COL764
Assignment 3

Learning to Rank

Debraj Dey
2021ME10973

Prof. Srikanta Bedathur

Indian Institute of Technology Delhi

1 Modeling of the Problem

The goal is to rank documents for a given query by predicting a relevance score for each query-document pair. This relevance score determines the ranking order of documents for each query, with higher scores indicating higher relevance.

This problem is modeled as a pointwise learning-to-rank task, where the model is trained to predict the relevance score of a query-document pair individually. Pointwise approaches treat ranking as a regression task where each query-document pair has a label (e.g., 0 or 1) representing its relevance. The model learns to predict a continuous score (based on features) that corresponds to the document's relevance with respect to the query.

The provided data consists of query-document pairs with pre-extracted features. Each pair has an associated relevance label, 0 or 1, that indicates how relevant the document is for the given query.

Models Used:

Experimentation was done with three types of machine learning models:

1. Support Vector Regressor (SVR)
2. GradientBoostedRegressor
3. MLPRegressor (Multilayer Perceptron Regressor)

Support Vector Regressor (SVR)

SVR is based on the Support Vector Machine (SVM) framework, and it is adapted for regression tasks. It works by fitting the best possible hyperplane in a high-dimensional feature space to predict a continuous value.The SVR model predicts a relevance score for each query-document pair based on the input features. The loss function used in SVR aims to minimize the errors only when they exceed a certain threshold (controlled by the epsilon parameter).

Loss Function: SVR uses the ε-insensitive loss function, which ignores errors smaller than epsilon. It tries to minimize errors above epsilon while maintaining a margin between predicted and actual relevance scores.

GradientBoostedRegressor (GBR)

Gradient Boosting is an ensemble learning method that combines the predictive power of many decision trees. Each tree in the ensemble is trained to predict relevance scores. The model combines these predictions to produce a final relevance score for each query-document pair.

Loss Function: We use the Mean Squared Error (MSE) as the loss function, which minimizes the square of the difference between predicted and actual relevance scores.

Multilayer Perceptron Regressor (MLP)

MLPRegressor is a type of neural network, where data is passed through multiple layers of neurons (nodes), and the network learns to map input features to output relevance scores. The network predicts relevance scores by passing the input features through multiple hidden layers. Each layer applies an activation function to introduce non-linearity into the model.

Loss Function: The network is trained using the Mean Squared Error (MSE) loss, which minimizes the squared difference between the predicted and actual relevance labels.

2 Hyperparameter Selection

Support Vector Regressor (SVR)

For SVR, the best parameters were selected out of the following choices:

1. **Kernel** : Chosen from [‘linear’, ‘poly’, ‘rbf’].
2. **Epsilon** : Specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. Chosen from [0.01, 0.05, 0.1, 0.3, 0.5].
3. **C** : Regularization parameter. Chosen from [0.01, 0.1, 1, 10, 50, 100].
4. **Gamma** : Kernel coefficient. Chosen from [’scale’, ’auto’, 0.01, 0.1, 1].
5. **Tol** : Tolerance for stopping criterion. Chosen as 10^{-4}.
6. **Cache Size** : Specify the size of the kernel cache. Chosen as 500 MB.

After training and testing for each permutation of hyperparameter choices, the best choices were:

Parameter	TD2003	TD2004
Kernel	‘rbf’	‘rbf’
Epsilon	0.5	0.5
C	50	0.1
Gamma	’scale’	’scale’
Tol	0.0001	0.0001
Cache	500	500

Table 1: Parameter values for Support Vector Regressor

GradientBoostedRegressor (GBR)

For GBR, the best parameters were selected out of the following choices:

1. **n_estimators** : First [100, 300 and 500] were chosen as choices. Best nDCG results were obtained at 100. Then it was observed that upon decreasing the number

of estimators (i.e. trees), the nDCG score increases and then decreases, peaking at around 35 among the choices [20,25,30,35,40,45,50,55,60]. But the score again increases when further decreasing the number of estimators to very low i.e. [1,2,3].

2. **Learning Rate** : The learning rate was chosen from [0.001,0.005,0.01,0.05,0.1].

3. **Max depth** : Maximum depth of the individual regression estimators. Chosen from [2,7].

4. **Min Sample Split** : The minimum number of samples required to split an internal node. Chosen from [2,5,10].

5. **Subsample** : The fraction of samples to be used for fitting the individual base learners. If smaller than 1.0 this results in stochastic Gradient Boosting. Chosen from (0.0,1.0].

6. **Criterion** : Default value 'friedman_mse' used.

7. **Min Samples Leaf** : The minimum number of samples required to be at a leaf node. Chosen from [1,2,5].

8. **Max Features** : The number of features to consider when looking for the best split. Chosen from (0.0,1.0] and and the features considered at each split will be max(1, int(max_features * n_features)).

9. **Loss** : 'squared_error' was chosen.

After training and testing for each permutation of hyperparameter choices, the best choices were:

Parameter	**TD2003**	**TD2004**
n_estimators	2	2
Learning Rate	0.005	0.01
Max Depth	2	2
Min Sample Split	10	10
Subsample	0.5	0.5
Criterion	'friedman_mse'	'friedman_mse'
Min Samples Leaf	2	5
Max Features	1.0	1.0
Loss	'squared_error'	'squared_error'

Table 2: Parameter values for Gradient Boosted Regressor

Multilayer Perceptron Regressor (MLP)

For MLP, the best parameters were selected out of the following choices:

1. **Hidden Layer Sizes** : Number of hidden layers and the number of neurons in each hidden layer. Chosen from [(20,), (50,), (100,), (150,), (200,), (22,22), (44,44), (88,88), (88,44), (100,50)].

2. **Activation** : Activation function for the hidden layer. Chosen from ['identity', 'logistic', 'tanh', 'relu'].
3. **Solver** : The solver for weight optimization. Chosen from ['lbfgs', 'sgd', 'adam']
4. **Alpha** : Strength of the L2 regularization term. Chosen from [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1].
5. **Batch Size** : Size of minibatches for stochastic optimizers. Chosen from [100, 200, 300, 400, 500].
6. **Learning Rate** : Learning rate schedule for weight updates. Chosen from ['constant', 'invscaling', 'adaptive']. 'invscaling' gradually decreases the learning rate learning_rate_ at each time step 't' using an inverse scaling exponent of 'power_t'.
7. **Learning Rate Init** : The initial learning rate used. It controls the step-size in updating the weights. Chosen from [0.1, 0.001, 0.0001].
8. **Max Iter** : Maximum number of iterations. For stochastic solvers ('sgd', 'adam'), this determines the number of epochs (how many times each data point will be used). Chosen from values of [150, 200, 300, 500, 750, 1000, 1500].
9. **Early Stopping** : Whether to use early stopping to terminate training when validation score is not improving. Set to 'True'.
10. **Beta 1** : Exponential decay rate for estimates of first moment vector in adam. Chosen from [0.5,0.9]
11. **Beta 2** : Exponential decay rate for estimates of second moment vector in adam. Default value 0.999 used.

After training and testing for each permutation of hyperparameter choices, the best choices were:

Parameter	**TD2003**	**TD2004**
Hidden Layer Sizes	[100,]	[100,50]
Activation	'relu'	'relu'
Solver	'adam'	'adam'
Alpha	0.0001	0.01
Batch Size	100	100
Learning Rate	'invscaling'	'invscaling'
Learning Rate Init	0.001	0.1
Max Iter	200	200
Early Stopping	True	True
Beta 1	0.9	0.7
Beta 2	0.999	0.999

Table 3: Parameter values for Multilayer Perceptron Regressor

3 Performance

Support Vector Regressor (SVR)

Fold		nDCG@5	nDCG@10	nDCG@100
Fold1	Train	0.06667	0.06696	0.10208
	Valid	0.13392	0.14404	0.18155
	Test	0.03008	0.02646	0.09821
Fold2	Train	0.06026	0.06138	0.11543
	Valid	0.03868	0.05803	0.12444
	Test	0.1	0.08657	0.14528
Fold3	Train	0.09891	0.10517	0.12719
	Valid	0.1	0.07820	0.13490
	Test	0.01696	0.01100	0.07032
Fold4	Train	0.07232	0.10469	0.16780
	Valid	0.04902	0.03711	0.07063
	Test	0.01312	0.05322	0.10664

Table 4: Performance of Support Vector Regressor for TD2003

Fold		nDCG@5	nDCG@10	nDCG@100
Fold1	Train	0.05699	0.05814	0.09255
	Valid	0.03904	0.05571	0.10228
	Test	0.09742	0.08335	0.10506
Fold2	Train	0.04444	0.05048	0.08954
	Valid	0.10874	0.09583	0.13620
	Test	0.05528	0.05853	0.12813
Fold3	Train	0.06702	0.06626	0.08571
	Valid	0.00875	0.03174	0.10316
	Test	0.07079	0.06143	0.12059
Fold4	Train	0.05198	0.05717	0.09864
	Valid	0.07079	0.06453	0.12497
	Test	0.0	0.01456	0.05910

Table 5: Performance of Support Vector Regressor for TD2004

GradientBoostedRegressor (GBR)

Fold		nDCG@5	nDCG@10	nDCG@100
Fold1	Train	0.04823	0.05198	0.12461
	Valid	0.26594	0.21942	0.27135
	Test	0.11695	0.13152	0.24876
Fold2	Train	0.13624	0.14473	0.18782
	Valid	0.15531	0.17264	0.26305
	Test	0.14233	0.14016	0.20814
Fold3	Train	0.22060	0.22286	0.27384
	Valid	0.23608	0.21483	0.26441
	Test	0.03836	0.05742	0.10840
Fold4	Train	0.18230	0.15073	0.20224
	Valid	0.03156	0.04455	0.12535
	Test	0.11920	0.16681	0.23599

Table 6: Performance of Gradient Boosted Regressor for TD2003

Fold		nDCG@5	nDCG@10	nDCG@100
Fold1	Train	0.03874	0.03992	0.12435
	Valid	0.07225	0.09619	0.18206
	Test	0.10711	0.09495	0.13219
Fold2	Train	0.02976	0.03013	0.13659
	Valid	0.10678	0.08920	0.12140
	Test	0.09511	0.08992	0.18815
Fold3	Train	0.01528	0.01392	0.13027
	Valid	0.09137	0.08635	0.18202
	Test	0.09802	0.08702	0.15544
Fold4	Train	0.04017	0.04293	0.14482
	Valid	0.05536	0.05041	0.13147
	Test	0.01760	0.07859	0.19722

Table 7: Performance of Gradient Boosted Regressor for TD2004

Multilayer Perceptron Regressor (MLP)

Fold		nDCG@5	nDCG@10	nDCG@100
Fold1	Train	0.05388	0.05112	0.10912
	Valid	0.25706	0.21200	0.24299
	Test	0.0	0.03744	0.11247
Fold2	Train	0.05359	0.05523	0.10376
	Valid	0.1	0.10694	0.19833
	Test	0.1	0.10043	0.17698
Fold3	Train	0.10084	0.09440	0.15470
	Valid	0.13868	0.14016	0.19363
	Test	0.09435	0.09900	0.11965
Fold4	Train	0.07203	0.07010	0.16899
	Valid	0.05531	0.04718	0.09658
	Test	0.13380	0.14945	0.17022

Table 8: Performance of Multilayer Perceptron Regressor for TD2003

Fold		nDCG@5	nDCG@10	nDCG@100
Fold1	Train	0.11790	0.12082	0.19593
	Valid	0.18411	0.17245	0.22734
	Test	0.11465	0.09302	0.12781
Fold2	Train	0.15422	0.15308	0.22368
	Valid	0.11465	0.09302	0.12781
	Test	0.07514	0.07566	0.14408
Fold3	Train	0.15137	0.14678	0.21375
	Valid	0.07514	0.07566	0.14408
	Test	0.12319	0.11194	0.15760
Fold4	Train	0.12464	0.11371	0.16686
	Valid	0.12319	0.11195	0.15767
	Test	0.15536	0.17485	0.28892

Table 9: Performance of Multilayer Perceptron Regressor for TD2004

4 Runtimes

Support Vector Regressor (SVR)

Dataset	Fold	Train Time (secs)	Test Time (secs)
TD2003	Fold 1	0.00790	5.80057
	Fold 2	0.00587	7.78576
	Fold 3	0.00684	8.67126
	Fold 4	0.00667	10.58580
TD2004	Fold 1	0.01015	12.97037
	Fold 2	0.01146	15.06704
	Fold 3	0.01512	13.85626
	Fold 4	0.01150	14.71416
Average		0.00944	11.18140

Table 10: Runtimes of Support Vector Regressor

GradientBoostedRegressor (GBR)

Dataset	Fold	Train Time (secs)	Test Time (secs)
TD2003	Fold 1	0.09183	0.00311
	Fold 2	0.08994	0.00336
	Fold 3	0.11393	0.00364
	Fold 4	0.10667	0.00397
TD2004	Fold 1	0.15257	0.00388
	Fold 2	0.15934	0.00445
	Fold 3	0.14413	0.00455
	Fold 4	0.15107	0.00420
Average		0.12618	0.00390

Table 11: Runtimes of GradientBoostedRegressor

Multilayer Perceptron Regressor (MLP)

Dataset	Fold	Train Time (secs)	Test Time (secs)
TD2003	Fold 1	3.13828	0.02964
	Fold 2	2.42518	0.02506
	Fold 3	2.11087	0.02434
	Fold 4	2.52045	0.02546
TD2004	Fold 1	1.47262	0.08606
	Fold 2	1.57373	0.08884
	Fold 3	2.03258	0.06387
	Fold 4	1.67124	0.06570
Average		2.11812	0.05112

Table 12: Runtimes of Multilayer Perceptron Regressor