NPFL103: Information Retrieval (9)
Vector Space Classification

Pavel Pecina

pecina@ufal.mff.cuni.cz

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University

Original slides are courtesy of Hinrich Schütze, University of Stuttgart.

Contents

Vector space classification

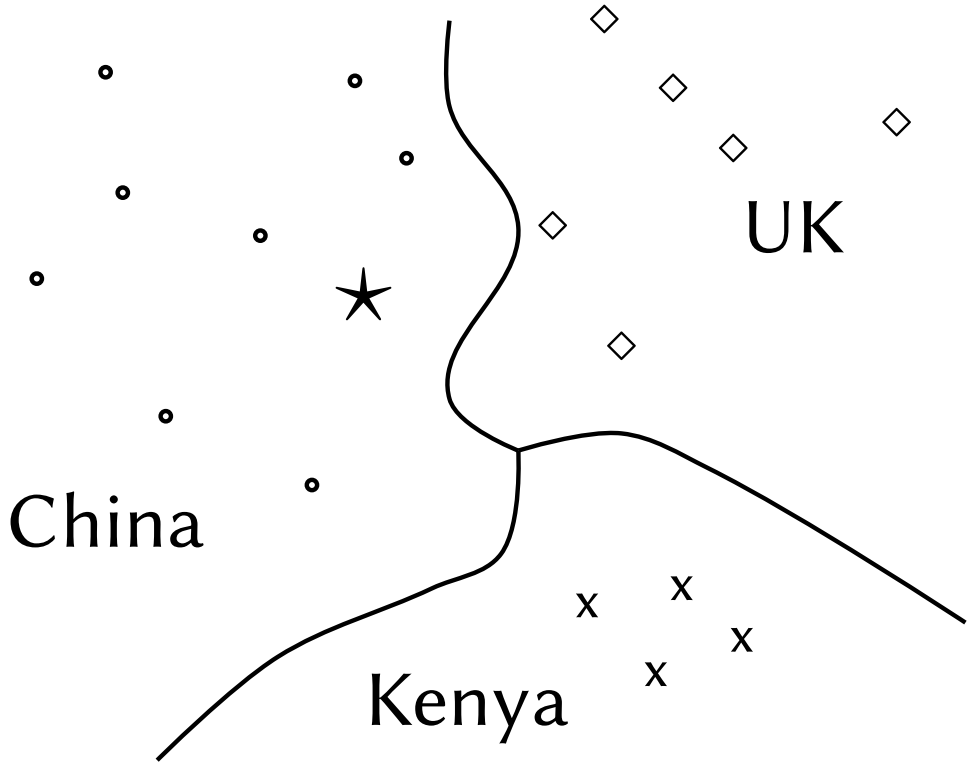
Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 100,000s of dimensions
- Normalize vectors (documents) to unit length
- How can we do classification in this space?

Vector space classification

- The training set is a set of documents, each labeled with its class.
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
- Premise 1: Documents in the same class form a contiguous region.
- Premise 2: Documents from different classes don't overlap.
- We define lines, surfaces, hypersurfaces to divide regions.

Classes in the vector space



Should the document ⋆ be assigned to *China*, *UK* or *Kenya*? Find separators between the classes Based on these separators: ⋆ should be assigned to *China* How do we find separators that do a good job at classifying new documents like ⋆?

k nearest neighbors

kNN classification

- kNN classification is another vector space classification method.
- It also is very simple and easy to implement.
- kNN is more accurate (in most cases) than Naive Bayes
- If you need to get a pretty accurate classifier up and running in a short time ...

  ...and you don't care about efficiency that much ...

  ...use kNN.

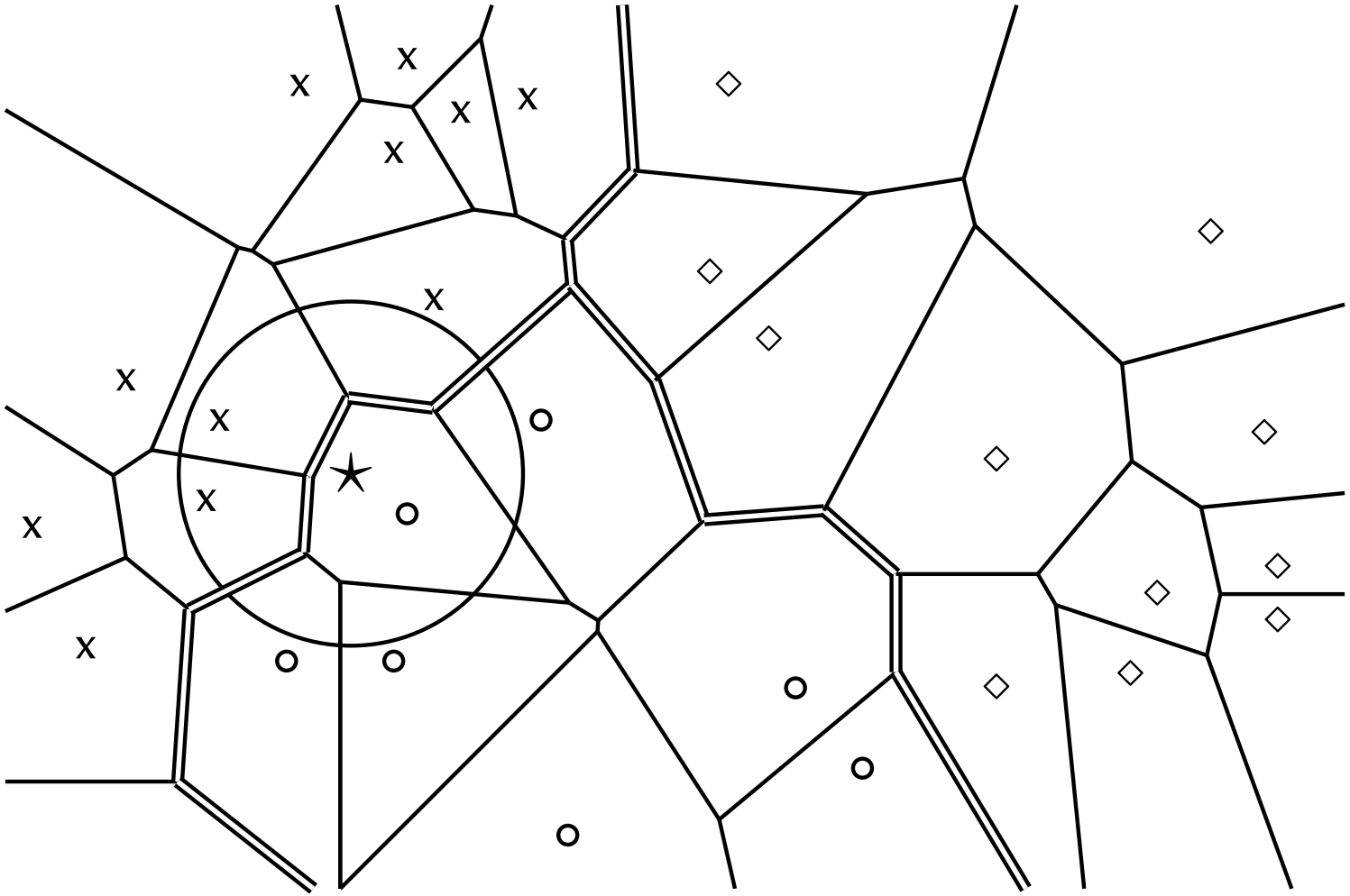
kNN classification

- kNN classification rule for $k = 1$ (1NN): Assign each test document to the class of its nearest neighbor in the training set.
- 1NN is not very robust, one document can be mislabeled or atypical.
- kNN classification rule for $k > 1$ (kNN): Assign each test document to the majority class of its k nearest neighbors in the training set.
- This amounts to locally defined decision boundaries between classes – far away points do not influence the classification decision.
- Rationale of kNN: We expect a test document d to have the same label as the training documents located in the local region surrounding d (contiguity hypothesis).

Probabilistic kNN

- Probabilistic version of kNN:
  $P(c|d)$ = fraction of k neighbors of d that are in c
- kNN classification rule for probabilistic kNN:
  Assign d to class c with highest $P(c|d)$

kNN is based on Voronoi tessellation

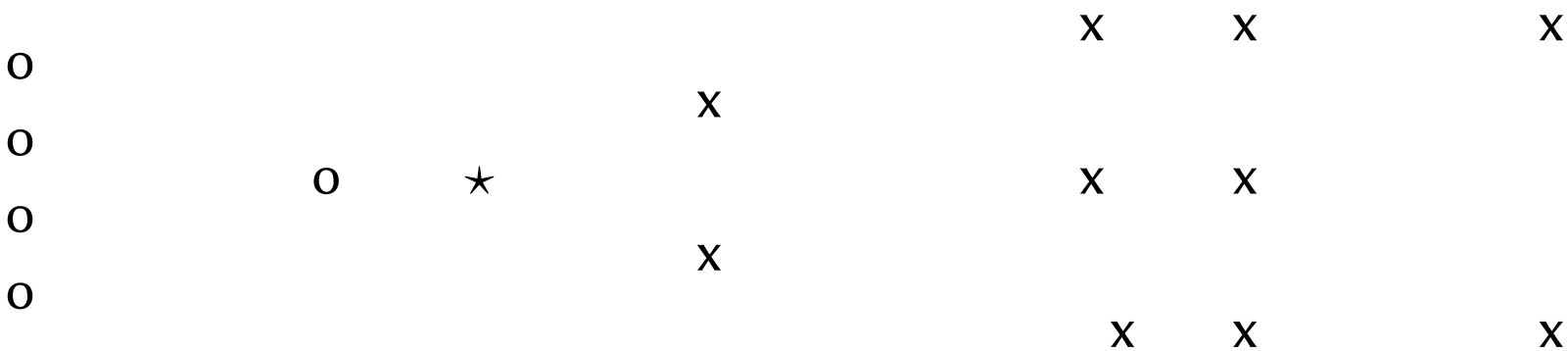


kNN algorithm

TRAIN-KNN($\mathbb{C}, \mathbb{D}$)
1 $\mathbb{D}' \leftarrow$ PREPROCESS($\mathbb{D}$)
2 $k \leftarrow$ SELECT-K($\mathbb{C}, \mathbb{D}'$)
3 **return** $\mathbb{D}', k$

APPLY-KNN($\mathbb{D}', k, d$)
1 $S_k \leftarrow$ COMPUTENEARESTNEIGHBORS($\mathbb{D}', k, d$)
2 **for each** $c_j \in \mathbb{C}(\mathbb{D}')$
3 **do** $p_j \leftarrow |S_k \cap c_j|/k$
4 **return** $\arg\max_j p_j$

Exercise



How is star classified by:
(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN

Time complexity of kNN

with preprocessing of training set

training $\Theta(|\mathbb{D}|L_{ave})$

testing $\Theta(L_a + |\mathbb{D}|M_{ave}M_a) = \Theta(|\mathbb{D}|M_{ave}M_a)$

without preprocessing of training set

training $\Theta(1)$

testing $\Theta(L_a + |\mathbb{D}|L_{ave}M_a) = \Theta(|\mathbb{D}|L_{ave}M_a)$

- M_{ave}, M_a is the size of vocabulary of a document (average, test)
- L_{ave}, L_a is the length of a document (average, test)
- kNN test time proportional to the size of the training set!
- The larger the training set, the longer it takes to classify a test doc.
- kNN is inefficient for very large training sets.

kNN with inverted index

- Naively finding nearest neighbors requires a linear search through $|\mathbb{D}|$ documents in collection.
- Finding k nearest neighbors is the same as determining the k best retrievals using the test document as a query to a database of training documents.
- Use standard vector space inverted index methods to find the k nearest neighbors.
- Testing time: $O(|\mathbb{D}|)$, that is, still linear in the number of documents. (Length of postings lists approximately linear in number of docs $\mathbb{D}$.)
- But constant factor much smaller for inverted index than for linear scan.

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
  - We always preprocess the training set, so in reality training time of kNN is linear.
- kNN is very accurate if training set is large.
- Optimality result: asymptotically zero error if Bayes rate is zero.
- But kNN can be very inaccurate if training set is small.

Linear classifiers

Linear classifiers

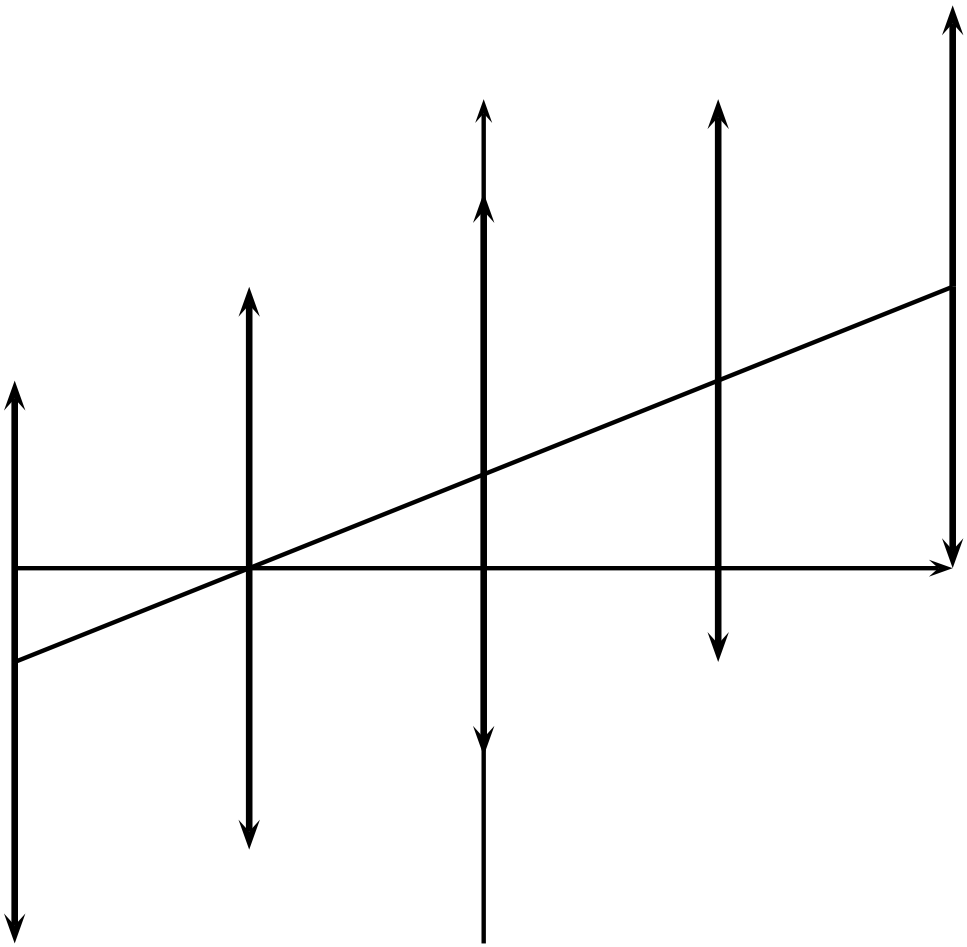
- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?

    ...where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.
- Methods for finding separator: Perceptron, Naive Bayes – as we will explain on the next slides
- Assumption: The classes are linearly separable.

A linear classifier in 1D

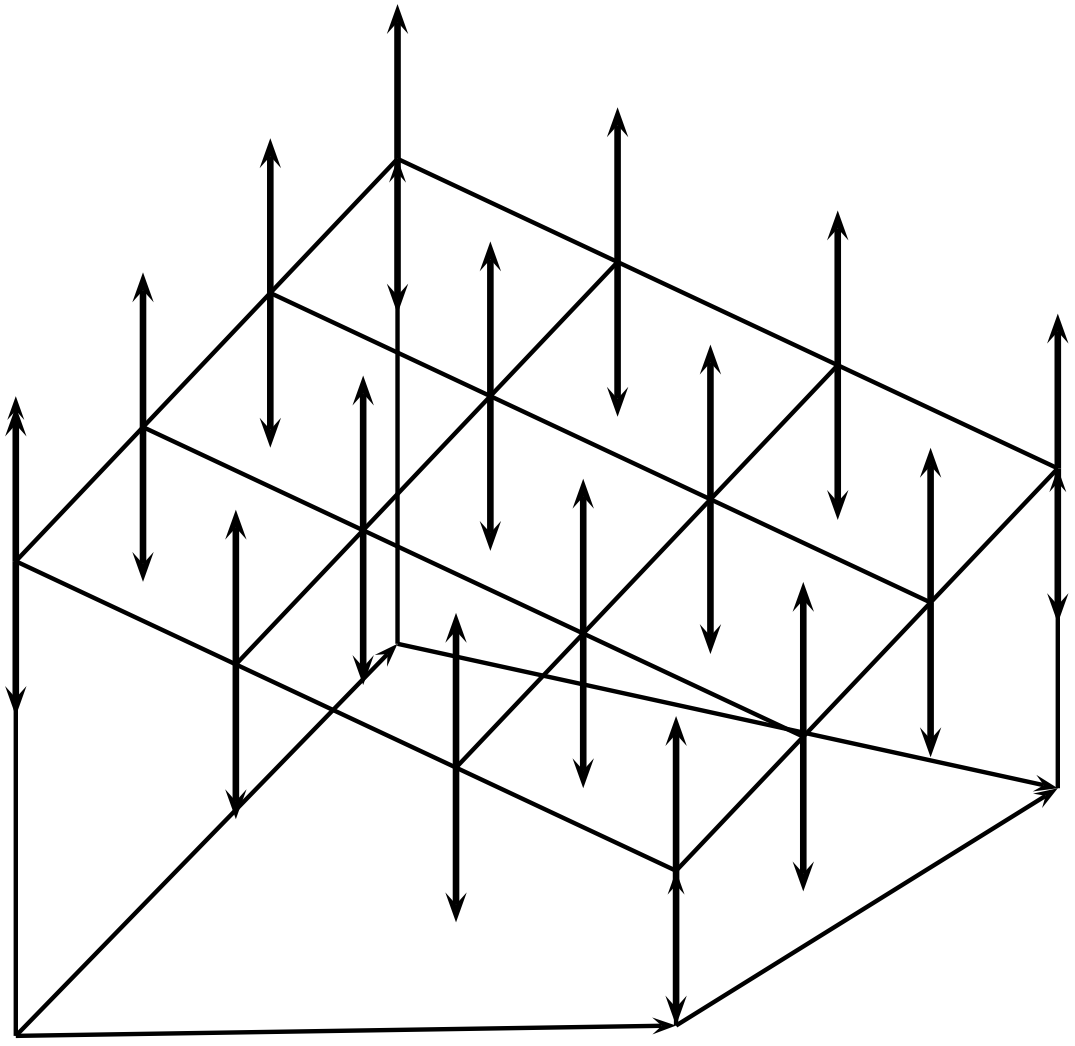
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1\ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1\ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\bar{c}$.

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1\ d_2\ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.
- Points $(d_1\ d_2\ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta$ are in the complement class $\bar{c}$.

Naive Bayes as a linear classifier

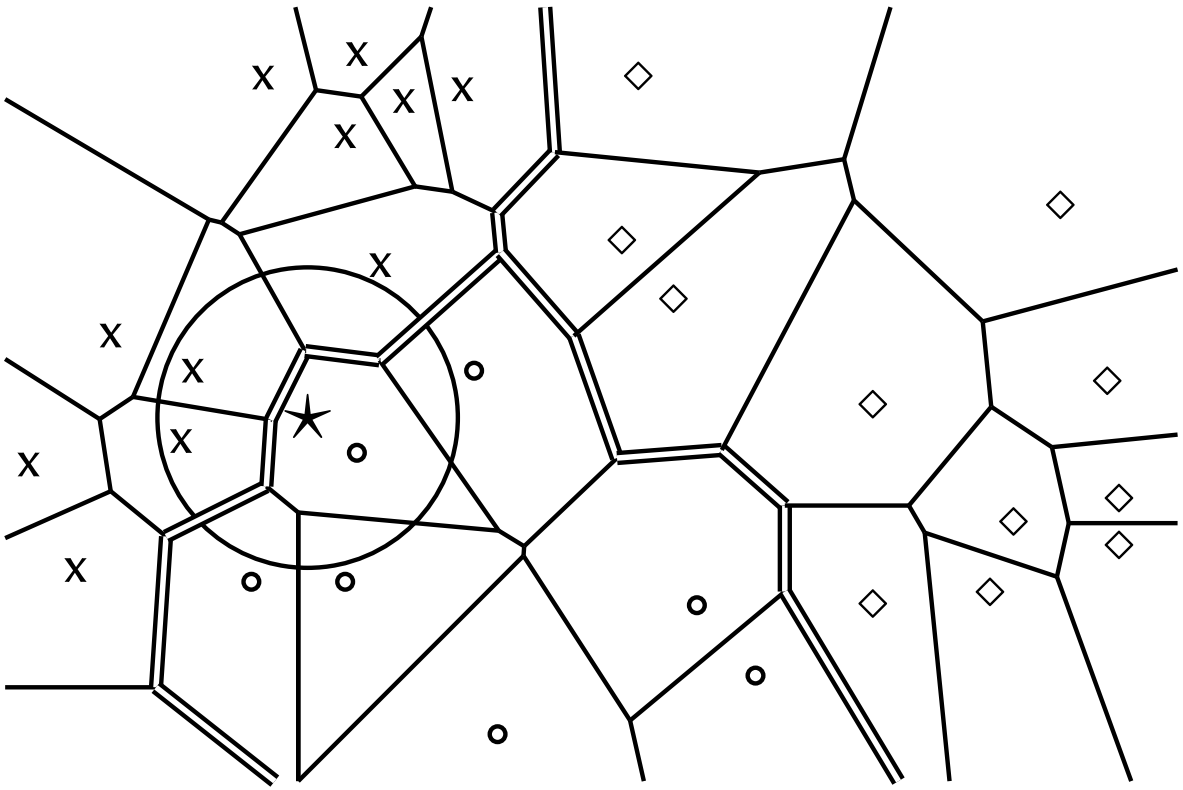
- Multinomial Naive Bayes is linear classifier (in log space) defined by:

$$\sum_{i=1}^{M} w_i d_i = \theta$$

- where
  - $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$,
  - d_i = number of occurrences of t_i in d, and
  - $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$.

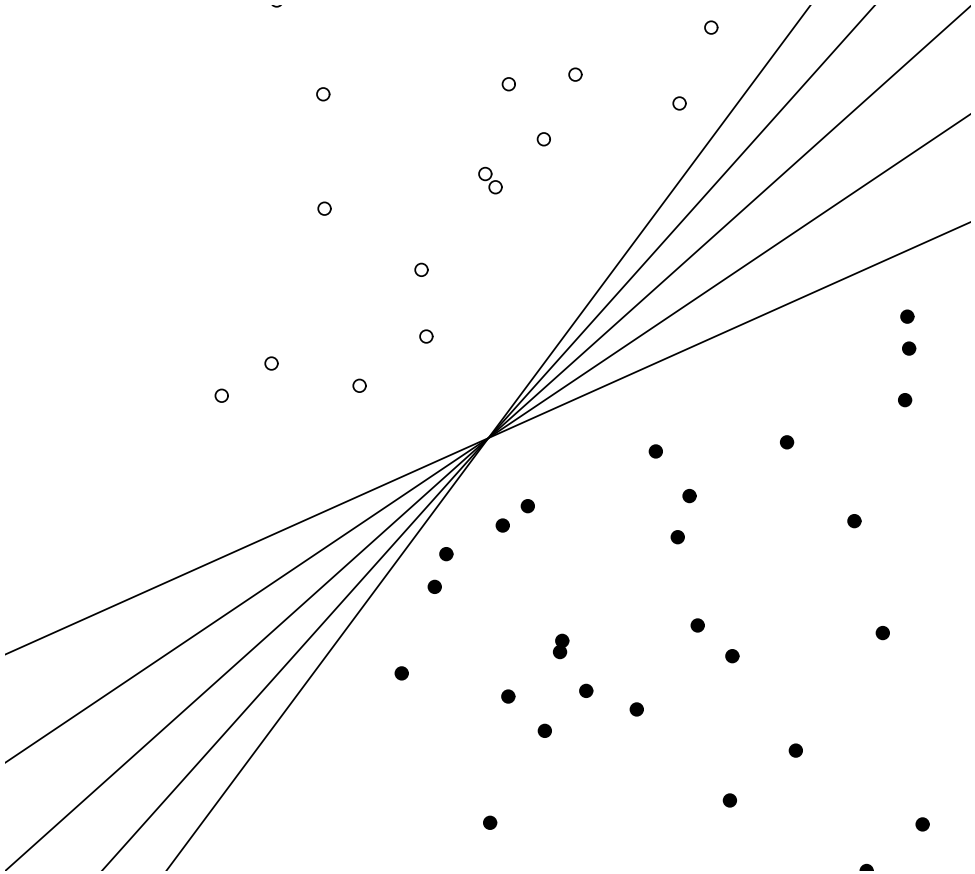
- Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary (not to positions in d as k did in our original definition of Naive Bayes)

kNN is not a linear classifier



- Classification decision based on majority of k nearest neighbors.
- The decision boundaries between classes are piecewise linear ...
- ...but they are in general not linear classifiers that can be described as $\sum_{i=1}^{M} w_i d_i = \theta$.

Which hyperplane?



Learning algorithms for vector space classification

- In terms of actual computation, there are two types of learning algorithms.
- (i) Simple learning algorithms that estimate the parameters of the classifier directly from the training data, often in one linear pass.
  - Naive Bayes, kNN are all examples of this.
- (ii) Iterative algorithms
  - Support vector machines
  - Perceptron
- The best performing learning algorithms usually require iterative learning.

Which hyperplane?

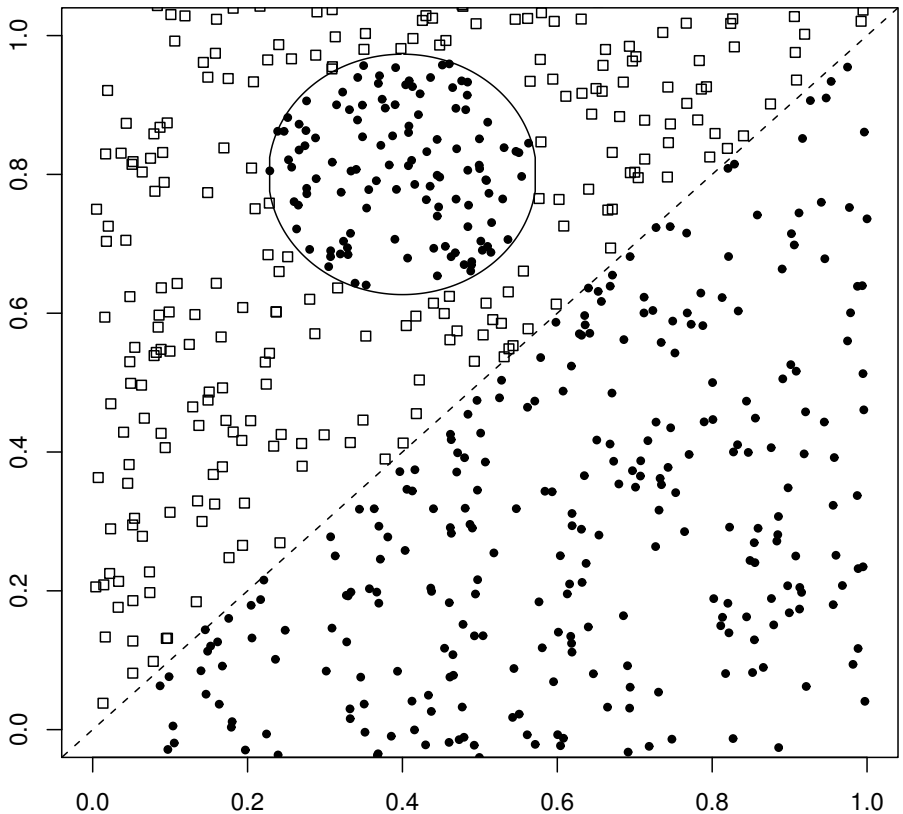
- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly ...

  ...but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?
- Perceptron: generally bad; Naive Bayes: ok, : linear SVM: good

Linear classifiers: Discussion

- Many common text classifiers are linear classifiers.
- Methods differ in the way of selecting the separating hyperplane.
- Huge differences in performance on test documents.
- Can we get better performance with more powerful nonlinear classifiers?
- Not in general: A given amount of training data may suffice for estimating a linear boundary, but not for estimating a more complex nonlinear boundary.

A nonlinear problem



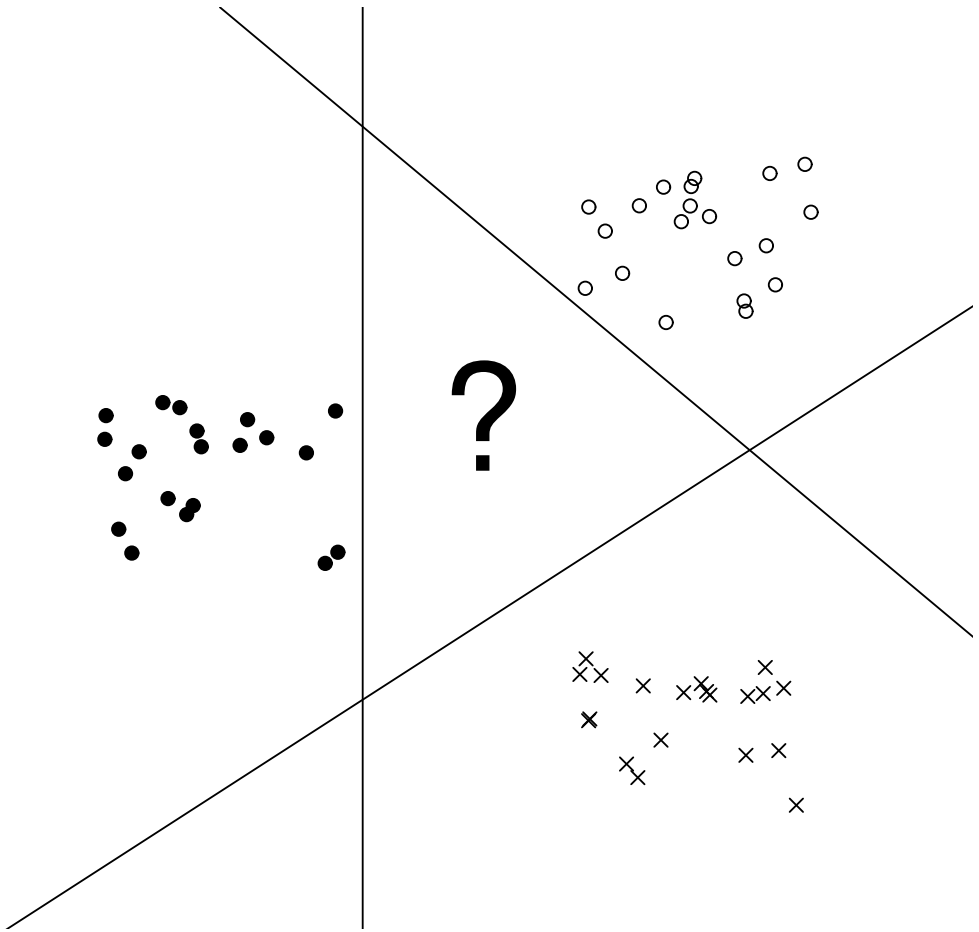
- Linear classifier like Naive Bayes does badly on this task.
- kNN will do well (assuming enough training data)

Which classifier do I use for a given TC problem?

- Is there a learning method 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?
  - How noisy is the problem?
  - How stable is the problem over time?

    (If unstable, it's better to use a simple and robust classifier.)

How to combine hyperplanes for > 2 classes?



One-of classification

- One-of or multiclass classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class.
  - Example: language of a document
    (assumption: no document contains multiple languages)
- Combine two-class linear classifiers as follows:
  - Run each classifier separately
  - Rank classifiers (e.g., according to score)
  - Pick the class with the highest score

Any-of classification

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves decisions open on all other classes.
  - A type of "independence" (but not statistical independence)
  - Example: topic classification
  - Usually: make decisions on the region, on the subject area, on the industry and so on "independently"
- Combine two-class linear classifiers as follows:
  - Simply run each two-class classifier separately on the test document and assign document accordingly

Support vector machines

Support vector machines

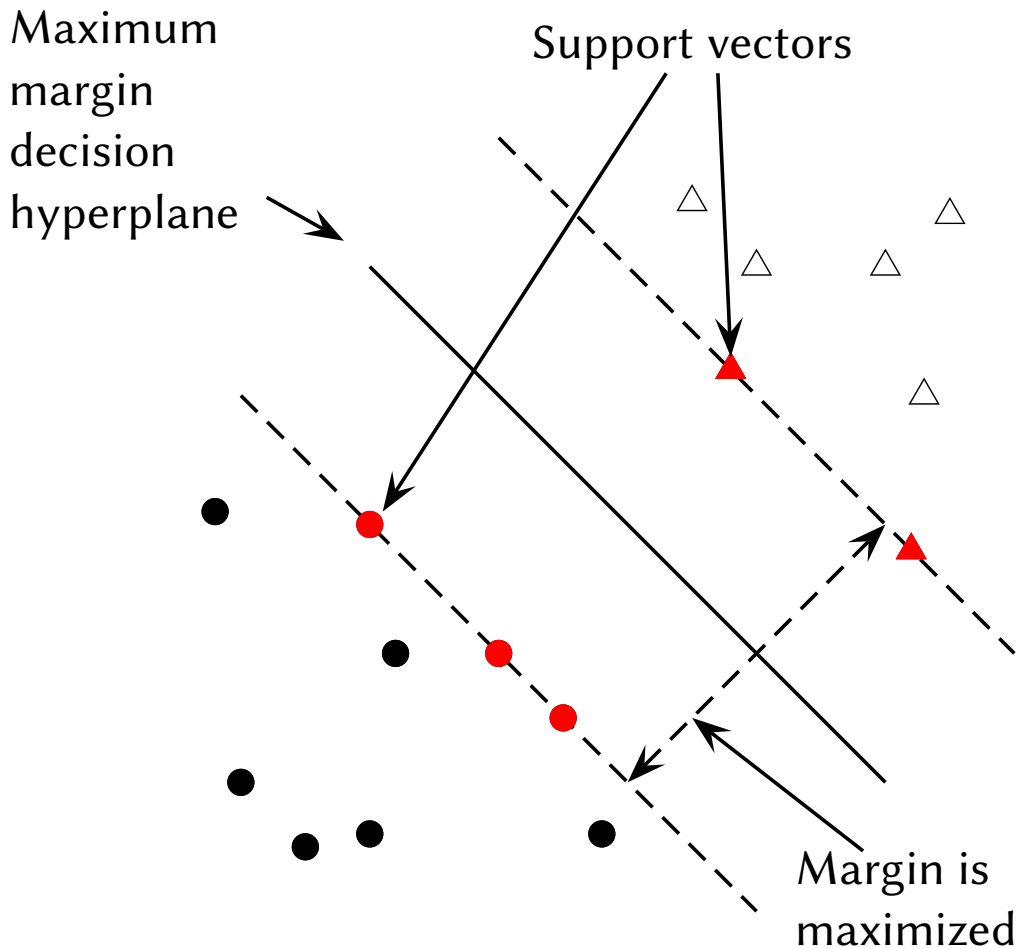
- Machine-learning research in the last two decades has improved classifier effectiveness.
- New generation of state-of-the-art classifiers: support vector machines (SVMs), boosted decision trees, regularized logistic regression, neural networks, and random forests
- Applications to IR problems, particularly text classification

SVMs: A kind of large-margin classifier

Vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

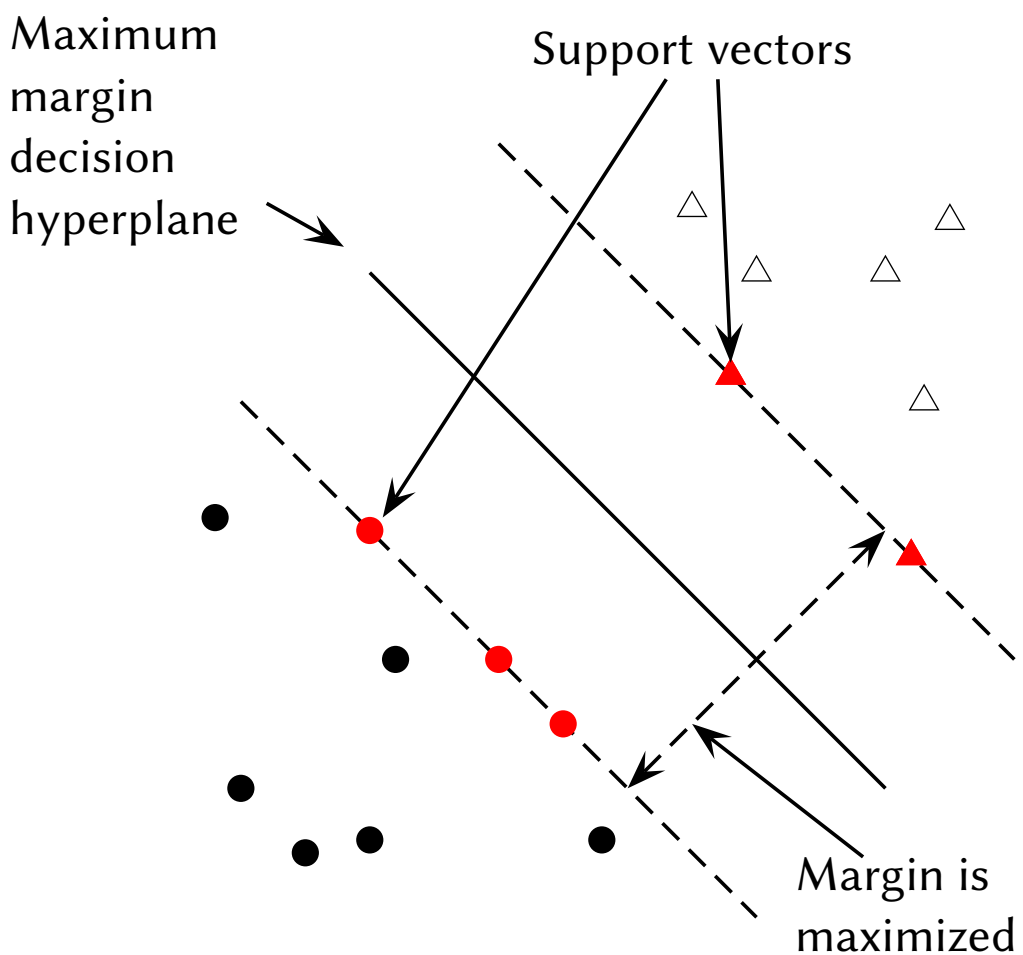
Support Vector Machines

- 2-class training data
- decision boundary → linear separator
- criterion: being maximally far away from any data point → determines classifier margin
- linear separator position defined by support vectors



Why maximize the margin?

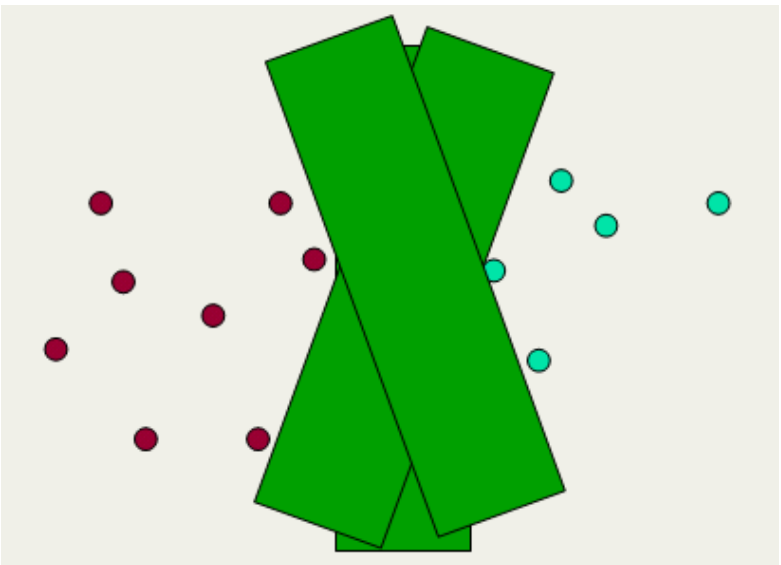
- Points near decision surface → uncertain classification decisions (50% either way).
- A classifier with a large margin makes no low certainty classification decisions.
- Gives classification safety margin w.r.t slight errors in measurement or doc. variation



Why maximize the margin?

SVM classifier: large margin around decision boundary

- compare to decision hyperplane: place fat separator between classes
- unique solution
- decreased memory capacity
- increased ability to correctly generalize to test data



Separating hyperplane: Recap

Hyperplane

An n-dimensional generalization of a plane (point in 1-D space, line in 2-D space, ordinary plane in 3-D space).

Decision hyperplane

Can be defined by:

- intercept term b
- normal vector $\vec{w}$ (**weight vector**) which is perpendicular to the hyperplane

All points $\vec{x}$ on the hyperplane satisfy:

$$\vec{w}^T\vec{x} = -b$$

Formalization of SVMs

Training set

Consider a binary classification problem:

- $\vec{x}_i$ are the input vectors
- y_i are the labels

For SVMs, the two data classes are $y_i = +1$ and $y_i = -1$, and the intercept term is explicitly represented as b.

The linear classifier is then:

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

A value of -1 indicates one class, and a value of $+1$ the other class.

Functional margin of a point

We are confident in the classification of a point if it is far away from the decision boundary.

Functional margin

The functional margin of the vector $\vec{x}_i$ w.r.t the hyperplane $\langle \vec{w}, b \rangle$ is:

$$y_i(\vec{w}^T \vec{x}_i + b)$$

The functional margin of a data set w.r.t a decision surface is twice the functional margin of any of the points in the data set with minimal functional margin

- factor 2 comes from measuring across the whole width of the margin

But we can increase functional margin by scaling $\vec{w}$ and b.

We need to place some constraint on the size of the $\vec{w}$ vector.

Geometric margin

Geometric margin of the classifier: maximum width of the band that can be drawn separating the support vectors of the two classes.

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

The geometric margin is clearly invariant to scaling of parameters: if we replace $\vec{w}$ by $5\vec{w}$ and b by $5b$, then the geometric margin is the same, because it is normalized by the length of $\vec{w}$.

Optimization problem solved by SVMs

Assume canonical distance

Assume that all data is at least distance 1 from the hyperplane, then:

$$y_i(\vec{w}^T\vec{x}_i + b) \geq 1$$

Since each example's distance from the hyperplane is $r_i = y_i(\vec{w}^T\vec{x}_i + b)/|\vec{w}|$, the geometric margin is $\rho = 2/|\vec{w}|$.

We want to maximize this geometric margin.

That is, we want to find $\vec{w}$ and b such that:

- $\rho = 2/|\vec{w}|$ is maximized
- For all $(\vec{x}_i, y_i) \in \mathbb{D}$, $y_i(\vec{w}^T\vec{x}_i + b) \geq 1$

Optimization problem solved by SVMs (2)

Maximizing $2/|\vec{w}|$ is the same as minimizing $|\vec{w}|/2$. This gives the final standard formulation of an SVM as a minimization problem:

Example

Find $\vec{w}$ and b such that:

- $\frac{1}{2}\vec{w}^T\vec{w}$ is minimized (because $|\vec{w}| = \sqrt{\vec{w}^T\vec{w}}$), and
- for all $\{(\vec{x}_i, y_i)\}$, $y_i(\vec{w}^T\vec{x}_i + b) \geq 1$

We are now optimizing a **quadratic function** subject to linear constraints. Quadratic optimization problems are standard mathematical optimization problems, and many algorithms exist for solving them (e.g. Quadratic Programming libraries).

Recap

- We start with a training set.
- The data set defines the maximum-margin separating hyperplane (if it is separable).
- We use quadratic optimization to find this plane.
- Given a new point $\vec{x}$ to classify, the classification function $f(\vec{x})$ computes the projection of the point onto the hyperplane normal.
- The sign of this function determines the class to assign to the point.
- If the point is within the margin of the classifier, the classifier can return "don't know" rather than one of the two classes.
- The value of $f(\vec{x})$ may also be transformed into a probability of classification

Soft margin classification

What happens if data is not linearly separable?

- Standard approach: allow the fat decision margin to make a few mistakes
- some points, outliers, noisy examples are inside or on the wrong side of the margin
- Pay cost for each misclassified example, depending on how far it is from meeting the margin requirement

Slack variable ξ_i: A non-zero value for ξ_i allows $\vec{x}_i$ to not meet the margin requirement at a cost proportional to the value of ξ_i.

SVM with slack variables

Slack variable ξ_i: a non-zero value for ξ_i allows $\vec{x}_i$ to not meet the margin requirement at a cost proportional to the value of ξ_i.

Example

Find $\vec{w}$ and b such that:

- $\frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i$ is minimized (because $|\vec{w}| = \sqrt{\vec{w}^T\vec{w}}$), and
- for all $\{(\vec{x}_i, y_i)\}$, $y_i(\vec{w}^T\vec{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$

Optimization problem: trading off how fat it can make the margin vs. how many points have to be moved around to allow this margin.

The sum of the ξ_i gives an upper bound on the number of training errors.

Soft-margin SVMs minimize training error traded off against margin.

Binary classification → One-of multiclass classification

- Many classification algorithms are binary.
- What do we do for one-of multiclass classification: we have $k > 2$ classes and the k classes are mutually exclusive?
- Common technique: build $|\mathbb{C}|$ one-versus-rest classifiers (commonly referred to as "one-versus-all" or OVA classification), and choose the class which classifies the test data with highest probability (probabilistic classifier) or greatest margin (SVM)
- Another strategy: build a set of one-versus-one classifiers, and choose the class that is selected by the most classifiers. While this involves building $|\mathbb{C}|(|\mathbb{C}| - 1)/2$ classifiers, the time for training classifiers may actually decrease, since the training data set for each classifier is much smaller.

Text classification

- Many commercial applications
- There are many applications of text classification for corporate Intranets, government departments, and Internet publishers.
- Often greater performance gains from exploiting domain-specific text features than from changing from one machine learning method to another.
- Understanding the data is one of the keys to successful categorization, yet this is an area in which many categorization tool vendors are weak.

Choosing what kind of classifier to use

When building a text classifier, first question:
How much training data is there currently available?

Practical challenge: creating or obtaining enough training data

Hundreds or thousands of examples from each class are required to produce a high performance classifier and many real world contexts involve large sets of categories.

- None?
- Very little?
- Quite a lot?
- A huge amount, growing every day?

No labeled training data

Use hand-written rules.

Example

IF (wheat OR grain) AND NOT (whole OR bread) THEN c = grain

In practice, rules get a lot bigger than this, and can be phrased using more sophisticated query languages than just Boolean expressions, including the use of numeric scores. With careful crafting, the accuracy of such rules can become very high (high 90% precision, high 80% recall). Nevertheless the amount of work to create such well-tuned rules is very large. A reasonable estimate is 2 days per class, and extra time has to go into maintenance of rules, as the content of documents in classes drifts over time.

Fairly little data and training a supervised classifier

Work out how to get more labeled data as quickly as you can.

- Best way: insert yourself into a process where humans will be willing to label data for you as part of their natural tasks.

Example

Often humans will sort or route email for their own purposes, and these actions give information about classes.

Active Learning

A system is built which decides which documents a human should label. Usually these are the ones on which a classifier is uncertain of the correct classification.

Fair amount of labeled data

Good amount of labeled data, but not huge

- Use everything that we have presented about text classification.
- Consider hybrid approach (overlay Boolean classifier).

Huge amount of labeled data

- Choice of classifier probably has little effect on your results.
- Choose classifier based on the scalability of training or runtime efficiency.

Rule of thumb: each doubling of the training

data size produces a linear increase in classifier performance, but with very large amounts of data, the improvement becomes sub-linear.