Intro to Machine Learning (CS771A, Autumn 2020)
Homework 2
Due Date: November 25, 2020 (11:59pm)

Instructions:

- Only electronic submissions will be accepted. Your main PDF writeup must be typeset in LaTeX (please also refer to the "Additional Instructions" below).
- The PDF writeup containing your solution has to be submitted via Gradescope `https://www.gradescope.com/` and the code for the programming part (to be submitted via this Dropbox link: `https://tinyurl.com/cs771-a20-hw2`)
- We have created your Gradescope account (you should have received the notification). Please use your IITK CC ID (not any other email ID) to login. Use the "Forgot Password" option to set your password.

Additional Instructions

- We have provided a LaTeX template file `hw2sol.tex` to help typeset your PDF writeup. There is also a style file `ml.sty` that contain shortcuts to many of the useful LaTeX commends for doing things such as boldfaced/calligraphic fonts for letters, various mathematical/greek symbols, etc., and others. Use of these shortcuts is recommended (but not necessary).
- Your answer to every question should begin on a new page. The provided template is designed to do this automatically. However, if it fails to do so, use the `\clearpage` option in LaTeX before starting the answer to a new question, to *enforce* this.
- While submitting your assignment on the Gradescope website, you will have to specify on which page(s) is question 1 answered, on which page(s) is question 2 answered etc. To do this properly, first ensure that the answer to each question starts on a different page.
- Be careful to flush all your floats (figures, tables) corresponding to question n before starting the answer to question $n+1$ otherwise, while grading, we might miss your important parts of your answers.
- Your solutions must appear in proper order in the PDF file i.e. solution to question n must be complete in the PDF file (including all plots, tables, proofs etc) before you present a solution to question $n+1$.
- For the programming part, all the code and README should be zipped together and submitted as a single file named `yourrollnumber.zip`. Please DO NOT submit the data provided.

Problem 1 (20 marks)

(Second-Order Optimization for Logistic Regression) Show that, for the logistic regression model (assuming each label $y_n \in \{0,1\}$, and no regularization) with loss function $\mathcal{L}(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n\boldsymbol{w}^\top\boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top\boldsymbol{x}_n)))$, iteration t of a *second-order* optimization based update $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - {\mathbf{H}^{(t)}}^{-1}\boldsymbol{g}^{(t)}$, where $\mathbf{H}$ denotes the Hessian and $\boldsymbol{g}$ denotes the gradient, reduces to solving an *importance-weighted* regression problem of the form $\boldsymbol{w}^{(t+1)} = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N}\gamma_n^{(t)}(\hat{y}_n^{(t)} - \boldsymbol{w}^\top\boldsymbol{x}_n)^2$, where γ_n denotes the importance of the n^{th} training example and $\hat{y}_n$ denotes a modified real-valued label. Also, clearly write down the expression for both, and provide a brief justification as to why the expression of γ_n makes intuitive sense here.

Problem 2 (20 marks)

(Perceptron with Kernels) We have seen that, due to the form of Perceptron updates $\boldsymbol{w} = \boldsymbol{w} + y_n\boldsymbol{x}_n$ (ignore the bias b), the weight vector learned by Perceptron can be written as $\boldsymbol{w} = \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n$, where α_n is the number of times Perceptron makes a mistake on example n. Suppose our goal is to make Perceptron learn nonlinear boundaries, using a kernel k with feature map ϕ. Modify the standard Perceptron algorithm to do this. In particular, for this kernelized variant of the Perceptron algorithm (1) Give the initialization, (2) Give the mistake condition, and (3) Give the update equation.

Problem 3 (20 marks)

(SVM with Unequal Class Importance) Sometimes it costs us a lot more to classify negative points as positive than positive points as negative. (for instance, if we are predicting if someone has cancer then we would rather err on the side of caution (predicting "yes" when the answer is "no") than vice versa). One way of expressing this in the support vector machine model is to assign different costs to the two kinds of mis-classification. The primal formulation of this is:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{||\boldsymbol{w}||^2}{2} + \sum_{n=1}^{N} C_{y_n}\xi_n$$

subject to $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0, \forall n$.

The only difference is that instead of one cost parameter C, there are two, C_{+1} and C_{-1}, representing the costs of misclassifying positive examples and misclassifying negative examples, respectively.

Write down the Lagrangian problem of this modified SVM. Take derivatives w.r.t. the primal variables and construct the dual, namely, the maximization problem that depends only on the dual variables $\boldsymbol{\alpha}$, rather than the primal variables. In your final PDF write-up, you need not give each of every step in these derivations (e.g., standard steps of substituting and eliminating some variables) but do write down the key steps. Explain (intuitively) how this differs from the standard SVM dual problem; in particular, how the C variables differ between the two duals.

Problem 4 (20 marks)

(SGD for K-means Objective) Recall the K-means objective function: $\mathcal{L} = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}||x_n - \mu_k||^2$. As we have seen, the K- means algorithm minimizes this objective by taking a greedy iterative approach of assigning each point to its closest center (finding the z_{nk}'s) and updating the cluster means $\{\mu_k\}_{k=1}^{K}$. The standard K-means algorithm is a batch algorithm and uses all the data in every iteration. It however can be made online by taking a random example x_n at a time, and then (1) assigning x_n "greedily" to the "best" cluster, and (2) updating the cluster means using SGD on the objective $\mathcal{L}$. Assuming you have initialized $\{\mu_k\}_{k=1}^{K}$ randomly and are reading one data point x_n at a time,

- How would you solve step 1?
- What will be the SGD-based cluster mean update equations for step 2? Intuitively, why does the update equation make sense?

- Note that the SGD update requires a step size. For your derived SGD update, suggest a good choice of the step size (and mention why you think it is a good choice).

Problem 5 (20 marks)

(Kernel K-means) Assuming a kernel k with an infinite dimensional feature map ϕ (e.g., an RBF kernel), we can neither store the kernel-induced feature map representation of the data points nor can store the cluster means in the kernel-induced feature space. How can we still implement the kernel K-means algorithm in practice? Justify your answer by sketching the algorithm, showing all the steps (initialization, cluster assignment, mean computation), clearly giving the mathematical operations in each. In particular, what is the difference between how the clusters means would need to be stored in kernel K-means versus how they are stored in standard K-means? Finally, assuming each input to be D-dimensional in the original feature space, and N to be the number of inputs, how does kernel K-means compare with standard K-means in terms of the cost of input to cluster mean distance calculation (please answer this using the big $\mathcal{O}$ notation)?

Problem 6 (Programming Problem, 50 marks)

Part 1: You are provided a dataset in the file `binclass.txt`. In this file, the first two numbers on each line denote the two features of the input $\boldsymbol{x}_n$, and the third number is the binary label $y_n \in \{-1, +1\}$.

Implement a generative classification model for this data assuming Gaussian class-conditional distributions of the positive and negative class examples to be $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_+, \sigma_+^2\mathbf{I}_2)$ and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_-, \sigma_-^2\mathbf{I}_2)$, respectively. Note that here $\mathbf{I}_2$ denotes a 2×2 identity matrix. Assume the class-marginal to be $p(y_n = 1) = 0.5$, and use MLE estimates for the unknown parameters. Your implementation need not be specific to two-dimensional inputs and it should be almost equally easy to implement it such that it works for any number of features (but it is okay if your implementation is specific to two-dimensional inputs only).

On a two-dimensional plane, plot the examples from both the classes (use red color for positives and blue color for negatives) and the **learned decision boundary** for this model. Note that we are not providing any separate test data. Your task is only to learn the decision boundary using the provided training data and visualize it.

Next, repeat the same exercise but assuming the Gaussian class-conditional distributions of the positive and negative class examples to be $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_+, \sigma^2\mathbf{I}_2)$ and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_-, \sigma^2\mathbf{I}_2)$, respectively.

Finally, try out an SVM classifier (with **linear kernel**) on this data (we've also provided the data in the format libSVM requires) and show the learn decision boundary. For this part, you do not need to implement SVM. There are many nice implementations of SVM available, such as the one in scikit-learn and the very popular libSVM toolkit. Assume the "C" (or λ) hyperparameter of SVM in these implementations to be 1.

Part 2: Repeat the same experiments as you did for part 1 but now using a different dataset `binclassv2.txt`. Looking at the results of both the parts, which of the two models (generative classification with Gaussian class-conditional and SVM) do you think seems to work better for each of these datasets, and in general?

Deliverables: Include your plots (use a separate, appropriately labeled plot, for each case) and experimental findings in the main writeup PDF. Submit your codes in a separate zip file on the provided Dropbox link. Please comment the code so that it is easy to read and also provide a README that briefly explains how to run the code. For the SVM part, you do not have to submit any code but do include the plots in the PDF (and mention the software used - scikit-learn or libSVM).

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Sajal Goyal
Roll Number: 180651
Date: November 27, 2020

QUESTION 1

For second order optimization, using concept of Taylor series expansion, we get:

$$L(\mathbf{w},\mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) + (\mathbf{w}-\mathbf{w}^{(t)})^T\nabla L(\mathbf{w}^{(t)}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}^{(t)})^T\nabla^2 L(\mathbf{w}^{(t)})(\mathbf{w}-\mathbf{w}^{(t)})$$

where loss function $L(\mathbf{w})$, with $y_n \in \{0,1\}$:

$$L(\mathbf{w}) = -\sum_{n=1}^{N}(y_n\mathbf{w}^T\mathbf{x}_n - \log(1+\exp{(\mathbf{w}^T\mathbf{x}_n)}))$$

So we need to calculate $\nabla L(\mathbf{w})$ and $\nabla^2 L(\mathbf{w})$ first. We get :

$$\nabla L(\mathbf{w}) = -\sum_{n=1}^{N}(y_n\mathbf{x}_n - \frac{\mathbf{x}_n\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})}) = -\sum_{n=1}^{N}(y_n\mathbf{x}_n - \mathbf{x}_n + \frac{\mathbf{x}_n}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})})$$

$$\nabla^2 L(\mathbf{w}) = 0 + 0 + \sum_{n=1}^{N}\frac{\mathbf{x}_n\mathbf{x}_n^T\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})^2} = \sum_{n=1}^{N}\frac{\mathbf{x}_n\mathbf{x}_n^T\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})^2}$$

Putting the values in the first equation, we get:

$$L(\mathbf{w},\mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) - (\mathbf{w}-\mathbf{w}^{(t)})^T\sum_{n=1}^{N}(y_n\mathbf{x}_n - \frac{\mathbf{x}_n\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})})$$

$$+\frac{1}{2}(\mathbf{w}-\mathbf{w}^{(t)})^T\sum_{n=1}^{N}\frac{\mathbf{x}_n\mathbf{x}_n^T\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})^2})(\mathbf{w}-\mathbf{w}^{(t)})$$

$$L(\mathbf{w},\mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) - \sum_{n=1}^{N}(y_n(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n - \frac{(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})})$$

$$+\frac{1}{2}\sum_{n=1}^{N}\frac{(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n\mathbf{x}_n^T(\mathbf{w}-\mathbf{w}^{(t)})\exp{(\mathbf{w}^T\mathbf{x}_n)}}{(1+\exp{(\mathbf{w}^T\mathbf{x}_n)})^2})$$

Let $\sigma = \frac{\exp{(\mathbf{w}^{(t)T}\mathbf{x}_n)}}{1+\exp{(\mathbf{w}^{(t)T}\mathbf{x}_n)}}$, Then:

$$L(\mathbf{w},\mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) - \sum_{n=1}^{N}y_n(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n + \sum_{n=1}^{N}(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n\sigma + \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n\mathbf{x}_n^T(\mathbf{w}-\mathbf{w}^{(t)})\sigma(1-\sigma)$$

$$L(\mathbf{w},\mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) + \sum_{n=1}^{N}(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n(\sigma - y_n) + \frac{\sigma(1-\sigma)}{2}\sum_{n=1}^{N}(\mathbf{w}-\mathbf{w}^{(t)})^T\mathbf{x}_n\mathbf{x}_n^T(\mathbf{w}-\mathbf{w}^{(t)})$$

To compare it with $\gamma_n^{(t)}$, we need to form a perfect square. So,

$$L(\mathbf{w}, \mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) + \sum_{n=1}^{N} \frac{\sigma(1-\sigma)}{2}[(\mathbf{w} - \mathbf{w}^{(t)})^T \mathbf{x}_n + \frac{\sigma - y_n}{\sigma(1-\sigma)}]^2 - \frac{(\sigma - y_n)^2}{2\sigma^2(1-\sigma)^2}$$

Collecting the $\mathbf{w}$ dependent terms, we get:

$$L(\mathbf{w}, \mathbf{w}^{(t)}) = \sum_{n=1}^{N} \frac{\sigma(1-\sigma)}{2}[(\mathbf{w} - \mathbf{w}^{(t)})^T \mathbf{x}_n + \frac{\sigma - y_n}{\sigma(1-\sigma)}]^2$$

Comparing with $\sum_N \gamma_n^{(t)}[\hat{y}_n^{(t)} - \mathbf{w}^T\mathbf{x}_n]^2$, we get :

$$\gamma_n^{(t)} = \frac{\sigma(1-\sigma)}{2} = \frac{\exp(\mathbf{w}^{(t)T}\mathbf{x}_n)}{(1 + \exp(\mathbf{w}^{(t)T}\mathbf{x}_n))^2}$$

$$\hat{y}_n^{(t)} = \mathbf{w}^{(t)T}\mathbf{x}_n + \frac{\sigma - y_n}{\sigma(1-\sigma)} = \mathbf{w}^{(t)T}\mathbf{x}_n + 1 + \exp(\mathbf{w}^{(t)T}\mathbf{x}_n) - \frac{y_n[1 + \exp(\mathbf{w}^{(t)T}\mathbf{x}_n)]^2}{\exp(\mathbf{w}^{(t)T}\mathbf{x}_n)}$$

σ lies between 0 and 1 and acts like a probability of any point belonging to some class. Expression for $\gamma_n^{(t)}$ is multiplication of σ and $(1-\sigma)$ and is maximum when $\sigma = (1-\sigma) = \frac{1}{2}$. This means that when n^{th} training example has very similar probability for two classes, then its importance is more.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Sajal Goyal
Roll Number: 180651
Date: November 27, 2020

QUESTION 2

Algorithm for Standard Perceptron:

```
Step 1: Initialize $w^{(0)} = 0, t = 0$
Step 2: Pick some $(\mathbf{x}_n, y_n)$ randomly
Step 3: if $y_n \mathbf{w}^{(t)T}\mathbf{x}_n < 0$ :
            $w^{(t+1)} = w^{(t)} + y_n\mathbf{x}_n$
            $t = t + 1$
Step 4: Repeat Step 2 until convergence.
```

After convergence, we get :

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

Through this, Perceptron learns only linear boundaries. In order to learn nonlinear boundaries, we replace the input $\mathbf{x}_n$ with $\phi(\mathbf{x}_n)$.
So then we get $\mathbf{w} = \sum_{j=1}^{N} \alpha_j y_j \phi(\mathbf{x}_j)$ and
Mistake rule $\Rightarrow$

$$y_n \mathbf{w}^{(t)T}\phi(\mathbf{x}_n) < 0 \Rightarrow y_n \sum_{j=1}^{N} \alpha_j y_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_n) < 0 \Rightarrow y_n \sum_{j=1}^{N} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_n) < 0$$

where $k(\mathbf{x}_j, \mathbf{x}_n)$ is the kernel function that gives dot product similarity between two inputs, $\mathbf{x}_j$ and $\mathbf{x}_n$ and does not require computing mapping of $\phi(\mathbf{x})$. We can't store $\mathbf{w}$ unless the feature mapping $\phi(\mathbf{x}_n)$ is finite dimensional and so here we will store the α_j and the training data for test time. For each mistake, we update only α_j with new value. α_j is kind of value which depicts the number of mistakes for each training example.

Algorithm for kernelized Perceptron:

```
Step 1: Initialize $\alpha_j = 0$ for all n          (Initialization)
Step 2: Pick some $(\mathbf{x}_n, y_n)$ randomly
Step 3: if $y_n \sum_{j=1}^{N} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_n) < 0$ : (Mistake Condition)
            $\alpha_n = \alpha_n + 1$                 (Update Equation)
Step 4: Repeat Step 2 until convergence.
```

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Sajal Goyal
Roll Number: 180651
Date: November 27, 2020

QUESTION 3

Standard SVM loss function:

$$L(\mathbf{w}, b, \alpha, \beta, \xi) = \frac{||\mathbf{w}||^2}{2} + C\sum_{n=1}^{N} \xi_n + \sum_{n=1}^{N} \alpha_n[1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) - \xi_n] - \sum_{n=1}^{N} \beta_n\xi_n$$

Dual problem for Standard SVM :

$$\max_{0<\alpha<C} \alpha^T\mathbf{1} - \frac{1}{2}\alpha^T G\alpha$$

Modified SVM loss function :

$$L(\mathbf{w}, b, \alpha, \beta, \xi) = \frac{||\mathbf{w}||^2}{2} + \sum_{n=1}^{N} C_{y_n}\xi_n + \sum_{n=1}^{N} \alpha_n[1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) - \xi_n] - \sum_{n=1}^{N} \beta_n\xi_n$$

$$\min_{\mathbf{w},b,\xi} \max_{\alpha>0,\beta>0} \frac{||\mathbf{w}||^2}{2} + \sum_{n=1}^{N} C_{y_n}\xi_n + \sum_{n=1}^{N} \alpha_n[1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) - \xi_n] - \sum_{n=1}^{N} \beta_n\xi_n$$

Solving the primal first, we get :

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} + \sum_{n=1}^{N} \alpha_n(-y_n\mathbf{x}_n) = 0$$

$$\Rightarrow \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

$$\frac{\partial L}{\partial \xi} = 0 \Rightarrow C_{y_n} + \alpha_n(-1) - \beta_n = 0$$

$$\Rightarrow C_{y_n} = \alpha_n + \beta_n$$

$$\frac{\partial L}{\partial \mathbf{b}} = 0 \Rightarrow \sum_{N} \alpha_n y_n = 0$$

We know that $\alpha, \beta > 0$ and $C_{y_n} = \alpha_n + \beta_n \Rightarrow$

$$0 < \alpha_n < C_{y_n}$$

Our problem simplifies to :

$$\max_{0<\alpha_n<C_{y_n}} \sum_{N} \alpha_n - \frac{1}{2}\sum_{n,m=1}^{N} \alpha_m\alpha_n y_m y_n(\mathbf{x}_m^T\mathbf{x}_n)$$

Dual problem (that depends only on the dual variables α) :

$$\max_{0<\alpha_n<C_{y_n}} \alpha^T\mathbf{1} - \frac{1}{2}\alpha^T G\alpha$$

The above solution is same as that for Standard SVM, just that range of α_n has changed. For Standard SVM, range of α_n just depends on single value C but for modified SVM, range of α_n depends on C_{+1} (when y=+1) and C_{-1} when y = -1). This is helpful in case where we cannot afford to have false negative (meaning person actually has cancer but results show that he/she does not have cancer) and so we want misclassification error of positive example to be as small as possible $\Rightarrow C_{+1} > C_{-1}$.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Sajal Goyal
Roll Number: 180651
Date: November 27, 2020

QUESTION 4

K-means objective function:

$$L = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}||\mathbf{x}_n - \mu_k||^2$$

where $z_{nk} \in \{0,1\}$ such that $z_{nk} = 1$ if $\mathbf{x}_n$ belongs to k^{th} cluster and 0 otherwise. So let the cluster assigned to input $\mathbf{x}_n$ be $k = a_n$ and so the objective function simplifies to :

$$L = \sum_{n=1}^{N} ||\mathbf{x}_n - \mu_{a_n}||^2$$

(μ_{a_n} is the centroid/mean of the cluster $k = a_n$)

Step 1 is we greedily choose the cluster for $\mathbf{x}_n$ whose Euclidean distance from centroid of $k = a_n$ is minimum.

Step 2. To update the mean, we need to optimize objective function as:

$$\frac{\partial L}{\partial \mu_{a_n}} = -2(\mathbf{x}_n - \mu_{a_n})$$

$$\Rightarrow \mu_{a_n}^{(t+1)} = \mu_{a_n}^{(t)} + \alpha(\mathbf{x}_n - \mu_{a_n}^{(t)}) \Leftrightarrow \mu_{a_n}^{(t+1)} - \mu_{a_n}^{(t)} = \alpha(\mathbf{x}_n - \mu_{a_n}^{(t)})$$

From above equation we can observe that $\mu_{a_n}^{(t)}$ gets closer to $\mathbf{x}_n$, the update gets small and μ_{a_n} converges to a finite value.

Step 3. Learning rate should be such that if there are many data points belonging to a particular cluster then step size should be small while if there are very few data points belonging to a particular cluster, then the step size should be large, that is, step size is inversely proportional to number of data points in that cluster. For the simplest case ,take :

$$\alpha(\text{Step size}) = \frac{1}{\text{number of data points in } (k = a_n) \text{ cluster}}$$

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Sajal Goyal
Roll Number: 180651
Date: November 27, 2020

QUESTION 5

For Kernel K-means algorithm, lets assume kernel $\mathbf{k}$ with an infinite dimensional feature map ϕ, we replace the input $\mathbf{x}_n$ and μ_k in the expressions of standard K-means with $\phi(\mathbf{x}_n)$ and $\phi(\mu_k)$ respectively. Let $\mathbf{C}_k$ be the set of examples assigned to cluster k, then

$$\hat{z}_k = \arg\min_k ||\mathbf{x}_n - \mu_k||^2 \Rightarrow \hat{z}_k = \arg\min_k ||\phi(\mathbf{x}_n) - \phi(\mu_k)||^2$$

$$\Rightarrow \hat{z}_k = \arg\min_k [\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_n) - 2\phi(\mathbf{x}_n)^T\phi(\mu_k) + \phi(\mu_k)^T\phi(\mu_k)]$$

$$\Rightarrow \hat{z}_k = \arg\min_k [\mathbf{k}(\mathbf{x}_n, \mathbf{x}_n) - 2\mathbf{k}(\mathbf{x}_n, \mu_k) + \mathbf{k}(\mu_k, \mu_k))]$$

and

$$\mu_k = \frac{1}{|\mathbf{C}_k|}\sum_{n\in\mathbf{C}_k} \mathbf{x}_n \Rightarrow \phi(\mu_k) = \frac{1}{|\mathbf{C}_k|}\sum_{n\in\mathbf{C}_k} \phi(\mathbf{x}_n)$$

Combining the above two equations, we get :

$$\hat{z}_k = \arg\min_k [\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_n) - \frac{2}{|\mathbf{C}_k|}\sum_{i\in\mathbf{C}_k}\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_i) + \frac{1}{|\mathbf{C}_k|^2}\sum_{i,j\in\mathbf{C}_k}\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)]$$

$$\hat{z}_k = \arg\min_k [\mathbf{k}(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{|\mathbf{C}_k|}\sum_{i\in\mathbf{C}_k}\mathbf{k}(\mathbf{x}_n, \mathbf{x}_i) + \frac{1}{|\mathbf{C}_k|^2}\sum_{i,j\in\mathbf{C}_k}\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)]$$

Algorithm 1 Kernel K-means Algorithm

```
0: for $k \leftarrow 1$ to $K$ do
0:     $N_k = 0$
0: end for
0: Choose K random $\mu_k$ vectors                                   //Initialization step
0: for $n \leftarrow 1$ to $N$ do
0:     $\hat{z}_n = \arg\min_k [\mathbf{k}(\mathbf{x}_n, \mathbf{x}_n) - 2\mathbf{k}(\mathbf{x}_n, \mu_k) + \mathbf{k}(\mu_k, \mu_k))]$      //Cluster assignment
0:     $N_{\hat{z}_n} = N_{\hat{z}_n} + 1$
0: end for
0: for $n \leftarrow 1$ to $N$ do
0:     $k_{prev} = \hat{z}_n$                                         //Storing previous Cluster
0:     $\hat{z}_k = \arg\min_k [\mathbf{k}(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{N_k}\sum_{i:z_i=k}\mathbf{k}(\mathbf{x}_n, \mathbf{x}_i) + \frac{1}{N_k^2}\sum_{i,j:z_i=z_j=k}\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)]$      //Update
0:     if $k_{prev} \neq \hat{z}_k$ then
0:         $N_{k_{prev}} = N_{k_{prev}} - 1$
0:         $N_{\hat{z}_n} = N_{\hat{z}_n} + 1$
0:     end if
0: end for
   Repeat above for loop until change in values tending to =0
```

Let $N_k = |\mathbf{C}_k|$. Then :

$$\hat{z}_k = \arg\min_k [\mathbf{k}(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{\mathbf{N}_k} \sum_{i:z_i=k} \mathbf{k}(\mathbf{x}_n, \mathbf{x}_i) + \frac{1}{\mathbf{N}_k^2} \sum_{i,j:z_i=z_j=k} \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)]$$

In standard K-means algorithm, we can store all mean vectors of different clusters. While for kernel K-means, Cluster mean, which is given below, cannot be stored as ϕ is infinite dimensional. So we have chosen random μ_k as initialization.

$$\mu_k = \phi^{-1}(\frac{1}{N_k} \sum_{n:z_n=k} \phi(\mathbf{x}_n))$$

Finally, given N inputs of D dimension, Computation cost per Cluster
for Standard K-Means := $O(ND)$
for kernel K-Means := $O(N^3 D)$

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Sajal Goyal
Roll Number: 180651
Date: November 27, 2020

QUESTION 6

Library used for SVM plots ⇒ scikit-learn
First four plots are made using Generative models while last two plots are made using SVM model

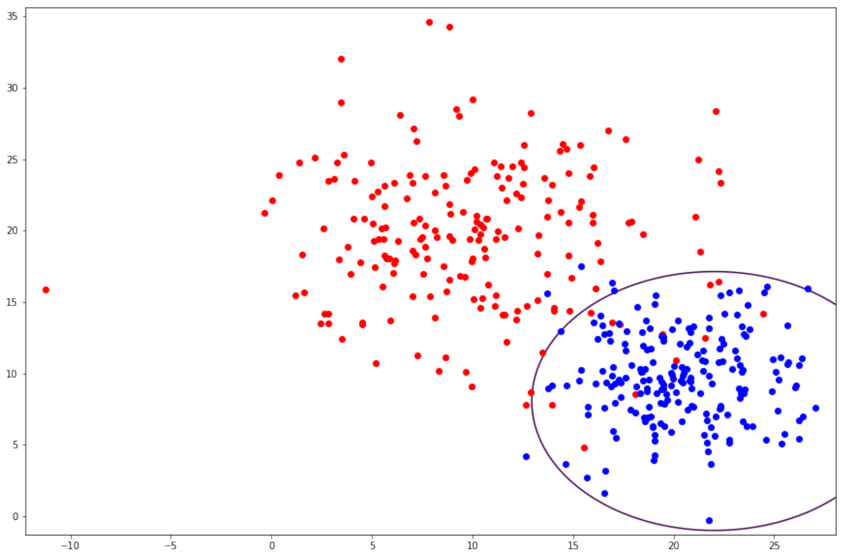

Figure 1: Decision boundary for binclass.txt data for different class covariances

We can see that Generative model with different Gaussian class conditionals perform better than SVM model because SVM model will try to separate points with a linear line while our data is linearly inseparable. So Generative model with different Gaussian class conditionals is better for both datasets. SVM will be better in case when data is linearly separable.

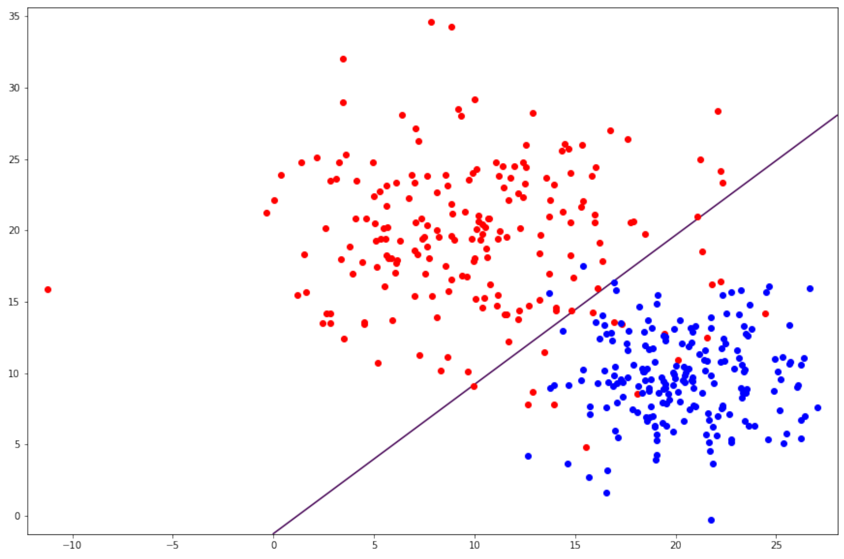

Figure 2: Decision boundary for binclass.txt data for equal class covariances

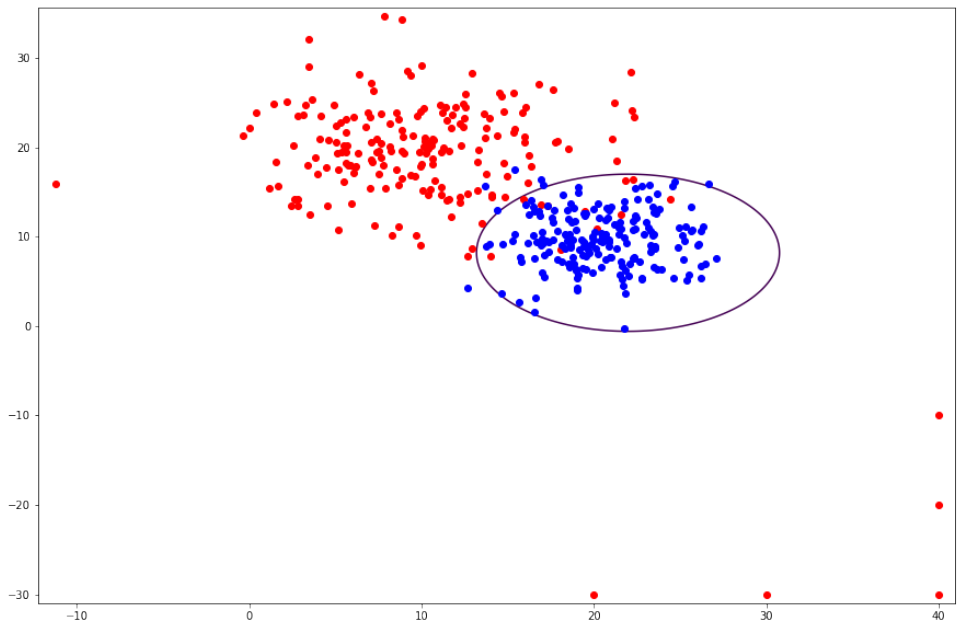

Figure 3: Decision boundary for binclassv2.txt data for different class covariances

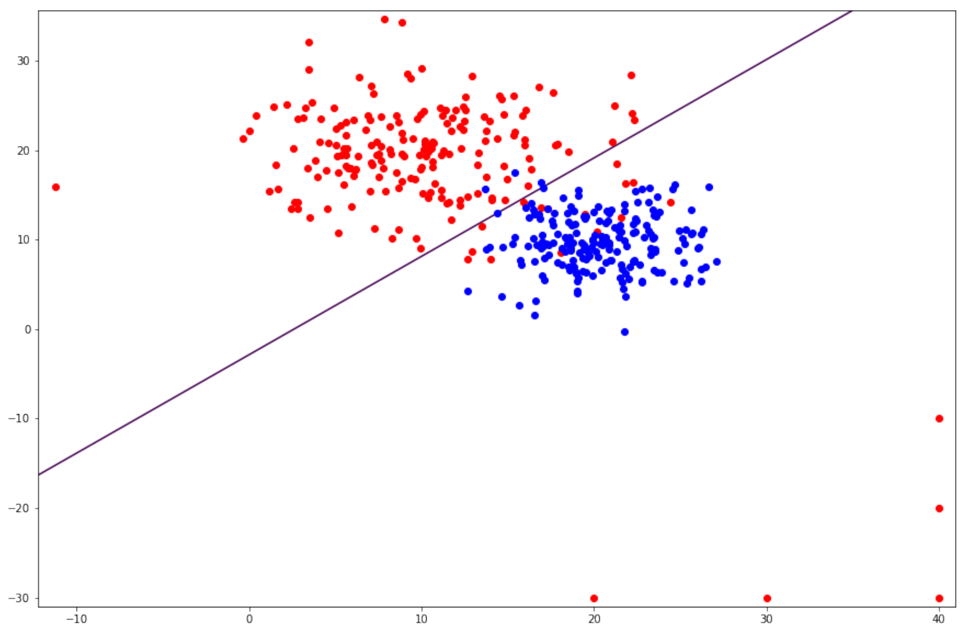

Figure 4: Decision boundary for binclassv2.txt data for equal class covariances

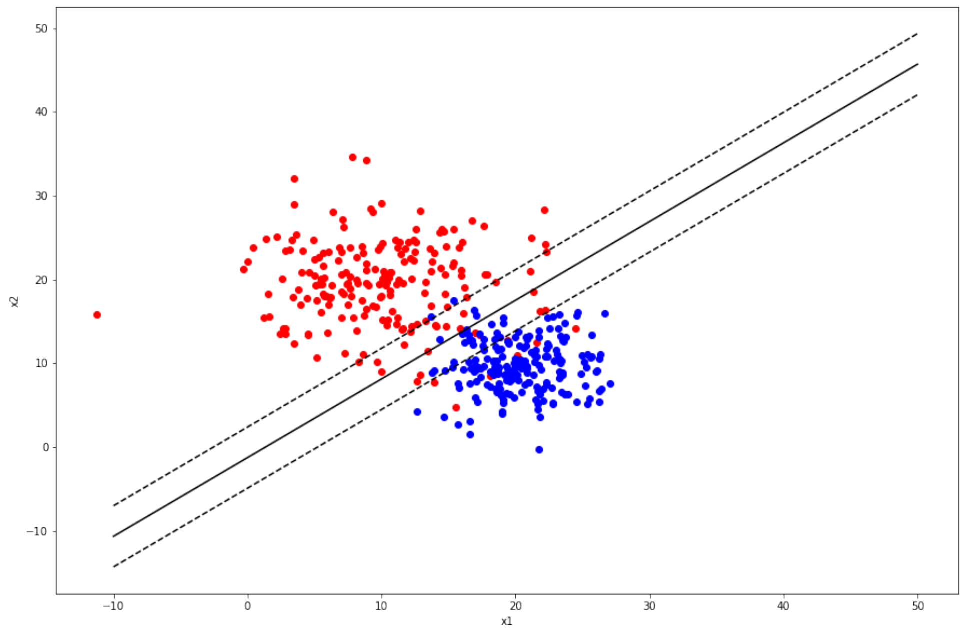

Figure 5: Decision boundary for binclass.txt data for linear SVM

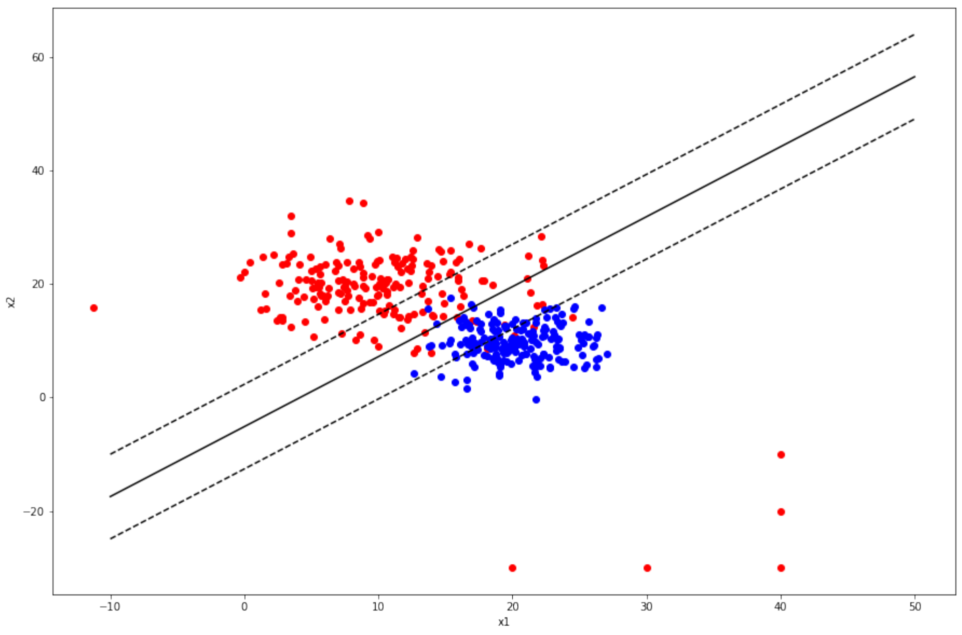

Figure 6: Decision boundary for binclassv2.txt data for linear SVM