Assignment 3
AI2000
Foundations of Machine Learning

Darpan Gaur
CO21BTECH11004

Problem 1

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N} \sigma_n (y_n - w^T \phi(x_n))^2$$

Given dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. Let $X = \begin{bmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_N) \end{bmatrix}^T$ and $Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$,

where $\phi(x_n)$ is the feature vector of x_n.

Error in matrix form:

$$E_D(w) = \frac{1}{2}(Y - Xw)^T \Sigma (Y - Xw)$$

where $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_N)$, is a diagonal matrix with σ_n on the diagonal.

Differentiating $E_D(w)$ w.r.t. w:

$$\frac{\partial E_D(w)}{\partial w} = \frac{1}{2}\frac{\partial}{\partial w}(Y - Xw)^T \Sigma (Y - Xw)$$

Using the property $\frac{\partial}{\partial w} x^T A x = x^T (A + A^T) \frac{\partial x}{\partial w}$, we get:

$$\frac{1}{2}\frac{\partial}{\partial w}(Y - Xw)^T \Sigma (Y - Xw) = \frac{1}{2}(Y - Xw)^T (\Sigma + \Sigma^T) \frac{\partial}{\partial w}(Y - Xw)$$

$$\implies \frac{\partial E_D(w)}{\partial w} = -(Y - Xw)^T \Sigma X$$

Putting $\frac{\partial E_D(w)}{\partial w} = 0$:

$$\implies -(Y - Xw)^T \Sigma X = 0$$

$$\implies X^T \Sigma Y = X^T \Sigma X w$$

$$\boxed{w = (X^T \Sigma X)^{-1} X^T \Sigma Y}$$

Problem 2

$$E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} log y_k(x_n, w)$$

For a given input x_n, Differentiating $E(w)$ w.r.t. a_k:

$$\frac{\partial E(w)}{\partial a_k} = -\sum_{k=1}^{K} \frac{\partial}{\partial a_k} t_k log y_k(x_n, w) = -\sum_{k=1}^{K} \frac{\partial}{\partial y_k} t_k log y_k(x_n, w) \frac{\partial y_k}{\partial a_k} = -\sum_{k=1}^{K} \frac{t_k}{y_k} \frac{\partial y_k}{\partial a_k}$$

$$\frac{\partial E(w)}{\partial a_k} = -\sum_{\substack{i=1 \\ i \neq k}}^{K} \frac{t_i}{y_i} \frac{\partial y_i}{\partial a_k} - \frac{t_k}{y_k} \frac{\partial y_k}{\partial a_k}$$

Differentiating y_i w.r.t. a_k:

$$\frac{\partial y_i}{\partial a_k} = \frac{\partial}{\partial a_k} \frac{e^{a_i}}{\sum_{j=1}^{K} e^{a_j}} = \frac{-e^{a_i} e^{a_k}}{(\sum_{j=1}^{K} e^{a_j})^2} = -y_i y_k$$

$$\frac{\partial y_k}{\partial a_k} = \frac{\partial}{\partial a_k} \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} = \frac{e^{a_k} \sum_{j=1}^{K} e^{a_j} - e^{a_k} e^{a_k}}{(\sum_{j=1}^{K} e^{a_j})^2} = y_k(1 - y_k)$$

Putting the values of $\frac{\partial y_i}{\partial a_k}$ and $\frac{\partial y_k}{\partial a_k}$ in $\frac{\partial E(w)}{\partial a_k}$:

$$\frac{\partial E(w)}{\partial a_k} = -\sum_{\substack{i=1 \\ i \neq k}}^{K} \frac{t_i}{y_i}(-y_i y_k) - \frac{t_k}{y_k} y_k(1 - y_k)$$

$$\implies \frac{\partial E(w)}{\partial a_k} = \sum_{\substack{i=1 \\ i \neq k}}^{K} t_i y_k + t_k y_k - t_k$$

$$\implies \frac{\partial E(w)}{\partial a_k} = \sum_{i=1}^{K} t_i y_k - t_k = y_k \sum_{i=1}^{K} t_i - t_k$$

$$\boxed{\frac{\partial E(w)}{\partial a_k} = y_k - t_k}$$

Problem 3

Given convex function $f(x) = x^2$, let $y_m(x) - f(x) = r_m$. As $y_m(x)$ is constant for a given x, we can say r_m is also convex.

$$E_{AV} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[(y_m(x) - f(x))^2] = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[r_m^2] = \mathbb{E}_x[\frac{1}{M}\sum_{m=1}^{M}r_m^2]$$

$$E_{ENS} = \mathbb{E}_x[(\frac{1}{M}\sum_{m=1}^{M}y_m(x) - f(x))^2] = \mathbb{E}_x[(\frac{1}{M}\sum_{m=1}^{M}r_m)^2]$$

We know that,

$$(\sum_{m=1}^{M}r_m)^2 = \sum_{m=1}^{M}r_m^2 + 2\sum_{\substack{i=1\\i\neq j}}^{M}r_i r_j$$

$$\implies (\sum_{m=1}^{M}r_m)^2 \leq \sum_{m=1}^{M}r_m^2$$

$$\implies \mathbb{E}_x[(\frac{1}{M}\sum_{m=1}^{M}r_m)^2] \leq \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[r_m^2]$$

$$\boxed{E_{ENS} \leq E_{AV}}$$

For general convex error function $Err(y_m(x), \hat{y}_m(x))$, by jensen's inequality

$$Err(\frac{1}{M}\sum_{m=1}^{M}y_m(x), f(x)) \leq \frac{1}{M}\sum_{m=1}^{M}Err(y_m(x), f(x))$$

Taking expectation on both sides:

$$\boxed{\mathbb{E}_x[Err(\frac{1}{M}\sum_{m=1}^{M}y_m(x), f(x))] \leq \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_x[Err(y_m(x), f(x))]}$$

Problem 4

Given,

$$y(x, w) = w_0 + \sum_{k=1}^{D}w_k x_k = x^T w$$

Now adding gaussian noise $\epsilon_k \sim \mathcal{N}(0, \sigma^2)$ to x_k:

$$y(x, w) = w_0 + \sum_{k=1}^{D}w_k(x_k + \epsilon_k) = (x + \epsilon)^T w$$

Sum of squared loss:

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y(x_i, w) - t_i)^2$$

Analyzing the effect of noise on $E(w)$, we see expected squared loss, where expectaion taken over ϵ:

$$L(w) = \mathbb{E}_\epsilon[\frac{1}{2}\sum_{i=1}^{N}((x_i + \epsilon_i)^T w - t_i)^2] = \frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_\epsilon[((x_i^T w - t_i) + \epsilon_i^T w)^2]$$

$$\implies L(w) = \frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_\epsilon[(x_i^T w - t_i)^2 + 2\epsilon_i^T w(x_i^T w - t_i) + w^T \epsilon_i \epsilon_i^T w]$$

$$\implies L(w) = \frac{1}{2}\sum_{i=1}^{N}[(x_i^T w - t_i)^2 + 2\,\mathbb{E}_\epsilon(\epsilon_i^T)w(x_i^T w - t_i) + w^T\,\mathbb{E}_\epsilon(\epsilon_i \epsilon_i^T)w]$$

As $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, we have $\mathbb{E}_\epsilon(\epsilon_i) = 0$ and $\mathbb{E}_\epsilon(\epsilon_i \epsilon_i^T) = \sigma^2 I$.

$$\implies L(w) = \frac{1}{2}\sum_{i=1}^{N}[(x_i^T w - t_i)^2 + w^T \sigma^2 I w]$$

$$L(w) = \frac{1}{2}\sum_{i=1}^{N}(x_i^T w - t_i)^2 + \frac{\sigma^2}{2} w^T w \tag{1}$$

For $L2$ regularization,

$$L(w) = \frac{1}{2}\sum_{i=1}^{N}(x_i^T w - t_i)^2 + \lambda w^T w \tag{2}$$

By comparing (1) and (2), we see that adding gaussian noise to input is equivalent to $L2$ regularization with $\lambda = \frac{\sigma^2}{2}$.

Problem 5

Part (a)

Custom implementation RF:

- Time taken: 15.8 seconds
- Accuracy: 92.614 %

Sklearn RF:

- Time taken: 0.026 seconds
- Accuracy: 93.483 %

Part (b)

Plot of accuracy vs number of features is shown in Figure 1. Accuracy initially increase with number of features, but after a certain point, it started oscillating.

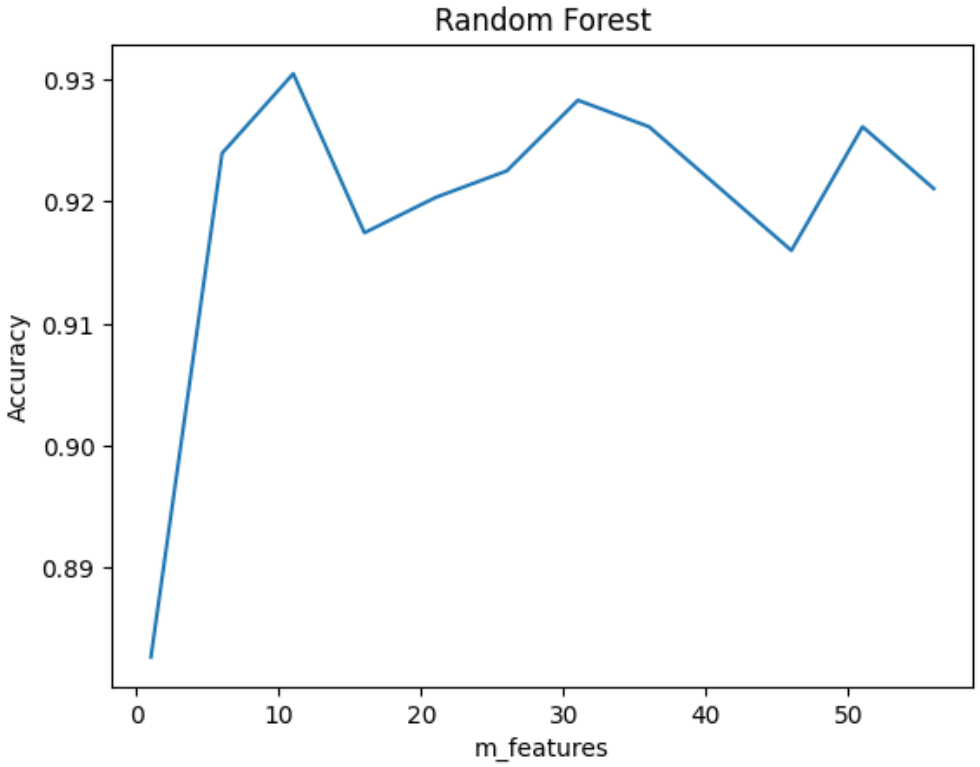

Figure 1: Accuracy vs Number of features

Part (c)

Plot of OOB error vs number of trees is shown in Figure 2. Here number of feature (m) set to 11.

- OOB error (0.092-0.10) is slightly higher than the test error (0.058-0.075).
- Both OOB error and test error are oscillating with number of trees.

Problem 6

Part (a)

Preprocessing steps:

- Find the null value percentage in each column, and removed coulmns with null value percentage greater than 20%.

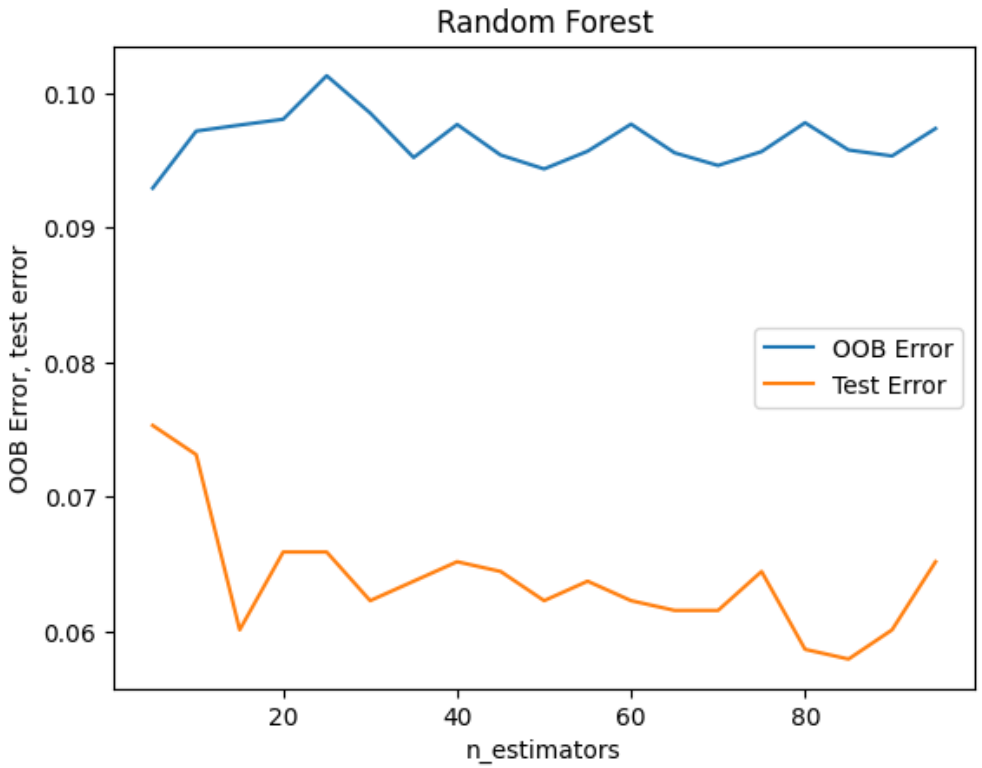

Figure 2: OOB error vs Number of trees

- Deal with columns having data type as object:
  - If column values can be directly converted to float, then convert them. Eg: int_rate have percentage values.
  - If unique values in column are less (≤ 12) and are useful convert them to one-hot encoding. Eg: grade have values A, B, C, D, E, F, G.
  - Else drop the column.
- Modify the target loan_status column to +1 for Fully Paid and -1 for Charged Off.
- Fill the null values with mean of the column.
- Find feature importance using Random Forest and drop columns with importance less than 0.0003.
- Drop id and member_id columns, as they are unique for each row.

Part (b)

Hyperparameters tuning:
Learning rate

Learning rate	Accuracy	Precision	Recall
0.001	0.84919	0.84919	1.0
0.01	0.97219	0.96829	1.0
0.05	0.98998	0.98834	1.0
0.1	0.99468	0.99385	0.99992
0.15	0.99545	0.99491	0.99975
0.2	0.996217	0.99573	0.99984

Number of trees

Number of trees	Accuracy	Precision	Recall
50	0.9951	0.99434	0.99992
100	0.996217	0.99573	0.99984
200	0.99727	0.99688	01.99992
300	0.99748	0.99712	0.99992
500	0.99769	0.99737	0.99992
700	0.99775	0.99745	0.99992
1000	0.99797	0.9977	0.99992
1250	0.99811	0.99786	0.99992
1500	0.99811	0.99786	0.99992
1700	0.99811	0.99786	0.99992

Increasing the number of trees increases the accuracy, but becomes stagnant after a certain point.

Max depth

Max depth	Accuracy	Precision	Recall
3	0.99811	0.99786	0.99992
5	0.99825	0.99794	1.0
7	0.99825	0.99794	1.0
9	0.99811	0.99778	1.0

After doing hyperparameter tuning, we get:

- Learning rate: 0.2
- Number of trees: 1250
- Max depth: 5
- Accuracy: 0.9982488091902494
- Precision: 0.997942048073757
- Recall: 1.0

Gradient Boosting vs Decision Tree:

- Gradient Boosting:

  - Accuracy: 0.9982488091902494
  - Precision: 0.997942048073757
  - Recall: 1.0

- Decision Tree:
  - Accuracy: 0.9969179041748389
  - Precision: 0.9979388243053838
  - Recall: 0.9984327311721521