CSE474/574: Introduction to Machine Learning(Fall 2015)

Instructor: Sargur N. Srihari
Teaching Assistants: Zhen Xu, Jun Chu, Yifang Liu

Project 2: Learning to Rank using Linear Regression

Due Date: Wednesday, Oct. 28

1 Overview

This project is to implement linear regression to solve regression problem and evaluate its performance. The objective is to learn how to map an input vector $\mathbf{x}$ into a scalar target t. In this project, you are required to finish four tasks:

1. Train a linear regression model on real dataset using batch method.
2. Train a linear regression model on real dataset using stochastic gradient descent.
3. Train a linear regression model on synthetic dataset using batch method and evaluate its performance.
4. Train a linear regression model on synthetic dataset using stochastic gradient descent and evaluate its performance.

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 Maximum Likelihood Solution

Maximum likelihood (ML) solution,

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

where β is the precision. Maximizing the likelihood is equivalent to minimizing the sum-of-squares error

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

1.3 Regularization to contain over-fitting

Append a quadratic regularization term in the error function

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

to avoid over-fitting, where

$$E_W(\mathbf{w})=\frac{1}{2}\sum_{j=0}^{M-1}|w_j|^2 \tag{6}$$

in which the coefficient λ governs the relative importance of the regularization term.

1.4 Solution

After choosing the basis functions, we solve linear regression and evaluate the optimal $\mathbf{w}_{ML}$ by setting the likelihood's gradient w.r.t $\mathbf{w}$ equal to zero. There are two ways to solve linear regression, closed-form solution and stochastic gradient descent.

1.4.1 Closed-form Maximum Likelihood Solution for w

The closed-form maximum likelihood solution to linear regression, assuming Gaussian noise, 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 outputs in the training data and $\boldsymbol{\Phi}$ is the design matrix:

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

Solution to maximum likelihood solution with quadratic regularization has the form

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

1.4.2 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 seperate validation dataset 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 validation dataset.

1.6 Datasets

The training dataset consists of N inputs $\mathbf{x}_1, ..., \mathbf{x}_N$ together with the corresponding target values $t_1, ..., t_N$. You are required to implement linear regression on two datasets.

1. A real world dataset from a problem in information retrieval known as Learning to Rank (LeToR, 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 Real-World Dataset

For real world dataset, we use the Microsoft LETOR 4.0 Dataset. LETOR is a package of benchmark data sets for research on Learning To Rank released by Microsoft Research Asia. The latest version, 4.0, 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 value 0, 1 or 2. This is the objective output y we expect our linear regression to give.
2. The second column `qid` is the query id. It is not used in the project.
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`, `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. We will give you a training dataset generated by the formula to train. However the true generating mechanism is classified. After you submit your model, we will test the performance of it on another group of generated data unseen to you.

4 Plan of Work

4.1 Tasks On Real-World Data

1. **Extract feature values and labels from the data**: Process the original text data file into a MATLAB matrix that contains the feature vectors and a MATLAB vector that contains the labels.
2. **Data Partition**: Partition the data into a training set, a validation set and 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 MAT-file**: Download the MAT-file from UBLearns and load it into MATLAB. It contains a matrix that contains the feature vectors and a vector that contains the labels. This is the public data generated by classified stochastic mechanism.

2. **Data Partition**: Partition the data into a training set and a validation set. The proportion is up to you and you DO NOT need a testing set, because our grading system will judge your performance on the classified testing data.

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, submit your hyper-parameters and model parameter. Our grading system will test your model's performance on the classified testing set. For this part, you will be graded only **ONCE**.

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 $\sigma_i^2 = \frac{1}{10}\mathrm{var}_i(\mathbf{x})$.

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

1. You will submit three files in your submission using CSE submit script:

   (a) MAT-file for regression implementation `proj2.mat`. It should include the following variables. Those with subscript "1" such as `M1` are used for real-world data tasks and those with subscript "2" such as `M2` are used for synthetic data tasks.

   `M1`, `M2`: A scaler. Number of basis functions M.

   `mu1`, `mu2`: A $D \times M$ array. Each column represents the $\boldsymbol{\mu}_j$ in the basis functions. D is the dimension of inputs $\mathbf{x}$.

   `Sigma1`, `Sigma2`: A $D \times D \times M$ array. They represent the Σ_j in the basis functions.

   `lambda1`, `lambda2`: A scalar. The regularization coefficient λ.

   `trainInd1`, `trainInd2`: A $N_1 \times 1$ array. It contains indices for the training data. N_1 is the number of training data.

   `validInd1`, `validInd2`: A $N_2 \times 1$ array. It contains indices for the validation data. N_2 is the number of validation data.

   `w1`, `w2`: A $M \times 1$ array. Weights $\mathbf{w}$ learned using closed form solution.

   `w01`, `w02`: A $M \times 1$ array. Initial weights $\mathbf{w}^{(0)}$ for stochastic gradient descent.

   `dw1`, `dw2`: A $M \times E$ array. Contains all update of weights $\Delta\mathbf{w}^{(\tau)}$. E is the number of iterations in your stochastic gradient descent.

   `eta1`, `eta2`: A $1 \times E$ array. Learning rate $\eta^{(\tau)}$ for task 2 and task 4.

   `trainPer1`, `trainPer2`: A scalar. Root mean square error on the training set.

   This file could be submitted unlimited times and we will grade the correctness of your model online like in project 1 before the due date. After the due date, we will evaluate your model on the classfied synthetic data. This evaluation is conducted only **ONCE**.

   (b) PDF file for project report `proj2.pdf`

   Write a complete project report. The following are some examples parts:

   i. Explain your model

   ii. Partition of the training, validation and test sets

iii. Explain how you choose the parameters and hyper-parameters and their relationships

iv. Discussion of model complexity and performance

v. Compare the performances of different solutions

Use graphs, tables and so on to elaborate your report.

(c) Your script used to generate the final version MAT-file `proj2.m`. This script should be able to generate `proj2.mat` given the real-world data file and synthetic data file.

2. Submit a hard copy of your project report before the end of the first class after the due date.

7 Due Date and Time

The due date is **11:59PM, Oct 28**. Hard copy of your project report must be handed in before the end of the first class after the due date.

8 Grading Policy

8.1 Score Distribution

The project scores are divided into five parts:

1. Train a Linear Regression model on real dataset using batch method. (15 points)
2. Train a Linear Regression model on real dataset using stochastic gradient descent. (15 points)
3. Train a Linear Regression model on synthetic dataset using batch method and evaluate its performance. (30 points)
4. Train a Linear Regression model on synthetic dataset using stochastic gradient descent and evaluate its performance. (30 points)
5. Project Report (10 points)

8.2 Anti-Cheating Measures

The hyper-parameters M, $\boldsymbol{\mu}$, Σ, λ, η, $\mathbf{w}_0$ are tuned individually and are thus prone to be different. Thus your learned model parameter $\mathbf{w}$ should also be distinct. Similar setting of hyper-parameters is not acceptable and is considered cheating! For the stochastic gradient descent, we will track the entire updating process $\Delta\mathbf{w}^{(\tau)}$, $\tau = 0, 1, 2, ...$ The learned final $\mathbf{w}$ is computed by summing all them up together with initial value $\mathbf{w}^{(0)}$. Thus it's impossible to simply copy the answer from the closed-form solution!