Listwise Approach to Learning to Rank – Theory and Algorithms

Presented by Jay Shah

Listwise Approach - Overview

- Takes ranked lists of objects as instances
- Trains a ranking function through a listwise loss function.
- Claimed to perform better on IR than Pointwise/Pairwise

Listwise Approach – Expected Loss

$$R(h) = \int_{X \times Y} l(h(x), y)\, dP(x, y)$$

$$loss\ function = l(h(x), y) = \begin{cases} 1, & if\ h(x) \neq y \\ 0, & if\ h(x) = y \end{cases}$$

Listwise Approach - Sampling

$$R_s(h) = \frac{1}{m}\sum_{i=1}^{m} l(h(x^{(i)})), y^{(i)})$$

$$h(x^{(i)}) = sort(g(x_1^{(i)}), \ldots, g(x_n^{(i)}))$$

$$R_s(g) = \frac{1}{m}\sum_{i=1}^{m} l(sort(g(x_1^{(i)}), \ldots, g(x_n^{(i)})), y^{(i)})$$

Listwise – Surrogate Loss

$$R_s(g) = \frac{1}{m}\sum_{i=1}^{m} l(sort(g(x_1^{(i)}),\ldots,g(x_n^{(i)})), y^{(i)})$$

$$\downarrow$$

$$R_s^{\phi}(g) = \frac{1}{m}\sum_{i=1}^{m} \phi(g(x^{(i)}), y^{(i)})$$

Theoretical Analysis

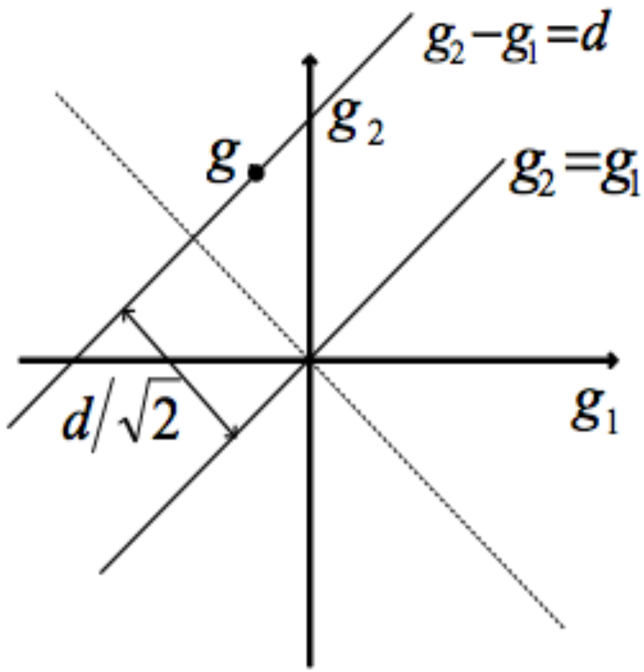
- Consistency
- Soundness
- Continuity, differentiability and convexity
- Computational efficiency

Case Studies

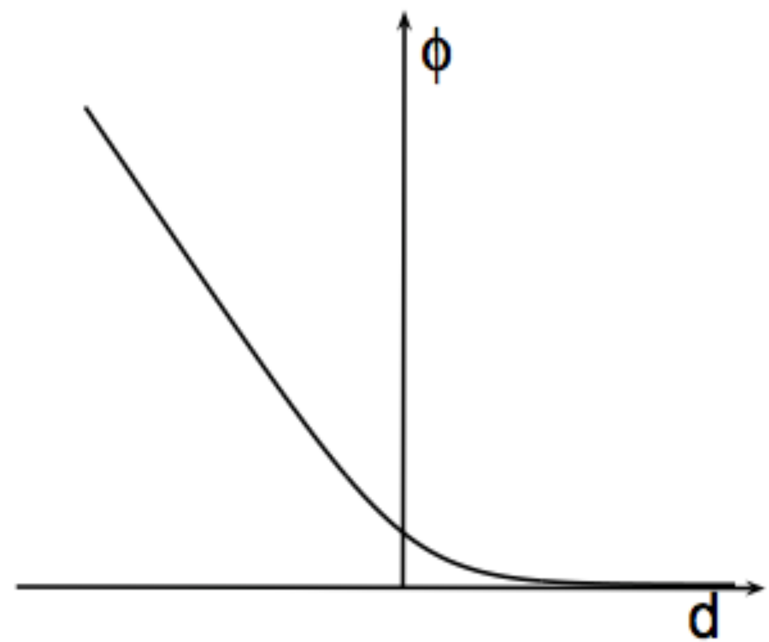
- Likelihood Loss → ListMLE
- Cosine Loss → RankCosine
- Cross Entropy Loss → ListNet

Likelihood Loss

$$\phi(g(x), y) = -\log \prod_{i=1}^{n} \frac{\exp(g(x_{y(i)}))}{\sum_{k=i}^{n} \exp(g(x_{y(k)}))}$$



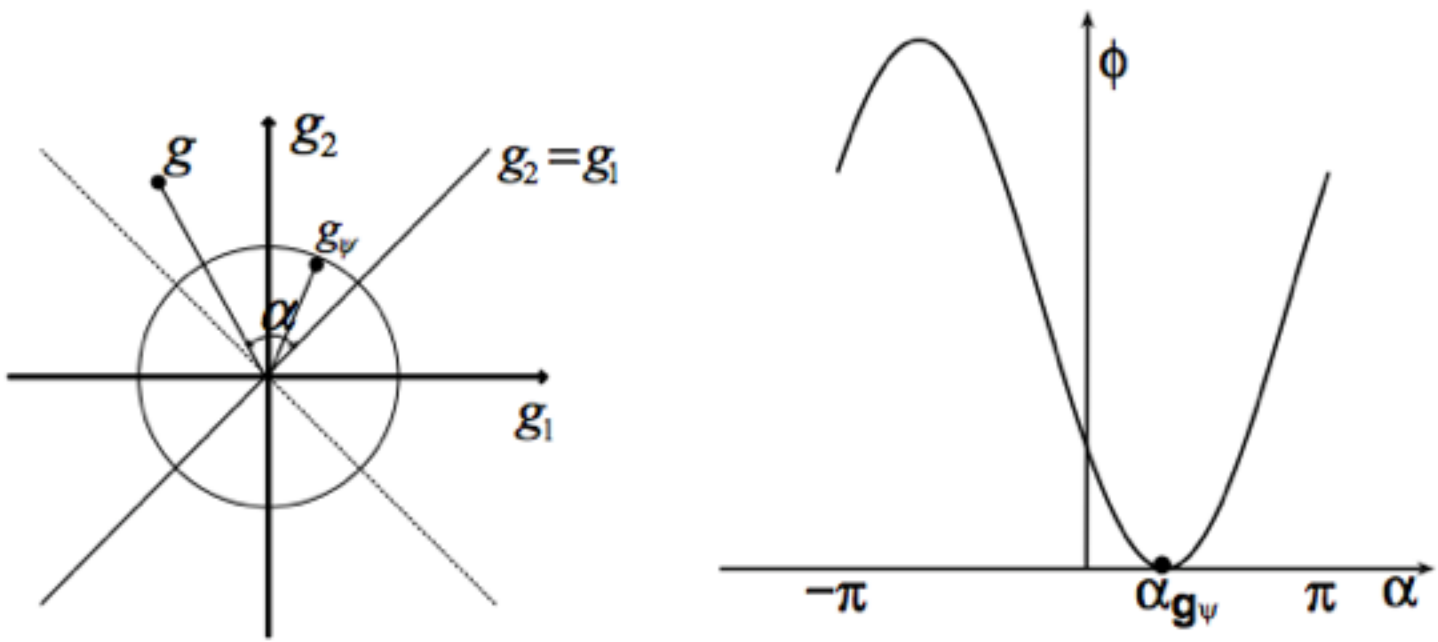
(a)



(b)

Cosine Loss

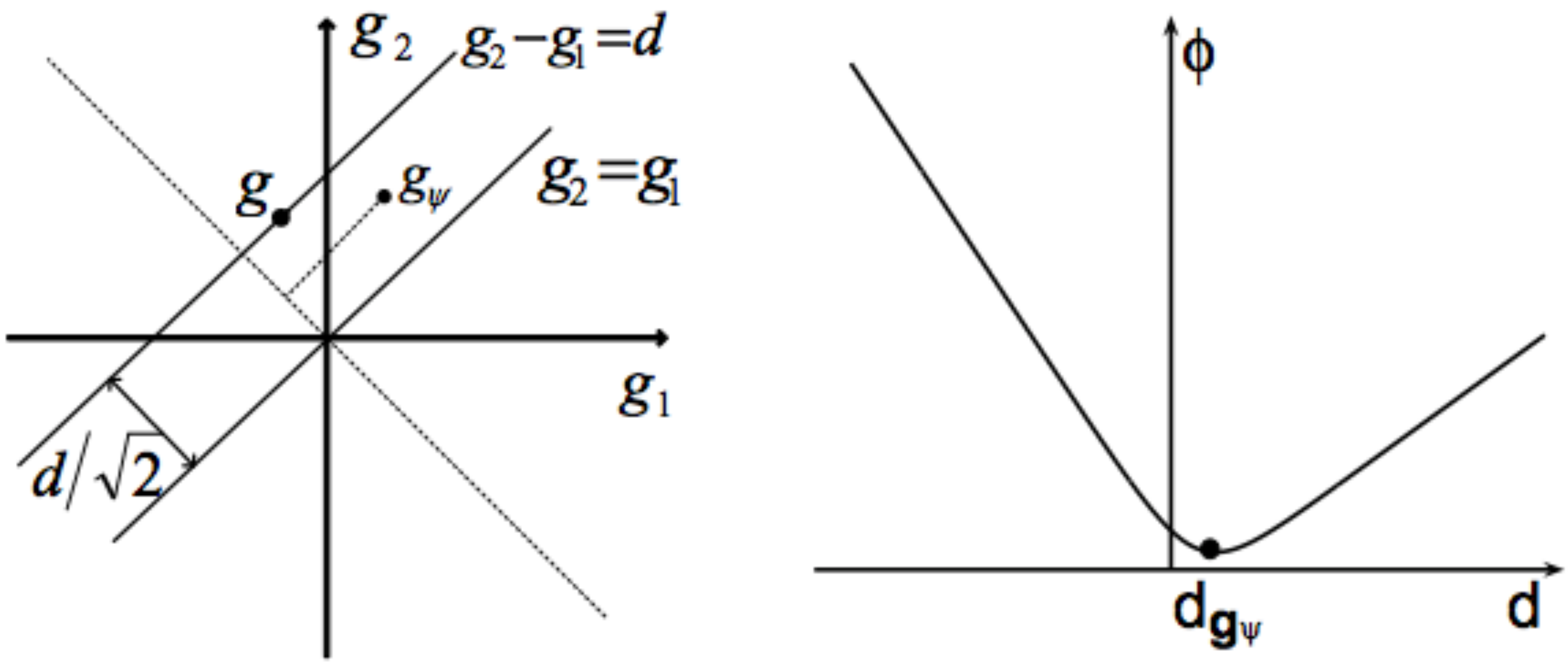
$$\phi(g(x), y) = \frac{1}{2}\left(1 - \frac{\psi_y(x)^T g(x)}{\|\psi_y(x)\| \, \|g(x)\|}\right)$$



(a) (b)

Cross Entropy Loss

$$\phi(g(x), y) = D(P(\pi \mid x; \psi_y) \,\|\, (P(\pi \mid x; g))$$



(a) (b)

Surrogate Loss Comparison

Loss	Consistency	Soundness	Continuity	Differentiability	Convexity	Complexity
Likelihood	✓	✓	✓	✓	✓	O(n)
Cosine	✓	✕	✓	✓	✕	O(n)
Cross Entropy	✓	✕	✓	✓	✓	O(n! x n)

ListMLE

Algorithm 1 ListMLE Algorithm

Input: training data$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$
Parameter: learning rate η, tolerance rate ϵ
Initialize parameter ω
repeat
  for $i = 1$ **to** m **do**
    Input $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ to Neural Network and compute gradient $\triangle\omega$ with current ω
    Update $\omega = \omega - \eta \times \triangle\omega$
  end for
  calculate likelihood loss on the training set
until change of likelihood loss is below ϵ
Output: Neural Network model ω

Experiment on Synthetic Data

- Randomly sample a point on area [0,1] x [0,1]
- Assign score using $y = x_1 + 10x_2 + \varepsilon$
- Generate 15 points and scores this way.

Experiment on Synthetic Data

Algorithm	Accuracy	MAP
ListMLE	0.92 ± 0.011	0.999 ± 0.002
ListNet-log	0.905 ± 0.010	0.999 ± 0.002
ListNet-sqrt	0.917 ± 0.009	0.999 ± 0.002
ListNet-l	0.767 ± 0.021	0.995 ± 0.003
ListNet-q	0.868 ± 0.028	0.999 ± 0.002
ListNet-exp	0.832 ± 0.074	0.997 ± 0.004
RankCosine-log	0.180 ± 0.217	0.948 ± 0.034
RankCosine-sqrt	0.080 ± 0.159	0.886 ± 0.056
RankCosine-l	0.917 ± 0.112	0.999 ± 0.002
RankCosine-q	0.102 ± 0.161	0.890 ± 0.060
RankCosine-exp	0.047 ± 0.163	0.746 ± 0.136

Experiment on OHSUMED Data

- 106 queries, 16,140 query-document pairs.
- Definitely relevant, possibly relevant, or not relevant.
- Normalized Discounted Cumulative Gain (NDCG).

Experiment on OHSUMED Data

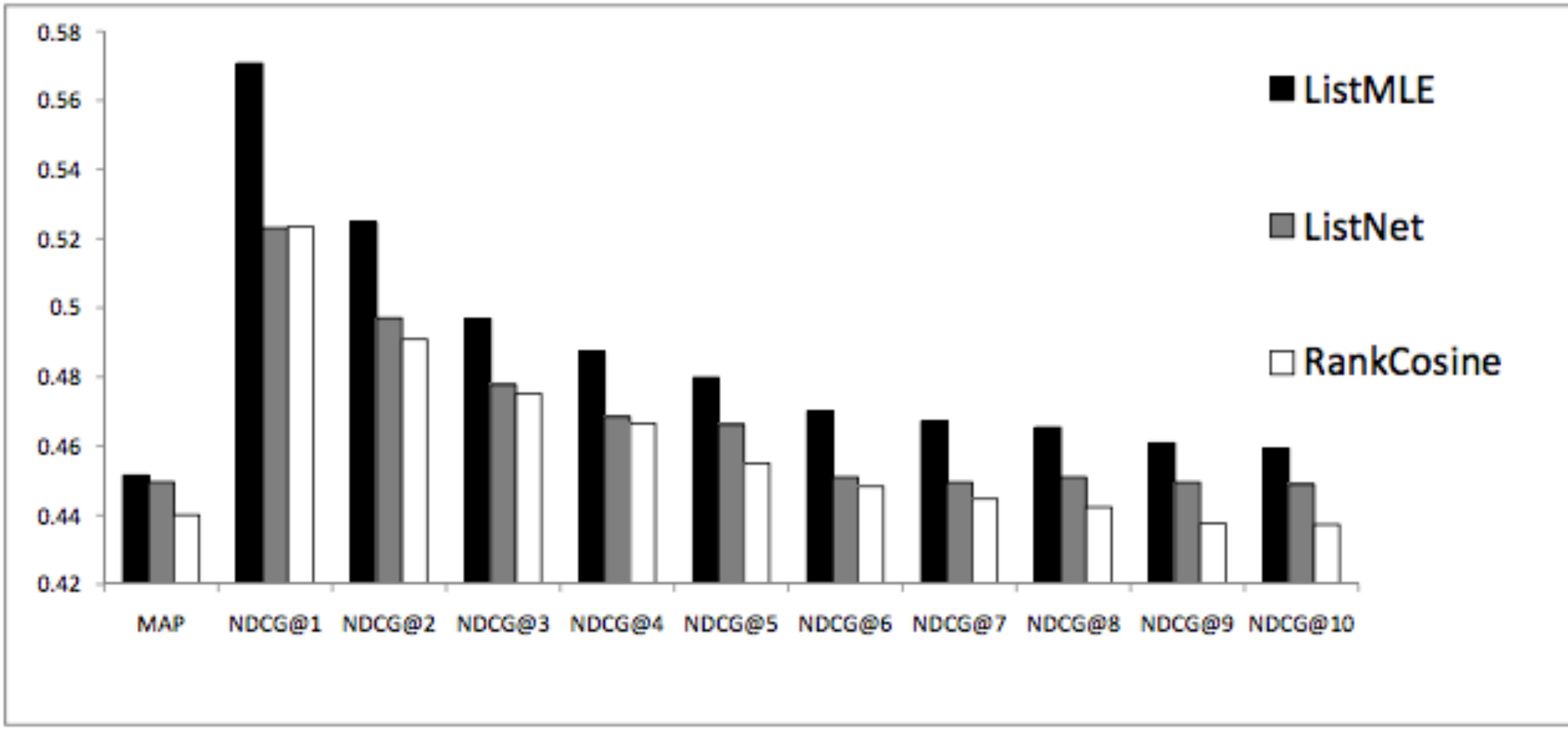


Figure 4. Ranking performance on OHSUMED data.

Future Work

- More theoretical analysis on properties of loss functions.
- Cost sensitive loss function instead of 0 - 1 loss.
- Investigate other surrogate loss functions

Thank you for listening

Questions?