Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Outline

**Morning program**
- Preliminaries
- Semantic matching
- **Learning to rank**
  - Overview & basics
  - Quick refreshers
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Toolkits
- Entities

**Afternoon program**
- Modeling user behavior
- Generating responses
- Recommender systems
- Industry insights
- Q & A
Learning to rank (LTR)

Definition
"... the task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance.” - Liu [2009]

LTR models represent a rankable item—e.g., a document—given some context—e.g., a user-issued query—as a numerical vector $\vec{x} \in \mathbb{R}^n$.

The ranking model $f : \vec{x} \to \mathbb{R}$ is trained to map the vector to a real-valued score such that relevant items are scored higher.

We only discuss offline LTR models here—see Grotov and de Rijke [2016] for an overview of online LTR.
Liu [2009] categorizes different LTR approaches based on training objectives:

- **Pointwise approach**: relevance label $y_{q,d}$ is a number—derived from binary or graded human judgments or implicit user feedback (e.g., CTR). Typically, a regression or classification model is trained to predict $y_{q,d}$ given $\tilde{x}_{q,d}$.

- **Pairwise approach**: pairwise preference between documents for a query $(d_i \succ_q d_j)$ as label. Reduces to binary classification to predict more relevant document.

- **Listwise approach**: directly optimize for rank-based metric, such as NDCG—difficult because these metrics are often not differentiable w.r.t. model parameters.
Features

Traditional LTR models employ hand-crafted features that encode IR insights. They can often be categorized as:

- **Query-independent or static** features (e.g., incoming link count and document length)
- **Query-dependent or dynamic** features (e.g., BM25)
- **Query-level** features (e.g., query length)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
    Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
A quick refresher - Neural models for different tasks

- **Regression**
  - Model: \( \text{features}_{\text{item}} \)
  - Loss: expected vs predicted

- **Classification**
  - Model: \( \text{features}_{\text{item, class 1}} \)
  - Model: \( \text{features}_{\text{item, class 2}} \)

A quick refresher - What is the Softmax function?

In neural classification models, the softmax function is popularly used to normalize the neural network output scores across all the classes

\[
p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \quad (\gamma \text{ is a constant})
\]
A quick refresher - What is Cross Entropy?

The cross entropy between two probability distributions $p$ and $q$ over a discrete set of events is given by,

$$ CE(p, q) = - \sum_i p_i \log(q_i) $$

(2)

If $p_{\text{correct}} = 1$ and $p_i = 0$ for all other values of $i$ then,

$$ CE(p, q) = - \log(q_{\text{correct}}) $$

(3)
Cross entropy with softmax is a popular loss function for classification

$$\mathcal{L}_{CE} = -\log\left(\frac{e^{\gamma z_{correct}}}{\sum_{z \in Z} e^{\gamma z}}\right)$$ (4)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
    Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Learning to rank

Pointwise objectives

Regression-based or classification-based approaches are popular

**Regression loss**

Given \( \langle q, d \rangle \) predict the value of \( y_{q,d} \)

E.g., square loss for binary or categorical labels,

\[
L_{\text{Squared}} = \|y_{q,d} - f(\overrightarrow{x}_{q,d})\|^2
\]  

where, \( y_{q,d} \) is the one-hot representation [Fuhr, 1989] or the actual value [Cossock and Zhang, 2006] of the label
Pointwise objectives

Regression-based or classification-based approaches are popular

Classification loss

Given \(\langle q, d \rangle\) predict the class \(y_{q,d}\)

E.g., Cross-Entropy with Softmax over categorical labels \(Y\) [Li et al., 2008],

\[
\mathcal{L}_{CE}(q, d, y_{q,d}) = -\log \left( p(y_{q,d}|q, d) \right) = -\log \left( \frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}} \right)
\]

where, \(s_{y_{q,d}}\) is the model’s score for label \(y_{q,d}\)
Outline

Morning program
  Preliminaries
  Semantic matching
  **Learning to rank**
    - Overview & basics
    - Quick refreshers
    - Pointwise loss
    - **Pairwise loss**
    - Listwise loss
    - Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Pairwise objectives

Pairwise loss minimizes the average number of inversions in ranking—i.e., $d_i \succ_q d_j$ but $d_j$ is ranked higher than $d_i$

Given $\langle q, d_i, d_j \rangle$, predict the more relevant document

For $\langle q, d_i \rangle$ and $\langle q, d_j \rangle$,
- Feature vectors: $\vec{x}_i$ and $\vec{x}_j$
- Model scores: $s_i = f(\vec{x}_i)$ and $s_j = f(\vec{x}_j)$

Pairwise loss generally has the following form [Chen et al., 2009],

$$L_{\text{pairwise}} = \phi(s_i - s_j)$$ (8)

where, $\phi$ can be,
- Hinge function $\phi(z) = \max(0, 1 - z)$ [Herbrich et al., 2000]
- Exponential function $\phi(z) = e^{-z}$ [Freund et al., 2003]
- Logistic function $\phi(z) = \log(1 + e^{-z})$ [Burges et al., 2005]
- etc.
RankNet

RankNet [Burges et al., 2005] is a pairwise loss function—an industry favourite [Burges, 2015]

Predicted probabilities:
\[ p_{ij} = p(s_i > s_j) \equiv \frac{e^{\gamma s_i}}{e^{\gamma s_i} + e^{\gamma s_j}} = \frac{1}{1 + e^{-\gamma(s_i-s_j)}} \]

and \[ p_{ji} \equiv \frac{1}{1 + e^{-\gamma(s_j-s_i)}} \]

Desired probabilities: \( \bar{p}_{ij} = 1 \) and \( \bar{p}_{ji} = 0 \)

Computing cross-entropy between \( \bar{p} \) and \( p \),

\[ \mathcal{L}_{RankNet} = -\bar{p}_{ij} \log(p_{ij}) - \bar{p}_{ji} \log(p_{ji}) \quad (9) \]

\[ = - \log(p_{ij}) \quad (10) \]

\[ = \log(1 + e^{-\gamma(s_i-s_j)}) \quad (11) \]
Cross Entropy (CE) with Softmax over documents

An alternative loss function assumes a single relevant document $d^+$ and compares it against the full collection $D$

Probability of retrieving $d^+$ for $q$ is given by the softmax function,

$$p(d^+ | q) = \frac{e^{\gamma \cdot s(q, d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}}$$  \hspace{1cm} (12)

The cross entropy loss is then given by,

$$\mathcal{L}_{CE}(q, d^+, D) = - \log \left( p(d^+ | q) \right)$$  \hspace{1cm} (13)

$$= - \log \left( \frac{e^{\gamma \cdot s(q, d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}} \right)$$  \hspace{1cm} (14)
If we consider only a pair of relevant and non-relevant documents in the denominator, CE reduces to RankNet.

Computing the denominator is prohibitively expensive—large body of work in NLP on this that may be relevant to future LTR models:
- Hierarchical softmax
- Sampling based approaches

In IR, LTR models typically consider few negative candidates [Huang et al., 2013, Mitra et al., 2017, Shen et al., 2014]
Avoid computing $p(d^+|q)$, group candidates $D$ into set of classes $C$, then predict correct class $c^+$ given $q$ followed by predicting $d^+$ given $\langle c^+, q \rangle$ [Goodman, 2001]

$$p(d^+|q) = p(d^+|c^+, q) \cdot p(c^+|q)$$

(15)

Computational cost is a function of $|C| + |c^+| << |D|$.

Employ hierarchy of classes [Mnih and Hinton, 2009, Morin and Bengio, 2005]

Hierarchy based on similarity between candidates [Brown et al., 1992, Le et al., 2011, Mikolov et al., 2013], or frequency binning [Mikolov et al., 2011]
Sampling based approaches

Alternative to computing exact softmax, estimate it using sampling based approaches

\[
\mathcal{L}_{CE}(q, d^+, D) = -\log \left( \frac{e^{\gamma \cdot s(q, d^+)} \sum_{d \in D} e^{\gamma \cdot s(q, d)}}{e^{\gamma \cdot s(q, d^+)} \sum_{d \in D} e^{\gamma \cdot s(q, d)}} \right) = -\gamma \cdot s(q, d^+) + \log \sum_{d \in D} e^{\gamma \cdot s(q, d)} \tag{16}
\]

Importance sampling [Bengio and Senécal, 2008, Bengio et al., 2003, Jean et al., 2014, Jozefowicz et al., 2016], Noise Contrastive Estimation [Gutmann and Hyvärinen, 2010, Mnih and Teh, 2012, Vaswani et al., 2013], negative sampling [Mikolov et al., 2013], BlackOut [Ji et al., 2015], and others have been proposed

See [Mitra and Craswell, 2017] for detailed discussion
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
  Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Listwise

**Blue**: relevant  **Gray**: non-relevant

NDCG and ERR higher for left but pairwise errors less for right

Due to strong position-based discounting in IR measures, errors at higher ranks are much more problematic than at lower ranks

But listwise metrics are non-continuous and non-differentiable

[Burges, 2010]
LambdaRank

Key observations:
- To train a model we don't need the costs themselves, only the gradients \(\text{of the costs w.r.t model scores}\).
- It is desired that the gradient be bigger for pairs of documents that produces a bigger impact in NDCG by swapping positions.

**LambdaRank** [Burges et al., 2006]
Multiply actual gradients with the change in NDCG by swapping the rank positions of the two documents.

\[
\lambda_{\text{LambdaRank}} = \lambda_{\text{RankNet}} \cdot |\Delta NDCG| \tag{17}
\]
ListNet and ListMLE

According to the Luce model [Luce, 2005], given four items \( \{d_1, d_2, d_3, d_4\} \) the probability of observing a particular rank-order, say \( [d_2, d_1, d_4, d_3] \), is given by:

\[
p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)}
\]

(18)

where, \( \pi \) is a particular permutation and \( \phi \) is a transformation (e.g., linear, exponential, or sigmoid) over the score \( s_i \) corresponding to item \( d_i \)
ListNet and ListMLE

**ListNet** [Cao et al., 2007]  
Compute the probability distribution over all possible permutations based on model score and ground-truth labels. The loss is then given by the K-L divergence between these two distributions.

This is computationally very costly, computing permutations of only the top-K items makes it slightly less prohibitive.

**ListMLE** [Xia et al., 2008]  
Compute the probability of the ideal permutation based on the ground truth. However, with categorical labels more than one permutation is possible which makes this difficult.
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
    Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Toolkits for off-line learning to rank

- RankLib: https://sourceforge.net/p/lemur/wiki/RankLib
- shoelace: https://github.com/rjagerman/shoelace [Jagerman et al., 2017]
- QuickRank: http://quickrank.isti.cnr.it [Capannini et al., 2016]
- RankPy: https://bitbucket.org/tunystom/rankpy
- pyltr: https://github.com/jma127/pyltr
- jforests: https://github.com/yasserg/jforests [Ganjisaffar et al., 2011]
- XGBoost: https://github.com/dmlc/xgboost [Chen and Guestrin, 2016]
- SVMRank: https://www.cs.cornell.edu/people/tj/svm_light [Joachims, 2006]
- sofia-ml: https://code.google.com/archive/p/sofia-ml [Sculley, 2009]
- pysofia: https://pypi.python.org/pypi/pysofia