Homework 1

2020 Spring CSCI 5525: Machine Learning

Due on Fabruary 16th 11:59pm

Please type in your info:

- **Name**:Arvind Renganthan
- **Student ID**:5595189
- **Email**:renga016@umn.edu
- **Collaborators, and on which problems:**

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to fill in above to specify which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online.
- Look up things/post on sites like Quora, StackExchange, etc.

Submission. Submit a PDF using this LaTeX template for written assignment part and submit Python script file (.py) for all programming part. You should upload all the files on Canvas.

Written Assignment

Instruction. For each problem, you are required to write down a full mathematical proof to establish the claim.

Problem 1. Two helpful matrices.

Let us first recall the notations in linear regression. The design matrix and the response vector are are defined as:

$$A = \begin{bmatrix} \leftarrow x_1^\intercal \rightarrow \\ \vdots \\ \leftarrow x_n^\intercal \rightarrow \end{bmatrix} \qquad \mathbf{b} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

For this problem, we will assume the covariance matrix $A^\intercal A$ is invertible, and so $(A^\intercal A)^{-1}$ is well-defined (**Clearly mention the properties of matrix operations used while solving**).

Problem 1.1. The residual matrix. For any weight vector $\mathbf{w}$, let us define the vector of least squares residuals as

$$e = \mathbf{b} - A\mathbf{w}$$

Now if $\mathbf{w}$ is the least square solution given by $\mathbf{w} = (A^\intercal A)^{-1}A^\intercal\mathbf{b}$, we can rewrite e as

$$e = \mathbf{b} - A(A^\intercal A)^{-1}A^\intercal\mathbf{b} = \left(I - A(A^\intercal A)^{-1}A^\intercal\right)\mathbf{b}$$

Now let $M = \left(I - A(A^\intercal A)^{-1}A^\intercal\right)$. Show that

- M is symmetric (i.e. $M = M^\intercal$). (**2 points**)
- M is idempotent (i.e. $M^2 = M$). (**2 points**)
- $MA = 0$. (**1 point**)

Solution To prove $M = M^T$
$M^T = (I - A(A^TA)^{-1}A^T)^T$
$= I - (A(A^TA)^{-1}A^T)^T$ [Expanding and I^T = I]
$= I - (A^T)^T[(A^TA)^{-1}]^TA^T$
$= I - A[((A^TA)^T)^{-1}]A^T$
$= I - A[((A^T(A^T)^T)^{-1}]A^T$
$= I - A[(A^TA)^{-1}]A^T$
=M [previous step same a M]
So M is symmetric

For idempotent $M^2 = M$
$M^2 = (I - A(A^TA)^{-1}A^T)(I - A(A^TA)^{-1}A^T)$
$= I^2 - 2A(A^TA)^{-1}A^T + (A(A^TA)^{-1}A^T)(A(A^TA)^{-1}A^T)$
$= I - 2A(A^TA)^{-1}A^T + A(A^TA)^{-1}A^TA(A^TA)^{-1}A^T)$
$= I - 2A(A^TA)^{-1}A^T + A.I.(A^TA)^{-1}A^T$ [as $(A^TA)^{-1}A^TA = I$]
$= I - 2A(A^TA)^{-1}A^T + A(A^TA)^{-1}A^T$
$= I - A(A^TA)^{-1}A^T$
$= M$
so M is idempotent

To prove MA = 0
$MA = (I - A(A^TA)^{-1}A^T)A$
$= I.A - A(A^TA)^{-1}A^TA$
$= A - A(A^TA)^{-1}A^TA$
$= A - A$ [as $(A^TA)^{-1}A^TA = I$]
$= 0$

Problem 1.2. The hat matrix. Using the residual maker, we can derive another matrix, the hat matrix or projection matrix $P = I - M = A(A^\intercal A)^{-1}A^\intercal$. Note that the predicted value by the least squares solution is given by $P\mathbf{b}$. Show that

- P is symmetric. (**1 point**)
- P is idempotent. (**1 point**)

Solution To prove P is symmetric P = P^T
$P^T = (A(A^TA)^{-1}A^T)^T$
$= (A^T)^T[(A^TA)^{-1}]^TA^T$
$= A[((A^TA)^T)^{-1}]A^T$
$= A[((A^T(A^T)^T)^{-1}]A^T$
$= A[(A^TA)^{-1}]A^T$
=P So P is symmetric

To prove P is idempotent P = P^2
$P^2 = (A(A^TA)^{-1}A^T)(A(A^TA)^{-1}A^T))$
$= A.I.(A^TA)^{-1}A^T$ [as $(A^TA)^{-1}A^TA = I$]
$= A(A^TA)^{-1}A^T$
=P
So P is idempotent

Problem 2. Gradient of conditional log-likelihood.

For any $a \in \mathbb{R}$, let $\sigma(a) = \frac{1}{1+\exp(-a)}$. For each example $(x_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$, the conditional log-likelihood of logistic regression is

$$\ell(y_i \mid \mathbf{x_i}, \mathbf{w}) = y_i \ln(\sigma(\mathbf{w}^\intercal \mathbf{x_i})) + (1 - y_i) \ln(\sigma(-\mathbf{w}^\intercal \mathbf{x_i}))$$

Derive the gradient of $\ell(y_i \mid \mathbf{x_i}, \mathbf{w})$ with respect to w_j (i.e. the j-th coordinate of $\mathbf{w}$), i.e. $\frac{\partial}{\partial w_j}\ell(y_i \mid \mathbf{x_i}, \mathbf{w})$ (**Clearly mention the properties of derivatives used while solving**). (**6 points**)

Solution let $a_n = w^TX_n$ so $a_n = w_0x_{n0} + w_1x_{n1} + w_2x_{n2} + \cdots + w_dx_{nd}$
let $p_i = \sigma(a_i)$
let $\sigma(a) = \frac{1}{1+\exp(-a)}$
$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial p_i}\frac{\partial p_i}{\partial a_i}\frac{\partial a_i}{\partial w_j}$ [Chain-Rule]
Let us start with cost function
$\frac{\partial L}{\partial p_i} = y_i\frac{1}{p_i} + (1 - y_i)\frac{1}{1-p_i}(-1) = \frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}$ -(1) [using rule for derivation x/y]
Next derive activation function we get
$p_i = \sigma(a_i) = \frac{1}{1+e^{-a_i}}$
$\frac{\partial p_i}{\partial a_i} = \frac{-1}{(1+e^{-a_i})^2}(e^{-a_i})(-1) = \frac{e^{-a_i}}{(1+e^{-a_i})^2} = \frac{1}{1+e^{-a_i}}\frac{e^{-a_i}}{1+e^{-a_i}}$ [using rule for derivation x/y and rule for derivation of x multiplied by y]
$\frac{\partial p_i}{\partial a_i} = p_i(1 - p_i)$ -(2)
Now for a_i
$\frac{\partial a_i}{\partial w_j} = x_{ij}$ - (3)
use (1),(2),and (3) in Chain rule
$\frac{\partial L}{\partial w_j} = \frac{y_i}{p_i}p_i(1 - p_i)x_{ij} - \frac{1-y_i}{1-p_i}p_i(1 - p_i)x_{ij}$
$= y_i(1 - p_i)x_{ij} - (1 - y_i)p_ix_{ij}$
$= [y_i - y_ip_i - p_i + y_ip_i]x_{ij}$
$= (y_i - p_i)x_{ij}$
$= (y_i - \sigma(a_i))x_{ij}$
$= (y_i - \sigma(w^TX_i))x_{ij}$

Problem 3. Derivation of Ridge Regression Solution.

Recall that in class we claim that the solution to ridge regression ERM:

$$\min_{\mathbf{w}} \ (\|A\mathbf{w} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{w}\|_2^2)$$

is $\mathbf{w}^* = (A^\intercal A + \lambda I)^{-1} A^\intercal \mathbf{b}$. Now provide a proof. (**6 points**)
(Hint: recall that $\nabla F(\mathbf{w}) = \mathbf{0}$ is a sufficient condition for $\mathbf{w}$ to be a minimizer of any convex function F. To get a full credit, you should be able to show why $A^\intercal A + \lambda I$ is invertible.)
(**Clearly mention the properties of matrix calculus used while solving**)

Solution. Expanding F we get $(b - Aw)^T(b - Aw) + \lambda w^T w$
$= b^T b - w^T A^T b - b^T Aw + w^T x^T Aw + \lambda w^T w$
$= b^T b - w^T A^T b - w^T A^T b + w^T A^T Aw + w^T \lambda I w$ with I being the identity matrix
$= b^T b - 2w^T A^T b + w^T \left(A^T A + \lambda I\right) w$
as $\nabla F(\mathbf{w}) = \mathbf{0}$ is a sufficient condition for $\mathbf{w}$ to be a minimizer of any convex function F.
Now we search for the w that minimizes our criterion.
$\frac{\partial F}{\partial w} = -2A^T b + 2\left(A^T A + \lambda I\right) w = 0$ (as w is symmetric)

$\left(A^T A + \lambda I\right) w = A^T b$
$\Rightarrow w^* = \left(A^T A + \lambda I\right)^{-1} A^T b$ (if $(A^T A + \lambda I)$ is invertible)

To prove $M = (A^T A + \lambda I)$ is invertible

M is the sum of (symmetric) positive semidefinite matrices $A^T A$ and λI [as For any column vector v, we have $v^T A^T Av = (Av)^T(Av) = (Av)(Av) \geq 0$, therefore $A^T A$ is positive definite as colums of X are linearly independent]
so M itself is positive semidefinite.
Moreover, since M is a positive semidefinite matrix, it is invertible if and only if it is positive definite.
c is an eigenvalue of $X^T X$ if and only if c + λ is an eigenvalue of $X^T X + \lambda I$. As $X^T X$ is positive semi definite it means that all its eigenvalues are non-negative. This in turn implies that all the eigenvalues of $X^T X + \lambda I$ are positive, i.e. it is positive definite so it is invertible

Problem 4. Minimizing a Squared Norm Plus an Affine Function.

A generalization of the least squares problem adds an affine function to the least squares objective,

$$\min_{\mathbf{w}} \ \|A\mathbf{w} - \mathbf{b}\|_2^2 + \mathbf{c}^\intercal \mathbf{w} + d$$

where $A \in \mathbb{R}^{m \times n}, \mathbf{w} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n, d \in \mathbb{R}$. Assume the column of A are linearly independent. This generalized problem can be solved by reducing it to a standard least squares problem, using a trick called *completing the square*.

Show that the objective of the problem above can be expressed in the form

$$\|A\mathbf{w} - \mathbf{b}\|_2^2 + \mathbf{c}^\intercal \mathbf{w} + d = \|A\mathbf{w} - \mathbf{b} + \mathbf{f}\|_2^2 + g$$

where $\mathbf{f} \in \mathbb{R}^m, g \in \mathbb{R}$.It follows that we can solve the generalized least squares problem by minimizing $\|A\mathbf{w} - (\mathbf{b} - \mathbf{f})\|_2^2$

(Hint: Express the norm squared term on the right-hand side as $\|(A\mathbf{w} - \mathbf{b}) + \mathbf{f})\|_2^2$ and expand it. Then argue that the equality above holds provided $2A^\intercal \mathbf{f} = \mathbf{c}$. One possible choice is $f = \frac{1}{2}(A^\dagger)^\intercal \mathbf{c}$.) (You must justify these statements.) (**6 point**)

Solution: we get $(Aw - b)^\top (Aw - b) + c^\top w + d$
Expressing norm squared term on the right-hand side as
$||(Aw - b) - f||_2^2 + g$
$= \|Aw - b\|_2^2 + 2\langle Aw - b, f\rangle + \|f\|^2 + g$
$= \|Aw - b\|_2^2 + 2f^\top (Aw - b) + \|f\|^2 + g$
$= \|Aw - b\|_2^2 + 2f^\top Aw + \|f\|^2 + g - 2f^\top b$
So comparing coefficients we have
$c^T = 2b^T A \quad d = ||f||^2 + g - 2f^T b$. [as each term which has been equated to d $\in \mathbb{R}$]
$c = 2A^\top b$ and $d = ||f||^2 + g - 2f^T b$. for equality

For minimizing the problem $\|Aw - (b - f)\|_2^2$
Now, L(x) $= \|(Aw - b) - f\|_2^2$
$= \|Aw - b\|_2^2 + 2f^T Aw + ||f||^2 - 2f^T b$
$\frac{\partial}{\partial n} L(w) = 0$
$\Rightarrow 2A^\top (Aw - b) = -2A^\top f$
$\Rightarrow \quad A^T Aw = -A^T f + A^T b$
$\Rightarrow A^\top Aw = A^\top (b - f)$
$\Rightarrow \quad w = \left(A^T A\right)^{-1} A^T (b - f)$ [Here $((A^T A)^{-1}$ is put in since columns of a are linearly independent]
also the term $\left(A^T A\right)^{-1} A^T$ is equal to A^+ which is pseudo inverse of A
$\Rightarrow w = A^+ (b - f)$

Problem 5. Iterative Method for Least Squares Problem.

In this exercise we explore an iterative method, due to the mathematician Lewis Richardson, that can be used to compute $\hat{\mathbf{w}} = A^\dagger \mathbf{b}$, we define $\mathbf{w}^{(1)} = 0$ and for $k = 1, 2, 3, ...,$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mu A^\intercal (A\mathbf{w}^{(k)} - \mathbf{b})$$

where μ is a positive parameter, and the superscripts denote the iteration number.This defines a sequence of vectors that converge to $\hat{\mathbf{w}}$ provided μ is not too large; The iteration is terminated when $A^\intercal (A\mathbf{w}^{(k)} - \mathbf{b})$ is small enough, which means the least squares optimality conditions are almost satisfied. To implement the method we only need to multiply vectors by A and by $A^\intercal$. If we have efficient methods for carrying out these two matrix-vector multiplications, this iterative method can be faster. Iterative methods are often used for very large scale least squares problems.

(a) Show that if $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)}$, we have $\mathbf{w}^{(k)} = \hat{\mathbf{w}}$. (**5 points**)

(b) Express the vector sequence $\mathbf{w}^{(k)}$ as a linear dynamical system with constant dynamics matrix and offset, i.e., in the form $\mathbf{w}^{(k+1)} = F\mathbf{w}^{(k)} + g$. (**3 points**)

Solution (a). Here it is given that $w^{(k+1)} = w^{(k)} - \mu A^\top \left(Aw^{(k)} - b\right) \quad (1)$
Here it is also given that $w^{(k+1)} = w^{(k)}$
Then by putting it in (1) we have $w^{(k)} = w^{(k)} - \mu A^T \left(Aw^{(k)} - b\right)$

$so, \quad \mu A^T \left(Aw^{(k)} - b\right) = 0$

$so, \quad Aw^{(k)} - b = 0 \quad \left[\text{since} \quad \begin{matrix} \mu \neq 0 \text{ and} \\ A^\top \neq 0 \end{matrix} \right]$

so $\quad Aw^{(k)} = b$

Multiplying by $A^\top$ on both sides we have,

$A^\top A \cdot w^{(k)} = A^\top \cdot b$

so, $w^{(k)} = A^\top b$[for the orthogonality]

so, $w^{(k)} = \hat{w} \quad$ [since $\hat{w} = A^\top b$ given]

so $\quad w^{(k)} = \hat{w}$, holds, only, when A in an orthogonal ie $A^\top A = 1$

(b)

From eq$n$ (1)

we have $w^{(k+1)} = w^{(k)} - \mu A^T \left(Aw^{(k)} - b\right)$

$So, w^{(k+1)} = w^{(k)} - \mu A^\top \cdot Aw^{(k)} + \mu A^\top \cdot b$

$so, w^{(k+1)} = w^{(k)} - \mu \cdot I \cdot w^{(k)} + \mu \cdot A^\top \cdot b$

So, $w^{(k+1)} = w^{(k)}(1 - \mu I) + \mu \cdot A^\top \cdot b$ [Let A be an orthogonal matrix,then $A^\top \cdot A = I$]

so $w^{(k+1)} = (1 - \mu I)w^{(k)} + \mu \cdot \hat{w}$ $\left[\because \hat{w} = A^\top \cdot b\right)$ we can write this as

$So, \quad w^{(k+1)} = F \cdot w^{(k)} + g$ [where $F = 1 - \mu I \quad and \quad g = \mu \cdot \hat{w}$] This is the reqd. expression

Programming Assignment

Instruction. For each problem, you are required to report descriptions and results in the PDF and submit code as python file (.py) (as per the question).

- **Python** version: Python 3.
- Please follow PEP 8 style of writing your Python code for better readability in case you are wondering how to name functions & variables, put comments and indent your code
- **Packages allowed**: numpy, pandas, matplotlib
- Please PROPERLY COMMENT your code in order to have utmost readability
- Please provide the required functions for each problem.
- There shall be NO runtime errors involved.
- There would be PENALTY if any of the above is not followed
- **Submission**: For programming parts, **ONLY THE PYTHON 3 SCRIPT FILE WILL BE ACCEPTED**

Problem 6. Iterative Method for Least Squares Problem (Cont'd).

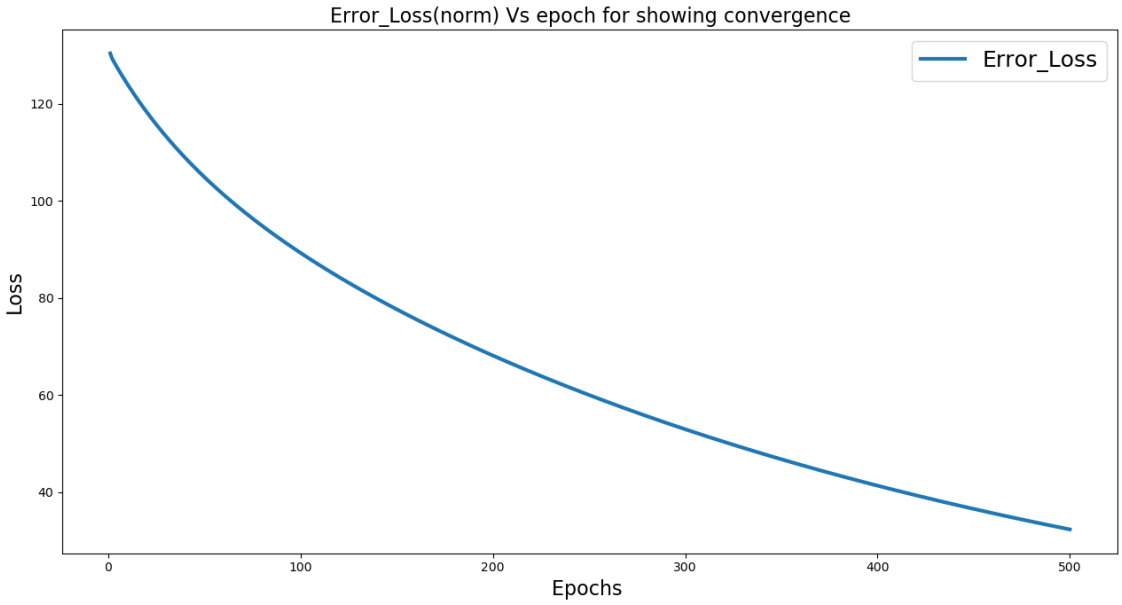
For this problem, you will implement Richardson algorithm introduced in Problem 5 and report the required graphs. Please submit a Python script file (name hw1_lsq_iter.py)

- Generate a random 20×10 matrix A and 20-vector b, and compute $\hat{\mathbf{w}} = A^{\dagger}\mathbf{b}$. Run the Richardson algorithm with $\mu = \frac{1}{\|A\|^2}$ for 500 iterations, and plot $\|\mathbf{w}^{(k)} - \hat{\mathbf{w}}\|$ to verify that $\mathbf{w}^{(k)}$ appears to be converging to $\hat{\mathbf{w}}$. Please properly analyze and describe your plot. (**12 points**)
- Note: function np.linalg.lstsq is not allowed.

  To get a full credit, the main script should output the required plots with explicitly named x-y axis and the following function should be provided:

  1. $\mathbf{w}$ = lsq(A, $\mathbf{b}$) where $\mathbf{w} = \arg\min_{\mathbf{w}} \ \|A\mathbf{w} - \mathbf{b}\|_2^2$ solved by closed form.
  2. $\mathbf{w}$ = lsq_iter(A, $\mathbf{b}$) where $\mathbf{w} = \arg\min_{\mathbf{w}} \ \|A\mathbf{w} - \mathbf{b}\|_2^2$ solved by Richardson algorithm.

Solution Plot



In the above plot by error_loss I mean value of $\|\mathbf{w}^{(k)} - \hat{\mathbf{w}}\|$.
We can see that as the epoch goes higher $\|\mathbf{w}^{(k)} - \hat{\mathbf{w}}\|$ value is going down in an exponential manner.
So from the graph we can come to conclusion that it will converge at the same value as $\hat{\mathbf{w}}$ we get from closed form solution eventually as it is something similar to a exponential decay.

Problem 7. Logistic regression.

For this problem, you will use the IRIS dataset. Features are in the file IRISFeat.csv and labels in the file IRISlabel.csv. The dataset has 150 samples. A python script (name hw1-logistic.py) with all the steps need to be submitted.

a) (**12 Points**) Your goal is to implement logistic regression. You can design or structure your code to fulfil the requirements but to get a full credit, the main script should output the required tables or plots with explicitly named x-y axis and the following function should be provided where X being features and y target.:

1. Cross validation: Make sure you randomly shuffle the dataset and partition it into almost equal (k=5) folds. Save each of the 5 folds into dictionary X_shuffled and y_shuffled.
   X_train, y_train, X_valid, y_valid = get_next_train_valid(X_shuffled, y_shuffled, itr) where itr is iteration number.
2. model_weights, model_intercept = train(X_train, y_train)
3. y_predict_class = predict(X_valid, model_weights, model_intercept)

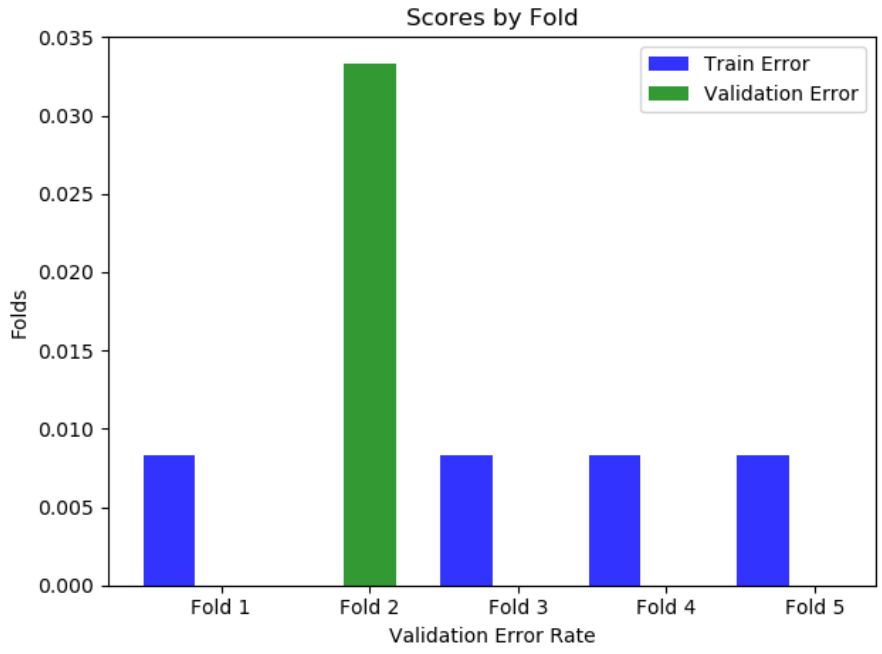
USE GRADIENT DESCENT to solve it. You should initialize the weights randomly to begin with.

b) (**3 Points**) At the beginning, briefly describe the approach in one paragraph along with any equations and methods used in your implementation.

c) (**5 Points**) Report the plot of training and validation set error rates (number of misclassified samples/total number of samples) and the confusion matrix for validation set from 5-fold cross validation. Explain your selection of learning rate and How does it affect the performance/training of your model?

(b) So the algorithm by taking gradient moves along the slope towards global minima through a number of iterations or till convergence condition is established.While we move down the slope we update the weights as well.
Steps followed in the algorithm are
1: Initialize at step t = 0 to w(0).
2: for t = 0, 1, 2, . . . do
3: $\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t))$
4: Move in the direction $v_t = g_t$
5: Update the weights: $\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y_* \mathbf{x}_* \left(\frac{\eta}{1+e^{y_* \mathbf{w}^{\mathrm{T}} \mathbf{x}_*}} \right)$ [For logistic regression]
6: Iterate 'until it is time to stop' or convergence condition.
7: end for
8: Return the final weights
(c)
The training error rates for 5 folds were [0.008333333333333304, 0.0, 0.008333333333333304, 0.008333333333333304, 0.008333333333333304]
The validation error rates for 5 folds were [0.0, 0.033333333333333326, 0.0, 0.0, 0.0]

Bar plot for the above



Confusion matrix

FOLD 1

		True Value		
		Positive	Negative	Total
Predicted	Positive	19	0	19
	Negative	0	11	11
	Total	19	11	30

FOLD 2

		True Value		
		Positive	Negative	Total
Predicted	Positive	21	0	21
	Negative	1	8	9
	Total	22	8	30

FOLD 3

		True Value		
		Positive	Negative	Total
Predicted	Positive	23	0	23
	Negative	0	07	07
	Total	23	07	30

FOLD 4

		True Value		
		Positive	Negative	Total
Predicted	Positive	18	0	18
	Negative	0	12	12
	Total	18	12	30

FOLD 5

		True Value		
		Positive	Negative	Total
Predicted	Positive	20	0	20
	Negative	0	10	10
	Total	20	10	30

Initially i selected a learning rate of 0.01 as I wanted to make sure I take the smallest learning rate possible so that i dont miss any local minima while travelling on slope for gradient descent. At the same time it was getting stuck in a platue region and took a long time to run so i increased the learning rate to 0.1 and made sure it started converging. It converged to final result fast and did not miss local minima at 0.1 so i fixed it as my learning rate.
For Larger training rate if it does converge it will converge faster whereas there is no assurance that it will converge.
Small training rate will definitely converge but it may take a very long time.