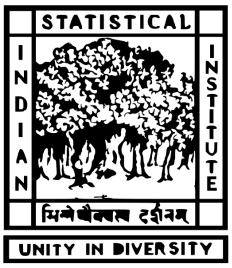


WINDMILL POWER MANAGEMENT USING ONLINE AND ACTIVE LEARNING APPROACH

By

Sankarsan Seal

Dissertation

Submitted in partial fulfilment of the requirements for

M. Tech (Computer Science) degree

of the

Indian Statistical Institute

2017

Under supervision of

Dr. Swagatam Das

Electronics and Communication Sciences Unit

Indian Statistical Institute, Kolkata

Certificate of Completion

This is certified that the dissertation on WINDMILL POWER MANAGEMENT USING ONLINE AND ACTIVE LEARNING APPROACH by Mr Sankarsan Seal(Roll No CS1511), in Partial fulfilment of the requirements for the award of the Degree of Master of Technology in Computer Science from Indian Statistical Institute, Kolkata is a report prepared by him under my guidance. It is certified that the work has not been submitted anywhere else for the award of any other diploma or degree of this or any other University.

Date:

Dr. Swagatam Das
Supervisor
Electronics and Communication Sciences Unit
Indian Statistical Institute, Kolkata

Contents

List of Tables

List of Figures

Acknowledgements

I like to acknowledge Indian Statistical Institute for selecting me as eligible candidate of M.Tech Computer Science curriculum and giving me opportunity to pursue my post graduate under great environment of research and ambience of excellence.

But as part of M.Tech course, I am fortunate to come in touch with great scholar but humble supervisor Dr. Swagatam Das. His relentless perseverance makes me to comprehend the importance of research. I am grateful to him to introduce me to vast arena of upcoming exploration on machine learning. It will help me to keep going research in future whatever be my field of research.

With the direction of Mr. Shaunak Datta, I could veer my research to right direction. He suggested me important topics to tackle with different problems I faced during study of immense repository of knowledge.

I am thankful to my family for keeping faith in me and encouraging me to face this challenge with positive mind frame.

- Sankarsan Seal

Abbreviations

ANN:	Artificial Neural Network
BP:	Back Propagation
COV:	Covariance
FNN:	Feed forward Neural Network
MLP:	Multi Layer Perceptron
NWP:	Numerical Weather Prediction
NORMA:	Naive Online Risk Minimization Algorithm
PCA:	Principal Component Analysis
RBF:	Radial Basis Function
RVFL:	Random Vector Functional Link
SGD:	Stochastic Gradient Descent
SVC:	Support Vector Machine for Classification
SVD:	Singular Value Decomposition
SVM:	Support Vector Machine

Chapter 1

Introduction

1.1 Abstract

Present day we are looking for alternative energy source instead of fossil fuel. Renewable energies which are available in our reach now days are solar, hydro and wind. We are considering wind power generation scenario in this paper. Wind energy is converted to electrical energy with help of windmill at high wind blowing regions. But we need to differentiate between the wind speed which is normal flow and which is storm like violent.

Whole process can be divided in two parts. First, collecting labelled training set from huge collection of unlabelled dataset. Second, training with those data.

Problem we face is that whole year we receive most usual and normal wind speed reading but storm like wind speed reading is very rare in kind. So it is imbalanced data problem. We know using imbalanced dataset we can get high accuracy but very poor precision and recall of minority class. Our problem is that we should shut our windmill power production when certain wind speed is crossed to save from over power generation and protecting

windmill from high speed rotation. Sudden or abrupt power generation cut off can create problem in power distribution grid. Using weather forecast like NWP(Numerical Weather Prediction) [16] we can predict the probable storm or typhoon like high speed wind flow over larger area but we can not say exact hour of impact. If we get the prediction for individual windmill farm using some reading and shut down the windmill production before some time like 1 hour, 6 hours or 12 hours then we can take decision how to redistribute the power requirement[21].

Different statistical approaches[16], [18], like time series analysis and ANN implementation[17],[19],[20] of wind speed prediction for usual operational speed limit are done previously with historical data. But new problem arises when we are doing rare high wind speed prediction.

1.2 Motivation

If we are equipped to receive data in batch and data are unlabelled and imbalanced in nature we can use Support Vector like classifier. But using all data points or instances are not necessary all the time. We can reduce the dataset size using active learning like approach. On the other hand, if it is imbalanced data and data are not coming in bulk or batch, we look for predicting event with online learning option. We know Multilayer perceptron, RBF neural net for online leaning. Those have some inherent problems of training and not good performer of imbalanced dataset. In this report, SVM using LIBSVM, Random Vector Functional Link, NORMA, Stochastic Gradient Descent Primal L1-SVM are explored for this issue.

1.3 Thesis Outline

The remaining chapters are organized in following sequence,

In Chapter 2, we consider different approaches to deal with imbalanced dataset. How different sampling methods are used and their drawbacks.

In Chapter 3, we discussed batch training of SVM and RVFL for labelling problem using active learning. Online prediction methods of Stochastic Gradient Descent of Primal L1-SVM(SGD-SVM) and NORMA are illustrated. Generic K-NN classifier is also discussed.

In Chapter 4, we proposed a way of using SVM (LIBSVM tool) and RVFL with active learning when batch dataset is present with us. Then we explained the use of SGD-SVM concept for online learning. Well known algorithm for online SVM ,Naive Online Risk minimization Algorithm (NORMA)[5], is also explained.

In Chapter 5, we mentioned the dataset description and how those attributes are selected. All experimental results and their causes are discussed. The comparison between SVM(LIBSVM) with quadratic programming,Online SVM(NORMA) and Primal L1-SVM (SGD-SVM) are there.

In Chapter 6, we mention the summary of observed results and difficulties found in applied algorithms. Future research direction to reduce the misclassification is discussed.

Chapter 2

Imbalanced Data

2.1 Problem with Imbalanced Dataset

In everyday life we encounter with different instances of observations. But some observations are prevalent in collection than others. Most of the time prevalent and more frequent data instances are not so important and do not bear any significant information. On contrary, most useful and relevant observation for classification are rare. For example, detection of cancerous cell among normal healthy cell is an imbalanced problem. Similarly, detection of fraudulent transaction using credit card, identifying faulty item in production or object detection in digital images.

Equivalently, wind speed for storm and typhoon are very rare in observation than normal wind speed. We usually divide the dataset into two parts when we find imbalanced data problem. The instances, which we like to identify is called positive instances, are very rare in dataset. Other instances are named as negative instances. It is called Class Imbalance Problem. This imbalanced instances actually represents very low prior probability in dataset. Using this prior probability, we may get higher accuracy after classification

but we miss minority class to identify, which eventually leads to failure of whole class labelling and classification process.

2.2 Under Sampling

To reduce the imbalance ratio between positive class and negative class, we can device different methods of under sampling which remove the majority class instances[24]. But it has severe problem too, as we are removing the instances from dataset, we may eliminate most important and relevant observations from dataset for majority class.

2.3 Over Sampling

In the case of minority class, we can oversample instances. But those instances are mostly replica of existing instances so it may increase number of observations but it is not helpful to provide new relevant instances which can help the classification process. When oversampling with replacement is used this kind of situation arises.

2.4 Higher Penalty for Minority Class Misclassification

Many classifiers facilitate the option to fix regularization parameter to trade off between optimal classifier and classification error[23]. When we are dealing with imbalance dataset, we can assign different weight to penalty for different classes. If we assign high cost for minority class then classifier may shift to minority class region to predict more accurately.

2.5 SMOTE

SMOTE(Synthetic Minority Oversampling TEchique) [3] is one of the many over sampling methods for minority class. Instead of replicating instances from existing data distribution this method generates "synthetic" minority instances. It generates new instances in feature space rather than data space. It uses the k-Nearest Neighbour concept for creating new observations. It uses points in the line joining k-nearest neighbours of same class. If we need many instances in vicinity then randomly many instances are chosen and use points on line segment.

2.6 Active Learning

We know that collecting useful data for classification is very crucial task[12], but labelling each observed data to proper class is also very cumbersome[28]. When we are dealing with billions of observations or instances, providing labels is not possible. In the case of supervised learning process we definitely provide labels to all training instances. And in the case of semi-supervised learning all data points are not labelled. Small set of whole data points is labelled by experts or user before training. After training with those data points, new unlabelled data points are introduced to already labelled dataset. Using k nearest neighbour approach we can labelled those unlabelled data and retrain the classifier. In this method we may label those points too which do not bear significant information for better classifier.

To overcome this drawback, we use active learning. Active learning is not actually any learning of classifier. It learns about labels of unlabelled data points[33, 34],without creating new artificial instance. The process of labelling data point or instance for active learning is called *Query* which is

answered by user or expert or oracle.

2.6.1 Query strategies

2.6.1.1 Uncertainty Sampling

With a classifier, a instance ,which is selected randomly without replacement, can not be classified with very high probability or confidence, is candidate for active learning instance[32]. The search for randomly selected instance which is not certainly classified is time consuming. We need to look for active learning sample with this strategy in whole dataset.

2.6.1.2 Query by Committee

In this strategy [11, 31] we have multiple classifiers, like multilayer perceptron, k -NN, RBF neural network. A selected instance is classified by these classifiers. We need to label only those instances which create maximum disparity in classification. Suppose we have six(6) such classifiers, if 3 out of 6 classification agree to a particular class but rest 3 of 6 are not then we have maximum conflict. If 5 out of 6 or 6 out of 6 are labelled to same class then we do not need to do labelling. The instance which has maximum entropy, we need to classify that instance.

2.6.1.3 Random Pool

Instead of looking for whole dataset for next active learning instance[13, 14, 30], we can select pool size of L instances, then we find the instance which is nearest to classifier or can not be classify with high confidence. Let M is the dataset size. let $1 - \eta$ be 95% of confidence such that selected element is nearest to classifier. Let p is the probability that top 5% of instances nearest

to classifier. So we can say,

$$1 - (1-p)^L = 1 - \eta$$

$$\implies (1-p)^L = \eta$$

taking log to both sides

$$\implies L = \frac{\log \eta}{\log (1-p)}$$

if $\eta = 0.05$ and $p = 0.05$ we can have

$$L \approx 59$$

It shows that if we select 59 or more instances randomly then we have high probability of getting an active learning instance. It does not depend on M, size of dataset, too. So instead of searching whole dataset we can look for small pool of unlabelled data points. It is called 59 trick[22].

2.6.1.4 Exploiting Active Learning with SVM

In subsequent chapter we will discuss the use of available SVM tools(like LIBSVM),[26, 25] which provide details of support vectors and indices of those vectors in whole dataset. While testing or predicting the class of a instance we can get the distance of the instance from classifier hyperplane. Data points or instances which are within the margin of +1 to -1 distance or any predefined range, those are most likely to be active learning instances [29] and can be used in retraining the classifier for betterment[28]. So instead of all dataset, we can retrain with support vectors in previous iteration and newly added active learning instances. So the number of training data points will be manageable.

Chapter 3

Learning Methods

3.1 Online Classifier

In the domain of online classifier, all data are not present before the training process starts. It gradually learns from instances appearing one after another and observes the error and uses the penalty to minimise future classification error. Although we have multiple batch classifiers with very good performance but those classifiers take lot of computational resources like memory and CPU time and another thing, it may produce classifier which is overfitted to given dataset. Online learning process has many advantages over batch process. It requires less powerful computational unit and lesser memory. Only issue with this process is that it uses multiple epochs to reduce the overall error of classification. That means same instance is to be presented to online classifier for training. In some online classifier like multilayer perceptron we use the concept of back propagation to update synaptic weights or in other words, learning from instance representation and error observation.

3.2 Random Vector Functional Link Network (RVFL)

Random Vector Functional Link Network[6] is one of the variants of ANN. It is single hidden layer neural network but little modification from conventional multilayer perceptron architecture. As we know initial machine learning architecture for conventional ANN[6] is shown in this figure 3.1.

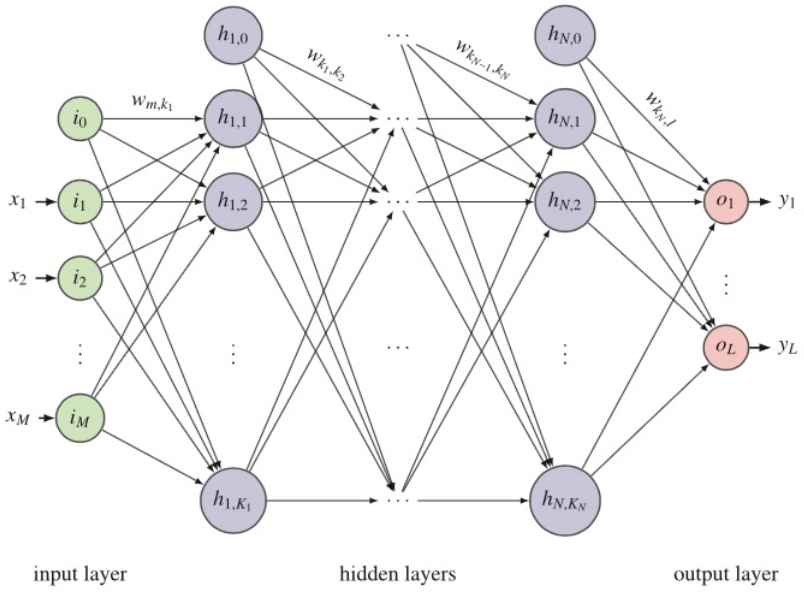


Figure 3.1: Artificial Neural Network with multiple hidden layers

We initially conceptualize the idea of Feed forward Neural Network(FNN). But present time we need to consider the previous output or result for successive prediction. This kind of neural networks are very useful for time series kind data. When a neural network architecture contains link from hidden layer to input layer or output layer to hidden layer, this kind of neural network variant is called Recurrent Neural Network(RNN). RVFL is a single

hidden layer feedforward neural network. The links of RVFL is shown [6] in the figure 3.2.

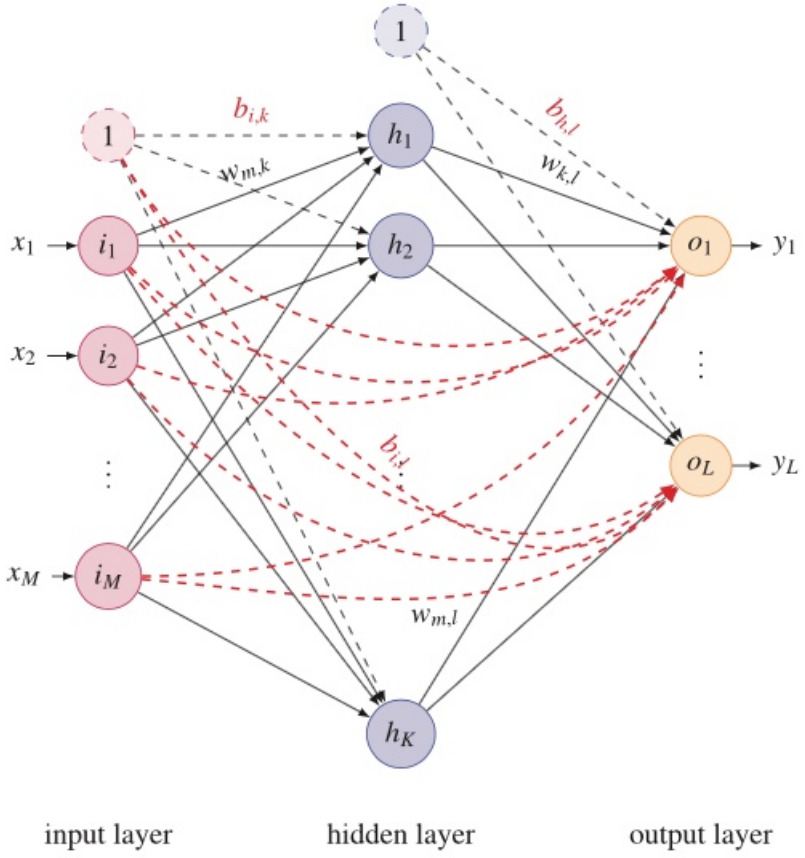


Figure 3.2: Random Vector Functional Link Neural Network Architecture

3.2.1 Quick Review of Neural Network

3.2.1.1 Activation Function

For neural network[27], we need an activation function which is differential with respect to weight of link between previous layer to present layer or

two successive layers. Usually we use **sigmoid function** and **hyperbolic tangent,** tanh **function**

$$logsig(x) = \frac{1}{1 + \exp^{(-x)}}$$

$$tanh(x) = \frac{\exp^{(x)} - \exp^{(-x)}}{\exp^{(x)} + \exp^{(-x)}}$$

$logsig(x) \in (0, 1)$ and $tanh(x) \in (-1, +1)$

Let $f(\cdot)$ be either $logsig(x)$ or $tanh(x)$ function and induced local field function be u.

For value of first hidden layer k^{th} neuron $h_{1,k}$ be

$$h_{1,k} = f(u)$$

where

$$u = \left(\sum_{i=0}^{M} W_{i,k} x_i \right), i = 0, \ldots, M$$

M is number of input node and $W_{i,k}$ is weight value between i^{th} neuron node of input layer to k^{th} neuron of consequent hidden layer . $W_{0,k} = b_k$ is bias if present.

Similarly, for output layer neurons

$$o_l = f(u)$$

where

$$u = \left(\sum_{i=0}^{K} W_{i,l} h_{k,i} \right)$$

3.2.1.2 Weight of Links

For training purpose, initially we do not know the weight of each link. When we start training of neural network, weight of each link is assigned with

random values. Usually it is assigned with small values. If we assign initially huge value to weight to links then it may not train as expected.

3.2.1.3 Training by Back Propagation

For Batch mode training, each instance of whole batch is presented to the neural network. The error or cost function is calculated as

$$J(W) = \frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{L}\left(o_{i,j} - desired\_value_{i,j}\right)^2$$

Where m is number training instances, L is number of output neurons.

For online leaning process

$$J(W) = \frac{1}{2}\sum_{j=1}^{L}\left(o_{i,j} - desired\_value_{i,j}\right)^2$$

for a particular instance.

We need to minimize $J(W)$ with respect to W taking partial derivative for individual $W_{k,j}$.

$$\Delta W_{k,j} = -\eta\frac{\partial J(W)}{\partial W_{k,j}}$$

New weight after update after t iterations

$$W_{k,j}^{(t+1)} = W_{k,j}^{(t)} + \Delta W_{k,j}$$

In this way, Back Propagation (BP) is utilized to update weight of links but this method requires an iterative training process which is computationally demanding. Further, BP is also highly likely to be trapped in a local minimum.

3.2.1.4 Random Vector and Functional Link

We know Multi Layer Perceptron(MLP) consists of hidden layers and it adds complexity to calculate optimal weights for classification. Along with that, hidden layer configuration is also heuristic process. Lesser number of hidden layer neurons may under fit the training dataset and excessive number of neurons of hidden layer may overfit. Although we can estimate the number of hidden layers and number of nodes on those hidden layers, initial weight value assignment and update are challenging task.

Random Vector concept suggests that instead of updating the weights between input layer to hidden layer and within hidden layers, we can assign weights of those hidden layer randomly with in uniform distribution [-1, +1]. It is very helpful while we are only considering BP between last hidden layer and output layer. But it has possibility of overfitting of weights between hidden layers. Although this method gives significant performance gain over MLP.

Functional Link utilizes higher combination of its inputs. It reduces number the hidden layers or removes them to make neural network training process faster although keeping non-linear separability feature. The Functional Link neural network architecture is basically a flat network without any hidden layer which has made the learning algorithm used in the network less complicated. The input vector is extended with a suitably enhanced representation of the input nodes, thereby artificially increasing the dimension of the input space. Those added enhanced nodes to input layer are called enhancement nodes.

As shown in the figure 3.2 links between input layer to output layer represent functional links which are trained with Back Propagation(BP).

3.3 Support Vector Machine

Support Vector Machine [1, 2] emphases the concept of classifier providing maximum margins between two classes. It is usually very useful with linearly separable data and it can be extended to non-separable data too. In the case of non-separable dataset it utilized higher dimensional mapping and *Kernel Trick* for classification with same mathematical foundation. Also, It requires regularization parameter as input.

3.3.1 Margin in SVM

Main aim of maximum margin is that it can provide classification with highest confidence. Many classifiers like single layer perceptron provide one of the many hyperplanes but that does not guarantee the best classifier with maximum margin between two classes on both side of hyperplane.

Let define w be weight vector and b is bias for classifier. we have a new instance to classify, say x, then $w^T x + b \gg 0$ or $w^T x + b \ll 0$ gives us the very high confidence while classification, where $w^T x + b = 0$ is equation of classifier. It is also known as separating hyperplane.

If we consider the points A, B and C in figure 3.3, using the above mentioned equation we can say with high confidence or probability, A is most certain to classify. About B, classification is moderately certain. But for C, it is very low probable that classification is sure or not.

3.3.2 Binary Classification with SVM

Let D is our feature space and $x \in D^n$ where m is number of instances and n is number of features. For an instance, $x = (x_1, x_2, \ldots, x_n)^T$ if we try to classify we need weight vector $w \in \mathbb{R}^n$ and bias $b \in \mathbb{R}$. Class label for each

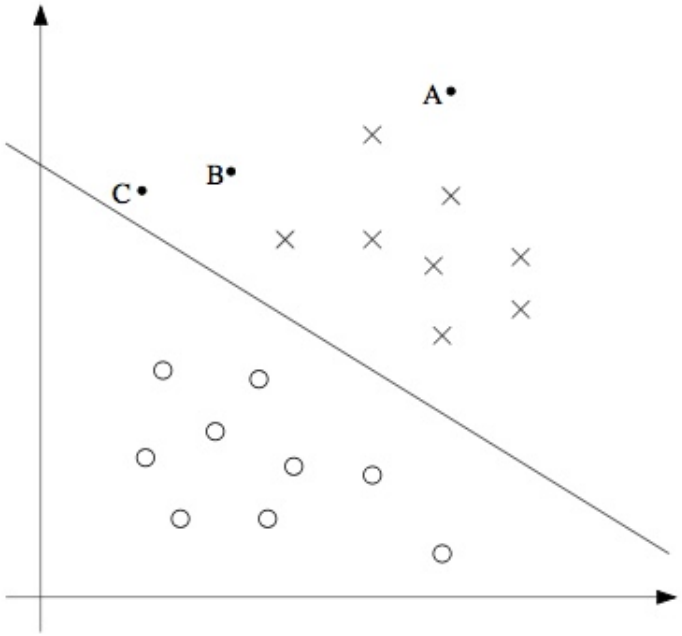


Figure 3.3: Linearly separable dataset by hyperplane

instance is identified by $y \in \{-1, +1\}$

Let define

$$g(z) = +1, \text{when}, z \geq 0$$

else

$$g(z) = -1, \text{when}, z < 0$$

Then hypothesis,

$$h_{w,b}(x) = g(w^T x + b)$$

3.3.3 Functional and Geometric Margin

Using previously mentioned equation,

$$h_{w,b}(x) = g(w^T x + b)$$

, where $h_{w,b}$ yields value $\in \{-1, +1\}$. But it is also true for

$$h_{w,b}(x) = g(2w^T x + 2b)$$

.

So it is not taking the actual distance from hyperplane for confidence level. We can not differentiate between two cases with functional margin.

Let define **functional margin** for w and b,

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

Let define

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}^{(i)}$$

We know that if we have an equation

$$w^T x + b = 0$$

of a straight line, then perpendicular distance from a given point $\hat{x}$ be

$$\frac{\left|w^T \hat{x} + b\right|}{\|w\|_2}$$

Using functional margin concept we can reduce that to

$$\frac{\hat{\gamma}^{(i)}}{\|w\|_2} = \frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|_2} = \frac{\left|w^T x^{(i)} + b\right|}{\|w\|_2}$$

$$\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|w\|_2} = y^{(i)} \left(\left(\frac{w}{\|w\|_2} \right)^T x^{(i)} + \frac{b}{\|w\|_2} \right)$$

and

$$\gamma = \min_{1\ldots m} \gamma^{(i)}$$

Where γ is called **geometric margin** .

Our objective is to maximize the geometric margin, so we can take functional margin -1 or $+1$.

We can write that

$$\max_{\hat{\gamma},w,b} \frac{\hat{\gamma}}{\|w\|}$$

such that,

$$y^{(i)}\left(w^T x + b\right) \geq \hat{\gamma}, i = 1, 2, \ldots, m$$

when $\hat{\gamma} = 1$, equation reduced to

$$\max_{\hat{\gamma},w,b} \frac{1}{\|w\|}$$

such that,

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, 2, \ldots, m$$

It can be converted to minimization problem as

$$\min_{\hat{\gamma},w,b} \frac{1}{2}\|w\|^2$$

such that,

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, 2, \ldots, m$$

above optimization problem is convex, using quadratic optimizer we can solve this to use as optimal margin classifier.

3.3.4 Lagrangian Duality

Suppose we have following objective function and constraints,

$$\min_{w} f(w)$$

such that,

$$h_i(w) \leq 0$$

for $i = 1, \ldots, l$ and

$$g_i(w) = 0$$

for $i = 1, \ldots, k$

So we can define *general Lagrangian Construct*

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{l} \alpha_i h_i(w) + \sum_{i=1}^{k} \beta_i g_i(w)$$

where α_i and β_i are *Lagrangian Multipliers.*

Then we can define

$$\Gamma_{\mathcal{P}}(w) = \max_{\alpha,\beta:\alpha,\beta\geq 0} \mathcal{L}(w, \alpha, \beta)$$

, where $\mathcal{P}$ stands for primal.

If all constraints are satisfied then

$$\Gamma_{\mathcal{P}}(w) = \max_{\alpha,\beta:\alpha,\beta\geq 0} f(w) + \sum_{i=1}^{l} \alpha_i h_i(w) + \sum_{i=1}^{k} \beta_i g_i(w)$$

otherwise

$$\Gamma_{\mathcal{P}}(w) = \infty$$

Optimization problem yields

$$\Gamma_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if w satisfies all constraints} \\ \infty & \text{Otherwise} \end{cases}$$

If we define dual of this above mentioned problem,

$$\Gamma_{\mathcal{D}}(\alpha, \beta) = \min_{w} \mathcal{L}(w, \alpha, \beta)$$

taking max of both sides,

$$\implies \max_{\alpha,\beta:\alpha\geq 0} \Gamma_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha,\beta:\alpha\geq 0} \min_{w} \mathcal{L}(w, \alpha, \beta)$$

It can be shown that $d^* = \max_{\alpha,\beta:\alpha\geq 0} \Gamma_{\mathcal{D}}(\alpha,\beta)$ and $p^* = \min_{w} \Gamma_{\mathcal{P}}(\alpha,\beta)$

$$d^* = p^*$$

If we assume that w^* is solution of primal optimization problem and α^* and β^* are solutions of dual optimization problem and $d^* = p^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$, where *KKT condition* (Karush-Kuhn-Tucker condition) can be applied, then we have

$$\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, i = 1, \ldots, n \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, i = 1, \ldots, k \\
\alpha_i^* h_i(w^*) &= 0, i = 1, \ldots, l \\
h_i(w^*) &\leq 0 \\
\alpha_i^* &\geq 0
\end{aligned}$$

3.3.5 Optimal Margin Classifier

As discussed in previous subsection 3.3.3,

$$\min_{\hat{\gamma},w,b} \frac{1}{2}\|w\|^2$$

such that,

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, 2, \ldots, m$$

It can be converted to

$$g_i(w) = -y^{(i)}\left(w^T x^{(i)} + b\right) + 1 \leq 0$$

From above equations we can make Lagrangian construction as

$$\mathcal{L}\left(w, b, \alpha\right) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y^{(i)} \left(w^T x^{(i)} + b \right) - 1 \right)$$

where α_i is *Lagrangian Multiplier*. We discussed in section 3.3.4 the *Lagrangian Multiplier*.

If we define

$$\Gamma_{\mathcal{P}}(w) = \max_{\alpha,\beta:\alpha,\beta\geq 0} \mathcal{L}(w, b, \alpha,)$$

and dual of this problem,

$$\Gamma_{\mathcal{D}}\left(\alpha\right) = \min_{w} \mathcal{L}\left(w, b, \alpha\right)$$

Using the *KKT condition*,

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

$$\implies w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

$$\nabla_b \mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Putting these above equations in

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left(x^{(i)} \right)^T x^{(j)}$$

So we get dual optimization problem

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left(x^{(i)} \right)^T x^{(j)}$$

such that

$$\alpha_i \geq 0, i = 1, \ldots, m$$

$$\sum_{i}^{m} \alpha_i y^{(i)} = 0$$

So we can solve the dual optimization problem instead of primal optimization problem as we know $d^* = p^*$.

3.3.6 Kernel Trick

Suppose we have inputs with an attribute, x, this is called information in input or data space. Now we define new function as vector

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

Function $\phi(x)$ maps the input to higher dimension, it is called higher dimensional mapping from input space to feature space.

So if we have any dot product $\langle x, z \rangle$ which can be translated to $\langle \phi(x), \phi(z) \rangle$ in feature space.

So we define Kernel as

$$K(x, z) = \left(\phi\left(x\right)\right)^T \phi\left(z\right)$$

For our discussion, we are considering, Kernel function

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

above equation is called Gaussian Kernel,

where σ is variance, a predefined constant.

Equations of kernel functions suggest us that if both x and z are very near in value of features then kernel function will yield big magnitude, but x and z are nearly orthogonal to each other then kernel value will tend to zero.

3.3.7 Mercer Kernel

Suppose, $K(x, z)$, is a valid kernel function as mentioned in section 3.3.6 using mapping function ϕ.

If we have m points then we can define a matrix using $m \times m$ operations of $K(x, z)$

$$K = \left(\left(K_{ij}\right)_{m\times m}\right)$$

where (i, j) entry of matrix K, K_{ij} is $K(x_i, x_j)$

for $i, j = 1, \ldots, m$

This kind of matrix is called *Kernel Matrix*.

Kernel Matrix has some interesting properties,

1. $K_{ij} = \left(\phi\left(x_i\right)\right)^T \phi\left(x_j\right) = \left(\phi\left(x_j\right)\right)^T \phi\left(x_i\right) = K_{ji}$, So it is symmetric matrix.

2. Suppose, z is a vector of size $m \times 1$,

$$\begin{aligned}
z^T K z &= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i K i j z_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i \left(\phi\left(x_i\right)\right)^T \left(\phi\left(x_j\right)\right) z_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} z_i \left(\sum_{k=1}^{l} \phi_k\left(x_i\right) \phi_k\left(x_j\right)\right) z_j \\
&= \sum_{k=1}^{l}\sum_{i=1}^{m}\sum_{j=1}^{m} z_i \phi_k\left(x_i\right) \phi_k\left(x_j\right) z_j \\
&= \sum_{k=1}^{l} \left(\sum_{i=1}^{m} z_i \phi_k\left(x_i\right)\right)^2 \\
&\geq 0
\end{aligned}$$

When a matrix holds this property it is called positive semi-definite matrix.

The matrix which is symmetric and positive semi-definite is called Mercer matrix.

3.4 Naive Online Risk Minimization Algorithm (NORMA)

This generalized algorithm [5] suggests a way to use SVM in online setup. It does learning in reproducing kernel Hilbert space with stochastic gradient descent method.

3.4.1 Reproducing Kernel Hilbert Space for Stochastic Gradient Descent

3.4.1.1 Properties of Reproducing Kernel Hilbert Space

If we have a kernel function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ with dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$

1. Kernel k has reproducing property

$$\big\langle f, k(x, \cdot) \big\rangle_{\mathcal{H}} = f(x)$$

   for all $x \in \mathbb{X}$

2. f can be expressed as linear combination of kernel function in $\mathcal{H}$. Inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ induces norm such that $f \in \mathcal{H}$ we can get the norm of f as

$$\|f\| = \langle f, f \rangle_{\mathcal{H}}^{\frac{1}{2}}$$

3.4.1.2 Risk Function

When the probability distribution is known to use in batch learning process such that $\mathcal{P} : \mathbb{X} \times \mathbb{Y} \to \mathbb{R}$, where $\mathcal{P} \in [0, 1]$, we can estimate the risk or error

in following manner,

$$R[f, \mathcal{P}] = E_{(x,y)} \left[l\left(f(x), y\right) \right]$$

where $l(\cdot)$ is called loss function.

In case of our classification problem

$$l\left(f(x), y\right)) = \max\left(0, \rho - yf(x)\right)$$

As we are dealing with unknown $\mathcal{P}$ and we get m instances for batch learning process we can take the *empirical risk* as

$$R_{emp}\left[f, \mathcal{S}\right] = \frac{1}{m} \sum_{i=1}^{m} l\left(f\left(x\right), y\right)$$

where $\mathcal{S}$ is the set of observation for training in batch mode.

To avoid overfitting while minimizing the $R_{emp}\left[f, \mathcal{S}\right]$ we can use *regularized risk* function as

$$R_{reg}\left[f, \mathcal{S}\right] = \frac{1}{m} \sum_{i=1}^{m} l\left(f\left(x\right), y\right) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$

, where $\lambda > 0$ and as given in section 3.4.1.1 , $\|f\| = \langle f, f \rangle_{\mathcal{H}}^{\frac{1}{2}}$.

While we are dealing in online training, we are getting single instance at a time, so that time the $R_{reg}\left[f, \mathcal{S}\right]$ will be treated as *instantaneous risk* for observation (x, y)

$$R_{inst}\left[f, x, y\right] = l\left(f\left(x\right), y\right) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$

, where $m = 1$

3.4.1.3 Stochastic Gradient Update Rule

By definition of simple gradient descent, we can update the f using equation

$$f_{t+1} = f_t - \eta \partial_{f_t} R_{inst}\left[f_t, x_t, y_t\right]$$

where η is learning rate and as we know

$$R_{inst}\left[f_t, x_t, y_t\right] = l\left(f(x_t), y_t\right) + \frac{\lambda}{2}\|f_t\|_{\mathcal{H}}^2$$

and

$$\partial_{f_t}\|f_t\|_{\mathcal{H}}^2 = 2f_t$$

$$f_{t+1} = (1 - \eta\lambda) f_t - \eta l^{'}\left(f_t\left(x_t\right), y_t\right) k(x_t, \cdot)$$

from the above equation if we use the concept of kernels multiplied with Lagrangian multiplier or coefficient α values,

new α_t we obtain as

$$\alpha_t = -\eta l^{'}\left(f_t\left(x_t\right), y_t\right)$$

and for

$$\alpha_i = (1 - \eta\lambda)\,\alpha_i$$

for $i = 1, \ldots, (t-1)$

If $g\left(x\right)$ is in the form

$$g\left(x\right) = f_t\left(x\right) + b$$

where b is bias and $b \in \mathbb{R}$

b bias will be updated as

$$b_{t+1} = b_t - \eta \partial_b R_{inst}\left[g_t, x_t, y_t\right]$$

$$\implies b_{t+1} = b_t - \eta l^{'}\left(f_t\left(x_t\right), y_t\right)$$

3.5 Primal L1-SVM

As discussed in previous section 3.3, we can get dual of primal of

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$

subjected to

$$y^{(i)}\left(w^T x_i + b\right) \geq 1 - \xi_i, i = 1, \ldots, m$$

$$\xi_i \geq 0$$

for linearly non-separable data space, or

$$y^{(i)}\left(w^T \phi\left(x_i\right) + b\right) \geq 1 - \xi_i, i = 1, \ldots, m$$

in higher dimension feature space, where $y^{(i)} \in \{-1, +1\}$, w is weight vector, ϕ is mapping function from lower data space to higher dimension feature space, b is bias $\in \mathbb{R}$ and C is margin parameter which determines trade off between maximization of margin and minimization of error ξ_i.

As classification error ξ_i is expressed in linear, for this kind of primal SVM is called L1-SVM[7].

We already know that main reason of converting primal problem in weight vector term w to dual problem in term of Lagrangian α coefficients is that dimension of w may be very large and we need to compute $w^T x$ for any given observations or input points.

To avoid that multiplication for very high dimension input space, we convert the primal problem to dual.

We use quadratic optimization solver for dual of minimization problem expressed in Lagrangian multipliers, α coefficients are mostly zero. There are non-zero α values for support vectors. But before getting α coefficient values, we need to deal with huge dataset for quadratic or superlinear operations which are some time very complicated to manipulate.

But when we know the dimension for particular feature space and which is reasonable to calculate $w^T x$, primal method with this approach can be used.

Suppose we have augmented and reflected space as discussed in section 4.3.1, $\mathbb{I} = \{(y_i x_i, y_i \rho)\}_{i=1}^{m}$ and $\mathbb{I} \subset \mathbb{R}^{n+1}$ where n is number of attributes in input or feature space.

Let $\hat{x}_i \in \mathbb{I}$ and w is the weight vector, $w \in \mathbb{R}^{n+1}$

The equation of regularized empirical risk will be

$$\frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\left(0, 1 - w^T \hat{x}_i\right)$$

, where λ is regularization parameter.

$$\implies \frac{1}{2}\|w\|^2 + \frac{1}{\lambda m}\sum_{i=1}^{m} \max\left(0, 1 - w^T \hat{x}_i\right)$$

taking $C = \frac{1}{\lambda m}$

$$\implies \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \max\left(0, 1 - w^T \hat{x}_i\right)$$

, we call C as penalty parameter.

as $\lambda > 0$ and $m > 0$, which implies $C > 0$

if $m = 1$, that is, we are considering single instance at a time, this equation can be treated for online learning too.

$$\implies \frac{1}{2}\|w\|^2 + C\max\left(0, 1 - w^T \hat{x}_i\right)$$

, where $C = 1/\lambda$

$$\implies \frac{1}{2}\|w\|^2 + \frac{1}{\lambda}\max\left(0, 1 - w^T \hat{x}_i\right)$$

Using gradient descent update rule,

$$w_{t+1} = w_t - \eta_t \nabla_{w_t} \left[\frac{1}{2} \|w_t\|^2 + \frac{1}{\lambda} \max\left(0, 1 - w_t^T \hat{x}_i\right) \right]$$

When $w_t^T \hat{x}_i \leq 1$

$$w_{t+1} = w_t - \eta_t \left[w_t + \frac{1}{\lambda} \left(-\hat{x}_i\right) \right]$$

If we take η_t at each step t as $\eta_t = \frac{1}{t+1}$

When $w_t^T \hat{x}_i \leq 1$

$$w_{t+1} = \frac{t}{t+1} w_t + \frac{\hat{x}_i}{\lambda\,(t+1)}$$

otherwise

$$w_{t+1} = \frac{t}{t+1} w_t$$

, that is, $\frac{t}{t+1}$ is acting as forgetting factor.

If we consider

$$w_t = \frac{\alpha_t}{\lambda t}$$

above update equations can be reduced to

When $\alpha_t^T \hat{x}_i \leq \lambda t$

$$w_{t+1} = \frac{t}{(t+1)} \frac{\alpha_t}{\lambda t} + \frac{\hat{x}_i}{\lambda\,(t+1)}$$

$$\implies \frac{\alpha_{t+1}}{\lambda\,(t+1)} = \frac{\alpha_t}{\lambda\,(t+1)} + \frac{\hat{x}_i}{\lambda\,(t+1)}$$

$$\implies \alpha_{t+1} = \alpha_t + \hat{x}_i$$

otherwise

$$w_{t+1} = \frac{t}{(t+1)} \frac{\alpha_t}{\lambda t}$$

$$\implies \frac{\alpha_{t+1}}{\lambda\,(t+1)} = \frac{\alpha_t}{\lambda\,(t+1)}$$

$$\implies \alpha_{t+1} = \alpha_t$$

Which is similar as classical perceptron update rules.

3.6 *k*-Nearest Neighbour

k-Nearest Neighbour classification algorithm or k-NN in short is easiest to comprehend and to implement. Initially k-NN starts with small number of instances which are labelled. After that when new instance with no label is presented in dataset, using k most nearest neighbours' class label information, we can find the label of that new instance.

As this classifier does not do any weight updating or optimization process when labelled training dataset is presented, it is called *Lazy learner*.

Most of the cases the value " k" is chosen as odd number so it will help to break the tie. It is non-parametric method of classification as only " k" is used and no other parameter is required, such as regularization parameter, variance or gamma constant for RBF. But it is very resource intensive algorithm and it requires *Euclidean distance* for each instance present in training dataset. When number of element in dataset is very huge it will not be very effective in term of performance and throughput.

Chapter 4

Utilized Algorithms

4.1 General Idea

We assume that meteorological data are available in bulk in many data repositories and we are taking assumption that weather patterns are not changing abruptly over span of 6 years. With those data, we tried to predict or test wind speed prediction prior 1 hour, 6 hours, 12 hours and 24 hours.

Here we did two types of experiments. First, in which we start with small labelled instances and try to provide correct labelling to other unlabelled instances using prior knowledge obtained from labelled instances. Second, predicting next wind speed prior 5 minutes, 10 minutes, 15 minutes, 30 minutes , 1 hour, 6 hours, 12 hours and 24 hours in online setup. Observations are coming one after another, and predicting next wind speed class.

When we are getting data one after another we are using Primal L1-SVM in online mode. The updating method is very much similar of online single layer neural network update. Our aim is to predict the wind speed category with ever changing situation.

Similarly, NORMA[5] is also used for online prediction.

4.2 SVM using LIBSVM

Support vector machine is very effective in term of binary classification. Although we can implement support vector machine for regression and novelty detection, we are focusing on classification version of SVM or SVC. We are interested in LIBSVM [26] as it is freely available and doing the optimization problem without any user's involvement to solve quadratic programming. While discussing Support Vector Machine in detail in section 3.3 we mentioned for solving the dual of primal optimization problem we need to use some sort of commercial quadratic program solver, we are using LIBSVM.

We can integrate LIBSVM in different interfaces like C, C++, JAVA, python and matlab.

Typical use of LIBSVM is to train the model with labelled data and predict the labels of unlabelled data. For SVC, we require to mention C parameter value of regularization parameter for trade off between error and maximization of margin. Along with regularization parameter, we can mention the kernel type and gamma parameter $\gamma = \left(\frac{1}{2*variance}\right)$.

Also LIBSVM provides some special setting to deal with imbalanced dataset. For using different cost values for classes, we can provide as argument to LIBSVM.

Although we are mentioning LIBSVM mainly for binary classification problem, it can do multi-class classification too.

4.2.1 Batch Mode Learning

4.2.1.1 Predicting Held Out Portion of Training Data

As for huge dataset and higher dimensional feature space, we can train model of SVM with a portion of training dataset which is in standardised form.

Using same parameter like C and γ, predict the held out dataset. All training data are standardised using following equation, Where μ and σ are mean and standard deviation of training dataset.

Standardised feature vector

$$\hat{x_i} = \frac{x_i - \mu}{\sigma}$$

4.2.1.2 Algorithm

1. Find the mean μ and standard deviation σ for training dataset.
2. Standardise whole training data using following formula

$$\hat{x_i} = \frac{x_i - \mu}{\sigma}$$

3. Find the C regularization parameter and other parameters related to kernel type using cross validation using grid search.
4. Train the SVM model with C and other parameters.
5. Use the model to predict held out dataset.

4.2.1.3 Prediction of Labels of Testing Data

In this case, similar to previous algorithm 4.2.1.2, mean (μ) and variance (σ) of training dataset are used, and testing dataset should be standardised using those values .

4.2.2 Active Learning Approach

In case of Active learning with SVM, we start with initial set of 60 points containing atleast one instance from each class for a particular year of dataset. Train the model with those 60 training instances. Try to predict the labels

of next randomly selected 60 instances from dataset and compute the decision confidence. We only consider those points which are within decision confidence -1 to +1 or between -2 and +2 or between +4 to -2 depending our experiment and those points are active learning instances. Add those instances to training set for next training. Training set of next iteration contains active learning instances and only support vectors of model of last training. This process continues until all training instances are used up obeying without replacement policy.

4.2.2.1 Algorithm

1. Select randomly, initial training set without replacement policy of size 60.
2. Train LIBSVM model.
3. Select next 60 instances for predicting labels without replacement.
4. Measure the decision confidence for each instance for label prediction, selected in last step(step 3).
5. Mark instances which are between predefined decision confidence values. These are active learning instances.
6. Create fresh training set with support vectors in last model training.
7. Add those active learning instances with corresponding labels to newly created training dataset.
8. Retrain the model as going to step 2, until dataset is exhausted.
9. Predict the labels of instances using model trained with active learning approach in previous steps.

4.3 Stochastic Gradient Descent Primal L1-SVM

As earlier mentioned in the section 3.5 for primal optimization problem of L1-SVM, here the algorithm and required data structure are discussed[7].

4.3.1 Data Structure

If given training set is $\{(x_i, y_i)\}_{i=1}^{m}$, where $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$, m is number of instances in training set, n is the number of features. New augmented training set is created from input data space by appending a new dimension to it. After adding new dimenstion to existing feature space we get $\{(x_i, \rho)\}_{i=1}^{m}$ in augmented space. We choose ρ as 1.

We multiply the each augmented instance with respective label. So we get augmented and reflected instances as $\mathbb{I} = \{(y_i x_i, y_i \rho)\}$.

4.3.2 Online Learning Approach

4.3.2.1 Algorithm

1. Set penalty parameter C.
2. Set regularization parameter as $\lambda = 1/C$
3. Set α vector to $\mathbf{0}$ where $\alpha \in \mathbb{R}^{n+1}$.
4. Set $t = 0$, where t is number of iterations still executed.
5. Calculate label of x_t observation where x_t in augmented space, when $t > 0$

$$y_t^{predicted} = sign\left(\frac{1}{\lambda t}\alpha^T x_t\right)$$

when $t = 0$, $y_t^{predicted} = 0$

6. if $\alpha^T (y_t x_t) \leq \lambda t$ then

$$\alpha = \alpha + y_t x_t$$

7. Increment t,

$$t = t + 1$$

8. Go to step 5.

4.4 NORMA

It is online learning methodology. It is using kernel and corresponding coefficient to control the effect of recently used data points instead of keeping all previous observations. It considers truncation mechanism to remove very old kernel too.

4.4.1 Algorithm

1. Set learning rate η and regularization parameter λ such that, $\lambda > 0$ and $\eta < 1/\lambda$.

2. Set *margin parameter*, ρ such that, $\rho \geq 0$.

3. Set *trunction parameter* τ. Whenever algorithm is running we are handling online last τ number of kernels and respective coefficients.

4. Set coefficient vector, say α, of size τ to 0.

5. Set τ number of kernels of size n to 0, where n is number of feature of a instance.

6. Set bias b to zero where $b \in \mathbb{R}$.

7. Calculate constant product term $\gamma = (1 - \eta\lambda)$. It is called forgetting factor.

8. Find new location for α_t, say, $alpha\_index$ in α vector.

$$alpha\_index = t\%\tau + 1$$

9. Calculate the function value $f(x_t) = \sum_{i=1}^{\tau} \alpha_i k(x_i, x_t)$ for observation x_t found on $t\text{-}th$ step iteration. $y_t^{predicted} = sign\left(f\left(x_t\right)\right)$.

10. if $y_t f(x_t) \leq \rho$ then

$$\sigma = 1$$

else

$$\sigma = 0$$

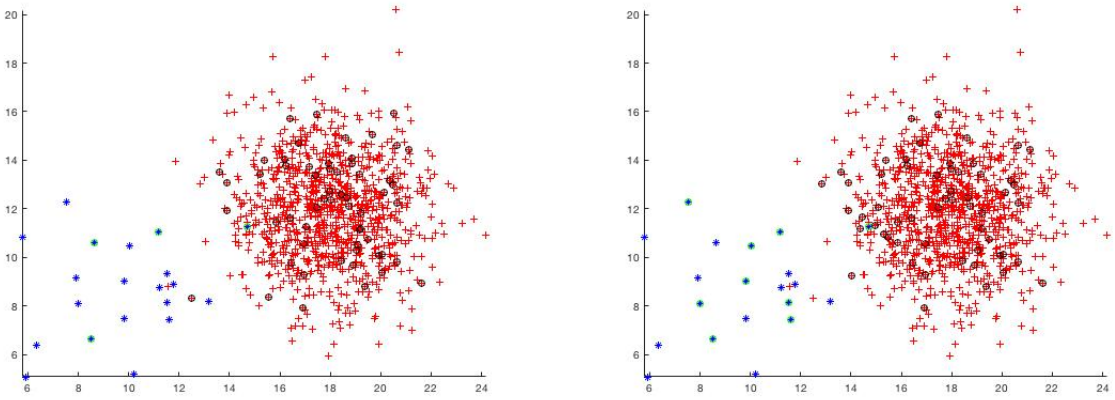
11. Calculate

    (a) $\alpha = \gamma\alpha$

    (b) $\alpha_{alpha\_index} = \eta\sigma y_t$.

    (c) $b = b + \eta\sigma y_t$

12. go to step 8.



(a) Before execution

(b) After execution

Figure 4.1: 2-dimensional data classification using proposed algorithm, left hand side, is picture of before execution and right side is picture of completion

4.5 Proposed Algorithm

We are in search for a simpler algorithm for online learning using concept of k-NN and support vector of SVM. It is in initial stage of exploration. We did it in experimental basis whether it can provide any reasonable outcome.

4.5.1 Algorithm

1. Set initial centroids for positive class instances and for negative class instances using available instances. Say, maximum number of centroids for each class is 10.

2. Set the variance parameter for RBF.

3. New instance is acquired for classification.

4. Calculate the RBF value from positive centroids and negative centroids.

5. Find the maximum value of RBF for positive(say $I_{positive}$) and negative($I_{negative}$) for the instance from centroids.

6. If $I_{positive} \geq I_{negative}$ then classify as positive else negative.

7. If number of centroids for respective class is less than maximum no of centroids then we can add instance as new centroid. Else go to next step.

8. If distance between two centroids which are nearest to instance, say $D_{centroids}$, $D_{centroids} \geq I_{positive}$ and $D_{centroids} \geq I_{negative}$, then replace nearest centroid with instance.

9. Calculate the average RBF value of each positive centroid from negative centroids, and average RBF value from other positive centroids. If average RBF value of positive centroid from other positive centroids is less than negative centroids then remove that centroid.

10. Similarly calculate the average RBF value of each negative centroids from positve centroids and average RBF value from other negative centroids. If average RBF value of negative centroid from other negative centroids is less than positive centroids then remove that centroid.

11. Go to step 3.

Chapter 5

Statistical Information

5.1 Dataset

We have downloaded all 6 years of wind speed dataset from https://maps.nrel.gov/wind-prospector/ . It contains variety of required dataset. We are only focusing at particular location. It is possible to download multiple datasets of different locations. These data are freely available for research pertaining to renewable energy.

5.1.1 Attributes present in dataset

1. Day
2. Month
3. Hour
4. Minute
5. Wind Power(MW)
6. Wind Direction(deg)

7. Wind Speed(m/s)

8. Temperature(K)

9. Surface Air Pressure($Pascal$)

10. Density(Kg/m^3)

Dataset is available for 2007 to 2012. Every five minute, above mentioned observations were collected. So every one hour we have 12 observations. It is found that above mentioned attributes of observations are most relevant in case of wind speed prediction. It was supported by experiments [15] that already present attributes are most relevant, so we do not need to do feature selection separately. While we are predicting in online setup, every observation is separated by 5 minutes from previous and next observation.

From that collection of observations, we prepare observations separated by 10 minutes, 15 minutes, 30 minutes and 1 hour. While we extracting data for 10 minutes and 15 minutes we only include average of 2 observations and 3 observations of original dataset,respectively.

1. Wind Power(MW)

2. Wind Direction(deg)

3. Wind Speed(m/s)

4. Temperature (Kelvin)

5. Surface Air Pressure (Pascal)

6. Air Density(Kg/m^3)

For 30 minutes and 1 hours, aggregated dataset are prepared by taking average of 6 and 12 observations respectively.

1. Wind Power(MW)
2. Wind Direction(deg)
3. Wind Speed(m/s)
4. Temperature (Kelvin)
5. Surface Air Pressure (Pascal)
6. Air Density (Kg/m^3)
7. Median of wind speed (of 6 or 12 observations)
8. Mode of wind speed (of 6 or 12 observations)
9. Variance of wind speed (of 6 or 12 observations)

As shown in above list, mean, mode and variance of wind speed are added as new attributes.

5.1.2 Definitions of Attributes

5.1.2.1 Wind Speed

Wind Speed is measure of airflow, how fast it is blowing. It is usually measured in knots, km/h or m/s. In this paper we are considering only in m/s unit. Wind speed is categorised in *Beaufort Wind Scale*. By the categorisation, speeds above 17 m/s is called *Gale* type wind.

Different types of wind

1. **Gust wind** is sudden wind speed increase for few seconds from average wind speed. Gust are usually 30 to 40 percent higher than average wind speed.

2. **Squall wind** is abrupt and large increase in wind speed which lasts for few minutes and diminishes suddenly.

3. **Lull wind** short duration decrease in wind speed than average speed for few seconds.

4. **Gale wind** when average wind speed is more than 17m/s.

5. **Storm** When average wind speed is above 25m/s.

In given dataset, all wind speed measurements are done at hight of $100m$ from surface. We are classifying speed which is greater than 18 m/s as +1 and rest as -1.

In the given figure 5.1, it is found that where data points recorded, high wind speed is directly related to direction. When wind direction is between 150° to 250°, the maximum wind speed is observed.

5.1.2.2 Wind Direction

As name suggests, it is providing the wind flow direction. It is measurement of wind flow from which direction. If wind flows from north direction to south direction then it is called northerly wind. Wind direction is measured in degrees clockwise from due north and so a wind coming from the south has a wind direction of 180°; one from the east is 90°.

Wind speed and direction are measured with help of anemometer and wind vane.

5.1.2.3 Temperature

Air temperature is the most frequently measured information pertaining to weather prediction. In this dataset temperatures are measured in Kelvin.

5.1.2.4 Surface Air Pressure

Wind pressure is also very helpful parameter for wind speed prediction. When atmospheric pressure is reduced that time probability of high wind speed increases. If we notice in the figure 5.2, where 5th and 6th column are air pressure or atmospheric pressure and wind speed respectively. It is prominently visible that whenever there is drop of air pressure, wind speed increases in that area. Where is low pressure, winds are flowing from other places to that location.

Air pressures are measured in *Pascal*, if q is the air pressure then $q = \frac{1}{2}\rho W^2$, where ρ is air density and W is wind speed in m/s.

5.1.2.5 Air Density

Air density is represented by ρ, which is the measure of air mass per unit volume.

In this dataset air density is measured in Kg/m^3 unit.

The mathematical relation between air density ρ and air pressure and temperature as follows,

$$\rho = \frac{p}{R_{const}T}$$

where

p in Pascal

T in Kelvin

R_{const} is air specific constant

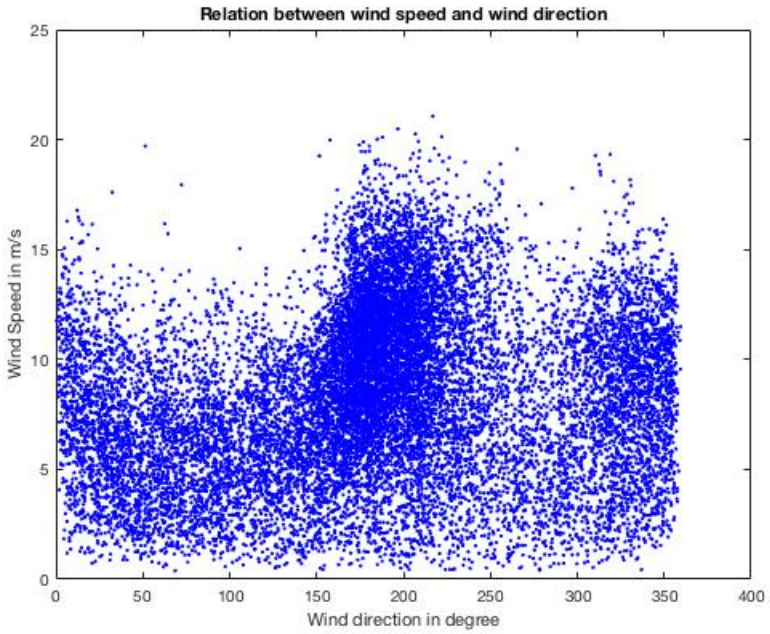


Figure 5.1: Relation of wind speed and wind direction

5.2 Feature Selection Methods

5.2.1 Principal Component Analysis

To identify the principal component, we did singular value decomposition. Steps are described in followings.

1. Calculate the mean of whole dataset.

2. Subtract mean vector from all observations. This step helps us to shift origin to centre of data points.

$$X_{centralized} = (X - \mu)$$

, where X is collection of all data and μ is mean vector of X.

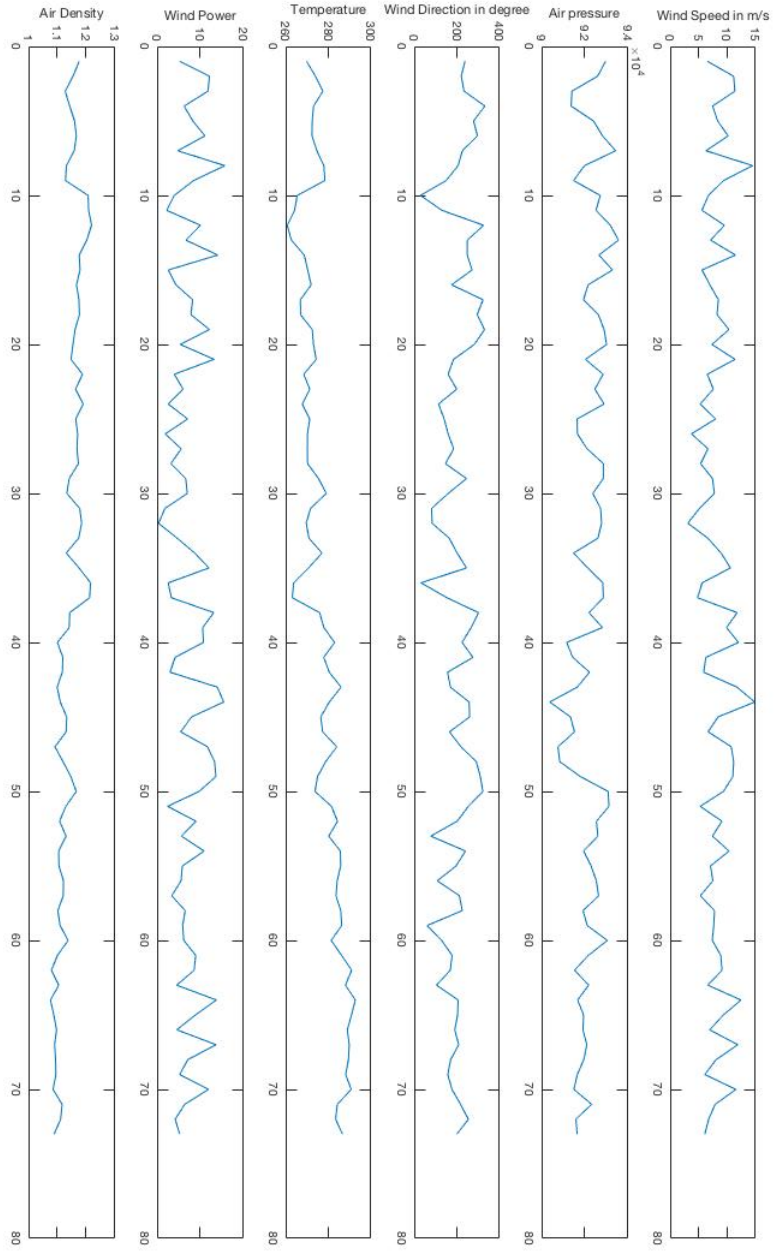


Figure 5.2: Relation of wind speed with different other parameters

3. Finding the covariance matrix of $X_{centralized}$

$$COV_{X_{centralized}} = \frac{1}{m-1}\left\{X^T_{centralized}X_{centralized}\right\}$$

, where m is total number of observations.

4. Compute singular value decomposition using matlab function

$$[U, S, V] = SVD(COV_{X_{centralized}})$$

,where each column of U represents the eigen vector with non-increasing variance.

5. Getting first K features from $X_{centralized}$ as

$$X_{pca} = X_{centralized}U(:, 1:K)$$

, where we need first K principal component.

5.2.2 Autocorrelated Features

Autocorrelation between wind speed of different lags are shown in figure 5.3

It is found that upto 72 lags, that is, 6 hours we have autocorrelation coefficient more than 0.5. So we added one sample for last 6 hours. So we have 6 additional features along with above mentioned features.

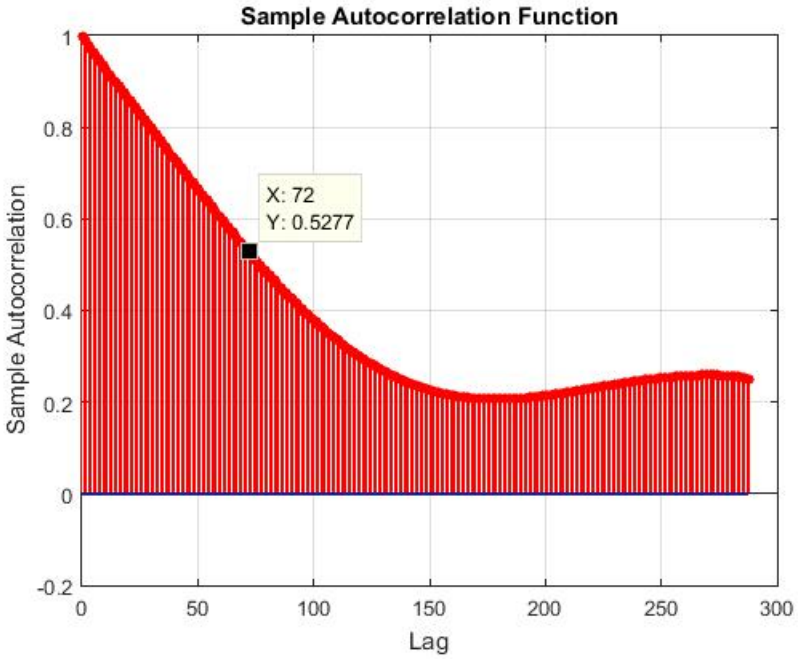


Figure 5.3: Autocorrelation of wind speed

5.3 Experimental Results

5.3.1 Usual SVM Approach to Check Training Performance

As shown in table 5.1, It shows different measures of training set while it is trained with all labelled instances and with predefined parameters $C = 200$ and $\gamma = 0.5$ for 1 hour ahead prediction. It is used as standard against which all other outcomes are compared.

Last row of the table 5.1 shows how many support vectors are required to classify the training set with high recall and precision.

Learning performance of SVM with All labelled Training Data							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.9296	0.9013	0.8436	0.9036	0.8784	0.8564	0.6382
Precision	0.9882	0.9468	0.9648	0.9944	0.9956	0.9514	0.9285
Specificity	0.9999	0.9996	0.9999	0.9999	0.9999	0.9998	0.9998
Gmean	0.9642	0.9492	0.9184	0.9505	0.9372	0.9253	0.7988
F1-Score/Measure	0.9580	0.9235	0.9001	0.9468	0.9333	0.9014	0.7565
No of SV	464	976	891	495	712	1106	5853

Table 5.1: SVM Learning performance from labelled dataset with $C = 200$ and $\gamma = 0.5$ for 1 hour ahead

5.3.2 Labelling Training Dataset with Active Learning

5.3.2.1 Experiment Type 1

In this experiment, initial 60 instances are selected as described in section 4.2.2.1. It ensures atleast one instance from minority class is present. The result of experiment is shown for 1 hour ahead prediction in table 5.2 and 5.3. In the table 5.2 margin of -1 to +1 is used. In table 5.3, margin from -2 to +4 is used to give preference to minority class. In the figure 5.4 shows how number of support vectors converges for active learning process with iterations.

It shows that if we use asymmetric margin for minority and majority class we can get better recall and precision.

We notice that in this configuration all active learning instances are not part of support vectors of learning model. Very few of the active instances are support vector of ultimate model.

5.3.2.2 Experiment Type 2

Similar type of active learning experiment is done with RVFL neural network. As it is binary classification problem, we are taking the difference of two output neurons. If the output difference is less than 0.01 for any instance, that instance is used for active learning.

This experiment is done to show, if we keep precision high then recall is less for neural network setup, (table 5.4). Neural network is not good performer of imbalanced dataset. Training of RVFL with 1:1 ratio of minority and majority class is also done but if suffers very low precision error.

5.3.2.3 Experiment Type 3

This active learning experiment is done with certain constraint on training set. The margin is from -1 to +0.5. Regularization parameter $C = 0.9$ and $\gamma = 100$ are decided after some trial and error.

But while we train, we keep positive(minority) and negative(majority) instances in 1:1 ratio. Select one positive active learning instance with one negative active learning instance.

When we get the active learning set which is not growing any more, we train with just those active learning instances at last step. Then we test the model with rest of the instances which are not selected as active learning instances.

From this experiment we can get great recall (1) but precision suffers. For this reason we can assure that we do not have False Negative error. When we get positive class predicted instances, we can verify as the number is low.

This problem is observed that, if we keep recall high, we suffer with precision.

Tests are done for 1 hour, 6 hour, 12 hour and 24 hour ahead prediction

Active Learning of Training Data with LIBSVM tool							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6407	0.7693	0.5829	0.5685	0.7317	0.6305	0.5161
Precision	0.9505	0.9444	0.8849	0.9655	0.7402	0.8571	0.8467
Specificity	0.9999	0.9997	0.9997	1.0000	0.9987	0.9994	0.9996
Gmean	0.8004	0.8770	0.7634	0.7540	0.8548	0.7938	0.7183
F1-Score/Measure	0.7655	0.8479	0.7029	0.7157	0.7359	0.7265	0.6413
No of SV	202	518	338	218	326	363	1686
No of Active Inst.	1223	2454	1446	1276	1384	1288	4938
Percentage of labelling	1.20	2.33	1.38	1.21	1.31	1.22	0.783

Table 5.2: Labelling of training set of size 105120 each with Active Learning, except last column which is for 6 years, where $C = 200$ and $\gamma = 0.5$, for 1 hour ahead prediction, instances within range of -1 and +1 are considered for active learning labelling.

labelling as shown in table 5.5,5.6, 5.7, 5.8 respectively.

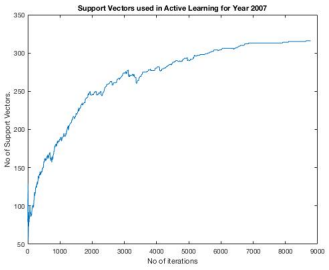
But it saves lot of effort for labelling other negative instances, quantity of which is more than 100000.

Last row of those table shows percentage of whole dataset we required to label to obtain displayed recall and precision.

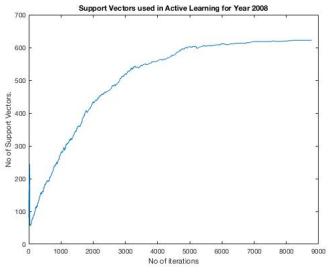
Another interesting observation is that mostly all active learning instances are support vector of learned model. We can check the row with No of SV and No of Active Instances.

Active Learning of Training Data with LIBSVM tool							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.8185	0.8160	0.7251	0.7107	0.8108	0.7356	0.5224
Precision	0.9866	0.9474	0.9623	0.9790	0.9976	0.9272	0.8968
Specificity	0.9999	0.9997	0.9999	0.9999	0.9999	0.9997	0.9997
Gmean	0.9047	0.9032	0.8515	0.8430	0.9004	0.8575	0.7227
F1-Score/Measure	0.8947	0.8768	0.8270	0.8235	0.8946	0.8203	0.6602
No of SV	316	640	598	278	531	657	3243
No of Active Inst.	9176	15426	9561	5664	9852	6144	34985
Percent. of labelling	8.73	14.67	9.10	5.39	9.37	5.84	5.55

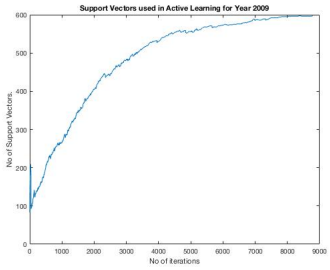
Table 5.3: Labelling of training set of size 105120 each with Active Learning, except last column which is for 6 years, where $C = 200$ and $\gamma = 0.5$, for 1 hour ahead prediction, instances within range of -2 and +4 are considered for active learning labelling.



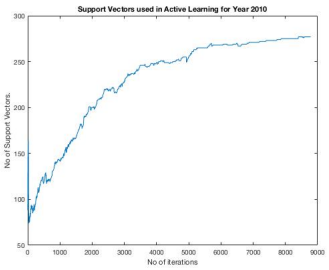
(a) Number of Support Vector change with iteration in active learning for 2007



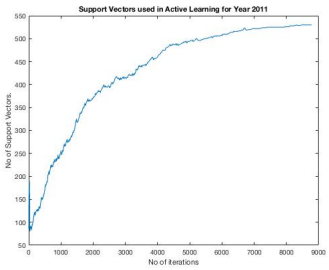
(b) Number of Support Vector change with iteration in active learning for 2008



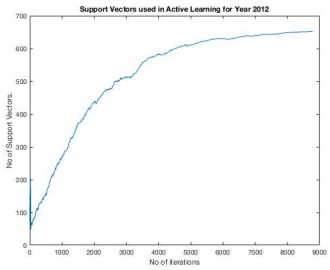
(c) Number of Support Vector change with iteration in active learning for 2009



(d) Number of Support Vector change with iteration in active learning for 2010



(e) Number of Support Vector change with iteration in active learning for 2011



(f) Number of Support Vector change with iteration in active learning for 2012

Figure 5.4: Relation between number of iterations and number of support vectors

Active Learning of Training Data with RVFL							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.3556	0.4427	0.0592	0	0.3861	0.2277	0.1851
Precision	0.7273	0.6917	0.6944	0	0.9009	0.6250	0.7143
Specificity	0.9997	0.9986	0.9999	1	0.9998	0.9993	0.9997
Gmean	0.5962	0.6649	0.2434	0	0.6213	0.4770	0.4302
F1-Score/Measure	0.4776	0.5398	0.1092	0	0.5405	0.3338	0.2940
No of Active Inst.	143	244	89	51	124	283	650
Percent. of labelling	0.14	0.23	0.084	0.049	0.12	0.27	0.10

Table 5.4: Labelling of training set of size 105120 each for 1 hour ahead prediction with Active Learning, except last column which is for 6 years, when difference between output values of two neurons is less than equal to 0.01, those instances are identified as active instance.

Testing of Active Learning from Training Data with LIBSVM tool							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	1 (15/15)	1 (125/125)	1 (20/20)	1 (2/2)	1 (85/85)	1 (102/102)	1 (335/335)
Precision	0.4054 (15/37)	0.6098 (125/205)	0.2667 (20/75)	0.0870 (2/23)	0.4670 (85/182)	0.5312 (102/192)	0.4583 (335/731)
Specificity	0.9998	0.9992	0.9995	0.9998	0.9991	0.9991	0.9994
Gmean	0.9999	0.9996	0.9997	0.9999	0.9995	0.9996	0.9997
F1-Score/Measure	0.5769	0.7576	0.4211	0.1600	0.6367	0.6939	0.6285
No of SV	511	1250	805	393	869	939	4783
No of Active Instances	511	1250	805	393	869	939	4786
Percentage of labelling	0.49	1.19	0.77	0.37	0.83	0.89	0.76

Table 5.5: Labelling of training set of size 105120 each with Active Learning, except last column which is for 6 years, where $C = 0.9$ and $\gamma = 100$, for 1 hour ahead prediction, instances within range of -1 and +0.5 are considered for active learning labelling.

Testing of Active Learning of Training Data with LIBSVM tool							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	1 (39/39)	1 (179/179)	1 (66/66)	1 (37/37)	1 (59/59)	1 (57/57)	1 (416/416)
Precision	0.3482 (39/112)	0.5375 (179/333))	0.2578 (39/256)	0.2387 (37/155)	0.2850 (59/207)	0.2457 (59/232)	0.2675 (416/1555)
Specificity	0.9993	0.9985	0.9982	0.9989	0.9986	0.9983	0.9982
Gmean	0.9997	0.9993	0.9991	0.9994	0.9993	0.9992	0.9991
F1-Score/Measure	0.5166	0.6992	0.4099	0.3854	0.4436	0.3945	0.4221
No of SV	463	1140	713	323	919	1031	4623
No of Active Instances	463	1143	713	323	919	1031	4624
Percentage of labelling	0.44	1.09	0.68	0.31	0.87	0.98	0.73

Table 5.6: Labelling of training set of size 105120 each with Active Learning, except last column which is for 6 years, where $C = 0.9$ and $\gamma = 100$, for 6 hour ahead prediction, instances within range of -1 and +0.5 are considered for active learning labelling.

Testing of Active Learning of Training Data with LIBSVM tool							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	1 (46/46)	1 (191/191)	1 (71/71)	1 (20/20)	1 (42/42)	1 (117/117)	1 (482/482)
Precision	0.3333 (46/138)	0.5668 (191/337)	0.3381 (71/210)	0.1818 (20/110)	0.2143 (42/196)	0.3993 (117/293)	0.3317 (482/1453)
Specificity	0.9991	0.9986	0.9987	0.9991	0.9985	0.9983	0.9984
Gmean	0.9996	0.9993	0.9993	0.9996	0.9993	0.9992	0.9992
F1-Score/Measure	0.5000	0.7235	0.5053	0.3077	0.3529	0.5707	0.4982
No of SV	449	1119	703	355	952	909	4491
No of Active Instances	449	1119	703	356	952	909	4492
Percentage of labelling	0.43	1.06	0.67	0.34	0.91	0.86	0.71

Table 5.7: Labelling of training set of size 105120 each with Active Learning, except last column which is for 6 years, where $C = 0.9$ and $\gamma = 100$, for 12 hour ahead prediction, instances within range of -1 and +0.5 are considered for active learning labelling.

Testing of Active Learning of Training Data with LIBSVM tool							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	1 (41/41)	1 (197/197)	1 (61/61)	1 (20/20)	1 (64/64)	1 (147/147)	1 (522/522)
Precision	0.2993 (41/137)	0.5310 (197/371)	0.2961 (61/206)	0.1575 (20/127)	0.3832 (64/167)	0.4900 (147/300)	0.3436 (522/1519)
Specificity	0.9991	0.9983	0.9986	0.9990	0.9990	0.9985	0.9984
Gmean	0.9995	0.9992	0.9993	0.9995	0.9995	0.9993	0.9992
F1-Score/Measure	0.4607	0.6937	0.4569	0.2721	0.5541	0.6577	0.5115
No of SV	463	1107	725	355	908	847	4412
No of Active Inst.	463	1107	725	355	908	848	4412
Percentage of labelling	0.44	1.05	0.69	0.34	0.86	0.81	0.70

Table 5.8: Labelling of training set of size 105120 each with Active Learning, except last column which is for 6 years, where $C = 0.9$ and $\gamma = 100$, for 24 hour ahead prediction, instances within range of -1 and +0.5 are considered for active learning labelling.

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.7519	0.8493	0.7725	0.6904	0.8417	0.8144	0.7082
Precision	0.7868	0.8811	0.8089	0.7083	0.8532	0.8318	0.8197
Specificity	0.9995	0.9992	0.9993	0.9995	0.9993	0.9991	0.9993
Gmean	0.8669	0.9212	0.8786	0.8307	0.9171	0.9020	0.8413
F1-Score/Measure	0.7689	0.8649	0.7903	0.6992	0.8474	0.8230	0.7599

Table 5.9: Wind speed prediction at interval of 5 minutes with Stochastic Gradient Descent Primal L1-SVM

5.3.3 Online Learning for Prediction

5.3.3.1 Experiment with Stochastic Gradient Descent Primal L1-SVM

Online classifier SGD primal L1-SVM is used to predict minority class of high wind speed for 5 minute, 10 minute, 15 minute, 1 hour, 6 hour , 12 hour and 24 hour ahead. Results are shown in tables 5.9,5.12,5.15,5.18,5.21,5.23 and 5.25.

It is clearly seen that as prediction gap increases the reliability of recall and precision decreases.

5.3.3.2 Experiment with NORMA

Experiments of NORMA outperform all the results of SGD primal L1-SVM. It is also done for 5 minute, 10 minute, 15 minute, 1 hour , 6 hour , 12 hour and 24 hour ahead prediction.

Observed results are shown in tables 5.10,5.13,5.16,5.19,5.22, 5.24 and 5.26.

Online Training data from 2007 to 2012 and overall 6 years with NORMA							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.8000	0.8880	0.8104	0.7157	0.8687	0.8406	0.8416
Precision	0.7970	0.8868	0.8143	0.7157	0.8687	0.8421	0.8420
Specificity	0.9995	0.9992	0.9993	0.9995	0.9993	0.9991	0.9993
Gmean	0.8942	0.9420	0.8999	0.8458	0.9318	0.9165	0.9171
F1-Score/Measure	0.7985	0.8874	0.8124	0.7157	0.8687	0.8414	0.8418

Table 5.10: Online wind speed prediction at interval of 5 minutes with NORMA

Online Training data from 2007 to 2012 and overall 6 years with proposed algorithm							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.7370	0.8480	0.7464	0.6497	0.8108	0.7723	0.7955
Precision	0.4761	0.7430	0.4730	0.3459	0.6688	0.6733	0.6618
Specificity	0.9979	0.9979	0.9966	0.9977	0.9980	0.9980	0.9982
Gmean	0.8576	0.9199	0.8625	0.8051	0.8996	0.8779	0.8911
F1-Score/Measure	0.5785	0.7920	0.5790	0.4515	0.7330	0.7194	0.7225

Table 5.11: Online wind speed prediction at interval of 5 minutes with proposed algorithm

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.7015	0.8011	0.6634	0.5333	0.7617	0.7382	0.6647
Precision	0.7287	0.8282	0.6154	0.5581	0.7959	0.7660	0.7617
Specificity	0.9993	0.9988	0.9984	0.9993	0.9990	0.9988	0.9991
Gmean	0.8373	0.8945	0.8138	0.7300	0.8723	0.8587	0.8149
F1-Score/Measure	0.7148	0.8144	0.6385	0.5455	0.7784	0.7519	0.7099

Table 5.12: Online wind speed prediction at interval of 10 minutes with stochastic gradient descent Primal L1-SVM with C=15, penalty variable

5.3.3.3 Experiment with Proposed Algorithm

Online experiments with proposed algorithm are carried for 5 minutes, 10 minutes, 15 minutes and 1 hour ahead prediction. Details of experimental results are shown in tables 5.11,5.14,5.17,5.20.

Online Training data from 2007 to 2012 and overall 6 years with NORMA							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.7313	0.8229	0.6829	0.5889	0.7852	0.7673	0.7566
Precision	0.7206	0.8229	0.6965	0.5824	0.7821	0.7729	0.7577
Specificity	0.9993	0.9988	0.9988	0.9993	0.9989	0.9988	0.9990
Gmean	0.8549	0.9066	0.8259	0.7671	0.8856	0.8754	0.8694
F1-Score/Measure	0.7259	0.8229	0.6897	0.5856	0.7836	0.7701	0.7572

Table 5.13: Online wind speed prediction at interval of 10 minutes with NORMA

Online Training from 2007 to 2012 and overall 6 years with proposed algorithm							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6493	0.7657	0.6585	0.5111	0.7422	0.6618	0.7265
Precision	0.2390	0.5643	0.2328	0.1447	0.4299	0.4777	0.4477
Specificity	0.9947	0.9958	0.9915	0.9948	0.9952	0.9962	0.9962
Gmean	0.8036	0.8732	0.8080	0.7131	0.8594	0.8120	0.8507
F1-Score/Measure	0.3494	0.6497	0.3439	0.2255	0.5444	0.5549	0.5540

Table 5.14: Online wind speed prediction at interval 10 minutes with proposed algorithm

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.7664	0.8116	0.7055	0.6750	0.8238	0.7977	0.7153
Precision	0.7736	0.8421	0.7188	0.6835	0.8386	0.8135	0.8087
Specificity	0.9993	0.9988	0.9987	0.9993	0.9990	0.9986	0.9987
Gmean	0.8751	0.9003	0.8394	0.8213	0.9072	0.8925	0.8452
F1-Score/Measure	0.7700	0.8266	0.7121	0.6792	0.8311	0.8055	0.7591

Table 5.15: Online wind speed prediction at interval of 15 minutes with stochastic gradient descent Primal L1-SVM , where C=15 , penalty constant

Online Training from 2007 to 2012 and overall 6 years with NORMA							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.8131	0.8333	0.7423	0.7125	0.8238	0.8093	0.8093
Precision	0.7909	0.8303	0.7423	0.7125	0.8202	0.8125	0.8125
Specificity	0.9993	0.9986	0.9988	0.9993	0.9988	0.9986	0.9986
Gmean	0.9014	0.9123	0.8611	0.8438	0.9071	0.8990	0.8990
F1-Score/Measure	0.8018	0.8318	0.7423	0.7125	0.8220	0.8109	0.8109

Table 5.16: Online wind speed prediction at interval 15 minutes with NORMA

Online Training from 2007 to 2012 and overall 6 years with proposed algorithm							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.7009	0.7899	0.6748	0.6250	0.7974	0.7626	0.7652
Precision	0.2941	0.5767	0.2619	0.1984	0.5764	0.7000	0.6913
Specificity	0.9948	0.9954	0.9911	0.9942	0.9962	0.9976	0.9975
Gmean	0.8351	0.8867	0.8178	0.7883	0.8912	0.8722	0.8737
F1-Score/Measure	0.4144	0.6667	0.3774	0.3012	0.6691	0.7300	0.7264

Table 5.17: Online wind speed prediction at interval 15 minutes with proposed algorithm

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6111	0.5741	0.3913	0.2500	0.6552	0.4146	0.4913
Precision	0.6471	0.5962	0.3214	0.1667	0.6129	0.4474	0.5346
Specificity	0.9993	0.9976	0.9978	0.9989	0.9986	0.9976	0.9986
Gmean	0.6286	0.7568	0.6249	0.4997	0.8089	0.6431	0.7005
F1-Score/Measure	0.6000	0.5849	0.3529	0.2000	0.6333	0.4304	0.5120

Table 5.18: Online wind speed prediction at interval of 1 hour with stochastic gradient descent Primal L1-SVM, where C=15, penalty parameter

Online Training from 2007 to 2012 and overall 6 years with NORMA							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6667	0.6111	0.3913	0.2500	0.6552	0.4634	0.5434
Precision	0.6667	0.6111	0.3913	0.2857	0.6333	0.4634	0.5434
Specificity	0.9993	0.9976	0.9984	0.9994	0.9987	0.9975	0.9985
Gmean	0.8162	0.7808	0.6250	0.4999	0.8089	0.6799	0.7366
F1-Score/Measure	0.6667	0.6111	0.3913	0.2667	0.6441	0.4634	0.5434

Table 5.19: Online wind speed prediction at interval of 1 hour with NORMA

Online Training from 2007 to 2012 and overall 6 years with proposed algorithm							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.5556	0.2844	0.7725	0.6904	0.8340	0.8161	0.5202
Precision	0.0385	0.5741	0.5790	0.4444	0.6636	0.6843	0.0635
Specificity	0.9714	0.9910	0.9977	0.9984	0.9979	0.9979	0.9747
Gmean	0.7346	0.7543	0.8779	0.8302	0.9123	0.9025	0.7121
F1-Score/Measure	0.0719	0.3804	0.6619	0.5408	0.7391	0.7444	0.1132

Table 5.20: Online wind speed prediction at interval of 1 hour with proposed algorithm

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6111	0.4815	0.3043	0.1250	0.5517	0.3902	0.3526
Precision	0.5789	0.5909	0.2917	0.0909	0.5517	0.3902	0.5446
Specificity	0.9991	0.9979	0.9981	0.9989	0.9985	0.9971	0.9990
Gmean	0.7814	0.6932	0.5511	0.3534	0.7422	0.6238	0.5935
F1-Score/Measure	0.5946	0.5306	0.2979	0.1053	0.5517	0.3902	0.4281

Table 5.21: Online wind speed prediction at interval of 6 hours with stochastic gradient decent Primal L1-SVM, where C=5.

Online Training from 2007 to 2012 and overall 6 years with NORMA							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6667	0.6296	0.3478	0.2500	0.6207	0.4878	0.5434
Precision	0.6667	0.6296	0.3636	0.2500	0.6207	0.4878	0.5465
Specificity	0.9993	0.9977	0.9984	0.9993	0.9987	0.9976	0.9985
Gmean	0.8162	0.7926	0.5893	0.4998	0.7873	0.6976	0.7366
F1-Score/Measure	0.6667	0.6296	0.3556	0.2500	0.6207	0.4878	0.5449

Table 5.22: Online wind speed prediction at interval of 6 hours with NORMA

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6111	0.4074	0.1739	0	0.5862	0.2927	0.2832
Precision	0.5500	0.5366	0.2500	0	0.6071	0.4000	0.5698
Specificity	0.9990	0.9978	0.9986	0.9990	0.9987	0.9979	0.9993
Gmean	0.7813	0.6376	0.4167	0	0.7652	0.5404	0.5320
F1-Score/Measure	0.5789	0.4632	0.2051	0	0.5965	0.3380	0.3784

Table 5.23: Online wind speed prediction at interval of 12 hours with stochastic gradient descent Primal L1-SVM, where C=5

Online Training from 2007 to 2012 and overall 6 years with NORMA							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6667	0.6296	0.3913	0.2500	0.6897	0.4878	0.5549
Precision	0.6667	0.6296	0.3913	0.2500	0.6667	0.4878	0.5549
Specificity	0.9993	0.9977	0.9984	0.9993	0.9989	0.9976	0.9985
Gmean	0.8162	0.7926	0.6250	0.4998	0.8300	0.6976	0.7444
F1-Score/Measure	0.6667	0.6296	0.3913	0.2500	0.6780	0.4878	0.5549

Table 5.24: Online wind speed prediction at interval of 12 hours with NORMA

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.2778	0.3704	0.0435	0.2500	0.4138	0.3171	0.1618
Precision	0.4167	0.5263	0.1111	0.1429	0.5217	0.4483	0.4667
Specificity	0.9992	0.9979	0.9991	0.9986	0.9987	0.9982	0.9994
Gmean	0.5268	0.6080	0.2084	0.4997	0.6429	0.5626	0.4022
F1-Score/Measure	0.3333	0.4348	0.0625	0.1818	0.4615	0.3714	0.2403

Table 5.25: Online wind speed prediction at interval of 24 hours with stochastic gradient descent, where C=5.

Online Training from 2007 to 2012 and overall 6 years with SGD Primal L1-SVM							
Year →	2007	2008	2009	2010	2011	2012	All 6 years
Recall/Sensitivity	0.6667	0.6296	0.3913	0.2500	0.6552	0.4762	0.5549
Precision	0.6667	0.6071	0.3913	0.2500	0.6552	0.4878	0.5455
Specificity	0.9993	0.9975	0.9984	0.9993	0.9989	0.9975	0.9985
Gmean	0.8162	0.7925	0.6250	0.4998	0.8090	0.6975	0.7444
F1-Score/Measure	0.6667	0.6182	0.3913	0.2500	0.6552	0.4819	0.5501

Table 5.26: Online wind speed prediction at interval of 24 hours with NORMA

Chapter 6

Conclusion

6.1 Summary

In this effort, we tried to do two types of experiments. First is for labelling huge unlabelled dataset. Second is try to predict violent wind speed ahead of few minutes to one day(24 hours). For very volatile wind speed and weather, generic static classifiers are not found. Generic classifiers like MLP, SVM, KNN may work for dataset where classes have nearly equal number of instances for training. But those classifiers fail if we have imbalanced dataset. So instead of providing right away any static classifier, in this report we tried to explore how good quality dataset for training can be produced. And we also considered the online prediction process too. In which training and test process are sequential. Online classifier is dynamic in nature such that it learns from previous instances and try to predict subsequent events. It is more reasonable in term of shifting the classifier hyperplane and computation overhead is less.

First kind of experiments will provide us more reliable training set than other sampling methods. Although in our first kind of experiments, minority

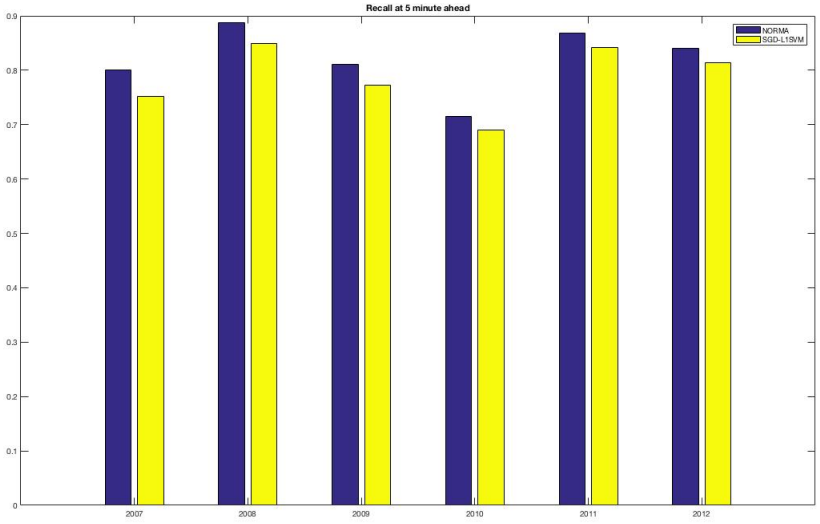


Figure 6.1: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 5 minutes ahead

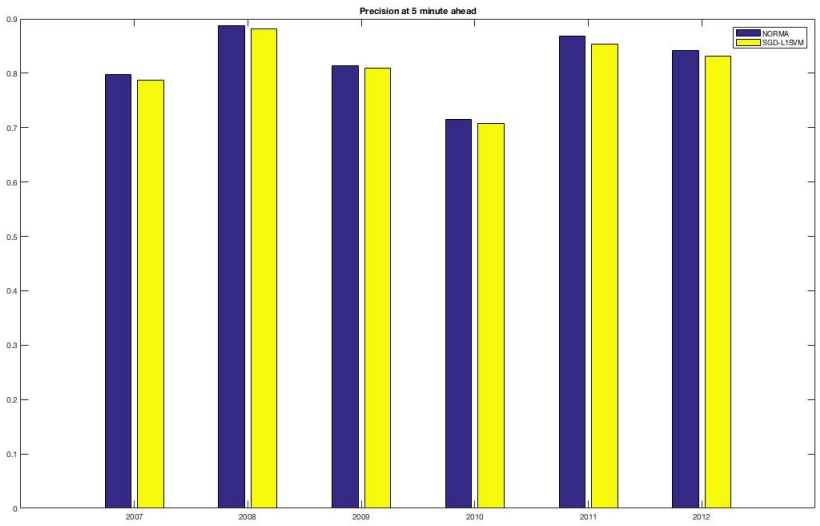


Figure 6.2: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 5 minutes ahead

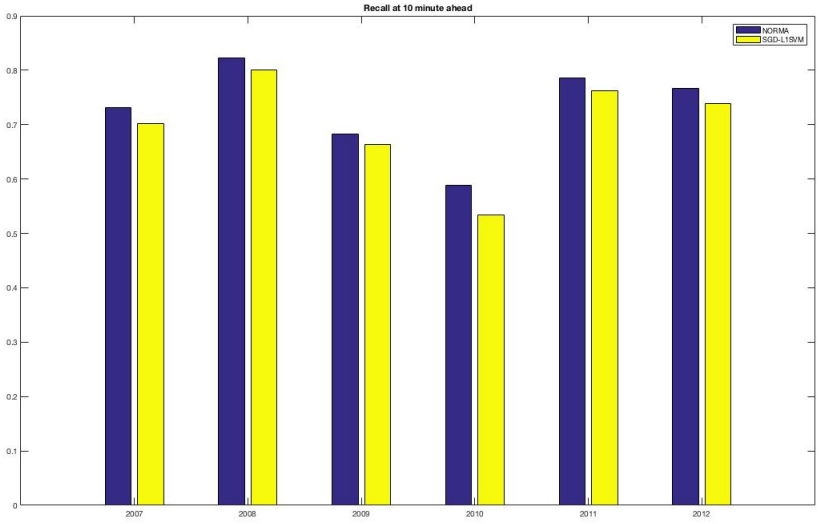


Figure 6.3: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 10 minutes ahead

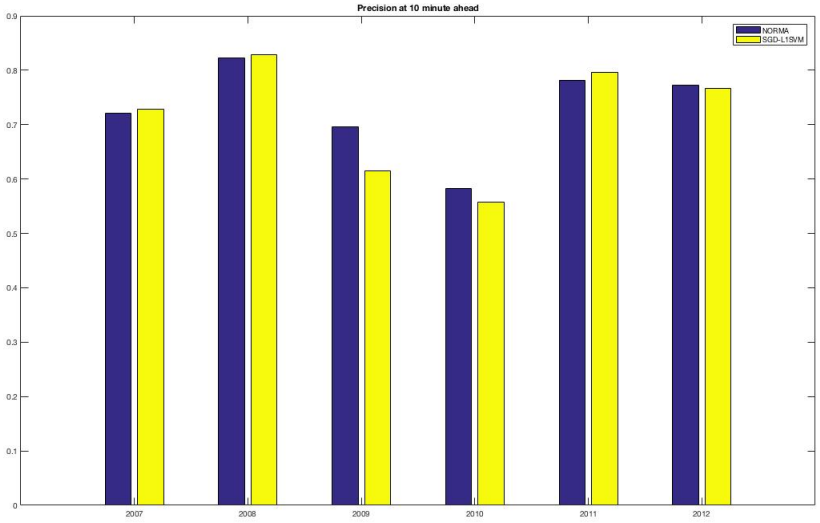


Figure 6.4: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 10 minutes ahead

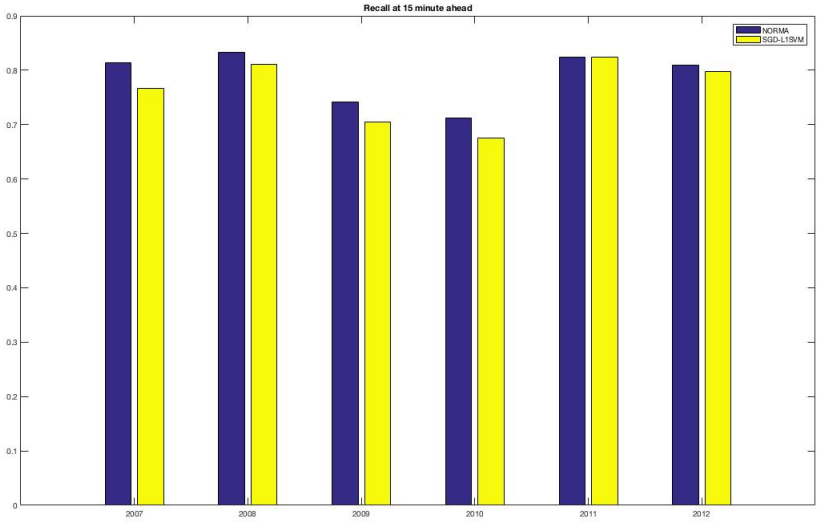


Figure 6.5: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 15 minutes ahead

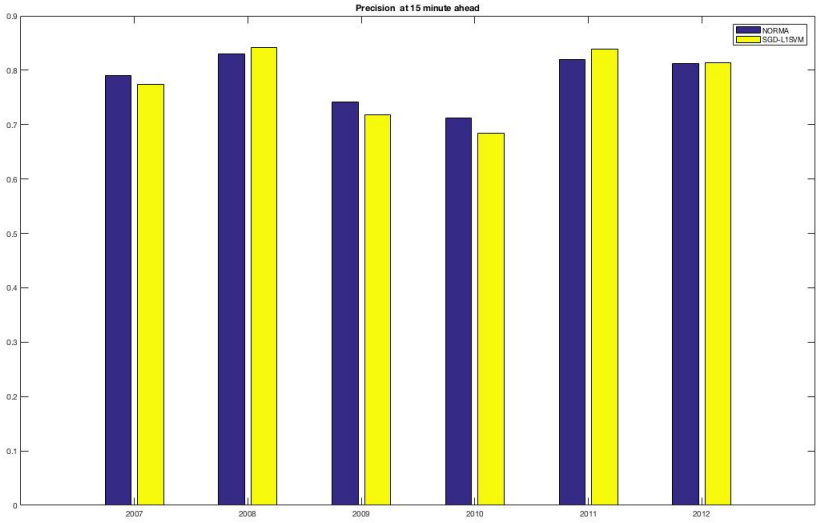


Figure 6.6: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 15 minutes ahead

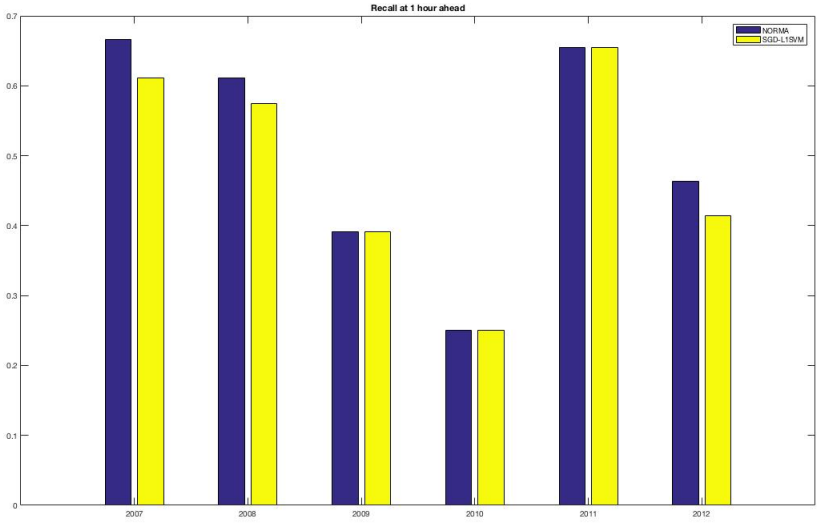


Figure 6.7: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 1 hour ahead

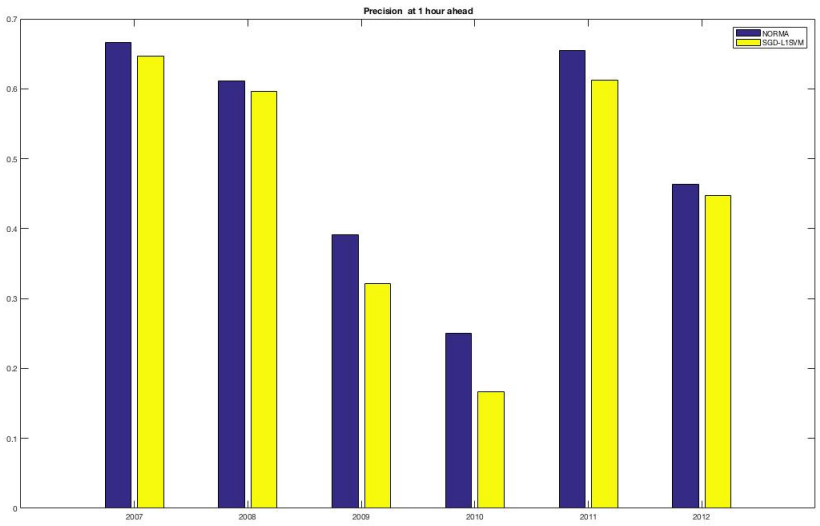


Figure 6.8: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 1 hour ahead

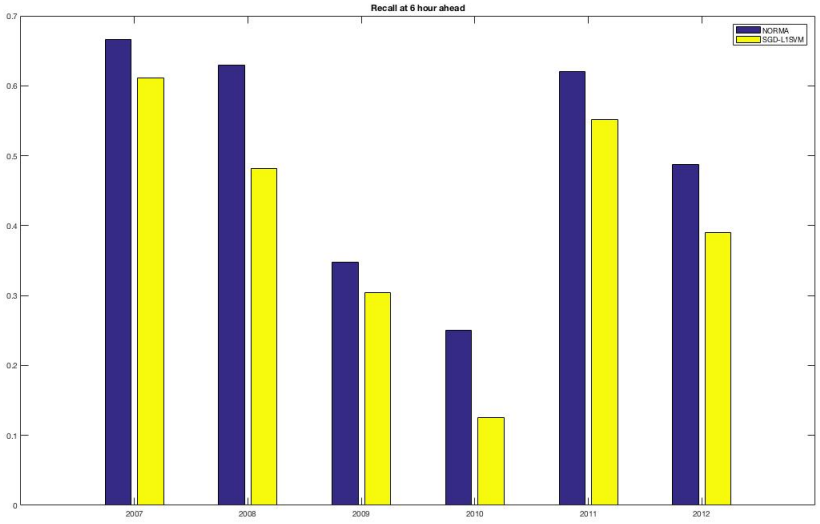


Figure 6.9: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 6 hour ahead

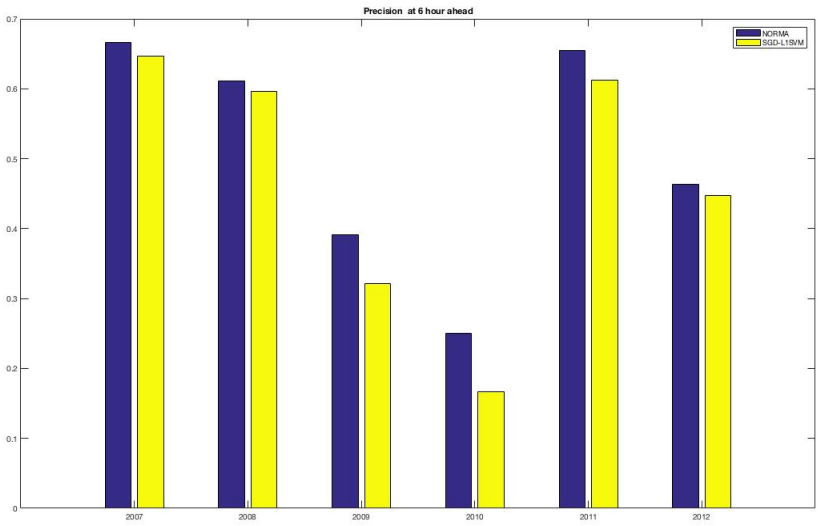


Figure 6.10: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 6 hour ahead

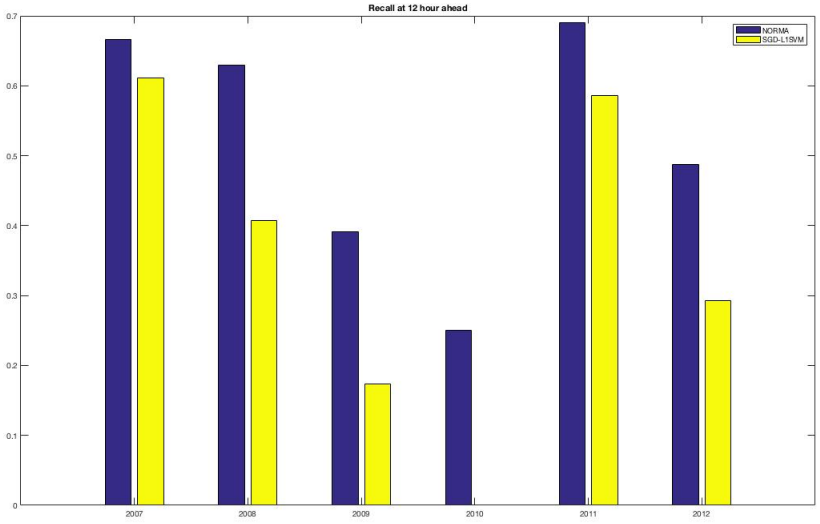


Figure 6.11: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 12 hour ahead

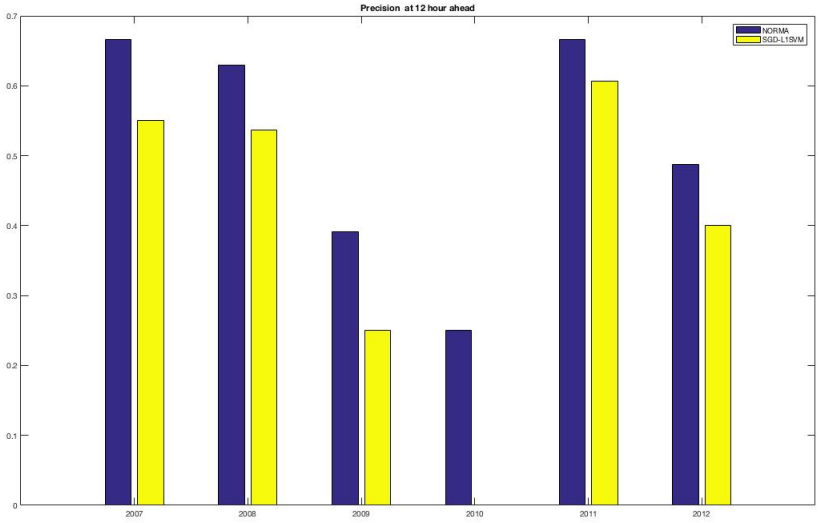


Figure 6.12: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 12 hour ahead

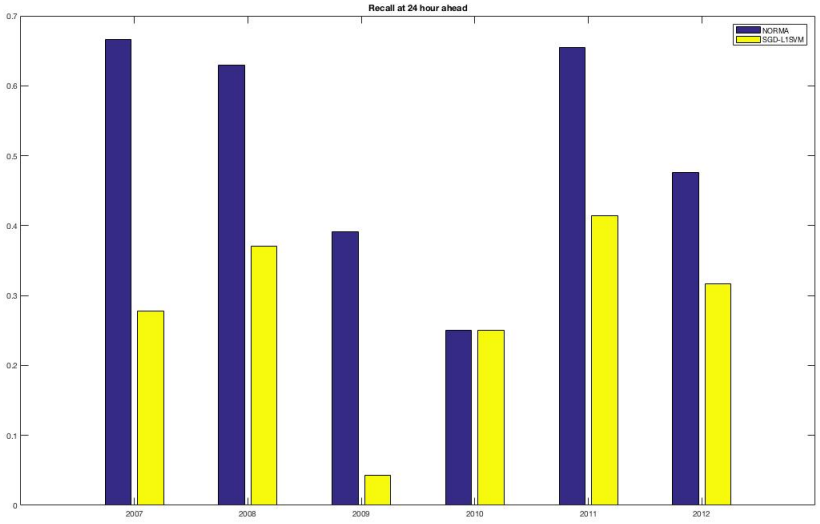


Figure 6.13: Comparison of minority class recall between NORMA and SGD Primal L1-SVM when predicting 24 hour ahead

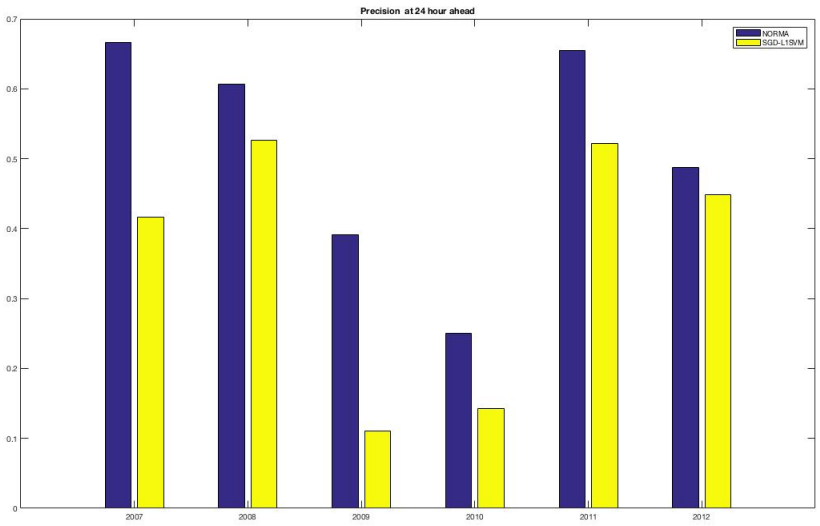


Figure 6.14: Comparison of minority class precision between NORMA and SGD Primal L1-SVM when predicting 24 hour ahead

class instances are very few, which is less than 1% that we can afford to label manually, but huge portion of operational wind speed labelling is unnecessary, it was found. Using active learning process, we can omit less important majority class instance labelling. It will save lots of time and effort for training data collection and may produce relevant instances for training. We did active learning with help of RVFL neural network architecture and SVM. We found that SVM with active learning performing well than RVFL neural network. If we train with whole dataset and measure recall and precision of training, generic SVM perform well as it is using all the necessary support vector as required and all labels. But we can easily find out in table 5.2 and table 5.3, using only 16% of labelled dataset we can acquire reasonable accuracy for majority class and also acceptable recall for minority class.

Second type of experiments, we implemented stochastic gradient descent primal L1-SVM and NORMA. Both are in online training mode. Both the classifier try to predict next value of wind speed for prescribed time interval. And it then learns from error reckoned from prediction and actual value difference. After Learning it, predicts next value. From the table 5.10, 5.13and 5.16, we found that it outperformed SGD primal L1-SVM, table 5.9, 5.12 and 5.15. But SGD primal L1-SVM is much more faster than NORMA. SGD primal L1-SVM requies few additions and condition checking. But NORMA produces better recall and reliability in cost of multiple kernel value calculations and storage of few past coefficients and centroids. It is also noticed that as gap between present time and predicted value time increasing, recall and precision are rapidly diminishing. Upto 1 hour of predictions are acceptable but beyond that it is not reliable.

6.2 Future Work

In the development process we could not find any static classifier for wind speed prediction where we have imbalanced dataset, our next arena of exploration will be whether it is possible to produce static classifier for this kind of weather prediction process where dataset is imbalanced.

We also consider own developed algorithm for prediction in our online learning process, it is using KNN and SVM like concept for identifying boundary instances from both classes of instances.Although that algorithm is providing good recall but precision is very poor. We like to convey in that direction to reduce the computation cost and increase precision using KNN and SVM like properties.

Bibliography

[1] Corinna Cortes, Vladimir Vapnik. *Support-Vector Networks*. Journal Machine Learning Volume 20 Issue 3, Sept. 1995 Pages 273 - 297

[2] Edgar Osuna, Robert Freund, Federico Girosi, *Support Vector Machines: Training and Applications*, Technical Report Support Vector Machines: Training and Applications

[3] Nitesh V. Chawla,Kevin W. Bowyer,Lawrence O. Hall, W. Philip. Kegelmeyer *SMOTE: synthetic minority over-sampling technique*. Journal of Artificial Intelligence Research Volume 16 Issue 1, January 2002 Pages 321-357.

[4] Guoqi Li, Changyun Wen, Zheng Guo Li. *Model-Based Online Learning With Kernels* IEEE Transactions on Neural Networks and Learning Systems (Volume: 24, Issue: 3, March 2013).

[5] Jyrki Kivinen , Alexander J. Smola , Robert C. Williamson *Online Learning with Kernels*, IEEE Trans. Signal Process., vol. 100, no. 10, pp. 21652176, Oct. 2004

[6] Y. Ren et al., *Random vector functional link network for short-term electricity load demand forecasting*, Information Sciences (2016), http://dx.doi.org/10.1016/j.ins.2015.11.039

[7] Constantinos Panagiotakopoulos, Petroula Tsampouka *The Stochastic Gradient Descent for the Primal L1-SVM Optimization Revisited*,ECML/PKDD (3), volume 8190 of Lecture Notes in Computer Science, page 65-80. Springer, (2013).

[8] Vapnik, V. *Statistical learning theory*. Wiley, Chichester (1998)

[9] Seyda Ertekin,Jian Huang,Leon Bottou, Lee Giles *Learning on the border: active learning in imbalanced data classification* Proceeding CIKM '07 Proceedings of the sixteenth ACM conference on Conference on information and knowledge management Pages 127-136

[10] Antoine Bordes,Seyda Ertekin,Jason Weston, *Fast Kernel Classifiers with Online and Active Learning* Journal of Machine Learning Research 6 (2005) 15791619

[11] H. S. Seung, M. Opper, H. Sompolinsky *Query by committee*, Proceeding COLT '92 Proceedings of the fifth annual workshop on Computational learning theory Pages 287-294

[12] David A. Cohn, Zoubin Ghahramani, Michael I. Jordan, *Active learning with statistical models* Journal of Artificial Intelligence Research Volume 4 Issue 1, Jnauary 1996 Pages 129-145

[13] Steven C. H. Hoi, Rong Jin, Jianke Zhu, Michael R. Lyu *Batch Mode Active Learning and Its Application to Medical Image Classification* ICML '06 Proceedings of the 23rd international conference on Machine learning Pages 417-424

[14] Rita Chattopadhyay, Zheng Wang, Wei Fan, Ian Davidson, Sethuraman Panchanathan , Jieping Ye *Batch Mode Active Sampling Based*

on Marginal Probability Distribution Matching ACM Transactions on Knowledge Discovery from Data (TKDD) - Special Issue on ACM SIGKDD 2012 TKDD Volume 7 Issue 3, September 2013 Article No. 13

[15] M. A. Ghorbani , R. Khatibi , B. Hosseini , M. Bilgili, *Relative importance of parameters affecting wind speed prediction using artificial neural networks*, Theor Appl Climatol (2013) 114:107114 DOI 10.1007/s00704-012-0821-9

[16] Tarek H. M. El-Fouly, Ehab F. El-Saadany , Magdy M. A. Salama, *One Day Ahead Prediction of Wind Speed and Direction*

[17] Kanna Bhaskar, S. N. Singh, *AWNN-Assisted Wind Power Forecasting Using Feed-Forward Neural Network*, IEEE Transactions on Sustainable Energy (Volume: 3, Issue: 2, April 2012)

[18] P. Louka,, G. Galanisa, , N. Siebert, G. Kariniotakis, P. Katsafados, I. Pytharoulis, , G. Kallos, *Improvements in wind speed forecasts for wind power prediction purposes using Kalman filtering*

[19] M.C. Alexiadis, P.S. Dokopoulos,H.S. Sahsamanoglou, *Wind speed and power forecasting based on spatial correlation models*,IEEE Transactions on Energy Conversion (Volume: 14, Issue: 3, Sep 1999)

[20] I.G. Damousis, M.C. Alexiadis,J.B. Theocharis,P.S. Dokopoulos *A fuzzy model for wind speed prediction and power generation in wind parks using spatial correlation*,IEEE Transactions on Energy Conversion (Volume: 19, Issue: 2, June 2004)

[21] Yan Xu, Zhao Yang Dong, Zhao Xu, Ke Meng, Kit Po Wong, *An Intelligent Dynamic Security Assessment Framework for Power Systems With*

Wind Power,IEEE Transactions on Industrial Informatics (Volume: 8, Issue: 4, Nov. 2012)

[22] Alex J. Smola,Bernhard Schkopf , *Sparse greedy matrix approximation for machine learning*,ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning Pages 911-918

[23] Pedro Domingos, *MetaCost: a general method for making classifiers cost-sensitive*,Proceeding KDD '99 Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining Pages 155-164

[24] Xu-Ying Liu, Jianxin Wu, Zhi-Hua Zhou, *Exploratory under-sampling for class-imbalance learning*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) (Volume: 39, Issue: 2, April 2009)

[25] Greg Schohn ,David Cohn, *Less is More: Active Learning with Support Vector Machines*, Proceeding ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning Pages 839-846

[26] C.-C. Chang, C.-J. Lin, *LIBSVM : a library for support vector machines*,ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011.

[27] F. Rosenblatt *The perceptron: A probabilistic model for information storage and organization in the brain*,Neurocomputing: foundations of research Pages 89-114

[28] Simon Tong, Daphne Koller *Support vector machine active learning with applications to text classification*, The Journal of Machine Learning Research archive Volume 2, 3/1/2002 Pages 45-66

[29] Colin Campbell,Nello Cristianini ,Alex J. Smola, *Query Learning with Large Margin Classifiers*,Proceeding ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning Pages 111-118

[30] Klaus Brinker, *Incorporating Diversity in Active Learning with Support Vector Machines*,In Proceedings of the 20th International Conference on Machine Learning(ICML-2003), Washington DC, 2003.

[31] Yoav Freund, H. Sebastian Seung, Eli Shamir, Naftali Tishby, *Selective Sampling Using the Query by Committee Algorithm*,Journal Machine Learning Volume 28 Issue 2-3, Aug./Sept. 1997 Pages 133 - 168

[32] David D. Lewis, William A. Gale, *A sequential algorithm for training text classifiers*, Proceeding SIGIR '94 Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval Pages 3-12

[33] Dana Angluin, *Queries and Concept Learning*,Journal Machine Learning Volume 2 Issue 4, April 1988 Pages 319 - 342

[34] J. -N. Hwang, J. J. Choi, S. Oh, R. J. Marks, II, *Query-based learning applied to partially trained multilayer perceptrons*, Journal IEEE Transactions on Neural Networks archive Volume 2 Issue 1, January 1991 Page 131-136

[35] Yuhong Guo, *Active Instance Sampling via Matrix Partition*, Proceeding NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems,2010 Pages 802-810

[36] Indre Zliobaite,Albert Bifet, Bernhard Pfahringer , Geoffrey Holmes, *Active Learning With Drifting Streaming Data*,IEEE Transactions on Neural Networks and Learning Systems,Volume: 25, Issue: 1, Jan. 2014