Ranking

There are several different ranking problems in machine learning, among them "pairwise ranking" and "ordinal regression."

Consider this problem of movie ranking:

I'm a user on Netflix. For pairs of movies I've seen, I tell Netflix which of the two I liked better. Netflix has a database of features for each movie (like director, year, genre, Ebert's rating, other users ratings, etc). Using this database, Netflix wants to create a ranked list of movies for me to watch, from most-recommended to least-recommended.

This ↗ is a pairwise ranking problem. Formally,

Given: $\{x_i\}_{i=1 \dots m}$

"truth function" $\Pi : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$

$$\Pi(x_i, x_k) = \begin{cases} 1 & \text{if } x_i \text{ should be ranked above } x_k \\ 0 & \text{otherwise} \end{cases}$$

Want: Construct $f : \mathcal{X} \to \mathbb{R}$ such that for a new pair of randomly chosen examples $\bar{x}$, $\hat{x}$, if $\Pi(\bar{x}, \hat{x}) = 1$ then $f(\bar{x}) > f(\hat{x})$ with high prob. (i.e., $\bar{x}$ is ranked above $\hat{x}$).

Note that the pairwise setting is more general than if I had simply ranked all the Netflix movies I'd seen or given them a score.

T Consider instead the problem of ordinal regression:

I'm a user on Netflix. I give each movie a rating in $[0, 5]$. For a movie I haven't yet seen, Netflix wants to give me a prediction of my score for that movie, where score $\in \{0, 1, 2, 3, 4, 5\}$.

Basically, the idea is to put a decision boundary between 0 & 1, 1 & 2, 2 & 3, etc. In that sense this is similar to classification. Algorithms for ordinal regression are generalizations of classification algorithms,

I eg "PRank" - perceptron ranking, Crammer & Singer 02, Shashua & Levin 03, Chu & Ghahramani 05

T

More generally, "listwise" ranking:

I give Netflix a score for each movie I've seen. Netflix must produce a ranked list that is close to (in some sense) the ranked list I would have made if I had seen all the movies.

I

Example application: Natural Language Processing. Named Entity Recognition

Niagara Falls
Old Fort Niagara } historic place name
Niagara Mohawk ← company name
James Niagara Jr. ← person name
Niagara Falls, NY ← town name

For each occurrence of the word "Niagara" want to determine the type of named entity it refers to, usually as a ranked list of entity types. Large corpus of text used as training data.

This is an example of a <u>bipartite ranking</u> problem, since each entity type is either right or wrong.

<u>Bipartite ranking</u> (subset of pairwise ranking problems)

Given: $\{x_i\}_{i=1\cdots m}$ and $y_i \in \underbrace{\{-1, 1\}}_{\text{each movie is either good or bad}}$, construct $f: X \to \mathbb{R}$

such that $f(x_i) > f(x_k)$ where $y_i = 1$ & $y_k = -1$.

(The good movies are ranked higher than the bad movies.)

Why are classification & bipartite ranking different?

- both have $y \in \{-1, 1\}$, and both want to construct $f: X \to \mathbb{R}$ with (+)'s above (-)'s....
- but they are very different. Consider misclassification loss:

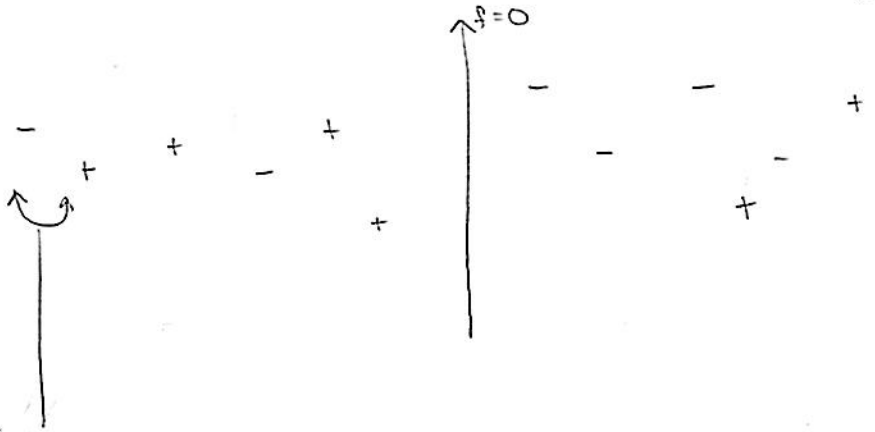
$$\#\text{misclassified examples} = \sum_{i=1}^{m} \mathbb{1}_{[\text{sign } f(x_i) \neq y_i]}$$

Consider bipartite misranking loss:

$$\#\text{misranked pos-neg pairs} = \sum_{\{i: y_i = 1\}} \sum_{\{k: y_k = -1\}} \mathbb{1}_{[f(x_i) \leq f(x_k)]}$$

$\{i: y_i = 1\}$: positive examples; $\{k: y_k = -1\}$: negative examples

$\mathbb{1}_{[f(x_i) \leq f(x_k)]}$: lose a point if x_i is ranked below x_k



If we swap these two examples, misclassification loss doesn't change, but misranking loss does!

Also, f is invariant to addition of a constant - there's no decision boundary for ranking.

Complexity of pairwise ranking is much higher: $(\#\text{pos})^*(\#\text{neg})$ compared to $\#\text{pos} + \#\text{negs}$ for classification.

The example above is a good way to show that SVM's do not rank!

Note: I'm simplifying these ranking problems very much. Many real-world issues such as (query, document) formulation for webpage ranking. Huge #'s of examples, so possibly do a 1st pass ranking/classification, cutoff, & use the learning alg to "re-rank".

Some measures of Ranking Quality (inspired by Eugene)

- Area Under the ROC curve ("AUC") which is in essence the Receiver Operator Characteristic

  misranking error for pairwise ranking, or Whitney-Mann-Wilcoxon statistic.

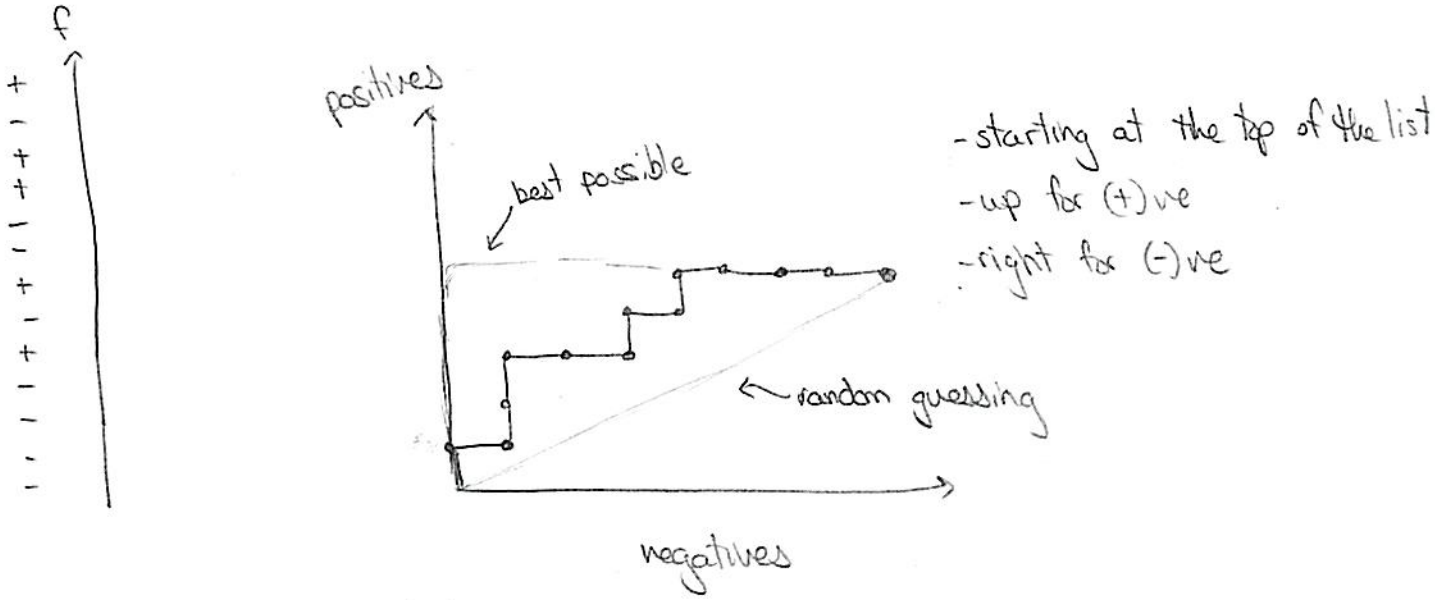
- precision, recall, F1 score, avg precision from information retrieval
- DCG - Discounted Cumulative Gain

The problem is that most of these measures are very complicated, so it is difficult to optimize them. RankBoost maximizes a proxy for the AUC (Schapire et al 02). Joachims & others (06-08) have been using "structured learning" to optimize other measures, but alg requires exponential constraints. (Cynthia has been using a weighted version of the AUC that concentrates at the top of the ranked list "p-norm push".)

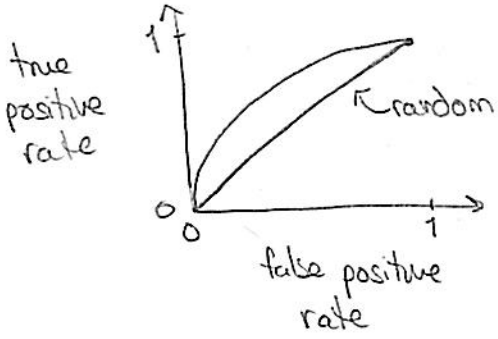
ROC Curves (Receiver Operator Characteristic)

- First used in WWII for analysis of radar signals. Following Pearl Harbor attack, US army began new research to increase the prediction of correctly detected Japanese aircraft from their radar signals.
- Used in medical research, particularly radiology.
- First applied to machine learning in 1989.
- ROC curve can be viewed as a graphical representation of a ranked list.

Simplest way to construct an ROC Curve for a ranked list.

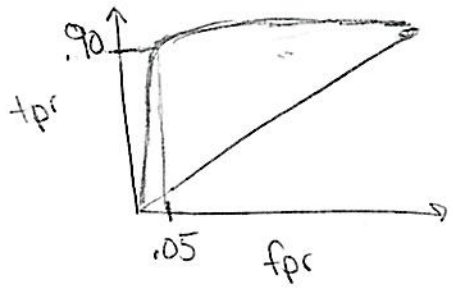


Most often, axes are rescaled to [0.1].



In this way, we can ask: When we place a decision boundary so that the true positive rate is 90% what is the false positive rate?

In the prenatal test for Downs Syndrome, if we insist on detecting 90% of Downs cases, what percent of false positive tests will we get? About 5%.



Note: the left of the ROC curve corresponds to the top of the ranked list.

Bipartite Misranking Error $= \sum_{\{i:\, y_i = 1\}} \sum_{\{k:\, y_k = -1\}} \mathbb{1}_{[f(x_i) \le f(x_k)]}$

$= |\#\text{pos}||\#\text{neg}|(1 - \text{AUC})$

$\propto$ "Wilcoxon-Mann-Whitney" statistic

General Misranking Error $= \sum_i \sum_k \mathbb{1}_{[f(x_i) \le f(x_k)]} \pi(x_i, x_k)$

(Close a point for every crucial pair that is misranked.)

RankBoost's Objective Function: use $\mathbb{1}_{[z \le 0]} \le e^{-z}$

misranking error $= \sum_i \sum_k \mathbb{1}_{[f(x_i) - f(x_k) \le 0]} \pi(x_i, x_k)$

$\le \sum_i \sum_k e^{-[f(x_i) - f(x_k)]} \pi(x_i, x_k) =: L_{RB}(f)$

Let $f(x) = \sum_j \lambda_j h_j(x)$ "weak rankers" or "rules of thumb" $h_j : \mathcal{X} \to \{0, 1\}$.

RankBoost minimizes $L_{RB}(\lambda) := \sum_i \sum_k e^{-\sum_j \lambda_j (h_j(x_i) - h_j(x_k))} \pi(x_i, x_k)$

for bipartite problem, i are the (+)ve examples, k are the (-)ve examples

these can be precomputed

Turns out that for binary weak rankers $h_j : \mathcal{X} \to \{0, 1\}$, AdaBoost minimizes $L_{RB}(\lambda)$ too. (R, Cortes, Mohri, Schapire '06, RS08).

Ranking Measures from Information Retrieval - These concentrate at the top of the list but are not often optimized directly. Meant for document retrieval & similar tasks.

Precision: fraction of documents returned that are relevant to the search.

prec@n : precision at cutoff rank n.

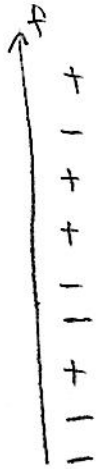
$y \in \{+1, -1\}$

+1 ↑ relevant, -1 ↑ irrelevant

Recall : fraction of the relevant documents that are successfully retrieved

recall@n : recall at cutoff rank n

Example:



prec@n	recall@n
1/1	1/4
1/2	1/4
2/3	2/4
3/4	3/4
3/5	3/4
3/6	3/4
4/7	4/4
4/8	4/4
4/9	4/4

Average Precision: mean of prec@n for relevant documents

$$1/4\,(1 + 2/3 + 3/4 + 4/7)$$

F1-score : $\dfrac{2 \cdot (\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}}$ (at a specified cutoff point)

Discounted Cumulative Gain (DCG) : Sum of $1/\log(1+\text{rank of relevant documents})$

$$1/\log(1+1) + 1/\log(1+3) + 1/\log(1+4) + 1/\log(1+7)$$

Mean Inverse Rank: sum of $1/(\text{rank of relevant documents})$

$$1/1 + 1/3 + 1/4 + 1/7$$

Which is the best? Paper in SIGIR 07 studied User satisfaction on Google, claims that user satisfaction is closest to precision & DCG. Studied 26 participants doing several queries each. ☺

True answer: It depends on the task!

Summary of this lecture:

You now (hopefully) know some interesting ranking problems in supervised machine learning, including pairwise ranking (where bipartite ranking is a special case) and ordinal regression.

You know that classification & bipartite ranking are fundamentally different.

You know how to interpret an ROC curve as a graphical representation of a ranked list.

Derivation of RankBoost's objective as a convex upper bound on misranking error

Definition of ranking measures from IR

Thanks!