Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Om Singh
Roll Number: 210686
Date: September 15, 2023

QUESTION 1

Sneak Peek:

In the end, we shall demonstrate that the loss function under consideration in this problem is linked to the **Learning with Prototypes** model. Instead of Euclidean distances used in the class, it just uses Mahalanobis distance. If $\mathbf{M}_c$ is taken to be the identity matrix I, it reduces to the usual Euclidean distance function, as we shall conclude in the end.

Setting up the problem:

To solve this optimisation problem, we need to find simultaneous solution to the equations: $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_c} = 0$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{M}_c} = 0$, where $\mathcal{L} = \sum \frac{1}{N_c}(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c (\mathbf{x}_n - \mathbf{w}_c) - log|\mathbf{M}_c|$., as per the first-order optimality conditions.

It can be shown that these optima points represent a minima, and not maxima or saddle point, by checking the second derivatives (excluded from our present analysis).

Solving for the optimality with respect to $\mathbf{w}_c$:

As per the problem statement, it is clear that $\mathbf{M}_c$, $\mathbf{x}_n$ and N_c are independent of $\mathbf{w}_c$. Also, as $\mathbf{M}_c$ is positive definite and real-valued, it must be a symmetric matrix. So,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}_c} &= \frac{\partial}{\partial \mathbf{w}_c}[\sum_{\mathbf{x}_n : y_n = c} \frac{1}{N_c}(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c (\mathbf{x}_n - \mathbf{w}_c) - log|\mathbf{M}_c|] \\
&= \frac{\partial}{\partial \mathbf{w}_c} \sum_{\mathbf{x}_n : y_n = c} \frac{1}{N_c}(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c (\mathbf{x}_n - \mathbf{w}_c) \\
&= \frac{1}{N_c} \sum_{\mathbf{x}_n : y_n = c} \frac{\partial}{\partial \mathbf{w}_c}[(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c (\mathbf{x}_n - \mathbf{w}_c)]
\end{aligned} \tag{1}$$

Using the chain rule of differentiation,

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}_c}[(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c (\mathbf{x}_n - \mathbf{w}_c)] &= \frac{\partial}{\partial (\mathbf{w}_c - \mathbf{x}_n)}[(\mathbf{w}_c - \mathbf{x}_n)^T \mathbf{M}_c (\mathbf{w}_c - \mathbf{x}_n)].\frac{\partial (\mathbf{w}_c - \mathbf{x}_n)}{\partial \mathbf{w}_c} \\
&= 2\mathbf{M}_c(\mathbf{w}_c - \mathbf{x}_n)
\end{aligned} \tag{2}$$

Substituting this back in (1) and using the fact that $|\mathbf{M}_c| \neq 0$, we get:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_c} = \frac{2\mathbf{M}_c}{N_c} \sum_{\mathbf{x}_n : y_n = c} (\hat{\mathbf{w}}_c - \mathbf{x}_n) = 0 \tag{3}$$

$$\implies \sum_{\mathbf{x}_n : y_n = c} (\hat{\mathbf{w}}_c - \mathbf{x}_n) = 0 \implies \sum_{\mathbf{x}_n : y_n = c} \hat{\mathbf{w}}_c = \sum_{\mathbf{x}_n : y_n = c} \mathbf{x}_n \implies N_c \hat{\mathbf{w}}_c = \sum_{\mathbf{x}_n : y_n = c} \mathbf{x}_n \tag{4}$$

So, we finally get:

$$\boxed{\hat{\mathbf{w}}_c = \frac{1}{N_c} \sum_{\mathbf{x}_n : y_n = c} \mathbf{x}_n} \tag{5}$$

Solving for the optimality with respect to $\mathbf{M}_c$ (continued on the next page):

Differentiating with respect to $\mathbf{M}_c$, we get:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{M}_c} &= \frac{\partial}{\partial \mathbf{M}_c}[\sum_{\mathbf{x}_n:y_n=c} \frac{1}{N_c}(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c(\mathbf{x}_n - \mathbf{w}_c) - log|\mathbf{M}_c|] \\ &= \frac{\partial}{\partial \mathbf{M}_c}\sum_{\mathbf{x}_n:y_n=c} \frac{1}{N_c}(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c(\mathbf{x}_n - \mathbf{w}_c) - \frac{\partial}{\partial \mathbf{M}_c} log|\mathbf{M}_c|\end{aligned} \tag{6}$$

Using the properties of differentiation with respect to matrices when the matrix is symmetric, we finally get:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{M}_c} &= \frac{1}{N_c}\sum_{\mathbf{x}_n:y_n=c} \frac{\partial}{\partial \mathbf{M}_c}[(\mathbf{x}_n - \mathbf{w}_c)^T \mathbf{M}_c(\mathbf{x}_n - \mathbf{w}_c)] - \frac{\partial}{\partial \mathbf{M}_c} log|\mathbf{M}_c| \\ &= \frac{1}{N_c}\sum_{\mathbf{x}_n:y_n=c} [(\mathbf{x}_n - \hat{\mathbf{w}}_c)(\mathbf{x}_n - \hat{\mathbf{w}}_c)^T] - \hat{\mathbf{M}}_c^{-1} = 0\end{aligned} \tag{7}$$

$$\implies \hat{\mathbf{M}}_c^{-1} = \frac{1}{N_c}\sum_{\mathbf{x}_n:y_n=c} (\mathbf{x}_n - \hat{\mathbf{w}}_c)(\mathbf{x}_n - \hat{\mathbf{w}}_c)^T \tag{8}$$

So, we finally get the following expression for $\hat{\mathbf{M}}_c$:

$$\begin{aligned}\hat{\mathbf{M}}_c &= (\frac{1}{N_c}\sum_{\mathbf{x}_n:y_n=c} (\mathbf{x}_n - \hat{\mathbf{w}}_c)(\mathbf{x}_n - \hat{\mathbf{w}}_c)^T)^{-1} \\ \implies & \boxed{\hat{\mathbf{M}}_c = N_c(\sum_{\mathbf{x}_n:y_n=c} (\mathbf{x}_n - \hat{\mathbf{w}}_c)(\mathbf{x}_n - \hat{\mathbf{w}}_c)^T)^{-1}}\end{aligned} \tag{9}$$

where $\mathbf{w}_c$ can be found from equation (5).

Final equations:

The optimal values of $\mathbf{w}_c$ and $\mathbf{M}_c$ for the given objective function are given as:

$$\hat{\mathbf{w}}_c = \frac{1}{N_c}\sum_{\mathbf{x}_n:y_n=c} \mathbf{x}_n \tag{10}$$

$$\hat{\mathbf{M}}_c = N_c(\sum_{\mathbf{x}_n:y_n=c} (\mathbf{x}_n - \hat{\mathbf{w}}_c)(\mathbf{x}_n - \hat{\mathbf{w}}_c)^T)^{-1} \tag{11}$$

Special case: When $\mathbf{M}_c$ is an identity matrix:

This model reduces to **Learning with Prototypes**!

In the special case when $\mathbf{M}_c$ is the identity matrix I, the objective function to be optimised becomes:

$$\mathcal{L} = \sum_{\mathbf{x}_n:y_n=c} \frac{1}{N_c}(\mathbf{x}_n - \mathbf{w}_c)^T \check{(}\mathbf{x}_n - \mathbf{w}_c) - 0 = \frac{1}{N_c}\sum_{\mathbf{x}_n:y_n=c} (\mathbf{x}_n - \mathbf{w}_c)^T \check{(}\mathbf{x}_n - \mathbf{w}_c) \tag{12}$$

as the determinant of the matrix becomes 1. Here, we can see that the optimisation problem now is independent of the matrix $\mathbf{M}_c$.

Now, the quantity under summation is simply the sum of distances of x_n from the the class vectors $\mathbf{w}_c$. If we treat w_c as being the mean of the class c being used in LwP model, we see a direct correspondence!

Even for the general case, it is same as the Learning with Prototypes model. It just uses a different distance function (Mahalanobis distance) instead of the Euclidean distance we employed in the class.

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Om Singh
Roll Number: 210686
Date: September 15, 2023

QUESTION 2

Short Answer: Yes, the one-nearest neighbour classification model is consistent in the given setting.

Detailed Explanation

Setting up the problem:

Let's assume that the training data contains the points $(\mathbf{x}_n, y_n)$, where v represents the training input and y_n represents its true, noise-free label. Let the actual target function be $f(x)$, such that $\forall \mathbf{x}_n, f(\mathbf{x}_n) = y_n$.

Let's discuss the probability of a point being misclassified:

Say the nearest neighbour of $\mathbf{x}$ for some test input $\mathbf{x}$ is given by $\mathbf{x}_{NN}$ among all the training inputs, based on some previously chosen distance function.

If the one-nearest neighbour algorithm is consistent, we must have $\mathbb{P}\left[f(\mathbf{x}) \neq f(\mathbf{x}_{NN})\right] \to 0$ as the number of training inputs increase (mathematically, as $n \to \infty$).

Checking for consistency:

Let the distance function being used be the Euclidean distance function. Also, let's assume the distance between $\mathbf{x}$ and $\mathbf{x}_{NN}$ is ϵ. Now, as $\mathbf{x}_{NN}$ is nearest to $\mathbf{x}$, we can be sure that:

$\forall i, ||\mathbf{x}_i - \mathbf{x}||_2 \geq \epsilon$, where equality exists iff $i = NN$. Now, we get:

$$\begin{aligned}\mathbb{P}[||\mathbf{x}_{NN} - \mathbf{x}||_2 > \epsilon] &= \mathbb{P}[\forall i, ||\mathbf{x}_i - \mathbf{x}||_2 > \epsilon] \\ &= \prod_{i=1}^{n} \mathbb{P}[||\mathbf{x}_i - \mathbf{x}||_2 > \epsilon]\end{aligned} \tag{13}$$

This $\to 0$ as $n \to \infty$, as values of all the probability can be at most 1. So, its conjugate probability,

$$\boxed{\mathbb{P}[||\mathbf{x}_{NN} - \mathbf{x}||_2 < \epsilon] \to 1; (n \to \infty)}.$$

With that, we can clearly conclude that as we increase the amount of training data, $\mathbf{x}_{NN}$ comes closer and closer to the test point $\mathbf{x}$, and hence, the one-nearest neighbour algorithm approximates the class of that point arbitrarily well, because the label for $\mathbf{x}_{NN}$ is noise-free.

Conclusion:

As we just saw, if the amount of training data is infinite in a noise-free setting, Bayes optimal error rate is zero. So, as per the definition of consistency for a Machine-Learning algorithm, in the setting of the problem, the one-nearest neighbour algorithm is consistent.

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Om Singh
Roll Number: 210686
Date: September 15, 2023

QUESTION 3

In case of Decision Trees, one of the parameters that can be used to quantify the purity in regression settings is the **Mean Squared Error (MSE in short)**.

Defining the quantity:

Let us assume our training set S to be of the form $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n$ is the training input, and y_n is its corresponding real-valued label. We can choose any measure of central tendency we want in the following discussion. Two of the most common ones are arithmetic mean and median. For the sake of further discussion, I am sticking with arithmetic mean unless otherwise specified. But, this can easily be extended and generalised for median as well.

Now, let's assume S' is some subset of the training set S. Then, we define the mean squared error (MSE) of S' as:

$$\mathbf{M.S.E.}(S') = \frac{1}{n(S')} \sum_{n,(x_n,y_n)\epsilon S'} (y_n - \hat{y})^2$$

where $\hat{y}$ is the arithmetic mean or median of the y_n's present in the set S', and $n(.)$ .is the cardinal function, which denotes the number of elements present in a set.

Choosing the feature and why it works?:

In the case of classification using Decision Trees, we used to choose the feature corresponding to the highest information gain. Similarly, for regression, we can choose the value of our independent variable as the value, at which the split performed ensures the resulting subsets have the least possible mean squared errors.

Conclusion:

So, the criteria to choose a feature to spilt on can be the Mean Squared Error as defined above, and the feature can be chosen in a way that minimised this M.S.E. error as we go from the parent node to the children node.

Choose the feature which maximises:

$$\boxed{\mathbf{M.S.E.}(S) - \frac{n(S')}{n(S)}\mathbf{M.S.E.}(S') - \frac{n(S) - n(S')}{n(S)}\mathbf{M.S.E.}(S - S')}$$

where S' is some subset of S, which is the original parent set.

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Om Singh
Roll Number: 210686
Date: September 15, 2023

QUESTION 4

Notations and setting up the problem:

Let us assume that we have N training examples $(\mathbf{x}_n, y_n)$, where each x_n is a d-dimensional vector. To keep the expressions simple by incorporating a bias term in the expression as well, let us assume that each input vector has an extra coordinate too, labelled $x_{n,0}$ such that $x_{n,0} = 1 \forall n$.

Now, say the given test point is $\mathbf{x}_*$. As per the closed form solution obtained in Unregularised Linear Regression by optimising the squared error loss function, the optimum weight vector comes out to be $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

Solving for the prediction:

n^{th} component of our optimum weight vector, $\hat{\mathbf{w}}_n$:

$$\hat{\mathbf{w}}_n = [(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}]_n = \sum_{n=1}^{N} e_{in}y_n \tag{14}$$

where e_{in} is the i^{th} row, n^{th} column element of the matrix $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.

$$\begin{aligned} f(\mathbf{x}_*) &= \sum_{d=0}^{D} \hat{\mathbf{w}}_d \mathbf{x}_{*,d} = \sum_{d=0}^{D}[\sum_{n=1}^{N} e_{dn}y_n]x_{*,d} \\ &= \sum_{d=0}^{D}\sum_{n=1}^{N} e_{dn}y_n x_{*,d} = \sum_{n=1}^{N}[\sum_{d=0}^{D} e_{dn}x_{*,d}]y_n \end{aligned} \tag{15}$$

As per the problem, we have $f(x_*) = \sum w_n y_n$. Comparing, we get the expression for w_n to be:

$$\boxed{w_n = \sum_{d=0}^{D} e_{dn}x_{*,d}}. \tag{16}$$

Symbols have their usual meaning, as aforementioned.

Comparison with the weights in weighted k-Nearest Neighbours:

Weights in a weighted version of k-Nearest Neighbours are such that $w_n = \frac{C}{d(\mathbf{x}_n,\mathbf{x}_*)}$, where C: constant, and $d(\mathbf{x}_i, \mathbf{x}_j)$ is a distance function.

Some of the ways in which the weights in unregularised linear regression are different from the weights in weighted k-Nearest Neighbours are as follows:

- In the case of weighted k-Nearest Neighbours, weight for a particular training point contains the information of that particular training point only. However, in the unregularised Linear Regression case, the weight vector depends on the entire set of training examples, as the expression contains the matrix $\mathbf{X}$.
- In the case of weighted k-Nearest Neighbours, if we hold all the other parameters constant, weight w_n decreases as $||\mathbf{x}_*||_2$ increases $\forall n$. On the other hand, in the case of unregularised Linear Regression, weight vector $[w_n]_{n=1}^{N}$ increases as $||\mathbf{x}_*||_2$ increases, as $\mathbf{x}_*$ appears in the numerator.

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Om Singh
Roll Number: 210686
Date: September 15, 2023

QUESTION 5

Setting up the problem:

In the upcoming discussion, $\mathbb{E}(.)$ refers to the expected value of the random variable enclosed in the brackets. Let the number of training examples be N. Then, the loss function which has to be minimised:

$$\mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} (y_n - \mathbf{w}^T \tilde{\mathbf{x}_n})^2$$

Here, $\tilde{\mathbf{x}_n} = \mathbf{x}_n \odot \mathbf{m}_n$, where $\mathbf{x}_n$ is the original training input and $\mathbf{m}_n$ is a $D \times 1$ binary masking vector with $m_{nd} = \text{Bernoulli}(p)$.

Solving for the expectation value:

Let us assume that T_n denotes the n^{th} term in the loss function above. Thus, we have $T_n = (y_n - \mathbf{w}^T \mathbf{x}_n)^2$. To find its expectation value, we need to express the dot product explicitly. Further, as there is not bias term, we can add an extra term in $\mathbf{w}$ and choose $\mathbf{x}_0 = 1$, and expand this expression as:

$$\mathbb{E}(T_n) = \mathbb{E}[(y_n - \mathbf{w}^T \mathbf{x_n})^2] = \mathbb{E}[(y_n - \sum_{i=0}^{D} w_i x_{n_i})^2] \tag{17}$$

where v_i denotes the i^{th} component of the vector $\mathbf{v}$.

Using the linearity of expectation, we have:

$$\mathbb{E}(T_n) = y_n^2 + \sum_{i=0}^{D} \mathbb{E}(w_i^2 x_{n_i}^2) - 2y_n \sum_{i=0}^{D} \mathbb{E}(w_i x_{n_i}) + 2 \sum_{0 \le i < j \le D} \mathbb{E}(w_i w_j x_{n_i} x_{n_j}) \tag{18}$$

Now, $\mathbb{E}(x_{n_i}^2) = p x_{n_i}^2$ as per the problem, because x_{n_i} is a Bernoulli random variable, and the probability of it being non-zero is p. Similarly, $\mathbb{E}(x_{n_i}) = p x_{n_i}$ and $\mathbb{E}(x_{n_i} x_{n_j}) = p^2 x_{n_i} x_{n_j}$.

Substituting all of these into the equation, we get:

$$\begin{aligned} \mathbb{E}(T_n) &= y_n^2 + p \sum_{i=0}^{D} (w_i^2 x_{n_i}^2) - 2p y_n \sum_{i=0}^{D} (w_i x_{n_i}) + 2p^2 \sum_{0 \le i < j \le D} w_i w_j x_{n_i} x_{n_j} \\ &= (y_n - p\mathbf{w}^\intercal \mathbf{x}_n)^2 + (p - p^2) \|\mathbf{w} \odot \mathbf{x}_n\|_2^2 \end{aligned} \tag{19}$$

Now to find the expected value of the entire loss function, we just need to find the sum of expected values of the individual terms T_i (using linearity of expectation):

$$\begin{aligned} \mathbb{E}(\mathcal{L}(\mathbf{w})) &= \sum_{n=1}^{N} (y_n - p\mathbf{w}^\intercal \mathbf{x}_n)^2 + (p - p^2) \sum_n \|\mathbf{w} \odot \mathbf{x}_n\|_2^2 \\ &= \sum_n (y_n - p\mathbf{w}^\intercal \mathbf{x}_n)^2 + \lambda R(\mathbf{w}) \end{aligned} \tag{20}$$

where $\lambda = p - p^2 \ge 0$ [as $p \epsilon (0, 1)$] is the regulariser parameter and $R(\mathbf{w}) = \sum_{n=1}^{N} \|\mathbf{w} \odot \mathbf{x}_n\|_2$ is a regulariser function.

Concluding remarks:

Therefore, minimising the expected value of the loss function $\mathcal{L}$ using masked inputs is equivalent to minimising the regularised loss function:

$$\boxed{\mathcal{L}^{'}(\mathbf{w}) = \sum_{n=1}^{N}(y_n - p\mathbf{w}^{\intercal}\mathbf{x}_n)^2 + \lambda R(\mathbf{w})} \tag{21}$$

where $\lambda = p - p^2$ and $R(\mathbf{w}) = \sum_{n=1}^{N} \|\mathbf{w} \odot \mathbf{x}_n\|_2$.

Here, p represents the probability that the feature under consideration is being retained under the Bernoulli mask!

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Om Singh
Roll Number: 210686
Date: September 15, 2023

QUESTION 6

Refer to the code for this problem to get to understand the implementation clearly. I have commented the major steps in the code, and have kept the code as modular as possible for good readability.

The README.md file contains all the information on where to place the files convex.py and regress.py, so that the path from where the data is being loaded remains valid. If the files are placed otherwise, the user shall require to change the path appropriately.

Test-Set Classification Accuracy for Convex Combination Method:

In this case, the testing set accuracy was found out to be: 46.89 %. This is in relation with the model's predictions against the ground-truth labels. 46.89 % of the total testing inputs were classified correctly!

Test-Set Classification Accuracy for Multi-Output Regression Method:

In this case, testing set accuracy was found out to be the following:

$\lambda = 0.01$ gives Testing-Set Accuracy of 58.09%.
$\lambda = 0.1$ gives Testing-Set Accuracy of 59.55%.
$\lambda = 1.0$ gives Testing-Set Accuracy of 67.39%.
$\lambda = 10.0$ gives Testing-Set Accuracy of 73.28%.
$\lambda = 20.0$ gives Testing-Set Accuracy of 71.68%.
$\lambda = 50.0$ gives Testing-Set Accuracy of 65.08%.
$\lambda = 100.0$ gives Testing-Set Accuracy of 56.47%.

All these accuracies are in relation to the ground-truth labels.

As is clear for the aforementioned values of λ, the best test-set classification accuracy is achieved at $\lambda = 10.0$ among the values tested. The accuracy achieved at this λ is 73.28%.

Other values of λ don't generalise as good as $\lambda = 10.0$.

For more details on the implementation and execution, read the README.md file, and check the comments present in the uploaded code.