Introduction to Machine Learning

************* September 6, 2018****************

Project 2: Learning to Rank using Linear Regression

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 two 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).

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

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

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

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 Dataset

You are required to implement linear regression on a learning to rank (LeToR) dataset. In the LeToR dataset 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.

The Microsoft LETOR 4.0 Dataset is a benchmark data set for research released by Microsoft Research Asia. It 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.

2 Plan of Work

2.1 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.

3 Strategies for Tuning Hyper-Parameters

3.1 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.

3.2 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

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2 \end{pmatrix} \tag{14}$$

   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.

3.3 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.