Classification Algorithms

UCSB 293S, 2017. T. Yang

Some of slides based on R. Mooney (UT Austin)

Table of Content

Classification

- Given:
  - A description of an instance, x
  - A fixed set of categories (classes): $C=\{c_1, c_2, \ldots c_n\}$
  - Training examples
- Determine:
  - The category of x: $h(x) \in C$, where $h(x)$ is a classification function
- A training example is an instance x, paired with its correct category $c(x)$: $\langle x, c(x) \rangle$

Sample Learning Problem

- Instance space: <size, color, shape>
  - size $\in$ {small, medium, large}
  - color $\in$ {red, blue, green}
  - shape $\in$ {square, circle, triangle}
- C = {positive, negative}
- D:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

General Learning Issues

- Many hypotheses are usually consistent with the training data.
- Bias
  - Any criteria other than consistency with the training data that is used to select a hypothesis.
- Classification accuracy (% of instances classified correctly).
  - Measured on independent test data.
- Training time (efficiency of training algorithm).
- Testing time (efficiency of subsequent classification).

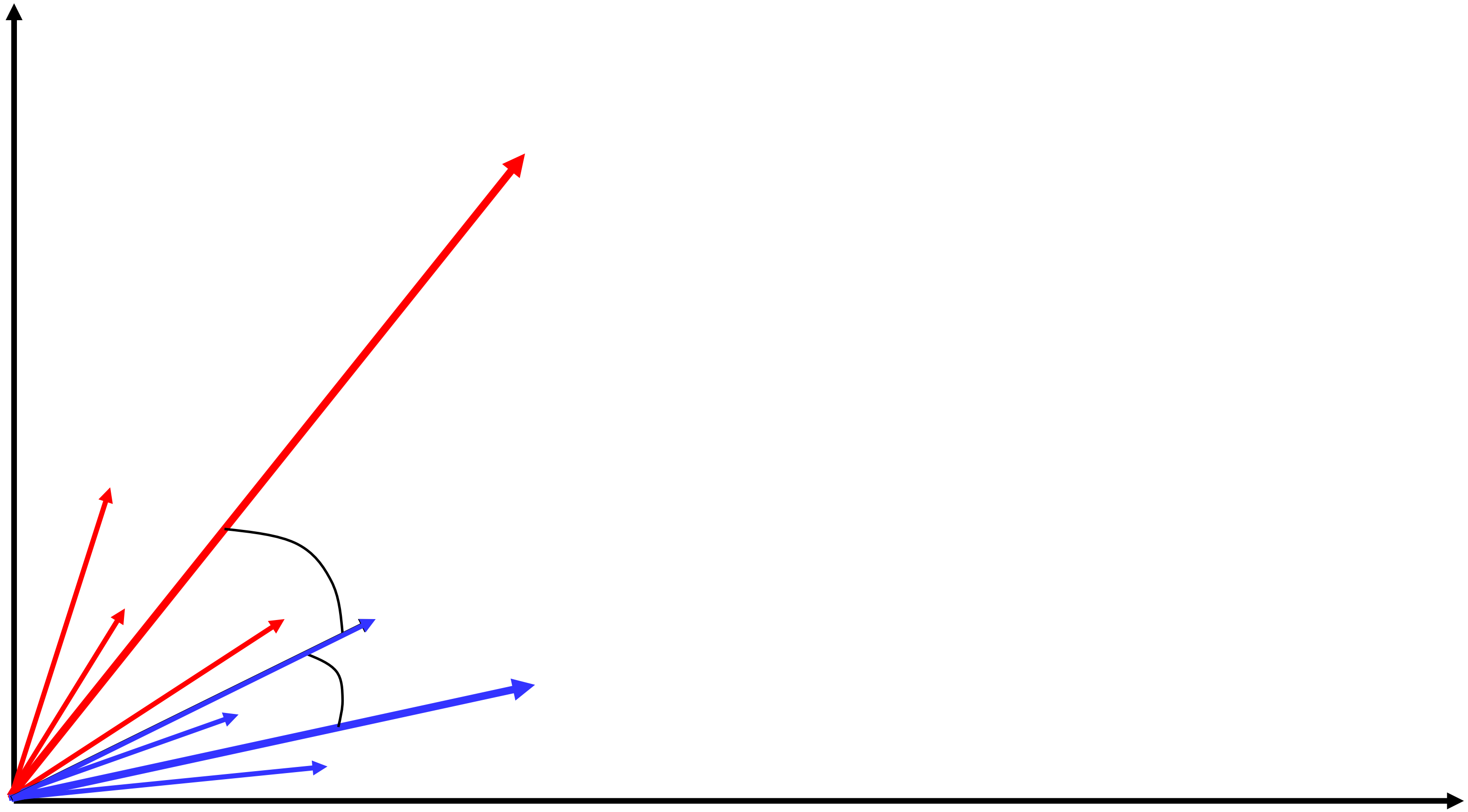
Text Categorization/Classification

- Assigning documents to a fixed set of categories.
- Applications:
  - Web pages
    - Recommending/ranking
    - category classification
  - Newsgroup Messages
    - Recommending
    - spam filtering
  - News articles
    - Personalized newspaper
  - Email messages
    - Routing
    - Prioritizing
    - Folderizing
    - spam filtering

Learning for Classification

- Manual development of text classification functions is difficult.
- Learning Algorithms:
  - **Bayesian (naïve)**
  - Neural network
  - **Rocchio**
  - Rule based (Ripper)
  - **Nearest Neighbor (case based)**
  - **Support Vector Machines (SVM)**
  - **Decision trees**
  - Boosting algorithms

Illustration of Rocchio method



Rocchio Algorithm

Assume the set of categories is $\{c_1, c_2, \ldots c_n\}$

Training:

Each doc vector is the frequency normalized TF/IDF term vector.

For i from 1 to n

Sum all the document vectors in c_i to get prototype vector $\mathbf{p}_i$

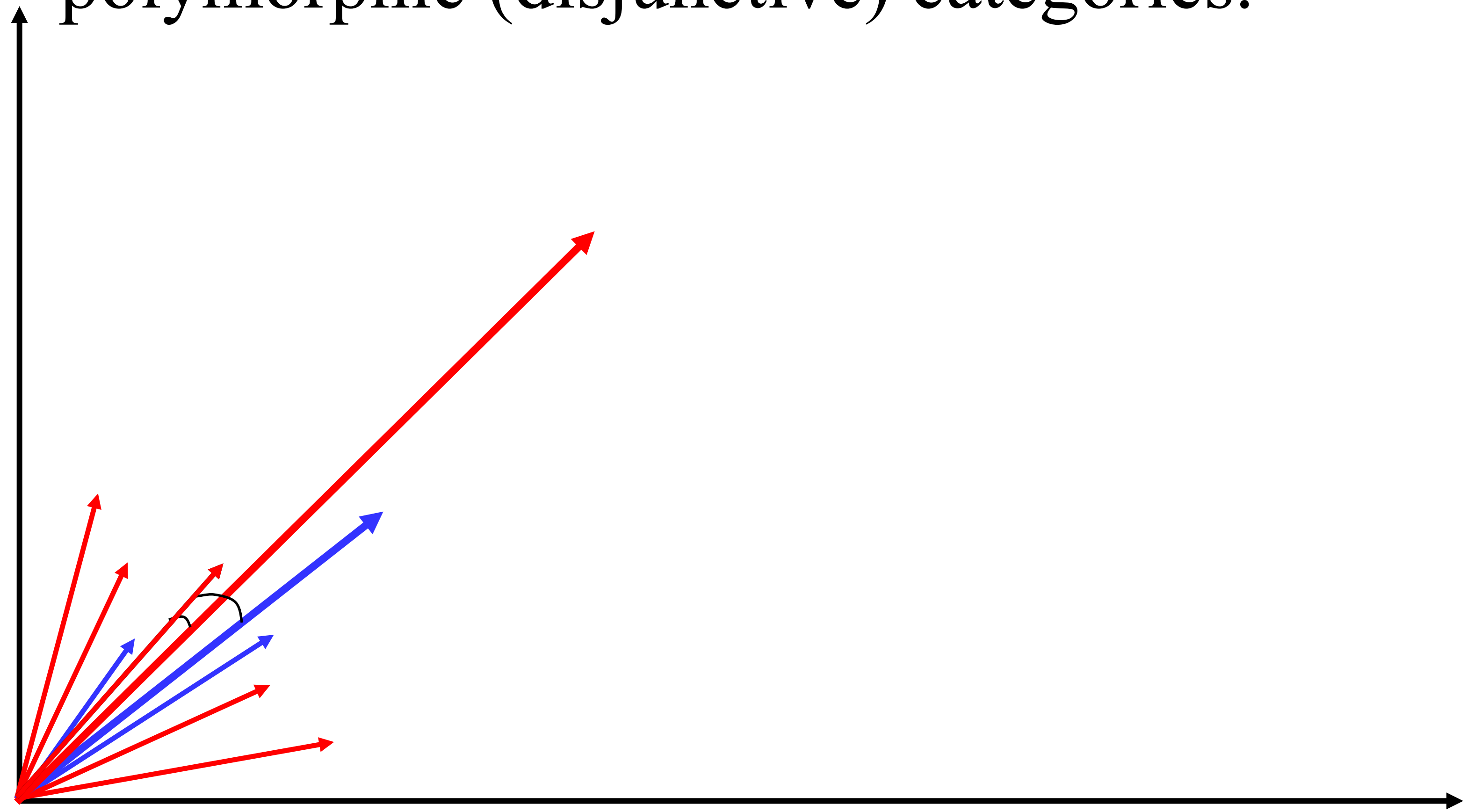
Testing: Given document x

Compute the cosine similarity of x with each prototype vector.

Select one with the highest similarity value and return its category

Rocchio Anomoly

- Prototype models have problems with polymorphic (disjunctive) categories.



Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in *D*.
- Testing instance *x*:
  - Compute similarity between *x* and all examples in *D*.
  - Assign *x* the category of the most similar example in *D*.
- Does not explicitly compute a generalization or category prototypes.
- Also called:
  - Case-based
  - Memory-based
  - Lazy learning

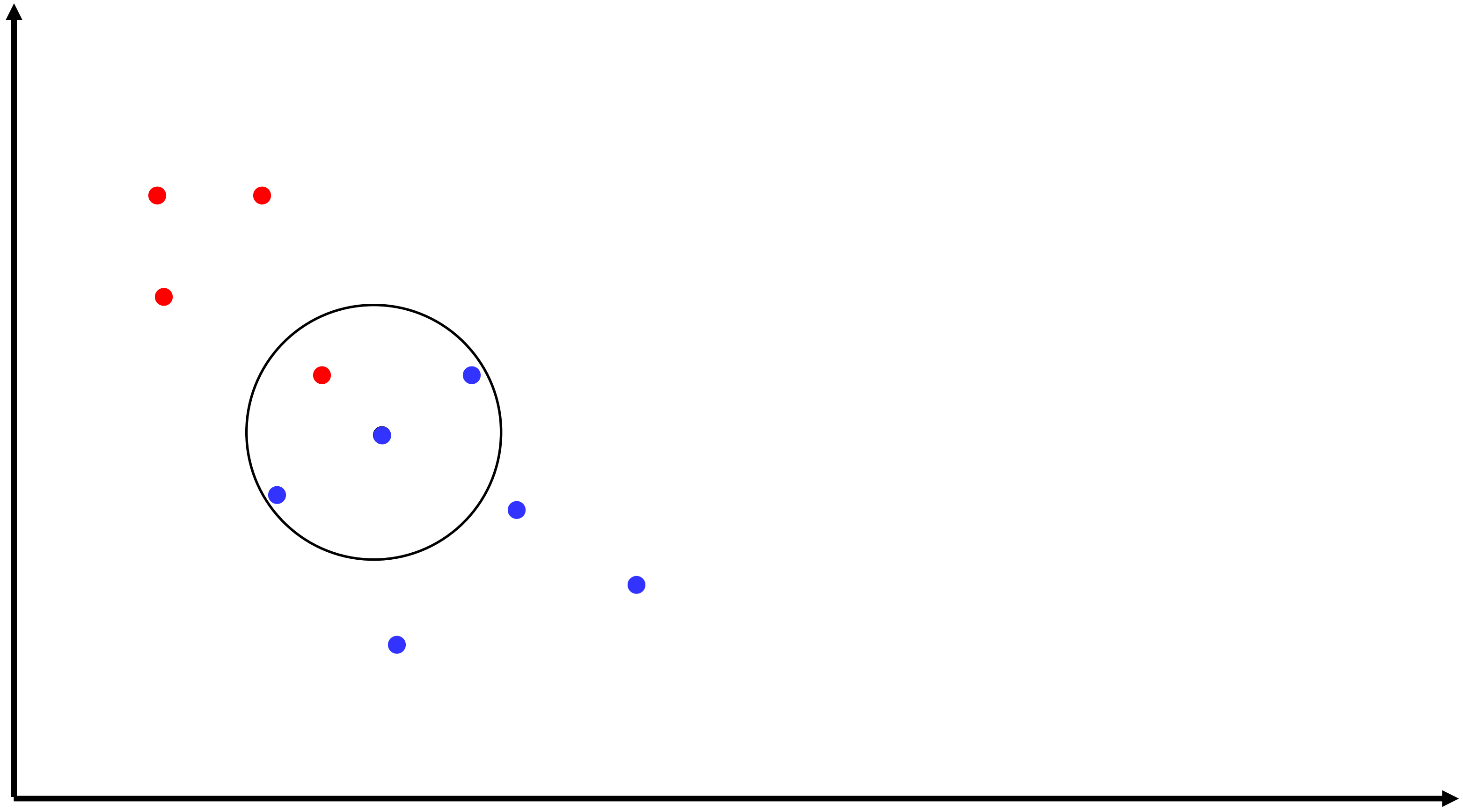
K Nearest-Neighbor

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

Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- Simplest for continuous *m*-dimensional instance space is *Euclidian distance*.
- Simplest for *m*-dimensional binary instance space is *Hamming distance* (number of feature values that differ).
- For text, cosine similarity of TF-IDF weighted vectors is typically most effective.

3 Nearest Neighbor Illustration
(Euclidian Distance)



K Nearest Neighbor for Text

Training:

For each each training example $<x, c(x)> \in D$

Compute the corresponding TF-IDF vector, $\mathbf{d}_x$, for document x

Test instance *y*:

Compute TF-IDF vector **d** for document y

For each $<x, c(x)> \in D$

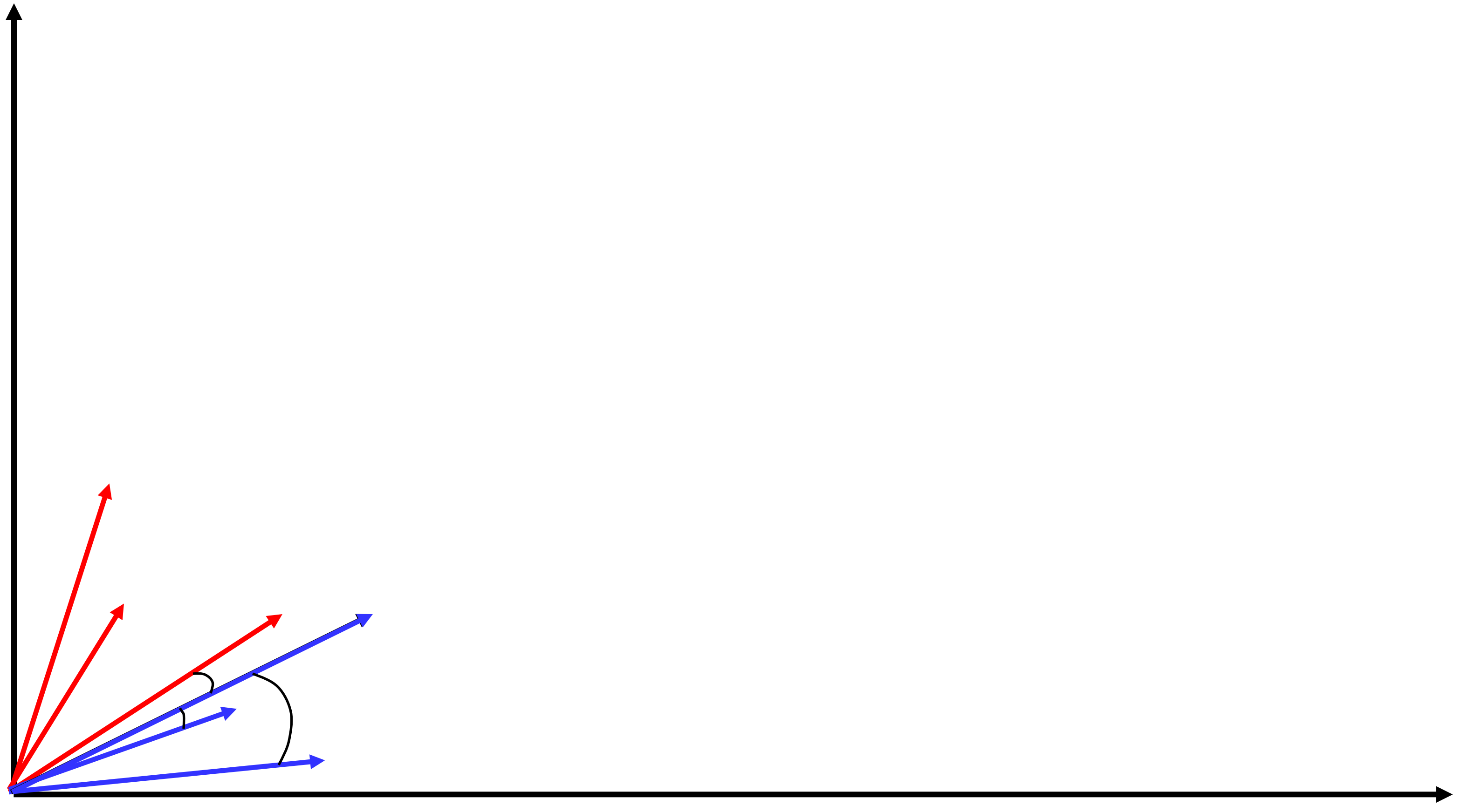
Let $s_x = \text{cosSim}(\mathbf{d}, \mathbf{d}_x)$

Sort examples, x, in D by decreasing value of s_x

Let N be the first k examples in D. (*get most similar neighbors*)

Return the majority class of examples in N

Illustration of 3 Nearest Neighbor for Text



Bayesian Classification

Bayesian Methods

- Learning and classification methods based on probability theory.
  - Bayes theorem plays a critical role in probabilistic learning and classification.
- Uses *prior* probability of each category
  - Based on training data
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

Basic Probability Theory

- All probabilities between 0 and 1

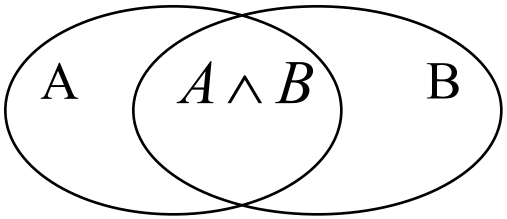
$$0 \leq P(A) \leq 1$$

- True proposition has probability 1, false has probability 0.

$$P(\text{true}) = 1 \qquad P(\text{false}) = 0.$$

- The probability of disjunction is:

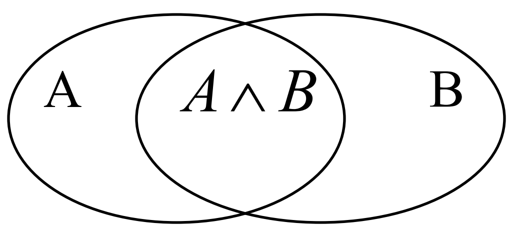
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



Conditional Probability

- P(A | B) is the probability of A given B
- Assumes that B is all and only information known.
- Defined by:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$



Independence

- A and B are *independent* iff:

$$P(A \mid B) = P(A)$$

$$P(B \mid A) = P(B)$$

These two constraints are logically equivalent

- Therefore, if A and B are independent:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} = P(A)$$

$$P(A \wedge B) = P(A)P(B)$$

Joint Distribution

- Joint probability distribution for $X_1,\ldots,X_n$ gives the probability of every combination of values: $P(X_1,\ldots,X_n)$
  - All values must sum to 1.

Category=positive

Color\shape	circle	square
red	0.20	0.02
blue	0.02	0.01

negative

	circle	square
red	0.05	0.30
blue	0.20	0.20

- Probability for assignments of values to some subset of variables can be calculated by summing the appropriate subset

$$P(red \wedge circle) = 0.20 + 0.05 = 0.25$$

$$P(red) = 0.20 + 0.02 + 0.05 + 0.3 = 0.57$$

- Conditional probabilities can also be calculated.

$$P(positive \mid red \wedge circle) = \frac{P(positive \wedge red \wedge circle)}{P(red \wedge circle)} = \frac{0.20}{0.25} = 0.80$$

Computing probability from a training dataset

Ex	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negitive
4	large	blue	circle	negitive

Test Instance *X*:
<medium, red, circle>

Probability	Y=positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.5	0.5
P(medium | *Y*)	0.0	0.0
P(large | *Y*)	0.5	0.5
P(red | *Y*)	1.0	0.5
P(blue | *Y*)	0.0	0.5
P(green | *Y*)	0.0	0.0
P(square | *Y*)	0.0	0.0
P(triangle | *Y*)	0.0	0.5
P(circle | *Y*)	1.0	0.5

Bayes Theorem

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Simple proof from definition of conditional probability:

$$P(H \mid E) = \frac{P(H \wedge E)}{P(E)} \quad \text{(Def. cond. prob.)}$$

$$P(E \mid H) = \frac{P(H \wedge E)}{P(H)} \quad \text{(Def. cond. prob.)}$$

$$P(H \wedge E) = P(E \mid H)P(H)$$

Thus: $$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Bayesian Categorization

- Determine category of instance x_k by determining for each y_i

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)}$$

- P(X=x_k) estimation is not needed in the algorithm to choose a classification decision via comparison.

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{\cancel{P(X = x_k)}}$$

- If really needed:

$$\sum_{i=1}^{m} P(Y = y_i \mid X = x_k) = \sum_{i=1}^{m} \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)} = 1$$

$$P(X = x_k) = \sum_{i=1}^{m} P(Y = y_i)P(X = x_k \mid Y = y_i)$$

Bayesian Categorization (cont.)

$$P(Y=y_i \mid X=x_k) = \frac{P(Y=y_i)P(X=x_k \mid Y=y_i)}{\cancel{P(X=x_k)}}$$

- Need to know:
  - Priors: $P(Y=y_i)$
  - Conditionals: $P(X=x_k \mid Y=y_i)$
- $P(Y=y_i)$ are easily estimated from training data.
  - If n_i of the examples in training data D are in y_i then $P(Y=y_i) = n_i / |D|$
- Too many possible instances (e.g. 2^n for binary features) to estimate all $P(X=x_k \mid Y=y_i)$ in advance.

Naïve Bayesian Categorization

- If we assume features of an instance are independent **given the category** (***conditionally independent***).

$$P(X \mid Y) = P(X_1, X_2, \cdots X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y)$$

- Therefore, we then only need to know P(X_i| Y) for each possible pair of a feature-value and a category.
  - n_i of the examples in training data D are in y_i
  - $n_{ij}$of the examples in D *with category* y_i
  - P(x_{ij} |Y=y_i) = n_{ij}/ n_i

Underflow Prevention:

Multiplying lots of probabilities may result in floating-point underflow. Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities.

Computing probability from a training dataset

Ex	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negitive
4	large	blue	circle	negitive

Test Instance *X*:
<medium, red, circle>

Probability	Y=positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.5	0.5
P(medium | *Y*)	0.0	0.0
P(large | *Y*)	0.5	0.5
P(red | *Y*)	1.0	0.5
P(blue | *Y*)	0.0	0.5
P(green | *Y*)	0.0	0.0
P(square | *Y*)	0.0	0.0
P(triangle | *Y*)	0.0	0.5
P(circle | *Y*)	1.0	0.5

Naïve Bayes Example

Probability	Y=positive	Y=negative
P(Y)	0.5	0.5
P(small | Y)	0.4	0.4
P(medium | Y)	0.1	0.2
P(large | Y)	0.5	0.4
P(red | Y)	0.9	0.3
P(blue | Y)	0.05	0.3
P(green | Y)	0.05	0.4
P(square | Y)	0.05	0.4
P(triangle | Y)	0.05	0.3
P(circle | Y)	0.9	0.3

Test Instance:
<medium ,red, circle>

Naïve Bayes Example

Probability	Y=positive	Y=negative
$P(Y)$	0.5	0.5
$P(\text{medium} \mid Y)$	0.1	0.2
$P(\text{red} \mid Y)$	0.9	0.3
$P(\text{circle} \mid Y)$	0.9	0.3

Test Instance:
<medium ,red, circle>

$P(\text{positive} \mid X) = P(\text{Positive})*P(X/\text{Positive})/P(X)$

$= P(\text{positive})*P(\text{medium} \mid \text{positive})*P(\text{red} \mid \text{positive})*P(\text{circle} \mid \text{positive}) / P(X)$

$0.5 \quad * \quad 0.1 \quad * \quad 0.9 \quad * \quad 0.9$

$= 0.0405 / P(X) = 0.0405 / 0.0495 = 0.8181$

$P(\text{negative} \mid X) = P(\text{negative})*P(\text{medium} \mid \text{negative})*P(\text{red} \mid \text{negative})*P(\text{circle} \mid \text{negative}) / P(X)$

$0.5 \quad * \quad 0.2 \quad * \quad 0.3 \quad * \quad 0.3$

$= 0.009 / P(X) \quad = 0.009 / 0.0495 = 0.1818$

$P(\text{positive} \mid X) + P(\text{negative} \mid X) = 0.0405 / P(X) + 0.009 / P(X) = 1$

$P(X) = (0.0405 + 0.009) = 0.0495$

Error prone prediction with small training data

Ex	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negitive
4	large	blue	circle	negitive

Probability	Y=positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.5	0.5
P(medium | *Y*)	0.0	0.0
P(large | *Y*)	0.5	0.5
P(red | *Y*)	1.0	0.5
P(blue | *Y*)	0.0	0.5
P(green | *Y*)	0.0	0.0
P(square | *Y*)	0.0	0.0
P(triangle | *Y*)	0.0	0.5
P(circle | *Y*)	1.0	0.5

Test Instance *X*:
<medium, red, circle>

P(positive | *X*) = 0.5 * 0.0 * 1.0 * 1.0 = 0

P(negative | *X*) = 0.5 * 0.0 * 0.5 * 0.5 = 0

Smoothing

- To account for estimation from small samples, probability estimates are adjusted or *smoothed*.
- Laplace smoothing using an *m*-estimate assumes that each feature is given a prior probability, *p*, that is assumed to have been previously observed in a "virtual" sample of size *m*.

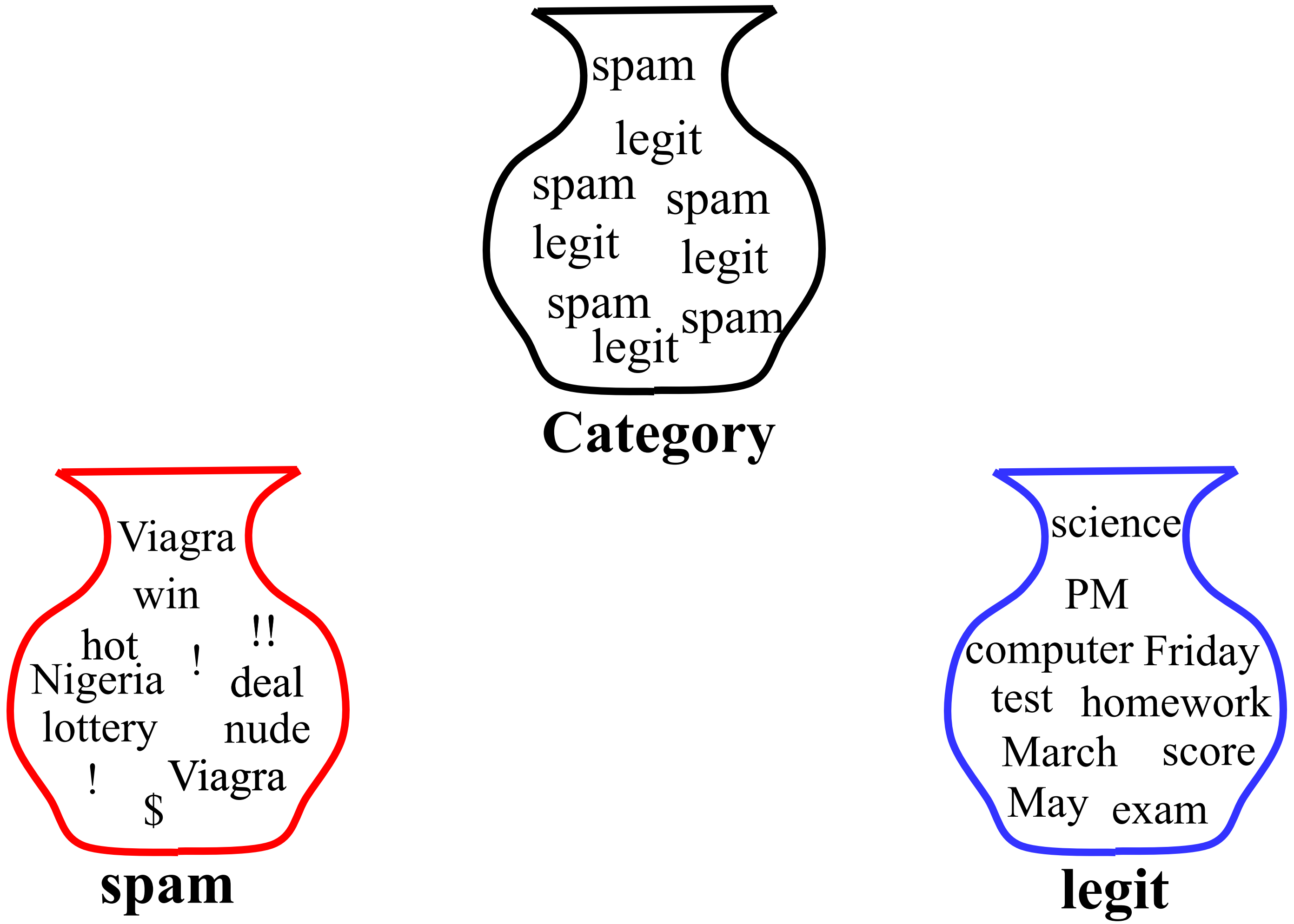
$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk} + mp}{n_k + m}$$

- For binary features, *p* is simply assumed to be 0.5.

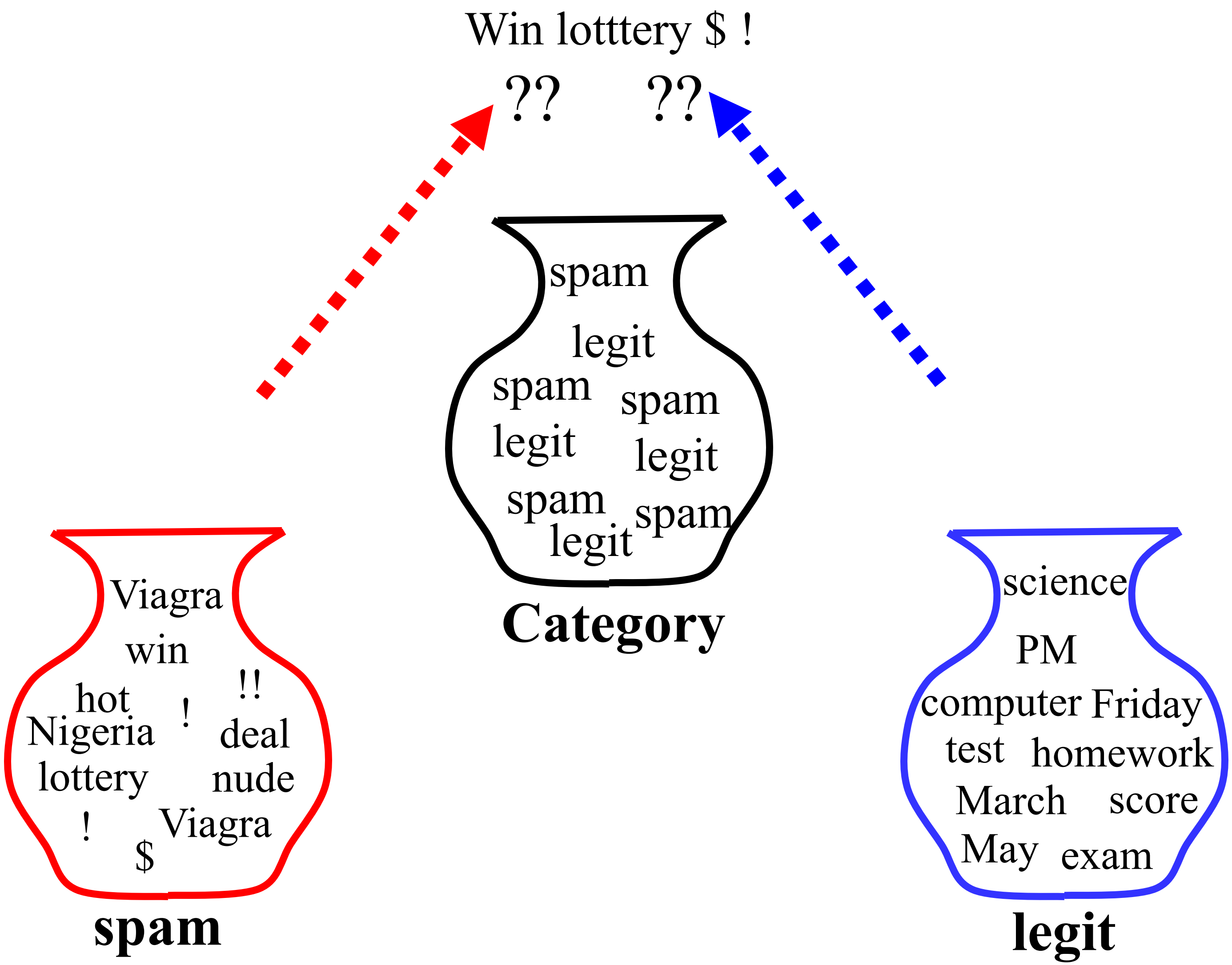
Laplace Smothing Example

- Assume training set contains 10 positive examples:
  - 4: small
  - 0: medium
  - 6: large
- Estimate parameters as follows (if m=1, p=1/3)
  - P(small | positive) = (4 + 1/3) / (10 + 1) = 0.394
  - P(medium | positive) = (0 + 1/3) / (10 + 1) = 0.03
  - P(large | positive) = (6 + 1/3) / (10 + 1) = 0.576
  - P(small or medium or large | positive) = 1.0

Bayes Training Example



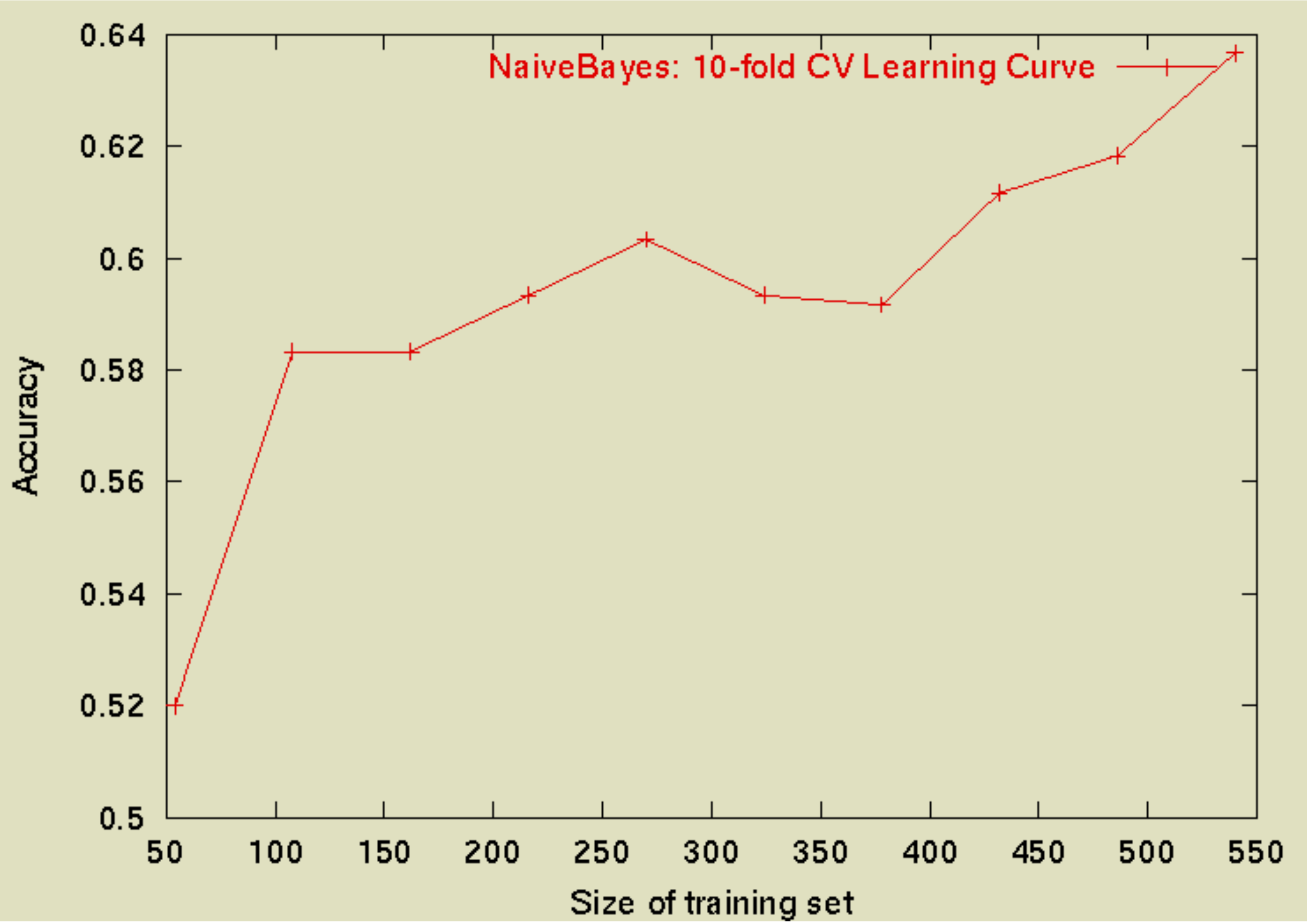
Naïve Bayes Classification



Evaluating Accuracy of Classification

- Evaluation must be done on test data that are independent of the training data
  - Classification accuracy: the number of test instances correctly classified divided by total number of test instances
  - Average results over multiple training and test sets (splits of the overall data) for the best results.
- Not enough labeled data? N-fold cross-validation
- Partition data into N equal-sized disjoint segments.
  - Run N trials, each time using a different segment of the data for testing, and training on the remaining N–1 segments.
  - This way, at least test-sets are independent.
  - Report average classification accuracy over the N trials.
  - Typically, $N = 10$.

Sample Learning Curve (Yahoo Science Data)

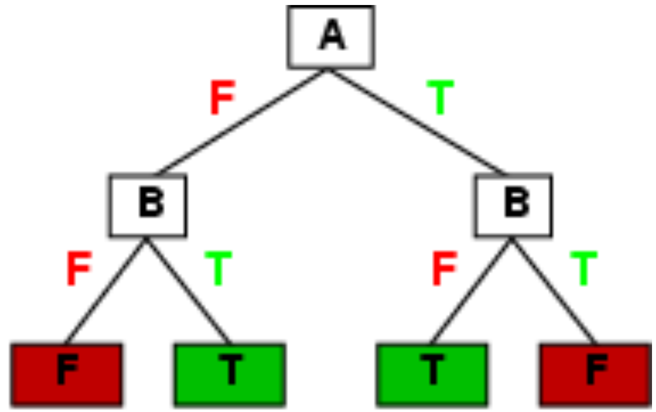


Classification with Decision Trees

Decision Trees

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples

- Prefer to find more compact decision trees: we don't want to memorize the data, we want to find structure in the data!

Decision Trees: Application Example

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Training data: Restaurant example

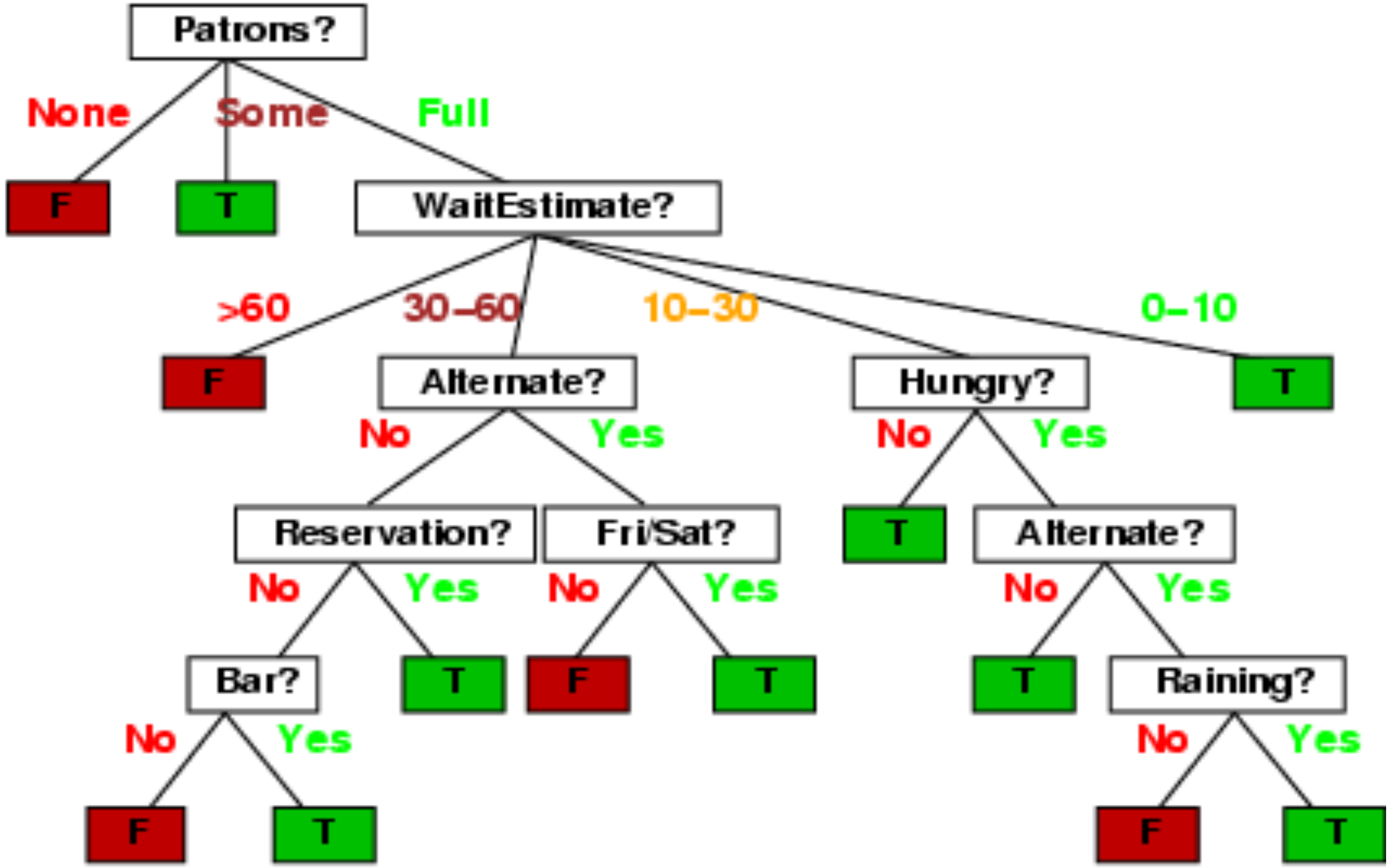
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

A decision tree to decide whether to wait

- imagine someone talking a sequence of decisions.

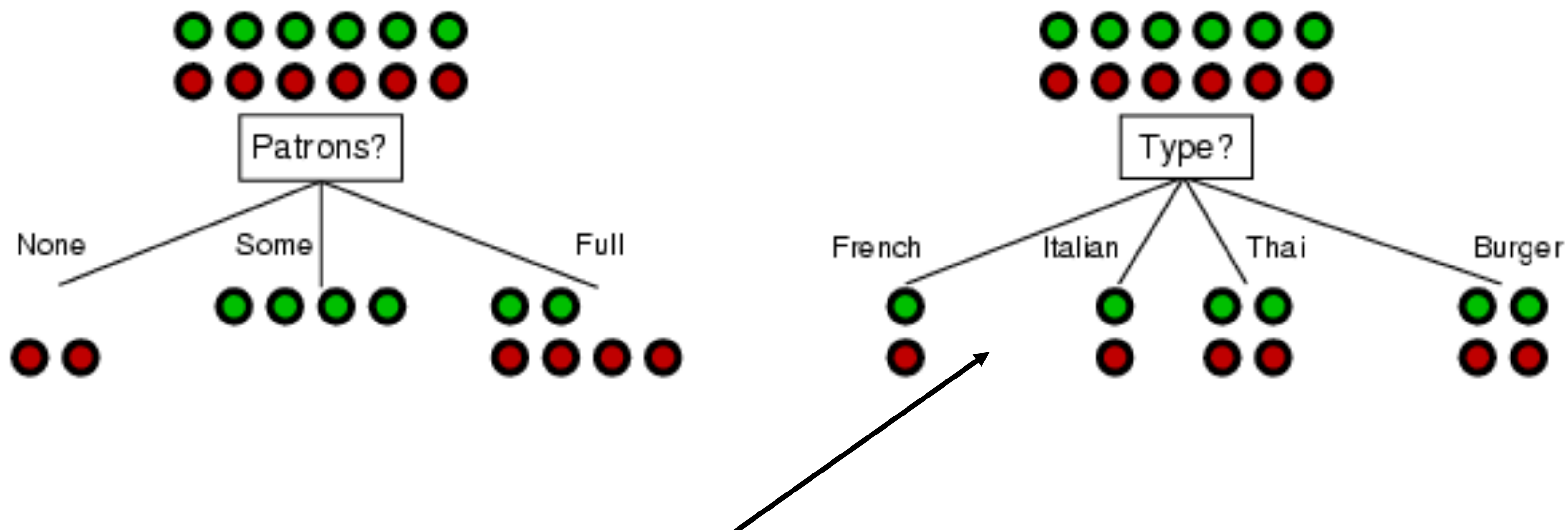


Decision tree learning

- If there are so many possible trees, can we actually search this space? (solution: greedy search).
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree.

Choosing an attribute for making a decision

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



To wait or not to wait is still at 50%.

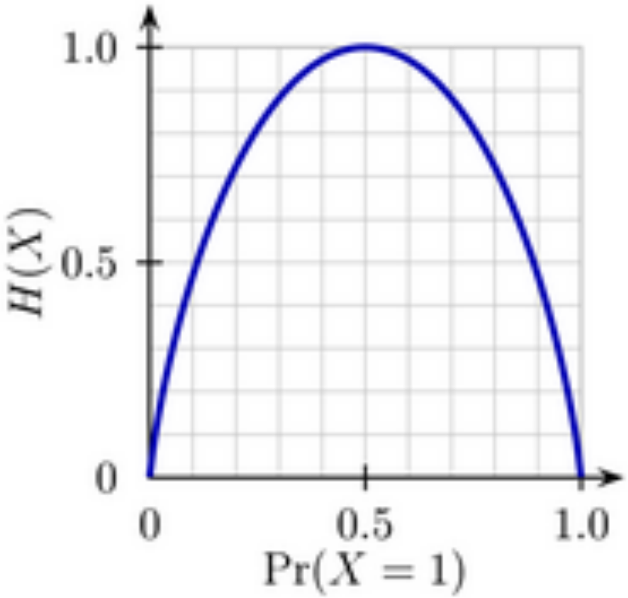
Information theory background: Entropy

- Entropy measures uncertainty

  $-p \log(p) - (1-p) \log(1-p)$

Consider tossing a biased coin.
If you toss the coin VERY often,
the frequency of heads is, say, p,
and hence the frequency of tails is
1-p.

Uncertainty (entropy) is zero if p=0 or 1
and maximal if we have p=0.5.



Using information theory for binary decisions

- Imagine we have p examples which are true (positive) and n examples which are false (negative).

- Our best estimate of true or false is given by:

$$P(true) \approx p / p + n$$
$$p(false) \approx n / p + n$$

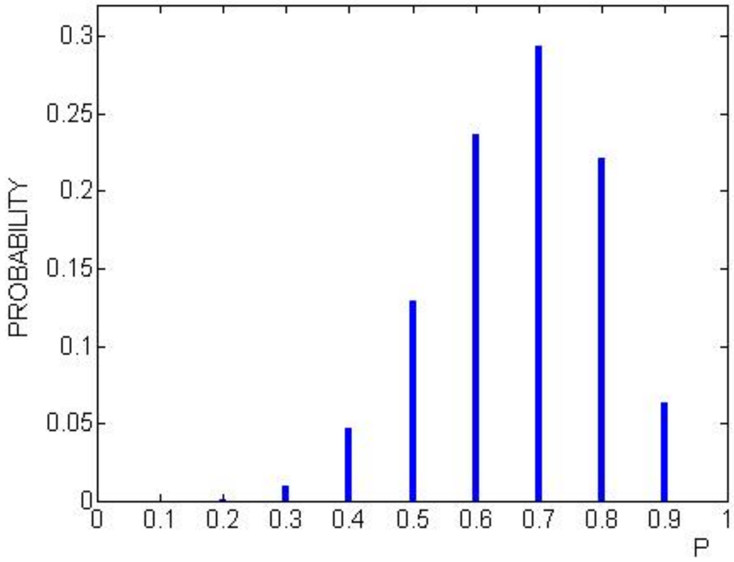
- Hence the entropy is given by:

$$Entropy(\frac{p}{p+n}, \frac{n}{p+n}) \approx -\frac{p}{p+n} \log \frac{p}{p+n} - \frac{n}{p+n} \log \frac{n}{p+n}$$

Using information theory for more than 2 states

- If there are more than two states s=1,2,..n we have (e.g. a die):

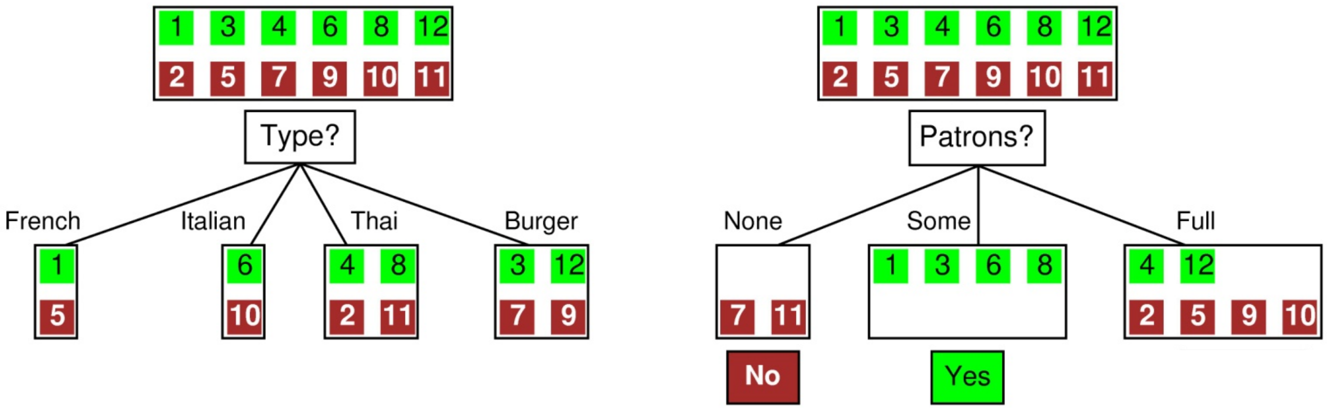
$$
\begin{aligned}
Entropy(p) = & -p(s=1)\log[p(s=1)] \\
& - p(s=2)\log[p(s=2)] \\
& \ldots \\
& - p(s=n)\log[p(s=n)]
\end{aligned}
$$



$$\sum_{s=1}^{n} p(s) = 1$$

ID3 Algorithm: Using Information Theory to Choose an Attribute

- How much information do we gain if we disclose the value of some attribute?
- ID3 algorithm by Ross Quinlan uses information gained measured by maximum entropy reduction:
  - IG(A) = uncertainty before – uncertainty after
  - Choose an attribute with the maximum IA



Before: Entropy = - ½ log(1/2) – ½ log(1/2)=log(2) = 1 bit:
There is “1 bit of information to be discovered”.

After: for Type: If we go into branch “French” we have 1 bit, similarly for the others.

French: 1bit
Italian: 1 bit
Thai: 1 bit
Burger: 1bit

On average: 1 bit and gained nothing!

After: for Patrons: In branch “None” and “Some” entropy = 0!,
In “Full” entropy = -1/3log(1/3)-2/3log(2/3)=0.92

So Patrons gains more information!

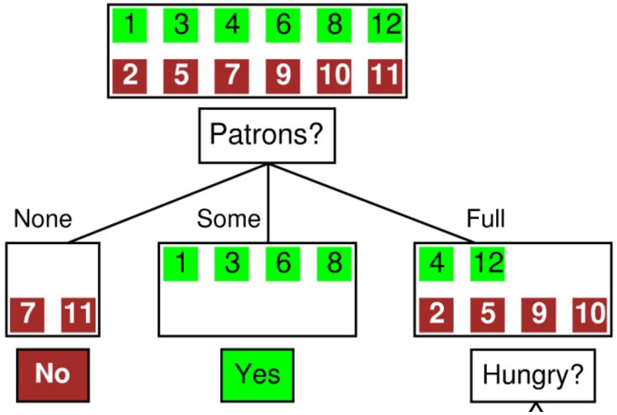
Information Gain: How to combine branches

•1/6 of the time we enter "None", so we weight"None" with 1/6.
Similarly: "Some" has weight: 1/3 and "Full" has weight ½.

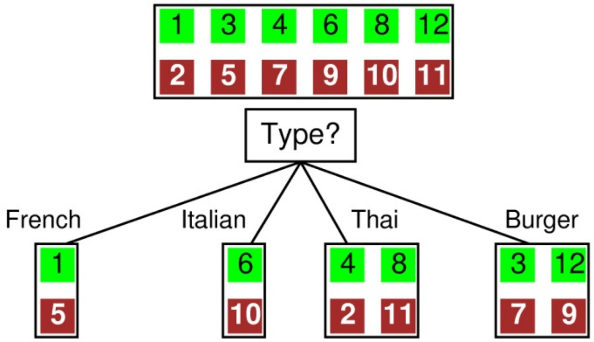
$$Entropy(A) = \sum_{i=1}^{n} \frac{p_i + n_i}{p + n} Entropy\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

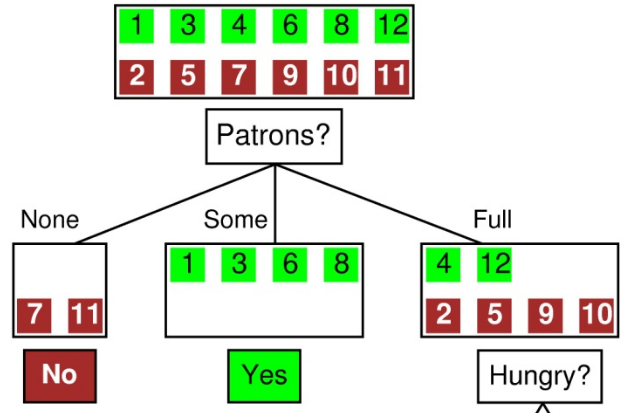
entropy for each branch.

weight for each branch



Choose an attribute: Restaurant Example





For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

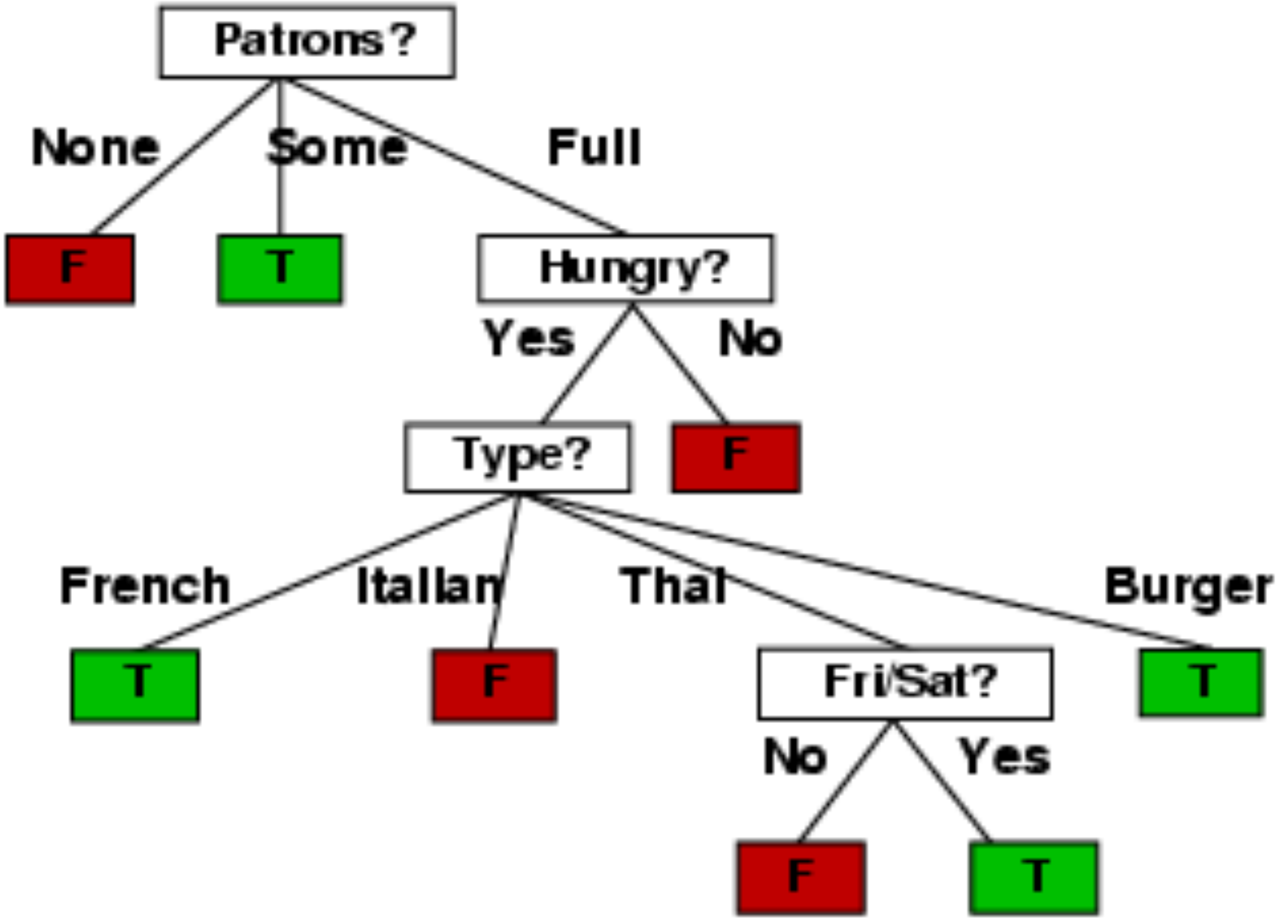
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Example: Decision tree learned

- Decision tree learned from the 12 examples:

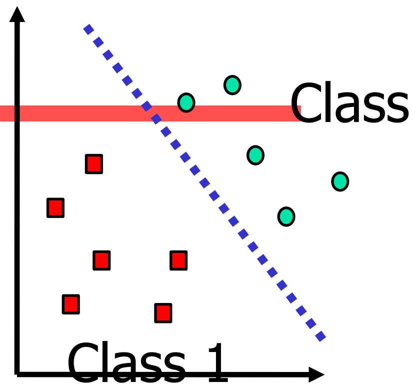


Issues

- When there are no attributes left:
  - Stop growing and use majority vote.
- Avoid over-fitting data
  - Stop growing a tree earlier
  - Grow first, and prune later.
- Deal with continuous-valued attributes
  - Dynamically select thresholds/intervals.
- Handle missing attribute values
  - Make up with common values
- Control tree size
  - pruning

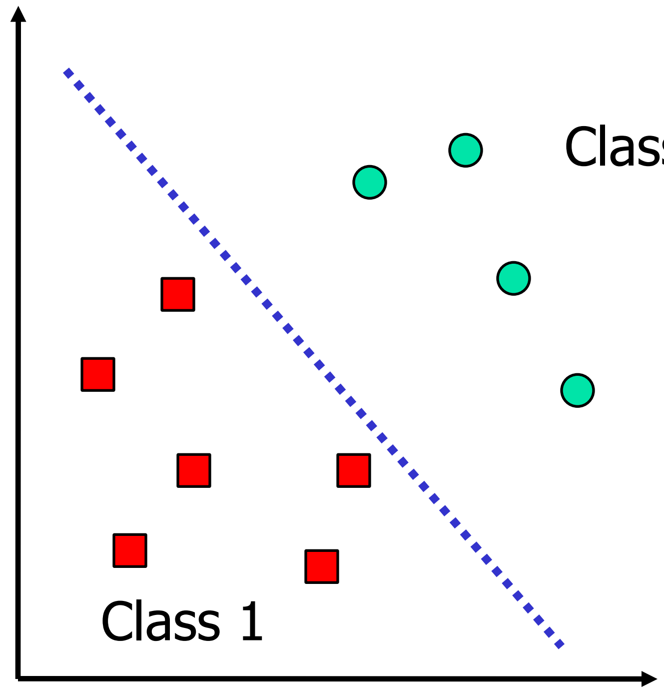
Classification with SVM

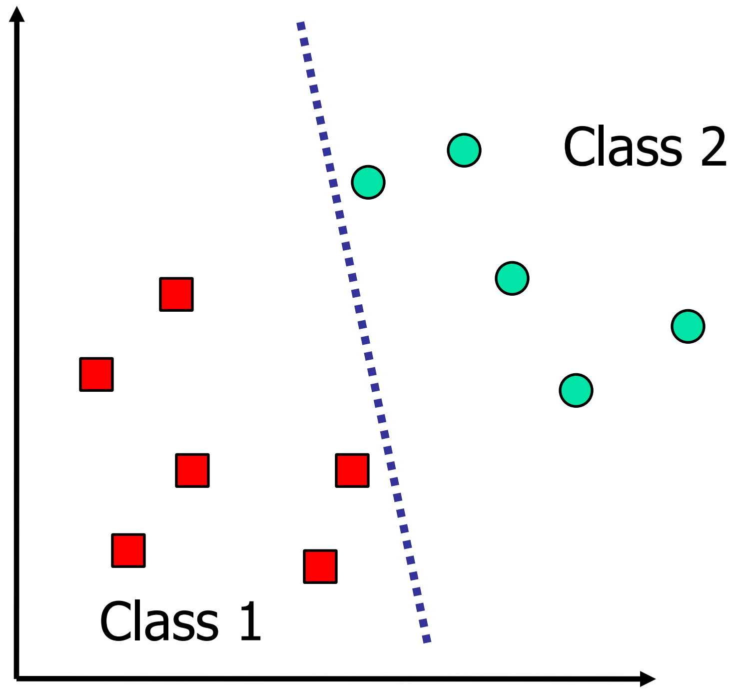
Two Class Problem: Linear Separable Case with a Hyperplane



Many decision boundaries can separate classes using a hyperplane.
Which one should we choose?

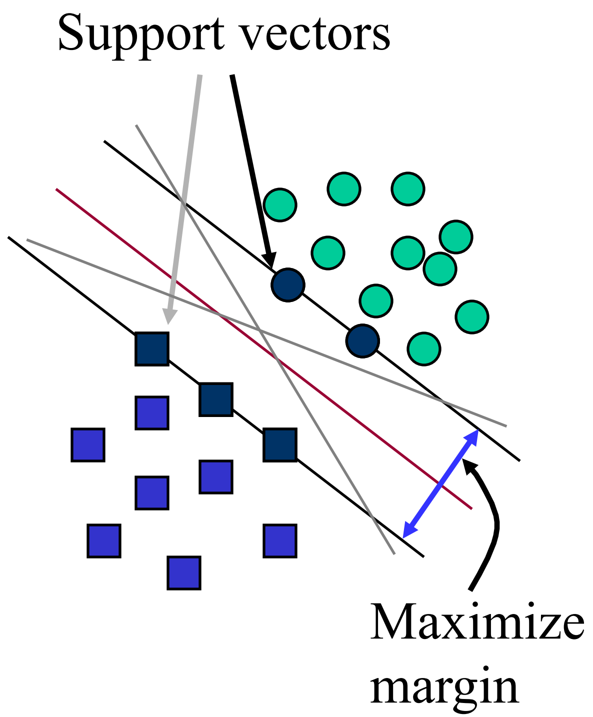
Example of Bad Decision Boundaries





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*.
- *Quadratic programming* problem



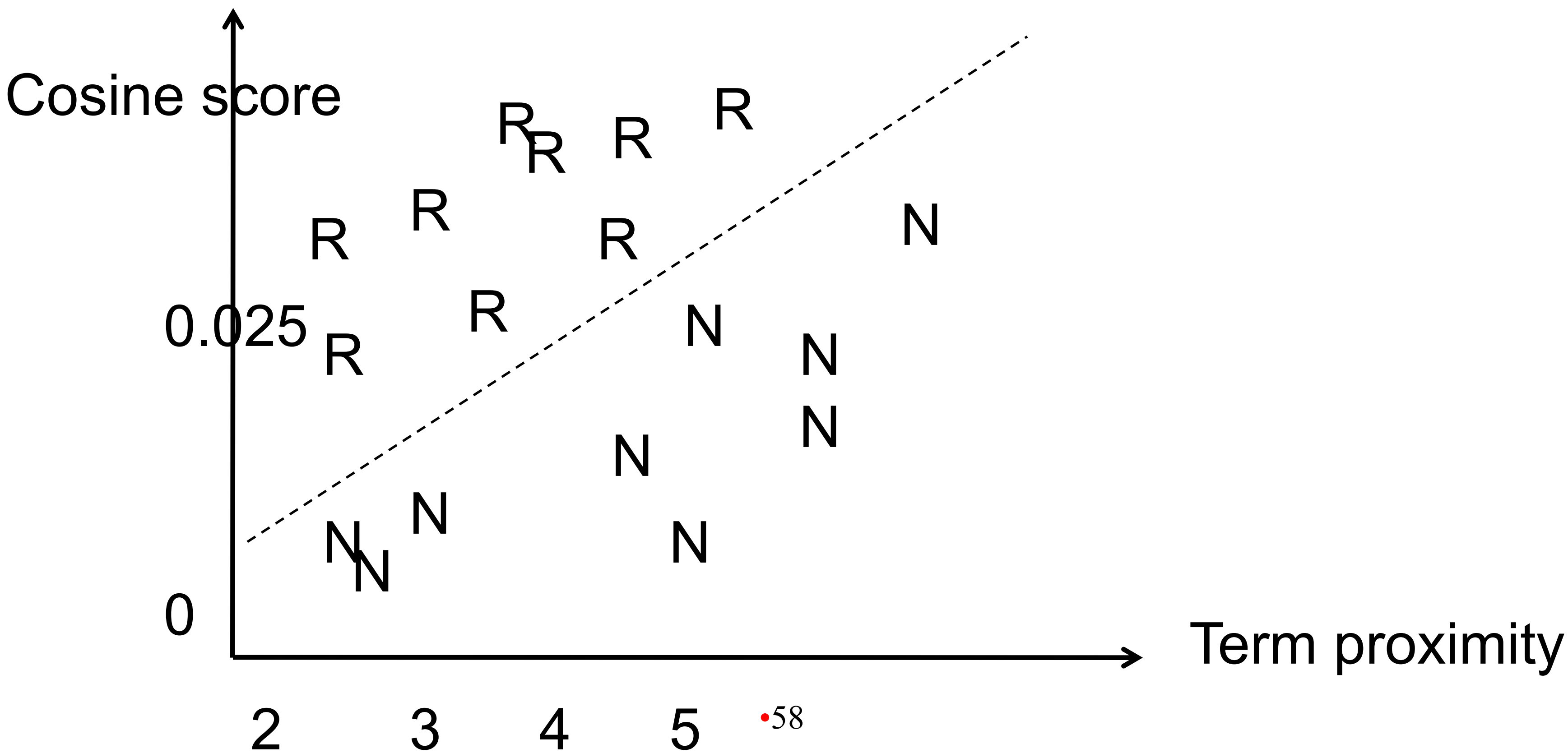
Training examples for document ranking

Two ranking signals are used (Cosine text similarity score, proximity of term appearance window)

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*
...	...	...	...		

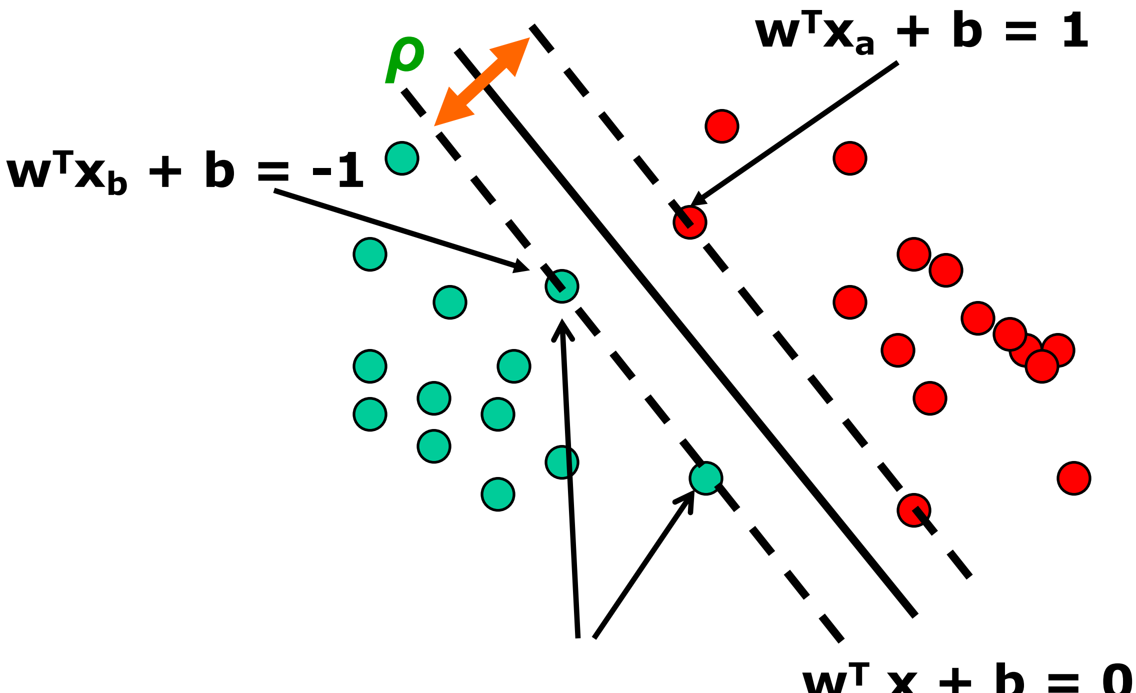
Proposed scoring function for ranking

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



Formalization

- w: weight coefficients
- x_i: data point i
- y_i: class result of data point i (+1 or -1)
- Classifier is: $f(x_i) = \text{sign}(w^T x_i + b)$



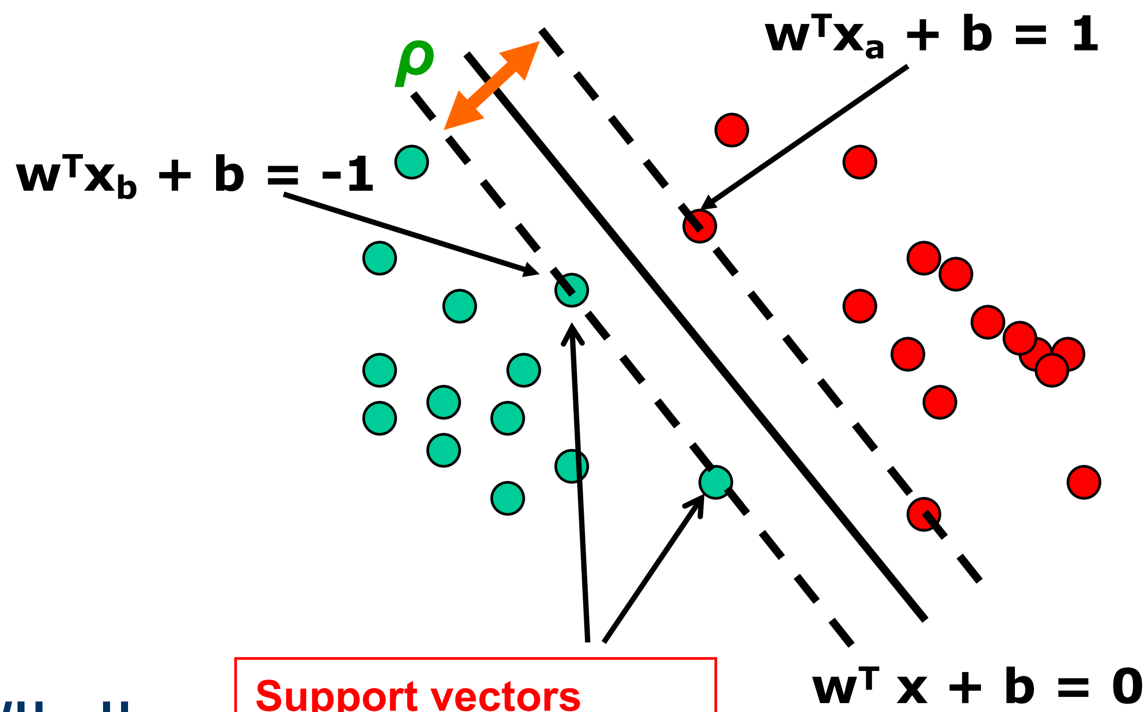
Linear Support Vector Machine (SVM)

- Hyperplane

$w^T x + b = 0$

$w^T x + b = 1$

$w^T x + b = -1$



Support vectors
datapoints that the margin
pushes up against

- $\rho = ||x_a - x_b||_2 = 2/||w||_2$
- $||w||^2 = w^T w$

Linear SVM Mathematically

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

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

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

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

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

- The margin of dataset is:

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

The Optimization Problem

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$
- A constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

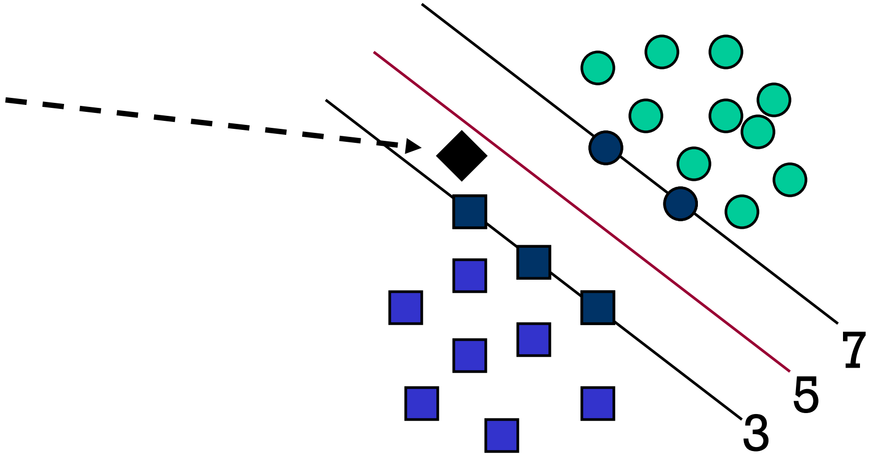
Classification with SVMs

- Given a new point (x_1,x_2), we can score its projection onto the hyperplane normal:
  - In 2 dims: score = $w_1x_1+w_2x_2+b$.
    - I.e., compute score: $wx + b = \Sigma\alpha_i y_i \mathbf{x_i}^{\mathbf{T}}\mathbf{x} + b$
  - Set confidence threshold t.

Score > t: yes

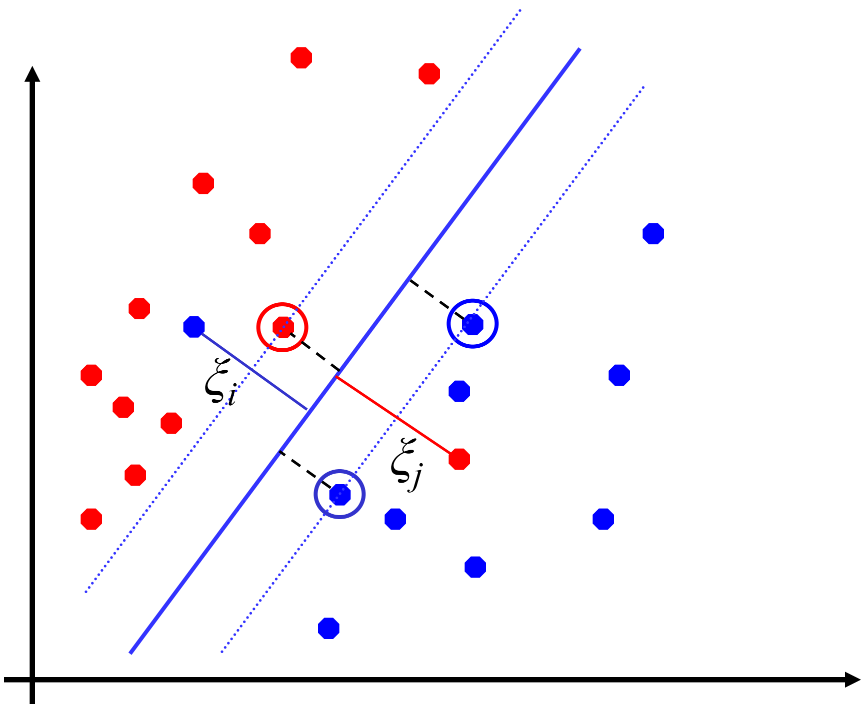
Score < -t: no

Else: don't know



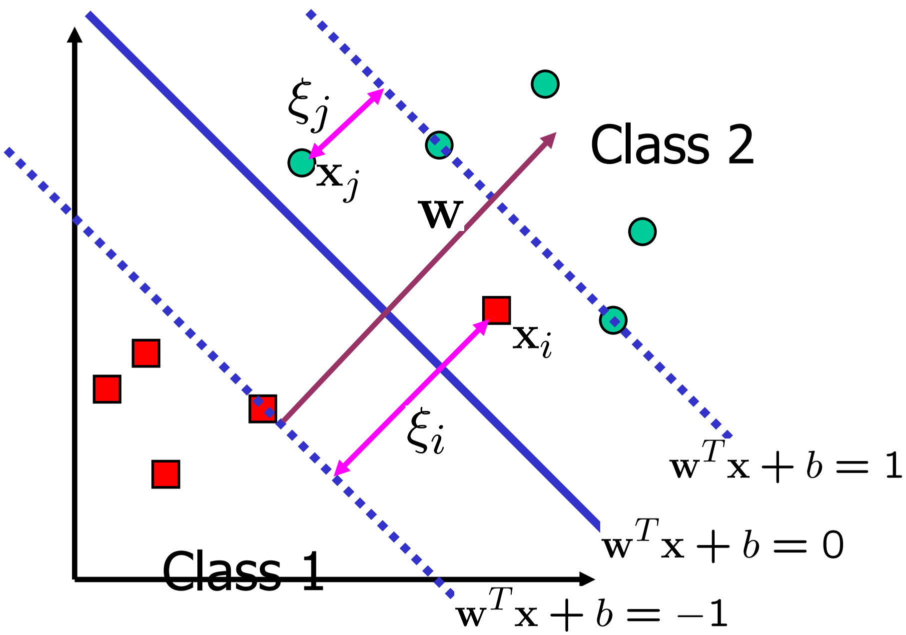
Soft Margin Classification

- If the training set 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

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}$+b
- ξ_i approximates the number of misclassified samples



New objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

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 $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^\mathbf{T}\mathbf{x_i} + \mathrm{b}) \geq 1$

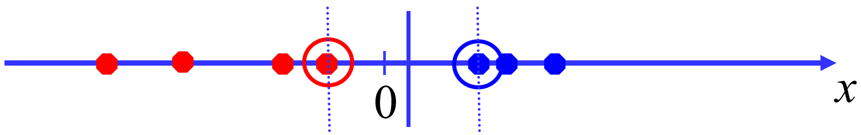
- The new formulation incorporating slack variables:

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 $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

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

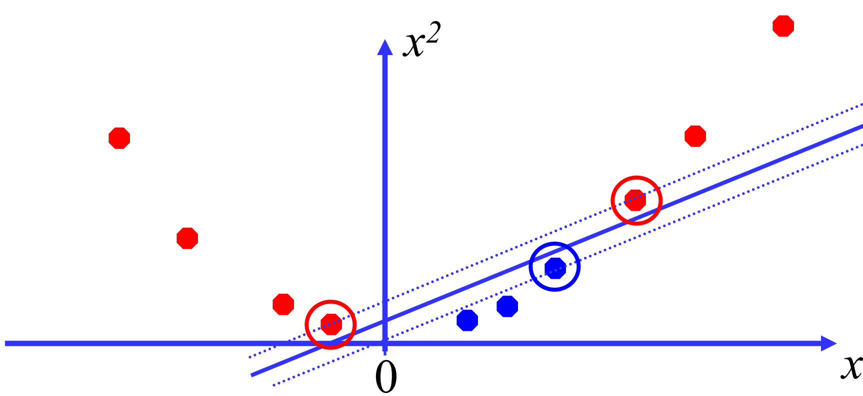
Non-linear SVMs

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



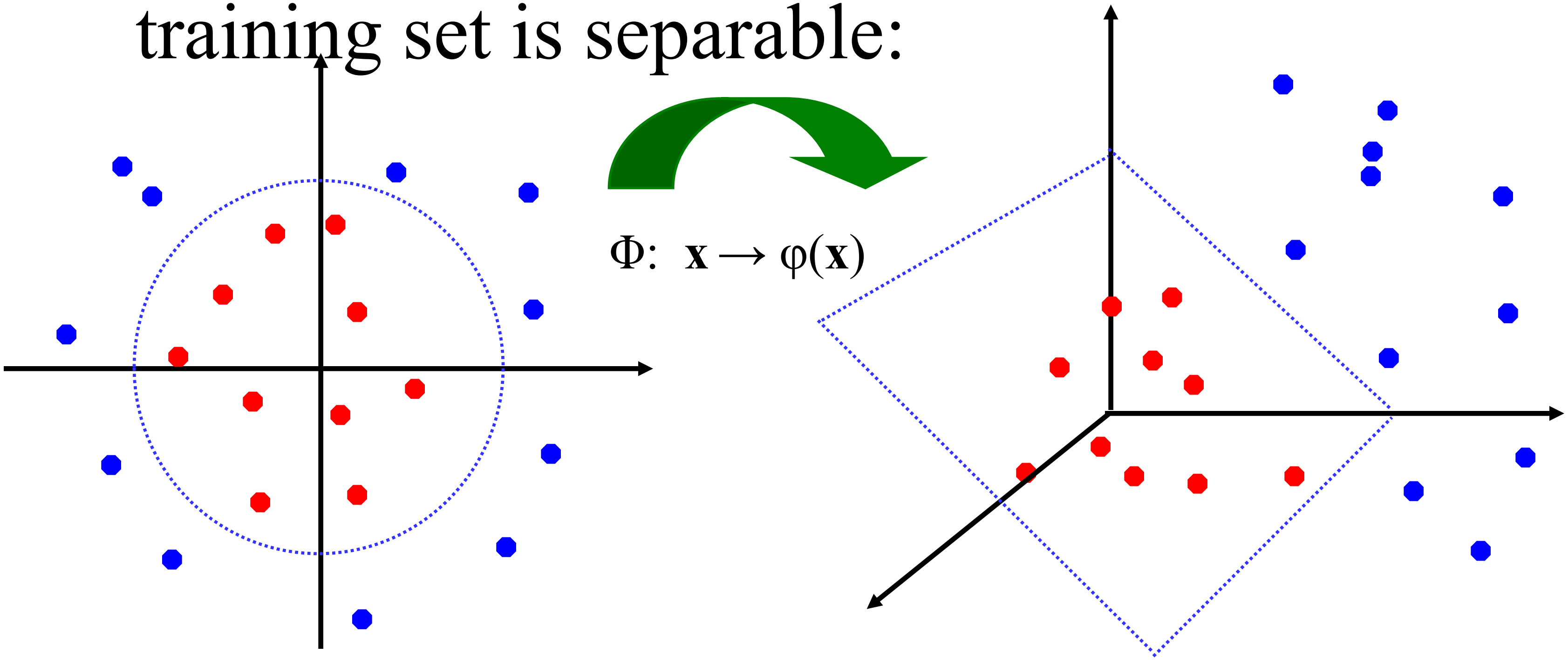
- But what are we going to do if the dataset is just too hard?

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



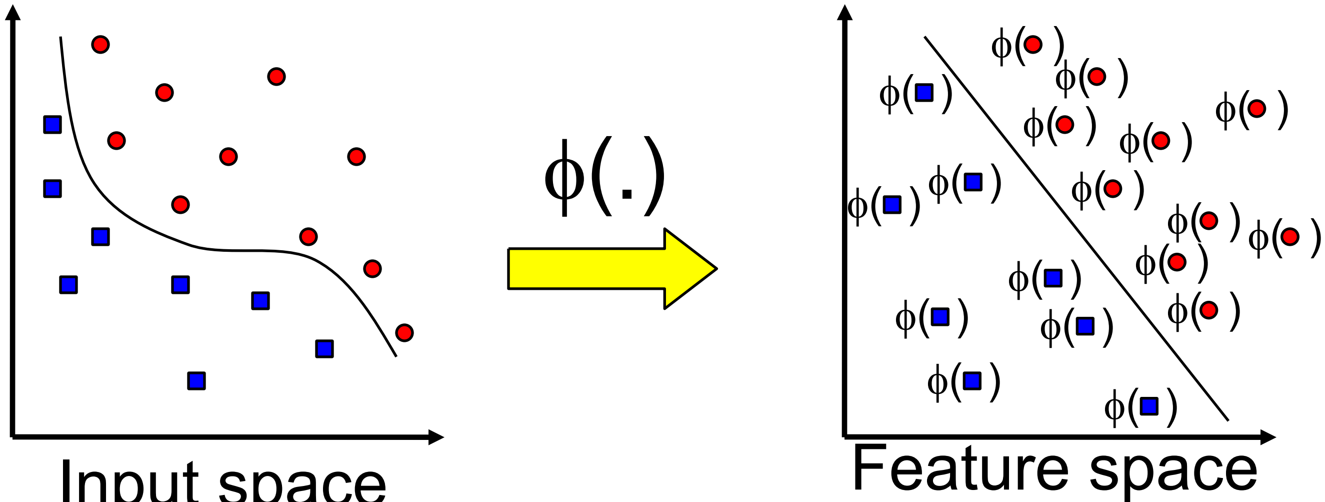
Non-linear SVMs: Feature spaces

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



Transformation to Feature Space

- "Kernel tricks"
  – Make non-separable problem separable.
  – Map data into better representational space



Example Transformation

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\begin{aligned} \langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

- Define the kernel function *K* (**x**,**y**) as

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- SVM computation involves pair-wise vector product. The inner product φ(.)φ(.) can be computed by *K* without going through the map φ(.) explicitly!

Choosing a Kernel Function

- Active research on kernel function choices for different applications
- Examples:
  - Polynomial kernel with degree *d* $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$
  - Radial basis function (RBF) kernel

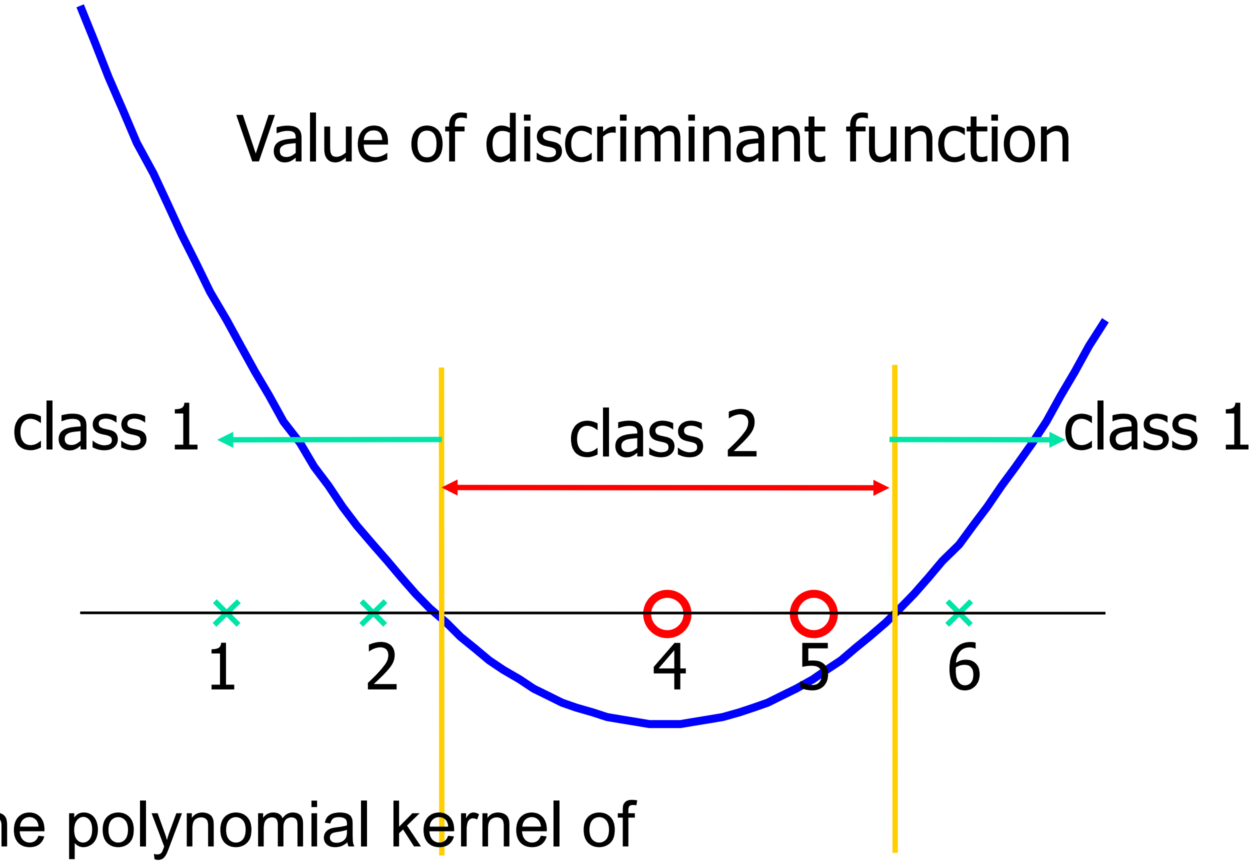
$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

or sometime $K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$

  - Closely related to radial basis function neural networks

- In practice, a low degree polynomial kernel or RBF kernel is a good initial try

Example: 5 1D data points



We use the polynomial kernel of degree 2

$K(x,y) = (xy+1)^2$

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Evaluation: Reuters News Data Set

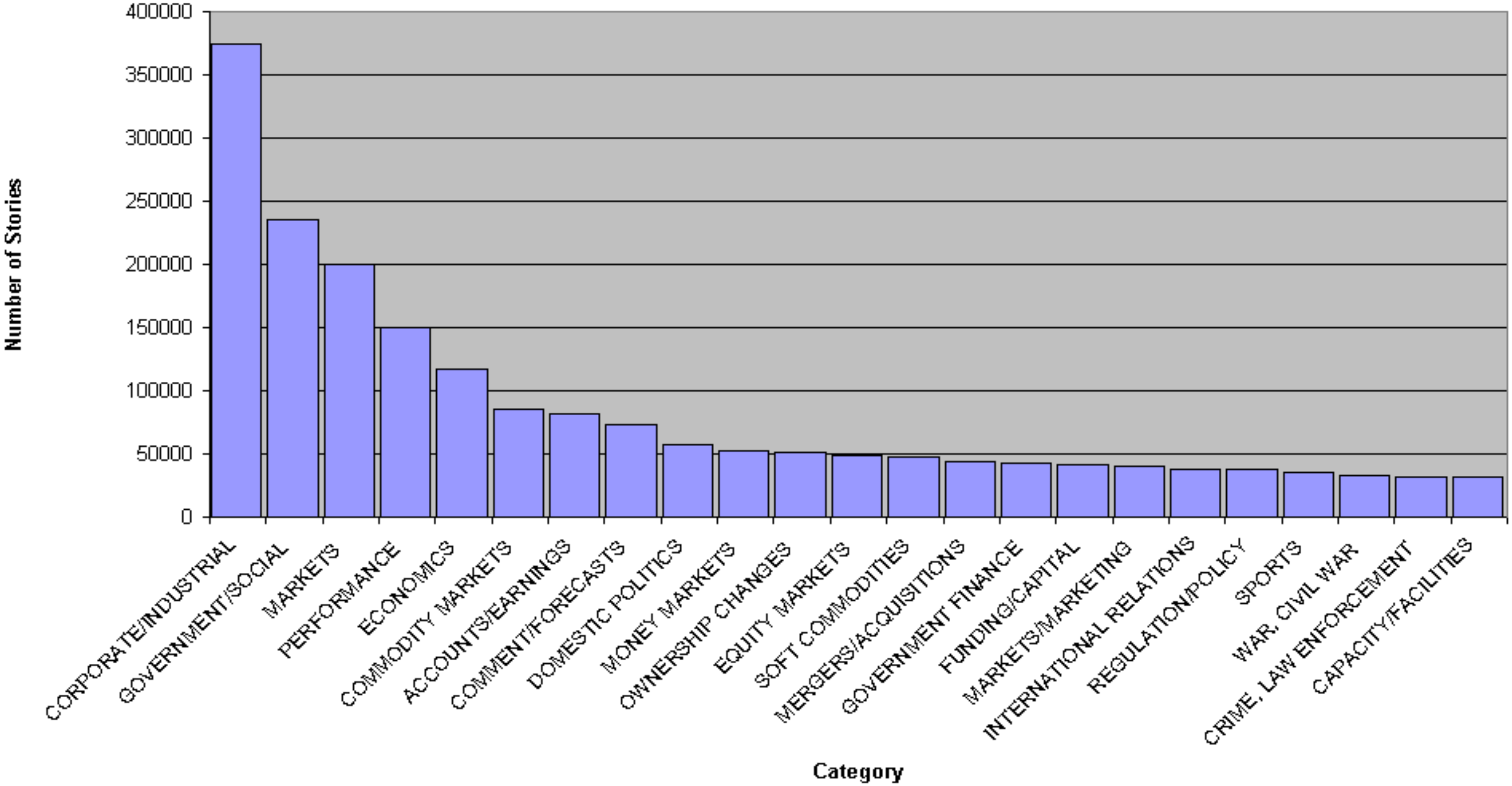
- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

New Reuters: RCV1: 810,000 docs

- Top topics in Reuters RCV1



Dumais et al. 1998: Reuters - Accuracy

	Rocchio	NBayes	Trees	LinearSVM	
earn	92.9%	95.9%	97.8%	98.2%	
acq	64.7%	87.8%	89.7%	92.8%	
money-fx	46.7%	56.6%	66.2%	74.0%	
grain	67.5%	78.8%	85.0%	92.4%	
crude	70.1%	79.5%	85.0%	88.3%	
trade	65.1%	63.9%	72.5%	73.5%	
interest	63.4%	64.9%	67.1%	76.3%	
ship	49.2%	85.4%	74.2%	78.0%	
wheat	68.9%	69.7%	92.5%	89.7%	
corn	48.2%	65.3%	91.8%	91.1%	
Avg Top 10	64.6%	81.5%	88.4%	91.4%	
Avg All Cat	61.7%	75.2%	na	86.4%	

Recall: % labeled in category among those stories that are really in category

Precision: % really in category among those stories labeled in category

Break Even: (Recall + Precision) / 2

Results for Kernels (Joachims 1998)

	Bayes	Rocchio	C4.5	k-NN	SVM (poly) degree $d =$ 1	2	3	4	5	SVM (rbf) width $\gamma =$ 0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average.
- Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

Micro- vs. Macro-Averaging: Example

Class 1	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Why this difference?

Micro.Av. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

The Real World

- How much training data do you have? None, very little, quite a lot, a huge amount and its growing
- Manually written rules
  - No training data, adequate editorial staff?
  - Never forget the hand-written rules solution!
    - If (wheat or grain) then categorize as grain
  - With careful crafting (human tuning on development data) performance is high:
    - 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
  - Amount of work required is huge
    - Estimate 2 days per class … plus maintenance

Which methods to use?

- A reasonable amount of data
  - Good with SVM, Trees
  - Be prepared with the "hybrid" solution.
- A huge amount of data
  - SVMs (train time) or kNN (test time) can be too expensive.
  - Naïve Bayes, logistic regression
  - Trees including boosting trees, random forests