ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«ΠΛΗΡΟΦΟΡΙΚΗ»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Αναγνώριση Εικόνων με τη Χρήση Τεχνητών Νευρωνικών Δικτύων** **Image Recognition with Neural Networks**
Ονοματεπώνυμο Φοιτητή	**ΓΕΩΡΓΙΟΣ ΤΑΣΙΟΥΛΗΣ**
Πατρώνυμο	**ΠΑΥΛΟΣ**
Αριθμός Μητρώου	**ΜΠΠΛ17055**
Επιβλέπων	**Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα
Δ. Σωτηρόπουλος *Επίκ. Καθηγητής*	Ε. Σακκόπουλος *Ανάπλ. Καθηγητής*	Γ. Τσιχριντζής *Καθηγητής*

Περίληψη

Η παρούσα μεταπτυχιακή διατριβή πραγματεύεται το ζήτημα της αναγνώρισης εικόνων με τη χρήση τεχνητών νευρωνικών δικτύων. Αρχικά γίνεται αναφορά στον τομέα της μηχανικής μάθησης και εξετάζεται ο αλγόριθμος της επικλινούς καθόδου που είναι βασικό συστατικό πολλών τεχνητών νευρωνικών δικτύων. Εν συνεχεία παρουσιάζεται η γενική δομή και λειτουργία των τεχνητών νευρωνικών δικτύων από τη στιγμή που τροφοδοτούνται με τα πρώτα δεδομένα έως το τελικό αποτέλεσμα που παράγουν. Ειδική αναφορά γίνεται στα τεχνητά νευρωνικά δίκτυα αισθητήρα, συνελικτικά, επαναλαμβανόμενα και παραγωγικά ανταγωνιστικά τεχνητά νευρωνικά δίκτυα. Τέλος γίνεται κατασκευή τεχνητού νευρωνικού δικτύου για κάθε μία από τις προαναφερθείσες κατηγορίες ώστε να εκπαιδευτεί, να δοκιμαστεί και να αξιολογηθεί σε πραγματικά δεδομένα εικόνων.

Λέξεις Κλειδιά: τεχνητό νευρωνικό δίκτυο, αναγνώριση εικόνας, μηχανική μάθηση, επιστήμη δεδομένων, τεχνητό νευρωνικό δίκτυο αισθητήρα, συνελικτικό τεχνητό νευρωνικό δίκτυο, επαναλαμβανόμενο τεχνητό νευρωνικό δίκτυο, παραγωγικό ανταγωνιστικό τεχνητό νευρωνικό δίκτυο, επικλινής κάθοδος, συνάρτηση ενεργοποίησης, συνάρτηση απώλειας, κανονικοποίηση, ρυθμός εκπαίδευσης.

Abstract

The present thesis describes the issue of image recognition with the use of artificial neural networks. It starts with an introduction to machine learning and the algorithm of gradient descent which is a key element to many artificial neural networks. Afterwards, the general structure and function of artificial neural networks is presented from the moment of the first input data till the production of the outcome. Emphasis is shown on single and multi-layer perceptrons, convolutional and recurrent neural networks and generative adversarial neural networks. Finally, a network of each of the aforementioned categories is constructed, trained, tested and evaluated on real image data.

Key Words: artificial neural network, image recognition, machine learning, data science, single layer perceptron, multi-layer perceptron, convolutional neural network, recurrent neural network, generative adversarial neural network, gradient descent, activation function, loss function, normalisation, learning rate.

Περιεχόμενα

1. Μηχανική Μάθηση

1.1 Τι είναι η Μηχανική Μάθηση

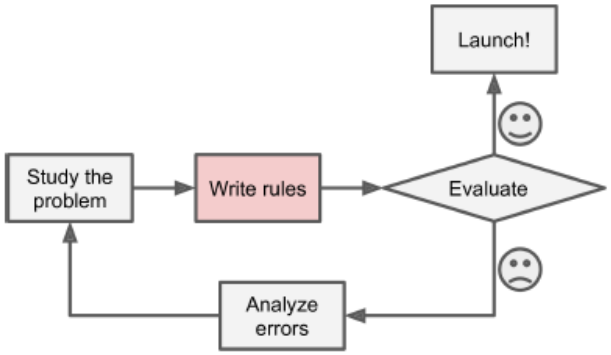
Η μηχανική μάθηση σχετίζεται με την εξαγωγή γνώσης από δεδομένα, αποτελώντας ένα επιστημονικό πεδίο ανάμεσα στους τομείς της στατιστικής, της τεχνητής νοημοσύνης και της επιστήμης των υπολογιστών (Muller & Guido, 2016). Η χρήση της μηχανικής μάθησης είναι πλέον καθημερινότητα, όπως για παράδειγμα οι αυτόματες συστάσεις για το ποιες ταινίες να δει κάποιο άτομο, προτάσεις για παραγγελίες φαγητού ή άλλων αγαθών και αναγνώριση των φιλικών μας προσώπων σε ηλεκτρονικές εφαρμογές. Οι εφαρμογές της μηχανικής μάθησης λαμβάνουν χώρα όχι μόνο σε εμπορικές εφαρμογές αλλά και επιστημονικές όπως η κατανόηση θεμάτων και προβλημάτων της αστροφυσικής, η ανακάλυψη νέων σωματιδίων, η ανάλυση του DNA και η αντιμετώπιση του καρκίνου. Η μηχανική μάθηση λοιπόν δεν αναφέρεται στη δημιουργία ενός ρομπότ του μέλλοντος με υπερανθρώπινες ιδιότητες αλλά είναι πλέον πραγματικότητα στην καθημερινότητά μας. Συνοψίζοντας, μπορούμε να ισχυριστούμε πως η μηχανική μάθηση είναι η επιστήμη (και η τέχνη) προγραμματισμού των υπολογιστών έτσι ώστε να μαθαίνουν από δεδομένα (Geron, 2019).

1.2 Γιατί είναι χρήσιμη η μηχανική μάθηση

Έστω ότι θέλουμε να κατασκευάσουμε ένα φίλτρο για σπαμ μέηλ χρησιμοποιώντας τις παραδοσιακές τεχνικές. Αυτά που ενδεχομένως να κάνουμε είναι:

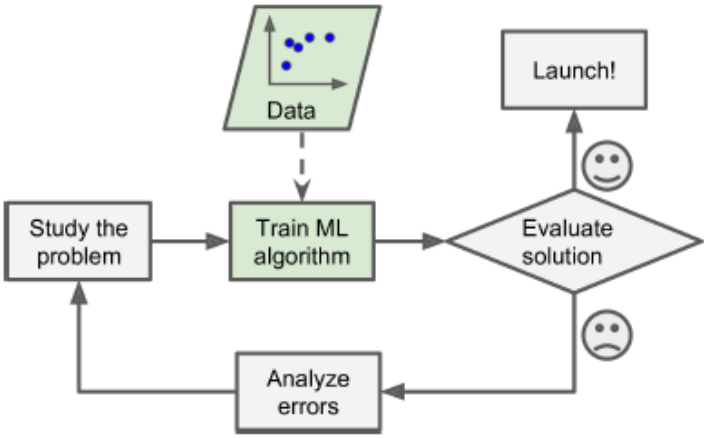
1. Αναζήτηση για το πως είναι δομημένο ένα σπαμ μέηλ. Προσπαθούμε να εντοπίσουμε λέξεις που είναι πολύ συνηθισμένες σε αυτά τα μέηλ όπως «πιστωτική κάρτα», «δωρεάν», «ευκαιρία» ή «καταπληκτικό» και ενδεχομένως τον τρόπο δομής τέτοιων μέηλ.
2. Γράφουμε έναν αλγόριθμο εντοπισμού των δομών σπαμ μέηλ που εντοπίσαμε και το πρόγραμμά μας σηματοδοτεί ως σπαμ τα μέηλ εκείνα στα οποία εντοπίζεται ένας αριθμός από αυτές τις δομές.
3. Κάνουμε δοκιμές στο πρόγραμμά μας και επαναλαμβάνουμε τα βήματα 1 και 2 έως ότου έχουμε ένα λειτουργικό πρόγραμμα.

Καθώς το πρόβλημα δεν είναι τετριμμένο είναι πολύ πιθανό να καταλήξουμε με ένα τεράστιο πρόγραμμα σύνθετων κανόνων το οποίο είναι αρκετά δύσκολο να συντηρηθεί, εικόνα 1.1.



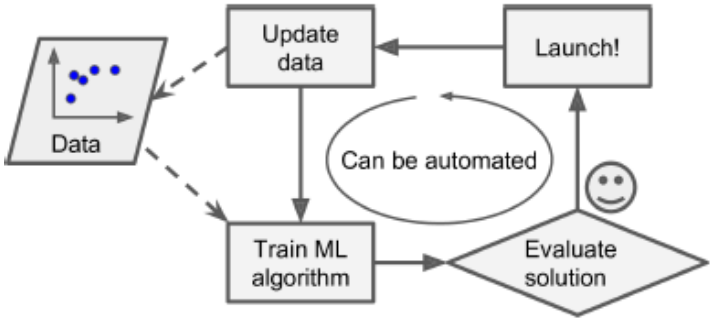
Εικόνα 1.1. Παραδοσιακή προσέγγιση σύνθετου προβλήματος (Geron, 2019).

Εν αντιθέσει, ένα φίλτρο ανεπιθύμητου μέηλ βασισμένο στη μηχανική μάθηση αυτόματα μαθαίνει ποιες λέξεις και φράσεις είναι ύποπτες για ανίχνευση ενός ανεπιθύμητου μέηλ εντοπίζοντας ασυνήθιστες συχνές δομές λέξεων στα ανεπιθύμητα μέηλ συγκρίνοντάς τα με τα επιθυμητά. Το τελικό πρόγραμμα είναι πολύ μικρότερο, ευκολότερο στη συντήρηση και πολύ πιθανότερα ακριβές, εικόνα 1.2.



Εικόνα 1.2. Προσέγγιση προβλήματος με μηχανική μάθηση. (Geron, 2019).

Επιπροσθέτως, αν οι δημιουργοί των ανεπιθύμητων μέηλ παρατηρήσουν πως τα μέηλ που περιέχουν τη φράση «4U» μπλοκάρονται, μπορούν να αλλάξουν τη φράση κάνοντάς την «for you». Ένα φίλτρο ανεπιθύμητων μέηλ που χρησιμοποιεί παραδοσιακές τεχνικές θα πρέπει να ανανεωθεί ώστε να μπορεί να αποκλείει τα νέα μέηλ. Αν οι σπάμερς πραγματοποιούν συνεχώς τέτοιες αλλαγές αυτό σημαίνει πως και το δικό μας πρόγραμμα θα πρέπει να αλλάζει συνέχεια. Από την άλλη μεριά, ένα φίλτρο σπαμ βασισμένο σε τεχνικές μηχανικής μάθησης, αυτόματα εντοπίζει πως η φράση «for you» εμφανίζεται συχνά στα σπαμ μέηλ και τα κατηγοριοποιεί αναλόγως, χωρίς τη δική μας επέμβαση, εικόνα 1.3.



Εικόνα 1.3. Αυτόματη προσαρμογή στην αλλαγή. (Geron, 2019).

Εν γένει μπορούμε να ισχυριστούμε πως η μηχανική μάθηση ενδείκνυται για:

- Προβλήματα των οποίων οι υπάρχουσες λύσεις απαιτούν συνεχείς ανθρώπινες παρεμβάσεις και έχουν πάρα πολλούς κανόνες. Ένας αλγόριθμος μηχανικής μάθησης μπορεί να απλοποιήσει τον κώδικα και να λειτουργεί αποτελεσματικότερα.
- Σύνθετα προβλήματα για τα οποία δεν υπάρχει καμία καλή λύση με τη χρήση των παραδοσιακών μεθόδων. Οι καλύτερες τεχνικές μηχανικής μάθησης μπορούν να βρουν μια λύση.
- Ευμετάβλητα περιβάλλοντα. Ένα σύστημα μηχανικής μάθησης μπορεί να προσαρμόζεται στα νέα δεδομένα.
- Λήψη γνώσης για σύνθετα προβλήματα και μεγάλα ποσά δεδομένων.

1.3 Τι είναι η Επιστήμη των Δεδομένων

Τη δεκαετία του 1980 η IBM ξεκίνησε την πρώτη σχεσιακή βάση δεδομένων όπου αποθήκευε πληροφορίες σχετικά με τους πελάτες ή τις πληρωμές επιχειρήσεων. Έχοντας όλα αυτά τα δεδομένα διαθέσιμα, δημιουργήθηκε ο προβληματισμός για το πως μπορούμε να εφαρμόσουμε αλγορίθμους για να εξάγουμε συγκεκριμένα μοτίβα – patterns από αυτά τα δεδομένα, το λεγόμενο data mining (Fayyad, Piatetsky, Smyth, 1996). Αρχικά αυτά τα μοτίβα εξάγονταν αποκλειστικά με τη χρήση της στατιστικής. Ωστόσο με την πρόοδο της τεχνολογίας ο συνδυασμός της εξόρυξης γνώσης από δεδομένα και της επιστήμης των υπολογιστών μας οδήγησε στην επιστήμη των δεδομένων (data science). Στις μέρες μας, τα δεδομένα δημιουργούνται από παντού, από το κάθε κλικ που κάνουμε στο διαδίκτυο μέσα από τον υπολογιστή μας ή από τις εφαρμογές που χρησιμοποιούμε στο κινητό μας τηλέφωνο, τα οποία μπορούν να καταγράψουν τις συνήθειές μας, την καταναλωτική συμπεριφορά μας ή τον τρόπο που ζούμε. Σίγουρα για μια επιχείρηση που πουλά προϊόντα εγκυμοσύνης θα ήταν άκρως πολύτιμο να γνωρίζει αν οι υπάρχοντες ή οι εν δυνάμει πελάτες της κυοφορούν και έτσι να τους προωθήσει τα προϊόντα της. Η επιστήμη των δεδομένων έρχεται να ξεκαθαρίσει το θολό τοπίο όλων αυτών των άπειρων δεδομένων και να τα μετατρέψει σε γνώση που θα έχει ουσιαστική αξία. Με μια πρόταση μπορούμε να ισχυριστούμε πως η επιστήμη των δεδομένων μετατρέπει τα δεδομένα σε αξία.

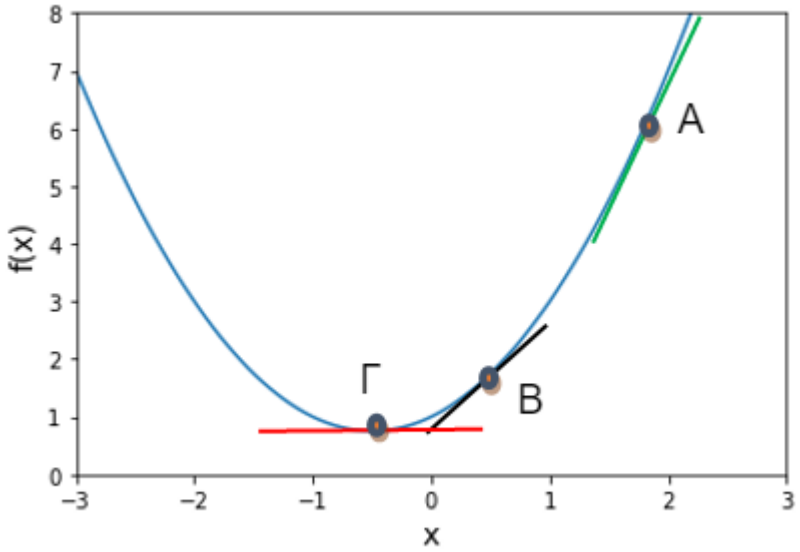
Για να μπορέσουμε να φτάσουμε στο σημείο να εξορύξουμε πολύτιμη γνώση από το θολό τοπίο των αρχικών μας δεδομένων είναι απαραίτητο να ακολουθήσουμε κάποια βήματα:

1. **Καθορισμός ερώτησης** στην οποία θέλουμε να απαντήσουμε. Μια καλώς διαμορφωμένη ερώτηση είναι καθοριστικής σημασίας για το ποια δεδομένα πρέπει να αναζητήσουμε και να επεξεργαστούμε.
2. **Συλλογή δεδομένων** που θα μας βοηθήσουν να απαντήσουμε στην ερώτηση.
3. **Καθαρισμός δεδομένων**. Τα δεδομένα που θα συλλέξουμε πρέπει να επεξεργαστούν ώστε να δούμε αν έχουμε ελλείψεις στα δεδομένα μας, αν υπάρχουν λάθη στα δεδομένα ή αν έχουμε τον επιθυμητό τύπο δεδομένων.
4. **Εξερεύνηση και οπτικοποίηση δεδομένων**. Ένα σχεδιάγραμμα οπτικοποιεί τα δεδομένα που έχουμε στην κατοχή μας και είναι πολύ πιο αποτελεσματικό στην κατανόησή τους από έναν πίνακα γεμάτο αριθμούς. Με την οπτικοποίηση μπορούμε να εντοπίσουμε ευκολότερα διάφορα προβλήματα στα δεδομένα μας, απομακρυσμένες τιμές (outliers), τάσεις ή μοτίβα στα δεδομένα. Ενδέχεται σε αυτό το βήμα να χρειαστεί να κάνουμε κάποιον επιπλέον καθαρισμό των δεδομένων, συνεπώς στην πράξη αυτό το βήμα συχνά γίνεται μαζί με το προηγούμενο.
5. **Εκπαίδευση αλγορίθμου**. Για να μπορέσουμε να εξορύξουμε γνώση από τα δεδομένα μας θα κάνουμε χρήση ενός αλγορίθμου τον οποίο πρέπει πρώτα να εκπαιδεύσουμε ώστε να μπορεί να ανιχνεύει τα πολύτιμα μοτίβα και δομές που αναζητούμε μέσα από τα δεδομένα.
6. **Αξιολόγηση αποτελέσματος**. Στο τελευταίο στάδιο πρέπει να δούμε πως τα πήγε ο αλγόριθμος. Απάντησε την αρχική μας ερώτηση και κατά πόσο ακριβής ήταν;

2. Επικλινής Κάθοδος – Gradient Descent

2.1 Ένα ελάχιστο

Έστω ότι έχουμε μία συνάρτηση κόστους της μορφής $f(x)=x^2+x+1$ είναι δηλαδή μια παραβολή την οποία θέλουμε να ελαχιστοποιήσουμε μιας και πρόκειται για συνάρτηση κόστους. Ιδανικά θέλουμε να φτάσουμε στο ελάχιστο σημείο της, το σημείο Γ. Μέσα από την κλίση της καμπύλης μπορούμε να μάθουμε ποιο είναι το ζητούμενο ελάχιστο σημείο καθότι αυτή η κλίση είναι ο ρυθμός μεταβολής. Παρατηρούμε πως η κλίση στη σημείο Α είναι έντονη, στο Β λιγότερο έντονη και στο Γ, όπου βρίσκεται και το ζητούμενο ελάχιστο σημείο είναι 0 (όσο μακρύτερα βρισκόμαστε από το ελάχιστο σημείο τόσο πιο έντονη είναι η κλίση), εικόνα 2.1. Για να βρούμε την κλίση θα χρειαστούμε την 1η παράγωγο της συνάρτησης.



Εικόνα 2.1. Γραφική αναπαράσταση της συνάρτησης $f(x)=x^2+x+1$ (γαλάζια γραμμή) και εφαπτόμενες στα σημεία Α, Β, Γ.

Εφαρμόζοντας τον αλγόριθμο της επικλινής καθόδου (gradient descent), ξεκινάμε από κάποιο σημείο της συνάρτησης, έστω το Α και σε κάθε βήμα αφαιρούμε, καθότι κινούμαστε αντίθετα από την κλίση, μία ποσότητα ίση με την παράγωγο της συνάρτησης στο σημείο που είμαστε, εν προκειμένω το Α, πολλαπλασιασμένη με μία ποσότητα που ονομάζεται ρυθμός εκπαίδευσης (learning rate). Αν εφαρμόσουμε αυτό τον αλγόριθμο για 30 επαναλήψεις λαμβάνουμε τα παρακάτω αποτελέσματα:

```
In [33]: # Gradient Descent
new_x = 3
previous_x = 0
step_multiplier = 0.1

for n in range(30):
    previous_x = new_x
    gradient = df(previous_x)
    new_x = previous_x - step_multiplier * gradient

print(f"Local min occurs at: {new_x}")
print(f"Slope or df(x) at this point is: {df(new_x)}")
print(f"f(x) value at this point {f(new_x)}")
```

```
Local min occurs at: -0.4956672098625011
Slope or df(x) at this point is: 0.008665580274997753
f(x) value at this point 0.7500187730703756
```

Εικόνα 2.2

Παρατηρούμε πως τα αποτελέσματα είναι μια προσέγγιση της πραγματικότητας (το ελάχιστο συμβαίνει στο σημείο όπου η παράγωγης της συνάρτησης είναι 0, δηλαδή 2x + 1 = 0 άρα x = -0.5.

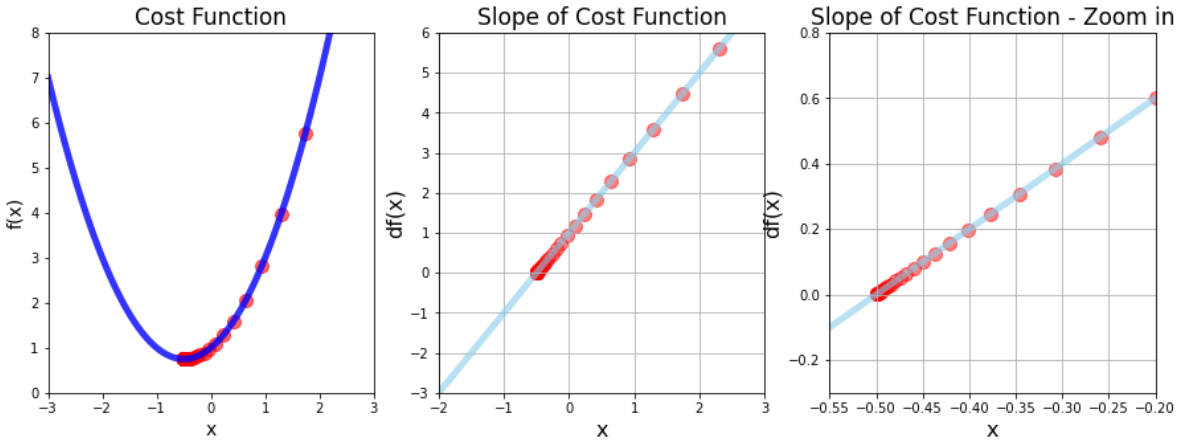
Αν τρέξουμε τον αλγόριθμο για περισσότερες επαναλήψεις λαμβάνουμε μια καλύτερη προσέγγιση της πραγματικότητας. Ανάλογα λοιπόν με το βαθμό ακρίβειας που επιθυμούμε, μπορούμε να καθορίσουμε τον αριθμό των επαναλήψεων (εικόνα 2.3).

```
In [34]: # Gradient Descent
new_x = 3
previous_x = 0
step_multiplier = 0.1

for n in range(100):
    previous_x = new_x
    gradient = df(previous_x)
    new_x = previous_x - step_multiplier * gradient

print(f"Local min occurs at: {new_x}")
print(f"Slope or df(x) at this point is: {df(new_x)}")
print(f"f(x) value at this point {f(new_x)}")
```

```
Local min occurs at: -0.4999999992870374
Slope or df(x) at this point is: 1.4259251557291464e-09
f(x) value at this point 0.75
```

Εικόνα 2.3

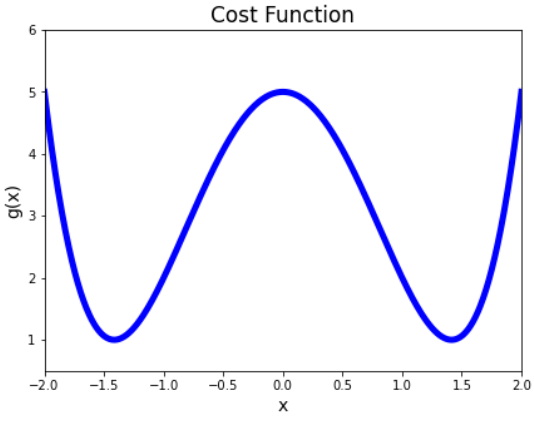
Στη γραφική αναπαράσταση της εκτέλεσης του αλγορίθμου μπορούμε να δούμε πως στα αρχικά στάδια οι μεταβολές είναι μεγαλύτερες, ενώ όσο πλησιάζουμε προς τον στόχο που είναι το ελάχιστο, αυτές οι μεταβολές γίνονται συνεχώς μικρότερες (εικόνα 2.4).



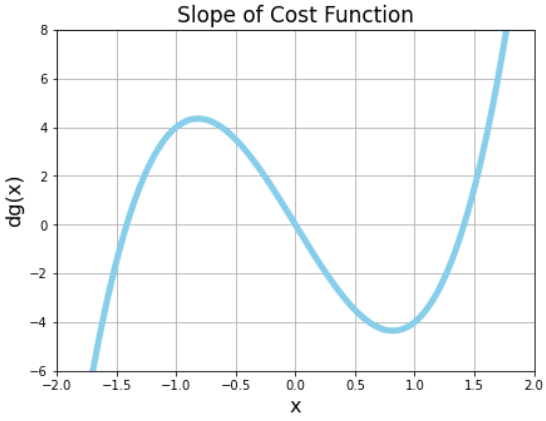
Εικόνα 2.4

2.2 Πολλαπλά τοπικά ελάχιστα

Αν θεωρήσουμε ως συνάρτησης κόστους που θέλουμε να ελαχιστοποιήσουμε την $g(x) = x^4 – 4x + 5$, παρατηρούμε από τη γραφική της παράσταση ότι έχει 2 ελάχιστα (εικόνα 2.5α).



Εικόνα 2.5α



Εικόνα 2.5β

Μπορούμε επίσης να κατασκευάσουμε την παρακάτω συνάρτηση για τον υπολογισμό της επικλινής καθόδου:

```python
# Gradient Descent function

def gradient_descent(derivative_func, initial_guess, multiplier=0.02, precision=0.001):
    new_x = initial_guess

    x_list = [new_x]
    slope_list = [derivative_func(new_x)]

    for n in range(100):
        previous_x = new_x
        gradient = derivative_func(previous_x)
        new_x = previous_x - multiplier * gradient

        step_size = abs(new_x - previous_x)

        x_list.append(new_x)
        slope_list.append(derivative_func(new_x))

        if step_size < precision:
            break

    return new_x, x_list, slope_list
```

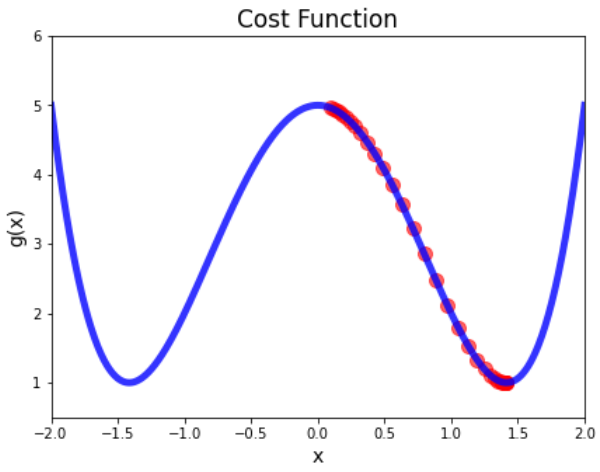
Εικόνα 2.6

Με αρχικό σημείο το 0.1 μπορούμε να καλέσουμε τη συνάρτηση και να αποτυπώσουμε τα αποτελέσματα (εικόνα 2.8α και 2.8β) όπου ο αλγόριθμος καταλήγει στο ελάχιστο 1.41

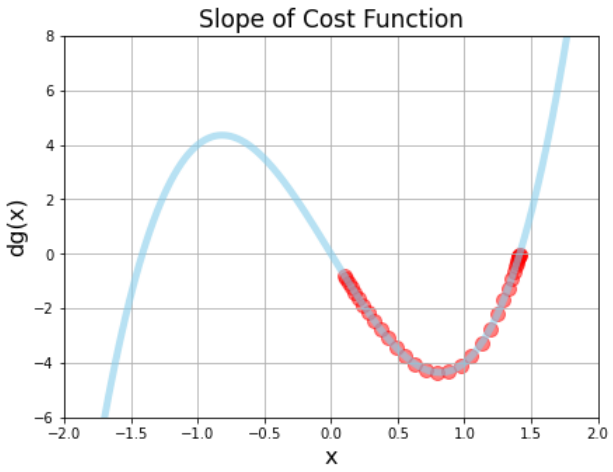
```python
local_min, list_x, deriv_list = gradient_descent(derivative_func=dg, initial_guess= 0.1)
print(f"Local min occurs at {local_min}")
print(f"Number of steps: {len(list_x)}")
```

```
Local min occurs at 1.4120887490901561
Number of steps: 34
```

Εικόνα 2.7



Εικόνα 2.8α



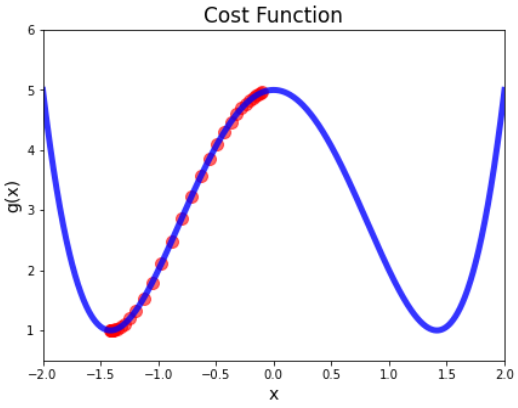
Εικόνα 2.9β

Αν ξεκινήσουμε από το σημείο -0.1 ο αλγόριθμος καταλήγει αυτή τη φορά στο ελάχιστο -1.41, εικόνα 2.10, 2.11α & 2.11β.

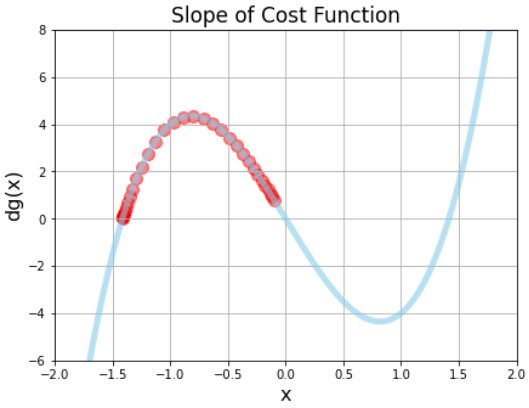
```
local_min, list_x, deriv_list = gradient_descent(derivative_func=dg, initial_guess= -0.1)
print(f"Local min occurs at {local_min}")
print(f"Number of steps: {len(list_x)}")
```

```
Local min occurs at -1.4120887490901561
Number of steps: 34
```

Εικόνα 2.10

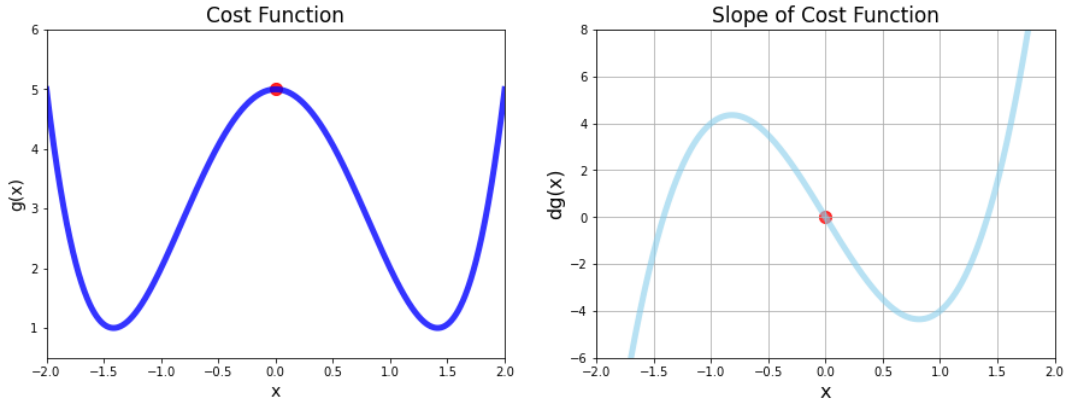


Εικόνα 2.11β



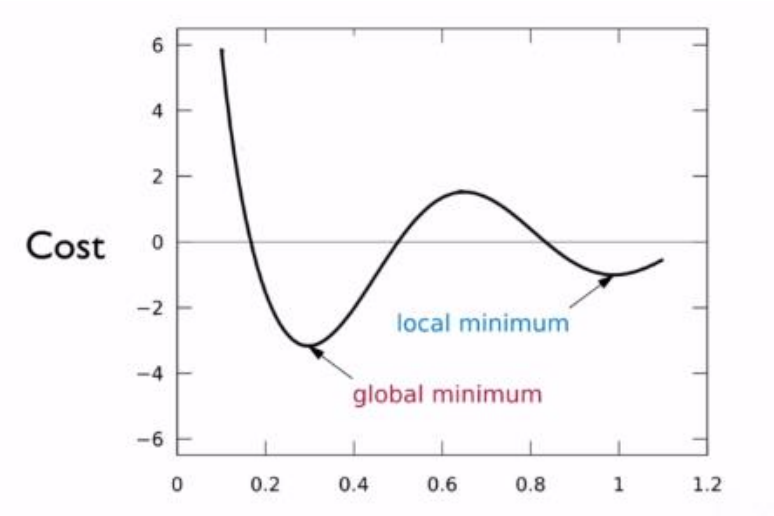
Εικόνα 2.11β

Αν επιλέξουμε να ξεκινήσουμε από το σημείο 0 παραητρούμε πως ο αλγόριθμος δεν κάνει κάθοδο προς καμία κατεύθυνση, απεναντίας μένει σε ένα τοπικό μέγιστο καθότι η κλίση σε εκείνο το σημείο είναι 0. (εικόνα 2.12)



Εικόνα 2.12

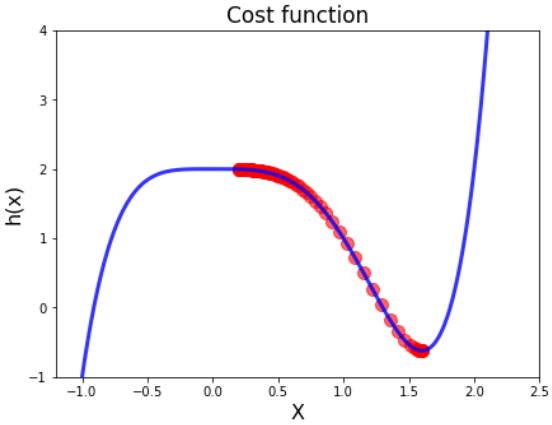
Επίσης αν η συνάρτηση κόστους είχε ένα τοπικό και ένα ολικό ελάχιστο, υπάρχει ο κίνδυνος εγκλωβισμού του αλγορίθμου στο τοπικό ελάχιστο, μεγαλύτερου κόστους, σε περίπτωση που το αρχικό σημείο ήταν στο δεξί μέρος της εικόνας 2.13 μη βρίσκοντας έτσι τη βέλτιστη λύση (εικόνα 2.13).



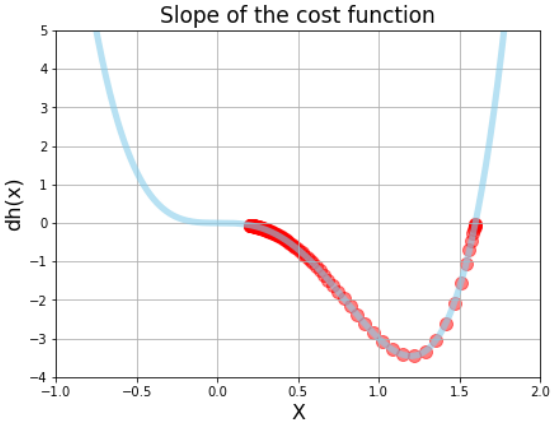
Εικόνα 2.13

Παρατηρούμε πως το μονοπάτι της καθόδου επιρεάζεται δραματικά από το αρχικό σημείο που θα επιλέξουμε, γεγονός το οποίο δείχνει την ευαισθησία του αλγορίθμου στην επιλογή της αρχικής θέσης. Μία λύση σε αυτό το ζήτημα είναι η εισαγωγή τυχαιότητας, ξεκινόντας από διαφορετικά αρχικά σημεία ώστε να δούμε ποιο θα βρει τη βέλτιστη λύση. Αυτή η λύση ενδείκνυται όταν δε γνωρίζουμε που ακριβώς είναι το ζητούμενο ελάχιστο.

Στην περίπτωση που έχουμε τη συνάρτηση κόστους $x^5 – 2x^4 + 2$ και ξεκινήσουμε τον αλγόριθμο από το σημείο 0.2 τότε έχουμε τα αποτλέσματα της εικόνας 2.14α και 2.14β.



Εικόνα 2.14α



Εικόνα 2.14β

Όμως αν ξεκινήσουμε από το σημείο -0.2 τότε το πρόγραμμα θα εμφανίσει πρόβλημα καθώς το ελάχιστο από την αριστερή πλευρά του γραφήματος τείνει στο άπειρο.

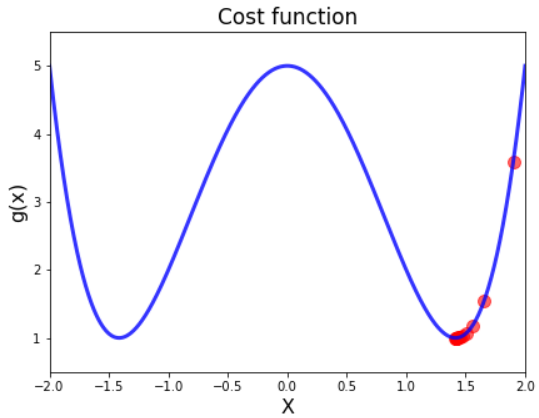
2.3 Ρυθμός Εκπαίδευσης – Learning Rate

Ο ρυθμός εκπαίδευσης είναι ένας αριθμός που καθορίζει το πόσο μεγάλο θα είναι το βήμα που θα εκτελέσει ο αλγόριρθμος της επικλινής καθόδου.

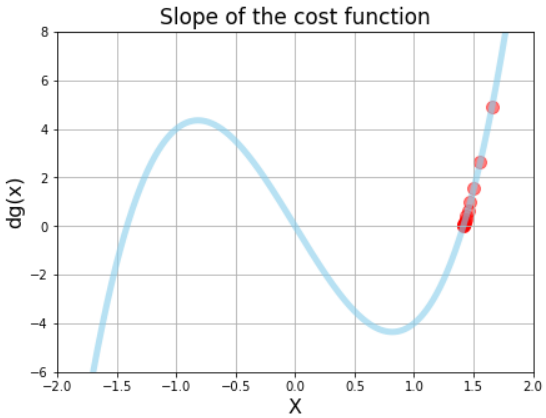
Παίρνοντας τη συνάρτηση $g(x) = x^4 + 4x^2 + 5$ και ξεκινώντας από το σημείο 1.9 (εικόνα 2.16α) με ρυθμό εκπαίδευσης 0.02 έχουμε την εύρευση του ελαχίστου σε 14 βήματα:

```
# Calling gradient descent function
local_min, list_x, deriv_list = gradient_descent(derivative_func=dg, initial_guess= 1.9,
                                                 multiplier=0.02, max_iter=500)
```

Εικόνα 2.5



Εικόνα 2.16α

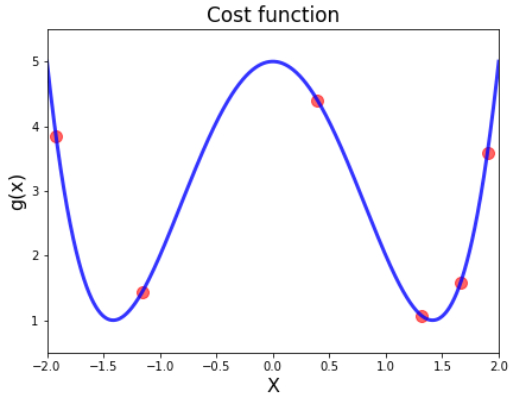


Εικόνα 2.16β

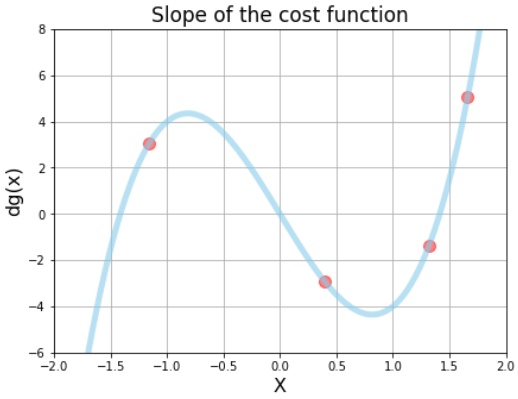
Αν αυξήσουμε το ρυθμό εκπαίδευσης στο 0.25 παρατηρούμε πως ο αλγόριθμος κάνει κάποιες τεράστειες αναπηδήσεις μη καταφέρνοντας να βρει κάποιο ελάχιστο (εικόνα 2.18α & 2.18β):

```
# Calling gradient descent function
local_min, list_x, deriv_list = gradient_descent(derivative_func=dg, initial_guess= 1.9,
                                                 multiplier=0.25, max_iter=5)
```

Εικόνα 2.17



Εικόνα 2.18α

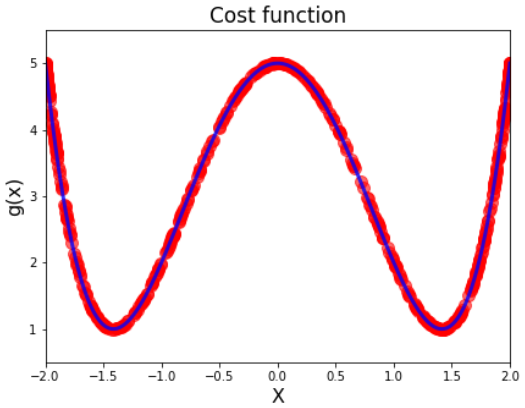


Εικόνα 2.18β

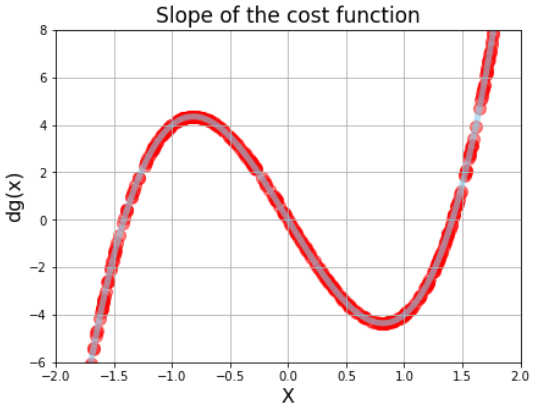
Αν κρατήσουμε το ρυθμό εκπαίδευσης στο 0.25 και αυξήσουμε και τον αριθμό επαναλήψεων παρατηρούμε πως ο αλγόριθμος πάλι δε συγκλείνει (εικόνα 2.20α & 2.20β):

```
# Calling gradient descent function
local_min, list_x, deriv_list = gradient_descent(derivative_func=dg, initial_guess= 1.9,
                                                 multiplier=0.25, max_iter=500)
```

Εικόνα 2.19

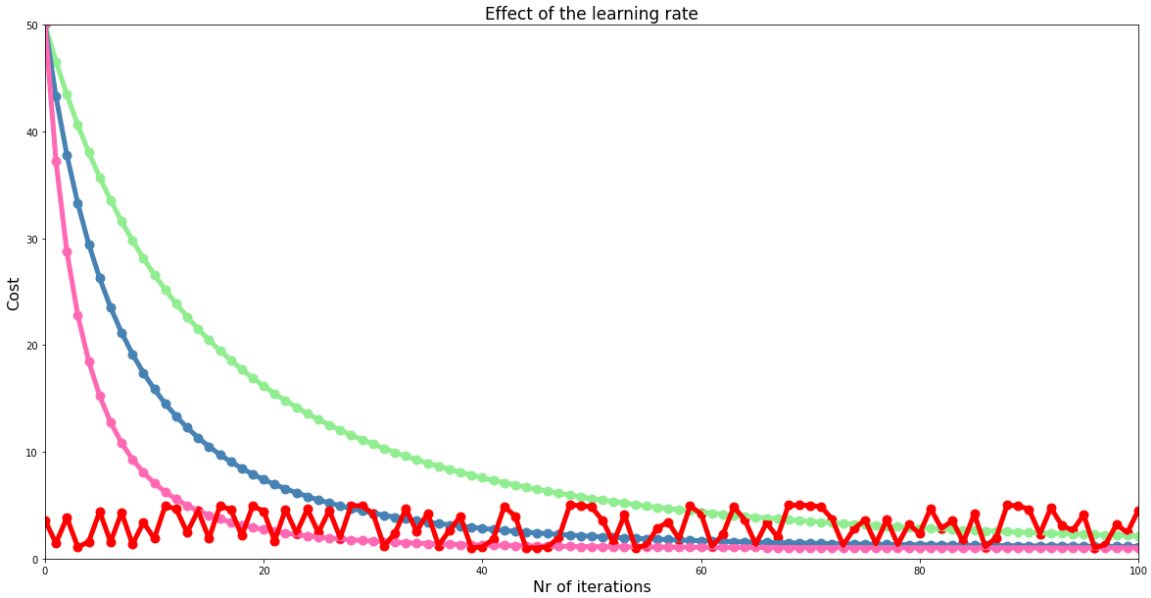


Εικόνα 2.20α



Εικόνα 2.20β

Αναπαριστώντας γραφικά τη σύγκλιση 4 διαφορετικών ρυθμών εκπαίδευσης λαμβάνουμε τα παρακάτω αποτελέσματα:



Εικόνα 2.21

Με το πράσινο χρώμα έχουμε έναν ρυθμό εκπαίδευσης της τάξης του 0.0005, με το γαλάζιο χρώμα της τάξης του 0.001 και με το ροζ χρώμα της τάξης του 0.002. Και στις 3 αυτές περιπτώσεις παρατηρούμε ότι ο αλγόριθμος συγκλίνει βρίσκοντας το ελάχιστο αλλά όσο αυξάνεται ο ρυθμός εκπαίδευσης συγκλίνει γρηγορότερα.

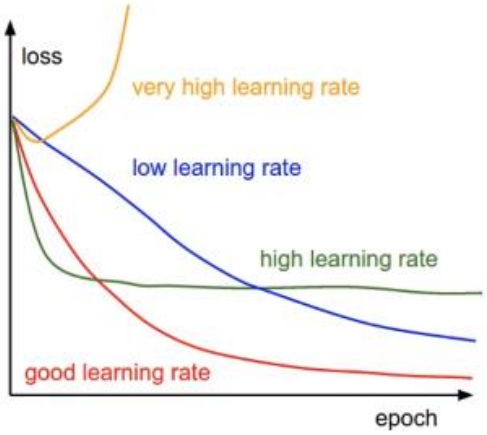
Αυτό όμως δεν ισχύει και στην περίπτωση του κόκκινου χρώματος όπου ο ρυθμός εκπαίδευσης είναι αρκετά υψηλότερος σε σχέση με αυτόν των άλλων χρωμάτων (0.25 έναντι 0.0005, 0.001 και 0.002) και έχει σημείο εκκίνησης το 1.9, δηλαδή ξεκίνα πολύ κοντινότερα σε κάποιο από τα ελάχιστα (τα +/- 1.4) σε σχέση με το σημείο εκκίνησης των άλλων χρωμάτων. Αυτό συμβαίνει γιατί το συγκεκριμένο μέγεθος ρυθμού εκπαίδευσης είναι αρκετά μεγάλο κάνοντας τον αλγόριθμο να μην μπορεί να συγκλίνει.

Συμπερασματικά μπορούμε να ισχυριστούμε ότι εν γένει πολύ μικροί ρυθμοί εκπαίδευσης θα έχουν περισσότερα βήματα και αργότερη σύγκληση ενώ μεγάλοι ρυθμοί εκπαίδευσης δημιουργούν τεράστιες «αναπηδήσεις» του αλγορίθμου ο οποίος ενδέχεται να μη συγκλίνει ποτέ. Δεν υπάρχει ένας μοναδικός τρόπος που να υποδεικνύει κάθε φορά ποιος είναι ο βέλτιστος ρυθμός εκπαίδευσης αλλά καθορίζεται ανάλογα με την περίπτωση και συχνά μετά από διάφορες δοκιμές. Επίσης ο ρυθμός εκπαίδευσης μπορεί να είναι μεταβλητός και να επανακαθορίζεται σε κάθε βήμα γεγονός που είναι χρήσιμο όταν είμαστε αρκετά μακριά από το ελάχιστο – στόχο και θέλουμε να κάνουμε μεγάλα βήματα ώστε να το προσεγγίσουμε και όσο το προσεγγίζουμε μικραίνουμε τα βήματά μας κατά συνέπεια και το ρυθμό εκπαίδευσης.

Συνοψίζοντας μπορούμε να ισχυριστούμε πως (Senior et al., 2013), εικόνα 2.22.

Με πολύ υψηλό ρυθμό εκπαίδευσης ελλοχεύει ο κίνδυνος ο αλγόριθμος να μη συγκλίνει ποτέ και η απώλεια να φθάσει μέχρι και το άπειρο.

Με πολύ χαμηλό ρυθμό εκπαίδευσης υπάρχει περίπτωση η σύγκλιση του αλγορίθμου να επιτευχθεί πάρα πολύ αργά ή να υπάρξει εγκλωβισμός σε τοπικό ελάχιστο.

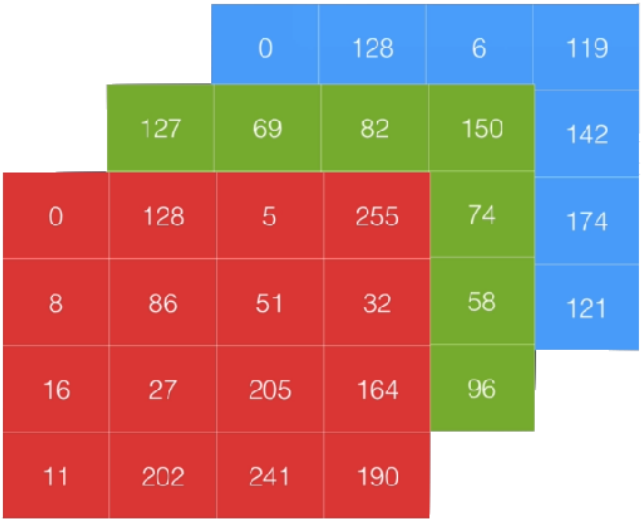


Εικόνα 2.23

3. Επεξεργασία Εικόνας

Όταν έχουμε ως δεδομένα φωτογραφίες, πρέπει να τις μετατρέψουμε σε μια τέτοια μορφή η οποία να είναι κατανοητή από την εφαρμογή και τον υπολογιστή. Για αυτό κάθε εικόνα θα μετατραπεί σε έναν πίνακα αριθμών που θα έχει πληροφορίες για κάθε πίξελ της. Πιο συγκεκριμένα κάθε πίξελ της εικόνας μπορεί να μετατραπεί σε έναν σύνολο από 3 αριθμούς. Κάθε ένας από αυτούς τους αριθμούς περιγράφει την ποσότητα κόκκινου, πράσινου και γαλάζιου χρώματος (RGB – Red, Green, Blue) που υπάρχει σε κάθε πίξελ. Κάθε χρώμα λαμβάνει αριθμούς από το 0 (που σημαίνει ανυπαρξία χρώματος) έως το 255 (που σημαίνει ότι το χρώμα είναι στην εντονότερη μορφή του). Για παράδειγμα ένα πίξελ με μοναδικό χρώμα το κόκκινο, έχει κάποια τιμή για το κόκκινο πχ 200, 0 (ελάχιστη για το πράσινο) και 0 (ελάχιστη για το γαλάζιο). Ένα πίξελ με μοναδικό χρώμα το πράσινο έχει 0 για το κόκκινο, μια τιμή για το πράσινο πχ 230 και 0 για το γαλάζιο, το λευκό έχει τη μορφή 255, 255, 255, το μαύρο 0, 0, 0 και συνδυάζοντας αυτές τις 3 τιμές από 0 έως 255 προκύπτουν όλα τα γνωστά χρώματα. Σε περίπτωση που μας ενδιαφέρουν οι περιπτώσεις του λευκού, μαύρου και των μεταξύ τους γκρι αποχρώσεων τότε μπορούμε να χρησιμοποιήσουμε μία τιμή μεταξύ του 0 που είναι το μαύρο και του 255 που είναι το λευκό.

Για κάθε λοιπόν εικόνα θα χρησιμοποιήσουμε έναν πίνακα 3 διαστάσεων, μία διάσταση για το ύψος της εικόνας, μία για το πλάτος και μία για το κανάλι χρωματισμού το οποίο αναφέρεται στις αναλογίες κόκκινου – πράσινου – γαλάζιου χρώματος. Έτσι δημιουργείται ένας πίνακας παρόμοιος με αυτόν της εικόνας 3.1.



Εικόνα 3.1

Η επεξεργασία εικόνας είναι αναπόσπαστο μέρος του πεδίου της όρασης υπολογιστών (computer vision) και χρησιμοποιείται σε εφαρμογές όπως (IBM, 2022):

- **Κατηγοριοποίηση Εικόνων (Image Classification)**: το μοντέλο δέχεται μια εικόνα και μπορεί να την κατατάξει σε μια κατηγορία (πχ. γάτα, σκύλος κτλ).

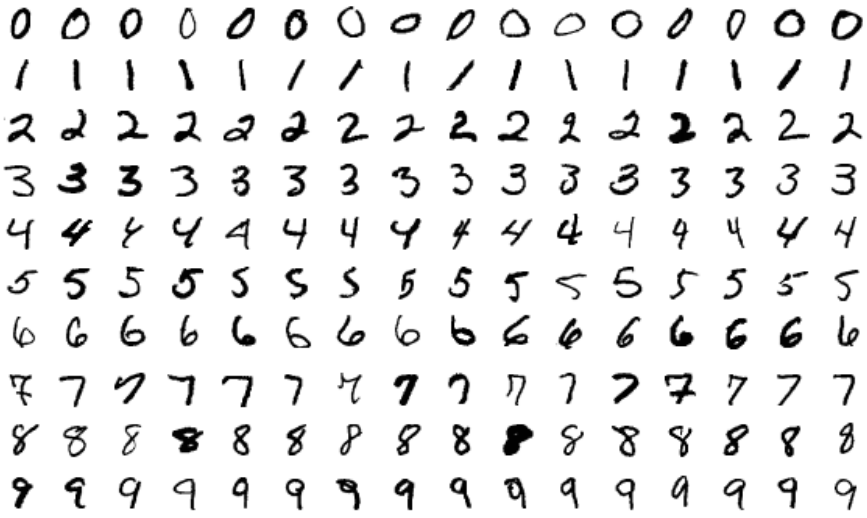
- **Ανίχνευση Αντικειμένων (Object Detection)**: Γίνεται χρήση της κατηγοριοποίησης εικόνων ώστε να ταυτοποιηθεί μια συγκεκριμένη κατηγορία εικόνας και στη συνέχεια η συγκεκριμένη κατηγορία να ανιχνευθεί σε κάποια εικόνα ή βίντεο.
- **Παρακολούθηση Αντικειμένων (Object Tracking):** Ένα αντικείμενο παρακολουθείται μόλις ανιχνευθεί και είναι χρήσιμο σε περιπτώσεις εικόνων που τραβήχτηκαν με μια σειρά ή αποτελούν μέρος ενός βίντεο. Για παράδειγμα, τα αυτόνομα αυτοκίνητα, εκτός από την ανίχνευση πεζών και άλλων αυτοκινήτων πρέπει να παρακολουθούν και την κίνησή τους στο χώρο ώστε να αποφευχθούν ατυχήματα.
- **Ανάκτηση Εικόνας με βάση κάποιο Περιεχόμενο (Content – Based Image Retrieval)**: Χρησιμοποιείται η όραση υπολογιστή για αναζήτηση και ανάκτηση εικόνων από μεγάλα σύνολα δεδομένων με βάση το περιεχόμενο των εικόνων. Μία εφαρμογή είναι η αυτόματη έναντι της χειρόγραφης τοποθέτησης σχολίων σε εικόνες.

4. Σύνολα Δεδομένων

Στις εφαρμογές της παρούσας εργασίας θα γίνει χρήση των συνόλων δεδομένων MNIST – Handwritten Digits και CIFAR 10 τα οποία είναι κατάλληλα για εφαρμογές μηχανικής μάθησης.

4.1 MNIST – Handwritten Digits

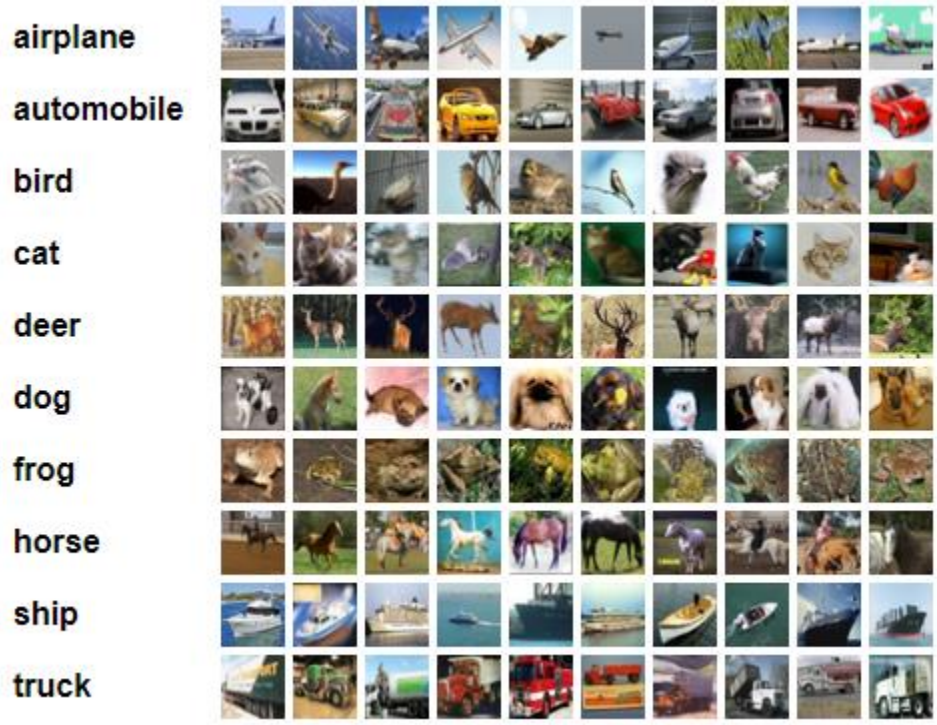
Το σύνολο δεδομένων MNIST – Handwritten Digits (Modified National Institute of Standards and Technology database) αποτελείται από εικόνες των 10 δεκαδικών ψηφίων (από το 0 έως το 9) σε χειρόγραφη μορφή. Το σύνολο αυτό έχει 60.000 παραδείγματα τέτοιων εικόνων που προορίζονται για εκπαίδευση των μοντέλων μηχανικής μάθησης και 10.000 παραδείγματα για δοκιμή. Είναι υποσύνολο της βάσης δεδομένων του NIST (http://yann.lecun.com/exdb/mnist/), οι εικόνες είναι σε αποχρώσεις του γκρι, συνεπώς υπάρχει ένα χρωματικό κανάλι με τιμές από το 0 για το απόλυτο λευκό έως το 255 για το απόλυτο μαύρο και οι διαστάσεις τους είναι 28 πίξελ ύψους επί 28 πίξελ πλάτους (εικόνα 4.1).



Εικόνα 4.1

4.2 CIFAR 10

Το σύνολο δεδομένων CIFAR 10 είναι μέρος ενός μεγαλύτερου συνόλου 80 εκατομμυρίων μικρών εικόνων (https://www.cs.toronto.edu/~kriz/cifar.html). Αποτελείται από 60.000 εικόνες συνολικά, που ανήκουν σε 10 κατηγορίες – κλάσεις. Από αυτές τις εικόνες οι 50.000 προορίζονται για εκπαίδευση του μοντέλου μηχανικής μάθησης και οι 10.000 για δοκιμή. Οι εικόνες αυτές είναι έγχρωμες, ;έχουν διαστάσεις 32 πίξελ ύψους επί 32 πίξελ πλάτους και ανήκουν στις κατηγορίες αεροπλάνο, αυτοκίνητο, πουλί, γάτα, τάρανδος, σκύλος, βάτραχος, άλογο, πλοίο και φορτηγό, εικόνα 4.2.


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

Εικόνα 4.2

5. Τεχνητά Νευρωνικά Δίκτυα

Σε αυτό το σημείο μπορούμε να περάσουμε από τον τομέα shallow learning της μηχανικής μάθησης, σε αυτόν του deep learning. Ο όρος shallow learning περιλαμβάνει τα μοντέλα που μελετήσαμε έως τώρα και στα οποία εμείς είμαστε υπεύθυνοι για την επιλογή εκείνων των χαρακτηριστικών που θα κάνουν το μοντέλο μας να πραγματοποιήσει καλύτερες προβλέψεις (Opela et al., 2002). Στον τομέα του deep learning, όπου ανήκουν τα τεχνητά νευρωνικά δίκτυα, το νευρωνικό δίκτυο αναλαμβάνει να επιλέξει τα σωστά χαρακτηριστικά που θα οδηγήσουν σε ακριβέστερες προβλέψεις. Το νευρωνικό δίκτυο θα εκπαιδευτεί στο να συνδυάσει τα απαραίτητα χαρακτηριστικά με τον πιο αποδοτικό τρόπο. Στην περίπτωση μάλιστα των νευρωνικών δικτύων, όσο πιο βαθύ είναι ένα δίκτυο, τόσο περισσότερα επίπεδα έχει. Στα πλεονεκτήματα των τεχνητών νευρωνικών δικτύων συγκαταλέγονται το ότι είναι ικανά να εντοπίζουν μη γραμμικές σχέσεις μεταξύ εξαρτημένων και ανεξάρτητων μεταβλητών καθώς και όλες τις πιθανές αλληλεπιδράσεις μεταξύ των μεταβλητών πρόβλεψης (predictor variables) και χρειάζονται λιγότερη τυπική στατιστική εκπαίδευση (Tu, 1996). Στα μειονεκτήματά τους περιλαμβάνεται το γεγονός του «μαύρου κουτιού», το γεγονός δηλαδή ότι η λειτουργία τους δεν είναι 100% ορατή στο χρήστη, έχουν μεγαλύτερο υπολογιστικό φόρτο και τείνουν στο overfitting – να προσαρμόζονται δηλαδή υπερβολικά στα δεδομένα εκπαίδευσης μη μπορώντας να ανταπεξέλθουν σε άλλα δεδομένα εκτός από αυτά και το κόστος κατασκευής τους πολλές φορές είναι αρκετά υψηλό (Tu, 1996).

5.1 Βιολογικά και Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα είναι μια σχετικά νέα περιοχή στις φυσικές επιστήμες, η οποία αντλεί την έμπνευσή της από το νευρικό σύστημα των ζώντων οργανισμών. Προσπαθούν να συνδυάσουν τον τρόπο σκέψης του ανθρώπινου εγκεφάλου (ο οποίος εκπαιδεύεται, θυμάται, ξεχνάει κτλ) με τον αφηρημένο μαθηματικό τρόπο σκέψης, ώστε να λύσουν κάθε είδους προβλήματα με τη βοήθεια των ηλεκτρονικών υπολογιστών. Το νευρικό σύστημα των οργανισμών αποτελείται από πολλά νευρωνικά δίκτυα τα οποία εξειδικεύονται σε διάφορες διεργασίες και κεντρική μονάδα όλου αυτού του συστήματος είναι ο εγκέφαλος. Ο εγκέφαλος και κατ’ επέκταση κάθε βιολογικό νευρωνικό δίκτυο έχει έναν μεγάλο αριθμό μονάδων που ονομάζονται νευρώνες, οι οποίες συνεχώς επεξεργάζονται πληροφορίες και τις στέλνουν σε άλλους νευρώνες του δικτύου, με αποτέλεσμα να περατώνουν περίπλοκες διεργασίες όπως η αναγνώριση εικόνων ή φωνής με ελάχιστη προσπάθεια. Ιδανικά, αυτήν την ικανότητα επιθυμούμε να μεταφέρουμε και στους ηλεκτρονικούς υπολογιστές, των οποίων βέβαια η δομή είναι αρκετά διαφορετική από αυτήν του εγκεφάλου και έτσι δημιουργήθηκε η έννοια των τεχνητών νευρωνικών δικτύων.

Ο ανθρώπινος εγκέφαλος έχει περίπου 10^{10} νευρώνες οι οποίοι είναι διαφορετικοί μεταξύ τους, κάθε νευρώνας συνδέεται με πολλούς άλλους νευρώνες μέσω συνδέσεων που ονομάζονται συνάψεις και υπολογίζεται πως κάθε νευρώνας έχει περίπου 10^4 συνάψεις (Αργυράκης 2001). Οι συνάψεις δημιουργούνται όταν ο εγκέφαλος αποκτά εμπειρίες όπως το να μαθαίνει, να αναγνωρίζει ή να κατανοεί. Ένας αριθμός νευρώνων με τις διασυνδέσεις τους είναι ένα βιολογικό νευρωνικό δίκτυο και

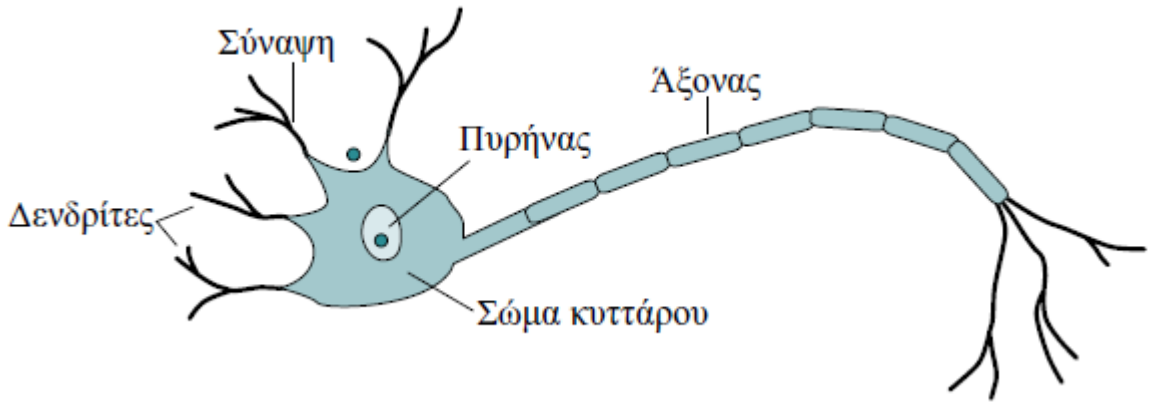
όλο το σύστημα των νευρωνικών δικτύων στον άνθρωπο δημιουργεί το κεντρικό νευρικό σύστημα που επεκτείνεται σε όλο το ανθρώπινο σώμα. Ο ρόλος τώρα του νευρώνα σε ένα νευρωνικό δίκτυο είναι να λαμβάνει τα σήματα που έρχονται από άλλους νευρώνες, να τα επεξεργάζεται και να μεταφέρει την επεξεργασμένη πλέον πληροφορία σε άλλους νευρώνες.

Ο βιολογικός νευρώνας είναι ένα κύτταρο αποτελούμενο από το κυρίως σώμα, τον άξονα και τους δενδρίτες και τα λειτουργικά του αυτά τμήματα λειτουργούν ως εξής (Διαμαντάρας, 2007):

- Οι δενδρίτες είναι οι πύλες εισόδου του νευρώνα καθώς δέχονται τα σήματα από άλλους νευρώνες.
- Ο άξονας είναι η πύλη εξόδου του νευρώνα καθώς στέλνει σήματα σε άλλους νευρώνες.
- Οι συνάψεις είναι τα σημεία ένωσης μεταξύ διακλαδώσεων του άξονα ενός νευρώνα και των δενδριτών από άλλους νευρώνες. Το ποσοστό της δραστηριότητας που μεταδίδεται στο δενδρίτη λέγεται συναπτικό βάρος και οι συνάψεις κατηγοριοποιούνται σε ενισχυτικές και ανασταλτικές ανάλογα με το αν το φορτίο που έλκεται από τη σύναψη ερεθίζει το νευρώνα κάνοντάς τον να παράγει ηλεκτρικούς παλμούς ή τον καταστέλλει εμποδίζοντάς τον να παράγει παλμούς.

Στους βιολογικούς νευρώνες, οι φορείς της πληροφορίας που είναι ηλεκτρικοί παλμοί, ταξιδεύουν στον άξονα κάθε νευρώνα και μέσω των συνάψεων διαδίδονται στους δενδρίτες των παραληπτών νευρώνων (Διαμαντάρας, 2007). Κάθε νευρώνας συλλέγει το ηλεκτρικό φορτίο που δέχεται από κάθε σύναψη στους δενδρίτες του, ζυγίζοντας το εισερχόμενο φορτίο με το αντίστοιχο συναπτικό βάρος. Όσο ισχυρότερη είναι η συναπτική ζεύξη τόσο εντονότερα συμμετέχει το συγκεκριμένο φορτίο εισόδου στο συνολικό άθροισμα. Αν αυτό το άθροισμα ξεπερνά κάποιο κατώφλι τότε ο άξονας του νευρώνα αρχίζει να παράγει ηλεκτρικούς παλμούς με μεγάλη συχνότητα και λέμε ότι ο νευρώνας πυροβολεί, ενώ αν το φορτίο δεν ξεπερνά το συγκεκριμένο αυτό όριο ο νευρώνας παραμένει αδρανής (Διαμαντάρας, 2007).

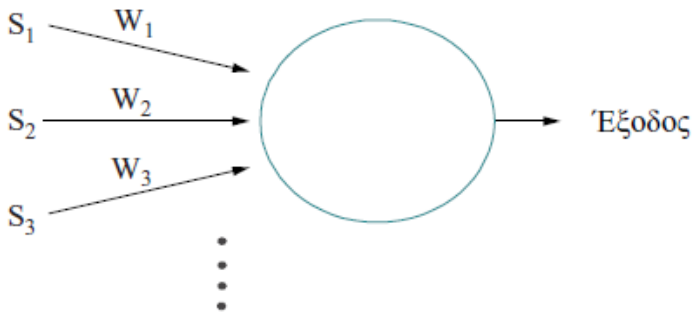
Στην παρακάτω εικόνα βλέπουμε την αναπαράσταση ενός βιολογικού νευρώνα



Εικόνα 5.1 (Αργυράκης, 2001)

Ωστόσο ο κόσμος δεν είναι μόνο για να παρατηρείται και να εξηγείται αλλά επίσης και για να χρησιμοποιείται ως έμπνευση για το σχεδιασμό τεχνουργημάτων με βάση την απλή αρχή ότι η φύση έχει κάνει μια αξιοθαύμαστη δουλειά εδώ και εκατομμύρια χρόνια (Σωτηρόπουλος – Τσιχριντζής, 2016). Τα τεχνητά νευρωνικά δίκτυα είναι ένα τρανταχτό παράδειγμα όπου οι φυσικές και βιολογικές διεργασίες θέτουν τη βάση για τη δημιουργία συστημάτων στο χώρο της πληροφορικής, της μηχανικής και εν γένει της τεχνολογίας.

Κατ' αναλογία με τα βιολογικά, έτσι και τα τεχνητά νευρωνικά δίκτυα αποτελούνται από νευρώνες, σε καθέναν από τους οποίους φθάνει ως είσοδος ένας αριθμός σημάτων (έστω S_1, S_2 κοκ). Ο τεχνητός νευρώνας μπορεί και αυτός, όπως ο βιολογικός, να ενεργοποιηθεί ή όχι και σε περίπτωση ενεργοποίησης να παράγει μία έξοδο. Κάθε σήμα που μεταδίδεται από τον έναν νευρώνα στον άλλον του νευρωνικού δικτύου συνδέεται με μια τιμή βάρους w η οποία είναι ενδεικτική για το πόσο συνδεδεμένοι είναι οι νευρώνες που συνδέονται με αυτό το βάρος. Όσο μεγαλύτερο είναι το βάρος τόσο μεγαλύτερη είναι και η συνεισφορά του σήματος. Όλα αυτά μπορούμε να τα δούμε σχηματικά στην εικόνα 5.2.



Εικόνα 5.2 (Αργυράκης, 2001). Ένας νευρώνας (κύκλος) με πολλές εισόδους (s1, s2, s3 κτλ.), αντίστοιχα βάρη (w1, w2, w3 κτλ) και μία έξοδο

Παρατηρούμε πως τα τεχνητά νευρωνικά δίκτυα έχουν ομοιότητες με τα βιολογικά και αναπτύχθηκαν ως γενικεύσεις μαθηματικών μοντέλων της ανθρώπινης νόησης ή νευροβιολογίας βασισμένα στις παρακάτω υποθέσεις (Faucett, 1993):

- Η επεξεργασία της πληροφορίας συμβαίνει σε πολλά απλά στοιχεία που λέγονται νευρώνες
- Τα σήματα μεταβιβάζονται μεταξύ των νευρώνων μέσα από συνδέσεις
- Κάθε σύνδεση σχετίζεται με ένα βάρος το οποίο σε ένα τυπικό νευρωνικό δίκτυο πολλαπλασιάζει το μεταδιδόμενο σήμα
- Κάθε νευρώνας εφαρμόζει μια συνάρτηση ενεργοποίησης, συνήθως μη γραμμική, στην είσοδό του (άθροισμα εισόδων) για να καθοριστεί η έξοδος.

Επίσης το νευρωνικό δίκτυο χαρακτηρίζεται από τη διάταξη των συνδέσεων μεταξύ των νευρώνων, που ονομάζεται αρχιτεκτονική, τη μέθοδο καθορισμού των βαρών των συνδέσεων που ονομάζεται εκπαίδευση και τη συνάρτηση ενεργοποίησης.

5.2 Ιστορική Αναδρομή

1943: Ο McCulloch, νευροφυσιολόγος και ο Pitts φοιτητής μαθηματικών παρουσιάζουν το πρώτο μοντέλο νευρωνικού δικτύου (McCoullogh – Pitts, 1943)
1947: Οι McCulloch – Pitts κάνουν ένα εξελιγμένο πρότυπο για την αναγνώριση σχημάτων όπου ο νευρώνας έχει δύο καταστάσεις, μπορεί να δέχεται πολλές εισόδους αλλά δίνει μόνο μία έξοδο.
1957: Ο Rosenblatt παρουσιάζει το μοντέλο του αισθητήρα (perceptron) που έχει μόνο δύο επίπεδα, της εισόδου και της εξόδου. Το σήμα προχωρά μονοδρομικά από την είσοδο στην έξοδο. Αρχίζει πλέον να επικρατεί η ιδέα πως τα νευρωνικά δίκτυα ίσως είναι η τεχνική που μπορεί να λύσει όλα τα προβλήματα.
1969: Οι Minsky και Papert αποδεικνύουν ότι το πρότυπο του αισθητήρα έχει περιορισμούς (πχ αδυναμία επίλυσης του απλού προβλήματος της λογικής πύλης X-0R) (Minsky – Papert, 1969)
1982: Ο Hopfield απέδειξε πως ένα νευρωνικό δίκτυο μπορεί να λειτουργήσει ως αποθηκευτικός χώρος αλλά και χώρος ανάκτησης της πληροφορίας (Hopfield, 1982)
1986: Οι McClelland – Rumelhart παρουσιάζουν ένα δίκτυο πολλών επιπέδων νευρώνων εκτός από την είσοδο και την έξοδο και προτείνουν μια νέα διαδικασία εκπαίδευσης, τη μέθοδο της οπισθοδιάδοσης, μια ιδιαίτερα χρήσιμη τεχνική ακόμα και σήμερα (McClelland – Rumelhart, 1986)

6. Εφαρμογές Τεχνητών Νευρωνικών Δικτύων

Οι εφαρμογές των νευρωνικών δικτύων καλύπτουν ένα ευρύτατο πεδίο επιστημών και αναγκών, μερικές από τις οποίες είναι οι ακόλουθες:

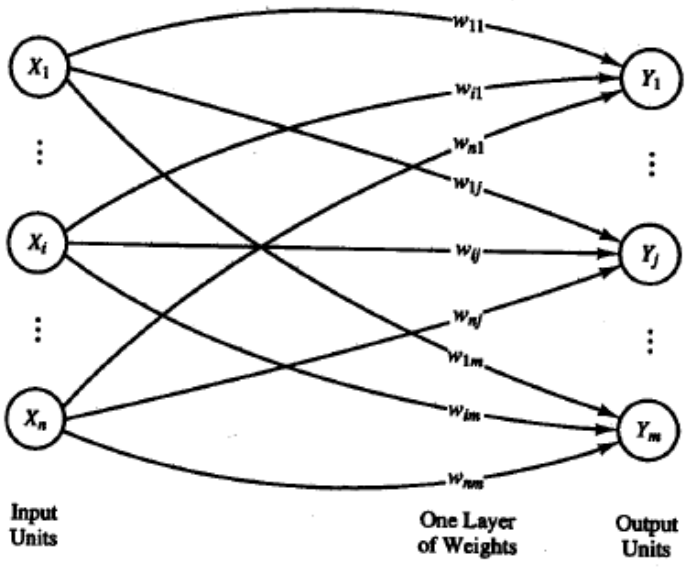
- Επεξεργασία σήματος. Μία από τις πρώτες εμπορικές εφαρμογές των τεχνητών νευρωνικών δικτύων είναι αυτή της εξάλειψης θορύβου στις τηλεφωνικές γραμμές (Faucett, 1993).
- Έλεγχος. Ένα παράδειγμα είναι ότι νευρωνικά δίκτυα παρέχουν βοήθεια στο χειρισμό μεγάλων οχημάτων όταν ο οδηγός κάνει όπισθεν (Nguyen & Widrow, 1989; Miller, Sutton, & Weros, 1990).
- Αναγνώριση σχεδίων. Νευρωνικά δίκτυα μπορούν να αναγνωρίσουν χειρόγραφους χαρακτήρες γράμματα ή ψηφία (Le Cun et al., 1990).
- Ιατρική. Ήδη από τη δεκαετία του 1980 έχει αναπτυχθεί μια εφαρμογή με το όνομα «Άμεσος Ιατρός». Το νευρωνικό δίκτυο αποθηκεύει ένα μεγάλο αριθμό ιατρικών ιστορικών με πληροφορίες για συμπτώματα, διαγνώσεις και θεραπεία για μια συγκεκριμένη περίπτωση. Μετά την εκπαίδευσή του μπορεί να βρει το πλήρες αποθηκευμένο σχέδιο που αντιπροσωπεύει την καλύτερη διάγνωση και θεραπεία (Anderson, 1986; Anderson, Golden, and Murphy, 1986).
- Παραγωγή λόγου. Η ορθή προφορά μιας γλώσσας όπως η αγγλική για παράδειγμα δεν είναι εύκολο καθότι η προφορά των γραμμάτων εξαρτάται πάντα από το που αυτά τα γράμματα εμφανίζονται. Νευρωνικά δίκτυα μετά από εκπαίδευση έχουν τη δυνατότητα να παρέχουν την ορθή προφορά λέξεων με μικρό ποσοστό λάθους (Sejnowski and Rosenberg, 1986).
- Αναγνώριση λόγου. Νευρωνικά δίκτυα μπορούν να αναγνωρίζουν ανεξάρτητους ομιλητές μέσα από το λόγο τους (Lippman, 1989).
- Χρήση από τράπεζες με σκοπό να εκτιμήσουν αν ένας εν δυνάμει δανειολήπτης είναι αξιόπιστος (McCord – Nelson M., W. T. Illingworth, 1991).
- Γρήγορες χημικές αναλύσεις με εφαρμογή στα αεροδρόμια για εντοπισμό εκρηκτικών ουσιών σε αποσκευές (Johnson, 1989).
- Στη βιομηχανία τα νευρωνικά δίκτυα χρησιμοποιούνται για αυτοματοποίηση ρομπότ και συστημάτων ελέγχου, επιλογή ανταλλακτικών κατά τη συναρμολόγηση, έλεγχο στη γραμμή παραγωγής και επιθεώρηση της ποιότητας κατά την κατασκευή (Αργυράκης, 2001).

7. Δομή Νευρωνικών Δικτύων

Τα δομικά συστατικά των νευρωνικών δικτύων είναι η αρχιτεκτονική τους και οι μέθοδοι καθορισμού των βαρών μέσω της εκπαίδευσης (Faucett, 1993).

7.1 Τυπικές Αρχιτεκτονικές

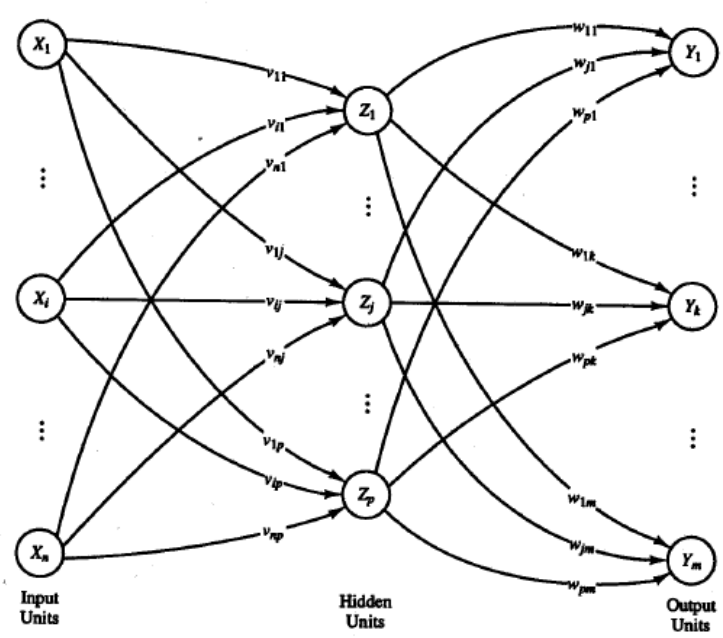
Συνήθως απεικονίζουμε τους νευρώνες ενός τεχνητού νευρωνικού δικτύου σε επίπεδα και οι νευρώνες του ίδιου επιπέδου συμπεριφέρονται με τον ίδιο τρόπο. Παράγοντες κλειδιά στον καθορισμό της συμπεριφοράς των νευρώνων είναι η συνάρτηση ενεργοποίησης και η διάταξη των συνάψεων με βάρη μεταξύ των νευρώνων. Συχνά οι νευρώνες στο ίδιο επίπεδο έχουν την ίδια συνάρτηση ενεργοποίησης. Σε αρκετά νευρωνικά δίκτυα υπάρχει ένα επίπεδο εισόδου, όπου η συνάρτηση ενεργοποίησης είναι ίση με το εξωτερικό σήμα εισόδου.
Μια γενική κατηγοριοποίηση των νευρωνικών δικτύων είναι σε ενός επιπέδου και πολλαπλών επιπέδων. Συνήθως στον καθορισμό των επιπέδων, οι νευρώνες που δέχονται το σήμα εισόδου δεν μετρούνται ως επίπεδο καθότι δεν κάνουν κάποιον υπολογισμό αλλά μεταφέρουν το σήμα εισόδου. Σε αυτή την περίπτωση ο αριθμός των επιπέδων καθορίζεται από τον αριθμό των επιπέδων των διασυνδέσεων με βάρη μεταξύ των νευρώνων. Στις εικόνες 3 και 4 βλέπουμε περιπτώσεις νευρωνικών δικτύων ενός και πολλών επιπέδων αντίστοιχα.



Εικόνα 7.1 (Faucett, 1993)

7.1.1 Τεχνητό Νευρωνικό Δίκτυο ενός επιπέδου

Οι κόμβοι X_1 έως X_n είναι νευρώνες εισόδου οι οποίοι δέχονται το εξωτερικό σήμα το οποίο θα μεταφέρουν στους νευρώνες Y_1 έως Y_m οι οποίοι είναι οι νευρώνες εξόδου. Οι συνάψεις – συνδέσεις μεταξύ των νευρώνων εισόδου και εξόδου έχουν όλες ένα βάρος w_{nm} (από το νευρώνα n στο νευρώνα m). Παρατηρούμε ότι υπάρχει ένα επίπεδο συνδέσεων (εικόνα 7.1).



Εικόνα 7.2 (Faucett, 1993)

7.1.2 Τεχνητό Νευρωνικό Δίκτυο πολλών επιπέδων.

Εδώ παρατηρούμε πως υπάρχουν 2 επίπεδα συνδέσεων (από τους νευρώνες X_n στους νευρώνες Z_P, και από τους νευρώνες Z_P στους νευρώνες Y_m, εικόνα 7.2). Οι νευρώνες X_n είναι οι νευρώνες εισόδου, οι νευρώνες Z_p είναι οι νευρώνες του κρυφού επιπέδου ή διαφορετικά κρυφοί νευρώνες και οι νευρώνες Y_m οι νευρώνες εξόδου.

Δεν υπάρχει ένας ενιαίος κανόνας για το πόσους νευρώνες πρέπει να χρησιμοποιήσουμε σε κάθε επίπεδο αλλά και για το πόσα επίπεδα νευρώνων είναι απαραίτητα. Αυτό εξαρτάται από το πρόβλημα που έχουμε να αντιμετωπίσουμε. Οι νευρώνες εισόδου εξαρτώνται από είδος των προτύπων που εισάγουμε στο νευρωνικό δίκτυο (αν πχ. έχουμε σημεία στο επίπεδο τότε θα χρειαστούμε 2 νευρώνες εισόδου, έναν για κάθε συντεταγμένη του σημείου) και οι νευρώνες εξόδου εξαρτώνται από τον αριθμό των κλάσεων που θέλουμε να κατηγοριοποιήσουμε τα δεδομένα μας (πχ με νευρώνες εξόδου που απαντούν ναι ή όχι μπορούμε να έχουμε το πολύ 2^1 δηλαδή δύο κλάσεις, με 2 τέτοιους νευρώνες μπορούμε να έχουμε το πολύ 2^2 δηλαδή τέσσερις κλάσεις κοκ). Επίσης δεν υπάρχει ένας ενιαίος κανόνας που να βρίσκει εφαρμογή σε κάθε πρόβλημα και να υποδεικνύει το πόσοι νευρώνες είναι συνδεδεμένοι και με ποιους. Αυτό που είναι σίγουρο είναι πως ο μέγιστος αριθμός συνδέσεων που μπορεί να υπάρξει σε ένα τεχνητό νευρωνικό δίκτυο είναι η περίπτωση που κάθε νευρώνας είναι συνδεδεμένος με όλους τους άλλους, οπότε ο αριθμός των συνδέσεων είναι N(N-1)/2 (N είναι ο αριθμός των νευρώνων) και ο ελάχιστος αριθμός των συνδέσεων είναι η περίπτωση όπου κάθε νευρώνας είναι συνδεδεμένος μόνο με έναν άλλο νευρώνα.

7.2 Εκπαίδευση Τεχνητού Νευρωνικού Δικτύου - Καθορισμός Βαρών

Η μέθοδος καθορισμού των βαρών των συνδέσεων μεταξύ των νευρώνων ενός νευρωνικού δικτύου συνιστά την εκπαίδευση του δικτύου (Faucett, 1993). Δύο γενικοί τύποι εκπαίδευσης είναι αυτοί με επιτήρηση και χωρίς επιτήρηση.

Στην εκπαίδευση με επιτήρηση (ή εκπαίδευση με δάσκαλο), τα διαθέσιμα δεδομένα δίδονται στη μορφή ζευγών εισόδου εξόδου (Σωτηρόπουλος – Τσιχριντζής, 2016). Συγκεκριμένα, κάθε δείγμα δεδομένων αποτελείται από ένα συγκεκριμένο διάνυσμα εισόδου και τη συσχετιζόμενη τιμή εξόδου. Ο πρωταρχικός σκοπός αυτής της μεθόδου εκπαίδευσης είναι να βρει μια συνάρτηση που να οδηγεί στο σωστό αποτέλεσμα όταν δίδεται στο νευρωνικό δίκτυο μια συγκεκριμένη είσοδος. Ο όρος μέθοδος με επίβλεψη χρησιμοποιείται λόγω του γεγονότος ότι τα αντικείμενα που μας ενδιαφέρουν είναι ήδη συσχετισμένα με τιμές – στόχους.
Αν τα δεδομένα είναι μόνο ένα δείγμα αντικειμένων χωρίς συσχετισμένες τιμές – στόχους τότε το πρόβλημα είναι γνωστό ως εκπαίδευση χωρίς επίβλεψη (Σωτηρόπουλος – Τσιχριντζής, 2016). Σε αυτή την περίπτωση, το δίκτυο τροποποιεί τα βάρη έτσι ώστε τα περισσότερο όμοια πρότυπα ανατίθενται στην ίδια μονάδα εξόδου ή συστάδα (Faucett, 1993).

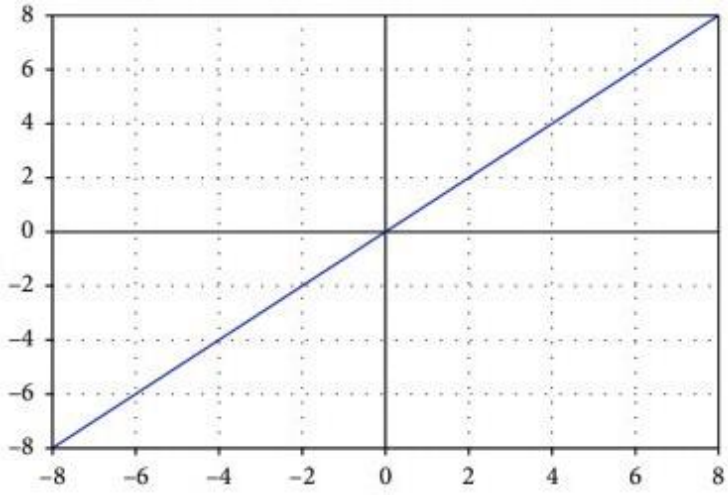
7.3 Συνάρτηση Ενεργοποίησης – Activation Function

Η βασική λειτουργία των τεχνητών νευρωνικών δικτύων περιλαμβάνει την άθροιση των γινομένων των σημάτων επί το βάρος σύναψης (weighted input signal) το οποίο εφαρμόζεται σε μια συνάρτηση ενεργοποίησης με σκοπό να παραχθεί η έξοδος. Τυπικά στους νευρώνες κάθε επιπέδου εφαρμόζεται η ίδια συνάρτηση ενεργοποίησης (Faucett, 1993). Οι συναρτήσεις αυτές είναι συνήθως μη γραμμικές ώστε να αποκομίσουμε στο μέγιστο τα οφέλη των δικτύων πολλαπλών επιπέδων καθώς η τροφοδότηση διαφορετικών επιπέδων με γραμμικώς επεξεργασμένα στοιχεία δε διαφέρει από αυτό που μπορεί να παραχθεί με τη χρήση ενός μόνο επιπέδου. Μερικές συναρτήσεις ενεργοποίησης είναι οι παρακάτω:

7.3.1 Γραμμική συνάρτηση

Χρησιμοποιείται συνήθως στους νευρώνες του επιπέδου εισόδου οι οποίοι μεταφέρουν το σήμα εισόδου χωρίς επεξεργασία. Το x αντιστοιχεί στον υπολογισμό του δυναμικού κάθε νευρώνα, το άθροισμα δηλαδή όλων των σημάτων, καθένα πολλαπλασιασμένο με το αντίστοιχο βάρος. Δίνεται από τον παρακάτω τύπο και τη γραφική της συνάρτηση τη βλέπουμε στην εικόνα 7.3:

$$f(x) = x$$

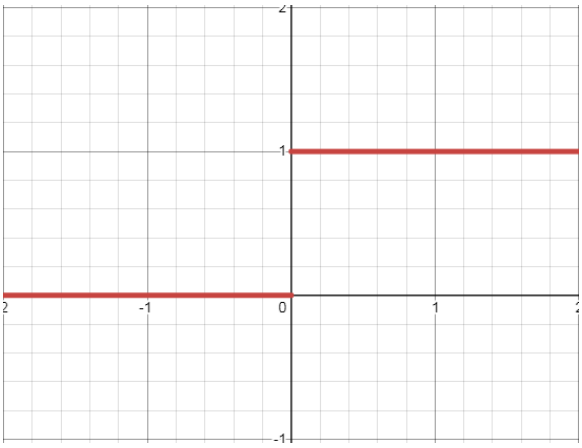


Εικόνα 7.3

7.3.2 Βηματική Συνάρτηση

Η βηματική συνάρτηση βασίζεται στην ύπαρξη κατωφλίου που σημαίνει πως ο νευρώνας ενεργοποιείται πάνω από ένα κατώφλι ενώ κάτω από αυτό παραμένει ανενεργός. Στο παρακάτω σχεδιάγραμμα το κατώφλι τίθεται στο 0. Αυτή η συνάρτηση είναι ιδιαίτερα χρήσιμη σε περιπτώσεις δυαδικής ταξινόμησης – binary classification. Ωστόσο δεν μπορεί να χρησιμοποιηθεί σε περιπτώσεις που υπάρχουν πολλαπλές τάξεις ταξινόμησης. Δίνεται από τον παρακάτω τύπο και τη γραφική της παράσταση τη βλέπουμε στην εικόνα 7.4.

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

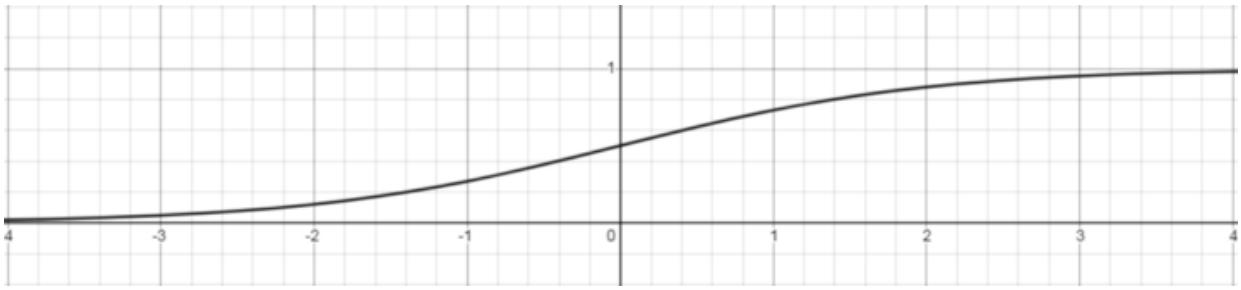


Εικόνα 7.3

7.3.3 Σιγμοειδής συνάρτηση

Η σιγμοειδής συνάρτηση έχει μια πιθανοτική προσέγγιση και η έξοδός της είναι στο εύρος από 0 έως 1. Χρησιμοποιείται συχνά σε νευρωνικά δίκτυα όπου οι επιθυμητές τιμές του αποτελέσματος είναι δυαδικές ή μεταξύ του διαστήματος 0 και 1 και είναι ιδιαιτέρως εύχρηστη όταν χρησιμοποιούμε τη μέθοδο εκπαίδευσης της οπισθοδιάδοσης λόγω της απλής της σχέσης μεταξύ της τιμής της συνάρτησης σε ένα σημείο και της της τιμής της παραγώγου σε αυτό το σημείο, μειώνοντας έτσι το υπολογιστικό βάρος (Faucett, 1993). Η παράγωγός της είναι f’(x) = f(x)(1 – f(x). Κανονικοποιεί την έξοδο κάθε νευρώνα, ωστόσο δεν κάνει σχεδόν καθόλου αλλαγές στην πρόβλεψη για πολύ μεγάλες ή πολύ μικρές εισόδους γεγονός το οποίο έχει ως αποτέλεσα το νευρωνικό δίκτυο να αρνείται να εκπαιδευτεί επιπλέον, ένα πρόβλημα που είναι γνωστό ως vanishing gradient. Δίνεται από τον παρακάτω τύπο και τη γραφική της παράσταση τη βλέπουμε στην εικόνα 7.5.

$$f(x) = \frac{1}{1 + \exp(-\sigma x)}$$

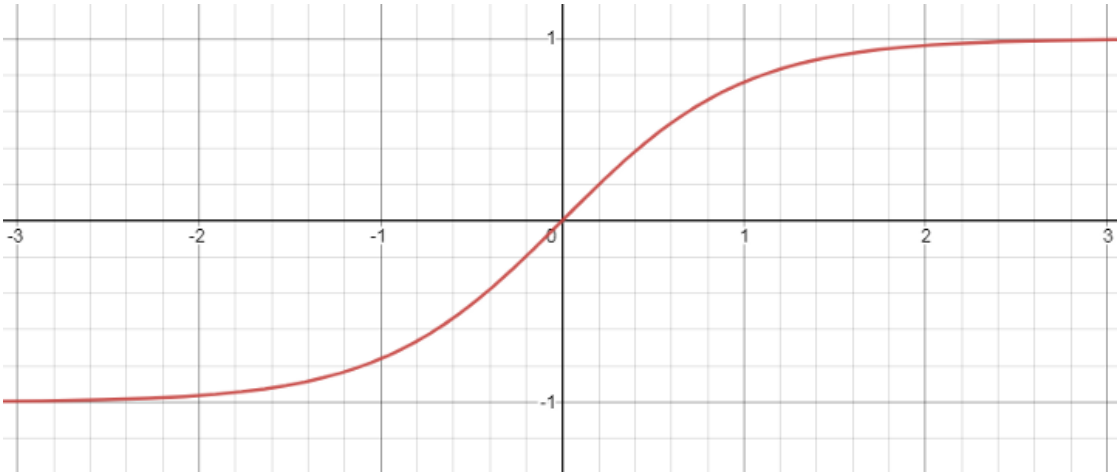


Εικόνα 7.5

7.3.4 Συνάρτηση υπερβολικής εφαπτομένης

Μοιάζει με τη σιγμοειδή συνάρτηση αλλά είναι ελαφρώς καλύτερη λόγω του γεγονότος ότι η έξοδός της έχει εύρος από -1 έως 1 επιτρέποντας έτσι τη λήψη αρνητικών αριθμών. Ωστόσο και αυτή έρχεται αντιμέτωπη με το πρόβλημα του vanishing gradient όπως ακριβώς και η σιγμοειδής. Δίνεται από τον παρακάτω τύπο και τη γραφική της παράσταση τη βλέπουμε στην εικόνα 7.6.

$$\frac{e^{2x}+1}{e^{2x}-1}$$

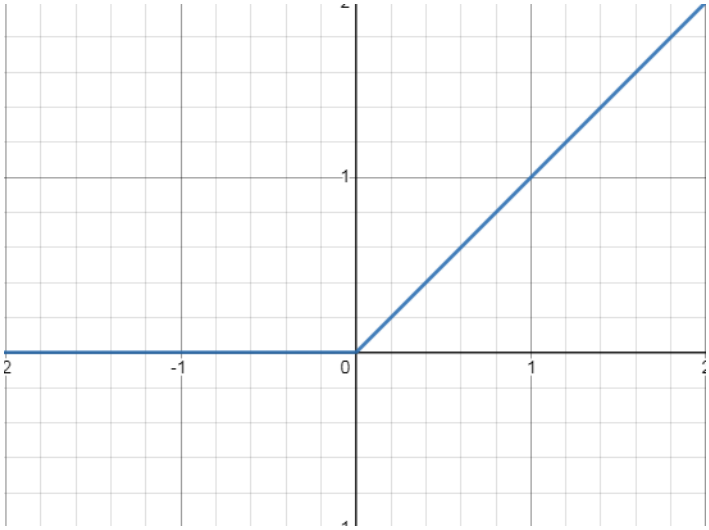


Εικόνα 7.6

7.3.5 RELU – Rectified Linear Unit Ανορθωμένη Γραμμική Μονάδα

Σε αυτή τη συνάρτηση, οι έξοδοι για τα τις θετικές εισόδους έχουν εύρος από το 0 έως το άπειρο αλλά όταν η είσοδος είναι μηδενική ή αρνητική, η συνάρτηση έχει ως έξοδο το 0 και εμποδίζει την οπισθοδιάδοση, ένα πρόβλημα γνωστό ως dying ReLU problem. Δίνεται από τον παρακάτω τύπο και τη γραφική της παράσταση τη βλέπουμε στην εικόνα 7.7.

f(x)= max(0,x)

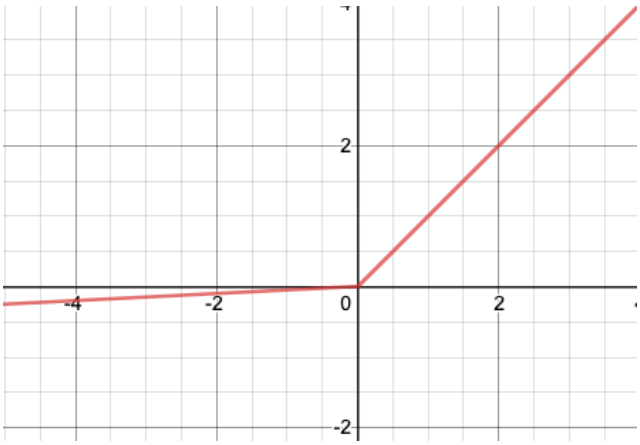


Εικόνα 7.7

7.3.6 Leaky ReLU

Η leaky ReLU είναι παραπλήσια της ReLU με τη διαφορά πως αντί να έχει έξοδο 0 για τις αρνητικές εισόδους, έχει έξοδο -αx, όταν το x είναι αρνητικό. Το α είναι μια υπερπαράμετρος και η τιμή της ποικίλει εν γένει από 0,01 έως 0,2. Αποτρέπει το πρόβλημα της dying ReLU και ενεργοποιεί την οπισθοδιάδοση. Ένα μειονέκτημά της είναι ότι η κλίση είναι προκαθορισμένη αντί να δίνεται η δυνατότητα στο νευρωνικό δίκτυο να την ανακαλύψει. Δίνεται από τον παρακάτω τύπο και τη γραφική της παράσταση τη βλέπουμε στην εικόνα 7.8.

$$y = LeakyRelU(x) = max(\alpha x, x).$$

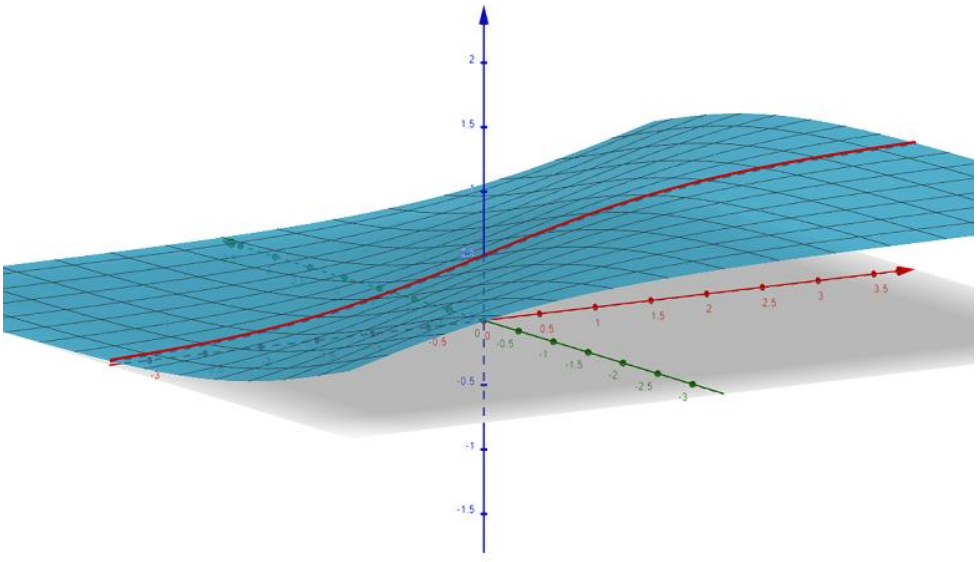


Εικόνα 7.8

7.3.7 Softmax

Η συνάρτηση Softmax χρησιμοποιείται για ταξινόμηση πολλών κλάσεων – multi-class classification, μετατρέποντας ένα διάνυσμα Κ πραγματικών αριθμών σε μια κατανομή πιθανότητας των Κ πιθανών αποτελεσμάτων. Η softmax αναθέτει μια υψηλή τιμή στο μέγιστο αποτέλεσμα που θα βρει αλλά επίσης κρατά κάποιες τιμές και για τα άλλα αποτελέσματα, ανάλογα πάντα με το μέγεθός τους. Η softmax παρομοιάζει την εφαρμογή πολλαπλών σιγμοειδών συναρτήσεων, μεταφράζοντας κάθε σιγμοειδή στην πιθανότητα να είναι σε μια συγκεκριμένη κλάση ή όχι, δίνοντας έμφαση στη μέγιστη πιθανότητα. Δίνεται από τον παρακάτω τύπο και τη γραφική της παράσταση τη βλέπουμε στην εικόνα 7.9.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



Εικόνα 7.9

7.4 Συνάρτηση Υπολογισμού Σφάλματος – Loss Function

Η αξιολόγηση των μοντέλων είναι μια βασική πτυχή της διαδικασίας ανάπτυξής τους και η συνάρτηση υπολογισμού του σφάλματος (loss function) είναι ένα μέτρο του πόσο καλά μπορεί το μοντέλο μηχανικής μάθησης να προβλέπει το αποτέλεσμα.

Η συνάρτηση υπολογισμού σφάλματος παίρνει δύο τιμές ως είσοδο: την τιμή εξόδου που παρήγαγε το μοντέλο μας και την πραγματική τιμή που επιθυμούμε να προβλέψει. Το αποτέλεσμα της συνάρτησης είναι το σφάλμα και είναι ενδεικτικό του κατά πόσο καλά λειτουργεί το μοντέλο μας.

Μία τέτοια συνάρτηση είναι αυτή του Μέσου Τετραγωνικού Σφάλματος (Mean Squared Error - MSE) και είναι ένα αξιόλογο κριτήριο της ποιότητας ενός μοντέλου που ασχολείται με προβλέψεις (Wallach & Goffinet, 1988). Η συνάρτηση δίνεται από τον παρακάτω τύπο, όπου $\hat{Y}_i$ είναι η τιμή πρόβλεψης που παρήγαγε το μοντέλο, Y_i η πραγματική τιμή που θέλουμε να προβλέψει και n ο αριθμός των δειγμάτων:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

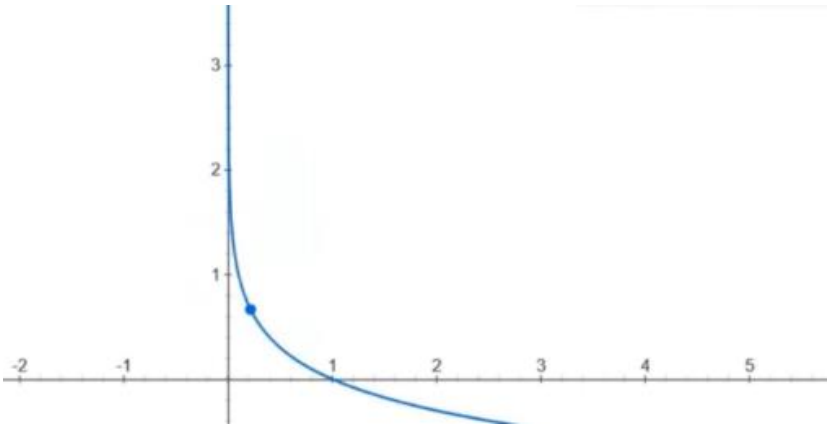
Το πλεονέκτημα της MSE είναι πως εγγυάται ότι το εκπαιδευόμενο μοντέλο μας δεν έχει «απομακρυσμένες» (outlier) προβλέψεις με τεράστια σφάλματα, καθώς η MSE βάζει μεγαλύτερο βάρος σε αυτά τα λάθη λόγω της ύψωσης στο τετράγωνο.

Το μειονέκτημα είναι πως αν το μοντέλο μας κάνει μια πολύ κακή πρόβλεψη, η ύψωση στο τετράγωνο μεγαλώνει ακόμα περισσότερο το σφάλμα. Βέβαια σε πολλές περιπτώσεις στην πράξη δεν ενδιαφερόμαστε πολύ για αυτές τις «απομακρυσμένες» τιμές (outliers) και στοχεύουμε περισσότερο σε ένα ολοκληρωμένο μοντέλο που έχει καλές επιδόσεις στην πλειοψηφία των περιπτώσεων.

Μία ακόμα συνάρτηση υπολογισμού σφάλματος είναι αυτή της Δυαδικής Διασταυρούμενης Εντροπίας (Binary Cross Entropy – BCE), η οποία δίνεται από τον παρακάτω τύπο, όπου y είναι η κλάση κατάταξης 0 ή 1, p η πρόβλεψη της πιθανότητας για την κλάση, c ο αριθμός των εξόδων που παράγει το δίκτυο για κάθε δείγμα και n ο αριθμός των δειγμάτων:

$$-\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{c}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

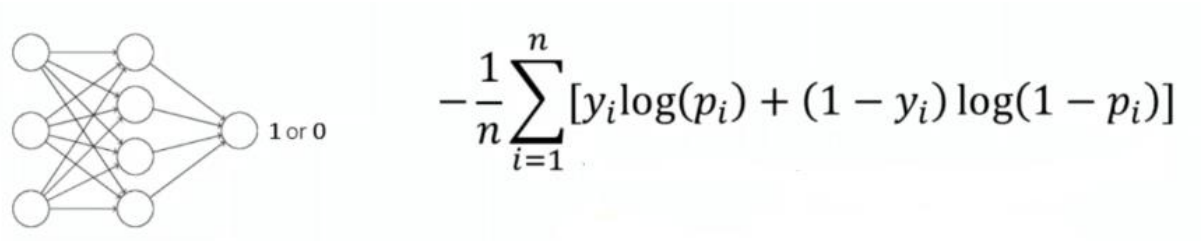
Η συνάρτηση είναι ιδιαίτερα χρήσιμη όταν τα αποτελέσματά μας κατηγοριοποιούνται σε 2 κλάσεις (πχ 0 – 1, ναι – όχι).
Μπορεί επίσης να χρησιμοποιηθεί και όταν το πραγματικό αποτέλεσμα παίρνει τιμές από 0 έως 1.
Η BCE χρησιμοποιεί την αρνητική λογαριθμική συνάρτηση η οποία έχει την παρακάτω γραφική απεικόνιση:



Εικόνα 7.10

Παρατηρούμε πως όταν η είσοδος στη συνάρτηση είναι μεγάλη, η έξοδος είναι μικρή (πχ. όταν η είσοδος είναι 1, η έξοδος είναι 0).

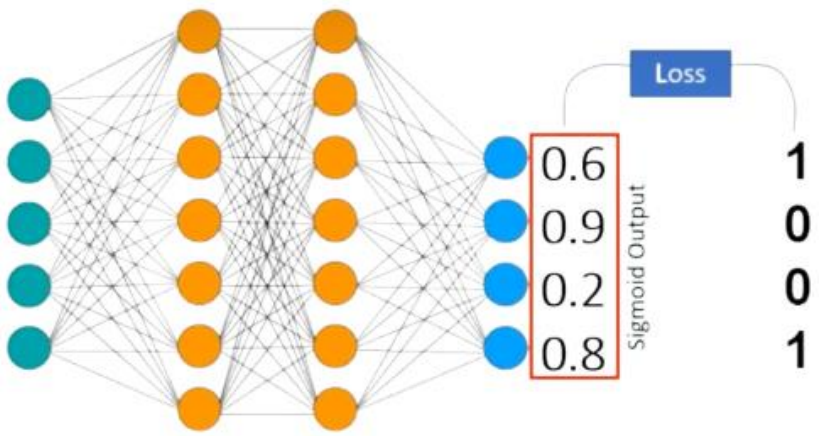
Ας δούμε μερικά παραδείγματα εφαρμογής αυτής της συνάρτησης σφάλματος BCE



Εικόνα 7.11

Σε αυτό το παράδειγμα (εικόνα 7.1), διαχωρίζουμε τα αποτελέσματα του νευρωνικού δικτύου σε δύο κλάσεις, 0 ή 1. Το αποτέλεσμα δε γίνεται να ανήκει ταυτόχρονα και στις 2 κλάσεις καθότι θα είναι είτε 0 είτε 1. Συνεπώς ο παράγοντας c είναι 1 και ο τύπος διαμορφώνεται όπως στην άνωθι εικόνα (εικόνα 7.11).
Επίσης κάθε ένα δείγμα i από τα n που υπάρχουν συνολικά θα ισχύει:
Αν το δείγμα ανήκει στην κλάση 1 τότε το 2^{ο} μέρος του τύπου μηδενίζεται καθότι 1- y_i = 1 -1 = 0 και έτσι έχουμε ως αποτέλεσμα-$y_i$log(p_i)
Αν το δείγμα ανήκει στην κλάση 0 τότε το 1^{ο} μέρος του τύπου μηδενίζεται καθότι y_i = 0 και έχουμε έτσι έχουμε ως αποτέλεσμα -log(1 - p_i)
Η διαδικασία αυτή βέβαια συνεχίζεται για όλα τα δείγματα n που έχουμε.

Στην περίπτωση που το δίκτυο χρειάζεται περισσότερους από έναν νευρώνες εξόδου με σκοπό να πραγματοποιήσει σωστή κατηγοριοποίηση των δεδομένων έχουμε τα εξής:



Εικόνα 7.12

Στο παράδειγμα της παραπάνω εικόνας (εικόνα 7.12) παρατηρούμε πως το δίκτυο παράγει 4 αποτελέσματα (κάνοντας χρήση της σιγμοειδούς συναρτήσεως ενεργοποίησης), θεωρούμε πως η σωστή κλάση κωδικοποιείται ως 1 και η λάθος ως 0 και για απλοποίηση θεωρούμε πως έχουμε ένα δείγμα δηλαδή n = 1. Συνεπώς έχουμε τον τύπο:

$$BCE\ Loss = -\sum_{i=1}^{c}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

Και το αποτέλεσμα είναι:

C	p_i	y_i
1	0.6	1
2	0.9	0
3	0.2	0
4	0.8	1

Πίνακας 1

Για c = 1:
$-[y_1 \log(p_1) + (1 - y_1)\log(1 - p_1)] =$
$-[1 \log(0.6) + (1 - 1)\log(1 - 0{,}6) =$
$-[\log(0{,}6) + 0] =$
$-\log(0.6)$

Για c = 2:
$-[y_2 \log(p_2) + (1 - y_2)\log(1 - p_2)] =$
$-[0 \log(0.9) + (1 - 0)\log(1 - 0.9) =$
$-[0 + 1 \log(0.1)] =$
$-\log(0.1)$

Ομοίως για c = 3 προκύπτει -log(0.8) και για c = 4 προκύπτει -log(0.8). Οπότε συνολικά έχουμε:
-[log(0.6) + log(0.1) + log(0.8) + log(0.8)]

Κατηγορική Διασταυρούμενη Εντροπία – Categorical Cross Entropy

Είναι το άθροισμα όλων των κατηγοριών – κλάσεων του μοντέλου (output size) για την πραγματική τιμή y_i του στόχου (label) το οποίο θα είναι 0 ή 1, επί το λογάριθμο της προβλεφθήσας πιθανότητας $\log\hat{y}_i$

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Αν για παράδειγμα υπάρχουν 2 κατηγορίες – τάξεις, πχ. μια εικόνα απεικονίζει μια γάτα ναι ή όχι, με τιμή $y_1 = 1$ όταν απεικονίζει γάτα και $y_0 = 0$ όταν δεν απεικονίζει γάτα, ο τύπος μετασχηματίζεται ως εξής:

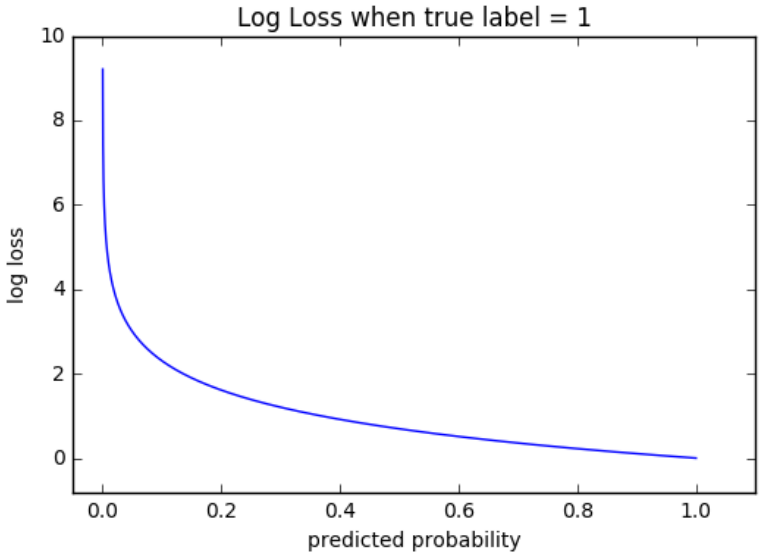
$CE = - (y_1 \log(\hat{y}_1) + y_0 \log(\hat{y}_0))$

Έστω ότι το μοντέλο προέβλεψε ότι η εικόνα απεικονίζει γάτα με πιθανότητα 1, δηλαδή $\hat{y}_1 = 1$ και $\hat{y}_0 = 0$:

$CE = - (1 \log(1) + 0 \log(0)$
$CE = - (1 \times 0 + 0 \times 1)$
$CE = 0$

Η τελική απώλεια είναι 0 καθότι το μοντέλο προέβλεψε σωστά το αποτέλεσμα με πιθανότητα 1.

Το γεγονός αυτό παρατηρείται και στη γραφική αναπαράσταση της συνάρτησης απώλειας. Η απώλεια τείνει στο 0 όσο η πρόβλεψη του μοντέλου είναι σωστή (πραγματική τιμή 1 και η πρόβλεψη του μοντέλο για την τιμή 1 τείνει στο 1 ή 100%). Από την άλλη πλευρά, αν η πραγματική τιμή είναι 1 και η πρόβλεψη του μοντέλου για την τιμή 1 τείνει στο 0%, η απώλεια μεγαλώνει τείνοντας στο άπειρο.



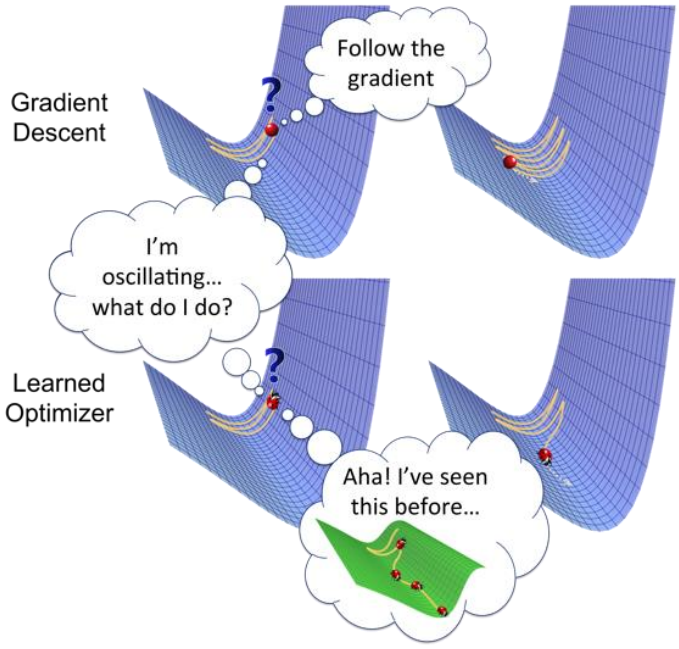
Εικόνα 7.13

7.5 Βελτιστοποίηση Νευρωνικών Δικτύων

Η βελτιστοποίηση των νευρωνικών δικτύων συνίσταται στη χρήση αλγορίθμων και μεθόδων μέσω των οποίων το δίκτυο μπορεί να βελτιώσει τα χαρακτηριστικά του όπως τα βάρη και το ρυθμό εκπαίδευσης έτσι ώστε να ελαχιστοποιηθούν τα σφάλματα και να παραχθούν τα ακριβέστερα δυνατά αποτελέσματα.

Υπάρχουν πολλοί τέτοιοι αλγόριθμοι και μέθοδοι όπως:

Η **επικλινής κάθοδος** (gradient descent). Χρησιμοποιείται ευρέως στη γραμμική παλινδρόμηση και σε αλγορίθμους ταξινόμησης (Dong & Zhou, 2007). Η μέθοδος οπισθοδιάδοσης τους λάθους χρησιμοποιεί την επικλινή κάθοδο. Ο αλγόριθμος εξαρτάται από την πρώτη παράγωγο της συναρτήσεως σφάλματος και υπολογίζει κατά ποιον τρόπο πρέπει να αλλαχθούν τα βάρη έτσι ώστε το λάθος να ελαχιστοποιηθεί.



Εικόνα 7.14 (Doshi, 2019)

Η **στοχαστική επικλινής κάθοδος** (stochastic gradient descent). Είναι παραλλαγή της επικλινούς καθόδου και προσπαθεί να ανανεώσει τις παραμέτρους του μοντέλου συχνότερα (Dong & Zhou, 2007).

Η **ορμή**, η οποία εφευρέθηκε για να μειώσει τη μεγάλη απόκλιση στη στοχαστική επικλινή κάθοδο και επιταχύνει τη σύγκλιση προς μία σχετική κατεύθυνση μειώνοντας τη διακύμανση προς την άσχετη κατεύθυνση.

Η τεχνική **ADAM – Adaptive Moment Estimation** είναι ένας αλγόριθμος βελτιστοποίησης για την επικλινή κάθοδο και είναι ιδιαιτέρως αποτελεσματική όταν υπάρχουν προβλήματα με μεγάλης κλίμακας δεδομένα ή παραμέτρους (Kingma & Lei Ba, 2015). Χρειάζεται λιγότερη μνήμη και συνδυάζει δύο μεθόδους επικλινούς καθόδου, αυτήν της ορμής και αυτήν της Root Mean Square Propagation (RMSP).

7.6 Εποχή – Επανάληψη – Ομάδα δεδομένων

Εποχή (epoch): Το νευρωνικό δίκτυο μαθαίνει το μοτίβο των δεδομένων εισόδου, διαβάζοντας αυτά τα δεδομένα και κάνοντας υπολογισμούς σε αυτά. Ωστόσο αυτό δε γίνεται μόνο μία φορά, αλλά το δίκτυο μαθαίνει συνέχεια χρησιμοποιώντας τα δεδομένα εισόδου ξανά καθώς και τα αποτελέσματα προηγούμενων προσπαθειών. Κάθε προσπάθεια του δικτύου να μάθει από τα δεδομένα εισόδου ονομάζεται εποχή. Η εποχή αναφέρεται σε έναν κύκλο που αφορά το σύνολο των δεδομένων εκπαίδευσης. Συνήθως η εκπαίδευση ενός νευρωνικού δικτύου χρειάζεται αρκετές εποχές. Βέβαια αυξάνοντας τον αριθμό των εποχών δε σημαίνει πάντα πως το δίκτυο θα βρίσκει καλύτερα αποτελέσματα (Baeldung, 2022).

Επανάληψη (iteration): Για κάθε ολοκληρωμένη εποχή υπάρχουν αρκετές επαναλήψεις. Η επανάληψη είναι ο αριθμός των ομάδων δεδομένων (batch) ή βημάτων στα χωρισμένα πακέτα των δεδομένων εκπαίδευσης που πρέπει να γίνουν ώστε να ολοκληρωθεί μια εποχή (Baeldung, 2022).

Ομάδα δεδομένων (batch): Είναι ο αριθμός των δεδομένων εκπαίδευσης που χρησιμοποιούμε σε μία επανάληψη. Όσο μεγαλύτερο είναι το μέγεθος της ομάδας δεδομένων τόσο περισσότερο χώρο στη μνήμη χρειαζόμαστε (Baeldung, 2022).

Ένα πρακτικό παράδειγμα για να την αποσαφήνιση αυτών των όρων είναι το εξής:
Έστω ότι υπάρχουν 2000 δεδομένα με βάση με τα οποία θέλουμε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο. Αυτά μπορούμε να τα χωρίσουμε σε 4 ομάδες (batches) των 500 δεδομένων. Αυτό σημαίνει πως θα έχουμε 4 επαναλήψεις (iterations) για να ολοκληρώσουμε μία εποχή (epoch).

7.7 Underfitting – Optimum - Overfitting

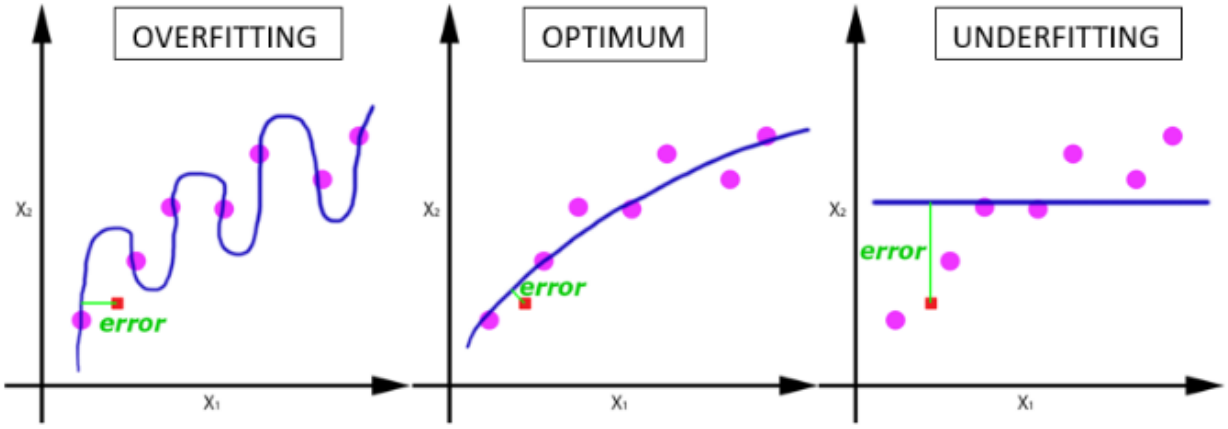
Είναι εξαιρετικά απίθανο το νευρωνικό δίκτυο να έχει εκπαιδευτεί σωστά με μία μόνο εποχή. Στην αρχή το νευρωνικό δίκτυο βρίσκεται σε μία κατάσταση όπου δεν έχει πληροφορία για τα δεδομένα ή ενδεχομένως δεν κατάφερε να εκπαιδευτεί σωστά και η πληροφορία που έχει δημιουργήσει δεν περιγράφει επαρκώς τα δεδομένα (πχ. αν το μοντέλο έχει εκπαιδευτεί μόνο για μία εποχή είναι πολύ πιθανό να μην επιφέρει τα επιθυμητά αποτελέσματα). Η κατάσταση αυτή ονομάζεται **underfitting**. Ένας τρόπος ασφαλούς διάγνωσης της κατάστασης overfitting είναι με τη δημιουργία ενός συνόλου δεδομένων επικύρωσης (validation test) και τη μελέτη συμπεριφοράς του. Το σύνολο αυτό υπάρχει μαζί με το σύνολο δεδομένων προς εκπαίδευσης (training set) χωρίς το validation set να ακυρώνει ή υποκαθιστά τη λειτουργία του training set.

Ένα νευρωνικό δίκτυο που έχει σχεδιαστεί ώστε να πραγματοποιεί ορθές γενικεύσεις, θα κάνει μια σωστή χαρτογράφηση εισόδου – εξόδου ακόμα και όταν η είσοδος είναι ελαφρώς διαφορετική από τα παραδείγματα που χρησιμοποιήθηκαν ώστε να εκπαιδευτεί το δίκτυο (Hayking, 2009). Η κατάσταση αυτή ονομάζεται **βέλτιστη** ή **optimum**.

Ωστόσο, όταν ένα νευρωνικό δίκτυο μαθαίνει υπερβολικά πολλά παραδείγματα εισόδων – εξόδου, το δίκτυο ενδέχεται να καταλήξει να απομνημονεύσει τα δεδομένα εκπαίδευσης. Ενδέχεται να πράξει τοιουτοτρόπως βρίσκοντας ένα χαρακτηριστικό (ίσως λόγω θορύβου) που είναι παρόν στα δεδομένα εκπαίδευσης, όχι όμως αληθές για τη λειτουργία που πρέπει να μοντελοποιήσει. Αυτό το φαινόμενο ονομάζεται **overfitting** ή **overtraining** και σε αυτή την περίπτωση το δίκτυο χάνει την ικανότητά του να γενικοποιεί παρόμοια σχέδια εισόδου – εξόδου (Hayking, 2009). Εν γένει, όσο πιο περίπλοκο είναι ένα

σύστημα, τόσο περισσότερο ρέπει προς το overfitting καθιστώντας τα τεχνητά νευρωνικά δίκτυα ιδιαίτερα επιρρεπή σε αυτό το πρόβλημα.

Τα φαινόμενα των καταστάσεων overfitting, βέλτιστης (optimum) και underfitting φαίνονται στο παρακάτω σχεδιάγραμμα (εικόνα 7.15).



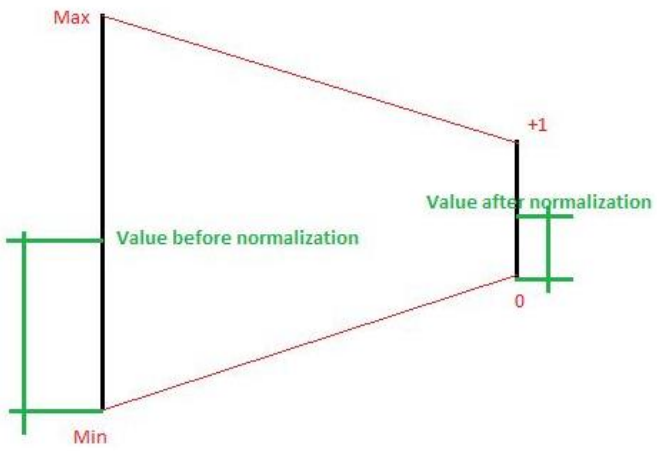
Εικόνα 7.15 (Sharma, 2017)

7.8 Κανονικοποίηση Δεδομένων

Η κανονικοποίηση των δεδομένων είναι μια τεχνική κλιμακοποίησής τους σε ένα πρώιμο στάδιο που μας επιτρέπει να βρούμε ένα νέο εύρος τιμών από ένα υπάρχον εύρος και είναι ιδιαίτερα χρήσιμη σε περιπτώσεις προβλέψεων (Patro & Sahu, 2015). Με αυτήν τη διαδικασία αποφεύγονται στρεβλώσεις όπως για παράδειγμα το γεγονός να δίνεται βαρύτητα μόνο στην τιμή ενός χαρακτηριστικού και όχι και στη μονάδα μέτρησής του (πχ. 5 χιλιόγραμμα - κιλά είναι το ίδιο με 5000 γραμμάρια ωστόσο αν αγνοηθεί η μονάδα μέτρησης η αριθμητική διαφορά του 5 από το 5000 είναι τεράστια παρόλο που πρακτικά απεικονίζουν την ίδια ποσότητα) και εν γένει αποτρέπουμε τους αλγορίθμους να δίνουν μεγαλύτερη βαρύτητα σε χαρακτηριστικά των οποίων οι αριθμητικές τιμές είναι μεγάλες, αγνοώντας ή μη δίνοντας την πρέπουσα βαρύτητα και σε χαρακτηριστικά των οποίων οι αριθμητικές τιμές είναι χαμηλότερες.

7.8.1 Κανονικοποίηση – Normalisation

Σε αυτή τη διαδικασία τα δεδομένα κλιμακοποιούνται σε ένα συγκεκριμένο εύρος (πχ από 0 έως 1 ή από -1 έως 1). Η διαδικασία είναι απαραίτητη όταν υπάρχουν μεγάλες διαφορές στα εύρη των διαφορετικών χαρακτηριστικών και είναι χρήσιμη όταν τα δεδομένα δεν έχουν ιδιαίτερα απομακρυσμένες τιμές – outliers (Peshawa & Rezhna, 2014). Μία γραφική αναπαράσταση της διαδικασίας κανονικοποίησης των δεδομένων φαίνεται στην εικόνα 7.16.



Εικόνα 7.16. Κανονικοποίηση δεδομένων σε εύρος 0 έως 1 (Peshawa & Rezhna, 2014)

Για να κανονικοποιήσουμε τα δεδομένα μπορούμε να χρησιμοποιήσουμε τον παρακάτω τύπο όπου x' είναι η τιμή του δεδομένου μετά την κανονικοποίηση, x η τιμή πριν την κανονικοποίηση, min και max η ελάχιστη και μέγιστη τιμή των δεδομένων που έχουμε αντίστοιχα:

$$x' = \frac{x - min}{max - min}$$

7.8.2 Κανονικοποίηση z-score (Z-score standardization)

Σε αυτήν τη διαδικασία δημιουργούμε ένα σετ δεδομένων με μέσο όρο 0 και τυπική απόκλιση 1 και αυτή η μέθοδος κλιμακοποίησης είναι χρήσιμη όταν τα δεδομένα ακολουθούν την κανονική κατανομή (Gaussian distribution – normal distribution) (Peshawa & Rezhna, 2014).

Ο υπολογισμός γίνεται από τον παρακάτω τύπο όπου x' είναι η τιμή του δεδομένου μετά την κανονικοποίηση z-score, $\bar{x}$ ο μέσος όρος και σ η τυπική απόκλιση:

$$x' = (x - \bar{x}) / \sigma$$

Ο μέσος όρος $\bar{x}$ (mean) ενός συνόλου N δεδομένων υπολογίζεται από τον τύπο:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Η τυπική απόκλιση σ ενός συνόλου n δεδομένων και μέσου όρου $\bar{x}$ υπολογίζεται από τον τύπο:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Συνοψίζοντας μπορούμε να πούμε τα παρακάτω για τις 2 αυτές μεθόδους:

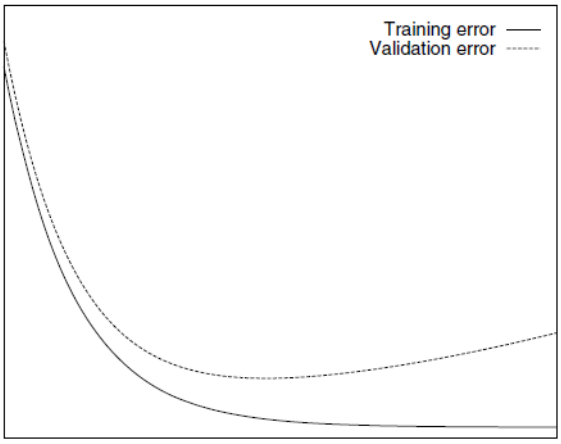
Normalisation	Standardisation
Η μέγιστη και η ελάχιστη τιμή των δεδομένων χρησιμοποιούνται για την κλιμακοποίηση	Ο μέσος και η τυπική απόκλιση χρησιμοποιούνται για την κλιμακοποίηση
Κλιμακοποιεί στο διάστημα [0, 1] ή [-1, 1]	Δε φράσσεται σε κάποιο εύρος
Επηρεάζεται σημαντικά από απομακρυσμένες τιμές (outliers)	Επηρεάζεται λιγότερο από απομακρυσμένες τιμές (outliers)
Στο πακέτο scikit learn μπορούμε να τη βρούμε ως MinMaxScaler	Στο πακέτο scikit learn μπορούμε να τη βρούμε ως StandardScaler
Χρήσιμο όταν δε γνωρίζουμε την κατανομή	Χρήσιμη όταν έχουμε κανονική (Gauss) κατανομή δεδομένων
Συναντάται συχνά ως Scaling Normalisation	Συναντάται συχνά ως Z-Score Normalisation

Πίνακας 2

7.8.3 Πρόωρο Σταμάτημα – Early Stopping

Μία τεχνική για να αποφευχθεί η κατάσταση overfitting είναι το πρόωρο σταμάτημα – early stopping. Η συγκεκριμένη τεχνική είναι η εξής (Prechelt, 1998):

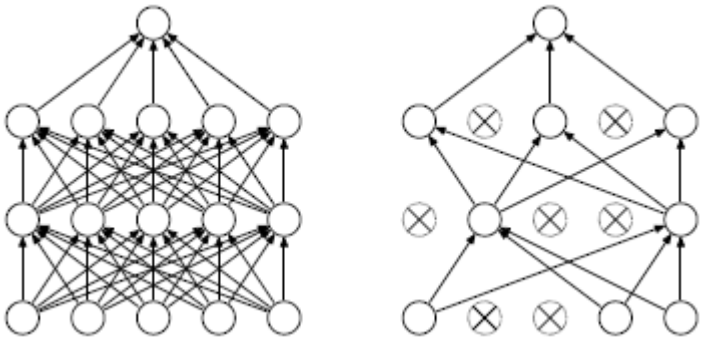
- Χωρισμός των δεδομένων εκπαίδευσης (training data) σε σύνολο εκπαίδευσης (training set) και σύνολο επικύρωσης (validation set) με κάποια αναλογία πχ. 2-1.
- Εκπαίδευση μόνο στο training set και εκτίμηση κατά διαστήματα στο λάθος ανά παράδειγμα το στο validation set πχ. κάθε $5^{η}$ εποχή.
- Τερματισμός εκπαίδευσης μόλις το σφάλμα στο validation set είναι μεγαλύτερο από ότι ήταν την τελευταία φορά που ελέγχθηκε.
- Χρήση των βαρών που είχε το δίκτυο στο προηγούμενο βήμα ως αποτέλεσμα της διαδικασίας εκπαίδευσης.



Εικόνα 7.17 Ιδανικές καμπύλες σφάλματος για training & validation. Οριζόντιος άξονας: χρόνος, κάθετος άξονας: σφάλματα (Prechelt, 1998)

7.8.4 Dropout

Η τεχνική dropout συνίσταται στην τυχαία αγνόηση μερικών νευρώνων και όλων των συνδέσεών τους κατά τη διάρκεια της εκπαίδευσης. Σε κάθε βήμα εκπαίδευσης μπορούν να αγνοηθούν διαφορετικοί νευρώνες. Με αυτόν τον τρόπο το τεχνητό νευρωνικό δίκτυο συρρικνώνεται και γίνεται λιγότερο περίπλοκο. Επιπλέον, με δεδομένο ότι κάθε φορά κάποιοι τυχαίοι νευρώνες δε λαμβάνονται υπόψιν, κάθε νευρώνας του δικτύου προσπαθεί να μη δίνει υπερβολικό βάρος σε συγκεκριμένες εισόδους.



Εικόνα 7.18. Αριστερά: TNN χωρίς dropout, Δεξιά: TNN με dropout (Srivastava et. al., 2014)

7.9 Παράμετροι Parameters – Υπερπαράμετροι Hyperparameters

Παράμετρος είναι μια μεταβλητή παραμετροποίησης που είναι εσωτερική στο μοντέλο και της οποίας η τιμή μπορεί να εκτιμηθεί από τα δεδομένα. Χαρακτηριστικά των παραμέτρων είναι:

- Χρειάζονται από το μοντέλο όταν κάνει προβλέψεις
- Η τιμή τους καθορίζει την ικανότητα του μοντέλου πάνω σε ένα πρόβλημα
- Εκτιμώνται ή μαθαίνονται από δεδομένα
- Συνήθως δεν καθορίζονται χειρωνακτικά από το χρήστη
- Συνήθως σώζονται ως μέρος του μοντέλου
- Οι τελικές παράμετροι που θα βρεθούν μετά την εκπαίδευση θα αποφασίσουν το πως το μοντέλο θα αποδώσει σε δεδομένα που δεν έχει ξαναδεί

Παραδείγματα παραμέτρων είναι τα βάρη ενός νευρωνικού δικτύου ή οι συντελεστές – coefficients στην παλινδρόμηση.

Η υπερπαράμετρος είναι μεταβλητή παραμετροποίησης που είναι εξωτερική στο μοντέλο και της οποίας η τιμή δεν μπορεί να εκτιμηθεί από τα βάρη. Χαρακτηριστικά τους είναι:

- Συχνά χρησιμοποιούνται σε διαδικασίες ώστε να βοηθήσουν να γίνει εκτίμηση των παραμέτρων του μοντέλου
- Συχνά καθορίζονται από τον χρήστη
- Συχνά μπορούν να καθοριστούν χρησιμοποιώντας ευρετικές μεθόδους
- Συχνά ρυθμίζονται για ένα πρόβλημα μοντελοποίησης πρόβλεψης
- Η επιλογή των υπερπαραμέτρων καθορίζει το πόσο αποτελεσματική είναι η εκπαίδευση. Για παράδειγμα στη μέθοδο του gradient descent, το learning rate καθορίζει πόσο αποτελεσματική και ακριβής είναι η διαδικασία βελτιστοποίησης στην εκτίμηση των παραμέτρων

Δεν μπορούμε να γνωρίζουμε τη βέλτιστη τιμή για την υπερπαράμετρο ενός μοντέλου για ένα πρόβλημα. Μπορούμε να χρησιμοποιήσουμε κάποιους εμπειρικούς κανόνες, να αντιγράψουμε τιμές που χρησιμοποιήθηκαν σε άλλα προβλήματα ή να αναζητήσουμε τη βέλτιστη τιμή μέσα από δοκιμές. Παραδείγματα υπερπαραμέτρων είναι ο ρυθμός εκπαίδευσης, τα επίπεδα και οι μονάδες των κρυφών επιπέδων των νευρωνικών δικτύων, ο αριθμός των εποχών, η επιλογή της συνάρτησης ενεργοποίησης, το μέγεθος του batch και η χρήση ορμής – momentum.

Διαφορετικά μοντέλα χρησιμοποιούν διαφορετικές υπερπαραμέτρους (Yang & Shami, 2020) και ο χρόνος εκπαίδευσης και δοκιμής ενός μοντέλου εξαρτάται σημαντικά από την επιλογή των υπερπαραμέτρων (Claesen & De Moor, 2015).

7.10 Αύξηση Δεδομένων – Data Augmentation

Η αύξηση δεδομένων είναι μια διαδικασία τεχνητής αύξησης της ποσότητας των δεδομένων, δημιουργώντας νέα δεδομένα από τα ήδη υπάρχοντα. Η διαδικασία αυτή συμπεριλαμβάνει και αλλαγές ελάσσονος σημασίας στα δεδομένα ή τη χρήση μοντέλων μηχανικής μάθησης για τη δημιουργία νέων δεδομένων στο λανθάνοντα χώρο των αυθεντικών δεδομένων. Η αύξηση των δεδομένων βελτιώνει τις επιδόσεις των μοντέλων μηχανικής μάθησης καθώς έτσι τα σύνολα δεδομένων γίνονται πιο ποικιλόμορφα και λαμβάνει χώρα σε πληθώρα εφαρμογών όπως η ανίχνευση

αντικειμένων και η κατηγοριοποίηση ή αναγνώριση εικόνων (Shorten & Khoshgoftaar, 2019). Η αύξηση των δεδομένων που αφορούν εικόνες μπορεί να γίνει με διάφορες τεχνικές όπως:

- Κόψιμο σε κάποιο συγκεκριμένο σημείο (πχ στο κέντρο) ή τυχαίο
- Περιστροφή
- Αλλαγή της φωτεινότητας
- Αλλαγή contrast
- Αλλαγή saturation

8. Τεχνητά Νευρωνικά Δίκτυα Αισθητήρα

8.1 Στοιχειώδης Αισθητήρας – Perceptron

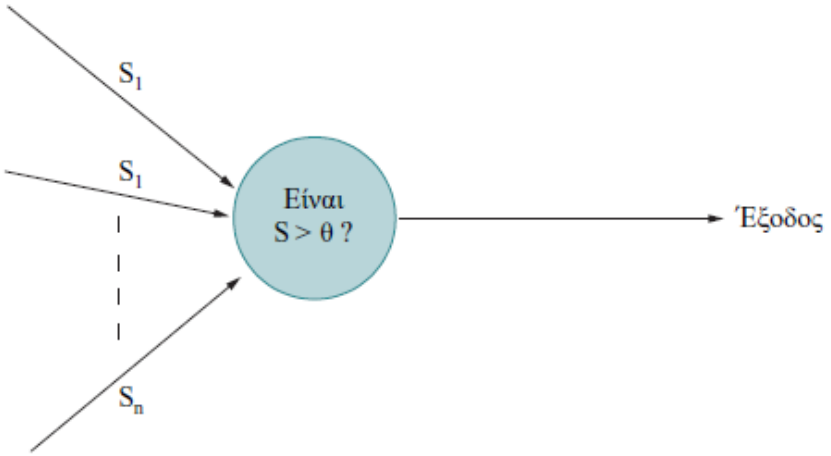
Ένα από τα πρώτα τεχνητά νευρωνικά δίκτυα που αναπτύχθηκαν είναι το μοντέλο του αισθητήρα (perceptron) το οποίο στην απλούστερή του μορφή (elementary perceptron) αποτελείται από μόνο έναν νευρώνα και είναι το πιο απλό και αυτοδύναμο σύστημα που υπάρχει και επιτελεί μια ορισμένη διεργασία (Αργυράκης, 2001). Έχει ορισμένο αριθμό εισόδων αλλά μία μόνο έξοδο. Κάθε εισερχόμενο σήμα συνδέεται με το νευρώνα με ένα βάρος w. Το δυναμικό S του νευρώνα προκύπτει από το άθροισμα των εισόδων s_i, κάθε μία πολλαπλασιασμένη με το αντίστοιχο βάρος της w_i, δηλαδή:

$$S = \sum_{i}^{n} s_i w_i$$

Μερικές φορές θεωρούμε ότι εκτός από τα εισερχόμενα σήματα και τα αντίστοιχα βάρη, ο νευρώνας έχει και ένα εσωτερικό βάρος που λέγεται «bias», b ή αλλιώς προδιάθεση ή παράγων προδιάθεσης του νευρώνα (Αργυράκης, 2001). Το βάρος αυτό είναι ξεχωριστό από τα υπόλοιπα αλλά δρα με τον ίδιο τρόπο, οπότε ο υπολογισμός του δυναμικού του νευρώνα δίνεται από τον τύπο:

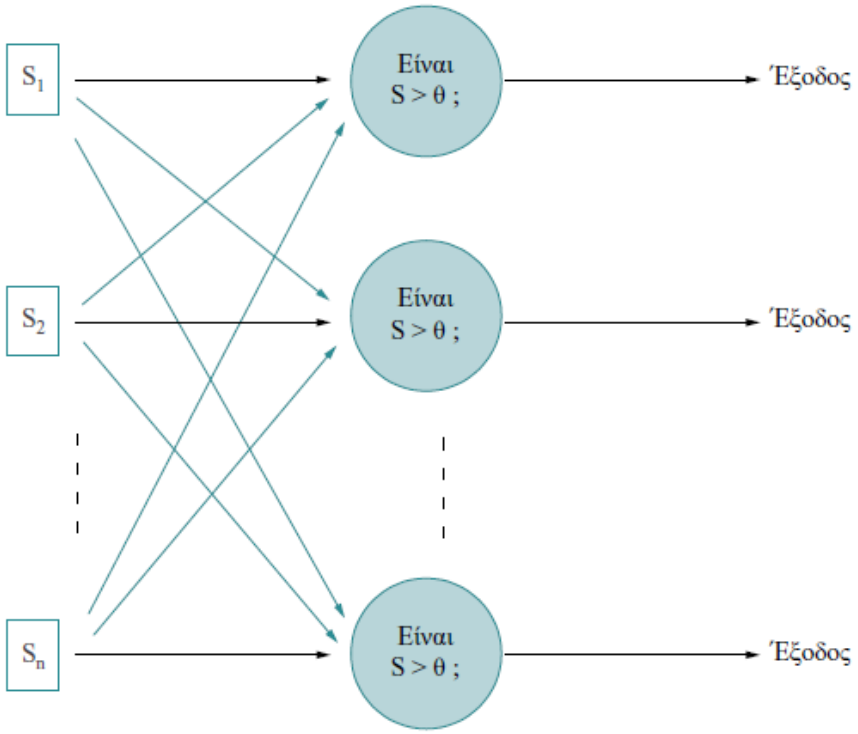
$$S = b + \sum_{i}^{n} s_i w_i$$

Αφού υπολογιστεί το δυναμικό του νευρώνα, στη συνέχεια περνάει ως παράμετρος στη συνάρτηση ενεργοποίησης που στην περίπτωση του perceptron είναι η βηματική συνάρτηση (στα δίκτυα τύπου αισθητήρα χρησιμοποιούνται συναρτήσεις ενεργοποίησης δυαδικής μορφής, Αυγενάκης 2001) και τέλος παράγεται η έξοδος.
Τον στοιχειώδη αισθητήρα perceptron μπορούμε να τον δούμε και στην παρακάτω εικόνα.



Εικόνα 8.1 (Αργυράκης, 2001)

Μπορούν να αναπτυχθούν και πιο προχωρημένα πρότυπα με περισσότερους του ενός νευρώνες και έτσι οδηγούμαστε στο μοντέλο του αισθητήρα με n νευρώνες όπως φαίνεται και στην παρακάτω εικόνα:



Εικόνα 8.2 (Αργυράκης, 2001). Μοντέλο αισθητήρα με n νευρώνες. Οι νευρώνες εισόδου συμβολίζονται με το σχήμα του τετραγώνου στα αριστερά και σε αυτούς δε συμβαίνει κάποια επεξεργασία αλλά χρησιμεύουν για τη λήψη του σήματος.

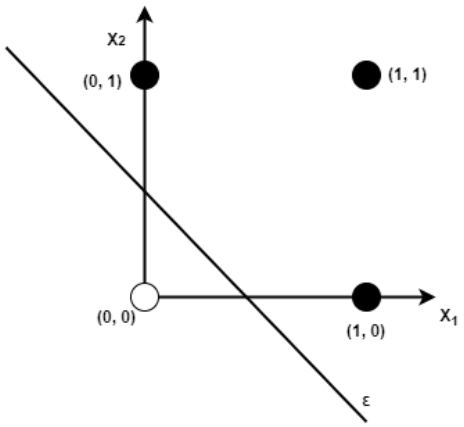
Από τον τρόπο λειτουργίας του αισθητήρα αντιλαμβανόμαστε πως η ενεργητικότητά του εξαρτάται από τα βάρη των συνδέσεων, τις τιμές των εισόδων και την τιμή κατωφλίου θ. Αυτό που μαθαίνει το σύστημα μέσω της εκπαίδευσης αποθηκεύεται στα βάρη των συνδέσεων τα οποία μεταβάλλονται κατά τη διαδικασία της εκπαίδευσης.

8.2 Εφαρμογή στη Λογική Πύλη OR

Ένα παράδειγμα πρακτικής εφαρμογής της απλούστερης μορφής αισθητήρα (elementary perceptron) είναι αυτό της λογικής πύλης OR, η οποία έχει δύο εισόδους X_1, X_2 και παράγει μία έξοδο Y. Η έξοδος Y είναι 0 όταν και οι δύο είσοδοι είναι 0 και 1 σε κάθε άλλη περίπτωση, όπως φαίνεται και στον ακόλουθο πίνακας αλήθειας:

Πίνακας Αλήθειας OR		
Είσοδοι		Έξοδος
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

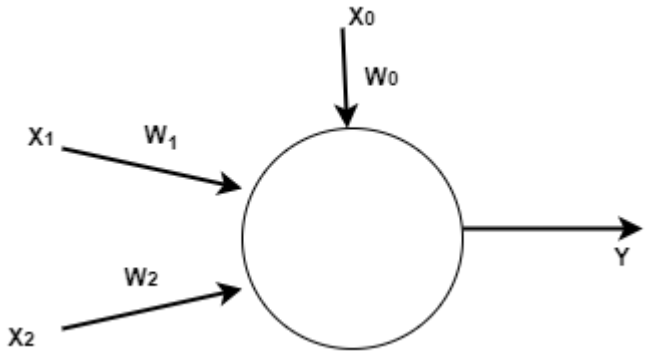
Πίνακας 3



Εικόνα 8.3. Το αποτέλεσμα 1 της εξόδου Y από τον πίνακα αλήθειας στα αριστερά (πίνακας 3), αναπαρίσταται με μαύρη κουκίδα στο σχήμα στα δεξιά και το αποτέλεσμα 0 με λευκή. Στις παρενθέσεις του σχήματος στα δεξιά έχουμε τις εισόδους X_1 και X_2 από τον πίνακα αλήθειας της OR και ε είναι η ευθεία του επιπέδου που διαχωρίζει τις δύο κλάσεις 0 και 1

Μιας και έχουμε 2 εισόδους, μπορούμε να αναπαραστήσουμε τον πίνακα αλήθειας της λογικής πύλης OR στους 2 άξονες που ορίζουν το επίπεδο. Παρατηρούμε στον πίνακα 3 πως οι τιμές της εξόδου Y είναι 0 ή 1, μπορούμε να θεωρήσουμε λοιπόν πως έχουμε δύο κλάσεις για το αποτέλεσμα εξόδου, τη 0 και την 1. Το επιθυμητό είναι ο νευρώνας να μπορεί να διαχωρίσει σωστά τα δεδομένα εισόδου στις δύο αυτές κλάσεις, να αποφασίσει δηλαδή για το ποια μπορεί να είναι η ευθεία ε στο δεξιό σχήμα της εικόνας 8.3. Οι μαύρες κουκίδες αντιστοιχούν σε αποτέλεσμα εξόδου Y = 1 και η λευκή σε Y = 0. Έτσι λοιπόν μπορούμε να πούμε πως το ημιεπίπεδο πάνω από την ευθεία ε θα είναι η κλάση για έξοδο 1 και κάτω από την ε η κλάση για έξοδο 0.

Συνοψίζοντας, έχουμε την είσοδο X_1 στην οποία αντιστοιχεί το βάρος W_1, την είσοδο X_2 στην οποία αντιστοιχεί το βάρος W_2 και την είσοδο X_0 την οποία θέτουμε ίση με -1 και το αντίστοιχο βάρος της W_o το οποίο ισοδυναμεί στο κατώφλι θ και έτσι οδηγούμαστε στο σχήμα της εικόνας 8.4.



Εικόνα 8.4

Με δεδομένο τον πίνακα αλήθειας της λογικής πύλης OR (πίνακας 3 - εικόνα 8.3-) ας υπολογίσουμε το δυναμικό του νευρώνα ανάλογα με τα σήματα στην είσοδο

Περίπτωση	X_1	X_2	Y	$X_1*W_1 + X_2*W_2 + X_0*W_0$ ή Δυναμικό
1	0	0	0	$0* W_1 + 0*W_2 - 1*\theta = -\theta$
2	0	1	1	$0* W_1 + 1*W_2 - 1*\theta = W_2 - \theta$
3	1	0	1	$1* W_1 + 0*W_2 - 1*\theta = W_1 - \theta$
4	1	1	1	$1* W_1 + 1*W_2 - 1*\theta = W_1 + W_2 - \theta$

Πίνακας 4

1η Προσέγγιση:

Θέλουμε να βρούμε την εξίσωση ευθείας ε (εικόνα 8.3) που θα διαχωρίζει κατά τον επιθυμητό τρόπο το επίπεδο σε 2 ημιεπίπεδα. Αν πάρουμε τα σημεία A(0.5, 0) και B(0, 0.5) τότε οδηγούμαστε στην παρακάτω εξίσωση ευθείας η οποία διαχωρίζει σωστά το επίπεδο:

$$\frac{x - x_1}{x_1 - x_2} = \frac{y - y_1}{y_1 - y_2}$$

$$\frac{x - 0.5}{0.5 - 0} \quad \frac{y - 0}{0 - 0.5}$$

$$0.5x + 0.5y - 0.25 = 0$$

Και δεδομένου ότι $X_1*W_1 + X_2*W_2 + X_0*W_0$ έχουμε $W_1 = 0.5$, $W_2 = 0.5$ και $\theta = 0.25$. Άρα βρήκαμε τα βάρη τα οποία αν εφαρμοστούν μας οδηγούν σε σωστά αποτελέσματα.

Περίπτωση	X_1	X_2	Y	$X_1*0.5 + X_2*0.5 + (-1)*0.25$	Βηματική Συνάρτηση
1	0	0	0	$0*0 + 0*0 -0.25 = -0.25 < 0$	0
2	0	1	1	$0*0 + 1*0.5 -0.25 = 0.25 > 0$	1
3	1	0	1	$1*0.5 + 0*0 -0.25 = 0.25 > 0$	1
4	1	1	1	$1*0.5 + 1*0.5 -0.25 = 0.75 > 0$	1

Πίνακας 5

Υπενθύμιση: η βηματική συνάρτηση ενεργοποίησης δίνει αποτέλεσμα 0 όταν το δυναμικό του νευρώνα είναι μικρότερο του 0 και 1 όταν το δυναμικό είναι μεγαλύτερο ή ίσο από το 0.

Συνεπώς η επιθυμητή ευθεία ε που διαχωρίζει το επίπεδο στις επιθυμητές κλάσεις της εικόνας 8.3 είναι η $0.5X_1 + 0.5X_2 -0.25$

2η Προσέγγιση:

Στην 1η περίπτωση του πίνακα 4 θέλουμε το αποτέλεσμα Y της εξόδου να είναι 0.
Στη βηματική συνάρτηση ενεργοποίησης αυτό συμβαίνει όταν το δυναμικό είναι μικρότερο του 0, δηλαδή $-\theta < 0$ ή $\theta > 0$.

Στη $2^η$ περίπτωση θέλουμε το αποτέλεσμα Y της εξόδου να είναι 1.
Στη βηματική συνάρτηση ενεργοποίησης αυτό συμβαίνει όταν το δυναμικό είναι μεγαλύτερο ή ίσο με το 0, δηλαδή $W_2 – θ ≥ 0$ ή $W_2 ≥ θ$.

Στην $3^η$ περίπτωση θέλουμε το αποτέλεσμα Y της εξόδου να είναι 1.
Στη βηματική συνάρτηση ενεργοποίησης αυτό συμβαίνει όταν το δυναμικό είναι μεγαλύτερο ή ίσο με το 0, δηλαδή $W_1 – θ ≥ 0$ ή $W_1 ≥ θ$.

Στη $4^η$ περίπτωση θέλουμε το αποτέλεσμα Y της εξόδου να είναι 1. Στη βηματική συνάρτηση ενεργοποίησης αυτό συμβαίνει όταν το δυναμικό είναι μεγαλύτερο ή ίσο με το 0, δηλαδή $W_1 + W_2 - θ ≥ 0$ ή $W_1 + W_2 ≥ θ$.

Συνοψίζοντας οδηγούμαστε στον πίνακα 6:

Περίπτωση	X_1	X_2	Y	$X_1*W_1 + X_2*W_2 + X_0*W_0$ ή Δυναμικό	Επιθ. Αποτ. Βηματικής f	Επιθυμητή Μορφή Δυναμικού
1	0	0	0	$0*W_1 + 0*W_2 - 1*θ = -θ$	0	$-θ < 0$ ή $θ > 0$
2	0	1	1	$0*W_1 + 1*W_2 - 1*θ = W_2 - θ$	1	$W_2 – θ > 0$ ή $W_2 > θ$
3	1	0	1	$1*W_1 + 0*W_2 - 1*θ = W_1 - θ$	1	$W_1 – θ > 0$ ή $W_1 > θ$
4	1	1	1	$1*W_1 + 1*W_2 - 1*θ = W_1 + W_2 - θ$	1	$W_1 + W_2 – θ > 0$ ή $W_1 + W_2 > θ$

Πίνακας 6

Αν θέσουμε $θ = 0.25$, $W_1 = 0.5$ και $W_2 = 0.5$ ικανοποιούνται όλες οι παραπάνω ανισώσεις.

Τιμές	Περίπτωση	Επιθυμητή Μορφή Δυναμικού
$W_1 = 0.5$ $W_2 = 0.5$ $θ = 0.25$	1	$θ > 0$ ή $0.25 > 0$ που ισχύει
	2	$W_2 > θ$ ή $0.5 > 0.25$ που ισχύει
	3	$W_1 > θ$ ή $0.5 > 0.25$ που ισχύει
	4	$W_1 + W_2 > θ$ ή $0.5 + 0.5 > 0.25$ που ισχύει

Πίνακας 7

Οδηγούμαστε δηλαδή στην ίδια μορφή εξίσωσης ευθείας που βρήκαμε και στην $1^η$ προσέγγιση καθότι

$X_1*W_1 + X_2*W_2 + X_0*W_0 =$
$X_1*0.5 + X_2*0.5 + (-1)*0.25 =$
$0.5X_1 + 0.5X_2 - 0.25$

Η ευθεία αυτή είναι η ευθεία ε η οποία διαχωρίζει το επίπεδο στις 2 επιθυμητές κλάσεις της εικόνας 8.3.

Να επισημάνουμε πως και στις 2 προσεγγίσεις βρίσκουμε μία από τις πολλές ευθείες που υπάρχουν και ικανοποιούν τους περιορισμούς μας.

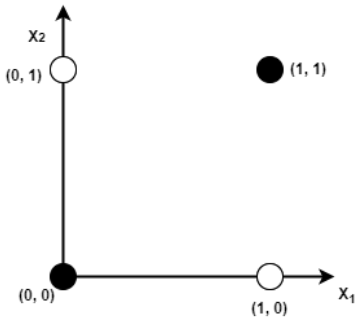
9. Γραμμική Διαχωρισιμότητα

9.1 Η Λογική Πύλη XOR

Ας υποθέσουμε πως θέλουμε να προσεγγίσουμε τη λογική πύλη του αποκλειστικού OR ή XOR με τη χρήση νευρωνικών δικτύων. Ο πίνακας αλήθειας και μια γραφική αναπαράσταση στο επίπεδο των εξόδων για την πύλη XOR φαίνονται στην εικόνα 9.1.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Πίνακας 8



Εικόνα 9.1

Παρατηρούμε πως δεν υπάρχει μία ευθεία η οποία από μόνη της να μπορεί να διαχωρίσει τα σημεία στις κλάσεις που θέλουμε, συνεπώς ένας μόνο νευρώνας που δημιουργεί μία μόνο ευθεία δεν είναι αρκετός. Προβλήματα παρόμοια με αυτή της περίπτωσης XOR που δεν μπορούν να παρασταθούν με το δίκτυο ενός μόνο νευρώνα ονομάζονται γραμμικά μη διαχωρίσιμα (Αργυράκης, 2001). Για το λόγο αυτό πρέπει να δημιουργήσουμε πιο πολύπλοκες δομές νευρωνικών δικτύων. Ένα μοντέλο νευρώνων 2 επιπέδων μπορεί να ξεχωρίσει σημεία που περιλαμβάνονται σε ανοικτές ή κλειστές κυρτές περιοχές και σε μοντέλα 3 επιπέδων δεν υπάρχουν περιορισμοί κυρτότητας καθότι ο νευρώνας του τρίτου επιπέδου δέχεται ως είσοδο μια ομάδα κυρτών πολυγώνων όπου ο λογικός συνδυασμός τους δεν είναι απαραίτητα κυρτός.

9.2 Κυρτή Περιοχή

Σχετικά με την κυρτότητα μιας περιοχής μπορούμε να αναφέρουμε τα παρακάτω (Παν. Πατρών, 2022)

- Μία περιοχή $\mathfrak{R} \subseteq R^2$ είναι κυρτή εάν για κάθε ζεύγος σημείων (p,q) με $p,q \in \mathfrak{R}$, το ευθύγραμμο τμήμα s(p,q) περιέχεται στην $\mathfrak{R}$.
- Ένα απλό πολύγωνο είναι κυρτό (convex polygon) εάν το εσωτερικό του είναι κυρτή περιοχή (convex region).
- Ισοδύναμα, ένα απλό πολύγωνο είναι κυρτό εάν καμία γωνία του εσωτερικού του πολυγώνου δεν ξεπερνά τις 180^o.
- Η κορυφή V ενός απλού πολυγώνου ονομάζεται αιχμή (cusp) εάν η εσωτερική γωνία στο V είναι μεγαλύτερη των 180^o.

Κλειστές κυρτές περιοχές *Ανοικτή κυρτή περιοχή* *Μη κυρτή περιοχή*

Εικόνα 9.2

10. Εκπαίδευση με τον Κανόνα Δέλτα

Τα νευρωνικά δίκτυα ξεκινούν από μία κατάσταση όπου δεν έχουν καμία προηγούμενη γνώση. Κατά τη διάρκεια της εκπαίδευσής τους, παρουσιάζονται διάφορα πρότυπα τα οποία ο αισθητήρας πρέπει να μάθει να αναγνωρίζει. Τα παραδείγματα αυτά αποτελούν την ομάδα παραδειγμάτων εκπαίδευσης η οποία έχει δύο τμήματα. Το τμήμα με τα σήματα εισόδου στο νευρωνικό δίκτυο και το τμήμα που περιλαμβάνει τους στόχους εκπαίδευσης, το αποτέλεσμα δηλαδή που θέλουμε να πετύχουμε. Για κάθε ομάδα εισερχομένων σημάτων υπάρχει ένας επιθυμητός στόχος και για όλα τα σήματα υπάρχει αντιστοιχία εισόδου – εξόδου (εκπαίδευση με επιτήρηση). Κάθε φορά που παρουσιάζεται ένα πρότυπο οι νευρώνες υπολογίζουν το δυναμικό τους, το συγκρίνουν με το κατώφλι θ, παράγουν (προβλέπουν) την έξοδο και η οποία συγκρίνεται με την επιθυμητή έξοδο – στόχο που είναι και γνωστή. Η διαφορά μεταξύ εισόδου στόχου και αυτής που προέβλεψε το νευρωνικό δίκτυο αποτελεί το σφάλμα, το οποίο χρησιμοποιείται για να διορθωθούν τα βάρη, εφόσον βέβαια αυτό υπάρχει. Όταν το δίκτυο βρει τιμές στα βάρη του τέτοιες μέσω των οποίων παράγεται η σωστή έξοδος για κάθε πρότυπο, τότε τα βάρη σταθεροποιούνται και δεν αλλάζουν. Ως κύκλο θεωρούμε τη διεργασία από την παρουσίαση των τιμών όλων των προτύπων στην είσοδο μέχρι τον στόχο στην έξοδο και τη διόρθωση των βαρών.

Σύμφωνα με τον κανόνα Δέλτα, ορίζεται η παράμετρος δ που αντιστοιχεί στη διαφορά μεταξύ τιμής στόχου t (target), που είναι η ορθή ή επιθυμητή και τιμής ο (output) που προέβλεψε το νευρωνικό δίκτυο.

$$\text{Δηλαδή } \boldsymbol{\delta = t - o}$$

Αντιλαμβανόμαστε πως αν δ = 0 τότε δεν υφίσταται σφάλμα, συνεπώς ουδεμία αλλαγή στα βάρη πραγματοποιείται.

Αν το δ δεν είναι 0 τότε πραγματοποιείται διόρθωση στα βάρη και υπολογίζουμε την ποσότητα Δ ως εξής (x_i η τιμή του σήματος και η μια σταθερά που δίνει το ρυθμό εκπαίδευσης που συνήθως κυμαίνεται από 0 έως 1, μικρό η δίνει μεγάλο χρόνο εκπαίδευσης καθότι οι αλλαγές θα γίνονται με μικρότερο ρυθμό και μεγάλο η δίνει μικρό χρόνο εκπαίδευσης):

$$\boldsymbol{\Delta_i = \eta * \delta * x_i}$$

Στη συνέχεια διορθώνεται το βάρος του επομένου βήματος n+1 ως εξής:

$$\boldsymbol{W_i(n+1) = w_i(n) + \Delta_i}$$

11. Μέθοδος της Οπισθοδιάδοσης του Λάθους

Η μέθοδος της οπισθοδιάδοσης του λάθους (error backpropagation) είναι ιδιαίτερα δημοφιλής για την εκπαίδευση δικτύων πολλών επιπέδων. Οι νευρώνες μέσα στο ίδιο επίπεδο δε συνδέονται μεταξύ τους, αλλά οι νευρώνες που ανήκουν σε διαφορετικά επίπεδα συνδέονται με συνάψεις. Το σημαντικό με αυτή τη μέθοδο είναι ότι καθίσταται δυνατόν να γίνουν μεταβολές και στα βάρη των ενδιαμέσων επιπέδων στα οποία δεν υπάρχει ο στόχος, κατά συνέπεια δεν είναι δυνατή η χρήση μιας απλής τεχνικής όπως ο κανόνας Δέλτα. Η βασική ιδέα είναι πως το δίκτυο ξεκινά τη μάθηση με τυχαίες τιμές στα βάρη του, αν δώσει λάθος απάντηση τα βάρη διορθώνονται και αυτό επαναλαμβάνεται μέχρι το λάθος να φθάσει ένα πολύ μικρό και ανεκτό επίπεδο.

Η γενική διαδικασία της μεθόδου μπορεί να περιγραφεί ως εξής (Αυγενάκης, 2001):

- Εισάγουμε ένα πρότυπο, από το σύνολο προτύπων που έχουμε για το πρόβλημά μας, στο επίπεδο εισόδου
- Υπολογίζουμε την έξοδο χρησιμοποιώντας μια συνάρτηση ενεργοποίησης (πχ. σιγμοειδή συνάρτηση)
- Προωθούμε την έξοδο του πρώτου επιπέδου στο επόμενο επίπεδο (κρυμμένο) επαναλαμβάνοντας το ίδιο σε όλα τα επίπεδα έως ότου φτάσουμε στο επίπεδο εξόδου
- Στο επίπεδο εξόδου υπολογίζουμε το σφάλμα
- Ανάλογα με το σφάλμα που προκύπτει μεταβάλλουμε τα βάρη ένα και επίπεδο προς επίπεδο, επιστρέφοντας από την έξοδο μέχρι την είσοδο.
- Προχωρούμε στο επόμενο πρότυπο και ακολουθούμε την ίδια διαδικασία για όλα τα πρότυπα.
-

Τα άνωθι βήματα αποτελούν έναν κύκλο εκπαίδευσης (εποχή). Η μέθοδος αυτή επιλύει περίπλοκα προβλήματα, όπως τα γραμμικώς μη διαχωρίσιμα, ξεπερνώντας τις δυνατότητες του μοντέλου του αισθητήρα και ουσιαστικό ρόλο διαδραματίζει η ύπαρξη τουλάχιστον ενός κρυμμένου επιπέδου.

Πιο συγκεκριμένα, ο αλγόριθμος της οπισθοδιάδοσης του λάθους εφαρμόζεται ακολουθώντας τα παρακάτω βήματα (Ψούνης, 2016):

Αρχικοποίηση:

- Αρχικοποιούμε τα διανύσματα:
  - Για κάθε πρότυπο 1 έως K κατασκευάζουμε το διάνυσμα $X_i = [X_{I0}, ..., X_{In}]$ και αρχικοποιούμε την επιθυμητή έξοδο d_i
- Δίνουμε αρίθμηση στους κόμβους
- Αρχικοποιούμε τις τιμές των βαρών
- Εντοπίζουμε τη συνάρτηση ενεργοποίησης για κάθε κόμβο (καθώς και την παράγωγό της)
- Δίνουμε τιμή στην παράμετρο μάθησης η (συνήθως $0 < η < 1$)

Προς τα εμπρός πέρασμα

- Οι νευρώνες εξετάζονται κατά την αύξουσα αρίθμηση $j = 1 … N$
- Θέτουμε ως y_i την έξοδο κάθε νευρώνα
- Για κάθε υπολογιστικό νευρώνα (συνήθως κρυφού επιπέδου και εξόδου):
  - υπολογίζουμε το δυναμικό ως εξής: $v_j = \Sigma^{v}_{i=0} w_{ij} * y_i$
  - υπολογίζουμε την έξοδο από τη συνάρτηση ενεργοποίησης $y_j = \phi(v_j)$
- Συμβολίζουμε με o_j την έξοδο των νευρώνων εξόδου

- Για κάθε νευρώνα εξόδου υπολογίζουμε το σφάλμα $e_j = d_j - o_j$ (επιθυμητή μείον πραγματική τιμή)

Προς τα πίσω πέρασμα

Υπολογισμός Τοπικών Κλίσεων

- Οι νευρώνες εξετάζονται κατά τη φθίνουσα αρίθμηση j = N, N-1, … 1
- Υπολογισμός τοπικής κλίσης δ για κάθε υπολογιστικό νευρώνα
  - Για τους νευρώνες εξόδου: $\delta_j(n) = e_j * \phi'_j(v_j)$
  - Για τους νευρώνες κρυφού επιπέδου: $\delta_j(n) = \phi'_j(v_j) * \Sigma_\kappa [\delta_\kappa(n) * w_{kj}(n)]$, όπου j ο υπό εξέταση νευρώνας και κ η κάθε έξοδος του νευρώνα
  - Για τους νευρώνες εισόδου δε γίνεται υπολογισμός τοπικής κλίσης

Διορθώσεις στα βάρη των ακμών

- Διορθώσεις σε όλα τα βάρη:
  - Υπολογισμός διόρθωσης των βαρών των ακμών: $\Delta w_{ij}(n) = \eta * \delta_j(n) * y_i(n)$, w_{ij} το βάρος που συνδέει τον I νευρώνα με τον j
  - Υπολογισμός των βαρών $w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n)$, n το τρέχον βήμα

12. Συνελικτικά Νευρωνικά Δίκτυα – Convolutional Neural Networks CNN

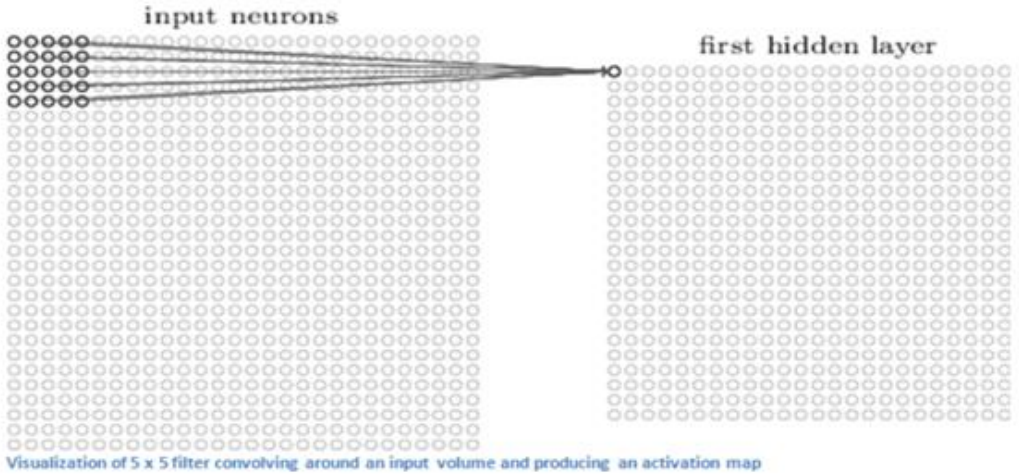
Τα συνελικτικά νευρωνικά δίκτυα – convolutional neural networks CNN κάνουν χρήση της μαθηματικής πράξης της συνέλιξης στην οποία οφείλουν και το όνομά τους και χρησιμοποιούν δεδομένα τα οποία έχουν μια γνωστή τοπολογία πλέγματος (Goodfellow et al., 2016). Η συνέλιξη είναι μια μαθηματική πράξη δύο συναρτήσεων f και g, συμβολίζεται ως f * g, η οποία παράγει μια τρίτη συνάρτηση εκφράζουσα το πως το σχήμα της μιας μεταβάλλεται από την άλλη και ορίζεται ως το ολοκλήρωμα του γινομένου των δύο συναρτήσεων μετά την αποτύπωση της μιας στον άξονα των ψ και τη μετατόπισή της (Καραγιάννης & Μαραγκός, 2011). Τα CNN εφαρμόζονται ευρέως στο πεδίο της όρασης του υπολογιστή και της αναγνώρισης εικόνων έχοντας ως στόχο να μας δώσουν μια πιθανότητα ότι μια εικόνα ανήκει σε μια κλάση (πχ. 0.80 πιθανότητα ότι η εικόνα έχει μια γάτα, 0.15 ένα σκύλο κοκ). Αυτό επιτυγχάνεται με το να επικεντρώνεται το μοντέλο σε χαρακτηριστικά χαμηλού επιπέδου όπως περιγράμματα ή καμπύλες που υπάρχουν στην εικόνα και εν συνεχεία να τα συνθέτει μέσα από μια σειρά επιπέδων.

Ένα τυπικό νευρωνικό δίκτυο CNN εμπεριέχει:

- Επίπεδα συνέλιξης.
- Κάθε επίπεδο συνέλιξης ακολουθείται από πράξη ενεργοποίησης.
- Επίπεδο pooling.
- Πλήρως συνδεδεμένο επίπεδο (fully connected layer).

12.1 Συνελικτικό Επίπεδο (Convolutional Layer)

Η είσοδος είναι ένας τανυστής (tensor) με το σχήμα: (αριθμός εισόδων) x (πίξελ ύψους εισόδου) x (πίξελ πλάτους εισόδου) x (χρωματικά κανάλια εισόδου). Σε αυτό το επίπεδο υπάρχει ένα φίλτρο (filter) ή νευρώνας (neuron) ή πυρήνας (kernel) ο οποίος σαρώνει κάποια από τα πίξελ της εικόνας εισόδου σύμφωνα πάντα με το μέγεθός του, εικόνα 12.1.



Εικόνα 12.1

Πρακτικά ο πυρήνας μετακινείται πάνω από την εικόνα εισόδου, πολλαπλασιάζοντας τις τιμές που έχει με αυτές των πίξελ της εικόνας εισόδου.

Τα αποτελέσματα των πολλαπλασιασμών συγκεντρώνονται και αποτυπώνονται σε έναν χάρτη χαρακτηριστικών (feature map) γνωστό και ως χάρτη ενεργοποίησης (activation map) σχήματος (αριθμός εισόδων) x (ύψος χάρτη χαρακτηριστικών) x (πλάτος χάρτη χαρακτηριστικών) x (χρωματικά κανάλια χάρτη χαρακτηριστικών) κάνοντας έτσι την αρχική εικόνα πιο αφηρημένη.

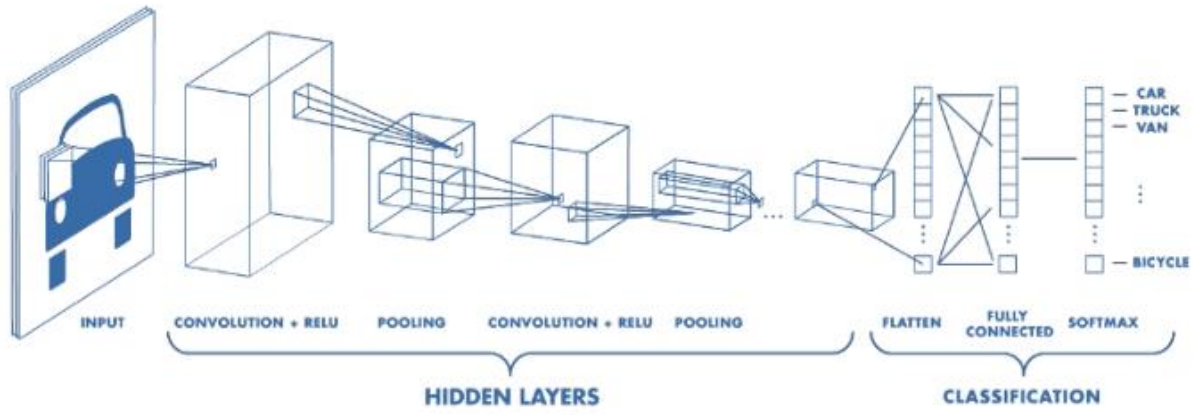
12.2 Επίπεδο Pooling

Τα CNN ενδέχεται να εμπεριέχουν μαζί με τα συνελικτικά επίπεδα και επίπεδα polling τα οποία μπορεί να είναι τοπικά ή καθολικά. Τα επίπεδα pooling μειώνουν τις διαστάσεις των δεδομένων συνδυάζοντας τις εξόδους συστάδων νευρώνων ενός επιπέδου σε ένα νευρώνα στο επόμενο επίπεδο. Τα τοπικά επίπεδα pooling συνδυάζουν μικρές συστάδες ενώ τα καθολικά ενεργούν σε όλους τους νευρώνες του χάρτη χαρακτηριστικών (Dan, 2011). Δύο συνηθισμένοι τύποι τέτοιων επιπέδων είναι το αυτοί του μέγιστου (max pooling) και του μέσου (average pooling). Το max pooling χρησιμοποιεί τη μέγιστη τιμή κάθε τοπικής συστάδας νευρώνων στο χάρτη χαρακτηριστικών και το average pooling τη μέση τιμή.

12.3 Πλήρως Συνδεδεμένο Επίπεδο – Fully Connected Layer

Τα πλήρως συνδεδεμένα επίπεδα συνδέουν κάθε νευρώνας ενός επιπέδου με κάθε νευρώνα του επόμενου επιπέδου κατά τον ίδιο παραδοσιακό τρόπο που λαμβάνει χώρα σε ένα τυπικό νευρωνικό δίκτυο πολυεπίπεδου perceptron (multilayer perceptron MLP).

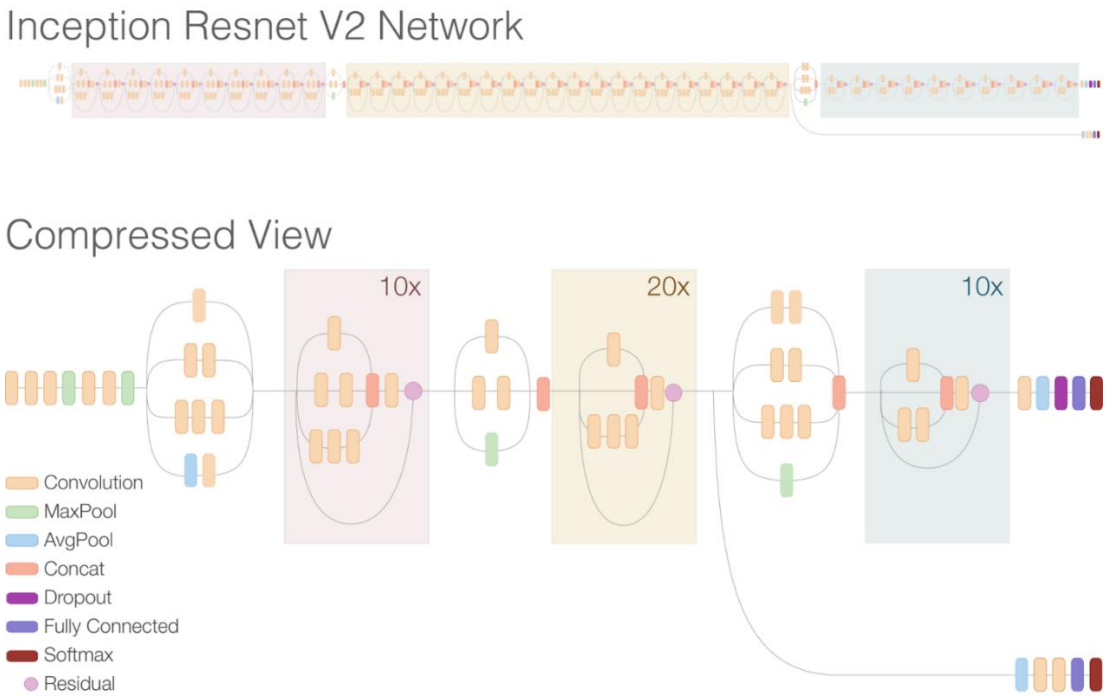
Μια αρχιτεκτονική ενός τυπικού CNN φαίνεται στην παρακάτω εικόνα:



Εικόνα 12.2, (Ejaz et al., 2019)

12.4 Inception ResNet v2

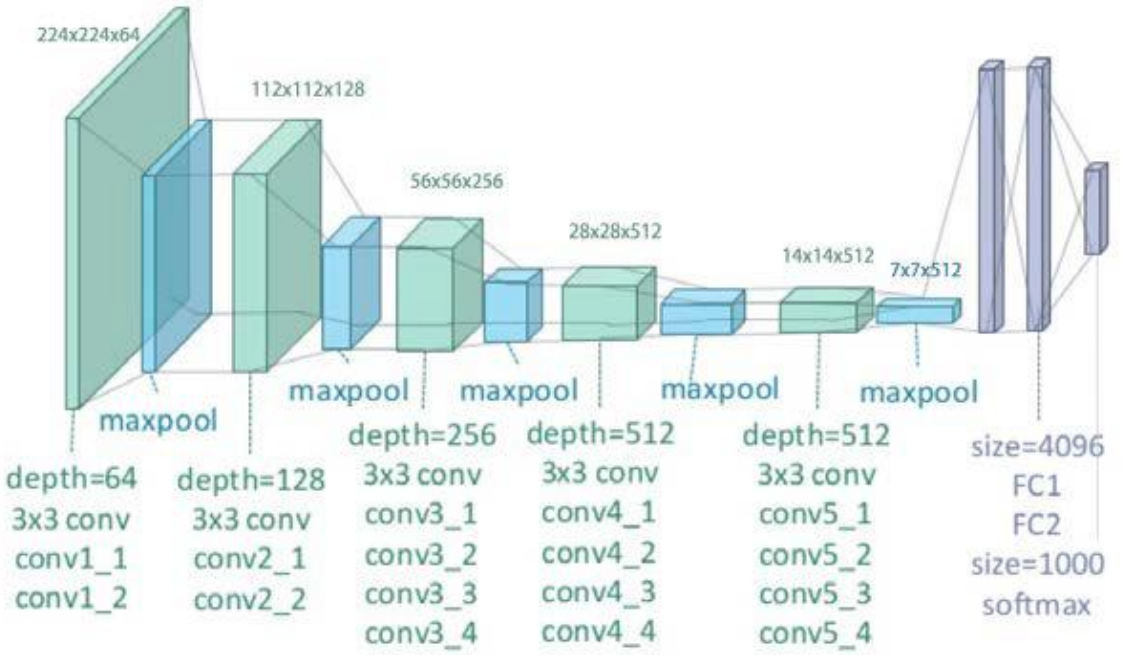
Το Inception ResNet v2 είναι ένα συνελικτικό νευρωνικό δίκτυο εκπαιδευμένο σε περισσότερες από 1.000.000 εικόνες από τη βάση δεδομένων ImageNet. Το δίκτυο έχει 164 επίπεδα και μπορεί να ταξινομεί εικόνες σε 1.000 κατηγορίες αντικειμένων όπως πληκτρολόγιο, ποντίκι, μολύβι και διάφορα ζώα (www.mathworks.com). Το δίκτυο έχει εκπαιδευτεί σε πλούσιες απεικονίσεις χαρακτηριστικών σε ένα μεγάλο εύρος εικόνων και λαμβάνει ως είσοδο εικόνες διαστάσεων 299 επί 299.



Εικόνα 12.3

12.5 Visual Geometry Group VGG 19

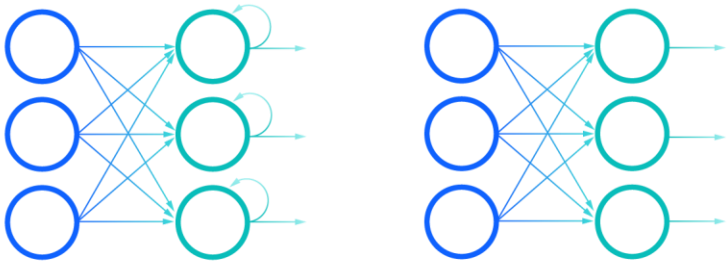
Το VGG 19 είναι ένα CNN νευρωνικό δίκτυο με 19 επίπεδα βάθος, εκπαιδευμένο σε περισσότερες από 1.000.000 εικόνες από τη βάση δεδομένων ImageNet. Το προ-εκπαιδευμένο αυτό δίκτυο μπορεί να ταξινομεί εικόνες σε 1.000 κατηγορίες αντικειμένων, όπως πληκτρολόγιο, ποντίκι, μολύβι και πολλές άλλες (www.mathworks.com). Το αποτέλεσμα είναι το δίκτυο να έχει μάθει μια πλούσια αναπαράσταση χαρακτηριστικών για ένα μεγάλο εύρος εικόνων και λαμβάνει ως είσοδο εικόνες μεγέθους 224 επί 224 πίξελ.



Εικόνα 12.4. VGG 19 (Zheng et al., 2018)

13. Επαναλαμβανόμενα Νευρωνικά Δίκτυα – Recurrent Neural Networks RNN

Τα επαναλαμβανόμενα νευρωνικά δίκτυα – recurrent neural networks RNN, χρησιμοποιούν διαδοχικά δεδομένα ή δεδομένα χρονοσειρών, κατά συνέπεια είναι ιδιαιτέρως χρήσιμα στην επίλυση προβλημάτων που σχετίζονται με τη γλωσσική μετάφραση, την επεξεργασία φυσικής γλώσσας, την αναγνώριση ομιλίας και τον υποτιτλισμό εικόνων. Χρησιμοποιούν δεδομένα για την εκπαίδευσή τους και ξεχωρίζουν για τη μνήμη τους (Dupond, 2019), καθώς λαμβάνουν την πληροφορία από τα τις προηγούμενες εισόδους για να επηρεάσουν την τρέχουσα είσοδο και έξοδο κάτι που δε συμβαίνει στα παραδοσιακά νευρωνικά δίκτυα, όπου οι είσοδοι και οι έξοδοι είναι ανεξάρτητοι, εικόνα 13.1.



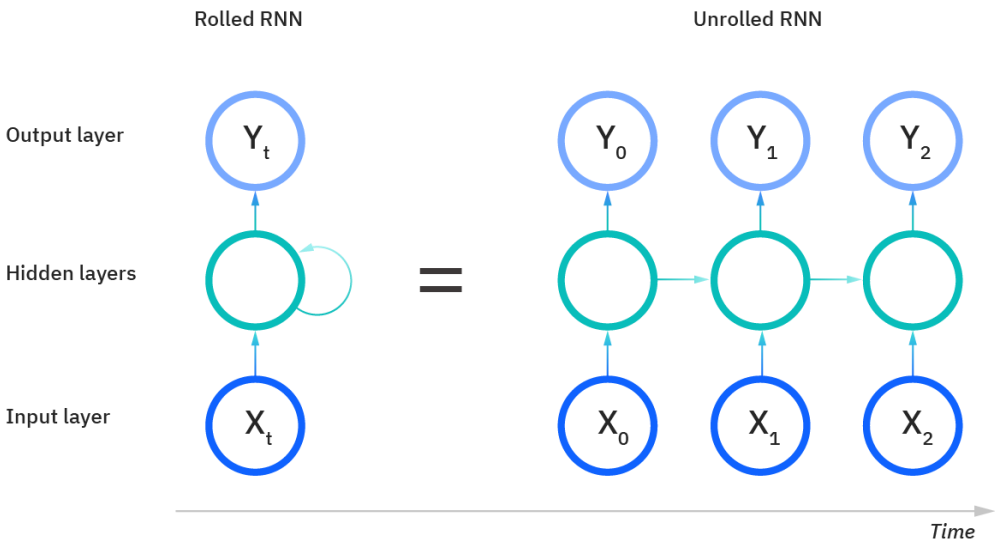
Εικόνα 13.1. Αριστερά: RNN, Δεξιά: Feedforward NN

Μπορούμε να λάβουμε ως παράδειγμα ένα ιδίωμα όπως τη φράση «μασάς τα λόγια σου» η οποία χρησιμοποιείται συχνά όταν κάποιο άτομο δε μιλάει καθαρά ή με σωστή άρθρωση ή, στη μεταφορική της έννοια, δεν περιγράφει την ακριβή κατάσταση της πραγματικότητας. Για να βγει νόημα από αυτή τη φράση πρέπει οι λέξεις της να διατυπωθούν με αυτήν ακριβώς τη σειρά και τα RNN να τροφοδοτηθούν με αυτήν ακριβώς τη σειρά λέξεων ώστε να μπορέσουν να την επεξεργαστούν ορθά.

Στην παρακάτω εικόνα βλέπουμε τη μορφή «rolled» του RNN η οποία αντιπροσωπεύει ολόκληρο το νευρωνικό δίκτυο ή διαφορετικά ολόκληρη την προβλεφθείσα φράση, όπως την «μασάς τα λόγια σου». Η μορφή «unrolled» αντιπροσωπεύει τα ξεχωριστά επίπεδα ή βήματα στο χρόνο του νευρωνικού δικτύου. Κάθε επίπεδο ταυτίζεται με μια λέξη της φράσης, όπως για παράδειγμα τα «λόγια». Κάθε προηγούμενη είσοδος, όπως «μασάς», «τα» θα παρουσιαστεί ως μια κρυφή κατάσταση στο τρίτο βήμα στο χρόνο για να προβλέψει την έξοδο της αλληλουχίας που θα είναι η λέξη «λόγια».

Ένα ακόμα χαρακτηριστικό των RNN είναι ότι μοιράζονται παραμέτρους σε κάθε επίπεδο του δικτύου. Ενώ τα feedforward networks έχουν διαφορετικά βάση σε κάθε κόμβο, τα RNN μοιράζονται τις ίδιες παραμέτρους βαρών σε κάθε επίπεδο του δικτύου. Αυτά τα βάρη προσαρμόζονται κατά τη διαδικασία της οπισθοδιάδοσης και του gradient descent ώστε να διευκολύνουν την ενισχυμένη μάθηση.

Τα RNN αξιοποιούν το αλγόριθμο της οπισθοδιάδοσης μέσα στο χρόνο (backpropagation through time – BPTT) για να καθορίσουν τις κλίσεις. Αυτός ο αλγόριθμος διαφέρει σε σχέση με την παραδοσιακή προσέγγιση της οπισθοδιάδοσης καθώς αθροίζει τα λάθη σε κάθε βήμα στο χρόνο ενώ τα feedforward δίκτυα δεν προβαίνουν σε αθροίσεις καθώς δε μοιράζονται παραμέτρους σε κάθε επίπεδο.



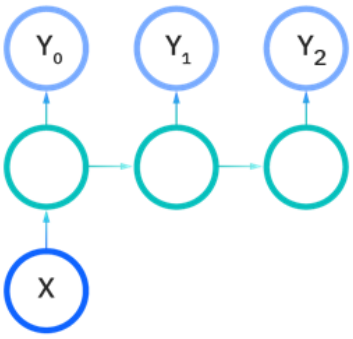
Εικόνα 13.2

Μέσα από αυτή τη διαδικασία τα RNN εμφανίζουν δύο προβλήματα γνωστά ως exploding gradients και vanishing gradients. Αυτά τα προβλήματα ορίζονται από το μέγεθος της κλίσης της συνάρτησης απώλειας. Όταν η κλίση είναι πολύ μικρή, τότε συνεχίζει να γίνεται μικρότερη, ανανεώνοντας τις παραμέτρους των βαρών έως ότου γίνουν ασήμαντα οπότε και ο αλγόριθμος σταματά να μαθαίνει. Το φαινόμενο του exploding gradient συμβαίνει όταν η κλίση είναι πολύ μεγάλη δημιουργώντας κατ' αυτόν τον τρόπο ένα ασταθές μοντέλο. Σε αυτή την περίπτωση τα βάρη μεγαλώνουν υπερβολικά έως ότου τελικά παρουσιαστούν με την τιμή NaN. Μια λύση σε αυτό το πρόβλημα είναι η μείωση του αριθμού των κρυφών επιπέδων, απαλείφοντας έτσι ένα μέρος της πολυπλοκότητας του RNN μοντέλου.

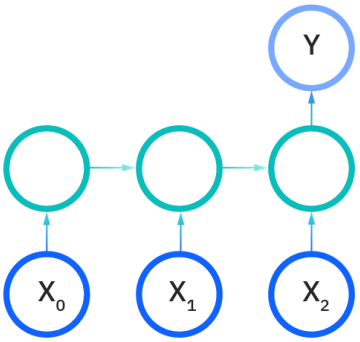
13.1 Τύποι RNN

Τα feedforward δίκτυα έχουν μία είσοδο σε μία έξοδο γεγονός που δεν ισχύει πάντα στα RNN. Απεναντίας, οι είσοδοι και έξοδοι στα RNN ποικίλουν στο μήκος και διακρίνονται οι παρακάτω περιπτώσεις:

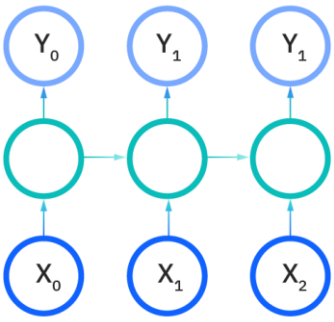
Εικόνα 13.3α. Ένα προς Ένα



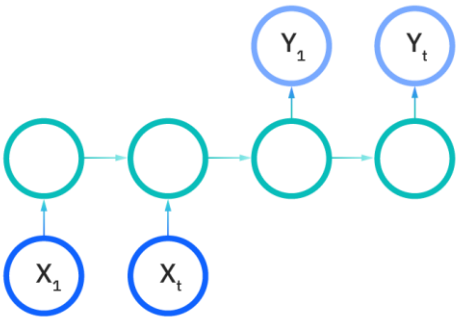
Εικόνα 13.3β. Ένα προς Πολλά



Εικόνα 13.4.Πολλά προς Ένα



Εικόνα 13.5α.Πολλά προς Πολλά



Εικόνα 13.5β. Πολλά προς Πολλά

Ως συναρτήσεις ενεργοποίησης συχνά χρησιμοποιούνται οι σιγμοειδής, η tanh και η relu.

13.2 Αρχιτεκτονικές RNN

Μια γνωστή αρχιτεκτονική RNN είναι τα **Bidirectional Recurrent Neural Networks (BRNN).** Ενώ τα Unidirectional RNN αντλούν πληροφορία μόνο από τις προηγούμενες εισόδους για να κάνουν προβλέψεις σχετικά με την τρέχουσα κατάσταση, τα Bidirectional RNN τραβούν μελλοντικά δεδομένα για να βελτιώσουν την ακρίβειά τους. Για παράδειγμα, σχετικά με την πρόβλεψη της φράσης «μασάς τα λόγια σου» το μοντέλο θα μπορούσε να προβλέψει ευκολότερα ότι η τρίτη λέξη είναι το «λόγια» αν ήξερε ότι οι δύο πρώτες είναι το «μασάς» και το «τα».

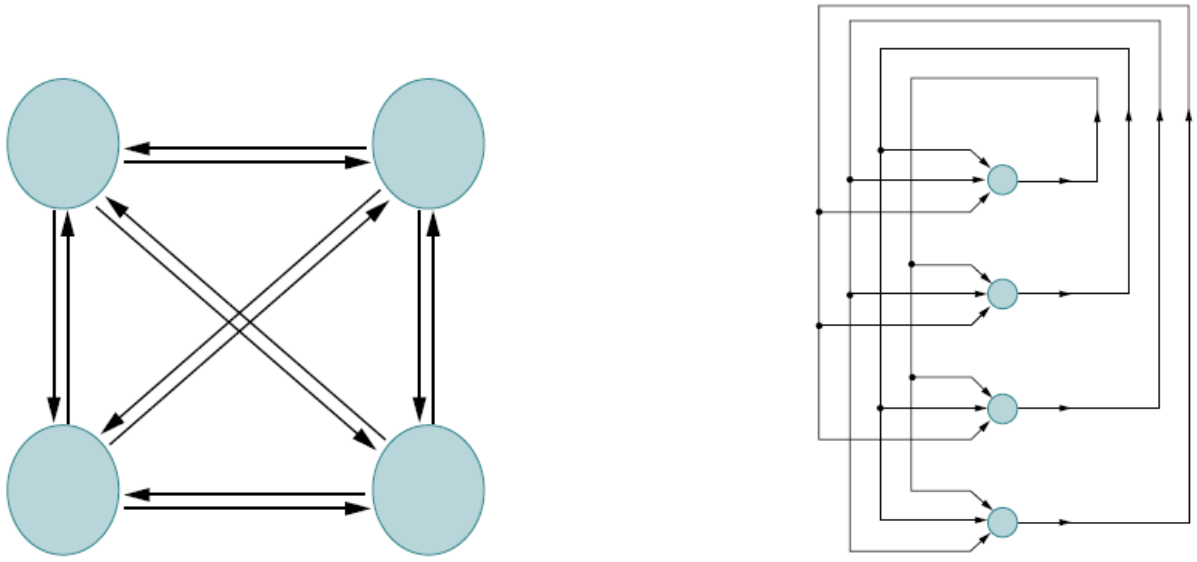
Long Short-Term Memory – LSTM. Αυτή η αρχιτεκτονική αποτελεί μια λύση στο πρόβλημα του vanishing gradient descent (Hochreiter – Schmidhuber, 1997). Με το πρόβλημα των long term dependencies (Sak et al., 2014), δηλαδή όταν η προηγούμενη κατάσταση που επηρεάζει την τρέχουσα πρόβλεψη, δεν είναι στο πρόσφατο παρελθόν, το μοντέλο RNN ενδέχεται να μην μπορεί να προβλέψει με ακρίβεια την τρέχουσα κατάσταση. Υποθέτουμε για παράδειγμα ότι θέλουμε να προβλέψουμε τις λέξεις με τα πλάγια γράμματα στη φράση «Ο Ορέστης έχει δυσανεξία στη λακτόζη. Δεν μπορεί να καταναλώσει *γαλακτοκομικά προϊόντα*». Η έννοια που εμπεριέχεται στις λέξεις «δυσανεξία στη λακτόζη» είναι ιδιαιτέρως χρήσιμη ώστε να αναμένουμε τι είδους προϊόντα δεν μπορεί κάποιο άτομο να καταναλώσει. Ωστόσο, αν η φράση «δυσανεξία στη λακτόζη» ήταν πολύ νωρίτερα από τη φράση «δεν μπορεί να καταναλώσει γαλακτοκομικά προϊόντα», θα ήταν από δύσκολο έως αδύνατο για ένα μοντέλο RNN να προβλέψει τις λέξεις «γαλακτοκομικά προϊόντα» και να κάνει τη σύνδεση μεταξύ των δύο φράσεων. Για να αντιμετωπιστεί αυτή η δυσκολία, τα LSTM έχουν κελιά στα κρυφά επίπεδα του νευρωνικού δικτύου τα οποία έχουν 3 πύλες, μία εισόδου, μία εξόδου και μία λήθης. Αυτές οι πύλες ελέγχουν τη ροή της πληροφορίας που είναι χρήσιμη για την πρόβλεψη της εξόδου του δικτύου. Για παράδειγμα, αν η αντωνυμία γένους «αυτός» επαναλήφθηκε πολλές φορές σε προηγούμενες προτάσεις, μπορεί μη συμπεριληφθεί από τη συγκεκριμένη κατάσταση κελιού.

Gated Recurrent Units – GRU. Αυτή η εκδοχή των RNN μοιάζει με τα LSTM καθώς προσπαθεί να δώσει λύση στο πρόβλημα της short term memory. Αντί της χρήσης μιας κατάστασης κελιού για να ελεγχθεί η πληροφορία, χρησιμοποιεί κρυφές καταστάσεις και αντί για τρεις πύλες, έχει δύο, την πύλη επαναφοράς (reset gate) και την πύλη επικαιροποίησης (update). Κατ’ αναλογία με τις πύλες στα LSTM, οι πύλες επαναφοράς και επικαιροποίησης ελέγχουν πόσες και ποιες πληροφορίες θα συγκρατηθούν.

14. Νευρωνικά Δίκτυα Hopfield

Τα δίκτυα τύπου Hopfield ανήκουν στην κατηγορία των RNN (Miljanovic, 2012), αποτελούνται από μόνο ένα επίπεδο με πολλούς νευρώνες. Δεν υπάρχει χωριστό επίπεδο εισόδου ή εξόδου αλλά κάθε νευρώνας δέχεται σήματα από το περιβάλλον και έχει εξόδους προς αυτό (Αργυράκης, 2001). Κάθε νευρώνας δίνει στην έξοδό του σήμα s_i προς όλες τις άλλες μονάδες j. Είναι δυαδικά συστήματα ως προς το ότι κάθε μονάδα χαρακτηρίζεται από μια δυαδική κατάσταση, δηλαδή μπορεί να έχει μία από τις δύο δυνατές τιμές (συνήθως οι τιμές είναι 0 ή 1). Οι μονάδες του δικτύου έχουν πλήρη συνδεσμολογία δηλαδή κάθε μονάδα συνδέεται με κάθε άλλη μονάδα στο σύστημα, υπάρχουν δηλαδή n(n-1) συνδέσεις, με n να είναι ο αριθμός των νευρώνων (κάθε μονάδα συνδέεται με κάθε άλλη μονάδα αλλά όχι με τον εαυτό της). Στη γενική περίπτωση οι συνδέσεις έχουν συγκεκριμένη κατεύθυνση έτσι ώστε σε κάθε ζευγάρι μονάδων που συνδέονται υπάρχει σύνδεση και προς τις δύο κατευθύνσεις, δηλαδή μεταξύ των μονάδων i και j υπάρχει η σύνδεση w_{ij} και η σύνδεση w_{ji}. Γενικά στα δίκτυα Hopfield θα έχουμε πάντοτε ότι $w_{ij} = w_{ji,}$δηλαδή τα βάρη είναι ίσα και προς τις δύο κατευθύνσεις και έτσι εξασφαλίζεται ότι το δίκτυο συγκλίνει και καταλήγει σε μια σταθερή κατάσταση. Μάλιστα οι Cohen – Grossberg απέδειξαν πως τα δίκτυα αυτά είναι ευσταθή αν η μήτρα των βαρών W_{ij} είναι συμμετρική με 0 στην κύρια διαγώνιο. Έτσι σε ένα δίκτυο με 4 νευρώνες η μήτρα των βαρών διαμορφώνεται ως εξής:

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} \\ w_{21} & 0 & w_{23} & w_{24} \\ w_{31} & w_{32} & 0 & w_{34} \\ w_{41} & w_{42} & w_{43} & 0 \end{bmatrix}$$



Εικόνα 14.1 (Αργυράκης 2001). Δύο αναπαραστάσεις του ιδίου δικτύου Hopfield με 4 νευρώνες. Παρατηρούμε πως γίνεται χρήση του μηχανισμού ανάδρασης (feedback mechanism) καθότι η έξοδος από κάθε νευρώνα ταυτόχρονα γίνεται είσοδος για τους άλλους νευρώνες.

Σε κάθε νευρώνα του δικτύου είναι δυνατόν να υπάρχει κάποιο κατώφλι. Ο υπολογισμός του δυναμικού γίνεται υπολογίζοντας το άθροισμα των βαρών επί τις τιμές εισόδου, δηλαδή

$$S_j = w_{ij} * s_i \ με \ i \neq j$$

Συνηθισμένες συναρτήσεις ενεργοποίησης που χρησιμοποιούνται είναι η συνάρτηση προσήμου όπου το αποτέλεσμα είναι 1 αν το x είναι μεγαλύτερο ή ίσο με το 0 και – 1 αν το x είναι μικρότερο του 0 ή μια παραλλαγή της σύμφωνα με την οποία το αποτέλεσμα είναι 1 αν το x είναι μεγαλύτερο από το 0, -1 αν το x είναι μικρότερο από το 0 και ίδιο με την έξοδο y_k του νευρώνα στο προηγούμενο βήμα αν x είναι ίσο με το 0.

$$f(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases} \qquad f(x) = \begin{cases} 1, x > 0 \\ -1, x < 0 \\ y_k, x = 0 \end{cases}$$

Τα δίκτυα Hopfield χρησιμοποιούνται για να αποθηκεύσουμε διανύσματα ίσης διάστασης. Έτσι αν θέλουμε να αποθηκεύσουμε τα N-διάστατα διανύσματα X_1, X_2, …., X_K θα χρειαστούμε ένα δίκτυο Hopfield N αισθητήρων. Η αποθήκευση γίνεται στα βάρη των ακμών με βάση τον παρακάτω τύπο:

$$W = X_1 * X_1^T + X_2 * X_2^T + X_K * X_K^T - K * I$$

Όπου K είναι ο αριθμός των διανυσμάτων και I ο μοναδιαίος πίνακας.

Για παράδειγμα αν θέλουμε να αποθηκεύσουμε τα διανύσματα $X_1 = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$ και $X_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ σε ένα τεχνητό νευρωνικό δίκτυο Hopfield έχουμε τον παρακάτω πίνακα βαρών:

$$W = X_1 * X_1^T + X_2 * X_2^T - 2I =$$

$$\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} [-1 \ \ 1 \ \ 1] + \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} [-1 \ \ 1 \ \ -1] \ -2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \ -2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Για να εξάγουμε διανύσματα από τον πίνακα βαρών γίνεται η πράξη

$$sign(W * X_i - \vartheta)$$

το αποτέλεσμα της οποίας θα πρέπει να είναι ίσο με X_i (sign είναι η συνάρτηση προσήμου)

Για τα διανύσματα του προηγούμενου παραδείγματος $X_1 = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$ και $X_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ με κατώφλι $\theta = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ έχουμε (λαμβάνουμε υπόψιν την παραλλαγή της συνάρτησης προσήμου όπου για x = 0 το αποτέλεσμα είναι το y_k που είχαμε στην είσοδο):

$$X_1 : sign(W * X_1 - \theta) = sign\left(\begin{bmatrix} 0 & 2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = sign\left(\begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = X_1$$

X_2 : sign(W * X_2 – θ) = sign$\left(\begin{bmatrix} 0 & 2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$ = sign$\left(\begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix}\right)$ = $\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ = X_2

Παρατηρούμε πως μετά την εκτέλεση των πράξεων όντως λαμβάνουμε τα διανύσματα X_1 και X_2

Ο μέγιστος αριθμός βασικών μνημών που μπορούμε να αποθηκεύσουμε χωρίς σφάλμα στην ανάκτηση είναι ο:

$$\boldsymbol{M_{max}} = \frac{\boldsymbol{N}}{\boldsymbol{4 * ln(N)}}$$

όπου N το πλήθος των νευρώνων άρα και η διάσταση του διανύσματος.

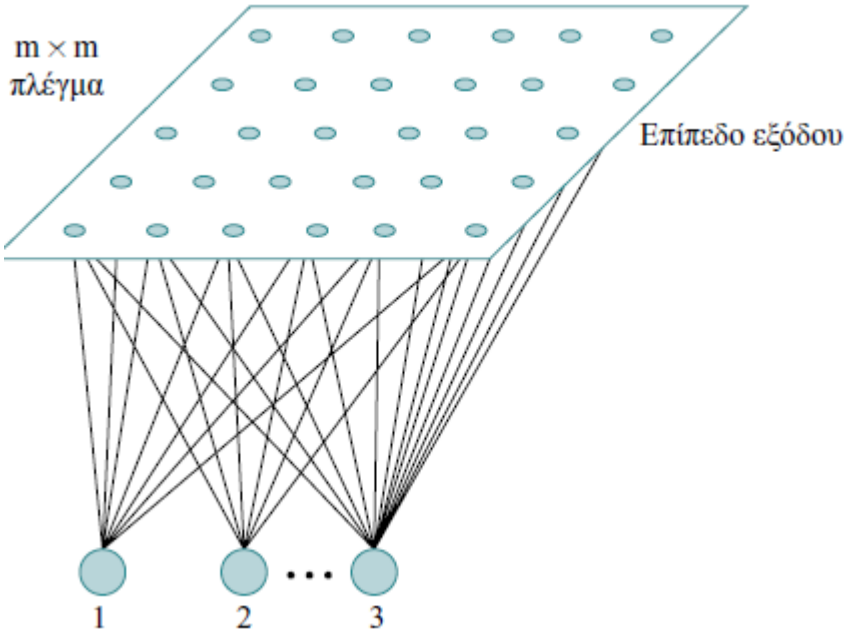
Ο μέγιστος αριθμός βασικών μνημών που μπορούμε να αποθηκεύσουμε ώστε περισσότερες από τις μισές βασικές μνήμες να ανακαλούνται χωρίς σφάλμα είναι

$$\boldsymbol{M_{max}} = \frac{\boldsymbol{N}}{\boldsymbol{2 * ln(N)}}$$

15. Νευρωνικά Δίκτυα Kohonen

15.1 Λειτουργία Δικτύων Kohonen

Τα δίκτυα Kohonen ανήκουν στην κατηγορία των τεχνητών νευρωνικών δικτύων χωρίς εκπαίδευση, δηλαδή εισάγονται τα πρότυπα στο δίκτυο χωρίς να δίδονται οι επιθυμητοί έξοδοι και αποτελούνται από δύο επίπεδα. Το πρώτο επίπεδο είναι συνήθως αυτό της εισόδου και το δεύτερο έχει το χαρακτηριστικό ότι είναι οργανωμένο σε μορφή πλέγματος οποιασδήποτε διάστασης (Αργυράκης, 2001). Τα δύο αυτά επίπεδα έχουν πλήρη συνδεσμολογία κατά συνέπεια αν έχουμε ν μονάδες εισόδου και ένα πλέγμα μ * μ μονάδων τότε έχουμε ένα σύνολο από ν * μ * μ συνδέσεις.



Εικόνα 15.1. Δίκτυο Kohonen (Αργυράκης, 2001)

Η λογική των δικτύων Kohonen είναι πως προσπαθούν να ομαδοποιήσουν τα πρότυπα που έχουν κάποια ομοιότητα (clustering).

Κάθε νευρώνας k (k = 1, … ,M) υπολογίζει την ευκλείδεια απόσταση d_k, του διανύσματος εισόδου x_i (το διάνυσμα εισόδου έχει συνιστώσες από 1 έως n) από το διάνυσμα των βαρών του (κάθε βάρος του νευρώνα k σχετίζεται με μια είσοδο i) [w_{1k}, w_{2k}, …, w_{nk}] σύμφωνα με τον τύπο:

$$\boldsymbol{d_k} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{W_{ik}} - \boldsymbol{X_i})^2}$$

Ο νευρώνας με τη μικρότερη τιμή είναι ο νικητής νευρώνας παράγοντας 1 στην έξοδο και οι υπόλοιποι παράγουν έξοδο 0.

Η διόρθωση βαρών γίνεται μόνο στο νικητή νευρώνα ως εξής:

Για κάθε βάρος w_{ik} του νευρώνα k (i: από τη συνιστώσα 1 έως n της εισόδου) υπολογίζεται η ποσότητα

$$\Delta W_{ik} = a(x_i - w_{jk})$$

όπου a είναι ο ρυθμός εκπαίδευσης.

Εν συνεχεία ανανεώνουμε το βάρος w_{ik} θέτοντας:

$$w_{ik} = w_{ik} + \Delta W_{ik}$$

Η διαδικασία συνεχίζεται έως ότου εκτελεστεί ένα πλήθος κύκλων εκπαίδευσης.

15.2 Παράδειγμα Νευρωνικού Δικτύου Kohonen

Υποθέτουμε πως έχουμε μαθητές στους οποίους έχουμε μετρήσει 2 χαρακτηριστικά.

- $1^{ον}$ τη βαθμολογία στην εργασία τετραμήνου
- $2^{ον}$ τη βαθμολογία στο τελικό διαγώνισμα τετραμήνου

Θέλουμε να κατατάξουμε αυτούς τους μαθητές σε δύο ομάδες με σκοπό να δούμε τις ανάγκες τους και πως μπορούμε να τους υποστηρίξουμε κατάλληλα. Έτσι, θα δημιουργήσουμε μία ομάδα μαθητών όπου τα παιδιά θα έχουν καλές επιδόσεις στα 2 άνωθι χαρακτηριστικά και θα χρειάζονται λιγότερη βοήθεια και μία ομάδα μαθητών με πιο αδύναμες επιδόσεις και που θα χρειάζονται περισσότερη υποστήριξη.

Μπορούμε λοιπόν να σχεδιάσουμε ένα νευρωνικό δίκτυο Kohonen, όπου στο επίπεδο εισόδου θα υπάρχουν 2 νευρώνες, ένας για το κάθε χαρακτηριστικό που μετράμε (βαθμολογία σε εργασία και βαθμολογία σε τελικό διαγώνισμα) και 2 νευρώνες εξόδου (ανταγωνιστικούς νευρώνες ή νευρώνες Kohonen) μιας και έχουμε αποφασίσει να χωρίσουμε τους μαθητές σε 2 κλάσεις (καλές και πιο αδύναμες επιδόσεις).

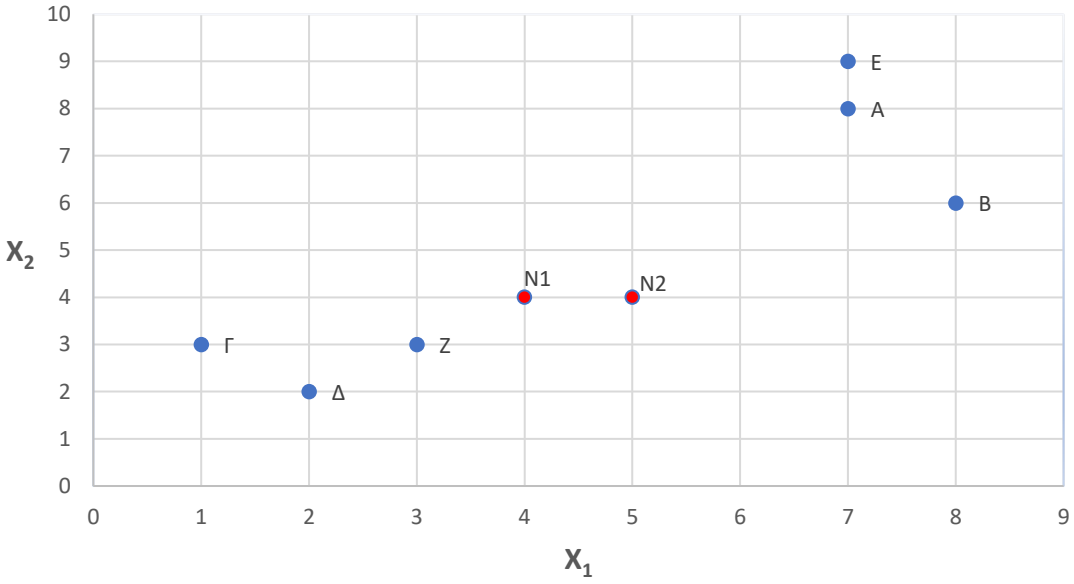
Δεδομένου ότι μετράμε 2 χαρακτηριστικά μπορούμε να κάνουμε μια γραφική απεικόνιση στο επίπεδο όπου X_1 θα είναι η βαθμολογία στην εργασία τετραμήνου και X_2 η βαθμολογία στο τελικό διαγώνισμα τετραμήνου.

Αρχικά τα βάρη των νευρώνων εξόδου λαμβάνουν τυχαίες τιμές (έστω ο $1^{ος}$ νευρώνας N1 έχει τις συντεταγμένες 4,4 και ο $2^{ος}$ νευρώνας N2 τις συντεταγμένες 4, 5. Οι νευρώνες αυτοί απεικονίζονται με τις– κόκκινες κουκκίδες στο επίπεδο της εικόνας 15.2) και έστω ο ρυθμός εκπαίδευσης a = 0.5.

Υποθέτουμε επίσης ότι τα δεδομένα X_1, X_2 για κάθε μαθητή είναι τα εξής (οι γαλάζιες κουκίδες στο επίπεδο της εικόνας 15.2):

Μαθητής	X_1	X_2
Α	7	8
Β	8	6
Γ	1	3
Δ	2	2
Ε	7	9
Ζ	3	3

Πίνακας 9



Εικόνα 15.2

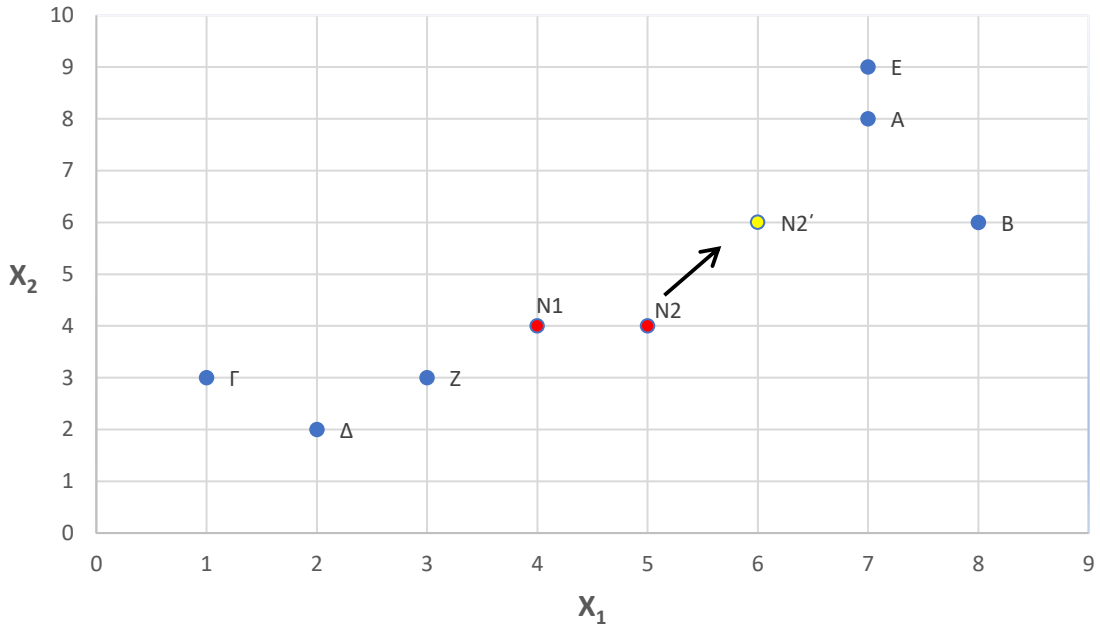
Υποθέτουμε πως εισάγεται πρώτα το πρότυπο Α, οπότε υπολογίζουμε την ευκλείδεια απόσταση του Α από κάθε νευρώνα.

Απόσταση Α από 1ο νευρώνα N1: $d_1 = \sqrt{(w_{11} - x_\perp)^2 + (w_{21} - x_2)^2}$ = $\sqrt{(4-7)^2 + (4-8)^2}$ = 5
Απόσταση Α από 2ο νευρώνα N2: $d_2 = \sqrt{(w_{12} - x_\perp)^2 + (w_{22} - x_2)^2}$ = $\sqrt{(5-7)^2 + (4-8)^2}$ = 4.47

Ο νευρώνας με τη μικρότερη απόσταση από το πρότυπο Α είναι ο N2, άρα είναι ο νικητής και ανανεώνουμε τα βάρη του ως εξής:

$\Delta W_{12} = a(x_1 - w_{12}) = 0.5(7 - 5) = 1$, $w_{12} = w_{12} + \Delta W_{12} = 5 + 1 = 6$
$\Delta W_{22} = a(x_2 - w_{22}) = 0.5(8 - 4) = 2$, $w_{22} = w_{22} + \Delta W_{22} = 4 + 2 = 6$

Άρα ο νευρώνας 2 πλέον μετακινείται στις συντεταγμένες 6, 6 (N2’ κίτρινη κουκίδα στην εικόνα 14.4) για να προσεγγίσει το πρότυπο Α.



Εικόνα 15.3

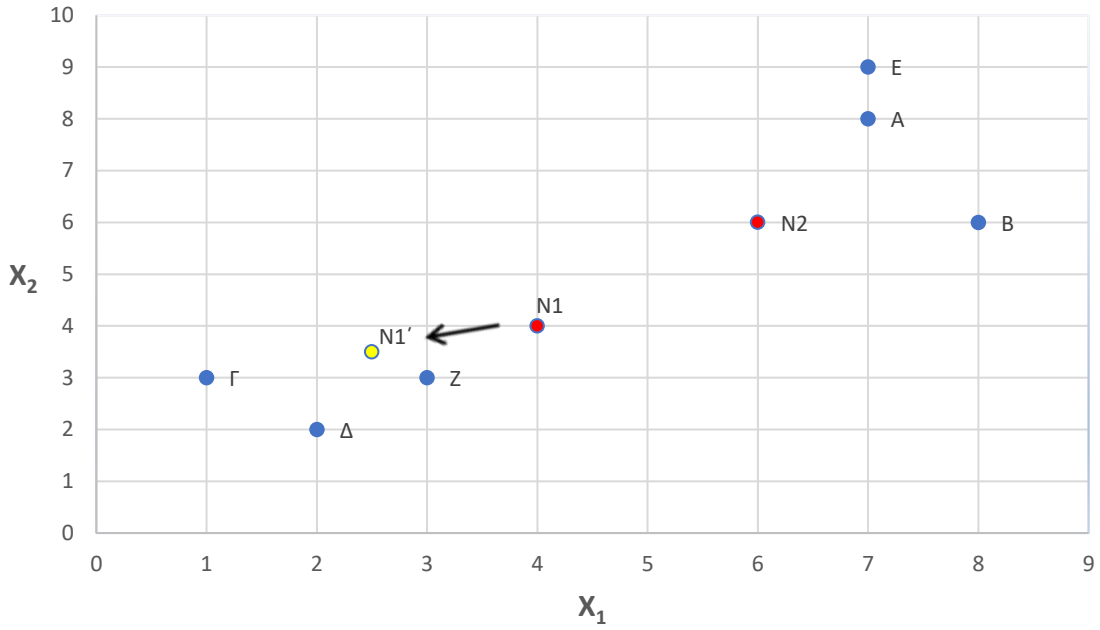
Υποθέτουμε πως στη συνέχεια εισάγεται το πρότυπο Γ (1, 3) και υπολογίζεται η απόστασή του από τους νευρώνες N1 και N2 (με τις ανανεωμένες πλέον συντεταγμένες σε όσους νευρώνες ανανεώθηκαν, εν προκειμένω του N2). Άρα:

Απόσταση Γ από 1ο νευρώνα N1: $d_1 = \sqrt{(w_{11} - x_\perp)^2 + (w_{21} - x_2)^2}$ = $\sqrt{(4-1)^2 + (4-3)^2}$ = 3.16
Απόσταση Γ από 2ο νευρώνα N2: $d_2 = \sqrt{(w_{12} - x_\perp)^2 + (w_{22} - x_2)^2}$ = $\sqrt{(6-1)^2 + (6-3)^2}$ = 5.83

Νικητής είναι ο νευρώνας N1 άρα τα βάρη του ανανεώνονται ως εξής:

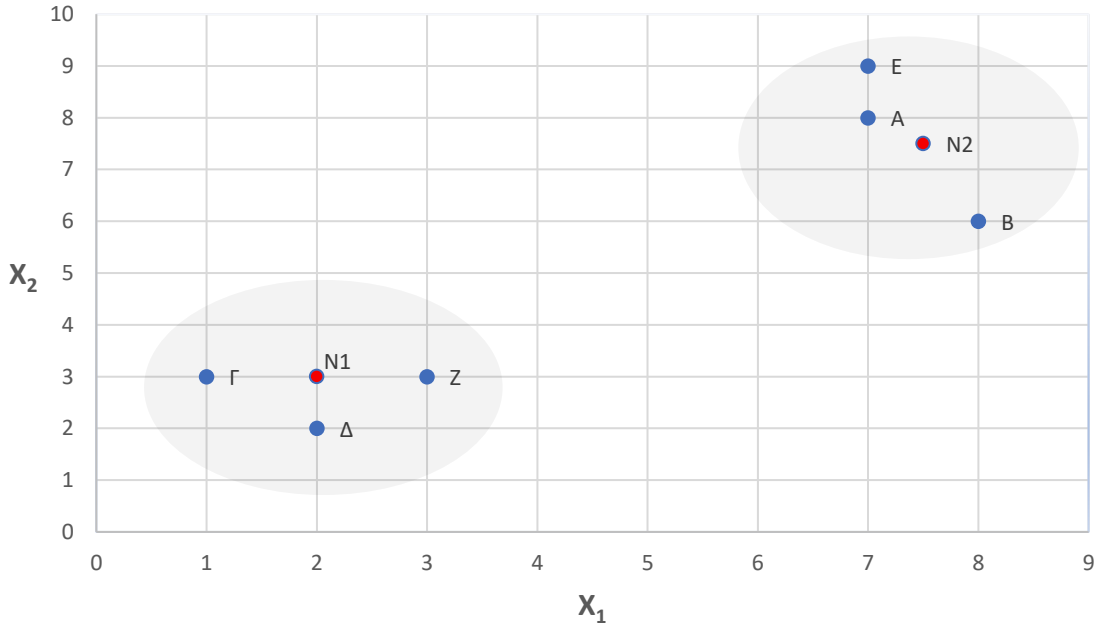
$\Delta W_{11} = a(x_1 - w_{11}) = 0.5(1 - 4) = -1.5$, $w_{11} = w_{11} + \Delta W_{11} = 4 - 1.5 = 2.5$
$\Delta W_{21} = a(x_2 - w_{12}) = 0.5(3 - 4) = -0.5$, $w_{21} = w_{21} + \Delta W_{21} = 4 - 0.5 = 3.5$

Συνεπώς ο νευρώνας N1 μετακινείται στις συντεταγμένες (2.5, 3.5) ώστε να προσεγγίσει το πρότυπο Γ (N1’ κίτρινη κουκκίδα στην εικόνα 15.4).



Εικόνα 15.4

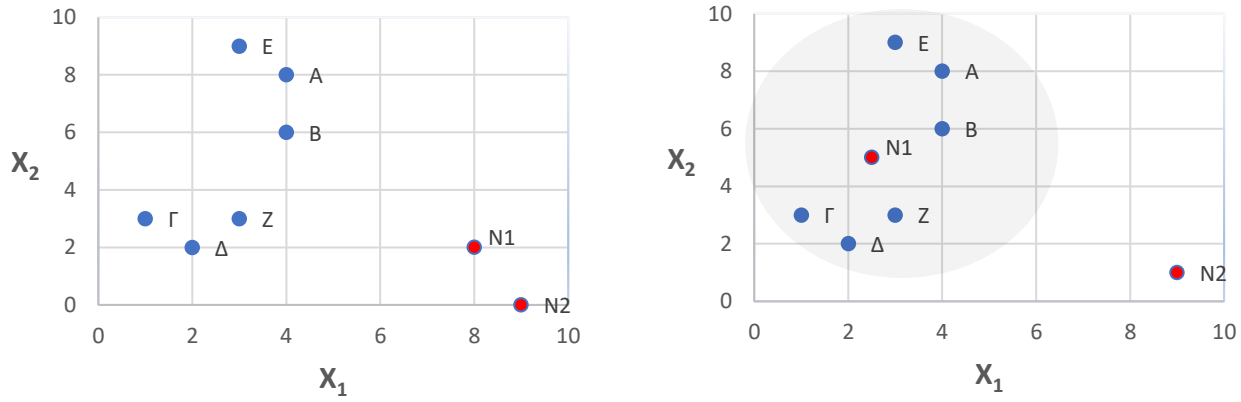
Η διαδικασία αυτή συνεχίζεται μέχρι να παρουσιαστούν όλα τα πρότυπα οπότε θα έχει ολοκληρωθεί μια εποχή και μετά από μερικές εποχές θα έχουμε πετύχει μια κατάσταση παρόμοια με αυτή του επομένου διαγράμματος (εικόνα 15.5) όπου θα έχουμε πετύχει την ομαδοποίηση των δεδομένων.



Εικόνα 15.5

15.3 Το Πρόβλημα των Νεκρών Νευρώνων

Ανάλογα με την αρχική τιμή των βαρών των νευρώνων Kohonen υπάρχει περίπτωση να εμφανιστεί το πρόβλημα των νεκρών νευρώνων, δηλαδή των νευρώνων εκείνων που δε θα νικάνε ποτέ, θα συντηρούν τα αρχικά τους βάρη και έτσι δε θα συμμετέχουν στη διαμόρφωση των ομάδων. Μία τέτοια κατάσταση περιγράφεται στο παρακάτω σχεδιάγραμμα (εικόνα 15.6), όπου μόνο ο νευρώνας N1 συμμετέχει στη λύση του προβλήματος ενώ ο N2 δεν μετακινείται καθόλου καθότι μονίμως είναι ο χαμένος νευρώνας.



Εικόνα 15.6

Λύσεις που μπορούν να εφαρμοστούν ώστε να ξεπεραστεί αυτό το πρόβλημα είναι:

Εισαγωγή μεταβλητού ρυθμού μάθησης:

Σε αυτήν την περίπτωση μπορεί να γίνει χρήση του παρακάτω τύπου:

$$a(n) = a(0)[1 - n/N],$$

όπου a(0) είναι η αρχική τιμή του ρυθμού εκπαίδευσης, n το τρέχον βήμα και N τα συνολικά βήματα εκπαίδευσης.

Τροποποίηση στα βάρη σε όλους της νευρώνες της γειτονιάς του νικητή

Η γειτονιά ορίζεται από την ακτίνα απόστασης από το νικητή νευρώνα, όσοι νευρώνες απέχουν το πολύ d(n) από το νικητή νευρώνα τροποποιούνται και μπορεί να οριστεί από τον παρακάτω τύπο:

$$d(n) = d(0)[1 - n/N]$$

όπου d(0) είναι η αρχική ακτίνα, n το τρέχον βήμα και N τα συνολικά βήματα.

16. Παραγωγικά Ανταγωνιστικά Δίκτυα - Generative Adversarial Networks GAN

Η κύρια χρήση των νευρωνικών δικτύων GAN είναι, όπως επισημαίνει και το όνομά τους, η παραγωγή δεδομένων. Μία τέτοια εφαρμογή είναι η παραγωγή εικόνων ακόμα και φωτογραφιών ανθρώπων που ενώ δεν υπάρχουν στην πραγματικότητα, η παραγόμενη φωτογραφία είναι εντελώς ρεαλιστική, εικόνα16.1 (Karras et al., 2018).

Εικόνα 16.1. (Karras et al., 2018)

16.1 Δομή GAN

Τα δίκτυα GANs έχουν ένα σύστημα δύο τεχνητών νευρωνικών δικτύων, το παραγωγικό – generative δίκτυο με στόχο να παράγει δεδομένα (πχ εικόνες) και το διαχωριστικό ΄- discriminator δίκτυο με στόχο να διαχωρίσει τα αληθινά από τα τεχνητά δεδομένα (πχ εικόνες) και τα δύο αυτά δίκτυα ανταγωνίζονται μεταξύ τους (Goodfellow et al., 2014), εικόνα 16.2.

Εικόνα 16.2

Γίνεται χρήση δύο loss function, μία για το generator δίκτυο και μία για το discriminator. Το δίκτυο discriminator πρέπει να προβλέψει αν το αποτέλεσμα είναι αληθινό ή τεχνητό, συνεπώς έχουμε ένα δυαδικό πρόβλημα – binary classification και για αυτές τις περιπτώσεις ενδείκνυται η χρήση της binary cross entropy loss function.

Όταν εκτελείται η τεχνική του gradient descent, παγώνουμε τα επίπεδα που βρίσκονται στον discriminator και η εκπαίδευση λαμβάνει χώρα μόνο στον generator. Στον generator διατηρείται η binary cross entropy loss function με τη διαφορά ότι οι στόχοι – labels αλλάζουν. Για παράδειγμα, αν για τον discriminator οι αληθινές εικόνες αντιστοιχούν στο 1 και οι τεχνητές στο 0, θα τον τροφοδοτήσουμε με τεχνητές εικόνες αλλά έχοντας το στόχο στο 1 έτσι ώστε να ενθαρρύνουμε το discriminator να αναγνωρίσει αυτές τις τεχνητές εικόνες ως αληθινές. Όμως τα βάρη του discriminator είναι παγωμένα οπότε δε θα αλλάξουν κατά τη διάρκεια αυτής της διαδικασίας. Μόνο τα βάρη του generator εκπαιδεύονται σε αυτό το στάδιο.

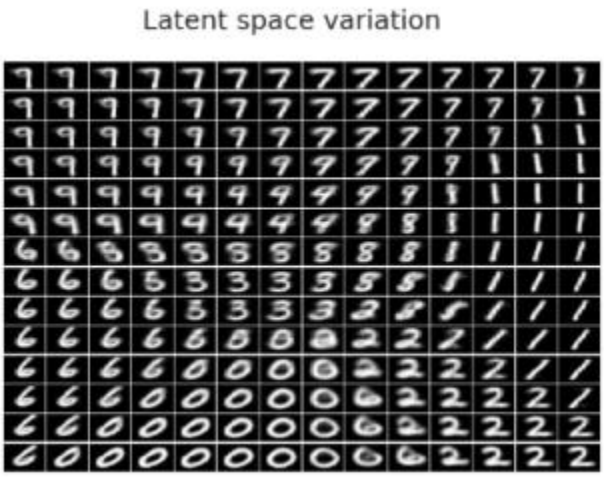
Γνωρίζουμε πως τα νευρωνικά δίκτυα πρέπει να έχουν εισόδους – inputs και εξόδους – outputs. Για το δίκτυο του discriminator αυτό είναι εύκολο καθότι έχει ως είσοδο τα δεδομένα, στην περίπτωσή μας μια εικόνα, και ως έξοδο απαντά στο αν είναι αληθινή ή όχι, εικόνα 16.3

Εικόνα 16.3

Στην περίπτωση του δικτύου του generator, γνωρίζουμε πως η έξοδος πρέπει να είναι η παραγωγή μιας εικόνας. Οι είσοδοι του generator θα είναι θόρυβος – noise ο οποίος θα δημιουργηθεί από μια τυποποιημένη κανονική κατανομή πολλών μεταβλητών (multivariate standard normal distribution) και ο generator θα μάθει να αποτυπώνει αυτόν τον θόρυβο σε μια εικόνα.

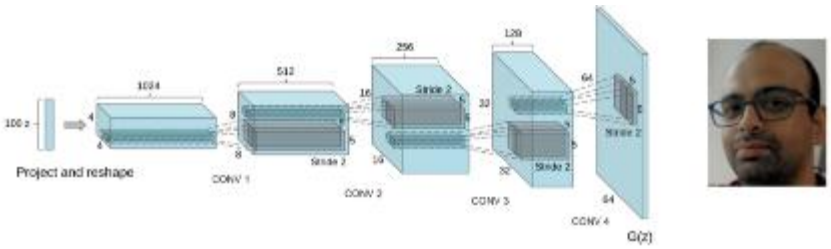
Εικόνα 16.4

Έχουμε δηλαδή ένα τυχαίο διάνυσμα z από μια τυποποιημένη κανονική κατανομή, μεγέθους για παράδειγμα D = 100, αν και πρόκειται για υπερ-παράμετρο (hyperparameter) οπότε υπάρχει δυνατότητα αλλαγής της, και αυτός ο χώρος των 100 διαστάσεων ονομάζεται λανθάνοντας χώρος – latent space. Μπορούμε να τον θεωρήσουμε ως έναν υποθετικό χώρο στον οποίο ο generator πιστεύει ότι βρίσκονται όλες οι εικόνες, εικόνα 16.5.



Εικόνα 16.5

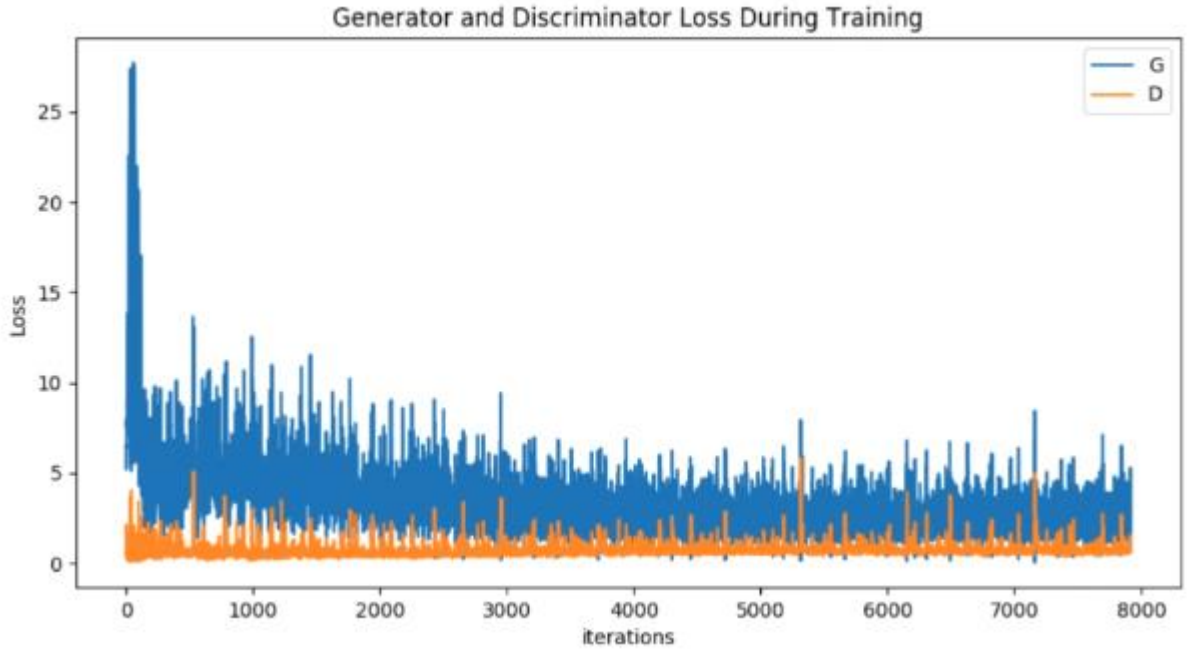
Παρατηρούμε πως στο χώρο αυτό τα ψηφία που μοιάζουν μεταξύ τους είναι σε διπλανές θέσεις (πχ το 1 με το 7 ή το 7 με το 9). Οι εικόνες που είναι μεταξύ των ορίων δύο ψηφίων δυσκολεύουν τόσο τους ανθρώπους όσο και το νευρωνικό δίκτυο στο να αποφασίσουν για το ποιο ψηφίο είναι στην πραγματικότητα. Ο generator μαθαίνει να συσχετίζει κάθε σημείο του λανθάνοντα χώρου με διαφορετικές εικόνες. Αργότερα, όταν παράγουμε εικόνες, διαλέγουμε ένα τυχαίο μέρος του λανθάνοντα χώρου και δημιουργούμε την εικόνα που αντιπροσωπεύεται στο συγκεκριμένο μέρος του χώρου. Συνοπτικά μπορούμε να πούμε πως στον generator ξεκινάμε με ένα διάνυσμα και το αποτυπώνουμε σε μια εικόνα, εικόνα 16.6.



Εικόνα 16.6

16.2 Αξιολόγηση GAN

Στα μοντέλα που έχουμε μελετήσει έως τώρα έχουμε παρατηρήσει πως με την πάροδο των εποχών η απώλεια μειώνεται και η ακρίβεια αυξάνεται. Στην περίπτωση όμως των GANs, ο generator και ο discriminator ανταγωνίζονται μεταξύ τους και έτσι οι συναρτήσεις απώλειας και για τον generator και για τον discriminator μοιάζουν με θόρυβο, εικόνα 16.7. Αν δούμε αποκλειστικά την απώλεια ανά επανάληψη – loss per iteration, φαίνεται σα να μη συμβαίνει τίποτα απολύτως. Ωστόσο και ο generator και ο discriminator βελτιώνονται γεγονός το οποίο δεν είναι εμφανές στο διάγραμμα loss per iteration καθότι ιδανικά και ο generator και ο discriminator βελτιώνεται με τον ίδιο ρυθμό.



Εικόνα 16.7

17. Αξιολόγηση Μοντέλου

17.1 Σωστά Θετικά – Λάθος Θετικά – Σωστά Αρνητικά – Λάθος Αρνητικά

Στο σημείο αυτό πρέπει να αναφερθούμε στο ζήτημα των σωστών και λάθος θετικών αποτελεσμάτων και σωστών και λάθος αρνητικών αποτελεσμάτων.

Αν υποθέσουμε πως δημιουργούμε έναν αλγόριθμο ο οποίος πρέπει να ξεχωρίζει την ανεπιθύμητη – σπαμ - από την επιθυμητή αλληλογραφία τότε ισχύει:

Σωστά Θετικά – True Positives: Τα μέηλ που είναι σπαμ και έχουν κατηγοριοποιηθεί από τον αλγόριθμο ως σπαμ. Η έννοια του θετικού αποτελέσματος για τον αλγόριθμο, σημαίνει κατηγοριοποίηση ενός μέηλ ως σπαμ.

Λάθος Θετικά – False Negatives: Τα μέηλ που δεν είναι σπαμ και έχουν κατηγοριοποιηθεί από τον αλγόριθμο ως σπαμ. Αυτός είναι ο λόγος που κατά καιρούς χρειάζεται να ελέγχουμε τον κατάλογο σπαμ μέηλ μας καθότι κάποιες φορές μπαίνουν εκεί κατά λάθος και επιθυμητά μέηλ.

Σωστά Αρνητικά – True Negatives: Τα μέηλ που δεν είναι σπαμ και έχουν κατηγοριοποιηθεί ως μη σπαμ.

Λάθος Αρνητικά – Wrong Negatives: Τα μέηλ που είναι σπαμ και έχουν κατηγοριοποιηθεί ως μη σπαμ. Πρόκειται για την περίπτωση που σπαμ μέηλ βρίσκονται στον φάκελο inbox των επιθυμητών μέηλ.

17.2 Δείκτης Recall

Ένας ακόμα δείκτης μέτρησης της απόδοσης του μοντέλου είναι το recall score το οποίο υπολογίζεται ως εξής:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Παρατηρούμε πως το μέγεθος των Λάθος Αρνητικών (το μέηλ κατηγοριοποιήθηκε ως μη σπαμ ενώ ήταν σπαμ, δηλαδή το μέηλ κατάφερε να περάσει στο inbox μας) έχει καθοριστική σημασία μιας και όταν γίνεται 0 τότε το κλάσμα παίρνει την τιμή 1 το οποίο είναι και η μέγιστη τιμή του recall score. Η ερμηνεία αυτού του δείκτη μπορεί να δοθεί μέσα από την ερώτηση: Από όλα τα σπαμ μέηλ, πόσα στην πραγματικότητα χαρακτήρισε το μοντέλο ως σπαμ; Ουσιαστικά είναι η αναλογία των σπαμ μέηλ που κατηγοριοποιήθηκαν σωστά (true positives) προς όλα τα σπαμ μέηλ που είχαμε στα δεδομένα μας (true positives & false negatives μιας και τα λάθος αρνητικά είναι και αυτά σπαμ μέηλ στην πραγματικότητα). Ο δείκτης προσπαθεί να μετρήσει την ικανότητα του μοντέλου μας να βρίσκει τα σημεία που μας ενδιαφέρουν, εν προκειμένω τα σπαμ μέηλ.

Ένα αρνητικό του δείκτη είναι ότι μπορούμε να χειραγωγήσουμε εύκολα το αποτέλεσμα. Αν για παράδειγμα χαρακτηρίσουμε όλα τα μέηλ ως σπαμ τότε δεν έχουμε καθόλου λάθος αρνητικά και το κλάσμα γίνεται 1 δίνοντας στο δείκτη τη μέγιστή του τιμή.

17.3 Δείκτης Precision – Positive Predicted Value

Ο συγκεκριμένος δείκτης ορίζεται ως εξής:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Πρόκειται για την αναλογία των σωστών θετικών, των σπαμ μέηλ που κατηγοριοποιήθηκαν ως σπαμ, προς το συνολικό αριθμό μέηλ που κατηγοριοποιήθηκαν ως σπαμ, είτε ήταν σωστή είτε λάθος η κατηρογοριοποίηση (αληθινά και λάθος θετικά). Η τιμή αυτού του δείκτη μεγαλώνει όταν μικραίνει το μέγεθος των λάθος θετικών αποτελεσμάτων του παρονομαστή.

17.4 Δείκτης F Score – F1 Score

Αυτός ο δείκτης συνδυάζει τους δείκτες recall και precision, οπότε λαμβάνει υπόψιν και τα λάθος θετικά και τα λάθος αρνητικά, και υπολογίζεται ως εξής:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

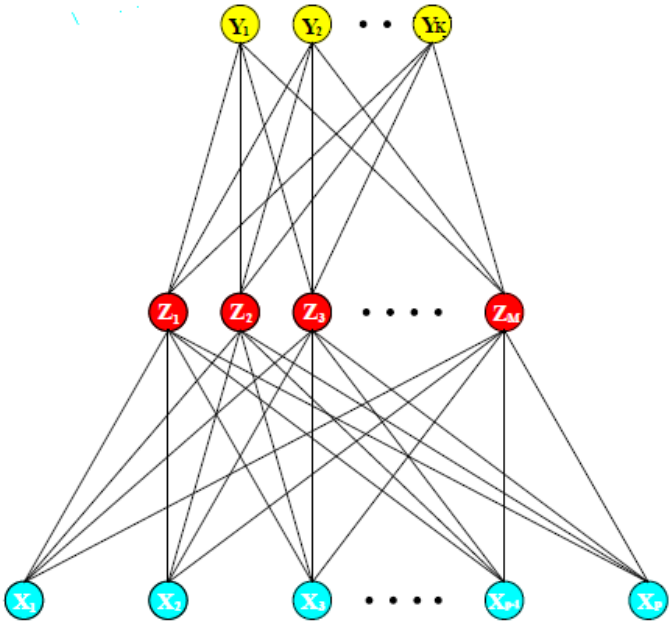
Η τιμή του κυμαίνεται μεταξύ 0 και 1.

18. Δημιουργία Τεχνητού Νευρωνικού Δικτύου Single Layer Perceptron Αναγνώρισης Εικόνων

Στο κεφάλαιο γίνεται χρήση του παρακάτω αρχείου:

- *18 Single Layer Perceptron.ipynb*

Μια μεγάλη κατηγορία νευρωνικών δικτύων είναι αυτά μονού επιπέδου τύπου perceptron (single layer perceptron) ή μονού κρυφού επιπέδου (single hidden layer) ή vanilla neural net (Hastie et al., 2009), εικόνα 18.1.



Εικόνα 18.1 (Hastie et al., 2009)

18.1 Συλλογή, Επεξεργασία και Εξερεύνηση Δεδομένων

Συγκεντρώνουμε τα δεδομένα του συνόλου MNIST (handwritten digits).

Οι τιμές των δεδομένων είναι στην κλίματα από 0 έως 255, συνεπώς τα κανονικοποιούμε διαιρώντας κάθε τιμή των δεδομένων με το 255.

Τέλος παρατηρούμε πως το σύνολο των δεδομένων για εκπαίδευση – train data είναι 60.000 και για δοκιμή 10.000.

Τα δεδομένα των χαρακτηριστικών (x_train, x_test) έχουν τις διαστάσεις των εικόνων του συνόλου MNIST, δηλαδή 28 επί 28 και τα δεδομένα των στόχων – labels περιέχουν τις κλάσεις στις οποίες πρέπει να κατηγοριοποιηθεί κάθε εικόνα, τα ψηφία δηλαδή από 0 έως 9, εικόνα 18.2.

```
# Load Data
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
# Noralize Data
x_train, x_test = x_train / 255.0, x_test / 255.0
print(f"x_train shape: {x_train.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
x_train shape: (60000, 28, 28)
y_train shape: (60000,)
x_test shape: (10000, 28, 28)
y_test shape: (10000,)
```

Εικόνα 18.2

18.2 Κατασκευή Μοντέλου

Στη συνέχεια κατασκευάζουμε το μοντέλο το οποίο στο επίπεδο εισόδου θα έχει το σύνολο των χαρακτηριστικών που είναι το πλήθος, 28 * 28 = 784, των πίξελ της κάθε εικόνας.

Εν συνεχεία δημιουργούμε ένα κρυφό επίπεδο με 128 νευρώνες και χρησιμοποιούμε ως συνάρτηση ενεργοποίησης τη Relu.

Τέλος δημιουργούμε το επίπεδο εξόδου με 10 νευρώνες, όσες και οι κλάσεις, με συνάρτηση ενεργοποίησης τη softmax.

```
# Build the model
model = tf.keras.models.Sequential([
  # Input 28 x 28
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

Εικόνα 18.3

Επιλέγουμε τον adam optimizer καθώς και την sparce categorical cross entropy loss function.

```
# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Εικόνα 18.4

18.3 Εκπαίδευση Μοντέλου

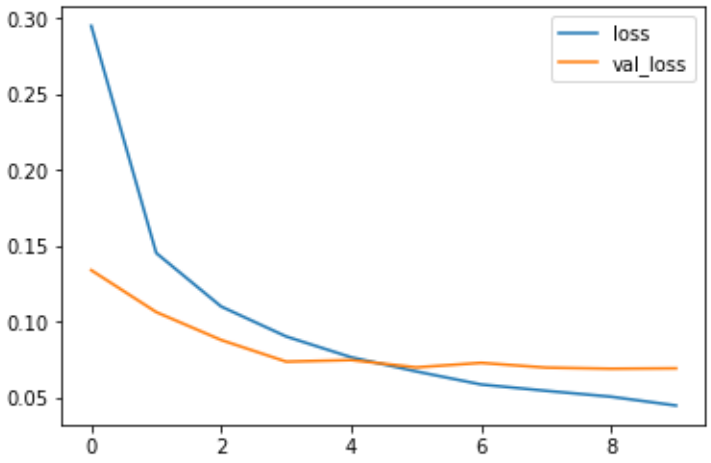
Εκπαιδεύουμε το μοντέλο για 10 εποχές, εικόνα 18.5

```
# Train the Model
r = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10)
```

Εικόνα 18.5

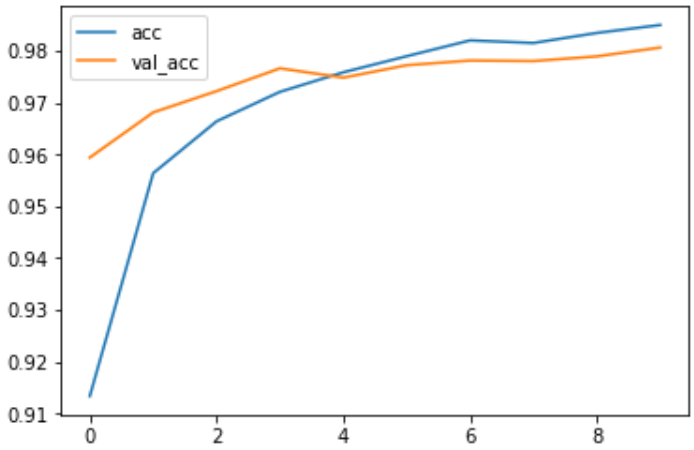
18.4 Αξιολόγηση Μοντέλου

Παρατηρούμε πως η απώλεια μειώνεται συνεχώς, τόσο στα δεδομένα εκπαίδευσης (γαλάζια γραμμή) όσο και στα δεδομένα δοκιμής (πορτοκαλί γραμμή) εικόνα 18.6.



Εικόνα 18.6

Αντίστοιχα παρατηρούμε την εξέλιξη της ακρίβειας η οποία αυξάνεται τόσο στα δεδομένα εκπαίδευσης όσο και στα δεδομένα δοκιμής με το πέρασμα των εποχών, εικόνα 18.7.



Εικόνα 18.7

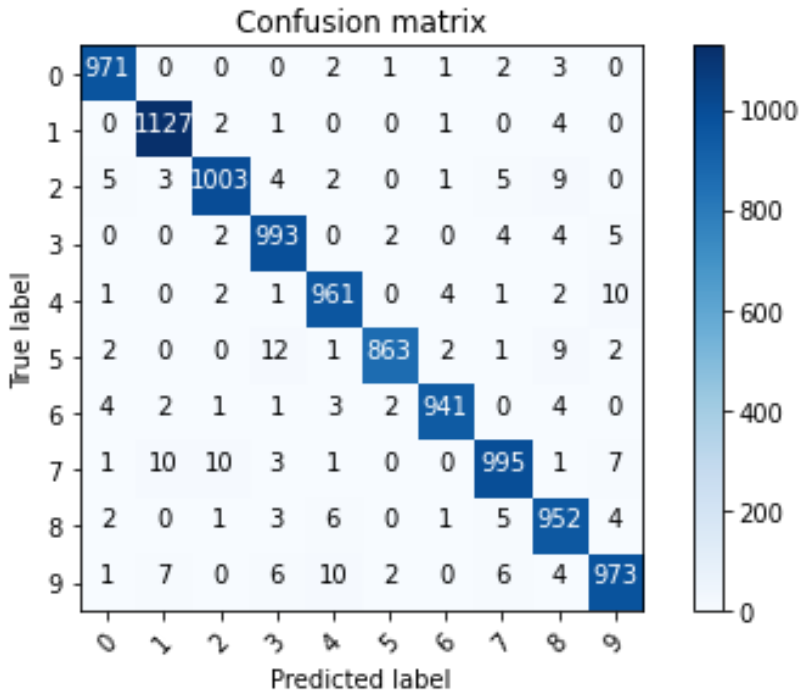
Κατασκευάζουμε την confusion matrix και περιμένουμε τα περισσότερα αποτελέσματα να είναι στη διαγώνιο μιας και η ακρίβεια του μοντέλου είναι ιδιαιτέρως υψηλή, εικόνα 18.8

```
# Evaluate the model
print(model.evaluate(x_test, y_test))
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0728 - accuracy: 0.9779
[0.07279146462678909, 0.9779000282287598]
```

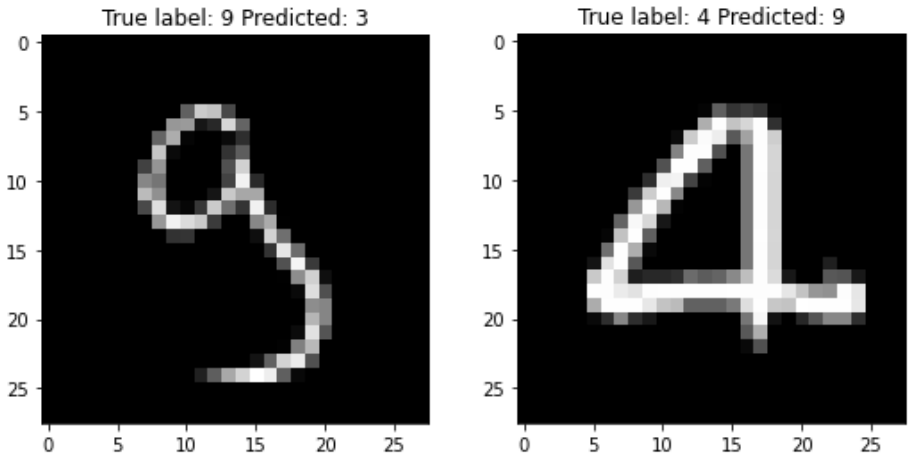
Εικόνα 18.8

Παρατηρούμε πως από τις μεγαλύτερες εσφαλμένες εκτιμήσεις είναι αυτές όπου στην εικόνα υπάρχει το ψηφίο 9 και το μοντέλο προέβλεψε 4 ή όπου στην εικόνα υπάρχει το ψηφία 7 και το μοντέλο προέβλεψε 2, λάθη που θα μπορούσε να κάνει εύκολα και ένας άνθρωπος , εικόνα 18.9



Εικόνα 18.9

Οπτικοποιούμε μερικά από τα λάθη του μοντέλου και παρατηρούμε την ομοιότητα των λάθος ψηφίων που προβλέφθηκαν με το πραγματικό ψηφίο, εικόνα 18.10.



Εικόνα 18.10

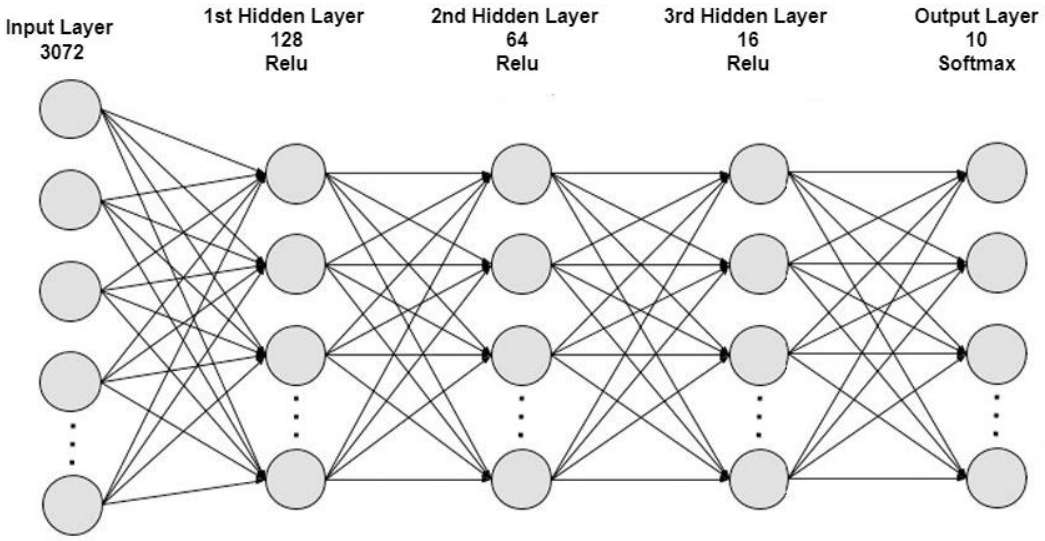
19. Δημιουργία Τεχνητού Νευρωνικού Δικτύου Multilayer Perceptron – MLP Αναγνώρισης Εικόνων

Στο κεφάλαιο γίνεται χρήση του παρακάτω αρχείου:

- *19 MLP CIFAR.ipynb*

Θα γίνει κατασκευή τεχνητού νευρωνικού δικτύου τύπου πολυεπίπεδου perceptron (multilayer perceptron) το οποίο θα έχει τη δυνατότητα να διακρίνει 10 διαφορετικές οντότητες σε εικόνες και πιο συγκεκριμένα σκύλους, πρόβατα, άλογα, βατράχους, ταράνδους, γάτες, φορτηγά, αυτοκίνητα και αεροπλάνα, θα έχει δηλαδή 10 διαφορετικές κλάσεις.

Συγκεκριμένα το τεχνητό νευρωνικό δίκτυο που θα κατασκευαστεί θα έχει 3072 νευρώνες στο επίπεδο εισόδου, 128 νευρώνες στο 1ο κρυφό επίπεδο, 64 νευρώνες στο 2ο κρυφό επίπεδο, 16 νευρώνες στο 3ο κρυφό επίπεδο και 10 νευρώνες στο επίπεδο εξόδου, εικόνα 20.1.



Εικόνα 19.1

19.1 Συλλογή Δεδομένων

Το σύνολο δεδομένων που θα χρησιμοποιηθεί σε αυτή την εφαρμογή είναι το CIFAR – 10 (https://www.cs.toronto.edu/~kriz/cifar.html) το οποίο περιέχει 60.000 φωτογραφίες από τις 10 κλάσεις που αναφέρθηκαν και χρησιμοποιείται συχνά για την εκπαίδευση μοντέλων στο θέμα της αναγνώρισης εικόνων (image recognition). Στα δεδομένα έχουμε τα χαρακτηριστικά x των εικόνων (πίξελ πλάτους, πίξελ ύψους, αριθμός χρωματικών καναλιών) και ο στόχος – ετικέτα (label) είναι η κλάση που πρέπει να αντιστοιχιστούν (σκύλος, γάτα κτλ). Επίσης τα δεδομένα χωρίζονται σε δεδομένα για εκπαίδευση και δεδομένα για δοκιμή του μοντέλου, εικόνα 19.2

```
# load_data() returns (x_train, y_train), (x_test, y_test), so we split data in train & test
(x_train_all, y_train_all), (x_test, y_test) = cifar10.load_data()
```

Εικόνα 19.2

19.2 Εξερεύνηση Δεδομένων

Τυπώνουμε τις 10 πρώτες εικόνες από το σύνολο δεδομένων για εκπαίδευση μαζί με το όνομα της κλάσης στην οποία ανήκει κάθε εικόνα.

```
plt.figure(figsize=(15, 5))

for i in range(10):
    # create subplots
    plt.subplot(1, 10, i+1)
    plt.imshow(x_train_all[i])
    # remove tick marks
    plt.xticks([])
    plt.yticks([])
    # insert label below image
    plt.xlabel(LABEL_NAMES[y_train_all[i][0]], fontsize=14)
```

Εικόνα 19.3

Παρατηρούμε πως κάθε εικόνα είναι αποθηκευμένη ως ένα πίνακας 3 διαστάσεων (πίξελ πλάτους, πίξελ ύψους, αριθμός RGB καναλιών σε κάθε πίξελ), εικόνα 19.4.

```
# shape of image: width x height x no of colour channels - rgb
x_train_all[0].shape
```

```
(32, 32, 3)
```

Εικόνα 19.4

```
# explore 1st image from x_train_all - the frog shown above
x_train_all[0]
```

```
array([[[ 59,  62,  63],
        [ 43,  46,  45],
        [ 50,  48,  43],
        ...,
        [158, 132, 108],
        [152, 125, 102],
        [148, 124, 103]],

       [[ 16,  20,  20],
        [  0,   0,   0],
        [ 18,   8,   0],
```

Εικόνα 19.5

Το σύνολο των δεδομένων των χαρακτηριστικών των εικόνων που προορίζονται για εκπαίδευση, x_train_all, είναι ένας πίνακας 4 διαστάσεων (αριθμός δεδομένων: 50.000, πίξελ πλάτους κάθε εικόνας: 32, πίξελ ύψους κάθε εικόνας: 32, αριθμός χρωματικών καναλιών: 3 – Red, Green, Blue), εικόνα 19.6.

```
nr_images, x, y, c = x_train_all.shape
print(f"No of images in x_train_all data set: {nr_images}")
print(f"No of width pixels in each image of x_train_all data set: {x}")
print(f"No of height pixels in each image of x_train_all data set: {y}")
print(f"No of colour channels (rgb) in x_train_all data set: {c}")
```

```
No of images in x_train_all data set: 50000
No of width pixels in each image of x_train_all data set: 32
No of height pixels in each image of x_train_all data set: 32
No of colour channels (rgb) in x_train_all data set: 3
```

Εικόνα 19.6

Αντίστοιχα το σύνολο δεδομένων με τους στόχους y_train_all έχει 50.000 εγγραφές, μία για κάθε εικόνα και είναι ένας ακέραιος αριθμός που αντιστοιχεί στην κλάση που ανήκει η εικόνα. Δίνουμε ονόματα σε αυτές τις 10 κλάσεις έτσι έχουμε:

0: Plane, 1: Car, 2: Bird, 3: Cat, 4: Deer, 5: Dog, 6: Frog, 7: Horse, 8: Ship, 9:Truck

Παρατηρούμε πως η 1η εικόνα από τα δεδομένα εκπαίδευσης αντιστοιχεί στην κλάση 6: Frog. Εικόνα 19.7.

```
print(f"No of labels in y_train_all set: {y_train_all.shape[0]}")
```

```
No of labels in y_train_all set: 50000
```

```
# 1st image belongs to class no 6 which is a Frog
y_train_all[0][0]
```

```
6
```

```
LABEL_NAMES[y_train_all[0][0]]
```

```
'Frog'
```

Εικόνα 19.7

Τα σύνολα δεδομένων των χαρακτηριστικών των και των στόχων των εικόνων που προορίζονται για δοκιμή του μοντέλου, x_test – y_test, έχουν από 10.000 δεδομένα, εικόνα 19.8.

```
# 10.000 testing images
x_test.shape
```

```
(10000, 32, 32, 3)
```

```
print(f"No of test images: {x_test.shape[0]}")
print(f"No of width pixels in each image of x_train_all data set: {x_test.shape[1]}")
print(f"No of height pixels in each image of x_train_all data set: {x_test.shape[2]}")
print(f"No of colour channels (rgb) in x_train_all data set: {x_test.shape[3]}")
```

```
No of test images: 10000
No of width pixels in each image of x_train_all data set: 32
No of height pixels in each image of x_train_all data set: 32
No of colour channels (rgb) in x_train_all data set: 3
```

```
# 10.000 testing labels
y_test.shape
```

```
(10000, 1)
```

Εικόνα 19.8

19.3 Επεξεργασία Δεδομένων

Όλα τα δεδομένα είναι μη προσημασμένοι (unsigned) ακέραιοι αριθμοί από το 0 έως το 255. Διαιρούμε όλους τους αριθμούς με το 255 ώστε να είναι σε μια κλίμακα από το 0 έως 1 καθώς ο ρυθμός εκπαίδευσης που θα χρησιμοποιηθεί είναι ένας αρκετά μικρός αριθμός και έτσι θα διευκολυνθεί ο υπολογισμός των απωλειών και η αναπροσαρμογή των βαρών, εικόνα 19.9.

```
# divide no in arrays by 255 to make them between 0 - 1 & convert them to floats
x_train_all, x_test = x_train_all / 255.0, x_test / 255.0
```

Εικόνα 19.9

Επίσης για διευκόλυνση των υπολογισμών μετατρέπουμε τους πίνακες με τα χαρακτηριστικά των εικόνων, x_train_all και x_test, από πίνακες 4 διαστάσεων (αριθμός δεδομένων X πίξελ πλάτους X πίξελ ύψους X αριθμός χρωματικών καναλιών) σε πίνακες 2 διαστάσεων (1η διάσταση: αριθμός δεδομένων X 2η διάσταση: [πίξελ πλάτους * πίξελ ύψους * αριθμός χρωματικών καναλιών]). Έτσι για τα δεδομένα εκπαίδευσης προκύπτει ένας πίνακας διαστάσεων 50.000 εγγραφών επί 32*32*3 = 50.000 επί 3072 και για τα δεδομένα δοκιμής ένας πίνακας διαστάσεων 10.000 επί 3072, εικόνες 19.10, 19.11, 19.12.

```
# flatten data from 4d to 1d. Collapse 3 final dimensions: width x height x colour channel
x_train_all = x_train_all.reshape(x_train_all.shape[0], TOTAL_INPUTS)
x_train_all
```

```
array([[0.23137255, 0.24313725, 0.24705882, ..., 0.48235294, 0.36078431,
        0.28235294],
       [0.60392157, 0.69411765, 0.73333333, ..., 0.56078431, 0.52156863,
        0.56470588],
       [1.        , 1.        , 1.        , ..., 0.31372549, 0.3372549 ,
```

Εικόνα 19.10

```
# new shape: 50,000 x 32*32*3 = 50,000 x 3072
x_train_all.shape
```

```
(50000, 3072)
```

```
x_test = x_test.reshape(len(x_test), TOTAL_INPUTS)
x_test
```

```
array([[0.61960784, 0.43921569, 0.19215686, ..., 0.08235294, 0.2627451 ,
        0.43137255],
       [0.92156863, 0.92156863, 0.92156863, ..., 0.72941176, 0.78431373,
        0.78039216],
```

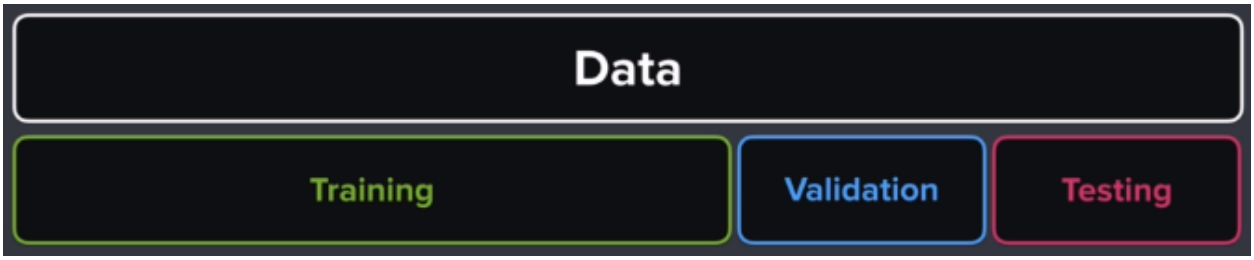
Εικόνα 19.11

```
x_test.shape
```

```
(10000, 3072)
```

Εικόνα 19.12

Στη συνέχεια χωρίζουμε ένα τμήμα από τα δεδομένα εκπαίδευσης με το οποίο θα δημιουργήσουμε ένα σύνολο δεδομένων επικύρωσης (validation data set), εικόνα 20.13. Αυτό το σύνολο μας παρέχει μια αμερόληπτη εκτίμηση για το πως συμπεριφέρεται το μοντέλο μας σε περίπτωση που χρειαστεί να κάνουμε κάποιες διορθώσεις μετά την εκπαίδευση, έτσι ώστε να διαλέξουμε την καλύτερη εκδοχή του. Το σύνολο με τα δεδομένα ελέγχου (test data set) θα χρησιμοποιηθεί για την τελική αξιολόγηση του μοντέλου. Μόνο η καλύτερη εκδοχή του μοντέλου θα χρησιμοποιήσει τα δεδομένα ελέγχου καθώς ο σκοπός του testing set είναι να μας δώσει μια ρεαλιστική εκδοχή για το πως θα λειτουργήσει το μοντέλο στον πραγματικό κόσμο. Σε περίπτωση που θέλουμε να κάνουμε κάποιες διορθώσεις στο μοντέλο και κάνουμε χρήση του training set ελλοχεύει ο κίνδυνος να συντονίσουμε το μοντέλο κατά τέτοιον τρόπο ώστε να έχει πολύ καλές επιδόσεις στα δεδομένα του training set μόνο και έτσι να καταλήξουμε με μη ρεαλιστικά αποτελέσματα.



Εικόνα 19.13

Δημιουργούμε το validation data set με τις 10.000 πρώτες εικόνες του training data set, εικόνα 19.14.

```
# first 10,000 images for validation data set
x_val = x_train_all[:VALIDATION_SIZE]
y_val = y_train_all[:VALIDATION_SIZE]
x_val.shape
```

```
(10000, 3072)
```

```
y_val.shape
```

```
(10000, 1)
```

Εικόνα 19.14

Οι επόμενες 40.000 εικόνες θα αποτελέσουν το ανανεωμένο training data set, εικόνα 19.15

```
# last 40.000 remaining images for x_train & y_train
x_train = x_train_all[VALIDATION_SIZE:]
y_train = y_train_all[VALIDATION_SIZE:]
x_train.shape
```

```
(40000, 3072)
```

```
y_train.shape
```

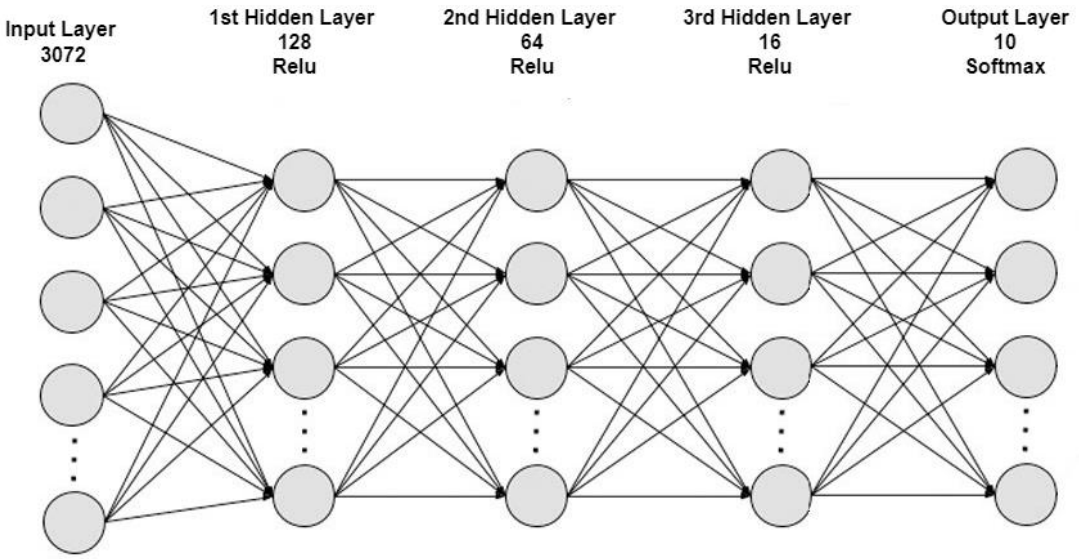
```
(40000, 1)
```

Εικόνα 19.15

19.4 Κατασκευή Νευρωνικού Δικτύου

Το νευρωνικό δίκτυο που θα κατασκευαστεί θα έχει, εικόνες 19.16, 19.17:

- ένα επίπεδο εισόδου: Το επίπεδο εισόδου θα έχει τόσους νευρώνες που θα καθοριστούν από τον αριθμό των πίξελ πλάτους της εικόνας επί τον αριθμό των πίξελ ύψους της εικόνας επί τον αριθμό των υφιστάμενων χρωματικών καναλιών. Κατά συνέπεια, στο συγκεκριμένο παράδειγμα υπάρχουν 32 πίξελ πλάτους, 32 πίξελ ύψους και 3 χρωματικά κανάλια σε κάθε εικόνα, έτσι οι νευρώνες εισόδου θα είναι 32*32*3 = 3072. 1^{ο} κρυφό επίπεδο με 128 νευρώνες.
- 2^{ο} κρυφό επίπεδο με 64 νευρώνες.
- 3^{ο} κρυφό επίπεδο με 16 νευρώνες.
- 1 επίπεδο εξόδου με 10 νευρώνες.
- Κάθε νευρώνας ενός επιπέδου συνδέεται με όλους τους νευρώνες του αμέσως επόμενου επιπέδου.
- Η συνάρτηση ενεργοποίησης των νευρώνων των 2 κρυφών επιπέδων θα είναι η ReLU.
- Η συνάρτηση ενεργοποίησης των νευρώνων του επιπέδου εξόδου θα είναι η softmax, που θα μεταμορφώσει τα αποτελέσματα εξόδου σε πιθανότητες (όλα τα αποτελέσματά της θα είναι αριθμοί μεταξύ του 0 και του 1 και το άθροισμά τους θα ισούται με 1).
- Η συνάρτηση υπολογισμού σφάλματος θα είναι η sparse categorical cross entropy.
- Θα χρησιμοποιηθεί ο Adam optimizer για βελτιστοποίηση.



Εικόνα 19.16

```
# create 1st model
model_1 = Sequential([
    # 1st hidden layer, 128 output units - layer neurons, 3072 inputs, activation f: relu
    Dense(units=128, input_dim=TOTAL_INPUTS, activation='relu', name='m1_hidden1'),
    # 2nd hidden layer, 64 output units - layer neurons, no need to specify inputs - keras figures it out
    Dense(units=64, activation='relu', name='m1_hidden2'),
    # 3hidden layer
    Dense(units=16, activation='relu', name='m1_hidden3'),
    # output layer
    Dense(units=10, activation='softmax', name='m1_output')
])

# compile model
model_1.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Εικόνα 19.17

Στην εικόνα 20.18 παρατηρούμε τα επίπεδα του μοντέλου (layer), τον αριθμό νευρώνων κάθε επιπέδου (output shape) και τις παραμέτρους που αναφέρονται στα βάρη.

Πιο συγκεκριμένα στο κρυφό επίπεδο 1 έρχονται είσοδοι από 32 x 32 x 3 = 3.072 νευρώνες εισόδου και το επίπεδο έχει 128 νευρώνες. Συνεπώς υπάρχουν 3.072 x 128 = 393.216 συνάψεις με βάρη. Επιπλέον κάθε νευρώνας του κρυφού επιπέδου έχει τη σύναψη με τον bias όρο, συνεπώς οι συνάψεις ανέρχονται στον αριθμό 393.215 + 128 = 393.344, εικόνα 19.18.

Στο 2ο κρυφό επίπεδο έρχονται πληροφορίες από τους 128 νευρώνες του 1ου κρυφού επιπέδου, οι νευρώνες του 2ου κρυφού επιπέδου είναι 64 άρα υπάρχουν 128 x 64 = 8.192 συνάψεις. Σε αυτές προσθέτουμε τις συνάψεις των νευρώνων αυτού του επιπέδου με τον όρο bias που αντιστοιχεί σε κάθε νευρώνα και έχουμε 8.192 + 64 = 8.256 συνάψεις, εικόνα 19.18.

Ομοίως στο 3ο κρυφό επίπεδο οι συνάψεις είναι 64 x 16 + 16 = 1.040, εικόνα 19.18.

Και στο επίπεδο εξόδου οι συνάψεις είναι 16 x 10 + 10 = 170, εικόνα 19.18.

Συνολικά υπάρχουν 393.344 + 8.256 + 1.040 + 170 = 402.810 συνάψεις (param), εικόνα 19.18.

```
# what model looks like
model_1.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 m1_hidden1 (Dense)          (None, 128)               393344

 m1_hidden2 (Dense)          (None, 64)                8256

 m1_hidden3 (Dense)          (None, 16)                1040

 m1_output (Dense)           (None, 10)                170

=================================================================
Total params: 402,810
Trainable params: 402,810
Non-trainable params: 0
```

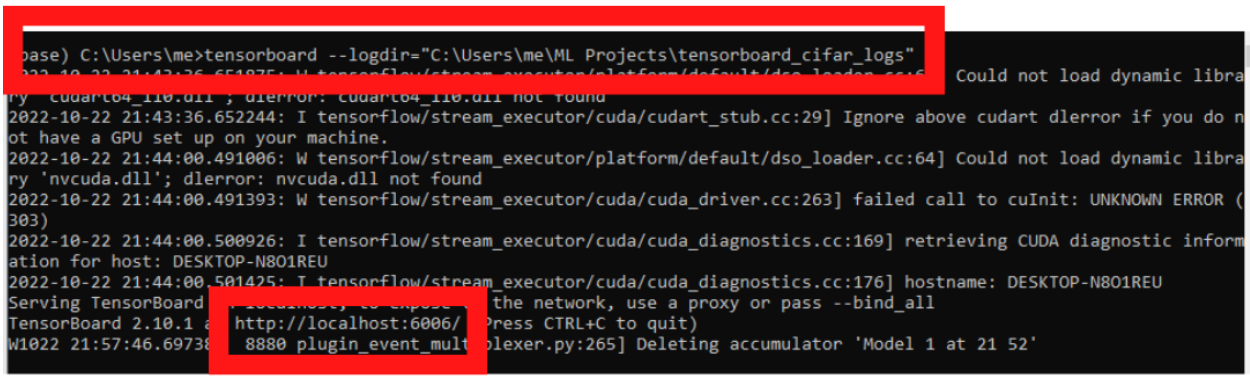
Εικόνα 19.18

Δημιουργώ μία συνάρτηση με σκοπό την κατασκευή αρχείων στα οποία θα έχει πρόσβαση το εργαλείο tensorboard ώστε να εξάγω συμπεράσματα, εικόνα 19.19.

```
def get_tensorboard(model_name):

    # set up folder of files tensorboard will use
    # get the current hour & minute to name folder name
    folder_name = f'{model_name} at {strftime("%H %M")}'

    # join root directory with folder name to create a path
    dir_paths = os.path.join(LOG_DIR, folder_name)

    # make directories
    try:
        os.makedirs(dir_paths)
    except OSError as err:
        print(err.strerror)
    else:
        print('Successfully created directory')

    return TensorBoard(log_dir=dir_paths)
```

Εικόνα 19.19

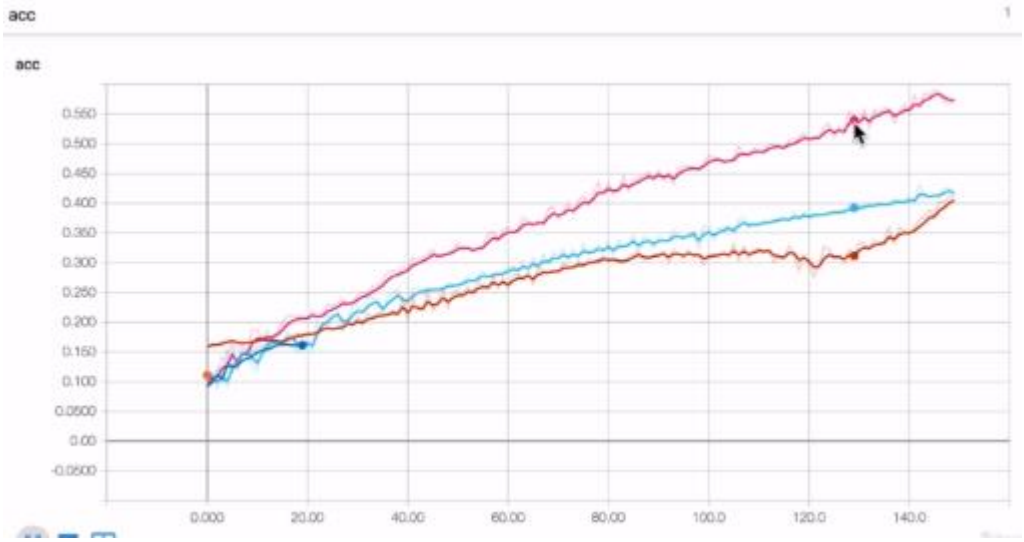
Στο Anaconda Prompt δίνω εντολή ώστε να μπορέσω να ανοίξω το tensorboard, εικόνα 19.20:



Εικόνα 19.20

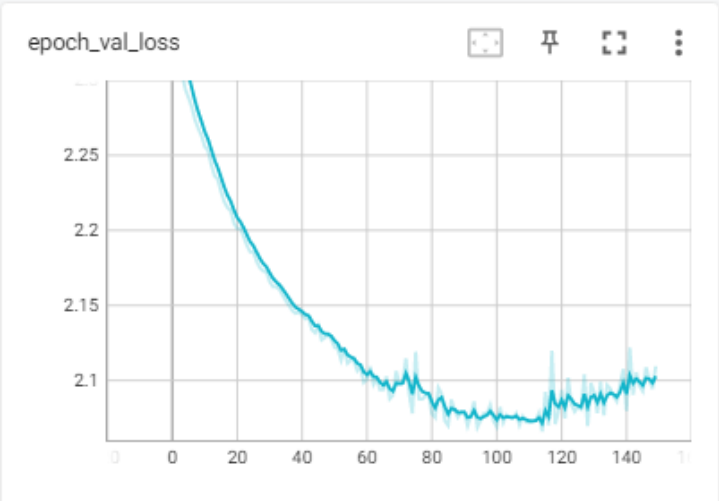
19.5 Εκπαίδευση Μοντέλου

Εκπαιδεύοντας το μοντέλο με ένα μικρό σύνολο δεδομένων για τον ίδιο αριθμό εποχών και για τα ίδια δεδομένα, παρατηρώ κάποιες διαφορές στην εξέλιξη της ακρίβειας. Αυτό συμβαίνει διότι ο optimizer εισάγει κάποια τυχαιότητα, όπως για παράδειγμα το σημείο εκκίνησης, εικόνα 19.21.



Εικόνα 19.21

Χρησιμοποιώντας και το σύνολο validation set παρατηρούμε ότι η απώλεια ξεκινάει από υψηλό επίπεδο, μειώνεται κατά την εκπαίδευση και από ένα σημείο και μετά αρχίζει και ανεβαίνει. Αυτό το πρόβλημα σχετίζεται με το overfitting, εικόνα 19.22. Στον εντοπισμό τέτοιων φαινομένων, η ύπαρξη του validation set είναι ιδιαίτερα χρήσιμη.



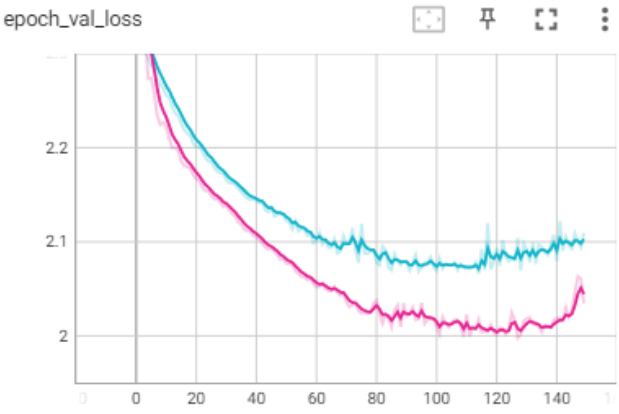
Εικόνα 19.22

Δημιουργούμε ένα νέο μοντέλο και εφαρμόζουμε την τεχνική του droupout με πιθανότητα μη λειτουργίας 0.2 στο σύνολο των νευρώνων του επιπέδου εισόδου, εικόνα 19.23.

```
# 2nd Model with Dropout probability 0.2
model_2 = Sequential()
model_2.add(Dropout(0.2, seed=42, input_shape=(TOTAL_INPUTS,)))
model_2.add(Dense(128, activation='relu', name='m2_hidden1'))
model_2.add(Dense(64, activation='relu', name='m2_hidden2'))
model_2.add(Dense(16, activation='relu', name='m2_hidden3'))
model_2.add(Dense(10, activation='softmax', name='m2_output'))

# compile model
model_2.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

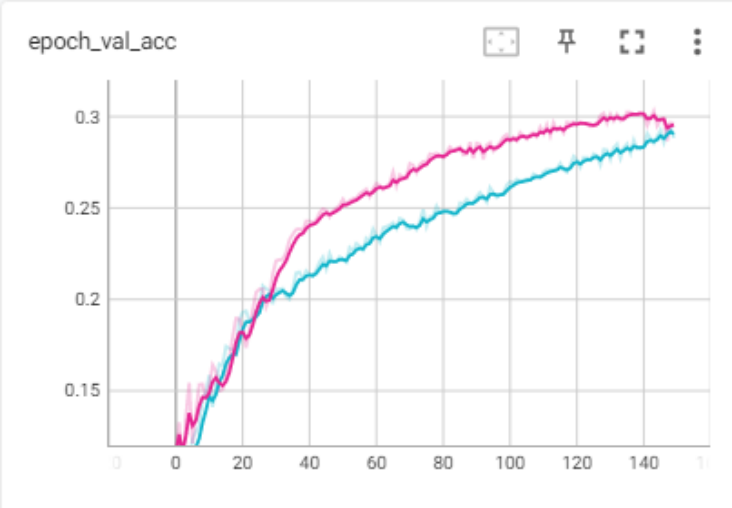
Εικόνα 19.24

Παρατηρούμε ότι το 2^{ο} μοντέλο με το dropout (ιώδης γραμμή στην εικόνα 19.25) μειώνει την απώλεια σε σχέση με το 1^{ο} μοντέλο (γαλάζια γραμμή) ωστόσο δεν το εξαλείφει, συνεπώς είναι προτιμότερο να χρησιμοποιηθεί η τεχνική του dropout σε συνδυασμό με το early stopping.



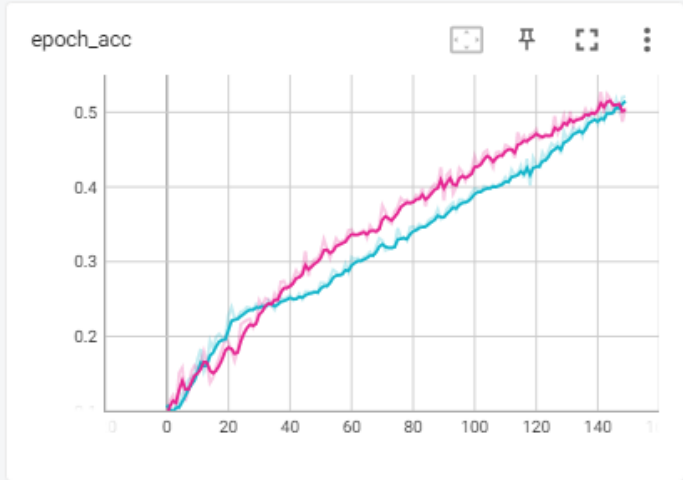
Εικόνα 19.25

Παρατηρούμε πως δεν έχουμε απώλειες στην ακρίβεια του 2ου μοντέλου και πως και τα δύο δίνουν ακρίβεια κοντά στο 0.3, εικόνα 19.26.



Εικόνα 19.26

Επίσης παρατηρούμε πως το 2ο μοντέλο με το dropout (γαλάζια γραμμή) μαθαίνει με πιο αργό ρυθμό σε σχέση με το 1ο μοντέλο, .



Εικόνα 19.27

Κατασκευάζουμε και ένα τρίτο μοντέλο στο οποίο προσθέτουμε ένα ακόμα dropout στο 1ο κρυφό επίπεδο, εικόνα 19.28.

```
# 3rd Model with 2 Dropouts - input layer + 1st hidden layer
model_3 = Sequential()
model_3.add(Dropout(0.2, seed=42, input_shape=(TOTAL_INPUTS,)))
model_3.add(Dense(128, activation='relu', name='m3_hidden1'))
model_3.add(Dropout(0.25, seed=42))
model_3.add(Dense(64, activation='relu', name='m3_hidden2'))
model_3.add(Dense(16, activation='relu', name='m3_hidden3'))
model_3.add(Dense(10, activation='softmax', name='m3_output'))

# compile model
model_3.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

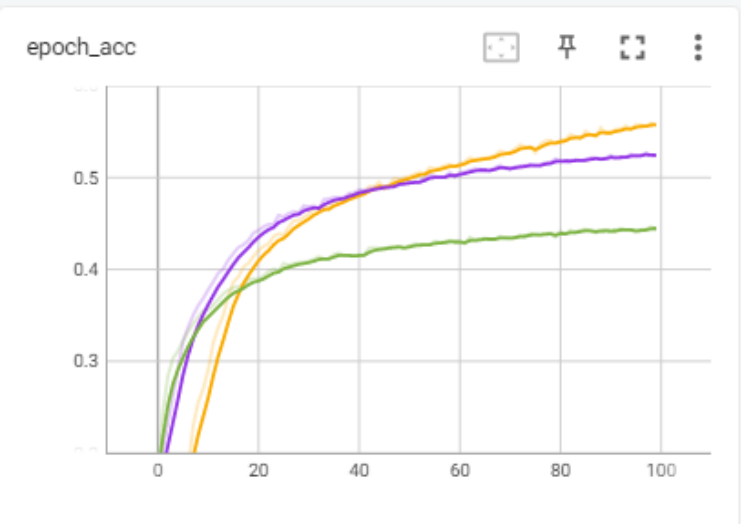
Εικόνα 19.28

Στη συνέχεια εκπαιδεύουμε τα 3 αυτά μοντέλα με το training set των 40,000 δεδομένων και έχουμε:

- Πορτοκαλί γραμμή: 1ο Μοντέλο - Καθόλου dropout
- Ιώδης γραμμή: 2ο Μοντέλο - dropout στο επίπεδο εισόδου
- Πράσινη γραμμή: 3ο Μοντέλο - dropout στο επίπεδο εισόδου και στο 1ο κρυφό επίπεδο

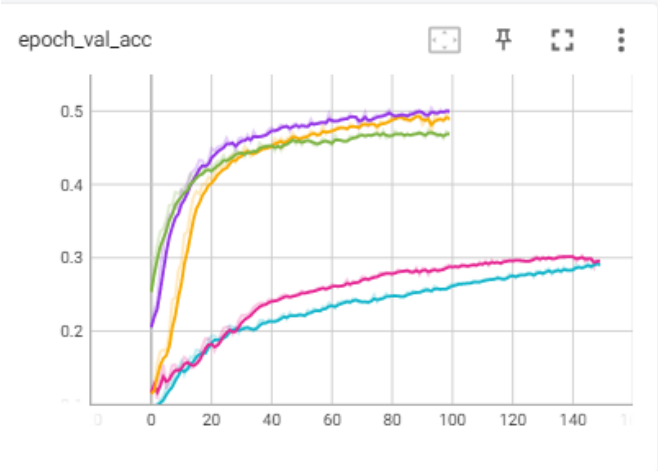
19.6 Αξιολόγηση Μοντέλου

Παρατηρούμε πως το droupout προκαλεί καθυστέρηση στην εκπαίδευση των μοντέλων εικόνα 19.29



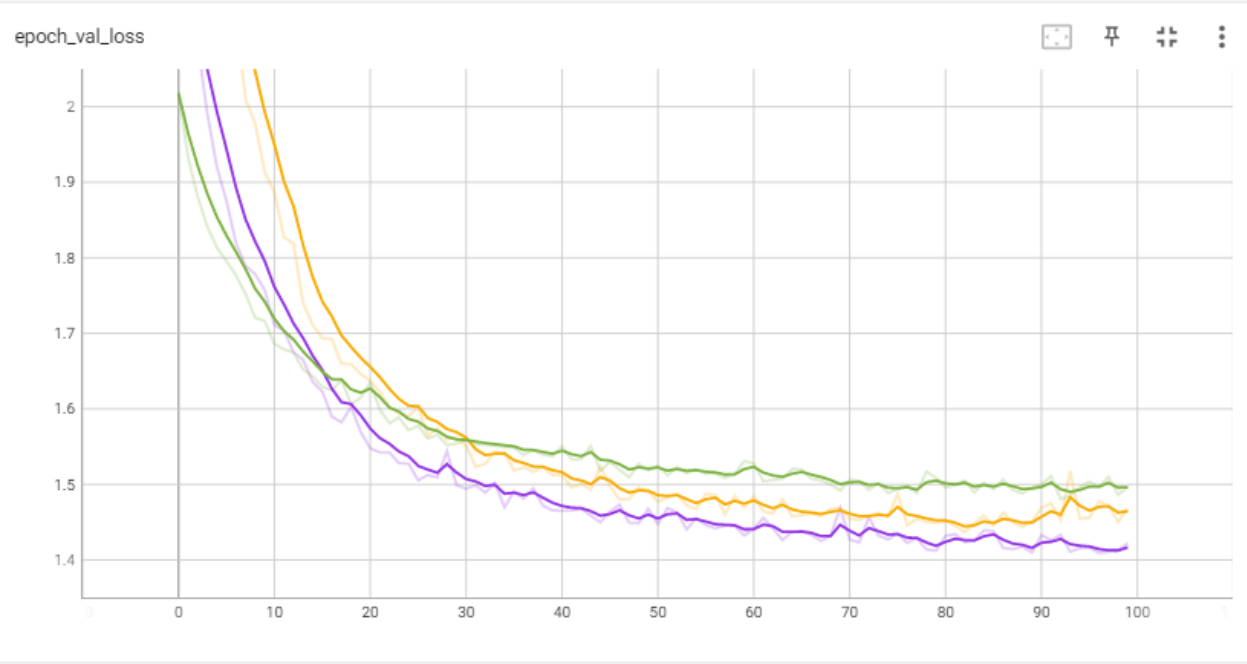
Εικόνα 19.29

Σχετικά με την ακρίβεια στην επικύρωση (validation accuracy) τα αποτελέσματα είναι πολύ διαφορετικά όταν εκπαιδεύουμε τα μοντέλα με λίγα δεδομένα και πολλά. Πιο συγκεκριμένα η ακρίβεια είναι σημαντικά υψηλότερη στην εκπαίδευση των 3 μοντέλων με το training set (40.000 δεδομένα) σε σχέση με την εκπαίδευση του 1ου μοντέλου (γαλάζια γραμμή) και του 2ου (ροζ γραμμή) με το extra small xs data set (1000 δεδομένα), εικόνα 19.30



Εικόνα 19.30

Στην απώλεια επικύρωσης (validation loss) παρατηρούμε πως τα μοντέλα 2 (ιώδες) και 3 (πράσινο) με το dropout έχουν μείωση της απώλειας ενώ στο 1ο μοντέλο (πορτοκαλί) χωρίς dropout από κάποια εποχή και μετά η απώλεια αυξάνεται, οπότε θα ήταν καλό η εκπαίδευση του μοντέλου 1 να σταματούσε λίγο νωρίτερα, εικόνα 19.31.



Εικόνα 9.31

Συμπερασματικά μπορούμε να ισχυριστούμε ότι:

- Αυξάνοντας τον αριθμό των δεδομένων εκπαίδευσης λαμβάνουμε πολύ καλύτερα αποτελέσματα ακρίβειας και overfitting.
- Το early stopping είναι σε πολλές περιπτώσεις απαραίτητο και μπορεί να συνδυαστεί με το dropout. Η εκπαίδευση των μοντέλων σε όλο και περισσότερες εποχές δεν επιφέρει απαραίτητα θετικά αποτελέσματα.

- Τα μοντέλα που χρησιμοποιούν την τεχνική του dropout για να μειώσουν το overfitting έχουν πιο αργή διαδικασία εκπαίδευσης, ωστόσο οδηγούν σε καλύτερη validation accuracy.
- Η αύξηση του αριθμού των δεδομένων (εν προκειμένω από 1.000 σε 40.000) είχε πολύ εντονότερα θετική επίπτωση στην ακρίβεια των μοντέλων από ότι το dropout.

Με τη χρήση των συνόλων δεδομένων δοκιμής (test data set) μπορούμε να αξιολογήσουμε τα μοντέλα μας.

Για το μοντέλο 2 για παράδειγμα λαμβάνουμε απώλεια – loss 1.42 και ακρίβεια 49.9% γεγονός το οποίο περιμέναμε καθώς έχουμε ήδη τα αποτελέσματα του validation test, εικόνα 19.32.

```
test_loss, test_accuracy = model_2.evaluate(x_test, y_test)
print(f"Test loos is {test_loss:0.3} and test accuracy is {test_accuracy:0.1%}")
```

```
Test loos is 1.42 and test accuracy is 49.9%
```

Εικόνα 19.32

Δημιουργούμε έναν πίνακα σύγχυσης (confusion matrix) για το μοντέλο.

True Positives: Το μοντέλο προέβλεψε το σωστό αποτέλεσμα. Είναι οι τιμές στη διαγώνιο της confusion matrix, εικόνα 19.33.

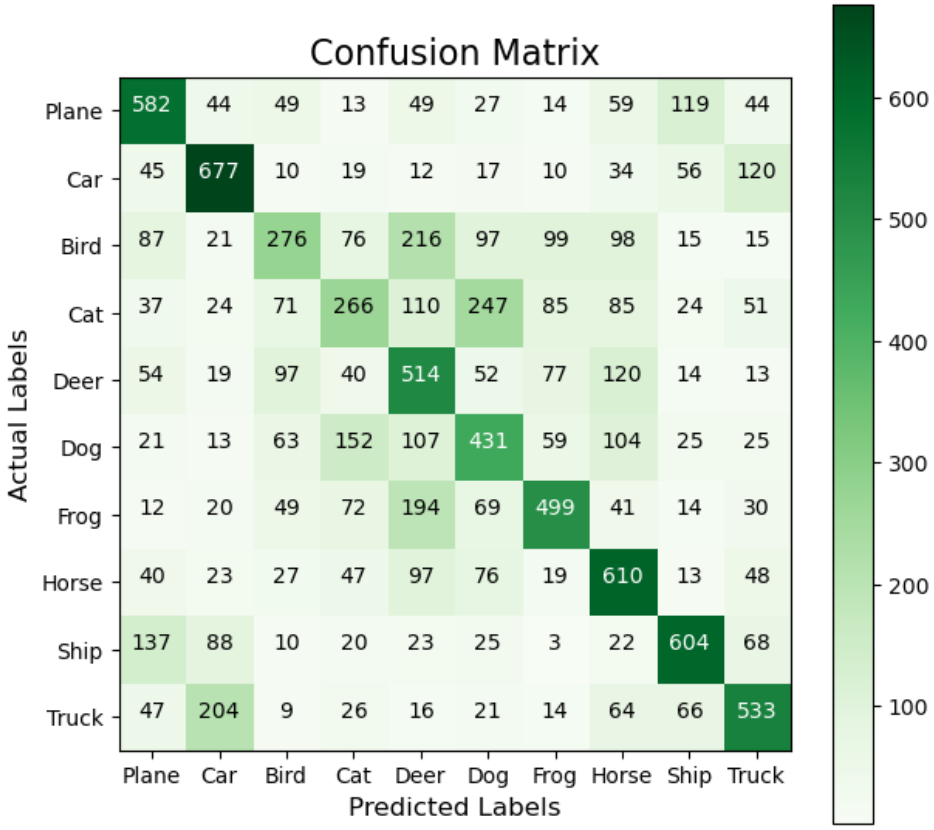
```
np.diag(conf_matrix)
```

```
array([582, 677, 276, 266, 514, 431, 499, 610, 604, 533], dtype=int64)
```

Εικόνα 19.33

False Positives: Αθροίζοντας τα στοιχεία κάθε στήλης, με εξαίρεση το σημείο της διαγωνίου, παίρνουμε το σύνολο των false positives για κάθε κλάση. Όλη η στήλη Plane, με εξαίρεση την τιμή 582, εικόνα 19.34, που είναι η σωστές προβλέψεις, είναι οι φορές που το μοντέλο προέβλεψε ότι υπάρχει αεροπλάνο ενώ στην πραγματικότητα δεν υπάρχει.

False Negatives: Αθροίζοντας τα στοιχεία κάθε γραμμής, με εξαίρεση αυτά που ανήκουν στη διαγώνιο, λαμβάνουμε τα false negatives της κάθε κλάσης. Για παράδειγμα, η πρώτη γραμμή Plane, έχει 585 ορθές προβλέψεις – το σημείου της διαγωνίου, εικόνα 19.34, και σε όλες τις άλλες περιπτώσεις το μοντέλο δεν προέβλεψε αεροπλάνο ενώ στην πραγματικότητα υπάρχει.



Εικόνα 19.34

Υπολογίζουμε το κριτήριο Recall για κάθε κλάση, εικόνα 19.36 και για ολόκληρο το μοντέλο είναι στο ύψους του 49.92%.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

```
recall = np.diag(conf_matrix) / np.sum(conf_matrix, axis=1)
recall
```

```
array([0.582, 0.677, 0.276, 0.266, 0.514, 0.431, 0.499, 0.61 , 0.604,
       0.533])
```

Εικόνα 19.35

```
# recall for the whole model
avg_recall = np.mean(recall)
print(f"Model 2 recall score is {avg_recall:.2%}")
```

```
Model 2 recall score is 49.92%
```

Εικόνα 19.36

Υπολογίζουμε το κριτήριο Precision για κάθε κλάση και για ολόκληρο το μοντέλο είναι στο ύψους του 49.76%, εικόνα 19.38.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Εικόνα 19.37

```
precision = np.diag(conf_matrix) / np.sum(conf_matrix, axis=0)
precision
```

```
array([0.548, 0.598, 0.418, 0.364, 0.384, 0.406, 0.568, 0.493, 0.636,
       0.563])
```

```
# precision for the whole model
avg_precision = np.mean(precision)
print(f"Model 2 precision score is {avg_precision:.2%}")
```

```
Model 2 precision score is 49.76%
```

Εικόνα 19.38

Υπολογίζουμε το κριτήριο F1 του μοντέλου που είναι στο ύψους του 49.92%, εικόνα 20.39.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

```
f1_score = 2 * (avg_precision * avg_recall) / (avg_precision + avg_recall)
print(f"Model 2 F1 Score: {f1_score:.2%}")
```

```
Model 2 F1 Score: 49.84%
```

Εικόνα 19.39

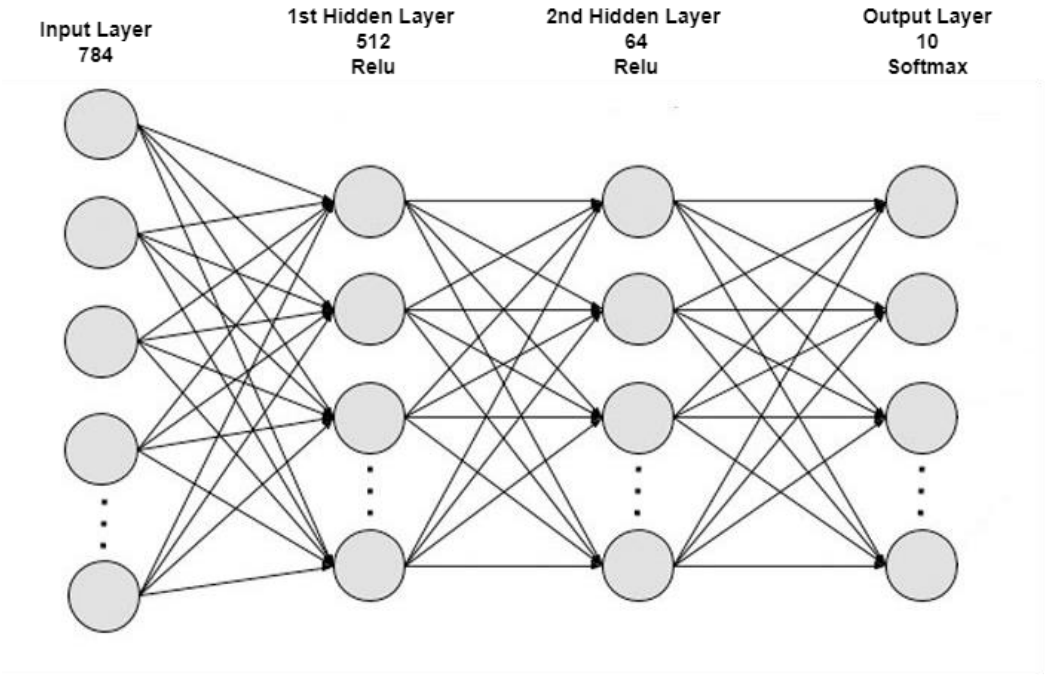
20. Δημιουργία Τεχνητού Νευρωνικού Δικτύου Multilayer Perceptron - MLP Ταξινόμησης Χειρόγραφων Ψηφίων

Στο κεφάλαιο γίνεται χρήση των παρακάτω αρχείων από το φάκελο 20 MLP MNIST:

- *20 MLP MNIST NN.ipynb*
- *Φάκελος MNIST, αρχεία:*
  - *Digit_xtest.csv*
  - *Digit_xtrain.csv*
  - *Digit_ytest.csv*
  - *Digit_ytrain.csv*
  - *Test_img.png*

Θα γίνει κατασκευή τεχνητού νευρωνικού δικτύου τύπου πολυεπίπεδου perceptron (multilayer perceptron) το οποίο θα έχει τη δυνατότητα να διακρίνει 10 διαφορετικά χειρόγραφα ψηφία (από το 0 έως το 9) σε εικόνες, θα έχει δηλαδή 10 διαφορετικές κλάσεις.

Συγκεκριμένα το τεχνητό νευρωνικό δίκτυο που θα κατασκευαστεί θα έχει 784 νευρώνες στο επίπεδο εισόδου, 512 νευρώνες στο 1ο κρυφό επίπεδο, 64 νευρώνες στο 2ο κρυφό επίπεδο, 16 νευρώνες και 10 νευρώνες στο επίπεδο εξόδου, εικόνα 20.1.



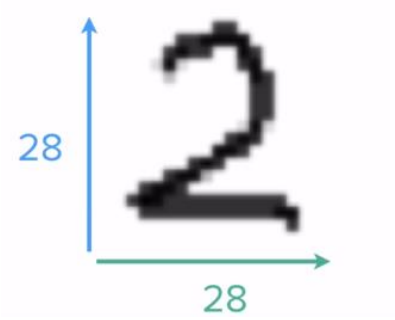
Εικόνα 20.1

20.1 Συλλογή Δεδομένων

Τα δεδομένα θα είναι εικόνες που θα απεικονίζουν χειρόγραφα ψηφία από το 0 έως το 9 και έχουν συλλεχθεί από τη βάση δεδομένων MNIST (http://yann.lecun.com/exdb/mnist/).

20.2 Εξερεύνηση Δεδομένων

Όλα τα δεδομένα είναι σε εικόνες οι οποίες έχουν λευκό, μαύρο και αποχρώσεις του γκρι. Κάθε εικόνα έχει 28 πίξελ στο πλάτος, 28 πίξελ στο ύψος και επειδή ο χρωματισμός είναι σε αποχρώσεις του γκρι, εικόνα 21.2, υπάρχει μόνο ένα κανάλι χρώματος (κάθε πίξελ έχει μια ακέραια τιμή από το 0 – εντελώς λευκό - έως το 255 – εντελώς μαύρο - που δείχνει πόσο φωτεινή ή σκοτεινή είναι η απόχρωση του γκρι που έχει). Αυτό συνεπάγεται πως οι είσοδοι – inputs για το μοντέλο θα είναι 28 x 28 x 1 = 784.



Εικόνα 20.2

Υπάρχουν 60.000 δεδομένα που προορίζονται για εκπαίδευση του μοντέλου (train data set). Το σύνολο x_train_all έχει τα δεδομένα που αφορούν τα χαρακτηριστικά – features που στην ουσία αφορούν τη γκρι απόχρωση του κάθε πίξελ της κάθε εικόνας. Για παράδειγμα η 1η εικόνα του x_train_all είναι ένας πίνακας 784 αριθμών από το 0 έως το 255. Παρατηρούμε πως τα δεδομένα έχουν επιπεδοποιηθεί – flattened (γεγονός το οποίο σημαίνει πως αντί για τις διαστάσεις 10.000 x 28 x 28 x 1 έχουμε τις διαστάσεις 10.000 x 1) για διευκόλυνση των υπολογισμών, εικόνα 21.3. Ωστόσο με αυτόν τον τρόπο χάνεται ένα μέρος της θεσιακής πληροφορίας (positional information) για το κάθε πίξελ, καθώς μετά την εφαρμογή της επιπεδοποίησης δεν μπορούμε να γνωρίζουμε ποια άλλα πίξελ γειτνιάζουν με το κάθε πίξελ.

```
# 60.000 Train Data, 28 x 28 x 1 = 784
x_train_all.shape
```

```
(60000, 784)
```

```
# Each example is an array of 784 int values 0 - 255
print(x_train_all[0].shape)
x_train_all[0]
```

```
(784,)

array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0   0   0   0   0   0   0   0   0   0   0   0   0
```

Εικόνα 20.3

Υπάρχουν επίσης 60.000 ετικέτες, y_train_all, κάθε μία από τις οποίες είναι ένας αριθμός από το 0 έως το 9, όσα δηλαδή και τα διαφορετικά ψηφία που απεικονίζονται σε κάθε εικόνα, εικόνα 20.4.

```
# 60.000 Train Labels
y_train_all.shape
```

```
(60000,)
```

```
# Labels of Digits from 0 to 9
y_train_all
```

```
array([5, 0, 4, ..., 5, 6, 8])
```

Εικόνα 20.4

Επίσης υπάρχουν και 10.000 δεδομένα που προορίζονται για δοκιμή του μοντέλου (test data set), χωρισμένα στα σύνολα των features (x_test) και labels (y_test), εικόνα 20.5.

```
# 10.000 Testing Data
x_test.shape
```

```
(10000, 784)
```

```
x_test
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
# 10.000 Testing Labels
y_test.shape
```

```
(10000,)
```

```
y_test
```

```
array([7, 2, 1, ..., 4, 5, 6])
```

Εικόνα 20.5

20.3 Επεξεργασία Δεδομένων

Οι ρυθμοί εκπαίδευσης των βελτιστοποιητών είναι πολύ μικροί αριθμοί και για αυτό το λόγο βοηθάει οι είσοδοι στο μοντέλο να έχουν τιμές μεταξύ του 0 και του 1. Συνεπώς, διαιρούμε τα σύνολα των χαρακτηριστικών με την τιμή 255, με αποτέλεσμα το απόλυτο λευκό θα συμβολίζεται με 0 και το απόλυτο μαύρο με 1, εικόνα 20.6.

```
# Re-scale features
x_train_all, x_test = x_train_all / 255.0, x_test / 255.0
```

Εικόνα 20.6

Μετατροπή των στόχων – labels σε μορφή one hot encoding. Παρατηρούμε πως ο 1ος στόχος του συνόλου y_train_all έχει την τιμή 5, το οποίο μετατρέπουμε σε μορφή 10 ψηφίων όπου όλα είναι 0 εκτός από το ψηφίο που αντιστοιχεί στη θέση που 5, δηλαδή: 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, εικόνα 20.7.

```
y_train_all[0]
```

```
5
```

```
y_train_all = np.eye(NR_CLASSES)[y_train_all]
y_train_all.shape
```

```
(60000, 10)
```

```
y_train_all
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.]])
```

```
y_train_all[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.])
```

Εικόνα 20.7

Η ίδια διαδικασία γίνεται και για το σύνολο των στόχων που προορίζονται για test (y_test), εικόνα 20.8.

```
# Convert y_test to One Hot Encoding
y_test = np.eye(NR_CLASSES)[y_test]
y_test.shape
```

```
(10000, 10)
```

```
y_test
```

```
array([[0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Εικόνα 20.8

Δημιουργούμε σύνολα επικύρωσης 10.000 δεδομένων για τα χαρακτηριστικά x και τους στόχους y, εικόνα 20.9.

```
# Create x, y Validation Sets of 10.000 data
x_val = x_train_all[:VALIDATION_SIZE]
y_val = y_train_all[:VALIDATION_SIZE]
```

```
print(x_val.shape)
print(y_val.shape)
```

```
(10000, 784)
(10000, 10)
```

```
# Update x, y Training Sets
x_train_all = x_train_all[VALIDATION_SIZE:]
y_train_all = y_train_all[VALIDATION_SIZE:]
```

```
print(x_train_all.shape)
print(y_train_all.shape)
```

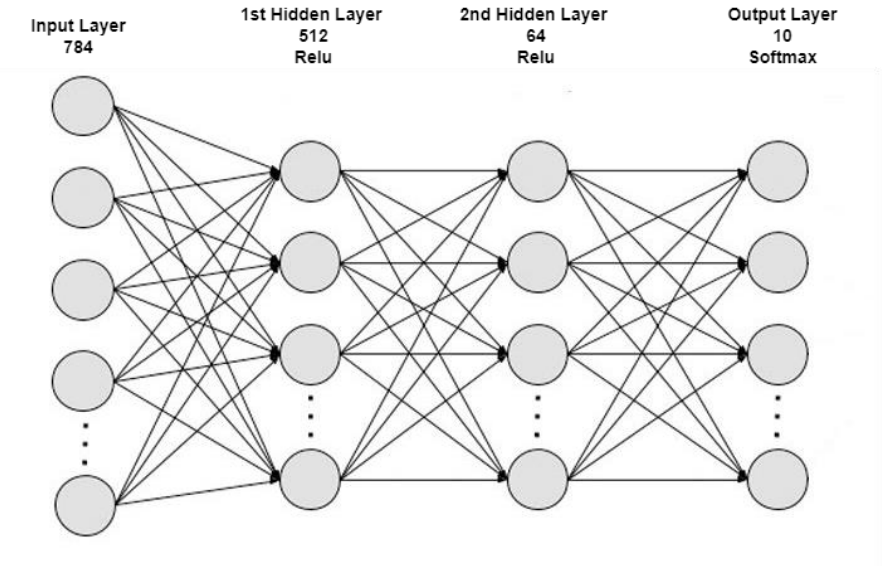
```
(50000, 784)
(50000, 10)
```

Εικόνα 20.9

20.4 Κατασκευή Νευρωνικού Δικτύου

Το τεχνητό νευρωνικό δίκτυο που θα κατασκευαστεί θα έχει τη μορφή:

- Επίπεδο εισόδου 784 νευρώνες
- 1ο κρυφό επίπεδο θα υπάρχουν 512 νευρώνες
- 2ο κρυφό επίπεδο 64, εικόνα XX.
- Στο επίπεδο εξόδου οι νευρώνες θα είναι όσες και οι κλάσεις ταξινόμησης δηλαδή 10.



Εικόνα 20.10

```
# 1st Hidden Layer Neurons
n_hidden1 = 512
# 2nd Hidden Layer Neurons
n_hidden2 = 64
```

Εικόνα 20.11

Η συνάρτηση setup_layer δημιουργεί τα επίπεδα του τεχνητού νευρωνικού δικτύου παίρνοντας ως παραμέτρους την είσοδο του κάθε επιπέδου, το αριθμό των βαρών που καθορίζεται από τις εισόδους σε κάθε επίπεδο επί τον αριθμό των νευρώνων του επιπέδου, το αριθμό των bias που αντιστοιχεί ένα σε κάθε νευρώνα, κατά συνέπεια θα είναι όσοι και οι νευρώνες του κάθε επιπέδου και το όνομα του επιπέδου.

Αν το επίπεδο είναι κρυφό τότε η συνάρτηση ενεργοποίησης καθορίζεται η Relu, ενώ αν το επίπεδο είναι εξόδου τότε η συνάρτησης ενεργοποίησης είναι η SoftMax, εικόνα 20.12. Η συνάρτηση επιστρέφει την έξοδο του κάθε επιπέδου.

```
def setup_layer(input, weight_dim, bias_dim, name):

    with tf.name_scope(name):
        # Create Weights for Layer (shape: inputs X neurons) using Truncated Normal Distribution
        initial_w = tf.truncated_normal(shape=weight_dim, stddev=0.1, seed=42)
        # Create Tensorfloat Variable holding all Weights of Layer
        w = tf.Variable(initial_value=initial_w, name='W')

        # Initialise Biases of Layer - All Start with Same Value 0.0
        initial_b = tf.constant(value=0.0, shape=bias_dim)
        b = tf.Variable(initial_value=initial_b, name='B')

        # Store Inputs Coming to Layer
        layer_in = tf.matmul(input, w) + b

        if name=='out':
            # Softmax Activation Function for Output Layer
            layer_out = tf.nn.softmax(layer_in)
        else:
            # Relu for Other Layers
            layer_out = tf.nn.relu(layer_in)

        tf.summary.histogram('weights', w)
        tf.summary.histogram('biases', b)

        return layer_out
```

Εικόνα 20.12

Κατασκευάζουμε το 1^{ο} και 2^{ο} κρυφό επίπεδο καθώς και το επίπεδο εξόδου, εικόνα 20.13:

```
layer_1 = setup_layer(X, weight_dim=[TOTAL_INPUTS, n_hidden1],
                      bias_dim=[n_hidden1], name='layer_1')

layer_2 = setup_layer(layer_1, weight_dim=[n_hidden1, n_hidden2],
                      bias_dim=[n_hidden2], name='layer_2')

output = setup_layer(layer_2, weight_dim=[n_hidden2, NR_CLASSES],
                      bias_dim=[NR_CLASSES], name='out')
```

Εικόνα 20.13

Η loss function που χρησιμοποιείται είναι η cross-entropy, εικόνα 20.14

```
with tf.name_scope('loss_calc'):
    # Logits: Outputs from Output Layer
    # Use the Avg due to Batchess
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y, logits=output))
```

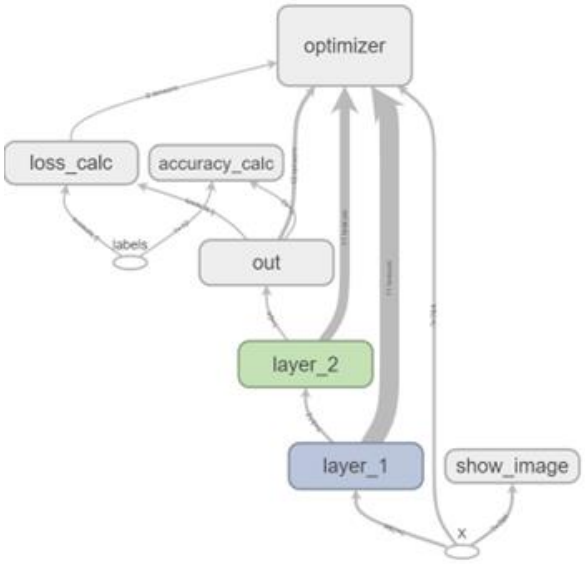
Εικόνα 20.14

Ο optimiser που χρησιμοποιείται είναι ο Adam στον οποίο καθορίζουμε εμείς το learning rate που επιθυμούμε κάθε φορά να χρησιμοποιήσουμε, εικόνα 20.15:

```
# Adam Optimiser
with tf.name_scope('optimizer'):
    # Use my Learning Rate
    optimizer = tf.train.AdamOptimizer(learning_rate)
    # Minimise Loss Calcuated from Cross Enotropy
    train_step = optimizer.minimize(loss)
```

Εικόνα 20.15

Μέσα από το εργαλείο tensorboard του tensorflow, μπορούμε να δούμε μια σχηματική περιγραφή του μοντέλου που κατασκευάσαμε, εικόνα 20.16



Εικόνα 20.16

Καθορίζουμε το μέγεθος του batch στα 1000 δεδομένα και δημιουργούμε μια συνάρτηση που να μας επιστρέφει τα δεδομένα (features & labels) του κάθε batch, εικόνα20.17 :

```
# Going from One Batch to Next
# data: features X, labels: Y
def next_batch(batch_size, data, labels):

    global num_examples
    global index_in_epoch

    start = index_in_epoch
    index_in_epoch += batch_size

    # Start Next Epoch
    if index_in_epoch > num_examples:
        start = 0
        index_in_epoch = batch_size

    # End of Bact
    end = index_in_epoch

    return data[start:end], labels[start:end]
```

Εικόνα 20.17

20.5 Εκπαίδευση Μοντέλου

Προχωρούμε στην εκπαίδευση του μοντέλου μέσα από μια επαναληπτική δομή for η οποία θα εξετάσει τα 50.000 δεδομένα των training set (μέσα από τις iterations των batch) για συγκεκριμένο αριθμό εποχών καθορισμένο από εμάς. Τα validation data, που είναι λιγότερα (συνολικά 10.000) δε θα χωριστούν σε batches, εικόνα 20.18

```
# Run Through All Epochs
for epoch in range(nr_epochs):

    # ============= Training Dataset =========
    # Iterate Through Data
    for i in range(nr_iterations):

        batch_x, batch_y = next_batch(batch_size=size_of_batch, data=x_train, labels=y_train)
        # Create Dictionary to Feed it to TensorFlow Session
        feed_dictionary = {X:batch_x, Y:batch_y}
        # Session Runs All Calculations - Training Step
        sess.run(train_step, feed_dict=feed_dictionary)

    # Accuracy Calculated on Batct - feed dictionary
    s, batch_accuracy = sess.run(fetches=[merged_summary, accuracy], feed_dict=feed_dictionary)

    train_writer.add_summary(s, epoch)

    print(f'Epoch {epoch} \t| Training Accuracy = {batch_accuracy}')

    # ================= Validation ====================
    # Not in Batches
    summary = sess.run(fetches=merged_summary, feed_dict={X:x_val, Y:y_val})
    validation_writer.add_summary(summary, epoch)

print('Done training!')
```

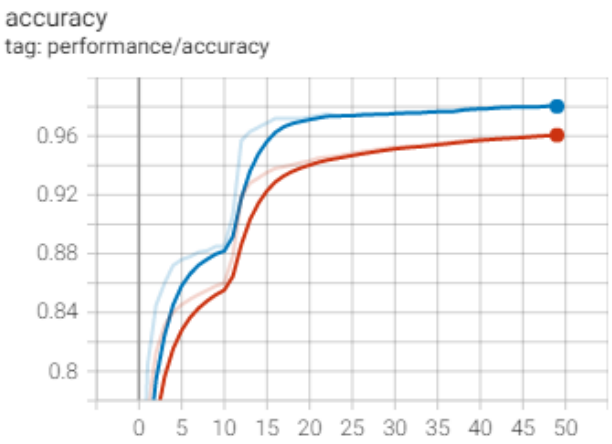
```
Epoch 0         | Training Accuracy = 0.8619999885559082
Epoch 1         | Training Accuracy = 0.8809999823570251
Epoch 2         | Training Accuracy = 0.9110000133514404
```

Εικόνα 20.18

Εκπαίδευση με learning rate 0.0001

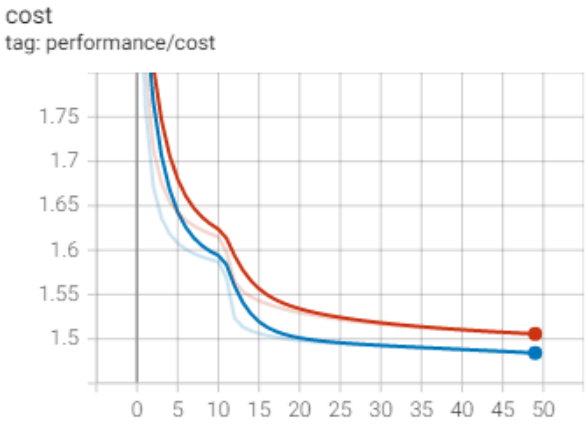
Εκπαιδεύουμε το τεχνητό νευρωνικό δίκτυο που δημιουργήσαμε με τα training data καθώς και με τα validation data, για 50 εποχές, με learning rate 0.0001. Σύμφωνα με το κριτήριο αξιολόγησης accuracy λαμβάνουμε την εικόνα 20.19.

Η γαλάζια γραμμή ανήκει στα training data και η κόκκινη στα validation. Παρατηρούμε πως το μεγαλύτερο μέρος της εκπαίδευσης λαμβάνει χώρα μέχρι την εποχή 20 και πως η εκπαίδευση στα validation data καθυστερεί σε σχέση με τα training data. Μετά την $20^{η}$ εποχή, η επίτευξη μεγαλύτερης ακρίβειας γίνεται με πολύ αργότερο ρυθμό, τα training data φθάνουν στο επίπεδο accuracy 0.98 και τα validation data στο επίπεδο 0.9613, εικόνα 20.19



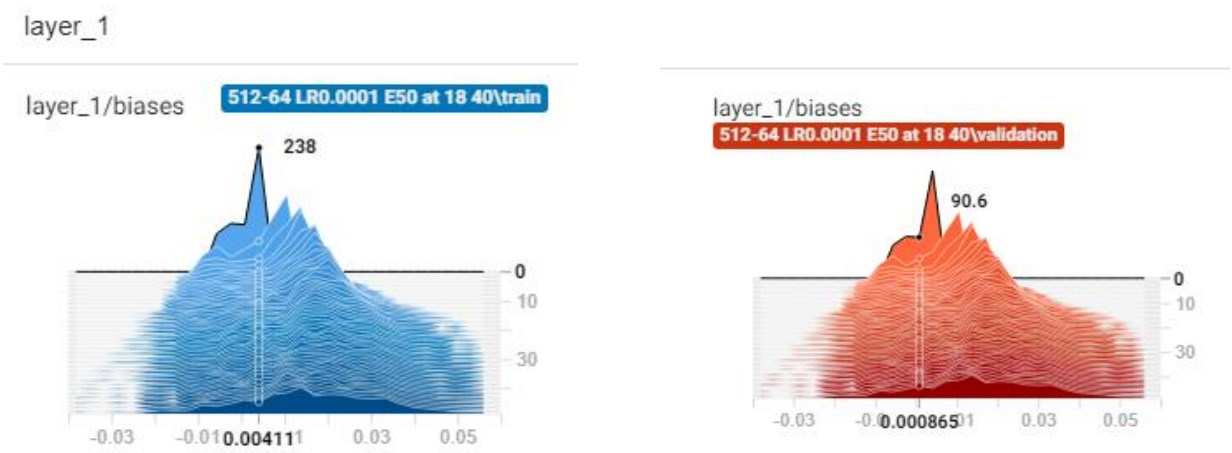
Εικόνα 20.19

Σχετικά με τις απώλειες, αυτές μειώνονται αρκετά μέχρι την εποχή 20 και μετά με αργότερους ρυθμούς, εικόνα 20.20



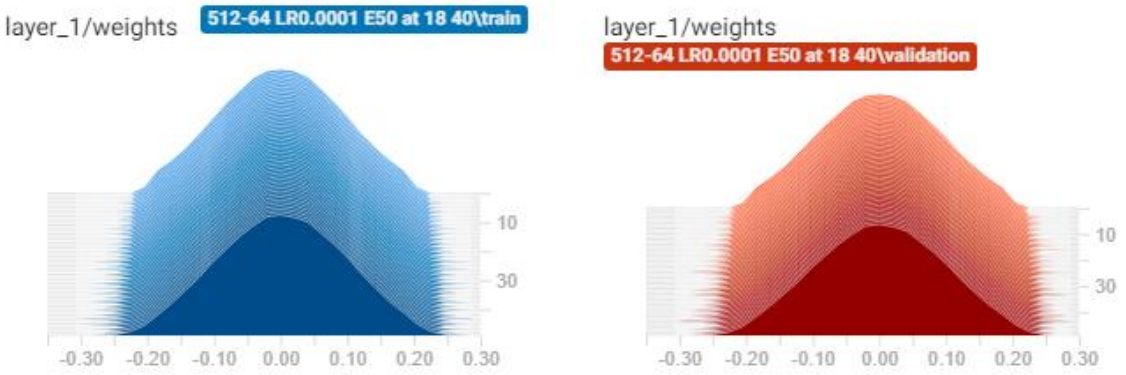
Εικόνα 20.20

Σχετικά με τις τιμές των biases στο σχεδιάγραμμα της εικόνας 20.21, με το γαλάζιο χρώμα έχουμε τα training data και με το κόκκινο χρώμα τα validation data. Τα biases αρχικοποιούνται όλα στο 0.0. Παρατηρούμε πως στο τέλος της 1ης εποχής (εποχή 0) 238 biases των training data έχουν ανανεωθεί παίρνοντας την τιμή 0.00411 και 91 biases των validation data έλαβαν την τιμή 0.000865. Η εξέλιξή τους στις 50 εποχές, μέσω των οποίων είχαν έντονες διαφοροποιήσεις στις τιμές τους, φαίνεται στην εικόνα 20.21



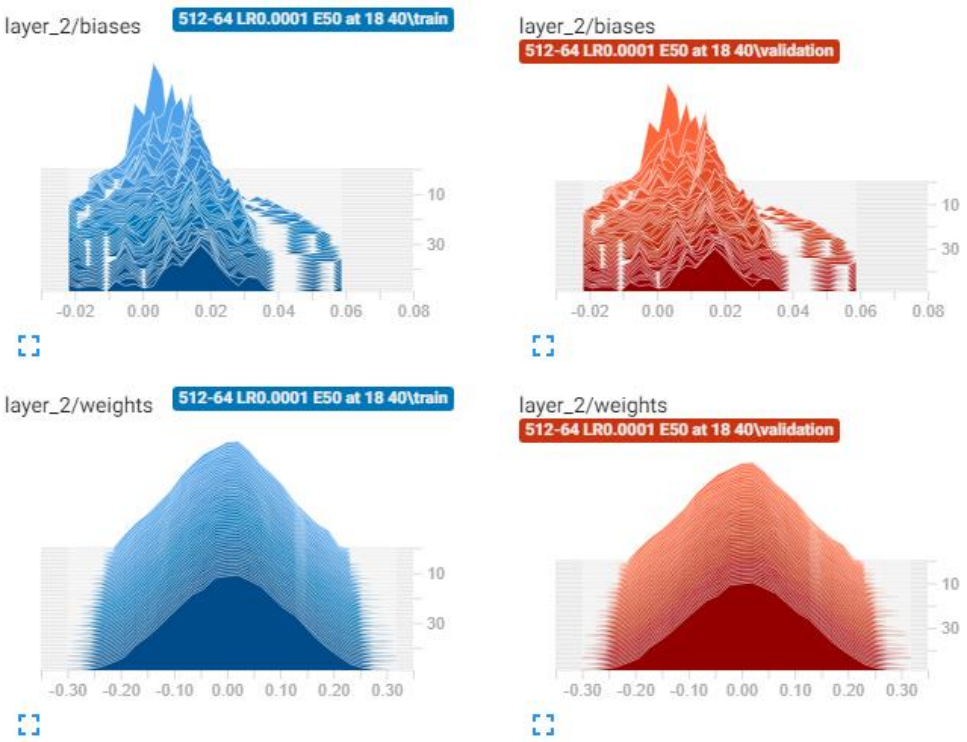
Εικόνα 20.21

Τα βάρη αρχικοποιήθηκαν μέσα από την truncated normal distribution και αυτή η κατανομή φαίνεται να τηρήθηκε σε γενικές γραμμές και στις 50 εποχές, γεγονός το οποίο μας ωθεί να σκεφτούμε πως το μεγαλύτερο μέρος της εκπαίδευσης έγινε στις τιμές biases, εικόνα 20.22



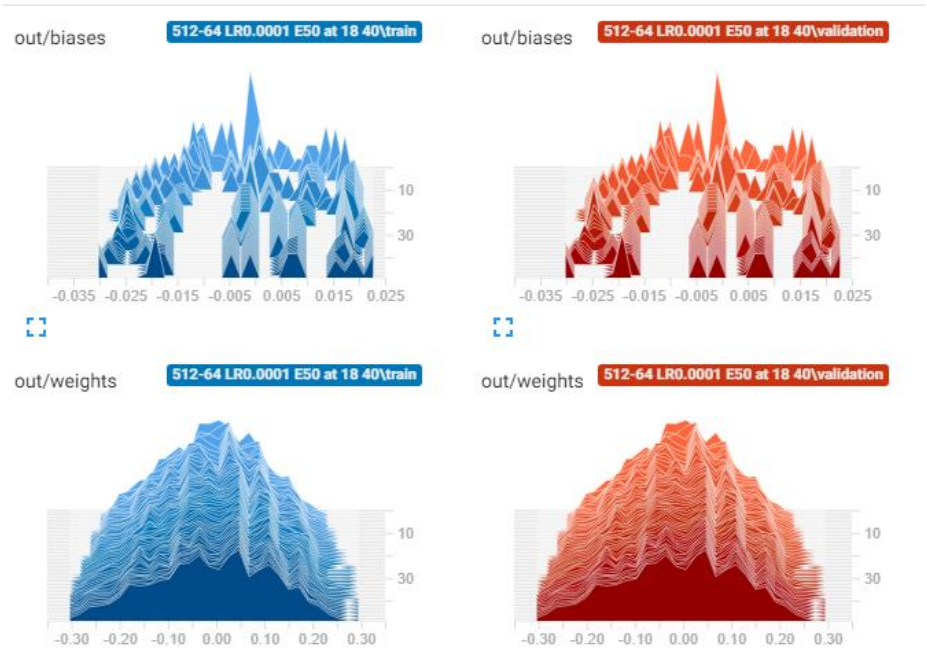
Εικόνα 20.22

Παρόμοια μοτίβα υπήρξαν και στο 2^{ο} κρυφό επίπεδο, εικόνα 20.23



Εικόνα 20.23

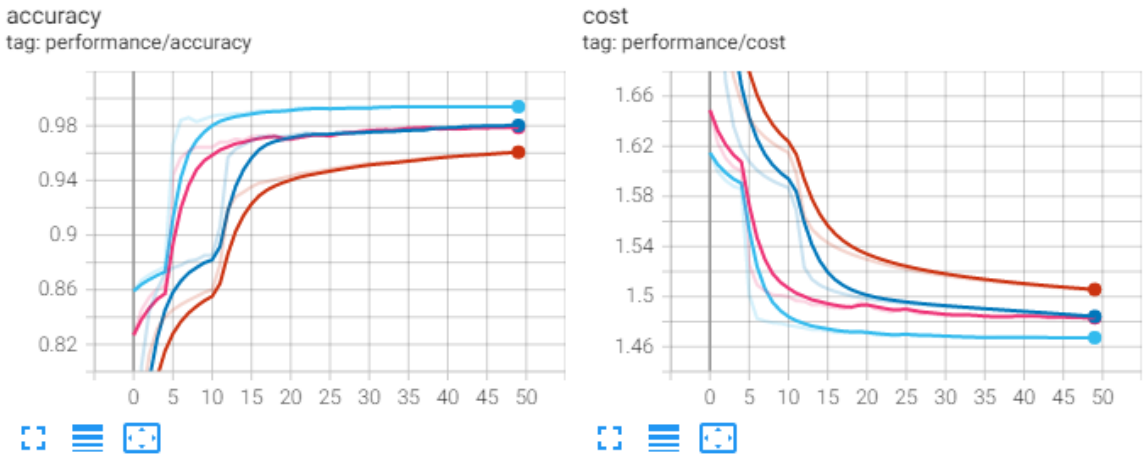
Σημαντικές διαφοροποιήσεις έλαβαν χώρα τόσο στα 10 biases όσο και στα βάρη του output layer, εικόνα 20.24



Εικόνα 20.24

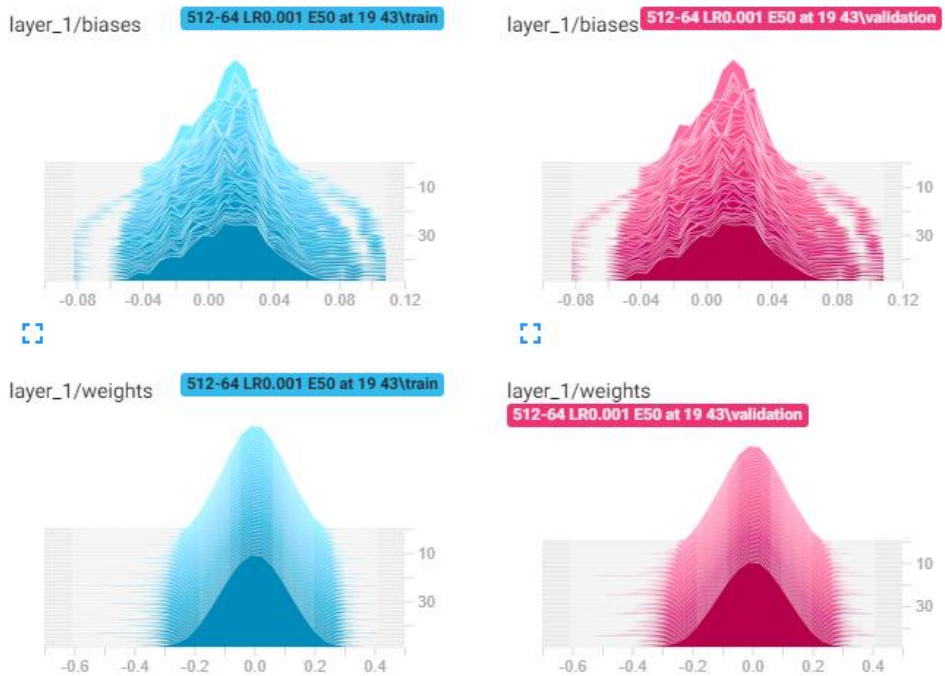
Εκπαίδευση με Learning Rate = 0.001

Αν αυξήσουμε το learning rate 10 φορές (από 0.0001 σε 0.001) παρατηρούμε πως επιτυγχάνεται μεγαλύτερη accuracy και γρηγορότερα (ανοικτή γαλάζια γραμμή – training data set, ροζ γραμμή – validation data set και learning rate 0.001, σκούρα γαλάζια γραμμή – training data set, κόκκινη γραμμή – validation data set με learning rate 0.001), εικόνα 20.25

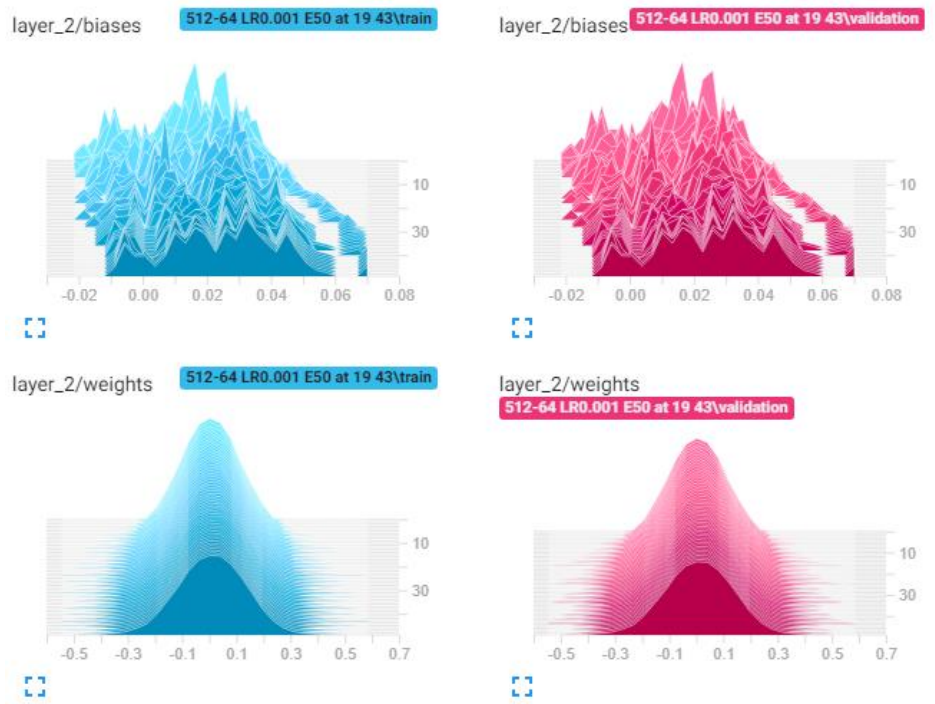


Εικόνα 21.25

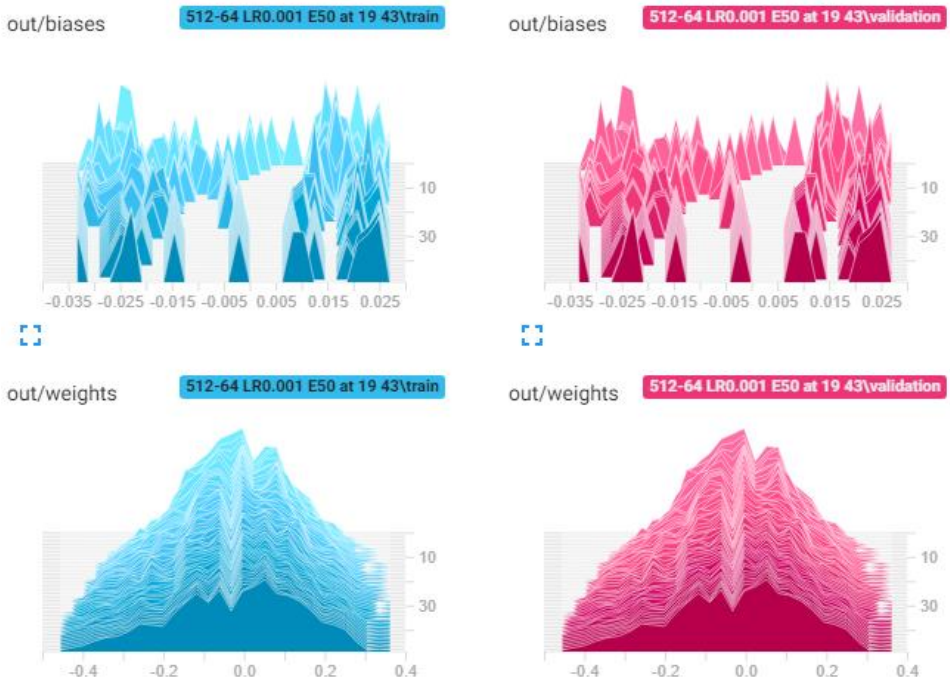
Επίσης οι διαφοροποιήσεις στις τιμές των βαρών και των biases είναι πιο έντονες όταν εφαρμόζουμε learning rate 0.001 σε σχέση με το 0.0001, εικόνες 20.26, 20.27, 20.28.



Εικόνα 20.26



Εικόνα 20.27



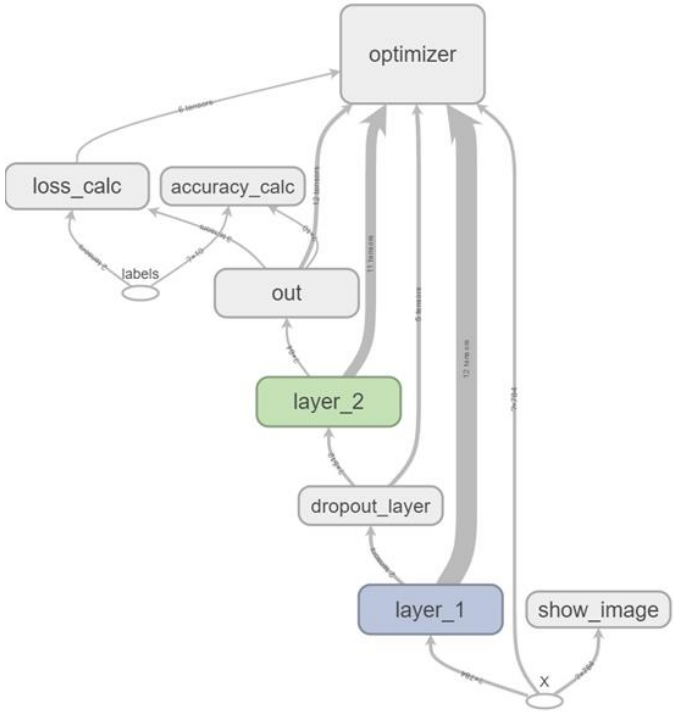
Εικόνα 20.28

Έως τώρα δεν έχουν παρατηρηθεί ισχυρές ενδείξεις overfitting στο μοντέλο (είτε με learning rate 0.0001 είτε με 0.001), ωστόσο μπορούμε να εφαρμόσουμε την τεχνική του dropout στο 1ο κρυφό επίπεδο με ποσοστό ενεργών νευρώνων 80% και να δούμε τα αποτελέσματα.

```
# Model Wth Dropout
layer_1 = setup_layer(X, weight_dim=[TOTAL_INPUTS, n_hidden1],
                      bias_dim=[n_hidden1], name='layer_1')

layer_drop = tf.nn.dropout(layer_1, keep_prob=0.8, name='dropout_layer')

layer_2 = setup_layer(layer_drop, weight_dim=[n_hidden1, n_hidden2],
                      bias_dim=[n_hidden2], name='layer_2')

output = setup_layer(layer_2, weight_dim=[n_hidden2, NR_CLASSES],
                      bias_dim=[NR_CLASSES], name='out')

model_name = f'{n_hidden1}-DO-{n_hidden2} LR{learning_rate} E{nr_epochs}'
```

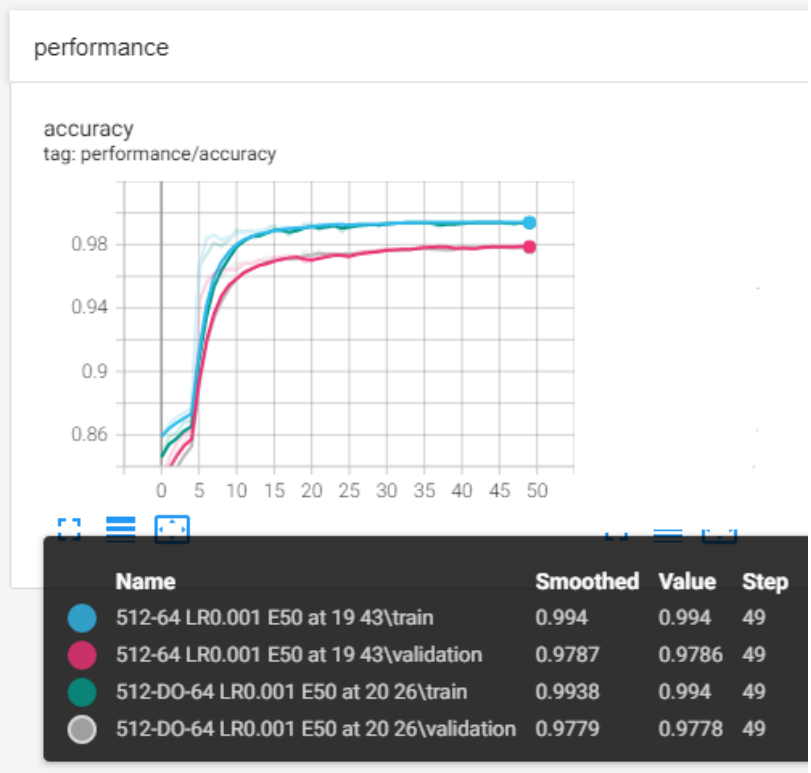
Εικόνα 20.29

Η σχηματική περιγραφή του μοντέλου με το dropout φαίνεται στην εικόνα 20.30



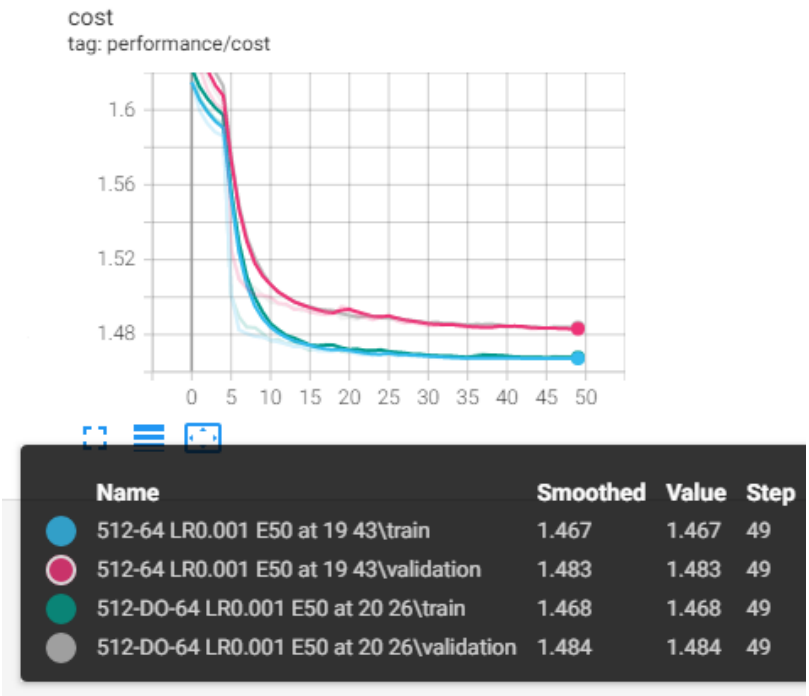
Εικόνα 21.30

Συγκρίνοντας το μοντέλο χωρίς dropout και με dropout (learning rate = 0.001 και στις 2 περιπτώσεις) μπορούμε να δούμε πως και τις 2 φορές το μοντέλο φθάνει σχεδόν στην ίδια accuracy, στην περίπτωση όμως του dropout καθυστερεί, (γαλάζια και ροζ γραμμή – training και validation χωρίς dropout, πράσινη και γκρι γραμμή – training και validation χωρίς με dropout) εικόνα 20.31.



Εικόνα 20.31

Σχετικά με τις απώλειες, έντονο φαινόμενο overfitting δεν παρατηρήθηκε ούτε χωρίς ούτε με το dropout, εικόνα 20.32.



Εικόνα 20.32

20.6 Αξιολόγηση Μοντέλου

Μπορούμε να χρησιμοποιήσουμε μια εικόνα, η οποία θα έχει τον χειρόγραφο αριθμό 2, ώστε να πραγματοποιήσουμε μια πρόβλεψη με το μοντέλο μας, εικόνα 20.33

```
img = Image.open('MNIST/test_img.png')
img
```

Εικόνα 20.33

Κάνοντας χρήση της τελευταίας έκδοσης του μοντέλου (με dropout και learning rate 0.001), το μοντέλο προβλέπει ότι ο αριθμός 2 απεικονίζεται στην εικόνα, εικόνα 20.34

```
prediction = sess.run(fetches=tf.argmax(output, axis=1), feed_dict={X:[test_img]})
```

```
print(f'Prediction for test image is {prediction}')
```

```
Prediction for test image is [2]
```

Εικόνα 20.34

Εφαρμόζοντας το σύνολο των δεδομένων για δοκιμή στο μοντέλο με dropout και learning rate 0.001, test data set, λαμβάνουμε ακρίβεια 97.93%, εικόνα 20.35.

```
test_accuracy = sess.run(fetches=accuracy, feed_dict={X:x_test, Y:y_test})
print(f'Accuracy on test set is {test_accuracy:0.2%}')
```

```
Accuracy on test set is 97.93%
```

Εικόνα 20.35

Τεχνικά νευρωνικά δίκτυα τύπου Multilayer Perceptron – MLP, μπορούν να πετύχουμε και μεγαλύτερη ακρίβεια στο θέμα της αναγνώρισης χειρόγραφων ψηφίων. Το 2010, κατασκευάστηκε MLP το οποίο έχοντας 6 επίπεδα με 2500, 2000, 1500, 1000, 500, 10 νευρώνες σε κάθε επίπεδο αντίστοιχα, πέτυχε απώλεια μόλις 0.35% (Cirecan et al., 2010).

21. Δημιουργία Convolutional Neural Network CNN Αναγνώρισης Εικόνων

Στο κεφάλαιο γίνεται χρήση του παρακάτω αρχείου:

- *21 CNN.ipynb*

21.1 Συλλογή, Επεξεργασία και Εξερεύνηση Δεδομένων

Θα γίνει χρήση των εικόνων του συνόλου δεδομένων CIFAR 10.

Συλλέγουμε τα δεδομένα από το dataset CIFAR 10 και μετατρέπουμε κάθε ένα από αυτά σε μια τιμή από το 0 έως το 1 διαιρώντας τα με το 255.

Παρατηρούμε πως το σύνολο των δεδομένων των χαρακτηριστικών x_train προς εκπαίδευση είναι 50.000, διαστάσεων 32 πίξελ πλάτους επί 32 πίξελ ύψους επί 3 χρωματικά κανάλια.

Τα σύνολα των δεδομένων για εκπαίδευση του μοντέλου, x_test – y_test, έχουν 10.000 δεδομένα και συνολικά υπάρχουν 10 κλάσεις στις οποίες κατατάσσονται οι εικόνες του CIFAR 10. Αυτές είναι οι airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

```
# Load Data
cifar10 = tf.keras.datasets.cifar10

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train, y_test = y_train.flatten(), y_test.flatten()
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print(y_train[0])
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)
```

```
x_train.shape:  (50000, 32, 32, 3)
y_train.shape:  (50000,)
6
x_test.shape:  (10000, 32, 32, 3)
y_test.shape:  (10000,)
```

```
# Number of Classes
K = len(set(y_train))
print("number of classes:", K)
```

```
number of classes: 10
```

Εικόνα 21.1

21.2 Κατασκευή Μοντέλου

Στο νευρωνικό δίκτυο που κατασκευάζεται υπάρχουν συνελικτικά επίπεδα μαζί με κανονικοποίηση κατά batch – batch normalisation και γίνεται χρήση της τεχνικής dropout πριν από κάθε dense επίπεδο, εικόνα 21.2.

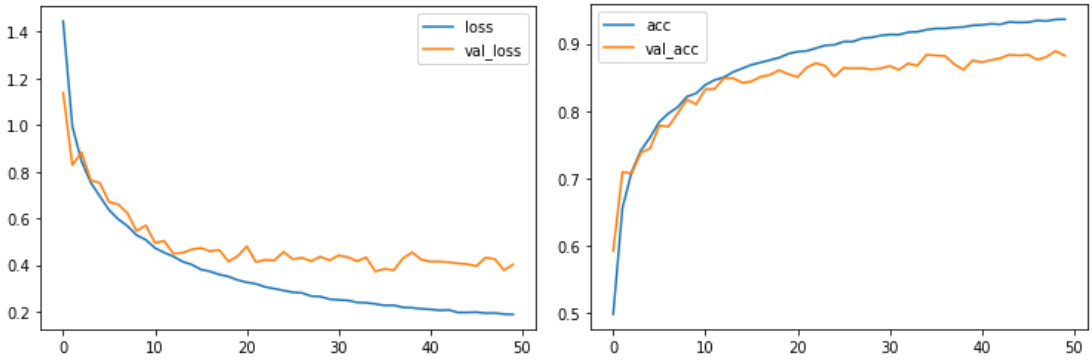
```
# Build Model
i = Input(shape=x_train[0].shape)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(i)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
x = Dense(K, activation='softmax')(x)

model = Model(i, x)
```

Εικόνα 21.2

21.3 Εκπαίδευση Μοντέλου

Εκπαιδεύουμε το μοντέλο για 50 εποχές (σε batch size 32) και λαμβάνουμε τα παρακάτω αποτελέσματα, όπου στην τελευταία εποχή η απώλεια των δεδομένων εκπαίδευσης είναι στο 0.1880 και των δεδομένων δοκιμής ελαφρώς μεγαλύτερη στο 0.4024, ενώ η ακρίβεια των δεδομένων εκπαίδευσης είναι στο 0.9363 και των δεδομένων δοκιμής ελαφρώς μικρότερη στο 0.883, εικόνα 21.3.

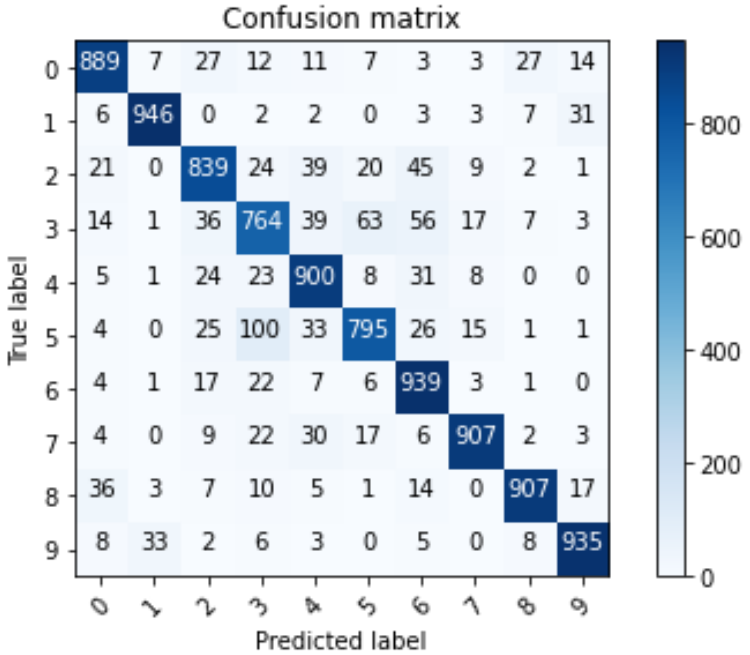
```
Epoch 50/50
1562/1562   - 35s 23ms/step - loss: 0.1880 - accuracy: 0.9363 - val_loss: 0.4024 - val_accuracy: 0.8821
```



Εικόνα 21.3

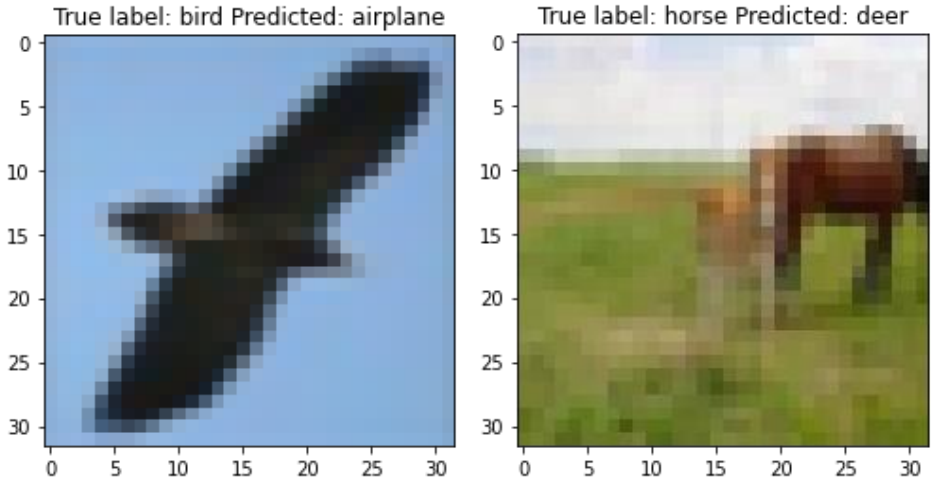
21.4 Αξιολόγηση Μοντέλου

Στη μήτρα σύγχυσης παρατηρούμε πως υπάρχουν πολλές λάθος προβλέψεις όταν η κλάση είναι 5 (σκύλος) και προβλέπεται 3 (γάτα), συνολικά 100 τέτοιες λάθος προβλέψεις, αλλά και το αντίστροφο, όταν δηλαδή η κλάση είναι 3 (γάτα) και προβλέπεται 5 (σκύλος), συνολικά 63 τέτοιες λάθος προβλέψεις. Με δεδομένο το γεγονός ότι οι εικόνες είναι χαμηλής ανάλυσης, μπορούμε να ισχυριστούμε πως το μοντέλο έχει μεγάλη ακρίβεια και τα λάθη του προσομοιάζουν αυτά που θα έκανε και ένας άνθρωπος.



Εικόνα 21.4

Βλέποντας τα αποτελέσματα από κάποιες λανθασμένα ταξινομημένες εικόνες μπορούμε να επισημάνουμε πως πρόκειται για λάθη που θα μπορούσε να κάνει και ένας άνθρωπος. Στην εικόνα 21.5 το πουλί μοιάζει με αεροπλάνο και το άλογο μοιάζει με τάρανδο.



Εικόνα 21.5

22. Εφαρμογή προ-Εκπαιδευμένων Τεχνητών Νευρωνικών Δικτύων Convolutional Neural Networks - CNN Αναγνώρισης Εικόνων

Στο κεφάλαιο γίνεται χρήση των παρακάτω αρχείων από το φάκελο 22 Pre Trained CNN:

- *22 Pre_Trained_NN.ipynb*
- *Pre-Trained NN Images*

Θα γίνει χρήση προ-εκπαιδευμένου τεχνητού νευρωνικού δικτύου το οποίο θα κληθεί να αναγνωρίσει τι απεικονίζεται σε εικόνες με τις οποίες θα το τροφοδοτήσουμε.

Η εφαρμογή θα πραγματοποιηθεί με τη βοήθεια του περιβάλλοντος google colaboratory (αρχείο 15 Pre_Trained_NN.ipynb).
Οι 11 φωτογραφίες – δείγματα που χρησιμοποιούνται στη συγκεκριμένη εφαρμογή προέρχονται από την ιστοσελίδα www.unsplash.com, σε ανάλυση 256 επί 256 πίξελ (φάκελος Pre-Trained NN Images).

Φόρτωση και προβολή 1ης φωτογραφίας από το δείγμα, εικόνα 22.1

```
# pull up 1st image
pic = load_img(FILE_1)
# display 1st image
display(pic)
```

Εικόνα 22.1

Δημιουργία πίνακα 3 διαστάσεων (256 πίξελ ύψος – 256 πίξελ πλάτους – 3 τιμές για χρώματα RGB) με τα πίξελ της φωτογραφίας:

```
# image -> array
pic_array = img_to_array(pic)
pic_array
```

```
array([[[115., 124., 131.],
        [133., 140., 150.],
        [147., 151., 163.],
        ...,
        [231., 237., 249.],
        [230., 238., 249.],
        [229., 239., 249.]],

       [[118., 127., 136.],
        [133., 140., 150.],
        [145., 149., 161.],
        ...,
        [230., 238., 249.],
        [230., 238., 249.],
        [230., 238., 249.]],

       [[117., 126., 135.],
        [134., 141., 151.],
        [146., 150., 161.],
        ...,
        [230., 238., 249.],
        [230., 238., 249.],
```

```
pic_array.shape
```

```
(256, 256, 3)
```

Εικόνα 22.2

22.1 Μοντέλο InceptionResNetV2

Για το μέρος του νευρωνικού δικτύου θα γίνει χρήση του ανοικτού λογισμικού Keras το οποίο παρέχει μια διεπαφή σε γλώσσα προγραμματισμού Python για τεχνητά νευρωνικά δίκτυα. Πιο συγκεκριμένα θα γίνει χρήση του προ-εκπαιδευμένου μοντέλου (τα βάρη είναι ήδη καθορισμένα) InceptionResNetV2, το οποίο ανήκει στην κατηγορία των Convolutional Neural Networks – CNNs η οποία είναι ιδιαίτερα αποδοτική για θέματα κατηγοριοποίησης εικόνων (He et al., 2015).

Τα βάρη του μοντέλου έχουν ήδη εκπαιδευτεί από το ImageNet, μια τεράστια και υψηλής ποιότητας βάση δεδομένων εικόνων για εκπαίδευση μεγάλης κλίμακας μοντέλων αναγνώρισης αντικειμένων (https://www.image-net.org/) και υποστηρίζεται κυρίως από τα πανεπιστήμια Princeton και Stanford των ΗΠΑ.

```
%%time
inception_model = InceptionResNetV2(weights='imagenet')
```

Εικόνα 22.3

Μετασχηματίζουμε την εικόνα ώστε να είναι συμβατή με τη μέθοδο predict() του μοντέλου, εικόνα 22.5:

```
# change picture size to 299x299
pic = load_img(FILE_1, target_size=(299, 299))
display(pic)
```

```
# image -> array
pic_array = img_to_array(pic)
pic_array.shape
```

```
(299, 299, 3)
```

Εικόνα 22.4

```
# image -> array
pic_array = img_to_array(pic)
pic_array.shape
```

```
(299, 299, 3)
```

```
# expand picture dimensions
expanded = np.expand_dims(pic_array, axis=0)
expanded.shape
```

```
(1, 299, 299, 3)
```

```
preprocessed = preprocess_input(expanded)
```

Εικόνα 22.5

Κάνουμε πρόβλεψη για μία φωτογραφία

```
prediction = inception_model.predict(preprocessed)
prediction
```

```
1/1 [==============================] - 0s 493ms/step
array([[1.81810858e-04, 1.20879748e-04, 9.83503851e-05, 1.30786706e-04,
        1.57059345e-04, 1.80344563e-04, 2.30266174e-04, 1.54684967e-04,
        1.02325481e-04, 1.56910231e-04, 6.77406206e-05, 1.16771254e-04,
        1.45441329e-04, 1.14305061e-04, 2.03759904e-04, 1.00941899e-04,
```

Εικόνα 22.6

Αποκωδικοποιώντας την πρόβλεψη λαμβάνουμε τις κορυφαίες 5 προβλέψεις του μοντέλου για την εικόνα. Η πρώτη πρόβλεψη του μοντέλου, εικόνα 22.7, είναι πως η φωτογραφία έχει ένα αλεξιβρόχιο – umbrella με πιθανότητα 0.829, ορειβατική σκηνή – mountain tent με πιθανότητα 0.0012, αδιάβροχο – trench coat με πιθανότητα 0.0011, τέμενος – mosque με πιθανότητα 0.0010 και τρούλο – dome με πιθανότητα 0.0010. Παρατηρούμε πως η πρόβλεψη είναι πολύ κοντά στα στοιχεία που απεικονίζονται στη φωτογραφία.

```
decode_predictions(prediction)
```

```
[[('n04507155', 'umbrella', 0.8292437),
  ('n03792972', 'mountain_tent', 0.0012268261),
  ('n04479046', 'trench_coat', 0.0011062953),
  ('n03788195', 'mosque', 0.0010572356),
  ('n03220513', 'dome', 0.001031099)]]
```

Εικόνα 22.7

Μπορούμε να συνοψίσουμε όλα τα παραπάνω βήματα σε μία συνάρτηση και να τα εφαρμόσουμε στη 2η φωτογραφία του δείγματός μας:

```
def format_img_inceptionresnet(filename):
  pic = load_img(filename, target_size=(299, 299))
  pic_arr = img_to_array(pic)
  expanded = np.expand_dims(pic_arr, axis=0)

  return preprocess_input(expanded)
```

Εικόνα 22.8

Η πρόβλεψη για τη 2η φωτογραφία, εικόνα 22.9, πάλι είναι κοντά στην πραγματικότητα καθώς προβλέψει την απεικόνιση γαμπρού - groom με πιθανότητα 0.704, νυφικού – gown με πιθανότητα 0.11, καρπού ελαιοκράμβης – rapeseed με πιθανότητα 0.016 μάλλον λόγω των φυτών που υπάρχουν στη φωτογραφία, αμμόλοφων – sandbar με πιθανότητα 0.012 και φούστα με στεφάνι - hoopskirt με πιθανότητα 0.007.

```
data = format_img_inceptionresnet(FILE_2)
prediction = inception_model.predict(data)
display(load_img(FILE_2))
decode_predictions(prediction)
```

```
1/1 [==============================] - 1s 753ms/step
```

```
[[('n10148035', 'groom', 0.70423746),
  ('n03450230', 'gown', 0.1166962),
  ('n11879895', 'rapeseed', 0.016582735),
  ('n09421951', 'sandbar', 0.012790689),
  ('n03534580', 'hoopskirt', 0.0072878003)]]
```

Εικόνα 22.9

22.2 Μοντέλο Virtual Geometry Group 19 – VGG19

Στη συνέχεια εφαρμόζουμε ένα άλλο μοντέλο στην κατηγορία των convolutional neural networks – CNN, που προσφέρεται από το Keras και σχετίζεται με την αναγνώριση εικόνας (image recognition), το Virtual Geometry Group 19 – VGG19. Τα βάρη και αυτού του μοντέλου έχουν προ-εκπαιδευτεί από την ImageNet.

Φορτώνουμε το μοντέλο VGG19

```
vgg19_model = VGG19()
```

Εικόνα 22.10

Δημιουργούμε μια συνάρτηση προ-επεξεργασίας εικόνων ώστε να μπορούν να αναγνωριστούν και να επεξεργαστούν στη συνέχεια από το μοντέλο VGG19, εικόνα 22.11.

```
def format_img_vgg19(filename):
  pic = load_img(filename, target_size=(224, 224))
  pic_arr = img_to_array(pic)
  expanded = np.expand_dims(pic_arr, axis=0)

  return preprocess_input_vgg19(expanded)
```

Εικόνα 22.11

Στη συνέχεια κάνουμε πρόβλεψη για την 3η φωτογραφία, εικόνα 22.12, και το 1ο αποτέλεσμα που λαμβάνουμε είναι ότι πρόκειται για ένα υποβρύχιο - submarine. Στη φωτογραφία δεν υπάρχει υποβρύχιο, ωστόσο τα σχήματα της φωτογραφίας είναι τέτοια που θυμίζουν υποβρύχιο. Το νευρωνικό δίκτυο έκανε ένα λάθος το οποίο θα μπορούσε να κάνει και ένας άνθρωπος αν κοίταζε από μακριά ή όχι τόσο προσεκτικά τη φωτογραφία. Η δεύτερη πρόβλεψη είναι σωστή καθώς υπάρχει κυματοθραύστης - breakwater στη φωτογραφία και οι άλλες 3 προβλέψεις σωσίβια λέμβος - lifeboat, ακρωτήρι - promontory και καταμαράν – catamaran, μοιάζουν με τα σχήματα της φωτογραφίας, όμως δεν εμπεριέχονται σε αυτήν.

```
data = format_img_vgg19(FILE_3)
pred = vgg19_model.predict(data)
display(load_img(FILE_3))
decode_predictions(pred)
```

```
1/1 [==============================] - 2s 2s/step
```

```
[[('n04347754', 'submarine', 0.16831164),
  ('n02894605', 'breakwater', 0.12554646),
  ('n03662601', 'lifeboat', 0.096701466),
  ('n09399592', 'promontory', 0.08424033),
  ('n02981792', 'catamaran', 0.08419327)]]
```

Εικόνα 22.12

Δοκιμάζουμε τα 2 αυτά μοντέλα, InceptionResNet και VGG19 με μια ακόμα φωτογραφία του δείγματός μας και λαμβάνουμε τα παρακάτω αποτελέσματα:

Το μοντέλο InceptionResNet, εικόνα 19.13, έχει ως 1η πρόβλεψη την πάνα – diaper, το καλάθι – hamper, την πετσέτα μπάνιου – bath towel, το φορείο – stretcher και τη μπανιέρα – bathtub. Αυτά τα αποτελέσματα σχετίζονται κατά κάποιον τρόπο ή θυμίζουν στοιχεία της εικόνας ωστόσο δεν υπάρχει πουθενά η πρόβλεψη για τα πόδια μωρού, το μωρό ή τα χέρια που είναι τα βασικά στοιχεία στην εικόνα. Επίσης το μοντέλο δεν είναι αρκετά σίγουρο για καμία από τις προβλέψεις του για αυτό άλλωστε και οι πιθανότητες για κάθε πρόβλεψη που έκανε είναι πολύ χαμηλές.

```
data = format_img_inceptionresnet('06 Feet.jpg')
prediction = inception_model.predict(data)
display(load_img('06 Feet.jpg'))
decode_predictions(prediction)
```

```
1/1 [==============================] - 1s 503ms/step
```

```
[[('n03188531', 'diaper', 0.096486986),
  ('n03482405', 'hamper', 0.0711633),
  ('n02808304', 'bath_towel', 0.05567977),
  ('n04336792', 'stretcher', 0.053134613),
  ('n02808440', 'bathtub', 0.045097623)]]
```

Εικόνα 22.13

Οι προβλέψεις της ίδιας εικόνας με το μοντέλο VGG19, εικόνα 22.14, είναι ότι υπάρχει ένα μπουρίτο, αφρικανικός γκρι παπαγάλος – African grey, ρόδα κεραμικής - potter’s wheel, ζύμη – dough, ινδική κόμπρα - Indian cobra. Εδώ τα αποτελέσματα είναι αρκετά πιο άσχετα με τα στοιχεία της εικόνας και από τις πιθανότητες φαίνεται πως πάλι το μοντέλο δεν είναι αρκετά σίγουρο για τις προβλέψεις που έκανε.

```
data = format_img_vgg19('06 Feet.jpg')
pred = vgg19_model.predict(data)
display(load_img('06 Feet.jpg'))
decode_predictions(pred)
```

```
1/1 [==============================] - 1s 833ms/step
```

```
[[('n07880968', 'burrito', 0.35974738),
  ('n01817953', 'African_grey', 0.095278166),
  ('n03992509', "potter's_wheel", 0.047266256),
  ('n07860988', 'dough', 0.03768623),
  ('n01748264', 'Indian_cobra', 0.02660745)]]
```

Εικόνα 22.14

23. Δημιουργία Recursive Neural Network RNN Αναγνώρισης Εικόνων

Στο κεφάλαιο γίνεται χρήση του παρακάτω αρχείου:

- *23 RNN.ipynb*

Στο νευρωνικό δίκτυο που θα κατασκευαστεί θα γίνει χρήση των δεδομένων του συνόλου MNIST Handwritten Digits. Όπως έχει ήδη αναφερθεί, το σύνολο αυτό περιέχει φωτογραφίες οι οποίες χαρακτηρίζονται από τα πίξελ ύψους και πλάτους και υπάρχει μόνο ένα χρωματικό κανάλι καθότι οι φωτογραφίες είναι σε γκρι αποχρώσεις. Μπορούμε λοιπόν να υποθέσουμε πως τα δεδομένα που περιγράφουν την κάθε εικόνα είναι μια μορφή πολυδιάστατης χρονοσειράς και το νευρωνικό δίκτυο RNN να δρα ως σαρωτής, περνώντας από κάθε πίξελ των εικόνων.

23.1 Συλλογή, Επεξεργασία και Εξερεύνηση Δεδομένων

Έχουμε 60.000 δεδομένα προς εκπαίδευση. Κάθε ένα από αυτά είναι μια εικόνα ενός χειρόγραφου ψηφίου από το 0 έως το 9 σε ασπρόμαυρες – γκρι αποχρώσεις, διαστάσεων 28 επί 28 πίξελ, εικόνα 23.1.

```
# Load Data
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
print("x_train.shape:", x_train.shape)
print("y_train.shape:", y_train.shape)
print("x_test.shape:", x_test.shape)
print("y_test.shape:", y_test.shape)
```

```
x_train.shape: (60000, 28, 28)
y_train.shape: (60000,)
x_test.shape: (10000, 28, 28)
y_test.shape: (10000,)
```

Εικόνα 23.1

Στη συνέχεια κατασκευάζουμε το μοντέλο δημιουργώντας ένα επίπεδο εισόδου – Input, ;ένα επίπεδο LSTM και ένα fully connected – Dense επίπεδο, εικόνα 23.2.

```
# Build Model
i = Input(shape=x_train[0].shape)
x = LSTM(128)(i)
x = Dense(10, activation='softmax')(x)

model = Model(i, x)
```

Εικόνα 23.2

23.2 Εκπαίδευση και Αξιολόγηση Μοντέλου

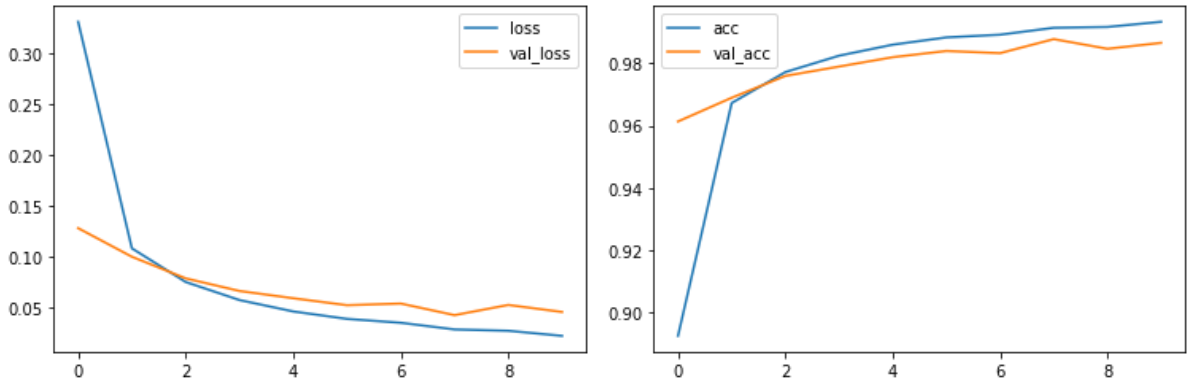
Εκπαιδεύοντας το μοντέλο με τον adam optimizer, loss function την sparse categorical cross entropy και για 10 εποχές λαμβάνουμε απώλεια δεδομένων εκπαίδευσης 0.0220, δεδομένων δοκιμής 0.0455 και ακρίβεια δεδομένων εκπαίδευσης 0.9933 και δεδομένων δοκιμής 0.9865, εικόνες 23.3, 23.4 και 23.5. Το μοντέλο φαίνεται να κάνει καλές γενικεύσεις χωρίς να αντιμετωπίζει το πρόβλημα αποθήκευσης πληροφορίας long distance στην κάθε εικόνα. Γίνεται λόγος για long distance πληροφορία καθότι δεν μπορούμε να γνωρίζουμε τι ψηφίο υπάρχει σε κάθε εικόνα αν δεν την υπολογίσουμε στο σύνολό της. Σίγουρα δε γίνεται να βασιστούμε αποκλειστικά στις πληροφορίες των τελευταίων γραμμών της εικόνας μόνο για να εξάγουμε συμπεράσματα για την κλάση στην οποία ανήκει η εικόνα καθότι το πιθανότερο είναι αυτά τα δεδομένα να αφορούν μαύρα πίξελ μιας και τα ψηφία βρίσκονται στο κέντρο κάθε εικόνας.

```
# Compile and Train Model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
r = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10)
```

Εικόνα 23.3

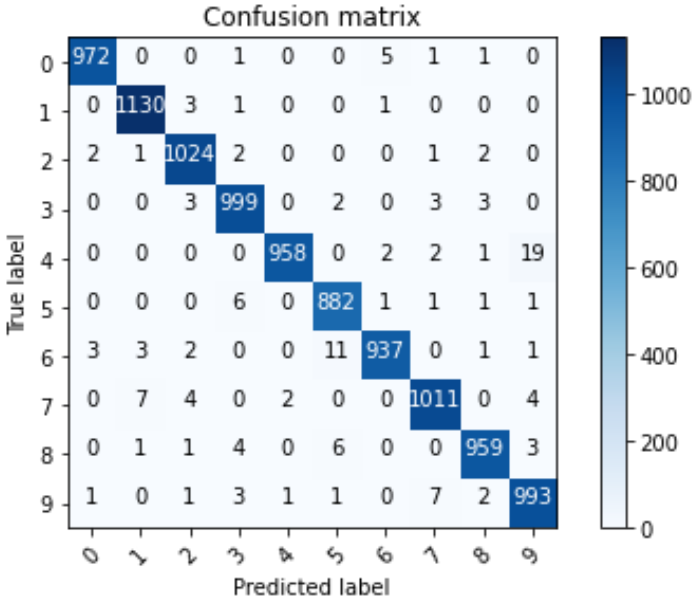
```
Epoch 10/10
1875/1875   - 9s 5ms/step - loss: 0.0220 - accuracy: 0.9933 - val_loss: 0.0455 - val_accuracy: 0.9865
```

Εικόνα 23.4

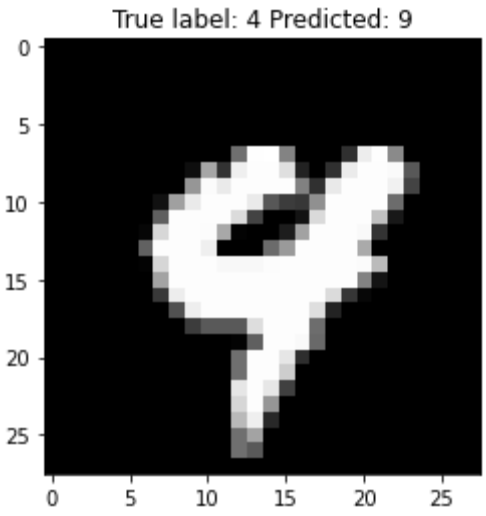


Εικόνα 23.5

Στη μήτρα σύγχυσης παρατηρούμε ότι τα περισσότερα δεδομένα είναι στην κύρια διαγώνιο, γεγονός που είναι αναμενόμενο λόγω της υψηλής ακρίβειας του μοντέλου. Τα περισσότερα λάθη εμφανίζονται στις περιπτώσεις που στην πραγματικότητα υπάρχει το ψηφίο 4 στην εικόνα και το μοντέλο προβλέπει 9, ένα λάθος που γίνεται συχνά και από ανθρώπους, εικόνες 23.6 και 23.7.



Εικόνα 23.6



Εικόνα 23.7

24. Δημιουργία Generative Adversarial Network GAN Αναγνώρισης Εικόνων

Στο κεφάλαιο γίνεται χρήση του παρακάτω αρχείου:

- *24 GAN.ipynb*

24.1 Συλλογή, Επεξεργασία και Εξερεύνηση Δεδομένων

Φορτώνουμε τα δεδομένα από το σύνολο MNIST και δημιουργούμε το σύνολο των χαρακτηριστικών x_train για εκπαίδευση με 60.000 δεδομένα διαστάσεων 28 επί 28, όσα και τα πίξελ ύψους επί πλάτους της κάθε εικόνας χειρόγραφου ψηφίου, y_train το σύνολο με 60.000 ετικέτες (κάθε ένα από τα 10 διαφορετικά ψηφία αντιστοιχεί σε μία κλάση) για εκπαίδευση, το σύνολο x_test με 10.000 δεδομένα χαρακτηριστικών για δοκιμή και το σύνολο y_test με 10.000 ετικέτες για δοκιμή. Εν συνεχεία μετασχηματίζουμε τα δεδομένα ώστε να λάβουν τιμές από -1 έως 1, έχοντας ως κέντρο το 0, με σκοπό τη διευκόλυνση της εκπαίδευσης, εικόνα 24.1.

```
# Load Data
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Map Inputs to (-1, +1) for Better Training / Images Centred around 0
x_train, x_test = x_train / 255.0 * 2 - 1, x_test / 255.0 * 2 - 1
print("x_train.shape:", x_train.shape)
print("y_train.shape:", y_train.shape)
print("x_test.shape:", x_test.shape)
print("y_test.shape:", y_test.shape)
```

```
x_train.shape: (60000, 28, 28)
y_train.shape: (60000,)
x_test.shape: (10000, 28, 28)
y_test.shape: (10000,)
```

Εικόνα 24.1

Μετασχηματίζουμε τα δεδομένα ώστε να έχουν μορφή πίνακα 2 διαστάσεων, εικόνα 24.2.

```
# Flatten Data to Be Tabular
N, H, W = x_train.shape
D = H * W
x_train = x_train.reshape(-1, D)
x_test = x_test.reshape(-1, D)
print("x_train.shape:", x_train.shape)
print("x_test.shape:", x_test.shape)
```

```
x_train.shape: (60000, 784)
x_test.shape: (10000, 784)
```

Εικόνα 24.2

Επιλέγουμε ο λανθάνων χώρος – latent space να έχει 100 διαστάσεις, εικόνα 24.3.

```
# Dimensionality of Latent Space
latent_dim = 100
```

Εικόνα 24.3

Ο generator παίρνει ως είσοδο ένα διάνυσμα από τον laten space, δημιουργούμε 3 κρυφά επίπεδα με συνάρτηση ενεργοποίησης τη leaky relu, γίνεται χρήση τους τεχνικής του batch normalization και τέλος δημιουργούμε ένα επίπεδο εξόδου με συνάρτηση ενεργοποίησης την tanh μιας και τα δείγματά τους έχουν τιμές από -1 έως 1, εικόνα 24.4.

```
# Generator Model
def build_generator(latent_dim):
  # Input from Latent Space
  i = Input(shape=(latent_dim,))
  x = Dense(256, activation=LeakyReLU(alpha=0.2))(i)
  x = BatchNormalization(momentum=0.7)(x)
  x = Dense(512, activation=LeakyReLU(alpha=0.2))(x)
  x = BatchNormalization(momentum=0.7)(x)
  x = Dense(1024, activation=LeakyReLU(alpha=0.2))(x)
  x = BatchNormalization(momentum=0.7)(x)
  x = Dense(D, activation='tanh')(x)

  model = Model(i, x)
  return model
```

Εικόνα 24.4

Στον discriminator δημιουργούμε ένα επίπεδο εισόδου, δύο κρυφά επίπεδα με συνάρτηση ενεργοποίησης τη leaky relu και ένα επίπεδο εξόδου. Μιας και η έξοδος του επιπέδου εξόδου εμπίπτει

στην κατηγορία binary classification (είναι αληθινή ή όχι η εικόνα), ως συνάρτηση ενεργοποίησης αυτού του επιπέδου χρησιμοποιείται η σιγμοειδής, εικόνα 24.5.

```
# Discriminator Model
def build_discriminator(img_size):
  i = Input(shape=(img_size,))
  x = Dense(512, activation=LeakyReLU(alpha=0.2))(i)
  x = Dense(256, activation=LeakyReLU(alpha=0.2))(x)
  x = Dense(1, activation='sigmoid')(x)
  model = Model(i, x)
  return model
```

Εικόνα 24.5

Ορίζουμε ως loss function την binary cross entropy στον discriminator και ως optimizer τον adam, εικόνα 24.6.

```
# Build and Compile Discriminator
discriminator = build_discriminator(D)
discriminator.compile(
    loss='binary_crossentropy',
    optimizer=Adam(0.0002, 0.5),
    metrics=['accuracy'])
```

Εικόνα 24.6

Κατασκευάζουμε τον generator καθώς και ένα μοντέλο συνδυασμού – combined model όπου μια είσοδος διέρχεται και από τον generator και τον discriminator. Τους στο combined model παγώνουμε τα βάρη του discriminator, έχουμε ως loss function την binary cross entropy και optimizer τον adam, εικόνα 24.7.

```
# Build and Compile Generator
generator = build_generator(latent_dim)

# Create Input to Represent Noise Sample from Latent Space
z = Input(shape=(latent_dim,))

# Pass Noise Through Generator to Get an Image
img = generator(z)

# Make Sure Only the Generator is Trained - Freeze Discriminator
discriminator.trainable = False

# The True Output is Fake, but we Label Them Real
fake_pred = discriminator(img)

# Create Combined Model
combined_model = Model(z, fake_pred)

# Compile Combined Model
combined_model.compile(loss='binary_crossentropy', optimizer=Adam(0.0002, 0.5))
```

Εικόνα 24.7

24.2 Εκπαίδευση και Αξιολόγηση Μοντέλου

Στο βρόχο εκπαίδευσης υπάρχουν δύο βήματα. Η εκπαίδευση του discriminator και η εκπαίδευση του generator .

Για την εκπαίδευση του discriminator χρειαζόμαστε αληθινές και τεχνητές εικόνες. Μπορούμε να πάρουμε αληθινές εικόνες από το σύνολο δεδομένων x_train. Εν συνεχεία θα χρειαστούμε κάποιες τεχνητές εικόνες, έτσι δημιουργούμε θόρυβο από την standard normal distribution του latent space. Περνάμε τους αληθινές εικόνες στον discriminator με την ένδειξη 1 για να επισημάνουμε ότι ανήκουν στη θετική κλάση και τους τεχνητές με την ένδειξη 0 για να επισημάνουμε ότι ανήκουν στην αρνητική κλάση. Υπολογίζουμε τη συνολική απώλεια και ακρίβεια παίρνοντας το μέσο όρο από τους αντίστοιχες τιμές τους αληθινές και τεχνητές εικόνες, εικόνα 24.8.

```
# Main training loop
for epoch in range(epochs):
  ############################
  ### Train Discriminator ###
  ############################

  # Select Random Batch of Images
  idx = np.random.randint(0, x_train.shape[0], batch_size)
  real_imgs = x_train[idx]

  # Generate Fake Images
  noise = np.random.randn(batch_size, latent_dim)
  fake_imgs = generator.predict(noise)

  # Train Discriminator
  # both loss and accuracy are returned
  d_loss_real, d_acc_real = discriminator.train_on_batch(real_imgs, ones)
  d_loss_fake, d_acc_fake = discriminator.train_on_batch(fake_imgs, zeros)
  d_loss = 0.5 * (d_loss_real + d_loss_fake)
  d_acc  = 0.5 * (d_acc_real + d_acc_fake)
```

Εικόνα 24.8

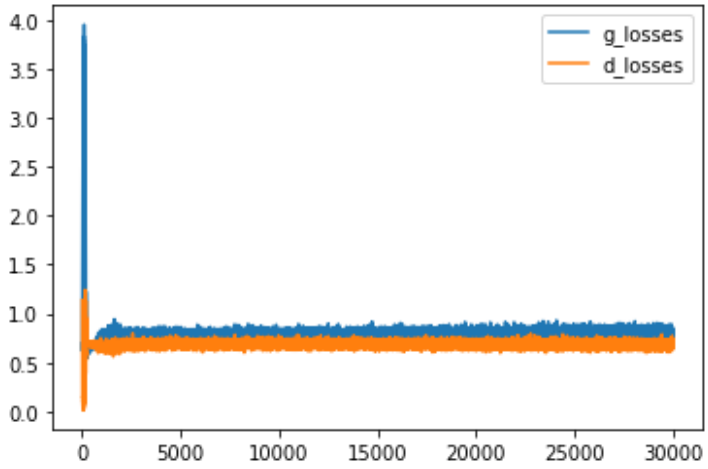
Για την εκπαίδευση του generator θα χρειαστούμε τεχνητές εικόνες, έτσι δημιουργούμε θόρυβο. Καλούμε το combined model με είσοδο το θόρυβο και στόχο ένα διάνυσμα από μονάδες – ones καθότι θέλουμε να κάνουμε τον discriminator να νομίζει ότι αυτές οι εικόνες είναι αληθινές και από αυτό λαμβάνουμε την απώλεια για τον generator, εικόνα 24.9.

```
noise = np.random.randn(batch_size, latent_dim)
g_loss = combined_model.train_on_batch(noise, ones)
```

Εικόνα 24.9

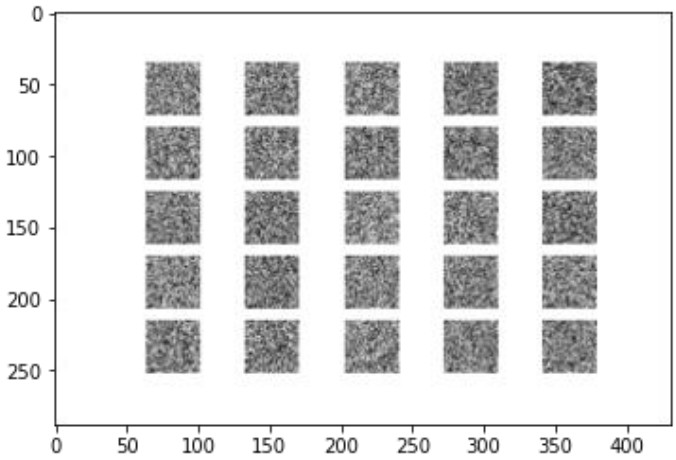
Παρατηρούμε ότι το μοντέλο δε φθάνει υψηλά επίπεδα ακρίβειας γεγονός που οφείλεται στο ότι ο discriminator βελτιώνεται, αλλά το ίδιο κάνει και ο generator, κάτι το οποίο είναι επιθυμητό ώστε ο generator να μπορέσει να δημιουργήσει αληθοφανείς εικόνες.

Σχετικά με τους απώλειες, παρατηρούμε πως μετά από κάποιες εποχές οι απώλειες και του generator και του discriminator κινούνται γύρω από τα ίδια επίπεδα, διότι υπάρχει αλληλεξάρτηση μεταξύ τους και βελτιώνονται παράλληλα, εικόνα 24.10.



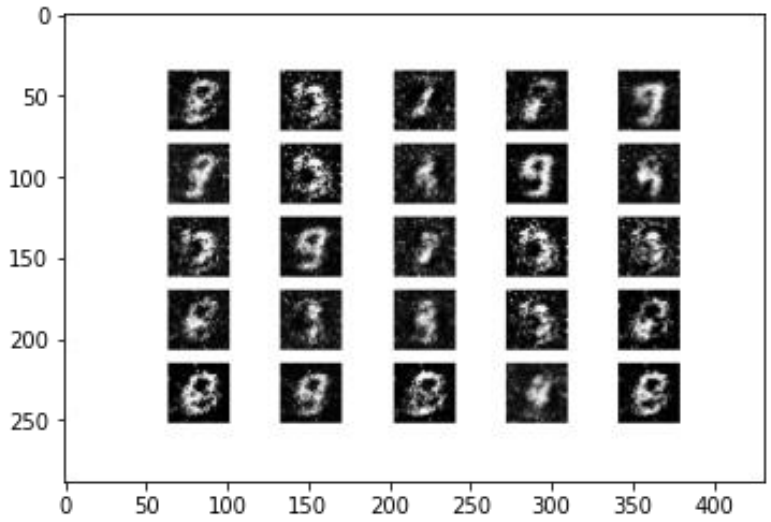
Εικόνα 24.10

Οπτικοποιώντας μερικά από τα αποτελέσματα βλέπουμε πως στην εποχή 0 ξεκινάμε με κάποιες ασπρόμαυρες εικόνες χωρίς κάποια σχέδια, εικόνα 24.11.



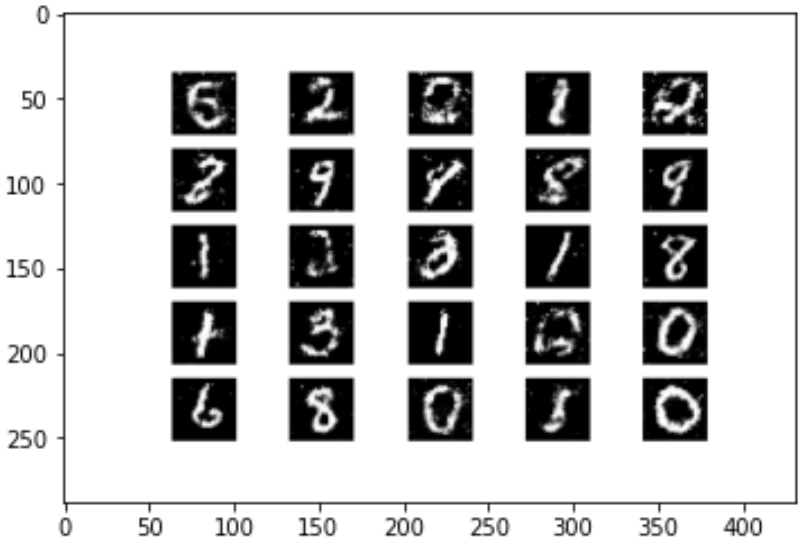
Εικόνα 24.11

Στην εποχή 1000 αρχίζουν και διαμορφώνονται κάποια σχήματα στις εικόνες, εικόνα 24.12



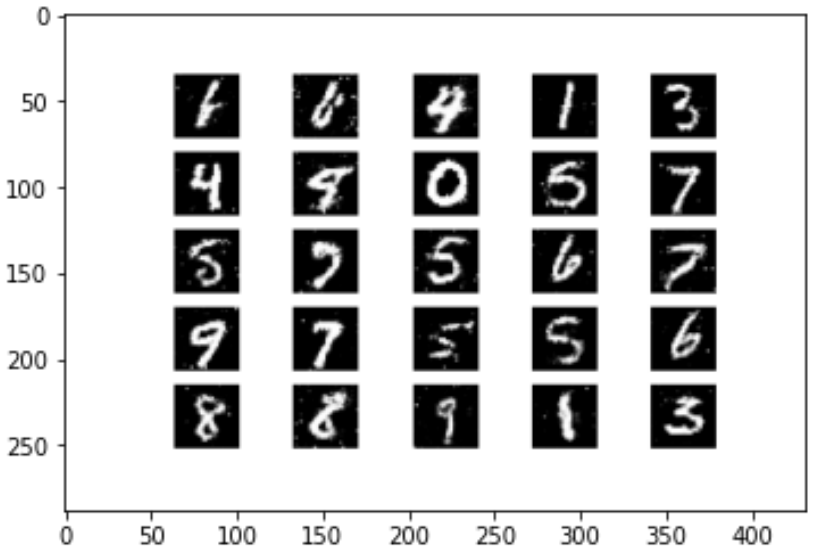
Εικόνα 24.12

Στην εποχή 10.000 αρχίζουμε και ξεχωρίζουμε κάποια ψηφία, εικόνα 24.13.



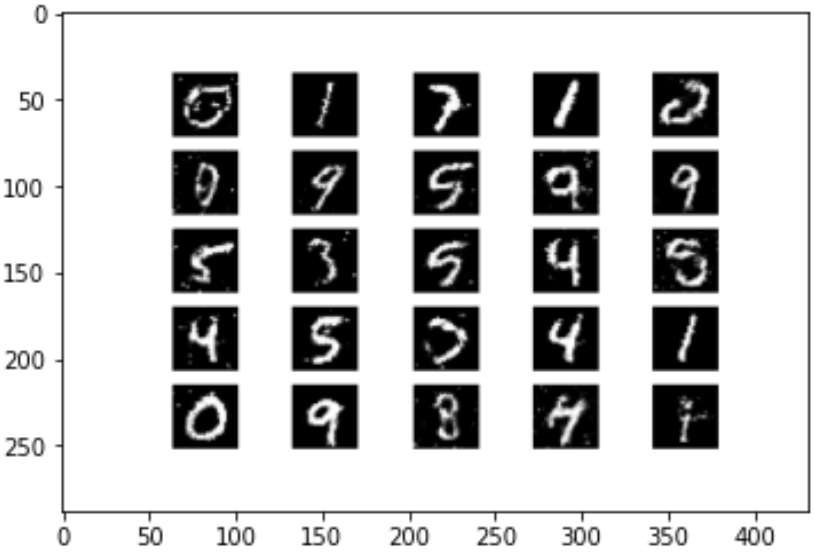
Εικόνα 24.13

Στην εποχή 20.000 τα ψηφία είναι ευκρινέστερα, εικόνα 24.14.



Εικόνα 24.14

Στην εποχή 29.800 παίρνουμε εικόνες οι οποίες θα μπορούσαν να είναι πραγματικές, εικόνα 24.15. Μπορούμε να ισχυριστούμε πως το εν λόγω μοντέλο εκπαιδεύτηκε αρκετά γρήγορα καθώς ήδη από την εποχή 10.000 τα αποτελέσματα είναι αρκετά ικανοποιητικά.



Εικόνα 24.15

Παράρτημα Ι

Στο παράρτημα Ι και ΙΙ παρατίθενται δύο εφαρμογές που συμπληρώνουν την κατανόηση της λειτουργίας της μηχανικής μάθησης.

Πρόβλεψη Εσόδων Ταινιών με τη Μέθοδο της Γραμμικής Παλινδρόμησης

Στο κεφάλαιο γίνεται χρήση των παρακάτω αρχείων από το φάκελο Appendix I Predict Movies Revenues - Linear Regression:

- *Predict Movies Revenues - Linear Regression.ipynb*
- *Cost_revenue_2columns.csv*
- *Cost_revenue_clean.csv*
- *Cost_revenue_dirty.csv*
- *Cost_revenue_no0.csv*

Έστω ότι θέλουμε να προβλέψουμε πόσα χρήματα μπορεί να δημιουργήσει μία ταινία. Ακολουθώντας τα προηγούμενα στάδια για να προσεγγίσουμε το πρόβλημα έχουμε:

Ι.1 Καθορισμός ερώτησης

Μπορούμε να ξεκινήσουμε από την εξής ερώτηση: «Πόσα χρήματα μπορεί να φέρει η ταινία». Η συγκεκριμένη ερώτηση όμως έχει κάποια προβλήματα. Αρχικά, τι εννοούμε με τον όρο «χρήματα»; Μπορούμε να είμαστε πιο συγκεκριμένοι; Ας διαμορφώσουμε λοιπόν την εξής ερώτηση: «Πόσα έσοδα μπορούμε να έχουμε από την ταινία;» Η ερώτηση αυτή είναι καλύτερη καθώς ο όρος έσοδα μπορεί να ποσοτικοποιηθεί, έχει συγκεκριμένη σημασία στον επιχειρηματικό κόσμο και μπορούμε να βρούμε δεδομένα πάνω σε αυτό. Το ζήτημα που τίθεται τώρα είναι από τι εξαρτώνται τα έσοδα των ταινιών (ηθοποιοί, προσωπικό, σκηνικά κτλ). Ενδεχομένως να πρέπει να δούμε όλο τον προϋπολογισμό της ταινίας για να μπορέσουμε να προβλέψουμε το πόσο καλά θα πάει με τα δυνητικά της έσοδα. Πολλές ταινίες με μεγάλους προϋπολογισμούς, είχαν μεγάλα έσοδα (πχ Τιτανικός, Άβαταρ), ωστόσο είναι αυτό το γεγονός αρκετό για να βγάλουμε κάποιο συμπέρασμα; Όχι, αλλά μπορούμε να διαμορφώσουμε ακόμα καλύτερα την αρχική μας υπόθεση ως εξής: «Μπορούμε να χρησιμοποιήσουμε τους προϋπολογισμούς ταινιών ώστε να προβλέψουμε τα έσοδά τους;» (εικόνα Ι.1). Αυτή η ερώτηση έχει το πλεονέκτημα ότι μπορεί να εξετασθεί και να δούμε αν υπάρχει κάποια σχέση μεταξύ του προϋπολογισμού και των εσόδων μιας ταινίας.

Μπορούμε να προσεγγίσουμε την τελική μορφή της ερώτησης χρησιμοποιώντας τη μέθοδο της γραμμικής παλινδρόμησης. Στην επιστήμη των δεδομένων τα «έσοδα» αποτελούν την εξαρτημένη μεταβλητή (dependent variable) ενώ στη μηχανική μάθηση το στόχο (target). Κατ' αναλογία στην

επιστήμη των δεδομένων ο «προϋπολογισμός» είναι η ανεξάρτητη μεταβλητή (independent variable) ενώ στη μηχανική μάθηση ονομάζεται χαρακτηριστικό (feature).

Εικόνα Ι.1

Ι.2 Συλλογή Δεδομένων

Για να απαντήσουμε στην ερώτηση «Μπορούμε να χρησιμοποιήσουμε τους προϋπολογισμούς ταινιών ώστε να προβλέψουμε τα έσοδά τους;» πρέπει να συλλέξουμε:

1. Δεδομένα για το χαρακτηριστικό, δηλαδή τους προϋπολογισμούς των ταινιών εκφρασμένους σε ένα νόμισμα.
2. Δεδομένα για τον στόχο, δηλαδή τα έσοδα των ταινιών εκφρασμένα σε ένα νόμισμα.

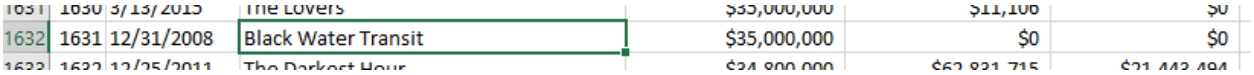
Μπορούμε να συλλέξουμε τα δεδομένα αυτά από την ιστοσελίδα www.the-movies.com, τα οποία οργανώνουμε σε ένα αρχείο excel (αρχείο cost_revenue_dirty.csv) και έτσι έχουμε στοιχεία για 5.391 ταινίες που αφορούν την ημερομηνία έναρξης προβολής, τον τίτλο ταινίας, τον προϋπολογισμό παραγωγής, τα παγκόσμια έσοδα σε δολάρια ΗΠΑ και τα έσοδά τους στις ΗΠΑ, εικόνα Ι.2.

Rank	Release Date	Movie Title	Production Budget ($)	Worldwide Gross ($)	Domestic Gross ($)
1	12/18/2009	Avatar	$425,000,000	$2,783,918,982	$760,507,625
2	12/18/2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$2,058,662,225	$936,662,225
3	5/24/2007	Pirates of the Caribbean: At Worldâ€™s E	$300,000,000	$963,420,425	$309,420,425
4	06-11-15	Spectre	$300,000,000	$879,620,923	$200,074,175
5	7/20/2012	The Dark Knight Rises	$275,000,000	$1,084,439,099	$448,139,099

Εικόνα Ι.2

Ι.3 Καθαρισμός Δεδομένων

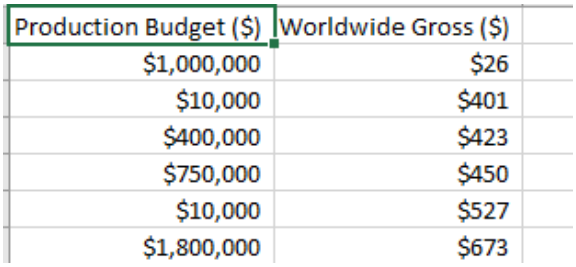
Στα δεδομένα παρατηρούμε πως υπάρχουν κάποιες ταινίες που στον προϋπολογισμό τους έχουν 0 έσοδα, όπως η ταινία Black Water Transit (εικόνα Ι.3). Η συγκεκριμένη ταινία, αν και είχε προγραμματίσει να προβληθεί στις 31 / 12 / 2008, δεν προβλήθηκε εν τέλει ποτέ εξαιτίας νομικών ζητημάτων που προέκυψαν. Δεδομένου ότι και άλλες ταινίες δεν κατάφεραν να προβληθούν ποτέ, η κάθε μία για τους δικούς της λόγους, μπορούμε να αποκλείσουμε από τους υπολογισμούς μας αυτές τις ταινίες των οποίων τα παγκόσμια έσοδα είναι 0 (αποκλείουμε 357 τέτοιες ταινίες, αρχείο cost_revenue_no0.csv).



1631	1630	3/13/2015	The Lovers	$35,000,000	$11,106	$0
1632	1631	12/31/2008	Black Water Transit	$35,000,000	$0	$0
1633	1632	12/25/2011	The Darkest Hour	$34,800,000	$62,831,715	$21,443,494

Εικόνα Ι.3

Επίσης για τους υπολογισμούς μας, μας ενδιαφέρουν μόνο οι στήλες του προϋπολογισμού (production budget) και των παγκόσμιων εσόδων (worldwide gross), συνεπώς αποφασίζουμε να διαγράψουμε όλες τις άλλες στήλες (εικόνα Ι.4).



Production Budget ($)	Worldwide Gross ($)
$1,000,000	$26
$10,000	$401
$400,000	$423
$750,000	$450
$10,000	$527
$1,800,000	$673

Εικόνα Ι.4

Τέλος αφαιρούμε το σύμβολο του δολαρίου $ καθώς και τα κόμματα από τους αριθμούς για να μπορέσουμε να επεξεργαστούμε τους αριθμούς και μετονομάζουμε τις στήλες σε production_budget_usd και worldwide_gross_usd ώστε να αποφύγουμε τυχόν τυπογραφικά λάθη που μπορούν να προκαλέσουν τα ειδικά σύμβολα και τα κενά, εικόνα Ι.5 (αρχείο cost_revenue_clean.csv).

production_budget_usd	worldwide_gross_usd
1000000	26
10000	401
400000	423
750000	450
10000	527

Εικόνα Ι.5

Ι.4 Εξερεύνηση και Οπτικοποίηση Δεδομένων

Η οπτικοποίηση των δεδομένων είναι εξαιρετικά σημαντική. Σε αυτό το βήμα θα γίνει χρήση του εργαλείου Jupyter notebook.

Αρχικά φορτώνουμε τα επεξεργασμένα πλέον – καθαρά δεδομένα μας (εικόνα Ι.6 και Ι.7).

```
In [9]: import pandas
        from pandas import DataFrame
        import matplotlib.pyplot as plt

In [10]: # upload data from csv file
         data = pandas.read_csv('cost_revenue_clean.csv')

In [11]: # show data
         data
```

Εικόνα Ι.6

Out[11]:

	production_budget_usd	worldwide_gross_usd
0	1000000	26
1	10000	401
2	400000	423
3	750000	450
4	10000	527
...	...	...
5029	225000000	1519479547
5030	215000000	1671640593
5031	306000000	2058662225
5032	200000000	2207615668
5033	425000000	2783918982

5034 rows × 2 columns

Εικόνα Ι.7

Παρατηρούμε κάποια στατιστικά στοιχεία των δεδομένων μας (πχ. μέγιστες και ελάχιστες τιμές, εικόνα Ι.8).

```
In [18]: data.describe()
```

Out[18]:

	production_budget_usd	worldwide_gross_usd
count	5.034000e+03	5.034000e+03
mean	3.290784e+07	9.515685e+07
std	4.112589e+07	1.726012e+08
min	1.100000e+03	2.600000e+01
25%	6.000000e+06	7.000000e+06
50%	1.900000e+07	3.296202e+07
75%	4.200000e+07	1.034471e+08
max	4.250000e+08	2.783919e+09

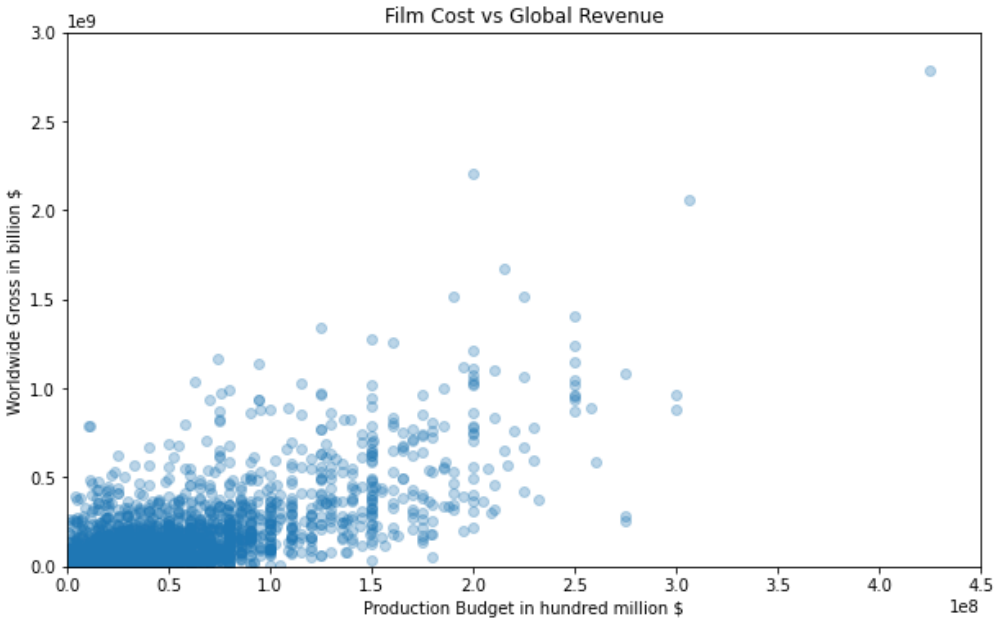
Εικόνα Ι.8

Δημιουργούμε ένα διάγραμμα διασποράς των δεδομένων μας (εικόνα Ι.9 και Ι.10).

```
In [12]: # X: production budgets
         X = DataFrame(data, columns=['production_budget_usd'])
         # y: revenue
         y = DataFrame(data, columns=['worldwide_gross_usd'])
```

```
In [19]: plt.figure(figsize=(10, 6))
         plt.scatter(X, y, alpha=0.3)
         plt.title('Film Cost vs Global Revenue')
         plt.xlabel('Production Budget in hundred million $')
         plt.ylabel('Worldwide Gross in billion $')
         # data don't go below zero on neither axis
         plt.xlim(0, 450000000)
         plt.ylim(0, 3000000000)
         plt.show()
```

Εικόνα Ι.9



Εικόνα Ι.10

Από το διάγραμμα διασποράς μπορούμε να παρατηρήσουμε τα εξής:

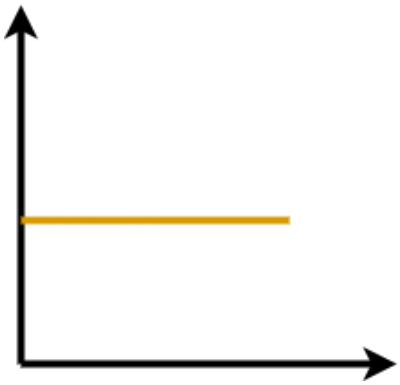
- Υπάρχουν κάποια απομακρυσμένα σημεία – outliers, όπως αυτό στο πάνω δεξιό μέρος του σχεδιαγράμματος. Συγκεκριμένα αυτή είναι η ταινία Avatar η οποία είχε υψηλό προϋπολογισμό και υψηλά έσοδα.
- Στο κάτω δεξί μέρος του σχεδιαγράμματος έχουμε τις ταινίες που κόστισαν αρκετά αλλά δεν είχαν αρκετά έσοδα.
- Οι περισσότερες ταινίες κόστισαν λιγότερο από 100 εκατομμύρια $, για αυτό και τα σημεία πυκνώνουν στο κάτω αριστερό μέρος του σχεδιαγράμματος.
- Από το σχεδιάγραμμα βλέπουμε πως φαίνεται να υπάρχει μια ανοδική τάση στα σημεία, δηλαδή ακριβότερες παραγωγές είχαν περισσότερα έσοδα.

Ι.5 Εκπαίδευση Αλγορίθμου

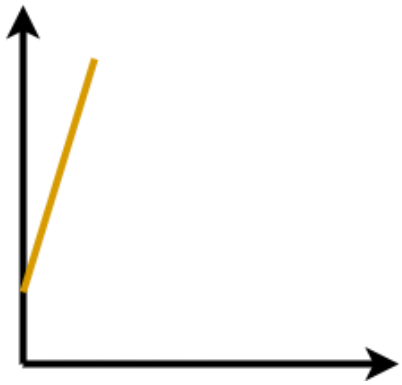
Θα χρησιμοποιήσουμε τον αλγόριθμο της γραμμικής παλινδρόμησης (linear regression), τον οποίον θα τροφοδοτήσουμε με 2 δεδομένα, τον προϋπολογισμό (χαρακτηριστικό – feature ή ανεξάρτητη μεταβλητή – independent variable) και τα έσοδα των ταινιών (αυτά προσπαθούμε να προβλέψουμε, στόχος – target ή εξαρτημένη μεταβλητή – dependent variable) εκφρασμένα σε δολάρια ΗΠΑ. Η γραμμική παλινδρόμηση θα προσπαθήσει να αποτυπώσει τη σχέση μεταξύ των 2 κατηγοριών δεδομένων σε μια ευθεία γραμμή.

Η γενική μορφή εξίσωσης ευθείας είναι ψ = αχ + β, όπου α είναι η κλίση της ευθείας και β το σημείο τομής της ευθείας με τον άξονα ψψ’. Η κλίση α φανερώνει πόσο θα αλλάξει το ψ για μια δεδομένη

μεταβολή του χ. Όσο μεγαλύτερη είναι η τιμή του α τόσο πιο απότομη είναι η κλίση α της ευθείας. Στην ακραία περίπτωση που τα δεδομένα δεν έχουν καμία απολύτως σχέση, τότε η ευθεία είναι παράλληλη στον άξονα χχ' και η κλίση α είναι 0, (εικόνα Ι.11α), ενώ όσο ισχυρότερη είναι η συσχέτιση των 2 κατηγοριών δεδομένων, τόσο πιο απότομη γίνεται η κλίση της ευθείας, εικόνα (εικόνα 14β) .



Εικόνα Ι.11α

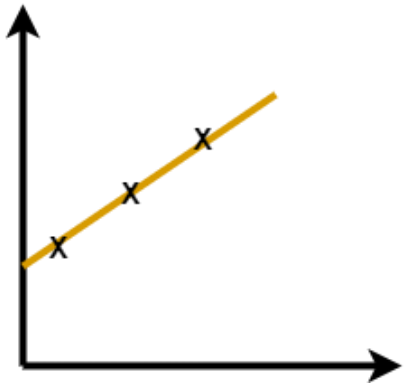


Εικόνα Ι.12β

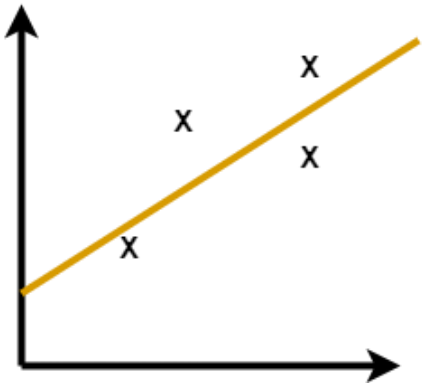
Στη μηχανική μάθηση οι τιμές της κλίσης α της ευθείας και της τομής της β με τον άξονα ψψ' είναι άγνωστες και αυτές προσπαθούμε να υπολογίσουμε. Οι τιμές αυτές ονομάζομαι παράμετροι και συχνά συμβολίζονται ως $\alpha = \theta_1$ και $\beta = \theta_0$, το ψ συμβολίζεται ως $h_{\theta(x)}$ και έτσι η εξίσωση ευθείας παίρνει τη μορφή:

$$\mathbf{h_\theta(x) = = \theta_0 + \theta_1 x}$$

Πρέπει να αποφασίσουμε ποια ευθεία έχει το καλύτερο δυνατό θ_0 και θ_1. Αν τα δεδομένα μας ήταν συγγραμμικά τότε το αποτέλεσμα θα ήταν απλό σχεδιάζονται την ευθείας που τα ενώνει, (εικόνα Ι.12α), στην πραγματικότητα όμως συνήθως έχουμε ένα νέφος από σημεία και πρέπει να αποφασίσουμε ποια ευθεία θα αποτυπώσει καλύτερα τη μεταξύ τους σχέση, (εικόνα Ι.12β).



Εικόνα Ι.12α



Εικόνα Ι.12β

Παρατηρούμε πως υπάρχει μια απόκλιση μεταξύ του πραγματικού δεδομένου X και της ευθείας που σχεδιάσαμε. Κάθε σημείο στην ευθεία ονομάζεται προβλεπόμενη τιμή (predicted / fitted value) και η διαφορά μεταξύ προβλεπόμενης y και πραγματικής $h_\theta(x)$ τιμής ονομάζεται κατάλοιπο ή εκτιμημένο σφάλμα (residual). Συνεπώς ο στόχος είναι ο υπολογισμός της ευθείας που θα ελαχιστοποιήσει αυτά τα κατάλοιπα. Κάποιες πραγματικές τιμές είναι κάτω από την ευθεία που θα σχεδιάσουμε και κάποιες πάνω και δεδομένου ότι εμείς χρειαζόμαστε το σύνολο όλων αυτών των διαφορών, υψώνουμε κάθε διαφορά στο τετράγωνο ώστε να βρούμε το συνολικό τους άθροισμα. Κατά συνέπεια ο στόχος μας είναι:

Για $h_\theta(x) = = \theta_0 + \theta_1 x$, να διαλέξουμε αυτά τα θ_0 και θ_1 ώστε να ελαχιστοποιηθεί η παράσταση:

RSS (Residual Sum of Squares): $(y^{(1)} - h_\theta(x^{(1)})^2 + \ldots\ldots + y^{(n)} - h_\theta(x^{(n)})^2)$

Για να εφαρμόσουμε τον αλγόριθμο της γραμμικής παλινδρόμησης θα χρησιμοποιήσουμε το πακέτο sklearn (εικόνα I.13).

```
from sklearn.linear_model import LinearRegression
```

Εικόνα I.13

Στη συνέχεια βρίσκουμε την κλίση (θ_1) καθώς και το σημείο τομής θ_o (εικόνα I.14).

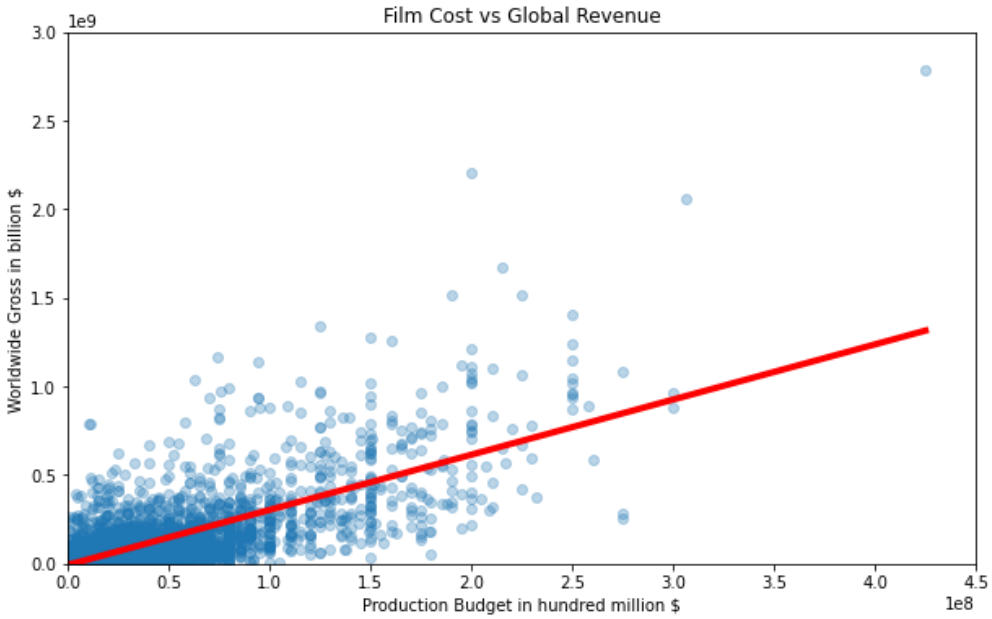
```
In [32]: # Linear Regression
         regression = LinearRegression()

         # run L.R. on features X and labels y
         regression.fit(X, y)

Out[32]: LinearRegression()

In [33]: # slope coefficient θ1
         regression.coef_

Out[33]: array([[3.11150918]])

In [34]: # Intercept θ0
         regression.intercept_

Out[34]: array([-7236192.72913975])
```

Εικόνα I.14

Τέλος μπορούμε να απεικονίσουμε την ευθεία γραμμικής παλινδρόμησης (εικόνα I.15 & I.16).

```
# plot L.R. line
plt.figure(figsize=(10, 6))
plt.scatter(X, y, alpha=0.3)

# for each X value we need the predicted values for the yy' axis
plt.plot(X.values, regression.predict(X), color='red', linewidth = 4)

plt.title('Film Cost vs Global Revenue')
plt.xlabel('Production Budget in hundred million $')
plt.ylabel('Worldwide Gross in billion $')
# data don't go below zero on neither axis
plt.xlim(0, 450000000)
plt.ylim(0, 3000000000)
plt.show()
```

Εικόνα Ι.15



Εικόνα Ι.16

Ι.6 Αξιολόγηση Αποτελέσματος

Στο σημείο αυτό πρέπει να αξιολογήσουμε το τι συμπεράσματα μπορούμε να εξάγουμε από τα δεδομένα των προϋπολογισμών των ταινιών σχετικά με τα έσοδά τους, εφαρμόζοντας τη γραμμική παλινδρόμηση.

Η κλίση, η παράμετρος θ_1 δηλαδή, είναι περίπου +3.11, γεγονός που σημαίνει πως υπάρχει θετική συσχέτιση μεταξύ προϋπολογισμού και εσόδων ταινιών. Πιο συγκεκριμένα αυτός ο αριθμός σημαίνει

πως για κάθε δολάριο ΗΠΑ που έχουμε δαπανήσει στον προϋπολογισμό μιας ταινίας αναμένουμε να έχουμε έσοδα της τάξης των 3.11 δολαρίων.

Η άλλη παράμετρος, θ_1, που τέμνει τον άξονα ψψ', είναι περίπου – 7.2 εκατομμύρια δολάρια, γεγονός που σημαίνει πως μία ταινία προϋπολογισμού 0 δολαρίων θα έχει αρνητικά έσοδα, δηλαδή έλλειμμα 7.2 εκατομμυρίων δολαρίων. Αυτό το συμπέρασμα είναι αρκετά μη ρεαλιστικό.

Η αλήθεια είναι πως το συγκεκριμένο μοντέλο αποτελεί μια υπεραπλούστευση της πραγματικότητας και για το λόγο αυτό θα πρέπει να το εμπιστευόμαστε με επιφύλαξη, ειδικά για τις τιμές που βρίσκονται στα άκρα του. Αν θέλουμε να κάνουμε μια πρόβλεψη για μια ταινία της οποίας ο προϋπολογισμός είναι 50 εκατομμύρια τότε η πρόβλεψη είναι πως τα έσοδα θα είναι περίπου τριπλάσια του προϋπολογισμού:

$$\mathbf{h_\theta(x) = = \theta_0 + \theta_1 x}$$

$$\mathbf{h_\theta(x) = = -7{,}236{,}192.7 + 3.1115 * 50{,}000{,}000 = 148{,}338{,}807}$$

Για να αξιολογήσουμε το αποτέλεσμα του εν λόγω μοντέλου, χρησιμοποιούμε το συντελεστή προσδιορισμού R^2 (R squared), ο οποίος εν προκειμένω φανερώνει το ποσό μεταβλητότητας των εσόδων μιας ταινίας που εξηγείται από τις μεταβολές στον προϋπολογισμό των ταινιών, παίρνει τιμές από 0 έως 1 και όσο μεγαλύτερη είναι η τιμή του τόσο καλύτερη είναι η προσαρμογή της γραμμικής παλινδρόμησης του δείγματος στα στοιχεία και αντίστροφα (εικόνα Ι.17).

```
In [40]: # R squared
         regression.score(X, y)

Out[40]: 0.5496485356985729
```

Εικόνα Ι.17

Ο συντελεστής R^2 είναι περίπου 0.55 γεγονός που σημαίνει πως το υπεραπλουστευμένο μοντέλο μας, με ένα μόνο χαρακτηριστικό, τον προϋπολογισμό των ταινιών, μπορεί να ερμηνεύσει το 55% της διακύμανσης που βλέπουμε στα παγκόσμια έσοδα των ταινιών.

Ι.7 Βελτιώσεις

Το μοντέλο θα είχε διαφορετική και ενδεχομένως ρεαλιστικότερη συμπεριφορά αν προσθέταμε και άλλα χαρακτηριστικά, πχ. πόσος καιρός χρειάστηκε για να παραχθεί η ταινία.

Στο μοντέλο θα μπορούσαν να χρησιμοποιηθούν και δεδομένα επιπλέον εκτός αυτών της εκπαίδευσης, δεδομένα δηλαδή τα οποία δεν έχει ξαναδεί.

Πολύ συχνά στον πραγματικό κόσμο η σχέση των δεδομένων δεν είναι γραμμική, συνεπώς πρέπει να μετασχηματίσουμε τα δεδομένα μας ώστε να μπορούν να ταιριάζουν στο μοντέλο.

Παράρτημα ΙΙ

Δημιουργία Ταξινομητή Ανεπιθύμητης - Επιθυμητής Ηλεκτρονικής Αλληλογραφίας με τη μέθοδο Naive Bayes

Στο κεφάλαιο γίνεται χρήση των παρακάτω αρχείων από το φάκελο Appendix II Spam Filter Naïve Bayes:

- *1 Bayes Classifier – Pre - Processing.ipynb (εικόνες σελίδες 153 – 169)*
- *2 Bayes Classifier – Training.ipynb (εικόνες σελίδες 169 – 175)*
- *3 Bayes Classifier – Testing, Inference & Evaluation.ipynb (εικόνες σελίδες 176 – 184)*
- *Φάκελος Δεδομένων - SpamData*

Ι.1 Καθορισμός ερώτησης

Η ερώτηση που πρέπει να απαντήσουμε στο συγκεκριμένο πρόβλημα είναι η εξής: Είναι ένα μέηλ σπαμ ή όχι;

Ι.2 Συλλογή δεδομένων

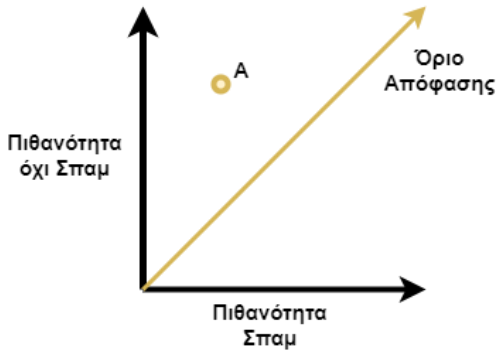
Για να μπορέσουμε να διαχωρίσουμε την ανεπιθύμητη από την επιθυμητή αλληλογραφία πρέπει αρχικά να καθορίσουμε ποια είναι τα χαρακτηριστικά εκείνα που κατατάσσουν ένα μέηλ στη μια κατηγορία ή την άλλην. Αυτό το πρόβλημα διαφέρει από εκείνα που αφορούν τις περιπτώσεις παλινδρόμησης με βάση την οποία προσπαθούμε να προσαρμοστούμε στα δεδομένα και αφορά τις περιπτώσεις κατηγοριοποίησης (classification) όπου πρέπει να γίνει διαχωρισμός των δεδομένων.

Μάλιστα για το εν λόγω πρόβλημα που θέλουμε να προσεγγίσουμε, αυτό του διαχωρισμού επιθυμητής και ανεπιθύμητης αλληλογραφίας, θα πρέπει να τροφοδοτήσουμε τον αλγόριθμο μηχανικής μάθησης που θα χρησιμοποιήσουμε με δεδομένα τα οποία θα έχουν τη μορφή κειμένου. Συνεπώς πρέπει να επεξεργαστούμε τα δεδομένα με τέτοιον τρόπο ώστε να είναι αντιληπτά από τον αλγόριθμο. Σε αυτό το βήμα πρέπει να προβούμε στη συλλογή των δεδομένων. Τα δεδομένα για το παρόν πρόβλημα θα τα αντλήσουμε από την ιστοσελίδα https://spamassassin.apache.org/old/publiccorpus/. Πρόκειται για μια ιστοσελίδα η οποία μας παρέχει πληθώρα διάφορων μέηλ, σπαμ και επιθυμητών, τα οποία μπορούμε να χρησιμοποιήσουμε για εκπαίδευση και δοκιμή του αλγορίθμου μας.

I.2.1. Corpus & Document

Η λέξη **corpus** στην τεχνική ορολογία που ασχολείται με σετ δεδομένων κειμένου σημαίνει ένα μεγάλο και δομημένο σετ κειμένων. Μία ακόμα λέξη που θα συναντήσουμε συχνά κατά τη διαδικασία συλλογής δεδομένων κειμένου είναι αυτή του **document** το οποίο στο παρόν πρόβλημα αναφέρεται σε ένα συγκεκριμένο μέλη του corpus. Εν ολίγοις το corpus είναι ένα σετ όλων των documents και το document αναφέρεται σε ένα συγκεκριμένο αντικείμενο μέσα στο corpus.

Θα κάνουμε χρήση του αλγορίθμου «Αφελής Bayes» (naive Bayes classifier) του οποίου σημαντικά πλεονεκτήματα είναι η ταχύτητα και η απλότητα (Muller – Guido, 2016). Στην πράξη ο αλγόριθμος αυτό που θα κάνει είναι να συγκρίνει την πιθανότητα ένα μέλη να είναι ανεπιθύμητο με ένα μέλη να είναι επιθυμητό. Αν για παράδειγμα η πιθανότητα ένα μέλη να είναι ανεπιθύμητο είναι μεγαλύτερη, τότε το μέλη θα καταταχθεί ως τέτοιο. Μπορούμε να αναπαραστήσουμε τις δύο αυτές πιθανότητες σε ένα σύστημα 2 αξόνων και η ευθεία που θα βρίσκεται ακριβώς στη μέση του σχεδιαγράμματος (η πορτοκαλί γραμμή στην εικόνα XX) ονομάζεται όριο απόφασης (decision boundary). Πάνω στο όριο απόφασης οι 2 πιθανότητες είναι ίσες. Το μοντέλο θα αποφασίσει για το αν ένα μέλη είναι ή όχι ανεπιθύμητο με βάση σε ποια από τις 2 πλευρές βρίσκεται. Για παράδειγμα σε περίπτωση που έρθει ένα μέλη και υπολογίσουμε την πιθανότητα να είναι σπαμ 70% και την πιθανότητα να μην είναι σπαμ 30%, τότε αυτό τοποθετείται στο σημείο A του σχεδιαγράμματος της εικόνας II.1, πάνω από το όριο απόφασης και εν τέλει ταξινομείται ως σπαμ. Κατ' αναλογία μπορούμε να τοποθετήσουμε στο σχεδιάγραμμα όλα τα μέλη που θα έχουμε, ανάλογα πάντα με την πιθανότητα που έχουν να είναι ή όχι σπαμ.



Εικόνα II.1

Για να μπορέσουμε να καταλάβουμε αν ένα μέλη είναι σπαμ ή όχι πρέπει να κοιτάξουμε το ίδιο το μέλη και να εντοπίσουμε τι είδους λέξεις ή θέματα συνήθως εμπεριέχονται σε αυτά τα μέλη (πχ. λέξεις όπως δωρεάν, πρόσβαση, προσφορά, ευκαιρία).

Ι.2.2 Πιθανότητες

Στο σημείο αυτό είναι σημαντικό να αναφερθούμε στα διάφορα είδη πιθανοτήτων που υπάρχουν.

Πιθανότητα: Είναι η προσδοκία να συμβεί ένα γεγονός. Αν για παράδειγμα θέλουμε να βρούμε την πιθανότητα ένα μέηλ που λάβαμε σε ένα έτος να είναι σπαμ, πρέπει να διαιρέσουμε τον αριθμό των σπαμ μέηλ που λάβαμε όλη τη χρονιά με το συνολικό αριθμό μέηλ που λάβαμε, σπαμ και μη, όλη τη χρονιά.

Η **συνδυασμένη πιθανότητα** είναι η πιθανότητα τομής δύο ενδεχομένων – γεγονότων Α και Β και συμβολίζεται P(A ∩ B). Για παράδειγμα αν ρίξουμε 2 φορές ένα κέρμα, τα πιθανά αποτελέσματα που έχουμε είναι κορώνα - κορώνα, κορώνα – γράμματα, γράμματα – γράμματα, γράμματα – κορώνα. Αν μας ενδιαφέρει η πιθανότητα και στην 1η ρίψη και στη 2η να έρθει κορώνα τότε μιλάμε για τη συνδυασμένη πιθανότητα (joint probability) να έρθει 2 φορές κορώνα και είναι μία ανάμεσα στα 4 πιθανά αποτελέσματα ή ¼ = 0.25% ή ½ * ½ = 0.25%.

Δεσμευμένη πιθανότητα: μετράει την πιθανότητα ενός γεγονότος Α δεδομένου ότι έχει συμβεί πριν κάποιο άλλο γεγονός Β και συμβολίζεται P(A | B). Για παράδειγμα, η πιθανότητα ένα μέηλ να είναι σπαμ αν εμπεριέχει τη λέξη δωρεάν.

Ένας ακόμα σημαντικός παράγοντας στον υπολογισμό των πιθανοτήτων είναι η **ανεξαρτησία των γεγονότων**. Το γεγονός δηλαδή στη 2η ρίψη ενός νομίσματος να λάβουμε κορώνα (ή γράμματα) δεν εξαρτάται καθόλου από το αποτέλεσμα της 1ης ρίψης

Αν τα ενδεχόμενα είναι ανεξάρτητα τότε η συνδυασμένη τους πιθανότητα είναι P(A ∩ B) = P(A)*P(B) και για τη δεσμευμένη πιθανότητα ισχύει P(A|B) = P(A).

Αν όμως τα γεγονότα είναι εξαρτημένα τότε η δεσμευμένη πιθανότητά τους, δηλαδή η πιθανότητα να γίνει το Α ενώ έχει γίνει το Β είναι:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

Ι.2.3. Θεώρημα Bayes

Το θεώρημα του Bayes προσφέρει έναν εναλλακτικό τρόπο υπολογισμού της δεσμευμένης πιθανότητας με τον εξής τρόπο:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Σχετικά με τον αλγόριθμο «Αφελής Bayes», το όνομα αφελές σχετίζεται με το γεγονός ότι ο αλγόριθμος υποθέτει ανεξαρτησία μεταξύ των ενδεχομένων εμφάνισης της κάθε λέξης σε ένα μέηλ. Έτσι λοιπόν η πιθανότητα ενός μέηλ να είναι σπαμ ενώ περιέχει τις λέξεις «προσφορά» και «δωρεάν» υπολογίζεται ως εξής: P(Spam | Προσφορά) ∩ : P(Spam | Δωρεάν). Κάθε λέξη στο μέηλ εξετάζεται απομονωμένα, η

συχνότητα κάθε λέξης στα μέηλ μετατρέπεται σε χαρακτηριστικό στο μοντέλο μας και επειδή ακριβώς κάθε λέξη εξετάζεται απομονωμένα, ο αλγόριθμος ονομάζεται αφελής. Αγνοούμε γραμματική, συντακτικό, σαρκασμό και εν γένει το ύφος των φράσεων. Επίσης, περιπτώσεις όπως Νέα Υόρκη θεωρούνται ως 2 λέξεις και οι εξαρτήσεις μεταξύ των λέξεων δε λαμβάνονται υπόψιν, καθότι θεωρούμε τις λέξεις ως ανεξάρτητες, όπως οι λέξεις σε μια σακούλα.

I.3 Καθαρισμός Δεδομένων

Ως μέρος της διαδικασίας καθαρισμού των δεδομένων μας αυτό που θα κάνουμε είναι:

- Να εξάγουμε τα σώματα των μέηλ
- Να μετατρέψουμε όλα τα μέηλ που έχουμε (βρίσκονται στα αρχεία spam_1, spam_2, easy_ham1, easy_ham2) από μορφή txt σε DataFrame
- Να ελέγξουμε για κενά μέηλ
- Να ελέγξουμε για τιμές που λείπουν

Τα μέηλ που έχουμε στη διάθεσή μας έχουν διάφορες πληροφορίες στις κεφαλίδες τους (headers) όπως πχ. διεύθυνση αποστολέα, λήπτη κτλ, όπως φαίνεται στην εικόνα II.2

```
From exmh-workers-admin@redhat.com  Thu Aug 22 12:36:23 2002
Return-Path: <exmh-workers-admin@spamassassin.taint.org>
Delivered-To: zzzz@localhost.netnoteinc.com
Received: from localhost (localhost [127.0.0.1])
        by phobos.labs.netnoteinc.com (Postfix) with ESMTP id D03E543C36
        for <zzzz@localhost>; Thu, 22 Aug 2002 07:36:16 -0400 (EDT)
Received: from phobos [127.0.0.1]
        by localhost with IMAP (fetchmail-5.9.0)
        for zzzz@localhost (single-drop); Thu, 22 Aug 2002 12:36:16 +0100 (IST)
Received: from listman.spamassassin.taint.org (listman.spamassassin.taint.org [66.187.233.211]) by
    dogma.slashnull.org (8.11.6/8.11.6) with ESMTP id g7MBYrZ04811 for
    <zzzz-exmh@spamassassin.taint.org>; Thu, 22 Aug 2002 12:34:53 +0100
```

Εικόνα II.2

Αυτό το μέρος των μέηλ δε μας ενδιαφέρει για την εκπαίδευση του αλγορίθμου αλλά μόνο το κύριο σώμα τους (e-mail body), εικόνα II.3 και μετατροπή των μέηλ σε DataFrame

```
def email_body_generator(path):

    for root, dirnames, filenames in walk(path):
        for file_name in filenames:

            filepath = join(root, file_name)

            stream = open(filepath, encoding='latin-1')

            is_body = False
            lines = []

            for line in stream:
                if is_body:
                    lines.append(line)
                elif line == '\n':
                    is_body = True

            stream.close()
            email_body = '\n'.join(lines)

            yield file_name, email_body
```

Εικόνα ΙΙ.3

```
def df_from_directory(path, classification):
    rows = []
    row_names = []

    for file_name, email_body in email_body_generator(path):
        rows.append({'MESSAGE': email_body, 'CATEGORY': classification})
        row_names.append(file_name)

    return pd.DataFrame(rows, index=row_names)
```

```
# create dataframe of spam mails
# CATEGORY 0 non spam - 1 spam
spam_emails = df_from_directory(SPAM_1_PATH, 1)
# append spam mails from the other folder
spam_emails = spam_emails.append(df_from_directory(SPAM_2_PATH, 1))
spam_emails.head()
```

Εικόνα ΙΙ.4

```
# create dataframe for ham mails
ham_emails = df_from_directory(EASY_NONSPAM_1_PATH, HAM_CAT)
ham_emails = ham_emails.append(df_from_directory(EASY_NONSPAM_2_PATH, HAM_CAT))
```

Εικόνα ΙΙ.5

```
# create dataframe of all emails spam + ham
data = pd.concat([spam_emails, ham_emails])
print(f"Shape of entire dataframe is: {data.shape}")
data.head()
```

```
Shape of entire dataframe is: (5799, 2)
```

	MESSAGE	CATEGORY
00001.7848dde101aa985090474a91ec93fcf0	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Tr...	1
00002.d94f1b97e48ed3b553b3508d116e6a09	1) Fight The Risk of Cancer!\n\nhttp://www.adc...	1
00003.2ee33bc6eacdb11f38d052c44819ba6c	1) Fight The Risk of Cancer!\n\nhttp://www.adc...	1
00004.eac8de8d759b7e74154f142194282724	##########...	1
00005.57696a39d7d84318ce497886896bf90d	I thought you might like these:\n\n1) Slim Dow...	1

Εικόνα II.6

Παρατηρούμε πως το σύνολο των μέηλ που έχω, σπαμ και μη, είναι 5799.

Διερευνούμε αν έχουμε κενά μέηλ ή τιμές που να λείπουν και παρατηρούμε πως έχουμε 3 κενά μέηλ, εικόνα II.7 και II.8

```
# check if message bodies are null
data['MESSAGE'].isnull().values.any()
```

```
False
```

```
# check if any mail is empty, string length zero
(data.MESSAGE.str.len() == 0).any()
```

```
True
```

Εικόνα II.7

```
# how many mails are of length 0
(data.MESSAGE.str.len() == 0).sum()
```

```
3
```

```
# check for  null/None values
data.MESSAGE.isnull().sum()
```

```
0
```

Εικόνα II.8

Εντοπίζουμε τα κενά μέηλ και παρατηρούμε πως το πρόβλημα εντοπίζεται στα αρχεία cmds τα οποία και δεν πρέπει να συμπεριλάβουμε στον αλγόριθμο, εικόνα II.9

```
data[data.MESSAGE.str.len() == 0].index
```

```
Index(['cmds', 'cmds', 'cmds'], dtype='object')
```

Εικόνα II.9

Αφαιρούμε αυτά τα 3 αρχεία από το dataframe το οποίο πλέον έχει 5796 αρχεία – μέηλ

```
# remove cmds files and updata data
data.drop(['cmds'], inplace=True)
```

```
data.shape
```

```
(5796, 2)
```

Εικόνα II.10

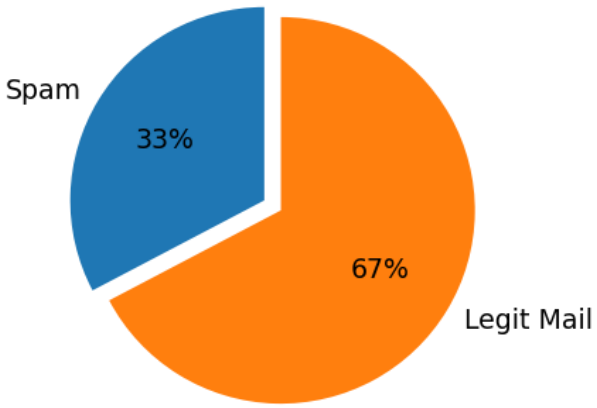
5.4 Εξερεύνηση και Οπτικοποίηση Δεδομένων - Επεξεργασία Φυσικής Γλώσσας

Συνολικά έχουμε 3900 μη σπαμ μέηλ (κατηγορία 0) και 1896 σπαμ μέηλ (κατηγορία 1), εικόνα II.11

```
data.CATEGORY.value_counts()
```

```
0    3900
1    1896
Name: CATEGORY, dtype: int64
```

Εικόνα II.11



Εικόνα II.12

Επεξεργασία Φυσικής Γλώσσας

Πρέπει να μετατρέψουμε τα δεδομένα μας που αυτή τη στιγμή είναι σε μορφή κειμένου, σε μια τέτοια μορφή την οποία ο αλγόριθμος θα είναι ικανός να κατανοήσει και να επεξεργαστεί. Κατά συνέπεια:

- Θα μετατρέψουμε όλα τα γράμματα των κειμένων σε πεζά
- Θα χωρίσουμε τις λέξεις των προτάσεων των κειμένων (**tokenization**)
- Θα αφαιρέσουμε τις λέξεις τύπου stop (**stop words**), δηλαδή λέξεις που απαντώνται πολύ συχνά σε κείμενα και χρησιμοποιούνται κυρίως για γραμματικούς σκοπούς παρά για να μεταφέρουν κάποιο νόημα (πχ άρθρα).
- Θα αφαιρέσουμε τα HTML tags από τα μέηλ
- Θα χρησιμοποιήσουμε τις ρίζες των λέξεων (**stemming**), ανεξάρτητα από το γραμματικό τους τύπο (χρόνος, έγκλιση, πτώση κτλ). Για παράδειγμα οι αγγλικές λέξεις going, goes, go μοιράζονται την ίδια ρίζα και το μόνο που αλλάζει είναι ο γραμματικός τύπος. Εμείς θέλουμε να αντιμετωπίσουμε αυτές τις 3 λέξεις με τον ίδιον τρόπο. Είναι πιθανό να καταλήξουμε σε μια λέξη η οποία δεν υπάρχει στη γλώσσα (για παράδειγμα οι λέξεις argue, argued, argues, arguing επεξεργάζονται και χρησιμοποιείται ο τύπους argu που δεν αντιστοιχεί σε πραγματική λέξη της αγγλικής γλώσσας).
- Θα αφαιρέσουμε τα σημεία στίξης.

Για να πετύχουμε τους παραπάνω σκοπούς θα γίνει χρήση του πακέτου nltk

Δημιουργούμε ένα σύνολο με όλα τα stop words της αγγλικής γλώσσας, εικόνα II.13:

```
# create a set of stopwords
stop_words = set(stopwords.words('english'))
```

Εικόνα II.13

Δημιουργούμε μία μέθοδο η οποία αφαιρεί όλα τα stop words καθώς τα html tags από τα μέηλ, εικόνα II.14.

```
def clean_msg_no_html(message, stemmer=PorterStemmer(),
                      stop_words=set(stopwords.words('english'))):

    # Remove HTML tags
    soup = BeautifulSoup(message, 'html.parser')
    cleaned_text = soup.get_text()

    # Converts to Lower Case and splits up the words
    words = word_tokenize(cleaned_text.lower())

    filtered_words = []

    for word in words:
        # Removes the stop words and punctuation
        if word not in stop_words and word.isalpha():
            filtered_words.append(stemmer.stem(word))

    return filtered_words
```

Εικόνα ΙΙ.14

Εφαρμόζουμε την παραπάνω μέθοδο σε όλα τα 5796 μέηλ που έχουμε και αποθηκεύουμε τις επεξεργασμένες πλέον λέξεις του κάθε μέηλ σε μια μεταβλητή με το όνομα nested_list, εικόνα ΙΙ.15.

```
nested_list.tail()
```

```
DOC_ID
5791    [http, bizarr, collect, stuf, anim, could, fet...
5792    [care, use, one, also, realli, cute, thing, ja...
5793    [sm, skip, montanaro, write, jeremi, put, anot...
5794    [mark, hammond, like, given, zodb, sound, attr...
5795    [hi, probabl, use, whatsoev, also, problem, re...
Name: MESSAGE, dtype: object
```

```
len(nested_list)
```

```
5796
```

Εικόνα ΙΙ.15

Εν συνεχεία δημιουργούμε ένα λεξικό με τις 2500 πιο κοινές και «καθαρές» λέξεις που απαντώνται στα μέηλ το οποίο και αποθηκεύουμε σε ένα csv αρχείο, εικόνα ΙΙ.16.

Create a Vocabulary DataFrame with a WORD_ID

```
# Create vocabulary to train the classifier
word_ids = list(range(0, VOCAB_SIZE))
vocab = pd.DataFrame({'VOCAB_WORD': frequent_words.index.values}, index=word_ids)
vocab.index.name = 'WORD_ID'
vocab.head()
```

WORD_ID	VOCAB_WORD
0	http
1	use
2	list
3	email
4	get

Save Vocabulary as a CSV File

```
vocab.to_csv(WORD_ID_FILE, index_label=vocab.index.name, header=vocab.VOCAB_WORD.name)
```

Εικόνα II.16

Διαπιστώνουμε πως ο μέγιστος αριθμός επεξεργασμένων λέξεων σε ένα μέηλ είναι 7671 εικόνα II.7:

```
# Python List Comprehension
clean_email_lengths = [len(sublist) for sublist in stemmed_nested_list]
print('Nr words in the longest email:', max(clean_email_lengths))
```

```
Nr words in the longest email: 7671
```

Εικόνα II.17

Κάθε λέξη από αυτές θα αποτελέσει ένα χαρακτηριστικό – feature για τον αλγόριθμο που θα χρησιμοποιήσουμε. Ανακατεύουμε και χωρίζουμε τα δεδομένα σε test & training tests με αναλογία 30% - 70%. Παρατηρούμε πως υπάρχει αντιστοιχία μεταξύ των DOC_ID στα X σύνολα των χαρακτηριστικών και στα y σύνολα των στόχων. Έτσι το 1^{ο} μέηλ στο σύνολο X_train είναι το 4844 και το 1^{ο} μέηλ στο σύνολο y_train είναι πάλι το 4844, εικόνα II.18.

```
X_train, X_test, y_train, y_test = train_test_split(word_columns_df, data.CATEGORY, test_size=0.3, random_state=42)
```

```
print(f"No of training samples: {X_train.shape[0]}")
print(f"Fraction of training samples: {X_train.shape[0] / word_columns_df.shape[0]}")
```

```
No of training samples: 4057
Fraction of training samples: 0.6999654934437544
```

```
X_train.index.name = X_test.index.name = 'DOC_ID'
X_train.head()
```

DOC_ID	0	1	2	3	4	5	6	7	8	9	...	7661	7662	7663	7664	7665	7666	7667	7668	7669
4844	ye	inde	agent	directori	verita	cd	unix	subdirectori	file	call	...	None	None	None	None	None	None	None	None	None
4727	problem	come	tri	instal	harddissssk	like	alreadi	mount	http	yahoo	...	None	None	None	None	None	None	None	None	None
5022	origin	messag	date	mon	aug	chad	norwood	sven	cc	subject	...	None	None	None	None	None	None	None	None	None
3504	inlin	folk	sever	major	internet	outag	morn	across	major	provid	...	None	None	None	None	None	None	None	None	None
3921	url	http	date	bath	chronicl	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None

5 rows × 7671 columns

Εικόνα ΙΙ.18

```
y_train.head()
```

```
DOC_ID
4844    0
4727    0
5022    0
3504    0
3921    0
Name: CATEGORY, dtype: int64
```

```
X_test.head()
```

DOC_ID	0	1	2	3	4	5	6	7	8	9	...	7661	7662	7663	7664	7665	7666	7667
4675	interest	alway	wonder	thing	bad	exampl	goto	languag	support	goto	...	None	None	None	None	None	None	None
4220	url	http	date	final	gdc	europ	review	confernec	session	ect	...	None	None	None	None	None	None	None
2484	stephen	william	mailto	swilliam	weaken	food	transact	argument	note	neighborhood	...	None	None	None	None	None	None	None
2418	el	mon	sep	bitbitch	wrote	eugen	mani	homo	friend	lover	...	None	None	None	None	None	None	None
5110	music	school	joke	american	conductor	european	conductor	talk	european	conductor	...	None	None	None	None	None	None	None

5 rows × 7671 columns

Εικόνα ΙΙ.19

```
: y_test.head()

: DOC_ID
  4675    0
  4220    0
  2484    0
  2418    0
  5110    0
  Name: CATEGORY, dtype: int64
```

Εικόνα ΙΙ.20

Σύνοψη

Ξεκινήσαμε με 5797 μέηλ συνολικά (σπαμ - μη σπαμ) και τα χωρίσαμε με αναλογία 70% σε μέηλ για εκπαίδευση αλγορίθμου – train (4057 μέηλ) και 30% για δοκιμή – τεστ του αλγορίθμου (1739).

Στη συνέχεια δημιουργούμε μία αραιή μήτρα (sparse matrix) στην οποία έχουμε τις πληροφορίες:

Κατηγορία μέηλ (LABEL: 0 για επιθυμητό, 1 για σπαμ, DOC_ID αύξων αριθμός μέηλ, OCCURRENCE: πόσες φορές συναντάται η λέξη στο μέηλ, WORD_ID: κωδικός της λέξης καθότι κάθε μία από τις 2500 λέξεις του λεξικού που δημιουργήσαμε την αντιστοιχίσαμε σε έναν κωδικό), εικόνα ΙΙ.21.

```
def make_sparse_matrix(df, indexed_words, labels):
    """
    Returns sparse matrix as dataframe.

    df: A dataframe with words in the columns with a document id as an index (X_train or X_test)
    indexed_words: index of words ordered by word id
    labels: category as a series (y_train or y_test)
    """

    nr_rows = df.shape[0]
    nr_cols = df.shape[1]
    word_set = set(indexed_words)
    dict_list = []

    for i in range(nr_rows):
        for j in range(nr_cols):

            word = df.iat[i, j]
            if word in word_set:
                doc_id = df.index[i]
                word_id = indexed_words.get_loc(word)
                category = labels.at[doc_id]

                item = {'LABEL': category, 'DOC_ID': doc_id,
                        'OCCURENCE': 1, 'WORD_ID': word_id}

                dict_list.append(item)

    return pd.DataFrame(dict_list)
```

Εικόνα ΙΙ.21

```
%%time
sparse_train_df = make_sparse_matrix(X_train, word_index, y_train)
```

```
CPU times: total: 20min 32s
Wall time: 25min 48s
```

```
sparse_train_df[:5]
```

	LABEL	DOC_ID	OCCURENCE	WORD_ID
0	0	4844	1	265
1	0	4844	1	1239
2	0	4844	1	504
3	0	4844	1	308
4	0	4844	1	254

```
sparse_train_df.shape
```

```
(429184, 4)
```

Εικόνα II.22

Ομαδοποιούμε τις λέξεις μας ανά μέηλ, εικόνα II.23:

```
train_grouped = sparse_train_df.groupby(['DOC_ID', 'WORD_ID', 'LABEL']).sum()
train_grouped.head()
```

			OCCURENCE
DOC_ID	WORD_ID	LABEL	
0	**2**	**1**	1
	3	**1**	2
	4	**1**	1
	7	**1**	3
	11	**1**	1

```
# word at word_id 2
vocab.at[2, 'VOCAB_WORD']
```

```
'list'
```

Εικόνα II.23

Μπορούμε να κάνουμε έναν έλεγχο των αποτελεσμάτων μας. Στην εικόνα II.24 βλέπουμε πως η λέξη με WORD_ID είναι η λέξη list, η οποία συναντάται 1 φορά (OCCURRENCE) στο μέηλ με DOC_ID 0:

```
# The word list occurs only 1 time in message doc_id 0
data.MESSAGE[0]
```

```
'<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">\n\n<HTML><HEAD>\n\n<META content=3D"text/html; charset=3Dwindow
s-1252" http-equiv=3DContent-T=\n\nype>\n\n<META content=3D"MSHTML 5.00.2314.1000" name=3DGENERATOR></HEAD>\n\n<BODY><!-- Inser
ted by Calypso -->\n\n<TABLE border=3D0 cellPadding=3D0 cellSpacing=3D2 id=3D_CalyPrintHeader_ r=\n\nules=3Dnone \n\nstyle=3D"C
OLOR: black; DISPLAY: none" width=3D"100%">\n\n  <TBODY>\n\n  <TR>\n\n    <TD colSpan=3D3>\n\n      <HR color=3Dblack noShade S
IZE=3D1>\n\n    </TD></TR></TD></TR>\n\n  <TR>\n\n    <TD colSpan=3D3>\n\n      <HR color=3Dblack noShade SIZE=3D1>\n\n    </TD
></TR></TBODY></TABLE><!-- End Calypso --><!-- Inserted by Calypso=\n\n --><FONT \n\ncolor=3D#000000 face=3DVERDANA,ARIAL,HELVE
TICA size=3D-2><BR></FONT></TD><=\n\n/TR></TABLE><!-- End Calypso --><FONT color=3D#ff0000 \n\nface=3D"Copperplate Gothic Bold"
size=3D5 PTSIZE=3D"10">\n\n<CENTER>Save up to 70% on Life Insurance.</CENTER></FONT><FONT color=3D#ff=\n\n0000 \n\nface=3D"Copp
erplate Gothic Bold" size=3D5 PTSIZE=3D"10">\n\n<CENTER>Why Spend More Than You Have To?\n\n<CENTER><FONT color=3D#ff0000 face=
3D"Copperplate Gothic Bold" size=3D5 PT=\n\nSIZE=3D"10">\n\n<CENTER>Life Quote Savings\n\n<CENTER>\n\n<P align=3Dleft></P>\n\n<
P align=3Dleft></P></FONT></U></I></B><BR></FONT></U></B></U></I>\n\n<P></P>\n\n<CENTER>\n\n<TABLE border=3D0 borderColor=3D#11
1111 cellPadding=3D0 cellSpacing=3D0 wi=\n\ndth=3D650>\n\n  <TBODY></TBODY></TABLE>\n\n<TABLE border=3D0 borderColor=3D#111111
cellPadding=3D5 cellSpacing=3D0 wi=\n\ndth=3D650>\n\n  <TBODY>\n\n  <TR>\n\n    <TD colSpan=3D2 width=3D"35%"><B><FONT face=3DV
erdana size=3D4>Ensurin=\n\ng your \n\n      family\'s financial security is very important. Life Quote Savings ma=\n\nkes \n\n
buying life insurance simple and affordable. We Provide FREE Access =\n\nto The \n\n      Very Best Companies and The Lowest Ra
tes.</FONT></B></TD></TR>\n\n  <TR>\n\n    <TD align=3Dmiddle vAlign=3Dtop width=3D"18%">\n\n      <TABLE borderColor=3D#111111
width=3D"100%">\n\n        <TBODY>\n\n        <TR>\n\n          <TD style=3D"PADDING-LEFT: 5px; PADDING-RIGHT: 5px" width=3D"10
0=\n\n%"><FONT \n\n            face=3DVerdana size=3D4><B>Life Quote Savings</B> is FAST, EAS=\n\nY and \n\n            SAVES y
ou money! Let us help you get started with the best val=\n\nues in \n\n            the country on new coverage. You can SAVE hu
ndreds or even tho=\n\nusands \n\n            of dollars by requesting a FREE quote from Lifequote Savings. =\n\nOur \n\n
service will take you less than 5 minutes to complete. Shop an=\n\nd \n\n            compare. SAVE up to 70% on all types of Li
fe insurance! \n\n</FONT></TD></TR>\n\n        <TR><BR><BR>\n\n          <TD height=3D50 style=3D"PADDING-LEFT: 5px; PADDING-RI
GHT: 5px" \n\n          width=3D"100%">\n\n            <P align=3Dcenter><B><FONT face=3DVerdana size=3D5><A \n\n            hr
ef=3D"http://website.e365.cc/savequote/">Click Here For Your=\n\n \n\n            Free Quote!</A></FONT></B></P></TD>\n\n
<P><FONT face=3DVerdana size=3D4><STRONG>\n\n          <CENTER>Protecting your family is the best investment you\'ll eve=\n\nr
\n\n          make!<BR></B></TD></TR>\n\n        <TR><BR><BR></STRONG></FONT></TD></TR></TD></TR>\n\n        <TR></TR></TBODY>
</TABLE>\n\n      <P align=3Dleft><FONT face=3D"Arial, Helvetica, sans-serif" size=3D2=\n\n></FONT></P>\n\n      <P></P>\n\n
<CENTER><BR><BR><BR>\n\n      <P></P>\n\n      <P align=3Dleft><BR></B><BR><BR><BR><BR></P>\n\n      <P align=3Dcenter><BR></P>
\n\n      <P align=3Dleft><BR></B><BR><BR></FONT>If you are in receipt of this=\n\n email \n\n      in error and/or wish to be
removed from our list, <A \n\n      href=3D"mailto:coins@btamail.net.cn">PLEASE CLICK HERE</A> AND TYPE =\n\nREMOVE. If you \n
\n      reside in any state which prohibits e-mail solicitations for insuran=\n\nce, \n\n      please disregard this \n\n
email.<BR></FONT><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR=\n\n><BR><BR><BR></FONT></P></CENTER></CENTER></TR></TBODY
></TABLE></CENTER></=\n\nCENTER></CENTER></CENTER></CENTER></BODY></HTML>\n\n\n\n\n\n\n'
```

Εικόνα ΙΙ.24

Τέλος αποθηκεύουμε τα δεδομένα σε ένα αρχείο txt με στήλες τις DOC_ID, WORD_IC, LABEL, OCCURRENCE, εικόνα ΙΙ.25.

```
np.savetxt(TRAINING_DATA_FILE, train_grouped, fmt='%d')
```

```
train_grouped.columns
```

```
Index(['DOC_ID', 'WORD_ID', 'LABEL', 'OCCURENCE'], dtype='object')
```

Εικόνα ΙΙ.25

Ακριβώς την ίδια διαδικασία ακολουθούμε και για τα μέηλ που προορίζονται για δοκιμή του αλγορίθμου (δημιουργία sparse matrix, ομαδοποίηση λέξεων ανά μέηλ, αποθήκευση σε αρχείο txt).

Παρατηρήσεις

Βλέπουμε πως από τα 4057 μέηλ που είχαμε για εκπαίδευση του αλγορίθμου, εν τέλη αποθηκεύτηκαν στο αρχείο txt 4014 μέηλ και από 1739 μέηλ που είχαμε για δοκιμή του αλγορίθμου, αποθηκεύτηκαν στο αρχείο txt 1724, εικόνα ΙΙ.26.

```
train_doc_ids = set(train_grouped.DOC_ID)
test_doc_ids = set(test_grouped.DOC_ID)
```

```
# no of individual mails in training set
len(train_doc_ids)
```

```
4013
```

```
# no of individual mails in testing txt file
len(test_doc_ids)
```

```
1724
```

```
len(X_test)
```

```
1739
```

Εικόνα II.26

Τα τεστ μέηλ που δεν συμπεριλήφθηκαν στο αρχείο txt είναι:

```
# X_test.index stores document ids
set(X_test.index.values) - test_doc_ids
```

```
{134, 179, 240, 274, 298, 339, 439, 471, 670, 734, 765, 945, 1544, 1670, 1700}
```

Εικόνα II.27

Αναζητώντας το 1ο μέηλ με τον κωδικό 134 λαμβάνουμε το παρακάτω αποτέλεσμα:

```
data.MESSAGE[134]
```

```
'------=_NextPart_000_00E8_85C13B1D.B7243B86\n\nContent-Type: text/html; charset="iso-8859-1"\n\nContent-Transfer-Encoding: bas
e64\n\n\n\n\n\nPGh0bWw+DQoNCjxib2R5IGJnY29sb3I9IiNGRkZGRkViIHRleHQ9IiMwMDAw\n\nMDAiPiANCjxwIGFsaWduPSJjZW50ZXIiPjxhIGhyZWY9Imh0
dHA6Ly93d3cu\n\nZGlyZWN0d2Vic3RvcmUuY29tL3RveXMvaW5kZXguaHRtbCI+PGltZyBzcmM9\n\nImh0dHA6Ly93d3cuZGlyZWN0d2Vic3RvcmUuY29tL21waWM
uanBnIiB3aWR0\n\naD0iNTAwIiBoZWlnaHQ9IjMzOSIgYm9yZGVyPSIwIj48L2E+PGJyPiANCjxm\n\nb250IHNpemU9IjMiIGZhY2U9IkFyaWFsLCBIZWx2ZXRpY2
EsIHNhbnMtc2Vy\n\naWYiPjxhIGhyZWY9Imh0dHA6Ly93d3cuZGlyZWN0d2Vic3RvcmUuY29tL3Rv\n\neXMvaW5kZXguaHRtbCI+PGI+RU5URVIgDQpOT1cgaWYge
W91IGFyZSAxOCBh\n\nbmQgb3ZlcjwvYj48L2E+PC9mb250PjwvcD4gDQo8cCBhbGlnbj0iY2VudGVy\n\nIj48Zm9udCBmYWNlPSJBcmlhbCwgSGVsdmV0aWNhLCBz
YW5zLXNlcmlmIiBz\n\naXplPSI0IiBjb2xvcj0iI0ZGMDAwMCI+PGI+U1BFQ0lBTCANCk9GRkVSPGJy\n\nPiANCjxmb250IHNpemU9IjUiPjMwIERheXMgPGk+RlJ
FRSBhY2Nlc3M8L2k+\n\nIDwvZm9udD48YnI+IA0KdG8gdGhlIGxhcmdlc3QgQWR1bHRzaXRlIG9uIHRo\n\nZSB3ZWIuPGJyPiANCjwvYj4gPGZvbnQgc2l6ZT0iMi
I+Zm9yIG9yZGVycyBv\n\ndmVyICQxMDA8L2ZvbnQ+PC9mb250PjwvcD4gDQo8cCBhbGlnbj0iY2VudGVy\n\nIj48Zm9udCBzaXplPSIyIiBmYWNlPSJBcmlhbCwgS
GVsdmV0aWNhLCBzYW5z\n\nLXNlcmlmIj48Yj48Zm9udCBzaXplPSI0IiBjb2xvcj0iIzAwMDBGRiI+UmVh\n\nZHkgDQp0byBnbyBTaG9wcGluZz8gPC9mb250Pjwv
Yj48YnI+IA0KWW91IGNh\n\nbiBmZWVsIHNhZmUgc2hvcHBpbmcgb3VyIHNlY3VyZSBvbmxpbmUgc3RvcmUu\n\nIDxicj4gDQpZb3VyIHByaXZhY3kgYW5kIGNvbmZ
pZGVudGlhbGl0eSBpcyBv\n\ndXIgcHJpb3JpdHkuIDxicj4gDQpBbGwgb3JkZXJzIGFyZSBwYWNrZWQgaW4g\n\nYSBkaXNjcmVldCBwcmlvcml0eSBtYWlsIGJveC
4gPGJyPiANCldlIG5ldmVy\n\nIHNoYXJlIHlvdXIgaW5mb3JtYXRpb24gd2l0aCBhbnlvbmUuIE9yZGVycyBz\n\naGlwcGVkIHdpdGhpbiAyNCBob3Vycy48L2Zvb
nQ+PC9wPiANCg0KPHAgYWxp\n\nZ249ImNlbnRlciI+PGZvbnQgc2l6ZT0iMyIgZmFjZT0iQXJpYWwsIEhlbHZl\n\ndGljYSwgc2Fucy1zZXJpZiI+PGEgaHJlZj0i
aHR0cDovL3d3dy5kaXJlY3R3\n\nZWJzdG9yZS5jb20vdG95cy9pbmRleC5odG1sIj48Yj48Zm9udCBzaXplPSI1\n\nIj5FTlRFUiANCk5PVzwvZm9udD48L2I+PC9
hPjwvZm9udD48L3A+IA0KPHAg\n\nYWxpZ249ImNlbnRlciI+Jm5ic3A7PC9wPiANCjxwIGFsaWduPSJjZW50ZXIi\n\nPiZuYnNwOzwvcD4gDQo8cCBhbGlnbj0iY2
VudGVyIj48Zm9udCBzaXplPSIx\n\nIiBmYWNlPSJBcmlhbCwgSGVsdmV0aWNhLCBzYW5zLXNlcmlmIj5UaGlzIEUt\n\nbWFpbGluZyANCmhhcyBiZWVuIHNlbnQgd
G8geW91IGFzIGEgcGVyc29uIGlu\n\ndGVyZXN0ZWQgaW4gdGhlIGluZm9ybWF0aW9uIGVuY2xvc2VkLiA8YnI+IA0K\n\nSWYgdGhpcyByZWFjaGVkIHlvdSBieSBl
cnJvciwgb3IgeW91IGRvIG5vdCB3\n\naayBvbiB0aGUgd29yZDxhIGhyZWY9Im1haWx0bzptaW5pcmVtQGZpcmVtYWls\n\nLmRlIj4gUkVNT1ZFIDwvYT5hbmQgDQ
pyZXBseSB0aGlzIGVtYWlsIHRvIHVz\n\nLCB5b3Ugd2lsbCBiZSB0YWtlbiBvZmYgb3VyIGxpc3QgaW1tZWRpYXRlbHkg\n\nYW5kIE5FVkVSIA0KcmVjZWl2ZSBhb
nkgZW1haWxzIGZyb20gdXMuPC9mb250\n\nPjwvcD4gDQo8cCBhbGlnbj0iY2VudGVyIj4mbmJzcDs8L3A+IA0KPHAgYWxp\n\nZ249ImNlbnRlciI+Jm5ic3A7PC9w
PiANCjxwIGFsaWduPSJjZW50ZXIiPiZu\n\nYnNwOzwvcD4gDQo8cCBhbGlnbj0iY2VudGVyIj4mbmJzcDs8L3A+IA0KPHAg\n\nYWxpZ249ImNlbnRlciI+Jm5ic3A
7PC9wPiANCjwvYm9keT4gDQo8L2h0bWw+\n\nDQoxMzkzUHdxYjAtMjYxU0JjSDAzNTdtaEdxMS0xMDN1VHVQOTA3NkFyd3gw\n\nLTI1Nm5ad2M0NjA2aWtDSzUtND
UyS2VzdzM1NjhFT29JNC00NjFQbDc3\n\n\n'
```

Εικόνα II.28

Για να αποκλείσουμε την περίπτωση να υπάρχει πρόβλημα με το encoding αναζητούμε το μέηλ στα αρχεία μας και λαμβάνουμε το αποτέλεσμα:

```
data.loc[134]
```

```
MESSAGE      ------=_NextPart_000_00E8_85C13B1D.B7243B86\n\...
CATEGORY                                                    1
FILE_NAME              00135.00e388e3b23df6278a8845047ca25160
Name: 134, dtype: object
```

Εικόνα II.29

```
File  Edit  Format  View  Help
    with asmtp; 28 Aug 0102 04:06:50 -0900
Reply-To: <new_adult_toys_0463b54@yahoo.com>
Message-Id: <014d73e75c7b$6247c3e0$8ce81ed4@ywdmjl>
From: <new_adult_toys_0463b54@yahoo.com>
To: <webmaster@efi.ie>
Subject: online catalog of articles, adult toys 2009vjtt6-179dML-15
Date: Tue, 27 Aug 0102 20:59:44 -0200
MIME-Version: 1.0
X-Priority: 3 (Normal)
X-Msmail-Priority: Normal
X-Mailer: QUALCOMM Windows Eudora Version 5.1
Importance: Normal
Content-Type: multipart/mixed; boundary="----=_NextPart_000_00E8_85C

------=_NextPart_000_00E8_85C13B1D.B7243B86
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: base64


PGh0bWw+DQoNCjxib2R5IGJnY29sb3I9IiNGRkZGRkYiIHRleHQ9IiMwMDAw
MDAiPiANCjxwIGFsaWduPSJjZW50ZXIiPjxhIGhyZWY9Imh0dHA6Ly93d3cu
ZGlyZWN0d2Vic3RvcmUuY29tL3RveXMvaW5kZXguaHRtbCI+PGltZyBzcmM9
Imh0dHA6Ly93d3cuZGlyZWN0d2Vic3RvcmUuY29tL21waWMuanBnIiB3aWR0
aD0iNTAwIiBoZWlnaHQ9IjMzOSIgYm9yZGVyPSIwIj48L2E+PGJyPiANCjxm
b250IHNpemU9IjMiIGZhY2U9IkFyaWFsLCBIZWx2ZXRpY2EsIHNhbnMtc2Vy
aWYiPjxhIGhyZWY9Imh0dHA6Ly93d3cuZGlyZWN0d2Vic3RvcmUuY29tL3Rv
eXMvaW5kZXguaHRtbCI+PGI+RU5URVIgDQpOT1cgaWYgeW91IGFyZSAxOCBh
bmQgb3ZlcjwvYj48L2E+PC9mb250PiANCjwvYm9keT4NCjwvaHRtbD4NCg==
```

Εικόνα II.30

Παρατηρούμε πως και το ίδιο το σώμα του μέηλ έχει ένα προβληματικό κείμενο και τέλος περνώντας το συγκεκριμένο μέηλ από τη συνάρτηση clean_msg_no_html() έχουμε ως αποτέλεσμα μια κενή λίστα,

εικόνα ΙΙ.31, η οποία δεν αντιστοιχεί σε καμία λέξη του λεξικού των 2500 λέξεων που κατασκευάσαμε και έτσι το μέηλ αυτό δε συμπεριλαμβάνεται στα αποτελέσματα της sparce matrix.

```
clean_msg_no_html(data.at[134, 'MESSAGE'])
[]
```

Εικόνα ΙΙ.31

Παρόμοια αποτελέσματα λαμβάνουμε και για τα άλλα μέηλ που εν τέλει δε συμπεριλήφθηκαν στα αρχεία txt.

5.5 Εκπαίδευση Αλγορίθμου

(χρήση αρχείου 2 Bayes Classifier – Training.ipynb)

Δημιουργούμε μια νέα πλήρη μήτρα (full matrix) με δεδομένα που θα τροφοδοτήσουμε στον αλγόριθμο. Συγκεκριμένα η πλήρης μήτρα θα έχει για γραμμές τα μέηλ της εκπαίδευσης, δηλαδή 4013 και για στήλες τις:

- Αύξοντας αριθμός του μέηλ εκπαίδευσης, DOC_ID
- Κατηγορία μέηλ, 0 για μη σπαμ, 1 για σπαμ
- Κάθε κωδικός (0 έως 2499) των 2500 λέξεων του λεξικού μας θα είναι μία στήλη. Τα στοιχεία της κάθε στήλης φανερώνουν το πόσες φορές εμφανίζεται η κάθε λέξη στο συγκεκριμένο μέηλ.

```
def make_full_matrix(sparse_matrix, nr_words, doc_idx=0, word_idx=1, cat_idx=2, freq_idx=3):
    """
    Form a full matrix from a sparse matrix. Return a pandas dataframe.
    Keyword arguments:
    sparse_matrix -- numpy array
    nr_words -- size of the vocabulary. Total number of tokens.
    doc_idx -- position of the document id in the sparse matrix. Default: 1st column.
    word_idx -- position of the word id in te sparse matrix. Default: 2nd column.
    cat_idx -- position of the label (spam is 1, nonspam is 0). Default: 3rd column.
    freq_idx -- position of occurene of word in sparse matrix. Default: 4th column.
    """
    # start with empty df
    column_names = ['DOC_ID'] + ['CATEGORY'] + list(range(0, VOCAB_SIZE))
    doc_id_names = np.unique(sparse_matrix[:, 0])
    full_matrix = pd.DataFrame(index=doc_id_names, columns=column_names)
    full_matrix.fillna(value=0, inplace=True)

    for i in range(sparse_matrix.shape[0]):
        doc_nr = sparse_matrix[i][doc_idx]
        word_id = sparse_matrix[i][word_idx]
        label = sparse_matrix[i][cat_idx]
        occurrence = sparse_matrix[i][freq_idx]

        #populate matrix with data
        # row no: doc_nr, col: 'DOC_ID' etc
        full_matrix.at[doc_nr, 'DOC_ID'] = doc_nr
        full_matrix.at[doc_nr, 'CATEGORY'] = label
        full_matrix.at[doc_nr, word_id] = occurrence

    # set index
    full_matrix.set_index('DOC_ID', inplace=True)
    return full_matrix
```

Εικόνα II.32

```
%%time
full_train_data = make_full_matrix(sparse_train_data, VOCAB_SIZE)
```

```
CPU times: total: 26.8 s
Wall time: 30 s
```

```
full_train_data
```

DOC_ID	CATEGORY	0	1	2	3	4	5	6	7	8	...	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
0	1	0	0	1	2	1	0	0	3	0	...	0	0	0	0	0	0	0	0	0	0
1	1	7	1	2	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
2	1	6	1	1	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
3	1	6	0	0	2	4	0	3	14	0	...	0	0	0	0	0	0	0	0	0	0
4	1	5	1	2	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5789	0	3	1	0	1	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
5790	0	1	1	1	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
5791	0	3	1	0	1	1	0	0	0	1	...	1	0	0	0	0	0	0	0	0	0
5794	0	1	1	1	0	0	1	2	0	0	...	0	0	0	0	0	0	0	0	0	0
5795	0	3	4	2	0	5	0	3	0	0	...	0	0	0	0	0	0	0	0	0	0

4013 rows × 2501 columns

Εικόνα II.33

Για την εκπαίδευση του αλγορίθμου θα χρειαστούμε τις πιθανότητες της κάθε λέξης (token probabilities).

Παραδείγματος χάριν, η δεσμευμένη πιθανότητα (conditional probability) ένα μέηλ να είναι σπαμ υπό την προϋπόθεση ότι εμπεριέχει τη λέξη βιάγκρα υπολογίζεται όπως έχουμε αναφέρει ως εξής:

$$P(Spam|Viagra) = P(Viagra|Spam) * P(Spam) / P(Viagra)$$

Αποδομώντας αυτόν τον τύπο έχουμε:

P(Spam): Η πιθανότητα ενός μέηλ να είναι σπαμ

P(Viagra): Η πιθανότητα η λέξη βιάγκρα να εμφανίζεται σε οποιοδήποτε μέηλ. Αν για παράδειγμα η λέξη βιάγκρα εμφανίζεται 75 φορές σε όλα τα μέηλ (σπαμ και μη) που εξετάζουμε και οι συνολικές λέξεις σε όλα τα μέηλ είναι 700.000 τότε η πιθανότητα είναι 75 / 700.000

P(Viagra|Spam): Η πιθανότητα ένα μέηλ να έχει τη λέξη βιάγκρα δεδομένου ότι είναι σπαμ. Αν για παράδειγμα η λέξη βιάγκρα εμφανίζεται 65 φορές στα σπαμ μέηλ και αυτά τα σπαμ μέηλ έχουν 370.000 λέξεις, αυτή η πιθανότητα είναι 65 / 370.000

$$P(Spam \mid Viagra) = \frac{P(Viagra \mid Spam)\, P(Spam)}{P(Viagra)}$$

Εμφάνιση σε σπαμ Όλες οι λέξεις των σπαμ

$$P(Spam \mid Viagra) = \frac{65 / 370{,}000 \qquad 0.55}{75 / 750{,}000}$$

Εμφάνιση σε όλα Όλες οι λέξεις όλων των μέηλ

Υπολογίζουμε την πιθανότητα ενός μέηλ (από τα μέηλ εκπαίδευσης) να είναι σπαμ η οποία είναι περίπου 31.1 %, εικόνα II.34.

```
# total no of mails
full_train_data.CATEGORY.size
```

```
4013
```

```
# total no of spams, non spam have category 0, spam have category 1
full_train_data.CATEGORY.sum()
```

```
1248
```

```
prob_spam = full_train_data.CATEGORY.sum() / full_train_data.CATEGORY.size
print(f"Probability of spam is: {prob_spam}")
```

```
Probability of spam is: 0.310989284824321
```

Εικόνα III.34

Επίσης υπολογίζουμε τον αριθμό των σπαμ μέηλ, των tokens στα σπαμ μέηλ, τα μη σπαμ μέηλ – χαμ και τον αριθμό των tokens στα μη σπαμ μέηλ ελέγχοντας παράλληλα ότι σπαμ + χαμ μέηλ = αριθμός μέηλ προς εκπαίδευση και πως spam tokens + ham tokens = tokens in all training mails:

```
spam_wc = spam_lengths.sum()
print(f"No of tokens in spam mails: {spam_wc}")
```

```
No of tokens in spam mails: 176318
```

```
ham_lengths = email_lengths[full_train_data.CATEGORY == 0]
print(f"No of ham mails: {ham_lengths.shape}")
```

```
No of ham mails: (2765,)
```

```
# check subsets are correct
email_lengths.shape[0] - spam_lengths.shape[0] - ham_lengths.shape[0]
```

```
0
```

```
nonspam_wc = ham_lengths.sum()
print(f"No of tokens in ham mails: {nonspam_wc}")
```

```
No of tokens in ham mails: 252866
```

```
# check
spam_wc + nonspam_wc == total_wc
```

```
True
```

Εικόνα II.35

Αθροίζουμε τα tokens που υπάρχουν στα σπαμ μέηλ, εικόνα II.36 και επαναλαμβάνουμε την ίδια διαδικασία για τα tokens στα μη σπαμ μέηλ.

```
# subset of all rows corresponding to spam messages
train_spam_tokens = full_train_features.loc[full_train_data.CATEGORY == 1]
train_spam_tokens
```

DOC_ID	0	1	2	3	4	5	6	7	8	9	...	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
0	0	0	1	2	1	0	0	3	0	0	...	0	0	0	0	0	0	0	0	0	0
1	7	1	2	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
2	6	1	1	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
3	6	0	0	2	4	0	3	14	0	0	...	0	0	0	0	0	0	0	0	0	0
4	5	1	2	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1885	1	0	0	2	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1887	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1889	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
1890	2	0	0	0	1	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
1895	1	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1248 rows × 2500 columns

Εικόνα III.36

```
# sum tokens in spam mails col by col
# a series of all word ids with no of times these tokens occur in spam messages
# Laplace smoothing: adding 1 to avoid 0 division
summed_spam_tokens = train_spam_tokens.sum(axis=0) + 1
summed_spam_tokens
# word with id 0 occurs 2179 times in all spam messages
```

```
0       2179
1        935
2       1217
3       2022
4       1219
       ...
2495       8
2496       1
2497       2
2498       1
2499      26
Length: 2500, dtype: int64
```

Εικόνα II.37

Υπολογίζουμε την πιθανότητα ένα τόκεν να εμφανιστεί σε ένα μέηλ δεδομένου ότι το μέηλ αυτό είναι σπαμ (το άθροισμα των πιθανοτήτων είναι 1), εικόνα ΙΙ.38.

```
# do this for all tokens simultaneously, add VOCAB_SIZE to implement Laplace smoothing technique
prob_tokens_spam = summed_spam_tokens / (spam_wc + VOCAB_SIZE)
prob_tokens_spam
```

```
0       0.012186
1       0.005229
2       0.006806
3       0.011308
4       0.006817
          ...
2495    0.000045
2496    0.000006
2497    0.000011
2498    0.000006
2499    0.000145
Length: 2500, dtype: float64
```

```
# check if above probabilities sum up to 1
prob_tokens_spam.sum()
```

```
1.0
```

Εικόνα ΙΙ.38

Υπολογίζουμε την πιθανότητα ενός τόκεν να εμφανιστεί υπό την προϋπόθεση ότι το μέηλ είναι μη σπαμ (το άθροισμα των πιθανοτήτων είναι 1), εικόνα ΙΙ.39:

```
prob_tokens_nonspam = summed_ham_tokens / (nonspam_wc + VOCAB_SIZE)
prob_tokens_nonspam
```

```
0       0.021475
1       0.010142
2       0.008008
3       0.003673
4       0.006313
          ...
2495    0.000106
2496    0.000023
2497    0.000110
2498    0.000098
2499    0.000012
Length: 2500, dtype: float64
```

```
# check if above probabilities sum up to 1
prob_tokens_nonspam.sum()
```

```
1.0
```

Εικόνα ΙΙ.39

Υπολογίζουμε την πιθανότητα ένα τόκεν να εμφανιστεί (είτε σε σπαμ είτε σε μη σπαμ μέηλ, το άθροισμα των πιθανοτήτων πρέπει να είναι 1), εικόνα ΙΙ.40:

P(Token) - Probability Token Occurs (in spam or ham mails)

```
prob_tokens_all = full_train_features.sum(axis=0) / total_wc
prob_tokens_all
```

```
0       0.017850
1       0.008209
2       0.007596
3       0.006892
4       0.006592
          ...
2495    0.000077
2496    0.000012
2497    0.000065
2498    0.000056
2499    0.000063
Length: 2500, dtype: float64
```

```
# check if above probabilites sum up to 1
prob_tokens_all.sum()
```

```
1.0
```

Εικόνα ΙΙ.40

Δημιουργούμε μια πλήρη μήτρα (full matrix) για τα δεδομένα δοκιμής (test data) του αλγορίθμου, εικόνα ΙΙ.41:

```
X_test = full_test_data.loc[:, full_test_data.columns != 'CATEGORY']
X_test
```

DOC_ID	0	1	2	3	4	5	6	7	8	9	...	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
8	0	0	1	4	2	1	2	4	1	2	...	0	0	0	0	0	0	0	0	0	0
12	6	1	1	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
14	0	0	1	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
15	0	2	1	1	2	0	0	3	0	4	...	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	1	0	1	0	0	0	...	0	0	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5783	2	1	0	2	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5786	5	5	2	2	1	2	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
5788	0	4	0	2	4	3	3	1	4	3	...	0	0	0	0	0	0	0	0	0	0
5792	2	2	0	1	0	0	2	1	0	0	...	0	0	0	0	0	0	0	0	0	0
5793	1	9	1	0	0	1	2	1	1	1	...	0	0	0	0	0	0	0	0	0	0

1724 rows × 2500 columns

```
y_test = full_test_data.CATEGORY
y_test
```

Εικόνα ΙΙ.42

(αρχείο 03 Bayes Classifier – Testing, Inference & Evaluation.ipynb)

Δοκιμή – Τεστ Αλγορίθμου

Όπως αναφέραμε η συνδυασμένη πιθανότητα (joint probability) δύο ανεξάρτητων γεγονότων A και B υπολογίζεται ως εξής: P(A ∩ B) = P(A)*P(B).

Την έννοια της ανεξαρτησίας θα χρησιμοποιήσουμε και στον αλγόριθμό μας. Αν για παράδειγμα ένα μέηλ έχει 2 τόκενς, το βιάγκρα και δωρεάν, τότε υπολογίζουμε την πιθανότητα το μέηλ να είναι σπαμ ως το γινόμενο των συνδυασμένων πιθανοτήτων: P(Spam | Viagra) x P(Spam | Free)

Στους υπολογισμούς μας θα χρησιμοποιήσουμε τους λογαρίθμους των πιθανοτήτων ώστε να απλοποιήσουμε τις υπολογιστικές πράξεις και έτσι οι ζητούμενες πιθανότητες μετατρέπονται σε:

$$P(Spam \mid X) = \frac{P(X \mid Spam)\, P(Spam)}{P(X)}$$

$$log(P(Tokens \mid Spam)) - log(P(Tokens)) + log(P(Spam))$$

```
joint_log_spam = X_test.dot(np.log(prob_token_spam) - np.log(prob_all_tokens)) + np.log(PROB_SPAM)
joint_log_spam
```

```
array([  24.27388126,    2.15807202,   20.58882974, ..., -374.67704567,
         -9.9043877 , -112.029553  ])
```

Εικόνα II.43

$$P(Ham \mid X) = \frac{P(X \mid Ham)\,(1 - P(Spam))}{P(X)}$$

$$log(P(Tokens \mid Spam)) - log(P(Tokens)) + log(1 - P(Spam))$$

```
joint_log_ham = X_test.dot(np.log(prob_token_ham) - np.log(prob_all_tokens)) + np.log(1-PROB_SPAM)
joint_log_ham
```

```
array([  25.06932962,    2.95352038,   21.3842781 , ..., -373.88159731,
         -9.10893934, -111.23410464])
```

Εικόνα II.44

Οι προβλέψεις μας θα βασιστούν στη σύγκριση των 2 άνωθι πιθανοτήτων και θα κατηγοριοποιούμε κάθε μέηλ με βάση ποια πιθανότητα έχει τη μεγαλύτερη τιμή.

$$P(Spam \mid X) > P(Ham \mid X)$$

OR

$$P(Spam \mid X) < P(Ham \mid X)$$

```
# vector of preditions
# y_hat for spam should be 1 and 0 for non spam
prediction = joint_log_spam > joint_log_ham
prediction
# false signifies non spam emails
```

```
array([False, False, False, ..., False, False, False])
```

```
y_test
```

```
array([1., 1., 1., ..., 0., 0., 0.])
```

Εικόνα II.45

Από την παραπάνω εικόνα, εικόνα II.45, παρατηρούμε πως οι 3 τελευταίες τιμές είναι False, συνεπώς τα μέηλ δεν είναι σπαμ και σε αυτές τις 3 δοκιμές ο αλγόριθμος αποφάσισε σωστά καθώς τα μέηλ αυτά έχουν είναι στην κατηγορία 0 (y_test) την οποία έχουν τα μη σπαμ μέηλ.

5. 6 Αξιολόγηση Αποτελέσματος

Για να έχουμε ένα μοντέλο που λειτουργεί σωστά, θα πρέπει αυτό το μοντέλο να μπορεί να χαρακτηρίζει τα σπαμ μέηλ ως σπαμ, τα μη σπαμ ως μη σπαμ και ιδανικά αυτό επιθυμούμε να γίνεται σωστά σε κάθε μέηλ.

Αρχικά είναι σημαντικό να γνωρίζουμε πόσα μέηλ κατάφερε να κατηγοριοποιήσει ορθά ο αλγόριθμος. Για αυτό θα συγκρίνουμε τις πραγματικές τιμές με αυτές που πρόβλεψε ο αλγόριθμος.

Ο αλγόριθμος κατέταξε σωστά 1685 μέηλ και λάθος 39 μέηλ, εικόνα II.46.

```
correct_docs = (y_test == prediction).sum()
print('Docs classified correctly', correct_docs)
numdocs_wrong = X_test.shape[0] - correct_docs
print('Docs classified incorrectly', numdocs_wrong)
```

```
Docs classified correctly 1685
Docs classified incorrectly 39
```

Εικόνα II.46

Για να μετρήσουμε την ακρίβεια του αλγορίθμου θα χρησιμοποιήσουμε το μέτρο: Αριθμός Ορθών Προβλέψεων / Συνολικός Αριθμός Προβλέψεων. Η ακρίβεια του μοντέλου μας είναι περίπου 97.7%, εικόνα II.47.

```
# Accuracy
correct_docs/len(X_test)
```

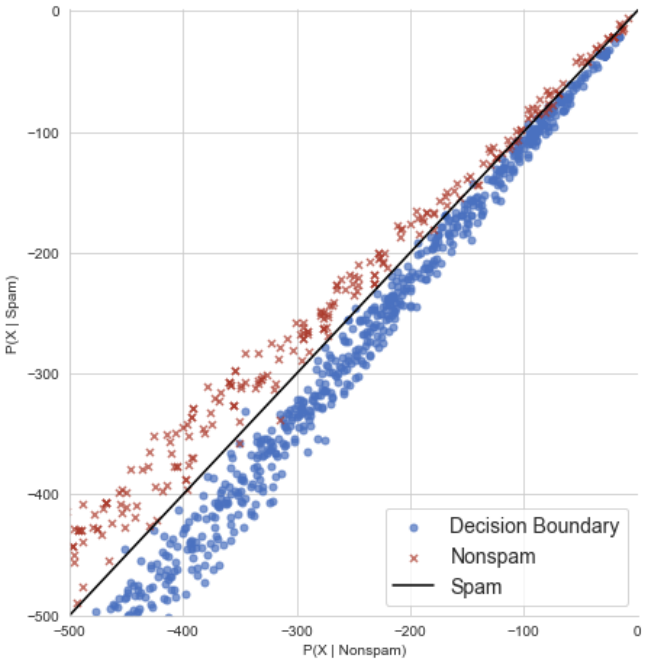
```
0.9773781902552204
```

```
fraction_wrong = numdocs_wrong/len(X_test)
print('Fraction classified incorrectly is {:.2%}'.format(fraction_wrong))
print('Accuracy of the model is {:.2%}'.format(1-fraction_wrong))
```

```
Fraction classified incorrectly is 2.26%
Accuracy of the model is 97.74%
```

Εικόνα II.47

Στο παρακάτω σχεδιάγραμμα, εικόνα II.48, απεικονίζεται η πιθανότητα ένα τόκεν X να εμφανίζεται υπό την προϋπόθεση ένα μέηλ να είναι σπαμ σε αντιπαράθεση με την πιθανότητα ένα τόκεν X να εμφανίζεται υπό την προϋπόθεση ένα μέηλ να είναι επιθυμητό (τα αποτελέσματα έχουν λογαριθμηθεί).



Εικόνα II.48

5.6.1 Σωστά Θετικά – Λάθος Θετικά – Σωστά Αρνητικά – Λάθος Αρνητικά

Στο σημείο αυτό πρέπει να αναφερθούμε στο ζήτημα των σωστών και λάθος θετικών αποτελεσμάτων και σωστών και λάθος αρνητικών αποτελεσμάτων.

Στο παράδειγμά μας ισχύουν τα εξής:

Σωστά Θετικά – True Positives: Τα μέηλ που είναι σπαμ και έχουν κατηγοριοποιηθεί από τον αλγόριθμο ως σπαμ. Η έννοια του θετικού αποτελέσματος για τον αλγόριθμο, σημαίνει κατηγοριοποίηση ενός μέηλ ως σπαμ.

Λάθος Θετικά – False Negatives: Τα μέηλ που δεν είναι σπαμ και έχουν κατηγοριοποιηθεί από τον αλγόριθμο ως σπαμ. Αυτός είναι ο λόγος που κατά καιρούς χρειάζεται να ελέγχουμε τον κατάλογο σπαμ μέηλ μας καθότι κάποιες φορές μπαίνουν εκεί κατά λάθος και επιθυμητά μέηλ.

Σωστά Αρνητικά – True Negatives: Τα μέηλ που δεν είναι σπαμ και έχουν κατηγοριοποιηθεί ως μη σπαμ.

Λάθος Αρνητικά – Wrong Negatives: Τα μέηλ που είναι σπαμ και έχουν κατηγοριοποιηθεί ως μη σπαμ. Πρόκειται για την περίπτωση που σπαμ μέηλ βρίσκονται στον φάκελο inbox των επιθυμητών μέηλ.

Στη δική μας περίπτωση ο αλγόριθμος πρόβλεψε 1136 μη σπαμ μέηλ και 588 σπαμ μέηλ, εικόνα II.49.

```
# non spam 1136, spam 588
np.unique(prediction, return_counts=True)
```

```
(array([False,  True]), array([1136,  588], dtype=int64))
```

Εικόνα II.49

Τα σωστά θετικά, δηλαδή τα μέηλ που είναι σπαμ και κατηγοριοποιήθηκαν ως σπαμ, ήταν 569, εικόνα II.50.

```
# single & for element by element comparison
true_pos = (y_test == 1) & (prediction == 1)
```

```
true_pos.sum()
```

```
569
```

```
true_pos
```

```
array([ True,  True,  True, ..., False, False, False])
```

Εικόνα II.50

Ομοίως τα λάθος θετικά ήταν 19 και τα λάθος αρνητικά 20, εικόνα II.51.

```
false_pos = (y_test == 0) & (prediction == 1)
false_pos.sum()
```

```
19
```

```
false_neg = (y_test == 1) & (prediction == 0)
false_neg.sum()
```

```
20
```

Εικόνα II.51

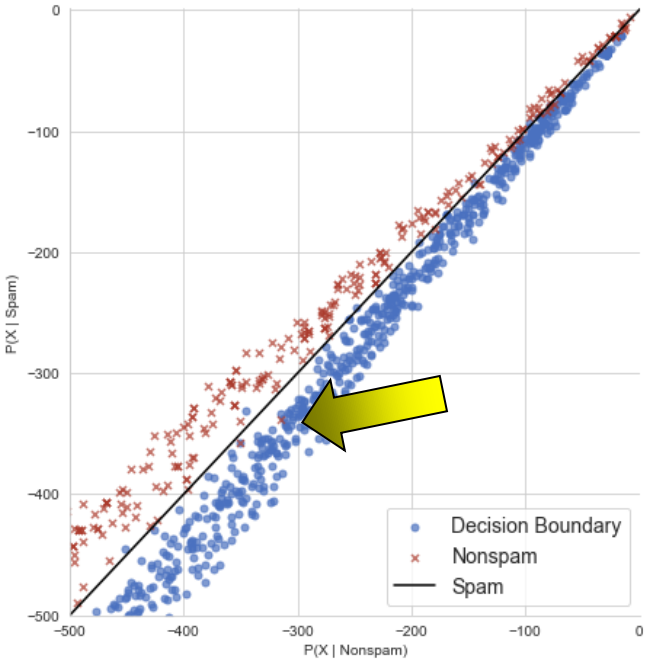
5.6.2 Δείκτης Recall

Ένας ακόμα δείκτης μέτρησης της απόδοσης του μοντέλου είναι το recall score το οποίο υπολογίζεται ως εξής:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Παρατηρούμε πως το μέγεθος των Λάθος Αρνητικών (το μέηλ κατηγοριοποιήθηκε ως μη σπαμ ενώ ήταν σπαμ, δηλαδή το μέηλ κατάφερε να περάσει στο inbox μας) έχει καθοριστική σημασία μιας και όταν γίνεται 0 τότε το κλάσμα παίρνει την τιμή 1 το οποίο είναι και η μέγιστη τιμή του recall score. Η ερμηνεία αυτού του δείκτη μπορεί να δοθεί μέσα από την ερώτηση: Από όλα τα σπαμ μέηλ, πόσα στην πραγματικότητα χαρακτήρισε το μοντέλο ως σπαμ; Ουσιαστικά είναι η αναλογία των σπαμ μέηλ που κατηγοριοποιήθηκαν σωστά (true positives) προς όλα τα σπαμ μέηλ που είχαμε στα δεδομένα μας (true positives & false negatives μιας και τα λάθος αρνητικά είναι και αυτά σπαμ μέηλ στην πραγματικότητα). Ο δείκτης προσπαθεί να μετρήσει την ικανότητα του μοντέλου μας να βρίσκει τα σημεία που μας ενδιαφέρουν, εν προκειμένω τα σπαμ μέηλ.

Τα κόκκινα σημεία που είναι κάτω από το όριο απόφασης στην εικόνα ΙΙ.52 είναι τα λάθος αρνητικά. Είναι σπαμ μέηλ που το μοντέλο τοποθέτησε κάτω από το όριο απόφασης, δηλαδή στην περιοχή των μη σπαμ μέηλ. Ο δείκτης recall μας βοηθά να δούμε πόσα σπαμ μέηλ πέτυχε ο αλγόριθμος σε σχέση με το πόσα έχασε.



Εικόνα ΙΙ.52

Εν προκειμένω ο αλγόριθμος πέτυχε ένα recall score της τάξης του 96.60%, εικόνα ΙΙ.53.

```
recall_score = true_pos.sum() / (true_pos.sum() + false_neg.sum())
print('Recall score is {:.2%}'.format(recall_score))
```

```
Recall score is 96.60%
```

Εικόνα II.53

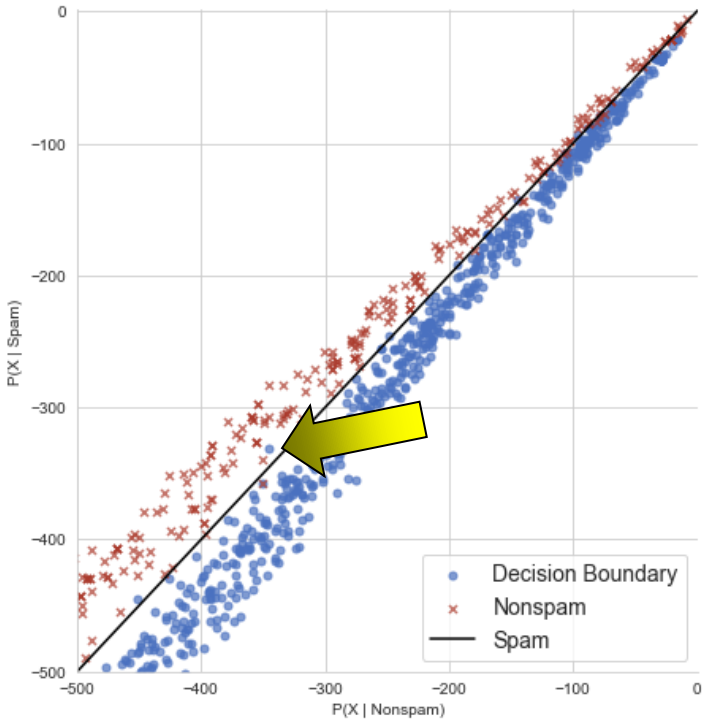
Ένα αρνητικό του δείκτη είναι ότι μπορούμε να χειραγωγήσουμε εύκολα το αποτέλεσμα. Αν για παράδειγμα χαρακτηρίσουμε όλα τα μέηλ ως σπαμ τότε δεν έχουμε καθόλου λάθος αρνητικά και το κλάσμα γίνεται 1 δίνοντας στο δείκτη τη μέγιστή του τιμή.

5.6.3 Δείκτης Precision – Positive Predicted Value

Ο συγκεκριμένος δείκτης ορίζεται ως εξής:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Πρόκειται για την αναλογία των σωστών θετικών, των σπαμ μέηλ που κατηγοριοποιήθηκαν ως σπαμ, προς το συνολικό αριθμό μέηλ που κατηγοριοποιήθηκαν ως σπαμ, είτε ήταν σωστή είτε λάθος η κατηρογοριοποίηση (αληθινά και λάθος θετικά). Τα λάθος θετικά είναι τα γαλάζια σημεία της εικόνας II.54, που βρίσκονται πάνω από το όριο απόφασης. Η τιμή αυτού του δείκτη μεγαλώνει όταν μικραίνει το μέγεθος των λάθος θετικών αποτελεσμάτων του παρονομαστή.



Εικόνα II.54

Ο εν λόγω δείκτης στο μοντέλο είναι 0.968, εικόνα II.55.

```
precision_score = true_pos.sum() / (true_pos.sum() + false_pos.sum())
print('Precision score is {:.3}'.format(precision_score))
```

```
Precision score is 0.968
```

Εικόνα II.55

5.6.4 Δείκτης F Score – F1 Score

Αυτός ο δείκτης συνδυάζει τους δείκτες recall και precision, οπότε λαμβάνει υπόψιν και τα λάθος θετικά και τα λάθος αρνητικά, και υπολογίζεται ως εξής:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Λαμβάνουμε μια τιμή 0.97 για το F Score, η οποία είναι υψηλή μιας και η τιμή του κυμαίνεται μεταξύ 0 και 1, εικόνα II.56.

```
f1_score = 2 * (precision_score * recall_score) / (precision_score + recall_score)
print('F Score is {:.2}'.format(f1_score))
```

```
F Score is 0.97
```

Εικόνα II.56

Μετά την εξέταση αυτών των 3 δεικτών και με δεδομένο το γεγονός ότι ο αριθμός των λάθος θετικών και λάθος αρνητικών αποτελεσμάτων είναι χαμηλός το αποτέλεσμα χαρακτηρίζεται ως θετικό για το μοντέλο. Βέβαια το μοντέλο επιδέχεται βελτιώσεις. Για παράδειγμα τα σύγχρονα φίλτρα μέηλ εξετάζουν τις κεφαλίδες και τις διευθύνσεις των μέηλ.

Βιβλιογραφία

Anderson, J. A. (1986). “Cognitive Capabilities of a Parallel System”. In E. Bienenstock, F. Fogelman-Souli, & G. Weisbuch, eds,. *Disordered Systems and Biological Organisation*, NATO ASI Series, F20, Berlin: Springe – Verlag

Anderson, J. A., R. . GOLDEN, & G. L. MURPHY. (1986). “Concepts in Distribited Systems.” In H. H. Su, ed., *Optical and Hybrid Computing*, 640: 260 – 272, Bellington, WA: Society of Photo-Optical Instrumentation Engineers.

Baeldung. (2022, June 20). *The Difference Between Epoch and Iteration in Neural Networks.* https://www.baeldung.com/cs/neural-networks-epoch-vs-iteration

Ciresan, C., D., Meier, U., Gambardella, L., M., Schmidhuber, J. (2010). Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*. Vol.22, Issue 12, p. 3207-3220

Claesen, M., De Moor, B. (2015). Hyperparameter Search in Machine Learning. *MIC 2015: The XI Metaheuristics International Conference*

Dan, C., Meier, U., Masci, J., Gambardella, L., M., Schmidhuber, J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification*. Proceeding of the Twenty-Second International Joint Conference on Artificial Intelligence. Vol. 2, 1237 - 1242*

Doshi, S. (2019). *Various Optimization Algorithms For Training Neural Network*. Towards Data Science. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6#:~:text=Many%20people%20may%20be%20using,help%20to%20get%20results%20faster

Dong, X., Zhou, D., X. (2007). Learning gradients by a gradient descent algorithm. *Journal of Mathematical Analysis and Applications* 341 (2008) 1018–1027

Dupond, S. (2019). A Thorough Review of the Current Advance of Neural Network Structures. *Annual Revies in Control*, 14: 200-230

Ejaz, S., Islam, R. (2019). Masked Face Recognition Using Convolutional Neural Network. *Conference: 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*

Faucett, L. (1993). Fundamentals of Neural Networks: Architectures, Algorithms And Applications. Pearson

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. AI Magazine, volume 17

Geron, Aurelien. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow, Concepts, Tools and Techniques to Build Intelligent Systems*. O’ Reilly

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press

Goodfellow, I., Abadie, J., P., Mirza, M., Xu, B., Farley, D., W., Ozair, S., Courville, A., Bengio. Y. D. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27* (NIPS 2014)

Hastie, T., Tibshirani, R., Jerome, Friedman. (2009). *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer

Hayking, S. (2009). *Neural networks and learning machines* (3rd ed.). Pearson education.

He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep Residual Learning for Image Recognition. Microsoft Research

Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation* 9(8): 1735-1780

Hopfield, J., J. (1982). Neural networks and physical systems with emergent collective computational abilities*. Proceedings of the National Academy of Sciences*, 79, 2554

IBM. (2022). *Computer Vision Examples*. https://www.ibm.com/topics/computer-vision

Johnson, R., C. (1989). Neural Nose to Sniff Out Explosives at JFK Airport. *Electronic Engineering Times* 536,1

Le Cun, Y. (1990). "Handwritten Digit Recognition with a Backpropagation Network" In D. S. Touretsky, ed., *Advances in Neural Information Processing Systems* 2. San Mateo, CA: Morgan Kaufman, pp. 396-404.

Karras, T., Aila, T., Laine, S., Lehtinen, J. (2018). Progressive Growing of GANs for Improved Quality, Stability and Variation. ICLR 2018

Kingma, D., P., Lei Ba, J. (2015). Adam: A Method for Stochastic Optimization. *Published as a conference paper at the 3rd International Conference for Learning Representations*

Lippmann, R. P. (1989). Review of Neural Networks for Speech Recognition. *Neural Computation*, 1:1-38.

McClelland, J., L., Rumelhart, D., E.)1986). Parallel Distributed Processing, Volumes 1 and 2. MIT Press, Cambridge, Mass

McCord – Nelson M., W. T. Illingworth. (1991). *A practical guide to Neural Nets.* Addison–Wesley Longman Publishing.

McCulloch, W., S., Pitts, W. (1943). A logical calculus of ideas immanent in nervous Activity. *Bulletin of Mathematical Biophysics*, 5**,** 115

McCulloch, W., Pitts, W.(1947). How we know universals the perception of auditory and visual forms. Bulletin of Mathematical Biophysics, 9, 127

Miller, W. T., R. S. Sutton, & P. J. Werbros, eds. (1990). *Neural Networks for Control*, Cambridge, MA: MIT Press.

Miljanovic, M. (2012). Comparative Analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction. Indian Journal of Computer and Engineering 3 (1)

Minsky, M., Papert, A,. S. (1969) Perceptrons, An Introduction to Computational Geometry. MIT Press
Nguyen, D., & B. Widrow. (1989). "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks." *International Joint Conference on Neural Networks*, Washington, DC, ii: 357 – 363.

Muller, A., Guido, S. (2016). *Introduction to Machine Learning with Python, A Guide for Data Scientists*. O' Reilly Media

Opela, M., Shindler, I., Kawulok, P., Kawolok, R., Rusz, S., Sauer, M. (2002). Shallow and deep learning of an artificial neural network model describing flow stress Evolution: A comparative study. Materials & Design

Patro, S, & Sahu, KK. (2015). Normalization: A preprocessing stage. Cornell University

Peshawa J., M, A., & Rezhna, H. (2014). "Data Normalization and Standardization: A Technical Report", Machine Learning Technical Reports, 1(1), pp 1-6.
https://docs.google.com/document/d/1x0A1nUz1WWtMCZb5oVzF0SVMY7a_58KQulqQVT8LaVA/edit#

Prechelt, L. (1998). Early Stopping - But When?. In: Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg.

Rosenblatt, F. (1962). Principles of Neurodynamics. *Spartan Books*. Washington DC

Sak, H., Senior, A., Beaufays, Francoise. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. Interspeech

Seinowski, T. J., & C. R. Rosenberg. (1986). NETtalk: A parallel Network That Learns to Read Aloud. The Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, 32 pp. Reprinted in Anderson & Rosenfeld [1988], pp. 663-672.

Senior, A., Heigold, G., Ranzato, M., A., Yang, K. (2013). *An Empirical Study of Learning Rates in Deep Neural Networks for Speech Recognition*. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing.

Sharma, S. (2017, September 23). *Epoch vs Batch Size vs Iterations*. Towards data science. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

Shorten, C., Khoshgoftaar, T., M. (2019). A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*

Sotiropoulos, D., N., Tsihrintzis, G., A. (2016). *Machine Learning Paradigms, Artificial Immune Systems and their Applications in Software Personalization*. Springer

Shrivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014) 1929-1958.

Tu, J., V. (1996). Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes. J Clin Epidemion Vol. 49, No. 22, pp. 1225 – 1231
Wallach, D., & Goffinet, B. (1988). Mean Squared Error of Prediction as a Criterion for Evaluating and Comparing System Models. *Ecological Modelling,* 44 (1989) 299-306. Elsevier Science Publishers

Yang, L., Shami, A. (2020). On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *Neurocomputing*, 415: 295 – 316

Zheng, Y., Yang, C., Merkulov, A. (2018). Breast Cancer Screening Using Convolutional Neural Network and Follow-Up Digital Mammography. Conference: Computational Imaging III

Αργυράκης, Π. (2001). *Νευρωνικά Δίκτυα και Εφαρμογές*. Ελληνικό Ανοικτό Πανεπιστήμιο

Καραγιάννης, Γ., Μαραγκός, Α., Π. (2011). *Βασικές Αρχές Σημάτων & Συστημάτων*. Εκδόσεις Παπασωτηρίου

Πανεπιστήμιο Πατρών (2022, Σεπτέμβριος). *Βασικά Γεωμετρικά Στοιχεία: Κυρτό Περίβλημα.* https://eclass.upatras.gr

Ψούνης, Δ. (2016, 30 Μαρτίου). *Τεχνητά Νευρωνικά Δίκτυα – Οπισθοδιάδοση του Λάθους*. https://www.youtube.com/watch?v=WetXB3UUqck&list=PLLMmbOLFy25EAcQ7RNvThvBm9JmEZ0Pi4&index=25&ab_channel=%CE%94%CE%B7%CE%BC%CE%AE%CF%84%CF%81%CE%B7%CF%82%CE%A8%CE%BF%CF%8D%CE%BD%CE%B7%CF%82