LEARNING TO RANK USING LINEAR REGRESSION

ABSTRACT

- This project aims to solve the Learning to Rank (LETOR) problem using linear regression, which is a machine learning approach.
- Linear regression converts input vectors to target output which is a scalar.
- Two different methods, namely closed form solution and Stochastic Gradient Descent, are used to solve the problem

MODEL DESCRIPTION

- In the linear regression model, the input for the model is a vector and the output obtained is a real valued scalar.
- In this case, the output is in the form of relevance label numbers 0, 1, 2, 3 etc. Higher the label, greater the relevance and hence, better the match.
- The output:

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

- This is a function in terms of x (input variable) and w (weight vector). The weight vector is learnt through the training process and can be updated to make the model more accurate.
- Gaussian radial basis function:

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

- This is the main function which is used to convert the input vector into the scalar output.
- We divide the available data set into 3 components. They are
- Training: 80 percent of the data set used to train the model/help it learn the details of the features associated with each document query pair instance.
- Validation: When the model has trained, it is deployed on the validation set (20 percent) of the data set.
- This is important as it used to calculate how well the model is performing against the known values of outputs.
- The difference in the values obtained by the model and the expected value gives us the error.
- In the Stochastic Gradient Descent method, the weights are iteratively updated to close the gap between the observed and expected values.

DATASET USED

- For this model, the LETOR dataset is used.
- This data set has a set of document query pairs along with various features associated with each such pair.
- Each instance has a set of features (46) which are used by the model to obtain a similarity-based ranking.
- Some of the features used:

Column in Output	Description
1	TF(Term frequency) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	IDF(Inverse document frequency) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF*IDF of body

METHOD 1: CLOSED FORM SOLUTION

- In this method of linear regression implementation, the model works by finding the weights by obtaining Moore Penrose pseudo inverse of design matrix Phi.

- The final weight is calculated once when all the training data sample values have been incorporated into Phi matrix unlike SGD method where for each iteration of training, the weights are updated by incorporating the E-rms value.
- Important terms:
- Design matrix:

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

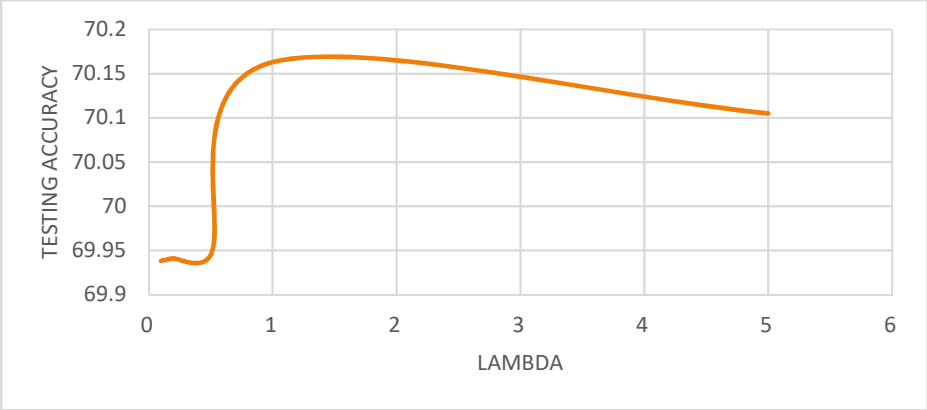
- This matrix holds all the values from training data

- Moore Penrose Pseudoinverse:
  $\mathbf{w}_{\mathrm{ML}} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$
- This value is the inverse of the matrix phi which denotes the closed form solution for the model.
- To prevent over-fitting, a regularization term is added. The solution with regularization incorporated is dented by $\mathbf{w}^* = (\lambda\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$.
- Here, the hyperparameter lambda controls the trade-off between complexity of model and over-fitting.

METHOD 2: STOCHASTIC GRADIENT DESCENT

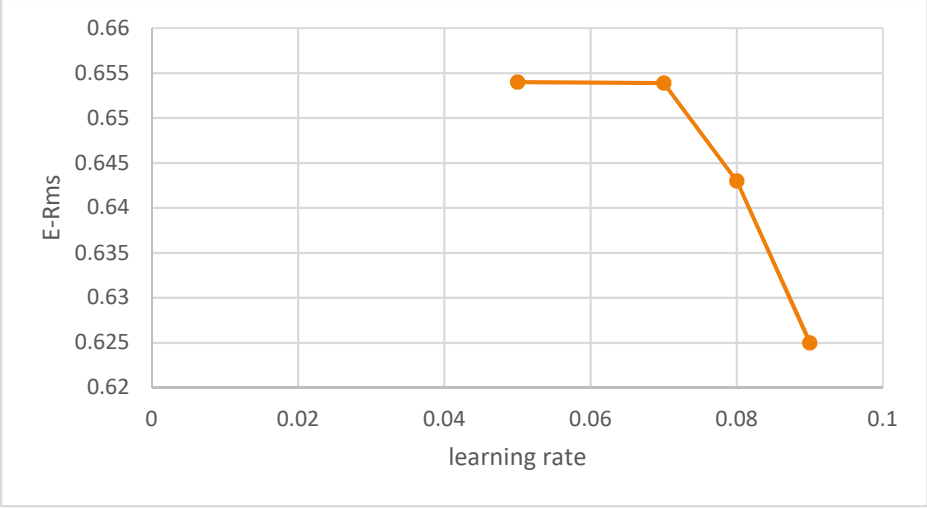
- It is an incremental iterative method in which a representation of the gradient decent of a function that can be iteratively used to generate optimal results.
- In this case, this is done by incrementally updating the weights by adjusting certain quantities known as hyper-parameters.
- Hyper-parameters used:
- Lambda: The tradeoff parameter which balances the complexity of the model and how well it is fitting the data.
- Eta: It is denoted by η which represents the learning rate or the rate or how much data the model is fitting.
- To avoid over-fitting and improve the overall accuracy of the model, regularization is introduced.
- In this method, initially the weight vector is taken as a random value. This value is then updated using the formula $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)}$
- Here, $\Delta\mathbf{w}^{(\tau)}$ decides the weight updates and is of the form $\Delta\mathbf{w}^{(\tau)} = -\eta^{(\tau)}\nabla E$

EXPERIMENT AND OBSERVATIONS

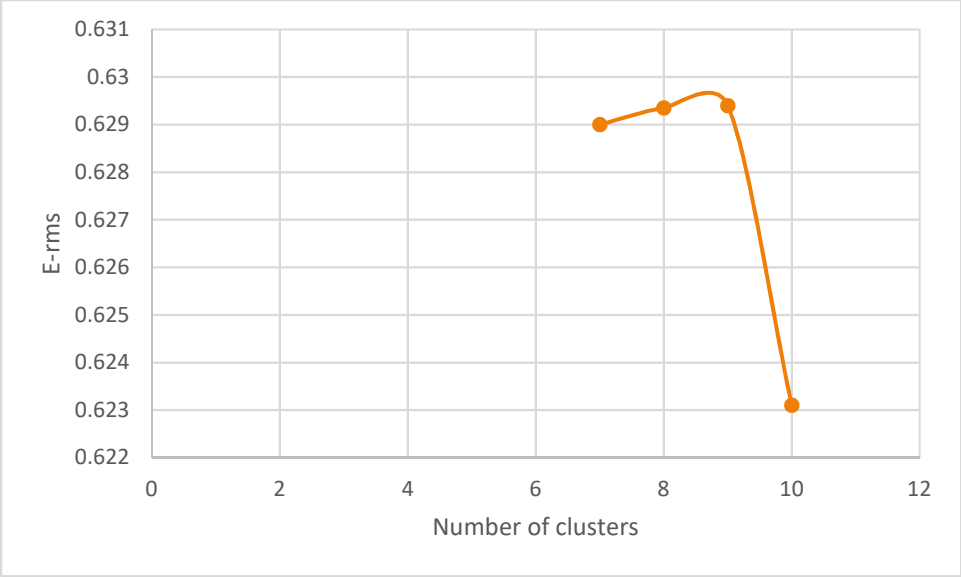


- Graph obtained by plotting testing accuracy vs lambda shows fluctuation in testing accuracy as lambda is increased.

- As the no of clusters increases, the testing accuracy drops initially, then increases.



- As learning rate increases, the root mean square error drops



- As the number of clusters increases, the root mean square error drops

CONCLUSION

- We find that on tuning the hyper-parameters mentioned above, we can observe changes in the generated values.
- Root Mean Square Error:
- This value, denoted by $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N_{\mathrm{V}}}$, is an important evaluation metric used to judge the performance of the model. It represents the difference in the values obtained and the values expected.
- Based on this parameter, we can estimate the amount by which the weights need to be updated.

REFERENCES

- https://en.wikipedia.org/wiki/Radial_basis_function
- https://en.wikipedia.org/wiki/Root-mean-square_deviation