Ranking and Learning

293S UCSB, Tao Yang, 2020

Partially based on Manning, Raghavan, and Schütze‘s text book.

Table of Content

- **Weighted scoring for ranking**
  - Ranking Features
- **Learning to rank:**
  - A simple example
  - Generalization
  - Type of learning-to-rank methods
- **Learning to ranking as classification**
  - learning-to-rank strategies
  - Convert ranking as SVM based classification

Aspects of Ranking Marching User Intent

- **Relevance**
  - Documents need to be relevant to a user query.
- **Authoritativeness.**
  - High quality content is normally preferred since users rely on trustful information to learn or make a decision.
- **Freshness.**
  - Latest information is desired for time-sensitive queries.
- **Preference**
  - Personal or geographical preference can impact the choices

Weighted Scoring

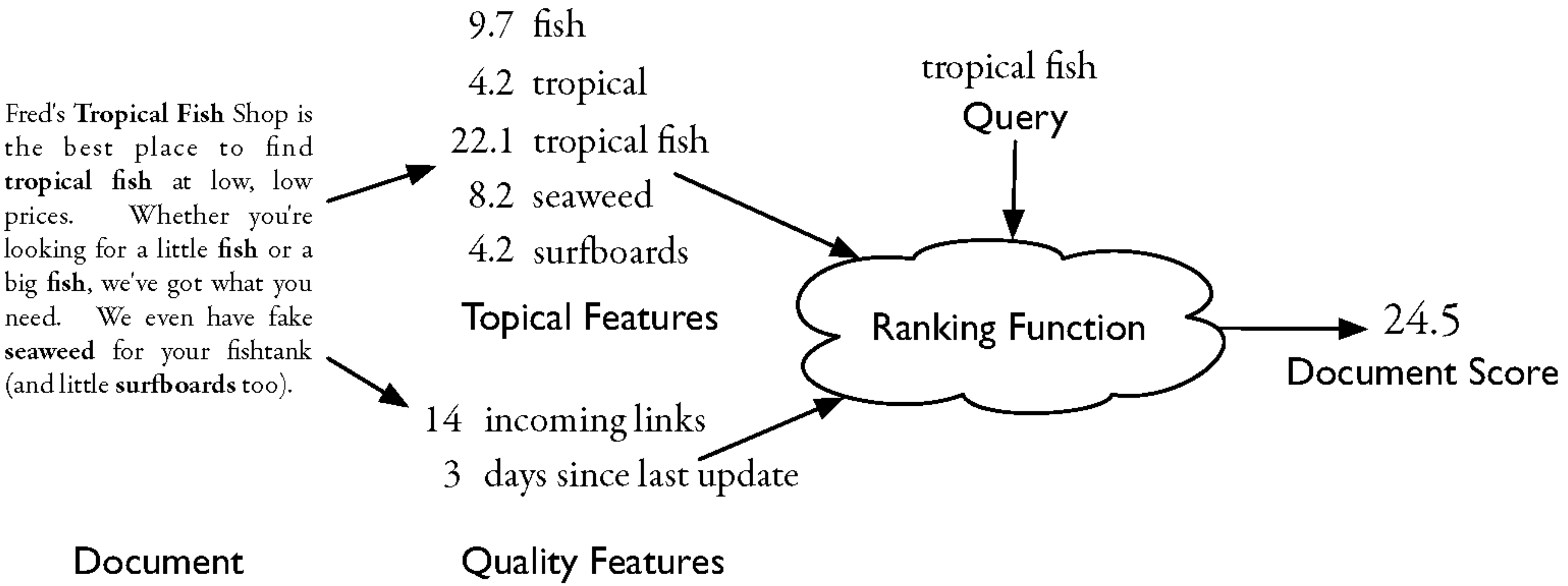
- **Scoring with weighted features**
  - Consider each document has subscores in each feature
  - Special case: Dot-product similarity of query and document
- **Example:**
  - **A simple weighted scoring method: use a linear combination of subscores:**
    - E.g.,

    Score = 0.6*< Title score> + 0.3*<Abstract score> + 0.1*<Body score>

Example with binary subscores
Query term “ucsb admission” appears in title, and ”ucsb” appears in body. Document score: (0.6 · 2) + (0.1 · 1) = 1.3.

Simple Model of Ranking with Similarity
[Croft, Metzler, Strohman's textbook slides]

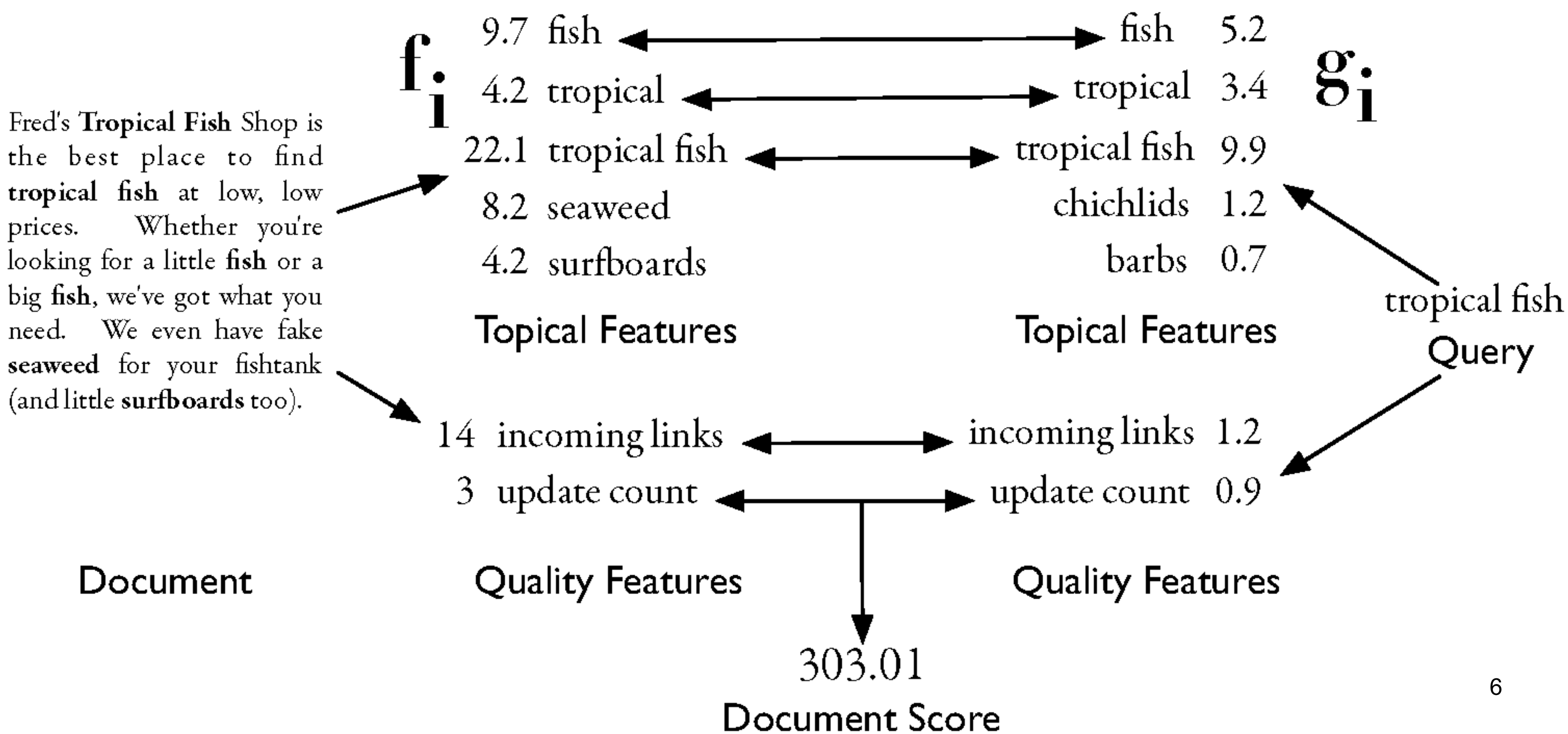
Document features are topical or quality-based



Simple Model of Ranking with Similarity
[Croft, Metzler, Strohman's textbook slides]

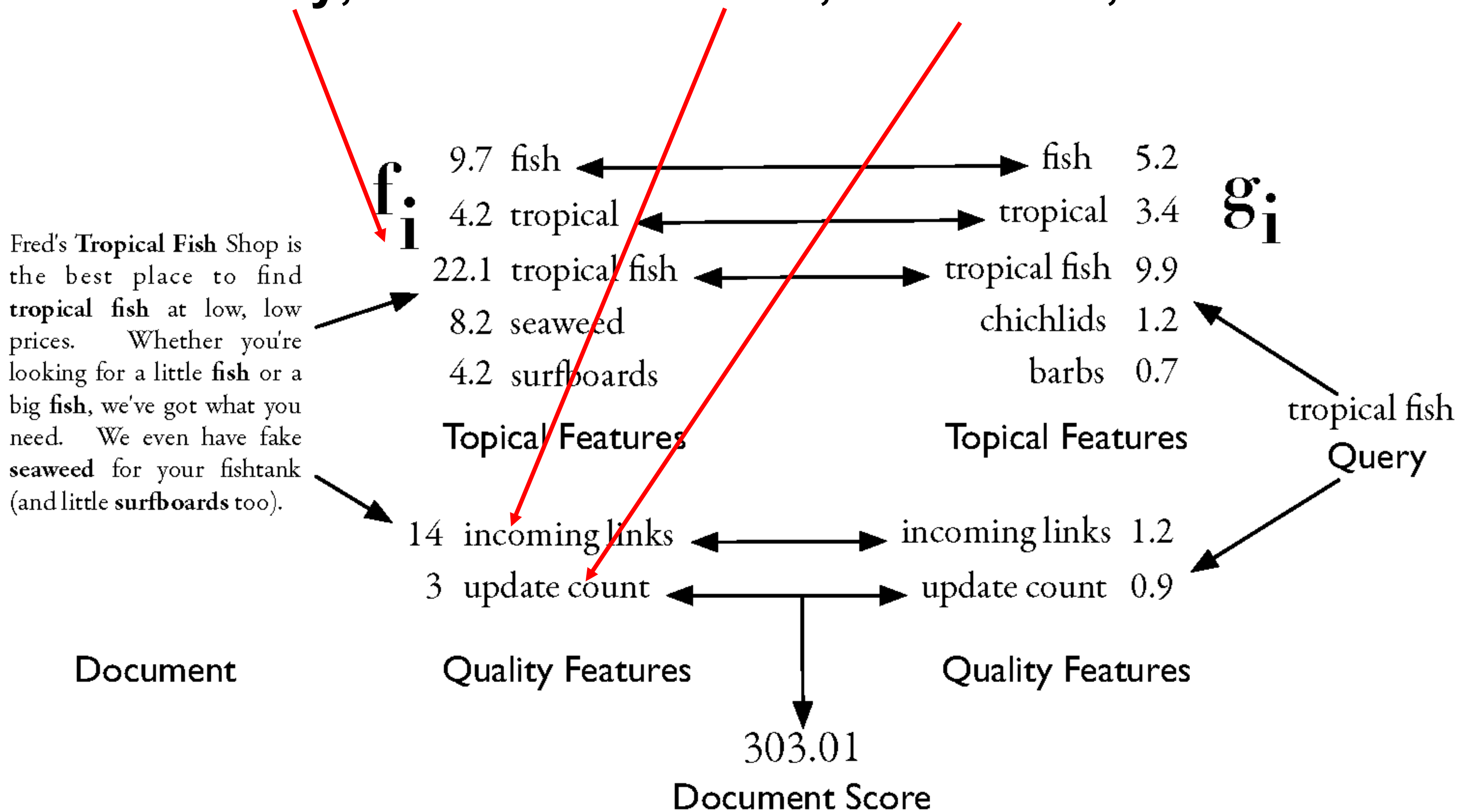
$$R(Q, D) = \sum_i g_i(Q) f_i(D)$$

f_i is a document feature function
g_i is a query feature function



Aspects of Ranking Marching User Intent

- **Relevancy, Authoritativeness, Freshness, Preference**



Ranking Features used in Web Search

- **Modern systems – especially on the Web – use a great number of features:**
  - Major web search engines use "hundreds" of such features – and they keep refinement
    - Text features: Query word frequency, Highlighted on page.
    - Document features: URL length, URL contains "~", Page length, Page freshness

- **Categories of ranking signals**
  - Query-dependent
  - Query-independent

Ranking Signals: Query Dependent

- **Text score**
  - Document text.
    - Text frequency: TFIDF, BM25
    - Text proxmity:
      - Closeness of keywords that appear in a document
      - Sum of $1/distance^2(w_1,w_2)$ for all keyword pairs
      - Query word span window
  - Anchor text
  - URL text
    - http://www.microsoft.com/en-us/download/

Ranking Signals: Query Dependent

- **Historical queries that yield document clicks**
  - www.marriott.com for mariott, marriot
- **Query classification and preference**
  - Local, commerical products, news, image, video
  - Geo-location
- **Link citation from documents that match the same query**
  - # citations from documents relevant to a query
  - Hub authority analysis

Ranking Signals: Query independent

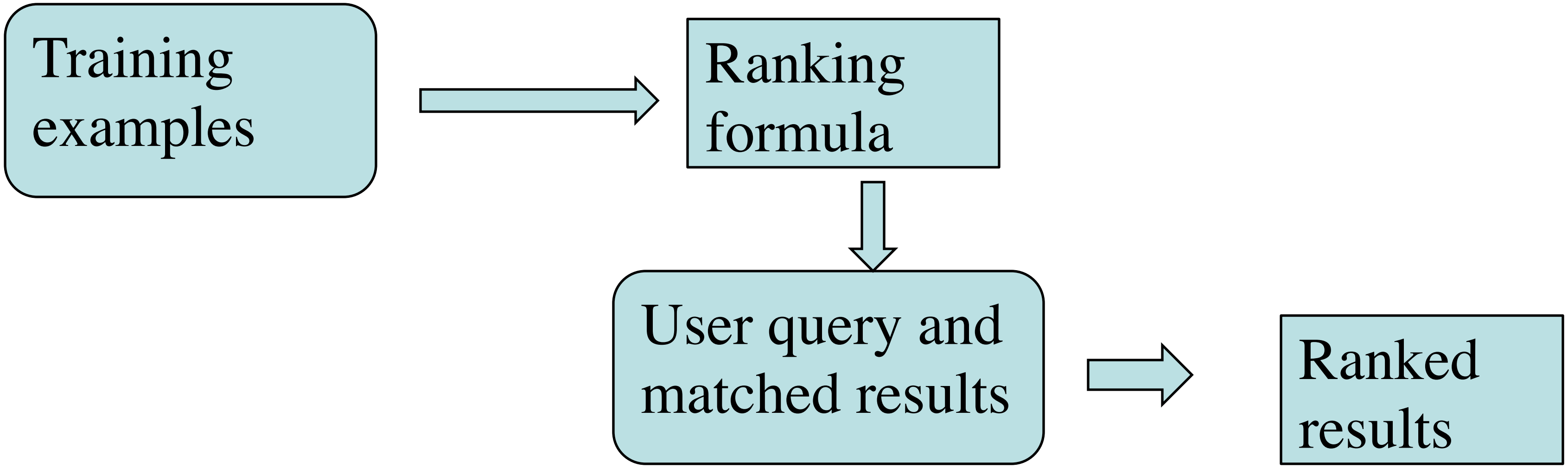
- **Document specific:**
  - Link analysis: Page Rank
    - #incoming links to a URL
  - Quality of documents:
    - Spam analysis
  - Page classification and properties
    - Geo location
    - Country/language classification
    - Homepage/personal page classification
  - Freshness
- **Site specific**
  - Site quality:
    - Well-known sites
  - Site classification: e.g. Country classification

Table of Content

- **Weighted scoring for ranking**
  - Ranking Features
- **Learning to rank:**
  - A simple example
  - Generalization
  - Type of learning-to-rank methods
- **Learning to ranking as classification**
  - learning-to-rank strategies
  - Convert ranking as SVM based classification

Machine learning for ranking

- **How do we combine these signals into a good ranker?**
  - How to derive weights if linear combination is used
  - What are other machine-learned models?
- **Learning to rank**
  - Learning from examples (called training data)



Learning weights: Methodology

- Given a set of **training examples**,
  - each contains (query *q*, document *d*, relevance score r).
  - r is relevance judgment for *d* on *q*
    - Simplest scheme
      - relevant (1) or nonrelevant (0)
    - More sophisticated: graded relevance judgments
      - 1 (bad), 2 (Fair), 3 (Good), 4 (Excellent), 5 (Perfect)

- Learn weights from these examples, so that the learned scores approximate the relevance judgments in the training examples

Simple example of learning-to-rank

- **Each doc has two zones, Title and Body**
- **For a chosen $w \in [0,1]$, score for doc d on query q**

$$score(d,q) = w \cdot s_T(d,q) + (1-w)s_B(d,q)$$

where:

$s_T(d, q) \in \{0,1\}$ is a Boolean denoting whether q matches the Title and

$s_B(d, q) \in \{0,1\}$ is a Boolean denoting whether q matches the Body

Examples of Training Data

Example	DocID	Query	s_T	s_B	Judgment
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Non-relevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Non-relevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_7	3191	driver	1	0	Non-relevant

From these 7 examples, learn the best value of w.

How?

- **For each example Φ_t we can compute the score based on** $score(d_t, q_t) = w \cdot s_T(d_t, q_t) + (1 - w)s_B(d_t, q_t)$.
- **We quantify Relevant as 1 and Non-relevant as 0**
- **Would like the choice of *w* to be such that the computed scores are as close to these 1/0 judgments as possible**
  - Denote by $r(d_t, q_t)$ the judgment for training instance Φ_t
- **Then minimize total squared regression error**

$$\sum_{\Phi_t} (r(d_t, q_t) - score(d_t, q_t))^2$$

Optimize the selection of weights

- **There are 4 kinds of training examples**
- **Thus only four possible values for score**
  - And only 8 possible values for error (relevant vs irrelevant)
- **Let n_{01r} be the number of training examples for which *title score* 0, *body score* 1, judgment = *Relevant*.**
- **Similarly define n_{00r}, n_{10r}, n_{11r}, n_{00i}, n_{01i}, n_{10i}, n_{11i}**

s_T	s_B	Score
0	0	0
0	1	$1-w$
1	0	w
1	1	1

Judgment=1 $\Rightarrow$ Error=w

Judgment=0 $\Rightarrow$ Error=1-w

Error:

$$[1-(1-\omega)]^2 n_{01r} + [0-(1-\omega)]^2 n_{01i}$$

Total error – then calculus

- **Add up contributions from various cases to get total error**

$$(n_{01r} + n_{10i})w^2 + (n_{10r} + n_{01i})(1 - w)^2 + n_{00r} + n_{11i}$$

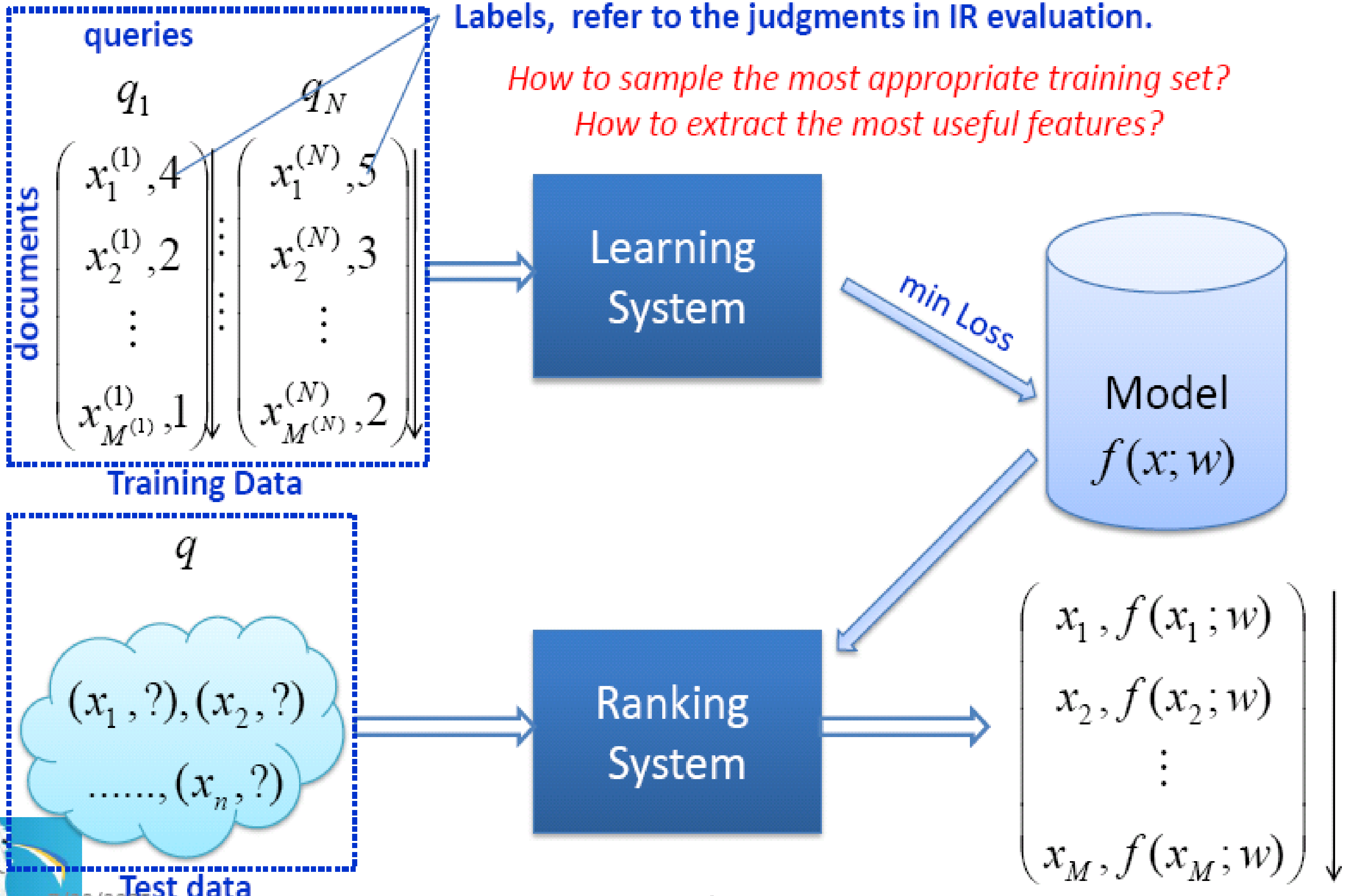
- **Now differentiate with respect to *w* to get optimal value of *w* as:**

$$\frac{n_{10r} + n_{01i}}{n_{10r} + n_{10i} + n_{01r} + n_{01i}}.$$

Generalizing this simple example

- **More (than 2) features**
- **Non-Boolean features**
  - What if the title contains some but not all query terms …
  - Categorical features (query terms occur in plain, boldface, italics, etc)
- **Scores are nonlinear combinations of features**
- **Multilevel relevance judgments (Perfect, Good, Fair, Bad, etc)**
- **Complex error functions**
- **Not always a unique, easily computable setting of score parameters**

Framework of Learning to Rank



Learning-based Web Search

- **Given features $x_1, x_2, \ldots, x_M$ for each document, learn a ranking function $f(x_1, x_2, \ldots, x_m)$ that minimizes the loss function L under a query**

- $f^* = \min L(f(x_1, x_2, \ldots, x_M), GroundTruth)$

- **Some related issues**
  - The functional space
    - linear/non-linear? continuous? Derivative?
  - The search strategy
  - The loss function

Table of Content

- **Weighted scoring for ranking**
  - Ranking Features
- **Learning to rank:**
  - A simple example
  - Generalization
- **Learning to ranking as classification**
  - Convert ranking as SVM based classification

Relationship to Classification Problem: An example

- **Collect a training corpus of (*q, d, r*) triples**
  - Relevance *r* is still binary for now
  - Document is represented by a feature vector
    - **x** = (α, ω)   α is cosine similarity, ω is minimum query window size
      - ω is the shortest text span that includes all query words (Query term proximity in the document)
- **Train a machine learning model to predict the class *r* of a document-query pair**

example	docID	query	cosine score	ω	judgment
Φ_1	37	linux operating system	0.032	3	*relevant*
Φ_2	37	penguin logo	0.02	4	*nonrelevant*
Φ_3	238	operating system	0.043	2	*relevant*
Φ_4	238	runtime environment	0.004	2	*nonrelevant*
Φ_5	1741	kernel layer	0.022	3	*relevant*
Φ_6	2094	device driver	0.03	2	*relevant*
Φ_7	3191	device driver	0.027	5	*nonrelevant*

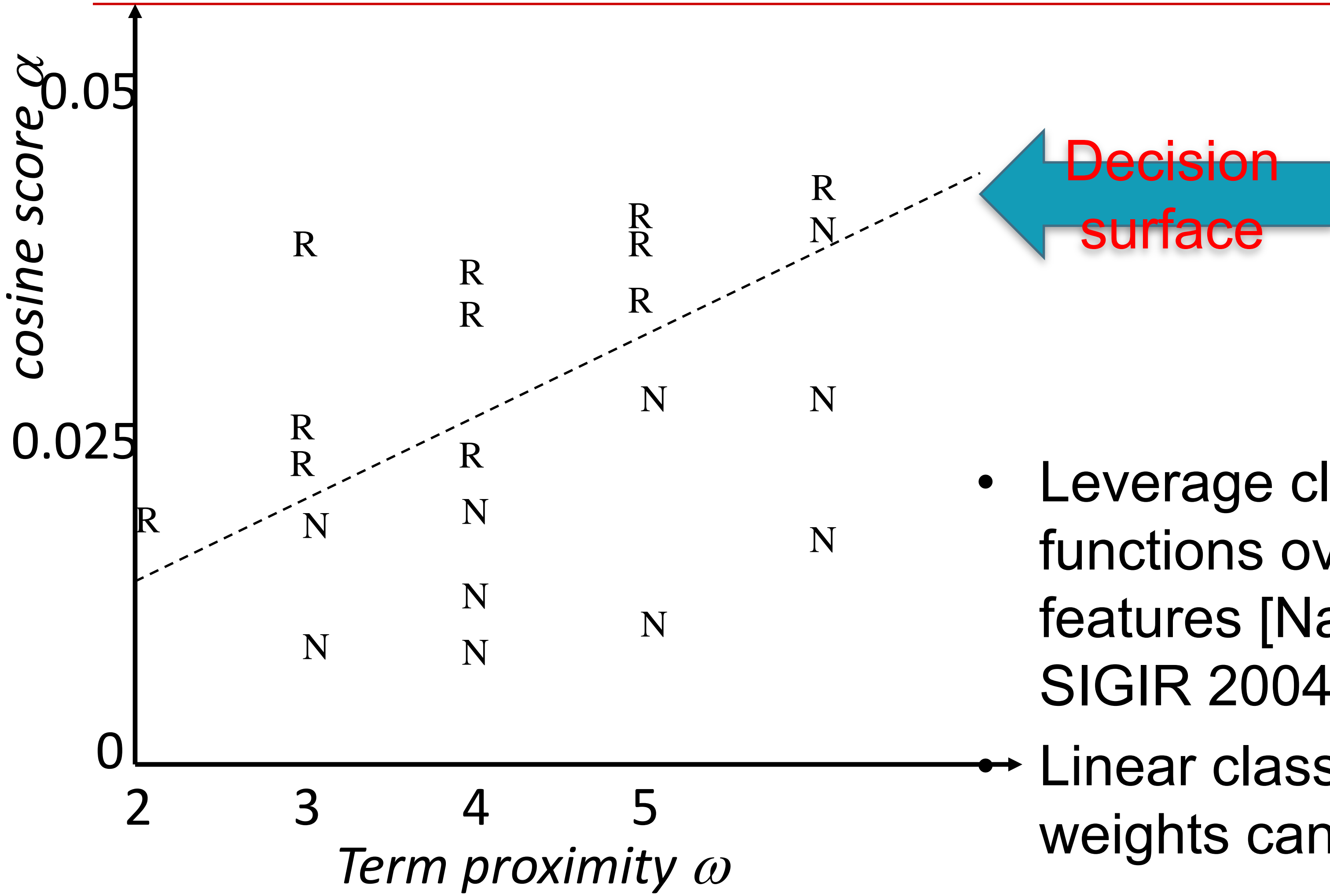
Window-based text span score

- **Query text span in a document** is the minimum length of word interval that covers all query words
- **Example document:**

  Fred's tropical fish shop is the best place to find tropical fish at low price

  - Span for query "tropical fish": 2
  - Span for query "Fred's fish shop": 4

Using classification for deciding relevance



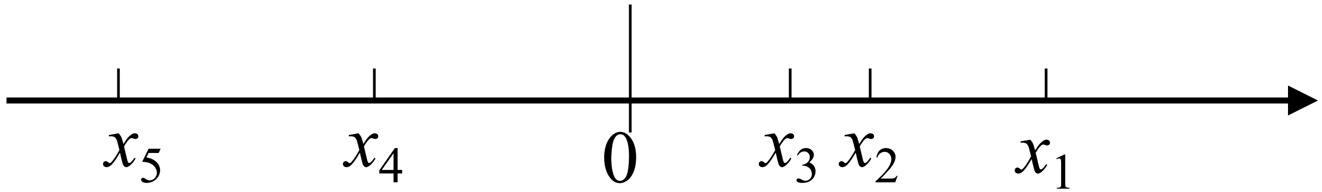
- Leverage classifier functions over many features [Nallapati SIGIR 2004]
- Linear classifier weights can be learned

A SVM classifier for relevance [Nallapati SIGIR 2004]

- Let $Score(d,q) = W \bullet Feature(d,q) + b$
  - *W is the weight vector*
  - *Feature(d,q) is the feature vector*
- Derive weights from the training examples:
  - want $Score(d,q) \leq -1$ for nonrelevant documents
  - $Score(d,q) \geq 1$ for relevant documents
- Testing:
  - decide relevant iff $Score(d,q) \geq 0$
- Train a classifier as the ranking function

Summary: Ranking vs. Classification

- **Classification**
  - Well studied: Bayesian, Neural network, Decision tree, SVM, Boosting, …
  - Training data: points: Positive: x1, x2, x3, Negative: x4, x5



- **Ranking**
  - Two ways to transform ranking problem to classification:
    1. Assign a document to a class (relevant/nonrelevant)

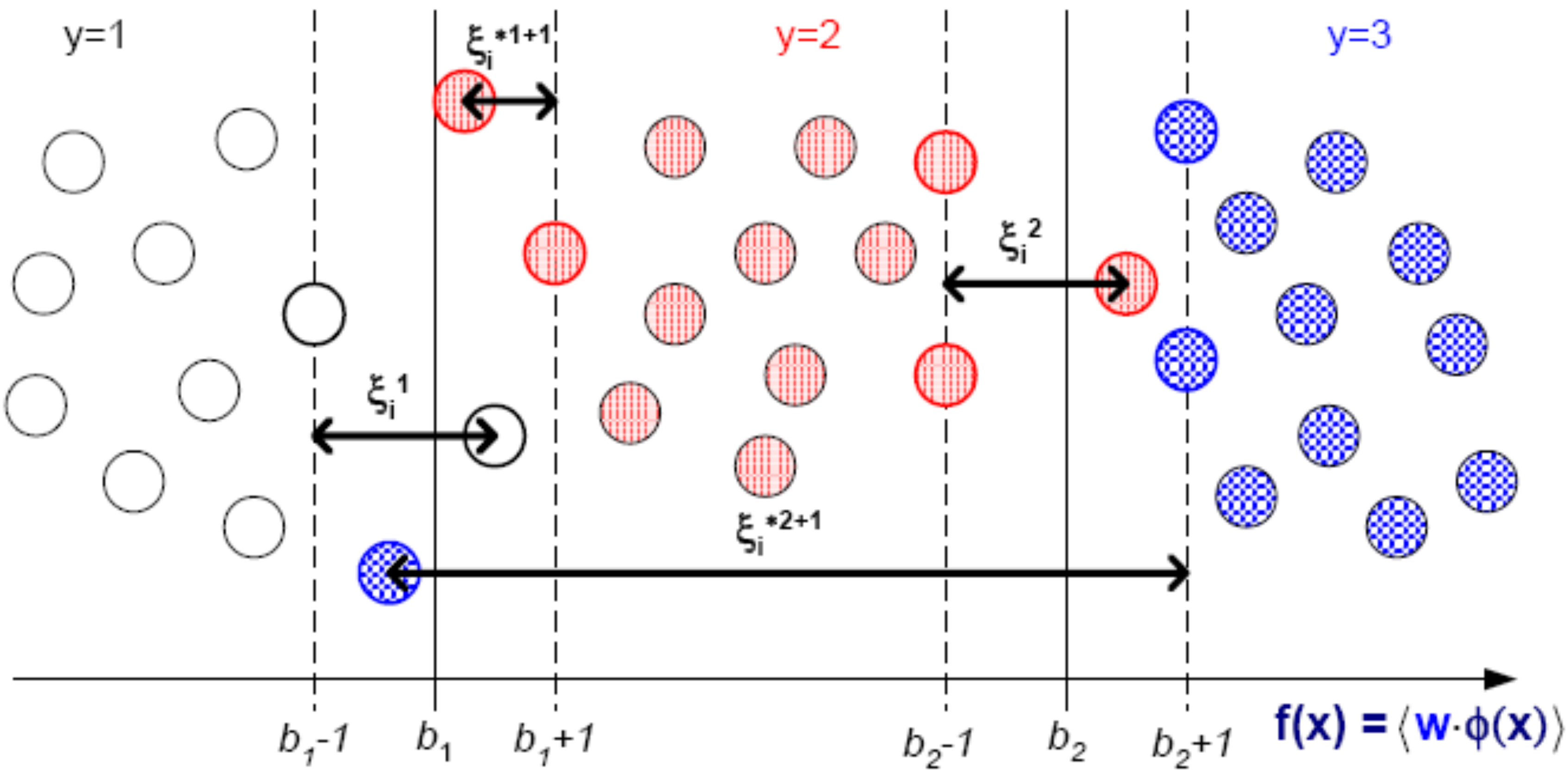
       Or assign to multiple classes such as perfect, excellent, good, fair, bad)
    2. Classify the relationship of two documents in answering a query

Strategies for "learning to rank"

- ***Point-wise* learning**
  - Given a query-document pair, predict a score (e.g. relevancy score)
    - Map f(x) to one of relevance vaules 0,1,2…
- ***Pair-wise* learning**
  - the input is a pair of results for a query, and the classification target is the relevance ordering relationship between them
  - Correct Order: $f(x_1) > f(x_2)$ if x_1 is more relevant than x_2
  - Otherwise incorrect.
- ***List-wise learning***
  - Directly optimize the ranking metric (e.g. NDCG) for each query with a list of ranked results

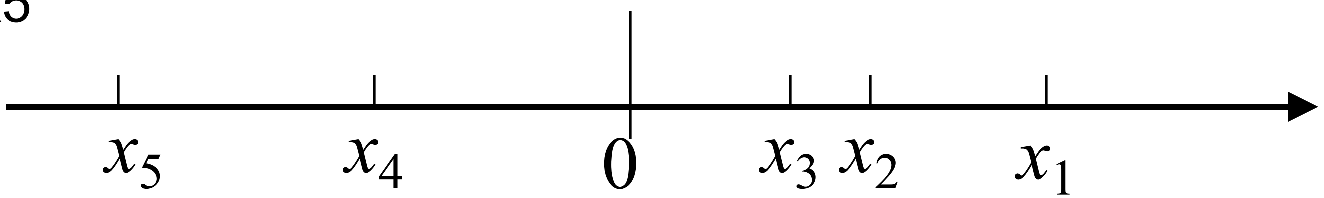
Point-wise learning: Example

- Goal is to learn a threshold to separate each rank
- Assume 3 relevance levels: 1, 2, 3



Pair-wise Learning

- A ranking should correctly classify the order of documents based on their relevance score:
  - Assume query q has matched documents ordered as x1, x2, x3, x4, x5



  - Correct order

(x1, x2), (x1, x3), (x1, x4), (x1, x5), (x2, x3), (x2, x4) …

  - Other orders are incorrect

Convert ranking into binary classification

Modified example for multi-class mapping with pair-wise learning

- **Collect a training corpus of (*q, d, r*) triples**
  - Relevance label *r* has 4 values
    - Perfect, Relevant, Weak, Nonrelevant
- **Train a machine learning model to predict the class *r* of a document-query pair**

example	docID	query	cosine score	ω	judgment
Φ_1	37	linux operating system	0.032	3	*Perfect*
Φ_2	37	penguin logo	0.02	4	*Nonrelevant*
Φ_3	238	operating system	0.043	2	*Relevant*
Φ_4	238	runtime environment	0.004	2	*Weak*
Φ_5	1741	kernel layer	0.022	3	*Relevant*
Φ_6	2094	device driver	0.03	2	*Perfect*
Φ_7	3191	device driver	0.027	5	*Nonrelevant*

The Ranking SVM : Pairwise Learning

[Herbrich et al. 1999, 2000; Joachims et al. KDD 2002]

- **Aim is to classify training instance pairs as**
  - correctly ranked
  - or incorrectly ranked
- **This turns an ordinal regression problem back into a binary classification problem**
- **We want a ranking function *f* such that c_i is ranked before c_k :**

$$c_i < c_k \text{ iff } f(\psi_i) > f(\psi_k)$$

- **Suppose that *f* is a linear function**

$$f(\psi_i) = \mathrm{w} \bullet \psi_i$$

- **Thus**

$$c_i < c_k \text{ iff } w(\psi_i - \psi_k) > 0$$

How many training examples formed for Ranking SVM?

example	docID	query	cosine score	ω	judgment
Φ_1	37	linux operating system	0.032	3	*Perfect*
Φ_2	37	penguin logo	0.02	4	*Nonrelevant*
Φ_3	238	operating system	0.043	2	*Relevant*
Φ_4	238	runtime environment	0.004	2	*Weak*
Φ_5	1741	kernel layer	0.022	3	*Relevant*
Φ_6	2094	device driver	0.03	2	*Perfect*
Φ_7	3191	device driver	0.027	5	*Nonrelevant*

1 training case:

Query: device driver Order: Doc 2094, 3192

How to derive (a, b, c) based on the training examples?

$$Score(d, q) = a\alpha + b\omega + c$$

$$Score(Doc\ 2094, q) \geq 1 + Score(Doc\ 3192, q)$$

$$0.03a + 2b + c \geq 1 + 0.027a + 5b + c$$

Ranking SVM

- **Training Set**
  - for each query q, we have a ranked list of documents totally ordered by a person for relevance to the query.
- **Features**
  - vector of features for each document/query pair

$$\psi_j = \psi(d_j, q)$$

  - feature differences for two documents d_i and d_j

$$\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$$

- **Classification**
  - Make the difference vector $\Phi(d_i, d_j, q)$ bigger than +1 if order is correct.
  - Otherwise less than −1.

Ranking SVM

OPTIMIZATION PROBLEM 1. (RANKING SVM)

$$\textit{minimize:} \quad V(\vec{w}, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum \xi_{i,j,k} \qquad (12)$$

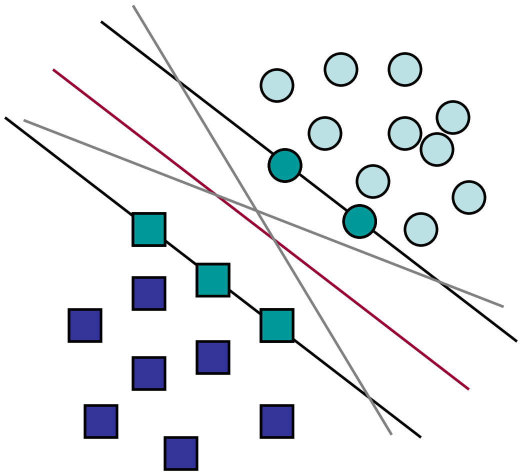
subject to:

$$\forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1}$$

$$\ldots \qquad (13)$$

$$\forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n}$$

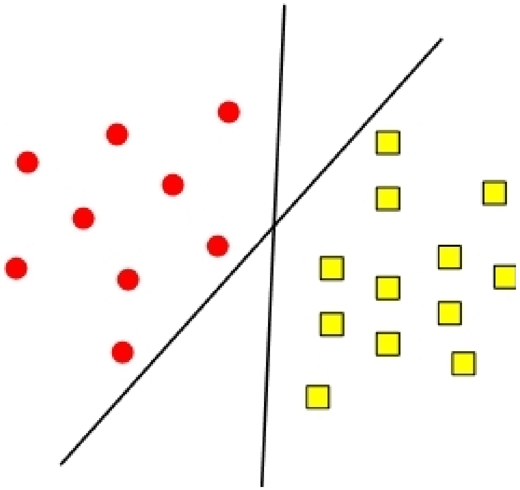
$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \qquad (14)$$



- Optimization problem is equivalent to that of a classification SVM on pairwise difference vectors $\Phi(q_k, d_i) - \Phi(q_k, d_j)$

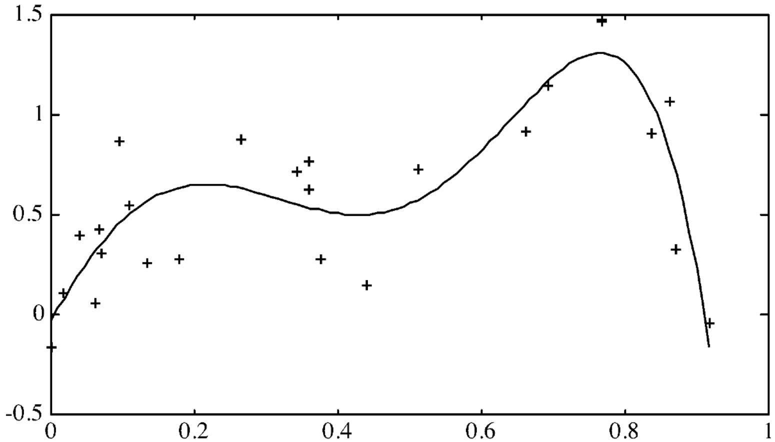
Classification vs. Regression

Classification



- Data in the form (x,y), where x is input vector. y is a category label
- Goal is to find indicator function estimation f .
- Loss: $$L(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}) \\ 1 & \text{if } y \neq f(\mathbf{x}) \end{cases}$$

Regression



- Data in the form (x,y), where x is input vector. y is real-valued output
- Goal is to find function estimation f.
- Example loss: $$L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$$

Classification vs. regression for learning to rank

- **Regression**
  - Find relative rank scores. E.g. Score = af_1+bf_2, what is weight a and b?
  - Not just classification labels.
- **Classification isn't the best model for rank score learning:**
  - Classification: Map to an unordered set of classes
  - Regression: Map to a real value
- **This regression formulation gives extra power:**
  - Relations between relevance levels are modeled
  - Fine grain scoring from highly relevant to irrelevant
  - Not an absolute scale of goodness

Summary

- **Weighted scoring for ranking**
  - Example: linear combination
  - Ranking features for web search
- **Learning to rank: A simple example**
  - Generalization to a general machine learning problem
- **Learning to ranking as classification**
  - Point-wise, pair-wise, & list-wise learning
    - Point-wise SVM classification
    - Pair-wise SVM classification
  - Classification vs. regression for ranking