# CptS 570 MACHINE LEARNING HOMEWORK-2

Sharath Kumar Karnati

15th October 2024

---

**1 Analytical Part (3 percent)**

**Question 1:**

## Questions

**1.** 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.,

$$C^+ = \frac{1}{n^+} \sum_{i:y_i=+1} x_i \quad and \quad C^- = \frac{1}{n^-} \sum_{i:y_i=-1} x_i.$$

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 $C^-$.

**Solution:**

We begin by calculating the weight vector $w$ for the classifier. Let $C^+$ represent the center of the positive class and $C^-$ represent the center of the negative class. The weight vector is defined as the difference between these two cluster centers:

$$w = C^+ - C^-$$

The decision boundary is defined by the hyperplane where the classification score equals zero:

$$sign(w \cdot x + b) = 0 \quad \Rightarrow \quad w \cdot x + b = 0$$

Now, substitute the expression for $w$ from equation (1) into equation (2):

$$(C^+ - C^-) \cdot x + b = 0$$

Next, we consider a point $x$ that lies exactly halfway between the two cluster centers, which would be the midpoint of the weight vector:

$$x = \frac{C^+ + C^-}{2}$$

Substitute this value of $x$ into equation (3):

$$(C^+ - C^-) \cdot \left( \frac{C^+ + C^-}{2} \right) + b = 0$$

Expanding this expression:

$$\frac{1}{2} \left( ||C^+||^2 - ||C^-||^2 \right) + b = 0$$

Solving for $b$, we get:

$$b = -\frac{1}{2} \left( ||C^+||^2 - ||C^-||^2 \right)$$

Thus, the weight vector and bias term are:

$$w = C^+ - C^-, \quad b = -\frac{1}{2} \left( ||C^+||^2 - ||C^-||^2 \right)$$

**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 ($\alpha$'s). How many of the training examples are support vectors?**

**Solution:** Now, let's compute the dual weights ($\alpha_i$). From the previous expression, the weight vector is:

$$w = C^+ - C^-$$

We know that the center $C^+$ of the positive class and the center $C^-$ of the negative class can be expressed as the average of the corresponding training examples:

$$C^+ = \frac{1}{n^+} \sum_{i:y_i=+1} x_i$$

$$C^- = \frac{1}{n^-} \sum_{i:y_i=-1} x_i$$

Substituting equations (2) and (3) into equation (1):

$$w = \frac{1}{n^+} \sum_{i:y_i=+1} x_i - \frac{1}{n^-} \sum_{i:y_i=-1} x_i$$

Now, we can express $w$ as a weighted sum over all the training examples:

$$w = \frac{1}{n^+ + n^-} \sum_{i=1}^{n^+ + n^-} y_i x_i$$

Recall that in the dual form of the SVM, the weight vector is expressed as:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

By comparing equations (4) and (5), we find that:

$$\alpha_i = \frac{1}{n^+ + n^-}$$

Since the weight vector is a linear combination of the training examples, and because the boundary is perfectly defined by the centers of the two clusters, only one support vector from each class is sufficient to define the decision boundary. Therefore, the number of support vectors is 1 from each class.

**Question 2.** Suppose we use the following radial basis function (RBF) kernel:

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

which has some implicit unknown mapping $\phi(x)$.

Prove that the mapping $\phi(x)$ corresponding to RBF kernel has infinite dimensions.

**Solution:**

The Radial Basis Function (RBF) kernel is defined as:

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

We can expand this expression as follows:

$$K(x_i, x_j) = \exp\left(-\frac{1}{2}\langle x_i - x_j, x_i - x_j\rangle\right)$$

Expanding the inner product:

$$K(x_i, x_j) = \exp\left(-\frac{1}{2}\left(\langle x_i, x_i\rangle - \langle x_i, x_j\rangle - \langle x_j, x_i\rangle + \langle x_j, x_j\rangle\right)\right)$$

$$K(x_i, x_j) = \exp\left(-\frac{1}{2}\left(\|x_i\|^2 + \|x_j\|^2 - 2\langle x_i, x_j\rangle\right)\right)$$

We can split the exponential function into two parts:

$$K(x_i, x_j) = \exp\left(-\frac{1}{2}\|x_i\|^2\right) \cdot \exp\left(-\frac{1}{2}\|x_j\|^2\right) \cdot \exp\left(\langle x_i, x_j\rangle\right)$$

Now, using the Taylor expansion for the exponential function, we can express $\exp(\langle x_i, x_j\rangle)$ as:

$$\exp(\langle x_i, x_j\rangle) = \sum_{n=0}^{\infty} \frac{\langle x_i, x_j\rangle^n}{n!}$$

Thus, the RBF kernel can be rewritten as an infinite sum of polynomial kernels:

$$K(x_i, x_j) = \exp\left(-\frac{1}{2}\|x_i\|^2\right) \cdot \exp\left(-\frac{1}{2}\|x_j\|^2\right) \cdot \sum_{n=0}^{\infty} \frac{\langle x_i, x_j\rangle^n}{n!}$$

We observe that the RBF kernel is formed by taking an infinite sum over polynomial kernels. Therefore, we can conclude that the mapping $\phi(x)$ corresponding to the 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 $\phi$ is less than 2, i.e., $\|\phi(x_i) - \phi(x_j)\|^2 \leq 2$.

**Solution:**

The squared distance between the feature mappings $\phi(x_i)$ and $\phi(x_j)$ in the feature space is given by:

$$\|\phi(x_i) - \phi(x_j)\|^2 = \|\phi(x_i)\|^2 + \|\phi(x_j)\|^2 - 2\langle\phi(x_i), \phi(x_j)\rangle$$

This can be rewritten using the kernel function $K(x_i, x_j)$:

$$\|\phi(x_i) - \phi(x_j)\|^2 = K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)$$

Substituting the RBF kernel expression:

$$\|\phi(x_i) - \phi(x_j)\|^2 = \exp\left(-\frac{1}{2}\|x_i - x_i\|^2\right) + \exp\left(-\frac{1}{2}\|x_j - x_j\|^2\right) - 2\exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right)$$

Since $\|x_i - x_i\|^2 = 0$ and $\|x_j - x_j\|^2 = 0$, this simplifies to:

$$\|\phi(x_i) - \phi(x_j)\|^2 = \exp(0) + \exp(0) - 2\exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right)$$

$$\|\phi(x_i) - \phi(x_j)\|^2 = 1 + 1 - 2\exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right)$$

Finally:

$$\|\phi(x_i) - \phi(x_j)\|^2 = 2 - 2\exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right)$$

Thus, we can conclude that:

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

This result shows that the squared distance between the feature mappings is bounded by 2, because the exponential term is always strictly less than 2.

**Question 3.** 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 $\alpha_i$ is the dual weight of the $i$th support vector.
Let us assume that we use the radial basis function (RBF) kernel:

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

and also assume that the training examples are linearly separable in the feature space $\phi$, 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 $R^d$). Prove that $f(x_{far}; \alpha, b) \approx b$.

**Solution:**
The function $f(x_{far}, \alpha, b)$ is given by:

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

Using the Radial Basis Function (RBF) kernel, $K(x_{far}, x_i)$ can be expressed as:

$$f(x_{far}, \alpha, b) = y_i \alpha_i \exp\left(-\frac{1}{2}\|x_{far} - x_i\|^2\right) + b$$

Since $\|x_{far} - x_i\|^2 > 0$, the exponential term decays rapidly. Therefore, for large values of $\|x_{far} - x_i\|$, we approximate:

$$\exp\left(-\frac{1}{2}\|x_{far} - x_i\|^2\right) \approx 0$$

Thus, the function simplifies to:

$$f(x_{far}, \alpha, b) = 0 + b$$

Hence, we have the final approximation:

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

This shows that for points $x_{far}$ far from the training data points $x_i$, the output of the function is approximately the bias term $b$.

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

**Solution:**

To verify whether a function is a valid kernel, it must satisfy the following two conditions:

1. **The kernel should be symmetric:**
   This means that $K(x_i, x_j) = K(x_j, x_i)$.

   $$K(x_i, x_j) = -\langle x_i, x_j \rangle$$

   Since the dot product is symmetric, i.e., $\langle x_i, x_j \rangle = \langle x_j, x_i \rangle$, we have:

   $$K(x_i, x_j) = -\langle x_j, x_i \rangle = K(x_j, x_i)$$

   Therefore, the kernel is symmetric.

2. **The kernel matrix should be positive definite:**
   For any vector $m$, the following condition must hold:

   $$m^T K m > 0$$

   For the kernel $K(x_i, x_j) = \langle x_i, x_j \rangle$, the kernel matrix $K$ is represented as:

   $$K = \langle x_1, x_1 \rangle \langle x_1, x_2 \rangle \langle x_2, x_1 \rangle \langle x_2, x_2 \rangle = abcd$$

   Now, if the kernel is modified to $K(x_i, x_j) = -\langle x_i, x_j \rangle$, the matrix becomes:

   $$K = -\langle x_1, x_1 \rangle - \langle x_1, x_2 \rangle - \langle x_2, x_1 \rangle - \langle x_2, x_2 \rangle = -a - b - c - d$$

   When computing the product $m^T K m$, where $m = n$
   $p$, the result is:

   $$m^T K m = np - a - b - c - dnp = -[(na + pb)n + (nc + pd)p]$$

   This results in $m^T K m < 0$, which shows that the kernel matrix is not positive definite, and hence, this is not a valid kernel.

   **Conclusion:** While the kernel is symmetric, it is not a positive definite matrix. Therefore, the kernel $K(x_i, x_j) = -\langle x_i, x_j \rangle$ is not a valid kernel.

**Question 5.** You are provided with $n$ training examples: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i$ is the input example, and $y_i$ 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.

How will you modify the Soft-margin SVM formulation to be able to leverage this extra information? Please justify your answer.

**Solution:**

In Support Vector Machines (SVM), the objective of the soft margin approach is to allow for some misclassification of data points while still striving for a maximal margin between the classes. The modified optimization problem can be expressed as follows:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C^+ \sum_{i \in S_1} \xi_n^+ + C^- \sum_{i \in S_1} \xi_n^-$$

Subject to: $\text{y}_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, N \quad and \quad \xi_n \geq 0$

In this formulation:

- $\|w\|^2$ represents the regularization term that aims to minimize the complexity of the model by keeping the weight vector $w$ small, which contributes to maximizing the margin. - $C^+$ and $C^-$ are positive constants that determine the penalty for misclassifying positive and negative examples, respectively. - The variables $\xi_n^+$ and $\xi_n^-$ are the slack variables that allow some flexibility in classifying data points. They measure the degree of misclassification of positive and negative examples.

The constraints ensure that correctly classified points satisfy the margin requirement, adjusted by the slack variables. By incorporating the penalties $C^+$ and $C^-$ into the optimization function, we can control the influence of misclassified outliers. This results in a more robust model that improves the margin while effectively handling errors in classification.

Overall, this soft margin formulation strikes a balance between maximizing the margin and minimizing misclassifications, thus leading to better generalization in the presence of noise or overlapping classes.

**Question 6.** You are provided with a set of $n$ training examples: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i$ is the input example, and $y_i$ 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 the large training set.
Tom wants to devise a solution based on the "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?

**Solution:**

Let $M_1, M_2, M_3, \ldots, M_k$ represent the $k^+$ positive clusters, and $N_1, N_2, N_3, \ldots, N_k$ represent the $k^-$ negative clusters. For the purpose of SVM training, we can utilize the centroids of these clusters as representative examples, assigning them corresponding labels.

Denote the centroid of the positive cluster $M_i$ as $C_{M_i}$. This centroid acts as a training sample for our classifier, labeled with 1 to signify a positive classification. If $X_{P_i}$ denotes the total number of examples in the cluster $M_i$, the centroid is calculated as follows:

$$C_{P_i} = \frac{1}{N_{P_i}} \sum_{j=1}^{N_{P_i}} P_{ij} x_j$$

where $x_j$ are the individual data points belonging to the positive cluster $M_i$.

The optimization problem for the soft margin SVM can be expressed as:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{k^+ + k^-} \xi_n Subjectto: y_n(w \cdot x_n + b) \geq 1 - \xi_n, \quad \forall n and \xi_n \geq 0, \quad \forall n$$

In this formulation:

- $\|w\|^2$ serves as a regularization term that encourages a larger margin between classes, thus improving the generalization of the model. - $C$ is a hyperparameter that balances the trade-off between maximizing the margin and minimizing the classification errors. - The slack variables $\xi_n$ allow for some misclassifications, providing flexibility in the model to accommodate noise or overlapping classes.

By using the centroids as training examples, this approach enhances the SVM's ability to classify data by leveraging the structure of positive and negative clusters effectively.

b) How to refine the clusters based on the resulting SVM classifier?

**Solution:** To enhance the clustering process based on the SVM classifier's outcomes, we can focus on refining the clusters that are situated near the decision boundary. The approach involves the following steps:

1. **Identify Clusters Near the Decision Boundary:** First, determine which clusters lie close to the decision boundary established by the SVM classifier. These clusters may exhibit overlapping characteristics, making them more susceptible to misclassification.

2. **Split Clusters into Smaller Subclusters:** For each identified cluster near the decision boundary, divide it into smaller subclusters. This can be achieved using clustering algorithms such as K-means, hierarchical clustering, or density-based clustering techniques, depending on the distribution of data points within the cluster.

3. **Train on the New Subclusters:** Use the newly formed subclusters as training examples for the SVM classifier. Each subcluster will have its centroid calculated, and these centroids can serve as representative data points, labeled according to their respective clusters.

4. **Iterate the Process:** After training the SVM with the subclusters, reevaluate the decision boundary. If any new clusters still lie near the decision boundary, repeat the process of splitting and training until the clusters are sufficiently refined.

By employing this method, we can enhance the SVM's performance, particularly in regions where data points are ambiguous or where class overlap occurs. This iterative refinement helps improve classification accuracy and the overall robustness of the model.

c) What is the stopping criterion?

**Solution:**

One of the terminating criteria parameters for the iterative process could be accuracy. If we observe that there is no meaningful progress during the iterations, we can halt the process once a specific threshold is reached.

For instance, let $x$ represent the accuracy at a certain iteration of the technique as we refine from coarse to fine clustering, and let $x'$ denote the new accuracy at the subsequent iteration. We can compute the difference in accuracy as follows:

$$|x' - x| \leq \phi$$

where $\phi$ is the predefined threshold for accuracy improvement. If the absolute difference between the two accuracies falls below this threshold, it indicates that further iterations are unlikely to yield significant improvements, prompting us to terminate the process.

**Optional question:** For what kind of problems will this solution fail?

**Solution:** In scenarios where there are insufficient training examples, the proposed solution may fail to perform effectively. The lack of diverse and representative training data can lead to overfitting or underfitting, compromising the model's ability to generalize well to unseen data. Consequently, it is crucial to ensure that an adequate number of training examples is available for the model to learn meaningful patterns and achieve reliable performance.

**7.** You are provided with a set of $n$ training examples: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i$ is the input example, and $y_i$ 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 is unbounded.

a) Suppose you have trained using the 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.

**Solution:**

- Let $m$ denote the number of support vectors $\alpha_i$ from the total $n$ training instances.
- Define $x_1, x_2, x_3, \ldots, x_m$ as the training instances that correspond to each of the support vectors.
- The activation function $w \cdot x + b$ indicates how much an example strays from the decision boundary.
- The support vectors that hold the most significance are those that are nearest to the decision boundary.
- Therefore, it is advisable to choose support vectors that have the lowest activation function values, extending the boundary up to a specified limit $B$.

b) Tom wants to train using the 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.

**Solution:**

During the training of the kernelized perceptron algorithm, we maintain a maximum of $B$ support vectors at each iteration. When we encounter a misclassified instance and have reached the limit of $B$ support vectors, we employ the following strategy to modify an existing support vector:

- We assess the magnitude of the activation function for each $\alpha_i$, which indicates how far the example is from the decision boundary.
- In selecting which example to replace, we prioritize the one that is farthest from the decision boundary. This is because such examples often contribute less critical information to the classification task due to their higher activation values.

**Programming and Empirical Analysis Part (6 percent grade):**

**1.**
**a.**

**Solution :**

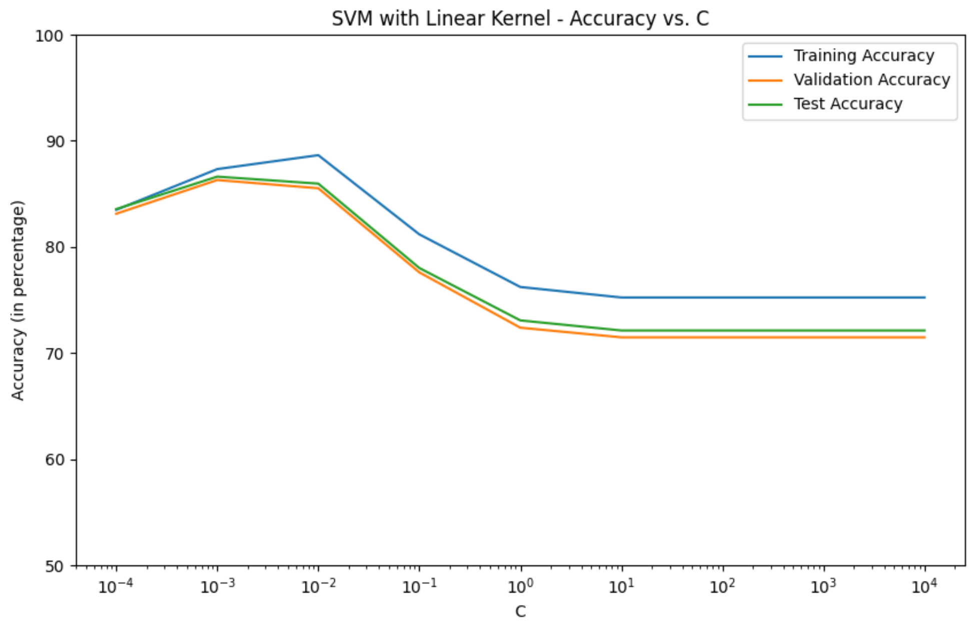

Figure 1: Accuracy Graph.

This graph shows us that the accuracy of training is higher than testing and validation's accuracy, but the difference is very small. Which shows that the data isn't being overfitted or underfitted. So the model is a good one. As C growing, the training,testing and validation accuracy is decreasing.

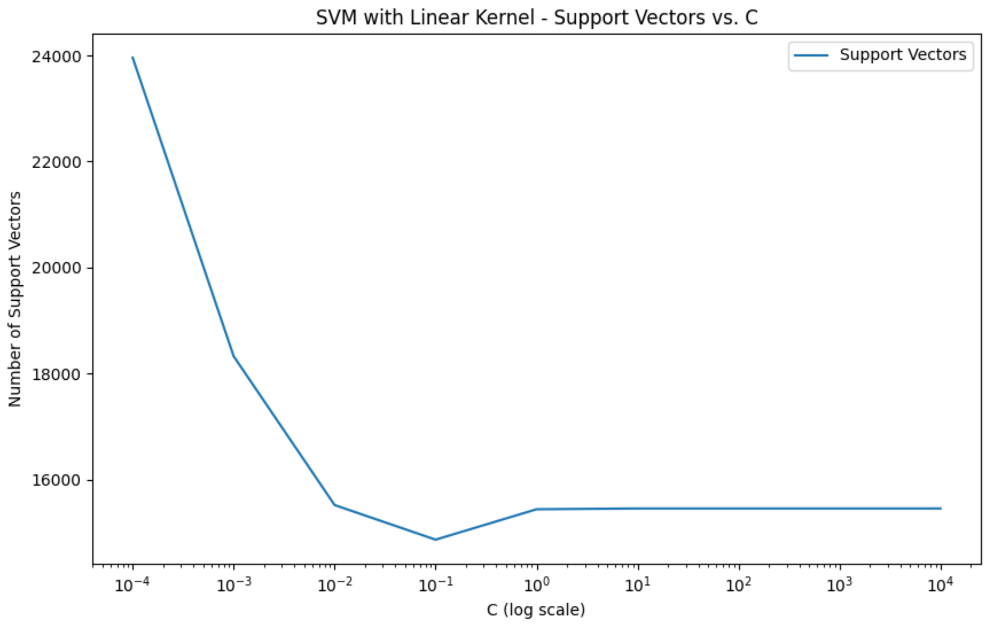

Figure 2: Support Vectors Graph.

The above graphs shows us the graph of number of support vectors using SVC kernal Function.

**b.**

**Solution :**

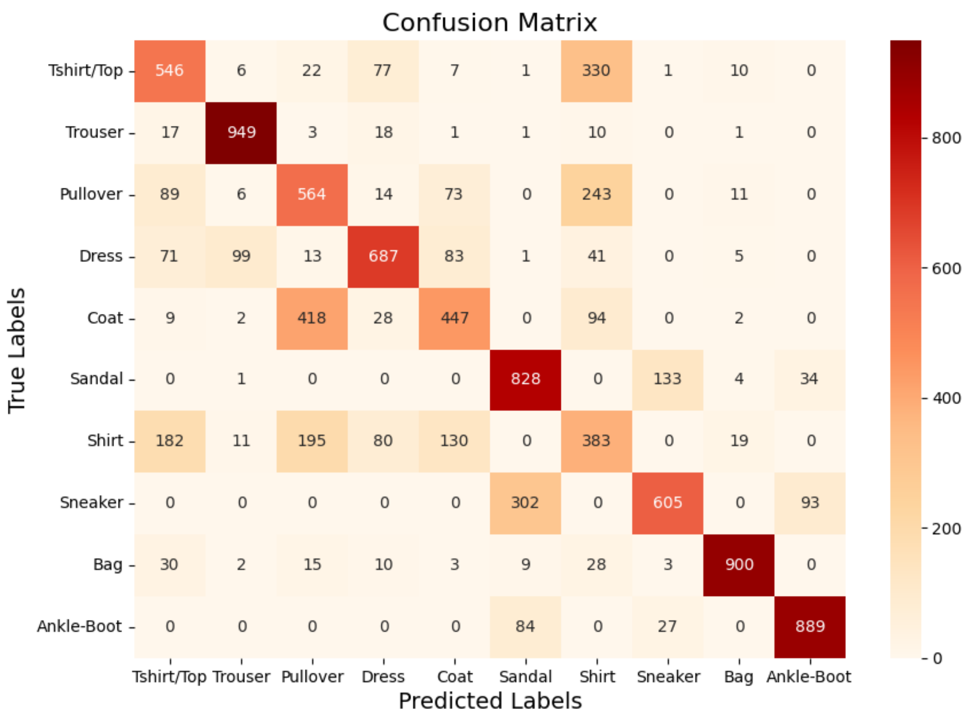

Figure 3: Confusion Matrix.

I acheived the accuracy of 72 percent.

**c.**

**Solution :**

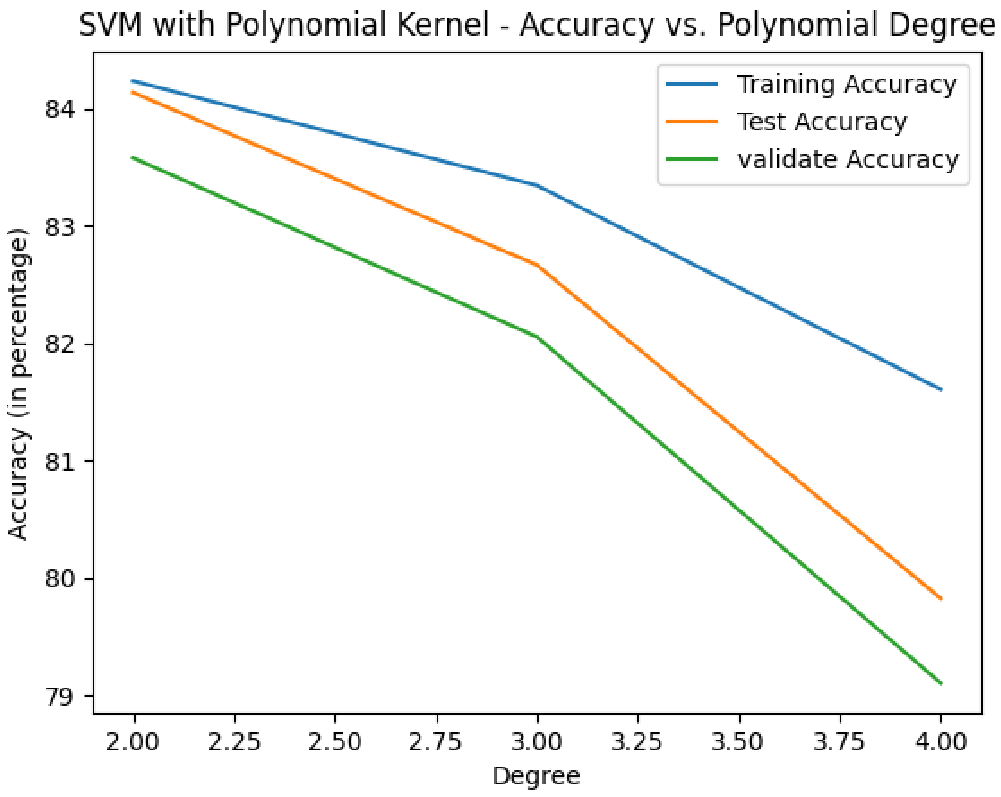

Figure 4: Accuracy vs Degree of polynomial.

The testing accuracy graph indicates that as the model complexity increases, the accuracy begins to decrease. This suggests that the model is well-calibrated for the data at lower complexities, but as it becomes more intricate, it may start to overfit the training data rather than generalize effectively

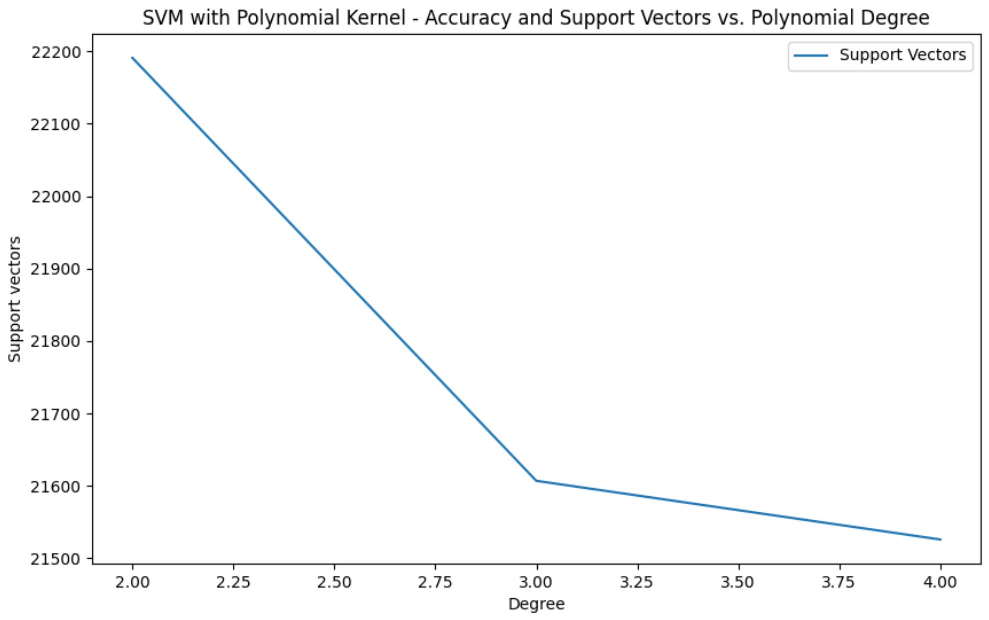

Figure 5: Number of Support vectors.

The observation highlights that as the degree increases, the count of support vectors tends to decrease, which makes sense. This trend suggests that moving to higher dimensions enhances the model's ability to separate the data more effectively, resulting in a clearer distinction between different classes.

**2 a.**

**Solution :**

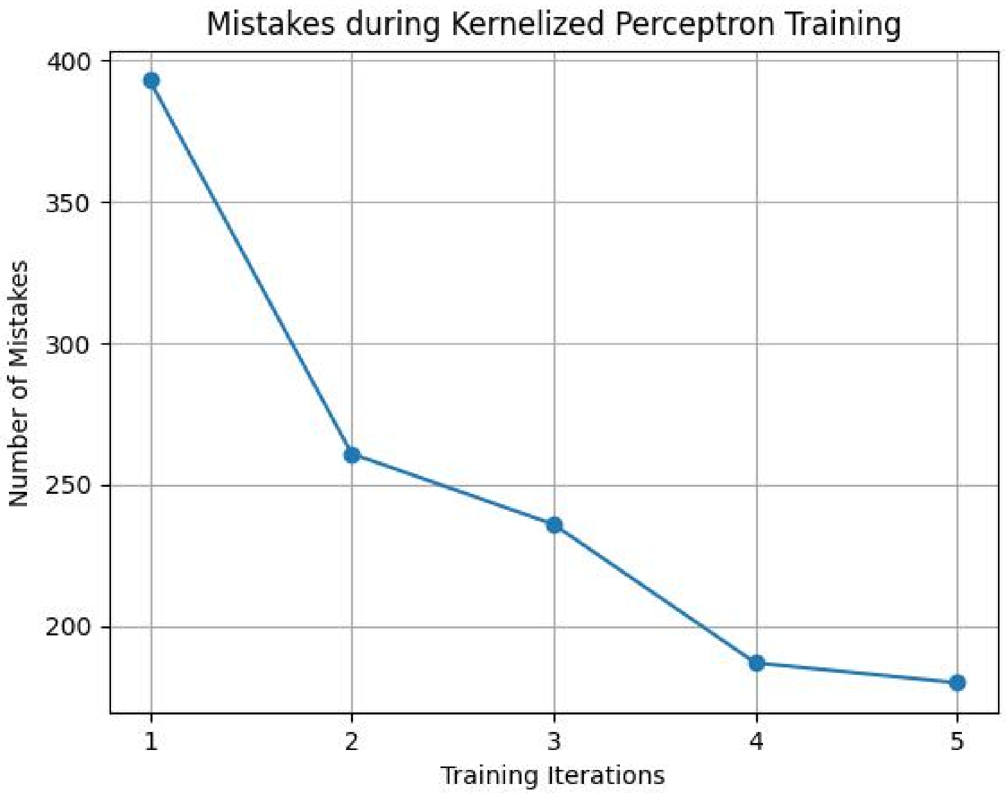

Figure 6: Kernalized Perceptron Mistakes.

Iteration 1: Mistakes = 393
Iteration 2: Mistakes = 261
Iteration 3: Mistakes = 236
Iteration 4: Mistakes = 187
Iteration 5: Mistakes = 180
Training Accuracy: 85.80
Validation Accuracy: 78.50
Testing Accuracy: 74.46

As the number of iterations increases, the model makes fewer errors, suggesting that it is able to acquire new insights from the same training data through repeated learning. Additionally, the accuracy of the model remains relatively consistent between the training and test datasets, indicating that it generalizes well to unseen data. Therefore, we can conclude that this classifier is well-suited for the task.

**3 a.b.c.d.**

**Solution:**

Codes are attached in the file.
Validation Accuracy (without pruning): 0.93
Test Accuracy (without pruning): 0.96
Validation Accuracy (after pruning): 0.93
Test Accuracy (after pruning): 0.93
**Observations:**

1. The decision tree without pruning achieved a validation accuracy of 0.93 (93 percent) and a test accuracy of 0.96 (96percent). This indicates that while the model performs well on both the training and test datasets, it may be prone to overfitting, as it captures noise along with the underlying patterns in the training data.

2. After applying pruning, the validation accuracy remained at 0.93 (93percent), while the test accuracy decreased to 0.93 (93percent). This slight drop in test accuracy suggests that while pruning may reduce the model's complexity and improve its generalization ability, it might also lead to a loss of some information that was helpful for making accurate predictions on the test set.

3. In summary, although the pruned decision tree shows comparable validation accuracy, its test accuracy reflects a tendency towards better generalization. Pruning helps prevent overfitting, which can lead to better performance on unseen data, although this comes at the cost of some accuracy on the training data. Thus, a balance between accuracy and generalization is crucial when evaluating model performance.