Learning to Rank using Language Models and SVMs

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

References:

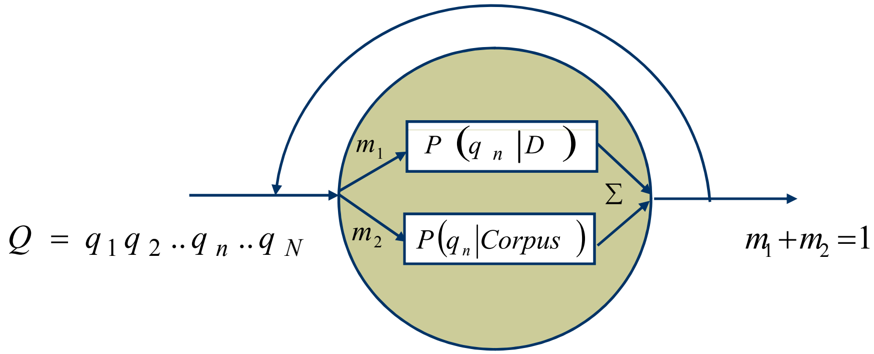
1. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Chapte 15 & associated slides, Cambridge University Press
2. Raymond J. Mooney's teaching materials
3. Berlin Chen et al., "*A discriminative HMM/N-gram-based retrieval approach for Mandarin spoken documents*," ACM Transactions on Asian Language Information Processing 3(2), June 2004.

Discriminatively-Trained Language Models (1/9)

- A simple document-based language model (LM) for information retrieval can be represented by

$$P(Q|D \text{ is } R) = \prod_{n=1}^{N} \left[m_1 P(q_n|D) + m_2 P(q_n|Corpus) \right]$$

  - The use of general corpus LM $P(q_n|Corpus)$ is for probability smoothing and better retrieval performance
  - Conventionally, the mixture weights m_1, m_2 $(m_1 + m_1 = 1)$ are empirically tuned or optimized by using the Expectation-Maximization (EM) algorithm



A mixture of *N* probability distributions

- D.R.H. Miller et al., "A hidden Markov model information retrieval system, *SIGIR 1999*.

- Berlin Chen et al., "An HMM/N-gram-based Linguistic Processing Approach for Mandarin Spoken Document Retrieval," *Interspeech* 2001

Discriminatively-Trained Language Models (2/9)

- For those documents with training queries, m_1 and m_2 can be estimated by using the Minimum Classification Error (MCE) training algorithm
  - The ordering of relevant documents D^* and irrelevant documents D' in the ranked list for a training query exemplar Q is adjusted to preserve the relationships $D^* \prec D'$; i.e., D^* should precede D' on the ranked list
    - A *learning-to-rank* algorithm
  - Documents thus can have different weights

- Berlin Chen et al., "A discriminative HMM/N-gram-based retrieval approach for Mandarin spoken documents," *ACM Transactions on Asian Language Information Processing* 3(2), June 2004.

Discriminatively-Trained Language Models (3/9)

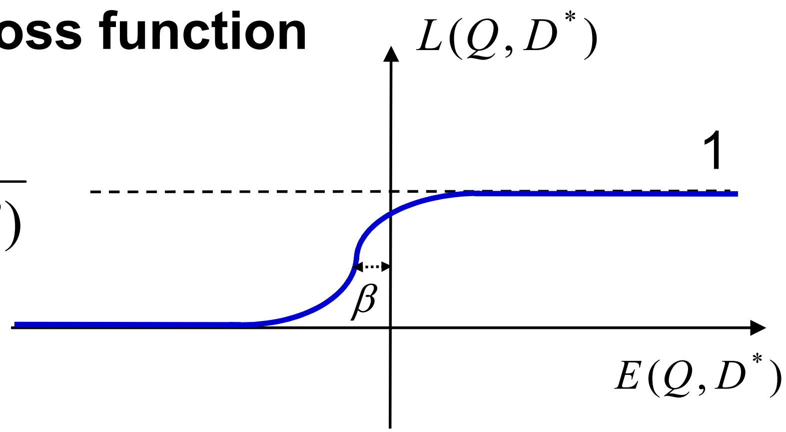
- Minimum Classification Error (MCE) Training
  - Given a query Q and a desired relevant doc D^*, define **the classification error function** as:

$$E(Q,D^*)=\frac{1}{|Q|}\left[-\log P\left(Q \middle| D^* \text{ is } R\right)+\max_{D'} \log P\left(Q \middle| D' \text{ is not } R\right)\right]$$

Also can take all irrelevant doc
in the answer set into consideration

">0": means misclassified; "<=0": means a correct decision

  - Transform the error function to **the loss function**

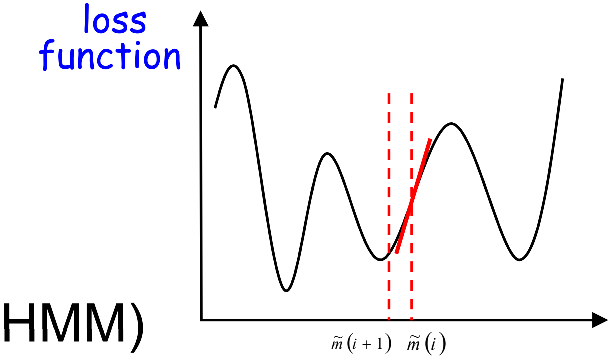


$$L(Q,D^*)=\frac{1}{1+\exp(-\alpha E(Q,D^*)+\beta)}$$

- In the range between 0 and 1
  - α : controls the slope
  - β : controls the offset

Discriminatively-Trained Language Models (4/9)

- Minimum Classification Error (MCE) Training
  - Apply the loss function to the MCE procedure for iteratively updating the weighting parameters



    - Constraints:

$$m_k \geq 0, \quad \sum_k m_k = 1$$

    - Parameter Transformation, (e.g.,Type I HMM)

$$m_1 = \frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} \quad \text{and} \quad m_2 = \frac{e^{\tilde{m}_2}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}}$$

  - Iteratively update m_1 (e.g., Type I HMM)

Gradient descent

$$\tilde{m}_1(i+1) = \tilde{m}_1(i) - \varepsilon(i) \cdot \frac{\partial L(Q, D^*)}{\partial \tilde{m}_1}\Big|_{D^* = D^*(i)}$$

    - Where,

$$\nabla_{D^*, \tilde{m}_1} = \varepsilon(i) \cdot \frac{\partial L(Q, D^*)}{\partial \tilde{m}_1} = \varepsilon(i) \cdot \frac{\partial L(Q, D^*)}{\partial E(Q, D^*)} \cdot \frac{\partial E(Q, D^*)}{\partial \tilde{m}_1},$$

$$\frac{\partial L(Q, D^*)}{\partial E(Q, D^*)} = \alpha \cdot L(Q, D^*) \cdot [1 - L(Q, D^*)]$$

Discriminatively-Trained Language Models (5/9)

- Minimum Classification Error (MCE) Training
  - Iteratively update m_1 (e.g., Type I HMM)

Note:

$$[\log f(x)]' = \frac{1}{f(x)} f'(x)$$

$$[f(x)g(x)]' = f'(x)g(x) + f(x)g'(x)$$

$$\left[\frac{f(x)}{g(x)}\right]' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

$$\frac{\partial E(Q,D^*)}{\partial \tilde{m}_1} = \frac{-1}{|Q|} \frac{\partial \left\{ \sum_{q_n \in Q} \log\left[\frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|D^*) + \frac{e^{\tilde{m}_2}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|Corpus) \right] \right\}}{\partial \tilde{m}_1}$$

$$= \frac{-1}{|Q|} \sum_{q_n \in Q} \left\{ \frac{ \frac{-e^{\tilde{m}_1}}{(e^{\tilde{m}_1} + e^{\tilde{m}_2})^2} \left[e^{\tilde{m}_1} P(q_n|D^*) + e^{\tilde{m}_2} P(q_n|Corpus) \right] + \frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|D^*) }{ \frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|D^*) + \frac{e^{\tilde{m}_2}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|Corpus) } \right\}$$

$$= \frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} - \frac{1}{|Q|} \sum_{q_n \in Q} \left\{ \frac{ \frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|D^*) }{ \frac{e^{\tilde{m}_1}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|D^*) + \frac{e^{\tilde{m}_2}}{e^{\tilde{m}_1} + e^{\tilde{m}_2}} P(q_n|Corpus) } \right\}$$

$$= -\left[-m_1 + \frac{1}{|Q|} \sum_{q_n \in Q} \frac{m_1 P(q_n|D^*)}{m_1 P(q_n|D^*) + m_2 P(q_n|Corpus)} \right],$$

Discriminatively-Trained Language Models (6/9)

- Minimum Classification Error (MCE) Training
  - Iteratively update m_1

$$\nabla_{D^*,\tilde{m}_1}(i) = -\varepsilon(i)\cdot\alpha\cdot L(Q,D^*)\cdot\left[1-L(Q,D^*)\right] \cdot\left[-m_1(i)+\frac{1}{|Q|}\sum_{q_n\in Q}\frac{m_1(i)P(q_n|D^*)}{m_1(i)P(q_n|D^*)+m_2(i)P(q_n|Corpus)}\right],$$

the new weight

$$m_1(i+1)=\frac{e^{\tilde{m}_1(i+1)}}{e^{\tilde{m}_1(i+1)}+e^{\tilde{m}_2(i+1)}}$$

$$\tilde{m}_1(i+1)=\tilde{m}_1(i)-\nabla_{D^*,\tilde{m}_1}(i)$$

$$=\frac{e^{\tilde{m}_1(i)}e^{-\nabla_{D^*,\tilde{m}_1}(i)}}{e^{\tilde{m}_1(i)}e^{-\nabla_{D^*,\tilde{m}_1}(i)}+e^{\tilde{m}_2(i)}e^{-\nabla_{D^*,\tilde{m}_2}(i)}}$$

$$=\frac{e^{\tilde{m}_1(i)}e^{-\nabla_{D^*,\tilde{m}_1}(i)}\Big/\left(e^{\tilde{m}_1(i)}+e^{\tilde{m}_2(i)}\right)}{\left[e^{\tilde{m}_1(i)}e^{-\nabla_{D^*,\tilde{m}_1}(i)}\Big/\left(e^{\tilde{m}_1(i)}+e^{\tilde{m}_2(i)}\right)\right]+\left[e^{\tilde{m}_2(i)}e^{-\nabla_{D^*,\tilde{m}_2}(i)}\Big/\left(e^{\tilde{m}_1(i)}+e^{\tilde{m}_2(i)}\right)\right]}$$

the old weight

$$=\frac{m_1(i)\cdot e^{-\nabla_{D^*,\tilde{m}_1}(i)}}{m_1(i)\cdot e^{-\nabla_{D^*,\tilde{m}_1}(i)}+m_2(i)\cdot e^{-\nabla_{D^*,\tilde{m}_2}(i)}},$$

Discriminatively-Trained Language Models (7/9)

- Minimum Classification Error (MCE) Training
  - Final Equations
    - Iteratively update m_1

$$\nabla_{D^*,\tilde{m}_1}(i) = -\varepsilon(i)\cdot\alpha\cdot L(Q,D^*)\cdot\left[1-L(Q,D^*)\right]$$
$$\cdot\left[-m_1(i)+\frac{1}{|Q|}\sum_{q_n\in Q}\frac{m_1(i)P(q_n|D^*)}{m_1(i)P(q_n|D^*)+m_2(i)P(q_n|Corpus)}\right]$$

$$m_1(i+1)=\frac{m_1(i)\cdot e^{-\nabla_{D^*,\tilde{m}_1}(i)}}{m_1(i)\cdot e^{-\nabla_{D^*,\tilde{m}_1}(i)}+m_2(i)\cdot e^{-\nabla_{D^*,\tilde{m}_2}(i)}}$$

- m_2 can be updated in the similar way

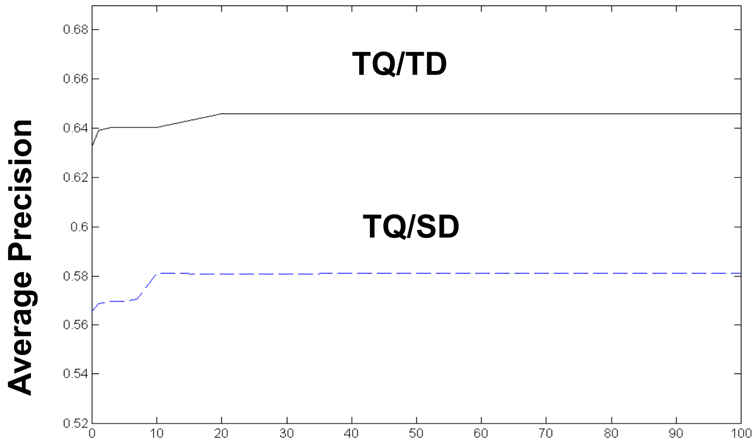
Discriminatively-Trained Language Models (8/9)

- Experimental results with MCE training

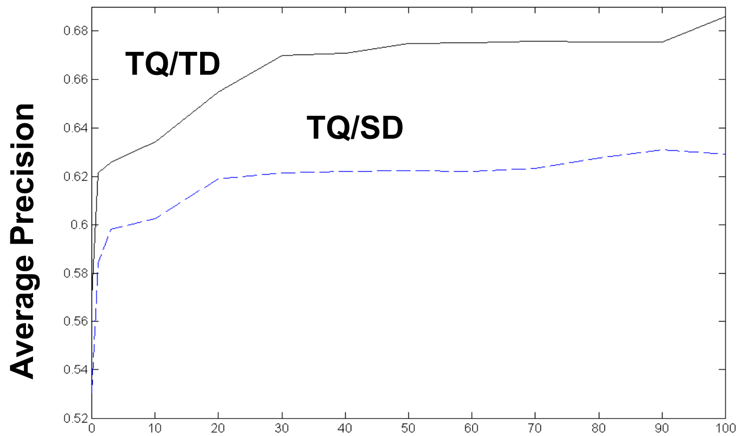
Average Precision		Word-level	Syllable-level	Fusion
		Uni	Uni+Bi*	
TDT2	TQ/TD	0.6459 (0.6327)	0.6858 (0.5718)	0.7329
	TQ/SD	0.5810 (0.5658)	0.6300 (0.5307)	0.6914

Before MCE Training → (values in parentheses)

Iterations=100



MCE Iterations (Word-based)



MCE Iterations (syllable-based)

- The results for the syllable-level indexing features were significantly improved

Discriminatively-Trained Language Models (9/9)

- Similar treatments have been recently applied to Document Topic Models (e.g., PLSA) and Word Topic Models (WTM) with good success

- For example, the ranking formula for PLSA can be represented by

$$
\begin{aligned}
P(q|D) &= \alpha \cdot \left(\beta \cdot \left[\sum_{T_k} P(q|T_k) P(T_k|D) \right] + (1-\beta) \cdot P(q|Corpus) \right) + (1-\alpha) \cdot P(q|D) \\
&= \sum_{T_k} \alpha\beta \cdot P(q|T_k) P(T_k|D) + \alpha(1-\beta) \cdot P(q|Corpus) + (1-\alpha) \cdot P(q|D) \\
&= \sum_{T_k} \left(\left[\alpha\beta \cdot P(q|T_k) + \alpha(1-\beta) \cdot P(q|Corpus) + (1-\alpha) \cdot P(q|D) \right] P(T_k|D) \right)
\end{aligned}
$$

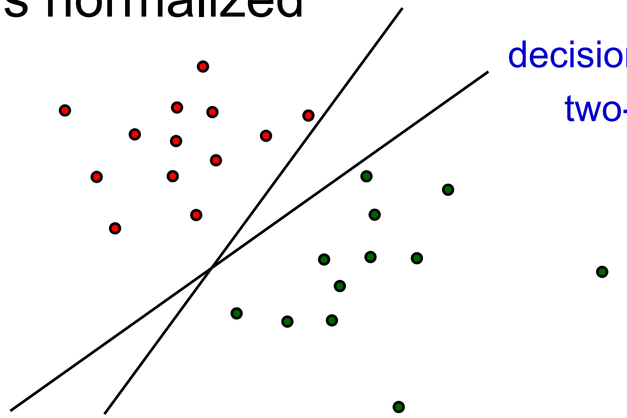
  – The weighting parameters α and β document topic distributions $P(T_k|D)$ can be trained by the MCE algorithm

Vector Representations

- Data points (e.g., documents) of different classes (e.g., relevant/non-relevant classes) are represented as vectors in a *n*-dimensional vector space
  - Each dimension has to do with a specific feature, whose value usually is normalized

decision function

$$f(\vec{x}) = \text{sign}\left(\vec{w}^{\mathrm{T}}\vec{x} + b\right)$$



decision hyperplane of a two-class problem

$$\vec{w}^{\mathrm{T}}\vec{x} + b$$

$$\mathbf{D} = \{\vec{x}_i, y_i\}_{i=1}^{N}$$

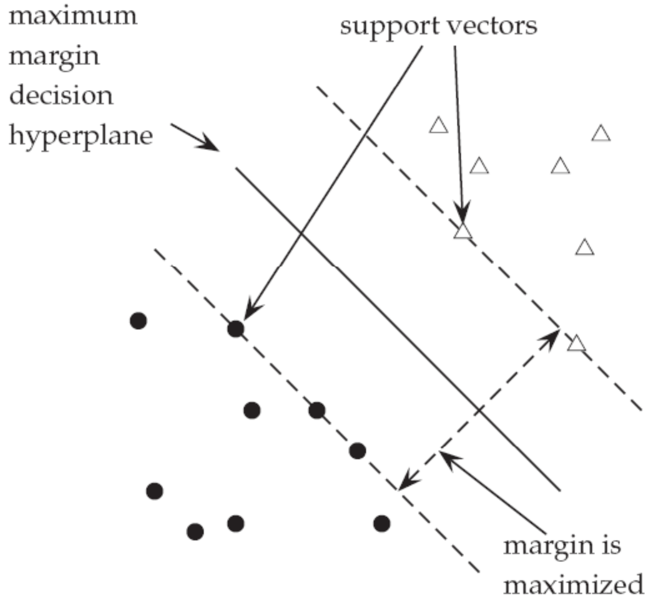
data point

class label (e.g., +1, -1)

- Support vector machines (SVM)
  - Look for a decision surface (or hyperplane) that is maximally far away from any data point
  - Margin: the distance from the decision surface to the closest data points on either side (or the support vectors)
  - SVM is a kind of large-margin classifier

Support Vectors

- SVM is fully specified by a small subset of the data (i.e., the support vectors) that defines the position of the separator (the decision hyperplane)



The support vectors are 5 points right up against the margin of the classifier

Intercept term

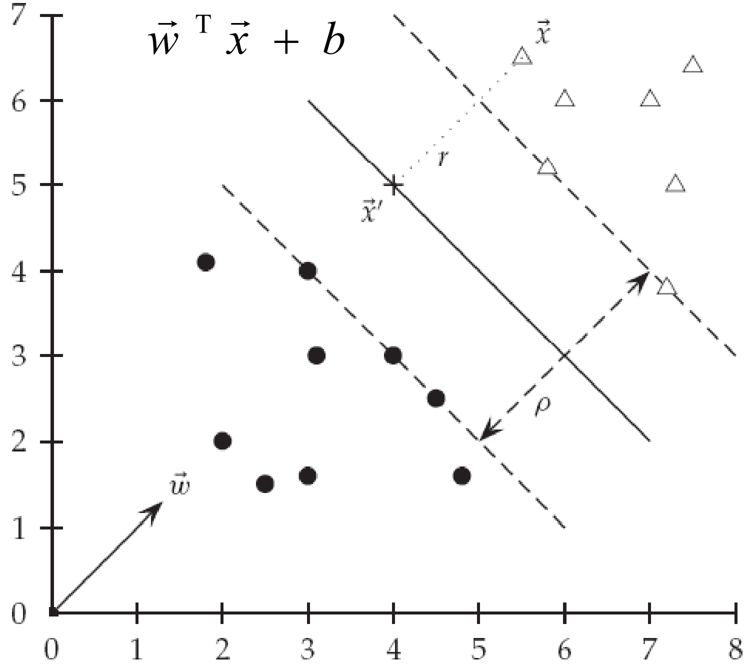
$$\vec{w}^{T}\vec{x} + b$$

normal (weight) vector of the hyperplane

- Maximization of the margin
  - If there are no points near the decision surface, then there are no very uncertain classification decisions
  - Also, a slight error in measurement or a slight document variation will not cause a misclassification

Formulation of SVM with Algebra (1/2)

- Assume here that data points are linearly separable
- Euclidean distance of a point to the decision boundary



Assume data points are linear separable !

1. The shortest distance between a point $\vec{x}$ to a hyperplane is perpendicular to the plane, i.e., parallel to $\vec{w}$

2. The point on the plane closest to $\vec{x}$ is $\vec{x}'$

$$\vec{x}' = \vec{x} - yr\frac{\vec{w}}{|\vec{w}|}$$

$$\Rightarrow \vec{w}^{\mathrm{T}}\left(\vec{x} - yr\frac{\vec{w}}{|\vec{w}|}\right) + b = 0$$

$$\Rightarrow r = \frac{y\left(\vec{w}^{\mathrm{T}}\vec{x} + b\right)}{|\vec{w}|} \text{ or } \frac{\left|\vec{w}^{\mathrm{T}}\vec{x} + b\right|}{|\vec{w}|}$$

3. We can scale $y\left(\vec{w}^{\mathrm{T}}\vec{x} + b\right)$, the so-called "functional margin", as we please; for example, to 1

Therefore, the margin defined by the support vectors is expressed by $\frac{2}{|\vec{w}|}$

(i.e., for support vectors $y\left(\vec{w}^{\mathrm{T}}\vec{x} + b\right) = 1$

; while for the others $y\left(\vec{w}^{\mathrm{T}}\vec{x} + b\right) \geq 1$)

Formulation of SVM with Algebra (2/2)

- SVM is designed to find $\vec{w}$ and b that can maximize the geometric margin
  - $\frac{2}{|\vec{w}|}$ (maximization of $\frac{2}{|\vec{w}|}$ is equivalent to minimization of $\frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w}$)
  - For all $\{\vec{x}_i, y_i\} \in \mathbf{D},\ y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) \geq 1$

Mathematical formulation (assume linear separability)

- Primal Problem
  - Minimize $\mathbf{L}_p$ with respect to $\vec{w}$ and b

$$\min \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} \text{ subject to } \quad y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) \geq +1, \forall\ i$$

To be minimized — To be maximized

$$\mathbf{L}_p = \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) - 1\right] \left(\alpha_i \geq 0\right)$$

?

$$= \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} - \sum_{t=1}^{N}\alpha_i y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) + \sum_{t=1}^{N}\alpha_i$$

1

2

$$\frac{\partial \mathbf{L}_p}{\partial \vec{w}} = 0 \Rightarrow \vec{w} = \sum_{t=1}^{N}\alpha_i y_i \vec{x}_i$$

$$\frac{\partial \mathbf{L}_p}{\partial b} = 0 \Rightarrow \sum_{t=1}^{N}\alpha_i y_i = 0$$

3

Formulation of SVM with Algebra (3/3)

- Dual problem (plug ② and ③ into ①)
  - Maximize $\mathbf{L}_d$ with respect to α_i

A convex quadratic-optimization problem

$$\mathbf{L}_d = \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} - \sum_{i=1}^{N}\alpha_i y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) + \sum_{i=1}^{N}\alpha_i$$

$$= \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} - \vec{w}^{\mathrm{T}}\underbrace{\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i}_{\vec{w}} - b\underbrace{\sum_{i=1}^{N}\alpha_i y_i}_{0} + \sum_{i=1}^{N}\alpha_i$$

$$= -\frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} + \sum_{i=1}^{N}\alpha_i$$

scalar

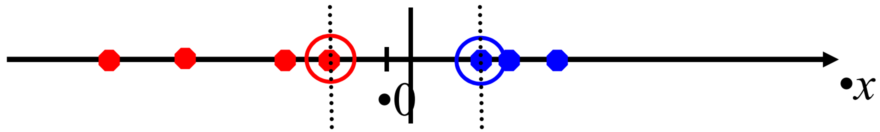
$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i^{\mathrm{T}}\vec{x}_j + \sum_{i=1}^{N}\alpha_i$$

Subject to the constraints that $\sum_{i=1}^{N}\alpha_i y_i = 0$ and $\alpha_i \geq 0 \ \forall i$

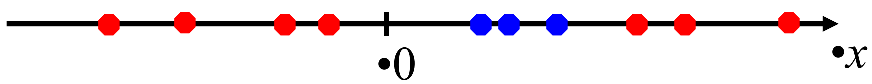
- Most α_i are 0 and only a small number have $\alpha_i > 0$ (they are support vectors)
- Have to do with the number of training instances, but not the input dimension

Dealing with Nonseparability (1/2)

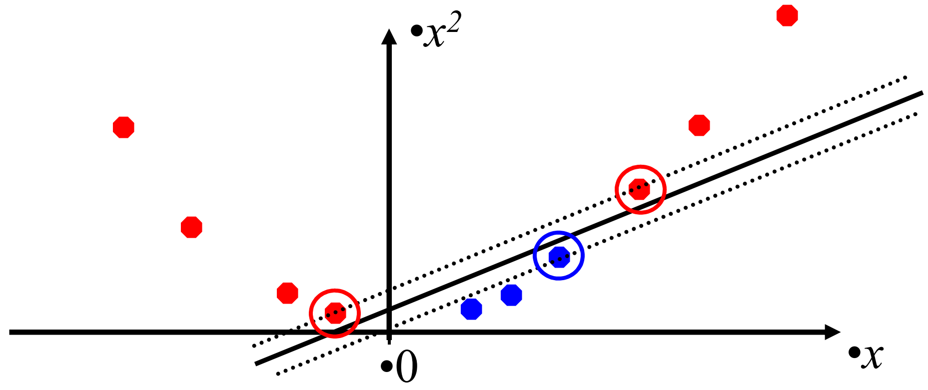
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

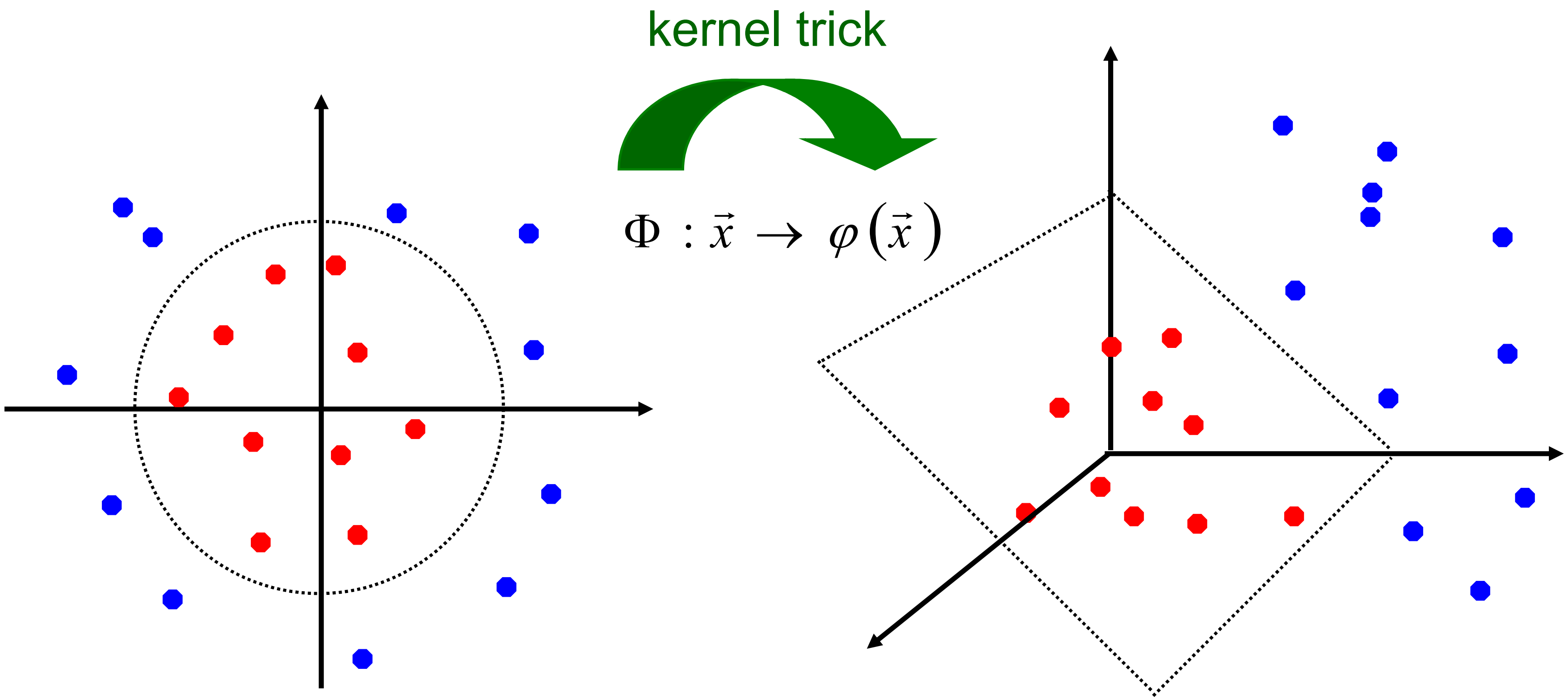


- How about mapping data to a higher-dimensional space?



Dealing with Nonseparability (2/2)

- General idea: The original feature space can always be mapped by a function $\varphi(\cdot)$ to some higher-dimensional feature space where the training set is separable



Purposes:

- Make non-separable problem separable
- Map data into better representational space

Kernel Trick (1/2)

- The SVM decision function for an input $\vec{x}$ at a high-dimensional (the transformed) space can be represented as

$$
\begin{aligned}
f(\vec{x}) &= \text{sign}\left(\vec{w}^{\mathrm{T}}\varphi(\vec{x})+b\right)\\
&= \text{sign}\left(\sum_{i=1}^{N}\alpha_i y_i \varphi(\vec{x}_i)^T \varphi(\vec{x})+b\right)\\
&= \text{sign}\left(\sum_{i=1}^{N}\alpha_i y_i \underline{K(\vec{x}_i,\vec{x})}+b\right)
\end{aligned}
$$

  - A kernel function $K(\vec{x}_i,\vec{x})$ is introduced, defined by the inner (dot) product of points (vectors) in the high-dimensional space
    - $K(\vec{x}_i,\vec{x})$ can be computed simply and efficiently in terms of the original data points
    - We wouldn't have to actually map from $\vec{x} \rightarrow \varphi(\vec{x})$

      (however, we still can directly compute $K(\vec{x}_i,\vec{x}) = \varphi(\vec{x}_i)^T \varphi(\vec{x})$)

Kernel Trick (2/2)

- Common Kernel Functions
  - Polynomials of degree q: $K(\vec{u},\vec{v}) = (\vec{u}^{\mathrm{T}}\vec{v}+1)^q$
    - Polynomial of degree two (quadratic kernel)

two-dimensional points

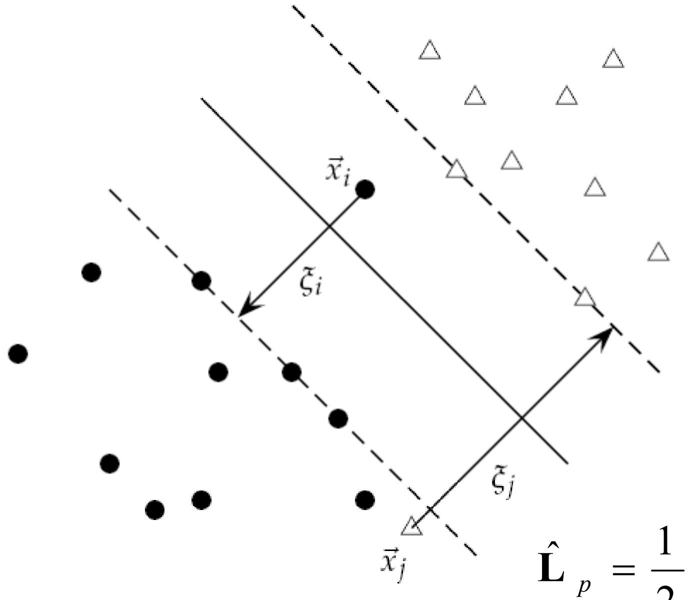
$$K(\vec{u},\vec{v}) = (\vec{u}^{\mathrm{T}}\vec{v}+1)^2$$
$$= (u_1v_1+u_2v_2+1)^2 \quad \left(\text{where } \vec{u}^{\mathrm{T}} = [u_1,u_2], \vec{u}^{\mathrm{T}} = [v_1,v_2]\right)$$
$$= 1+2u_1v_1+2u_2v_2+2u_1u_2v_1v_2+u_1^2v_1^2+u_2^2v_2^2$$
$$\phi(\vec{u}) = \left[1,\sqrt{2}u_1,\sqrt{2}u_2,\sqrt{2}u_1u_2,u_1^2,u_2^2\right]^T$$

  - Radial-basis function (Gaussian distribution): $K(\vec{u},\vec{v}) = e^{-(\vec{u}-\vec{v})^2/(2\sigma^2)}$
  - Sigmoidal function: $K(\vec{u},\vec{v}) = \tanh(2\vec{u}^{\mathrm{T}}\vec{v}+1)$

The above kernels are not always very useful in text classification !

Soft-Margin Hyperplane (1/2)

- Even for very high-dimensional problems, data points could be linearly inseparable
- We can instead look for the hyperplane that incurs the least error
  - Define slack variables $\xi_i \geq 0$ that store the variation from the margin for each data points



- Reformulation the optimization criterion with slack variables
  - Find $\vec{x}$, b , and $\xi_i \geq 0$ such that
    - ☞ $\frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} + C\sum_{i=1}^{N}\xi_i$ is minimum
    - ☞ For all $\{\vec{x}_i, y_i\} \in \mathbf{D}$, $y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) \geq 1 - \zeta_i$

$$\hat{\mathbf{L}}_p = \frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} + C\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\vec{w}^{\mathrm{T}}\vec{x}_i + b\right) - 1 + \zeta_i\right] + \sum_{i=1}^{N}\mu_i\zeta_i$$

Soft-Margin Hyperplane (2/2)

– Dual Problem

$$\hat{\mathbf{L}}_d = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i^{\mathrm{T}} \vec{x}_j$$

$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \ \text{ and } \ 0 \le \alpha_i \le C \quad \forall i$$

- Neither slack variables ξ_i nor their Lagrange multipliers μ_i appear in the dual problem!
- Again, $\vec{x}$ with non-zero α_i will be support vectors
- Solution to the dual problem is:

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

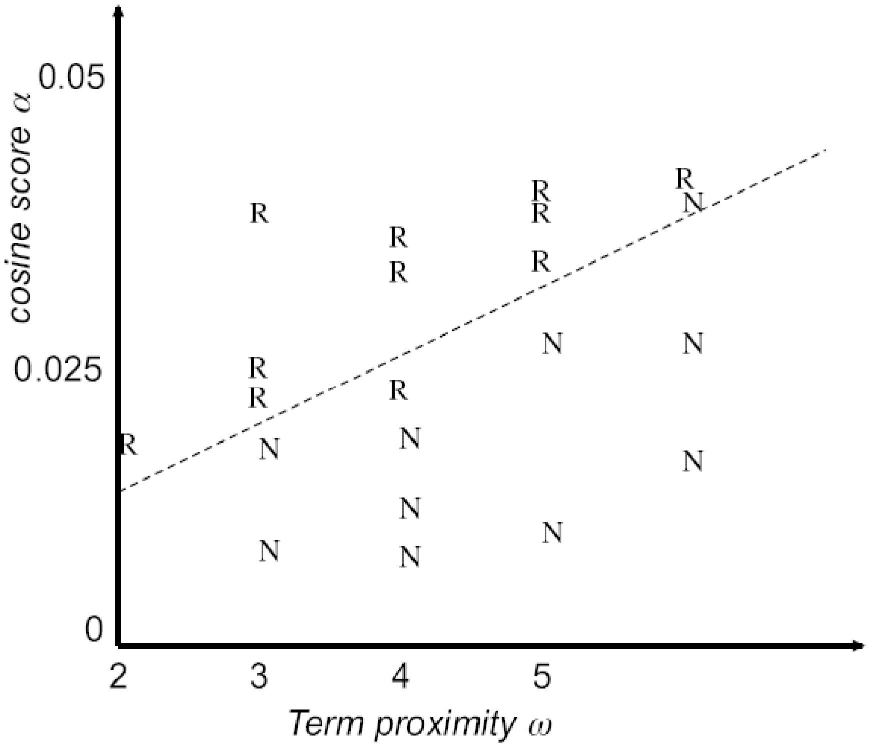
$$b = y_k (1 - \xi_k) - \vec{w}^{\mathrm{T}} \vec{x}_k \quad \text{for } k = \arg\max_k \alpha_k$$

– Parameter C can be viewed as a way to control overfitting – a regularization term

- The larger the value C , the more we should pay attention to each individual data point
- The smaller the value C , the more we can model the bulk of the data

Using SVM for Ad-Hoc Retrieval (1/2)

- For example, documents are simply represented by two-dimensional vectors $\psi(d_i, q)$ consisting of cosine score and term proximity



► **Figure 15.7** A collection of training examples. Each R denotes a training example labeled *relevant*, while each N is a training example labeled *nonrelevant*.

Using SVM for Ad-Hoc Retrieval (2/2)

- Examples: Nallapati, Discriminative Models for Information Retrieval, *SIGIR 2004*
  - Basic Features used in SVM

	Feature		Feature
1	$\sum_{q_i \in Q \cap D} log(c(q_i, D))$	4	$\sum_{q_i \in Q \cap D}(log(\frac{\|C\|}{c(q_i,C)}))$
2	$\sum_{i=1}^{n} log(1 + \frac{c(q_i,D)}{\|D\|})$	5	$\sum_{i=1}^{n} log(1 + \frac{c(q_i,D)}{\|D\|} idf(q_i))$
3	$\sum_{q_i \in Q \cap D} log(idf(q_i))$	6	$\sum_{i=1}^{n} log(1 + \frac{c(q_i,D)}{\|D\|} \frac{\|C\|}{c(q_i,C)})$

  - Compared with LM and ME (maximum entropy) models

Train ↓ Test →		Disks 1-2 (151-200)	Disk 3 (101-150)	Disks 4-5 (401-450)	WT2G (426-450)
Disks 1-2 (101-150)	LM ($\mu^* = 1900$)	**0.2561 (6.75e-3)**	0.1842	0.2377 (0.80)	0.2665 (0.61)
	SVM	0.2145	0.1877 (0.3)	0.2356	0.2598
	ME	0.1513	0.1240	0.1803	0.1815
Disk 3 (51-100)	LM ($\mu^* = 500$)	**0.2605 (1.08e-4)**	0.1785 (0.11)	0.2503 (0.21)	0.2666
	SVM	0.2064	0.1728	0.2432	0.2750 (0.55)
	ME	0.1599	0.1221	0.1719	0.1706
Disks 4-5 (301-350)	LM ($\mu^* = 450$)	**0.2592 (1.75e-4)**	**0.1773 (7.9e-3)**	**0.2516 (0.036)**	0.2656
	SVM	0.2078	0.1646	0.2355	0.2675 (0.89)
	ME	0.1413	0.0978	0.1403	0.1355
WT2G (401-425)	LM ($\mu^* = 2400$)	**0.2524 (4.6e-3)**	0.1838 (0.08)	0.2335	0.2639
	SVM	0.2199	0.1744	**0.2487 (0.046)**	**0.2798 (0.037)**
	ME	0.1353	0.0969	0.1441	0.1432
Best TREC runs (Site)		0.4226 (UMass)	N/A	0.3207 (Queen's College)	N/A

Tested on 4 TREC collections

Ranking SVM (1/2)

- Construct an SVM that not only considers the relevance of documents to the a training query but also the order of each document pair on the ideal ranked list
  - First, construct a vector of features $\psi(d_i, q)$ for each document-query pair
  - Second, capture the relationship between each document pair by introducing a new vector representation $\phi(d_i, d_j, q)$ for each document pair

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

  - Third, if d_i is more relevant than d_j given q (denoted $d_i \prec d_j$, i.e., d_i should precede d_j on the ranked list), then associate they with the label $y_{ijq} = +1$; otherwise, $y_{ijq} = -1$

Cf. T. Joachims and F. Radlinski, Search Engines that Learn from Implicit Feedback, *IEEE Trans. on Computer* 40(8), pp. 34-40, 2007

Ranking SVM (2/2)

- Therefore, the above ranking task is formulated as:
  - Find $\vec{x}$, b, and $\xi_{ijq} \geq 0$ such that
    - $\frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w} + C\sum_{i,j,q}\xi_{i,j,q}$ is minimized
    - For all $\{\phi(d_i, d_j, q) : d_i \prec d_j\}$, $\vec{w}^{\mathrm{T}}\phi(d_i, d_j, q) + b \geq 1 - \xi_{i,j,q}$

      (Note that y_{ijq} are left out here. Why?)