CptS 570 Machine Learning, Fall 2018
Homework #1
Due Date: Thu, Sept 27 (9:10am)

NOTE 1: Please use a word processing software (e.g., Microsoft word or Latex) to write your answers and submit a printed copy to me at the begining of class on Feb 27. The rationale is that it is sometimes hard to read and understand the hand-written answers.

NOTE 2: Please ensure that all the graphs are appropriately labeled (x-axis, y-axis, and each curve). The caption or heading of each graph should be informative and self-contained.

1. **(5 points)** Answer the following questions with a yes or no along with proper justification.
   a. Is the decision boundary of voted perceptron linear?
   b. Is the decision boundary of averaged perceptron linear?

2. **(5 points)** In the class, we saw the Passive-Aggressive (PA) update that tries to achieve a margin equal to *one* after each update. Derive the PA weight update for achieving margin M.

3. **(10 points)** Consider the following setting. You are provided with n training examples: $(x_1, y_1, h_1), (x_2, y_2, h_2), \cdots, (x_n, y_n, h_n)$, where x_i is the input example, y_i is the class label (+1 or -1), and $h_i > 0$ is the importance weight of the example. The teacher gave you some additional information by specifying the importance of each training example.
   a. How will you modify the perceptron algorithm to be able to leverage this extra information? Please justify your answer.
   b. How can you solve this learning problem using the standard perceptron algorithm? Please justify your answer. I'm looking for a reduction based solution.

4. **(10 points)** Consider the following setting. You are provided with n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$, where x_i is the input example, and y_i is the class label (+1 or -1). However, the training data is highly imbalanced (say 90% of the examples are negative and 10% of the examples are positive) and we care more about the accuracy of positive examples.
   a. How will you modify the perceptron algorithm to solve this learning problem? Please justify your answer.
   b. How can you solve this learning problem using the standard perceptron algorithm? Please justify your answer. I'm looking for a reduction based solution.

5. (**10 points**) Suppose we have n_+ positive training examples and n_- negative training examples. Let C_+ be the center of the positive examples and C_- be the center of the negative examples, i.e., $C_+ = \frac{1}{n_+} \sum_{i:\ y_i=+1} x_i$ and $C_- = \frac{1}{n_-} \sum_{i:\ y_i=-1} x_i$. Consider a simple classifier called CLOSE that classifies a test example x by assigning it to the class whose center is closest.

   - Show that the decision boundary of the CLOSE classifier is a linear hyperplane of the form $sign(w \cdot x + b)$. Compute the values of w and b in terms of C_+ and C_-.
   - Recall that the weight vector can be written as a linear combination of all the training examples: $w = \sum_{i=1}^{n_+ + n_-} \alpha_i \cdot y_i \cdot x_i$. Compute the dual weights (α's). How many of the training examples are support vectors?

6. (**5 points**) Suppose we use the following radial basis function (RBF) kernel: $K(x_i, x_j) = exp(-\frac{1}{2}\|x_i - x_j\|^2)$, which has some implicit unknown mapping $\phi(x)$.

   - Prove that the mapping $\phi(x)$ corresponding to RBF kernel has infinite dimensions.
   - Prove that for any two input examples x_i and x_j, the squared Euclidean distance of their corresponding points in the higher-dimensional space defined by ϕ is less than 2, i.e., $\|\phi(x_i) - \phi(x_j)\|^2 \leq 2$.

7. (**5 points**) The decision boundary of a SVM with a kernel function (via implicit feature mapping $\phi(.)$) is defined as follows:
   $w \cdot \phi(x) + b = \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b = f(x; \alpha, b)$
   , where w and b are parameters of the decision boundary in the feature space *phi* defined by the kernel function K, SV is the set of support vectors, and α_i is the dual weight of the i^{th} support vector.

   Let us assume that we use the radial basis function (RBF) kernel $K(x_i, x_j) = exp(-\frac{1}{2}\|x_i - x_j\|^2)$; also assume that the training examples are linearly separable in the feature space ϕ and SVM finds a decision boundary that perfectly separates the training examples.

   If we choose a testing example x_{far} that is far away from any training instance x_i (distance here is measured in the original feature space $\Re^d$). Prove that $f(x_{far}; \alpha, b) \approx b$.

8. (**5 points**) The function $K(x_i, x_j) = -\langle x_i, x_j \rangle$ is a valid kernel. Prove or Disprove it.

9. (**5 points**) You are provided with n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n,)$, where x_i is the input example, y_i is the class label (+1 or -1). The teacher gave you some additional information by specifying the costs for different mistakes C_+ and C_-, where C_+ and C_- stand for the cost of misclassifying a positive and negative example respectively.
   a. How will you modify the Soft-margin SVM formulation to be able to leverage this extra information? Please justify your answer.

10. **(5 points)** Consider the following setting. You are provided with n training examples: $(x_1, y_1, h_1), (x_2, y_2, h_2), \cdots, (x_n, y_n, h_n)$, where x_i is the input example, y_i is the class label (+1 or -1), and $h_i > 0$ is the importance weight of the example. The teacher gave you some additional information by specifying the importance of each training example.
    a. How will you modify the Soft-margin SVM formulation to be able to leverage this extra information? Please justify your answer.
    b. How can you solve this learning problem using the standard SVM training algorithm? Please justify your answer.

11. **(15 points)** You are provided with a set of n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n,)$, where x_i is the input example, y_i is the class label (+1 or -1). Suppose n is very large (say in the order of millions). In this case, standard SVM training algorithms will not scale due to large training set.
    Tom wants to devise a solution based on "Coarse-to-Fine" framework of problem solving. The basic idea is to cluster the training data; train a SVM classifier based on the clusters (coarse problem); refine the clusters as needed (fine problem); perform training on the finer problem; and repeat until convergence. Suppose we start with k_+ positive clusters and k_- negative clusters to begin with (a cluster is defined as a set of examples). Please specify the *mathematical formulation* (define all the variables used in your formulation) and concrete algorithm for each of the following steps to instantiate this idea:
    a) How to define the SVM training formulation for a given level of coarseness: a set of k_+ positive clusters and a set of k_- negative clusters?

b) How to refine the clusters based on the resulting SVM classifier?
c) What is the stopping criteria?
Optional question: For what kind of problems will this solution fail?

12. **(20 points)** You are provided with a set of n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n,)$, where x_i is the input example, y_i is the class label (+1 or -1). Suppose n is very large (say in the order of millions). In this case, online kernelized Perceptron algorithms will not scale if the number of allowed support vectors are *unbounded*.
a) Suppose you have trained using kernelized Perceptron algorithm (without any bounds on support vectors) and got a set of support vectors SV. Tom wants to use this classifier for real-time prediction and cannot afford more than B kernel evaluations for each classification decision. Please give an algorithm to select B support vectors from SV. You need to motivate your design choices in order to convince Tom to use your solution.
b) Tom wants to train using kernelized Perceptron algorithm, but wants to use at most B support vectors during the training process. Please modify the standard kernelized Perceptron training algorithm (from class slides) for this new setting. You need to motivate your design choices in order to convince Tom to use your solution.

13. **(50 points)** Programming and empirical analysis question.
Implement a binary classifier with both perceptron and passive-aggressive (PA) weight update as shown below.

Algorithm 1 Online Binary-Classifier Learning Algorithm

Input: $\mathcal{D}$ = Training examples, T = maximum number of training iterations
Output: w, the final weight vector

Initialize the weights $w = 0$
for each training iteration $itr \in \{1, 2, \cdots, T\}$ **do**
  for each training example $(x_t, y_t) \in \mathcal{D}$ **do**
    $\hat{y}_t = sign(w \cdot x_t)$ *// predict using the current weights*
    if mistake **then**
      $w = w + \tau \cdot y_t \cdot x_t$ *// update the weights*
    end if
  end for
end for
return final weight vector w

For standard perceptron, you will use $\tau = 1$, and for Passive-Aggressive (PA) algorithm, you will compute the learning rate τ as follows.

$$\tau = \frac{1 - y_t \cdot (w \cdot x_t)}{\|x_t\|^2} \tag{1}$$

You are provided with the income data. See `https://archive.ics.uci.edu/ml/datasets/census+income` for a description of different features. You are provided with a training set, development (held out) set, and testing set. The format of the file is as follows. Each line is one classification example: sequence of feature values with the class label at end ($\leq$50K or $>$50K) in comma separated values (CSV) format.

Perceptron, Passive-Aggressive, and Averaged Classifier

- Convert all features into binary format (0 or 1). Describe your conversion. How many features do you have (i.e., the dimensionality)?
- Implement Perceptron and Passive-Aggressive based classifier from these binary features.
- Compute the online learning curve for both Perceptron and PA algorithm by plotting the number of training iterations (1 to 5) on the x-axis and the number of mistakes on the y-axis. Compare the two curves and list your observations.
- Compute the accuracy of both Perceptron and PA algorithm on the training data, development data, and testing data for each training iteration (1 to 5). So you will have three accuracy curves for Perceptron and another three accuracy curves for PA algorithm. Compare the six curves and list your observations.
- Implement the averaged perceptron algorithm in both the naive way (one I wrote on the white board) and the smart way (Algorithm 7 from Hal's book chapter). Measure the training time to perform 5 iterations for both implementations. Compute the accuracies on training data, development data, and testing data with averaged classifier and compare with those obtained using standard perceptron. List your observations.
- Compute the general learning curve (vary the number of training examples starting from 5000 in the increments of 5000) for 5 training iterations. Plot the number of training examples on x-axis and the accuracy (development and testing) on the y-axis. List your observations from these curves.
- Optional things you can try. (a) Shuffling the training examples during each iteration. What did you observe?; and (b) Variable learning rate with a schedule of your choice. What did you observe?

14. **(50 points) Support Vector Machines (SVM) Classifier and Kernels**

- (**30 points**) Empirical analysis question. You can use a publicly available SVM classifier implementation (e.g., LibSVM or scikit-learn) for SVM related experiments. LibSVM (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/` and scikit-learn (`http://scikit-learn.org/stable/modules/svm.html`).

  (a) Using a linear kernel, train the SVM on the training data for different values of C parameter: $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}$. Compute the training accuracy, validation accuracy, and testing accuracy for the SVM obtained with different values of the C parameter. Plot the training accuracy, validation accuracy, and testing accuracy as a function of C (C value on x-axis and Accuracy on y-axis) – one curve each for training, validation, and testing data. Also, plot the number of support vectors (see the training log at the end of SVM training for LibSVM or use the scikit-learn API if applicable) as a function of C. List your observations.

  (b) Select the best value of hyper-parameter C based on the accuracy on validation set and train a linear SVM on the *combined* set of training and validation examples. Compute the testing accuracy and the corresponding confusion matrix: a 2×2 matrix.

  (c) Repeat the experiment (a) with the best C value from (a) with polynomial kernel of degree 2, 3, and 4. Compare the training, validation, testing accuracies, and the number of support vectors for differnt kernels (linear, polynomial kernel of degree 2, polynomial kernel of degree 3, and polynomial kernel of degree 4). List your observations.

- (**20 points**) Programming question. You will implement the standard kernelized Perceptron training algorithm (from class slides) for binary classification.

  (a) Train the binary classifier for 5 iterations with polynomial kernel (pick the best degree out of 2, 3, and 4 from the above experiment). Plot the number of mistakes as a function of training iterations. Compute the training, development, and testing accuracy at the end of 5 iterations.

Instructions for Code Submission and Output Format.

Please follow the below instructions. It will help us in grading your programming part of the homework. We will provide a dropbox folder link for code submission.

- Supported programming languages: Python, Java, C++
- Store all the relevant files in a folder and submit the corresponding zipfile named after your student-id, e.g., 114513209.zip
- This folder should have a script file named

  `run_code.sh`

  Executing this script should do all the necessary steps required for executing the code including compiling, linking, and execution
- Assume relative file paths in your code. Some examples:

  ‘‘./filename.txt’’ or ‘‘../hw1/filename.txt’’
- The output of your program should be dumped in a file named “output.txt”
- Make sure the output.txt file is dumped when you execute the script

  `run_code.sh`
- Zip the entire folder and submit it as

  `<student_id>.zip`

Grading Rubric

Each question in the students work will be assigned a letter grade of either A,B,C,D, or F by the Instructor and TAs. This five-point (discrete) scale is described as follows:

- **A) Exemplary (=100%)**.
  Solution presented solves the problem stated correctly and meets all requirements of the problem.
  Solution is clearly presented.
  Assumptions made are reasonable and are explicitly stated in the solution.
  Solution represents an elegant and effective way to solve the problem and is not overly complicated than is necessary.

- **B) Capable (=75%).**
  Solution is mostly correct, satisfying most of the above criteria under the exemplary category, but contains some minor pitfalls, errors/flaws or limitations.

- **C) Needs Improvement (=50%).**
  Solution demonstrates a viable approach toward solving the problem but contains some major pitfalls, errors/flaws or limitations.

- **D) Unsatisfactory (=25%)**
  Critical elements of the solution are missing or significantly flawed.
  Solution does not demonstrate sufficient understanding of the problem and/or any reasonable directions to solve the problem.

- **F) Not attempted (=0%)**
  No solution provided.

The points on a given homework question will be equal to the percentage assigned (given by the letter grades shown above) multiplied by the maximum number of possible points worth for that question. For example, if a question is worth 6 points and the answer is awarded a B grade, then that implies 4.5 points out of 6.

Assignment 1

Sheryl Mathew (11627236)

October 2, 2018

1. Decision boundary of Voted Perceptron and Average Perceptron

a. The decision boundary of voted perceptron is non-linear. During training, we get a set of weights based on the feature vectors of the dataset provided. For every example we get a weight vector. The decision boundary for that example will be perpendicular to the weight vector. In voted perceptron we have different weight vectors for each classifier. Therefore different decision boundaries for each weight vector. Therefore the decision boundary of the voted perceptron will be the intersection of all the decision boundaries which will not be linear.

b. The decision boundary of average perceptron is linear. During training, we get a set of weight based on the feature vectors of the dataset provided. For every example we get a weight vector. The decision boundary for that example will be perpendicular to the weight vector. In average perceptron we take the average of all the weight vectors from all the classifiers and get only a single weight vector. Therefore the decision boundary of the average perceptron will be linear.

2. Passive Aggressive Weight Update of Margin M

Let w_{t+1} be the solution of the following optimization problem

$W_{t+1} = \min_{w} \frac{1}{2} \|w - w_t\|^2$ s.t $y_t(w \cdot x_t) \geq M$ …… Equation (1)
where M is the margin

Lagrangian of Equation (1)

$L(w, \tau) = \frac{1}{2} \|w - w_t\|^2 + \tau (M - y_t(w \cdot x_t))$ …… Equation (2)

$= \frac{1}{2} \|w - w_t\|^2 + \tau M - \tau y_t(w \cdot x_t))$

$\frac{\partial L}{\partial w} = \|w - w_t\| - \tau y_t(x_t))$

$0 = w - w_t - \tau y_t(x_t))$

$w = w_t + \tau y_t(x_t))$ …… Equation (3)

Substitute Equation (3) in Equation (2)

$L(\tau) = \frac{1}{2} \|(w_t + \tau y_t(x_t)) - w_t\|^2 + \tau (M - y_t(w_t + \tau y_t(x_t)) \cdot x_t))$

$= \frac{1}{2} \|\tau\, y_t\, x_t\|^2 + \tau\,(M - y_t . x_t w_t - \tau\,(y_t\, x_t)^2)$

$= \frac{-1}{2} \|\tau\, y_t\, x_t\|^2 + \tau\,(M - y_t . x_t w_t)$

$= \frac{-\tau^2}{2} \|x_t\|^2 + \tau\,(M - y_t . x_t w_t)$ [Since $y_t^2 = 1$]

Dual optimization problem

$\underbrace{max}_{w} \frac{-\tau^2}{2} \|x_t\|^2 + \tau\,(M - y_t\,(x_t . w_t))$ …… Equation (4)

$\frac{\partial \tau}{\partial w} = \frac{-2\tau}{2} \|x_t\|^2 + M - y_t(x_t . w_t)$

$0 = \tau \|x_t\|^2 + M - y_t\,(x_t . w_t)$

$\tau = \frac{M - y_t\,(x_t . w_t)}{\|x_t\|^2}$ …… Equation (5)

Therefore

$$\tau = \max\{0, \frac{M - y_t\,(x_t . w_t)}{\|x_t\|^2}\}$$

Passive Aggressive Weight Update,

$$w_{t+1} = w_t + \tau\, y_t\, x_t$$
$$= w_t + [\frac{M - y_t\,(x_t . w_t)}{\|x_t\|^2}]\, y_t\, x_t$$

3. Importance of weight in Perceptron

a. Modify Perceptron Algorithm

In Standard Perceptron,

Let Learning Rate =1
If no mistake,
$w_{t+1} = w_t$
$b_{t+1} = b_t$
If mistake occurs,
$w_{t+1} = w_t + y_t . x_t$
$b_{t+1} = b_t + y_t$

In Modified Perceptron,

Let Learning Rate =1

If no mistake,

$w_{t+1} = w_t$

$b_{t+1} = b_t$

If mistake occurs,

$w_{t+1} = w_t + y_t \cdot x_t \cdot h_t$

$b_{t+1} = b_t + y_t \cdot h_t$

We include the importance weight of the example in the weight and bias update of the standard perceptron. This is because if the importance of the example is high and the perceptron makes a mistake on that example then there will be a huge weight update which will increase the weight vector significantly and hence the decision boundary. If the importance of the example is low and the perceptron makes a mistake on that example then the weight update will not increase the weight vector by much and hence the decision boundary will not have a major impact.

b. Solve using Perceptron Algorithm

X	Y	H
1,1	1	10
2,2	-1	3
3,3	1	1
4,4	1	4
5,5	-1	5

We can use Standard Perceptron to solve the above problem in different ways:

- We can do sub-sampling based on the weights ie take positive and negative examples in such a way that they have equal weight importance. In the above dataset for training we will take (1,1) for positive and (2,2), (5,5) for negative examples. So the samples become balanced.
- We can use scaling ie based on importance we increase the number of examples. Since (1,1) has importance 10 then we increase the example (1,1) by 10 times.

Let Learning Rate =1

If no mistake,

$w_{t+1} = w_t$

$b_{t+1} = b_t$

If mistake occurs,

$w_{t+1} = w_t + y_t' \cdot x_t$ where $y_t' = y_t \cdot h_t$

$b_{t+1} = b_t + y_t'$ where $y_t' = y_t \cdot h_t$

4. Imbalanced Data

a. Modify Perceptron Algorithm

In Standard Perceptron,

Let Learning Rate =1

If no mistake,

$w_{t+1} = w_t$

$b_{t+1} = b_t$

If mistake occurs,

$$w_{t+1} = w_t + y_t \cdot x_t$$
$$b_{t+1} = b_t + y_t$$

In Modified Perceptron,

Let Learning Rate =1
If no mistake,

$$w_{t+1} = w_t$$
$$b_{t+1} = b_t$$

If mistake occurs,

$$w_{t+1} = w_t + y_t \cdot x_t \cdot h_t$$
$$b_{t+1} = b_t + y_t \cdot h_t$$

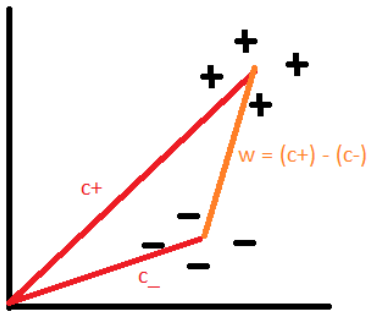
The data that we have is imbalanced ie 90% negative or 10% positive. Inorder to include this mismatch in the data we can add another parameter h which will account for the mismatch. Since we have less positive examples we need to give a weight like 10 to them and for negative examples give a lower weight 1. So that the data becomes balanced.

b. Solve using Perceptron Algorithm

We can use Standard Perceptron to solve the above problem in different ways:

- We can do sub-sampling ie we take only 10% of the negative examples. Therefore the perceptron will train on a balanced data which makes it more efficient.
- We can use scaling ie increase the number of the given positive examples itself so that the number of positive and negative examples become equal.

5. CLOSE Classifier



a. C_+ is the weight vector of the positive cluster and C_- is the weight vector of the negative cluster**.** By vector formula, the distance between both the vectors C_+ and C_- will give the weight vector.

Therefore,

$w = C_+ - C_-$ …….. Equation (1)

sign(w . x + b) = 0

w . x + b = 0 ……... Equation (2)

Substitute Equation (1) in Equation (2)

$(C_+ - C_-)\, x + b = 0$ ……... Equation (3)

Consider a x value which lies between the 2 clusters, ie the mid-point of the weight vector

$x = \frac{C_+ + C_-}{2}$ ……... Equation (4)

Substitute Equation (4) in Equation (3)

$((C_+ + C_-)\ (\frac{C_+ + C_-}{2}))+ b = 0$

$\frac{1}{2}\ (\|C_+\|^2 - \|C_-\|^2) + b = 0$ [Since $\|X\|^2 = X^T X$]

$b = \frac{-1}{2}\ (\|C_+\|^2 - \|C_-\|^2)$

Therefore,

$w = C_+ - C_-$
$b = \frac{-1}{2}\ (\|C_+\|^2 - \|C_-\|^2)$

We have only one weight vector for both the clusters of positive and negative examples. Therefore the decision boundary of the Close Classifier will be linear. Hence it is a linear classifier as the decision boundary separates the clusters of positive and negative examples into 2 separate halves by a line.

b. Compute α

$w = C_+ - C_-$ ……... Equation (1)

$C_+ = \frac{1}{n_+} \sum_{i,y=1}^{n} x_i$ ……... Equation (2)

$C_- = \frac{1}{n_-} \sum_{i,y=-1}^{n} x_i$ ……... Equation (3)

Substitute Equation (2) and Equation (3) in Equation (1)

$w = C_+ - C_-$

$= \frac{1}{n_+} \sum_{i,y=1}^{n} x_i - \frac{1}{n_-} \sum_{i,y=-1}^{n} x_i$

$= \frac{1}{n_+ + n_-} \sum_{i,}^{n} x_i y_i$ [Where y = 1 for positive examples and y = -1 for negative example]

$= \sum_{i,}^{n} \frac{1}{n_+ + n_-} x_i y_i$ ……... Equation (4)

We know that,

$w = \sum_{i,}^{n} \alpha_i x_i y_i$ ……... Equation (5)

Compare Equation (4) and Equation (5)

$\alpha_i = \frac{1}{n_+ + n_-}$

Number of support vectors is the fraction of the total examples. Ie

Number of support vectors = $\frac{1}{n}$ where n is the total number of examples

6. RBF Kernel

a. RBF Kernel has infinite dimensions

$K(x_i, x_j) = \exp[-\frac{1}{2} \|x_i - x_j\|^2]$

$= \exp[-\frac{1}{2} < x_i - x_j,\ x_i - x_j >]$ [Since $(a-b)^2 = (a-b)(a-b)$]

$= \exp[-\frac{1}{2} (x_i < x_i - x_j > - x_j < x_i - x_j >)]$

$= \exp[-\frac{1}{2} (< x_i, x_i > - < x_i, x_j > - < x_j, x_i > + < x_j, x_j >)]$

$= \exp[-\frac{1}{2} (< x_i, x_i > - < x_i, x_j > - < x_j, x_i > + < x_j, x_j >)]$

$= \exp[-\frac{1}{2} (\|x_i\|^2 - 2 < x_i, x_j > + \|x_j\|^2)]$ [Since $\|X\|^2 = X^T X$]

$= \exp[-\frac{1}{2} \left(\|x_i\|^2 + \|x_j\|^2\right) - \frac{1}{2} (-2 < x_i, x_j >)]$

$= \exp[-\frac{1}{2} \left(\|x_i\|^2 + \|x_j\|^2\right) + < x_i, x_j >]$

$= \exp\left[-\frac{1}{2} \left(\|x_i\|^2 + \|x_j\|^2\right)\right] \times \exp[< x_i, x_j >]$ [Since exp(a+b) = exp(a) x exp(b)]

= C exp[$< x_i, x_j >$] [Since exp $\left[-\frac{1}{2}\left(\|x_i\|^2 + \|x_j\|^2\right)\right]$ = Constant C]

By Taylor Series,

= C $\sum_{n=0}^{\infty} \frac{x_i^n x_j^n}{n!}$ [Since exp[x] = $\sum_{n=0}^{\infty} \frac{x^n}{n!}$]

= C $\sum_{n=0}^{\infty} \frac{(x_i x_j)n}{n!}$

= C $\sum_{n=0}^{\infty} \frac{K(x_i,\ x_j)}{n!}$ [Since polynomial kernel, K(x, x') = (x, x') ^{n}]

Therefore we find the value for RBF Kernel by taking an infinite series over a polynomial kernel.

b. $\|\emptyset(x_i) - \emptyset(x_j)\|^2 \leq 2$

$\|\emptyset(x_i) - \emptyset(x_j)\|^2$

= $\|\emptyset(x_i)\|^2 + \|\emptyset(x_j)\|^2 - 2 < \emptyset(x_i)\emptyset(x_j) >$ [Since $(a - b)^2 = a^2 - 2ab + b^2$]

= $< \emptyset(x_i)\emptyset(x_i) > + < \emptyset(x_j)\emptyset(x_j) > - 2 < \emptyset(x_i)\emptyset(x_j) >$ [Since $\|X\|^2 = X^T X$]

= $K(x_i, x_i) + K(x_j, x_j) - 2\ K(x_i, x_j)$ [Since $< \emptyset(x_i)\emptyset(x_j) > = K(x_i, x_j)$]

= exp[$-\frac{1}{2}\|x_i - x_i\|^2$] + exp[$-\frac{1}{2}\|x_j - x_j\|^2$] − 2 exp[$-\frac{1}{2}\|x_i - x_j\|^2$]

[Since $K(x_i, x_j)$ =exp[$-\frac{1}{2}\|x_i - x_j\|^2$]

= exp[$-\frac{1}{2}$ (0)] + exp[$-\frac{1}{2}$ (0)] − 2 exp[$-\frac{1}{2}\|x_i - x_j\|^2$]

= 1 + 1 − exp[$\|x_i - x_j\|^2$] [Since exp(0) =1]

= 2 − exp[$\|x_i - x_j\|^2$]

Therefore

$\|\emptyset(x_i) - \emptyset(x_j)\|^2 \leq 2$

Since is because exp[$\|x_i - x_j\|^2$] will always be separated from 2

7. $f(x_{far}, \alpha, b) \approx b$

$f(x_{far}, \alpha, b)$

$= \sum y_i \alpha_i K(x_{far}, x_i) + b$

$= \sum y_i \alpha_i \exp[-\frac{1}{2} \|x_{far} - x_i\|^2] + b$ [Since $K(x_i, x_j) = \exp[-\frac{1}{2} \|x_i - x_j\|^2]$

$= 0 + b$ [Since $\|x_{far} - x_i\|^2 > 0$ implies $\exp[-\frac{1}{2} \|x_{far} - x_i\|^2] \approx 0$]

$\approx b$

8. **$K(x_i, x_j) = - < x_i, x_j >$ is a valid kernel**

A kernel is valid if it satisfier Mercer's theorem. To satisfy Mercer's therorem we need to satufy the 2 conditions:

- Kernel should be symmetric
- Kernel should be positive definite matrix

Kernel should be symmetric : $K(x_i, x_j) = K(x_j, x_i)$

$K(x_i, x_j) = - <x_i, x_j>$

$= - <x_j, x_i>$

$= K(x_j, x_i)$

Therefore it is symmetric

Kernel should be positive definite matrix : $m^T K m > 0$

$K = <x_i, x_j>$

$= \begin{bmatrix} < x_1, x_1 > & < x_1, x_2 > \\ < x_2, x_1 > & < x_2, x_2 > \end{bmatrix}$

$m^T K m = [n \ \ p] \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} n \\ p \end{bmatrix}$

$= [(n.a + n.c) \ (n.b + n.d)] \begin{bmatrix} n \\ p \end{bmatrix}$

$= [(n.a + n.c) \ n + (n.b + n.d) \ p]$

> 0

But $- <x_j, x_i> = - \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$= \begin{bmatrix} -a & -b \\ -c & -d \end{bmatrix}$

$m^T K m = [-(n.a + n.c) \ n - (n.b + n.d) \ p]$

= - [(n.a + n.c) n + (n.b + n.d) p]

< 0

Therefore it is not a positive definite matrix

Hence $K(x_i , x_j) = - \langle x_i , x_j \rangle$ is not a valid kernel

9. SVM leverage

In Soft Margin SVM maximization,

$$\min_{w,b} \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{N} \varepsilon_i$$

such that

$$y_i \, (w \, . x_i + b) \geq 1 - \varepsilon_i \qquad \text{i = 1……..N}$$

$$\varepsilon_i \geq 0 \qquad \text{i = 1……..N}$$

In Modified Soft Margin SVM maximization,

$$\min_{w,b} \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{N} h_i \, \varepsilon_i$$

such that

$$y_i \, (w \, . x_i + b) \geq 1 - \varepsilon_i \qquad \text{i = 1……..N}$$

$$\varepsilon_i \geq 0 \qquad \text{i = 1……..N}$$

$$h_i = \begin{cases} C_+ \, , y_i = +1 \\ C_- \, , y_i = -1 \end{cases} \qquad \text{i = 1……..N}$$

We have added a variable h_i to the Empirical error that is along with the slack variable ε_i which tries to reduce the number of errors. Therefore we add the cost of the mistake $C_+ \; or \; C_-$ to ε_i so that the penalty is increased and hence we improve the margin while reducing the errors.

10. Importance of Weight in SVM

a. Modify SVM

In Soft Margin SVM maximization,

$$\min_{w,b} \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{N} \varepsilon_i$$

such that

$$y_i \, (w \, . x_i + b) \geq 1 - \varepsilon_i \qquad \text{i} = 1 \ldots\ldots\ldots \text{N}$$

$$\varepsilon_i \geq 0 \qquad \text{i} = 1 \ldots\ldots\ldots \text{N}$$

In Modified Soft Margin SVM maximization,

$$\min_{w,b} \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{N} h_i \, \varepsilon_i$$

such that

$$y_i \, (w \, . x_i + b) \geq 1 - \varepsilon_i \qquad \text{i} = 1 \ldots\ldots\ldots \text{N}$$

$$\varepsilon_i \geq 0 \qquad \text{i} = 1 \ldots\ldots\ldots \text{N}$$

We include the importance weight h_i of the example along with the slack variable ε_i in the maximization of the margin. This is because if the importance of the example is high, the error will be highly minimized then the margin will be made high.

b. Solve SVM

X	Y	H
1,1	1	10
2,2	-1	3
3,3	1	1
4,4	1	4
5,5	-1	5

We can use SVM to solve the above problem in different ways:

- We can do sub-sampling based on the weights ie take positive and negative examples in such a way that they have equal weight importance. In the above dataset for training we will take (1,1) for positive and (2,2), (5,5) for negative examples. So the samples become balanced.
- We can use scaling ie based on importance we increase the number of examples. Since (1,1) has importance 10 then we increase the example (1,1) by 10 times.

11. Coarse to Fine Framework

a. Consider S to be the set of n training examples ie S = $\{x_1, x_2 \dots x_n\}$**.** Run a clustering algorithm to produce K_+ and K_- clusters

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \frac{1}{2} \|w - w_m\|^2 + c \sum_{m=1}^{k} \sum_{i=1}^{N_m} \varepsilon_i^m$$

where m is the m^{th} cluster

b. For each K_+ and K_- clusters, run a clustering algorithm to combine similar points ie K_+ will be further classified. Classify the clusters in such a way that the values closer to the classifier are clustered together.

c. Perform clustering until each cluster cannot be further classified ie when the number of support vectors becomes equal to the number of feature vectors or when the accuracy after every iteration does not change my much.

d. It might fail when there are a lot of redundant training examples

12. Kernelized Perceptron Algorithm

a. Solve Standard Kernelized Perceptron,

Step 1: Consider a set of support vectors SV containing both positive and negative support vectors
Step 2: Split SV into 2 arrays one containing the positive the other containing the negative support vectors. SV is split into S_+ and S_-
Step 3: Sort S_+ and S_- in the descending order
Step 4: From the sorted S_+ take $\frac{B}{2}$ values and from the sorted S_- take $\frac{B}{2}$ values.
Step 5: Therefore we get a B support vectors.

The support vectors which are closer to the decision boundary will have the highest value and has the highest impact on increasing the margin and reducing the error. That is why we have selected half of the highest support vectors from S_+ and the other half from the highest support vectors from S- to have a set of B support vectors.

b. Modify Kernelized Perceptron

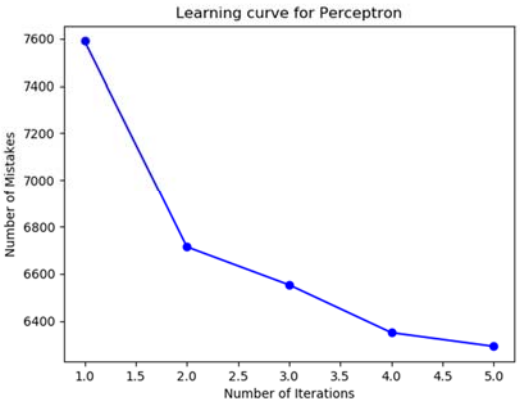
Step 1: For each training example draw a circle in such a way that it comprises only those examples of the same class. Count the number of examples in each circle and store it as N
Step 2: If a training example has a large value of N then ignore the example as it will not be close to the decision boundary
Step 3: Select only the examples which have the smallest values for N ie they are closer to the decision boundary
Step 5: Run Kernelized Perceptron over the smaller dataset for only B kernel evaluations

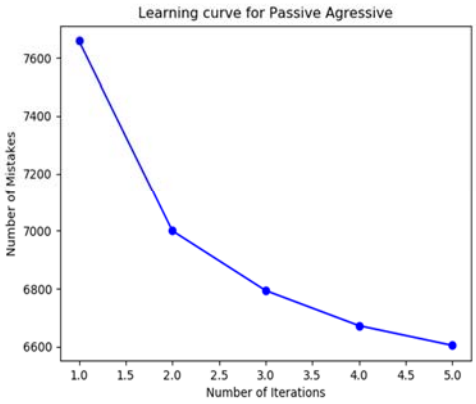
Examples that are closer to the decision boundary play the most important role in maximizing the margin and keeping the error minimum. Therefore we run the algorithm over only those examples and perform only B kernel evaluations which reduces time complexity.

13. Online Learning Algorithm Coding

a. We first separate the feature vectors and class labels separately. We convert the continous feature vectors into categorical values. Eg: Hours per week is split into Low, Medium and High. After conversion we use pandas get_dummies function to convert all the categorical feature vectors into binary feature vectors. There are 96 features.

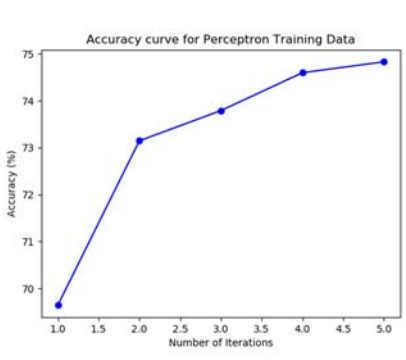
b. Comparison of Online Learning Curve for Perceptron and Passive Aggressive

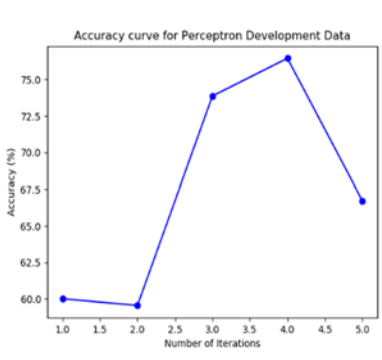


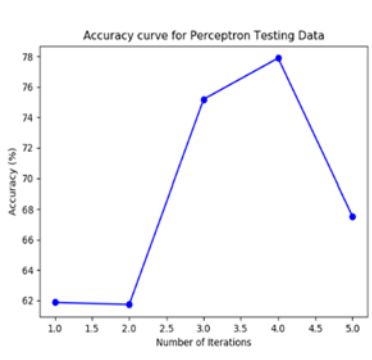


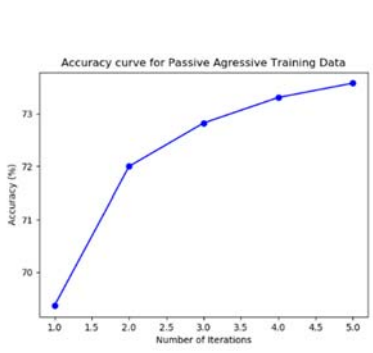
Perceptron at its 1st iteration has 7600 mistakes, 2nd iteration 6700 mistakes and so on. Passive Aggressive at its 1st iteration has 7700 mistakes, 2nd iteration 7000 mistakes and so on. Therefore Perceptron Algorithm learns faster when compared to Passive aggressive for few iterations.

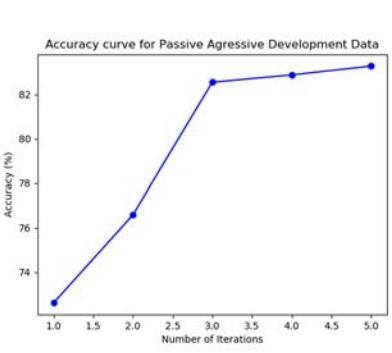
c. Comparison of Accuracy Curves for Perceptron and Passive Aggressive

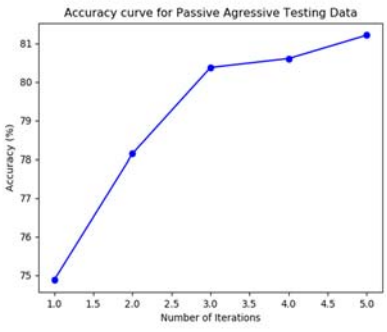












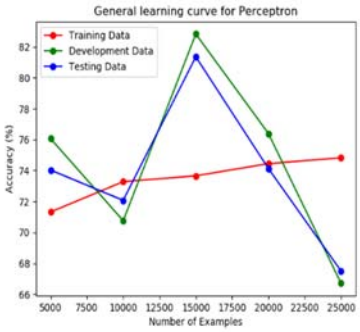
The accuracy curve for training data of perceptron is higher compared to that of passive aggressive for every iteration. But for development and test data passive aggressive has an increase in the accuracy for every iteration. But for perceptron there is an increase in the accuracies and a sharp drop in accuracy when number of iterations increases.

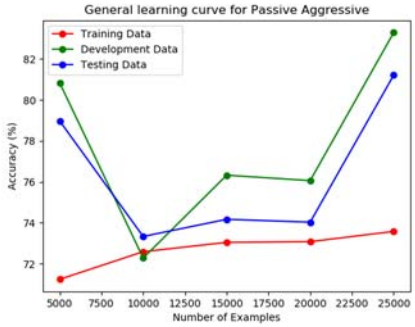
d. Comparison of Accuracies for Perceptron and Average Perceptron

Algorithm	Training Accuracy	Development Accuracy	Test Accuracy
Average Perceptron Naive	74.83	82.63	81.35
Average Perceptron Hal	74.95	83.36	81.45
Standard Perceptron	74.83	66.71	67.51

We notice that the accuracies for training, development and test data is higher for either of the Average Perceptron implementation when compared to Standard Perceptron. But the accuracies for training, development and test data Average Perceptron Hal implementation is higher compared to the Naive Algorithm.

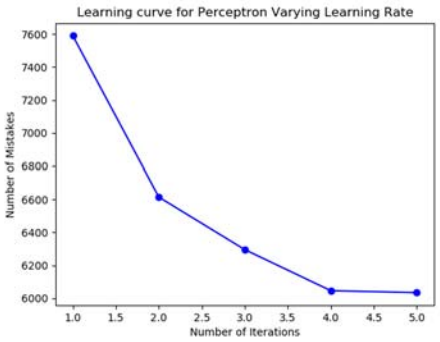
e. Comparison of General Learning Curve for Perceptron and Passive Aggressive

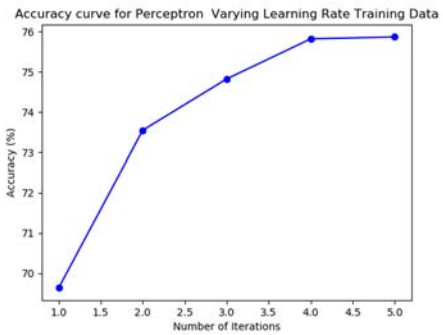


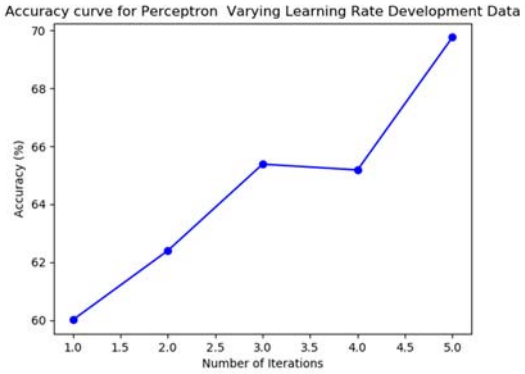


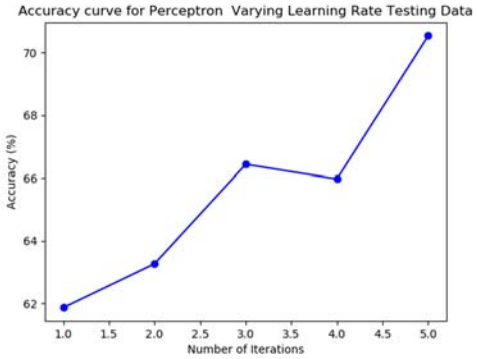
The general learning curve for train of both Perceptron and Passive Aggressive does not change by much when number of examples are increased for every iteration. In perceptron for development and testing data when the number of examples increases initially there is a rise and then drop in accuracy but for passive aggressive initially there is a drop in accuracy when number of iterations increase but afterwards there is a steady increase.

f. Observing Perceptron when learning rate is varied





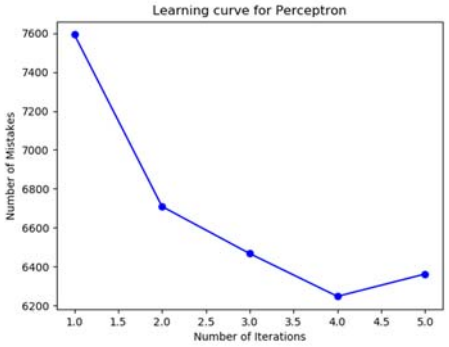


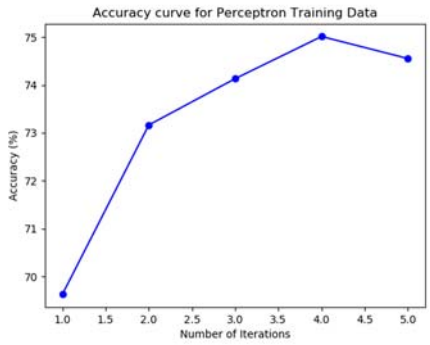


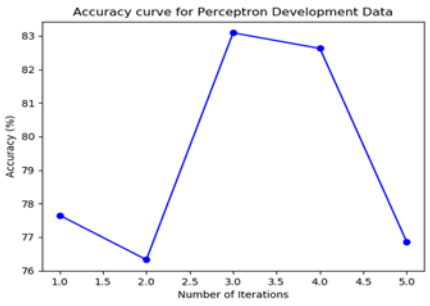
Algorithm	Training Accuracy	Development Accuracy	Test Accuracy
Perceptron Vary Learning Rate	75.86	69.76	70.54
Standard Perceptron	74.83	66.71	67.51

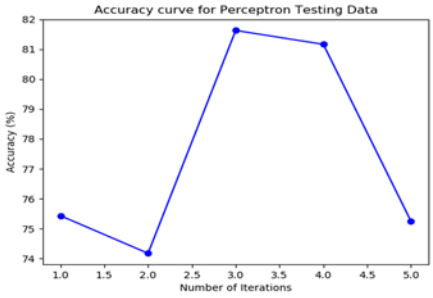
The learning rate is reduced by 0.1 for every iteration. The online learning curve has almost the same number of mistakes for iteration 4 and 5 (almost converge). But the accuracy curve has increased for every iteration without any drop in accuracy when iterations increases. The accuracy has also increased when compared to standard perceptron.

g. Observing Perceptron when training examples are shuffled







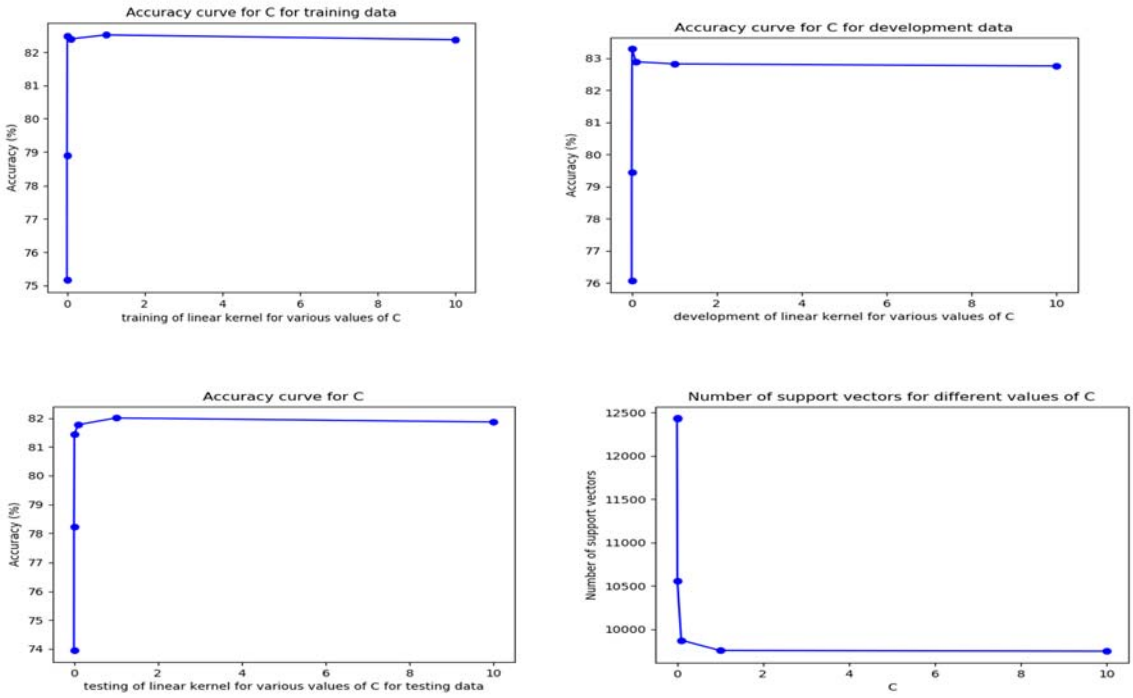


Algorithm	Training Accuracy	Development Accuracy	Test Accuracy
Perceptron Shuffle Examples	74.55	76.86	75.24
Standard Perceptron	74.83	66.71	67.51

The training examples are shuffled every iteration. The online learning curve is varying for every iteration. But the accuracy curve is similar to standard perceptron. Accuracy for training is less while development and test accuracy has increased when compared to standard perceptron.

14. SVM Coding

a. Observations for Varying C



In the accuracy curve, the accuracy for 0.0001 to 0.01 is increasing after that it is almost constant. In the support vectors curve, the number of support vectors for 0.0001 to 0.01 is decreasing after that it is almost constant.

b. Confusion Matrix

Confusion matrix for Training

[[18373 1564]

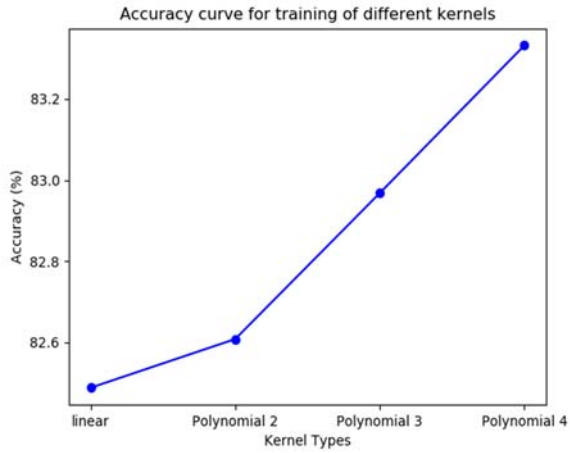
[3083 3488]]

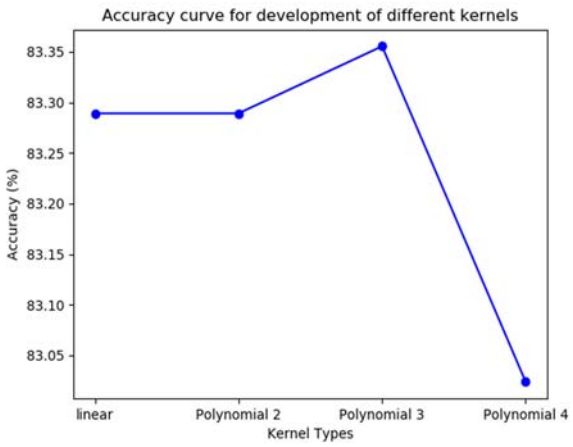
Confusion matrix for Testing

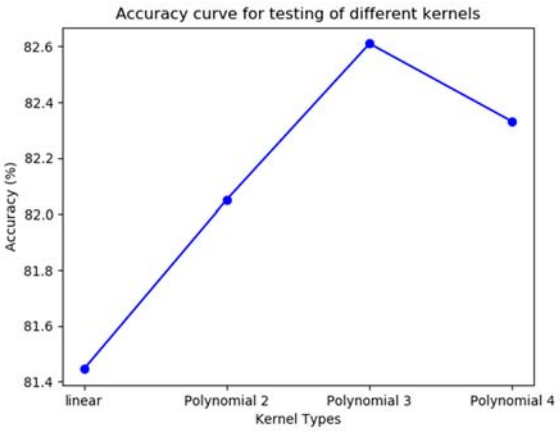
[[1452 134]

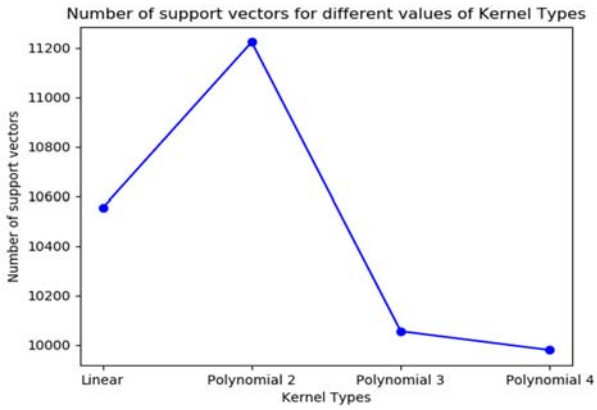
[262 297]]

c. Observations for Different Kernels









Polynomial of Degree 3 Kernel has the highest development and testing accuracy. While Polynomial of degree 4 Kernel has the least supprt vectors.