Learning to Rank using Gradient Descent

Paper Review

Kaustav Vats
B.Tech in Computer Science and Engineering
Indraprastha Institute of Information Technology
Delhi, India
kaustav16048@iiitd.ac.in

I. Research Question

Proposed a system that present result to a user that are ordered by the user's utility function. Ranking functions are used for many purposes, example- Search Engines, Query based answers etc. This is very important to provide accurate and most relevant results first and ranking functions are used by many of the platforms.

II. Summary

Authors of the paper have proposed a probabilistic cost function by understanding the gradient descent methods for ranking functions. They have proposed a neural network architecture which uses the ranking function. They have taken a query based data-set, which basically has a set of query and for each query there are a set of returned documents which are ranked for training purpose by labeling them as ex-good match, excellent match etc. Their approach to solve the problem involve learning a ranking function. Their approach provide an ordering and avoiding mapping the output to a rank, so it doesn't require rank boundaries. Authors considered using neural network because of their flexibility and they are often faster than the kernel methods in training phase.

Authors are using toy data and data gathered from the internet search engines. Data consist of the 17,400 queries and each query returns up to 1000 documents, mostly the top documents.

Their model takes a set of pair of samples $[A, B]$ with target probabilities P_{AB}. Model function $f : R^d \rightarrow R$. Rank of the sample is given by f. For $f(x1) > f(x2)$ means that x1 is ranked higher than the x2. Denoting the model posterior value $P(x1 > x2)$ by P_{ij} then the cross entropy loss of the model is $C_{ij} = -P_{ij} \log(P_{ij}) - (1 - P_{ij}) \log(1 - P_{ij})$. Equation after using the logistic function $C_{ij} = -P_{ij} a_{ij} + \log(1 + \exp a_{ij})$. Now C_{ij} is a linear function. Using above function is likely to perform better than a quadratic cost. If P_{ij} is 0.5 then C_{ij} becomes symmetric and gives a way of trainin on the patterns with the same rank.

Model puts consistency on the requirement of the value of posterior. If the consistency is not met, then there wont be a set of output that gives desired pair wise probabilities. For fixed set of A_i's, pairwise probability are $P(A_i | A_i or A_j) = P_i/(P_i + P_j)$. For a single output, cost function is a function of difference of two training samples. The cost function f is chosen to be monotonically increasing. First forward propagation of each sample, Activation and gradient values are stored. Authors have shown that on dropping the index on the last layer, gives a form of difference of the terms which is dependent on the value of the x. Sum of the weights of the neural network are not considered in single output layer and for more layers the sum appears normal. Mostly modification for Neural network are in back propagation to change the network to RankNet.

Authors have also performed some experiments with the Artificial Data. Data consist of 50 dimension feature vector, which has random value interval $[-1, 1]$. Authors also constructed two ranking functions. For the first function they used a 2 layer neural network. First construction of neural network consist of 50 inputs, 10 hidden neurons and one output layer. Each of these neurons have random weights. Authors considered six relevance levels. Labels of each of the sample were computed via neural network and output was stored in one of the relevant bin. For the second architecture, they computed mean of three terms of an input vector. Each term is was scaled to mean 0 and 1 variance of the data. first term was calculated by taking dot product of x with a random vector. For the 2nd term they computed a quadratic equation taking 1 to d randomly permuting and calculating dot product. For the 3rd term, author took 2 random permutation of to form cubic polynomial. They used above 2 ranking functions to create 1000 files and 50 feature vector each. In testing on search engine task, it returns total 50 documents. around 800 were given for training and 100 for validation and rest 100 for testing.

Authors observed that the neural network with same architecture with 1st layer having zero weight value and 2nd layer having random weights, could learn 1000 train vector with zero error. They also observed that on changing the learning rate to 0.001, the pair-wise error and the average cost function were found to be decreasing monotonically on the training set.

Authors extracted query dependent features from different sources. In pre-processing they replaced the counts with log to reduce to the range. This helps the neural network to learn more easily. authors also shuffled their data and observed that unlabeled data were given 0 rating. So ranking accuracies were computed using a normalized discounting cumulative gain. Results were sorted based on the score outputs. They also observed that poor rating was also given sometimes to some relevant documents. To solve this issue authors also trained model on randomly chosen unlabeled data.

Authors trained multiple models like PRank, which is a linear and quadratic model. For the RankNet a linear layer and 2 layer net with 10 hidden nodes were used. rather learning the quadratic equation they simply added a layer of pre-processing, taking features and every quadratic combination, as a new feature set. For each test, each of the algorithm was trained on 100 epochs. Model that gave the best result was kept and used to test on the validation set. They observed that quadratic PRank model is slow because of quadratic features.

Authors have proposed a probabilistic cost function for training a system to learn ranking orders. Their approaches are general and can be used with any other different function. RankNet is simple to train and gives great performance with real world ranking problems like search engines. For future work author suggested to extend the approach to other machine learning methods to do ranking, considering the speed and simplicity of the model makes it more useful for ranking problems.

REFERENCES

[1] Learning to rank gradient descent https://icml.cc/2015/wp-content/uploads/2015/06/icml$_ranking.pdf$