CSE474/574: Introduction to Machine Learning(Fall 2014)

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

**********DRAFT of September 26, 2014**********

Project 1: Linear Regression with Basis Functions

Due Date: Tuesday, Oct. 14^{th}

1 Overview

This project is to implement and evaluate several supervised machine learning approaches to the task of linear regression. The objective is to learn how to map an input vector $\mathbf{x}$ into a target value t using the model

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

where $\mathbf{w} = (w_0, w_1, .., w_{M-1})$ is a weight vector to be learnt from training samples and $\boldsymbol{\phi} = (\phi_0, .., \phi_{M-1})^T$ is a vector of M basis functions. Each basis function $\phi_j(\mathbf{x}), \quad j = 0, ..M-1$ converts input vector $\mathbf{x}$ into a scalar value. An example is the Gaussian basis function

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

where $\boldsymbol{\mu}_j$ is a vector in feature space and s is an isotropic spatial scale.

The training dataset consists of N exemplar vectors $\mathbf{X} = (\mathbf{x}_1, .., \mathbf{x}_N)$ together with the corresponding target values $\mathbf{t} = (t_1, ..t_N)$.

1.1 Methods

Three solutions are to be implemented, the last of which is optional:

1. Maximum likelihood (ML) solution, which is equivalent to minimizing the sum-of-squares error

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2 \tag{3}$$

   using the closed-form solution given in Section 7: Appendix of this project description.

   Use an additive regularization term with the error function

$$E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}) \tag{4}$$

to avoid over-fitting, where

$$E_W(\mathbf{w}) = \frac{1}{2}\sum_{j=0}^{M-1} |w_j|^q \tag{5}$$

where coefficient λ governs the relative importance of the regularization term and we choose $q = 2$ to reflect quadratic regularization.

2. Maximum likelihood with Gradient Descent

3. Bayesian linear regression (Bonus part)

1.2 Application

The data set for this task is from a problem in information retrieval known as Learning to Rank (LeToR), You are given datasets downloaded from Microsoft LETOR 4.0. It contains queries and urls represented by IDs, and feature vectors extracted from query-url pairs $\mathbf{x}$ along with relevance judgment labels t which are 0, 1, 2 with higher values indicating higher relevance[1].

1.3 Evaluation

Evaluate your solution using one or both of the following two criteria:

1. Root mean square (RMS) error, defined as

$$E_{RMS} = \sqrt{2E(\mathbf{w}*)/N_T} \tag{6}$$

where $\mathbf{w}*$ is the solution and N_T is the size of the test set.

2. Discounted Cumulative Gain (DCG), which is a measure used to evaluate search engines:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{7}$$

where rel is the relevance label and p is the rank position, you can set $p = 20$.

2 Details of Microsoft LETOR 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, QueryLevelNorm. In this project, only QueryLevel- Norm version `"Querylevelnorm.txt"` will be used. The entire data contains $N = 69623$ query-document pairs (rows) and $d = 46$ dimensions of features (columns).

Here are two sample rows from the MQ2008 dataset.

[1]Note that the feature descriptions are not disclosed, only the feature values are given.

```
   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
```

Each row is as follows.

1. The first column is relevance label of this pair with one of the values 0, 1 or 2. The larger the relevance label, the more relevant the query-document pair.
2. The second column is query id,
3. The following 46 columns are features. A query-document pair is represented by a 46-dimensional feature vector of real numbers. All the features have been normalized to be in the [0,1] range. When a feature is undefined for a set, its value is 0.
4. The end of the row is a comment about the pair, including id of the document.

3 Plan of Work

Partition the data into training and test tests. For maximum likelihood solutions, the training set can be further partitioned to get a validation set. Training set takes around 80% of the total, validation set takes about 10% and the rest for testing. The dataset is neatly formatted in full feature vectors with undefined feature values to be 0.

1. **Extract feature values from data:** Process the original text data into a matrix containing relevance labels (the first column) and feature vectors. This input matrix will be feed into your regression model.
2. **Name the training programs:** `"train_cfs.m"`, `"train_gd.m"`, **and** `"train_bs.m"` which correspond to closed-form, gradient descent, and Bayesian respectively. The input of training scripts consists of an $X \in \Re^{N \times D}$ matrix, where N is the number of the training samples (rows/urls) and D is the length/dimension of each training sample vector, and a $T \in \Re^N$ matrix (vector) is the target value of each training sample.
3. **Validation:** Once the function is learnt, for each new feature vector, the target value can be predicted. Your validation program should take a $Y \in \Re^{N' \times D}$ matrix as an input and output a $T \in \Re^{N'}$ vector of relevance values.

   Separate the validation data into ground truth (the column containing labels to be predicted) and feature vectors with size of $N' \times D$. Use the evaluation metrics i to evaluate and adjust the parameters of your model (e.g. parameters in your basis function, regularization coefficient, etc) to avoid overfitting. You should elaborate your validating process in your report.
4. **Testing:** Once you have tuned your model for the best accuracy it can achieve, you can test your model. As you did in the validation phase, a test set with size of $N'' \times D$ will be fed into your model. Compare the predicted labels with the real ones and report the final results in your testing programs `"test_cfs.m"`, `"test_gd.m"`, and `"test_bs.m"` ("cfs", "gd", and "bs" follow the same meanings as in the training phase).

5. **Output:** You need to output the results at the end of your program using sprintf. The sample code is provided as follows:

   fprintf('My ubit name is %s\n',yourubitname);
   fprintf('My student number is %d \n',yournumber);
   fprintf('the model complexity M_cfs is %d\n', M_cfs);
   fprintf('the model complexity M_gd is %d\n', M_gd);
   fprintf('the regularization parameters lambda_cfs is %4.2f\n', lambda_cfs);
   fprintf('the regularization parameters lambda_gd is %4.2f\n', lambda_gd);
   fprintf('the root mean square error for the closed form solution is %4.2f\n', rms_cfs);
   fprintf('the root mean square error for the gradient descent method is %4.2f\n', rms_gd);
   fprintf('the root mean square error for the Bayesian solution is %4.2f\n', rms_bs); (if you do it)

   Please exactly follow the same output format (including the same words and lower/uppercase) as the above. You code should also be self-contained, which means we can directly run it without any further work. You should include your input data file named project1_data.mat in your submission.

4 Deliverables

There are two parts in your submission:

1. **Report** describing your implementations, e.g., what were your training, validation and test sets, a discussion of model complexity and performance using graphs, tables, etc. Write a complete project report, which mainly has four parts:

   (a) Explain your model
   (b) Explain how you choose the parameters and hyper-parameters and their relationships
   (c) Evaluate the model
   (d) Compare the performances of different solutions

   Your report should be edited in pdf or word format, graphs should be carefully plotted to show your results. Additional grading considerations will include creativity in choice of models, accuracy and completeness in interpreting your statistics, and the clarity and flow of your report.

2. **Code** for your implementations. MATLAB is recommended for the project.

   Code Submission consists of:

   (a) Six Matlab functions `"train_cfs.m"`, `"train_gd.m"`, `"test_cfs.m"`, and `"test_gd.m"`, that train the linear regression model and predict the labels should be submitted.
   (b) One wrap-up script `"project1.m"` that can call all necessary functions, finish one whole training and testing procedure and output the required information. (Note that "yourubitname" should be substituted with your real ubitname).
   (c) Your input data file named `"project1_data.mat"`, learned weight matrices `"W_cfs.mat"` and `"W_gd.mat"`, hyper-parameters mu (mean) `"mu_cfs.mat"` and `"mu_gd.mat"`, hyper-parameters s (variance) `"s_cfs.mat"` and `"s_gd.mat"`.

(d) A Readme file includes short usage information about arguments, return values of your functions in order for us to verify that your code produces the results you submitted.

(e) If you choose to do the bonus part, you need two extra files, `"train_bs.m"` and `"test_bs.m"`.

You are required to submit only one **zip** file `"yourubitname_proj1.zip"` that contains all the above files.

All outputs and files should have exactly the same formats and names as specified. Each violation can result in 5 points penalty,

The report and code should be submitted through cse_submit command.

All the files mentioned above should be submitted via the CSE submit script, e.g.)
Submit for undergraduates: `submit_cse474 yourubitname_proj1.zip`
Submit for graduates: `submit_cse574 yourubitname_proj1.zip`

5 Due Date and Time

The due date is **Oct. 14th, 11:59PM**. After you have finished the project, you need to make a demonstration of your project to the TAs.

6 Grading Policy

The project grading has three parts as follows:

1. Maximum Likelihood Closed-form Solution (30 points)
2. Stochastic Gradient Descent (30 points)
3. Project Report (40 points)

Bonus Part: Bayesian Solution, if you implement all code by yourself, you can get extra 20 points. If you use a package, you can get extra 5 points.

7 Appendix

7.1 Closed-form Maximum Likelihood Solution

The closed-form maximum likelihood solution to linear regression (assuming Gaussian noise) has the form

$$\mathbf{w}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t} \tag{8}$$

where $\mathbf{t} = \{t_1, .., t_N\}$ is the vector of outputs in the training data and the input vectors $X = \{\mathbf{x}_1, ..\mathbf{x}_N\}$ are organized into the design matrix Φ as follows:

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

The quadratic regularizer extension has the form

$$\mathbf{w}_{ML} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \tag{9}$$

7.2 Stochastic Gradient Descent

The stochastic gradient descent algorithm has the form

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{\tau} + \eta(t_n - \mathbf{w}^{(\tau)T}\phi_n)\phi_n \tag{10}$$

7.3 Bayesian Linear Regression (Optional)

The predictive distribution has the form

$$p(t|\mathbf{x}, \mathbf{t}, \alpha.\beta) = N(t|\mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x})) \tag{11}$$

where $\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^T \mathbf{t}$, $S_N^{-1} = \alpha \mathbf{I} + \beta \Phi^T \Phi$, α is the spread in the prior and β is the noise precision.

Alternatively, you can try a prepackaged Gaussian Process
(The link is `http://www.gaussianprocess.org/gpml/code/matlab/doc/`).