ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ

ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ ΚΑΙ ΑΝΑΛΥΤΙΚΗ

ACTIVE LEARNING WITH SUPPORT VECTOR MACHINES

Ενεργή Μάθηση με Μηχανές Διανυσμάτων Στήριξης

ΕΜΜΑΝΟΥΗΛ ΚΑΒΑΚΑΚΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΟΡΕΣΤΗΣ ΤΕΛΕΛΗΣ

ΝΟΕΜΒΡΙΟΣ 2018

TABLE OF CONTENTS

ABSTRACT

In the field of Machine Learning all learning methods require a substantial amount of labeled data in order for the model to be properly fitted. In today's world of IoT (Internet of Things) and Big Data, where everything is controlled and monitored by software applications, unlabeled data are very easily acquired as they are continuously generated. However, the process of finding and annotating the true class to those dataset's instances, often requires more effort and time than the actual training of the model. ***Active learning*** aims to tackle this problem by enabling machine learning algorithms to perform equally well without reliance on the existence of huge training datasets. To accomplish this, an active learning algorithm is allowed to **query an oracle** (usually a human expert) for the true label of an unlabeled training example. There are a number of different ***strategies*** as well as ***learning scenarios*** that can be followed for this interaction which will be presented in later sections of this report. Active learning algorithms are basically wrapping around traditional supervised learning methods such as Support Vector Machines (SVMs), Logistic Regression etc. Apart from the topic of ***Active Learning***, this report offers a walkthrough of the theory behind Support Vector Machines and tries to present the various researched methods that combine these two topics.

ΠΕΡΙΛΗΨΗ

Στον τομέα της Μηχανικής Μάθησης όλες οι μέθοδοι εκμάθησης απαιτούν την ύπαρξη μίας σημαντικής ποσότητας κατηγοριοποιημένων δεδομένων προκειμένου οποιοδήποτε μοντέλο να εκπαιδευτεί σωστά. Στον σημερινό κόσμο του Διαδικτύου (Internet of Things) και των Μεγάλων Δεδομένων, όπου όλα ελέγχονται και παρακολουθούνται από εφαρμογές λογισμικού, τα μη κατηγοριοποιημένα δεδομένα αποκτώνται πολύ εύκολα καθώς παράγονται συνεχώς. Ωστόσο, η διαδικασία εύρεσης και σχολιασμού της αληθούς κλάσης στις περιπτώσεις αυτών των δεδομένων, απαιτεί συχνά περισσότερη προσπάθεια και χρόνο από την πραγματική εκπαίδευση του μοντέλου. Η ενεργή μάθηση στοχεύει στην αντιμετώπιση αυτού του προβλήματος επιτρέποντας στους αλγόριθμους μηχανικής μάθησης να αποδίδουν εξίσου καλά χωρίς να βασίζονται στην ύπαρξη τεράστιων συνόλων κατηγοριοποιημένων δεδομένων. Για να επιτευχθεί αυτό, ένας αλγόριθμος ενεργής μάθησης επιτρέπεται να ερωτά έναν 'προφήτη' (συνήθως έναν άνθρωπο εμπειρογνώμονα) για την αληθινή κατηγορία ενός μη επισημασμένου παραδείγματος εκπαίδευσης. Υπάρχουν ποικίλες διαφορετικές στρατηγικές καθώς και σενάρια εκμάθησης που μπορούν να ακολουθηθούν για αυτή την αλληλεπίδραση, τα οποία θα παρουσιαστούν σε επόμενα τμήματα αυτής της

διπλωματικής εργασίας. Οι αλγόριθμοι ενεργής μάθησης είναι ουσιαστικά μετα-αλγόριθμοί που περιβάλλουν τις παραδοσιακές μεθόδους μάθησης όπως οι Μηχανές Διανυσμάτων Υποστήριξης (SVM), η λογιστική παλινδρόμηση κλπ.. Εκτός από το θέμα της ενεργής μάθησης, αυτή η εργασία προσφέρει μια περίληψη της θεωρίας πίσω από τις Μηχανές Διανυσμάτων Υποστήριξης και προσπαθεί να παρουσιάσει τις διάφορες ενεργητικές μεθόδους μάθησης που συνδυάζουν αυτούς τους δύο τομείς της Μηχανικής Μάθησης.

1 INTRODUCTION

One of the goals of this paper is to introduce even the most inexperienced readers to the various methods of Machine Learning (ML). This is crucial to understand the benefits and flaws of the more complex and sophisticated learning schemes employed by the Active Learning scenarios that will be presented later on. Learning methods typically fall into three different categories: Supervised learning, Unsupervised learning and Semi-Supervised or Active learning. Those methods are derived from the form and qualities of the dataset that the model seeks to learn from. Following this small introduction, we will make a presentation of the respective concepts and methodologies of the biggest ML families in hope of making clear vague terms that the reader may have stumbled upon when reading the Abstract of this paper.

1.1 SUPERVISED LEARNING

Supervised learning is used in the majority of machine learning algorithms. Its main concept is that the training instances, represented as vectors $\vec{x}_i$, are labeled with a variable y_i that describes their true class. The model that we wish to fit to the data is essentially a mapping function $f(\vec{x}_i)$ such that $f(\vec{x}_i) = y_i$. Using the labeled training data, the model iteratively makes predictions and is corrected when those predictions are contrary to the true class of each training example. This procedure is terminated when the error becomes small enough or after some predefined number of iterations. Supervised learning can be further broken down into two categories: Classification and Regression. In the case of Classification, the model is designed to predict between two or more discrete values representing the different classes. For example, if a case of cancer is Benign or Malignant. Well known algorithms for classification are Decision Trees [1], Logistic Regression [2] and SVC (Support Vector Classifier) [3]. In Regression problems the model aims to predict a continuous output variable such as stock prices, a person's weight, salary etc. Common Regression algorithms are Linear Regression and SVR (Support Vector Regression) [4].

1.2 UNSUPERVISED LEARNING

In contrast to supervised learning, unsupervised learning algorithms make use of training datasets without the need for the training instances to be annotated with their respective class. The models learn from identifying structures, distributions or patterns in the input data. The data are grouped according to some metric or similarity measure, for example the Euclidean distance between the various training instances. The groups of data that are generated through this procedure are called clusters. Clustering techniques are divided into three main categories, the ***Hierarchical***, the ***Bayesian*** and the ***Partitional***. In ***Hierarchical*** algorithms, previously established clusters help in finding successive clusters. The result of the process is the formulation of a hierarchy of clusters usually represented by a tree. This family of algorithms can be broken down even more into two approaches, the ***agglomerative*** ('bottom-up') [5] and the ***divisive*** ('top-down') [6]. In the agglomerative approach each training instance is considered as a separate cluster and is afterwards merged into larger ones. The Divisive approach considers the entire dataset as a cluster and iteratively divides it into successively smaller ones. The ***Bayesian*** algorithms calculate the posterior probabilities of a datum instance to belong to each cluster and assign that instance to the cluster that achieves the greatest probability [7]. Finally, the ***Partitional*** algorithms, whose most well-known member is the K-means algorithm, directly assign each training instance to a cluster without regard to any probability model that describes the data. Instead they use some similarity or dissimilarity measure such as the distance from each cluster center (*centroid*) [8].

1.3 ACTIVE (SEMI-SUPERVISED) LEARNING

Supervised learning assumes that there is an abundance of labeled datasets that can be used for training purposes. However, there is a chance that for some problems there are either not enough labeled data or even if there are some, they do not effectively describe the different classes that compose it. Generally, the real class of the data instances is annotated by some human actor. In certain scenarios this can be done in large volumes by some user contribution e.g. music streaming services' subscribers adding a genre to a song and submitting it to the provider. Yet, in some fields such as the study of proteins this user contribution is not feasible. Therefore, you can imagine the cost both in time and other resources of having human business experts labeling novel datasets consisted of hundreds or thousands of instances. In addition to that, traditional supervised learning algorithms learn by iterating through all training samples no matter how important they are in finding the optimal solution. This can be considered as a waste in time, in computational resources and in the case of outliers or noise it might even reduce the quality of the results. Active learning can be considered as a special case of supervised learning that tackles the aforementioned issues. The main concept of active learning is that if the learning algorithm could choose the training instances that will mostly benefit the training process, it would reduce the amount of data needed while achieving great accuracy. Also, since it chooses the most effective training examples it reduces the time needed to reach a certain performance threshold and additionally will ignore outliers and noise.

At this point the reader should have become accustomed to the main pillars of ML. In the following chapter we will present the origins and theory of the learning model that will be used later on, during the practical presentation of some of the most common Active Learning strategies, Support Vector Machines.

2 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning model that can be used for both classification and regression problems. It accomplishes its purpose by constructing a hyperplane in multidimensional space that separates the different classes. This is achieved by identifying the optimal hyperplane, called the decision boundary, which separates the training data with a maximum margin. The original version of SVM was invented by Vladimir N.Vapnik and Alexey Ya. Chervonenkis in 1963. Later work [9] by Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik, introduced an efficient way of mapping the training input into higher or infinite dimension space. This in known as the Kernel Trick. The result of this is that it enables SVM to efficiently capture much more complex relationships between the data points without having to perform extensive transformations. The current incarnation of soft margin was proposed by Corinna Cortes and Vapnik [10]. SVMs are used in a plethora of applications. Some common examples are for Face Detection, Text and Hypertext categorization, Classification of Images, Bioinformatics, Handwriting recognition and Geospatial data analysis. The following analysis is heavily inspired by the notes of Andrew Ng[1] and we refer the reader to them for further reading.

2.1 Problem Formulation

Considering the linearly separable case, we need to identify a straight line that separates the training data. However, there is an infinite number of lines that can be fitted to the training dataset. Some of those can be considered as ‘good’ options but some others not that good. Figure 1 depicts some candidate lines that separate the two classes. At first glance all of them suffer from some issues that wouldn’t make them the best choice. For example, lines A and C would both misclassify the points marked with x even though it’s obvious to which class they belong. Line D is just a separator based on the vertical height of the points. But how do Support Vector Machines confidently calculate the optimum decision boundary that achieves this purpose? They do that by placing the line in such a way so the separation between the training examples of different class is as wide as possible.

[1] CS229 Lecture notes, Part V Support Vector Machines

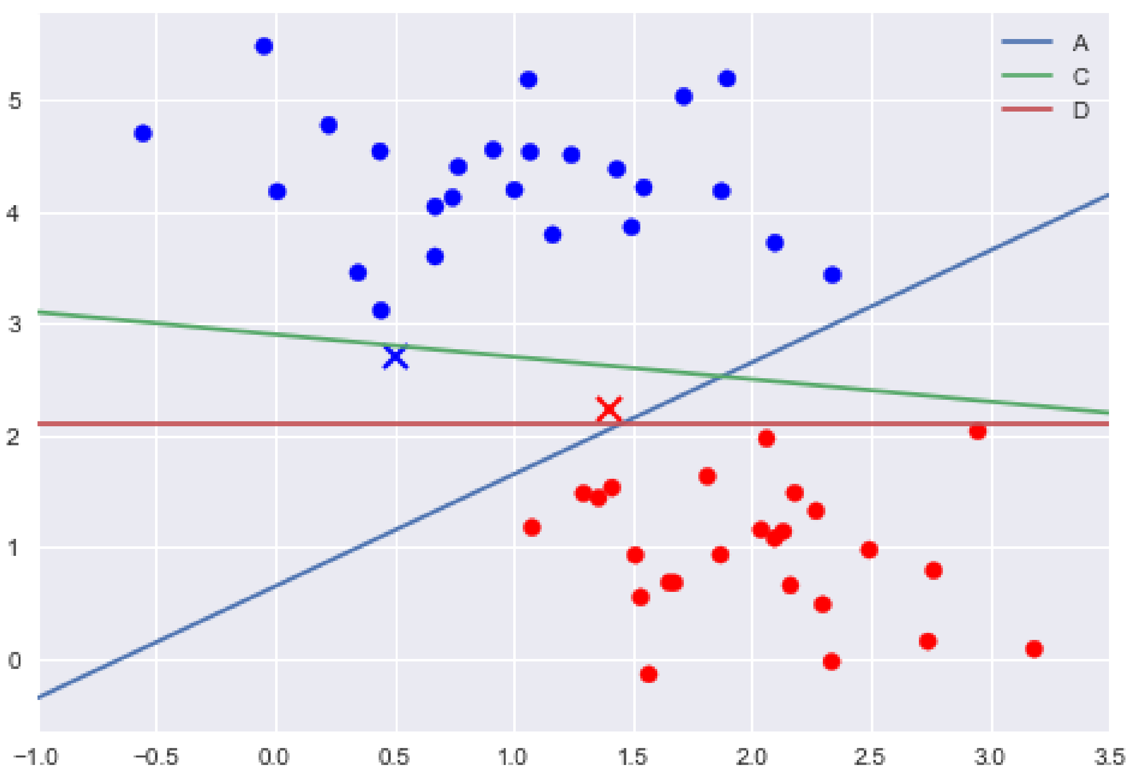


Figure 1 Various Decision Boundaries

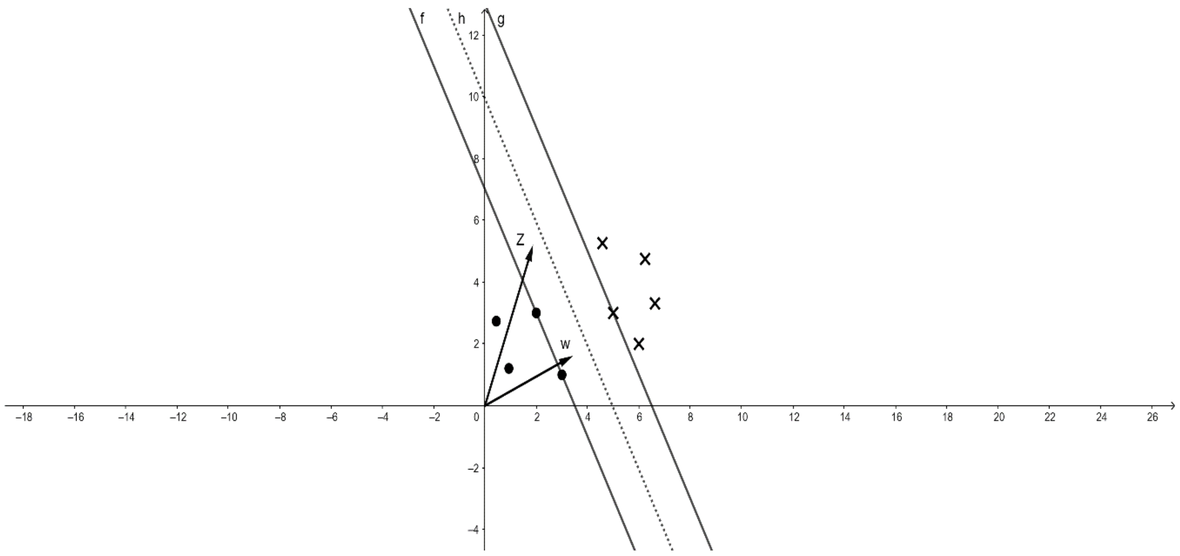


Figure 2

2.2 THE OPTIMAL MARGIN CLASSIFIER

Consider a normal vector $\vec{w}$ that is perpendicular to the median (dotted) line (Figure 2). Also consider an unknown vector $\vec{z}$. The goal is to identify if the unknown $\vec{z}$ lies on the side of the positive samples or the side of the negative samples. This can be expressed as follows:

$$\vec{w} \cdot \vec{z} \geq c, \textit{c is a constant} \qquad (1)$$

Intuitively, this means that if the projection of $\vec{z}$ to $\vec{w}$ is big enough it will eventually cross the median line and $\vec{z}$ will be classified as a positive sample. Without loss of generality, by setting

$c = -b$ in (2) we can assume that if :

$$\vec{w} \cdot \vec{z} + b \geq 0, \textit{b is a constant} \qquad (2)$$

then $\vec{z}$ is a positive sample. Relation (2) is called the decision rule.

In order to use the decision rule, the values of $\vec{w}$ and b must be calculated. For a positive sample x^+ we require that:

$$\vec{w} \cdot \overrightarrow{x^+} + b \geq 1 \qquad (3)$$

Similarly, for a negative sample x^- we require that:

$$\vec{w} \cdot \overrightarrow{x^-} + b \leq -1 \qquad (4)$$

To make the formulation of the problem more convenient we will now introduce a new variable y_i such that:

$$y_i = \begin{cases} +1, for\ positive\ samples \\ -1, for\ negative\ samples \end{cases}$$

By multiplying (3) and (4) with their respective value of y_i, we receive the following inequality for both cases:

$$y_i(\vec{w} \cdot \vec{x_i} + b) - 1 \geq 0 \qquad (5)$$

$$Constrained\ by\ y_i(\vec{w} \cdot \vec{x_i} + b) - 1 = 0, \qquad (6)$$

for those $\vec{x_i}$ that lie on the lines that define the margin. Essentially, Support Vector Machines maximize the margin between the different class training points that lie closest to the decision boundary.

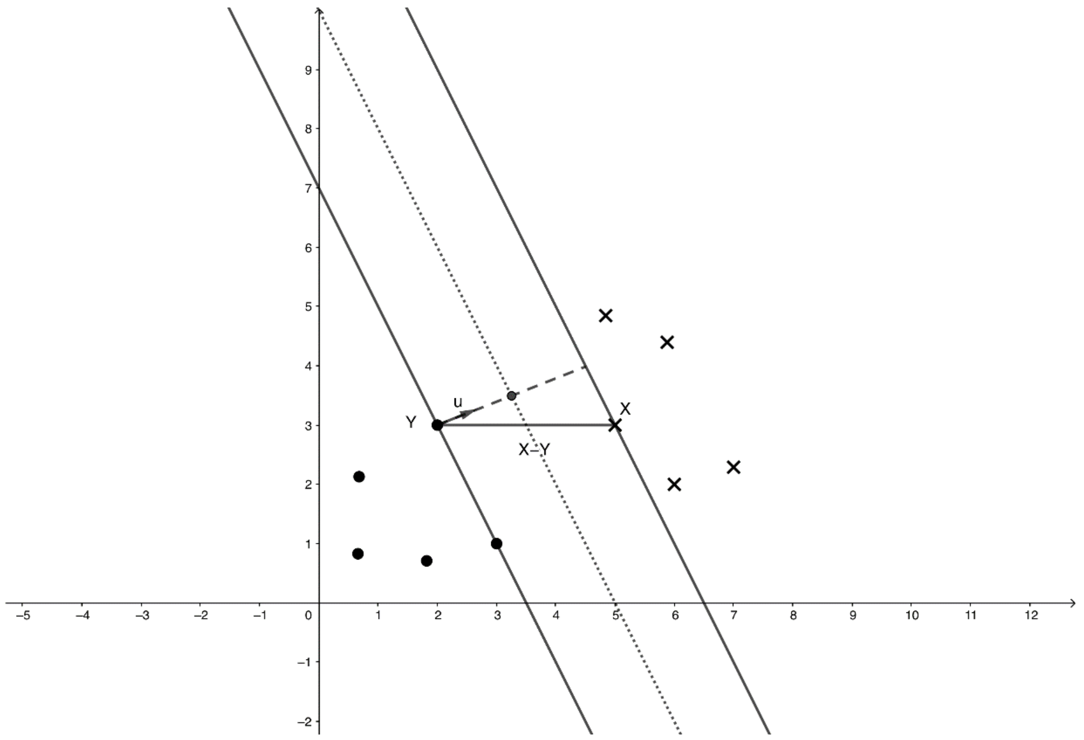


Figure 3

As Figure 2 suggests, the margin is the width of the segment that connects point X to point Y, where X is a positive sample and Y is a negative sample. So the margin can be expressed as:

$$margin = (\vec{X} - \vec{Y}) \cdot \vec{u}, \quad \vec{u} \text{ is a normal unit vector}$$

Earlier, we defined the vector $\vec{w}$ to be normal, so we can apply it to the margin equation by transforming it to a unit vector. The margin can now be expressed in relation to the vector $\vec{w}$:

$$margin = (\vec{X} - \vec{Y}) \cdot \frac{\vec{w}}{\|\vec{w}\|} \qquad (7)$$

Using the constraint (6) we acquire that:

$$\vec{w} \cdot \vec{X} = 1 - b$$

$$\vec{w} \cdot \vec{Y} = -b - 1$$

and substituting in (7):

$$\text{margin} = \frac{2}{||\vec{w}||}$$

So, in order to maximize the width of the margin we have to maximize $\frac{1}{||\vec{w}||}$, which is equivalent to minimizing $\frac{1}{2}||\vec{w}||^2$.

Finally, the optimization problem that we have to face in order to find the optimum decision boundary is the following:

$$min \frac{1}{2}||\vec{w}||^2 \qquad \textit{(8)}$$

$$s.t.\ y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1\,, i = 1, \dots, m$$

The above Optimization Problem contains a convex, quadratic function and can be solved using Quadratic Programming. Later on, we will reformulate this problem to its dual form and this will enable us to use Kernels. Kernels will allow us to solve the cases where the training data are not linearly separable and to apply algorithms that solve the optimization problem much more efficiently than Quadratic Programming.

2.3 GENERALIZATION PARAMETERS

So far, the formulation of the Optimization Problem was based on the case of the training data being linearly separable. However, real world datasets are rarely that well defined. They contain problematic features such as noise, overlapping classes and outliers that hinder the training of an accurate model. With the current formulation the model will most likely consider the noise inside the dataset as useful information and will learn from it leading to undesirable results. To tackle this problem, it is better to allow some training samples to violate the margin constraints in (9). This is achieved by applying positive variables ξ_i to the constraints, called slack variables.

$$min \frac{1}{2}||\vec{w}||^2 + C \sum_i^m \xi_i \qquad \textit{(9)}$$

$$s.t\ \ y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 - \ \xi_i\,, i = 1, \dots, m$$

$$\xi_i \geq 0, i = 1, \dots, m$$

The training examples are now allowed to have a margin less than one. This also results in the margin area being non-empty, meaning that some of the constraints have failed and we have allowed some data points to reside there. The constant C is a regularization parameter that controls the tradeoff between achieving a low cost and minimizing the norm of $\vec{w}$. By using a large value for C, the optimization algorithm will produce a model with a very small margin that will try to classify correctly all training examples. This leads to loss in the model's ability to generalize and eventually overfitting. On the other hand, if the value of C is too small then the resulting model will allow a large margin that will lead to large training error. At the time, there is no method to predetermine the best value of C, so in most cases experimentation on the dataset at hand is needed for choosing the best value.

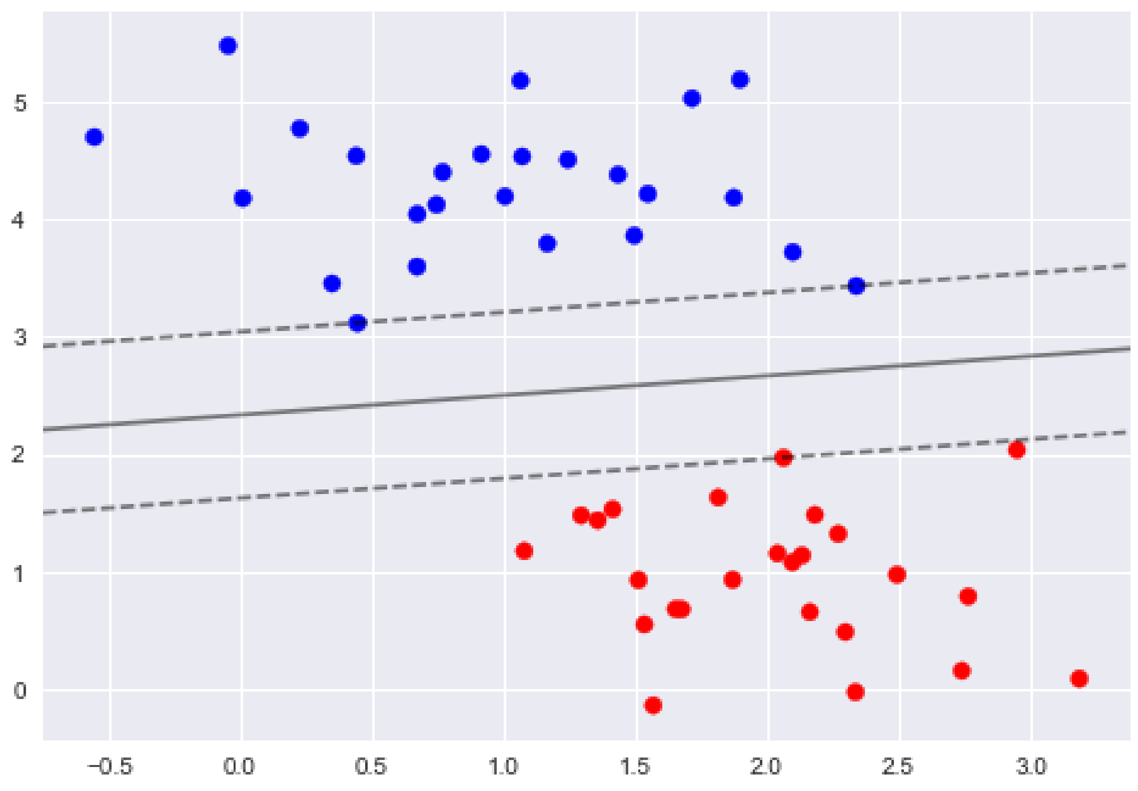


Figure 4 Optimal Margin Classifier

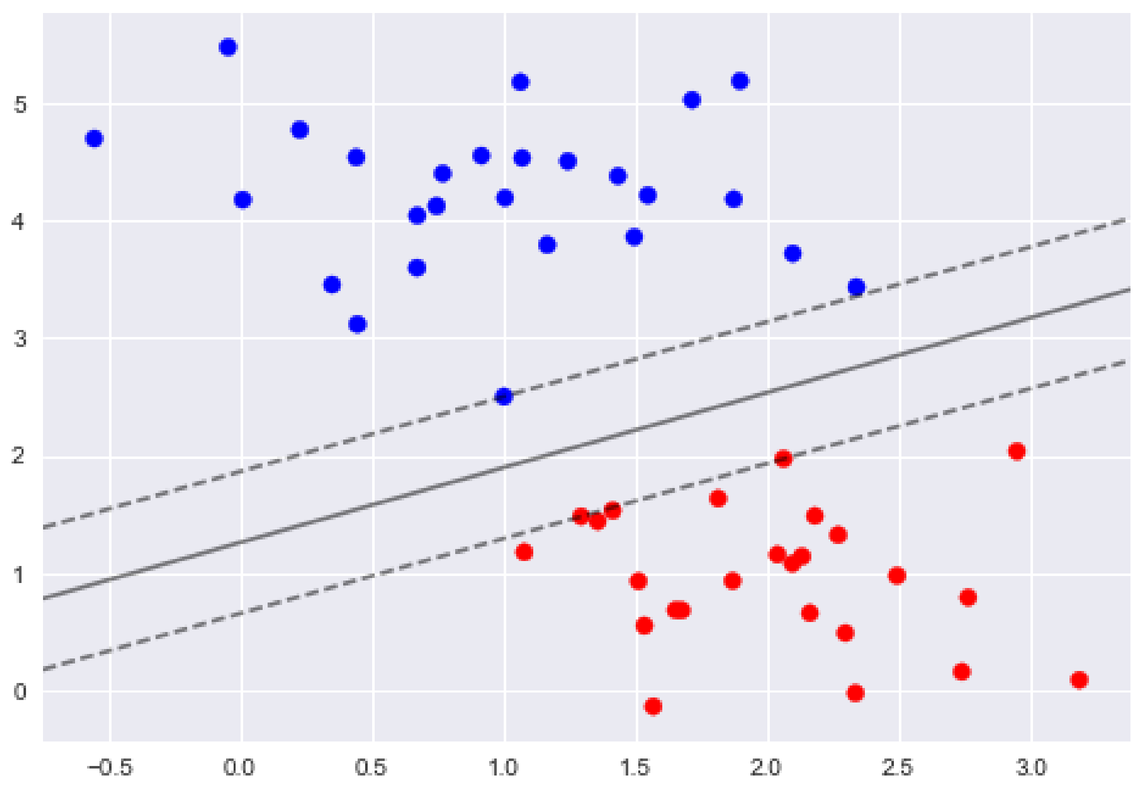


Figure 5 The Impact on the margin's width after adding a borderline point.

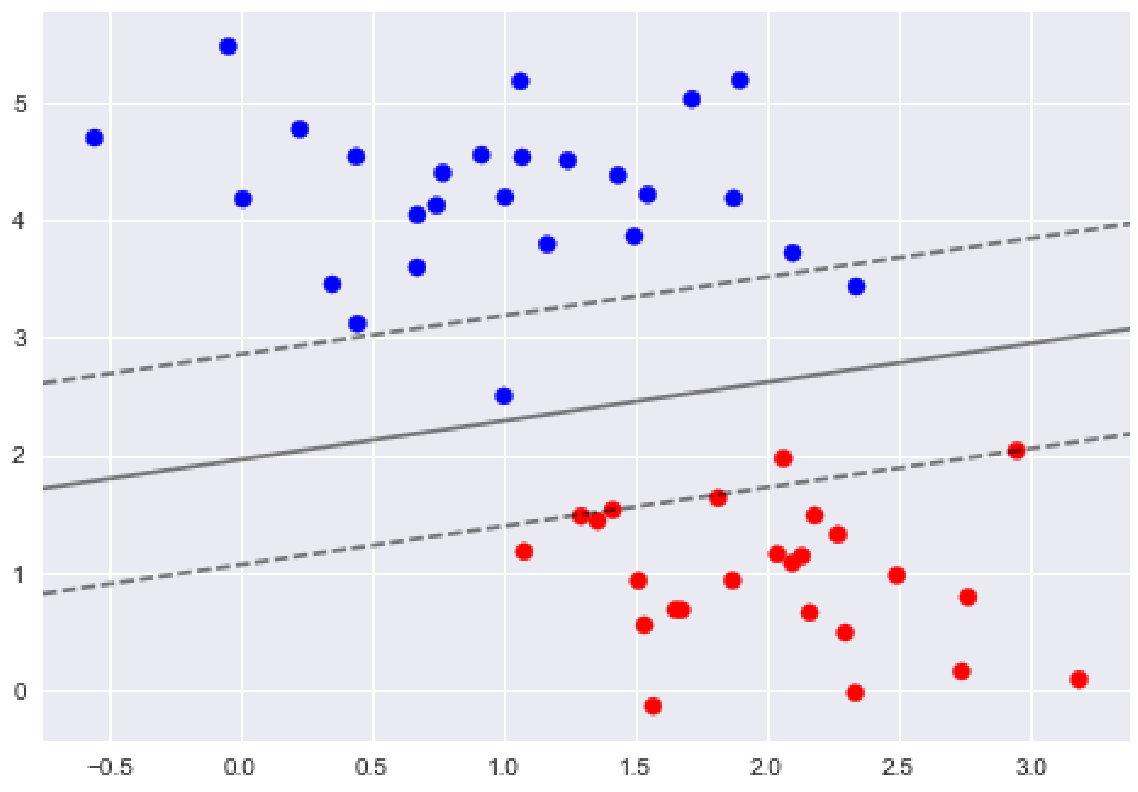


Figure 6 Allow for bigger margin by using the Regularization Parameters

2.4 LAGRANGE DUALITY

Before delving into the dual presentation of (8) we must present the method that we are going to use to solve it. Lagrange Multipliers are a mathematical method for solving optimization problems of differentiable functions. In contrast to other methods of finding the extremums of functions like the second derivative test, Lagrange multipliers take into consideration various constraints that may be applied to the function. Given a nonlinear programming problem known as the primal problem, there exists another nonlinear programming problem closely related to it, called the dual problem. Consider F_P being the primal objective function and let F_D be the objective function of the dual problem. If $\vec{x}$ is a feasible solution for the **minimization** of F_P and $\vec{y}$ is a feasible solution for the **maximization** of F_D the **Weak Duality Theorem** states that $F_D(\vec{y}) \leq F_P(\vec{x})$. This means that under certain convexity assumptions the primal and dual problems can have equal optimal objective values. Thus, we can solve the dual problem instead of the primal. The main concept of the method is that for optimizing a function $f(x_{1,} \dots, x_n) \to R^n$ subject to a constraint $g(x_{1,} \dots, x_n) \leq 0$ the gradients of f and g point to the same direction and differ at most by a scalar factor λ.

$$\nabla f(x) = \lambda \nabla g(x) \qquad (10)$$

$$g(x) \leq 0 \qquad (11)$$

Combining (10) and (11) into the so called Lagrangian equation:

$$L(x, \lambda) = f(x) - \lambda g(x) \qquad (12)$$

We are searching for the points that satisfy:

$$\nabla L(x, \lambda) = 0 \qquad (13)$$

In case of multiple inequalities $g_i(x)$ and equalities $h_i(x)$, (12) generalizes to:

$$\mathrm{L}(\mathrm{x}, \lambda, \mu) = \mathrm{f}(\mathrm{x}) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

The scalars $\lambda_{i,} \mu_j$ are called the **Lagrange Multipliers**. In order to calculate the gradient of $\mathrm{L}(\mathrm{x}, \lambda, \mu)$ we must take the first partial derivatives of $\mathrm{L}(\mathrm{x}, \lambda, \mu)$ with respect to x, λ, μ, set them to zero and solve for x, λ, μ.

Additionally, for the Primal $\mathrm{L_P}$ and dual L_D to share an optimal feasible solution there must exist a vector $\vec{\mathrm{x}}$ that is a solution to the primal problem and $\lambda_{i,} \mu_\iota$ that are the solution to the dual problem and satisfy the **Karush-Kuhn-Tucker (KKT)** conditions:

$$\frac{\partial \mathrm{L}}{\partial \vec{\mathrm{x}}} L(\vec{x}, \lambda, \mu) = 0, \forall i = 0, \dots, n$$

$$\frac{\partial \mathrm{L}}{\partial \mu_i} L(\vec{x}, \lambda, \mu) = 0, \forall i = 0, \dots, l$$

$$\lambda_i g_i(\vec{x}) = 0, \forall i = 0, \dots, k$$

$$g_i(\vec{x}) \le 0, \forall i = 0, \dots, k$$

$$\lambda_i \ge 0, \forall i = 0, \dots, k$$

By plugging our original optimization problem (8) into the Lagrangian (12) we get the primal presentation of our problem:

$$L_P = \frac{1}{2}\|\vec{w}\|^2 - \sum \lambda_i [\, y_i(\vec{w} \cdot \vec{x_i} + b) - 1] \qquad (14)$$

$$= \frac{1}{2}\sqrt{\vec{w} \cdot \vec{w}}^2 - \sum \lambda_i [\, y_i(\vec{w} \cdot \vec{x_i} + b) - 1]$$

$$= \frac{1}{2}\vec{\mathrm{w}} \cdot \vec{\mathrm{w}} - \sum \lambda_{\mathrm{i}} [\, y_i(\vec{w} \cdot \vec{x_i} + b) - 1]$$

Calculating the partial derivatives:

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum \lambda_i y_i \vec{x_i} = 0 \rightarrow \vec{w} = \sum \lambda_i y_i \vec{x_i} \qquad (15)$$

$$\frac{\partial L}{\partial b} = - \sum \lambda_i y_i = 0 \rightarrow \sum \lambda_i y_i = 0 \qquad (16)$$

The important thing to note in (15) is that the vector $\vec{w}$ is **eventually the linear sum of the training examples**. All or some of them, because for some training examples the multipliers λ_i will be zero. This is made sure by the **KKT dual complementary conditions $\boldsymbol{\lambda_i g_i(\vec{x}) = 0}$ and $\boldsymbol{\lambda_i \ge 0}$.** For those training examples that their respective multiplier $\boldsymbol{\lambda_i}$ has a value larger or equal to zero**,** the corresponding constraint $\boldsymbol{g_i(\vec{x})}$ will have zero value. Those training examples will lie on the lines that define the margin and will be called the **Support Vectors**.

Using (15), (16) on the Lagrangian primal objective function (14) we get the Lagrangian dual objective function:

$$L_D = \frac{1}{2}\sum_i \lambda_i y_i \vec{\mathrm{x}_i} \cdot \sum_j \lambda_j y_j \vec{\mathrm{x}_j} - \sum_i \lambda_i y_i \vec{\mathrm{x}_i} \cdot \sum_j \lambda_j y_j \vec{\mathrm{x}_j} - \sum_i \lambda_i y_i b + \sum_i \lambda_i \rightarrow$$

$$L_D = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i \, y_j \, \vec{x_i} \cdot \vec{x_j} \qquad (17)$$

Thus, the optimization problem that we need to solve is:

$$max \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i \, y_j \, \vec{x_i} \cdot \vec{x_j} \quad (18)$$

$$s.t. \sum \lambda_{\mathrm{i}} \mathrm{y}_{\mathrm{i}} = 0$$

$$\lambda_i \ge 0$$

Finally plugging (15) to the decision rule we get that for an unknown $\vec{z}$ to be considered as a positive sample the following inequality must be met:

$$\sum_i \lambda_i y_i \vec{x}_i \cdot \vec{z} + b \geq 0 \qquad (19)$$

Both relations (17) and (18) show that the optimization problem as well as the decision rule depend only on the dot product of the training examples and the unknowns. This is **important** because it is what enables as to use the **Kernel Trick**.

2.5 KERNELS

The examples presented so far assumed datasets that can be separated using a straight line. In real life problems this is hardly ever the case, as data are **randomly distributed**. **Error! Reference source not found.** displays the case of a dataset consisted of two clusters. It is obvious that the generated linear classifier cannot separate the classes accurately as one class is enclosed by the other. However, there is a way of **tricking** a linear classifier into solving a linear problem even though the dataset is **not** linearly separable. This can happen because an inseparable dataset can become separable if we project it to higher dimensions. For an input vector $\vec{x}$ there is a feature mapping function $\varphi(\vec{x})$ that acts as a map from n-dimensional to m-dimensional space, where m is larger than n. So, since $\varphi(\vec{x})$ transforms the input to higher dimensions we could replace the dot product of $\overrightarrow{\mathrm{X_i}} \cdot \overrightarrow{\mathrm{X_j}}$ with the dot product of their feature mapping functions $\varphi(\overrightarrow{\mathrm{X_i}}) \cdot \varphi(\overrightarrow{\mathrm{X_j}})$. However, this would not be efficient since the computations required for producing $\varphi(\overrightarrow{\mathrm{X_i}})$ and $\varphi(\overrightarrow{\mathrm{X_j}})$ are very expensive. Also, after all the trouble of going up to the m-dimensional space the actual result of the dot product is just a scalar. By choosing the right Kernel function we remove the need for such complex computations. As a Kernel function $K(\vec{x}, \vec{y})$ of two n-dimensional input vectors we define the result of the dot product between their respective feature mappings, without having to explicitly calculate them.

$$K(\vec{x}, \vec{y}) = \varphi(\vec{x}) \cdot \varphi(\vec{y})$$

To make this understandable, an example is presented:

Consider the input vectors $\vec{x}, \vec{y}$ such that $\vec{x} = (1, 2, 3), \overline{y} = (4, 5, 6)$. Also consider a feature mapping function $\varphi(\vec{x}) = (x_1x_1,\ x_1x_2,\ x_1x_3,\ x_2x_1,\ x_2x_2,\ x_2x_3\ , x_3x_1\ , x_3x_2\ , x_3x_3\ )$. For the input vectors $\vec{x}, \vec{y}$ we calculate their mappings and dot product:

$$\varphi(\vec{x}) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$$

$$\varphi(\vec{y}) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$$

$$\varphi(\vec{x}) \cdot \varphi(\vec{y}) = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

It is obvious that even for small dimensional vectors the amount of calculations needed is very large $(O(n^2),\ n\ being\ the\ dimension\ of\ the\ vectors)$. Now using the equivalent Kernel function $K(\vec{x},\vec{y}) = (\vec{x} \cdot \vec{y})^2$:

$$K(\vec{x},\vec{y}) = (4 + 10 + 18)^2 = 1024$$

Both ways produce the same result but in the case of the Kernel function the operations needed are reduced dramatically $(O(n))$.

As described above, by using the Polynomial Kernel $K(\vec{x},\vec{y}) = (a\vec{x} \cdot \vec{y} + c)^d$ with d=2, we were able to accurately separate the same dataset that the linear separator in Figure 7 failed. The resulting decision boundary and margins are depicted in Figure 8.

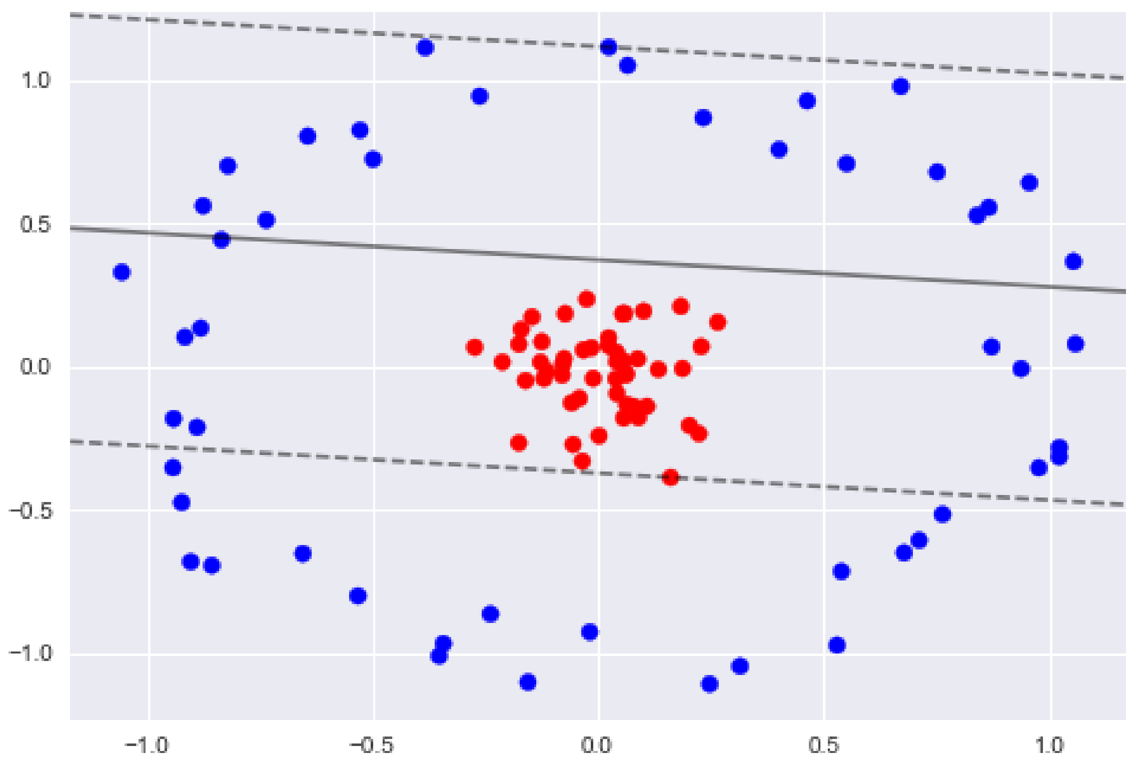


Figure 7 Attempt to separate a non linearly separable dataset using the linear SVM

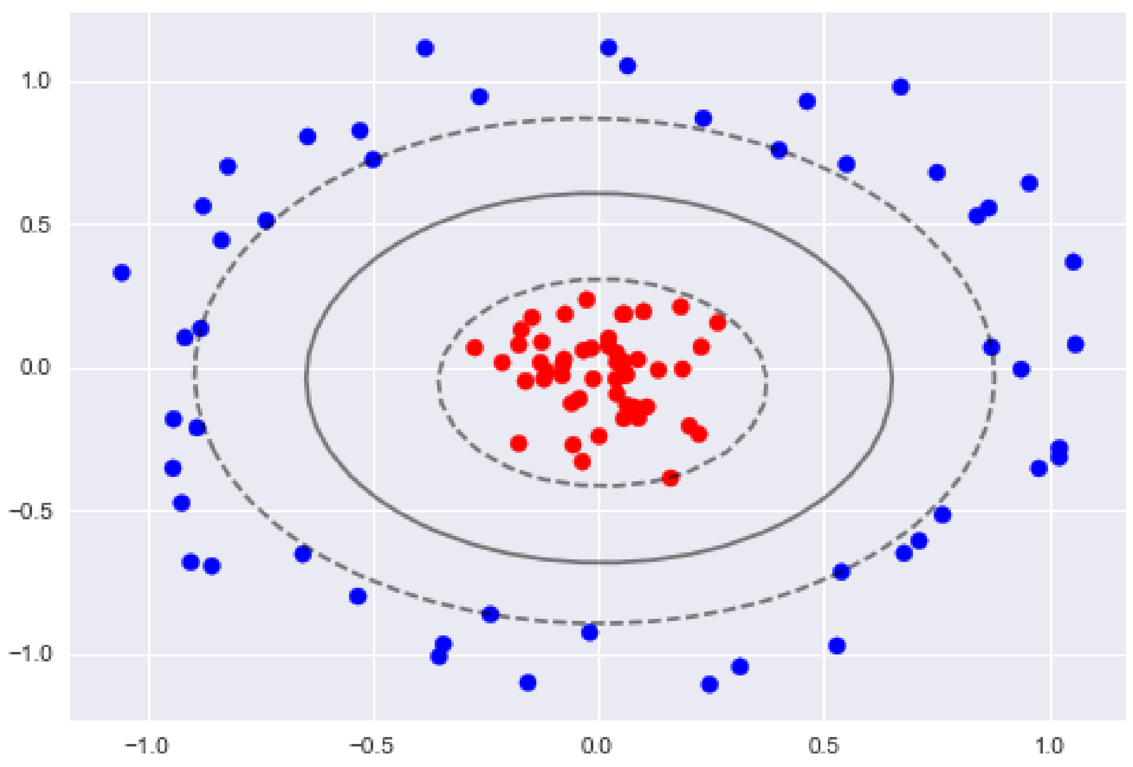


Figure 8 SVM using the polynomial Kernel

2.5.1 **Common Kernels**

Using the right Kernel function is not a trivial task, as it depends on the problem at hand. For example, a linear Kernel only allows to pick out lines or hyperplanes, a polynomial Kernel models feature conjunctions up to a specified polynomial order and the radial basis function allows to pick out circles or hyperspheres. After choosing the correct Kernel function, the user must also cope with the tedious task of fine tuning the various hyper parameters to achieve maximum performance.

1. Linear Kernel

$$K(x, y) = x^T \cdot y + c$$

2. Polynomial Kernel

$$K(x, y) = (ax^T \cdot y \ + c)^d$$

Where a is the the slope and d is the order of the polynomial.

3. Gaussian Kernel (Radial Basis)

$$K(x, y) = \exp(-\frac{\|x - y\|^2}{2\sigma^2})$$

A simpler representation of the RBF Kernel can be obtained by defining $\gamma = \frac{1}{2\sigma^2}$

$$K(x, y) = \exp(-\gamma\|x - y\|^2)$$

The σ parameter plays a major role in the performance of the RBF Kernel function. Larger values of σ lead to a general classifier but overestimating its value can lead to the model losing its nonlinear power. On the other hand, small values of σ tend to make a classifier that is sensitive to small local clusters but underestimating its value leads to lack of generalization and unwanted sensitivity to noise.

4. Sigmoid Kernel

$$K(x, y) = \tanh(\gamma x^T y + c)$$

3 ACTIVE LEARNING

As it was mentioned in the introductory chapter, Active Learning algorithms construct/expand their initial training datasets by identifying informative unlabeled training instances. After some basic training with a small dataset, an active learner will query an oracle (usually a human expert) whenever it decides that an unlabeled training instance will have substantial positive impact to the process. This impact can be defined in various ways, depending on which active learning strategy is being used, e.g. uncertainty sampling, variance reduction etc. In addition to the strategies that can be used to configure the learner, active learners follow one of three main scenarios. Intuitively, as scenarios we define the variants in the origin that the active learner can query training instances from. Figure 9 illustrates an abstraction of the procedure that every active learner follows, called the Active Learning Cycle.

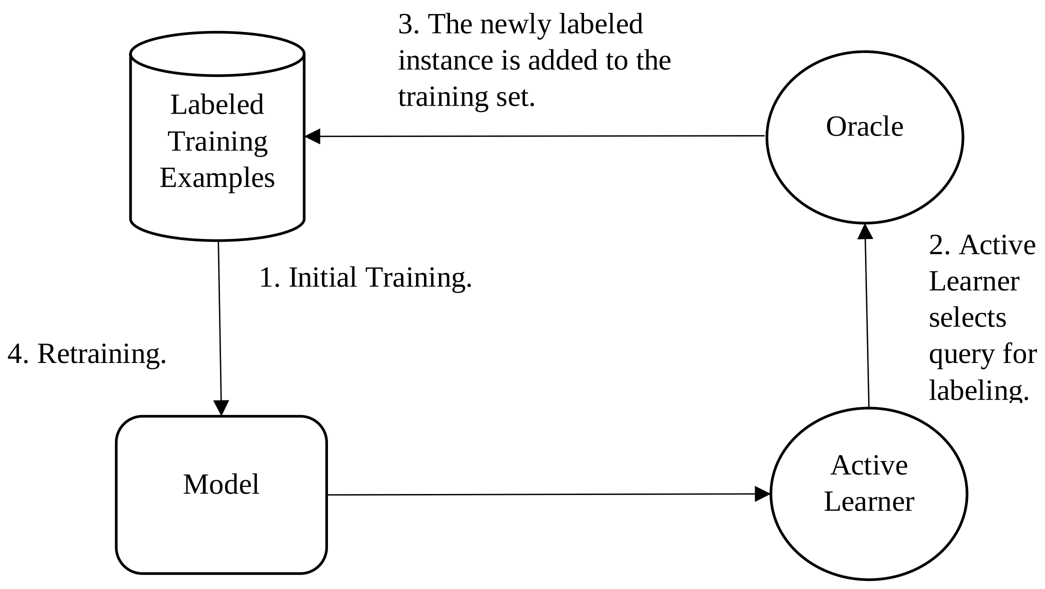

3. The newly labeled instance is added to the training set.
Labeled Training Examples
Oracle
1. Initial Training.
2. Active Learner selects query for labeling.
4. Retraining.
Model
Active Learner

Figure 9 The Active Learning Cycle

3.1 ACTIVE LEARNING SCENARIOS

Unlabeled data can be freely generated and provided by several different sources. For example, sensors may collect data and propagate them to a central repository or stream them directly to the consumers. It is only natural for active learners to be able to adjust to these variants and process unlabeled instances no matter their origin. In the following sections we will present the three main scenarios that are employed to enable active learning frameworks to make use of diverse data sources. For a condensed presentation of these strategies we refer the reader to Figure 10.

3.1.1 Membership Query Synthesis

In Membership Query Synthesis the learning algorithm doesn't rely on a pool of unlabeled data to draw the sample it wishes labeled by the oracle. Instead, the learner constructs the instances from some underlying natural distribution. For example, in the case of character recognition an active learner will create an image that is similar to a character and will send it to the oracle in order to label it. Since the synthesized training instances are only restricted by the input space, there are cases where samples drawn using the Membership Query Synthesis reduce the predictive error rate faster than the pool based approach [11]. However, this approach has certain limitations due to the nature of artificially generating training instances. In some settings, for example when generating letters, some samples may amount to no known character. In such situations, querying a human oracle for something that applies to no real world logic is considered a bad use of this learning scenario. In an application described by King et al [12] the role of the oracle is assigned to an automated robot. The robot, devises experiments and judging from their results, assigns a label to each of them. Since the conducted experiments were designed, implemented and labeled by the 'robot scientist' the need for a human oracle was eliminated. This resulted in 100-fold decrease in cost[2] compared to running random experiments, while the accuracy was on par with having a human scientist labeling the results. Another way to tackle the problem of synthesized queries being unknown to the human oracle is proposed in an article by L. Wang et al. [13]. The proposed hybrid algorithm combines the Membership Query Synthesis scenario to the pool-based sampling. Using labeled data, it synthesizes an instance close to the decision boundary and then it calculates the closest real unlabeled instance from the unlabeled pool of samples. Then, the real and human recognizable instance is sent to the oracle to label it.

3.1.2 Stream-based Sampling

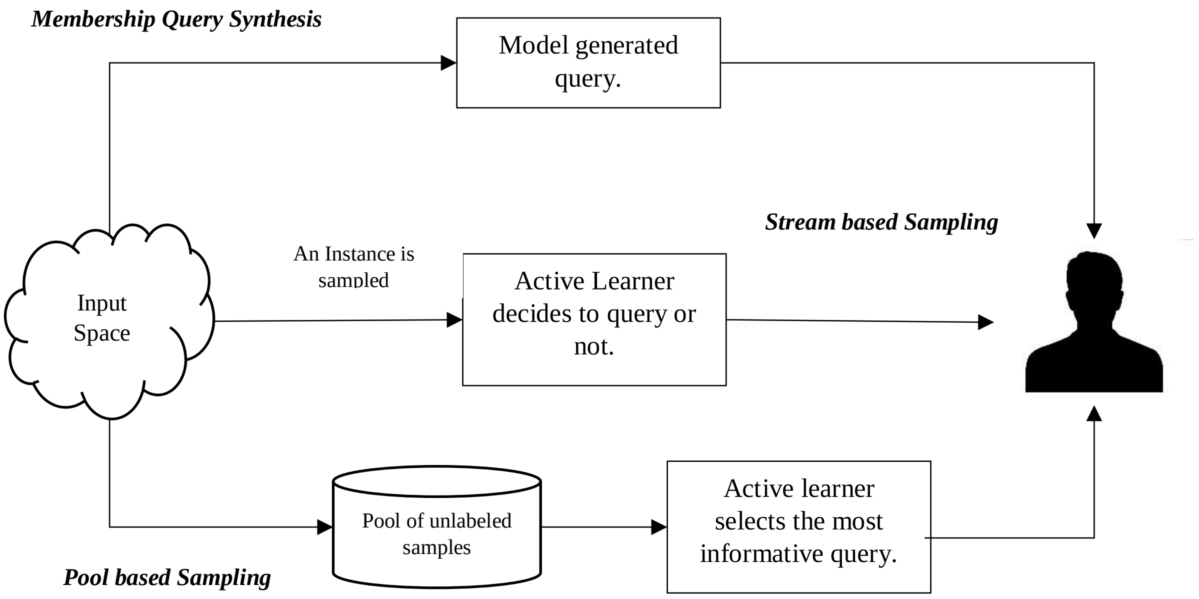
In Stream-based or selective sampling we make the assumption that acquiring an unlabeled instance is free or inexpensive. A stream of samples reaches the active learner who then determines whether to query the current instance's label. It is important to note that in real world applications, streams are often too large for the learner to store the data in a background pool and then decide which to label. So, this procedure has to run in parallel to the stream. The learner decides on having

[2] In their experiments, the researchers defined cost as a function of time and price.

the current instance labeled or not, based on some informativeness criterion. This criterion can be based on a selection strategy such as uncertainty sampling, where the learner computes some kind of metric for each sample e.g. its distance from the decision boundary. If the distance reaches a certain threshold the oracle is asked to label it. Another way for the learner to choose the most informative samples is by tracking the areas of the input space for which it knows the least. If the current stream sample belongs to those areas it will be sent to the oracle for labeling. Wei Chu et al. in [14] illustrate an algorithm for online active learning, trained on real user generated data (UGC) from a Yahoo! News portal. The purpose of the research is to use stream-based sampling combined with importance sampling [15] to make detection of abusive UGC smarter. Their results show great promise in reducing labelling efforts in the setting of such dynamic problems where there is concept drift. Stream-based sampling is also successfully utilized in the field of robotic vision where the robot needs to be adaptive to rapid changes of its environment. As it is illustrated in [16] by Alexander Narr, Rudolph Trieber and Daniel Cremers, self-driven vehicles actively learn from a stream of 3D point data and can modify their model by learning new semantics with the knowledge they gain during their operation.

3.1.3 **Pool-based Sampling**

Pool based sampling has been the most practiced scenario in applied active learning. It has been used in a wide variety of application contexts from text classification [17], [18] to image classification [19], [20] and cancer diagnosis [21] . Its wide acceptance is due to the fact that in most real-world applications there is an abundance of training samples but only a small portion of those is labeled. In most cases the pool of the unknown samples is static, which means that no additional instances are being added along the training phase. During the learning process the active learning algorithm will use a selection strategy in order to compute an informativeness measure for all elements of the pool. In some settings where the pool is too large for all its instances to be considered, the algorithm may choose a random subset of it. Then, the instance that scored the greatest informativeness value will be sent to the oracle for its true label and the learning cycle will continue. Since the model changes after every query selection, the algorithm must continuously re-compute the selection criterion in every training cycle. Due to this need, pool-based sampling has great computational requirements. Thus, in cases where memory or processing power are limited, pool-based sampling is not considered as the most appropriate choice and other alternative methods, like stream-based sampling must be considered.



Membership Query Synthesis
Model generated query.
Stream based Sampling
An Instance is sampled
Input Space
Active Learner decides to query or not.
Pool of unlabeled samples
Active learner selects the most informative query.
Pool based Sampling

Figure 10 Overview of Active Learning Scenarios

3.2 ACTIVE LEARNING STRATEGIES

In the previous section we presented the different sources that an active learner may receive its input from. During this presentation we often mentioned that the learning algorithm uses an informativeness criterion or selection strategy to decide which unlabeled instance to send to the oracle. Query selection strategies are the core of active learning and there has been a plethora of ways of defining the informativeness of training samples. The following part will present an overview of the most common selection strategies.

3.2.1 **Uncertainty Sampling**

Uncertainty Sampling can be considered as the most intuitive and easy to grasp query selection strategy. In this setting, the active learner chooses to label the instance for which is less certain. Certainty can be defined in a number of different ways. For example, a way of defining certainty is by translating it to the probability of an instance belonging to a certain class. For a binary classification problem, a logistic regression active learner would choose to label an instance with posterior probability of belonging to a certain class that is close to 0.5. From now on, throughout the presentation of the sampling strategies we will define as x^*, the sample that is chosen as the most appropriate ***(least certain)*** for sending to the oracle. Furthermore, we define as X the set of the unknown elements and Y the set of different classes *(labels)*. Following from the example of the logistic regression we can define the decision rule for identifying the most uncertain training instance x^* in a more formal way:

$$x^* = argmax_X\ 1 - P_w(\hat{y}|x)$$

$$\hat{y} = argmax_Y P_w(y|x)$$

The above equations describe the **Least Confidence (LC)** strategy. As $\hat{y}$ we define the class that, under a model w, a training sample x has the greatest posterior probability of belonging to. Since least confidence strategy takes into consideration only the most probable class, it is not suited for cases where information about the remaining classes' distribution is important. A great presentation of such a case can be found in the work of Scheffer, Christian Decomain and Stefan Wrobel [22] regarding information extraction from textual documents using Active learning Hidden Markov models. The researchers used ***Margin Sampling*** to incorporate the probability of the two most probable classes in their decision rule. Using margin sampling, the most informative unlabeled training instance can be defined as follows:

$$x^* = argmin_X\ P_w(\hat{y}_1|x) - P_w(\hat{y}_2|x)$$

$$\hat{y}_1: the\ most\ probable\ class\ for\ the\ training\ instance\ x$$

$$\hat{y}_2: the\ second\ most\ probable\ class\ for\ the\ training\ instance\ x$$

What margin sampling implies, is that given an unlabeled training example, if the difference in the probability of the two most probable classes is big, then those two classes are separated by a large margin. This means that there is big confidence that the instance belongs to the most probable class and it is not worth the cost of sending it to the oracle for labeling. On the other hand, if the margin is small then there is little confidence about the true label of the instance and the model will have significant gain from learning its true label. Finally, the most widely used uncertainty sampling variant is ***Shannon Entropy*** [23]. The entropy of a training instance x is given by:

$$H(x) = -\sum_i P_w(y_i|x) \log P_w(y_i|x)$$

In physics, entropy represents the log of the number of states or configurations that a system can achieve. If the particles inside a system can move around multiple positions then the system has high entropy. If the stay fixed in a certain position the system has low entropy. In information theory, entropy can be defined as the opposite of knowledge. Imagine tricking a friend of yours into playing with an uneven coin that will only land on heads. You make a bet of getting 2 heads in a row. Of course, since it is your coin the probability of winning is high and so is your knowledge of the outcome. On the contrary, the entropy of the system represented by the series of throws is low. Overall, the probability of winning is 1 while the entropy is:

$$-(1 * \log_2 1 - 1 * \log_2 1) = 0$$

In the case of an even coin your knowledge of the outcome is much lower and so is the probability of winning:

$$P(win) = \frac{1}{2} * \frac{1}{2} = 1/4$$

While the entropy of the system is bigger:

$$-\left(\frac{1}{2} * \log_2 \frac{1}{2} + \frac{1}{2} * \log_2 \frac{1}{2}\right) = 2$$

In the case of active learning, choosing to label the training instance that holds the biggest entropy will give the greatest informational gain to the model. The most informative training sample is given by:

$$x_H^* = argmax_X - (\sum_i P_w(y_i|x) \log P_w(y_i|x))$$

The most obvious advantage of ***Entropy sampling*** over the ***least confidence*** and ***margin sampling*** is that it takes into consideration the full set of different labels that exist in the dataset. In the case of binary classification all three methods are mathematically equivalent. All the aforementioned strategies, use probability in order to calculate the most informative instance. This fact may trick someone into thinking that it is inappropriate to use Uncertainty sampling for non-probabilistic models. As it was mentioned in the beginning of this section, certainty or uncertainty can be translated in a number of different ways, two of them being the probability and entropy. In the case of SVMs, we can define certainty as the distance between the decision boundary and some unlabeled instance. Thus the most informative unlabeled instance will be the one closest to the decision boundary [17].

3.2.2 Query by committee

Another widely used selection strategy is the query by committee (QBC). In this setting, a set of classifiers trained on different configurations collaborate in order to choose the most informative sample. QBC's goal is to minimize the version space by using as few labeled training examples as possible. In the case of SVMs, given a set of labeled training data and a Kernel, there is a set of different hyperplanes that separate the data in the feature space [17]. This set of hypotheses is called the version space [24]. The QBC algorithm maintains a committee of different classifiers f_i. Each of the classifiers is asked to predict the label of a specific unlabeled instance. After the labelling process has finished, the instance on which the most members of the committee have disagreed upon is chosen as the next query to the oracle. In order for the strategy to be accurate, there is a number of issues to be considered. It has been proven [25] that for a set or ***ensemble*** of classifiers to be more accurate than any of its individual members, certain conditions must be met.

The first condition is that every classifier in the set has to be accurate. As an accurate classifier we define one that has better error rate than just random guessing the label of an instance x. The second condition is that the members of the committee have to be diverse. This means that the classification errors made by the members of the committee have to be uncorrelated. If not, the classifiers will rarely disagree and the voting process will not be productive. There are multiple approaches of constructing such a committee of classifiers. Using ***Bayesian Voting*** an ensemble can be defined as the set of all the hypotheses of the version space, each weighted by the posterior probability $P(f \mid S)$ where S is a training example. Ensembles can also be created by ***manipulating the training examples***. In this method the training set is broken down into separate subsets each of which is used to produce a different hypothesis. The way in which the subsets are created is specific to the different variants of this family of algorithms such as ***Bagging*** [26] and ***ADABOOST*** [27] [28]. Another technique for creating an ensemble of classifiers is to ***manipulate the input features*** instead of the training samples. Each hypothesis in the committee is trained on a subset of features usually ones that are related or have been observed using the same technique [29]. Finally, ensembles can be created by ***injecting randomness*** in the models' training. The most typical use of this method is in the domain of neural networks. Using the same training dataset but different initial weights can lead to an ensemble of neural networks that differ substantially from one to another [30]. An interested reader should read more about the different ensemble methods in machine learning in the paper by Dietterich [31] regarding Ensemble methods in machine learning. In order to conclude the QBC overview we need to present the different variants of measuring the committee's disagreement. Similar to entropy based uncertainty sampling, the most informative unlabeled training instance x^* can be considered the one that gathers the biggest ***vote entropy*** [32] across the different committee members. Formally, vote entropy can be defined as such:

$$x^* = argmax_X - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}$$

As y_i we define the various classes that exist among the training samples X, $V(y_i)$ is the number of votes that a specific label has accumulated and C is the size of the committee. Lastly, another disagreement measure is the average ***Kullback-Leibler (KL) divergence***. Generally, KL $D(p \,||\, q)$ describes the divergence between the probability distributions p and q. A common and easily digested way of describing KL divergence is as the distance between the two distributions. However, this is inaccurate and fails because, in comparison to real distance measures, the KL divergence from p to q isn't always equal to the KL divergence from q to p. What KL divergence actually measures is how much information each sample drawn from $p(x)$, will on average bring when trying to distinguish $p(x)$ from $q(x)$. In the active learning setting, the KL divergence is calculated between every committee member and the whole ensemble. The most informative sample is the one that its average classification from any one of the committee participants differs the most from the committee as a whole. KL divergence is defined as follows:

$$x^* = argmax_X \frac{1}{C} \sum_{c=1}^{C} D(P_{\theta_c} || \, P_C)$$

Where

$$D\left(P_{\theta_c} || P_C\right) = \sum_i P_{\theta_c}(y_i|x) \log \frac{P_{\theta_c}(y_i|x)}{P_C(y_i|x)}$$

3.2.3 Expected model change

The techniques discussed in the previous sections were mainly focused in discovering the most informative unlabeled training instances by using the current model to calculate some kind of informativeness measure e.g. entropy. Thus, they did not consider how the model would actually react to the chosen sample being added to the training set. They were in some way sample centric. Furthermore, it has been suggested that both uncertainty sampling and QBC can sometimes fail by querying samples that are only partially informative, meaning that they contain little information about the overall distribution of the dataset e.g. outliers [33] [34] [35]. To address this, the expected model change (EMC) query method chooses to query the unlabeled instance that would have the greatest impact on the model's state ***if*** we knew its label. For models trained using gradient descend, the informativeness can be naturally measured by the norm of the gradient imparted by an unlabeled instance. This approach is called the Expected Gradient Length (EGL) and it has been successfully used by Burr Settles and Mark Kraven in [36] for the task of Sequence labeling as well as for speech recognition [37]. For a labeled training set X and the model's parameters w, we define the gradient of the log-likelihood l as $\nabla l(X; w)$. For a new training example $< x, y >$ being added to the training set X the new gradient for l takes the form of $\nabla l(X^{+<x,y>}; w)$. Note that the query algorithm doesn't know a priory the true label of the added training example. To accommodate for this, we calculate the expected gradient length $EGL(x)$ over the N most probable labels:

$$EGL(x) = \sum_{y \in N} P(y|x\,; w)\ \|\nabla l(X^{+<x,y>}; w)\|$$

$$x^* = argmax\ EGL(x)$$

Where $\|\nabla l(X^{+<x,y>}; w)\|$ is the Euclidean norm of each resulting gradient. As it is stated in [36], since the gradient $\nabla l(X; w)$ will be almost zero from the previous training step, we can define, for computational efficiency purposes, that:

$$\nabla l(X^{+<x,y>}; w) \approx\ \nabla l(< x, y >; w).$$

3.2.4 Expected Error Reduction

Inevitably, no matter what query selection strategy we choose, the criterion for an accurate model is its performance error. However, none of the previous methods have taken this measure under consideration and the reason is that closed form calculation of the expected error is hard to control. Expected error reduction (EER) sampling method, aims to select a query x^* such that when its true label y is incorporated in the training set X, the model that is re-trained using the expanded dataset ($X + < x^*, y >$) will have a lower error than if any other sample x' had been chosen. Since the true label of x^* is unknown before making the query, we have to make an expectation calculation of the estimated error for each possible label y, weighted by the current model's posterior probability $P_\theta(y|x)$. The resulting algorithm starts by training a classifier from the initial training labeled examples. Then, for each unlabeled example x, a pair of $< x, y_i >$ for every possible label y_i is created and added to the training set. The model is retrained using the new training set and the resulting expected loss is calculated. The average expected loss, weighted by the model's posterior probability $P_w(y|x)$ is assigned to the current training example x. Finally, the unlabeled example x that scored the lowest expected error will be sent to the oracle for labelling. There are multiple issues with the above algorithm that have to be remedied in order to make it efficient. The cost of incrementally training the model has to be smaller than the cost of retraining the model using the new enlarged dataset (this applies to every active learning algorithm). Additionally, the model needs to reclassify only those training instances that the predictions about them are likely to change due to the additional training examples. Since the unlabeled pool is usually large, calculating the error for every instance in the pool after a candidate query is added is very costly. To make this efficient, the calculations need to be made only for those instances in the neighborhood of the candidate query. That way the algorithm also saves the calculations that otherwise would have been made for uninformative training samples such as outliers. Generally, there are two approaches of calculating the expected error. The first approach is to minimize the expected 0/1 loss:

$$x^*_{0/1} = argmin_X \sum_i P_w\,(y_i|x)(\sum_{u=1}^{U} 1 - P_{w^{+(x,y_i)}}(y|x^{(u)}))$$

where $w^{+(x,y_i)}$ is the model after it has been retrained with the added training tuple $(x,\ y_i)$ and U is the pool of unlabeled samples. The second approach is to minimize the expected log-loss:

$$x^*_{log} = argmin_X \sum_i P_w\,(y_i|x)(-\sum_{u=1}^{U}\sum_j P_{w^{+(x,y_i)}}(y_j|x^{(u)}) \log P_{w^{+(x,y_i)}}(y_j|x^{(u)}))$$

which aims to maximize the informational gain of the query or equivalently to minimize the expected entropy over the unlabeled pool U. Expected error reduction strategy has been extensively employed in research papers. Roy and McCallum in [38] have proposed this strategy for text classification using naïve Bayes. Later on, to tackle the problem of having to cycle through all the unlabeled samples, the researchers used Monte Carlo sampling to reduce the size of the unlabeled pool. Guo and Greiner [39] employed logistic regression to create an active learning framework, based on error reduction strategy that could change its selection rule in order to cope with unexpected labels. Moskovitch et al in [40] made use of SVMs in order to quickly identify malicious code (Worms). In their setting, the error reduction strategy was utilized to reduce the noise in the training set and the business need of having human experts to label new and unknown worms.

3.2.5 **Variance Reduction**

Due to the ***lack of a closed form*** expression, the Expected error reduction strategy can be computationally expensive since both of the described equations need to be calculated for each of the different unlabeled instances and possible classes. In addition to that, in order to test the effect that every newly added training instance has over the model's error, a new version of the model has to be incrementally trained. Gemen et al. [41] showed that the overall generalization error can be expressed as the sum of the true label noise, the model's bias and the model's variance. Of those three terms, only the variance is highly dependent on the training examples. Thus, we could create an active learning strategy that chooses to query the training instances that mostly reduce the variance and in doing so the model's generalization error. Cohn et al. [42] [35] provided the first statistical analyses on active learning using the model's variance and proved that it can be calculated in closed form for a variety of different models such as neural networks, Mixture models and linear regression. Specifically, the variance can be expressed in terms of the gradient with respect to the model parameters and the Fischer Information Matrix [43]. Known works using Fischer information matrices in the context of active learning are those by Schein and Ungar [44] and Zhang and Oles [45]. Specifically, Schein and Ungar run experiments on the most popular heuristic active learning strategies such as query by committee and uncertainty sampling as well as experimental methods like variance reduction (also known as A-optimality). They report that the heuristic methods produced mixed results, performing ***even worse*** than random sampling in some stages of the evaluation, with ***margin sampling*** being the most promising and cost effective of all variants. Additionally, they note that variance reduction strategy proved to be too computationally expensive, both in memory and complexity, prohibiting the evaluation on two of the larger document classification tasks.

3.2.6 **Density Weighted methods**

Another family of sampling methods that, along with variance reduction and expected model change, overcome the issue of querying outliers are the density weighted methods. Density weighted methods prefer the unlabeled instances that are ***representative*** to the overall unlabeled dataset's distribution and essentially guide the learner to concentrate on the most ***important*** uncertain data instead of the most uncertain data. Based on this strategy McCallum and Nigam [46] propose a QBC learner for text classification that chooses to query an instance that not only achieves ***high*** classification variance among the committee members, but is also ***similar*** to many other instances of the unlabeled pool. In their work, the density around an unlabeled instance is approximated by calculating the average distance ***(exponentiated KL divergence)*** from that sample to all others. Another way to formalize similarity between training samples is by taking advantage of the natural clusters that exist in the unlabeled dataset distribution. Clustering can be pre-computed without any human interaction and stored for later use. The information obtained through clustering can be utilized in a number of ways. As far as labelling is concerned the centroids of each cluster can be considered as the most important instances. Moreover, since unlabeled instances that belong to the same cluster are likely to share the same label, the algorithm is relieved of the cost of having to label a great number of them. Fujii et al. in [47], tackle the problem of word sense disambiguation using an active learner, based on an interpretation certainty measure similar to the one proposed by Lewis and Gale in [18]. In addition to that, the researchers incorporated the measure of similarity, in the form of the number of nearest neighbours when choosing the optimal query. For an unlabeled instance x to be considered optimal it has to be in the same neighborhood as the most unlabeled instances. In other words, x is optimal when the total interpretation certainty of the unlabeled instances increase the most after the model has been trained with it. Since x is similar to all other instances in the neighborhood, the algorithm doesn't have to loop through them too, thus becoming more efficient. Similarly, their active learner chooses to avoid instances x' that may have great number of unlabeled neighbors but are similar to instances that have already been labeled. Xu et al. [48] developed a formula that incorporates three kinds of measures for choosing the best query in the field of document relevance feedback. Apart from using KL divergence for calculating the relevance of an unlabeled instance x, the researchers used J-divergence [49], to calculate two additional measures, the density and the diversity of the sample x. The density of the area around a specific document is measured by calculating the average distance from that document to all others. The J-divergence for two documents is defined as:

$$JD(d_i \| d_j) = KL(d_i \| d_j) + KL(d_j \| d_i)$$

and the density of the area around a document d_i:

$$density(d_i) = -\frac{1}{\|D\|} \sum_{j \in D} JD(d_i \| d_j)$$

The diversity of document d_i and a document set is measured as the minimum distance (J-divergence) between d_i and any member of that set. Nguyen and Smeulders in [50] have used k-medoid clustering to find K representatives so as to minimize the sum of the distance from the data samples to the nearest representatives. Afterwards, logistic regression is deployed to calculate the probability of a cluster instance to be of a specific label in order to propagate this information to the rest of the cluster members. Finally, the criterion for choosing the data to be labeled gives priority to samples near the decision boundary and to centroids of ***dense*** clusters. In the work of Zhao Xu et al. [51] the use of plain uncertainty sampling is characterized as 'myopic' since it greedily selects the unlabeled instance closest to the decision boundary, without taking into consideration the underlying data distribution. Moreover, they note that the structure of the natural clusters could be very beneficial to the training process. In the algorithm for text classification that they propose, a SVM classifier is trained on the labeled dataset. Then, a k-means clustering is performed to the unlabeled data points that lie in the margin of the SVM classifier. The unlabeled instances that lie closest to the centroids of the clusters are considered the most informative and are sent to the oracle for labeling.

3.3 CONSIDERATIONS REGARDING ACTIVE LEARNING

As it was mentioned, Active Learning aims to make the training of machine learning algorithms more economical, both in time and other resources. However, as such practices become more popular and integrated into real-world working systems, many of the assumptions made in the various studies are violated. A common issue that arises, is that the training sets being generated by means of an active learning framework are closely tied (biased) to the model and the selection strategy that was used to query the training examples that compose them. Therefore, the generated dataset's distribution does not fully represent the real one. Thus, in the case of swapping the current model for another one, the actively annotated training set might not be as effective as before. Such a case can be found in the work of Baldridge and Osborne [52], where a training set created by a parser model leads to worse training results when used to train different parser models. By sacrificing computing power (which is not always an option), we can reduce the bias of the generated training sets using an ensemble of classifiers. Intuitively, an ensemble of heterogeneous classifiers is bound to produce a training set that is more general compared to one that was gathered by a single model. However, there are some studies reporting opposite results and indicate that datasets collected by an active learner have positive effects when used to train a different model [53] [54] [18]. For example, Lewis and Catlett [55] trained a decision tree using a dataset assembled by a naive Bayes model using uncertainty sampling. They report that a decision tree model trained with the aforementioned training set produced higher accuracy than another decision tree that was trained with a ten times larger, randomly selected dataset.

Another parameter that should be taken into consideration is the quality of the oracle. The foundation of active learning is based on the assumption that the oracle produces labels of high quality. However, no matter which form the oracle takes, e.g. human or machine, some unlabeled training examples can be difficult to determine. Moreover, in the case of human oracles the quality of labels is correlated to the physical condition of the annotator e.g. confusion, fatigue. Recently, the use of mechanisms such as Crowdsourcing [56] by the means of tools like Amazon Mechanical Turk (AMT) [57], suggest the use of multiple annotators. Remember that one of the motives of active learning is the economical procurement of labeled data. There are cases where it is more economical to acquire labelings from multiple non- expert annotators than by a single expert oracle. These approaches introduce even more issues regarding the quality of the labeling process. Some annotators may be more reliable than others or some may be intentionally misinforming the model (case of internet chatbots using ambiguous user input for training purposes). Therefore, a need arises for machine learning models to be annotator aware. In simpler terms, when multiple annotators are available, the model should be able to choose the most suitable annotator that will mostly improve the training process. Some works tackle this by suggesting the ability of the active learner to query multiple times the true label of an unlabeled training example [58] [59] [60]. They don't however take into consideration the properties that characterize the individual oracles. Such an approach can be found in the work by Yan et al in [61] where after selecting the most informative unlabeled training example, the active learner attempts to determine the annotator that will provide the most confident label. The model they propose, utilizes the training data X, the set Y that contains all the ***observed*** labelings of all annotators and produces an estimation of the ground truth Z, a classifier for labeling any new instance x and one more model for estimating the annotator's quality (expertise) given the input x.

So far, every query strategy that we have described selects queries one at a time. It is obvious that in practice this can be highly inefficient as the learning process is being stalled by the amount of time the oracle takes to label a single element. Additionally, consider the case of multiple available annotating environments with the ability to perform labelings in parallel. In order to make use of the above capabilities, it would be preferable to send a larger number of queries (batch) for labeling. The most simple and naïve approach is the Q-best where the active learner sends the Q best instances to the oracle. Like uncertainty sampling where subsequent queries can unwittingly refer to similar points of the distribution, the Q-best approach suffers from information overlap among the instances of the batch. Similar to 3.2.6, other batch selection techniques remove this problem by incorporating diversity among the instances of the batch and choose the ones that are both informative and diverse. Researches regarding Batch selection strategies when using a SVM active learner can be found in the works by Brinker [62] and Xu et al. [48] which construct their batches incorporating diversity and density measures respectively. Unlike the heuristic scores of diversity and density utilized by the aforementioned batch selection strategies, Guo and Schuurmans [63] face the construction of the batch as a continuous optimization problem. Their problem formulation aims to maximize the discriminative classification performance of the target classifier while taking the unlabeled data into account. A similar approach using discriminative models can be found in the work by Wang, Zengmao, et al in [64] regarding Hyperspectral Image Classification.

There are settings where the input of the training process is not composed of training examples that are individually labeled. Instead, the learner receives labeled ***bags*** of features. This area of machine learning is called ***Multiple Instance (MI)*** learning and presents a way of reducing the need of exhaustingly labeling datasets by widening the level of annotation granularity. For a bag to be labeled as negative, all of the contained examples have to be labeled as negative. On the other hand, for a bag to be labeled positive at least one of the training examples has to be positive. This kind of approach has been used in the field of Content-based image retrieval and text classification tasks. In text classification, a document represents a bag of instances which in turn are represented by short passages inside the document. A text document is considered to be positive if any of the passage(s) inside it are of the target class. It is obvious, that the biggest problem in MI is that for an accurate model to be produced the learner must be able to determine which instances in positive bags are actually positive. Since labels can be needed in various levels of detail, there is a chance that the cost of annotating some or all passages inside the various training examples can be prohibitively high. To reduce the labeling cost of annotating datasets for MI we must consider the formulation of active learning scenarios that could fit in such cases. As it was mentioned, MI oriented datasets are labeled in a lesser level of detail than typical training datasets.

There are active learning MI approaches that allow the learner to query the oracle for labels at varying levels of granularity. Settles, Craven and Ray [65] argue that while MI learning reduces the effort of labeling by providing labels at a rougher level of detail, it appears that selecting some instances of finer granularity can improve the learning process. Their research is targeted on the scenario where the learner is only allowed to query for instances inside positively labeled bags. After experimenting with several known active learning strategies (Uncertainty sampling, Expected gradient length) as well as a variant of their own called MI Uncertainty they note that all active learning methods perform significantly better than passive learning using random sampling. Moreover, their proposed approach (MI Uncertainty) takes into consideration not only the uncertainty regarding an instance's class but also its contribution to allowing the learner to 'explain' the bag that it belongs. That way it tackles oxymoronic cases of querying the most uncertain training instance even though it belongs to the most positive bag. Liu, Dong, et al. [66] reference Settles, Craven and Ray's report [65] and add that MI learning can also benefit by using a mixed approach of labeling instances and bags simultaneously. In their paper they compared three different selection strategies: selecting instances only, selecting bags only and selecting a combination of bags and instances. To accomplish that, they created a unified MI active learning framework based on SVMs that could estimate the similarity between instances only, bags only, as well as instances and bags. Additionally, for the reasons discussed in 3.2.6, their selection measure combined uncertainty, diversity and novelty. Using various numbers of labeled training examples, the mixed selection approach outperformed the others by scoring the greatest AUROC (Area under ROC) score. Also, as expected, all active learning strategies outperformed random sampling.

The varying levels of image annotations motivated Vijayanarasimhan and Grauman's work in [67] to address the need for the learner to be able to adopt to the multiple levels of annotation granularity and also decide ***which unlabeled item and at which annotation detail*** is best to be queried while being cost-effective. Their results show that this multi-level query approach achieves significant improvements at a lower cost over all the compared methods (Single level active learning, Single level random sampling and multiple level random sampling) and also improving accuracy with as many as ten labeled instances. Apart from querying instances, Raghavan et al. [68] introduced the concept of querying features in addition to instances in the setting of text categorization. The reason for this approach, is that human annotators perform much faster when annotating features rather than instances or documents. Moreover, in general concepts such as 'sports' you don't need an expert oracle to distinguish the discriminative value of a word (cost reduction) e.g. 'is the word basketball able to sufficiently describe documents about sports?'. The feature oracle has knowledge of all document labels and using information gain (entropy) as a measure of importance, it produces a ranked list of features in decreasing order of importance for each document. Their algorithm combines labeling of features and documents and their results indicate: i) improved performance over uncertainty sampling with as few as 7 labeled examples, ii) naïve users can provide effective feedback on the most relevant features, iii) the average time needed to label features takes one fifth of the time needed to label documents on average.

Last but not least, throughout this report it becomes apparent that the majority of the proposed algorithms and selection measures require for sophisticated implementations in order to work efficiently. So, the reader should also consider the software engineering overhead required for building such algorithms.

4 EXPERIMENTAL EVALUATION

This paper would not be complete without presenting applied examples of some of the mentioned query selection methods. Since a whole chapter was dedicated to the formalization of SVMs it is only fitting to use SVMs as the model in our experiments. Moreover, their geometrical foundation makes them suitable for illustrating some of the most popular selection strategies like Margin Sampling, as well as some variants of Density Weighted sampling methods such as the one proposed by Zhao Xu et al. [51]. All models, performance measure techniques and dataset manipulation methods used for the experiments come from the Python scikit-learn machine learning library [69]. The classifier in use is a Support Vector Classifier whose implementation is based on libsvm [70]. The major part of the active learning selection strategies implementation comes from Google's active learning project available on GitHub[3].

4.1 BASELINE PRESENTATION

The following section uses toy datasets and provides the reader with illustrative examples that depict the model, training sets, unlabeled datasets and selected queries. Thus, it serves as a baseline before presenting the results when testing the same methods against benchmark datasets like the MNIST database.

4.1.1 Margin Sampling

Even amongst active learning sceptics' works, margin sampling's simplicity both in inception and implementation, has been shown to outperform other more sophisticated and elaborate query selection strategies [44]. In short, margin sampling considers as the most informative unlabeled instance, the one that lies closest to the decision boundary. This is often mentioned as its biggest flaw, since it is characterized as a rather myopic approach that doesn't take into consideration the underlying class distribution. Afterwards, the selected query or queries (in the case of batch selections) are sent to the oracle for labeling, the queried instances are added to the training data and the training procedure continues. There are two approaches regarding the handling of the training after the new instances have been added. The model can either be retrained entirely or incrementally. Ideally, in order to save time and computational resources the learning phase should not start fresh after the newly labeled item is added to the training set. Of course, this applies to every active learning algorithm no matter the selection strategy in use. In the case of

[3] Google/active-learning
https://github.com/google/active-learning

SVMs incremental training can be achieved by readjusting the model's parameters after recalculating the gradients using a single training instance. An example of such an implementation can be found in the work by Shalev-Shwartz, Shai, et al [71] and their proposed algorithm called Pegasos. The Pegasos algorithm is a variant of Stochastic gradient descend where in each training step the model's parameters are readjusted considering only a single ***randomly*** picked training instance. In our case, due to limitations set by the used libraries the notion of incrementally training the model was not put into consideration when implementing the various learning strategies. Figure 11 depicts a sample dataset that will be used to illustrate the way margin sampling chooses the best query.

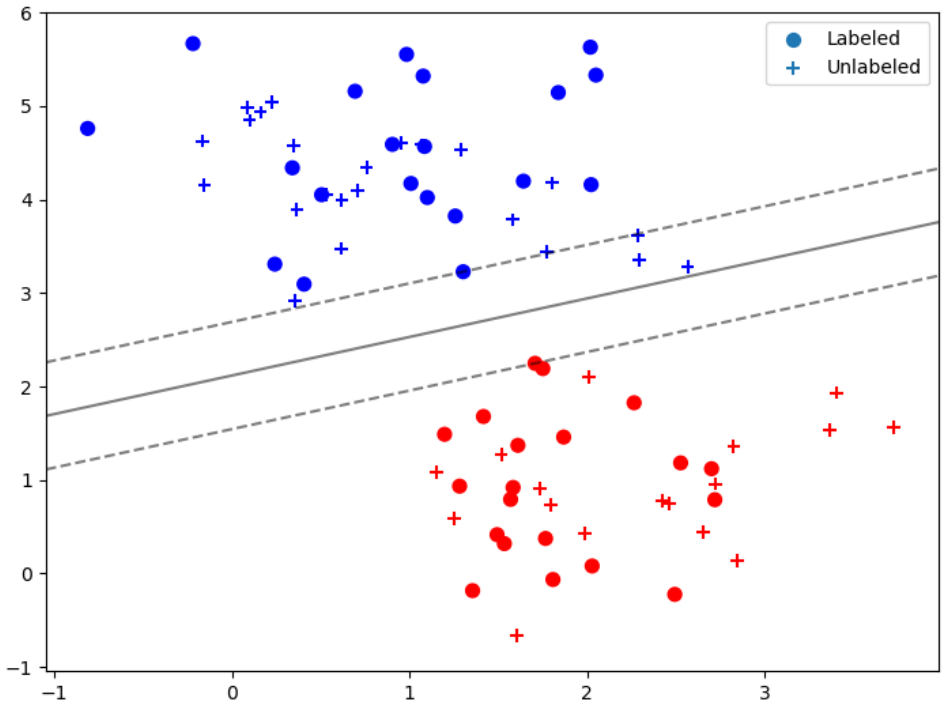


Figure 11 Sample Dataset for Margin Sampling and Decision Boundary

The dataset consists of two classes, class 'red' and class 'blue'. The points represented by the dots are used for the initialization the SVM classifier. The resulting decision boundary is used to calculate the distance from each unlabeled point, represented by the crosses. The chosen unlabeled instance that lies closest to decision boundary is depicted in Figure 12. In the case of multiclass classification problems, we choose the unlabeled instance with the smallest margin for the two most probable classes.

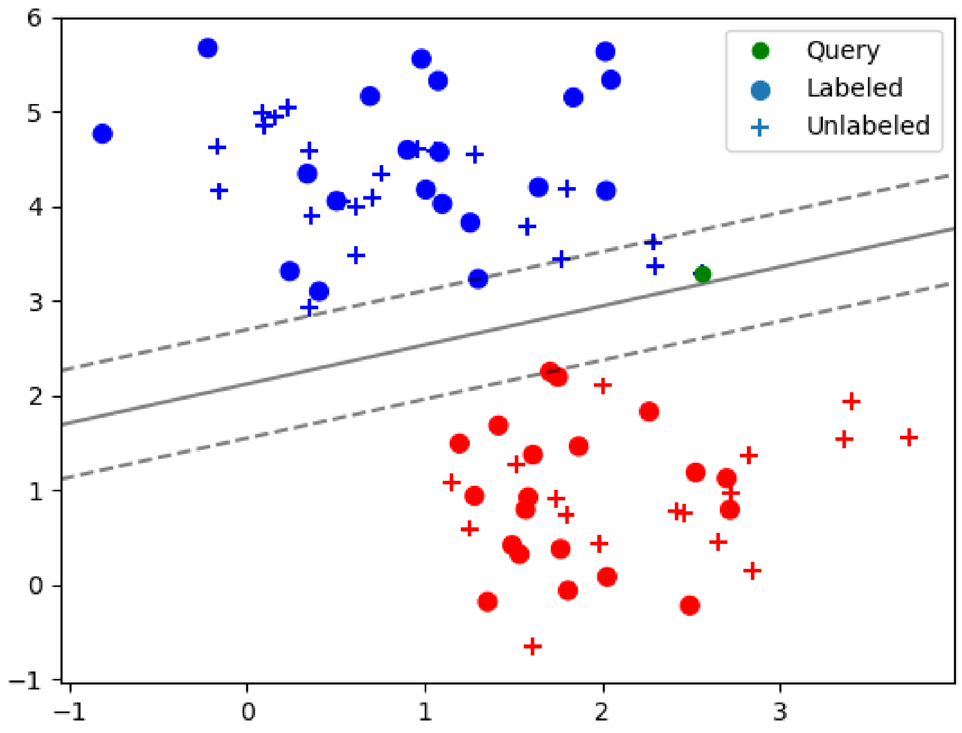


Figure 12 Most Informative query using Margin Sampling

4.1.2 Representative Cluster mean sampling

Due to the way that margin sampling chooses the queries, there is the possibility for the resulting, actively labeled dataset to be biased in favour of some of the underlying classes. To negate this effect, the selection criterion has to be aware of how the different classes are distributed inside the area of interest, in our case the area inside the margin. Representative cluster mean sampling, tries to gain knowledge of the underlying data distribution inside the margin by clustering the labeled points that lie in it. After the initial training, a k-means clustering is performed in order to acquire the cluster centroids who are considered to be the points that concentrate the most information. Figure 13 illustrates the state of the query strategy after the completion of the clustering procedure of a dataset consisting of two different classes.

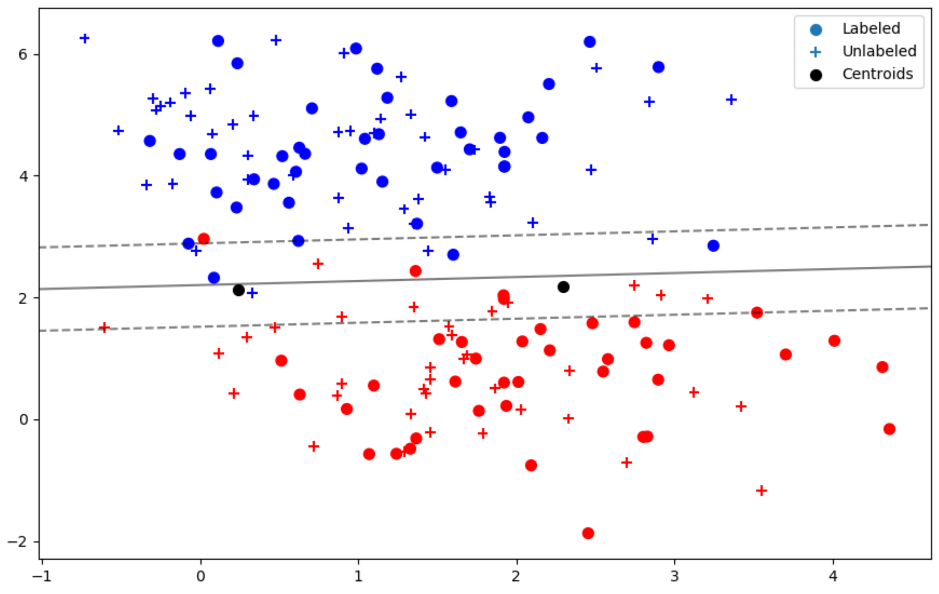


Figure 13 Centroids of points inside the margin

After acquiring the cluster centres and since they are not actually members of the dataset, the selection algorithm chooses the unlabeled instances (batch) that lie closest to them. The chosen instances are considered as the most informative as well as representative in regards to the underlying distribution (Figure 14) and are sent to the oracle for labeling. In cases when the points inside the margin are less than the requested batch size the selection strategy reduces to plain margin sampling.

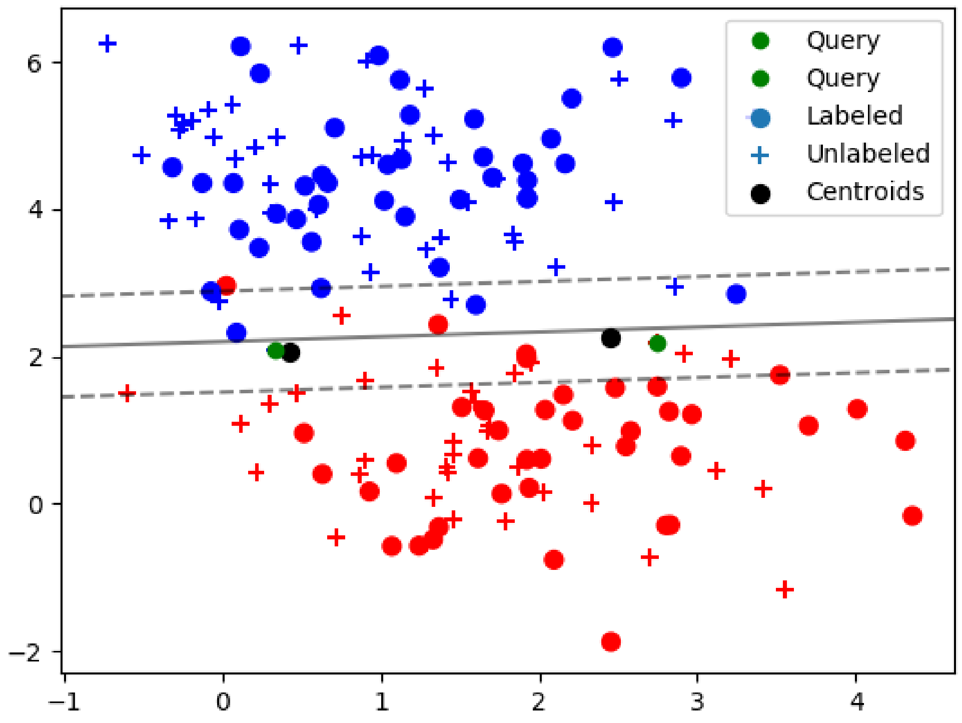


Figure 14 Most Informative query using Representative Cluster Mean Sampling

4.1.3 **Informative and diverse sampling**

Similarly to Representative Cluster Mean Sampling, this method aims to minimize bias in the actively queried dataset. The sampler in question collects points with small margin and creates batches of queries whose distribution respects the one of the entire dataset. Of course with the batch size is set to one, the selection strategy is identical to Margin Sampling. Before the sampling procedures initiates, the algorithm applies a k-means clustering to the unlabeled pool and calculates the probability of each cluster. In essence, this is fraction of each clusters' members to the overall number of instances. Following the initial training of the Support Vector Classifier, the unlabeled instances are sorted by their distance from the margin and are iteratively used to calculate the impact that they would have to the batch's distribution if they were to be added to it. In the case that their addition doesn't alter the various cluster distribution, meaning that the probability of each cluster stays true to the original dataset, the selected instances are considered as good candidates and are officially members of the batch. This procedure is repeated until the batch's size reaches a predefined size or until the pool is exhausted. Figure 15 illustrates a sample dataset composed of 3 classes that will be used to illustrate this strategy. In order to make the underlying distribution more obvious we remove the labeled instances that will be used for the initial training (Figure 16).

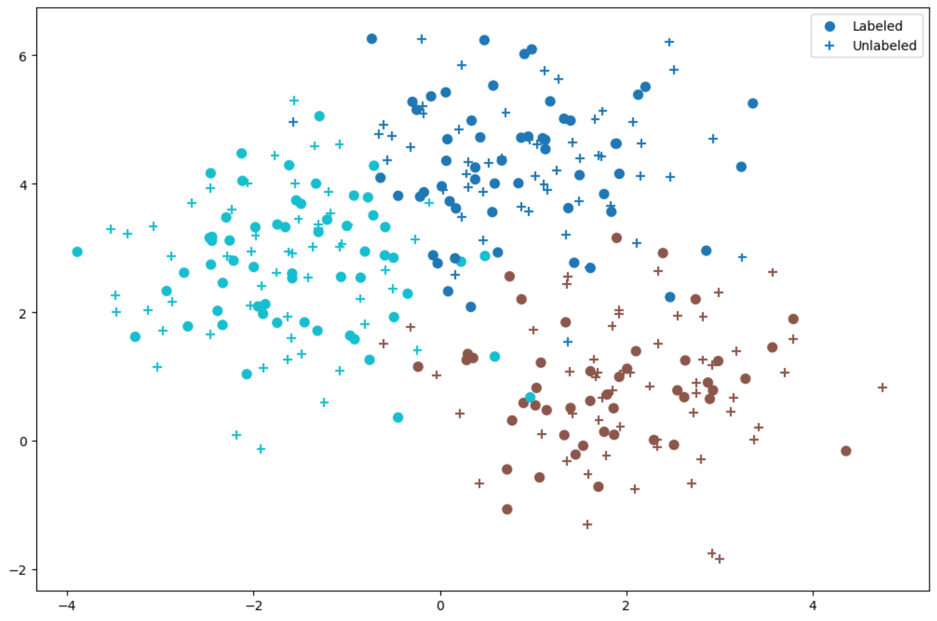


Figure 15 Sample Dataset for Informative and Diverse Sampling

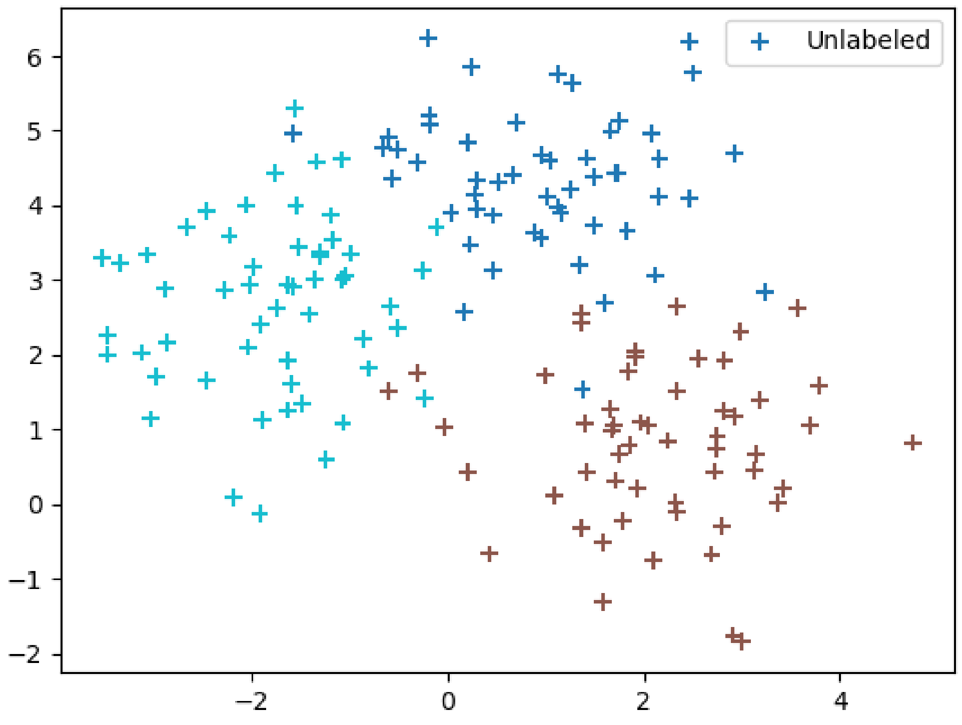


Figure 16 Unlabeled points only Sample Dataset

After applying the aforementioned algorithm for a batch size of 10, we depict the chosen queries in Figure 17.

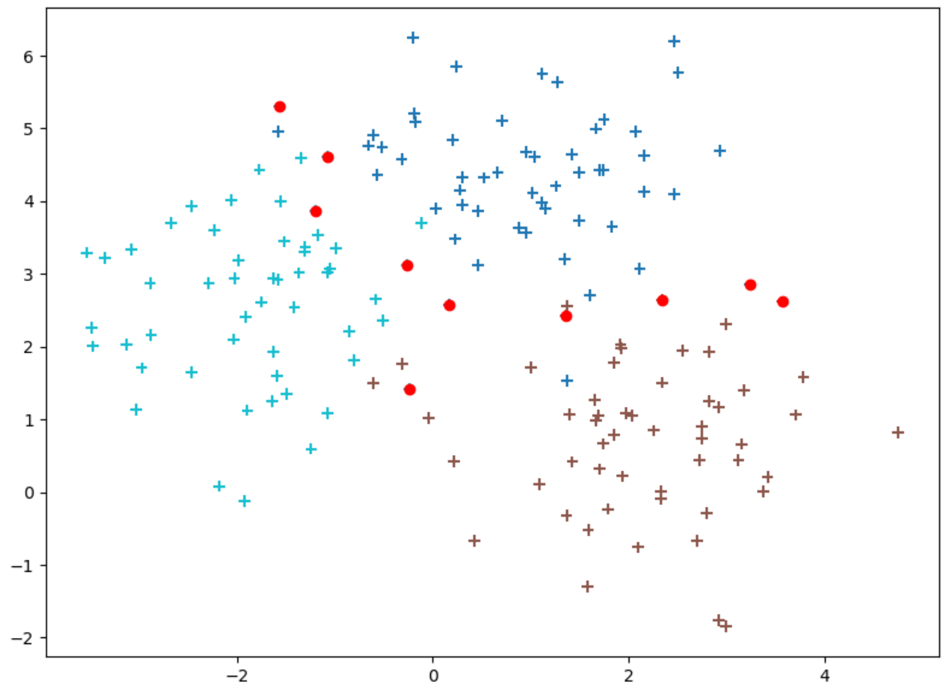


Figure 17 Most Informative queries using Informative Diverse Sampling

Since Informative-Diverse Sampling shares the same core as Margin Sampling, it would be interesting to compare their respective queries when performed on the same dataset. Figure 18 demonstrates the queries made by Margin sampling for a batch of size 10. The reader should notice the concentration of queries that lie in the area between the classes represented by the brown and dark blue color. Using the knowledge obtained by training on those new points the resulting classifier should be able to distinguish these two classes more efficiently than before. From the remaining 4 queries, all of them are related to the separation of the classes in the shades of blue However, none of the selected instances will aid in the distinction of the brown class from the light blue class. This is the result of Margin Sampling blindly choosing the points lying closest to the decision boundary and will cause the expanded dataset to be biased towards the remaining classes. On the other hand, Figure 17 shows that the queries in the batch selected by the Informative-Diverse Sampling are more spread out across all areas that lie close to the decision boundaries. Thus, the resulting dataset will show more respect to the dataset’s original distribution.

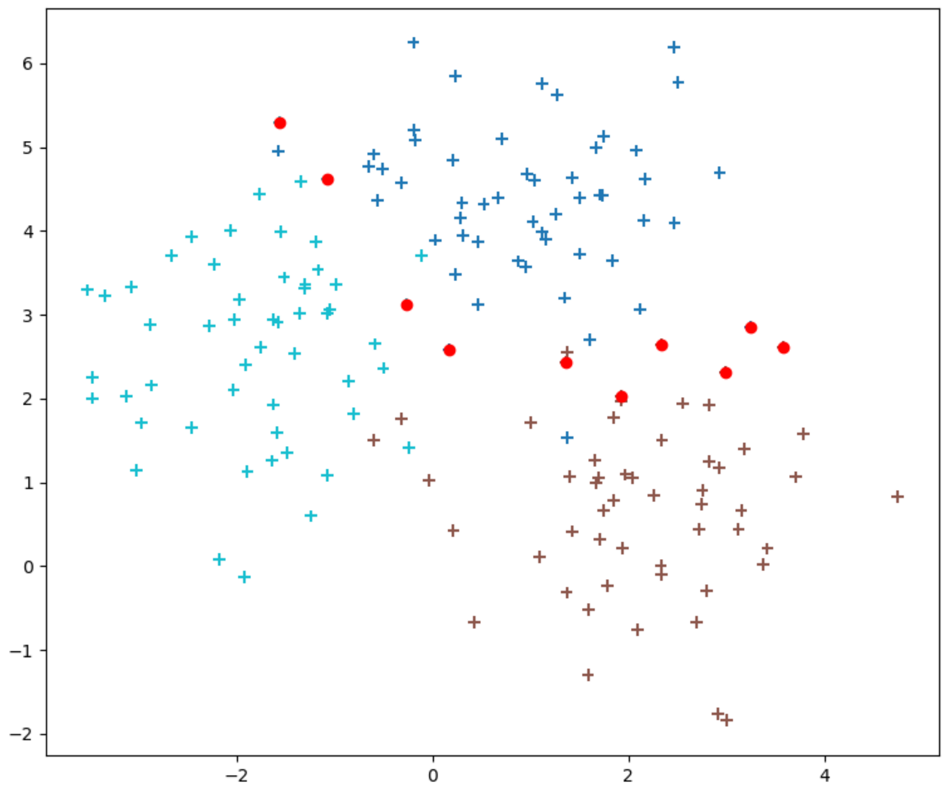


Figure 18 Margin Sampling selected queries

4.2 EVALUATION ON MNIST DATABASE

For the task of evaluating the Active Learning strategies illustrated in 4.1 we used the MNIST (Modified National Institute of Standards and Technology) database. MNIST is consisted of 70000 28x28 images of handwritten digits and is commonly used for the training and testing of models in the field of Machine Learning. There has been a great number of research works with the goal of achieving the lowest error rate when using this dataset, with the best so far achieving an error rate of 0.23 percent using a committee of 25 Convolutional Neural Networks [72]. In our case rather than striving to achieve the best accuracy possible, our interest is focused on the effect that Actively queried samples have on the training of a classifier when compared to Passively (Randomly) sampled ones. Our tests will vary from one to another in terms of the number of batches and the size of batches and will illustrate the fluctuation of accuracy after each new batch is added to the model's training set. To make things fair, the training sets are equivalent among all classifiers and consist of just 50 training samples. Lastly, all classifiers no matter the selection strategies have been tuned using the same hyper-parameters(Table 1).

Kernel	**γ**	**C**
RBF	0.001	2.8

Table 1 Model's Training Hyper-parameters

Right from the beginning it was stated that the goal of Active Learning is the economical procurement of labeled data. The most obvious way to accomplish this is by achieving great accuracy with the need of labeling as little as possible unlabeled instances. Figure 19 illustrates the performance of Active learning strategies when compared to Passive sampling after 10 rounds of additional training with batches consisted of 5 sampled instances.

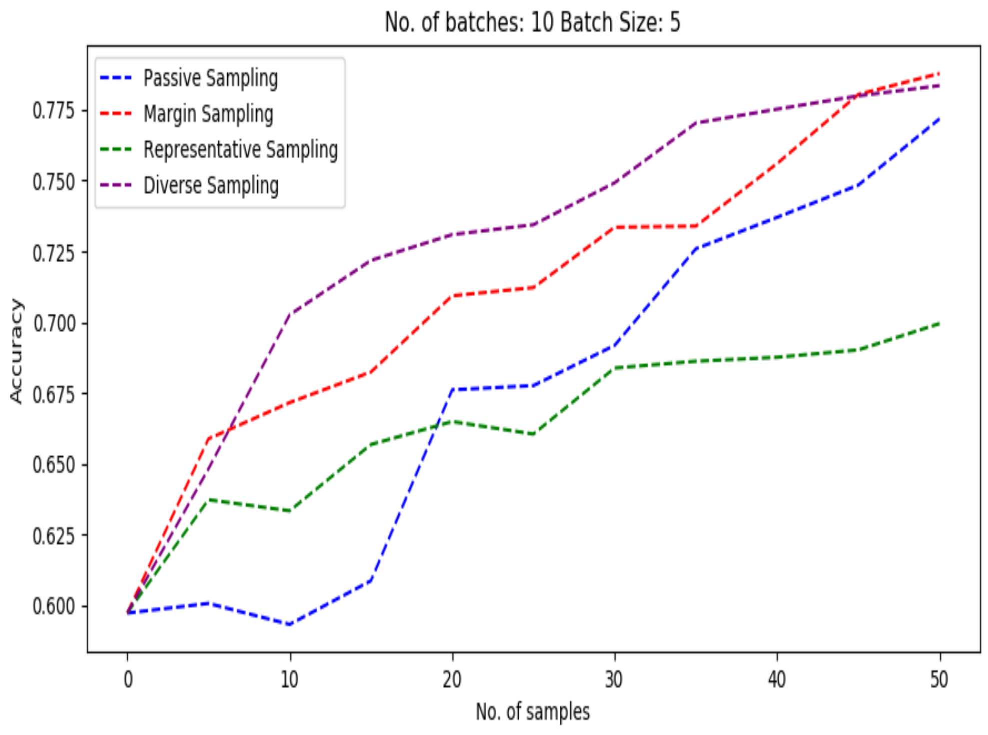


Figure 19

The most important thing to note is how immediate is the increase in the accuracy achieved by all active learning strategies. Passive learning needs 4 times the number of instances to reach the same level of performance that Margin and Diverse sampling achieve after the first batch. Figure 20 shows how Margin and Diverse sampling continue achieving superior performance even after doubling the number of batches. In contrast to these two strategies, Representative Sampling, is outperformed in most training stages apart from the initial ones. All experiments indicated that as batch size increases, Passive Sampling seems to close the gap in performance (Figure 21). Eventually in Figure 22 the amount of the information gained by big batches helps the Passive learning model to achieve equivalent levels of performance to the actively trained ones.

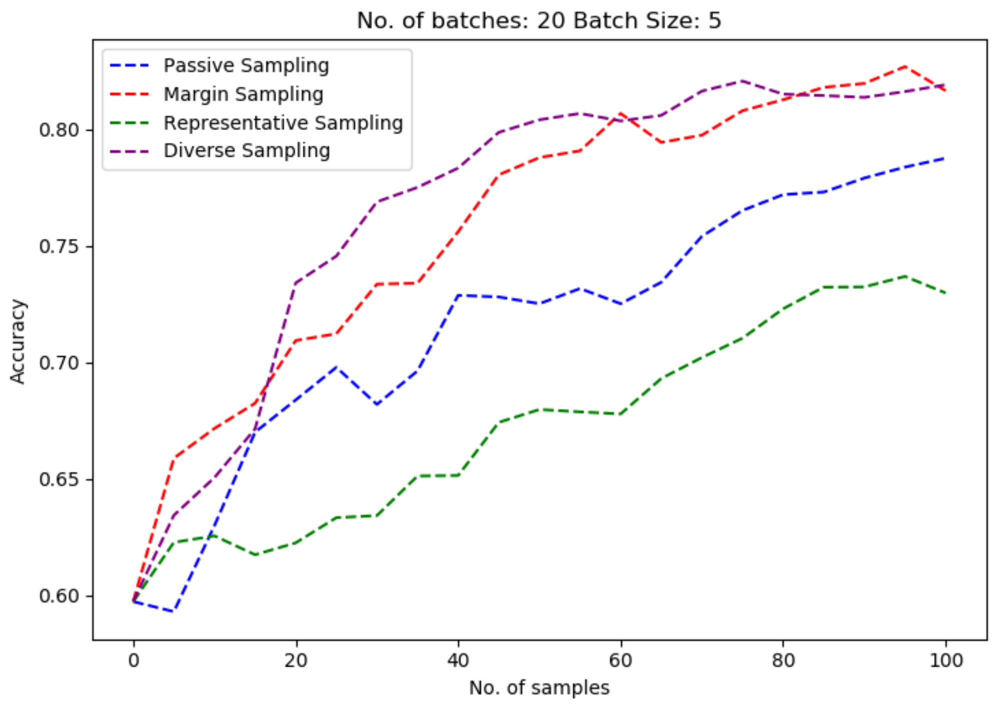


Figure 20

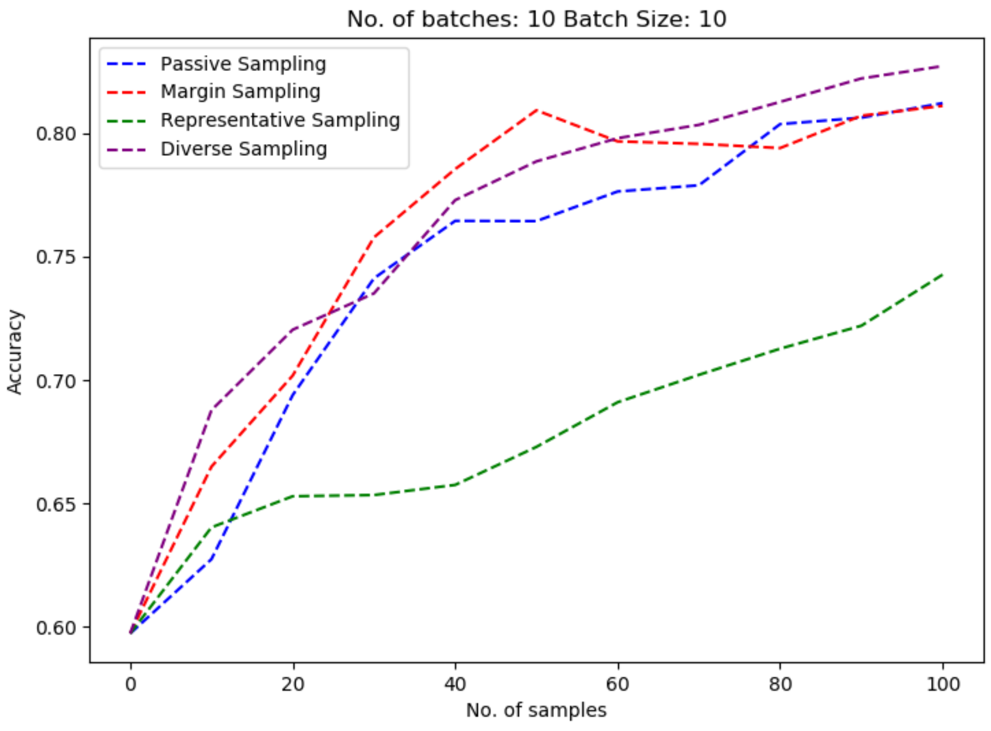


Figure 21

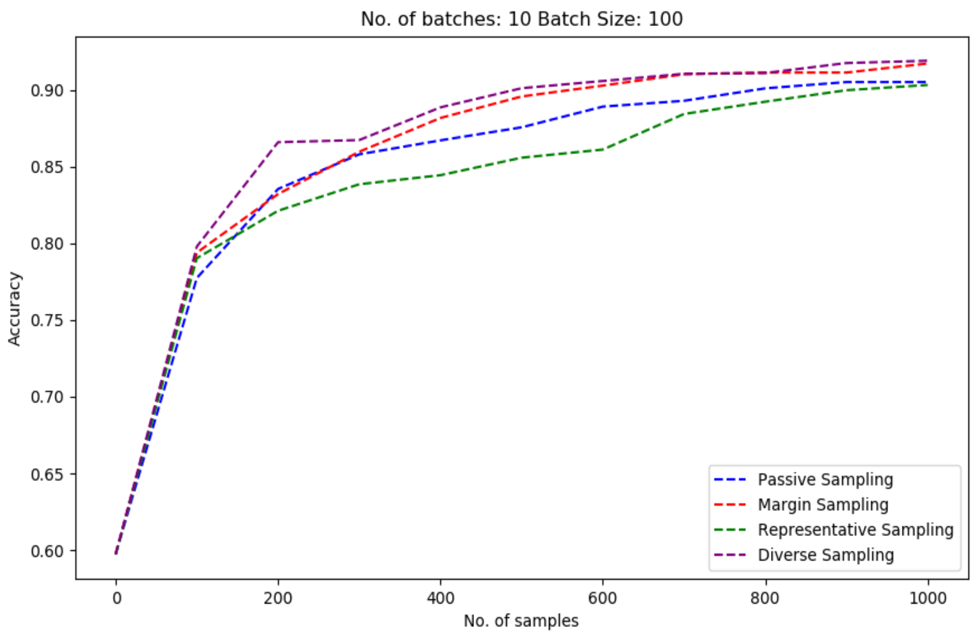


Figure 22

4.3 Evaluation On Cifar-10

To enhance previous findings, we ran additional experiments on a larger and generally more challenging dataset. The cifar-10 dataset consists of 60000 32x32 colour images separated in ten different classes that depict various objects like cars, trucks etc as well as animals like cats, dogs and horses. The dataset is broken down in 50000 training instances and 10000 test instances. Cifar-10 has been extensively used for evaluating various supervised machine learning algorithms with Convolutional Neural Networks [73] achieving the best performance so far. Like the previous tests, our supervised training model is a Support Vector Classifier (SVC) configured with the hyper parameters depicted in Table 2 and Table 3. The original training of all models (Passive and Active) was performed on a training set composed of 1000 randomly sampled training examples. Unlike the experiments performed on MNIST where the results showed a consistency of performance between the various tested techniques, the results using cifar-10 are somewhat mixed and related to the configuration of the model. Using the configuration depicted in Table 2, the performance of Active strategies is often worse than passive sampling between different runs (Figure 23, Figure 24). An interpretation of this could be that the initial seed of data is important to how the model is shaped during the rest of the training especially in cases like ours where the additional batches are of limited size and cannot rectify the overall performance. However, it seems that active learning strategies help the model into keeping a more balanced performance between different runs, even though they might get outperformed. What is also intriguing is the bad performance of Margin Sampling. This can be attributed to the fact that cifar-10 contains objects similar to one another. For example, we can imagine that the decision boundary between the classes of cars and trucks will have a great number of instances close to it. Since Margin Sampling cannot balance the batch's class distribution, some batches will be heavily biased towards collecting information for these specific classes.

Kernel	**γ**	**C**
RBF	0.001	1

Table 2. Configuration 1 of Model's training Hyper-parameters.

Kernel	**γ**	**C**
RBF	0.001	10

Table 3. Configuration 2 of Model's training Hyper-parameters.

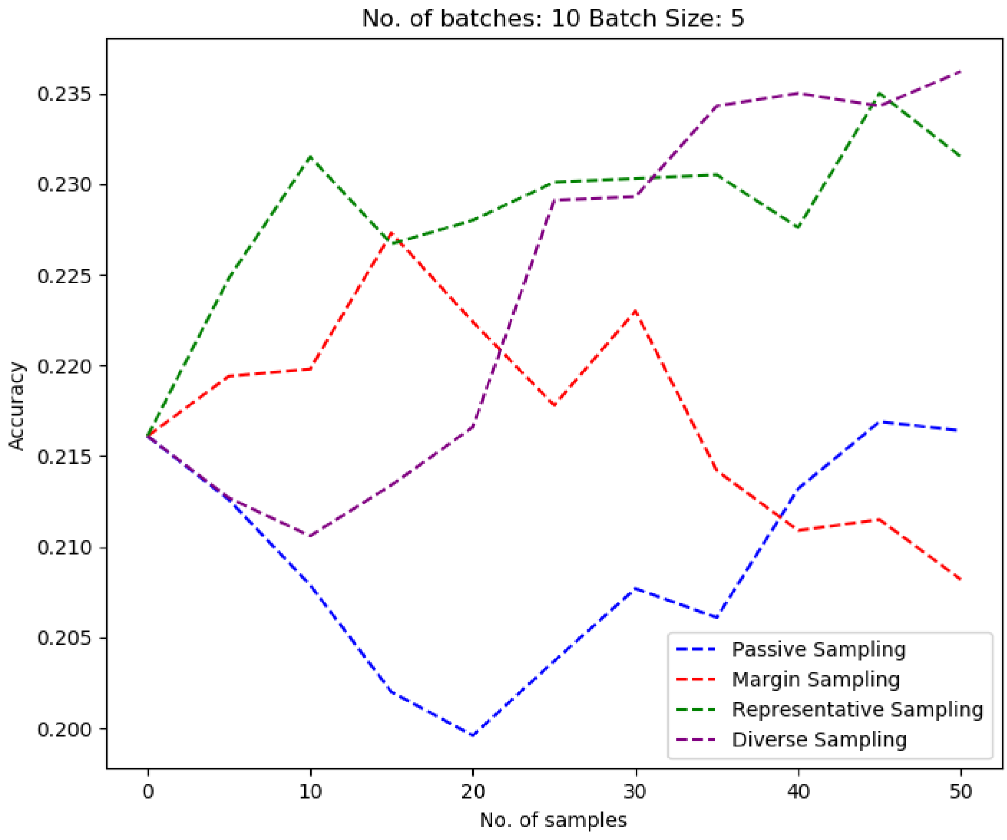

No. of batches: 10 Batch Size: 5
Accuracy
0.235
0.230
0.225
0.220
0.215
0.210
0.205
0.200
0
10
20
30
40
50
No. of samples
Passive Sampling
Margin Sampling
Representative Sampling
Diverse Sampling

Figure 23

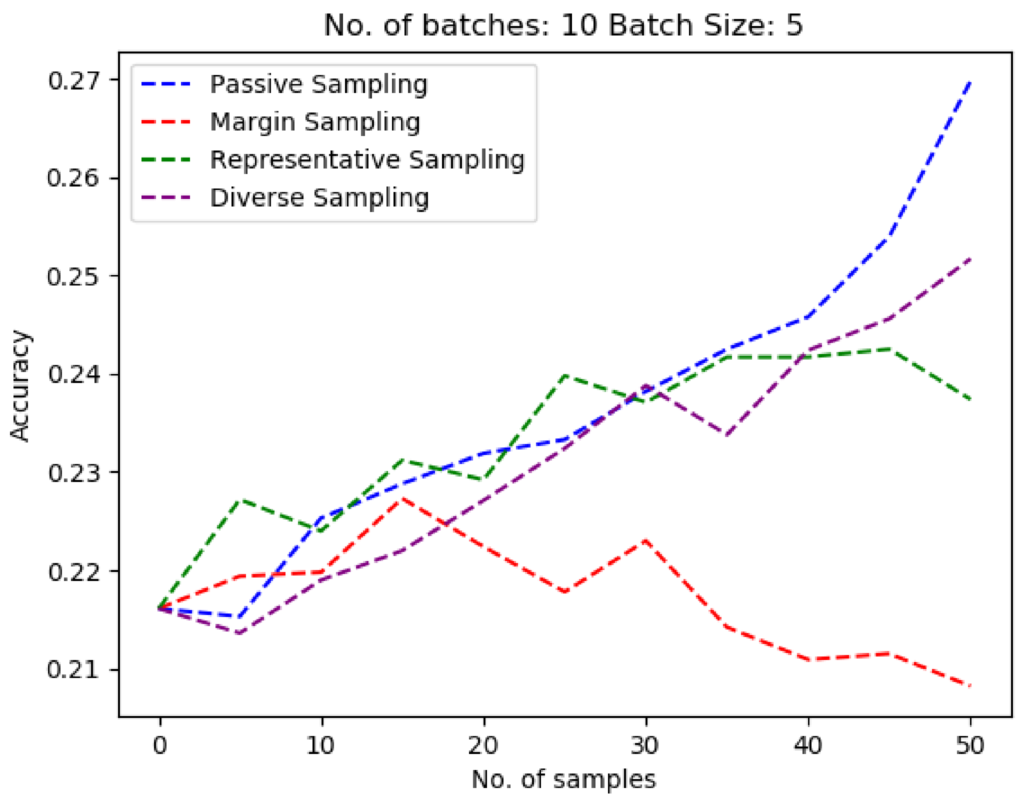


Figure 24

Changing the configuration of our model to the one depicted in Table 3 we observe a similar behaviour to the one depicted in the results produced by the MNIST dataset. Active Learning strategies are generally more accurate than passive sampling. Interestingly, Margin Sampling keeps showing the same unwanted behaviour as before, with sudden drops of accuracy between batches (Figure 25, Figure 26). This negative effect is corrected when increasing the size of the batches, therefore allowing them to be more balanced in terms of the number of instances of each class (Figure 27, Figure 28). Another thing to note is how much better the performance of Representative Sampling is when compared to the one when shown by the experiments on the MNIST dataset. This indicates that the choice of learning strategy is related to the nature and traits of the dataset in question. As it was previously explained, CIFAR-10 has areas that plain Margin Sampling cannot explain adequately. Thus, strategies that incorporate more measures, like diversity and similarity, when choosing the best queries are bound to be more effective when facing datasets with traits similar to CIFAR-10.

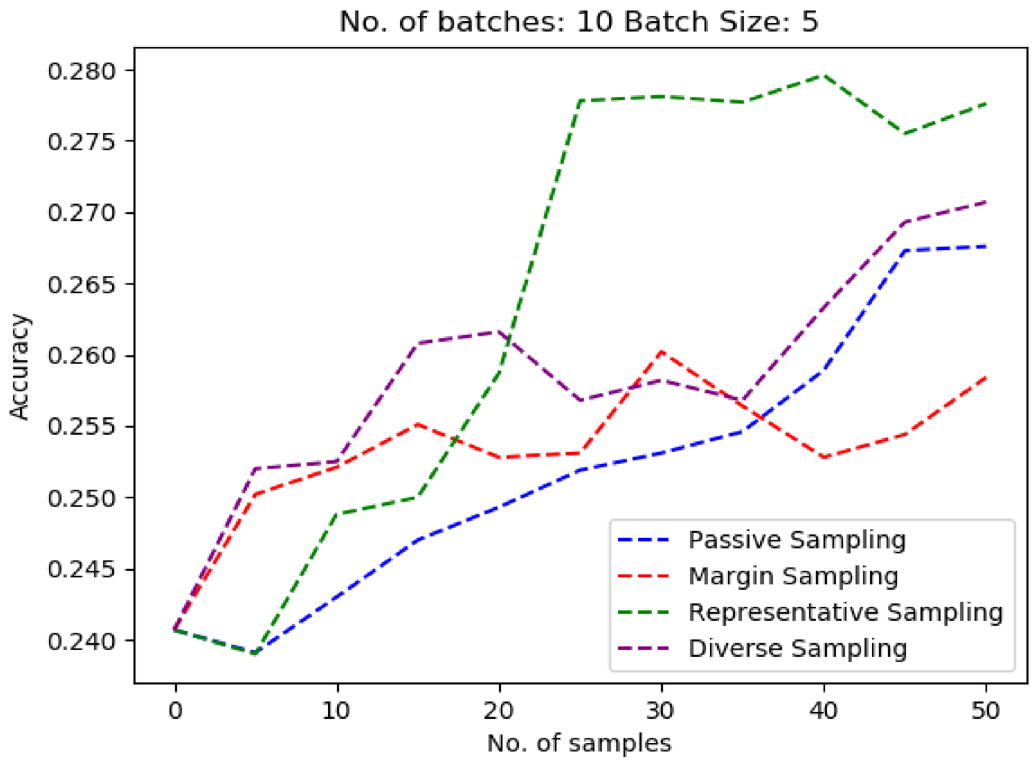


Figure 25

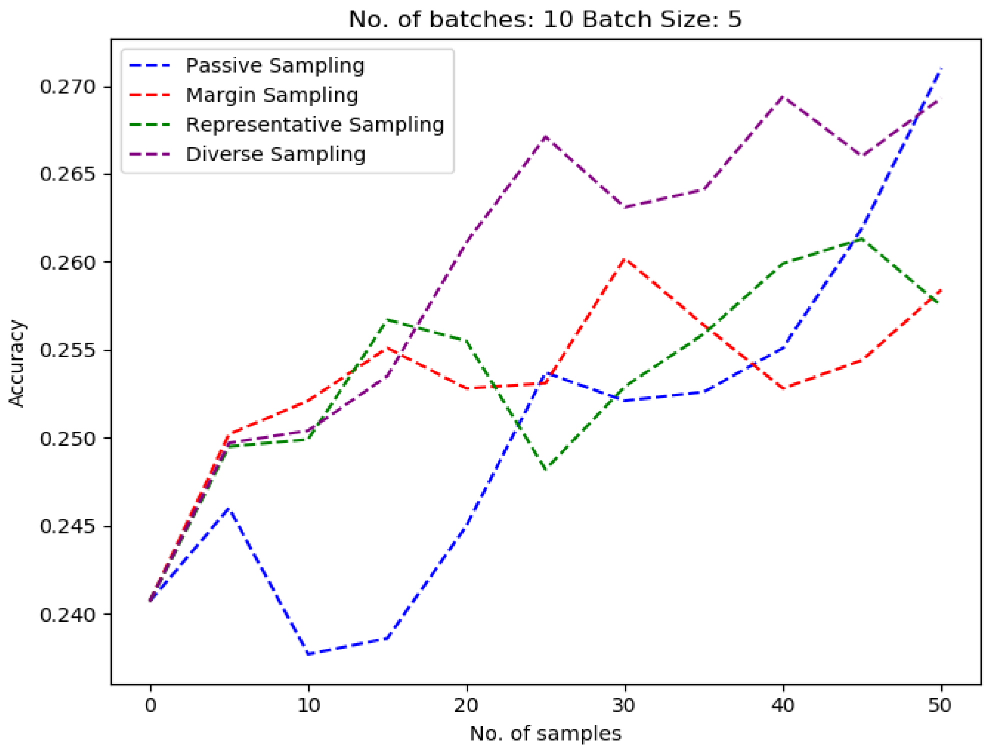


Figure 26

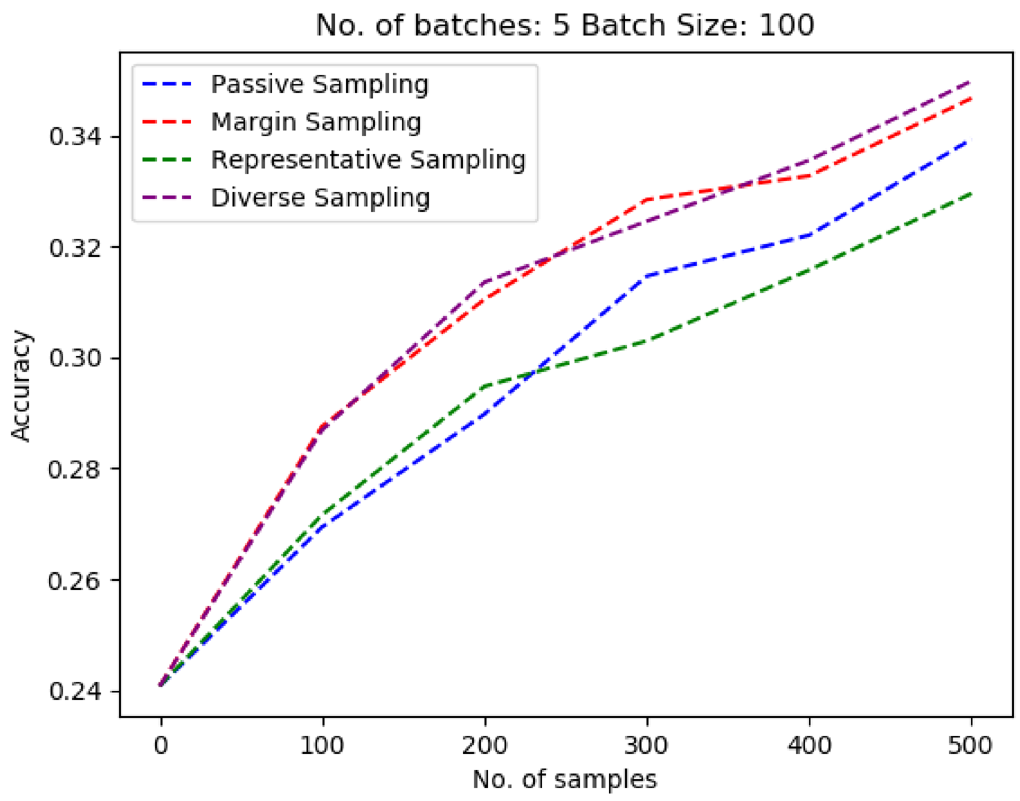


Figure 27

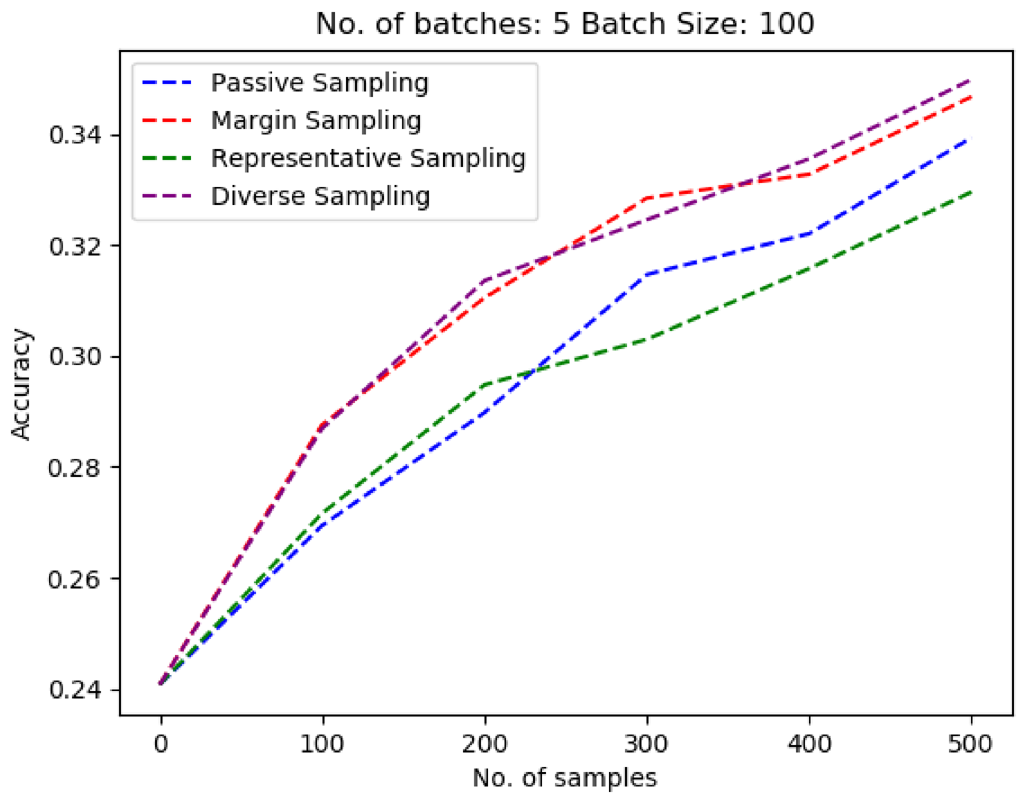


Figure 28

5 CONCLUSION

The concept of Active Learning drew breath from the undeniable truth that hand-labeling large and complex datasets is a tedious task that requires effort and a substantial amount of resources. Luckily, there have been a great number of works suggesting ways of utilizing the human factor in an efficient and beneficial way for the faster and generally more productive labeling of training datasets. Moreover, the newly generated datasets are specifically oriented in increasing the quality of the model that was used to construct the queries. This leads to the following important effects when compared to randomly sampling instances: i) higher accuracy using a limited number of training data ii) faster achievement of high accuracy even when utilized in a passive learning environment. This means that Active Learning can also be used as a means of accelerating the learning process as it eliminates unnecessary training steps. Of course, when the various research methods where put to the test in realistic scenarios many implications rose that created the need for a further investigation into how to tackle them and open the way for the wider use of Active Learning methods.

Throughout this report we tried to present the major Active Learning Frameworks, providing their theoretical foundations and the needs that motivated their research. As it is natural, we have demonstrated only a small number of Active Learning strategies that we consider to be the most representative of their kind. Since the ways that we can define informational gain and leverage it to gain knowledge can only be limited by our creativity, we our bound to come across many more incarnations of Active Learning scenarios in the future. For example, one can consider how the various properties of Support Vector Machines were utilized in different ways when choosing queries. Some strategies use the decision boundary, others the margin while some others the version space.

Using benchmark datasets we proved that the most important selection strategies like Margin Sampling lead to the more efficient training of our supervised model of choice (SVC). Moreover, we witnessed that there is some kind of relation between datasets and strategies that should be evaluated before deciding on which Active Learning method should be used. We often mentioned that the Active Learning Cycle can be implemented in two ways, traditionally by repeating the training process or incrementally by considering only the queried instances. Future plans involve evaluating how this decision can affect the performance of the model in a variety of ways e.g. training time, accuracy etc. For any reader interested in advancing the work shown here we encourage them to replicate the experiments using a model that follows the notion of incremental training and comparing the results to the ones presented in the final chapter. Finally, it would be beneficial to know if given a specific dataset there could be a way of determining which of all learning strategies would provide the most benefit without having to exhaust them by evaluating them one by one.

6 REFERENCES

[1] J. Quinlan, “Induction of Decision Trees,” *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986.

[2] D. W. Hosmer and S. Lemeshow, Applied Logistic Regression, 1989.

[3] C.-W. Hsu, C.-C. Chang and C.-J. Lin, “A Practical Guide to Support Vector Classification,” 2003.

[4] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola and V. Vapnik, “Support Vector Regression Machines,” in *Advances in Neural Information Processing Systems 9*, 1997.

[5] R. Sibson, “SLINK: An optimally efficient algorithm for the single-link cluster method,” *The Computer Journal,* vol. 16, no. 1, pp. 30-34, 1973.

[6] L. Kaufman and P. J. Rousseeuw, Finding Groups in Data, 1990.

[7] C. Chen, C. Chen, E. Durand, F. Forbes and O. Francois, “Bayesian clustering algorithms ascertaining spatial population structure: A new computer program and a comparison study,” *Molecular Ecology Notes,* vol. 7, no. 5, pp. 747-756, 2007.

[8] Y. Tarabalka, J. A. Benediktsson and J. Chanussot, “Spectral–Spatial Classification of Hyperspectral Imagery Based on Partitional Clustering Techniques,” *IEEE Transactions on Geoscience and Remote Sensing,* vol. 47, no. 8, pp. 2973-2987, 2009.

[9] B. E. Boser, I. M. Guyon and V. N. Vapnik, “A training algorithm for optimal margin classifiers". Proceedings of the fifth annual workshop on Computational learning theory”.

[10] C. Cortes and V. N. Vapnik, “Support-vector networks,” 1995.

[11] C. X. Ling and J. Du, “Active learning with direct query construction,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008.

[12] R. D. King, K. E. Whelan, F. M. Jones, P. G. K. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell and S. G. Oliver, “Functional genomic hypothesis generation and experimentation by a robot scientist,” *Nature,* vol. 427, no. 6971, pp. 247-252, 2004.

[13] L. Wang, X. Hu, B. Yuan and J. Lu, “Active learning via query synthesis and nearest neighbour search,” *Neurocomputing,* vol. 147, no. 1, pp. 426-434, 2015.

[14] W. Chu, M. Zinkevich, L. Li, A. Thomas and B. L. Tseng, “Unbiased online active learning in data streams,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[15] A. Beygelzimer, S. Dasgupta and J. Langford, “Importance weighted active learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[16] A. Narr, R. Triebel and D. Cremers, “Stream-based Active Learning for efficient and adaptive classification of 3D objects,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[17] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *Journal of Machine Learning Research,* vol. 2, no. 1, pp. 45-66, 2002.

[18] D. D. Lewis and W. A. Gale, “A sequential algorithm for training text classifiers,” in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994.

[19] C. Zhang and T. Chen, “An active learning framework for content-based information retrieval,” *IEEE Transactions on Multimedia,* vol. 4, no. 2, pp. 260-268, 2002.

[20] S. Tong and E. Y. Chang, “Support vector machine active learning for image retrieval,” in *Proceedings of the ninth ACM international conference on Multimedia*, 2001.

[21] Y. Liu, “Active Learning with Support Vector Machine Applied to Gene Expression Data for Cancer Classification,” *Journal of Chemical Information and Computer Sciences,* vol. 44, no. 6, pp. 1936-1941, 2004.

[22] T. Scheffer, C. Decomain and S. Wrobel, “Mining the Web with active hidden Markov models,” in *Proceedings 2001 IEEE International Conference on Data Mining*, 2001.

[23] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal,* vol. 27, no. 3, pp. 379-423, 1948.

[24] T. M. Mitchell, “Generalization as search,” *Artificial Intelligence,* vol. 18, no. 2, pp. 203-226, 1982.

[25] L. K. Hansen and P. Salamon, “Neural network ensembles,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 12, no. 10, pp. 993-1001, 1990.

[26] L. Breiman, “Bagging predictors,” *Machine Learning archive,* 1996.

[27] Y. Freund, “Schapire: Experiments with a new boosting algorithm,” , 1996.

[28] Y. Freund, “Schapire RE: A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences,* 1997.

[29] K. Tumer and J. Ghosh, “Error Correlation and Error Reduction in Ensemble Classifiers,” *Connection Science,* vol. 8, pp. 385-404, 1996.

[30] J. F. Kolen and J. B. Pollack, “Back Propagation is Sensitive to Initial Conditions,” in *Advances in Neural Information Processing Systems 3*, 1991.

[31] T. G. Dietterich, “Ensemble Methods in Machine Learning,” *multiple classifier systems,* pp. 1-15, 2000.

[32] I. Dagan and S. P. Engelson, “Committee-Based Sampling For Training Probabilistic Classifiers,” in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[33] A. Culotta and A. McCallum, “Reducing labeling effort for structured prediction tasks,” in *AAAI'05 Proceedings of the 20th national conference on Artificial intelligence - Volume 2,* 2005.

[34] S. Kim, Y. Song, K. Kim, J.-W. Cha and G. G. Lee, “MMR-based Active Machine Learning for Bio Named Entity Recognition,” in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, 2006.

[35] D. A. Cohn, Z. Ghahramani and M. I. Jordan, “Active learning with statistical models,” *Journal of Artificial Intelligence Research,* vol. 4, no. 1, pp. 129-145, 1996.

[36] B. Settles and M. Craven, “An Analysis of Active Learning Strategies for Sequence Labeling Tasks,” in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* 2008.

[37] J. Huang, R. Child, V. Rao, H. Liu, S. Satheesh and A. Coates, “Active Learning for Speech Recognition: the Power of Gradients.,” *arXiv preprint arXiv:1612.03226,* 2016.

[38] N. Roy and A. McCallum, “Toward Optimal Active Learning through Sampling Estimation of Error Reduction,” in *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning,* 2001.

[39] Y. Guo and R. Greiner, “Optimistic active learning using mutual information,” in *IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence*, 2007.

[40] R. Moskovitch, N. Nissim, D. Stopel, C. Feher, R. Englert and Y. Elovici, “Improving the Detection of Unknown Computer Worms Activity Using Active Learning,” *KI '07 Proceedings of the 30th annual German conference on Advances in Artificial Intelligence,* pp. 489-493, 2007.

[41] S. Geman, E. Bienenstock and R. Doursat, “Neural networks and the bias/variance dilemma,” *Neural Computation,* vol. 4, no. 1, pp. 1-58, 1992.

[42] D. A. Cohn, L. E. Atlas and R. E. Ladner, “Improving Generalization with Active Learning,” *Machine Learning,* vol. 15, no. 2, pp. 201-221, 1994.

[43] D. J. C. MacKay, “Information-based objective functions for active data selection,” *Neural Computation,* vol. 4, no. 4, pp. 590-604, 1992.

[44] A. I. Schein and L. H. Ungar, “Active learning for logistic regression,” , 2005.

[45] T. Zhang and F. J. Oles, “A probability analysis on the value of unlabeled data for classification problems,” in *ICML,* 2000.

[46] A. Mccallum and K. Nigam, “Employing EM and Pool-Based Active Learning for Text Classification,” *Proceedings of ICML-98, 15th International Conference on Machine Learning,* pp. 350-358, 1998.

[47] A. Fujii, T. Tokunaga, K. Inui and H. Tanaka, “Selective sampling for example-based word sense disambiguation,” *Computational Linguistics,* vol. 24, no. 4, pp. 573-597, 1998.

[48] Z. Xu, R. Akella and Y. Zhang, “Incorporating diversity and density in active learning for relevance feedback,” in *ECIR'07 Proceedings of the 29th European conference on IR research*, 2007.

[49] J. Lin, “Divergence measures based on the Shannon entropy,” *IEEE Transactions on Information Theory,* vol. 37, no. 1, pp. 145-151, 1991.

[50] H. T. Nguyen and A. W. M. Smeulders, “Active learning using pre-clustering,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004.

[51] Z. Xu, K. Yu, V. Tresp, X. Xu and J. Wang, “Representative sampling for text classification using support vector machines,” *european conference on information retrieval,* pp. 393-407, 2003.

[52] J. Baldridge and M. Osborne, “Active Learning and the Total Cost of Annotation.,” in *EMNLP,* 2004.

[53] K. Tomanek and U. Hahn, “Semi-Supervised Active Learning for Sequence Labeling,” in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

[54] R. Hwa, “On minimizing training corpus for parser acquisition,” in *ConLL '01 Proceedings of the 2001 workshop on Computational Natural Language Learning - Volume 7*, 2001.

[55] D. D. Lewis and J. Catlett, “Heterogenous uncertainty sampling for supervised learning,” in *ICML'94 Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, 1994.

[56] J. Howe, Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business, 2008.

[57] R. Snow, B. O'Connor, D. Jurafsky and A. Y. Ng, “Cheap and Fast -- But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks,” in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

[58] . Smyth, . Padhraic, . Fayyad, . Usama, . Burl, . Michael, . Perona, . Pietro, . Baldi and . Pierre, “Inferring Ground Truth from Subjective Labelling of Venus Images,” in *Proceedings of the 7th International Conference on Neural Information Processing Systems*, 1994.

[59] P. Donmez and J. G. Carbonell, “Proactive learning: cost-sensitive active learning with multiple imperfect oracles,” in *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008.

[60] V. Sheng, F. Provost and P. Ipeirotis, “Get Another Label? Improving Data Quality and Data Mining,” , 2008.

[61] Y. Yan, G. M. Fung, R. m. Rosales and J. G. Dy, “Active Learning from Crowds,” in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[62] K. Brinker, “Incorporating diversity in active learning with support vector machines,” in *ICML'03 Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, 2003.

[63] Y. Guo and D. Schuurmans, “Discriminative Batch Mode Active Learning,” in *Advances in Neural Information Processing Systems 20*, 2008.

[64] Z. Wang, B. Du, L. Zhang and L. Zhang, “A batch-mode active learning framework by querying discriminative and representative samples for hyperspectral image classification,” *Neurocomputing,* vol. 179, pp. 88-100, 2016.

[65] B. Settles, M. Craven and S. Ray, “Multiple-Instance Active Learning,” in *Advances in Neural Information Processing Systems 20*, 2008.

[66] D. Liu, X.-S. Hua, L. Yang and H.-J. Zhang, “Multiple-Instance Active Learning for Image Categorization,” in *MMM '09 Proceedings of the 15th International Multimedia Modeling Conference on Advances in Multimedia Modeling*, 2009.

[67] S. Vijayanarasimhan and K. Grauman, “Multi-Level Active Prediction of Useful Image Annotations for Recognition,” in *Advances in Neural Information Processing Systems 21*, 2009.

[68] H. Raghavan, O. Madani and R. Jones, “Active Learning with Feedback on Features and Instances,” *Journal of Machine Learning Research,* vol. 7, pp. 1655-1686, 2006.

[69] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[70] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 3, p. 27, 2011.

[71] S. Shalev-Shwartz, Y. Singer, N. Srebro and A. Cotter, “Pegasos: primal estimated sub-gradient solver for SVM,” *Mathematical Programming,* vol. 127, no. 1, pp. 3-30, 2011.

[72] D. Ciregan, U. Meier and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition,* 2012.