# Learning to Rank

**Dr. Fayyaz ul Amir Afsar Minhas**
PIEAS Biomedical Informatics Research Lab
Department of Computer and Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad, Pakistan
http://faculty.pieas.edu.pk/fayyaz/

These slides have been adapted from Shivani Agarwal's tutorial on "Ranking Methods in Machine Learning"
https://www.siam.org/meetings/sdm10/tutorial1.pdf

# Ranking SVM

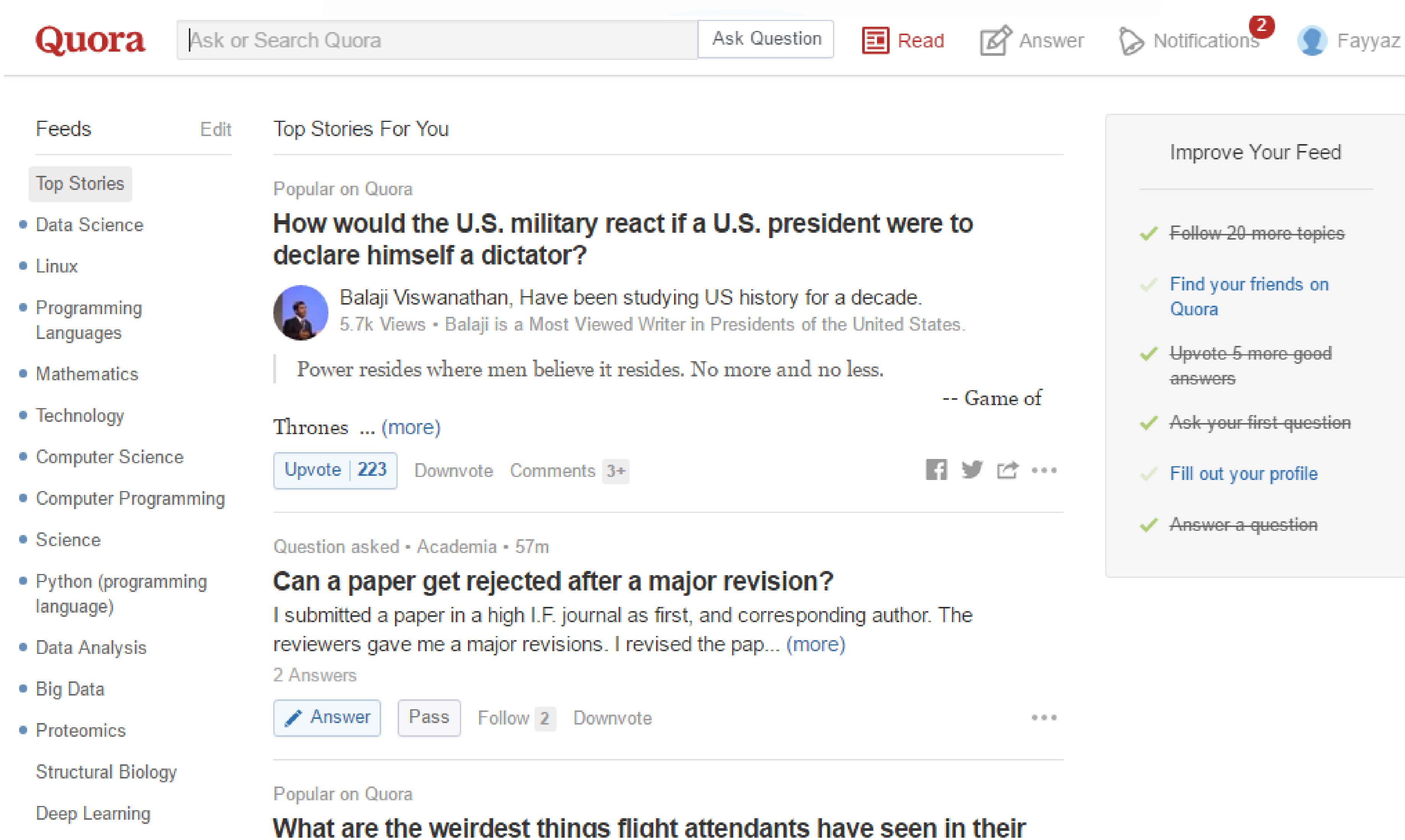
- **Assign a rank to an input example**
- **Apple grading**
  - **U.S. Extra Fancy**
  - **U.S. Fancy**
  - **U.S. No. 1**
  - **U.S. No. 1 Hail**
  - **U.S. Utility**

If $i$ is 'US Extra fancy' and $j$ is any other category then we want the score of $i$ to be higher than that of $j$, i.e.,:

$$f(x_i) \geq f(x_j) + 1 - \xi_{ij}$$

These are the constraints in a ranking SVM. The objective function minimizes the sum of slacks and maximizes the margin.

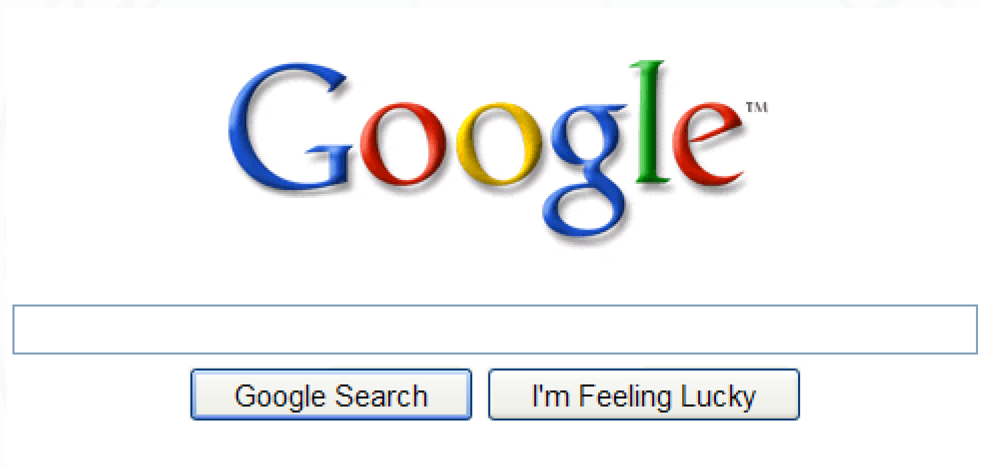
# Ranking in Recommendation Systems

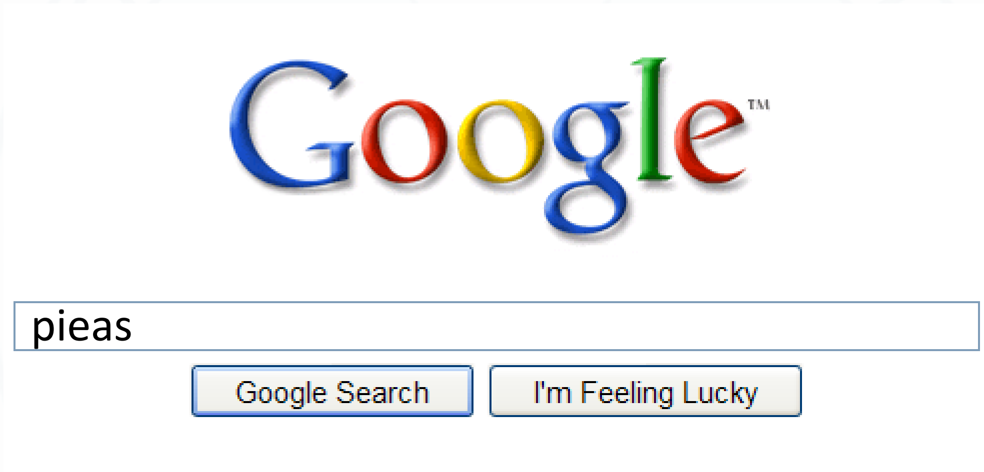


# Ranking in Recommendation Systems

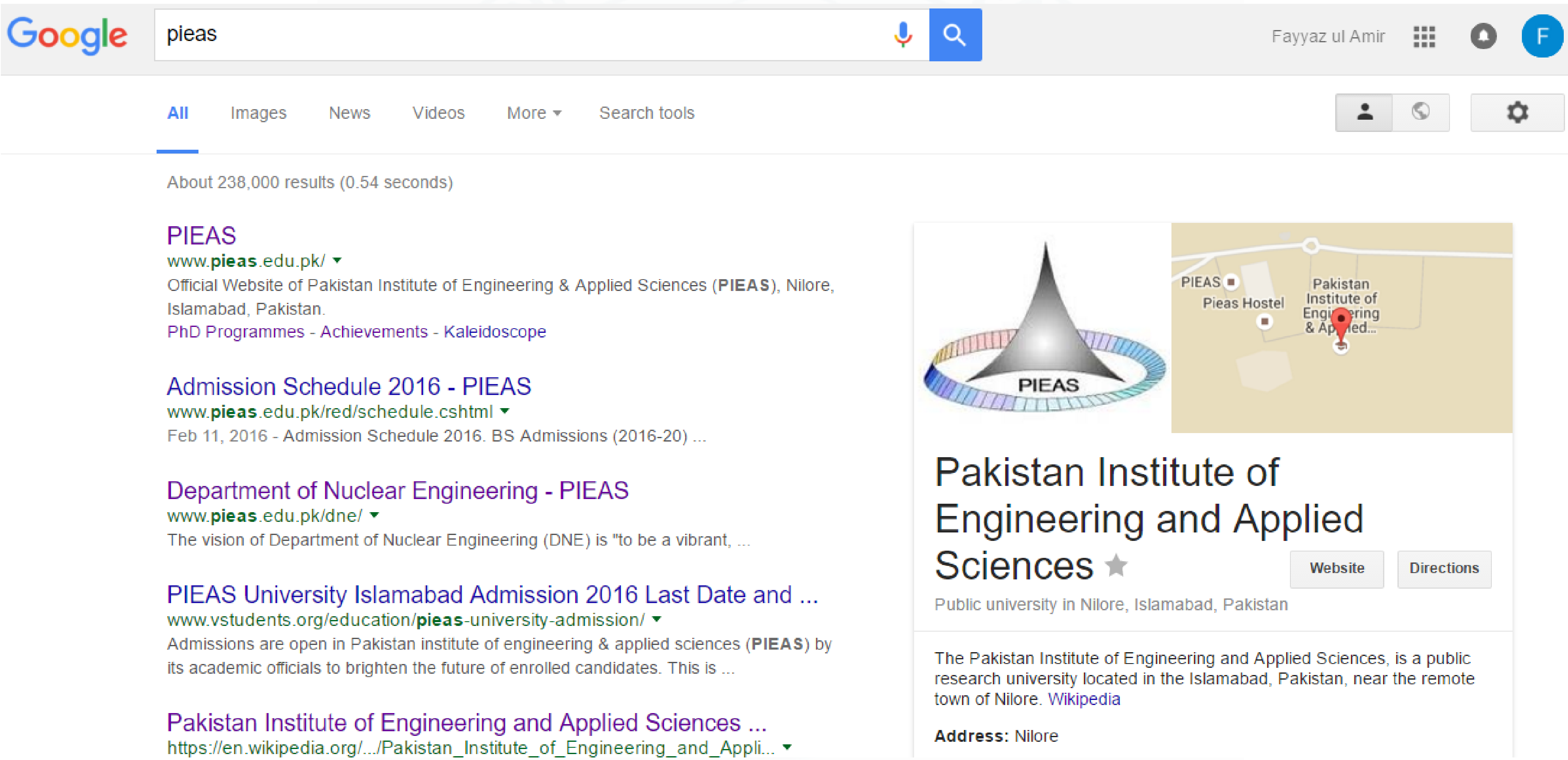

- Ranking of Facebook posts

# Ranking in Information Retrieval



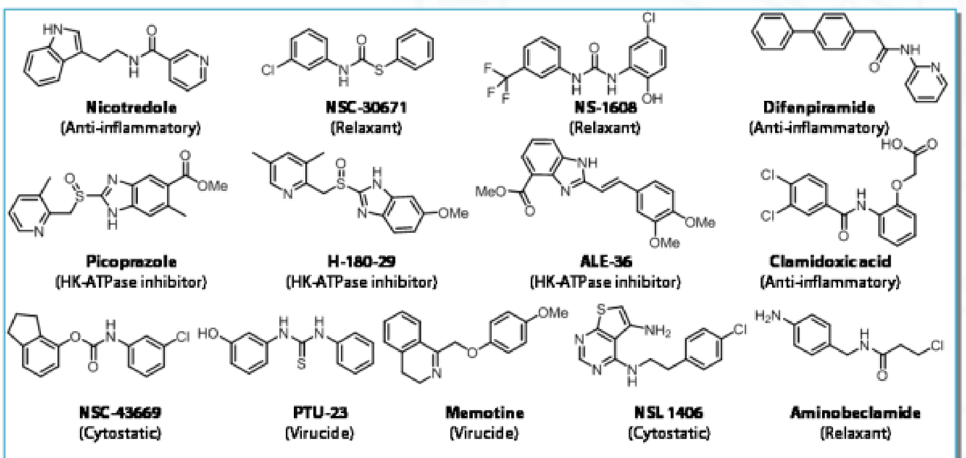

Google
pieas
Google Search
I'm Feeling Lucky

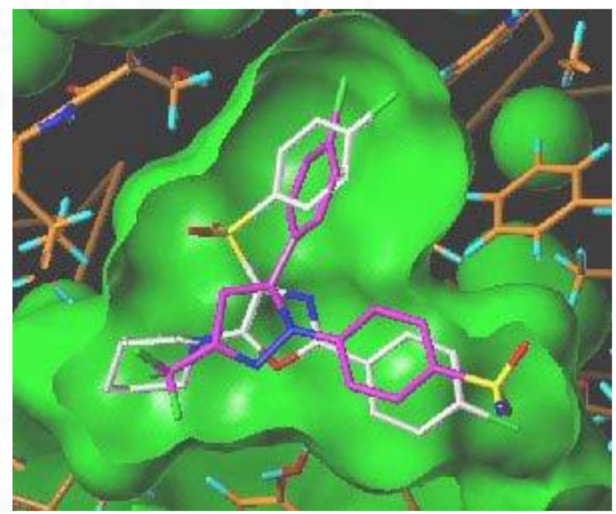
# Ranking in Information Retrieval



# Ranking in Drug Discovery

- Given: A large number of molecules from a chemical library
- Output: Which one is the most promising?



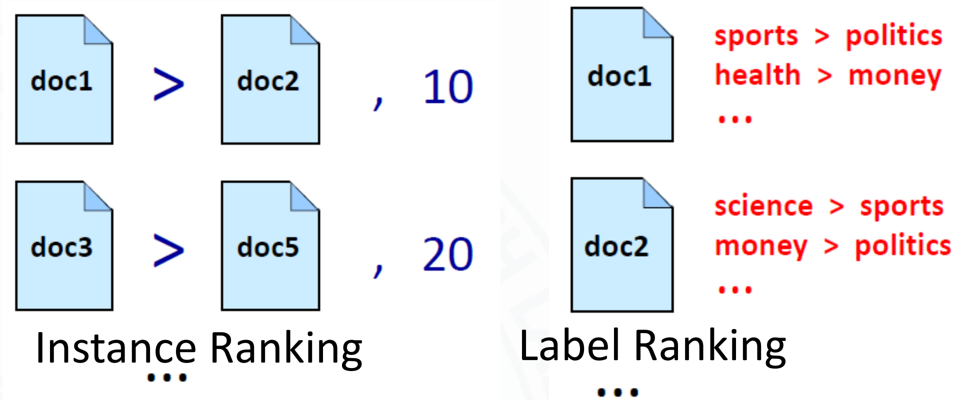



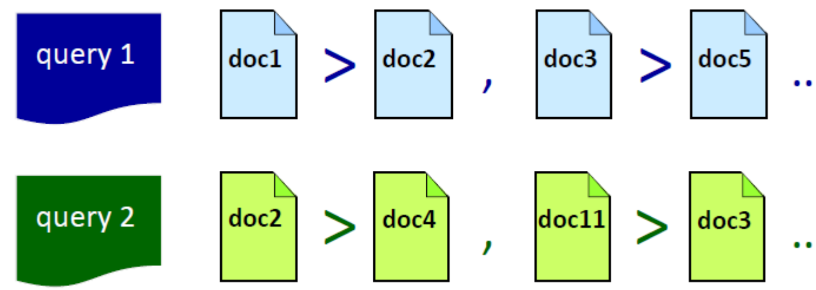
# Ranking in Bioinformatics

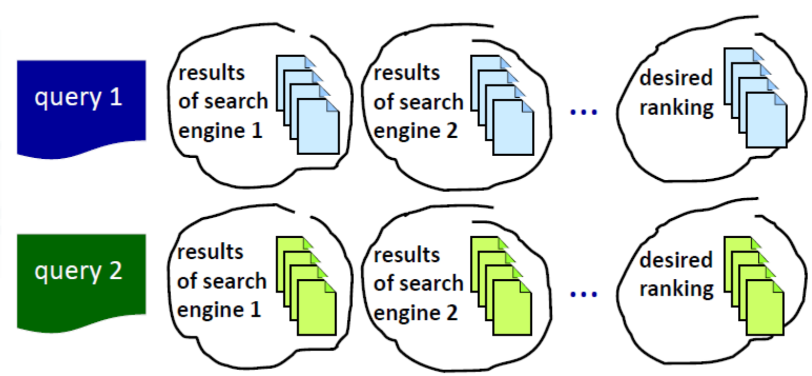
- Gene Discovery
  - Given: A set of genes and their activity in different conditions
  - Required output: What genes are important for a certain condition?

# Types of Ranking Problems

- Instance Ranking
- Label Ranking
- Subset Ranking
- Rank Aggregation



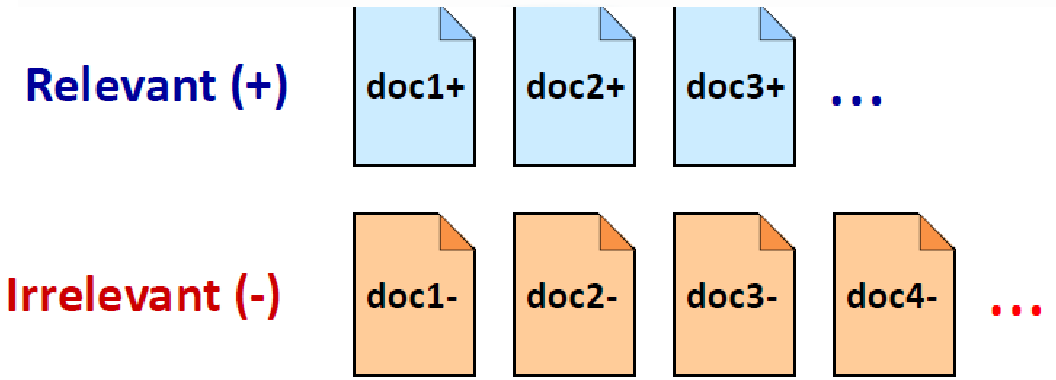




...

# What we will cover

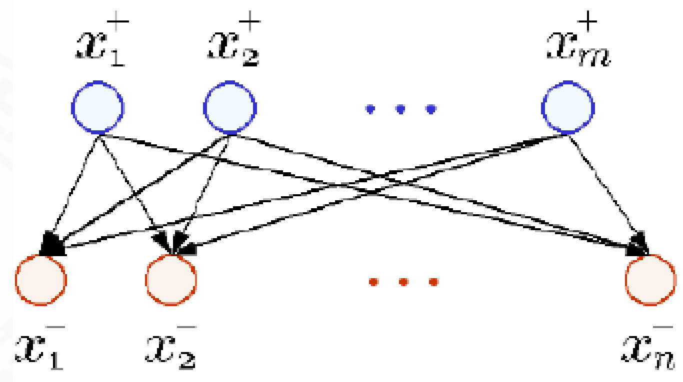
- Theory for Instance Ranking
  - Bipartite Ranking
  - K-partite Ranking
  - Ranking with Real-Valued Labels
  - General Instance Ranking
  - Properties of ranking methods
    - Ranking vs. Classification vs. Regression
    - AUC-ROC maximization
- Applications
  - Bioinformatics
  - Information Retrieval
- Further Reading and resources
  - Codes, etc.

# Bipartite Ranking




- Instance space: $X$
- Input: $S = \{S_+, S_-\}$
  - $S_+ = \{x_1^+, \dots, x_m^+\}$ (positive examples)
  - $S_- = \{x_1^-, \dots, x_n^-\}$ (negative examples)
- Output: Ranking function $f: X \rightarrow \mathbb{R}$
  - Isn't this the same as classification?
  - Or is it more closer to regression?

# Bipartite Ranking Function




- In classification we require
  - $f(x) > 0$ for positive examples
  - $f(x) < 0$ for negative examples
- In regression
  - $f(x)$ should be as close as possible to the corresponding output
- The ranking function should be such that
  - $f(x) > f(x')$ if instance $x$ should rank higher than $x'$
    - Any given positive instance should rank higher than any negative instance

# Ranking Error

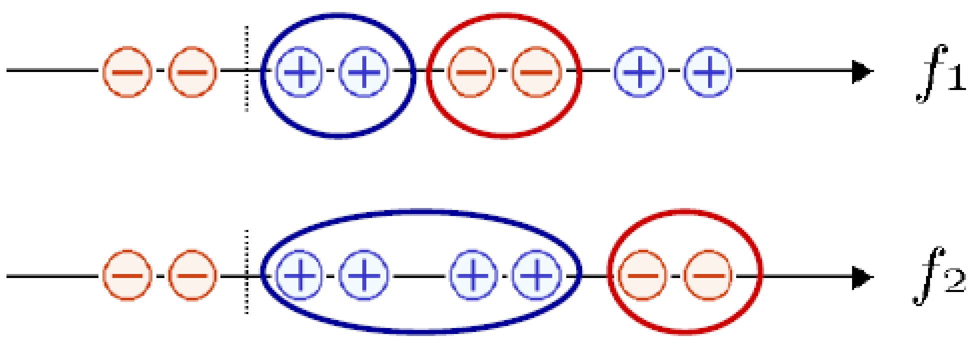
- If we have m positive instances and n negative instances, the number of possible ‘pairings’ becomes mn
  - Thus, the error for a given ranking function will be given by:

$$\widehat{\mathbf{er}}_S(f) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}(f(x_i^+) < f(x_j^-))$$

# Ranking Error

- What will be the classification and ranking errors of the following ranking functions?

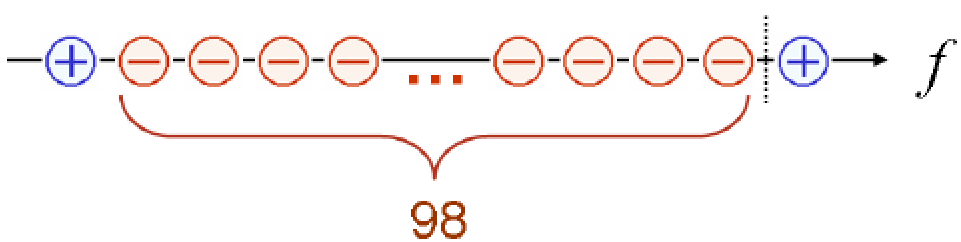
Example 1




Classification error = $\frac{1}{4}$
Ranking error = $\frac{1}{4}$

Classification error = $\frac{1}{4}$
Ranking error = $\frac{1}{2}$

Example 2




Classification error = $\frac{1}{100}$
Ranking error = $\frac{1}{2}$

# Ranking with Structured Risk Minimization

- SRM requires

$$f^* = argmin_f \underbrace{\{L(X,Y;f) + \lambda g(\|f\|)\}}_{\textbf{Structural Risk}}$$

Regularization Classifier Complexity (smoothing) term

Empirical Loss (or risk) term

$X, Y$ is the training data
$f$ is the learning function

- For ranking, we know the empirical error can be given by

$$\widehat{\mathbf{er}}_S(f) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}(f(x_i^+) < f(x_j^-))$$

# Ranking with SRM

- Find me a ranking function that minimizes the empirical error, possibly with some regularization, over some class of ranking functions
  - Examples: $f(x) = < w, x >$
- Mathematically

$$\min_{f \in \mathcal{F}} \left[ \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \ell(f, x_i^+, x_j^-) + \lambda N(f) \right]$$

| | | |
|---|---|---|
| $\ell(f, x_i^+, x_j^-)$ | : | convex upper bound on $\mathbf{1}(f(x_i^+) < f(x_j^-))$ |
| $N(f)$ | : | regularizer |
| $\lambda > 0$ | : | regularization parameter |
| $\mathcal{F}$ | : | class of ranking functions |

# Ranking with SRM

- What should be loss function?
  - The ideal loss function would be:
    - Zero-one loss $\mathbf{1}(f(x_i^+) < f(x_j^-))$
      - Incurs a loss of 1 only if the ranking is not proper
      - Otherwise the loss is zero
      - Problems?
        - » Non-convex
        - » Discontinous
    - Hinge-Loss

$$\ell_{\text{hinge}}(f, x_i^+, x_j^-) \;=\; \Big(1 - \big(f(x_i^+) - f(x_j^-)\big)\Big)_+ \quad \big[u_+ = \max(u, 0)\big]$$

# Bipartite RankSVM Algorithm

- Primal

$$min_w \, \lambda\|w\|^2 + \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} l\left(x_i^+, x_j^-; w\right)$$

- Alternatively,

$$min_{w,\xi>0} \, \lambda\|w\|^2 + \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \xi_{ij}$$

*Such that, for all i= 1…m and j = 1…n*

$$f(x_i) \geq f\left(x_j\right) + 1 - \xi_{ij}$$

# Bipartite RankSVM Algorithm

Introducing Lagrange variables in the primal

$$\{\alpha_{ij} : 1 \leq i \leq m, 1 \leq j \leq n\}$$

we can get the dual optimization problem shown below

$$\min_{\alpha} \left[\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n}\alpha_{ij}\alpha_{kl}\,\phi(x_i^+, x_j^-, x_k^+, x_l^-) - \sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_{ij}\right]$$

subject to $0 \leq \alpha_{ij} \leq C \quad \forall\, i, j$

where

$$\phi(x_i^+, x_j^-, x_k^+, x_l^-) = \left(K(x_i^+, x_k^+) - K(x_i^+, x_l^-) - K(x_j^-, x_k^+) + K(x_j^-, x_l^-)\right)$$

$$C = \frac{1}{\lambda mn}$$

Can be solved using a standard QP Solver or other methods (Chapelle & Keerthi, 2010)

# Other Algorithms

- RankBoost (Freund et al., 2003)
  - Combine outputs of multiple 'weak' ranking functions

$$\min_{f \in \mathcal{L}(\mathcal{F}_{\text{base}})} \left[ \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \ell_{\text{exp}}(f, x_i^+, x_j^-) \right]$$

$$\begin{aligned} \ell_{\text{exp}}(f, x_i^+, x_j^-) &= \text{exp}\left(-\left(f(x_i^+) - f(x_j^-)\right)\right) \\ \mathcal{L}(\mathcal{F}_{\text{base}}) &= \text{linear combinations of functions in some} \\ &\quad\ \text{base class } \mathcal{F}_{\text{base}} \end{aligned}$$

- RankNet (Burges et al, 2005): Neural network

$$\min_{f \in \mathcal{F}_{\text{neural}}} \left[ \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \ell_{\text{logistic}}(f, x_i^+, x_j^-) \right]$$

$$\begin{aligned} \ell_{\text{logistic}}(f, x_i^+, x_j^-) &= \log\left(1 + \text{exp}\left(-\left(f(x_i^+) - f(x_j^-)\right)\right)\right) \\ \mathcal{F}_{\text{neural}} &= \text{functions represented by some class of} \\ &\quad\ \text{neural networks} \end{aligned}$$

# K-partite ranking

Input: Training sample $S = (S_1, S_2, \ldots, S_k)$:

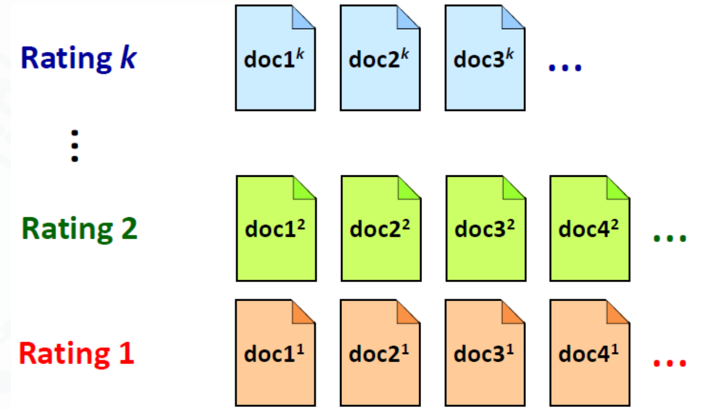
$$S_k = (x_1^k, \ldots, x_{n_k}^k) \in X^{n_k} \quad \text{(examples of rating } k\text{)}$$

$$\vdots$$

$$S_2 = (x_1^2, \ldots, x_{n_2}^2) \in X^{n_2} \quad \text{(examples of rating 2)}$$

$$S_1 = (x_1^1, \ldots, x_{n_1}^1) \in X^{n_1} \quad \text{(examples of rating 1)}$$

Output: Ranking function $f : X \to \mathbb{R}$




Empirical error:

$$\widehat{\mathbf{er}}_S(f) = \left( \frac{1}{\sum_{1 \le a < b \le k} n_a n_b} \right) \sum_{1 \le a < b \le k} \sum_{i=1}^{n_b} \sum_{j=1}^{n_a} (b - a)\, \mathbf{1}(f(x_i^b) < f(x_j^a))$$

# K-partite Ranking with SRM

Minimize a convex upper bound on the empirical ranking error, possibly with some regularization, over some class of ranking functions:

$$\min_{f \in \mathcal{F}} \left[ \left( \frac{1}{\sum_{1 \le a < b \le k} n_a n_b} \right) \sum_{1 \le a < b \le k} \sum_{i=1}^{n_b} \sum_{j=1}^{n_a} \ell(f, x_i^b, x_j^a, (b-a)) + \lambda N(f) \right]$$

where

$\ell(f, x_i^b, x_j^a, (b-a))$ : convex upper bound on $(b-a)\,\mathbf{1}(f(x_i^b) < f(x_j^a))$

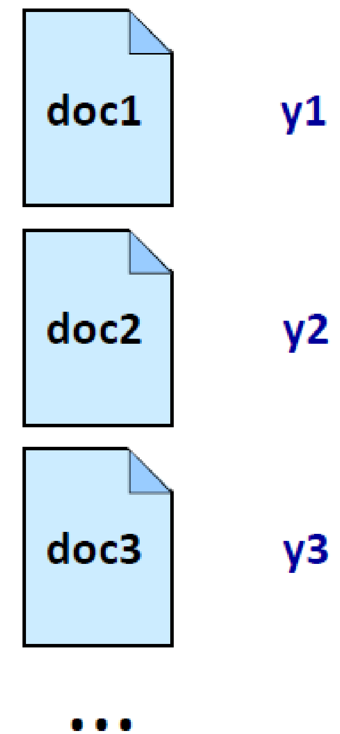
$N(f)$ : regularizer

$\lambda > 0$ : regularization parameter

$\mathcal{F}$ : class of ranking functions

# Ranking with Real-Valued Labels

- Instance space $X$
- Real-valued labels $Y = \mathbb{R}$
- Input: Training sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (X \times \mathbb{R})^m$
- Output: Ranking function $f : X \rightarrow \mathbb{R}$




- Empirical error:

$$\widehat{\mathbf{er}}_S(f) = \frac{1}{\binom{m}{2}} \sum_{1 \leq i < j \leq m} |y_i - y_j| \, \mathbf{1}\left((y_i - y_j)(f(x_i) - f(x_j)) < 0\right)$$

# Real-Valued Ranking with SRM

Minimize a convex upper bound on the empirical ranking error, possibly with some regularization, over some class of ranking functions:

$$\min_{f \in \mathcal{F}} \left[ \frac{1}{\binom{m}{2}} \sum_{1 \le i < j \le m} \ell(f, (x_i, y_i), (x_j, y_j)) + \lambda N(f) \right]$$

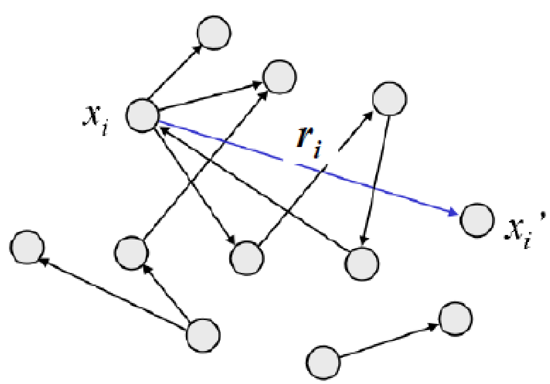
where

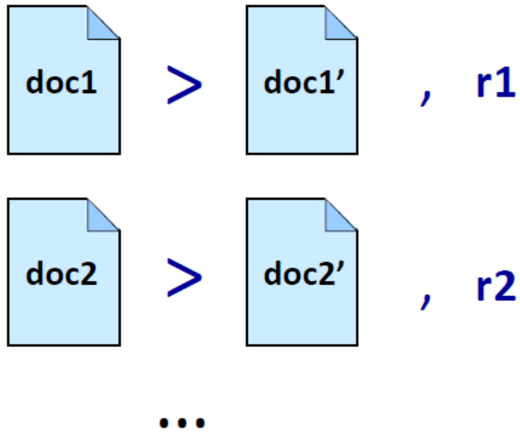
$$\begin{aligned}
\ell(f, (x_i, y_i), (x_j, y_j)) \quad &: \quad \text{convex upper bound on} \\
& \quad\ |y_i - y_j| \, \mathbf{1}\Big( (y_i - y_j)(f(x_i) - f(x_j)) < 0 \Big) \\
N(f) \quad &: \quad \text{regularizer} \\
\lambda > 0 \quad &: \quad \text{regularization parameter} \\
\mathcal{F} \quad &: \quad \text{class of ranking functions}
\end{aligned}$$

# Generalized Instance Ranking

- Each example is paired with another example and the difference between their ranks is given during training

- Instance space $X$
- Input: Training sample $S = ((x_1, x'_1, r_1), \ldots, (x_m, x'_m, r_m)) \in (X^2 \times \mathbb{R}_+)^m$
- Output: Ranking function $f : X \rightarrow \mathbb{R}$




- Empirical error: $\widehat{\mathbf{er}}_S(f) = \frac{1}{m} \sum_{i=1}^{m} r_i \, \mathbf{1}(f(x_i) < f(x'_i))$

# Generalized Instance Ranking with SRM

Minimize a convex upper bound on the empirical ranking error, possibly with some regularization, over some class of ranking functions:

$$\min_{f \in \mathcal{F}} \left[ \frac{1}{m} \sum_{i=1}^{m} \ell(f, x_i, x_i', r_i) + \lambda N(f) \right]$$

where

| | | |
|---|---|---|
| $\ell(f, x_i, x_i', r_i)$ | : | convex upper bound on $r_i \mathbf{1}(f(x_i) < f(x_i'))$ |
| $N(f)$ | : | regularizer |
| $\lambda > 0$ | : | regularization parameter |
| $\mathcal{F}$ | : | class of ranking functions |

# General RankSVM

$$\min_{f \in \mathcal{F}_K} \left[ \frac{1}{m} \sum_{i=1}^{m} \ell_{\text{hinge}}(f, x_i, x_i', r_i) + \frac{\lambda}{2} \|f\|_K^2 \right]$$

$$\ell_{\text{hinge}}(f, x_i, x_i', r_i) = \Big(r_i - \big(f(x_i) - f(x_i')\big)\Big)_+ \quad \left[u_+ = \max(u, 0)\right]$$

$\mathcal{F}_K$ = reproducing kernel Hilbert space (RKHS) with kernel function $K$

$$N(f) = \frac{\|f\|_K^2}{2}$$

**[Herbrich et al, 2000; Joachims, 2002]**

# SVMRank Tools

# $SVM^{rank}$

## Support Vector Machine for Ranking

Author: Thorsten Joachims <thorsten@joachims.org>
Cornell University
Department of Computer Science

Version: 1.00
Date: 21.03.2009

### Overview

*$SVM^{rank}$* is an instance of *$SVM^{struct}$* for efficiently training Ranking SVMs as defined in [Joachims, 2002c]. *$SVM^{rank}$* solves the same optimization problem as $SVM^{light}$ with the '-z p' option, but it is much faster. On the LETOR 3.0 dataset it takes about a second to train on any of the folds and datasets. The algorithm for solving the quadratic program is a straightforward extension of the ROC-area optimization algorithm described in [Joachims, 2006] for multiple rankings using the one-slack formulation of *$SVM^{struct}$*. However, since I did not want to spend more than an afternoon on coding *$SVM^{rank}$*, I only implemented a simple separation oracle that is quadratic in the number of items in each ranking (not the O[k*log k] separation oracle described in [Joachims, 2006]). While this makes the implementation not very suitable for the special case of ordinal regression [Herbrich et al, 1999], it means that it is nevertheless fast for small rankings (i.e. k<1000) and scales linearly in the number of rankings (i.e. queries).

### Source Code

The program is free for scientific use. Please contact me, if you are planning to use the software for commercial purposes. The software must not be further distributed without prior permission of the author. If you use $SVM^{rank}$ in your scientific work, please cite as

- T. Joachims, *Training Linear SVMs in Linear Time*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2006. [Postscript (gz)] [PDF]

The implementation was developed on Linux with gcc, but compiles also on Solaris, Cygwin, Windows (using MinGW) and Mac (after small modifications, see FAQ). The source code is available at the following location:

http://download.joachims.org/svm_rank/current/svm_rank.tar.gz

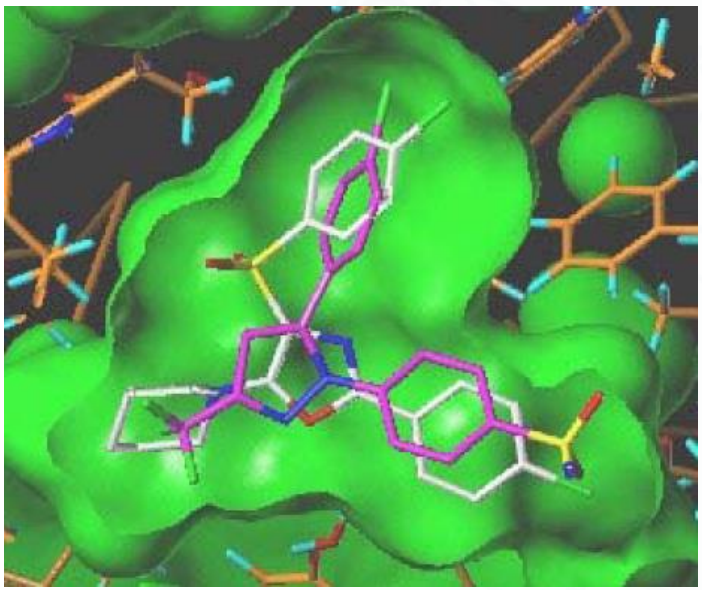
https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
https://sourceforge.net/p/lemur/wiki/RankLib/ (Java)
http://fa.bianp.net/blog/2012/learning-to-rank-with-scikit-learn-the-pairwise-transform/
http://faculty.pieas.edu.pk/fayyaz/studpro.html#pylemmings-a-software-suite-for-multiple-instance-learning

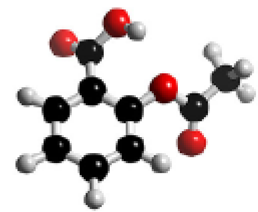
# Properties

- Ranking SVM maximizes the AUC score
  – Because it ensures that positive examples rank higher than negative examples
    - Ulf Brefeld, Tobias Scheffer. n.d. “AUC Maximizing Support Vector Learning.”
- Similar methods for maximizing Average Precision
  – Yue, Yisong, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. “A Support Vector Method for Optimizing Average Precision.” In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 271–78. SIGIR ’07. New York, NY, USA: ACM. doi:10.1145/1277741.1277790.

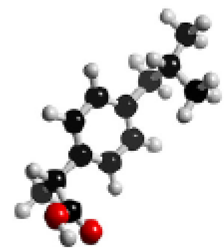
# Applications

- Drug Discovery
  - Input

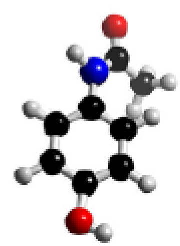




$pIC_{50} = 5.6718$



$pIC_{50} = 8.2991$



$pIC_{50} = 4.1317$

...

# Agarwal et al., 2010

- Dataset (from Sutherland et al., 2004)

| Data Set | No. of Compounds | No. of Chemical (2.5D) Descriptors | $pIC_{50}$ Values |
|---|---|---|---|
| DHFR inhibitors | 361 | 70 | 3.3 – 9.8 |
| COX2 inhibitors | 292 | 74 | 4.0 – 9.0 |

2.5D chemical descriptors
Gaussian kernel

| Training size | Ranking error | |
|---|---|---|
| | SVR | RankSVM |
| 24 | 0.4755 | **0.4601** |
| 48 | **0.3430** | 0.3509 |
| 72 | 0.2840 | **0.2726** |
| 96 | 0.2483 | **0.2351** |
| 120 | 0.2171 | **0.2121** |
| 144 | **0.2023** | 0.2032 |
| 168 | 0.2019 | **0.1817** |
| 192 | 0.1808 | **0.1749** |
| 216 | 0.1816 | **0.1722** |
| 237 | 0.1714 | **0.1681** |

FP2 molecular fingerprints
Tanimoto kernel

| Training size | Ranking error | |
|---|---|---|
| | SVR | RankSVM |
| 24 | 0.3793 | **0.3546** |
| 48 | 0.2905 | **0.2896** |
| 72 | 0.2517 | **0.2421** |
| 96 | 0.2343 | **0.2201** |
| 120 | 0.2147 | **0.2052** |
| 144 | 0.2166 | **0.1988** |
| 168 | 0.2096 | **0.1966** |
| 192 | 0.2056 | **0.1962** |
| 216 | 0.1907 | **0.1787** |
| 237 | 0.1924 | **0.1798** |

# Potential Application Areas

- Student Grades Prediction
- Binding Affinity Prediction
- Amyloidogenicity Prediction
- Document Ranking
- Remote Homology Detection
  - Liu, Bin, Junjie Chen, and Xiaolong Wang. 2015. "Application of Learning to Rank to Protein Remote Homology Detection." *Bioinformatics* 31 (21): 3492–98. doi:10.1093/bioinformatics/btv413.

# Issues

- The number of constraints becomes quadratic
- Efficient Algorithms Required

# Required Reading

- Yu, Hwanjo, and Sungchul Kim. 2012. “SVM Tutorial — Classification, Regression and Ranking.” In *Handbook of Natural Computing*, edited by Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, 479–506. Springer Berlin Heidelberg. http://link.springer.com/referenceworkentry/10.1007/978-3-540-92910-9_15.

- Yue, Yisong, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. “A Support Vector Method for Optimizing Average Precision.” In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 271–78. SIGIR ’07. New York, NY, USA: ACM. doi:10.1145/1277741.1277790.

We want to make a machine that will be proud of us.

- Danny Hillis