Homework 1
Linear Regression and Logistic Regression[1]

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: **YOUR NAME HERE**

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

[1]Compiled on Sunday 17 April, 2022 at 21:18

found 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[2].

Part 1. Derivation (10 pts)

1. **(8 pts)** Suppose the input feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ and the target output matrix $\boldsymbol{T} \in \mathbb{R}^{N \times d}$, 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:

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

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

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

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

Solution

[2] 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

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($\boldsymbol{X}$, $\boldsymbol{T}$)** (**6 pts**): The linear regression model fits to the training set (X, T), i.e., finding the optimal weight matrix W. Available unit tests: `TestFit`
- **LinearReg.predict($\boldsymbol{X}$)** (**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 $\phi(\boldsymbol{x} = [x_1, x_2, \cdots, x_D])$, gives you:

  $$\phi(\boldsymbol{x} = [x_1, x_2, \cdots, x_D]) = [x_1^2, x_1x_2, \cdots, x_1x_D, x_2^2, x_2x_3, \cdots, x_2x_d, \cdots, x_d^2]$$

  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

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

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

Solution

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, $\cdots$.

Solution

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

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

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[3].

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

[3] 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., $t_n = 1$:

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

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):

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

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

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

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

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

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

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

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($\boldsymbol{X}$, t)** (**12 pts**): The logistic regression model fits to the training set ($\boldsymbol{X}$, **t**), i.e., finding the optimal weight vector w.
  - **LogisticReg.compute_loss($\boldsymbol{X}$, t)** (**4 pts**): Compute and return the (average) loss value given a training batch. Available unit tests: `TestLoss`.
  - **LogisticReg.compute_gradient($\boldsymbol{X}$, 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($\boldsymbol{X}$)** (**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

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

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 $p >$ threshold) and the different metrics.

Solution

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

Solution

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

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