

# HOMework 1

## LINEAR REGRESSION AND LOGISTIC REGRESSION<sup>1</sup>

GEC-1539: ML&DS - DEVELOPMENT OF APPLICATIONS

OUT: Sunday, Apr 17th, 2022

DUE: Sunday, May 1st, 2022, 11:00pm, Beijing Time

Lecturer: Mark Vogelsberger

Mentor: Haizhou Shi

TAs: Huazhi Dong, Yu Wang

Reporter: **HAODONG LI**

## Instructions

Homework 1 covers the basic algorithms in Machine Learning, including linear regression and logistic regression. Each problem includes some short derivation questions to help you go through the necessary knowledge to complete the following coding task. Read through the Instructions below before you start working on the homework.

- **Collaboration Policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., “Jane explained to me what is asked in Question 2.1”). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Late Submission Policy:** The submission of an assignment made past the deadline will get 70% of the credits. Any assignment submitted more than 3 days past the deadline will get zero credit.
- **Submission Policy**
  - **Written problems submission:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please write your solution in the LaTeX files provided in the assignment and submit in a PDF form. Put your answers in the question boxes (between `\begin{soln}` and `\end{soln}`) below each problem. **Handwritten solutions are not accepted and will receive zero credit.** Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found

---

<sup>1</sup>Compiled on Monday 2 May, 2022 at 07:44

then points will be deducted. Please upload the written problems via the Eds platform.

- **Code submission:** All code must be submitted under the specific instruction given in the problem. We will be using the Edstem platform to auto-grade your code. **If you do not submit your code there, you will not receive any credit for your assignment.** The Edstem platform will auto-detect the possible plagiarism. Please make sure you familiarize yourself with the academic integrity information for this course.

### Important Notes on the Programming Problems:

- Do not use any toolboxes except those already imported in the code template.
- Read the doc-strings/comments in the template very carefully before you start.
- Reach out for help on Edstem Platform or during office hours when you struggle.
- Do not change any function signatures because your code will be auto-graded.
- Try to vectorize the computation as much as possible (e.g. compute in the form of matrix multiplication, utilize numpy functions instead of loops, etc.)
- Use Python 3.6 or above, and the latest version of numpy.

## Problem 1 (40 pts)

In this problem we will implement the linear regression model and apply it to a simple problem. The data we use here is the Fish Market dataset<sup>2</sup>.

### Part 1. Derivation (10 pts)

1. (8 pts) Suppose the input feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times D}$  and the target output matrix  $\mathbf{T} \in \mathbb{R}^{N \times d}$ , where  $N$  is the number of data points, and  $D$  and  $d$  is the number of features of a single data point and response variable, respectively. The multivariate linear regression model is represented as:

$$\mathbf{y}(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \mathbf{W}^\top \mathbf{x} + \mathbf{b},$$

where  $\mathbf{W} \in \mathbb{R}^{D \times d}$ ,  $\mathbf{b} \in \mathbb{R}^d$ . Try to derive the closed form solution to the least-square loss, where the objective is defined as:

$$(\mathbf{W}, \mathbf{b})^* = \arg \min_{\mathbf{W}, \mathbf{b}} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n; \mathbf{W}, \mathbf{b}) - \mathbf{t}_n\|_2^2$$

Hint: try to convert the input feature matrix to the extended version  $\mathbf{X} = \begin{bmatrix} -\mathbf{x}_1^\top, 1 \\ \vdots \\ -\mathbf{x}_N^\top, 1 \end{bmatrix}$ .

Solution

$$\begin{aligned} \mathbf{Y}(\mathbf{x}; \mathbf{W}, \mathbf{b}) &= \mathbf{W}^\top \mathbf{x} + \mathbf{b} \\ &= \mathbf{X} \cdot \mathbf{W}_b \end{aligned}$$

$$\mathbf{X} = \begin{bmatrix} -\mathbf{x}_1^\top, 1 \\ \vdots \\ -\mathbf{x}_N^\top, 1 \end{bmatrix}, \quad \mathbf{W}_b = \begin{bmatrix} \mathbf{W} \\ \mathbf{b} \end{bmatrix}$$

$$\begin{aligned} \Rightarrow (\mathbf{W}, \mathbf{b})^* &= \arg \min_{\mathbf{W}, \mathbf{b}} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n; \mathbf{W}, \mathbf{b}) - \mathbf{t}_n\|_2^2 \\ &= \arg \min_{\mathbf{W}_b} \|\mathbf{X} \cdot \mathbf{W}_b - \mathbf{T}\|^2 \end{aligned}$$

$$\text{We assume } E(\mathbf{W}_b) = \|\mathbf{X} \cdot \mathbf{W}_b - \mathbf{T}\|^2$$

$$\begin{aligned} \Rightarrow \frac{\partial E(\mathbf{W}_b)}{\partial \mathbf{W}_b} &= \mathbf{X}^\top \mathbf{X} \mathbf{W}_b - \mathbf{X}^\top \mathbf{T} = 0 \\ \Rightarrow \mathbf{W}_b &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{T} \end{aligned}$$

<sup>2</sup><https://www.kaggle.com/datasets/aungpyaeap/fish-market>

2. (2 pts) Given a new data point  $x \in \mathbb{R}^D$ , write out the prediction based on the closed form solution yielded before.

Solution

$$y = \mathbf{W}_b x = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{T} x$$

## Part 2. Implementation (12 pts)

Below is a list of classes and functions we will implement. Implement wherever the code template says `pass`. Reading `tests.py` might help you debug your implementation.

- **LinearReg.fit( $\mathbf{X}$ ,  $\mathbf{T}$ ) (6 pts)**: The linear regression model fits to the training set  $(\mathbf{X}, \mathbf{T})$ , i.e., finding the optimal weight matrix  $\mathbf{W}$ . Available unit tests: `TestFit`
- **LinearReg.predict( $\mathbf{X}$ ) (3 pts)**: The prediction the linear regression model makes using the learned weights. Available unit tests: `TestPredict`.
- **second\_order\_basis( $x$ ) (3 pts)**:  
The second-order polynomial basis function  $\phi(\mathbf{x} = [x_1, x_2, \dots, x_D])$ , gives you:

$$\phi(\mathbf{x} = [x_1, x_2, \dots, x_D]) = [x_1^2, x_1 x_2, \dots, x_1 x_D, x_2^2, x_2 x_3, \dots, x_2 x_d, \dots, x_d^2]$$

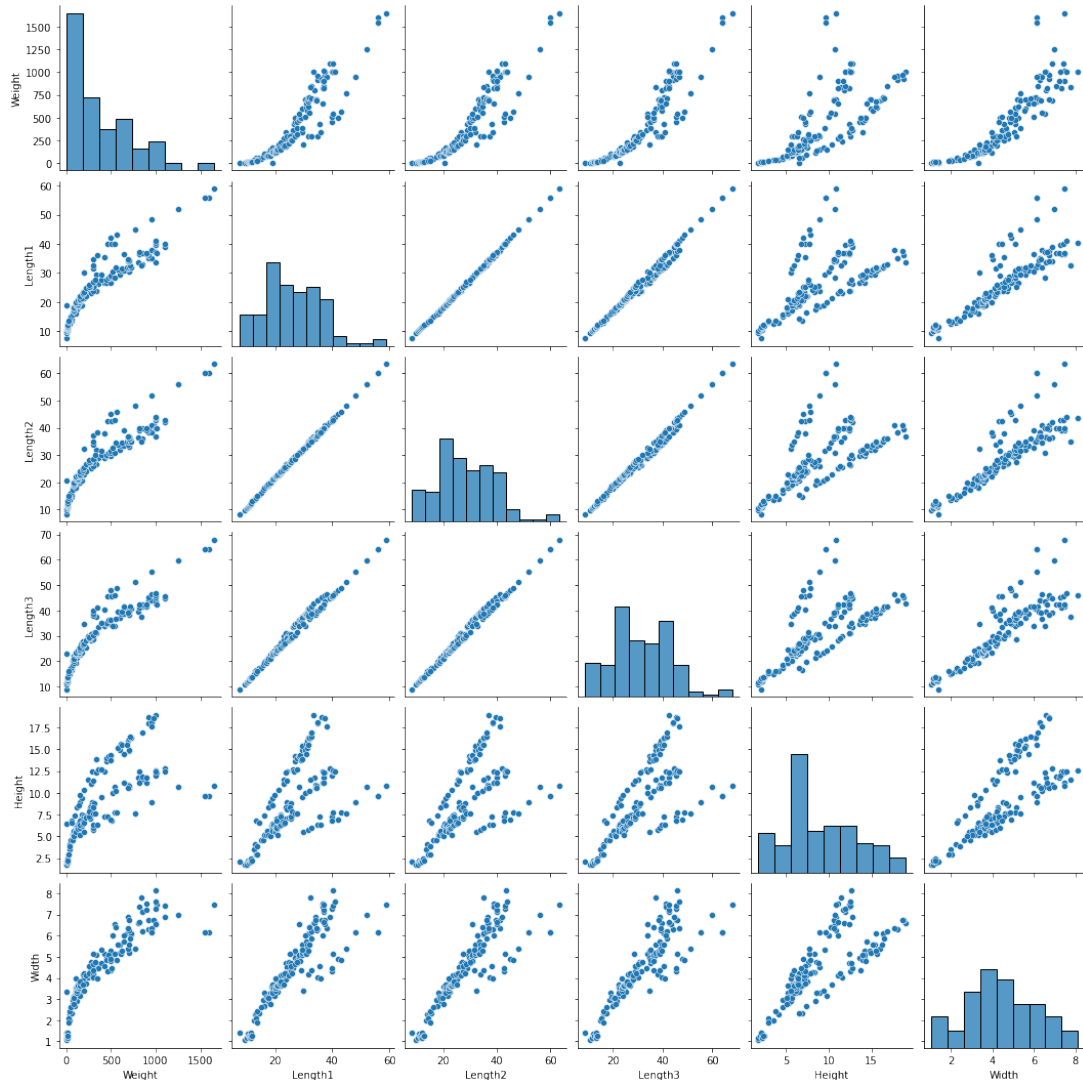
Note: the output doesn't need to be of the same order as above.

### Part 3. Experiments (18 pts)

In this section, we will train and test the linear regression model we implemented, and evaluate their performances. Observe and pre-process the data before you apply your model to it. In this section, you don't need to submit the code, only report the results you get.

1. (2 pts) Observe the data by plotting the pairwise correlation of all the features. Show the result and describe what you see.

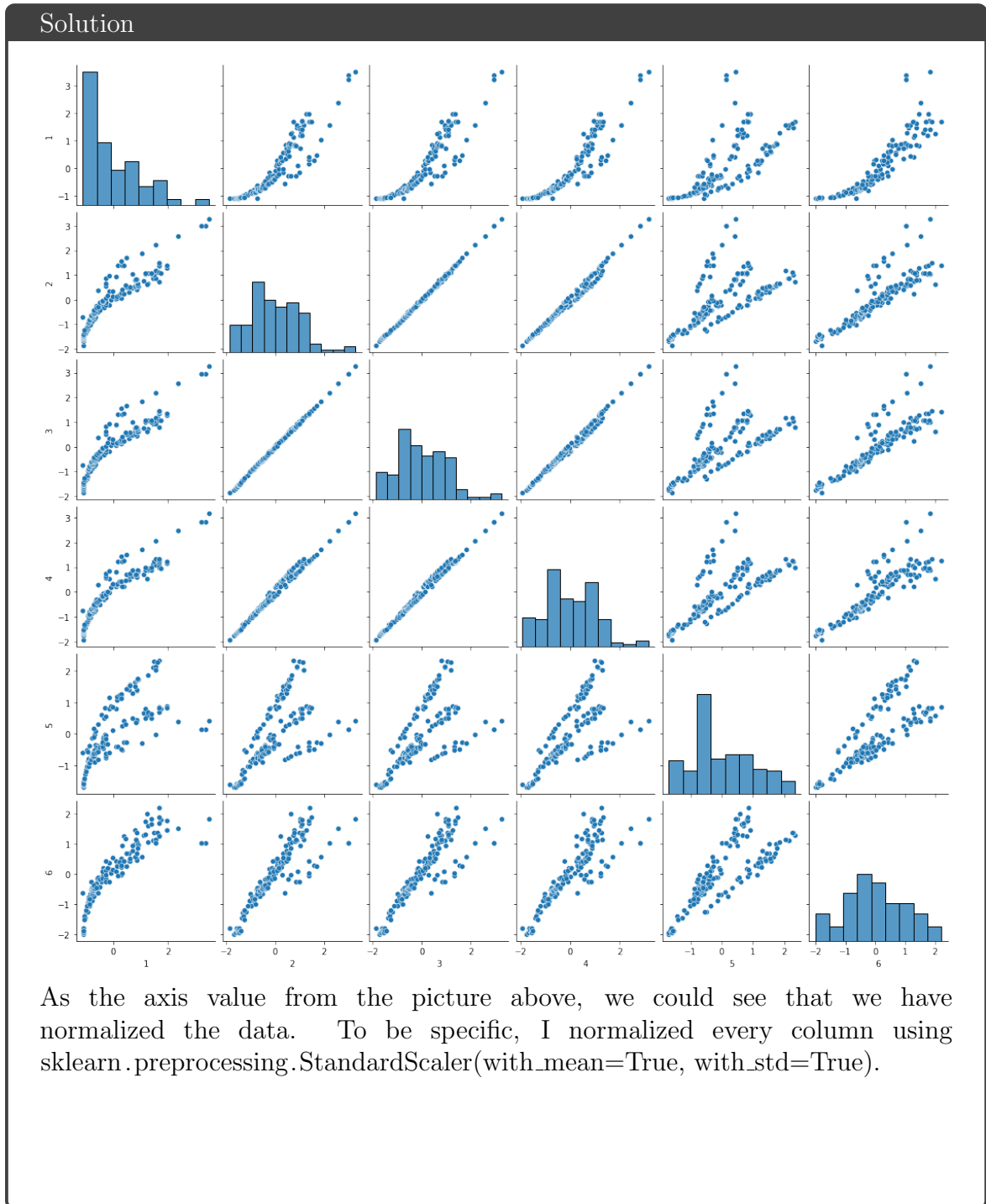
#### Solution



- (a) There are very obvious correlation among length 1, length 2, and length 3. The value of correlation coefficient between them is very close to 1.
- (b) We could also find some correlation relationships between other dimensions, but they are weaker.

2. (2 pts) **Normalize** (some times this is called **standardization**) the data by the following formula and compare the results.

$$\mathbf{x}' = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sqrt{\text{var}[\mathbf{x}]}}$$



3. (6 pts) Use the one-hot encoding to encode the species of the fish, compare and discuss the results of other two measures:
- Discard the information of species.
  - Use a scaler to encode the species, e.g., Perch=0, Bream=1,  $\dots$ .

### Solution

```
# We will make use of the species for regression
# by one-hot encoding.
data_scaled_one_hot = pd.get_dummies(data_scaled.Species, prefix='Species')
data_scaled_one_hot.head()
```

	Species_Bream	Species_Parkki	Species_Perch	Species_Pike	Species_Roach	Species_Smelt	Species_Whitefish
0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0

```
data_scaled_one_hot.tail()
```

	Species_Bream	Species_Parkki	Species_Perch	Species_Pike	Species_Roach	Species_Smelt	Species_Whitefish
154	0	0	0	0	0	1	0
155	0	0	0	0	0	1	0
156	0	0	0	0	0	1	0
157	0	0	0	0	0	1	0
158	0	0	0	0	0	1	0

- If we discard the information of species, we will lose the information of labels, which will make it impossible for us to do classification task later. Therefore, we can not discard the name of different species.
- If we use a scaler to encode the species, although we do keep the information of species maintained, it is still an unwise method. That is because later when we have trained an appropriate model, which when we input a set of numbers representing Length, Height, and other information, will output a set of possibilities representing the possibility of different species of this fish. The shape of this possibility array is  $[1, \text{number\_of\_species}]$ , and it is the same as the shape of the one-hot encoding form of each specie. Moreover, if we intend to predict other features instead of the species, it is important to unify the magnitude of feature "Species". Using 1, 2, 3,  $\dots$  will absolutely introduce more noise to the data, which might be harmful.
- In summary, one-hot is the most convenient and reasonable choice, like the figure above.

4. (8 pts) Fit the data using the **LinearReg** we implemented before, compare the training loss and the testing loss. What's your conclusion? Can you try to alleviate the identified problem?

#### Solution

- (a) training\_loss  $\rightarrow$  0.06233668660413258
- (b) testing\_loss  $\rightarrow$  0.07599127005373792
- (c) Personally, the result is good enough, but definitely it could be improved. In many cases, Least Squares Regression (LSR) could perfectly solve the problem, but due to the linearity, the ability of linear regression is limited. Some simple non-linear task such as OR operation could not be well solved by linear regression.
- (d) In future sections, I will use more linearity models as well as non-linearity models, and even neural networks to solve this problem. Hope I will get lower loss.



5. **(5 pts BONUS)** Do some further analysis on your model, try to improve it with any tricks you can think of. You can use the second-order polynomial basis functions for this bonus question.

#### Solution

Using the second-order polynomial basis functions:

- (a) training\_loss  $\rightarrow$  0.007369456930476613
- (b) testing\_loss  $\rightarrow$  0.03773699994208539
- (c) Obviously, there is a significant decrease in MSE loss of testing data.

Other solutions:

- (a) I have tried several linear regression models, the test loss are below:
  - LinearRegression  $\rightarrow$  0.07599127005376081
  - Ridge  $\rightarrow$  0.06674792314167687
  - RidgeCV  $\rightarrow$  0.05940370767567825
  - SGDRegressor  $\rightarrow$  0.06264622101036103
  - ElasticNet  $\rightarrow$  0.23724292194861565
  - ElasticNetCV  $\rightarrow$  0.07011677675984486
- (b) Also, I tried Decision Tree, Random Forest, K Neighbors, Gradient Boosting, and SVR:
  - DecisionTreeRegressor  $\rightarrow$  0.04903785914820777
  - GradientBoostingRegressor  $\rightarrow$  0.028634178077987002
  - KNeighborsRegressor  $\rightarrow$  0.034789994040148244
  - RandomForestRegressor  $\rightarrow$  0.03880698748980298
  - SVR  $\rightarrow$  0.015913159612975993
- (c) In order to get lower loss, I also applied some deep learning technologies, like neural networks. For this task, the model I built is:
  - (0): Linear(in\_features=12, out\_features=64, bias=True)
  - (1): ReLU()
  - (2): Linear(in\_features=64, out\_features=32, bias=True)
  - (3): ReLU()
  - (4): Linear(in\_features=32, out\_features=1, bias=True)
  - L1Loss and Adam  $\rightarrow$  0.011546974779458131
  - L1Loss and SGD  $\rightarrow$  0.00728513199862852
  - MSE and Adam  $\rightarrow$  0.01958152014774434
  - MSE and SGD  $\rightarrow$  0.010919815335035075

## Problem 2 (60 pts)

In this problem we will implement the linear regression model and apply it to a simple problem. The data we use here is the The Cleveland Heart Disease Dataset<sup>3</sup>.

### Part 1. Derivation (20 pts)

1. (3 pts) Derive the derivatives of the logistic function (sigmoid function):

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Solution

$$\begin{aligned}\sigma(x) &= \frac{1}{1 + e^{-x}} \\ \Rightarrow \sigma'(x) &= \frac{-(-e^{-x})}{(1 + e^{-x})^2} \\ &= \frac{e^{-x}}{(1 + e^{-x})^2} \\ &= \frac{e^{-x}}{1 + e^{-x}} \cdot \frac{1}{1 + e^{-x}} \\ &= (1 - \sigma(x)) \cdot \sigma(x)\end{aligned}$$

---

<sup>3</sup><https://archive.ics.uci.edu/ml/datasets/heart+disease>

2. (7 pts) In logistic regression, given the dataset  $D = \{\mathbf{x}_n, t_n\}$ , where  $t_n \in \{-1, 1\}$ . We use the sigmoid function  $\sigma(\cdot)$  to model the probability (likelihood) of a data sample belonging to the “positive class”, i.e.,  $t_n = 1$ :

$$P(t_n = 1 | \mathbf{x}_n, \mathbf{w}, w_0) = \sigma(\mathbf{w}^\top \mathbf{x}_n + w_0) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x}_n + w_0)}}.$$

Since  $P(t_n = 1 | \mathbf{x}_n, \mathbf{w}, w_0) + P(t_n = -1 | \mathbf{x}_n, \mathbf{w}, w_0) = 1$ , we can write the likelihood in a unified form (details are omitted):

$$P(t_n | \mathbf{x}_n, \mathbf{w}, w_0) = \sigma(t_n(\mathbf{w}^\top \mathbf{x}_n + w_0))$$

The loss function of logistic regression is the negative log-likelihood function of the dataset, which is defined as:

$$\mathcal{L} = - \sum_{n=1}^N \log \sigma(t_n(\mathbf{w}^\top \mathbf{x}_n + w_0)).$$

Please derive the derivatives of the loss function  $\mathcal{L}$ .

**Tips:** You can re-write the  $\mathbf{w}^\top \mathbf{x}_n + w_0$  part into a simple dot product as in Problem 1.

Solution

$$\begin{aligned} L &= - \sum_{n=1}^N \log \sigma(t_n(\mathbf{w}^\top \mathbf{x}_n + w_0)) \\ &= - \sum_{n=1}^N \log \sigma(t_n \cdot X_n \cdot W^\top) \\ \Rightarrow L' &= - \sum_{n=1}^N \frac{\sigma'(t_n \cdot X_n \cdot W^\top)}{\sigma(t_n \cdot X_n \cdot W^\top)} \cdot t_n \cdot X_n \\ &= - \sum_{n=1}^N \frac{(1 - \sigma(t_n X_n W^\top)) \cdot \sigma(t_n X_n W^\top)}{\sigma(t_n X_n W^\top)} \cdot t_n \cdot X_n \\ &= - \sum_{n=1}^N [1 - \sigma(t_n X_n W^\top)] \cdot t_n \cdot X_n \\ &= \sum_{n=1}^N \sigma(t_n \cdot X_n \cdot W^\top) \cdot t_n \cdot X_n - t_n \cdot X_n \\ &= \sum_{n=1}^N \frac{t_n \cdot X_n}{1 + e^{-t_n X_n W^\top}} - t_n \cdot X_n \end{aligned}$$

3. (10 pts) Show that the loss function of logistic regression is a convex function, where the function is defined as:

$$\mathcal{L}(\mathbf{w}, w_0) = - \sum_{n=1}^N \log \sigma(t_n(\mathbf{w}^\top \mathbf{x}_n + w_0)).$$

**Tips:**

- The definition of a convex function:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y), \quad \forall \lambda$$

- The addition of two convex functions is also convex.

**Solution**

Since we have already get the first-order derivatives of  $\mathcal{L}$ , if we want to show the convexity of it, we need to show the second-order derivatives of  $\mathcal{L}$  is always positive.

- First, since the derivative of the sigmod function exists. The second-order derivative of  $\mathcal{L}$  exists also.
- Now let us find the second-order derivative of  $\mathcal{L}$ :

$$\begin{aligned} L' &= \sum_{n=1}^N \sigma(t_n X_n W^\top) \cdot t_n \cdot X_n \\ \Rightarrow L'' &= \sum_{n=1}^N [1 - \sigma(t_n X_n W^\top)] \cdot \sigma(t_n X_n W^\top) \cdot (t_n \cdot X_n)^2 \\ &\quad \sigma(t_n X_n W^\top) \in (0, 1), \quad (t_n \cdot X_n)^2 \geq 0 \\ \Rightarrow 1 - \sigma(t_n X_n W^\top) &> 0, \quad \sigma(t_n X_n W^\top) > 0 \\ \Rightarrow L'' &> 0 \end{aligned}$$

- In summary, the loss function of logistic regression is convex.

## Part 2. Implementation (15 pts)

Below is a list of classes and functions we will implement. Implement wherever the code template says `pass`. Reading `tests.py` might help you debug your implementation.

- **LogisticReg.fit( $X$ ,  $t$ ) (12 pts)**: The logistic regression model fits to the training set  $(X, t)$ , i.e., finding the optimal weight vector  $w$ .
  - **LogisticReg.compute\_loss( $X$ ,  $t$ ) (4 pts)**: Compute and return the (average) loss value given a training batch. Available unit tests: `TestLoss`.
  - **LogisticReg.compute\_gradient( $X$ ,  $t$ ) (6 pts)**: Compute and return the (average) gradient value given a training batch. Available unit tests: `TestGrad`.
  - **LogisticReg.update(grad, lr) (2 pts)**: Update the weight vector given the gradient `grad` and the learning rate `lr`. Available unit tests: `TestUpdate`.
- **LogisticReg.predict( $X$ ) (3 pts)**: The prediction the logistic regression model makes using the learned weights. Available unit tests: `TestPredict`.

### Part 3. Experiments (25 pts)

In this section, we will train and test the logistic regression model we implemented, and evaluate its performance. Observe and pre-process the data before you apply your model to it. In this section, you don't need to submit the code, only report the results you get.

1. (2 pts) Pre-process the data following the guide. Explain the difference between `fit_transform` and `transform` method of the `StandardScaler` in sklearn.

#### Solution

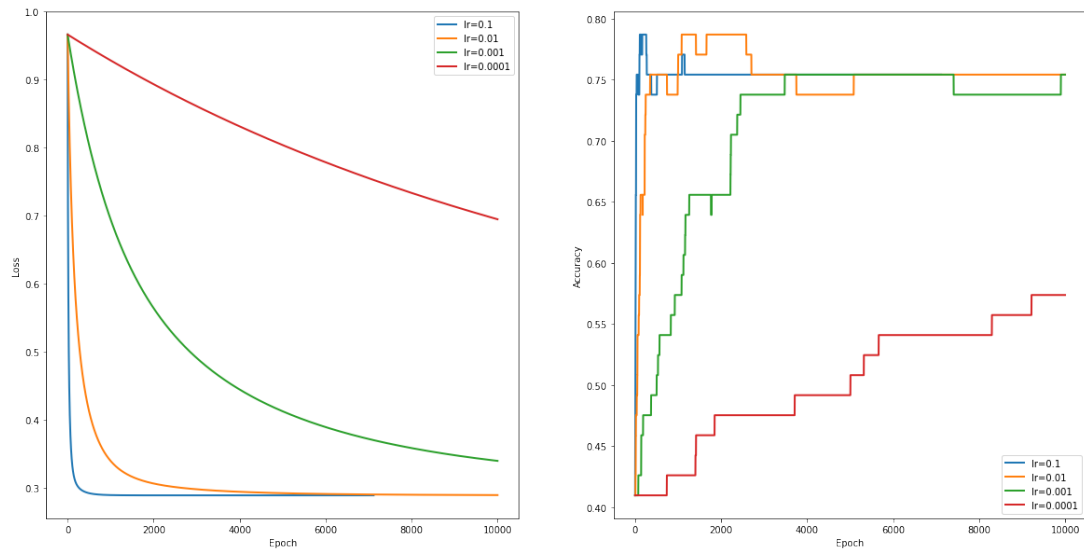
- (a) In the `fit()` method, where we use specific algorithms to fit the model based on the input data. After this process we will get the model with appropriate parameters.
- (b) For changing the data as what we want through the model, we could use `transform()` method to apply the calculations that we have calculated in `fit()` to every data point. **It should be taken attention to that we can not use `transform()` method without applying `fit()` before it.**
- (c) This `fit_transform()` method is basically the combination of `fit` method and `transform` method, it is equivalent to `fit().transform()`.

2. (10 pts) Train the logistic regression model with  $lr=0.1$ ,  $0.01$ ,  $0.001$ ,  $0.0001$  for 10000 epochs. Report the following:

- The loss trajectory of the training and testing set;
- The accuracy trajectory of the training and testing set.

Write down your conclusion in the solution panel.

### Solution



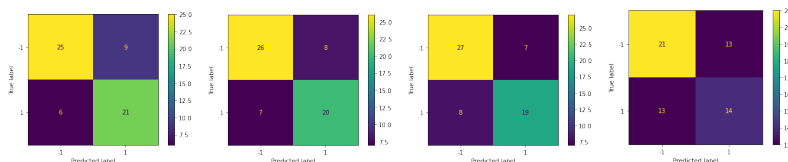
- (a) Like the figure above, the blue curve is plotted when  $lr=0.1$ , the orange one is for  $lr=0.01$ , the green one is for  $lr=0.001$ , and the red curve is plotted when  $lr=0.0001$ .
- (b) We could easily find that when learning rate is bigger, the velocity of loss decreasing and accuracy increasing is faster. There is also an interesting point. The optimum point could only be reached with  $lr=0.1$  and  $lr=0.01$ .

3. (6 pts) Report the performances of the logistic regression model, evaluated under the metrics other than simple accuracy, including

- Confusion Matrix
- F1-Score
- AUC-ROC curve

Explore the relationship between the threshold (we decide a sample belongs to the positive class if  $p > \text{threshold}$ ) and the different metrics.

### Solution



The confusion matrix is above, and the F1 score and roc-auc-score is below:

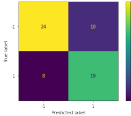
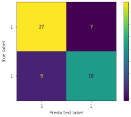
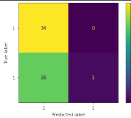
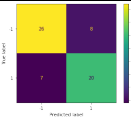
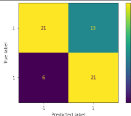
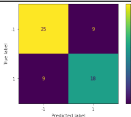
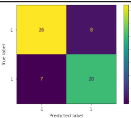
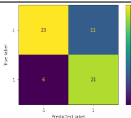
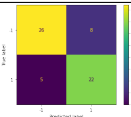
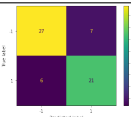
Learning Rate	F1 Score	ROC-AUC Score
0.1	0.7368421052631577	0.7565359477124183
0.01	0.7272727272727273	0.7527233115468408
0.001	0.7169811320754716	0.7489106753812637
0.0001	0.5185185185185185	0.568082788671024

Below is the relationships between the threshold and the different metrics:

- When the threshold getting larger, the True-Negative value will be larger, while the True-Positive value will significantly decrease. The False-Negative value will increase and the False-Positive value will decrease because there are less data samples being predicted as positive.
- When the threshold getting smaller, the True-Negative value will be smaller, while the True-Positive value will significantly increase. The False-Negative value will decrease and the False-Positive value will increase because there are more data samples being predicted as positive.
- Personally, since our data is almost averagely distributed, the value of F1 score and ROC-AUC score will decrease.



4. (7 pts) Use the sklearn library to try out other classification models and compare them with the logistic regression model.

Solution			
Model	Confusion Matrix	F1 Score	ROC-AUC Score
Nearest Neighbors		0.6785714285714286	0.7047930283224401
Linear SVM		0.6923076923076923	0.7303921568627451
RBF SVM		0.07142857142857142	0.5185185185185185
Gaussian Process		0.7272727272727273	0.7527233115468408
Decision Tree		0.6885245901639345	0.6977124183006537
Random Forest		0.6666666666666666	0.7009803921568628
Neural Net		0.7272727272727273	0.7527233115468408
AdaBoost		0.711864406779661	0.7271241830065359
Naive Bayes		0.7719298245614035	0.7897603485838779
QDA		0.7636363636363638	0.7859477124183007

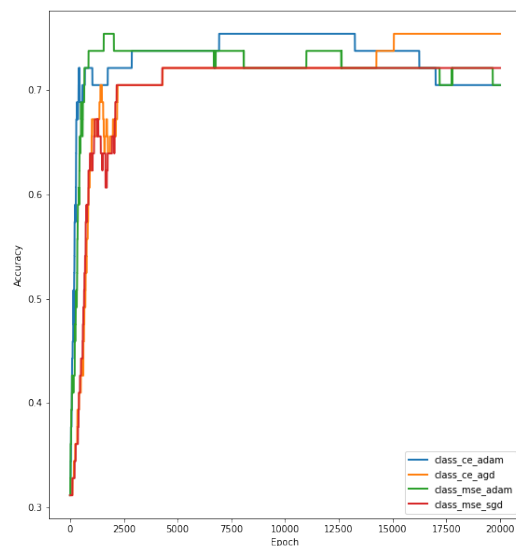
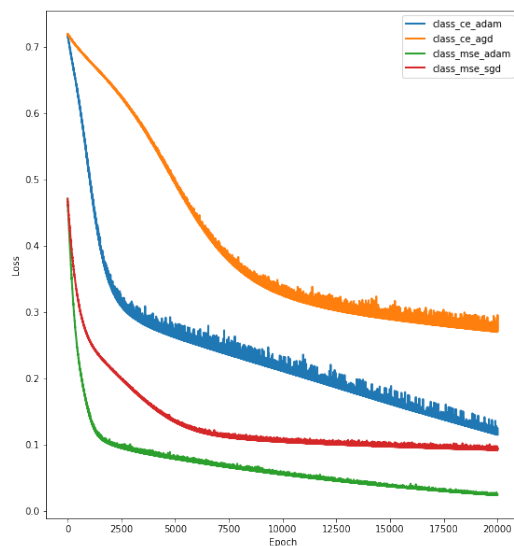
5. (5 pts BONUS) Do some further analysis on your model, try to improve it with any tricks you can think of.

**Tips:**

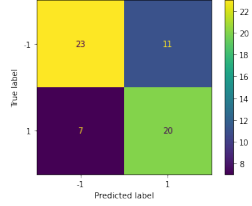
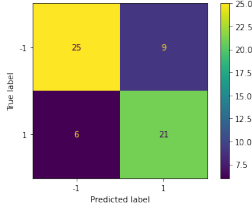
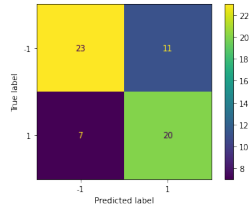
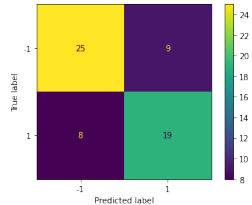
- l2-regularization to overcome the problem of overfitting
- Newton's method to help the logistic regression model converge faster

**Solution**

```
MLP(  
  (layers): Sequential(  
    (0): Linear(in_features=13, out_features=64, bias=True)  
    (1): ReLU()  
    (2): Linear(in_features=64, out_features=32, bias=True)  
    (3): ReLU()  
    (4): Linear(in_features=32, out_features=2, bias=True)  
  )  
)
```



- (a) In order to get high accuracy and lower loss, I applied neural networks implemented in PyTorch.
- (b) I have tried four corporations of loss functions and optimizes in total, the model structure as well as the loss and accuracy log is above.
- (c) Below are the Confusion Matrix, F1 score, and ROC-AUC Score

Model	Confusion Matrix	F1 Score	ROC-AUC Score
CrossEntropy and Adam		0.689655172413793	0.7086056644880174
CrossEntropy and SGD		0.7368421052631577	0.7565359477124183
MSELoss and Adam		0.689655172413793	0.7086056644880174
MSELoss and SGD		0.6909090909090909	0.7194989106753813

**Collaboration Questions** Please answer the following:

After you have completed all other components of this assignment, report your answers to the the following collaboration policy questions.

1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details.
3. Did you find or come across code that implements any part of this assignment ? If so, include full details even if you have not used that portion of the code.

**Solution**

1. No.
2. No.
3. No.