574 Project 2 Report

Jingyi Zhao (50245749)
Zheng Kai (50247576)
Can Yao (50243637)

Abstract

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 obtain the linear model that fits to observed data(on training data set). Then we use the validation set to optimize the model before we test the model performance on test set. In these process, we use two methods to compute weight, one is Closed-form Solution which employes K-means cluster with given model complexity, the other is (SGD)Stochastic Gradient Descent . We curb overfitting by using use early stopping during training, tuning the 2 parameter (learning rate/ complexity) before testing its performance. we report the RMSE for modeling task 1 being 0.566, task 2 being 0.570, task 3 being 0.698, and task 4 being 0.817. For small feature space, it's okay to use both method; for a large feature space, we recommending using SGD method.

Introduction

In this project, we train linear regression models on two different datasets. One is Letor 4.0 Dataset from real word, the other is synthetic data that generated using known functions added with noise. For each dataset, we use two different methods to get the linear regression models, one is closed-form solution and the other is stochastic gradient descent method.
Some of the formulas used in the project are listed below
linear regression function form

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

In this project, we use the Gaussian radial basis functions

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

Minimizing the sum-of-squares error

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

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

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

Closed-form Solution for **w:**

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

the design matrix Φ:

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

Stochastic Gradient Descent Solution for **w**

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

$$\nabla E = \nabla E_D + \lambda \nabla E_W$$

Evaluate solutions on a test set using Root Mean Square (RMS) error:

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

Experiment Design

- General Design

Task 1 and 2 are based on LeToR data. The LeToR training data consists of pairs of input values x and target values t. The input values are real-valued vectors ,the target values are scalars that take one of three values 0, 1, 2. Data is partitioned into training set, validation set, and test set. The training set takes around 80% of the total. The validation set takes about 10% .Validation set is used to pick the best training numbers in order to avoid overfitting; Test set is to evaluate the overall performance of the model. Task 3 and Task 4 are based on

Synthetic Data. The data is manually synthesized based on original LeToR data, with fewer features and sample numbers. we perform linear regression using closed form approach from matrix operation in Task 1 and 3, and linear regression using gradient descent for task 2 and 4. We would like to compare the performance between 2 different training methods. We compare the result of Task 1, 3 and Task 2,4 to verify if our training models has certain extensibility..

- Validation Process:

Validation data is used to minimize overfitting. This set does not adjust the weight. If the accuracy based on the training data is increased after the adjustment of the training data, but the accuracy of the verification data is not increased or decreased, it means overfitting occur and early stopping should be introduced. we are able to obtain the best training model for one pair of learning rate & # of basis. Furthermore, best models from different combinations of learning rate & # of basis are compared based on their validation error. Eventually we are able to determine the best pair of parameters for each task after this validation process.

- Choice of parameters:

we did not arbitrarily choose any parameters. Instead, learning rate η and number of Basis are all chosen during parameter tuning. For η, we pick a range of 1 ro 0.001. Learning rate beyond 1 is typically too large in machine learning. For number of Basis, we pick a range of 1 to maximum number of features. The best pair of parameter is chosen after each model's validation performance is compared.

Data analysis

We observe overfitting during training, Figure 1 is an example of overfitting when training Task 4.

- Early stopping:

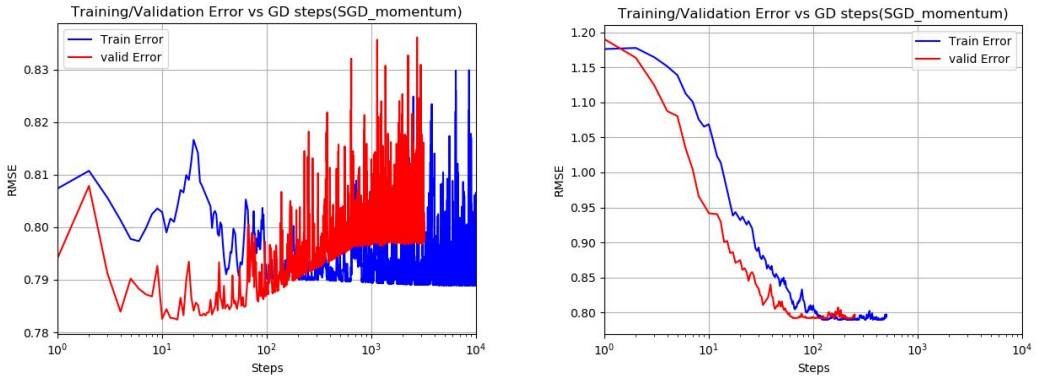

Figure 1: a) overfitting occurs during training for task 4. b) use early stop to prune the training model.

- Training Phase in gradient descent:

when training the data using gradient descent, we use early stopping to avoid overfitting. During training phase, after training goes beyond 500 steps, we check the validation error every 2 steps. We consider the model not stable when training steps are too small and decided not to early stop at below 500 training steps. When validation error starts to bounces up while training error continue to decrease, early stopping could occur and capture the model that give the best performance on validation set. Whether early stopping happens or not depends on the learning rate and the initial randomization of weight.

- Parameter tuning phase:

We plot the validation error at each learning rate vs its # of basis. we pick the # of basis that give smallest validation error, to avoid over complexing the model and overfitting. Figure 2-5 are graphs showing the Tuning of parameter (learning rate / # of basis).

- **Tune for Task 1**

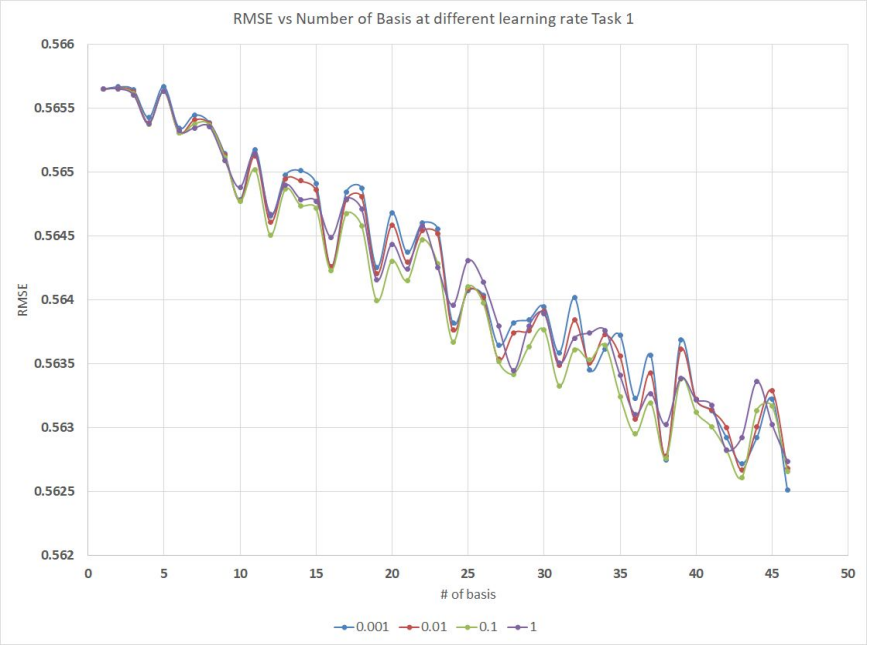

Figure 2: parameter performance for Task 1

we provide here a plot of parameter tuning using closed form method on synthetic data. we see here the best parameter is $\eta = 0.001$ and # of basis = 46.

- **Tune for Task 2**

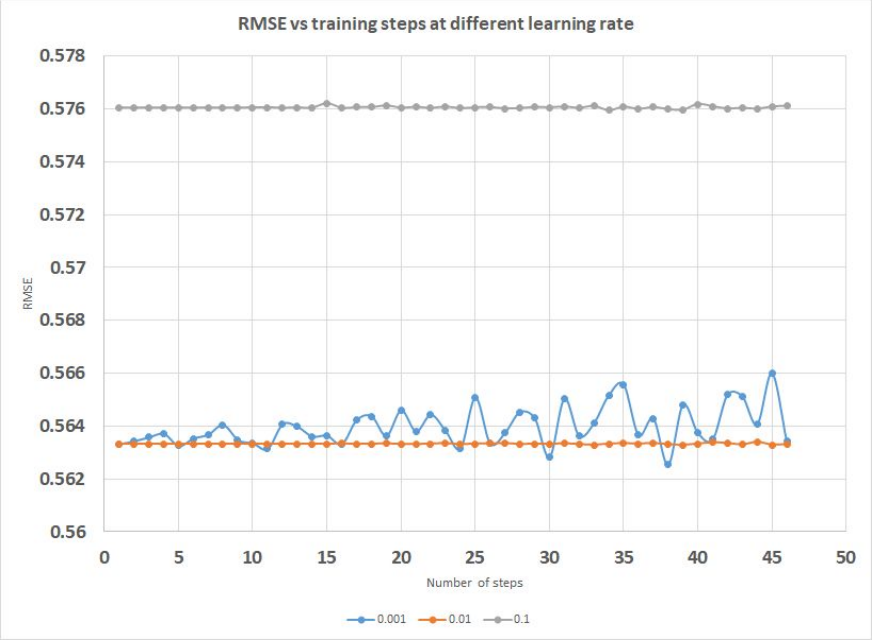

Figure 3: parameter performance for Task 2

When we use learning rate = 1 to tune task 2, the error blows up to 20000 so we discard it. According to the graph, the best parameter η = 0.001 and # of basis = 36.

- **Tune for Task 3**

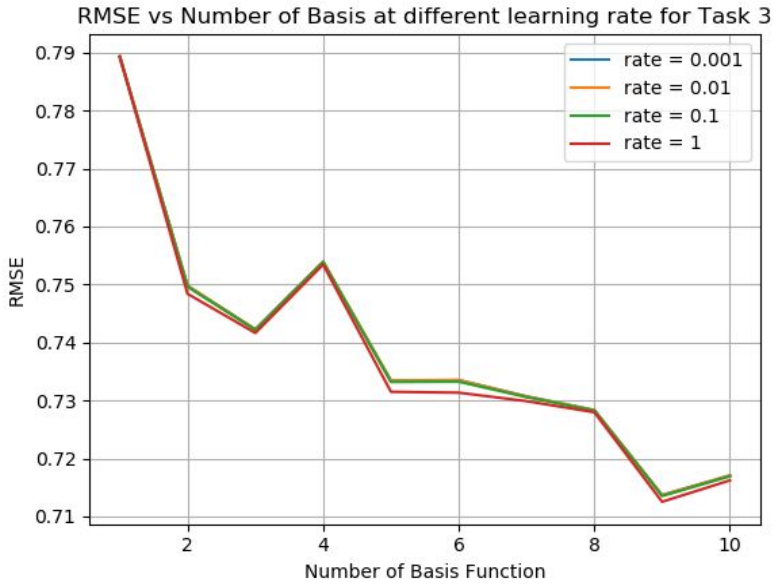

Figure 4: parameter performance for Task 3

we provide here a plot of parameter tuning using closed form method on synthetic data. we see here the best parameter is η = 1 and # of basis = 9.

- **Tune for Task 4**

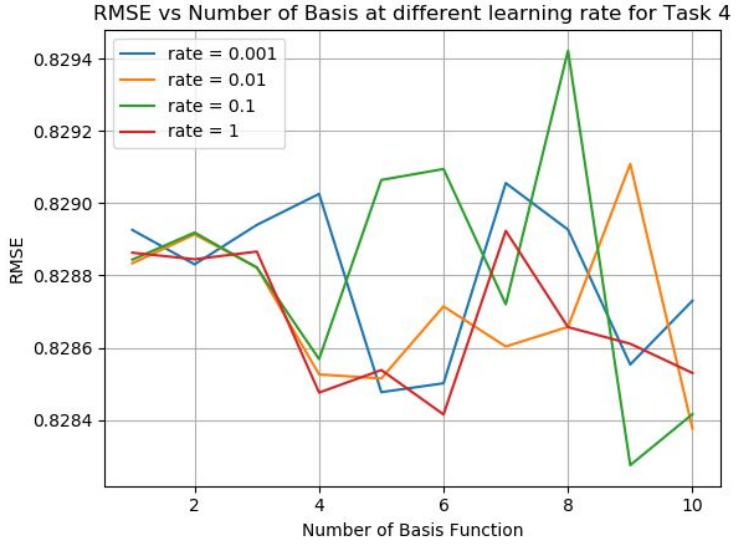

Figure 5： parameter performance for Task 4

we provide here a plot of parameter tuning using gradient descent method on synthetic data. we see here the best parameter is $\eta = 0.1$ and # of basis = 9. However, amongst all parameter combination, difference in error never goes beyond 0.01%. The performance for each optimized model in each test set is shown in Figure 6

Task #	Learning rate	# of basis	Train Error	Validation Error	Test Error
1	0.01	46	0.567	0.574	0.566
2	0.01	36	0.572	0.577	0.570
3	1	9	0.699	0.694	0.698
4	0.1	9	0.826	0.812	0.817

Figure 6： performance on test set

- Analysis on Closed form linear fitting:

When the data is trained in closed form, learning rate does not have significant impact on the model performance, whereas the more complex the model, the higher the performance of the model would be. Because the model complexity never goes beyond actual input feature numbers, we did not actually see a large extend of overfitting incurred by model complexity, despite at Tuning for Task 3, # of basis = 10 seems to be introducing a little overfitting.

- Analysis on SGD

When the data is trained in stochastic gradient descent, the model performance is subject highly to learning parameters. A higher learning parameter would have a higher probability to induce a high error in model. Moreover, Figure 3 shows that the SGD's result is more likely to be locked inside the local minima when learning rate is high. Thus, to further optimize our parameter tuning method, we should introduce much smaller learning rate like 1e-5, 1e-6. Otherwise, gradient descent performance on test set is within 0.7 % deviation for LeTor Set, and 17% deviation for Synthetic data set. We do not treat it as a significant difference.

- TestSet performance, running time comparison

Synthetic data test sets have a > 50% deviation in both trained methods compared with their LeToR counterparts. It is safe to conclude that the current model trained from LeToR data

sets from closed form approach and SGD approach need to be examined before transferred beyond LeToR data set.

For the closed form linear regression, given 4 being the number of learning rate, N being the total training sample, M being the total feature space.It takes 4 x Nx M to calculate the design matrix for a specific M. It takes 4 x (N x M) x M operations to determine best learning rate and model complexity. When M is small, it takes O(N) time .However, if the scale is large, it takes O(N^3). We already experienced a significant amount of decrease in training time during training task 3 compared with task 1. (complexity 9 vs complexity 46). Gradient descent on the other hand, takes 4α x M times to select the best parameter in a feature space of M, where α is defined as the steps beyond which model converges. SGD at most takes O(n^2). In this current project, we employed another gradient descent strategy using momentum, which guarantees the steps to converge at around 20 steps. Below is the comparison between SGD with momentum swing, and SGD.

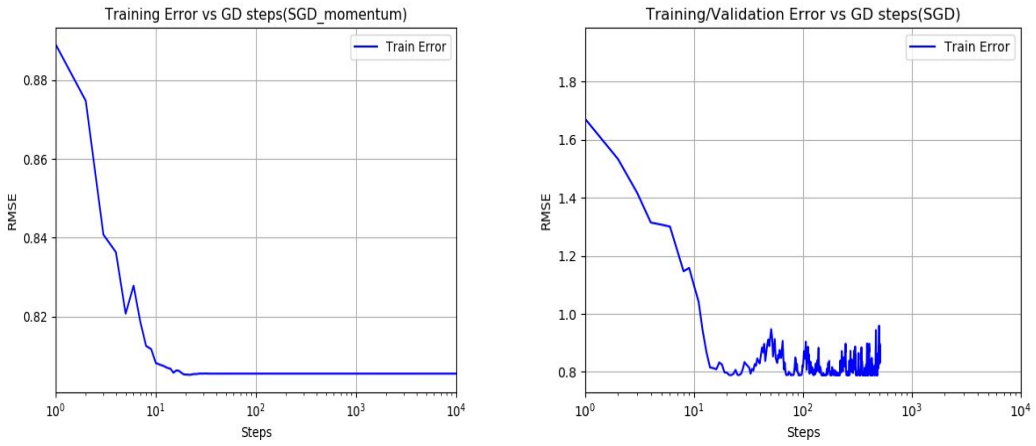

Figure 7：a) SGD optimized with momentum swing, b) SGD

Figure 7 a) converges at step 20 and Figure b) stops at step 600 due to early stopping which is a 30 times increase. Thus, we suggest use SGD with momentum to train data set with a large feature space, and the time complexity could be somewhere between O(n) and O(n^2).

Conclusion

Thus, we have used 2 different linear regression method to train the 2 different data pools. we did not see significant change in the performance of 2 training methods. However, for large

dataset with large feature space, we suggest training with SGD with momentum instead of computing closed form matrix during training.