Lab 5: Regression & Neural Networks

> CS410: Artificial Intelligence
> Shanghai Jiao Tong University, Fall 2021
> Due Time: 2021.12.18 23:59

Exercise 1: Linear Regression

Consider a linear regression model $y = \theta^\top x,$ where $y \in \mathbb{R}$, $\theta \in \mathbb{R}^5$ and $x \in \mathbb{R}^5$. Recall the geometric interpretation of linear regression given in Lecture 8, Slide 18 that the predicted value $\hat{Y} = X\hat{\theta} = X\left(X^\top X\right)^{-1} X^\top Y$ could be viewed as the projection of the true value Y onto the subspace spanned by the column vectors in X, where $X \in \mathbb{R}^{100\times 5}$ and $Y \in \mathbb{R}^{100}$. First load the data using `np.load('Data1_X.npy')` and `np.load('Data1_Y.npy')` to get X and Y respectively. Then compute $\hat{Y}$ and verify the aforementioned geometric interpretation with reasonable discussions.

Exercise 2: Logistic Regression

Consider a binary classification problem with 100 samples, each of which has a 2-dimensional feature $x_i \in \mathbb{R}^2$ and a binary label $y_i \in \{0, 1\}$. The task in this exercise is to solve this binary classification problem using the logistic regression model introduced in Lecture 8, Slide 67.

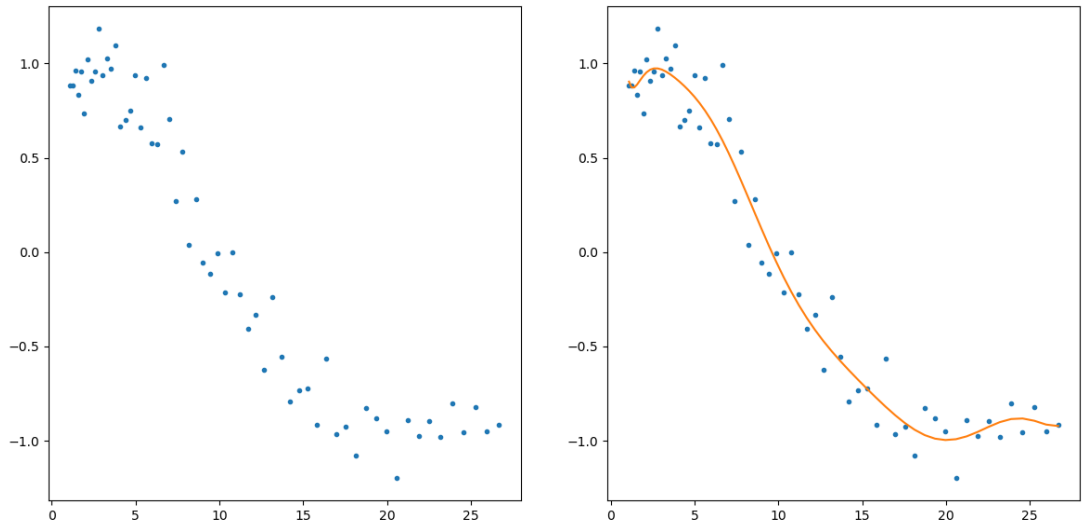
1. First load the data using `np.load('Data2_X.npy')` and `np.load('Data2_Y.npy')` to get X and Y respectively, where $X \in \mathbb{R}^{100\times 2}$ and $Y \in \{0, 1\}^{100}$. Then finish the `CELoss_binary()` function to compute the binary cross entropy loss and `gradient()` to compute the mini-batch gradient of the binary cross entropy loss function with respect to the parameter θ for a given size of the mini-batch.
2. Train your logistic regression model. Plot the binary cross entropy loss and the precision (defined in Lecture 8, Slide 69) against the number of iterations using stochastic gradient descent, mini-batch gradient descent and (batch) gradient descent respectively under 3 different learning rates and 3 different values of the threshold. Discuss your findings about the effects of learning rates, types of gradients and the values of the threshold to the loss and precision.
3. Visualize the decision boundary of predictions of your model with different values of threshold and discuss the impact of thresholds on the decision boundary as in Lecture 8, Slide 72.

Important:

- Using third-party libraries that incorporate the logistic regression model (e.g., scikit-learn) or integrate the automatic differentiation (e.g., PyTorch, TensorFlow) is not allowed in this exercise.

Exercise 3: L1/L2 Regularization

Consider a toy regression problem as shown below, where we have a simulated sine curve (between $60°$ and $300°$) with Gaussian noise (details in code). We are going to estimate the sine function using polynomial regression ($n = 16$), i.e., a model of the form $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon$. Essentially, we are applying a linear regression model for 16-d input data.



Implement ridge regression and lasso regression with different values of λ, and answer the following questions with plots, numbers, etc.

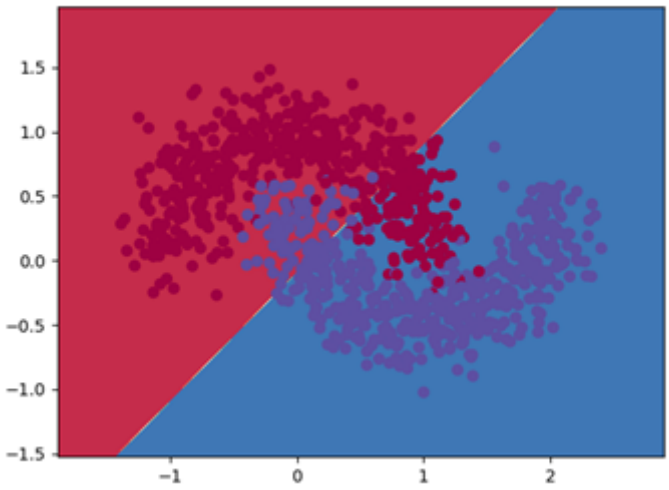
1. Which approach is less computationally expensive?
2. Which approach tends to create a sparser output?
3. What is the relationship between regularization and generalizability (especially when using a complex model)?

Important:

- Library Matplotlib is required in this exercise.
- You are allowed to use libraries such as scikit-learn in this exercise. However, make sure to import them explicitly within `ridge_regression()` and `lasso_regression()` to let us aware.
- We provide an example script for you to answer the questions. You can modify `main()` to do what you want.

Exercise 4: Two-layer Perceptron Network

Consider a toy binary classification problem as shown below, which is provided by scikit-learn.



Implement backpropagation of two-layer perceptron network with ReLU as the activation function and mean squared error as the loss function using **NumPy** (details listed in code). Change the number of hidden neurons and discuss your findings.

Important:

- Library Matplotlib and scikit-learn is required in this exercise.
- Update weights after all the gradients are calculated (i.e., with **old weights** as described in p46, Lecture 9).
- We provide an example script for you to answer the questions. You can modify `main()` to do what you want.

Submission

Here are the files you need to submit (please do **NOT** rename any file):

- `P1.py` for exercise 1, `P2.py` for exercise 2, `P3.py` for exercise 3, and `P4.py` for exercise 4.
- `report.pdf` for your **brief** report.

Compress the above three files into one `*.zip` or `*.rar` file and name it with your student ID, and submit it on Canvas.