Learning to Rank

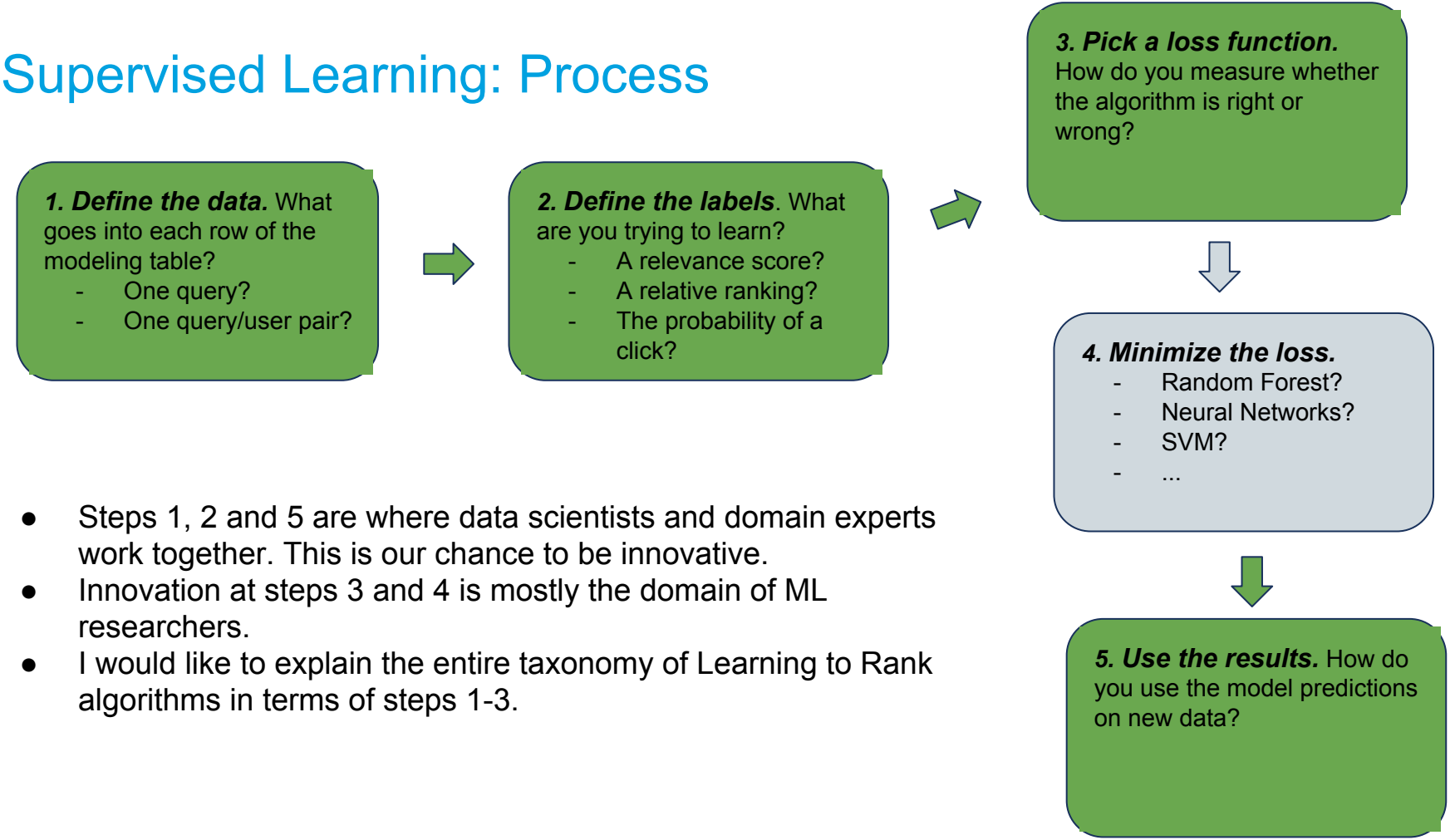
University of Washington - Applied Machine Learning

Overview

Goal: High-level overview of techniques for Learning to Rank

1. A few words about supervised learning.
2. Logistic regression for search relevance
   a. Some detail here because it is good scaffolding for the rest of the talk.
3. Comparison of loss functions
4. Taxonomy of methods for Learning to Rank
   a. Pointwise
   b. Pairwise
   c. Listwise
5. Example: Experimenting with Salesforce Search Logs.
6. What we really do at Salesforce (Implicit Feedback)

Supervised Learning: Process



- Steps 1, 2 and 5 are where data scientists and domain experts work together. This is our chance to be innovative.
- Innovation at steps 3 and 4 is mostly the domain of ML researchers.
- I would like to explain the entire taxonomy of Learning to Rank algorithms in terms of steps 1-3.

Example: Logistic Regression for Relevance

- Use a specific, and relatively simple ML algorithm to illustrate the idea that we can think about all of our problems in terms of steps 1-3.
- Logistic regression is the baseline binary classifier---analogous to linear regression.
- The goal is to use a set of features to predict a binary outcome. For example, fraud or not fraud, spam or not spam, relevant or not relevant, etc.

Example: Logistic Regression

Supervised Learning for Relevance

Step 1: Define the data.

- Each row is a query/document pair
- Each column is some measured feature of relevance

Query-Doc	Cosine Distance	Recency
1-1	.2	1
1-2	.0	2
1-3	.3	8
2-1	.6	2
2-2	.2	0
2-3	.0	2

Example: Logistic Regression

Supervised Learning for Relevance

Step 2: Define the labels.

- Decide that relevance is binary.
- Each document is either relevant or not.
- 1 = relevant, 0 = not relevant
- For now, we assume that someone had to go through and manually label some training data.

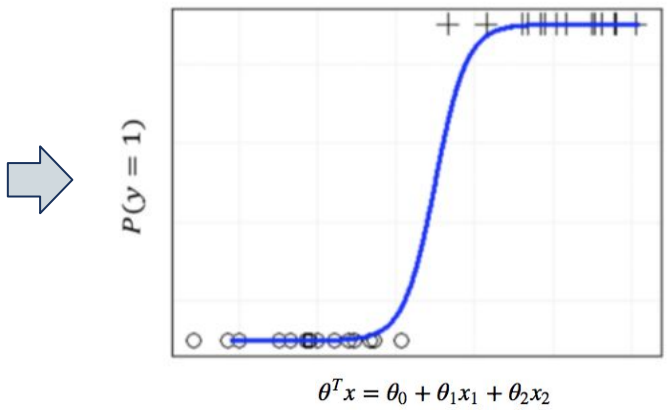
Query-Doc	Cosine Distance	Recency	Relevant
1-1	.2	1	0
1-2	.0	2	1
1-3	.3	8	1
2-1	.6	2	0
2-2	.2	0	1
2-3	.0	2	0

Derive the Loss function

Logistic regression is defined by the following assumption:

$$P(y = 1 \mid x\,;\theta) := \mathrm{F}(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Cosine Distance	Recency	Relevant
.2	1	0
.0	2	1
.3	8	1
.6	2	0
.2	0	1
.0	2	0

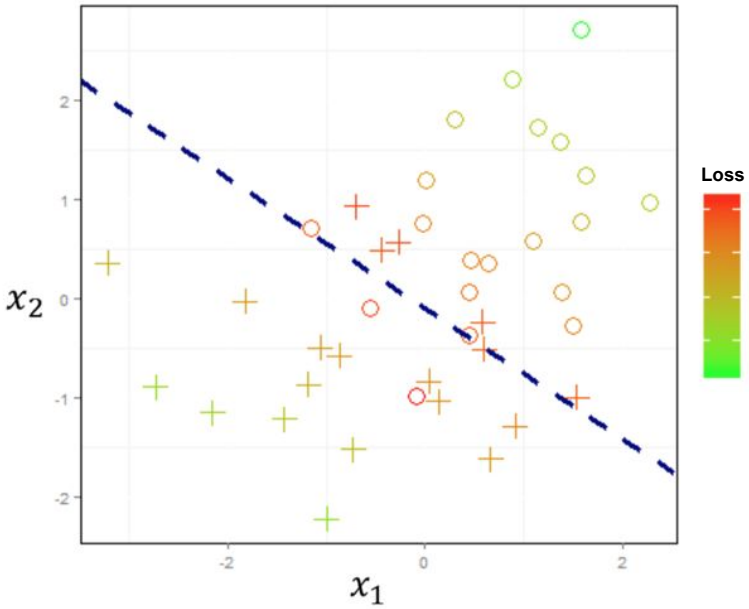


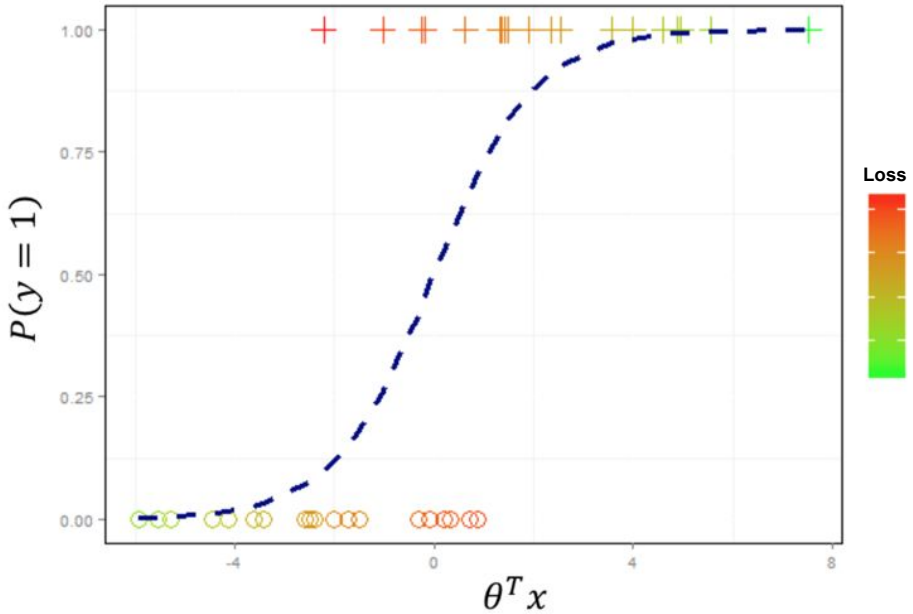
That is, the higher the $\theta^T x$ for a query/document pair, the more likely it is that the document is relevant to that query.

Training

The loss function for logistic regression is *cross-entropy*

$$\text{Loss}(F(x_i), y_i) = -y_i \log(F(x_i)) - (1 - y_i) \log(1 - F(x_i))$$





Training consists of finding the θ that minimizes the total loss, or equivalently, that maximizes the probability of the data (given our assumptions) --- using gradient descent.

Logistic Regression for Binary Relevance

Step 1: Define the data.

- Each row is a query/document pair
- Each column is some measured feature of relevancy

Query-Doc	Cosine Distance	Recency	Label	Prediction
1-1	.2	1.8	0	**.2**
1-2	.0	2	1	**.87**
1-3	.2	1.8	1	**.79**
2-1	.0	2	0	**.1**
2-2	.2	1.8	1	**.93**
2-3	.0	2	0	**.34**

Step 2: Define the labels.

- Decide that relevance is binary.
- Each document is either relevant or not.
- 1 = relevant, 0 = not relevant
- For now, we assume that someone had to go through and manually label some training data.

Step 3: Define the loss.

- Cross-Entropy

$$\text{Loss}(F(x_i), y_i) = -y_i \log(F(x_i)) - (1 - y_i) \log(1 - F(x_i))$$

Step 5: Use the predictions.

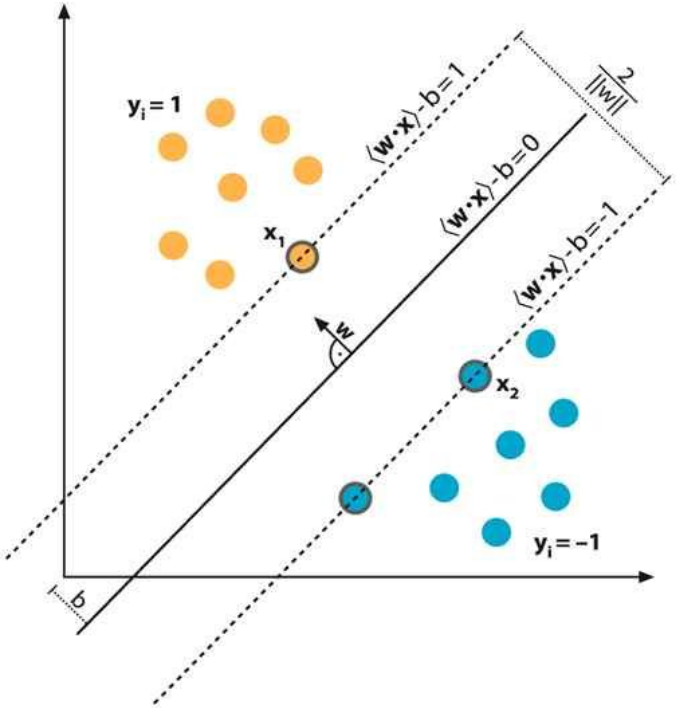
- The predictions have a natural probabilistic interpretation. The higher the output of the model, the more likely the document is relevant to the query.
- Then $\theta^T x$ can be used as a scoring function. (Since the logistic function is monotonic).

Logistic Regression vs SVM

Two ways of finding a separating hyperplane

- SVMs are another way of finding a separating hyperplane, according to a different criteria.
- Find the separating hyperplane that *maximizes the margin between classes*

Support Vector Machines



- Support Vector Machines find a separating hyperplane that maximizes the margin between the classes.
- The output of the predictor is the signed distance of the new point from the hyperplane.

Support Vector Machine for Binary Relevance

Step 1: Define the data.
- Each row is a query/document pair
- Each column is some measured feature of relevancy

Step 2: Define the labels.
- 1 → The document is relevant to the query
- -1 → The document is not relevant to the query

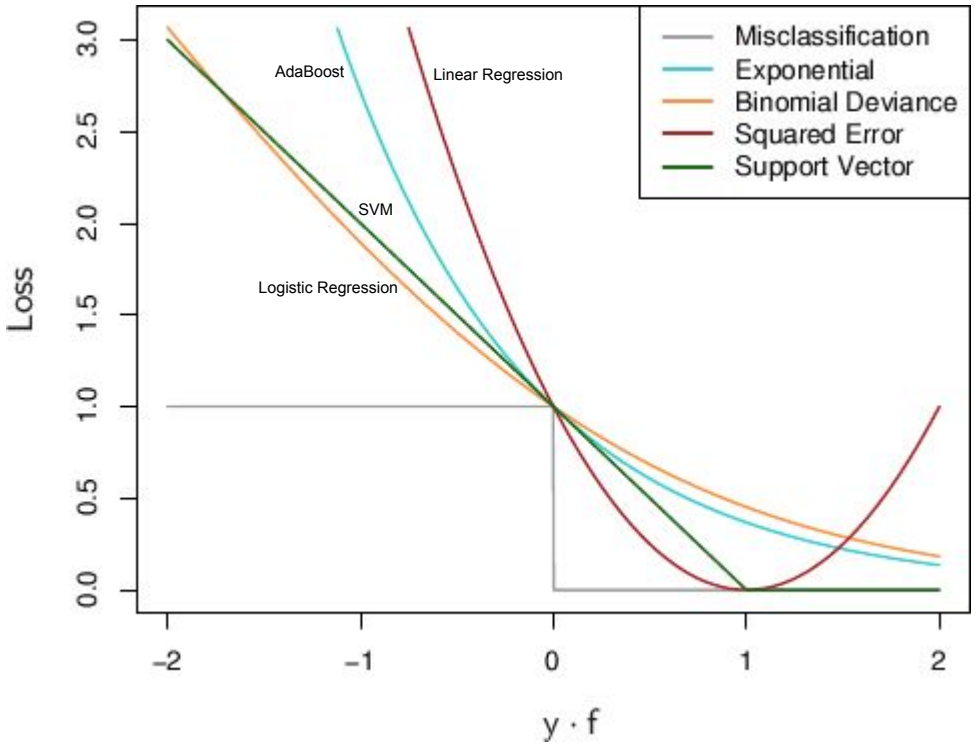
Step 3: Define the loss.
- Hinge Loss
- Loss(x, y) = max(0, 1 - y · ϕ(x))

Step 5: Use the predictions
- The sign of the output is the predicted class and the size of the output is the distance from the separating hyperplane.
- So we can again use ϕ(x) as a scoring function (the higher the better.

Query-Doc	Cosine Distance	Recency	Label	ϕ(x) (signed distance from separating hyperplane)
1-1	.2	1	-1	**-1.9**
1-2	.0	2	1	**2.4**
1-3	.3	8	1	**.09**
2-1	.6	2	-1	**-23**
2-2	.2	0	1	**-.2**
2-3	.0	2	-1	**.34**

* There is additional power in SVMs because ϕ can contain nonlinear transformations. I won't talk about that here.

Loss Functions



f: scoring function

- Random Forests, Gradient Boosted Decision Trees, Neural Networks, etc., are all just different approaches for minimizing these same loss functions.
- They all work. The important thing (for most problems) is the choice of data and labels, and that the loss function makes sense
- Comparing hinge and logistic loss: hinge is very robust to outliers and provides no additional benefit for being “really, really correct”

Evaluation

- How would you decide between a logistic regression model and and SVM?
- The loss functions are different, and hence not directly comparable.
- A couple of choices
  - Good: ROC curves as a measure of classifier performance
  - Better: A domain specific metric.

Evaluation

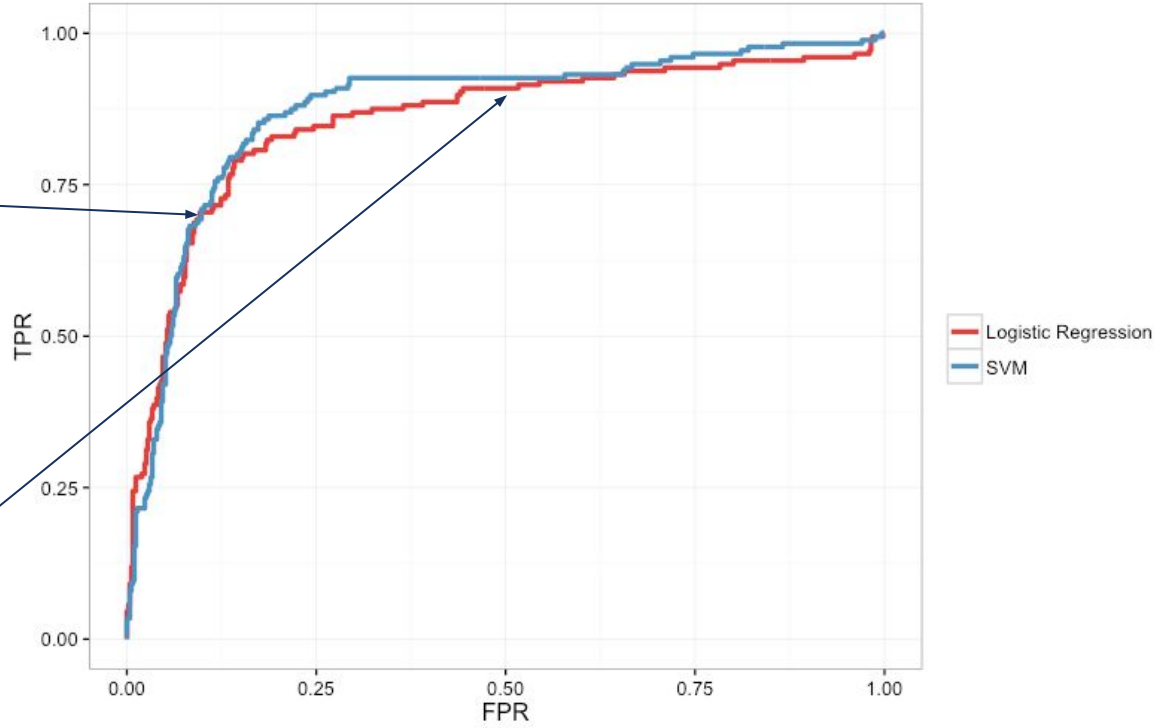
ROC Curves

Predict relevant if model output > 2

T = 2	Predicted Relevant	Predicted Not Relevant
Relevant	70	30
Not Relevant	300	2700

Predict relevant if predicted probability > .77

T = .77	Predicted Relevant	Predicted Not Relevant
Relevant	90	10
Not Relevant	1500	1500



ROC Curves let us compare binary classifiers, even when they are optimized on different loss functions.

Metrics for Ranking

Mean Average Precision

- Global metric for one query and set of returned documents
- Works for rankings of documents where relevance is binary.
- Average precision is the average over all relevant results in a query of the fraction of results up to that position that are relevant
- Mean Average Precision is the average over all queries of the Average Precision.

Query j	**Relevant**	Precision at Position k
Position 1	1	1
Position 2	1	1
Position 3	1	1
Position 4	0	--
Position 5	1	4/5
Position 6	1	5/6
Position 7	0	--

AP = (1 + 1 + 1 + ⅘ + ⅚) / 5

Taxonomy of Learning to Rank

Four basic approaches

- Pointwise
  - **Data**: One observation = query/record pair
  - **Labels**: Binary or ordinal
  - **Loss**: Standard loss functions
- Pairwise
  - **Data**: One observation = two query/record pairs *from the same query*
  - **Labels**: Binary
  - **Loss**: Standard
- Listwise
  - **Data**: One observation = one query
  - **Labels**: List of ordinal relevance rankings (one for each record)
  - **Loss**: Standard relevance metrics (e.g., MAP, NDCG)
- Implicit Feedback
  - **Data:** One observation = query/record pair, or one query.
  - **Labels:** User clicks
  - **Loss:** Negative-log likelihood (i.e. using probabilistic models)

A note on data labeling

Bing, Google, etc. (are there still others??) rely heavily on human-labeled data. Typically on a binary scale or a scale from 1-5. Here's an excerpt from Microsoft's Learning to Rank test dataset:

Dataset Descriptions

The datasets are machine learning data, in which queries and urls are represented by IDs. The datasets consist of feature vectors extracted from query-url pairs along with relevance judgment labels:

(1) The relevance judgments are obtained from a retired labeling set of a commercial web search engine (Microsoft Bing), which take 5 values from 0 (irrelevant) to 4 (perfectly relevant).

(2) The features are basically extracted by us, and are those widely used in the research community.

In the data files, each row corresponds to a query-url pair. The first column is relevance label of the pair, the second column is query id, and the following columns are features. The larger value the relevance label has, the more relevant the query-url pair is. A query-url pair is represented by a 136-dimensional feature vector. The details of features can be found here.

Labeled Data

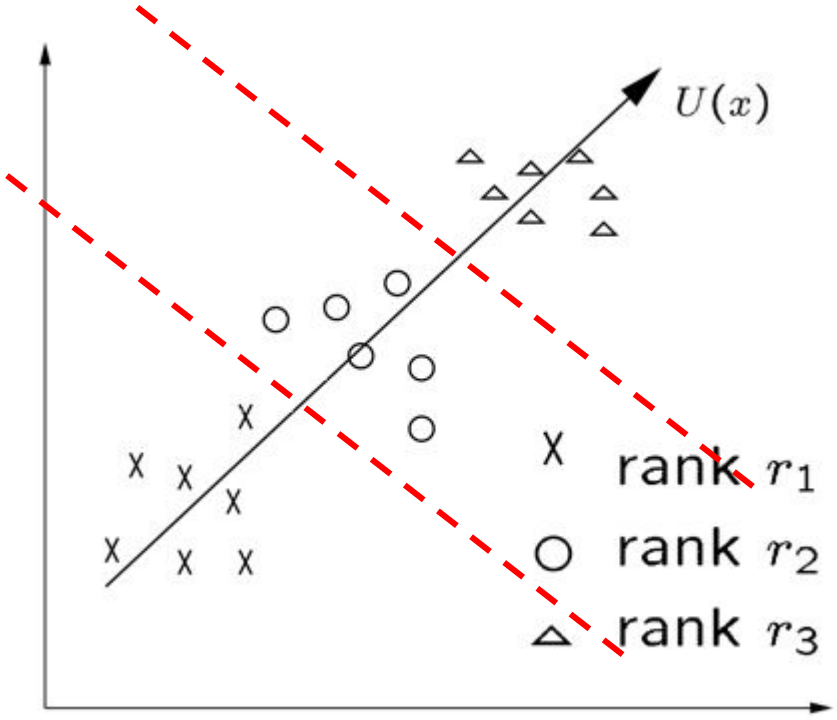
The vast majority of Learning to Rank research starts with data like this

Query-Doc	Cosine Distance	Recency	Rating
1-1	.2	1.8	1
1-2	.0	2	4
1-3	.2	1.8	3
2-1	.0	2	5
2-2	.2	1.8	5
2-3	.0	2	1

Ordinal Classification

- Pointwise
  - **Data**: One observation = query/record pair
  - **Labels**: Binary or ordinal
  - **Loss**: Standard loss functions
- Pairwise
  - **Data**: One observation = two query/record pairs *from the same query*
  - **Labels**: Binary
  - **Loss**: Standard
- Listwise
  - **Data**: One observation = one query
  - **Labels**: List of ordinal relevance rankings (one for each record)
  - **Loss**: Standard relevance metrics (e.g., MAP, NDCG)
- Implicit Feedback
  - **Data:** One observation = query/record pair, or one query.
  - **Labels:** User clicks
  - **Loss:** Negative-log likelihood (i.e. using probabilistic models)

Ordinal SVM - *Shashua and Levin*



The Ordinal SVM solves an ordered classification problem instead of binary classification.

The idea is to find a set of parallel hyperplanes that divide the data into n ordered classes.

Ordinal SVM

Step 1: Define the data.

- Each row contains the *difference in features* for two documents returned for the same query.

Step 2: Define the labels.

- Judged by humans
- Ordinal scale from 1-5
- 1=Least Relevant, 5=Most Relevant

Step 3: Define the loss.

- Sum of hinge loss functions (one for each hyperplane)

Step 5: Use the predictions

- The output of $\phi(x)$ is tested against the 5 different intervals between hyperplanes, corresponding to 5 different relevance levels.
- Bigger is always better, so again $\phi(x)$ can be directly used as a scoring function.

Query-Doc	Cosine Distance	Recency	Label	$\phi(x)$ (signed distance from separating hyperplane)
1-1	.2	1	1	**-1.9**
1-2	.0	2	3	**2.4**
1-3	.3	8	4	**.09**
2-1	.6	2	5	**-23**
2-2	.2	0	2	**-.2**
2-3	.0	2	4	**.34**

Pairwise Ranking

- Pointwise
  - **Data**: One observation = query/record pair
  - **Labels**: Binary or ordinal
  - **Loss**: Standard loss functions
- Pairwise
  - **Data**: One observation = two query/record pairs *from the same query*
  - **Labels**: Binary
  - **Loss**: Standard
- Listwise
  - **Data**: One observation = one query
  - **Labels**: List of ordinal relevance rankings (one for each record)
  - **Loss**: Standard relevance metrics (e.g., MAP, NDCG)
- Implicit Feedback
  - **Data:** One observation = query/record pair, or one query.
  - **Labels:** User clicks
  - **Loss:** Negative-log likelihood (i.e. using probabilistic models)

Ranking SVM - *Herbrich, Graepel, Obermayer*

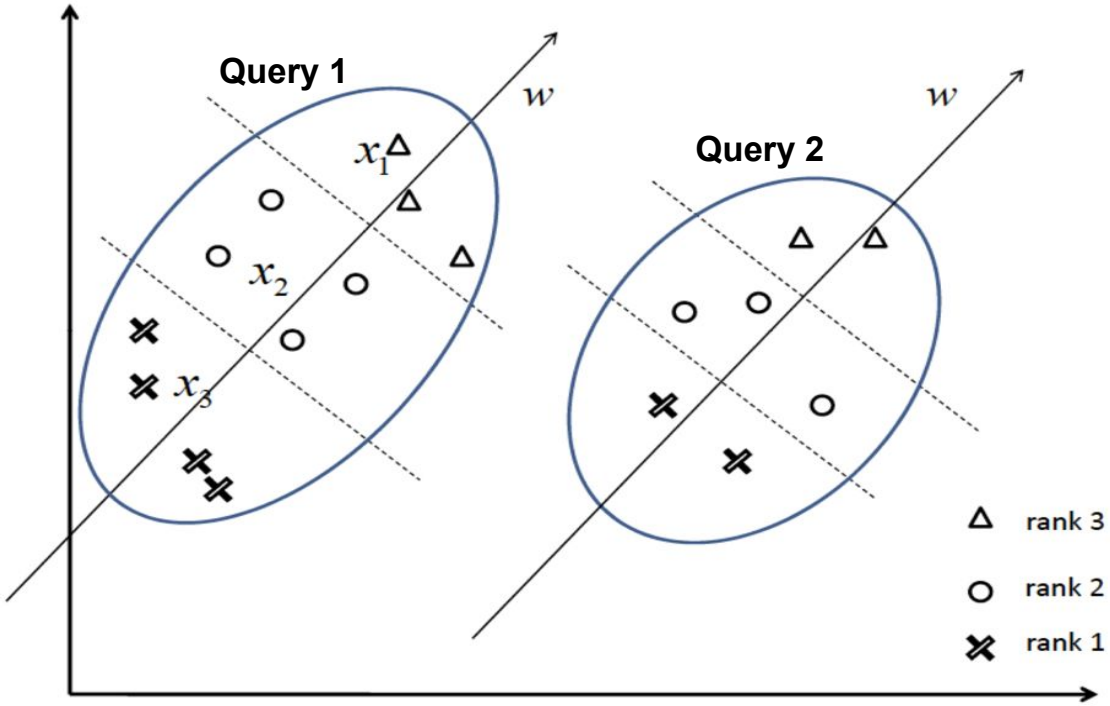
- Classify instance pairs as correctly ranked or incorrectly ranked
- Turn ordinal regression back into a binary classification problem
- We want a ranking function ϕ such that

$$y_i > y_k \text{ iff } \phi(\mathbf{x}_i) > \phi(\mathbf{x}_k)$$

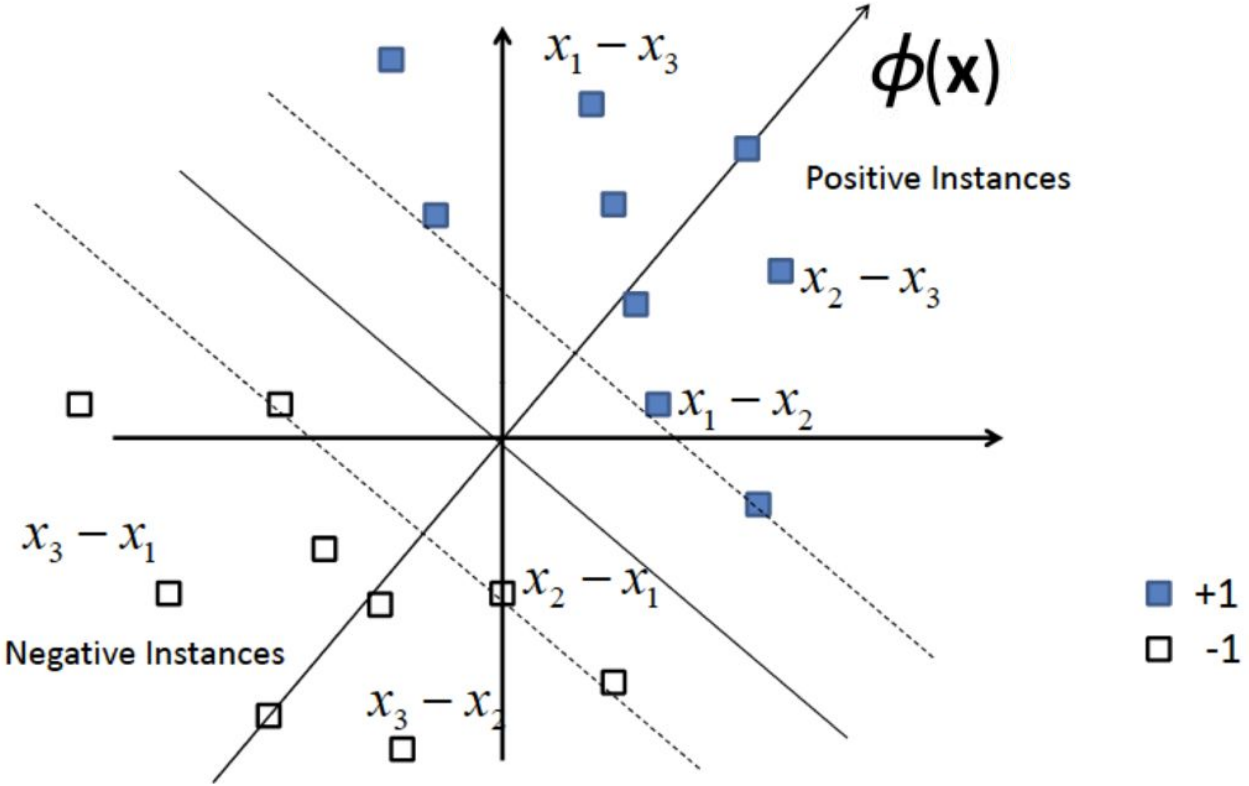
- That is, scoring function applied to $query_j/doc_i$ should be greater than the scoring function applied to $query_j/doc_k$ whenever d_i is more relevant than d_k (within query j)

Pairwise SVM

Use the same scoring function for all queries, but different queries don't have to compare to each other.



As long as ϕ is linear, then $\phi(\mathbf{x}_i) > \phi(\mathbf{x}_k)$ iff $\phi(\mathbf{x}_i - \mathbf{x}_k) > 0$



The Ranking SVM

Step 1: Define the data.

- Each row contains the *difference in features* for two documents returned for the same query.

Step 2: Define the labels.

- 1 → The first query is more relevant
- -1 → The second query is more relevant

Step 3: Define the loss.

- Hinge Loss
- Loss(x, y) = max(0, 1 - y · ϕ(x))
- As long as ϕ is linear

Step 5: Use the predictions

- ϕ is *still* a scoring function. The higher the better.

Query-Doc	Cosine Distance (diff)	Recency (diff)	Label	ϕ(x) (signed distance from separating hyperplane)
1-1 vs 1-2	.2	1	1	**1.9**
1-2 vs 1-3	.0	2	-1	**0.4**
1-3 vs 1-1	-.3	3	1	**.09**
2-1 vs 2-2	.6	-2	-1	**-23**
2-2 vs 2-3	-.2	0	1	**0.2**
2-3 vs 2-1	.0	2	-1	**-.34**

RankNet

Step 1: Define the data.

- Each row contains the *difference in features* for two documents returned for the same query.

Step 2: Define the labels.

- 1 → The first query is more relevant
- -1 → The second query is more relevant

Step 3: Define the loss.

- **Logistic loss (cross-entropy)**

Step 5: Use the predictions

- f is *still* a scoring function. The higher the better.

Query-Doc	Cosine Distance (diff)	Recency (diff)	Label	f(x)
1-1 vs 1-2	.2	1	1	**.88**
1-2 vs 1-3	.0	2	0	**0.4**
1-3 vs 1-1	-.3	3	1	**.09**
2-1 vs 2-2	.6	-2	0	**.04**
2-2 vs 2-3	-.2	0	1	**0.69**
2-3 vs 2-1	.0	2	0	**.22**

LambdaRank

Step 1: Define the data.

- Each row contains the *difference in features* for two documents returned for the same query.

Step 2: Define the labels.

- 1 → The first query is more relevant
- -1 → The second query is more relevant

Step 3: Define the loss.

- **Logistic loss (cross-entropy)**
- **Weight each observation according to how much a global metric will change if the label is swapped (e.g. MAP)**

Step 5: Use the predictions

- f is *still* a scoring function. The higher the better.

Query-Doc	Cosine Distance (diff)	Recency (diff)	Label	f(x)
1-1 vs 1-2	.2	1	1	**.88**
1-2 vs 1-3	.0	2	0	**0.4**
1-3 vs 1-1	-.3	3	1	**.09**
2-1 vs 2-2	.6	-2	0	**.04**
2-2 vs 2-3	-.2	0	1	**0.69**
2-3 vs 2-1	.0	2	0	**.22**

LambdaMART

Step 1: Define the data.

- Each row contains the *difference in features* for two documents returned for the same query.

Step 2: Define the labels.

- 1 → The first query is more relevant
- -1 → The second query is more relevant

Step 3: Define the loss.

- Logistic loss (cross-entropy)
- Weight each observation according to how much a global metric will change if the label is swapped (e.g. MAP)

Step 5: Use the predictions

- f is *still* a scoring function. The higher the better.

Query-Doc	Cosine Distance (diff)	Recency (diff)	Label	f(x)
1-1 vs 1-2	.2	1	1	**.88**
1-2 vs 1-3	.0	2	0	**0.4**
1-3 vs 1-1	-.3	3	1	**.09**
2-1 vs 2-2	.6	-2	0	**.04**
2-2 vs 2-3	-.2	0	1	**0.69**
2-3 vs 2-1	.0	2	0	**.22**

Instead of using standard gradient descent to minimize the loss (logistic regression), use Gradient Boosted Decision Trees (GBDT)

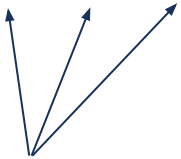
Learning to Rank with Implicit Feedback

- Our ranking problem is harder than Google's. We have labels for the data (clicks), but they are *explicitly biased.*
- Need a model that accounts for (a) relevance of results, and (b) positional bias, in a way that the two components can be pulled apart.
- Solution: Create a probabilistic model for clicks
  - P(click) = F(feat1, feat2, …, position) = F(feat1,feat2,...)G(position)

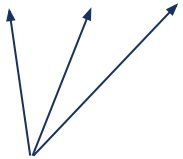
Example: Query/Click Data (KB Articles)

Goal: Find *ideal* relevance equation.
F = F(PC, PV, Luc)

queryId	1-PC	1-PV	1-Luc	2-PC	2-PV	2-Luc	3-PC	3-PV	4-Luc	4-PC	4-PV	5-Luc	5-PC	5-PV	5-Luc	clickRank
-10k6lm3qk	0	4	0.544	2	6	0.015	0	5	0.02	0	7	0.016	2	0	0.014	4
-10ycleuvj	0	0	9.664	0	0	9.368	0	0	5.465	0	0	5.461	0	0	5.038	1
-11dkusrix	2	7	7.247	5	5	2.505	0	0	4.555	0	0	4.545	0	0	4.505	1
-12zz6w1qa	4	5	1.178	4	6	0.937	0	0	1.976	0	0	1.961	3	5	0.927	1
-13cbi5bc9	4	7	15.405	0	5	3.016	0	0	3.602	0	0	3.438	3	5	0.514	1
-13y4wnbyx	0	5	6.209	5	8	3.518	6	8	2.783	0	0	6.091	0	0	5.956	1



# of child pages, # of page views, and lucene score, for the ***first*** record returned



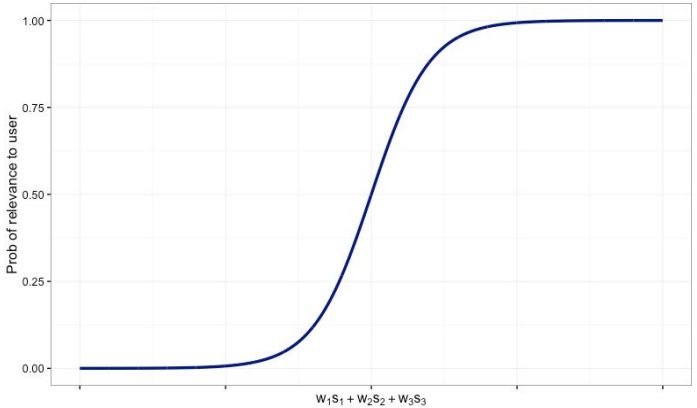
# of child pages, # of page views, and lucene score, for the ***third*** record returned

Position that was clicked

Probabilistic Model: Assumptions

1. The probability that a returned record is actually relevant to a given user, independent of the other results, is a logistic function of returned parameters.

s_1, s_2, s_3 = f(query, record).
We assume there is w_1, w_2, w_3, so that the picture looks like this.



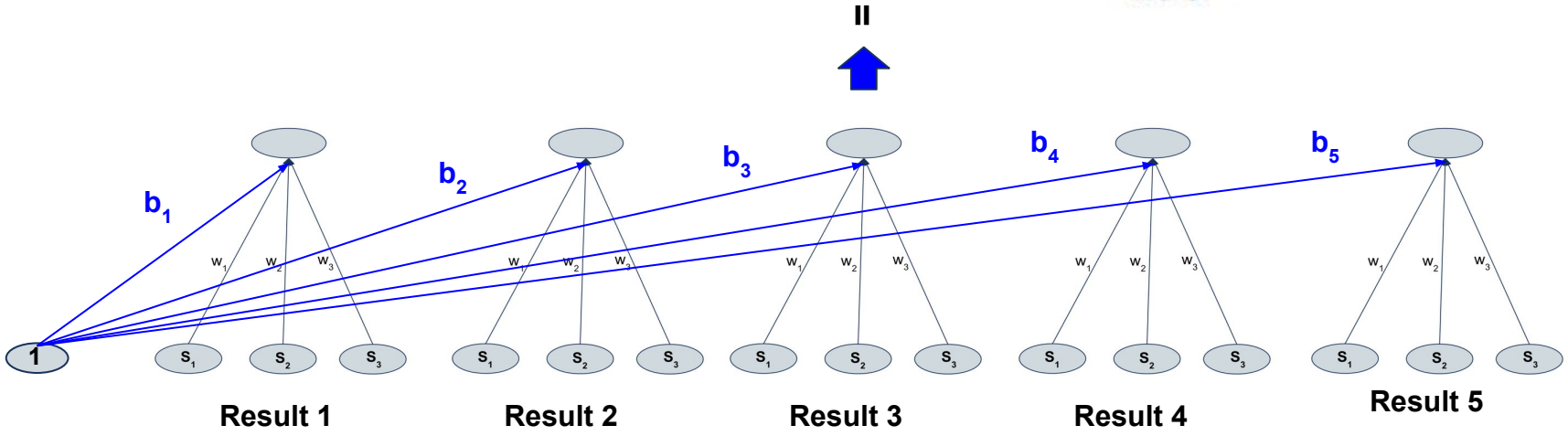
2. The *odds* of a record being clicked is the odds of the record being relevant, scaled by a constant positional bias factor

$$\frac{P(\text{click} = j)}{1 - P(\text{click} = j)} = b_j \frac{P(\text{j most relevant})}{1 - P(\text{j most relevant})}$$

That is, there is a linear scoring function such that the higher the score, the more likely the document is relevant to that query.

Probabilistic Model for Clicks

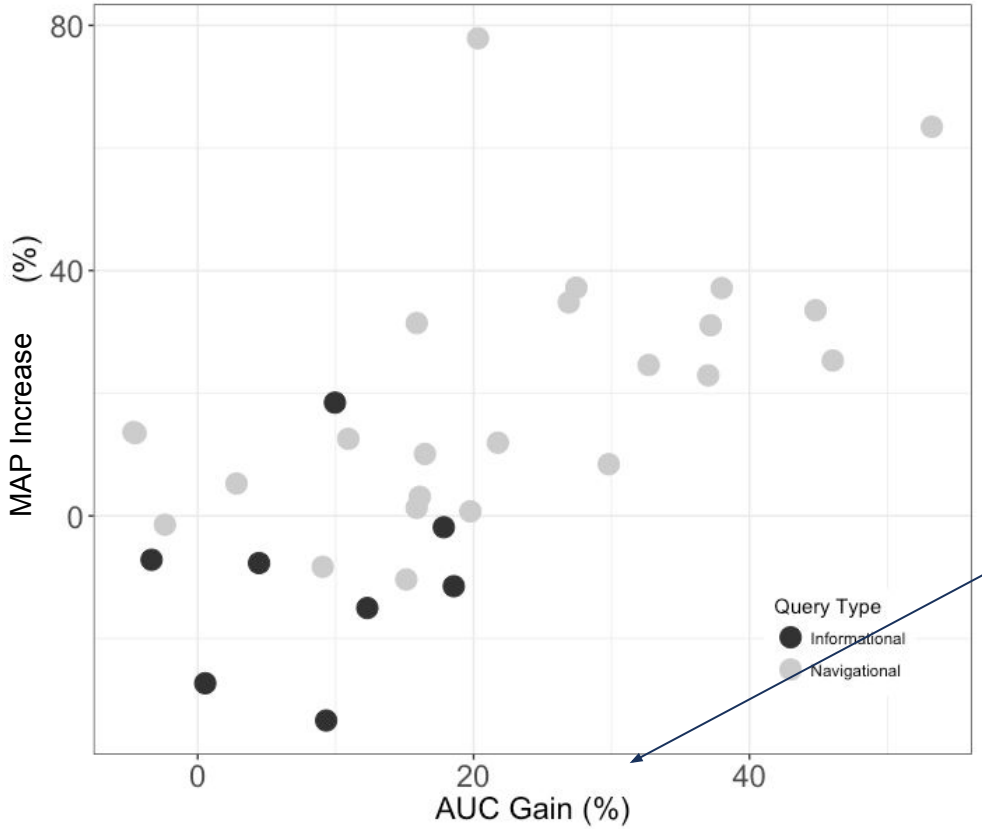
$$P(\text{click} = j \mid q, (R_1, \ldots, R_n); \mathbf{w}, (b_1, \ldots, b_n)) = \frac{e^{\mathbf{w}^T \mathbf{s}_j + b_j}}{\sum_{i=1}^{n} e^{\mathbf{w}^T \mathbf{s}_i + b_i}} = \hat{y}_j$$



Minimizing the cross entropy is equivalent to maximizing the likelihood of the observed click logs *according to our assumptions.*

$$\text{Loss}(q) := -\sum_{j=1}^{5} y_j \log(\hat{y}_j) - (1 - y_j) \log(1 - \hat{y}_j)$$

Evaluation (AB Tests)



Side-by-side performance of the new model and the old model in an AB test

Measure of how well our model is able to predict clicks *above and beyond the baseline* (click-through-rate by position)

thank you