CSCE 5063-001: Assignment 1

Due 11:59pm Friday, February 8, 2019

1 Linear regression

In this problem, you will implement linear regression to predict the death rate from other variables, 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). Columns 2-16 are input features, and column 17 is the output feature. 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.

1.1 Feature normalization

By looking at the feature values in the data, note 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. The method is:

1. Subtract the mean value of each feature from the dataset.
2. After subtracting the mean, additionally divide the feature values by their ranges of values (max-min) (an alternative is to take their respective standard deviations).

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 Gradient descent with quadratic regularization

To recap, the loss function for linear regression with quadratic regularization is given by

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)}) \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2, \tag{1}$$

where $h_\theta(x^{(i)}) = \theta^T x^{(i)}$ is the linear regression model. To minimize this function, we first obtain the gradient with respect to each parameter θ_j as follows:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j.$$

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

Algorithm 1: Batch Gradient Descent

```
k = 0;
while convergence criteria not reached do
    for j = 1, ..., n do
        Update θ_j ← θ_j − α ∂J(θ)/∂θ_j;
    Update k ← k + 1;
```

where α is the learning rate. Note that you need to simultaneously update θ_j for all j. The *convergence criteria* for the above algorithm is $\Delta_{\%cost} < \epsilon$, where

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

where $J_k(\theta)$ is the value of Eq. (1) at kth iteration, and $\Delta_{\%cost}$ is computed at the end of each iteration of the while loop. Initialize $\theta = \mathbf{0}$ and compute $J_0(\theta)$ with these values.

Task: Load the dataset in `data.txt`, split it into 80% / 20% training/test, and learn the linear regression modeling using the training data. Plot the value of loss function $J_k(\theta)$ vs. the number of iterations (k). After the training completes, compute the squared loss (w/o regularization function) on the test data.

1.3 Gradient descent with lasso regularization

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

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} |\theta_j|. \tag{2}$$

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 $\frac{\partial |\theta_j|}{\theta_j} = 1$ when $\theta_j = 0$.

Task: Load the dataset in `data.txt`, split it into 80% / 20% training/test, and learn the linear regression modeling using the training data. Round each parameter to 0 if its absolute value is smaller than 5×10^{-3}. Compare the number of non-zero parameters of the linear regression models obtained in Sections 1.2 and 1.3.

1.4 What to submit

In this assignment, use $\alpha = 0.01$, $\lambda = 1$, and $\epsilon = 0.1$.

1. Plot the value of loss function $J_k(\theta)$ vs. the number of iterations (k) for Section 1.2.
2. The squared loss on the test data for Section 1.2.
3. Equation for the gradient of Eq. (2).
4. Numbers of non-zero parameters of the models obtained in Sections 1.2 and 1.3.
5. The source code (.java, .py, .r, or .m files).

2 Decision tree

In this problem, we want to construct a decision tree to find out if a person will enjoy beer.

Definitions: Let there be k binary-valued attributes in the data. We pick an attribute that maximizes the gain at each node using Gini index:

$$G = I(D) - (I(D_L) + I(D_R)), \tag{3}$$

where D is the given dataset, and D_L and D_R are the subsets on left and right hand-side branches after splitting. Ties may be broken arbitrarily. We define $I(D)$ as follows:

$$I(D) = |D| \times \sum_i p_i(1 - p_i) = |D| \times \left(1 - \sum_i p_i^2\right),$$

where $|D|$ is the number of items in D; p_i is the probability distribution of the items in D, or in other words, p_i is the fraction of items that take value $i \in \{+, -\}$. Put differently, p_+ is the fraction of positive items and p_- is the fraction of negative items in D.

Task: Let $k = 3$. We have three binary attributes that we could use: "likes wine", "likes running" and "likes pizza". Suppose the following:

- There are 100 people in sample set, 60 of whom like beer and 40 who don't.
- Out of the 100 people, 50 like wine; out of those 50 people who like wine, 30 like beer.
- Out of the 100 people, 30 like running; out of those 30 people who like running, 20 like beer.
- Out of the 100 people, 80 like pizza; out of those 80 people who like pizza, 50 like beer.

What are the values of G (defined in Eq. (3)) for wine, running and pizza attributes? Which attribute would you use to split the data at the root if you were to maximize the gain G using the Gini index metric defined above?

2.1 What to submit

- Values of G for wine, running and pizza attributes.
- The attribute you would use for splitting the data at the root.