CSE474/574: Introduction to Machine Learning(Fall 2017)

Instructor: Sargur N. Srihari

********** October 7, 2017**********

Project 2: Learning to Rank using Linear Regression

Due Date: Monday, Oct 23, Report: Wednesday, Oct. 25

1 Overview

The goal of this project is to use machine learning to solve a problem that arises in Information Retrieval, one known as the Learning to Rank (LeToR) problem. We formulate this as a problem of linear regression where we map an input vector $\mathbf{x}$ to a real-valued scalar target $y(\mathbf{x}, \mathbf{w})$.

There are four tasks:

1. Train a linear regression model on LeToR dataset using a closed-form solution.
2. Train a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).
3. Train a linear regression model on a synthetic dataset using a closed-form solution.
4. Train a linear regression model on the synthetic dataset using SGD.

The LeToR training data consists of pairs of input values $\mathbf{x}$ and target values t. The input values are real-valued vectors (features derived from a query-document pair). The target values are scalars (relevance labels) that take one of three values 0, 1, 2: the larger the relevance label, the better is the match between query and document. Although the training target values are discrete we use linear regression to obtain real values which is more useful for ranking (avoids collision into only three possible values).

1.1 Linear Regression Model

Our linear regression function $y(\mathbf{x}, \mathbf{w})$ has the form:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) \tag{1}$$

where $\mathbf{w} = (w_0, w_1, .., w_{M-1})$ is a weight vector to be learnt from training samples and $\boldsymbol{\phi} = (\phi_0, .., \phi_{M-1})^\top$ is a vector of M basis functions. Assuming $\phi_0(\mathbf{x}) \equiv 1$ for whatever input, w_0 becomes the bias term. Each basis function $\phi_j(\mathbf{x})$ converts the input vector $\mathbf{x}$ into a scalar value. In this project, you are required to use the Gaussian radial basis functions

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^\top \Sigma_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j)\right) \tag{2}$$

where $\boldsymbol{\mu}_j$ is the center of the basis function and Σ_j decides how broadly the basis function spreads.

1.2 Noise model and Objective function

We use a probabilistic model in which the output target value is subject to noise. More specifically, we assume that the output has a normal distribution, with mean $y(\mathbf{x}, \mathbf{w})$ and precision β. With input samples $X = \{\mathbf{x}_1, .., \mathbf{x}_n\}$ and target values $\mathbf{t} = \{t_1, ..t_n\}$, the likelihood function has the form:

$$p\left(\mathbf{t}|X, \mathbf{w}, \beta\right) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\mathbf{w}^\top \boldsymbol{\phi}\left(\mathbf{x}_n\right), \beta^{-1}\right) \tag{3}$$

Maximizing (3) is equivalent to minimizing the sum-of-squares error

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^\top \boldsymbol{\phi}\left(\mathbf{x}_n\right)\}^2 \tag{4}$$

1.2.1 Regularization to contain over-fitting

To obtain better generalization and avoid overfitting, we add a regularization term to the error function, with the form:

$$E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}) \tag{5}$$

where the *weight decay* regularizer is

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{6}$$

The coefficient λ in (5) governs the relative importance of the regularization term.

The goal is to find $\mathbf{w}^*$ that minimizes (5). This can be done by taking the derivative of (5) with respect to $\mathbf{w}$, setting it equal to zero, and solving for $\mathbf{w}$ We consider two linear regression solutions: closed-form and stochastic gradient descent (SGD).

1.3 Closed-form Solution for w

The closed-form solution of (4), i.e., sum-of-squares error *without* regularization, has the form

$$\mathbf{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{t} \tag{7}$$

where $\mathbf{t} = \{t_1, .., t_N\}$ is the vector of outputs in the training data and $\boldsymbol{\Phi}$ is the design matrix:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

The quantity (7) is known as the Moore-Penrose pseudo-inverse of the matrix Φ.

The closed-form solution with least-squared regularization, as defined by (5) and (6) is

$$\mathbf{w}^* = (\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{t} \tag{8}$$

1.4 Stochastic Gradient Descent Solution for w

The stochastic gradient descent algorithm first takes a random initial value $\mathbf{w}^{(0)}$. Then it updates the value of $\mathbf{w}$ using

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} \tag{9}$$

where $\Delta\mathbf{w}^{(\tau)} = -\eta^{(\tau)}\nabla E$ is called the weight updates. It goes along the opposite direction of the gradient of the error. $\eta^{(\tau)}$ is the learning rate, deciding how big each update step would be. Because of the linearity of differentiation, we have

$$\nabla E = \nabla E_D + \lambda\nabla E_W \tag{10}$$

in which

$$\nabla E_D = -(t_n - \mathbf{w}^{(\tau)\top}\boldsymbol{\phi}(\mathbf{x}_n))\boldsymbol{\phi}(\mathbf{x}_n) \tag{11}$$

$$\nabla E_W = \mathbf{w}^{(\tau)} \tag{12}$$

1.5 Evaluation

Evaluate your solution on a test set using Root Mean Square (RMS) error, defined as

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N_{\mathrm{V}}} \tag{13}$$

where $\mathbf{w}^*$ is the solution and N_{V} is the size of the test dataset.

1.6 Datasets

You are required to implement linear regression on two datasets.

1. The LeToR dataset where the input vector is derived from a query-URL pair and the target value is human value assignment about how well the URL corresponds to the query.
2. A synthetic dataset generated using a mathematical formula plus some noise.

2 The LeToR Dataset

The Learning to Rank dataset is the Microsoft LETOR 4.0 Dataset. It is a package of benchmark data sets for research released by Microsoft Research Asia. The latest version, can be found at

`http://research.microsoft.com/en-us/um/beijing/projects/letor/letor4dataset.aspx`

It contains 8 datasets for four ranking settings derived from the two query sets and the Gov2 web page collection. For this project, download `MQ2007`. There are three versions for each dataset: "NULL", "MIN" and "QueryLevelNorm". In this project, only the "QueryLevelNorm" version "`Querylevelnorm.txt`" will be used. The entire dataset consists of 69623 query-document pairs(rows), each having 46 features. Here are two sample rows from the MQ2008 dataset.

```
2 qid:10032 1:0.056537 2:0.000000 3:0.666667 4:1.000000 ...  46:0.076923

    #docid = GX029-35-5894638 inc = 0.0119881192468859 prob = 0.139842

0 qid:10032 1:0.279152 2:0.000000 3:0.000000 4:0.000000 ...  46:1.000000

    #docid = GX030-77-6315042 inc = 1 prob = 0.341364
```

The meaning of each column are as follows.

1. The first column is the relevance label of the row. It takes one of the discrete values 0, 1 or 2. The larger the relevance label, the better is the match between query and document. Note that objective output y of our linear regression will give a continuous value rather than a discrete one– so as to give a fine-grained distinction.
2. The second column `qid` is the query id. It is only useful for indexing the dataset and not used in regression.
3. The following 46 columns are the features. They are the 46-dimensional input vector $\mathbf{x}$ for our linear regression model. All the features are normalized to fall in the interval of $[0, 1]$.
4. We would NOT use item `docid` (which is the ID of the document), `inc`, and `prob` in this project. So just ignore them.

3 Synthetic Dataset

The only way to truly determine the generalization power of your model would be to test it on an separate unseen dataset. Therefore we will generate synthesized data using some sort of mathematical formula

$$y = f(\mathbf{x}) + \varepsilon \tag{14}$$

where ε is noise. However the true generating mechanism is classified. We will give you a training dataset generated by the formula to train. Each data point consists of 10 features. All the target values of the training data are from a set discrete $\{0, 1, 2\}$. The objective of the regression algorithm is to map the input to a real value in the interval $[0, 2]$. The target values are discrete. But the regression algorithm learns a continuous value. In the solution we assume the output to be Gaussian distributed–which is a continuous distribution. Thus the learning algorithm will learn a continuous value; the mean of the output is going to be around 1 if outputs of 0,1, and 2 are equally likely. While most of the outputs will be in $[0, 2]$, some outputs will lie outside of $[0, 2]$ and some outputs could even be negative. We are doing this so that the output can be used for ranking rather than classification, e.g., an output of 2.7 is ranked higher than an output of 2.6.

4 Plan of Work

4.1 Tasks On The LeToR Dataset

1. **Extract feature values and labels from the data**: Process the original text data file into a Numpy matrix that contains the feature vectors and a Numpy vector that contains the labels.
2. **Data Partition**: Partition the data into a training set, a validation set and a testing set. The training set takes around 80% of the total. The validation set takes about 10% . The testing set takes the rest. The three sets should NOT overlap.
3. **Train model parameter**: For a given group of hyper-parameters such as M, $\boldsymbol{\mu}_j$, Σ_j, λ, $\eta^{(\tau)}$, train the model parameter $\mathbf{w}$ on the training set.
4. **Tune hyper-parameters**: Validate the regression performance of your model on the validation set. Change your hyper-parameters and repeat step 3. Try to find what values those hyper-parameters should take so as to give better performance on the validation set.
5. **Test your machine learning scheme on the testing set**: After finishing all the above steps, fix your hyper-parameters and model parameter and test your model's performance on the testing set. This shows the ultimate effectiveness of your model's generalization power gained by learning.

4.2 Tasks On Synthetic Data

1. **Load feature values and labels from the CSV file**: Download the CSV file from UBLearns and load it into Python. One CSV file contains a matrix that contains the feature vectors. Another CSV file contains a vector that contains the labels. This is the public data generated by classified stochastic mechanism.
2. **Data Partition**: Like before, partition the data into a training set, a validation set and a testing set.
3. **Train model parameter**: For a given group of hyper-parameters such as M, $\boldsymbol{\mu}_j$, Σ_j, λ, $\eta^{(\tau)}$, train the model parameter $\mathbf{w}$ on the training set.
4. **Tune hyper-parameters**: Validate the regression performance of your model on the validation set. Change your hyper-parameters and repeat step 3. Try to find what values those hyper-parameters should take so as to give better performance on the validation set.

5. **Submit your model for testing**: After finishing all the above steps, fix your hyper-parameters and model parameter and test your model's performance on the testing set. This shows the ultimate effectiveness of your model's generalization power gained by learning.

5 Strategies for Tuning Hyper-Parameters

5.1 Choosing Number of Basis Functions M and Regularization Term λ

For tuning M and λ, you can simply use grid search. Starting from small values of M and λ, gradually try bigger values, until the performance does not improve. Please refer to

https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search.

5.2 Choosing Basis Functions

1. **Centers for Gaussian radial basis functions $\boldsymbol{\mu}_j$**: A simple way is to randomly pick up M data points as the centers.

2. **Spread for Gaussian radial basis functions Σ_j**: A first try would be to use uniform spread for all basis functions $\Sigma_j = \Sigma$. Also constrain Σ to be a diagonal matrix

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2 \end{pmatrix} \tag{15}$$

   Choose σ_i^2 to be proportional to the ith dimension variance of the training data. For example, let $\sigma_i^2 = \frac{1}{10}\text{var}_i(\mathbf{x})$.

3. **k-means clustering**: A more advanced method for choosing basis functions is to first use k-means clustering to partition the observations into M clusters. Then fit each cluster with a Gaussian radial basis function. After that use these basis functions for linear regression.

5.3 Choosing Learning Rate $\eta^{(\tau)}$

The simplest way would be to use fixed learning rate η. But this would lead to very poor performance. Choosing too big a learning rate could lead to divergence and choosing too small a learnng rate could lead to intolerably slow convergence. A more advanced method is to use Learning Rate Adaption based on changing of performance. Please refer to

https://en.wikipedia.org/wiki/Gradient_descent.

6 Deliverables

There are two parts in your submission:

1. Report

   The report describes your implementations and results using graphs, tables, etc. Write a concise project report, which includes a description of how you implement the models and tune the parameters. Your report should be edited in PDF format. Additional grading considerations will include the performance of the training, creativity in paramter tuning and the clarity and flow of your report. Highlight the innovative parts and do not include what is already in the project description. You should also include the printed out results from your code in your report.

Submission:

Submit the PDF on a CSE student server with the following script:

`submit_cse474 proj2.pdf` for undergraduates

`submit_cse574 proj2.pdf` for graduates

In addition to the PDF version of the report, you also need to hand in the hard copy version on the first class after due date or else your project will not be graded.

2. Code

   The code for your implementations. Code in Python is the only accepted one for this project. You can submit multiple files, but the name of the entrance file should be main.py. All Python code files should be packed in a ZIP file named `proj2code.zip`. After extracting the ZIP file and executing command `python main.py` in the first level directory, it should be able to generate all the results and plots you used in your report and print them out in a clear manner.

   Submission:

   Submit the Python code on a CSE student server with the following script:

   `submit_cse474 proj2code.zip` for undergraduates

   `submit_cse574 proj2code.zip` for graduates

7 Due Date and Time

The due date is **11:59PM, Oct 23**. Hard copy of your project report must be handed in before the end of the first class after the due date. After finishing the project, you may be asked to demonstrate it to the TAs if your results and reasoning in your report are not clear enough.