Learning to Rank Using Mixture Of Experts And Matching Loss Functions

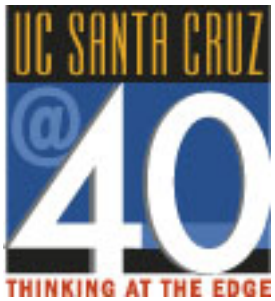
CMPS 290C Advanced Machine Learning Project Report

by

Seinjuti Chakraborty

under the guidance of

Prof. M.K. Warmuth



Department of Computer Science
University Of California,Santa Cruz
June 2007

1 Abstract

Modern Retrieval Engines have been regularly overwhelmed by vast repositories of documents.Organizing this information in an effective ranked list is an important part of retrieval.Though much research has already been done the problem of ranking documents is still far from being solved completely.One of the important reason being the personalized notion of relevance feedback when a query is submitted to the user which causes the ranking order to change continuously.
In this report we propose a novel online approach to solving the ranking problem.A mixture of experts have been used to combine partial ranking functions in order to surmise a global scheme.Another interesting approach that we have investigated is varying the loss functions as new batches of training data come in, in order to determine flexibly the important ranks from a user's perspective.

2 Introduction

Information Retrieval is based on retrieving the top ranked documents corresponding to a user's query using content and/or context information.But if the user is not an experienced searcher, the retrieved documents may not satisfy his current information need.Capturing this process is called relevance feedback.Relevance feedback is an interactive iteration between the user and the retrieval engine to refine and improve the search results,and hence improving the ranked order of returned documents. However since different users have different personalities the documents which might be relevant to one user (hence ranked high) may not be relevant to a different one. Hence the optimal ranking should be sensitive to the user's profile and personality as well as dynamic.
However in the context of the current "Learning to Rank" problem we do not consider the dynamic nature of the problem and instead work towards a better learning scheme which shall improve the performance in a static context.Optimal ranking under current context would be achieved if our ranking mechanism can place all the relevant documents in the highest order ranks followed by less important non-relevant documents.In most cases the topmost ranks are important because the user might not be interested in examining the lower order results at all.
We shall briefly discuss the existing approaches to solve this problem. Herbrich et al.[6] proposed addressing the issue using SVM and Yunbo et al.[1] suggested a modified SVM ranking based on Hinge Loss,which solves a pairwise ranking problem i.e problem of classifying instances as correctly ranked or incorrectly ranked. PRank[4] based on Perceptron usually maintains a set of weight vectors and a set of thresholds, based on which the instance is mapped to an actual rank by projecting each instance onto a real number line. RankBoost (Singer et al. [5])use partial rank orders based on feature scores of instances and combine them to a global ordering.Listwise Ranking Approach [2],on the other hand,

treats an entire list or permutation of the documents as a learning instance and defines a list-wise loss function.
The approach we have discussed in this paper,is an online learning algorithm based on the Winnow family where the experts are weak rankers, and the master algorithm generates a global ranking based on a mistake driven approach on the predicted rankings of the experts.The other interesting approach we have discussed is an adaptive matching loss function which keeps changing based on the number of relevant documents in the current batch of training data.The remainder of the paper is organized as follows.In section 2 we discuss the experts framework,in section 3 we discuss the matching loss functions,in section 4 the evaluation metrics ,in section 5 the experimental results and in section 6 we summarize our current research and future directions.

3 Background

We will be using RankSVM and RankBoost as baseline algorithms for comparison of performance. Hence the 2 algorithms have been briefly described here. The loss function for the RankSVM[1] is given by the following equation

$$L(w) = \sum_{i=1}^{l} \tau_{k(i)} \mu_{q(i)} [1 - z_i \left\langle w_i, x_i^{(1)} - x_i^{(2)} \right\rangle] + \lambda ||w||^2 \tag{1}$$

where μ adjusts the bias across queries and τ adjust the bias between rank pairs. The objective is to penalize queries with more number of relevant documents and give more important to higher rank orders then lower orders.The penalty recieved by the i-th instance pair is determined by the product $\tau \times \mu$. The following plot shows the hinge loss function used by SVM where the x-axis represents the dot product $z.(f(x_1) - f(x_2))$ and y-axis the loss. Here z is the actual label (+/-1) and f is the discriminant function. if $z.(f(x_1) - f(x_2)) >= 1$ loss is 0 and if $z.(f(x_1) - f(x_2)) < 1$ losses are linearly decreasing functions with different slopes of $\tau \times \mu$.
In RankBoost[5] the goal is to combine a set of weak ranking features $f_1, f_2, ...f_n$ where each feature defines a linear ordering of the instances.The feedback information is in the form of relative ranks of individual pairs of instances which can be summarized by a function ϕ where $\phi(x_0, x_1) > 0$ means that x_1 should be ranked above x_0.The learning algorithm attempts to find a final ranking that is similar to the given feedback function.The algorithm tries to find a final ranking H with a small number of crucial-pair misorderings, called the ranking loss $rloss_D(H)$ given by

$$\sum_{x0,x1} D(x0, x1)[[H(x1) <= H(x0)]] = \Pr_{(x0,x1) \in D}[H(x1) < H(x0)]$$

where [[]] indicates predicate 1 or 0 and

$$\sum_{x0,x1} D(x0, x1) = 1$$

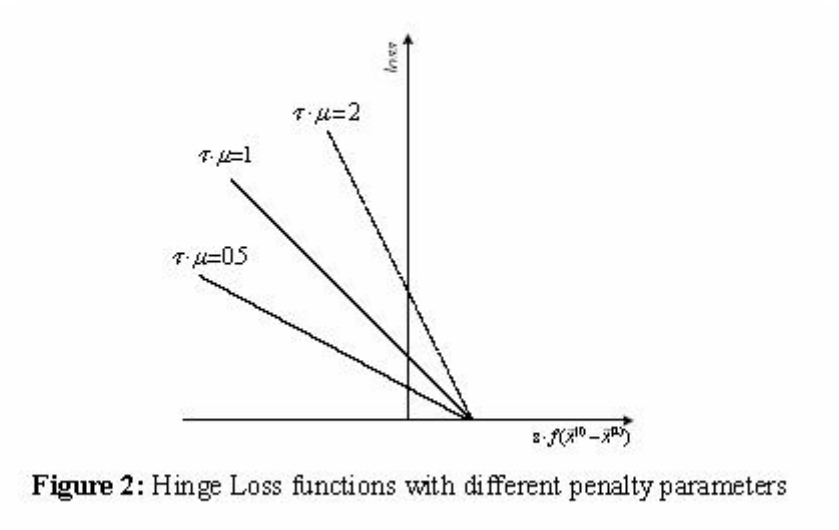


Figure 2: Hinge Loss functions with different penalty parameters

4 Expert Framework

Different versions of Winnow(LittleStone et al.[8])have been successfully used in tasks like text categorization.During the learning phase on each trial,learner makes a prediction and receives a feedback based on which the hypothesis(that can be a set of weights)are updated.It is a mistake driven algorithm and makes conservative update i.e updates only when mistakes are made.We have extended the applications of multiplicative update beyond text categorization,to learning a ranking order.In the algorithm, each expert is a document feature and the feature scores are used as prediction of experts.The individual expert predictions are combined by a greedy ordering algorithm to form a master prediction.The feedback received is in the form of paired instances of relevant and non relevant documents and updates are made based on whether the rank orders predicted by experts place the relevant documents above the non relevant documents.The algorithm has been described below.

Algorithm RankingExperts()

$\beta \in [0, 1]$, initial weight vector $w_i^0 \in [0, 1]$ with $\sum_{i=1}^{N} w_i^0 = 1$

N ranking experts, number of rounds=queries T

Do for t=1,2,...T

1. Receive a training set of documents(labeled relevant or non relevant) X^t and preference functions R_1^t,.....R_N^t

2. use algorithm Order-by-Preferences() to compute ordering function ρ^t

3. Order X^t using ρ^t.

4. Receive feedback F^t from the user.F^t is the pair of docs (u,v) where u is relevant and v is non relevant

5. Evaluate Loss(R_i^t, F^t)= $\sum_{(u,v)\in F^t}$ and $_{f_i(v)>f_i(u)}$ $\frac{R_i(u,v)}{|F^t|}$
where $f_i(v)$ is the score of feature i in doc v and $R_i(u,v) = |f_i(v) - f_i(u)|$
This implies that loss of an expert is the sum of difference in features scores for all pairs where nonrelevant doc has more score than the relevant doc for that expert

6. Set the new weights $w_i^{t+1} = \frac{w_i^t e^{-\eta Loss(R_i^t, F^t)}}{Z_t}$
such that $\sum_{i=1}^N w_i^{t+1} = 1$

7. Fixed share to past $w_i^{t+1} = (1-\alpha) \times w_i^{t+1} + \alpha \times w_i^0$
end for

Algorithm OrderByPreferences()(Singer et al.[9])
(Greedy Ordering)

1. Inputs:A set of vertices(documents)V,a function
PREF(u,v)=$\sum_{i:f_i(u)>f_i(v)} w_i R_i(u,v)$

2. Outputs: An ordering function ρ^t

3. for each $v \in V potential(v) = \sum_{u\in V} PREF(v,u) -$
$\sum_{u\in V} PREF(u,v)$ *(net outflow - net inflow)*
end for

4. **while V is not empty**

   - let $t = argmax_{v\in V} potential(v)$
   - let $\rho^t = |V|$ and $V = V - t$

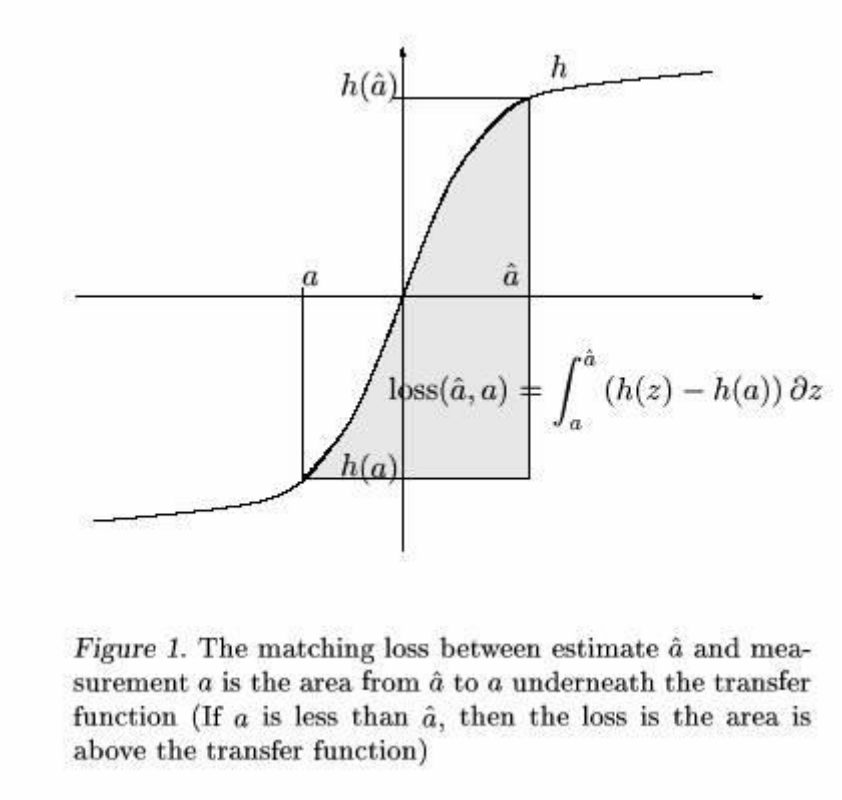


Figure 1. The matching loss between estimate $\hat{a}$ and measurement a is the area from $\hat{a}$ to a underneath the transfer function (If a is less than $\hat{a}$, then the loss is the area is above the transfer function)

- for each v in V update $potential(v) = potential(v) + PREF(t, v) - PREF(v, t)$

end while

The loss bounds in this setting has been shown[3] to be the cumulative loss of the Preference function $PREF^t$ which is bounded by the loss of the best ranking expert

$$\sum_{t=1}^{T} Loss(PREF^t, F^t) <= a_\beta min_i \sum_{t=1}^{T} Loss(R_i^t, F^t) + c_\beta lnN$$

Thus if one of the experts has low loss so will be the loss of the combined preference function. The performance of the actual ordering ρ^t can be related to PREF using triangle inequality

$$Loss(R_\rho, F) <= \frac{DISAGREE(\rho, PREF)}{|F|} + Loss(PREF, F)$$

where

$$DISAGREE(\rho, PREF) = \sum_{u,v:\rho_u > \rho_v} (1 - PREF(u, v))$$

5 Matching Loss

Matching Loss functions are asymmetric loss functions which are used for batch learning where certain instances carry more weightage than other instances.They are based on Bregman Divergence (Manfred et al.[7]).Figure (1) and (2) show that the loss function is

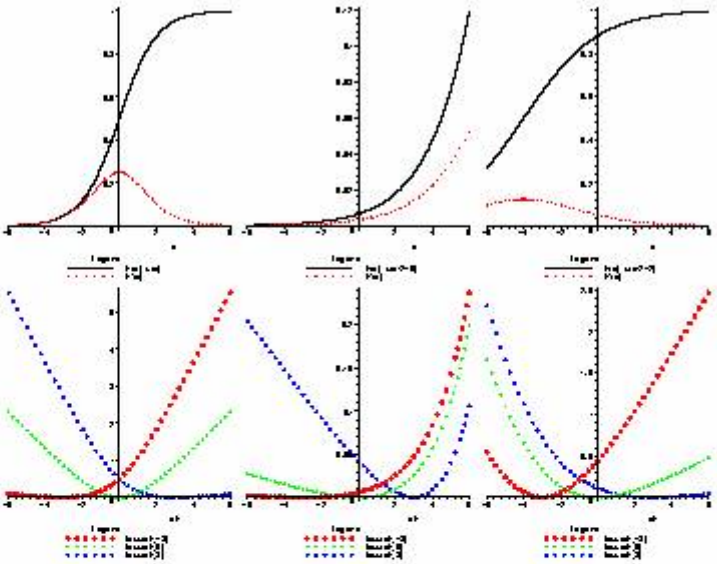


Figure 2. On the top row we plot three choices of the transfer function $h(a)$ and their derivatives: The first is the sigmoid function, i.e. $h(a) = s(a) := \frac{e^a}{1+e^a}$; the others two are shifted versions of the sigmoid. In the bottom row we plot the loss$(\hat{a}, a)$ as a function of the estimate $\hat{a}$ for fixed activities $a = -3, 0, 3$ (The transfer function is always specified in the plot above). Note that locally the losses are quadratic and the steepness of the bowl is determined by $h'(a)$.

the area under the transfer function(sigmoid),it can be varied over instances by a proper choice of h(a) and $h'(a)$.This fact can be used for ranking functions where the top order ranks have higher loss functions than lower order ranks.

5.1 Example Of Matching Loss Functions for a rank order 1 to 10 being important

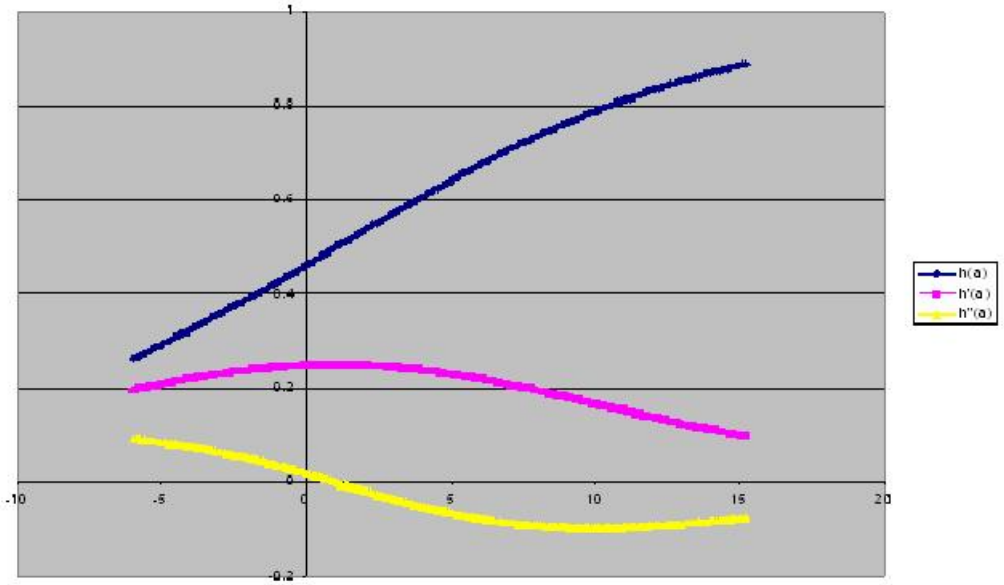


Figure : steeper slope of shifted sigmoid between 1 to 10, then becomes smooth

From figure above, we can derive $h(a) = \frac{e^{\frac{a}{\alpha}+\beta}}{(1+e^{\frac{a}{\alpha}+\beta})}$ $h'(a)$ should maximize at 1 and $h''(a)$ should minimize at 10. So, if x is the number of important rank(no of relevant documents) $\alpha = (x-1)/1.63$ and $\beta = \frac{-1}{\alpha}$

5.2 Algorithm for Ranking with Matching Loss

Assume rank of a document is approximately linear in its features $W.\phi(x)$.The predicted rank $\hat{a} = W.\phi(x)$ where W is a weight vector and $\phi(x)$ is a polynomial in the features of a document such that

$$\phi(x) = (1, x_1, x_2, ...x_1x_1, ..x_ix_j...x_{44}x_{44})$$

where we have been given 44 features. $\phi(x)$ can be chosen to be higher order polynomial or exponential or gaussian kernel.Actual ranks are generated by making the top documents relevant followed by nonrelevant docs in the training set. For each iteration the loss is given by

$$\text{Loss}(\hat{a}, a) = \int_{a}^{W.\phi(x)} (h(z) - h(a))dz$$

where h(a) is the transfer function.In our case we have a sigmoid as a transfer function and hence a logistic loss.

Deriving the loss update The optimal weight vector is given by

$$w^* = \arg\min(w) \left(\lambda ||w||^2 + \sum_{t=1}^{T} loss(w.\phi(x), a) \right)$$

and the loss is the integral

$$Loss(W.\phi(x), a) = \int_{a}^{W.\phi(x)} (h(z) - h(a))dz$$

=>

$$Loss(W.\phi(x), a) = H(W.\phi(x)) - H(a) - (W.\phi(x) - a)h(a)$$

=>

$$Loss(W.\phi(x), a) = \nabla_H(W.\phi(x), a)$$

where $h = \nabla H$, $h(a) = \frac{e^a}{1+e^a}$ and $H(a) = ln(1 + e^a)$

Solving and substituting gives us the logistic loss function

$$Loss(W.\phi(x)) = ln(1 + e^{W.\phi(x)}) - \frac{e^a}{1+e^a}W.\phi(x) + \frac{e^a}{1+e^a}ln(\frac{e^a}{1+e^a}) + (1 - \frac{e^a}{1+e^a})ln(1 - \frac{e^a}{1+e^a})$$

Hence the loss update is

$$\frac{\delta Loss(W.\phi(x)}{\delta W} = \frac{e^{W.\phi(x)}}{1 + e^{W.\phi(x)}}\phi(x) - \frac{e^a}{1+e^a}\phi(x)$$

=>

$$\frac{\delta Loss(W.\phi(x)}{\delta W} = (h(\hat{a}) - h(a))\phi(x)$$

Update the weights by Gradient Descent

Input :Training sample S=set of all documents for all queries Learning rate $\eta > 0$ and penalty weight λ

1. $w = 0$
2. while(stop condition is not met)
3. $\triangle w = 0$
4. $for(i = 0; i < S.size(); i++)$
5. $\triangle w = \triangle w + (h(a) - h(\hat{a}_i))\phi(x_i)$
6. end for
7. $\triangle w = \triangle w - 2\lambda w$
8. $w = w + \eta \triangle w$
9. end while

---

6 Evaluation Metrics

The following methods are used for evaluating the performance of our overall algorithm which needs a little explanation Precision@n

measures the relevance of the top n results of the ranking list with respect to a given query

$$P@n = \frac{\text{no of relevant docs in top n results}}{n}$$

Average precision is defined as average of P@n values for all relevant documents.

$$AP = \frac{\sum_{i=1}^{N}(P@n \times rel(i))}{\#\text{total rel docs for this query}}$$

MAP is the average AP across all queries.

Normalized Discounted Cumulative Gain
NDCG handles multiple levels of relevance.It follows 2 rules:

- Highly relevant docs are more important than marginally relevant docs
- The lower the rank of a doc (irrespective of relevance level) less valuable it is for the user because it is likely not to be examined

4 steps of computing NDCG

1. Compute gain of each document
2. Discount the gain of each document by its ranking position
3. Cumulate the discounted gain of the list
4. Normalize the discounted cumulative gain of the list
5. NDCG value for a position n in a ranking list:

$$N(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{log(1+j)}$$

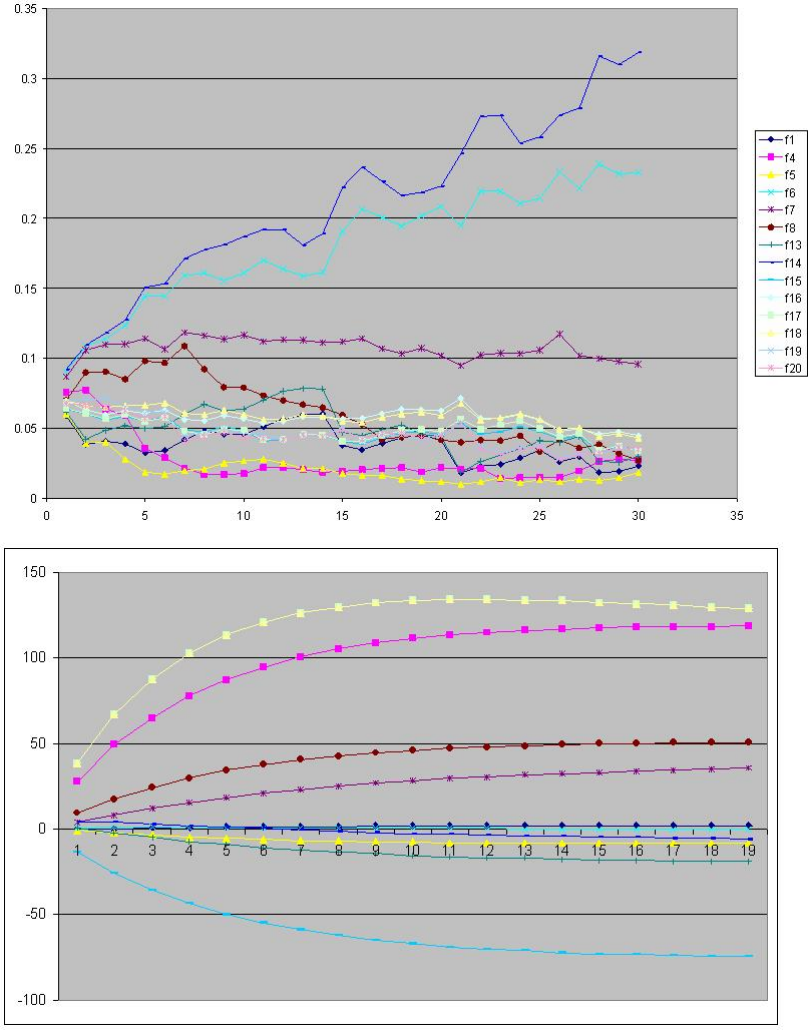
r(j) is the rating of jth document

7 Performance

7.1 Dataset

The dataset we have used is TREC 2003 provided by L̈earning To rank" conference at SIGIR Workshop 2007.The dataset consists of 5 folders, each folder containing a training set,a test set and a validation set.On an average each training set has about 50 queries,each query has on an average 1000 documents;the test set and the validation set have 30 queries each.The documents are vectors of feature scores, the features being BM25,LMIR scores,page rank and HITS scores,link analysis,anchor text etc.[9]

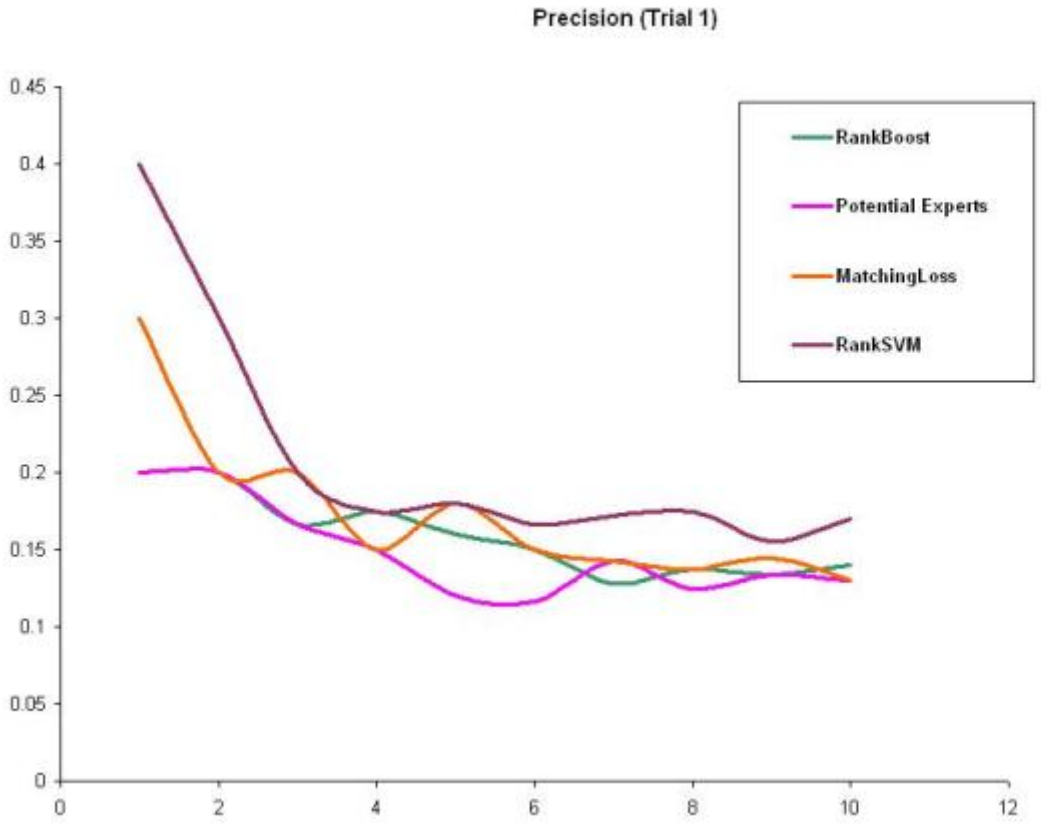
7.2 Weights plot for Experts Framework and Matching Loss

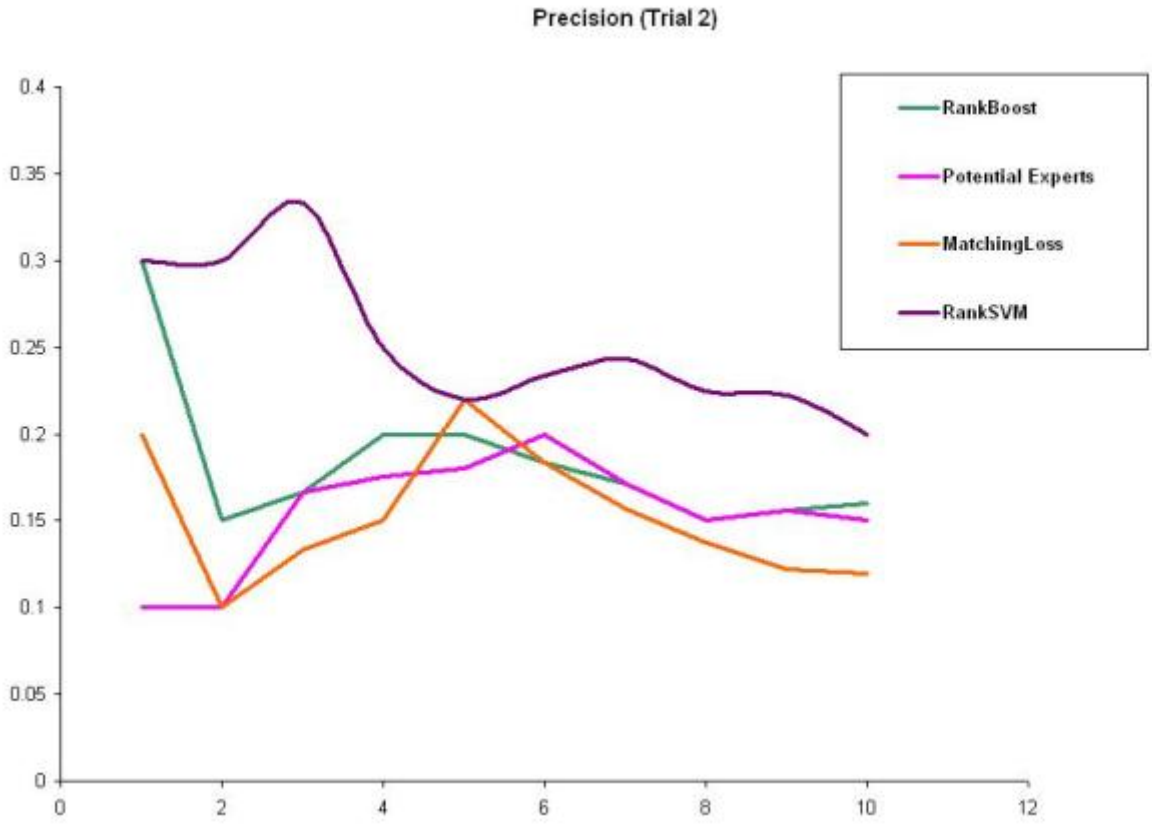


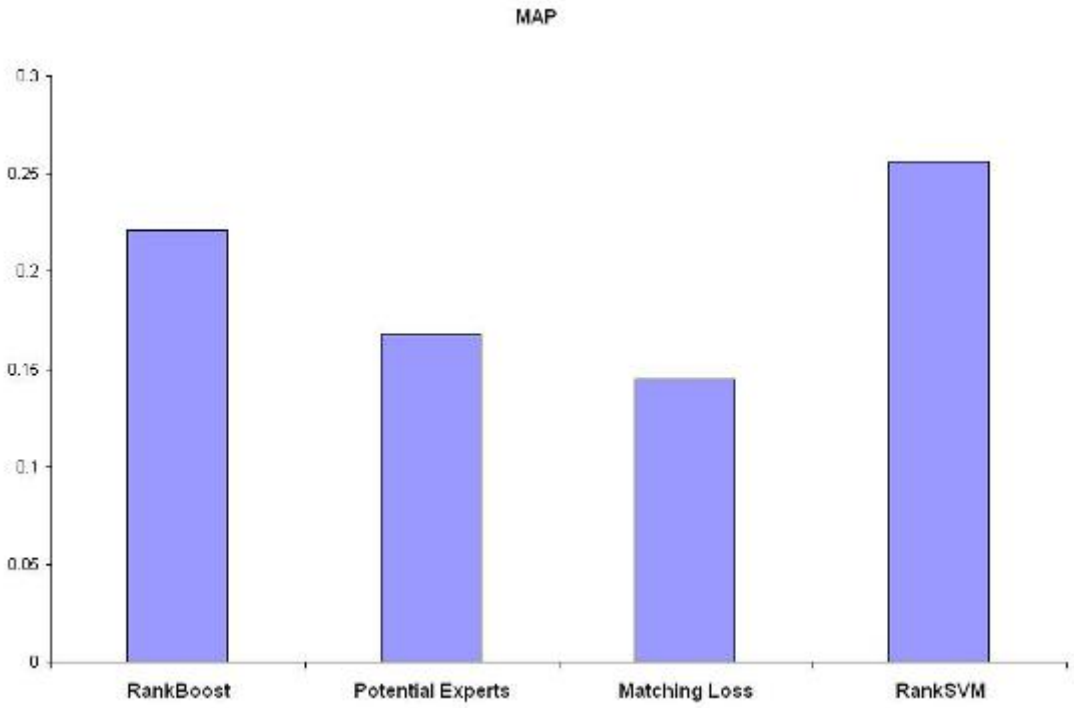
Figures (3)and (4) show the movement in the weight values of the features by application of the experts framework and the matching loss algorithm respectively.In the experts framework 2 particular experts gain majority weight over others.In the matching loss case, the respective weights(of the higher order features) converge and stabilize to a particular value depending on the learning rate.

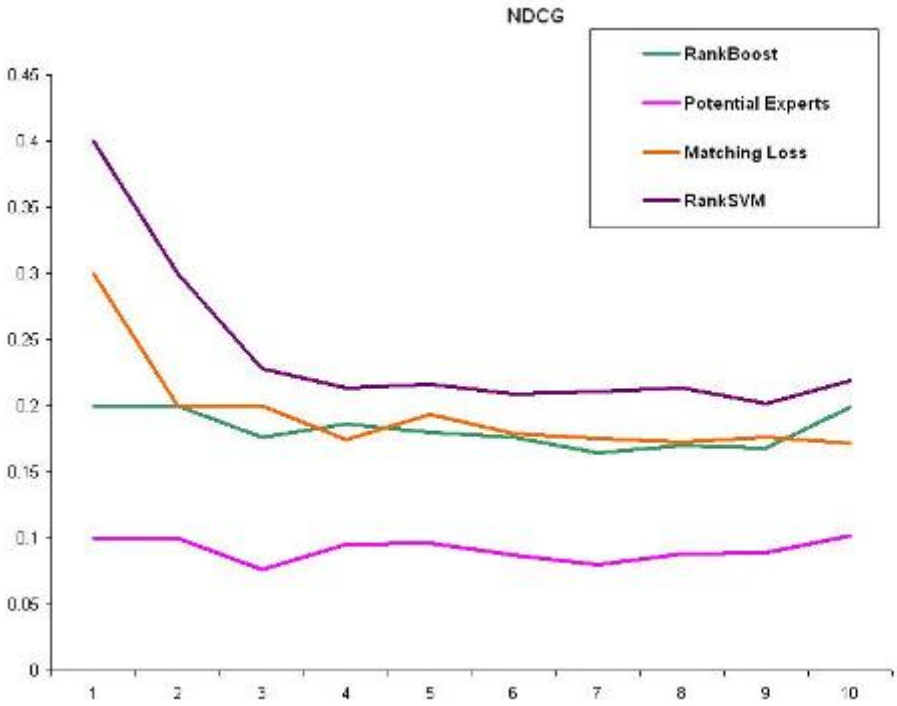
7.3 Precision,MAP and NDCG Measures

The precision plot for 2 different batch modes (Trial 1 and Trial 2) have been shown below which illustrates that the matching loss based algorithm has comparable performance with RankBoost but RankSVM still outperforms all other methods.The MAP and NDCG plots have also been shown in the figures below.We hope to improve performance more in future.


Precision (Trial 1)
RankBoost
Potential Experts
MatchingLoss
RankSVM
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
12


Precision (Trial 2)
RankBoost
Potential Experts
MatchingLoss
RankSVM
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
12





8 Future Work

We still need to improve the loss functions for a better performance.We intend on working towards proving rigorous loss bounds for the experts framework and the matching loss.The parameters of the algorithm still need to be tuned with respect to the validation set provided.As a future work we intend to try conservative updates,exponentiated gradient descent updates and

compare and contrast with the current gradient descent updates. Matching Loss seems to be be a very promising idea and can be extended to any classification problem.The system requirements of these online or batch updates is very reasonable which gives fast computing power to the algorithms. Another new algorithm that we would also like to investgate is the TotalBoost algorithm and compare its performance to RankBoost

9 Acknowledgements

Our sincere thanks to Professor M.K Warmuth for supporting this work and giving new insights into solving this problem. We thank our friends whose valuable discussions and critique helped us overcome some of the doubts and questions we had regarding the algorithms. It was a wonderful course to learn new research directions.

References

[1] Y. Cao, J. Xu, T.Liu, H.Li, Y.Huang, and H. Hon. Adapting ranking svm to document retrieval. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 186–193. ACM SIGIR, 2006.

[2] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank:from pairwise approach to listwise approach. *Microsoft Research Technical Report*, 40, April 2007.

[3] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal Of Artifical Intelligence Research*, 10:243–270, 1999.

[4] K. Crammer and Y. Singer. Pranking with ranking. *Advances in Neural Information Processing Systems*, 1(14):641–648, 2002.

[5] Y. Freund, R. Iyer, R. E.Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal Of Machine Learning Research*, 4(5):933–969, 2003.

[6] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[7] J.Kivinen and M.K.Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, December 2001.

[8] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, April 1988.

[9] T. Qin, T.-Y. Liu, and H. Li. The trec datasets in letor. 2007.