Lecture 16: Supervised Methods, Neural Networks and Learning to Rank

IR Spring 22

Instructor: Prof. Somak Aditya

Slides Courtesy: Learning to Rank Tutorial

Categorization/Classification

Given:

- A representation of a document d
  - Issue: how to represent text documents.
  - Usually some type of high-dimensional space – bag of words, vector space
- A fixed set of classes: $C = \{c_1, c_2, \ldots, c_J\}$

Determine:

- The category of d: $\gamma(d) \in C$, where $\gamma(d)$ is a classification function
- We want to build classification functions ("classifiers").

Classification Examples

- Classify an email as spam / legitimate
- Classify a news article as per its topic (Politics / Sports / Entertainment / ..)

Classification Methods

- Manual Classification

pol.
scienu
arts

- Rule-based Classification

q=? d=✓

- Given: A document d, A fixed set of classes: C = {c1, c2,…, cJ}
  - A training set D of documents each with a label in C
  - Determine: A learning method or algorithm which will enable us to learn a classifier γ
  - For a test document d, we assign it the class γ(d) ∈ C

$\sigma(\theta^T x)$

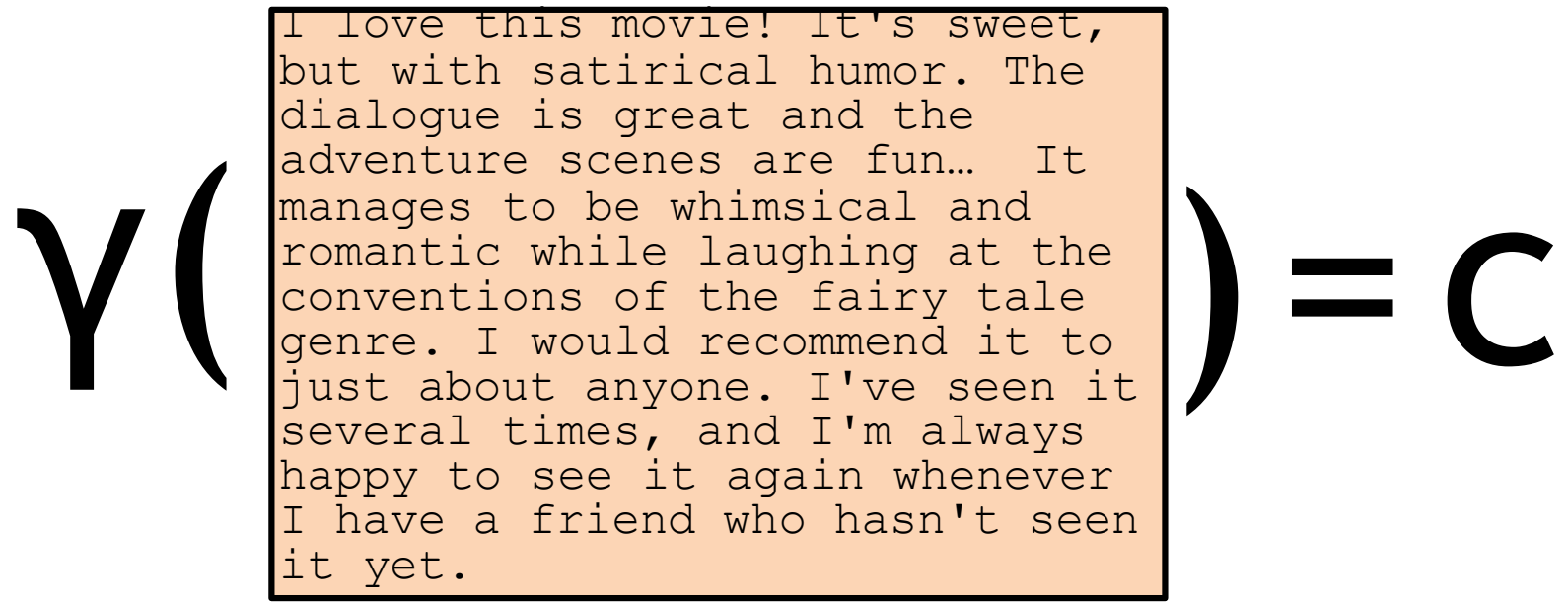
Classification Methods

- Supervised learning
  - Naive Bayes (simple, common)
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (generally more powerful)
  - Decision trees → random forests → gradient-boosted decision trees (e.g., xgboost)
  - Neural models
  - …
- No free lunch: need hand-classified training data
- Many commercial systems still use some of them, but mostly NN/DL dominates.

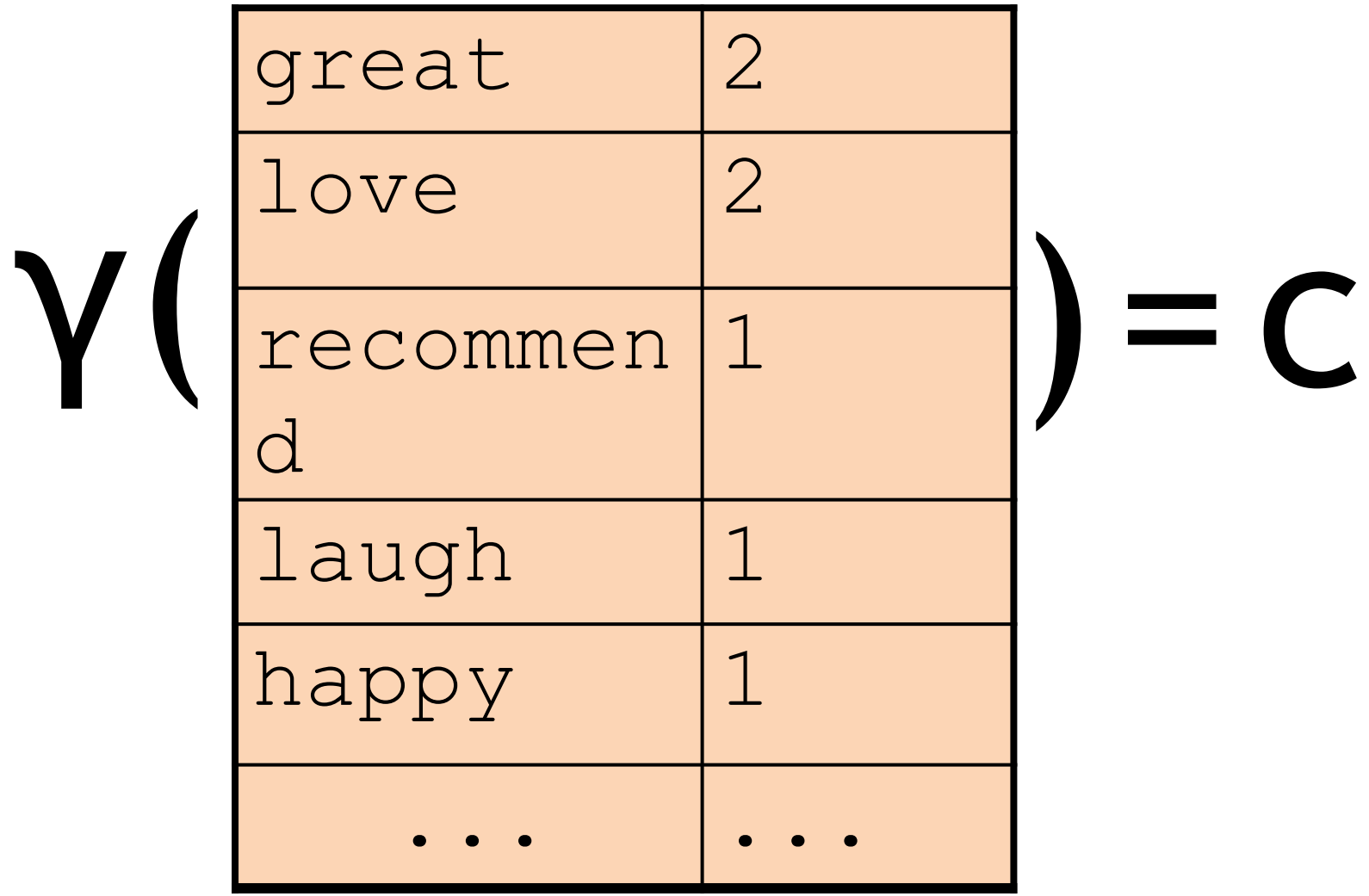
How to Classify?

- Three aspects
  - Input Representation (features – handcrafted, now we learn them)
  - Function representation (hypothesis space)
  - Output classes
- Supervised learning classifiers can use any sort of feature
  - URL, email address, punctuation, capitalization, dictionaries, network features
- In the simplest bag of words view of documents
  - We use only word features

The bag of words representation



The bag of words representation



Remember: Vector Space Representation

- Each document is a vector, one component for each term (= word).
- Normalize vectors to unit length.
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space

- How can we do classification in this space?

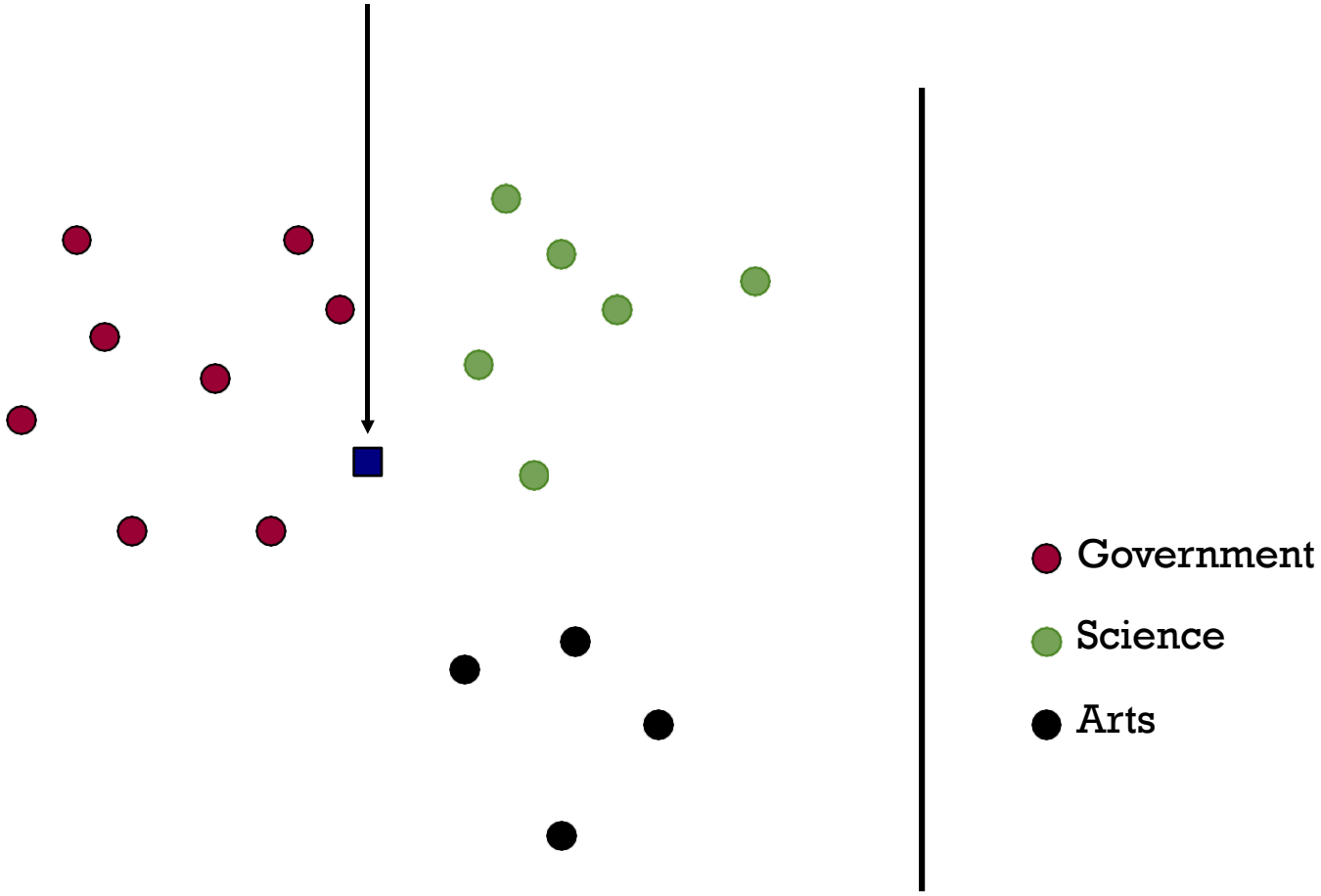
Classification Using Vector Spaces

- In vector space classification, training set corresponds to a labeled set of points (equivalently, vectors)

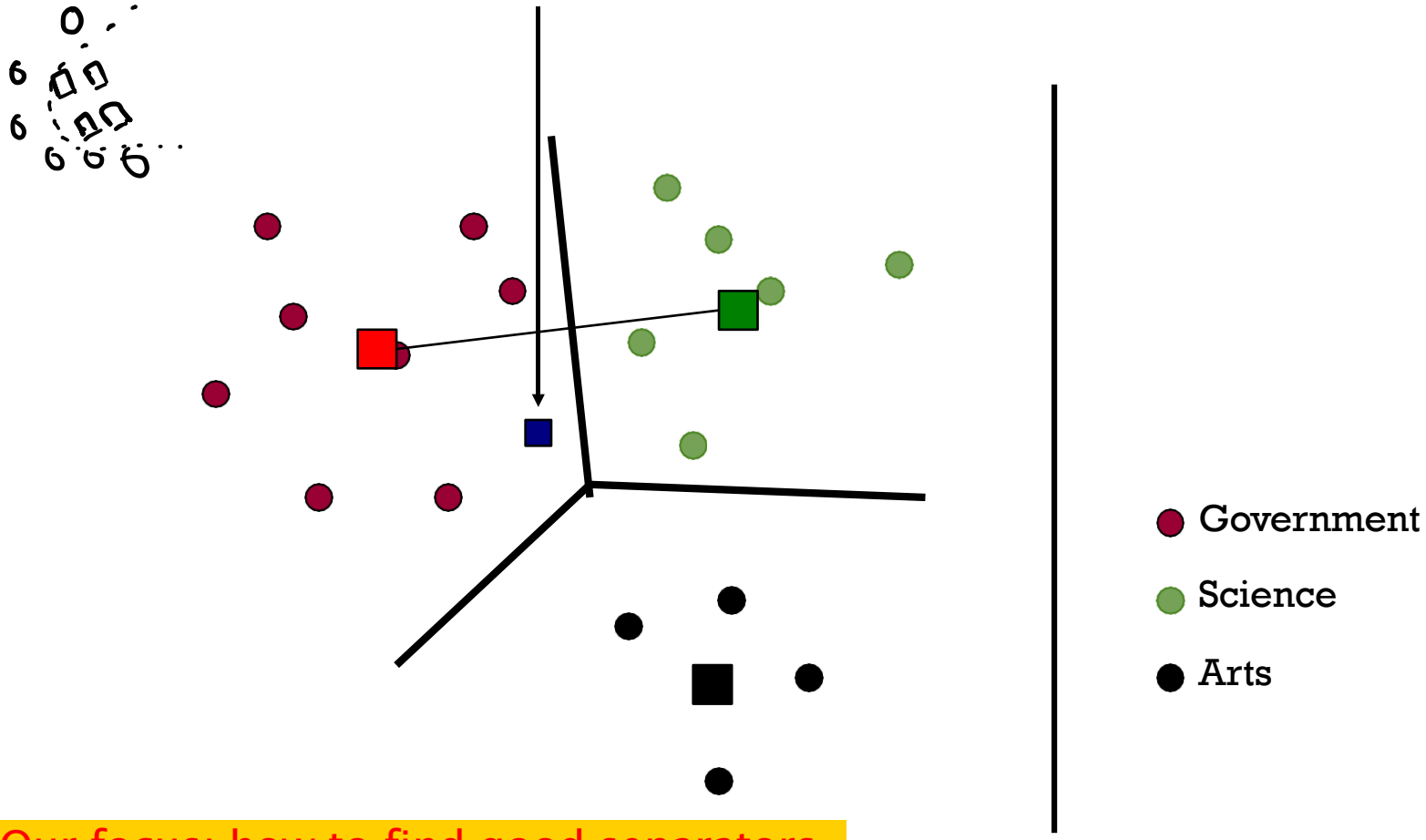
- Premise 1: Documents in the same class form a contiguous region of space
- Premise 2: Documents from different classes don't overlap (much)

- Learning a classifier: build surfaces to delineate classes in the space

Test Document of what class?



Test Document = Government



Our focus: how to find good separators

Another supervised learning problem: Regression

- Instead of mapping instances (e.g., documents) to discrete labels/classes …
- Map instances to a continuous (real) value
- E.g.,
  - Given the features (e.g., height, food habit, occupation, ~~weight~~) of many individuals, predict the weight of a new person for whom the other features are known
  - Given the bag of words representation of an email, predict the probability of it being spam

$x^2 + y^2 + 1 = 0$

$f(x) = y \quad y \in \mathbb{R}$ 0/1 0.8

SUPERVISED METHODS FOR RETRIEVAL

Machine learning for IR ranking?

- We've looked at methods for ranking documents in IR
  - Cosine similarity, inverse document frequency, BM25, proximity, pivoted document length normalization, Pagerank, …
- We've discussed supervised learning problems – classification and regression

I: q, d , O: 0/1

0-1 =

- Can we can use *machine learning* to rank the documents displayed in search results?
  - Sounds like a good idea
  - Known as "machine-learned relevance" or "learning to rank"
  - Actively researched and used by Web search engines

Simple example: Using classification for ad hoc IR

I, O, [objective]
↓
$\vec{x}$

Collect a training corpus of (*q*, *d*, *r*) triples

- Relevance *r* is here binary (but may be multiclass, with 3–7 values)
- Query-Document pair is represented by a feature vector
- Train a machine learning model to predict the class *r* of a document-query pair

$I \rightarrow \vec{x}$?
q, d
VSM
$\underline{f}(x) = y$

|V| |V|
2|V|
$f(q, d_1) = 0.8$

brut ∨ ceaser
Doc : . . . . .
Label : 1
$f(q, d_2) = 0.8$

“Learning to rank”

- Classification probably isn’t the right way to think about approaching ad hoc IR:
  - Classification problems: Map to an unordered set of classes
  - Regression problems: Map to a real value
  - Ordinal regression (or “ranking”) problems: Map to an *ordered* set of classes

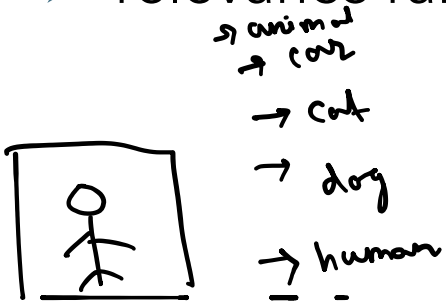
“Learning to rank”

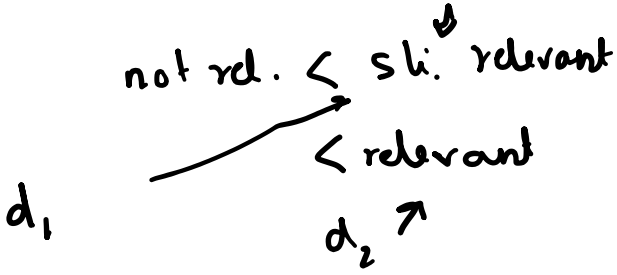
- Assume a number of categories **C** of relevance exist
  - These are totally ordered: $c_1 < c_2 < \ldots < c_J$
  - This is the ordinal regression setup

0, 1, 2

0 < 1 < 2

- Assume training data is available consisting of document
  - query pairs (d, q) represented as feature vectors x_i with
  - relevance ranking c_i





LEARNING TO RANK

Learning to rank (L2R)

Defnition

- "... 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]

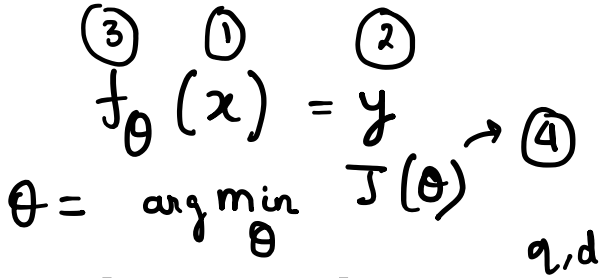
L2R models represent

- 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: $\vec{x} \rightarrow \mathbb{R}$ is **trained** to map the vector to a real-valued score such that relevant items are scored higher

Approaches



Based on training objectives [Liu 2009]:

$q, d, y_{q,d} = 0.7$
0/1
0/1/2

- Pointwise approach: Relevance label $y_{q,d}$ is a number
  - Supervision: binary or graded human judgments or implicit user feedback (e.g., CTR).
  - Classification/regression to predict $y_{q,d}$, given $\vec{x}_{q,d}$.

- Pairwise approach: pairwise preference between documents for a query ($d_i \succ_q d_j$) as label.

q, d_1, d_2 1 0
0 1

  - Supervision: pairwise preference
  - Task: Given $\langle q, d_i, d\_j \rangle$, predict 1 (if d_i is preferred) or 0 otherwise.

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

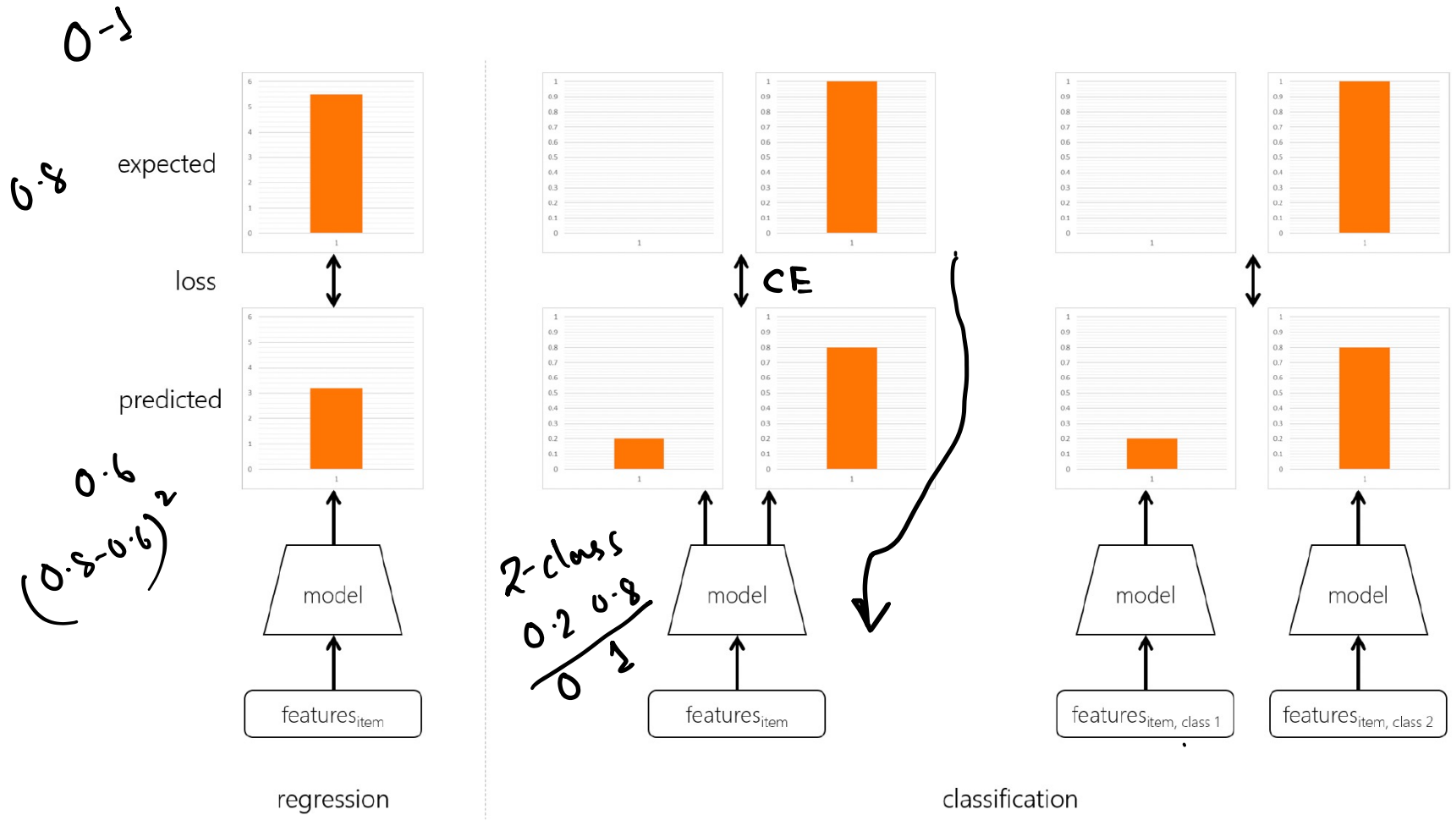
Features

Traditional L2R models employ hand-crafted features

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)

Short Intro: Cross Entropy + NN



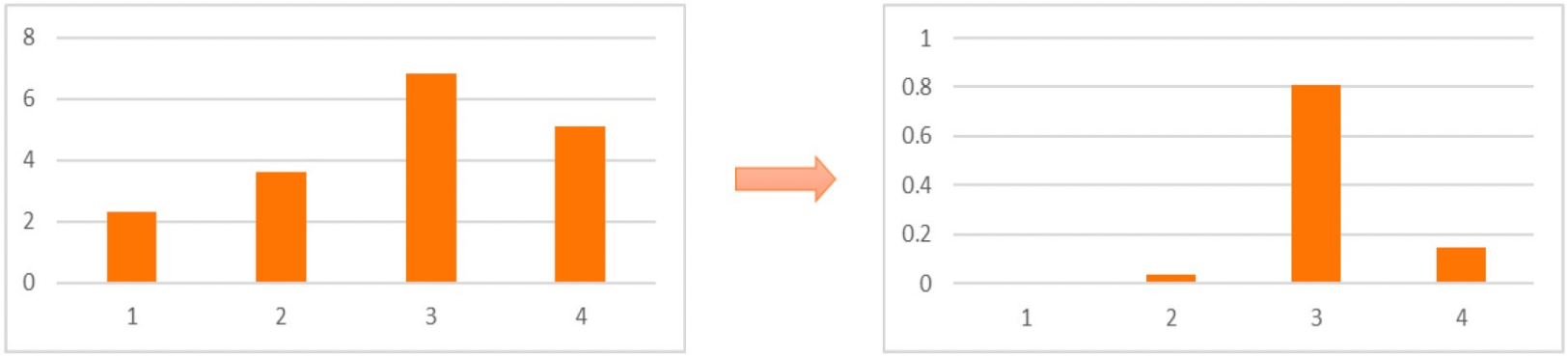
Short Intro : What is softmax

- The softmax function is popularly used to normalize the neural network output scores across all the classes

$h_\theta(x) = y$

$g(\theta^T x) = \begin{bmatrix} 0.7 & 0.5 \\ 0.8 & 0.2 \end{bmatrix}$

$$p(z_i) = \frac{e^{z_i}}{\sum_z e^z}$$

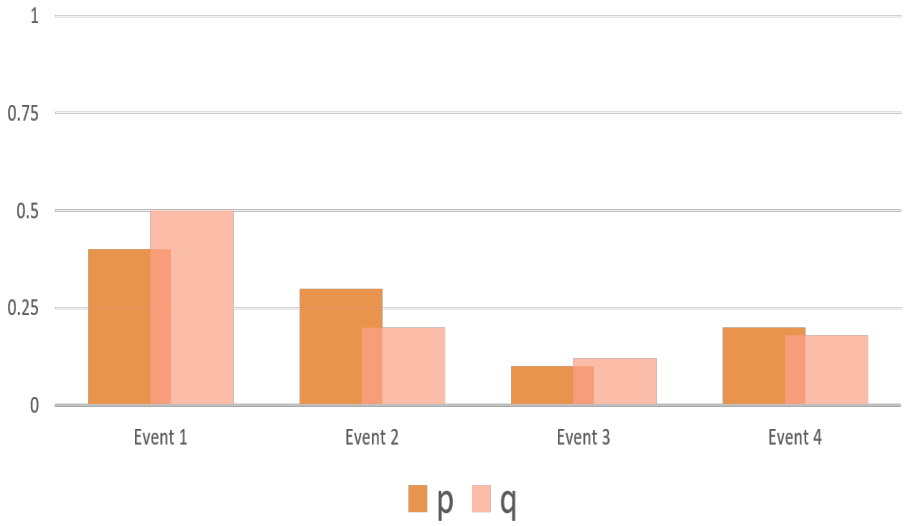


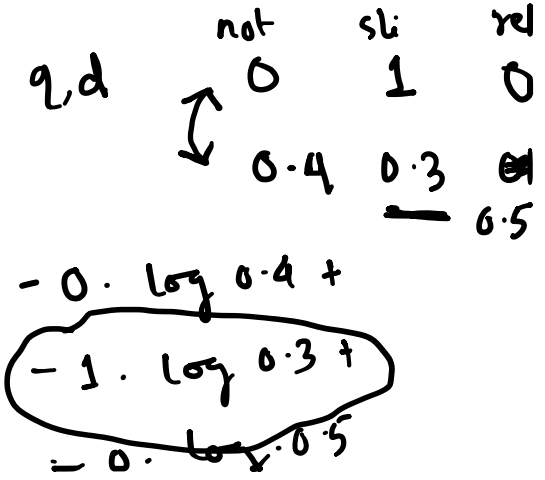
Short Intro: Cross-entropy

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

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

- Single-label classification: $CE(p, q) = -\log(q_{correct})$



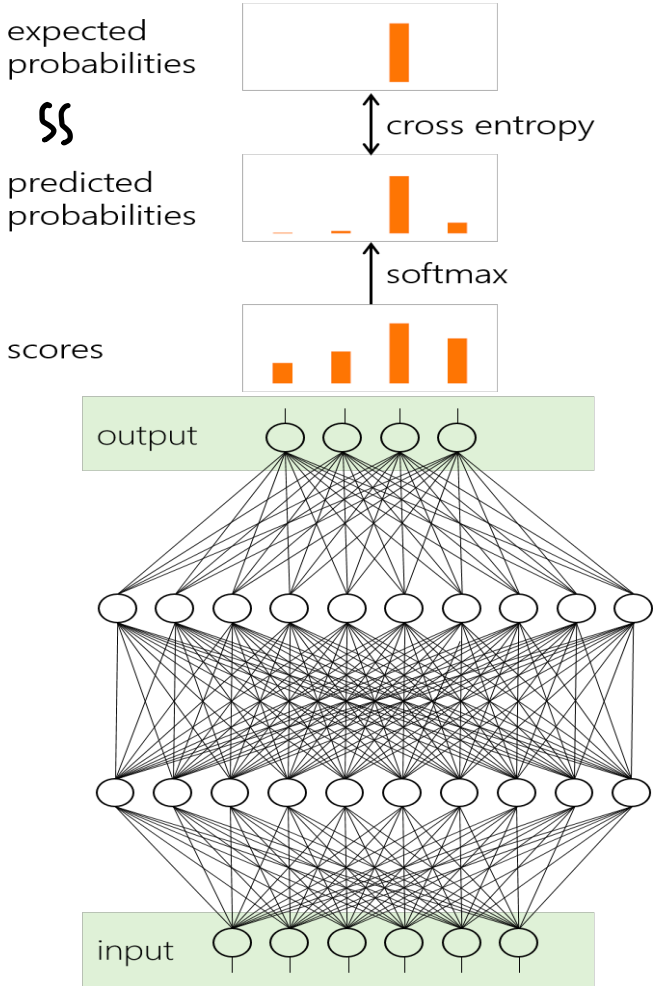


Short Intro: CE with softmax

- Cross entropy with softmax is a popular loss function for classification

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

$J(\theta)$



L2R Loss Functions

PointWise Loss

PairWise Loss

ListWise Approaches

Pointwise Loss

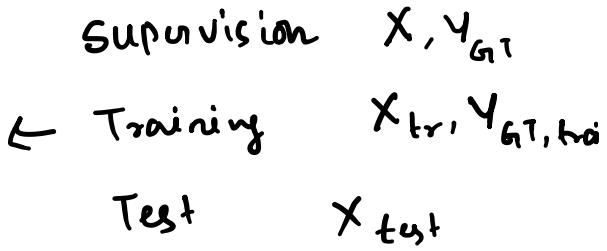
Regression-based or classification-based approaches are popular

- Regression loss
  - Given $\langle q, d \rangle$ predict the value of $y_{q,d}$
  - E.g., square loss for binary or categorical labels,

$$L_{Squared} = \left\| y_{q,d} - f\left(\vec{x}_{q,d}\right) \right\|^2$$

  - where, $y_{q,d}$ is (generally) the actual value of the label

Pointwise Loss



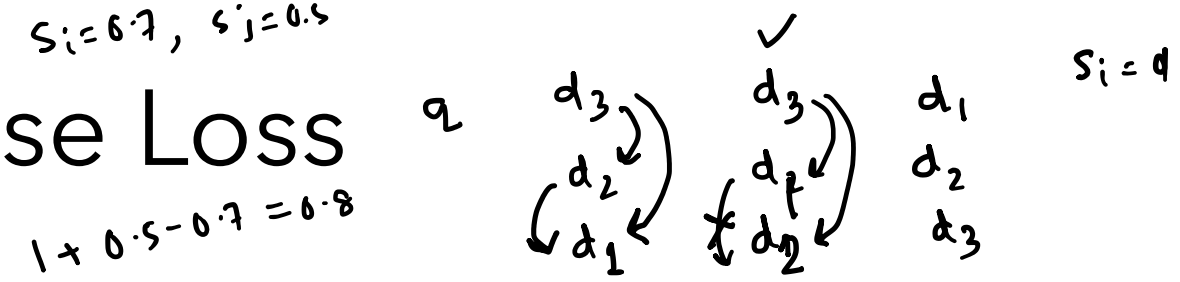
Regression-based or classification-based approaches are popular

- Classification loss
  - Given $\langle q, d \rangle$ predict the value of $y_{q,d}$
  - E.g., Cross-Entropy with Softmax over categorical labels Y,

$$L_{CE}(q, d, y_{q,d}) = -\log\left(p(y_{q,d}|q, d)\right) = -\log\left(\frac{e^{s_{y_{q,d}}}}{\sum_{y \in Y} e^{s_y}}\right)$$

  - where, $s_{y_{q,d}}$ is model's score for label $y_{q,d}$

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

- 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 $s_j = f(\vec{x}_j)$
  - Say, d_i is more relevant. ➔ $s_i > s_j$

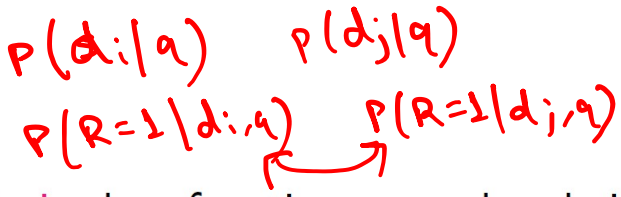
- Pairwise loss generally has the following form[Chen et al., 2009],

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

where, ϕ can be,

- Hinge function $\phi(z) = \max(0;\ 1 - z)$
- Logistic function $\phi(z) = \log(1 + e^{-z})$

RankNet



RankNet [Burges et al., 2005] is a pairwise loss function—popular choice for training neural L2R models and also an industry favourite [Burges, 2015]

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

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,

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

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

$$= log(1 + e^{-\gamma(s_i - s_j)}) \quad (11)$$

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,

$$p(d^+|q) = \frac{e^{\gamma \cdot s\left(q,d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \tag{12}$$

1M x 64

The cross entropy loss is then given by,

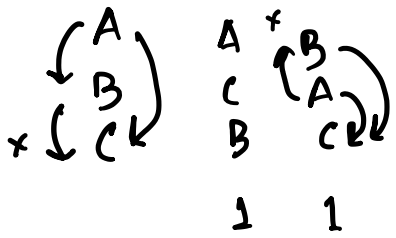
$$\mathcal{L}_{\mathsf{CE}}(q, d^+, D) = -log\Big(p(d^+|q)\Big) \tag{13}$$

$$= -log\Big(\frac{e^{\gamma \cdot s\left(q,d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}}\Big) \tag{14}$$

CE vs RankNet

- 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 -- L2R models typically consider few negative candidates
- Large body of work in NLP to deal with similar issue that may be relevant to future L2R models
  - Importance sampling, negative sampling

ListWise

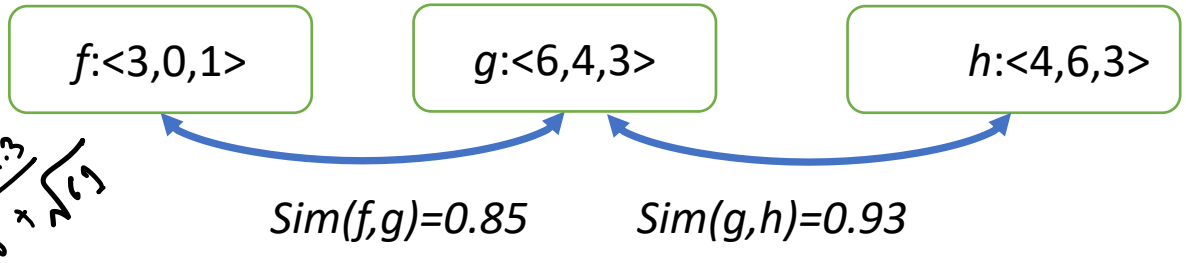


A Simple Example:

- function *f*: *f*(A)=3, *f*(B)=0, *f*(C)=1 ACB
- function *h*: *h*(A)=4, *h*(B)=6, *h*(C)=3 BAC
- **ground truth *g*: *g*(A)=6, *g*(B)=4, *g*(C)=3 ABC**

Question: which function is closer to ground truth?

- Based on pointwise similarity: *sim*(*f*,*g*) < *sim*(*g*,*h*).
- Based on pairwise similarity: *sim*(*f*,*g*) =*sim*(*g*,*h*)
- Based on cosine similarity between score vectors?



- Acc. To position-wise discount f should be closer to g.

Permutation Probability Distribution

ABC A B C
A 0/1 0/1 0/1

A-B
A-C
B-C

- Question:
  - How to represent a ranked list?

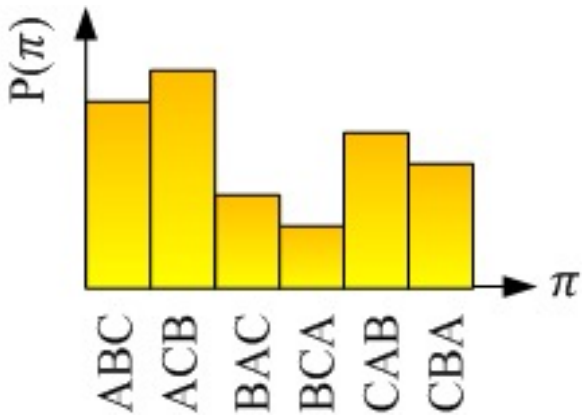
Solution

- Ranked list ←→ Permutation probability distribution
- More informative representation for ranked list: permutation and ranked list has 1-1 correspondence.

1 0 0 0 0 0

f: $f(A)=3, f(B)=0, f(C)=1$;
Ranking by f: ABC

↔



Luce Model: Defining Permutation Probability

- Probability of permutation π is defined as

$$P_s(\pi) = \prod_{j=1}^{n} \frac{\varphi(s_{\pi(j)})}{\sum_{k=j}^{n} \varphi(s_{\pi(k)})}$$

P(ABC) >
P(ACB)

- Example:

$$P_f(\mathrm{ABC}) = \frac{\varphi(f(\mathrm{A}))}{\varphi(f(\mathrm{A}))+\varphi(f(\mathrm{B}))+\varphi(f(\mathrm{A}))} \cdot \frac{\varphi(f(\mathrm{B}))}{\varphi(f(\mathrm{B}))+\varphi(f(\mathrm{C}))} \cdot \frac{\varphi(f(\mathrm{C}))}{\varphi(f(\mathrm{C}))}$$

P(A ranked No.1)

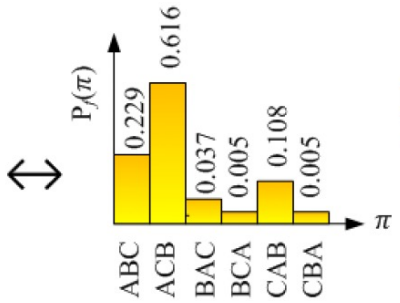
P(B ranked No.2 | A ranked No.1)
= P(B ranked No.1)/(1- P(A ranked No.1))

P(C ranked No.3 | A ranked No.1, B ranked No.2)

Distance between Ranked Lists

$\varphi = exp$

f: $f(\mathrm{A}) = 3, f(\mathrm{B})=0, f(\mathrm{C})=1;$
Ranking by f: ABC

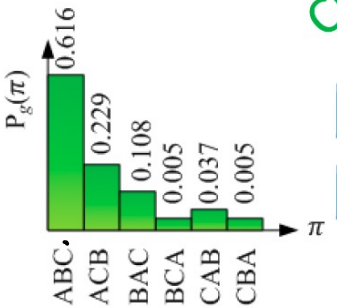


*Using **KL-divergence** to measure difference between distributions*

Closer!

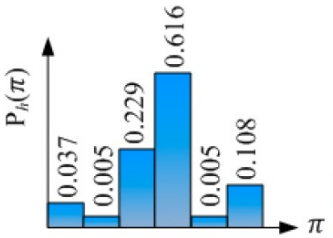
$dis(f,g) = 0.46$

g: $g(\mathrm{A}) = 6, g(\mathrm{B})=4, g(\mathrm{C})=3;$
Ranking by g: ABC



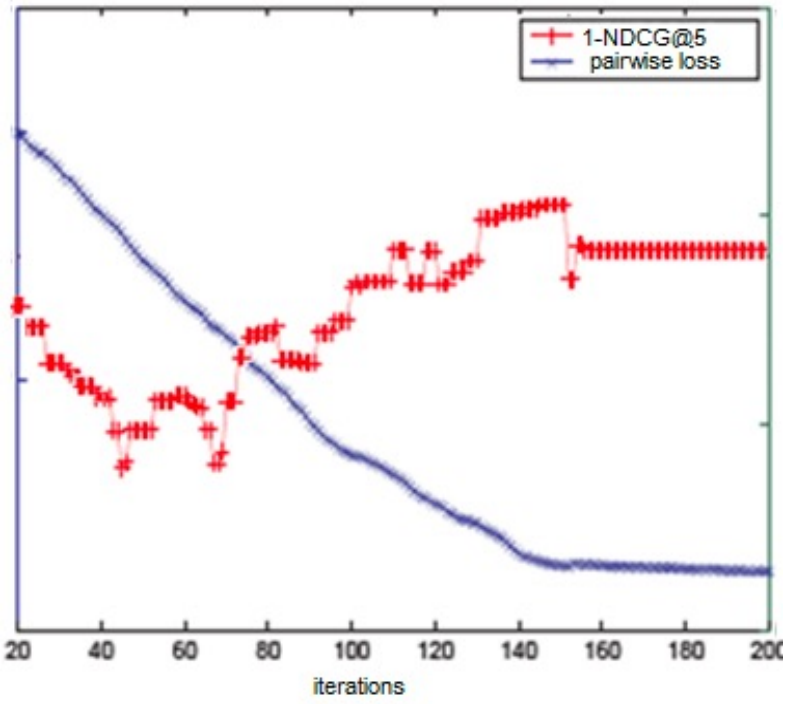
$dis(g,h) = 2.56$

h: $h(\mathrm{A}) = 4, h(\mathrm{B})=6, h(\mathrm{C})=3;$
Ranking by h: ACB

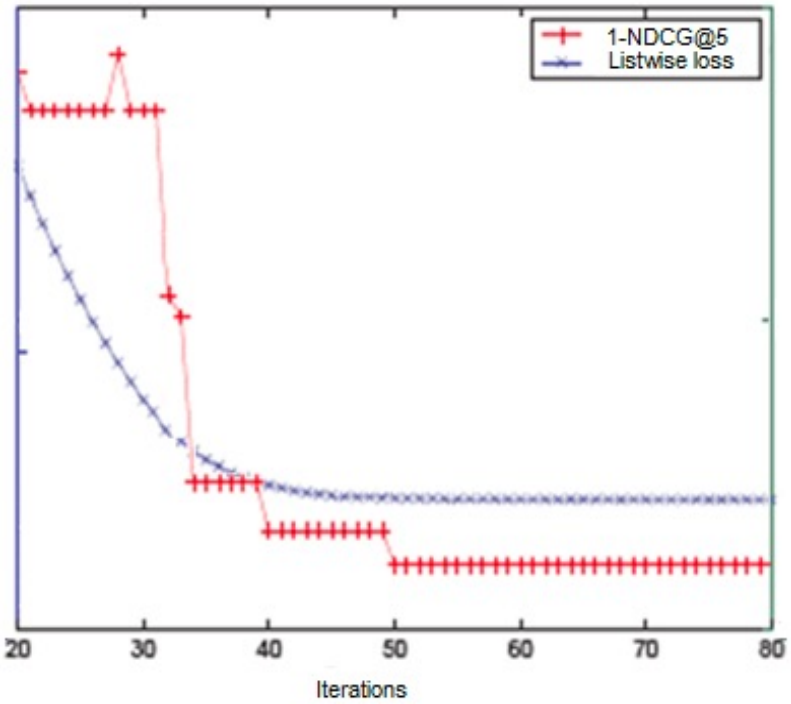


Experimental Results (ListNet)

(Z. Cao, T. Qin, T. Liu, et al. ICML 2007)



Pairwise (RankNet)



ListWise (ListNet)

Training Performance on TD2003 Dataset

ListNet vs 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

ideal MAP

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.

Supervision/Annotations

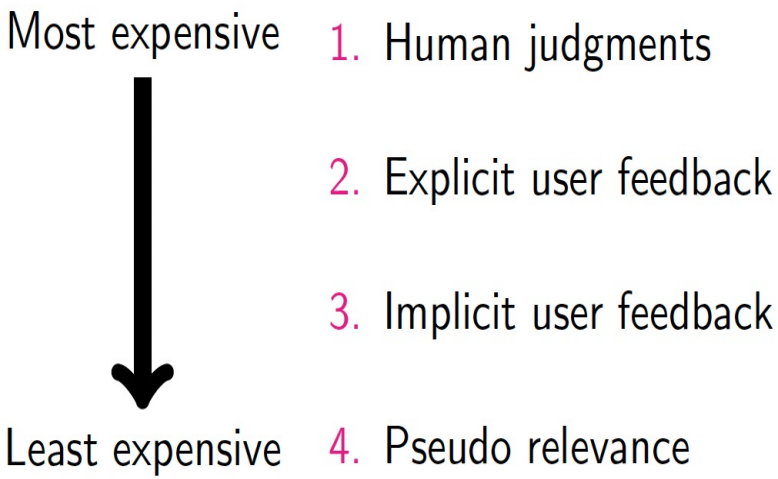
Different Levels of Supervision

Data requirements for training an offline L2R system

- Query/document pairs that encode an ideal ranking given a particular query.
- Ideal ranking? Relevance, preference, importance [Liu, 2009], novelty & diversity [Clarke et al., 2008].
- What about personalization? Triples of user, query and document.
- Related to evaluation. Pairs also used to compute popular offline evaluation measures.
- Graded or binary. "documents may be relevant to a di erent degree"
- Absolute or relative? Zheng et al. [2007]

Satisfying Data-Hungry Models

There are different ways to obtain query/document pairs.



Human Judgements

Human judges determine the relevance of a document for a given query.

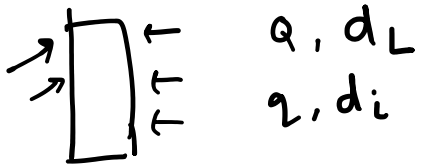
How to determine candidate query/document pairs?

- Obtaining human judgments is expensive.
- List of queries: sample of incoming traffic or manually curated.
- Use an existing rankers to obtain rankings and pool the outputs [Sparck Jones and van Rijsbergen, 1976].
- Trade-off between number of queries (shallow) and judgments (deep) [Yilmaz and Robertson, 2009].

Explicit User Feedback

- When presenting results to the user, ask the user to explicitly judge the documents.
- Unfortunately, users are only rarely willing to give explicit feedback [Joachims et al., 1997].

Extracting pairs from click-through data (training)



Extract implicit judgments from search engine interactions by users.

- Assumption: user clicks ⟹ relevance (or, preference).
- Virtually unlimited data at very low cost, but interpretation is more difficult.
- Presentation bias: users are more likely to click higher-ranked links.
  - How to deal with presentation bias? Joachims [2003] suggest to interleave different rankers and record preference.
- Chains of queries (i.e., search sessions) can be identified within logs and more fine-grained user preference can be extracted [Radlinski and Joachims, 2005].