Text Classification

Pabitra Mitra
pabitra@cse.iitkgp.ernet.in

Text Classification: definition

- The classifier:
  - *Input*: a document x
  - *Output*: a predicted class y from some fixed set of labels $y_1,\ldots,y_K$
- The learner:
  - *Input:* a set of m hand-labeled documents $(x_1,y_1),\ldots,(x_m,y_m)$
  - *Output:* a learned classifier $f{:}x \rightarrow y$

Text Classification: Examples

- Classify news stories as *World, US, Business, SciTech, Sports, Entertainment, Health, Other*
- Add MeSH terms to Medline abstracts
  - e.g. “Conscious Sedation” [E03.250]
- Classify business names by industry.
- Classify student essays as *A,B,C,D,* or *F.*
- Classify email as *Spam, Other.*
- Classify email to tech staff as *Mac, Windows, ..., Other.*
- Classify pdf files as *ResearchPaper, Other*
- Classify documents as *WrittenByReagan, GhostWritten*
- Classify movie reviews as *Favorable,Unfavorable,Neutral.*
- Classify technical papers as *Interesting, Uninteresting.*
- Classify jokes as *Funny, NotFunny.*
- Classify web sites of companies by Standard Industrial Classification (SIC) code.

Text Classification: Examples

- Best-studied benchmark: *Reuters-21578* newswire stories
  - 9603 train, 3299 test documents, 80-100 words each, 93 classes

> ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS
>
> BUENOS AIRES, Feb 26
>
> Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:
>
> - Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).
> - Maize Mar 48.0, total 48.0 (nil).
> - Sorghum nil (nil)
> - Oilseed export registrations were:
> - Sunflowerseed total 15.0 (7.9)
> - Soybean May 20.0, total 20.0 (nil)
>
> The board also detailed export registrations for subproducts, as follows....

→ Categories: grain, wheat (of 93 binary choices)

Classification Methods

- Supervised learning
  - Naive Bayes (simple, common)
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (generally more powerful)
  - Boosting
  - … plus many other methods
  - No free lunch: requires hand-classified training data
  - But data can be built up (and refined) by amateurs
- Many commercial systems use a mixture of methods

Representing text for classification

f(

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

- Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).
- Maize Mar 48.0, total 48.0 (nil).
- Sorghum nil (nil)
- Oilseed export registrations were:
- Sunflowerseed total 15.0 (7.9)
- Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....

)=y

?

simplest useful

What is the ~~best~~ representation for the document *x* being classified?

Bag of words representation

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

- Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).
- Maize Mar 48.0, total 48.0 (nil).
- Sorghum nil (nil)
- Oilseed export registrations were:
- Sunflowerseed total 15.0 (7.9)
- Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....

Categories: grain, wheat

Bag of words representation

xxxxxxxxxxxxxxxxxxxx GRAIN/OILSEED xxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxx grain xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx grains, oilseeds xxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxx tonnes, xxxxxxxxxxxxxxxxx shipments xxxxxxxxxxxx total xxxxxxxxx total xxxxxxxx xxxxxxxxxxxxxxxxxxxx:

- Xxxxx wheat xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, total xxxxxxxxxxxxxxxx
- Maize xxxxxxxxxxxxxxxxx
- Sorghum xxxxxxxxxx
- Oilseed xxxxxxxxxxxxxxxxxxxxx
- Sunflowerseed xxxxxxxxxxxxxx
- Soybean xxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx....

→ Categories: grain, wheat

Bag of words representation

xxxxxxxxxxxxxxxxxxx GRAIN/OILSEED xxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxx grain xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx grains, oilseeds xxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx tonnes, xxxxxxxxxxxxxxxxx shipments
xxxxxxxxxxxx total xxxxxxxxx total xxxxxxxx xxxxxxxxxxxxxxxxxxxxxx:

- Xxxxx wheat xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, total xxxxxxxxxxxxxxxx
- Maize xxxxxxxxxxxxxxxxx
- Sorghum xxxxxxxxxx
- Oilseed xxxxxxxxxxxxxxxxxxxxx
- Sunflowerseed xxxxxxxxxxxxxx
- Soybean xxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx....

word	*freq*
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

Categories: grain, wheat

Text Classification with Naive Bayes

- Represent document x as set of (w_i,f_i) pairs:
  - $x = \{(grain,3),(wheat,1),...,(the,6)\}$
- For each y, build a probabilistic model Pr(X|Y=y) of “documents” in class y
  - Pr(X=*{(grain,3),...}*|Y=*wheat*) = ...
  - Pr(X=*{(grain,3),...}*|Y=*nonWheat*) = ...
- To classify, find the *y* which was most likely to *generate x—i.e.,* which gives *x* the best score according to Pr(x|y)
  - $f(x) = \mathrm{argmax}_y \Pr(x/y)*\Pr(y)$

Bayes Rule

$$\Pr(y \mid x) \cdot \Pr(x) = \Pr(x, y) = \Pr(x \mid y) \cdot \Pr(y)$$

$$\Rightarrow \qquad \Pr(y \mid x) = \frac{\Pr(x \mid y) \cdot \Pr(y)}{\Pr(x)}$$

$$\Rightarrow \ \arg\max_y \Pr(y \mid x) = \arg\max_y \Pr(x \mid y) \cdot \Pr(y)$$

Text Classification with Naive Bayes

- How to estimate Pr(X|Y) ?
- *Simplest useful* process to generate a bag of words:
  - pick word 1 according to Pr(W|Y)
  - repeat for word 2, 3, ...
  - each word is generated *independently* of the others (which is clearly not true) but means

$$\Pr(w_1,\ldots,w_n \mid Y=y) = \prod_{i=1}^{n} \underbrace{\Pr(w_i \mid Y=y)}_{\text{How to estimate Pr(W|Y)?}}$$

Text Classification with Naive Bayes

- How to estimate Pr(X|Y) ?

$$\Pr(w_1,\ldots,w_n \mid Y=y)=\prod_{i=1}^{n}\Pr(w_i \mid Y=y)$$

Estimate *Pr(w|y)* by looking at the data...

$$\Pr(W=w \mid Y=y)=\frac{\text{count}(W=w \text{ and } Y=y)}{\text{count}(Y=y)}$$

This gives score of zero if x contains a brand-new word w_{new}

Text Classification with Naive Bayes

- How to estimate Pr(X|Y) ?

$$\Pr(w_1, \ldots, w_n \mid Y = y) = \prod_{i=1}^{n} \Pr(w_i \mid Y = y)$$

... and also imagine *m* examples
with *Pr(w|y)=p*

$$\Pr(W = w \mid Y = y) = \frac{\text{count}(W = w \text{ and } Y = y) + mp}{\text{count}(Y = y) + m}$$

Terms:

- This Pr(W|Y) is a *multinomial distribution*
- This use of *m* and *p* is a *Dirichlet prior* for the multinomial

Text Classification with Naive Bayes

- How to estimate Pr(X|Y) ?

$$\Pr(w_1, \ldots, w_n \mid Y = y) = \prod_{i=1}^{n} \underbrace{\Pr(w_i \mid Y = y)}$$

for instance: *m=1, p=0.5*

$$\Pr(W = w \mid Y = y) = \frac{\text{count}(W = w \text{ and } Y = y) + 0.5}{\text{count}(Y = y) + 1}$$

Text Classification with Naive Bayes

- Putting this together:
  - for each document x_i with label y_i
    - for each word w_{ij} in x_i
      - count[w_{ij}][y_i]++
      - count[y_i]++
      - count++
  - to classify a new $x=w_1 \ldots w_n$, pick y with top *score*:

$$score(y, w_1 \ldots w_k) = \lg \frac{\text{count}[y]}{\text{count}} + \sum_{i=1}^{n} \lg \frac{\text{count}[w_i][y] + 0.5}{\text{count}[y] + 1}$$

key point: we only need counts for words that actually appear in *x*

Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words … and more
- Selection may make a particular classifier feasible
  - Some classifiers can't deal with 1,000,000 features
- Reduces training time
  - Training time for some methods is quadratic or worse in the number of features
- Makes runtime models smaller and faster
- Can improve generalization (performance)
  - Eliminates noise features
  - Avoids overfitting

Feature Selection: Frequency

- The simplest feature selection method:
  - Just use the commonest terms
  - No particular foundation
  - But it make sense why this works
    - They' re the words that can be well-estimated and are most often available as evidence
  - In practice, this is often 90% as good as better methods
  - Smarter feature selection

Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data
  - Sometimes use cross-validation (averaging results over multiple training and test splits of the overall data)
- Easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set)

Evaluating Categorization

- Measures: precision, recall, F1, classification accuracy
- Classification accuracy: r/n where n is the total number of test docs and r is the number of test docs correctly classified

WebKB Experiment (1998)

- Classify webpages from CS departments into:
  - student, faculty, course, project
- Train on ~5,000 hand-labeled web pages
  - Cornell, Washington, U.Texas, Wisconsin
- Crawl and classify a new site (CMU) using Naïve Bayes
- Results

	Student	Faculty	Person	Project	Course	Departmt
Extracted	180	66	246	99	28	1
Correct	130	28	194	72	25	1
Accuracy:	72%	42%	79%	73%	89%	100%

Faculty

associate	0.00417
chair	0.00303
member	0.00288
ph	0.00287
director	0.00282
fax	0.00279
journal	0.00271
recent	0.00260
received	0.00258
award	0.00250

Students

resume	0.00516
advisor	0.00456
student	0.00387
working	0.00361
stuff	0.00359
links	0.00355
homepage	0.00345
interests	0.00332
personal	0.00332
favorite	0.00310

Courses

homework	0.00413
syllabus	0.00399
assignments	0.00388
exam	0.00385
grading	0.00381
midterm	0.00374
pm	0.00371
instructor	0.00370
due	0.00364
final	0.00355

Departments

departmental	0.01246
colloquia	0.01076
epartment	0.01045
seminars	0.00997
schedules	0.00879
webmaster	0.00879
events	0.00826
facilities	0.00807
eople	0.00772
postgraduate	0.00764

Research Projects

investigators	0.00256
group	0.00250
members	0.00242
researchers	0.00241
laboratory	0.00238
develop	0.00201
related	0.00200
arpa	0.00187
affiliated	0.00184
project	0.00183

Others

type	0.00164
jan	0.00148
enter	0.00145
random	0.00142
program	0.00136
net	0.00128
time	0.00128
format	0.00124
access	0.00117
begin	0.00116

SpamAssassin

- Naïve Bayes has found a home in spam filtering
  - Widely used in spam filters
  - But many features beyond words:
    - black hole lists, etc.
    - particular hand-crafted text patterns

SpamAssassin Features:

- Basic (Naïve) Bayes spam probability
- Mentions: Generic Viagra
- Regex: millions of (dollar) ((dollar) NN,NNN,NNN.NN)
- Phrase: impress ... girl
- Phrase: ‘Prestigious Non-Accredited Universities’
- From: starts with many numbers
- Subject is all capitals
- HTML has a low ratio of text to image area
- Relay in RBL, http://www.mail-abuse.com/enduserinfo_rbl.html
- RCVD line looks faked
- http://spamassassin.apache.org/tests_3_3_x.html

Naive Bayes is Not So Naive

- Very fast learning and testing (basically just count words)
- Low storage requirements
- Very good in domains with many <u>equally important</u> features
- More robust to irrelevant features than many learning methods

  Irrelevant features cancel each other without affecting results

Naive Bayes is Not So Naive

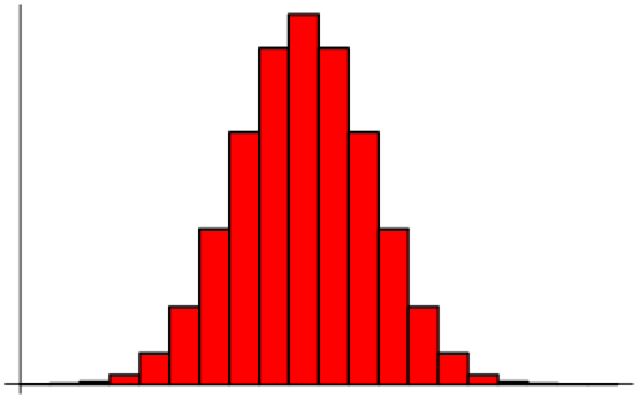
- More robust to concept drift (changing class definition over time)
- Naive Bayes won 1st and 2nd place in KDD-CUP 97 competition out of 16 systems

  Goal: Financial services industry direct mail response prediction: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- A good dependable baseline for text classification (but not the best)!

Multinomial, Poisson, Negative Binomial

- Within a class *y,* usual NB learns one parameter for each word *w:* p_w=Pr(W=w).
- ...entailing a particular distribution on word frequencies F.
- Learning two or more parameters allows more flexibility.



$$\Pr(F = f \mid p, N) = \binom{N}{f} p^f (1-p)^{N-f}$$

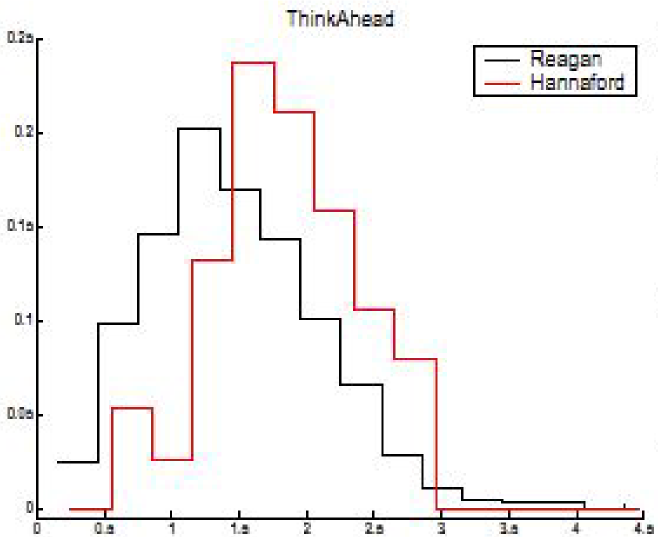
$$\text{Poisson}: \quad \Pr(F = f \mid \omega, \mu, N) = \frac{e^{-\omega\mu}(\omega\mu)^f}{f!}$$

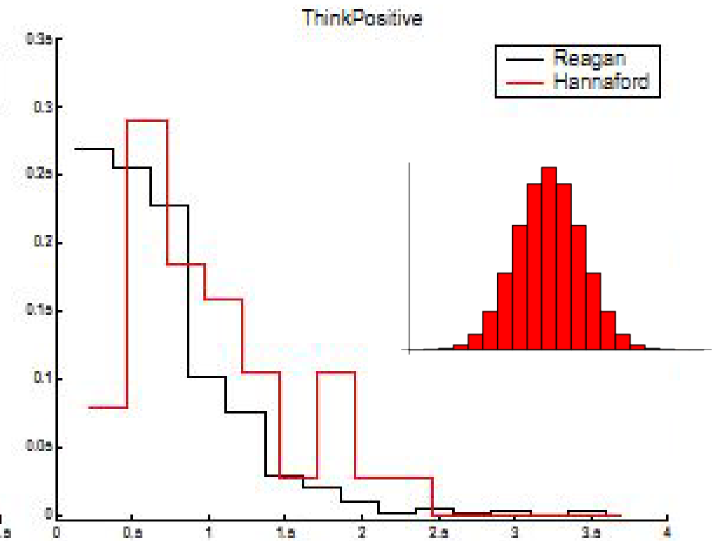
$$\text{Negative Binomial}: \quad \Pr(F = f \mid \omega, \mu, \delta, \kappa, N) = \frac{\Gamma(f+\kappa)}{f!\Gamma(\kappa)}(\omega\delta)^f (1+\omega\delta)^{-(f+\kappa)}$$

Multinomial, Poisson, Negative Binomial

- Binomial distribution does **not** fit **frequent** words or phrases very well. For some tasks frequent words are very important...e.g., classifying text by *writing style*.
  - "Who wrote Ronald Reagan's radio addresses?", Airoldi & Fienberg, 2003
- Problem is worse if you consider high-level features extracted from text
  - DocuScope tagger for "semantic markers"

Modeling Frequent Words





	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14+
Obsv.	146	171	124	81	55	42	20	13	9	3	8	3	1	1	2
Neg-Bin	167	152	116	82	56	37	25	16	10	7	4	3	2	1	1
Poisson	67	155	180	139	81	37	15	4	1						

“OUR” : Expected versus Observed Word Counts.

Extending Naive Bayes

- Putting this together:
  - for each *w,y* combination, build a histogram of frequencies for *w,* and fit Poisson to that as estimator for $Pr(F_w=f|Y=y)$.
  - to classify a new $x=w_1...w_n$, pick *y* with top *score*:

$$score(y, w_1 ... w_k) = \lg \Pr(y) + \sum_{i=1}^{n} \lg \Pr(F_{w_i} = f_{w_i} \mid y)$$

Dataset	Binomial	Poisson
5 Newsgroups (head)	2.300%	2.060%
5 Newsgroups (no-head)	4.680%	3.440%
Reagan's Addresses	8.500%	7.810%
Dealtime	12.068%	8.248%
Prostate Cancer	10.991%	11.920%
Nigerian Scam	0.501%	0.390%
Movie Reviews	31.643%	30.071%

More Complex Generative Models

[Blei, Ng & Jordan, JMLR, 2003]

- Within a class *y,* Naive Bayes constructs each *x:*
  - pick N words $w_1,\ldots,w_N$ according to $Pr(W|Y=y)$
- A more complex model for a class *y:*
  - pick K topics $z_1,\ldots,z_k$ and $\beta_{w,z}=Pr(W=w|Z=z)$ (according to some Dirichlet prior α)
  - for each document *x:*
    - pick a distribution of topics for *X,* in form of K parameters $\vartheta_{z,x}=Pr(Z=z|X=x)$
    - pick N words $w_1,\ldots,w_N$ as follows:
      - pick z_i according to $Pr(Z|X=x)$
      - pick w_i according to $Pr(W|Z=z_i)$

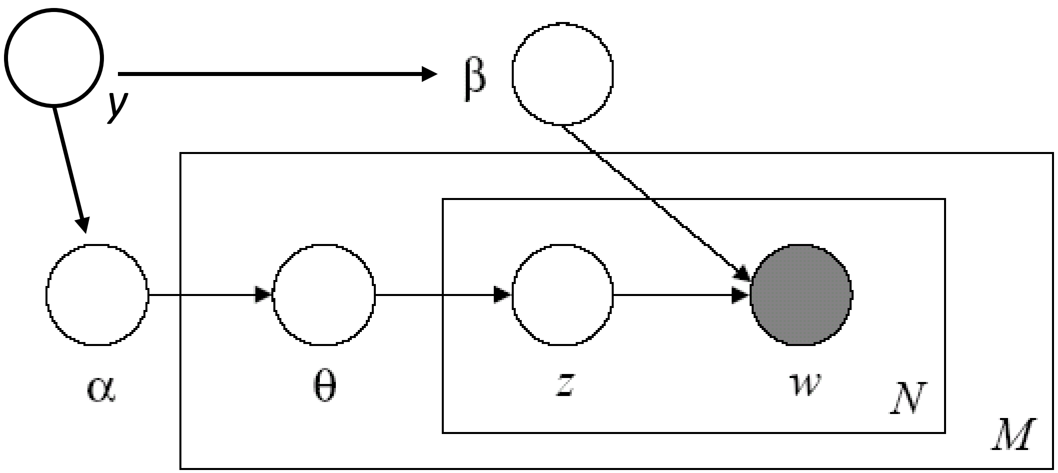
LDA Model: Example

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

More Complex Generative Models

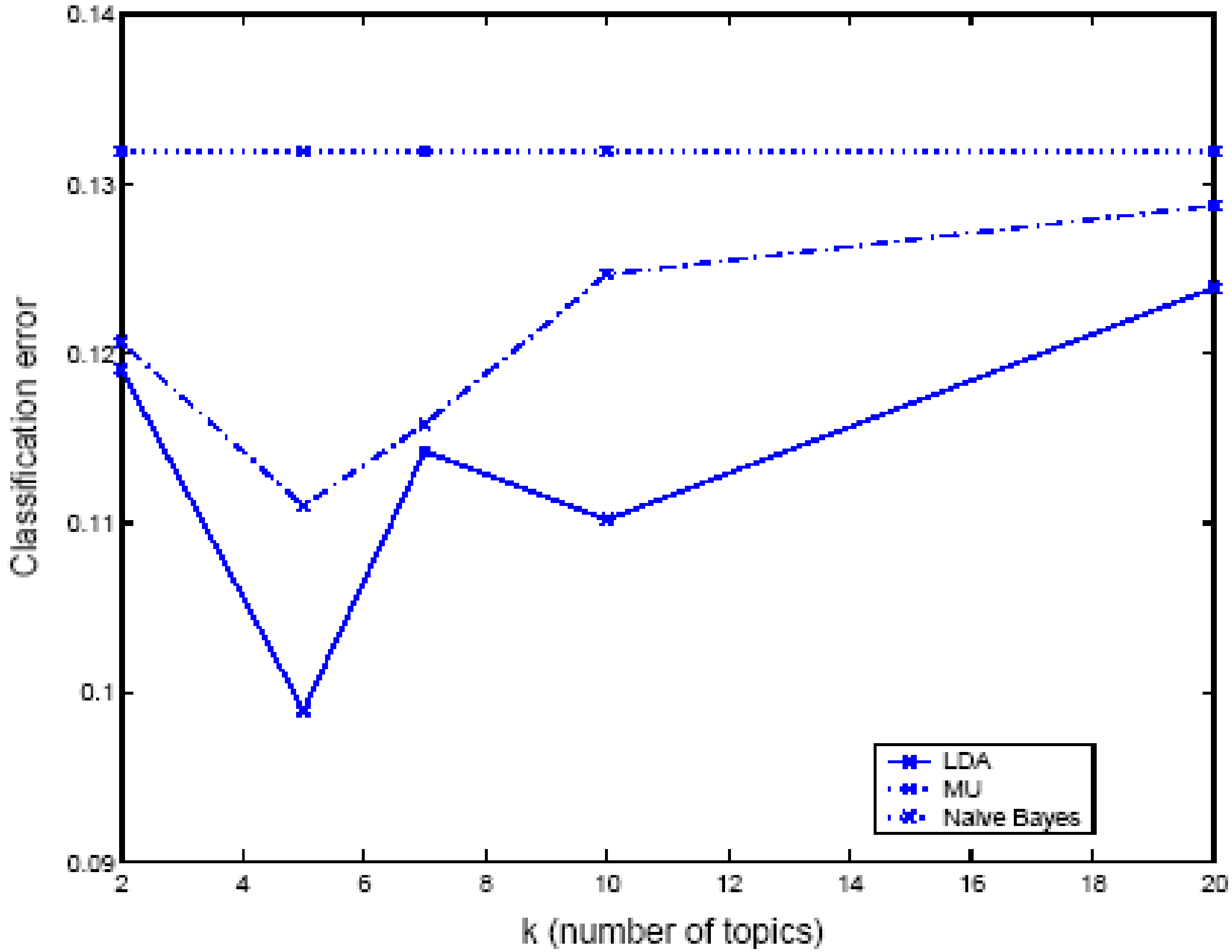
- pick K topics $z_1,\ldots,z_k$ and $\beta_{w,z}=Pr(W=w|Z=z)$ (according to some Dirichlet prior α)
- for each document $x_1,\ldots,x_M$:
  - pick a distribution of topics for x, in form of K parameters $\vartheta_{z,x}=Pr(Z=z|X=x)$
  - pick N words $w_1,\ldots,w_N$ as follows:
    - pick z_i according to $Pr(Z|X=x)$
    - pick w_i according to $Pr(W|Z=z_i)$



Learning:

- If we knew z_i for each w_i we could learn θ's and β's.
- The z_i's are *latent* variables (unseen).
- Learning algorithm:
  - pick β's randomly.
  - make "soft guess" at *zi's* for each *x*
  - estimate θ's and β's from "soft counts".
  - repeat last two steps until convergence

LDA Model: Experiment



Beyond Generative Models

Loglinear Conditional Models

Getting Less Naive

$$\Pr(y \mid x) = \frac{1}{Z} \Pr(y) \Pr(x \mid y)$$ where $Z = \Pr(x) = \sum_y \Pr(x \mid y)$

$$= \frac{1}{Z} \Pr(y) \prod_{j=1}^{n} \Pr(W_j = w_k \mid y)$$ for *j,k's* associated with *x*

$$= \frac{1}{Z} \hat{p}_y \prod_{j=1}^{k} \hat{p}_{j,k,y}$$ for *j,k's* associated with *x*

Estimate these based on
naive independence assumption

Getting Less Naive

$$\Pr(y \mid x) = \frac{1}{Z}\Pr(y)\Pr(x \mid y)$$

$$= \frac{1}{Z}\Pr(y)\prod_{j=1}^{n}\Pr(W_j = w_k \mid y)$$

"indicator function"
f(x,y)=1 if condition is true, f(x,y)=0 else

$$= \frac{1}{Z}\hat{p}_y\prod_{j=1}^{k}\hat{p}_{j,k,y} = \frac{1}{Z}\hat{p}_y\prod_{j,k,y}\exp\left((\ln\hat{p}_{j,k,y})\cdot\left\langle W_j = w_k, Y = y\right\rangle\right)$$

$$= \frac{1}{Z}\lambda_0\prod_{j,k,y}\exp\left(\lambda_{j,k,y}\cdot\left\langle W_j = w_k, Y = y\right\rangle\right)$$

Getting Less Naive

$$\Pr(y \mid x) = \frac{1}{Z}\Pr(y)\Pr(x \mid y)$$

$$= \frac{1}{Z}\Pr(y)\prod_{j=1}^{n}\Pr(W_j = w_k \mid y)$$

indicator function

$$= \frac{1}{Z}\hat{p}_y\prod_{j=1}^{k}\hat{p}_{j,k,y} = \frac{1}{Z}\hat{p}_y\prod_{j,k,y}\exp\left((\ln \hat{p}_{j,k,y})\cdot\left\langle W_j = w_k, Y = y\right\rangle\right)$$

simplified notation

$$= \frac{1}{Z}\lambda_0\prod_{j,k,y}\exp\left(\lambda_{j,k,y}\cdot f_{j,k,y}(x)\right)$$

Getting Less Naive

$$\Pr(y \mid x) = \frac{1}{Z}\Pr(y)\Pr(x \mid y)$$

$$= \frac{1}{Z}\Pr(y)\prod_{j=1}^{n}\Pr(W_j = w_k \mid y)$$

indicator function

$$= \frac{1}{Z}\hat{p}_y\prod_{j=1}^{k}\hat{p}_{j,k,y} = \frac{1}{Z}\hat{p}_y\prod_{j,k,y}\exp\left((\ln \hat{p}_{j,k,y})\cdot\left\langle W_j = w_k, Y = y\right\rangle\right)$$

simplified notation

$$= \frac{1}{Z}\lambda_0\prod_i\exp\left(\lambda_i \cdot f_i(x,y)\right)$$

Getting Less Naive

$$\Pr(y \mid x) = \frac{1}{Z}\lambda_0 \prod_i \exp\big(\lambda_i \cdot f_i(x,y)\big) \ = \frac{1}{Z}\lambda_0 \exp\left(\sum_i \lambda_i \cdot f_i(x,y)\right)$$

- each $f_i(x,y)$ indicates a property of x (word k at j with y)
- we want to pick each λ in a less naive way
- we have data in the form of (x,y) pairs
- one approach: pick λ's to *maximize*

$$\prod_i \Pr(y_i \mid x_i) \text{ or equivalently } \sum_{\mathrm{i}} \lg \Pr(y_i \mid x_i)$$

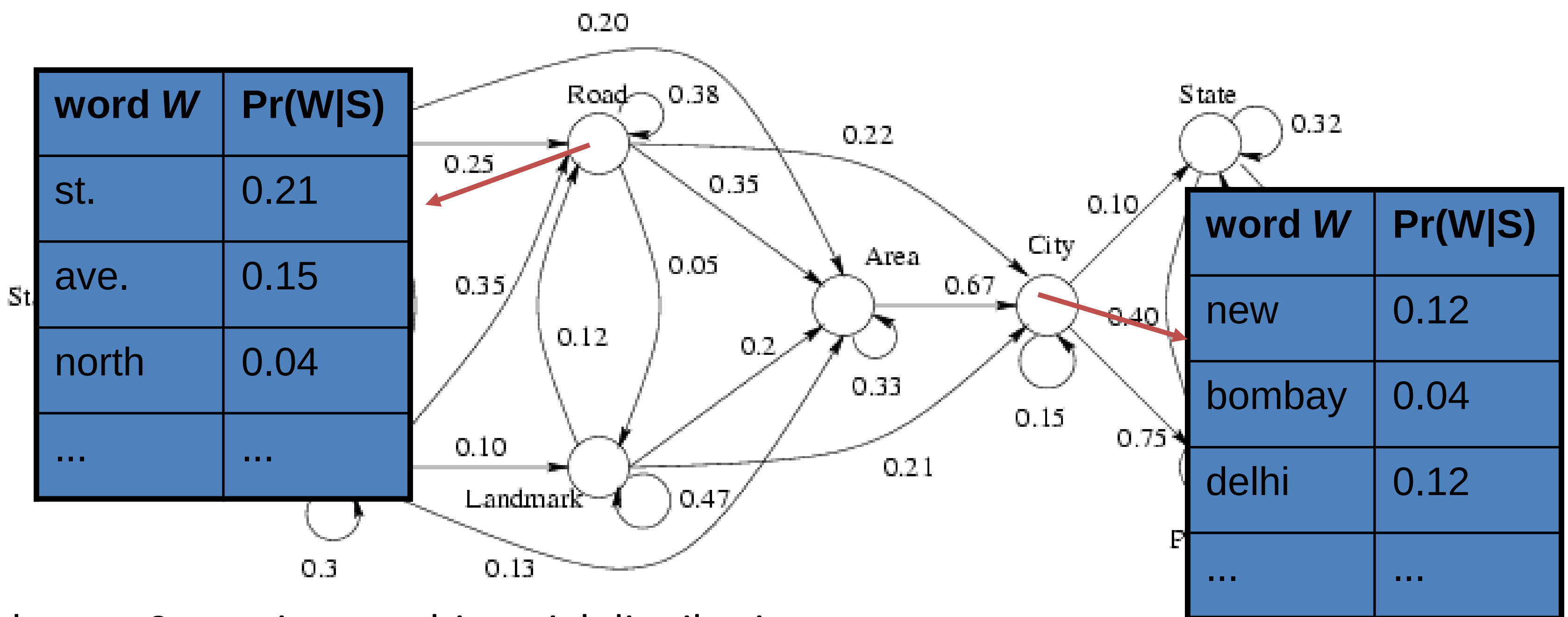
Getting Less Naive

- Putting this together:
  - define some likely properties $f_i(x)$ of an x,y pair
  - assume $$\Pr(y \mid x) = \frac{1}{Z} \lambda_0 \prod_i \exp\big(\lambda_i \cdot f_i(x,y)\big)$$
  - learning: optimize λ's to maximize $$\sum_{\mathrm{i}} \lg \Pr(y_i \mid x_i)$$
    - gradient descent works ok
      - recent work (Malouf, CoNLL 2001) shows that certain heuristic approximations to Newton's method converge surprisingly fast
    - need to be careful about sparsity
      - most features are zero
    - avoid "overfitting": maximize $$\sum_{\mathrm{i}} \lg \Pr(y_i \mid x_i) - c\Big(\sum_k \lambda_k\Big)$$

HMMs and CRFs

Hidden Markov Models

- The representations discussed so far ignore the fact that text is **sequential.**
- One sequential model of text is a *Hidden Markov Model*.



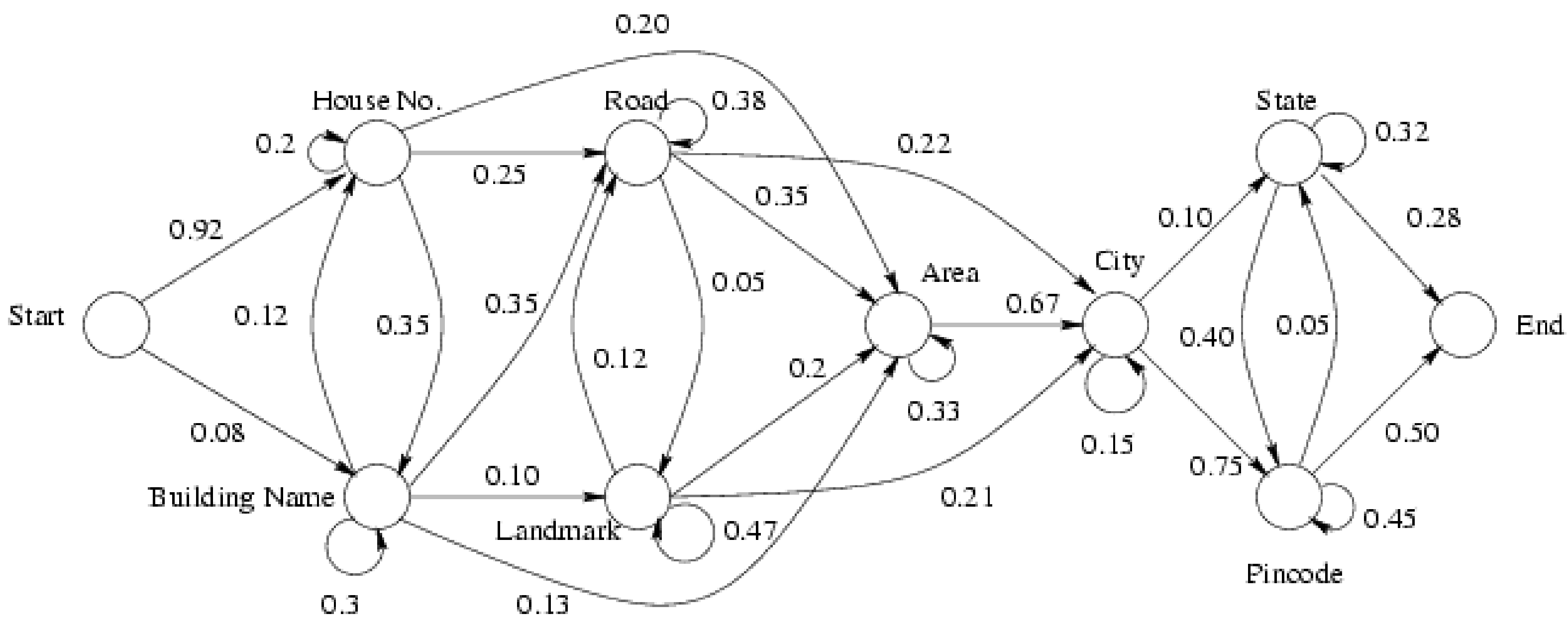
word *W*	Pr(W|S)
st.	0.21
ave.	0.15
north	0.04
...	...

word *W*	Pr(W|S)
new	0.12
bombay	0.04
delhi	0.12
...	...

Each state *S* contains a multinomial distribution

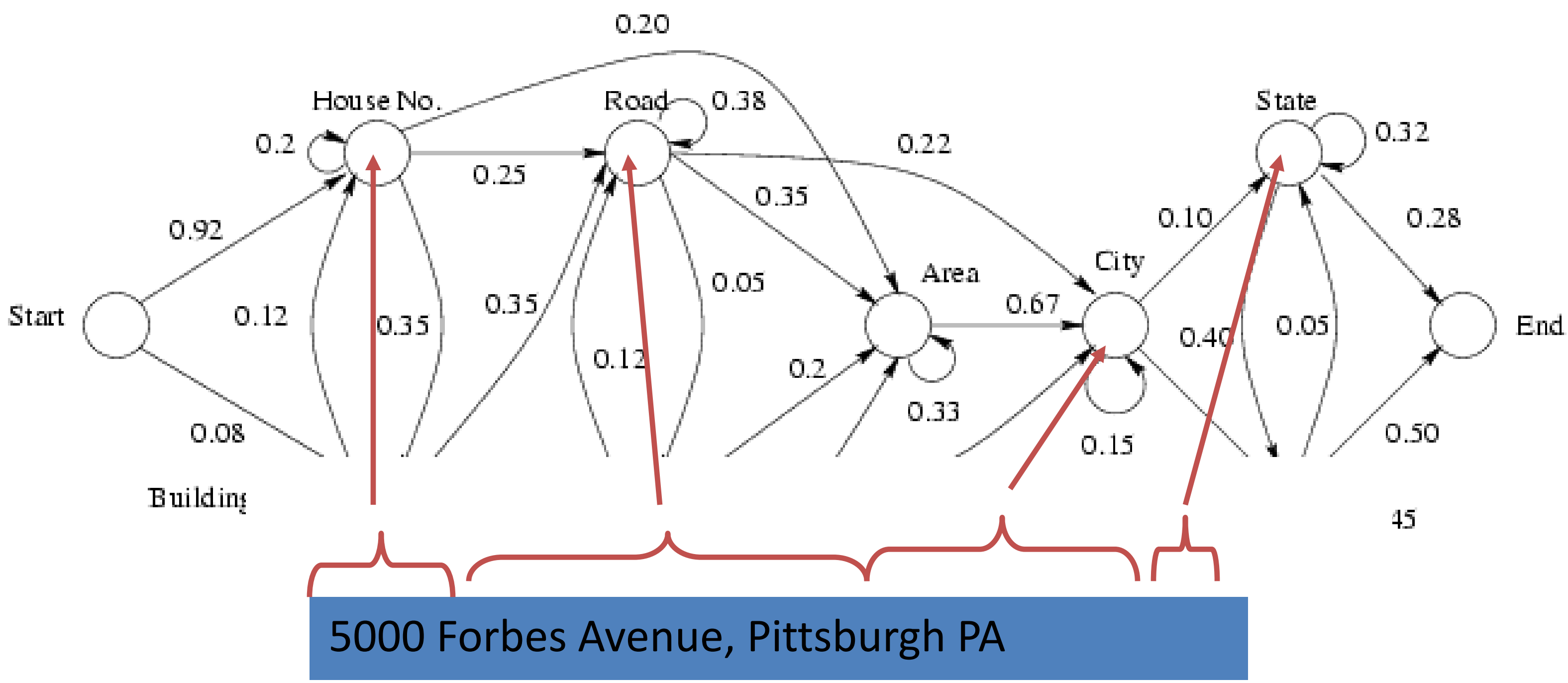
Hidden Markov Models

- A simple process to generate a sequence of words:
  - begin with $i=0$ in state $S_0=START$
  - pick S_{i+1} according to $Pr(S'|S_i)$, and w_i according to $Pr(W|S_{i+1})$
  - repeat unless $S_n=END$



Hidden Markov Models

- Learning is **simple** if you know $(w_1,\ldots,w_n)$ and $(s_1,\ldots,s_n)$
  - Estimate Pr(W|S) and Pr(S'|S) with counts
- This is quite reasonable for some tasks!
  - Here: training data could be pre-segmented addresses



Hidden Markov Models

- Classification is **not** simple.
  - Want to find $s_1,\ldots,s_n$ to maximize $Pr(s_1,\ldots,s_n \mid w_1,\ldots,w_n)$
  - Cannot afford to try all $|S|^N$ combinations.
  - However there is a trick—the *Viterbi* algorithm

time t	*Prob(S_t=s| $w_1,\ldots,w_n$)*						
	START	*Building*	*Number*	*Road*	…	*END*	
t=0	1.00	0.00	0.00	0.00	…	0.00	
t=1	0.00	0.02	0.98	0.00	…	0.00	5000
t=2	0.00	0.01	0.00	0.96	…	0.00	Forbes
…	…	…	…	…	…	…	Ave

Hidden Markov Models

- *Viterbi* algorithm:
  - each line of table depends **only** on the word at that line, and the line **immediately above it**
  - ➔ can compute $Pr(S_t=s \mid w_1,\ldots,w_n)$ quickly
  - a similar trick works for $\text{argmax}[s_1,\ldots,s_n]\ Pr(s_1,\ldots,s_n \mid w_1,\ldots,w_n)$

time t	*Prob(S_t=s| w_1,…,w_n)*						
	START	*Building*	*Number*	*Road*	…	*END*	
t=0	1.00	0.00	0.00	0.00	…	0.00	
t=1	0.00	0.02	0.98	0.00	…	0.00	5000
t=2	0.00	0.01	0.00	0.96	…	0.00	Forbes
…	…	…	…	…	…	…	Ave

Hidden Markov Models

Extracting Names from Text

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

Richard Stallman, founder of the Free Software Foundation, countered saying…

→

Microsoft Corporation
CEO
Bill Gates
Microsoft
Gates
Microsoft
Bill Veghte
Microsoft
VP
Richard Stallman
founder
Free Software Foundation

Hidden Markov Models

Extracting Names from Text

October 14, 2002, 4:00 a.m. PT

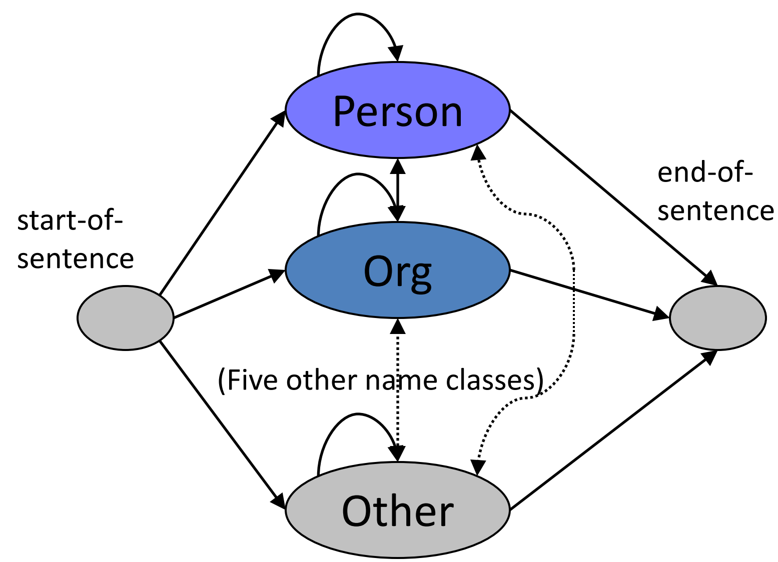
For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

Richard Stallman, founder of the Free Software Foundation, countered saying…

Nymble (BBN's 'Identifinder')



[Bikel et al, MLJ 1998]

Getting Less Naive with HMMs

- Naive Bayes model:
  - generate class *y*
  - generate words $w_1,..,w_n$ from $Pr(W|Y=y)$
- HMM model:
  - generate states $y_1,...,y_n$
  - generate words $w_1,..,w_n$ from $Pr(W|Y=y_i)$
- Conditional version of Naive Bayes
  - set parameters to maximize

$$\sum_i \lg \Pr(y_i \mid x_i)$$

- Conditional version of HMMs
  - *conditional random fields* (CRFs)

Getting Less Naive with HMMs

- Conditional random fields:
  - training data is set of pairs $(y_1 \ldots y_n, x_1 \ldots x_n)$
  - you define a set of *features* $f_j(i, y_i, y_{i-1}, x_1 \ldots x_n)$
    - for HMM-like behavior, use indicators for <$Y_i=y_i$ *and* $Y_{i-1}=y_{i-1}$> and <$X_i=x_i$>
  - I'll define

$$F_j(\vec{x}, \vec{y}) = \sum_i f_j(i, y_i, y_{i-1}, \vec{x})$$

Recall for maxent:

$$\Pr(y \mid x) = \frac{1}{Z} \lambda_0 \exp\left(\sum_i \lambda_i \cdot f_i(x, y) \right)$$

For a CRF:

$$\Pr(\vec{y} \mid \vec{x}) = \frac{1}{Z} \lambda_0 \exp\left(\sum_j \lambda_j \cdot F_j(\vec{x}, \vec{y}) \right)$$

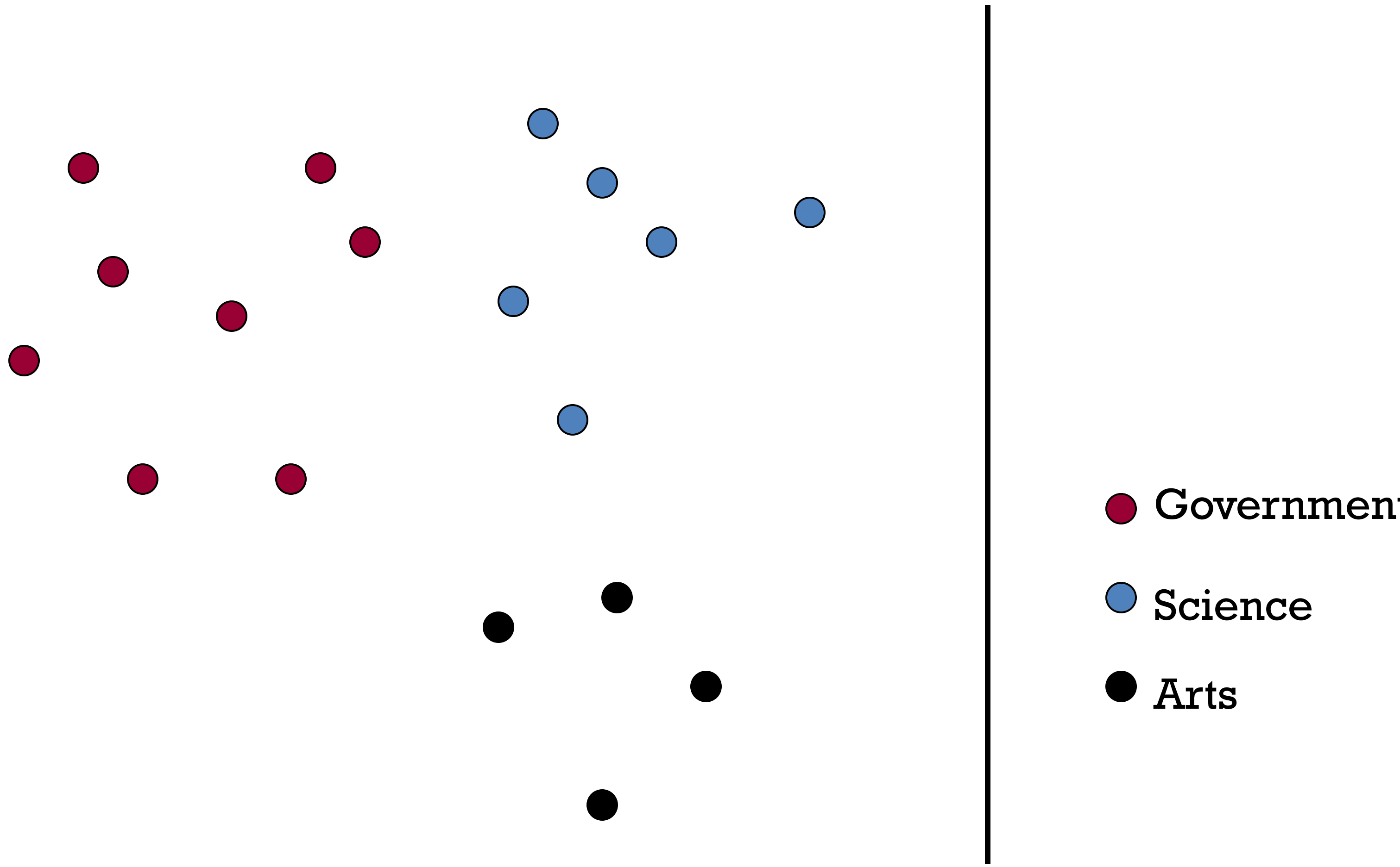
Learning requires HMM-computations to compute gradient for optimization, and Viterbi-like computations to classify.

Beyond Probabilities

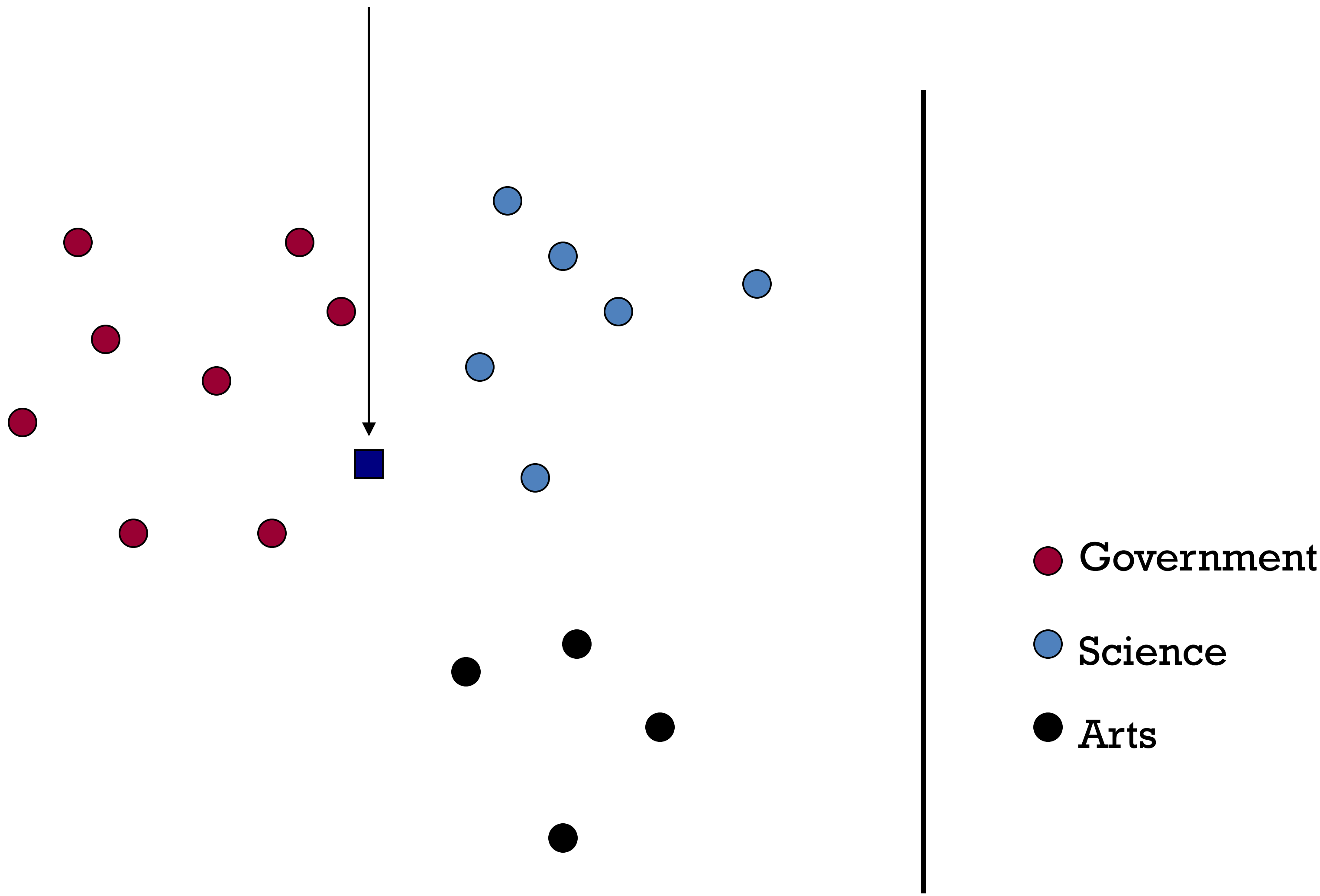
Classification Using Vector Spaces

- In vector space classification, training set corresponds to a labeled set of points (equivalently, vectors)
- Premise 1: Documents in the same class form a contiguous region of space
- Premise 2: Documents from different classes don’t overlap (much)
- Learning a classifier: build surfaces to delineate classes in the space

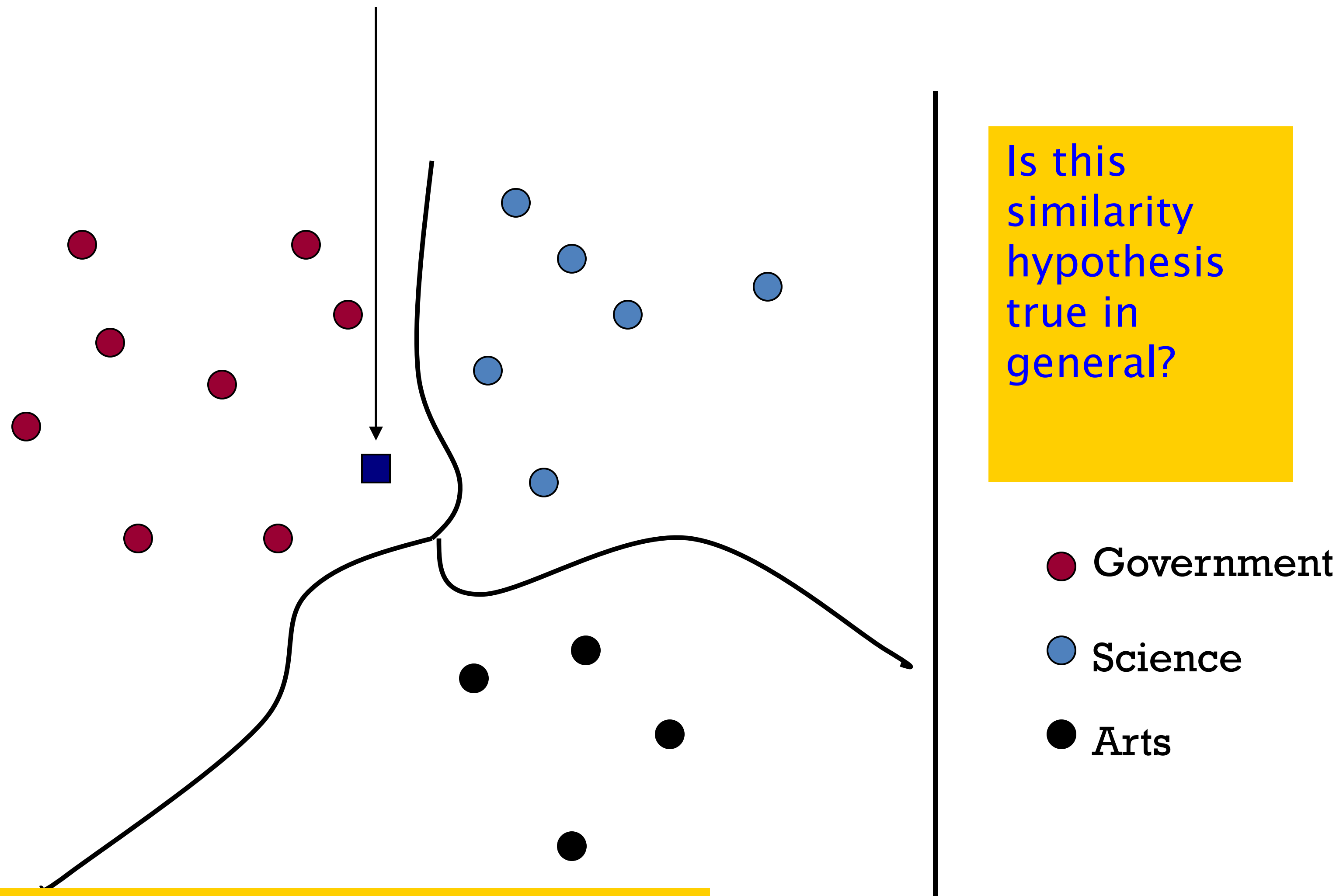
Documents in a Vector Space



Test Document of what class?



Test Document = Government



Is this similarity hypothesis true in general?

Our focus: how to find good separators

Definition of centroid

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- Where D_c is the set of all documents that belong to class c and $v(d)$ is the vector space representation of d.

- *Note that centroid will in general* not *be a unit vector even when the inputs are unit vectors.*

Rocchio classification

- Rocchio forms a simple representative for each class: the centroid/prototype
- Classification: nearest prototype/centroid
- It does not guarantee that classifications are consistent with the given training data

Rocchio classification

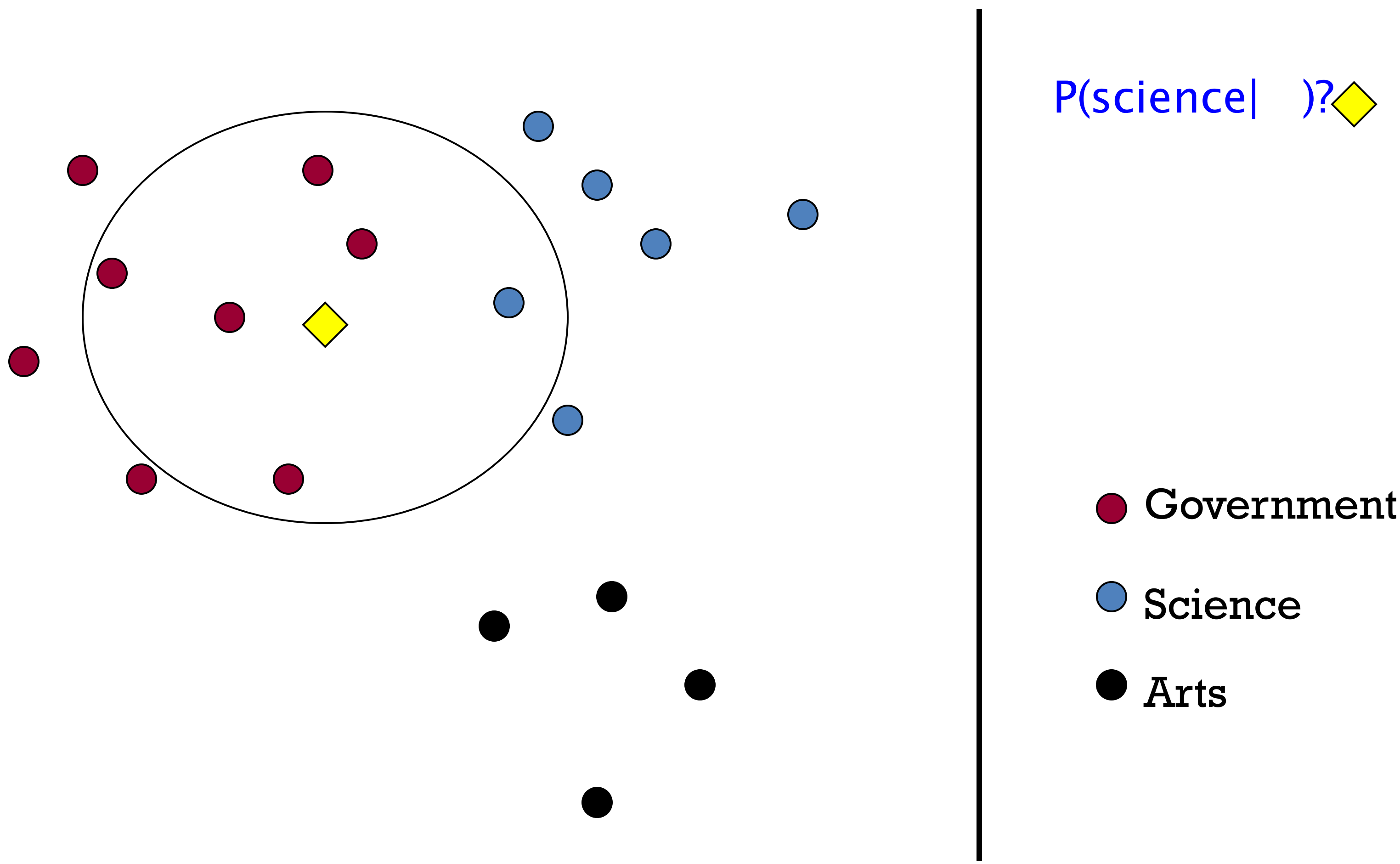
- Little used outside text classification
  - It has been used quite effectively for text classification
  - But in general worse than Naïve Bayes
- Again, cheap to train and test documents

k Nearest Neighbor Classification

- kNN = *k* Nearest Neighbor

- To classify a document *d*:
- Define *k*-neighborhood as the *k* nearest neighbors of *d*
- Pick the majority class label in the *k*-neighborhood

Example: k=6 (6NN)



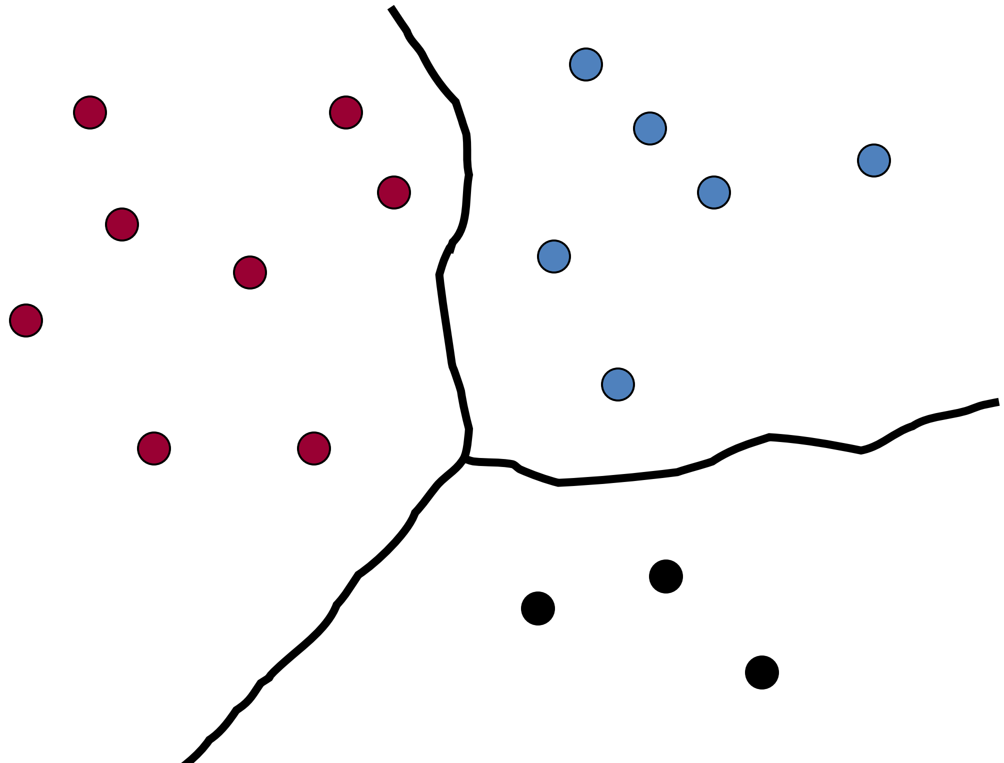
Nearest-Neighbor Learning

- Learning: store the labeled training examples *D*
- Testing instance *x (under 1NN)*:
  – Compute similarity between *x* and all examples in *D*.
  – Assign *x* the category of the most similar example in *D*.
- Does not compute anything beyond storing the examples
- Also called:
  – Case-based learning
  – Memory-based learning
  – Lazy learning
- Rationale of kNN: contiguity hypothesis

k Nearest Neighbor

- Using only the closest example (1NN) subject to errors due to:
  - A single atypical example.
  - Noise (i.e., an error) in the category label of a single training example.
- More robust: find the *k* examples and return the majority category of these *k*
- *k* is typically odd to avoid ties; 3 and 5 are most common

kNN decision boundaries



Boundaries are in principle arbitrary surfaces – but usually polyhedra

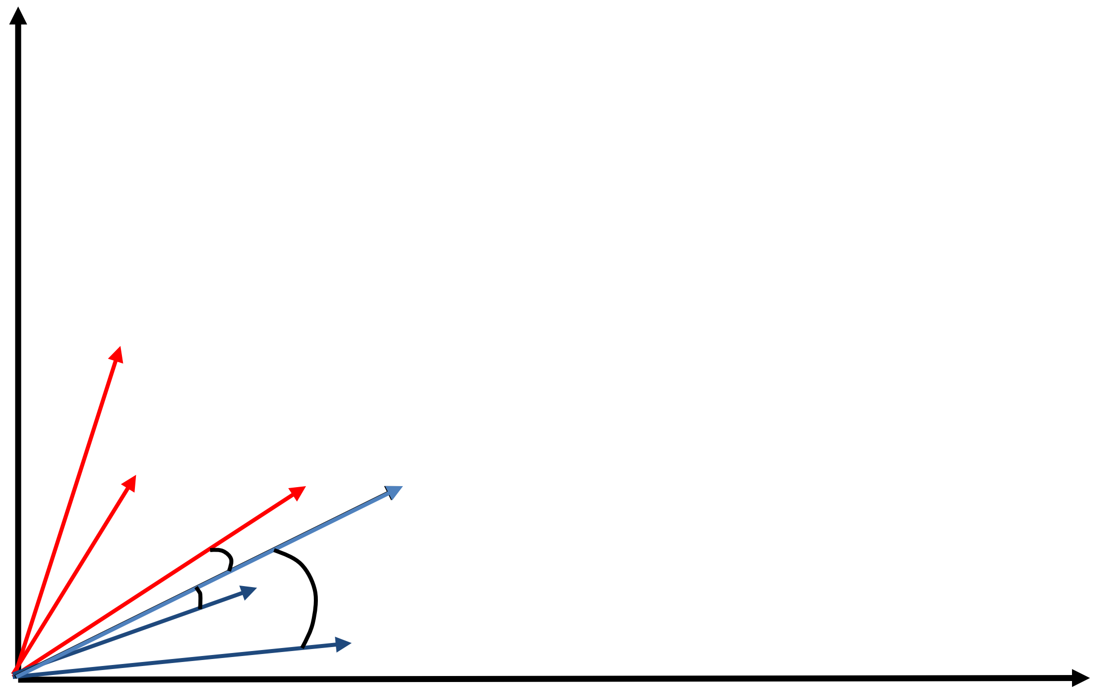
Government

Science

Arts

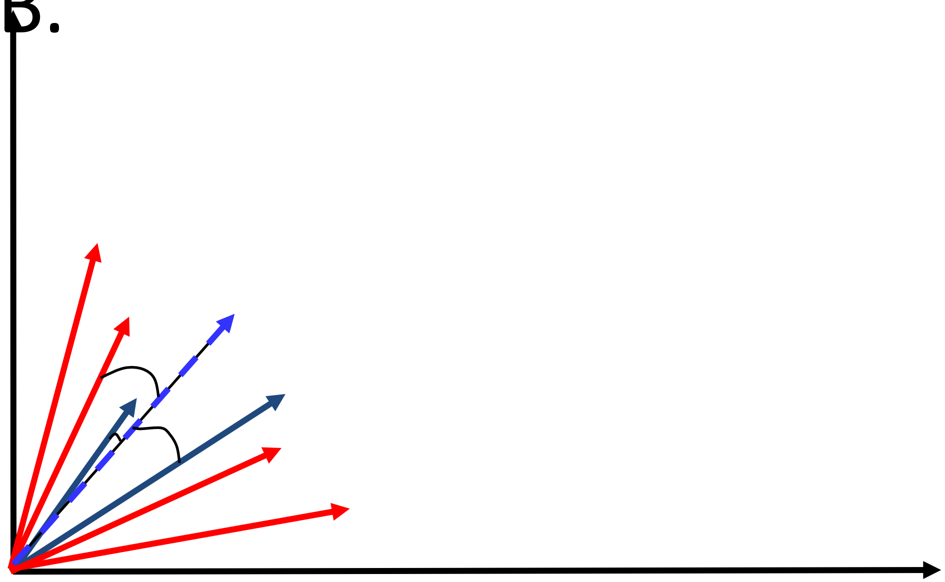
kNN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, Rocchio, etc.)

Illustration of 3 Nearest Neighbor for Text Vector Space



3 Nearest Neighbor vs. Rocchio

- Nearest Neighbor tends to handle polymorphic categories better than Rocchio/NB.



kNN: Discussion

- No feature selection necessary
- No training necessary
- Scales well with large number of classes
  - Don't need to train *n* classifiers for *n* classes
- Classes can influence each other
  - Small changes to one class can have ripple effect
- May be expensive at test time
- In most cases it's more accurate than NB or Rocchio

Bias vs. capacity – notions and terminology

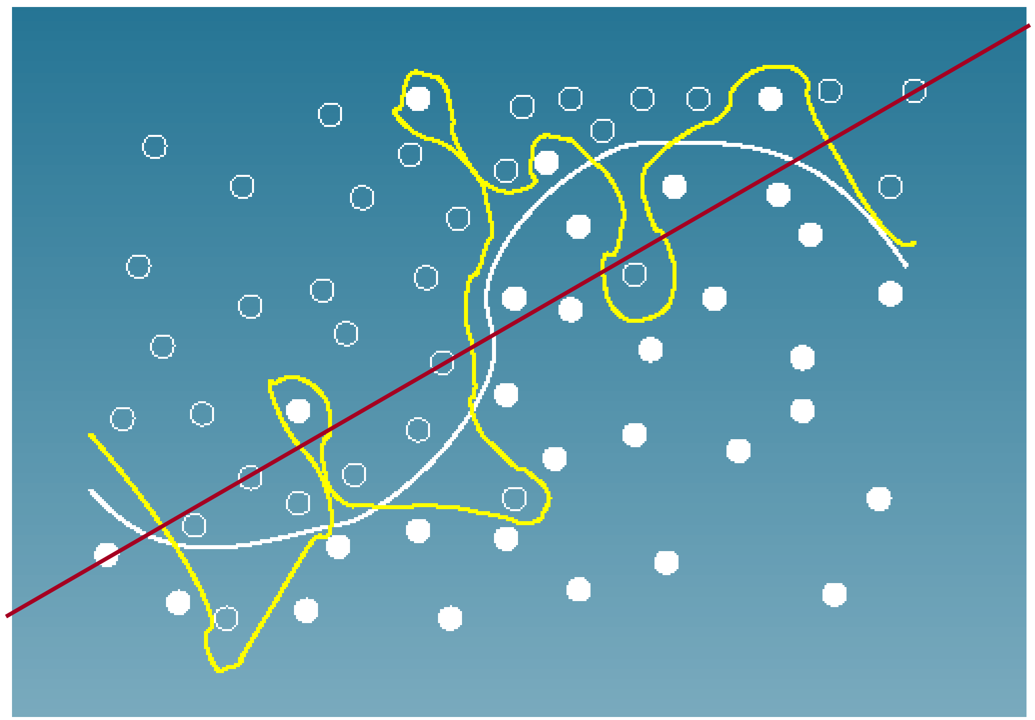
- Consider asking a botanist: Is an object a tree?
  - Too much *capacity*, low *bias*
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., different # of leaves)
  - Not enough capacity, high bias
    - Lazy botanist
    - Says "yes" if the object is green
  - You want the middle ground

(Example due to C. Burges)

kNN vs. Naive Bayes

- Bias/Variance tradeoff
  - Variance ≈ Capacity
- kNN has high variance and low bias.
  - Infinite memory
- NB has low variance and high bias.
  - Linear decision surface (hyperplane – see later)

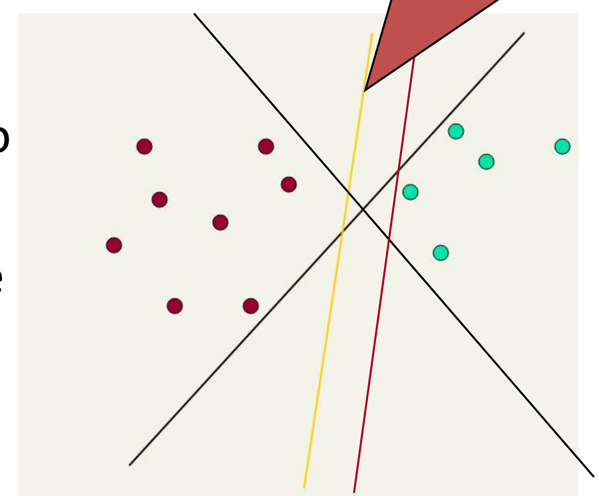
Bias vs. variance: Choosing the correct model capacity



Linear classifiers: Which Hyperplane?

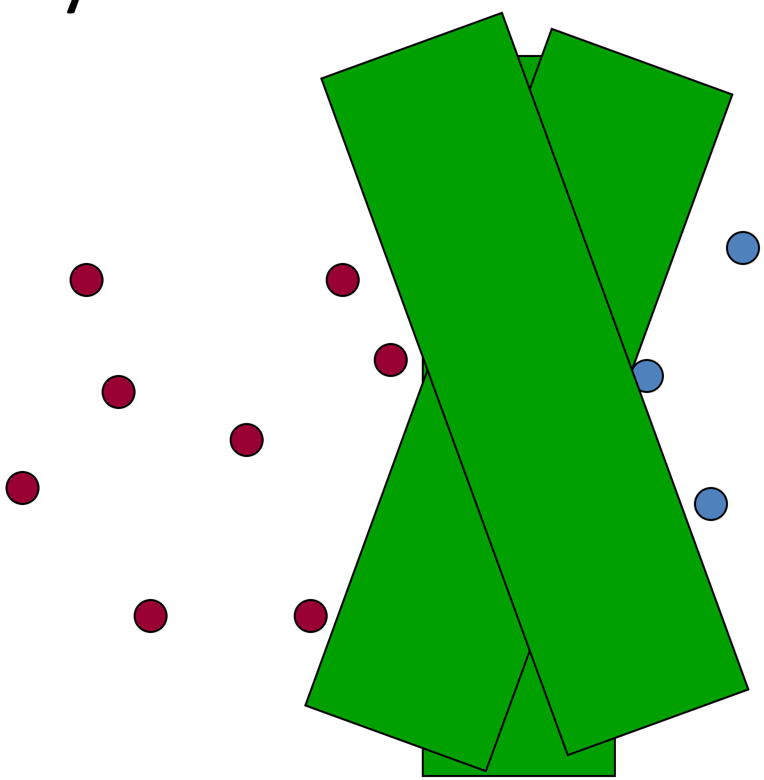
- Lots of possible choices for *a, b, c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- A Support Vector Machine (SVM) finds an optimal* solution.
  - Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:
$ax + by - c = 0$



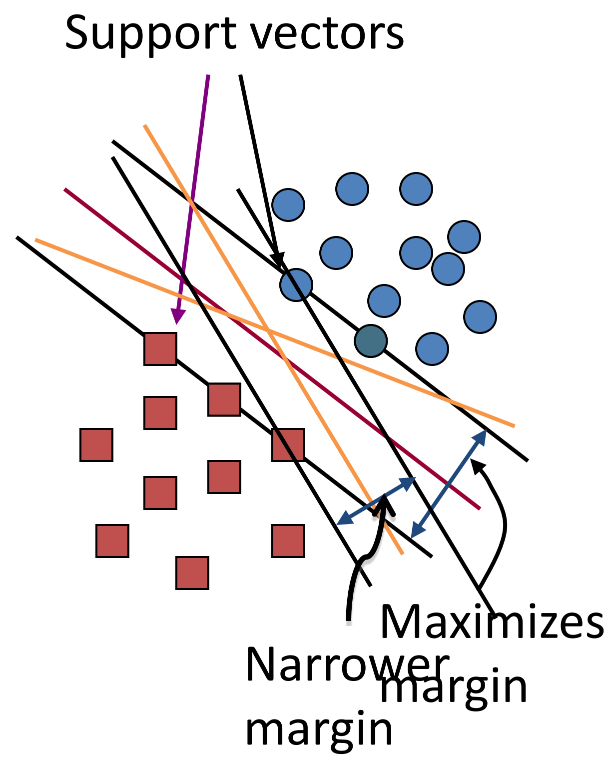
Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*



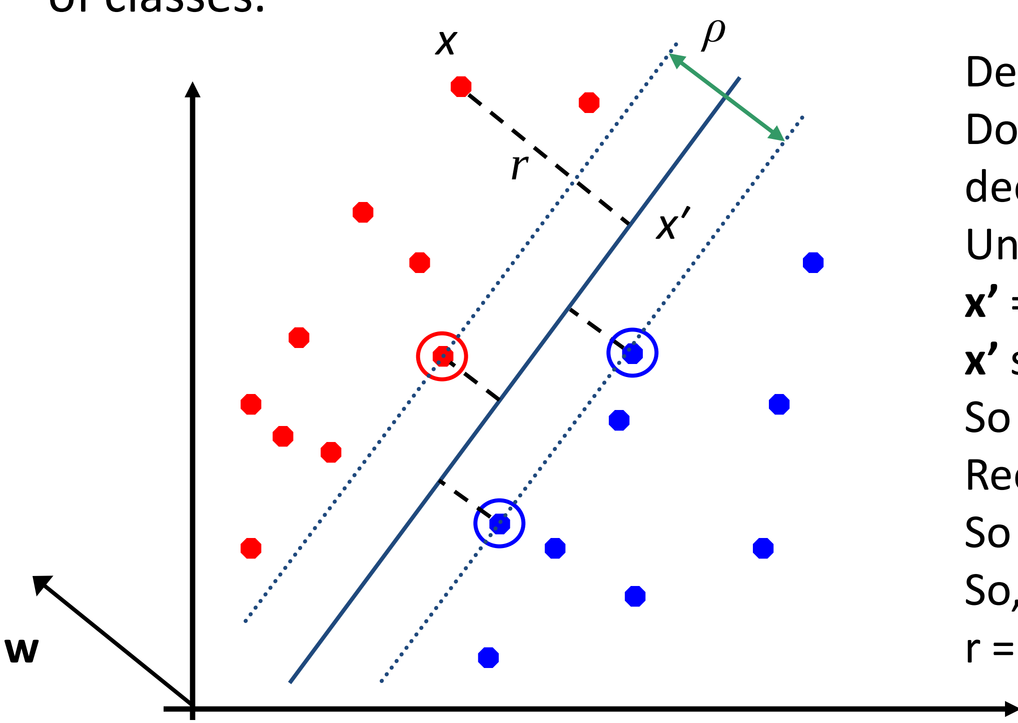
*but other discriminative methods often perform very similarly

Maximum Margin: Formalization

- **w**: decision hyperplane normal vector
- $\mathbf{x}_i$: data point *i*
- y_i: class of data point *i* (+1 or -1) NB: Not 1/0
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- Functional margin of $\mathbf{x}_i$ is: $y_i\,(\mathbf{w}^T\mathbf{x}_i + b)$
- The functional margin of a dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin
- **Problem:** we can increase this margin simply by scaling **w**, **b**….

Geometric Margin

- Distance from example to the separator is $$r = y\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** *ρ* of the separator is the width of separation between support vectors of classes.



Derivation of finding *r:*
Dotted line **x’** − **x** is perpendicular to decision boundary so parallel to **w**.
Unit vector is **w**/|**w**|, so line is r**w**/|**w**|.
x’ = **x** – yr**w**/|**w**|.
x’ satisfies $\mathbf{w}^T\mathbf{x'}$ + b = 0.
So $\mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$
Recall that |**w**| = sqrt($\mathbf{w}^T\mathbf{w}$).
So $\mathbf{w}^T\mathbf{x} - yr|\mathbf{w}| + b = 0$
So, solving for r gives:
$r = y(\mathbf{w}^T\mathbf{x} + b)/|\mathbf{w}|$

Linear SVM Mathematically

The linearly separable case

- Assume that the functional margin of each data item is at least 1, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b \geq 1 \quad \text{if } y_i = 1$$
$$\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

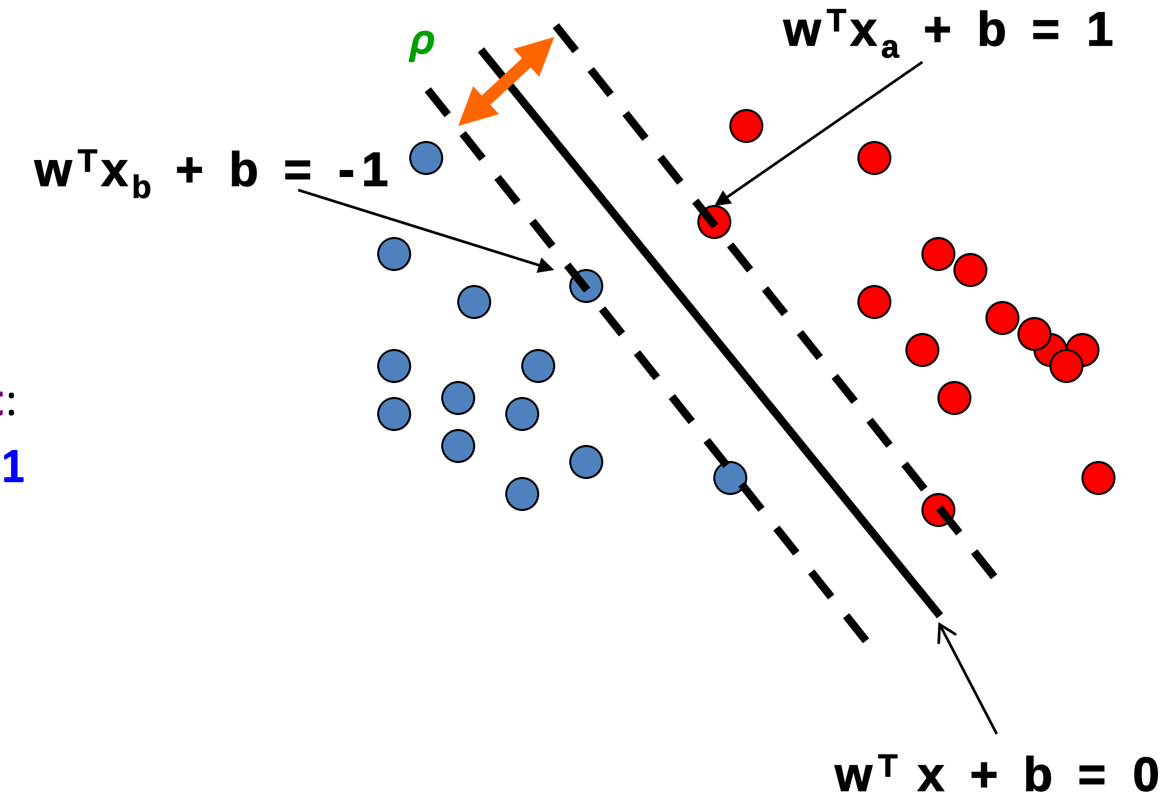
$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

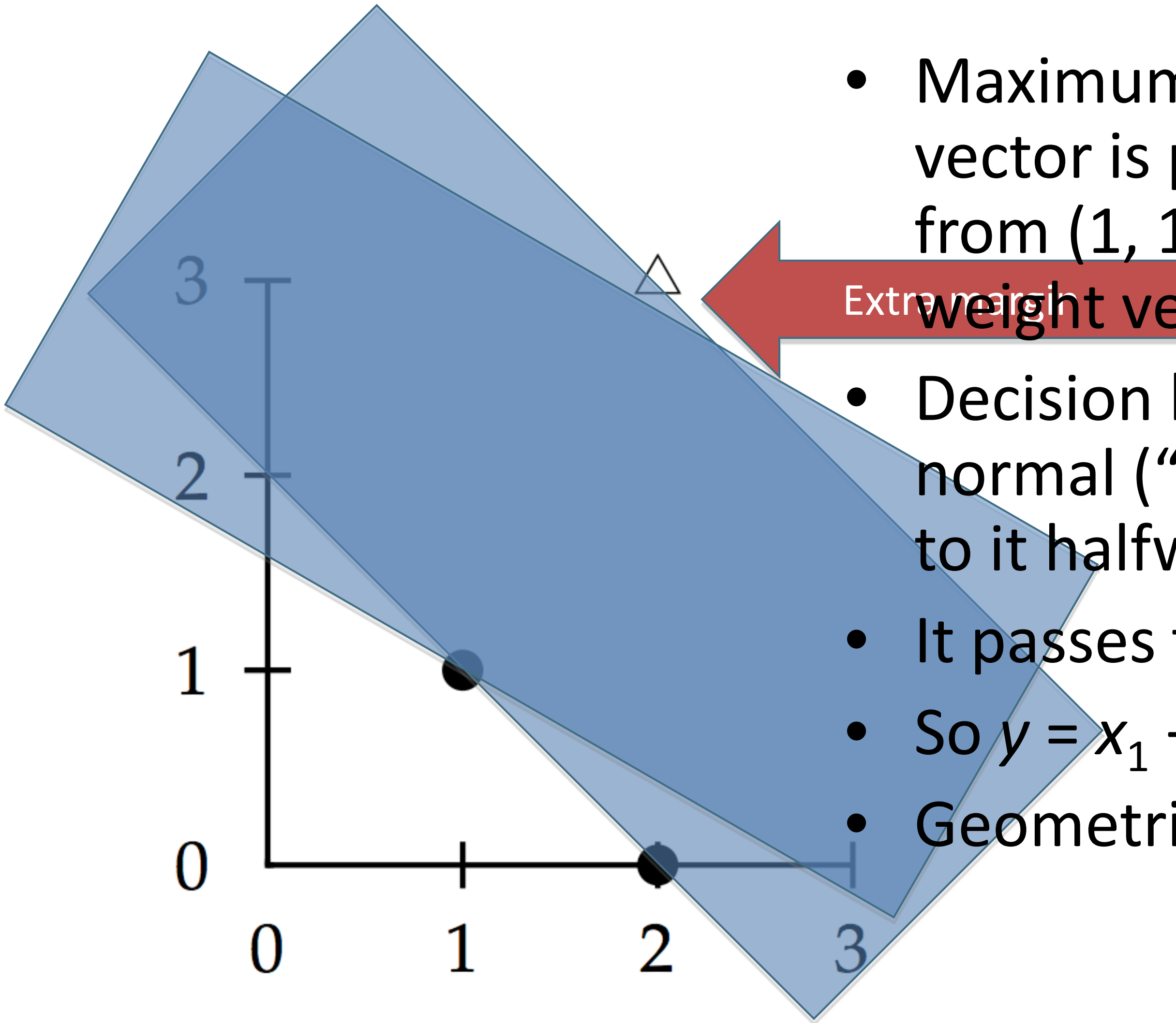
- **Hyperplane**
  $\mathbf{w}^T \mathbf{x} + b = 0$

- **Extra scale constraint**:
  $\min_{i=1,\ldots,n} |w^T x_i + b| = 1$

- This implies:
  $w^T(x_a - x_b) = 2$
  $\rho = \|x_a - x_b\|_2 = 2/\|w\|_2$

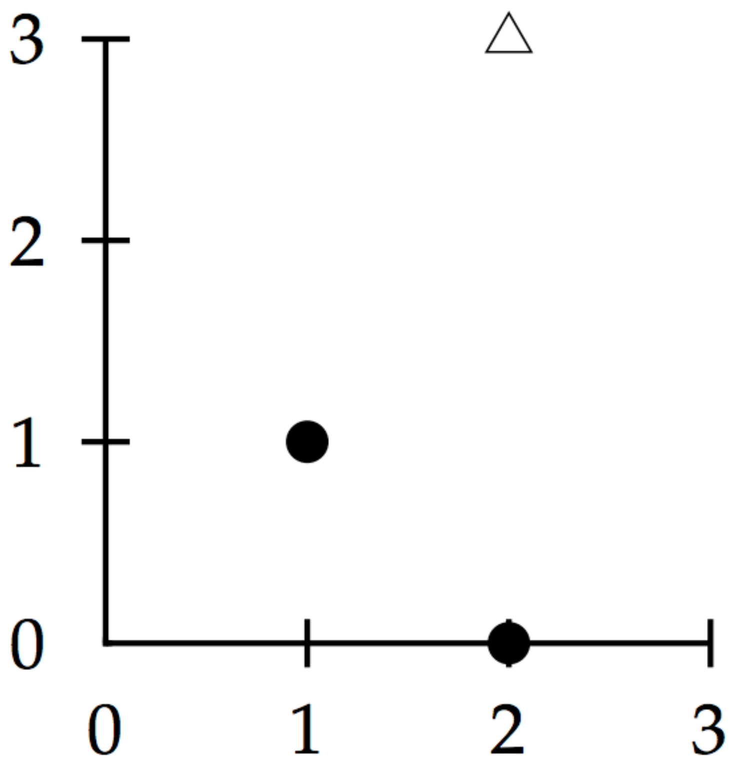


Worked example: Geometric margin



- Maximum margin weight vector is parallel to line from (1, 1) to (2, 3). So weight vector is (1, 2).
- Decision boundary is normal ("perpendicular") to it halfway between.
- It passes through (1.5, 2)
- So $y = x_1 + 2x_2 - 5.5$
- Geometric margin is $\sqrt{5}$

Worked example: Functional margin



- Let's minimize w given that $y_i(w^T x_i + b) \geq 1$
- Constraint has = at SVs; $w = (a, 2a)$ for some a
- $a+2a+b = -1 \qquad 2a+6a+b = 1$
- So, $a = 2/5$ and $b = -11/5$
  Optimal hyperplane is:
  $w = (2/5, 4/5)$ and $b = -11/5$
- Margin ρ is $2/|w|$
  $= 2/\sqrt{(4/25+16/25)}$
  $= 2/(2\sqrt{5}/5) = \sqrt{5}$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

> Find **w** and *b* such that
>
> $\rho = \dfrac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}\ ,\ y_i)\}$
>
> $\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i=1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min **w** = max 1/ **w**):

> Find **w** and *b* such that
>
> $\mathbf{\Phi}(\mathbf{w}) = ½\ \mathbf{w}^T\mathbf{w}$ is minimized;
>
> and for all $\{(\mathbf{x_i}\ , y_i)\}$: $y_i\ (\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\boldsymbol{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\boxed{\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w^T x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0}$$

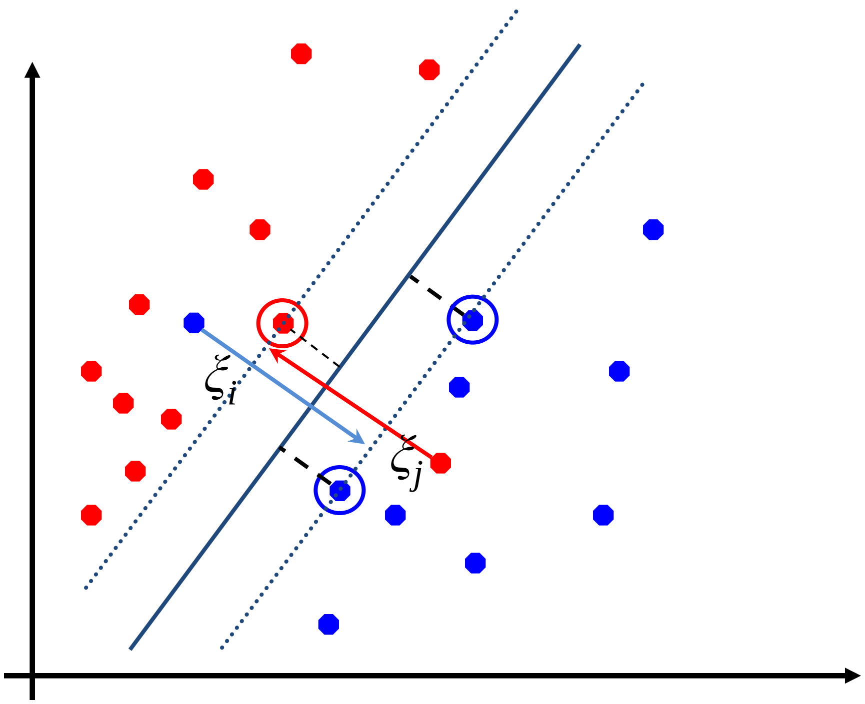
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$\boxed{f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i}^\mathbf{T} \mathbf{x} + b}$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^\mathbf{T}\mathbf{x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

> Find $\mathbf{w}$ and b such that
> $\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
> $y_i\, (\mathbf{w^T x_i} + \mathrm{b}) \geq 1$

- The new formulation incorporating slack variables:

> Find $\mathbf{w}$ and b such that
> $\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
> $y_i\, (\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter *C* can be viewed as a way to control overfitting
  - A regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

$$\boxed{\begin{array}{l}\text{Find } \alpha_1 \ldots \alpha_N \text{ such that} \\ \mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j} \text{ is maximized and} \\ (1)\ \ \Sigma\alpha_i y_i = 0 \\ (2)\ \ 0 \leq \alpha_i \leq C \text{ for all } \alpha_i \end{array}}$$

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

$$\boxed{\begin{array}{l}\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \\ b = y_k(1 - \xi_k) - \mathbf{w}^{\mathbf{T}}\mathbf{x}_k \text{ where } k = \underset{k'}{\operatorname{argmax}}\ \alpha_{k'}\end{array}}$$

w is not needed explicitly for classification!

$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^{\mathbf{T}}\mathbf{x} + b}$$

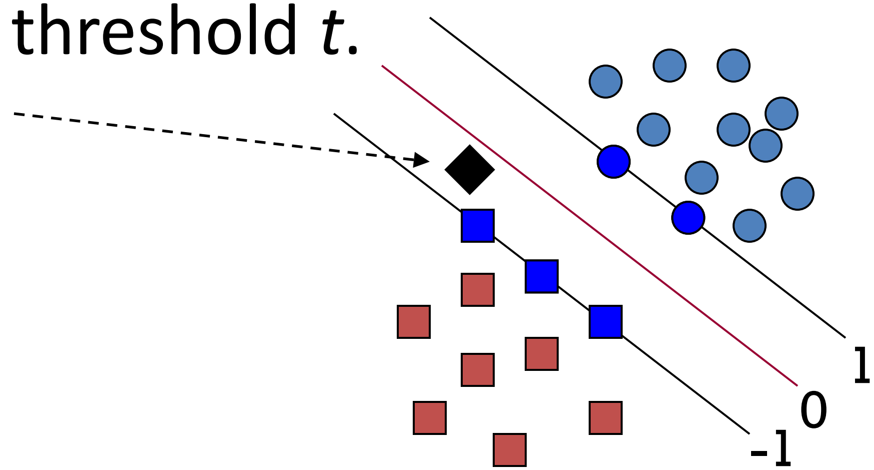
Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $\mathbf{w}^T\mathbf{x} + b = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$
    - Decide class based on whether < or > 0
  - Can set confidence threshold t.

Score > t: yes

Score < -t: no

Else: don't know



Linear SVMs: Summary

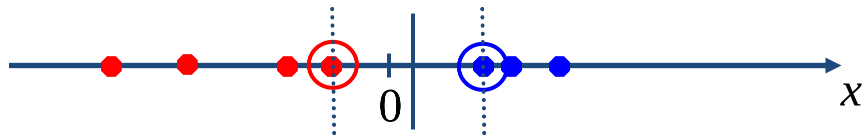
- The classifier is a *separating hyperplane.*
- The most "important" training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x_i}$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

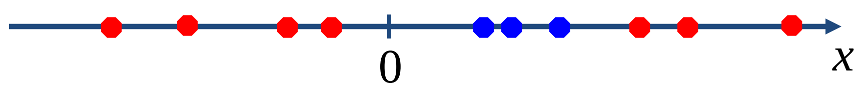
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x_i}^{\mathbf{T}}\mathbf{x}} + b$$

Non-linear SVMs

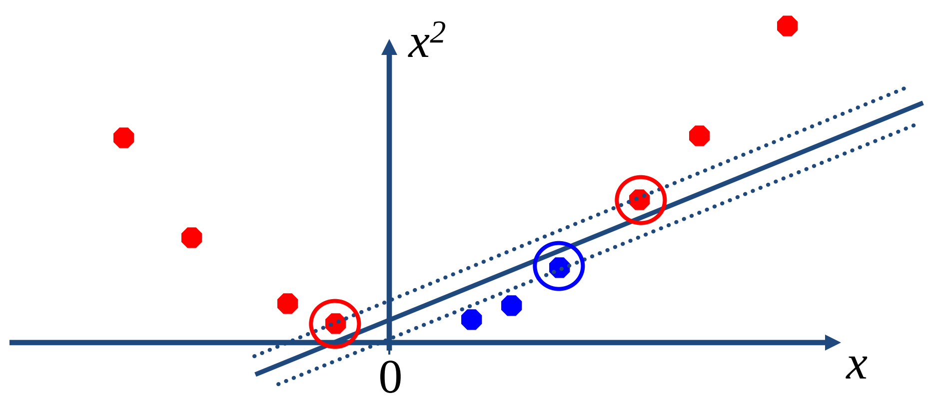
- Datasets that are linearly separable (with some noise) work out great:



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

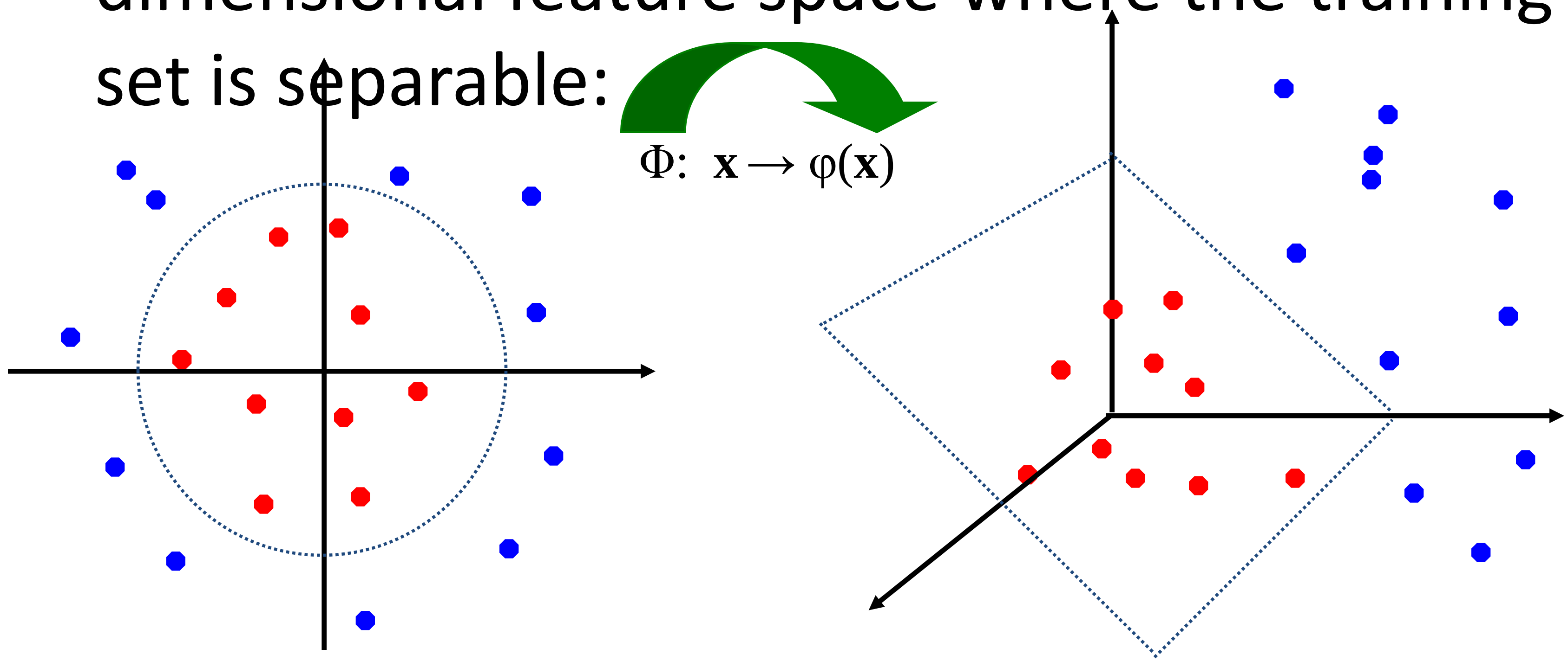


- How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T\, [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

  $= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ where $\phi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial **$K(x,z) = (1+x^T z)^d$**
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

- Have $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$ n

Summary: Representation of Text Categorization Attributes

- Representations of text are usually very high dimensional
- High-bias algorithms that prevent overfitting should generally work best in high-dimensional space
- For most text categorization tasks, there are many relevant features and many irrelevant ones

Which classifier do I use for a given text classification problem?

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the data?
  - How stable is the problem over time?
    - For an unstable problem, its better to use a simple and robust classifier.

Conclusions

- There are huge number of applications for text categorization.
- Bag-of-words representations generally work better than you’d expect
  - Naive Bayes are fastest to learn and easiest to implement
  - Linear classifiers that like wide margins tend to do best.
  - Probabilistic classifications are sometimes important.
- Non-topical text categorization (e.g., sentiment detection) is much less well studied than topic text categorization.

Some Resources for Text Categorization

- Surveys and talks:
  - *Machine Learning in Automated Text Categorization*, Fabrizio Sebastiani, *ACM Computing Surveys*, 34(1):1-47, 2002 , http://faure.isti.cnr.it/~fabrizio/Publications/ACMCS02.pdf
  - (Naive) Bayesian Text Classification for Spam Filtering http://www.daviddlewis.com/publications/slides/lewis-2004-0507-spam-talk-for-casa-marketing-draft5.ppt (and other related talks)
- Software:
  - Minorthird: toolkit for extraction and classification of text: http://minorthird.sourceforge.net
  - Rainbow: fast Naive Bayes implementation of text-preprocessing in C: http://www.cs.cmu.edu/~mccallum/bow/rainbow/
  - SVM Light: free support vector machine well-suited to text: http://svmlight.joachims.org/
- Test Data:
  - Datasets: http://www.cs.cmu.edu/~tom/, and http://www.daviddlewis.com/resources/testcollections

Thank You!