Information Retrieval

INFO 4300 / CS 4300

- Retrieval models
  - Older models
    - » Boolean retrieval
    - » Vector Space model
  - Probabilistic Models
    - » BM25
    - » Language models
  - Web search
    - » Learning to Rank

Search Taxonomy

- ***Informational***
  - finding information about some topic which may be on one or more web pages
  - **Topical** search
- ***Navigational***
  - finding a particular web page that the user has either seen before or is assumed to exist
- ***Transactional***
  - finding a site where a task such as shopping or downloading music can be performed

[Broder, 2002]

Web Search

- For effective navigational and transactional search, need to combine features that reflect ***user relevance***
- Commercial web search engines combine evidence from hundreds of features to generate a ranking score for a web page
  - page content, page metadata, anchor text, links (e.g., PageRank), and user behavior (click logs)
  - page metadata – e.g., “age”, how often it is updated, the URL of the page, the domain name of its site, and the amount of text content

Search Engine Optimization

- ***SEO***: understanding the relative importance of features used in search and how they can be manipulated to obtain better search rankings for a web page
  - E.g., improve the text used in the title tag, improve the text in heading tags, make sure that the domain name and URL contain important keywords, and try to improve the anchor text and link structure
  - Some of these techniques are regarded as not appropriate by search engine companies

Web Search

- In TREC evaluations, the most effective features for navigational search are:
  - Text in the title, body, and heading (h1, h2, h3, and h4) parts of the document, the anchor text of all links pointing to the document, the PageRank number, and the inlink count
- Given size of Web, many pages will contain all query terms
  - Ranking algorithm focuses on discriminating between these pages
  - Word proximity is important, e.g. n-gram models

Machine Learning and IR

- Considerable interaction between these fields
  - Rocchio algorithm (60s) is a simple learning approach
  - 80s, 90s: learning ranking algorithms based on user feedback
  - 2000s: text categorization
- Limited by amount of training data
- Web query logs have generated new wave of research
  - e.g., "Learning to Rank"

Information Retrieval

INFO 4300 / CS 4300

- Retrieval models
  - Older models
    - » Boolean retrieval
    - » Vector Space model
  - Probabilistic Models
    - » BM25
    - » Language models
  - Web search
    - » Learning to Rank

Generative vs. Discriminative

- All of the probabilistic retrieval models presented so far fall into the category of *generative models*
  - A ***generative model*** assumes that documents were generated from some underlying model (in this case, usually a multinomial distribution) and uses training data to estimate the parameters of the model
  - Probability of belonging to a class (i.e. the relevant documents for a query) is then estimated using Bayes' Rule and the document model

Generative vs. Discriminative

- A ***discriminative*** model estimates the probability of belonging to a class directly from the observed features of the document based on the training data
- Generative models perform well with low numbers of training examples
- Discriminative models usually have the advantage given enough training data
  - Can also easily incorporate many features

Discriminative Models for IR

- Discriminative models can be trained using explicit relevance judgments or click data in query logs
  - Click data is much cheaper, more noisy
  - e.g. Ranking Support Vector Machine (SVM) takes as input ***partial rank*** information for queries
    - » partial information about which documents should be ranked higher than others

Ranking SVM

- Training data is

$$(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)$$

  - r is partial rank information

Example

1. **Kernel Machines**
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. **SVM-Light Support Vector Machine**
   http://ais.gmd.de/~thorsten/svm_light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel Methods References
   http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
   http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html
7. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
   http://www.support-vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
    http://www.cs.wisc.edu/dmi/lsvm

Ranking SVM

- Training data is

$$(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)$$

  - r is partial rank information
    - » if document d_a should be ranked higher than d_b, then $(d_a, d_b) \in r_i$
  - partial rank information comes from relevance judgments (allows multiple levels of relevance) or click data
    - » e.g., d_1, d_2 and d_3 are the documents in the first, second and third rank of the search output, only d_3 clicked on → (d_3, d_1) and (d_3, d_2) will be in desired ranking for this query

Ranking SVM

- Learning a linear ranking function $\vec{w}.\vec{d_a}$
  - where *w* is a weight vector that is adjusted by learning
  - d_a is the vector representation of the features of document
  - *non-linear* functions also possible
- Weights represent importance of features
  - learned using training data
  - e.g.,

$$\vec{w}.\vec{d} = (2,1,2).(2,4,1) = 2.2 + 1.4 + 2.1 = 10$$

Ranking SVM

- Learn *w* that satisfies as many of the following conditions as possible:

$$\begin{aligned} \forall (d_i, d_j) \in r_1 \quad &: \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} \\ &\ldots \\ \forall (d_i, d_j) \in r_n \quad &: \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} \end{aligned}$$

- Can be formulated as an ***optimization*** problem

Ranking SVM

$$\begin{aligned} &minimize: && \frac{1}{2}\vec{w}.\vec{w} + C\sum \xi_{i,j,k} \\ &subject\ to: && \\ \forall (d_i, d_j) \in r_1 \quad &: && \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,1} \\ &\ldots && \\ \forall (d_i, d_j) \in r_n \quad &: && \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,n} \\ &&& \forall i \forall j \forall k : \xi_i, j, k \geq 0 \end{aligned}$$

  - ξ, known as a slack variable, allows for misclassification of difficult or noisy training examples, and *C* is a parameter that is used to prevent overfitting

Ranking SVM

- Software available to do optimization
- Each pair of documents in our training data can be represented by the vector:

$$(\vec{d_i} - \vec{d_j})$$

- Score for this pair is:

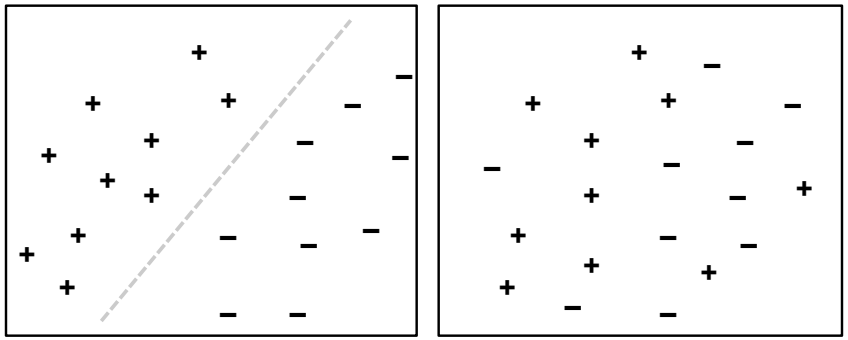
$$\vec{w}.(\vec{d_i} - \vec{d_j})$$

- SVM classifier will find a *w* that makes the smallest score as large as possible
  - make the differences in scores as large as possible for the pairs of documents that are hardest to rank

Support Vector Machines

- Based on geometric principles
- Given a set of inputs labeled '+' and '-', find the "best" hyperplane that separates the '+'s and '-'s
- Questions
  - How is "best" defined?
  - What if no hyperplane exists such that the '+'s and '-'s can be perfectly separated?

Separable vs. Non-Separable Data



Separable

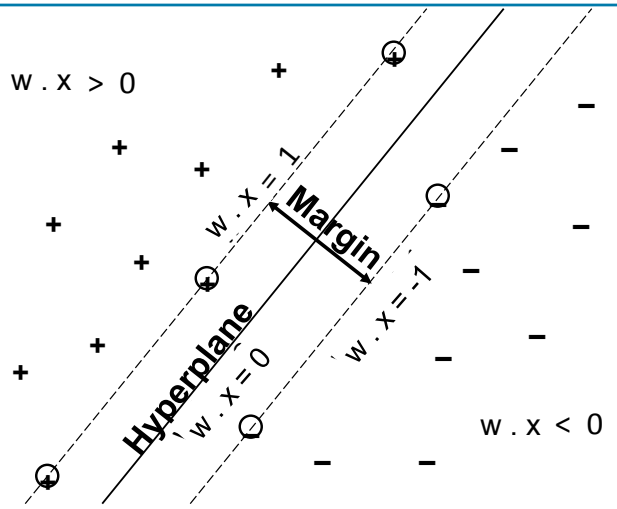
Non-Separable

"Best" Hyperplane?

- First, what is a hyperplane?
  - A generalization of a line to higher dimensions
  - Defined by a vector *w*
- With SVMs, the best hyperplane is the one with the ***maximum margin***
- If x^+ and x^- are the closest '+' and '-' inputs to the hyperplane, then the margin is:

$$\text{Margin}(w) = \frac{|w \cdot x^-| + |w \cdot x^+|}{||w||}$$

Support Vector Machines



Linear Separable Case

- In math:

$$\begin{aligned} &minimize: && \tfrac{1}{2}||w||^2 \\ &subject\ to: && \\ & && w \cdot x_i \geq 1 \quad \forall i \text{ s.t. Class}(i) = + \\ & && w \cdot x_i \leq -1 \quad \forall i \text{ s.t. Class}(i) = \text{-} \end{aligned}$$

- In English:
  - Find the largest margin hyperplane that separates the '+'s and '-'s

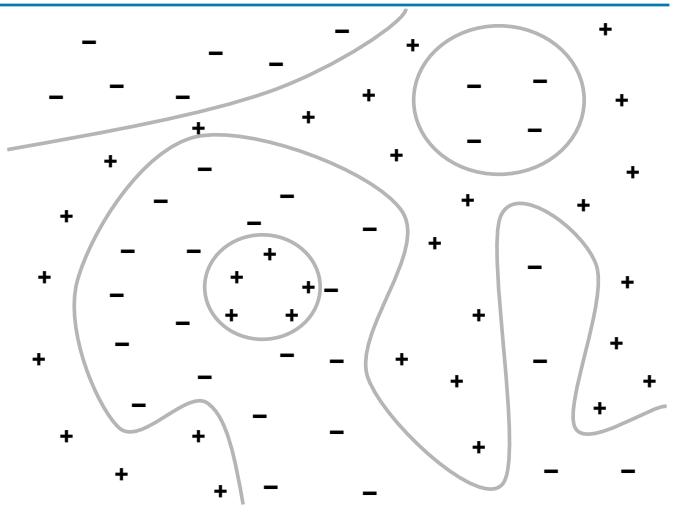
Linearly Non-Separable Case

- In math:

$$\begin{aligned} &minimize: && \tfrac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ &subject\ to: && \\ & && w \cdot x_i \geq 1 - \xi_i \quad \forall i \text{ s.t. Class}(i) = + \\ & && w \cdot x_i \leq -1 + \xi_i \quad \forall i \text{ s.t. Class}(i) = \text{-} \\ & && \xi_i \geq 0 \quad \forall i \end{aligned}$$

- In English:
  - ξ_i denotes how misclassified instance *i* is
  - Find a hyperplane that has a large margin and lowest misclassification cost

Nearest Neighbor Classification



The Kernel Trick

- Linearly non-separable data may become linearly separable if transformed, or mapped, to a higher dimension space
- Computing vector math (i.e., dot products) in very high dimensional space is costly
- The kernel trick allows very high dimensional dot products to be computed efficiently
- Allows inputs to be implicitly mapped to high (possibly infinite) dimensional space with little computational overhead

Non-Binary Classification with SVMs

- One versus all
  - Train "class *c* vs. not class *c*" SVM for every class
  - If there are K classes, must train *K* classifiers
  - Classify items according to:

$$\text{Class}(x) = \arg\max_c w_c \cdot x$$

- One versus one
  - Train a binary classifier for every pair of classes
  - Must train *K*(*K*-1)/2 classifiers
  - Computationally expensive for large values of *K*

SVM Tools

- Solving SVM optimization problem is not straightforward
- Many good software packages exist
  - SVM-Light
  - LIBSVM
  - R library
  - Matlab SVM Toolbox