ECE 8550: Assignment 1

1 Linear regression

In this section, you will implement linear regression to predict the death rate from several features, including annual precipitation, temperature, population, income, pollution, etc. The data for this assignment is given in file `data.txt`, which contains the data description, 17 columns (features) and 60 rows (examples). In the data matrix, columns 2-16 are input features, and column 17 is the output target. Note that Column 1 is the index and should not be used in the regression. You will implement **gradient descent** for learning the linear regression model.

We split the data to a training set and a testing set following the 80/20 rule. Thus, we have 48 training samples ($n = 48$) with 16 features ($d = 16$). The i-th sample is denoted by $\mathbf{x}_i$ where x_i^j is the value of the j-th feature this sample.

1.1 Feature normalization

By looking at the feature values in the data, you may notice that some features are about 1000 times the other features. When features differ by orders of magnitude, first performing feature normalization can make gradient descent converge much more quickly. There are different ways to do the feature normalization. For simplicity, we use the following method: (1) subtract the minimal value of each feature; (2) divide the feature values by their ranges of values. Equivalently, it is given by the following equation:

$$\text{for each } X, \text{ let } X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Similarly normalize the target Y. Note that to make prediction for a new data point, you also need to first normalize the features as what you did previously for the training dataset.

1.2 Feature appending

Classic regression is defined as

$$f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i + b = \sum_{j=1}^{d} w^j \times x_i^j + b,$$

where b_i is the intercept. For simplicity, we let $b = w^{d+1} \times x_i^{d+1}$ where $x_i^{d+1} = 1$ and w^{d+1} is unknown but learnable parameter. Thus, we can combine $\mathbf{w}$ and b via letting

$$\begin{aligned}\mathbf{w} \leftarrow &[w^1,\ w^2,\ w^3,\ \ldots,\ w^d, w^{d+1}]^T;\\ \mathbf{x}_i \leftarrow &[x_i^1,\ x_i^2,\ x_i^3,\ \ldots,\ x_i^d, 1]^T.\end{aligned}$$

1.3 Gradient descent with ridge regularization

To recap, the prediction function for linear regression is given by

$$f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i = \mathbf{w}^T \mathbf{x}_i = \sum_{j=1}^{d+1} w^j \times x_i^j, \tag{1}$$

where

- $\mathbf{x}_i$ is the feature vecter of the i-th sample;
- x_i^j is the value of the j-th feature for the i-th sample;
- w_j is the j-th parameter of $\mathbf{w}$;

The loss function for linear regression with ridge regularization is given by

$$J(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(f(\mathbf{x}_i) - y_i\right)^2 + \frac{\lambda}{2(d+1)} \sum_{j=1}^{d+1} (w^j)^2. \tag{2}$$

To minimize this function, we first obtain the gradient with respect to each parameter w_j as:

$$\frac{\partial J(\mathbf{w})}{\partial w^j} = \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathbf{x}_i) - y_i\right) x_i^j + \frac{\lambda}{d+1} w^j. \tag{3}$$

Then, the (full) gradient descent algorithm is given as:

where α is the learning rate. The *convergence criteria* for the above algorithm is $\Delta_{\%cost} < \epsilon$, where

$$\Delta_{\%cost} = \frac{|J_{k-1}(\mathbf{w}) - J_k(\mathbf{w})| \times 100}{J_{k-1}(\mathbf{w})},$$

Algorithm 1: Batch Gradient Descent

$k = 0$;
while *convergence criteria not reached* **do**
  for $j = 1, \ldots, m$ **do**
    Update $w^j \leftarrow w^j - \alpha \frac{\partial J(\mathbf{w})}{\partial w^j}$
  Update $k \leftarrow k + 1$;

where $J_k(\mathbf{w})$ is the value of Eq. (2) at k-th iteration, and $\Delta_{\%cost}$ is computed at the end of each iteration of the while loop. Initialize $\mathbf{w} = \mathbf{0}$ and compute $J_0(\mathbf{w})$ with these values.

Important: You must simultaneously update w^j for all j.

Task: Load the dataset in `data.txt`, split it into 80% / 20% training/test. The dataset is already shuffled so you can simply use the first 80% examples for training and the remaining 20% for test. Learn the linear regression model using the training data. Plot the value of loss function $J_k(\mathbf{w})$ vs. the number of iterations (k). For each iteration, compute the mean squared loss (w/o regularization function) on the test data. Plot the mean squared loss over the test data v.s. the number of iterations (k) in the same figure. The mean squared loss is defined as

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} \left[y_i - \mathbf{w} \cdot \mathbf{x}_i \right]^2 .$$

1.4 Gradient descent with lasso regularization

The loss function for linear regression with lasso regularization is given by

$$J(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i) \right)^2 + \frac{\lambda}{2(d+1)} \sum_{j=1}^{d+1} |w^j|. \tag{4}$$

To minimize the loss function, you will need to derive the gradient by yourself. The gradient descent algorithm is the same as the above.

Hint: For simplicity, you can consider $\frac{\partial |w^j|}{w^j} = 1$ when $w^j = 0$.

Task: Load the dataset in `data.txt`, split it into 80% / 20% training/test,. Learn the lasso regression model using the training data. Plot the value of loss function $J_k(\mathbf{w})$ vs. the number of iterations (k). Plot the mean squared loss over the test data v.s. the number of iterations (k) in the same figure.

1.5 What to submit

In this assignment, use $\alpha = 0.01$ and $\epsilon = 0.001$. Use $\lambda = 2$ for ridge regularization, and use $\lambda = 0.3$ for lasso regularization.

1. (50 pts) The source code (.py) without running errors in Anaconda 2021.11 (Python 3.9).
2. (20 pts) Plot the value of loss function $J_k(\mathbf{w})$ and MSE vs. the number of iterations (k) for Section 1.2, report the squared loss on the test data for Section 1.3.
3. (10 pts) The number of elements in $\mathbf{w}$ whose absoluate value is smaller than 0.01 for Section 1.3.
4. (10 pts) Equation for the gradient of Eq. (4) which should be similar to Eq. 3.
5. (10 pts) Plot the value of loss function $J_k(\mathbf{w})$ and MSE vs. the number of iterations (k) for Section 1.3, report the squared loss on the test data for Section 1.4.
6. (10 pts) The number of elements in $\mathbf{w}$ whose absoluate value is smaller than 0.01 for Section 1.4.

1.6 Requirements

1. This assignment is implemented using Python in Anaconda 2021.11 (Python 3.9).
2. No libraries are allowed except Python built-ins, numpy, and matplotlib.
3. Figures, equations, and numbers should be in a PDF file.
4. Zip the Python source code (.py) and your PDF file before uploading.