Metrics and losses

Vladislav Goncharenko
Video recommendations
team lead, Dzen

girafe ai

MSU & MIPT,
fall 2023

Lecture plan

- Offline metrics
- Online metrics
- Special properties
- Losses
  - RankNet
  - LambdaRank
  - LambdaMART
  - YetiRank

RMSE

girafe
ai

01

Offline metrics

RMSE – was offered at the famous Netflix Prize competition

$$RMSE = \sqrt{\sum_{i=1}^{N} (score_i - rating_i)^2}$$

RMSE: disadvantages (for recommendation systems)

RMSE evaluates the quality of predicted ratings for all objects that users have interacted with, while the goal of most recommendation systems is to find the top N most relevant objects. Moreover, rmse can greatly over/under-tell, which is very bad.

It follows from this: ranking metrics used in information search are well suited for evaluating recommendation systems

Precision@k

girafe ai

02

Precision@k

The proportion of relevant recommendations among the top k recommended

$$P@k = \frac{the\ number\ of\ relevant\ documents\ in\ top\ k}{k}$$

The key drawback is that it does not take into account the positions of documents

Precision@k

Prediction	Truth
0.90	?
0.88	1
0.78	?
0.67	0
0.66	1

Precision@3 = ?

Precision@k

Not all ratings are known!

Only known ratings should be taken into account when sorting!

Prediction	Truth
0.90	?
0.88	1
0.78	?
0.67	0
0.66	1

Precision@3 = 0.66

Deep Structured Semantic Models: how to teach?

The sum of P@k for relevant documents divided by the number of relevant documents the user has.

$$AP@k = \frac{1}{n\_rel} \sum_{k=1}^{k} I \cdot P@k$$

where I=1 if the document is relevant and 0 otherwise

Solves the problem of accounting for positions

Mean Avg Precision@k (MAP@K)

Averaging AP@k over the entire test sample of size N

$$\mathrm{MAP@}k = \frac{1}{N}\sum_{i=1}^{N} AP@k_i$$

Discounted Cumulative Gain (DCG)

girafe ai

03

DCG: Background

1. The most important documents in the top issue
2. Confusing the positions of the 90th and 100th aitems is not the same as confusing the 1st and 10th

How to ensure that the quality of ranking in the top of the output has a greater impact on the value of the metric?

Discount according to the position of the document!

DCG

DCG is the accumulated amount of document relevancy discounted by the logarithm of the document position in the output. Not normalized.

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$$

DCG

Why discount on the logarithm?

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$$

DCG

Why discount on the logarithm? The logarithm mathematically expresses the premises of the formula:

$$\log_2 1 = 0, \log_2 10 = 3.32$$

$$\log_2 100 = 6.64, \log_2 110 = 6.78$$

NDCG

girafe
ai

04

NDCG

NDCG – normalization on DCG of ideal ranking

$$NDCG_k = \frac{NDCG_k}{IDCG_k}$$

NDCG: Example

Position in the issue	Ground Truth		Recommendations	
	Order of documents	Rating	Order of documents	Rating
1	d4	2	d3	2
2	d3	2	d2	1
3	d2	1	d4	2
4	d1	0	d1	0
DCG	**3.76**		**3.26**	
NDCG	**1.0**		**0.86**	

$$DCG_{GT} = 1 + \frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} = 3.76$$

$$DCG_{rec} = 1 + \frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4} = 3.26$$

$$\text{N}DCG = \frac{DCG_{rec}}{DCG_{GT}} = 0.86$$

Mean Reciprocal Rank (MRR)

girafe ai

05

Mean Reciprocal Rank (MRR)

RR – reverse rank of the first relevant document

$$RR = \frac{1}{rel\_1st\_pos}, \qquad MRR = \frac{1}{N}\sum_{i=1}^{N} RR$$

Disadvantages: takes into account only the first relevant document, ignoring the subsequent ones

Online metrics

girafe
ai

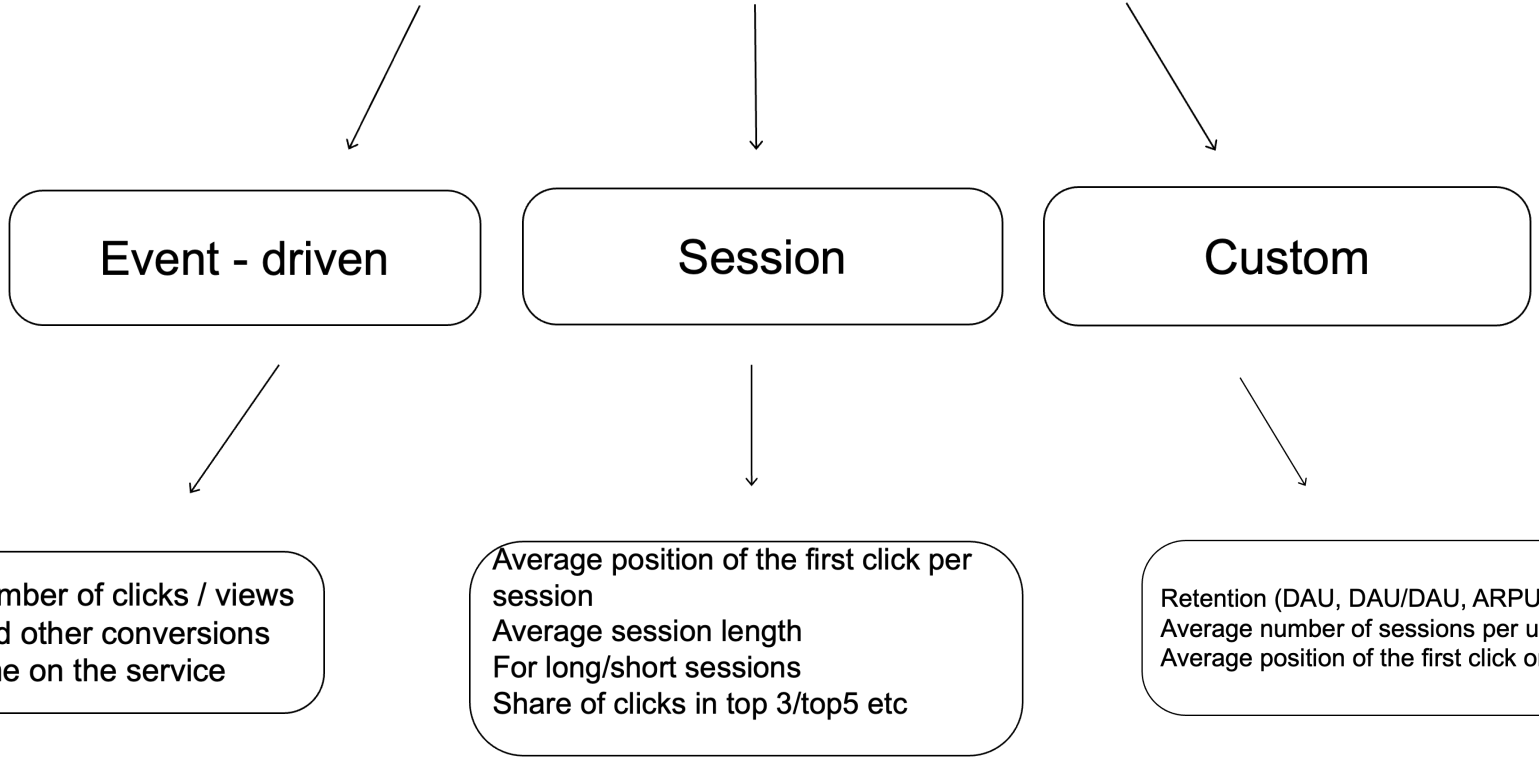
06

Online metrics

- Depend on the product
- Depend on business goals
- Influence the development of the product
- They are calculated on real users (most often in ab tests)
- and are most often the truth of last resort for making a decision about rolling out the model into production

It is critically important to choose the right "instruments" to assess the quality of recommendations

Online metrics



Product metrics: details

It is best to use custom metrics, because most often when we make a change in a product, we want to influence the user's perception of our product.

Event metrics are very dangerous. It is easy to make a mistake in interpreting the result if the metrics are implemented incorrectly. If you are faced with the task of designing an a/b testing platform – be careful

Event/session metrics are strongly influenced by robots / spam /fraud

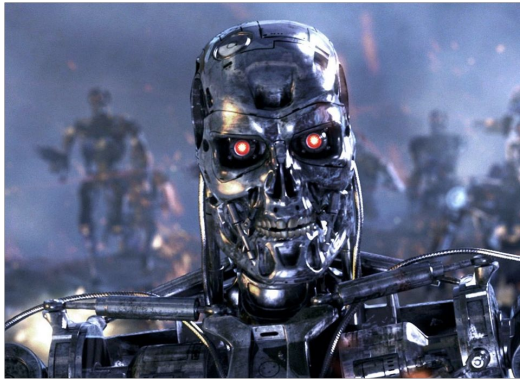
Product metrics: details

Example

Let our metric be the number of calls for an ad.

Run the a/b test, divide the sample into 2 splits. We run different recommendation algorithms on splits.

We see that one model gives the growth of the metric. However, robots could have screwed up this metric and, perhaps, the result of the split is incorrect. What to do?



Product metrics: details

Example

Let our metric be the number of calls for an ad.

Run the a/b test, divide the sample into 2 splits. We run different recommendation algorithms on splits.

We see that one model gives the growth of the metric. However, robots could have screwed up this metric and, perhaps, the result of the split is incorrect. What to do?

It is necessary to beat users by N packages within the split, calculate the average for them and compare with another split.

It turns out that we compare N honest random variables with similar random variables in another split, Such a trick will minimize the influence of robots on the metric value.

Product metrics is all you need?

Focusing strictly on product metrics does not always lead to good results. It is important to measure the quality of service and the "happiness of users" so that a product based on a recommendation system has the potential for long-term growth.

In other words, product metrics can approximate the quality of the service, but they do it poorly for long-term periods, you need to add other metrics to compensate for this myopia.

Special properties

girafe
ai

07

Special properties

- Completeness of documents (coverage)
- Novelty (novelty)
- Diversity / personalization
- Wow effect (serendipity)

Completeness (Coverage)

Share of recommended objects I_P among all objects I

$$Coverage = \frac{|I_P|}{|I|}$$

Novelty / diversity

The key idea is that the less popular an object is, the more likely it is that it will be new to the user.

For each recommended object $i \in R$, we calculate the probability that a random user will have it $\mathrm{P}(i) = \frac{m_i}{N}$, , where m_i is how many people were shown the i-th object, and N is the total number of users.

This metric is sometimes called MSI (mean self-information)

$$Novelty_{user} = \frac{1}{R}\sum_{i \in R} -\log(P(i))$$

Diversity

						
Born This Way	*Pink Friday*	*Dangerously in Love*	*Born This Way – The Remix*	*Femme Fatale*	*Can't be Tamed*	*Teenage Dream*
Lady Gaga	Nicki Minaj	Beyoncé	Lady Gaga	Britney Spears	Miley Cyrus	Katy Perry

Diversity

Intra List Similarity (ILS) – we analyze the similarity of embeddings of items within the framework of recommendations.

We minimize the metric to achieve diversity.

$$ILS_{user} = \frac{1}{R} \sum_{i \in R} \sum_{j \in R} sim(i, j)$$

Diversity

Aggregate Diversity is the cumulative number of unique R_u objects that the model recommends to all U users.

$$Diversity_{agg} = \left| \bigcup_{u \in U} R_u \right|$$

Serendipity – creating a WOW effect

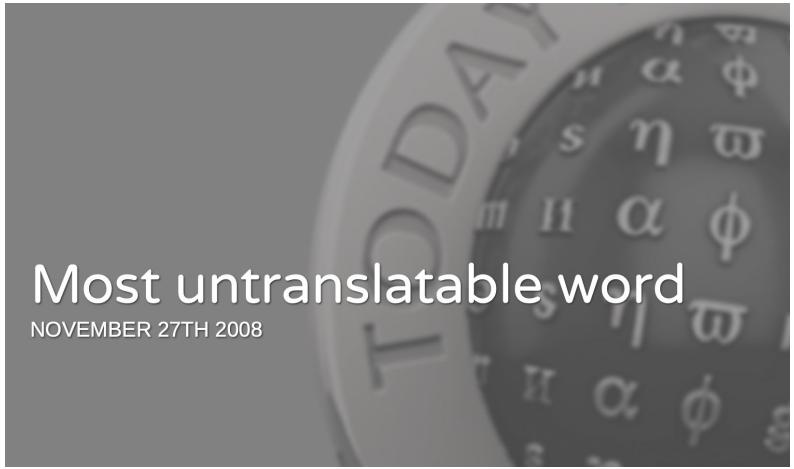
From the point of view of the recommendation system, it is the ability to recommend objects that are not only relevant to the user, but also differ significantly from what objects the user interacted with in the past.

Serendipity is a rather subjective property and it is difficult to formalize it, moreover, such recommendations are rare, which complicates their understanding and the possibility of measurement. There is no consensus on which metric to measure Serendipity

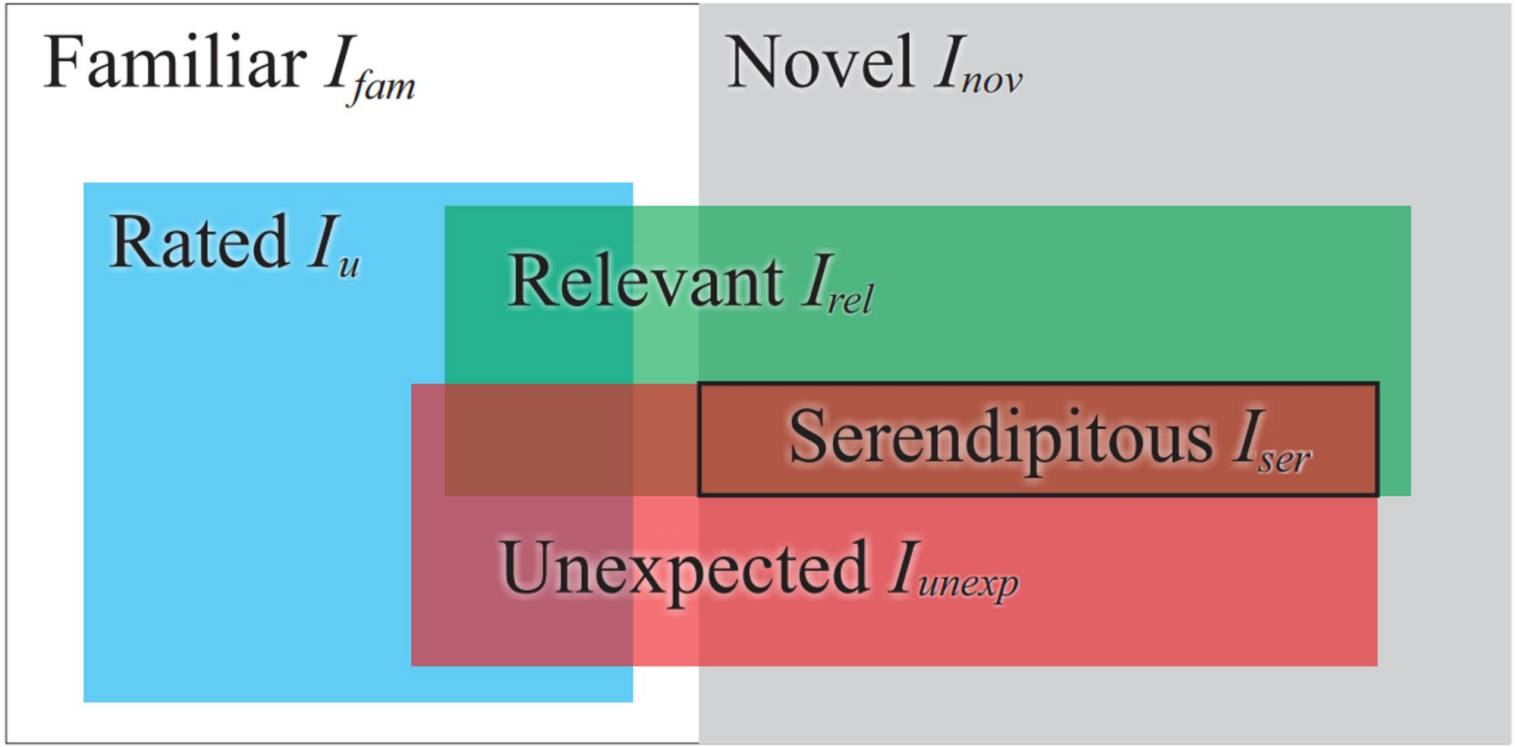
Serendipity - intuitive foresight

Senedipity is a term that entered the top10 words in 2008 with no explicit translation.

It is often translated into Russian as intuitive foresight



Serendipity



Serendipity

Let R_u be a list of recommendations for the user, $Pr_u(i)$ - prediction of the model, for each item from the list, and $Prim_u(i)$ - the prediction of a primitive model, and rel is the known relevance of the item for the user, then Serendipity is calculated as follows:

$$Serendipity_u = \sum_{i \in R_u} \max(Pr_u(i) - Prim_u(i), 0) \cdot rel_u(i)$$

To simplify the formula, the primitive model can be replaced by the predicate of the original model for a random user or by the average predicate of the model for all users

Results

girafe
ai

09

Results

Perfect metrics hardly exist

The key metrics always come from the product

It is important to remember that the growth of some metrics can cause the fall of others (the growth of novelty drops the accuracy, etc.)

Examples of metrics implementation
https://github.com/statisticianinstilettos/recmetrics/blob/master/recmetrics/metrics.py

Learning to rank

Vladislav Goncharenko

Spring 2023, MIPT

Lecture plan

- Setting the task of ranking training
- Evolution of approaches (RankNet, LambdaRank, LambdaMART, YetiRank, ...)
- Adaptation of ranking to boosting
- Overview of batch implementations
- What else to read

2 types of loss functions in recommendations

girafe ai

01

2 types of loss functions in recommendation

- Some are trained to restore the *interaction matrix*, others - to rank

Problem statement

girafe ai

02

Problem statement

Given N queries (queries) $Q = \{q_1, \ldots, q_N\}$.
Each request consists of K documents $D = \{d_1, \ldots, d_K\}$ and their relevance labels $L = \{l_1, \ldots, l_K\}$
Each d_i document is a vector of features of dimension M.

Then the goal of the training is to build a model that will best restore the order of documents in accordance with the true relevance labels of L.

Problem statement

BMW X3 20i xDrive II (F25) — 2012

1 685 000 ₽

от 34 900 ₽ / мес.

2.0 л / 184 л.с. / Бензин — Полный
Автомат — Серый
Внедорожник 5 дв.

Безопасная сделка | Отчёт ПроАвто

Рязань (180 км от Москвы), 40 минут назад

BMW X3 20d xDrive III (G01) — 2018

3 790 000 ₽

от 62 150 ₽ / мес.

2.0 л / 190 л.с. / Дизель — Полный
Автомат — Белый
Внедорожник 5 дв.

Рязань (180 км от Москвы), 4 часа назад

BMW 8 серии 840d xDrive II (G14/G15/G16) — 2018

6 544 000 ₽

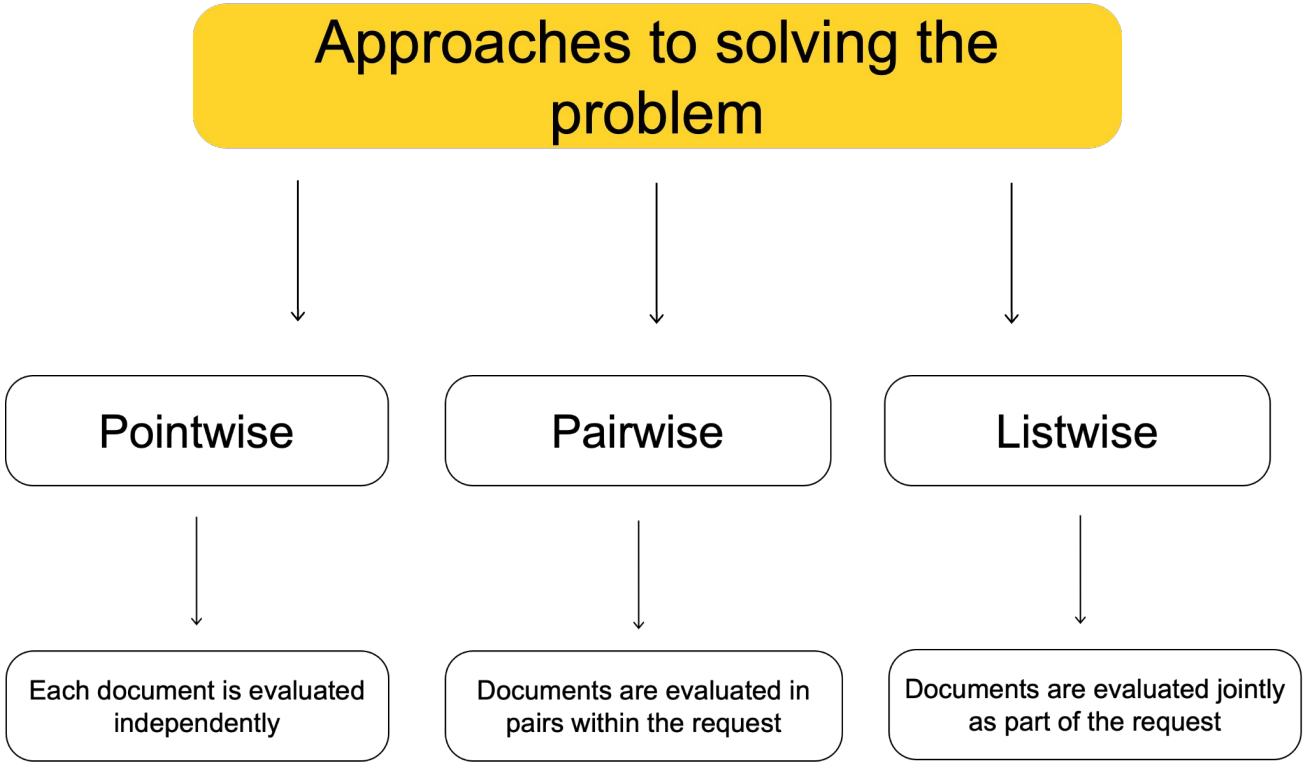
от 62 150 ₽ / мес.

3.0 л / 320 л.с. / Дизель — Полный
Автомат — Синий
Купе

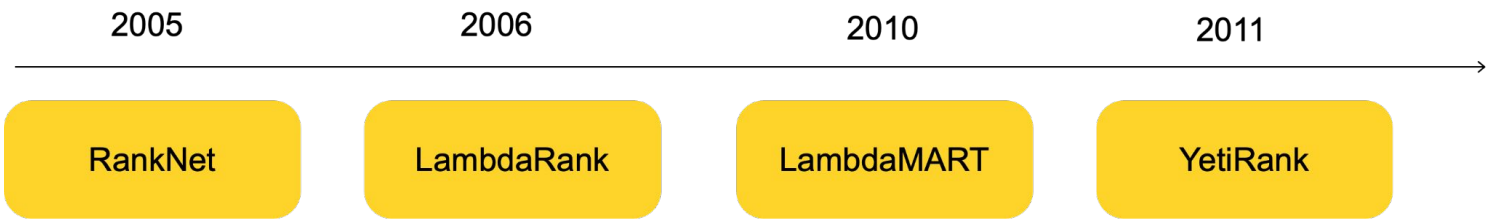
Отчёт ПроАвто

Саларьево, Филатов Луг, Москва, 1 час назад

Problem statement



Evolution of approaches



RankNet

girafe ai

03

RankNet: Ranking training

$$P_{ij} \equiv P(U_i \rhd U_j) \equiv \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

We train the model to predict the relevance of s for a pair of documents s_i, s_j. We substitute the difference of predicted relevancies into the sigmoid function (note that sigma in the denominator is a constant that regulates the scale - wtf MS research??) to predict the probability P_{ij} that the U_i document is more relevant than the U_j document

RankNet: loss function

$$C = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$$

$$\bar{P}_{ij} = \frac{1}{2}(1 + S_{ij})$$

Let S_{ij} be a target equal to one if the ith object is more relevant than the jth, minus one if the opposite is true and zero if the relevancy is equal.

RankNet: loss function and weight update

$$C = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$$

$$w_k \to w_k - \eta \frac{\partial C}{\partial w_k} = w_k - \eta \left(\frac{\partial C}{\partial s_i} \frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j} \frac{\partial s_j}{\partial w_k} \right)$$

RankNet: substituting values into the formula

$$C = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$$

$$C = \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log\left(1 + e^{-\sigma(s_i - s_j)}\right)$$

RankNet: the magic of differentiation

$$C = \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log\left(1 + e^{-\sigma(s_i - s_j)}\right)$$

Statement: the losses are symmetrical. The value of the derivative of the i-th aitem will be exactly equal to the value of the derivative of the j-th item with a minus sign. This property will be useful to us.

$$\frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C}{\partial s_j}$$

RankNet: Factorization

$$\frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C}{\partial s_j}$$

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial w_k} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

$$= \lambda_{ij}\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

We decompose the formula into the product of 2 components. This will speed up the training of the model and achieve almost linear complexity relative to the quadratic one before factorization.

Here we also introduce the concept of Lambda vectors λ_{ij}, which are gradients according to the predictions of the model for the pair i, j.

RankNet: due to what acceleration of calculations?

$$\delta w_k = -\eta \sum_{\{i,j\} \in I} \left(\lambda_{ij} \frac{\partial s_i}{\partial w_k} - \lambda_{ij} \frac{\partial s_j}{\partial w_k} \right) \equiv -\eta \sum_i \lambda_i \frac{\partial s_i}{\partial w_k}$$

The lambda gradient vector λ_i for document i is aggregated over all pairs (i,j) that can be formed in this query with the participation of the i-th document (where the relevance for the i-th is higher)

$$\lambda_i = \sum_{j:\{i,j\} \in I} \lambda_{ij} - \sum_{j:\{j,i\} \in I} \lambda_{ij}$$

Weights are now updated once per request, whereas previously they were updated for each pair of documents in the request. **Computational complexity is now linear in the number of documents inside the query, whereas it used to be quadratic.**

RankNet: the physical meaning of lambda vectors

Lambda gradient vector is a vector calculated for each document. The direction of the vector indicates whether the document needs to be moved up or down in relevance in order to improve the ranking.
The length of the vector is how much to move.

LambdaRank

girafe
ai

04

LambdaRank: optimize wisely

What are the disadvantages of RankNet? Let's take an example:

Suppose the model works almost perfectly, but it made 2 mistakes on the training sample for a given search result consisting of 99 documents:

1. Confuses the ranks of the 1st and 99th documents
2. Confuses the ranks of 97 and 98 documents

Which of these mistakes is more important to correct? **First**

Is the price of these errors the same in terms of the loss function? **Yes, although it is obvious that their price is different.**

Why is that? **Because RankNet optimizes the number of pairwise inversions in the ranked documents. Regardless of their positions in the output.**

What to do? **Optimize directly some good ranking metric... for example, NDCG**

LambdaRank: the key idea

LambdaRank uses the same lambda gradients as RankNet, only multiplies them by changing the DCG metric when rearranging the i-th and j-th document in places. The authors proved that in this case **NDCG is optimized directly!**

$$\lambda_{ij} = \frac{\partial C(s_i - s_j)}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} |\Delta_{NDCG}|$$

Moreover, it is true that it is possible to directly optimize an arbitrary ranking metric, be it MAP, MR, etc.

LambdaRank: again about lambda vectors

The gradients of RankNet are highlighted in black, LambdaRank is highlighted in red.

LambdaRank divides documents so as to maximize ndcg, while the gradients of the RankNet grow so as to minimize the number of pairwise ranking errors

LambdaMART

girafe ai

05

LambdaMART: adaptation to boosting

LambdaMART = LambdaRank + MART

MART (Multiple Additive Regression Trees) – a family of boosting algorithms that can solve any class of problems (regression, classification, ranking). MART models the same lambda gradients, so it's easy to adapt it for ranking training.

YetiRank

girafe ai

06

YetiRank

Intuition:

1) Not all document pairs are equally useful. You need to weigh it.
2) The most important pairs are those that meet in the top of the rating
3) The model's confidence is important that the i-th document is relevant to the j-th. Regularization for uncertain prediction.

$$\mathbb{L} = -\sum_{(i,j)} w_{ij} \log \frac{e^{x_i}}{e^{x_i} + e^{x_j}},$$

where x_i, x_j are the relevance predictions for the given documents, w_{ij} is the weight for the pair i, j

MART is used as a boosting algorithm

Overview of batch implementations

XGBoost (LambdaMART, ...) https://xgboost.readthedocs.io/en/latest/parameter.html

CatBoost (YetiRank, ...)
https://catboost.ai/en/docs/concepts/loss-functions-ranking#pairwise-objectives-and-metrics
Tutorial https://github.com/catboost/catboost/blob/master/catboost/tutorials/ranking/ranking_tutorial.ipynb

LightGBM (LambdaRank, ...) https://lightgbm.readthedocs.io/en/latest/Parameters.html

Realization RankNet, LambdaRank на PyTorch
https://github.com/haowei01/pytorch-examples/tree/master/ranking

Useful materials

1) C.J.C.Burges,T.Shaked,E.Renshaw,A.Lazier,M.Deeds,N.HamiltonandG.Hullender. Learning to Rank using Gradient Descent. Proceedings of the Twenty Second International Conference on Machine Learning, 2005

2) C.J.C. Burges, R. Ragno and Q.V. Le. Learning to Rank with Non-Smooth Cost Functions. Advances in Neural Information Processing Systems, 2006

3) J.H. Friedman. Greedy function approximation: A gradient boosting machine. Technical Report, IMS Reitz Lecture, Stanford, 1999; see also Annals of Statistics, 2001.

4) C.J.C. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Microsoft Research Technical Report MSR-TR-2010-82

5) Gulin, A., Kuralenok, I., Pavlov, D.: Winning the transfer learning track of Yahoo!'s learning to rank challenge with YetiRank. JMLR: Workshop and Conference Proceedings 14 (2011) 63–76