



CSCE 670 - Information Storage and Retrieval

Lecture 10: Learning to Rank (Cont'd) and Course Project Information Session

Yu Zhang

yuzhang@tamu.edu

September 25, 2025

Course Website: <https://yuzhang-teaching.github.io/CSCE670-F25.html>

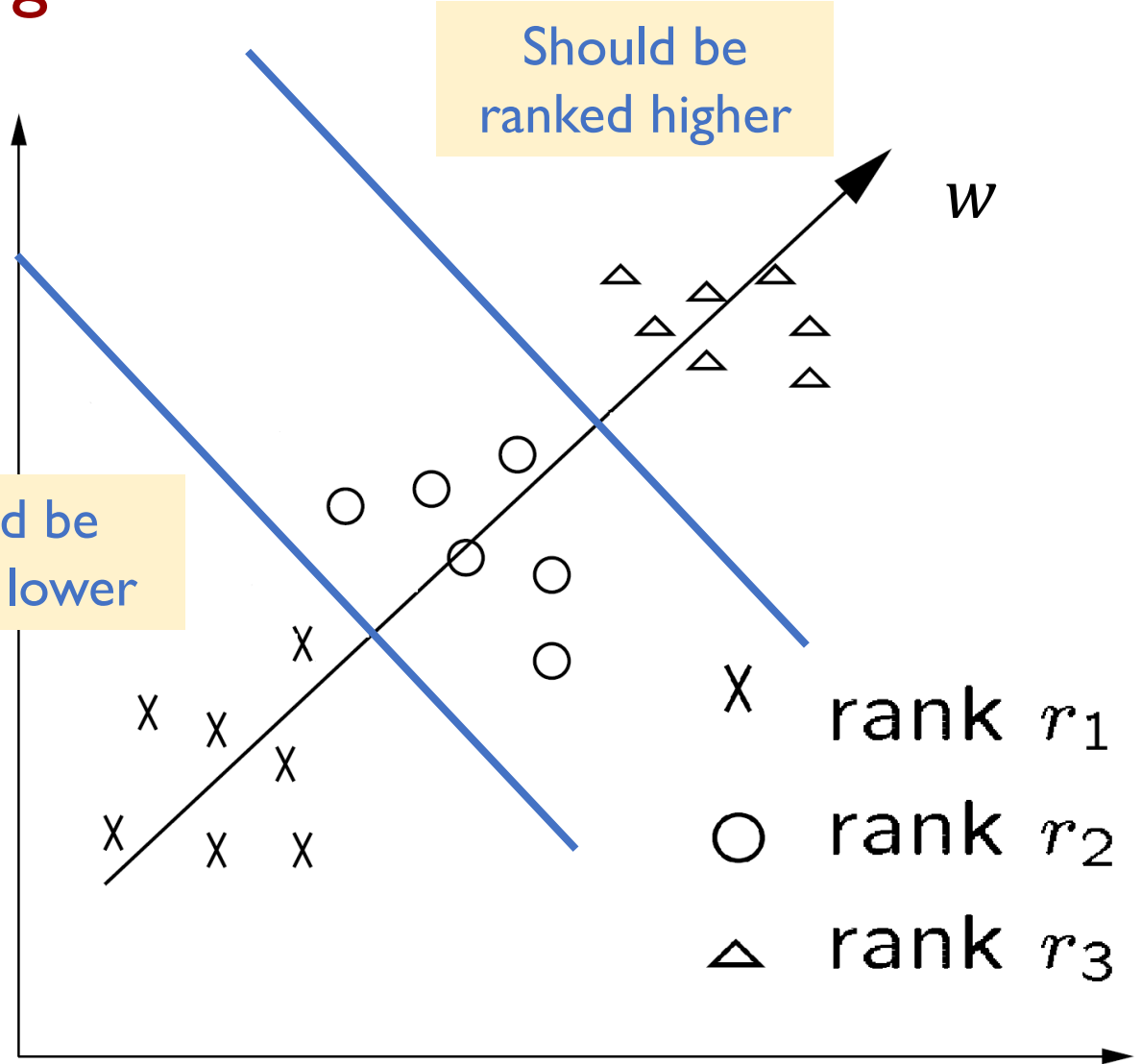
Recap: Pointwise Learning to Rank

Ranking function:

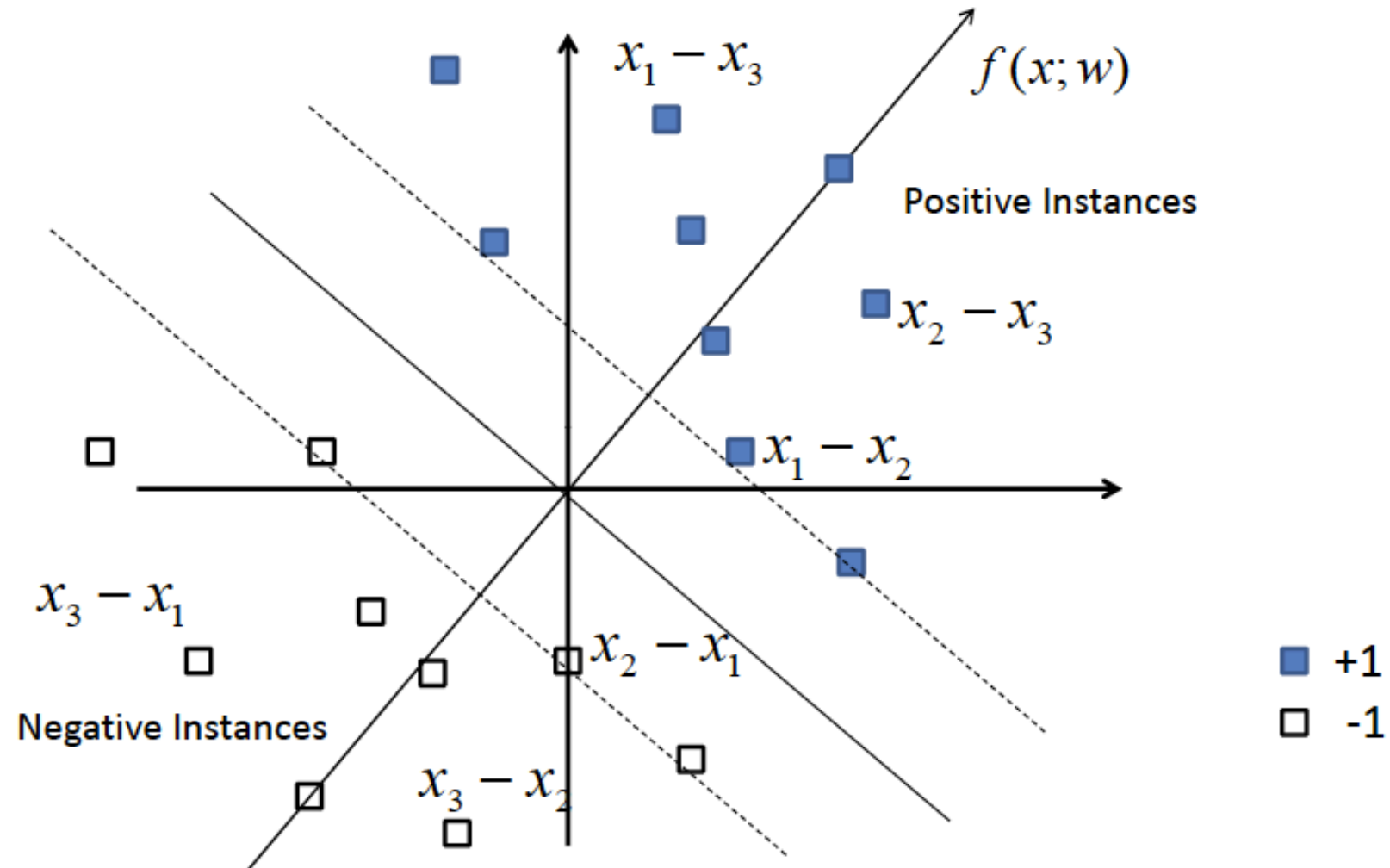
$$f(\psi_i) = w^T \psi_i$$

Should be
ranked lower

Should be
ranked higher



Recap: Pairwise Learning to Rank



Learning to Rank using Gradient Descent

Keywords: ranking, gradient descent, neural networks, probabilistic cost functions, internet search

Chris Burges

CBURGES@MICROSOFT.COM

Tal Shaked*

TAL.SHAKED@GMAIL.COM

Erin Renshaw

ERINREN@MICROSOFT.COM

Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399

Ari Lazier

ARIEL@MICROSOFT.COM

Matt Deeds

MADEEDS@MICROSOFT.COM

Nicole Hamilton

NICHAM@MICROSOFT.COM

Greg Hullender

GREGHULL@MICROSOFT.COM

Microsoft, One Microsoft Way, Redmond, WA 98052-6399

Abstract

We investigate using gradient descent methods for learning ranking functions; we pro-

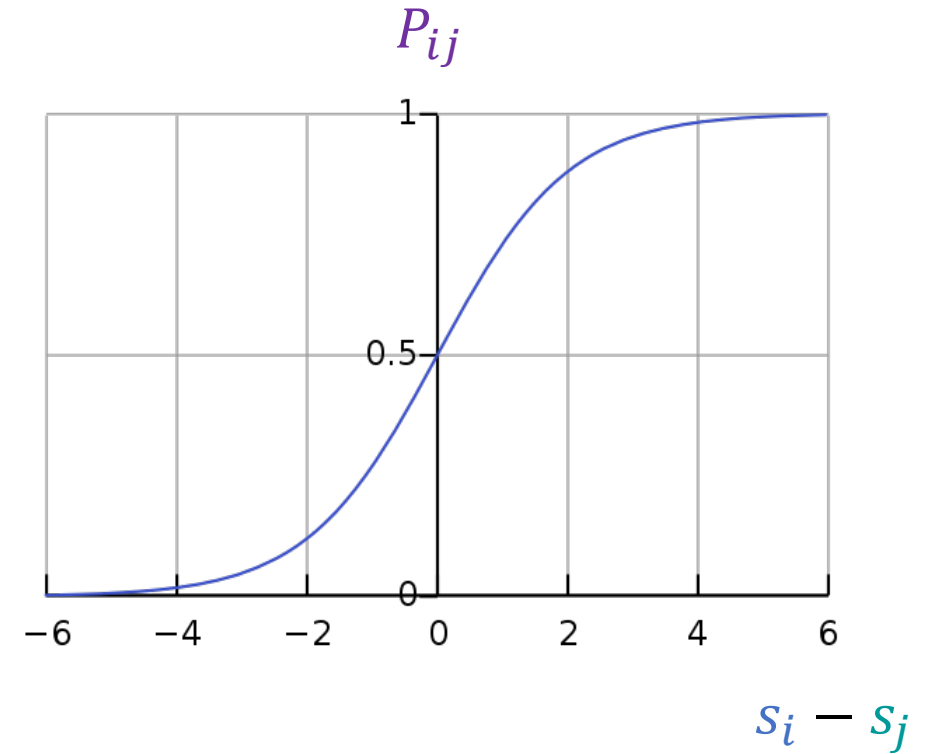
that maps to the reals (having the model evaluate on pairs would be prohibitively slow for many applications). However (Herbrich et al., 2000) cast the rank-

RankNet

- We have documents represented in a feature space: x is an input feature vector
- For document i the representation is x_i
- Our goal is to learn a function f that outputs a score (just a number)
 - $f(x_i) = s_i$
- Suppose we have two documents i and j that we represent as: x_i and x_j
- We can score each document according to our function f
 - So, we have $f(x_i) = s_i$ and $f(x_j) = s_j$
- Let $P_{ij} = \text{Prob}(\text{document } i \succ \text{document } j)$
 - That is, i is better than j for this query (the pairwise learning to rank setting)

RankNet

- $f(x_i) = s_i$ and $f(x_j) = s_j$
- Let $P_{ij} = \text{Prob}(\text{document } i \succ \text{document } j)$
- We can use a Sigmoid function to define P_{ij} :
 - $P_{ij} = \frac{e^{s_i}}{e^{s_i} + e^{s_j}}$ (a.k.a., Bradley–Terry Model)
- What if $s_i \gg s_j$? $P_{ij} \rightarrow 1$
- What if $s_i \ll s_j$? $P_{ij} \rightarrow 0$
- What if $s_i = s_j$? $P_{ij} = 0.5$
- Why not directly define $P_{ij} = \begin{cases} 1, & \text{if } s_i > s_j \\ 0.5, & \text{if } s_i = s_j \\ 0, & \text{if } s_i < s_j \end{cases}$



Loss Function

- Loss function tells us what we pay when the function we are trying to learn makes mistakes.
- Here we use the **cross-entropy** loss function:

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - (1 - \hat{P}_{ij})\log(1 - P_{ij})$$

- P_{ij} is the probability given by the model (i.e., $\frac{e^{s_i}}{e^{s_i} + e^{s_j}}$)
- \hat{P}_{ij} is the actual (i.e., 1, 0.5, or 0, depending on if document $i >$ document j in our training data)
- \log : natural logarithm, to the base of e

Cross-Entropy Loss

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - (1 - \hat{P}_{ij})\log(1 - P_{ij})$$

- Good for binary classification tasks (like in our case, is one document “better” than the other document?)
- Example:
 - Suppose the ground truth is document i \succ document j
 - $\hat{P}_{ij} = 1$
 - Our model predicts document i \succ document j with $P_{ij} = 0.99$
 - $\mathcal{L} \approx 0.01$

Cross-Entropy Loss

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - (1 - \hat{P}_{ij})\log(1 - P_{ij})$$

- Good for binary classification tasks (like in our case, is one document “better” than the other document?)
- Example:
 - Suppose the ground truth is document i \succ document j
 - $\hat{P}_{ij} = 1$
 - Our model predicts document i \succ document j with $P_{ij} = 0.50$
 - $\mathcal{L} \approx 0.69$

Cross-Entropy Loss

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - (1 - \hat{P}_{ij})\log(1 - P_{ij})$$

- Good for binary classification tasks (like in our case, is one document “better” than the other document?)
- Example:
 - Suppose the ground truth is document i \succ document j
 - $\hat{P}_{ij} = 1$
 - Our model predicts document i \succ document j with $P_{ij} = 0.01$
 - $\mathcal{L} \approx 4.61$

Loss Function

- $P_{ij} = \frac{e^{s_i}}{e^{s_i} + e^{s_j}}$
- $\mathcal{L} = -\hat{P}_{ij} \log P_{ij} - (1 - \hat{P}_{ij}) \log(1 - P_{ij})$
- After doing some math, you will have
- $\mathcal{L} = -\hat{P}_{ij}(s_i - s_j) + \log(1 + e^{s_i - s_j})$
- Given the cost function, our goal is to learn **the best model parameters** to minimize the cost. That is, we hope to pick some combination of features that do a good job of guessing when a document is preferred to another document in our training data.
- What are **the model parameters** in this case?
 - $f(x_i) = s_i$
 - E.g., $f(x_i) = w^T s_i$

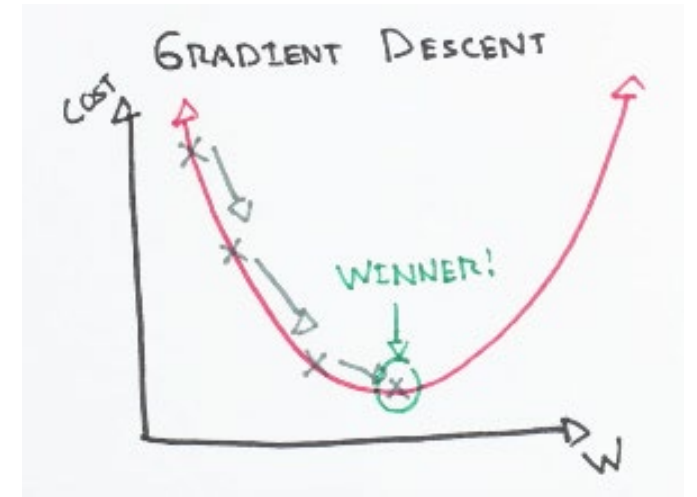
How to minimize the cost?

- Gradient descent!
- To minimize a function $f(x)$:
 - Take the derivative of f : ∇f
 - Start at some point x_0 and evaluate $\nabla f(x_0)$
 - Make a step in the reverse direction of the gradient: $x_1 = x_0 - \eta \nabla f(x_0)$
 - $\eta > 0$ is the learning rate
 - Repeat until converged

How to minimize the cost?

- Example

- $f(x) = (x - 2)^2 + 1$
- $\nabla f(x) = 2(x - 2)$
- If $x_0 < 2$, then $\nabla f(x_0) < 0$, so $x_1 = x_0 - \eta \nabla f(x_0) > x_0$
 - If we still have $x_1 < 2$, then $x_2 = x_1 - \eta \nabla f(x_1) > x_1$
- If $x_0 > 2$, then $\nabla f(x_0) > 0$, so $x_1 = x_0 - \eta \nabla f(x_0) < x_0$
- Finally, we always have x converge to 2, which minimizes $f(x)$



How to minimize the cost?

- Example

- $f(x, y) = (x - 2)^2 + (y - 3)^2 + 1$

- $\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2(x - 2) \\ 2(y - 3) \end{bmatrix}$

- $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \eta \begin{bmatrix} 2(x_0 - 2) \\ 2(y_0 - 3) \end{bmatrix}$

- If $x_0 < 2$, then $x_1 > x_0$

- If $x_0 > 2$, then $x_1 < x_0$

- Regardless of how y changes, x keeps moving closer to 2

- Similarly, regardless of how x changes, y keeps moving closer to 3

- Finally, we always have $\begin{bmatrix} x \\ y \end{bmatrix}$ converge to $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, which minimizes $f(x, y)$

Going Back to RankNet

- $\mathcal{L} = -\hat{P}_{ij}(s_i - s_j) + \log(1 + e^{s_i - s_j})$
- $\frac{\partial \mathcal{L}}{\partial s_i} = -\hat{P}_{ij} + \frac{e^{s_i - s_j}}{1 + e^{s_i - s_j}}$ Let's denote this as λ_{ij}
- $\frac{\partial \mathcal{L}}{\partial s_j} = \hat{P}_{ij} - \frac{e^{s_i - s_j}}{1 + e^{s_i - s_j}} = -\lambda_{ij}$
- Then how can we update a model parameter w_k in $f(x)$ (e.g., $f(x_i) = w^T s_i$)?
- Gradient Descent:
$$w_k \leftarrow w_k - \eta \frac{\partial \mathcal{L}}{\partial w_k}$$
- But how to calculate $\frac{\partial \mathcal{L}}{\partial w_k}$?

Going Back to RankNet

- Then how can we update a model parameter w_k in $f(x)$ (e.g., $f(x_i) = w^T s_i$)?
- Gradient Descent:

$$w_k \leftarrow w_k - \eta \frac{\partial \mathcal{L}}{\partial w_k}$$

- But how to calculate $\frac{\partial \mathcal{L}}{\partial w_k}$?
 - (Multi-variable) chain rule:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \frac{\partial \mathcal{L}}{\partial s_i} \frac{ds_i}{dw_k} + \frac{\partial \mathcal{L}}{\partial s_j} \frac{ds_j}{dw_k} = \lambda_{ij} \frac{ds_i}{dw_k} + (-\lambda_{ij}) \frac{ds_j}{dw_k} = \lambda_{ij} \left(\frac{ds_i}{dw_k} - \frac{ds_j}{dw_k} \right)$$

Interesting Observation

$$\frac{\partial \mathcal{L}}{\partial w_k} = \lambda_{ij} \left(\frac{ds_i}{dw_k} - \frac{ds_j}{dw_k} \right)$$

- This is the gradient for only one training sample (document i > document j)
- What if you have lots of training samples?
 - Add these gradients together!
 - If document i appears in the following training samples:
 - $(i > j_1), (i > j_2), \dots, (i > j_{10})$
 - $(j_{11} > i), (j_{12} > i), \dots, (j_{18} > i)$
 - What would be the term related to document i in the total gradient?

$$(\lambda_{ij_1} + \lambda_{ij_2} + \dots + \lambda_{ij_{10}} - \lambda_{j_{11}i} - \lambda_{j_{12}i} - \dots - \lambda_{j_{18}i}) \frac{ds_i}{dw_k}$$

Interesting Observation

- What would be the term related to document i in the total gradient?

$$(\lambda_{ij_1} + \lambda_{ij_2} + \dots + \lambda_{ij_{10}} - \lambda_{j_{11}i} - \lambda_{j_{12}i} - \dots - \lambda_{j_{18}i}) \frac{ds_i}{dw_k}$$

- λ_{ij} describes the desired change of scores for the pair of training sample (document $i >$ document j).
- The sum of all λ_{ij} 's and λ_{ji} 's related to document i describes the desired direction of i 's movement within the entire ranking list.

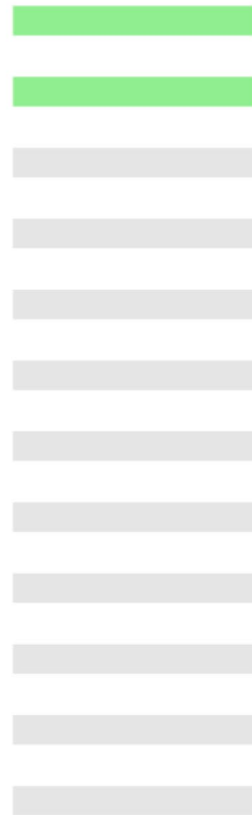
$$\lambda_i = \sum_{j:(i > j)} \lambda_{ij} - \sum_{j:(j > i)} \lambda_{ji}$$

Example

Each blue arrow represents the λ_i for the corresponding document i .

Green: Relevant
to the query

Gray: Irrelevant
to the query



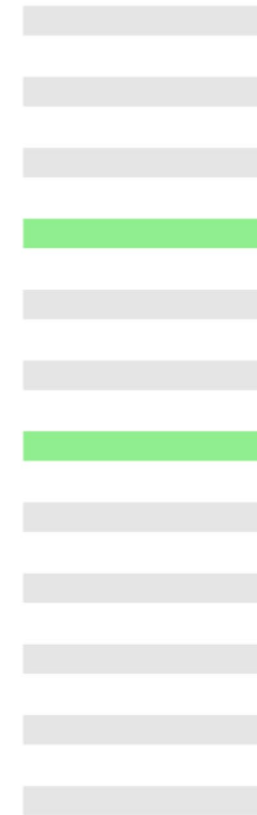
(a)

Perfect



(b)

10 pairwise errors



(c)

8 pairwise errors

Example

However, based on what we know about the importance of ranking in search engines, the **red** arrow seems more reasonable. (Why?)

Green: Relevant
to the query

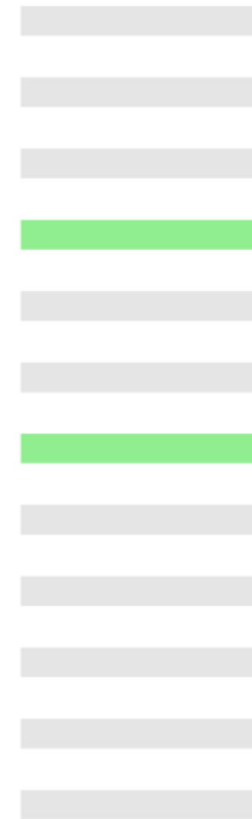
Gray: Irrelevant
to the query



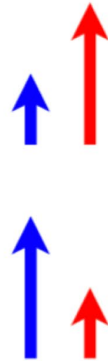
(a)



(b)



(c)



RankNet → LambdaRank [Burges et al., NIPS 2006]

- Instead of working directly with pairwise ranking errors, scale them by their impact on nDCG.
- **Idea:** Multiply λ_{ij} by the difference of an IR measure (e.g., nDCG) when document i and document j are swapped

Learning to Rank with Nonsmooth Cost Functions

Christopher J.C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
cburges@microsoft.com

Robert Ragno
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
rragno@microsoft.com

Quoc Viet Le
Statistical Machine
Learning Program
NICTA, ACT 2601, Australia
quoc.le@anu.edu.au

LambdaRank → LambdaMART [Burges 2010]

- A boosted tree version of LambdaRank
- MART = multiple additive regression trees

From RankNet to LambdaRank to LambdaMART: An Overview

Christopher J.C. Burges

Microsoft Research Technical Report MSR-TR-2010-82


Questions?

Course Project Information Session

Group Project

- Teams of **3 or 4** (any deviation from this size requires prior approval from the instructor)
 - **Option 1**: A prototype (search engine or recommender system)
 - **Option 2**: A research project (e.g., reasoning-search interleaved LLMs)
 - **Option 3**: A survey (e.g., recent studies on search-enhanced LLMs)
- **Topic-wise**: your choice, as long as it is related to the themes of this course!
- Proposal is due after you will have seen basics of both search engines and recommender systems
 - Think of this as a launching pad for your project and not a constraint

Themes of this Course

-  **Phase 1: Search Engines**
 - basics, Boolean and ranked retrieval, link analysis, evaluation, learning to rank (ML + ranking), ...
- **Phase 2: Recommender Systems**
 - basics, non-personalized recommendation, collaborative filtering, matrix factorization, implicit recommendation, ...
- **Phase 3: From Foundations to Modern Methods**
 - embedding learning, Transformer, “small” language models, ... (for search and recommendation)
- **Phase 4: Large Language Models (!!)**



Proposal Due

Proposal (2% of your Overall Grade)

- **October 18** (Saturday) by 11:59pm CT
- Post to Canvas
 - Only one team member needs to submit the PDF
 - TA will create the teams on Canvas after grading all proposals, since some groups may deviate from the required size without notifying us in advance and we may need to adjust their teams a bit
- ACL 2023 template: https://2023.aclweb.org/calls/style_and_formatting/
 - Keep it within **2 pages** maximum
- In the proposal PDF, include **all team member names and NetID**
- List each of the following questions and provide a brief answer (pick the set of questions appropriate for your project: **prototype**, **research**, or **survey**)

Proposal: Prototype

- What is exactly the function of your tool? That is, what will it do?
- Why would we need such a tool and who would you expect to use it and benefit from it?
- Does this kind of tool already exist? If similar tools exist, how is your tool different from them? Would people care about the difference? How hard is it to build such a tool? What is the challenge?
- How do you plan to build it? You should mention the data you will use and the core algorithm that you will implement.
- What existing resources can you use?
- How will you demonstrate the usefulness of your tool?
- A rough timeline to show when you expect to finish what. List a couple of milestones.

Additional Requirements for a Prototype Project

- **Requirement 1:** You must use *non-trivial* data
 - Examples of *non-trivial* data
 - Sample using an API
 - Download an existing collection
 - Write a crawler
 - Examples of *trivial* data
 - You spend 10 minutes writing 3 queries and 10 candidate sentences
- **Requirement 2:** You should spend 3-4 minutes giving a live demo of the prototype in your final project presentation.
 - Example (in terms of form, not content):
<https://www.youtube.com/watch?v=aIIQZz0cUUc>

Proposal: Research

- What is your research question? Clearly define the research problem/question.
- Why is this an interesting question to ask and why would we care about the answer to this question or a solution to the problem.
- Has any existing research work tried to answer the same or a similar question, and if so, what is still unknown?
- How do you plan to work out the answer to the question. (At the proposal stage, you are only expected to have a sketch of your methods.)
- How would you evaluate your solution. That is, how do you plan to demonstrate that your solution/answer is good or is reasonable.
- A rough timeline to show when you expect to finish what. List a couple of milestones.

Additional Requirements for a Research Project

- **Requirement 1:** You should identify a recent high-quality research paper (e.g., published at SIGIR, WWW, KDD, WSDM, RecSys, ACL, EMNLP, NAACL, etc.), better with an existing codebase
 - Technically, focus on one clean insight to improve the model proposed in the paper
 - Empirically, you do NOT have to show an improvement (remember, this is a short-fuse class project), but you need to identify key findings
- **Requirement 2:** Code must live on GitHub
 - Public code repository; or private + share with TA and me
 - Write a detailed README
 - Example (in terms of form, not content): <https://github.com/yuzhimanhua/MATCH>

Proposal: Survey

- What is the scope of your survey? Clearly define the area, topic, or set of methods you plan to cover.
- Why is this an important area to survey? Who would benefit from such a survey?
- Has there already been a survey or overview written on this topic? If so, what gaps or limitations exist that your survey could address?
- How will you structure your survey? For example, by taxonomy, chronology, methodology, application, or another organizing principle.
- What do you hope the reader will learn from your survey? What kinds of insights or take-aways will you emphasize?
- A rough timeline to show when you expect to finish what. List a couple of milestones.

Additional Requirements for a Survey Project

- **Requirement 1:** At least 20 papers, with no fewer than half published within the past three years.
 - Unpublished preprints (e.g., on arXiv) can also be included.
- **Requirement 2:** The works must be organized in a coherent and systematic order, rather than simply listed/appended.
 - A taxonomy of surveyed works should be provided in your final presentation/report
 - Example (in terms of form, not content): Figure 3 in <https://arxiv.org/pdf/2502.17504>
- **Requirement 3:** A corresponding GitHub repository (usually named as “Awesome-XXX”) is required (a README file suffices).
 - Public; or private + share with TA and me
 - Example (in terms of form, not content or number of papers surveyed): <https://github.com/yuzhimanhua/Awesome-Scientific-Language-Models>

Suggestions

- **Suggestion 1:** Clearly distinguish your project from existing efforts
 - You are welcome to use any existing resources that you like
 - We evaluate what you **add**, rather than what you use
- **Suggestion 2:** Have a sharp focus
 - Do something well, rather than a million things poorly
- **Suggestion 3:** If you do not have great data, consider ChatGPT as a “**data augmentor**”.

I like horror movies. My favorites are *The Shining* and *The Texas Chain Saw Massacre*. What songs could you recommend that have similar vibes?

Got it! Since you like *The Shining* and *The Texas Chain Saw Massacre*, you're drawn to unsettling, eerie, and tension-filled vibes—sometimes slow-burning dread, sometimes raw, chaotic energy. Here's a mix of songs that capture those moods:

1. Creepy/Atmospheric Horror Vibes (*Shining*-like):

- Bohren & der Club of Gore – “Prowler” → slow, jazzy, ominous.
- John Carpenter – “Halloween Theme” → minimalist, suspenseful, iconic tension.
- Aphex Twin – “Come to Daddy (Pappy Mix)” → unsettling electronic chaos.
- Angelo Badalamenti – “Twin Peaks Theme” → dreamy but with an underlying unease.

2. Raw, Disturbing Energy (*Texas Chain Saw Massacre*-like):

- Nirvana – “Milk It” → abrasive, claustrophobic, twisted energy.
- Death Grips – “Get Got” → chaotic and unsettling industrial-rap vibes.

I like horror movies. My favorites are *The Shining* and *The Texas Chain Saw Massacre*. Which song would I prefer: a) *Wildest Dreams* by Taylor Swift or b) *Yellow Submarine* by The Beatles. Please explain your reasoning.

Given your taste in horror movies—*The Shining* and *The Texas Chain Saw Massacre*—you seem drawn to eerie, unsettling, or tense atmospheres rather than bright, whimsical, or romantic ones.

- **“Wildest Dreams” by Taylor Swift** → dreamy, cinematic, and a bit moody, but primarily romantic and polished. It has some atmospheric quality, though not particularly dark or horror-inspired.
- **“Yellow Submarine” by The Beatles** → cheerful, colorful, and whimsical—essentially the opposite of horror vibes.

Reasoning: Between the two, *Wildest Dreams* aligns slightly closer to a cinematic, moody feel, which could resonate more with someone who appreciates tension and atmosphere, even though it’s not truly “horror.” *Yellow Submarine* is too lighthearted and playful to match your taste.

✅ You would likely prefer “Wildest Dreams.”

Tell me 10 recent papers on search-enhanced LLMs

Certainly! Here are 10 recent academic papers focused on search-enhanced large language models (LLMs), highlighting advancements in integrating search capabilities to enhance LLM performance:

1. Search-o1: Agentic Search-Enhanced Large Reasoning Models

This paper introduces a framework that combines LLMs with search capabilities to enhance reasoning tasks, enabling models to retrieve and utilize external information effectively.

2. Enhancing Large Language Models through Structured Reasoning

The authors propose methods to integrate structured reasoning processes into LLMs, improving their ability to handle complex tasks by leveraging external search data.

3. Reinforcement Learning Enhanced LLMs: A Survey

This survey examines the integration of reinforcement learning with LLMs, discussing how search-enhanced techniques can improve model performance in dynamic environments.

4. Agent-Enhanced Large Language Models for Political Science Research

The paper demonstrates how LLMs, augmented with predefined functions and specialized tools, can serve as dynamic agents capable of streamlining tasks such as data collection, preprocessing, and

Presentation (9% of your Overall Grade)

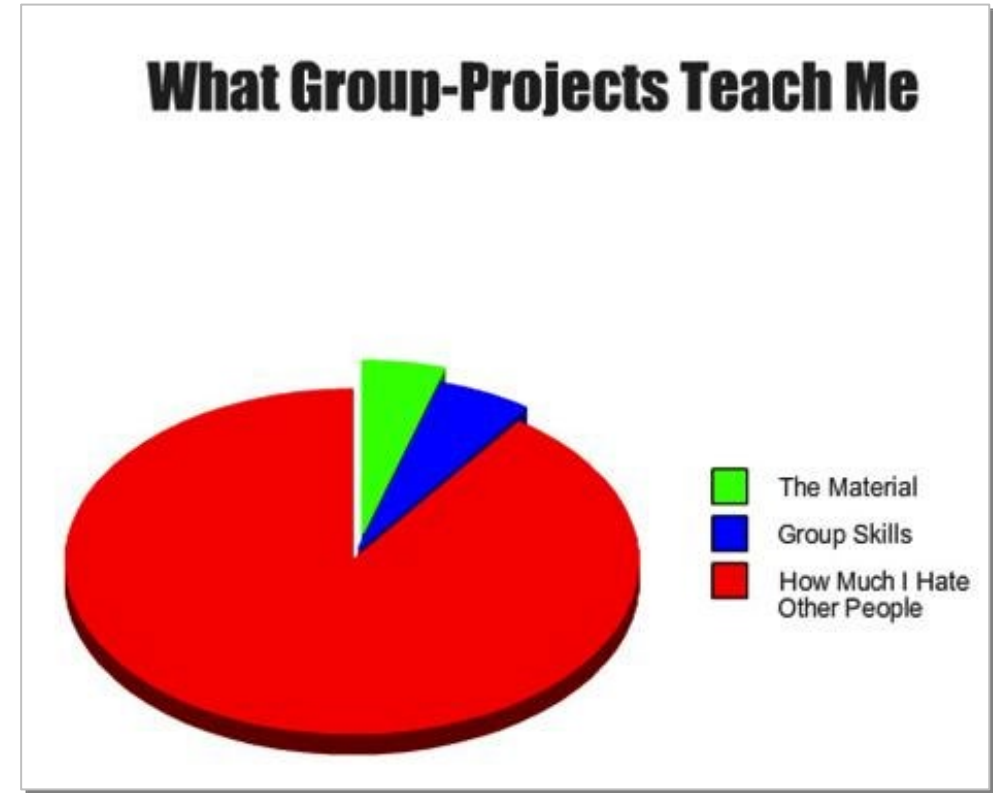
- December 2 and December 4 classes
- Zoom: <https://tamu.zoom.us/j/6411788612>
- The duration depends on the number of groups, with an estimated 8-10 minutes per group.
 - The 3-4 minute live demo by prototype groups is included within the estimated 8-10 minutes per group.
- Not everyone in the group needs to participate in the presentation.
- Any slide template is okay.

Final Report (9% of your Overall Grade)

- **December 9** (Tuesday) by 11:59pm CT
- Post to Canvas
- ACL 2023 template: https://2023.aclweb.org/calls/style_and_formatting/
 - **Prototype**: 3-5 pages
 - Example (form, not content): <https://aclanthology.org/2020.acl-demos.5.pdf>
 - **Research**: 6-8 pages
 - Example (form, not content): <https://aclanthology.org/2024.findings-acl.11.pdf>
 - **Survey**: 6-8 pages
 - Example (form, not content): <https://aclanthology.org/2024.emnlp-main.498.pdf>
- Each team only needs to submit one PDF

Teamwork

- You will receive an **overall grade** based on the performance of your entire team.
- We will also assess your **individual contribution** (through peer feedback, contributions to the code repository, etc.).
- Typically, the individual rating can increase or depress your project grade by **only some small delta**.



Questions?



Thank You!

Course Website: <https://yuzhang-teaching.github.io/CSCE670-F25.html>