Department of Computational and Data Sciences
DS226: Introduction to Computing for Artificial Intelligence and Machine Learning
November 11, 2022

Assignment 3

Instructions:

- **Submit a typed report in PDF format on Moodle. Use LaTeXpreferably. Handwritten reports will not be accepted.**
- **Solutions should be in the same order as the questions.**
- **Discussion is encouraged, but answers should be your own. Do not copy answers. Plagiarism will be penalised severely.**
- **For late submissions, the following late submission policy:**

Delay	**% of Credit that will be considered**
0-24 hours	**$99 - x$, example if late by 1.1 hour $x = 2$ (so will vary from 98 to 75)**
24-48 hours	**50**
48 hours - 1 week	**25**
Beyond 1 week	**No credit**

Deadline: 27th October, 11:59 AM
Total Credits: 100

Nomenclature

α	Learning rate
$\hat{w}_0$	Intercept (or bias)
$\hat{w}_i$	Coefficients ($i = 1, 2, \cdots, N$)
$\hat{y}$	Predicted output
$\hat{\mathbf{w}}$	Coefficient vector $\begin{bmatrix}\hat{w}_0 & \hat{w}_1 & \dots & \hat{w_M}\end{bmatrix}^T$
$\hat{\mathbf{Y}}$	Target vector $\begin{bmatrix}\hat{y}^1 & \hat{y}^2 & \dots & \hat{y}^N\end{bmatrix}^T$
$\mathbf{X}$	Feature matrix
$\mathbf{x}$	Input features , $\mathbf{x} = (x_1, x_2, \cdots, x_M)$
M	Number of features
N	Number of samples (or observations)

Question 1. Linear Regression from scratch. (20)

Consider a simple linear regression model

$$\hat{y} = \hat{w_0} + \sum_{j=1}^{M} x_j \hat{w_j}$$

Suppose we have N observations $(\mathbf{x}^i, y^i), i = 1, 2, \cdots, N$, where $\mathbf{x}^i \in \mathbb{R}^M, y^i \in \mathbb{R}$. We can write the linear model in vector form as an inner product,

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{w}} \tag{1.1}$$

where,

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_M^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_M^2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^N & x_2^N & \cdots & x_M^N \end{bmatrix}$$

Recall that by using the method of least squares,for a given set of outputs $\mathbf{y}$ in the training set, the solution of the normal equation gives us

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.2}$$

I. **Starting from equation 1.1, derive equation 1.2. With this result, implement a linear estimator in Python using the template given in `NormalEquation.py`**

Now, consider a linear regression problem in one variable,

$$\hat{y} = \hat{w_0} + x\hat{w_1}$$

II. **Using the least squares method, prove that for the best fit, the coefficient and intercept are given by**

$$\hat{w_0} = \frac{(\sum_{i=1}^N y^i \sum_{i=1}^N (x^i)^2) - (\sum_{i=1}^N x^i \sum_{i=1}^N x^i y^i)}{(N \sum_{i=1}^N (x^i)^2 - (\sum_{i=1}^N x^i)^2)}$$

$$\hat{w_1} = \frac{(N \sum_{i=1}^N x^i y^i) - (\sum_{i=1}^N x^i \sum_{i=1}^N y^i)}{(N \sum_{i=1}^N (x^i)^2 - (\sum_{i=1}^N x^i)^2)}$$

Implement the above relation in Python to build a linear estimator in one variable. Use the template proved in `LinearRegression1D.py`

The required implementation must be object-oriented. Instructions on using the template are given in `Instructions.pdf`

Question 2. Ridge Regression. (30)

Recall that in Ridge regression, the ridge coefficients minimize a penalized residual sum of squares,

$$\sum_{i=1}^{N} \left(y^i - w_0 - \sum_{j=1}^{M} x_j^i w_j \right)^2 + \lambda \sum_{j=1}^{M} w_j^2 \tag{2.1}$$

where $\lambda \geq 0$ is a complexity parameter which controls the amount of shrinkage.

(a) [**10 Credits**]

Assuming that $\bar{x} = 0$ such that the input data has been centered, show that the optimizer for

$$Z(w_0, \mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0\mathbf{1})^T (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0\mathbf{1}) + \lambda \mathbf{w}^T\mathbf{w} \tag{2.2}$$

is

$$\hat{w}_0 = \bar{y}$$
$$\hat{\mathbf{w}}^c_{ridge} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Note that equation 2.2 and 2.1 are same.

(b) [20 Credits]

Recall that $\hat{\mathbf{w}}_{ridge} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$.
Let $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}_{ridge}$

- Taking the SVD of the centered input matrix $\mathbf{X}$ as $\mathbf{X} = \mathbf{USV}^T$, get the expression of $\hat{\mathbf{y}}$ in terms of $\mathbf{u}_j$, s_j and $\mathbf{y}$, where $\mathbf{u}_j$ are columns of $\mathbf{U}$ and s_j are the diagonal elements of $\mathbf{S}$ (Hint: It will be in this form $\hat{\mathbf{y}} = \sum_{j=1}^{D}\mathbf{u}_j f(s_j,\lambda)\mathbf{u}_j^T\mathbf{y}$)

- Further show that for least squares prediction, doing similar analysis one can come to the conclusion that $\hat{\mathbf{y}} = \sum_{j=1}^{D}\mathbf{u}_j\mathbf{u}_j^T\mathbf{y}$. (Hint: For least squares, use $\mathbf{w}_{ls} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$)

- Comparing the expressions for $\hat{\mathbf{y}}$ for Least Squares and Ridge and analysing the function $f(s_j,\lambda)_{ridge}$, comment during which conditions the shrinking done by the Ridge regression is greater?

For more detailed intuition on connection of ridge regression and PCA, refer to section 7.5 of the book "Machine Learning: a Probabilistic Perspective" by Kevin Patrick Murphy.

Note: For part **(a)** and **(b)** brief derivations are expected.

Question 3. Elastic net (ridge and lasso combined). (10)

Let

$$Z_1(\mathbf{w}) = (\mathbf{y} - \mathbf{Xw})^T(\mathbf{y} - \mathbf{Xw}) + \lambda_2|\mathbf{w}|^2 + \lambda_1|\mathbf{w}|_1 \tag{3.1}$$

$$Z_2(\mathbf{w}) = \left(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w}\right)^T\left(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w}\right) + c\lambda_1|\mathbf{w}|_1 \tag{3.2}$$

where $c = (1+\lambda_2)^{-0.5}$ and $\tilde{\mathbf{X}} = c\begin{pmatrix}\mathbf{X}\\ \sqrt{\lambda_2}\,\mathbf{I}_d\end{pmatrix}$, $\tilde{\mathbf{y}} = \begin{pmatrix}\mathbf{y}\\ \mathbf{0}_{d\times 1}\end{pmatrix}$

With above information show that,

$$Z_1(c\mathbf{w}) = Z_2(\mathbf{w})$$

and hence you will be showing that the elastic net problem 3.1 can be reduced to a lasso problem on modified data 3.2

Question 4. Logistic regression as an extension of Linear Regression. (40)

Here, you are given a set of 10,000 tweets and you have to classify them as "positive" or "negative" using logistic regression. Recall that in linear regression we had a linear relationship between $\hat{y}$ and $\mathbf{x}$ of the form,

$$\hat{y} = \hat{\mathbf{w}}^T \mathbf{x}$$

where $\hat{\mathbf{w}}$ is of the form a

$$\hat{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

and $\mathbf{x}$ is of the form

$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Now we introduce a logistic function (sigmoid) that maps these values onto (0,1) given by

$$h(y) = \frac{1}{1 + \exp^{-\hat{y}}}$$

The loss function that is of concern here is

$$Z(w) = -\frac{1}{N} \sum_{i=1}^{N} p^{(i)} \log(h^{(i)})) + (1 - p^{(i)}) \log(1 - h^{(i)})$$

and is known as the cross-entropy loss.
$p^{(i)}$ is the actual label of the i-th training point.
$h(\hat{y}(\hat{\mathbf{w}})^{(i)})$ is the model's prediction for the i-th training point.

To update the weight w_j, we use:

$$w_j = w_j - \alpha \times \nabla_{w_j} Z(w)$$

where

$$\nabla_{w_j} Z(w) = \frac{1}{N} \sum_{i=1}^{N} (h^{(i)} - \hat{y}^{(i)}) x_j$$

'i' is the index across all N training examples.
'j' is the index of w_j and x_j
α is the learning rate

The output of the linear regression is passed as $\hat{y}$ into this function and this allows us to perform binary classfication. In this problem if $h(\hat{y}) >= 0.5$ we consider the tweet to be positive and if it $h(\hat{y}) < 0.5$ we consider the tweet to be negative.

You are provided with a Jupyter notebook which does the pre-processing of the dataset and converts the dataset into a matrix of numbers. Have a look at the code and do the following

- Use 75% of the data for training and remaining 25 % for testing.
- Prepare a scatter plot of number of "positive" words vs number of "negative" words for the tweets in the training dataset. Is there any correlation you see? (2)

- Implement Gradient descent and the other functions mentioned in the notebook for logistic regression. Also check the misclassified tweets. You can change the function definitions given in the notebook if required (25)
- Prepare plots for variation of accuracy and loss as a function of number of iterations for the gradient descent algorithm and report your training and test accuracies at the end of training. (5)
- Predict the scores for the sentences given at the end of the notebook and classify them as positive or negative. (2)
- Based on the analysis done above what can you say about the limitations of this model when trying to classify tweets. How would you improve the same model. (2)
- How would your results change if you were to remove all 5 of these emoticons mentioned below from your analysis (result with respect to training and test accuracies that your code gives with these modifications). Can you give an intuitive explanation for your answer? (6)
  1) :)
  2) :(
  3) :-)
  4) :-(
  5) :d
- **Deliverables** : Jupyter notebook (**All comments and analysis must be written in the notebook itself in markdown. Plots also should be shown in the notebook itself. No need for an additional pdf**)and python file (.py) with the same code . Jupyter notebook will be evaluated and .py file will be used for plagiarism checking.