Recommender Systems || Lecture 12

Summer 2022

Dr. Karthik Mohan

UW, Seattle

August 9, 2022

Today

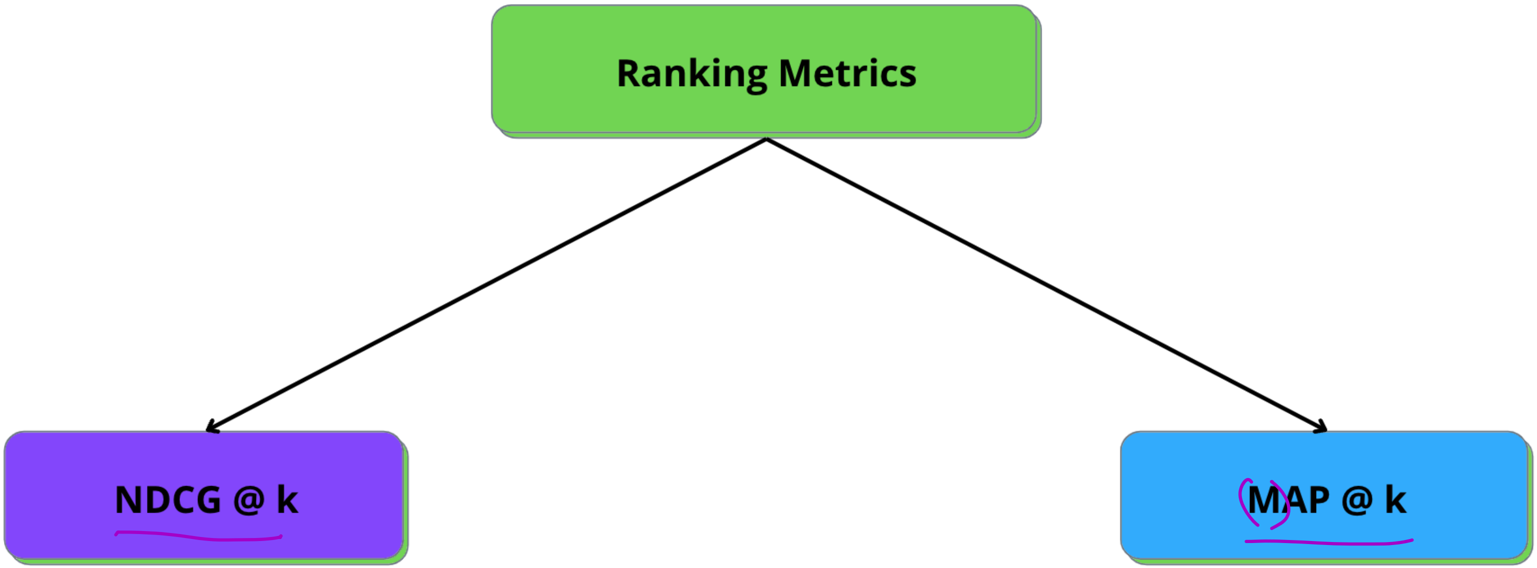
Today

1. Ranking loss function and metrics

Today

1. Ranking loss function and metrics
2. Page Ranking and Ranking Algorithms

Ranking Metrics



AP – Avg. Precision
(Average of Precision)

MAP – multiple queries
(mean of AP over queries)

Learning to Rank Survey

Ranking Metrics Example

Recommendations (model)

N=5

Compare

Purchases (Ground Truth)

N=6

Metrics

Let's compute all the metrics we know!

`Precision`, `Recall`, `Precision @ k`,
`Average Precision @ k`, `NDCG @ k`

Ranking Metrics Example

Recommendations

1 0 1 0 1

Purchases

Precision ?

$= 3/5$ → #Relevant recommendation / → #recommendations $= 0.6$

Assume:-
(care about the exact flavor/product)

Ranking Metrics Example

Recommendations

1 0 1

Purchases

Precision @ 3 ?

$$= \frac{2}{3} = 0.67$$

Ranking Metrics Example

vit. water zero

Recommendations

1 1 1

Purchases

Category Precision @ 3 ? $= \frac{3}{3} = 1$

Assume:- We don't care about the particular flavor

Model Metrics
Business Metrics
Gap?!

Ranking Metrics Example

Recommendations

} 5

Purchases
(Ground Truth)

} 6

Recall ?

$\frac{3}{6} \rightarrow$ # purchases (Ground Truth)

ICE # 1

Recommendations

Purchases

Assume:- Flavor doesn't matter for a match

Recall @ 3 ?

1. 0.3
2. 0.4
3. 0.5
4. 0.6

$\frac{3}{6}$ → Great ~~Good~~ Recall@3

pollev.com/karthikmohan088

Average Precision

Recommendations

might be lesser than K summation

$$AP@K = \frac{\sum_{i:\, i \text{ is relevant}} P@i}{K}$$

Purchases

Average Precision ?

Relevant!

$$AP@2 = \frac{\sum_{i:\, i \text{ is relevant}} P(i)}{2} = \frac{P@1 + P@2}{2} = \frac{1 + 2/2}{2} = 1$$

ICE # 2: Average Precision @ 4

Recommendations

?

Purchases

Average Precision @ 4 ?

1. 0.5
2. 0.75
3. 0.9
4. 1

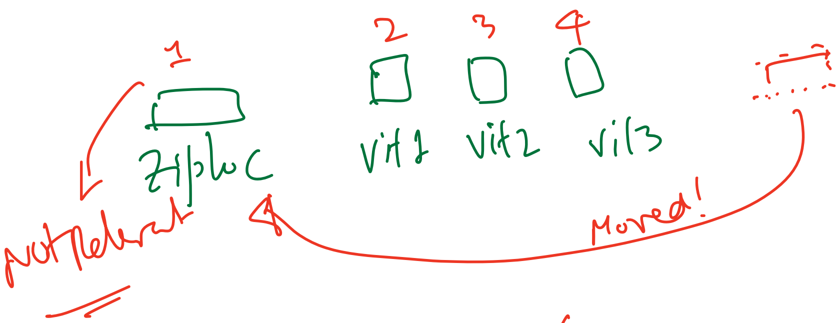
Relevant Relevant Rel Not Relevant

$$\frac{P@1 + P@2 + P@3 + \cancel{P@4}}{4} \leftarrow (@4!)$$

Relevant

$$= \frac{1+1+1}{4} = 0.75$$

pollev.com/karthikmohan088



$$AP@4 = \frac{\cancel{P@1} + P@2 + P@3 + P@4}{4}$$

$$= \frac{1/2 + 2/3 + 3/4}{4}$$

$$= \frac{0.5 + 0.67 + 0.75}{4} = \frac{1.92}{4} \approx 0.5!!$$

AP@4 moved from 0.75 to 0.5 with a single shift in an recommended item

P@4 = 0.75 (still the same!!)

DCG

Discounted Cumulative Gain

$$\mathrm{DCG} = \underbrace{\sum_{i=1}^{N}}_{\text{Cumulative}} \frac{\overbrace{y(d_i)}^{\text{Gain}}}{\underbrace{\log_2(\mathrm{rank}(d_i) + 1)}_{\text{Discount}}}$$

Gain (explanational rating)

Ranking Metrics Example — NDCG @ 3

1 0.8 0.9 0 0.7 Gradation / Grade

1 1 1 | 0 Relevant or not 1

Recommendations

Purchases

NDCG @ 3

$$DCG@3 = \frac{2^{1}}{\log(1+1)} + \frac{2^{0.8}}{\log(2+1)} + \frac{2^{0.9}}{\log(3+1)}$$

$$\max DCG@3 = \frac{2^1}{\log(1+1)} + \frac{2^{0.9}}{\log(2+1)} + \frac{2^{0.8}}{\log(3+1)}$$

(corresponds to perfect ranking)

$$NDCG@3 = \frac{DCG@3}{\max DCG@3} = 0.995 \simeq 1$$

$0 \leq NDCG \leq 1$!

Ranking is close to perfect!!

Summary of Metrics

Metric	Description
Precision	
Recall	
Precision @ k	
F-score	
Average Precision @ k	
NDCG @ k	

$\frac{\#\text{ Relevant items}}{\#\text{ rec items}}$

$\frac{\#\text{ Relevant items Recomm}}{\#\text{ Relevant items}}$

Proxy to determine good Ranking

Learning to Ranking Formulation

Loss Algo (SGD) Search/Page

Given a query q and a set of documents $D = (d_1, \ldots, d_n)$:

Pointwise

Input: Single candidate

$x = (q, d_i)$

Loss: How accurate is the predicted score $s_i \approx y_i$?

Solution: Transform task into Regression.

Pairwise

Input: Pair of candidates

$x_i = (q, d_i)$ and $x_j = (q, d_j)$

→ Ground Truth

Loss: If $y_i > y_j$, then are the predicted scores $s_i > s_j$?

prediction

Solution: Transform task into Binary Classification.

Listwise

Input: Whole list of candidates

$x_1 = (q, d_1)$ .. $x_n = (q, d_n)$

Loss: Takes into account position of all retrieved docs.

Solution: Incorporate evaluation metrics (e.g. DCG) into loss.

1o item

True Ranking   Recommended ranking

Learning to Ranking Formulation

Point wise loss

predicted score

→ Regression Loss

$$L(\underline{s}, \underline{y}) = \sum_{i=1}^{n} (s_i - y_i)^2$$

Truth

Learning to Ranking Formulation

Drawback:
No position info included in Loss!!

Pairwise (Binary Cross Entropy loss)

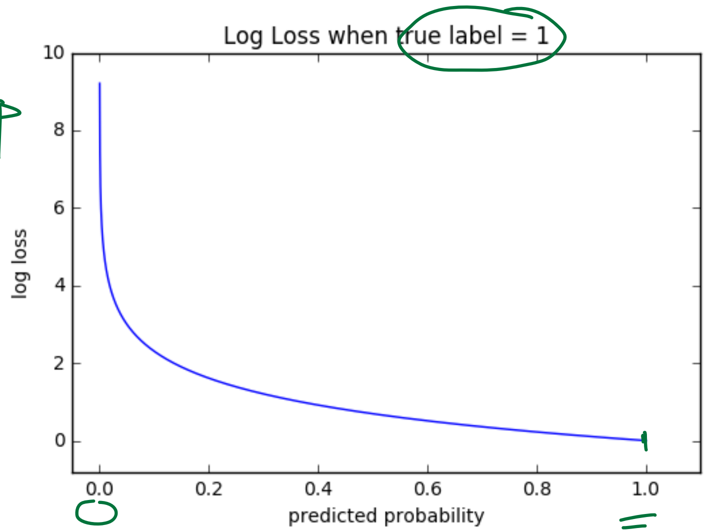
Let $y_{ij} = 1$

$$L(\underline{s}, \underline{y}) = -\sum_{i,j=1}^{n} y_{ij} \log(s_{ij}) + (1 - y_{ij}) \log(1 - s_{ij})$$

(i,j) d_i, d_j

Cross Entropy behavior

Loss $\to \infty$ at 0



$y_{ij} = 1 \Rightarrow d_i$ is more relevant than document j

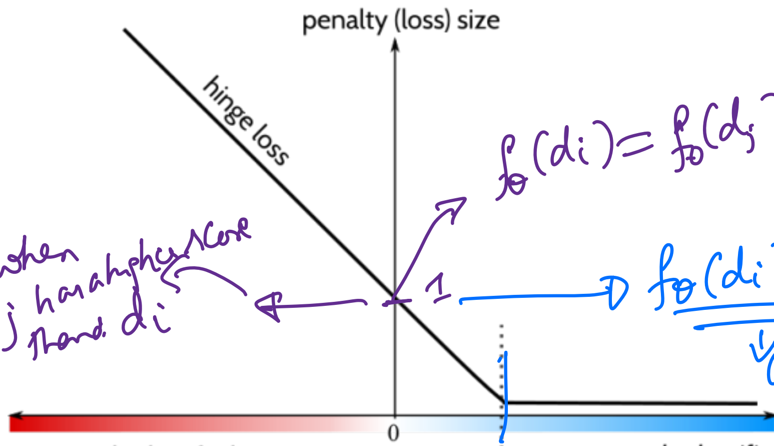
Pros/ Advantage:-
Pairwise comparison
Easy to obtain Ground Truth!

Learning to Ranking Formulation

Pairwise Hinge loss

$$\mathcal{L}_{\textsf{pairwise}} = \sum_{\underbrace{y(d_i) > y(d_j)}_{y_{ij}=1}} \max\Big(0,\ 1 - \big(f_\theta(d_i) - f_\theta(d_j)\big)\Big)$$

Hinge Loss Behavior



when d_j has a higher score than d_i

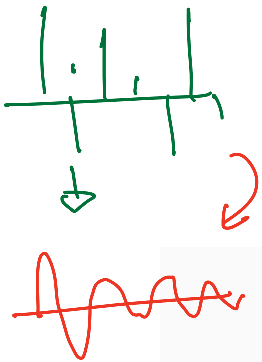
1

$f_\theta(d_i) = f_\theta(d_j)$

$f_\theta(d_i) > f_\theta(d_j)$

Good place

DCG



Incorporate position in Learning to Rank

Approx

Smoother ver.

$$\text{DCG} = \sum_{i=1}^{N} \frac{y(d_i)}{\log_2(\text{rank}(d_i) + 1)}$$

Not Differentiable

rank(di) = f(Sorting(Scores))

not differentiable!!

$\Rightarrow$ Cannot compute gradient

$\Downarrow$

Cannot use SGD!

Lambda Rank Proxy Loss

Heuristic

SOTA

Logistic loss (Logistic Regression)

Proxy for Gradient

$$\underbrace{\mathcal{L}_{LambdaRank}}_{\text{Model Algo}} = \sum_{y(d_i)>y(d_j)} \log\left(1 + e^{f_\theta(d_j)-f_\theta(d_i)}\right) \cdot \underbrace{|\Delta \mathrm{DCG}|}_{\text{Accounts for position}}.$$

Doesn't account for position

Learning to Ranking Formulation

Given a query q and a set of documents $D = (d_1, \ldots, d_n)$:

Pointwise

Input: Single candidate
$x = (q, d_i)$

↓

Loss: How accurate is the predicted score $s_i \approx y_i$?

↓

Solution: Transform task into Regression.

Pairwise

Input: Pair of candidates
$x_i = (q, d_i)$ and $x_j = (q, d_j)$

↓

Loss: If $y_i > y_j$, then are the predicted scores $s_i > s_j$?

↓

Solution: Transform task into Binary Classification.

Listwise

Input: Whole list of candidates
$x_1 = (q, d_1)$ .. $x_n = (q, d_n)$

↓

Loss: Takes into account position of all retrieved docs.

↓

Solution: Incorporate evaluation metrics (e.g. DCG) into loss.

Mapping Loss Functions to Ranking Models

Loss Function Type	**Loss Function**	**Ranking Model**
Pointwise	Quadratic	
Pairwise	Binary Cross-Entropy/ Hinge Loss	RankNet, LambdaRank, LambdaMart
Listwise		SoftRank

Optimizes Approx DCG

Next Class

Chris Burges

1. Deeper look into Rank models including LambdaRank
2. Neural models for Search Ranking and Re-ranking