

Part 10: Vector space classification



Francesco Ricci

Most of these slides comes from the
course:

Information Retrieval and Web Search,
Christopher Manning and Prabhakar
Raghavan

Content

- Recap on naïve Bayes
- Vector space methods for Text Classification
 - K Nearest Neighbors
 - Bayes error rate
 - Decision boundaries
 - Vector space classification using centroids
 - Decision Trees (briefly)
- Bias/Variance decomposition of the error
- Model selection
- Generalization

Recap: Multinomial Naïve Bayes classifiers

- Classify based on prior weight of class and conditional parameter for what each word says:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$$

- Training is done by counting and dividing:

$$P(c_j) \leftarrow \frac{N_{c_j}}{N} \quad P(x_k | c_j) \leftarrow \frac{T_{c_j x_k} + \alpha}{\sum_{x_i \in V} [T_{c_j x_i} + \alpha]}$$

- Don't forget to smooth.

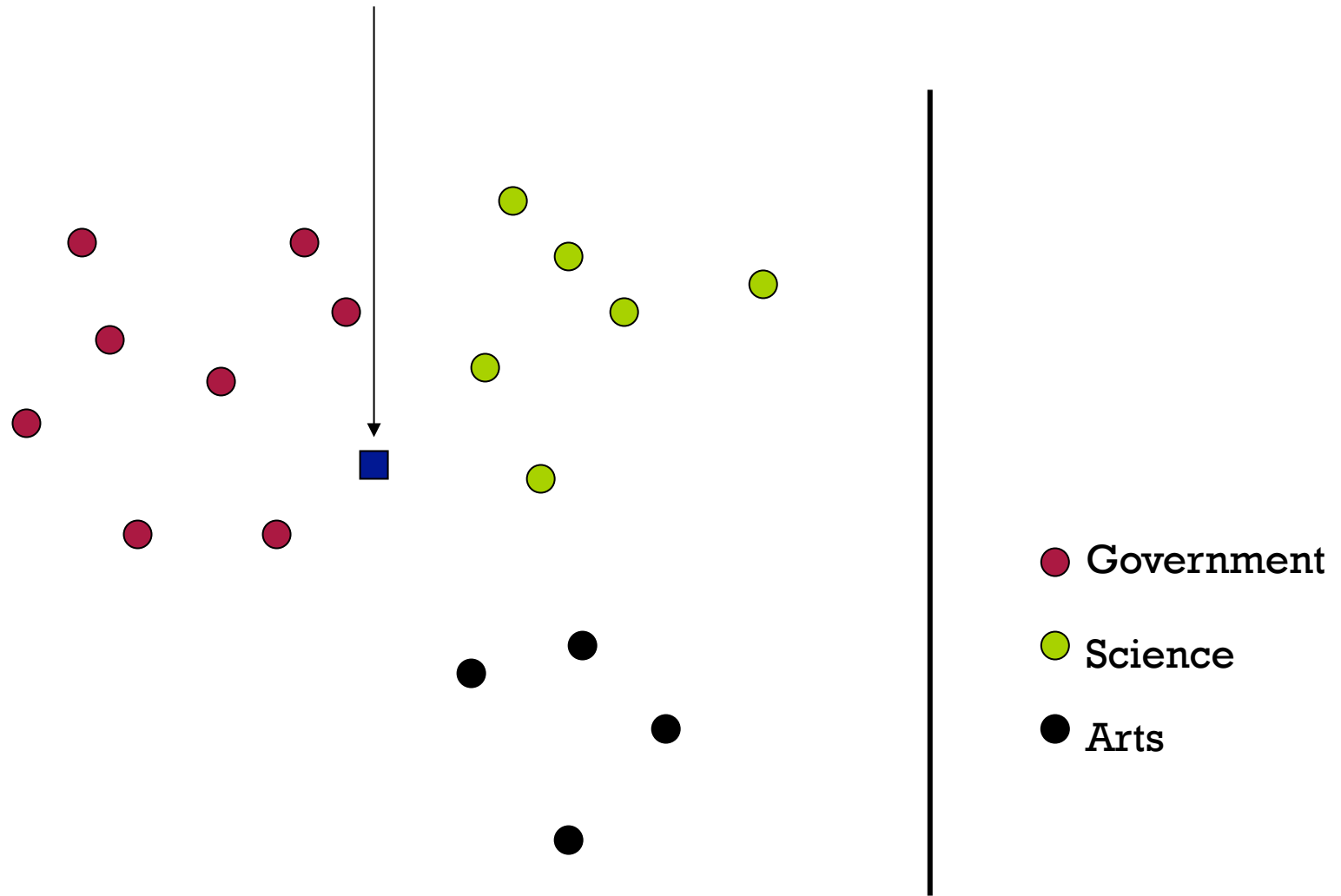
Vector Space Representation

- Each document is a vector, one component for each term (= word)
- Normally normalized vectors to unit length
- High-dimensional vector space:
 - Terms are axes
 - 10,000+ dimensions, or even 100,000+
 - Docs are vectors in this space
- How can we do classification in this space?
- How we can obtain high classification accuracy on data unseen during training?

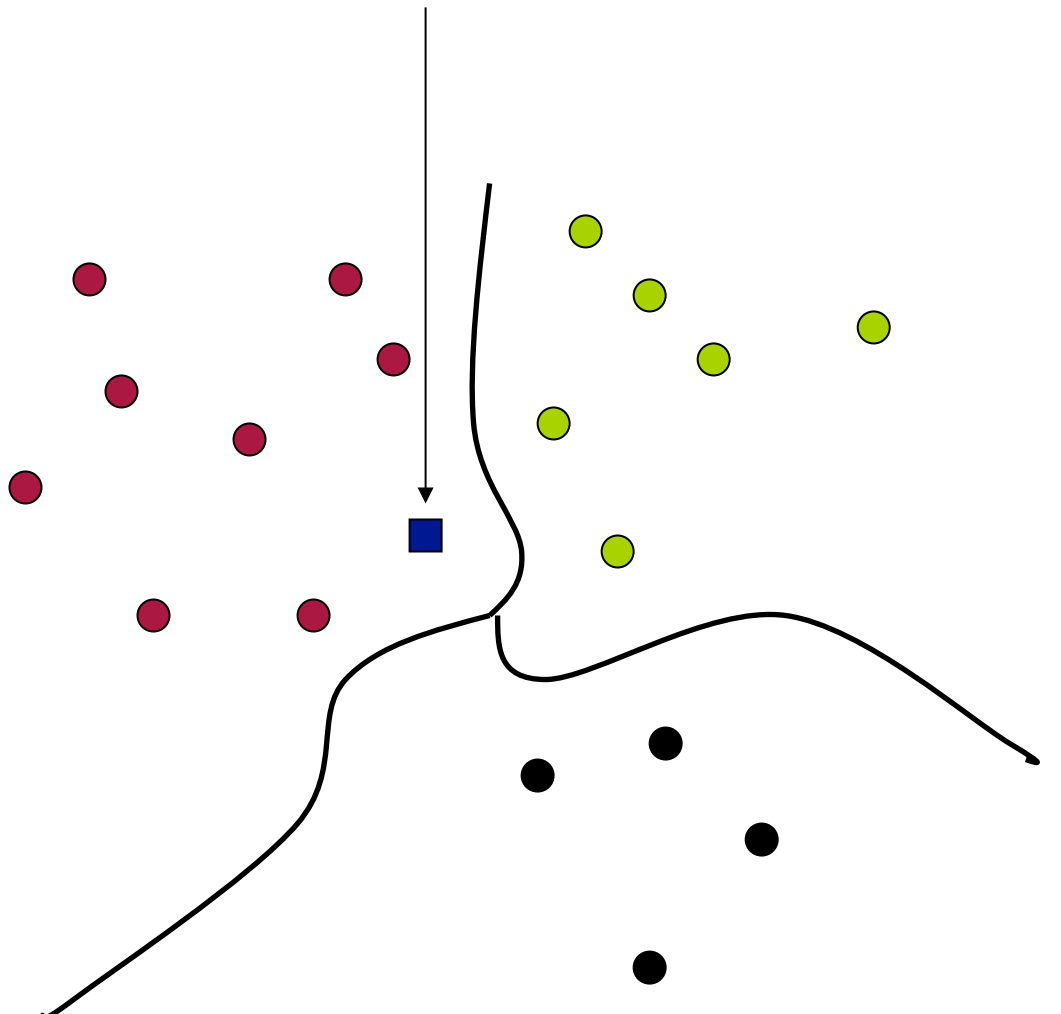
Classification Using Vector Spaces

- As before, the training set is a set of documents, each labeled with its class (e.g., topic)
- In vector space classification, this set corresponds to a labeled set of points (or, equivalently, vectors) in the vector space
- *Premise 1:* Documents in the same class form a contiguous region of space
- *Premise 2:* Documents from different classes don't overlap (much)
- *Goal:* Search for surfaces to delineate classes in the space.

Test Document of what class?



Test Document = Government

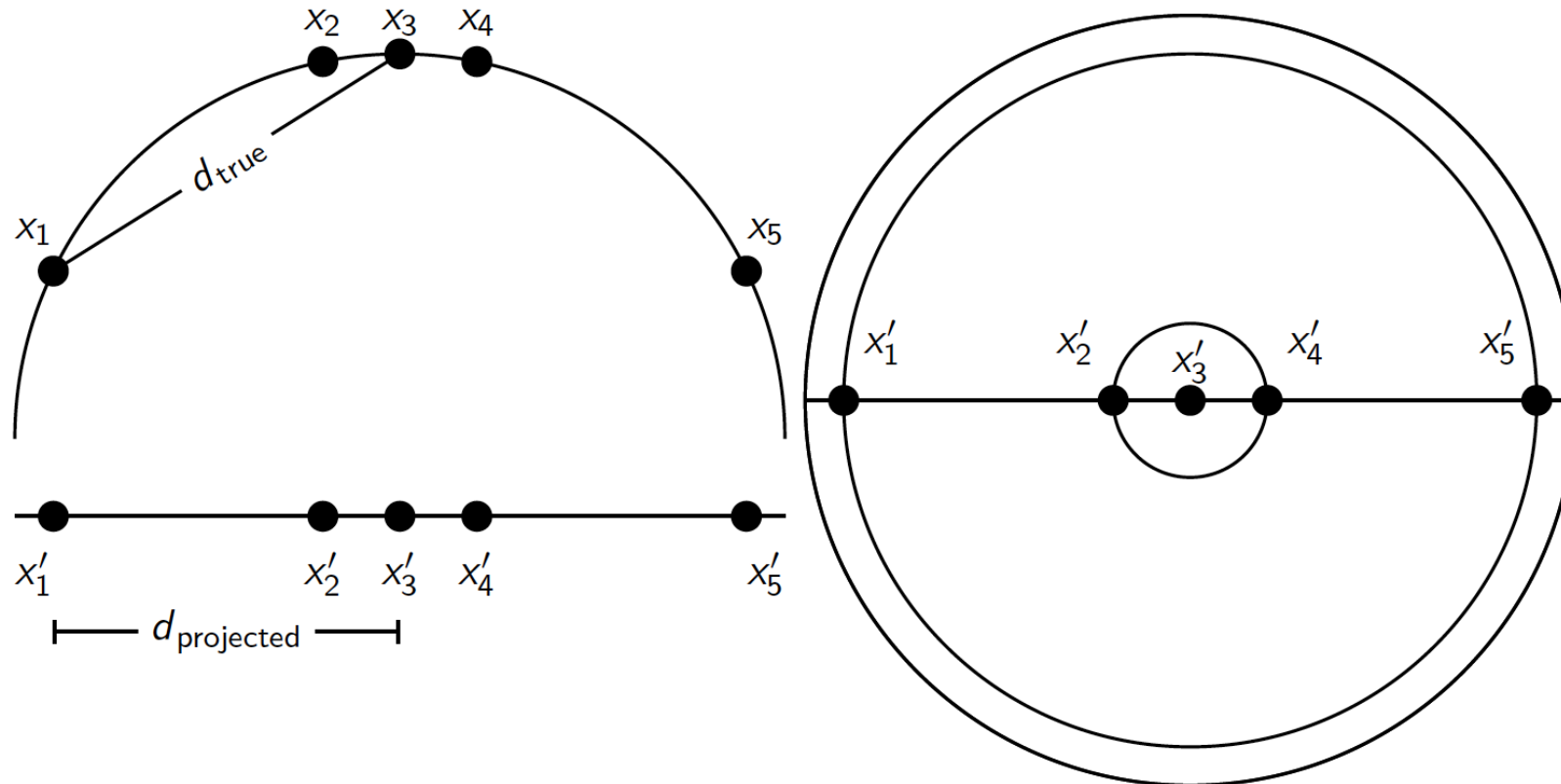


Is this similarity hypothesis true in general?

- Government
- Science
- Arts

Our main topic today is how to find good separators

Aside: 2D/3D graphs can be misleading



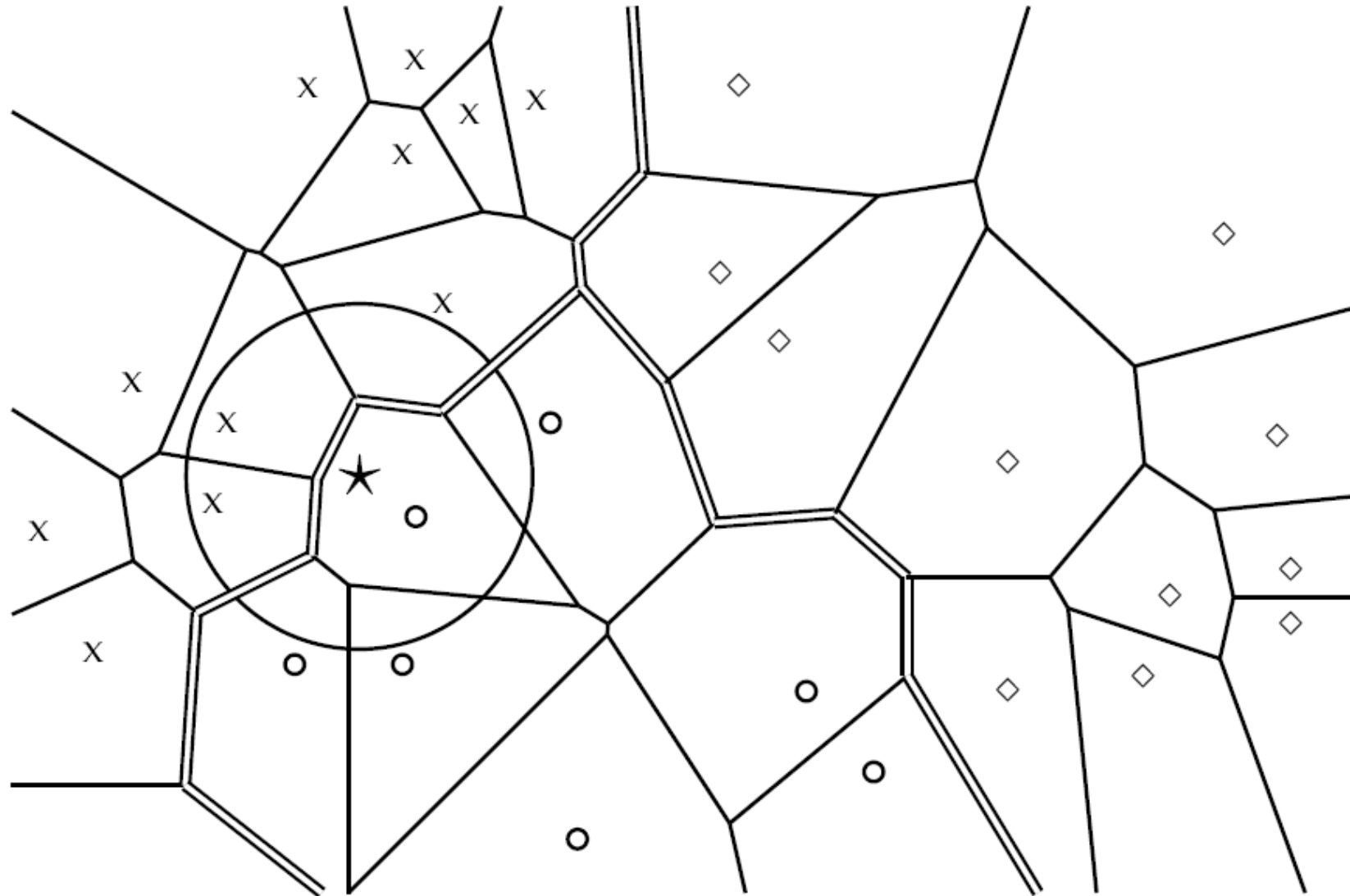
Left: A projection of the 2D semicircle to 1D. For the points x_1, x_2, x_3, x_4, x_5 at x coordinates $-0.9, -0.2, 0, 0.2, 0.9$ the distance $|x_2x_3| \approx 0.201$ only differs by 0.5% from $|x'_2x'_3| = 0.2$; but $|x_1x_3|/|x'_1x'_3| = d_{\text{true}}/d_{\text{projected}} \approx 1.06/0.9 \approx 1.18$ is an example of a large distortion (18%) when projecting a large area. *Right:* The corresponding projection of the 3D hemisphere to 2D.

Nearest-Neighbor Learning Algorithm

- ❑ **Learning: just storing** the training examples in D
- ❑ **Testing** a new instance x (under 1NN):
 - Compute similarity between x and all examples in D
 - Assign example x to the category of the most similar example in D
- ❑ Does not explicitly compute a **generalization** or **category prototypes**
- ❑ Also called:
 - Case-based learning
 - Memory-based learning
 - Lazy learning
- ❑ Rationale of kNN: **contiguity hypothesis.**

Is Naïve Bayes building such a *generalization*?

Decision Boundary: Voronoi Tessellation



<http://www.diku.dk/hjemmesider/studerende/duff/Fortune/>

k Nearest Neighbor

- Using only the closest example (1NN) to determine the class is **subject to errors** due to:
 - A **single atypical example** may be close to the test examples
 - **Noise** (i.e., an 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.

Editing the Training Set

- Not all points are really needed.

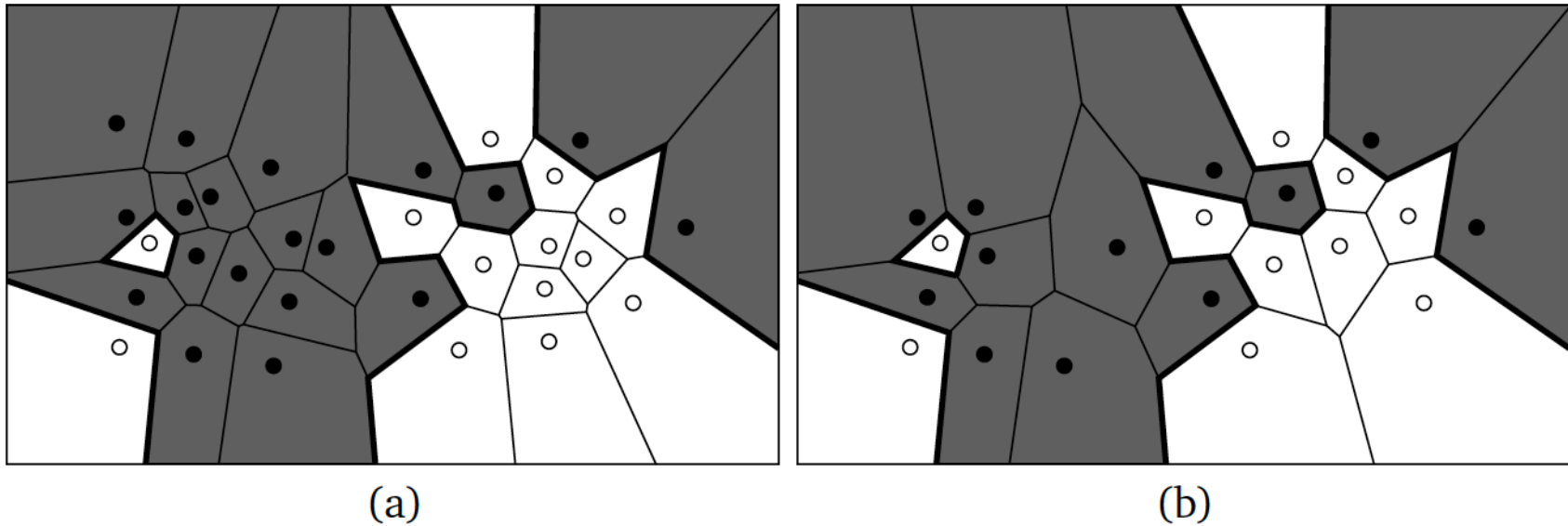


Fig. 1. The Voronoï diagram (a) before Voronoï condensing and (b) after Voronoï condensing. Note that the decision boundary (in bold) is unaffected by Voronoï condensing. Note: In this figure, and all other figures, red points are denoted by white circles and blue points are denoted by black disks.

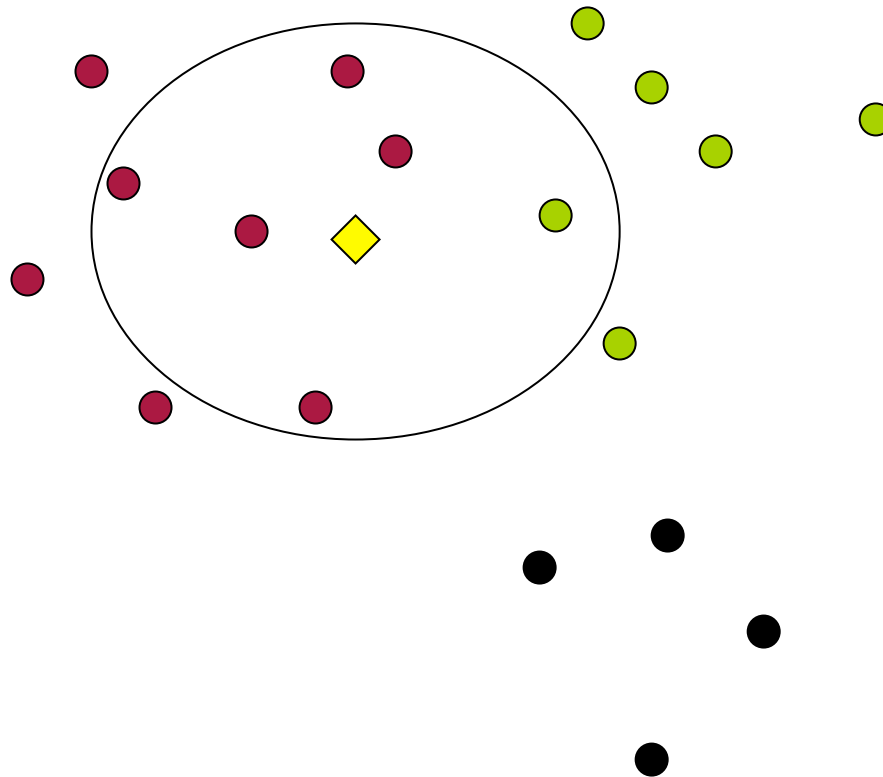
David Bremner, Erik Demaine, Jeff Erickson, John Iacono, Stefan Langerman, Pat Morin, and Godfried Toussaint. 2005. Output-Sensitive Algorithms for Computing Nearest-Neighbour Decision Boundaries. *Discrete Comput. Geom.* 33, 4 (April 2005), 593-604.

k Nearest Neighbor Classification

- kNN = k Nearest Neighbor
- Learning: **just storing** the representations of the training examples in D
- To classify document d into class c :
 - Define the k -neighborhood U as the k nearest neighbors of d
 - Count c_U : number of documents in U that belong to c
 - Estimate $P(c|d)$ as c_U/k
 - Choose as class $\operatorname{argmax}_c P(c|d)$ [= majority class].

Why we do not do smoothing?

Example: k=6 (6NN)



$P(\text{science} | \diamond)$?

- Government
- Science
- Arts

Illustration of 3 Nearest Neighbor for Text Vector Space



Distance-based Scoring

- Instead of using the number of nearest neighbours in a class as measure of class probability one can use cosine distance-based score

$$\text{score}(c, d) = \sum_{d' \in S_k(d)} I_c(d') \cos(\vec{v}(d'), \vec{v}(d))$$

- $S_k(d)$ is the set of nearest neighbours of d , $I_c(d') = 1$ iff d' is in class c and 0 otherwise
- $P(c_j|d) = \text{score}(c_j, d) / \sum_i \text{score}(c_i|d)$.

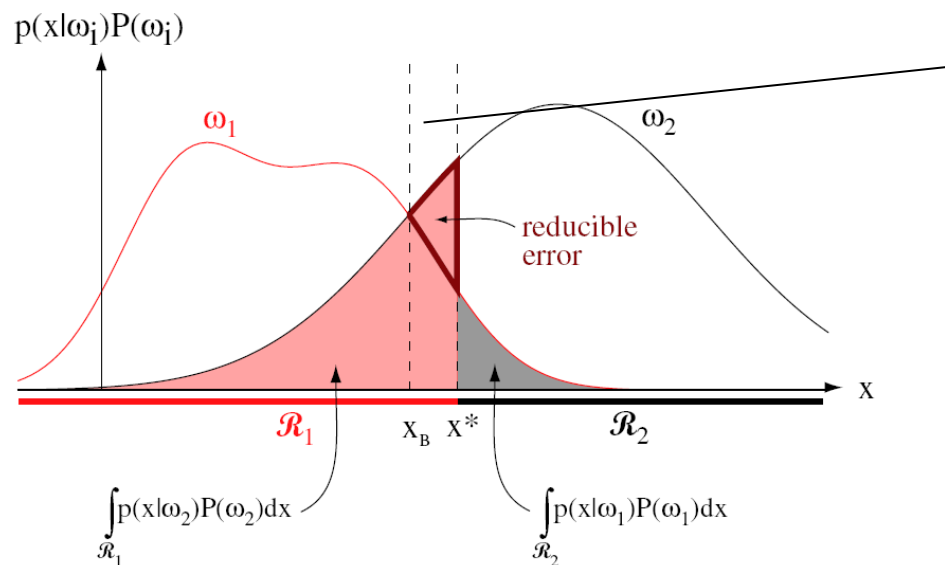
kNN is Close to Optimal

- Cover and Hart (1967)
- Asymptotically, the error rate of **1-nearest-neighbor** classification is less than twice the Bayes rate
 - *What is the meaning of "asymptotic" here?*
- Corollary: 1NN asymptotic error rate is 0 if Bayes rate is 0 (why?)
- **k-nearest neighbour** is guaranteed to approach the Bayes error rate, for some value of k (where k increases as a function of the number of data points).

Bayes Error Rate

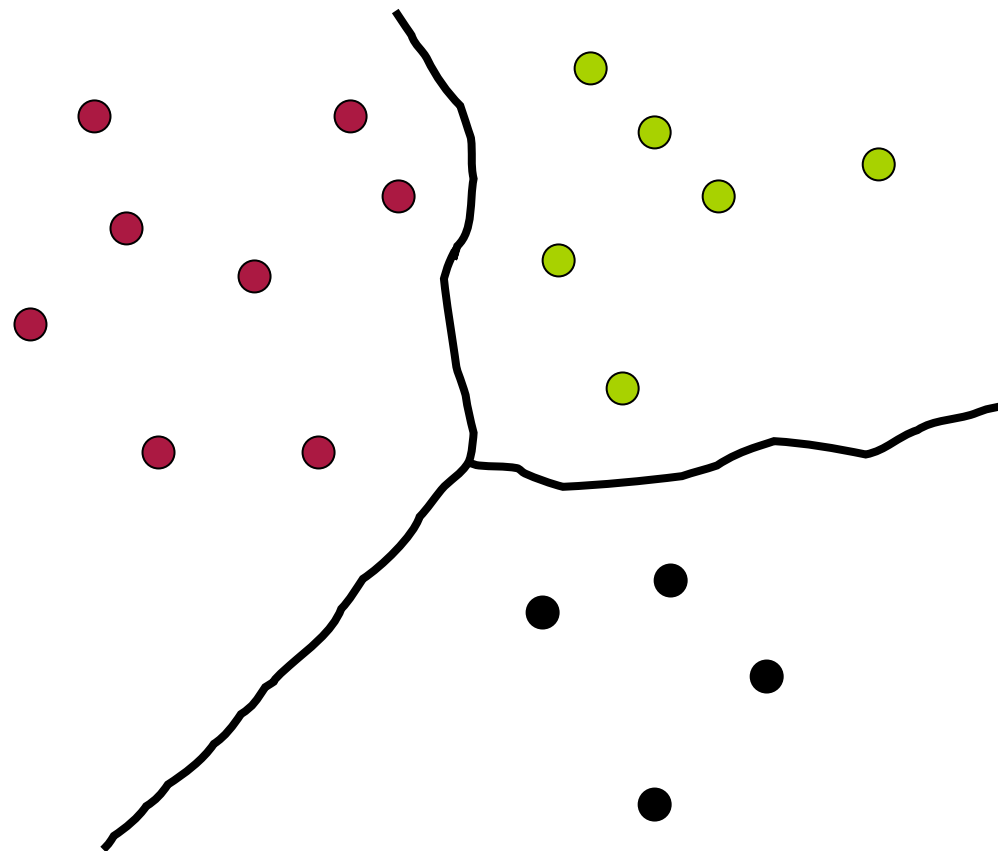
- R_1 and R_2 are the two regions defined by the classifier
- ω_1 and ω_2 are the distributions of the examples in the two classes

$$\begin{aligned}
 P(\text{error}) &= P(\mathbf{x} \in \mathcal{R}_2, \omega_1) + P(\mathbf{x} \in \mathcal{R}_1, \omega_2) \\
 &= P(\mathbf{x} \in \mathcal{R}_2 | \omega_1)P(\omega_1) + P(\mathbf{x} \in \mathcal{R}_1 | \omega_2)P(\omega_2) \\
 &= \int_{\mathcal{R}_2} p(\mathbf{x} | \omega_1)P(\omega_1) d\mathbf{x} + \int_{\mathcal{R}_1} p(\mathbf{x} | \omega_2)P(\omega_2) d\mathbf{x}.
 \end{aligned}$$



The error is minimal if x_B is the selected class separation. But there is still an "unavoidable" error.

kNN decision boundaries



Boundaries are in principle arbitrary surfaces – but for knn are 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.)

Similarity Metrics

- ❑ Nearest neighbor method **depends** on a similarity (or distance) metric – *different metric -> different classification*
- ❑ Simplest for continuous m-dimensional instance space is **Euclidean** distance
- ❑ Simplest for m-dimensional binary instance space is **Hamming** distance (number of feature values that differ)
- ❑ When the input space is made of numeric and nominal features use **Heterogeneous** distance functions
- ❑ Distance functions can be also defined locally – different distances for different part of the input space
- ❑ For text, cosine similarity of tf.idf weighted vectors is typically most effective.

Nearest Neighbor with Inverted Index

- Naively finding nearest neighbors requires a linear search through $|D|$ documents in collection
- But determining 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:** $O(B|V_t|)$ where B is the average number of training documents in which a test-document word appears, and $|V_t|$ is the dimension of the vector space
 - Typically $B \ll |D|$

kNN: Discussion

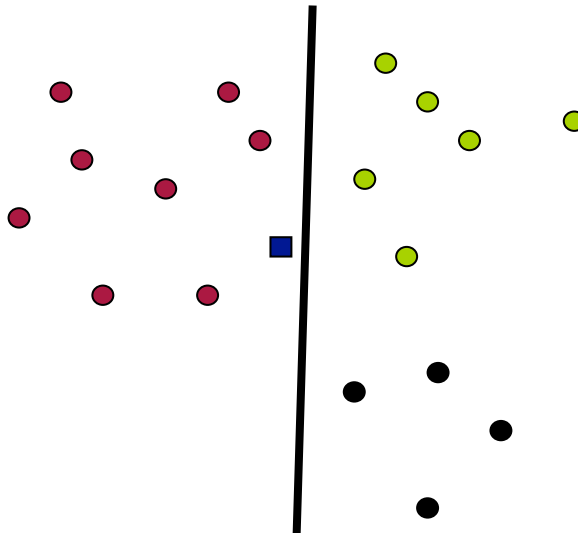
- ❑ **No feature selection** necessary – *but it is sometime useful*
- ❑ Scales well with large number of classes
 - Don't need to train n classifiers for n classes
- ❑ Classes can influence each other
 - Small changes to one class can have ripple effect
- ❑ No training necessary
 - Actually: perhaps not true - Data editing, etc. (edited NN techniques)
- ❑ May be more expensive at test time.

Linear classifiers and binary classification

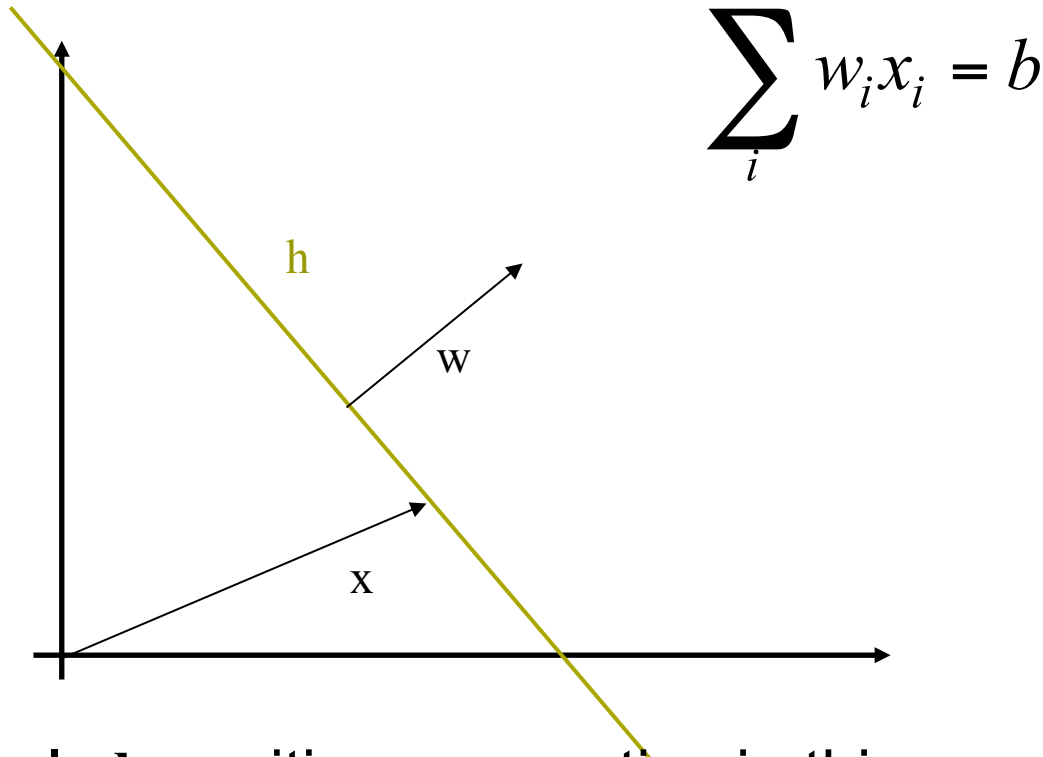
- Consider 2 class problems
 - Deciding between two classes, perhaps, government and non-government
 - It is also the situation when we want to solve the problem: One-versus-rest classification (if there are more classes)
- How do we define (and find) the **separating surface**?
- How do we decide which region a test doc is in?

Separation by Hyperplanes

- A strong **high-bias** assumption is *linear separability*:
 - in 2 dimensions, can separate classes by a line
 - in higher dimensions, need hyperplanes
- Can find separating hyperplane by *linear programming*
- Or can iteratively fit solution via perceptron
- separator can be expressed as $w_1x + w_2y = b$



The hyperplane equation

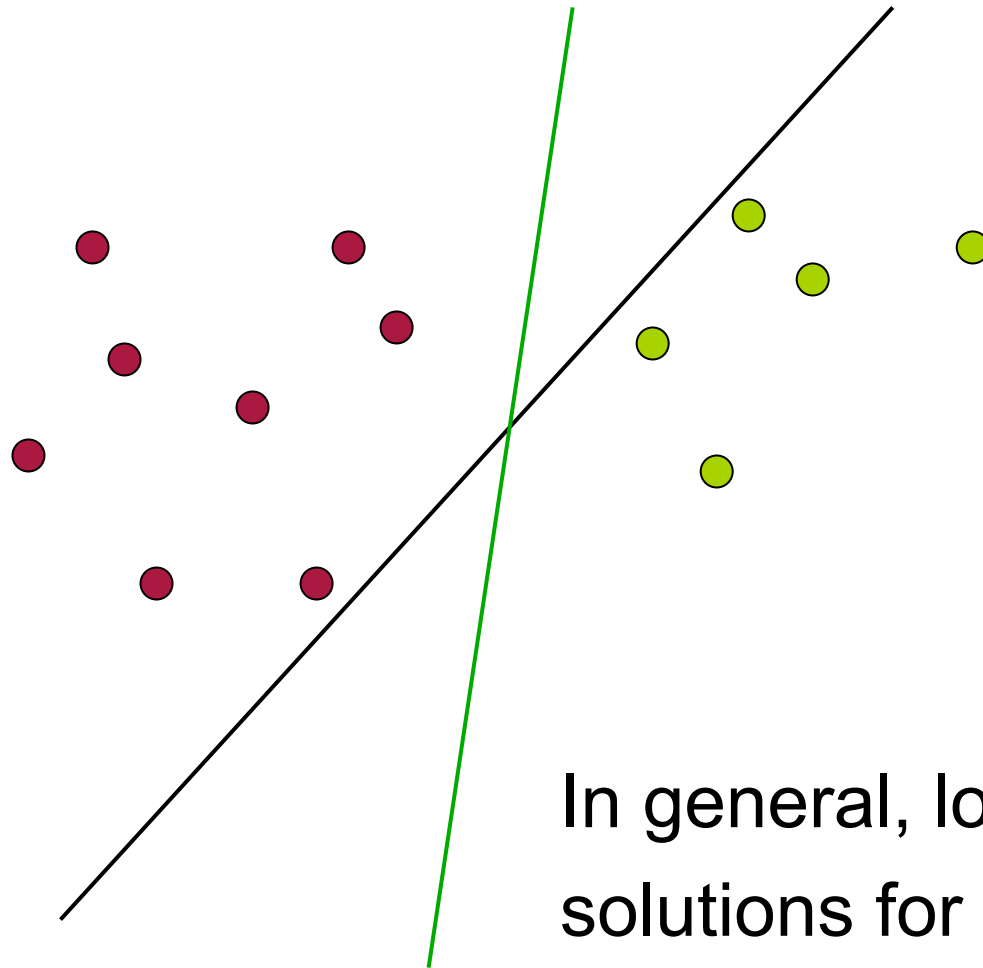


$$\sum_i w_i x_i = b$$

Is b positive or negative in this example?

What is the geometric interpretation of $\sum w_i x_i$ if w has unit length?

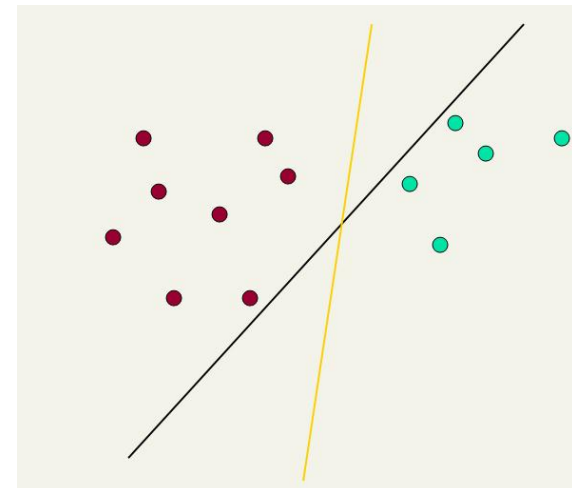
Which Hyperplane?



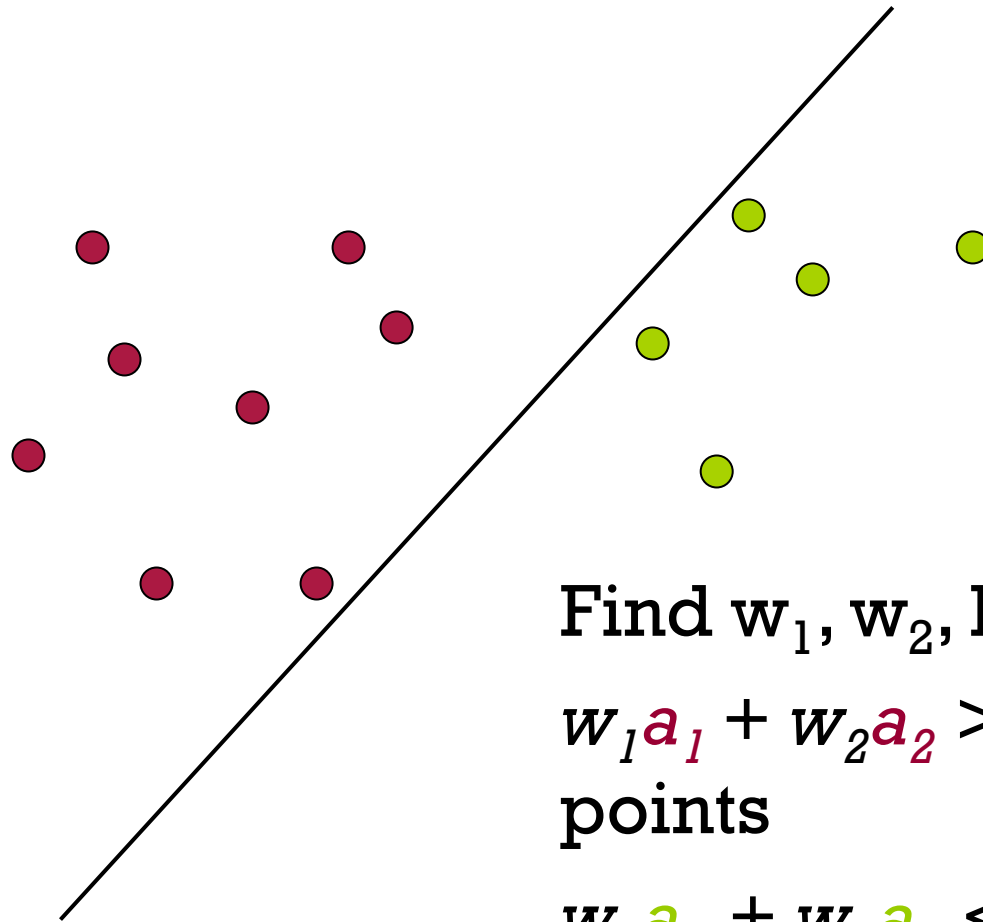
In general, lots of possible solutions for w_1, w_2, b

Which Hyperplane?

- Lots of possible solutions for w_1, w_2, b
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
 - E.g., perceptron
- Most methods find an optimal separating hyperplane
- Which points should influence optimality?
 - **All points**
 - Linear regression
 - Naïve Bayes
 - Only “**difficult points**” close to decision boundary
 - Support vector machines.



Linear programming / Perceptron



Find w_1, w_2, b , such that

$w_1 a_1 + w_2 a_2 > b$ for red points

$w_1 a_1 + w_2 a_2 < b$ for green points

Linear Programming

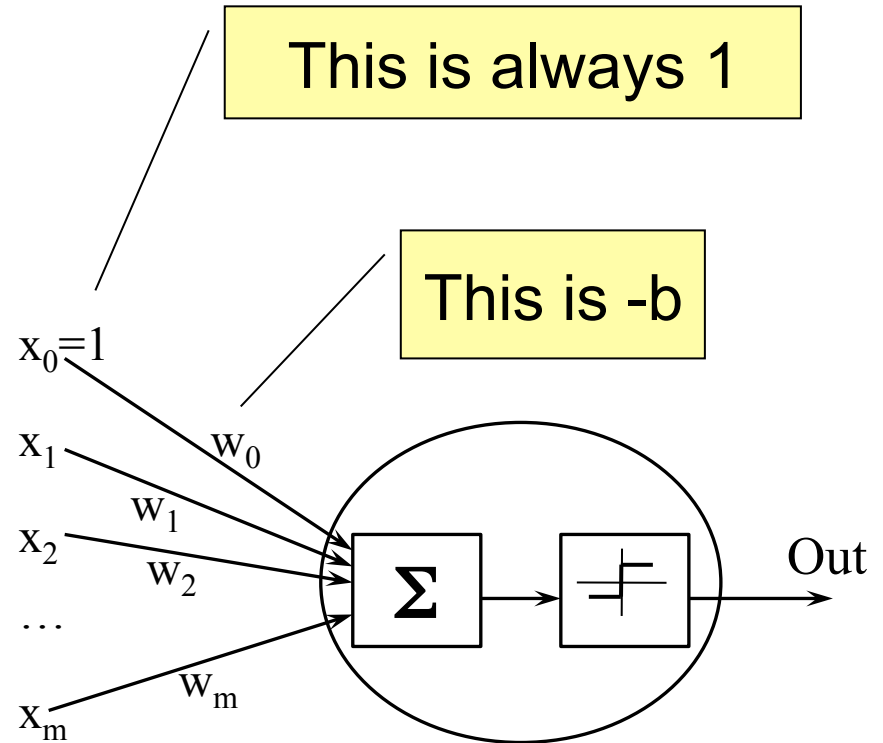
- Linear programming is a technique for the **optimization** of a linear objective function, subject to **linear equality** and **linear inequality constraints**
 - **Maximize $\mathbf{c}^T \mathbf{w}$** (\mathbf{c} and \mathbf{w} are n dimensional vectors)
 - **Subject to $\mathbf{A}\mathbf{w} \leq \mathbf{b}$** (\mathbf{A} is a $m \times n$ matrix, \mathbf{b} is a m dimensional vector)
- Example from previous slide
 - \mathbf{c}^T is not defined (chose what you want)
 - \mathbf{A}_{ij} is the matrix defined in this way:
 - A row (A_{i1}, A_{i2}) for each green point (A_{i1}, A_{i2}) , since we want $A_{i1}w_1 + A_{i2}w_2 \leq b$ ($b_i = b$)
 - A row $(-A_{j1}, -A_{j2})$ for each red point (A_{j1}, A_{j2}) , since we want $A_{j1}w_1 + A_{j2}w_2 \geq b$ ($b_j = -b$)

Perceptron

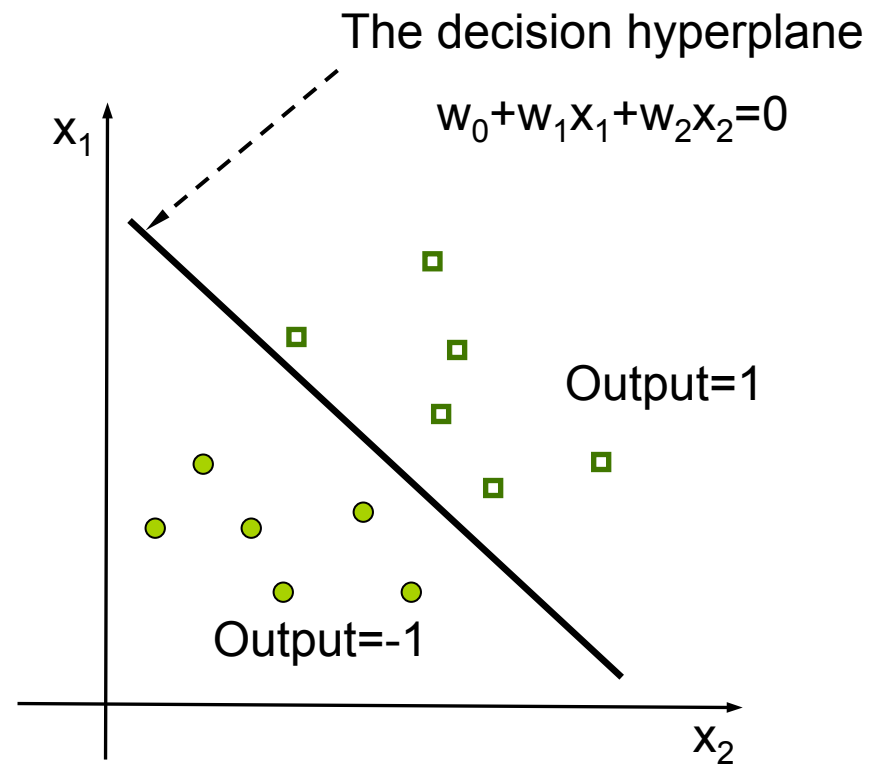
- A perceptron is the simplest type of Artificial Neural Network
- Use the hard-limit activation function
- For an instance \mathbf{x} , the perceptron output is:

$$Out = \text{sign}(\text{Net}(w, x)) = \text{sign}\left(\sum_{j=0}^m w_j x_j\right)$$

- 1, if $\text{Net}(\mathbf{w}, \mathbf{x}) > 0$
- -1, otherwise



Perceptron – Illustration



Perceptron – Learning

- Given a training set $D = \{(\mathbf{x}, d)\}$
 - \mathbf{x} is the input vector
 - d is the desired output value (i.e., -1 or 1)
- The perceptron learning is to determine a weight vector that makes the perceptron produce the correct output (-1 or 1) **for every training instance**
- If a training instance \mathbf{x} is correctly classified, then no (weight) update is needed
- If $d=1$ but the perceptron outputs -1 (i.e., $\text{Out}=-1$), then the weight \mathbf{w} should be updated so that $\text{Net}(\mathbf{w}, \mathbf{x})$ is increased
- If $d=-1$ but the perceptron outputs 1 (i.e., $\text{Out}=1$), then the weight \mathbf{w} should be updated so that $\text{Net}(\mathbf{w}, \mathbf{x})$ is decreased.

Perceptron_incremental(D, η)

Initialize \mathbf{w} ($w_i \leftarrow$ an initial (small) random value)

do

for each training instance $(\mathbf{x}, d) \in D$

 Compute the real output value Out

 if ($Out \neq d$)

$$\mathbf{w} \leftarrow \mathbf{w} + \eta (d - Out) \mathbf{x}$$

 end for

until all the training instances in D are correctly classified

return \mathbf{w}

You can check that if $Out < d$ then with the new weights $w^T x$ is larger than before

If the data are linearly separable!

Linear classifier: Example

- Class: "interest" (as in interest rate)
- Example features of a linear classifier

w_i	t_i	w_i	t_i
• 0.70	prime	• -0.71	dlrs
• 0.67	rate	• -0.35	world
• 0.63	interest	• -0.33	sees
• 0.60	rates	• -0.25	year
• 0.46	discount	• -0.24	group
• 0.43	bundesbank	• -0.24	dlr

- To classify, find dot product of feature vector and weights.

Linear Classifiers

- Many common text classifiers are linear classifiers:
 - Naïve Bayes
 - Perceptron
 - Rocchio
 - Logistic regression
 - Support vector machines (with linear kernel)
 - Linear regression
- Despite this similarity, noticeable performance differences
 - For separable problems, there is an infinite number of separating hyperplanes. Which one do you choose?
 - What to do for non-separable problems?
 - Different training methods pick different hyperplanes
- Classifiers more powerful than linear often don't perform better on text problems. Why?

Naive Bayes is a linear classifier

- Two-class Naive Bayes, we compute:

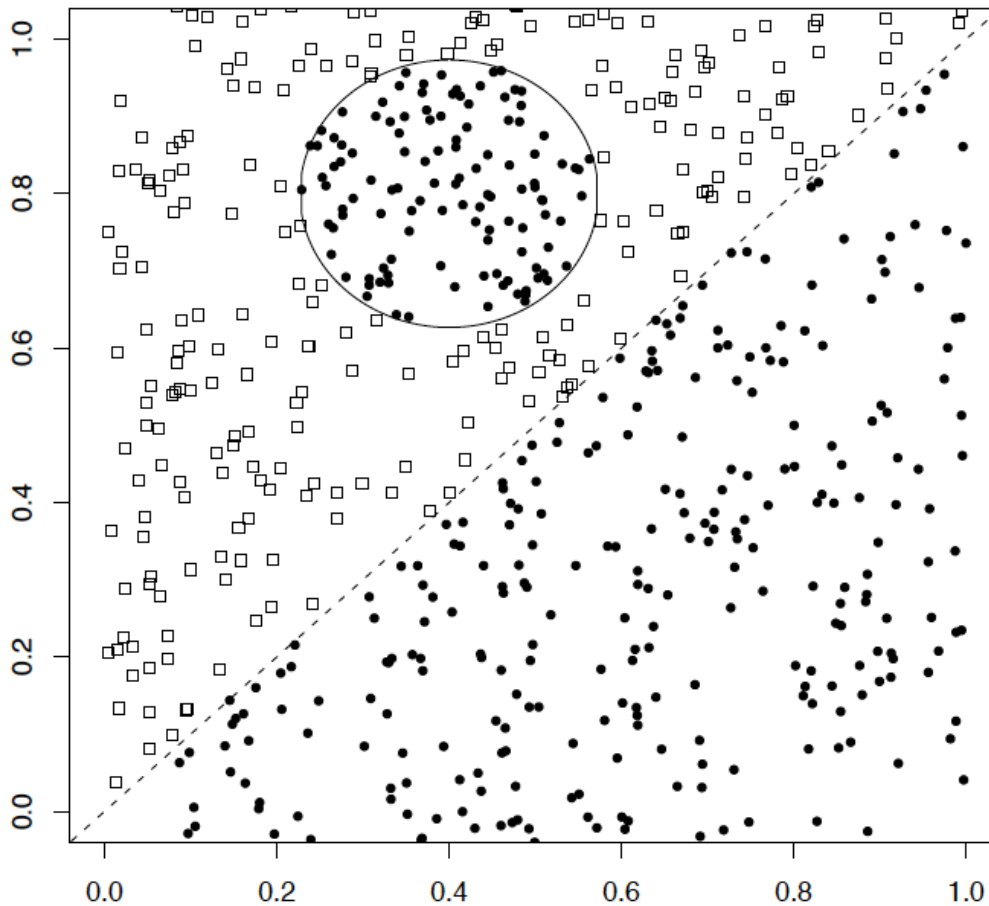
$$\log \frac{P(C | d)}{P(\bar{C} | d)} = \log \frac{P(C)}{P(\bar{C})} + \sum_{w \in d} \log \frac{P(w | C)}{P(w | \bar{C})}$$

- Decide class C if the odds is greater than 1, i.e., if the log odds is greater than 0.
- So decision boundary is hyperplane:

$$\alpha + \sum_{w \in V} \beta_w \times n_w = 0 \quad \text{where } \alpha = \log \frac{P(C)}{P(\bar{C})};$$

$$\beta_w = \log \frac{P(w | C)}{P(w | \bar{C})}; \quad n_w = \# \text{ of occurrences of } w \text{ in } d$$

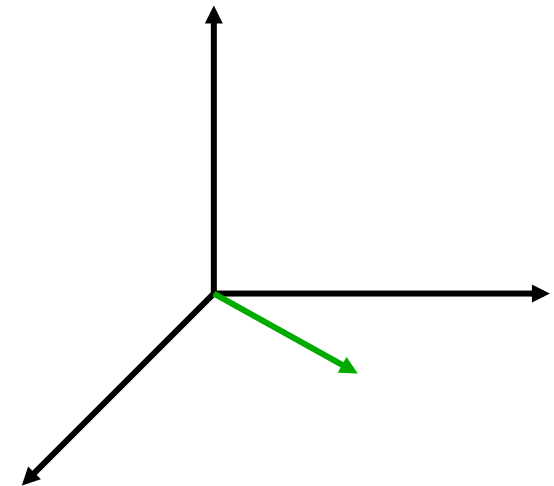
A nonlinear problem



- A linear classifier like Naïve Bayes does badly on this task
- kNN will do very well (assuming enough training data)

High Dimensional Data

- ❑ Pictures like the one at right are absolutely misleading!
- ❑ Documents are zero along almost all axes
- ❑ Most document pairs are very far apart (i.e., not strictly orthogonal, but only share very common words and a few scattered others)
- ❑ In classification terms: often document sets are separable, for most any classification
- ❑ This explain why linear classifiers are quite successful in this domain.



More Than Two Classes

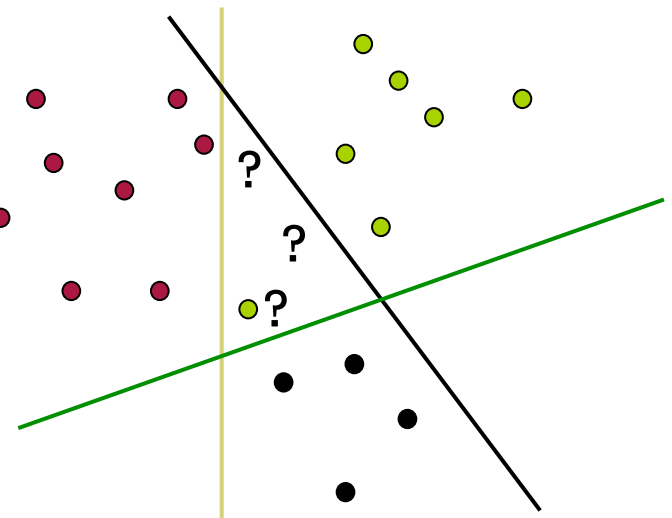
- **Any-of or multivalued classification**
 - Classes are independent of each other
 - A document can belong to 0, 1, or >1 classes
 - Decompose into n binary problems
 - Quite common for documents
- **One-of or multinomial or polytomous classification**
 - Classes are mutually exclusive
 - Each document belongs to exactly one class
 - E.g., digit recognition is polytomous classification
 - Digits are mutually exclusive.

Set of Binary Classifiers: Any of

- **Build a separator** between **each class and its complementary set** (docs from all other classes)
- Given **test doc**, evaluate it for **membership in each class independently**
- Though **maybe** you could do better by considering dependencies between categories.

Set of Binary Classifiers: One of

- Build a separator between each class and its complementary set (docs from all other classes)
- Given test doc, evaluate it for membership in each class
- Assign document to class with:
 - maximum score
 - maximum confidence
 - maximum probability
- Why different from **multiclass/any of** classification?



Using Rocchio for text classification

- **Relevance feedback** methods can be adapted for **text categorization**
 - As noted before, relevance feedback can be viewed as 2-class classification
 - Relevant vs. non-relevant documents
- Use standard TF/IDF weighted vectors to represent text documents
- For **training** documents in each category, compute a prototype vector by summing the vectors of the training documents in the category
 - Prototype = **centroid** of members of class
- Assign **test** documents to the category with the closest prototype vector based on cosine similarity.

Rocchio example

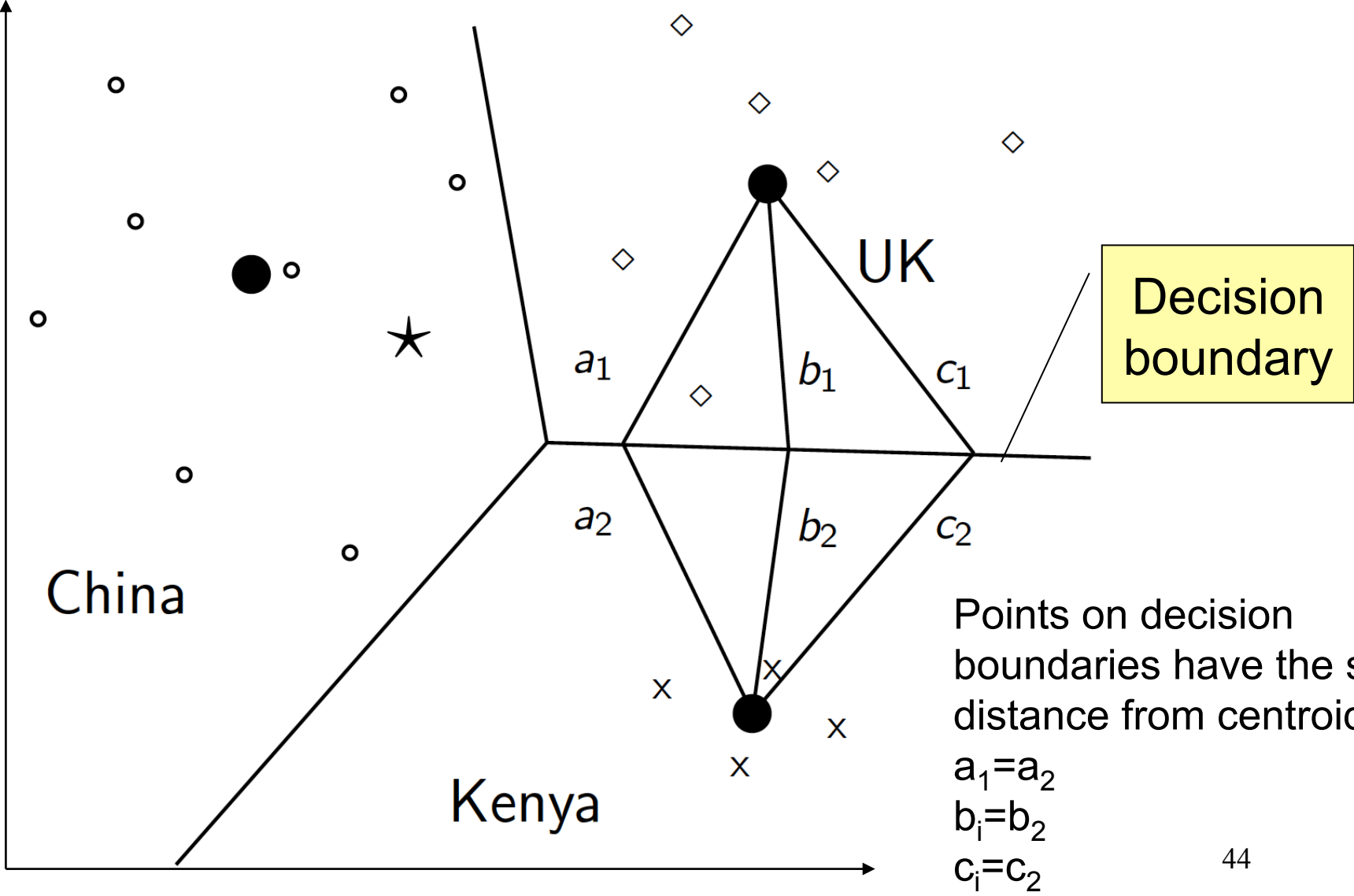
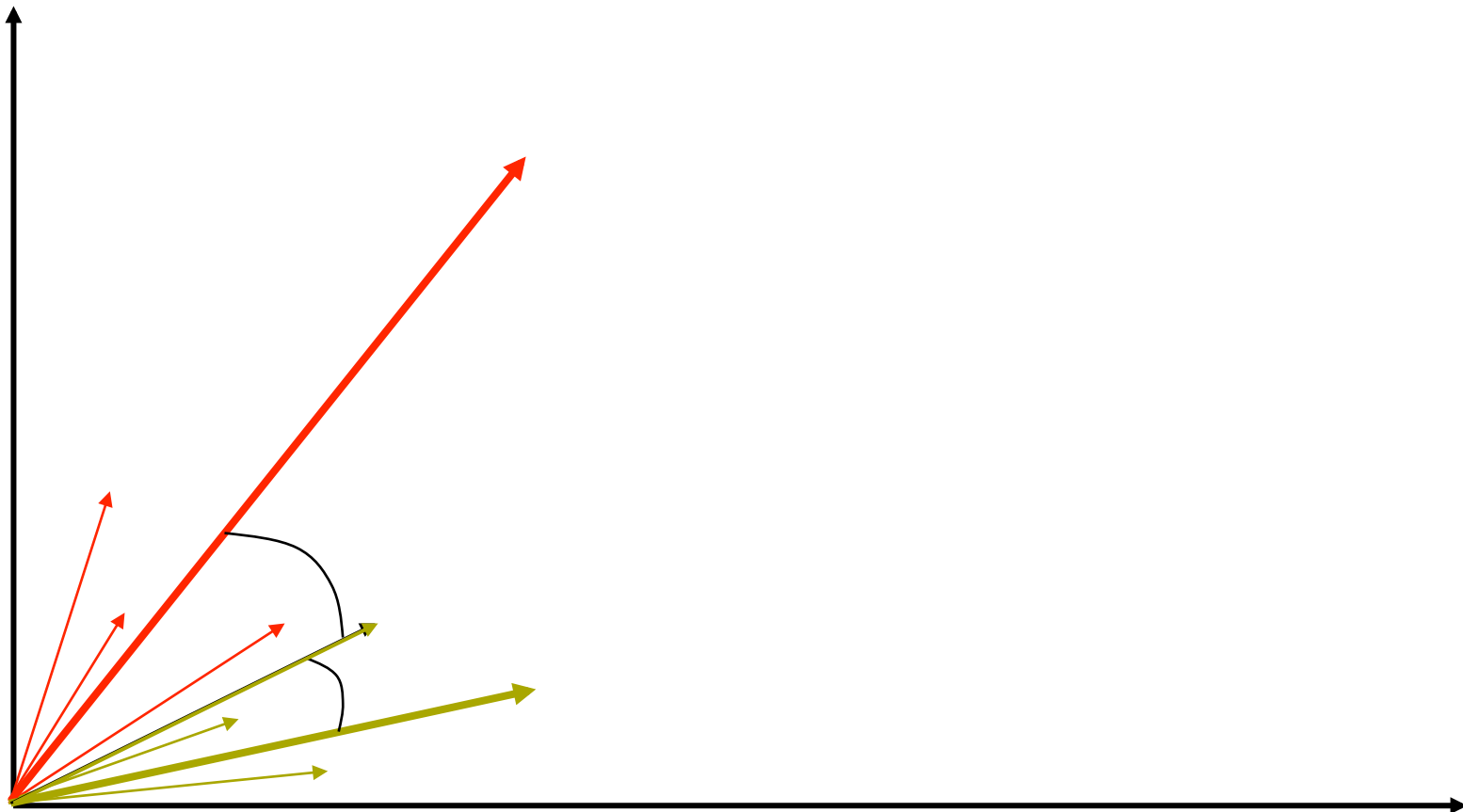


Illustration of Rocchio Text Categorization



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.*

Train and Test: Rocchio

```
TRAINROCCHIO( $\mathbb{C}, \mathbb{D}$ )  
1  for each  $c_j \in \mathbb{C}$   
2  do  $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$   
3      $\vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$   
4  return  $\{\vec{\mu}_1, \dots, \vec{\mu}_J\}$ 
```

```
APPLYROCCHIO( $\{\vec{\mu}_1, \dots, \vec{\mu}_J\}, d$ )  
1  return  $\arg \min_j |\vec{\mu}_j - \vec{v}(d)|$ 
```

Class whose
prototype has
minimal Euclidean
distance from test
document

- ❑ One can use also the cosine similarity – how you must change the algorithm?
- ❑ If there are only two classes the decision line is a simple hyperplane ... see later

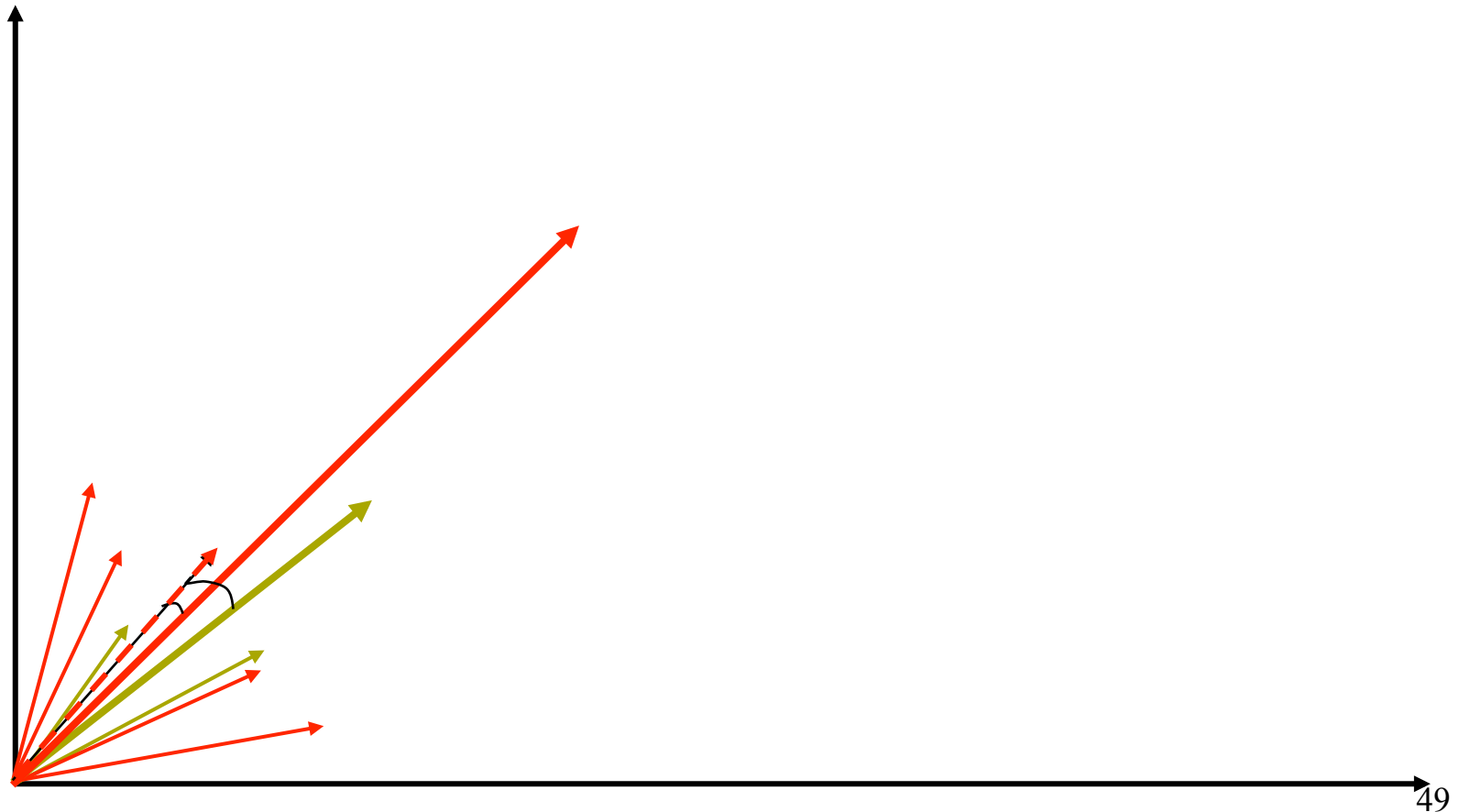
Rocchio Properties

- ❑ Forms a simple generalization of the examples in each class (a *prototype*)
- ❑ The **decision boundary** between two classes is the set of points with equal distance from the two corresponding centroids
- ❑ Classification is based on similarity to class prototypes
- ❑ Does not guarantee classifications are consistent with the given training data.

Why not?
Is that bad?

Rocchio Anomaly

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

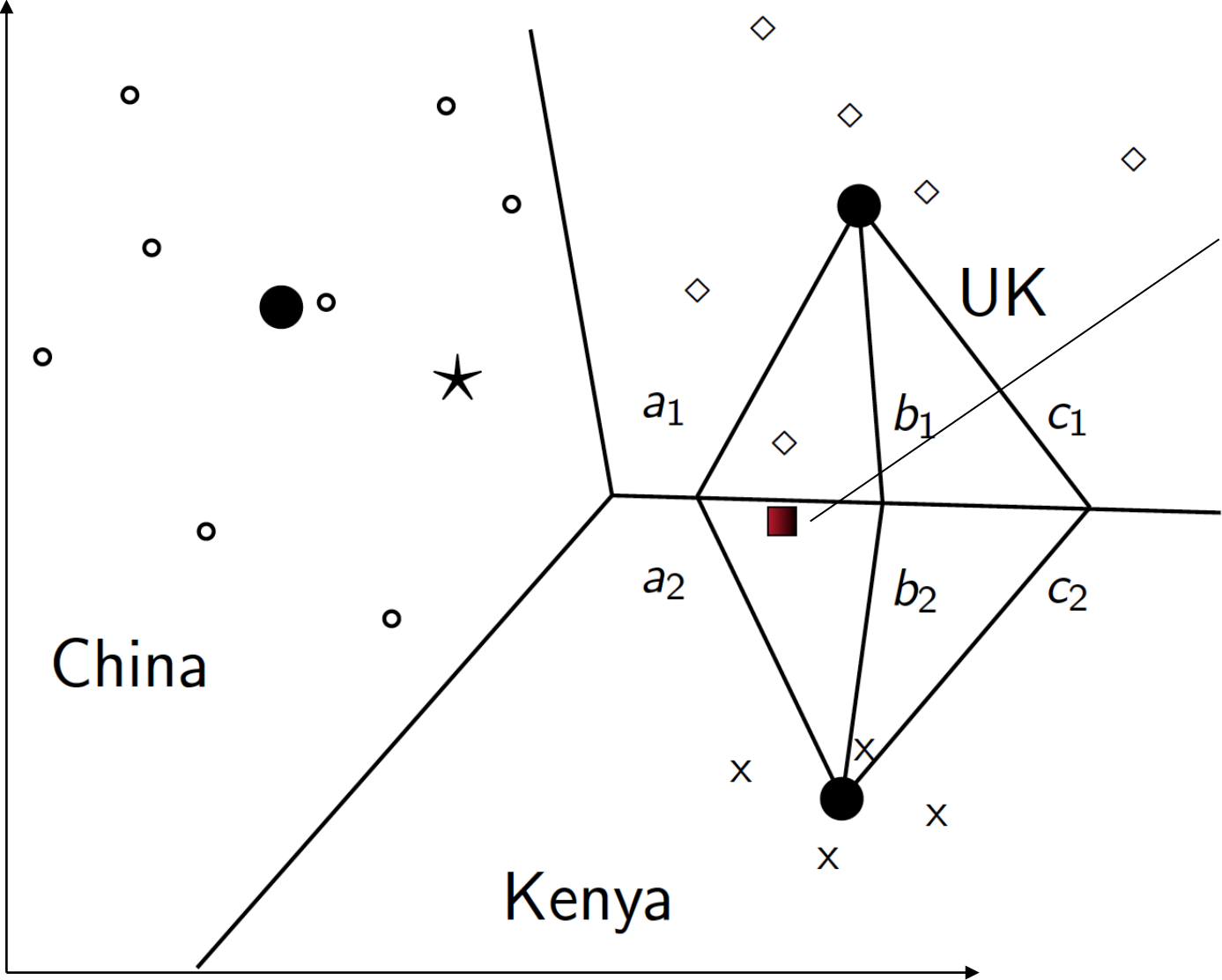


3 Nearest Neighbor Comparison

- Nearest Neighbor tends to handle polymorphic categories better.

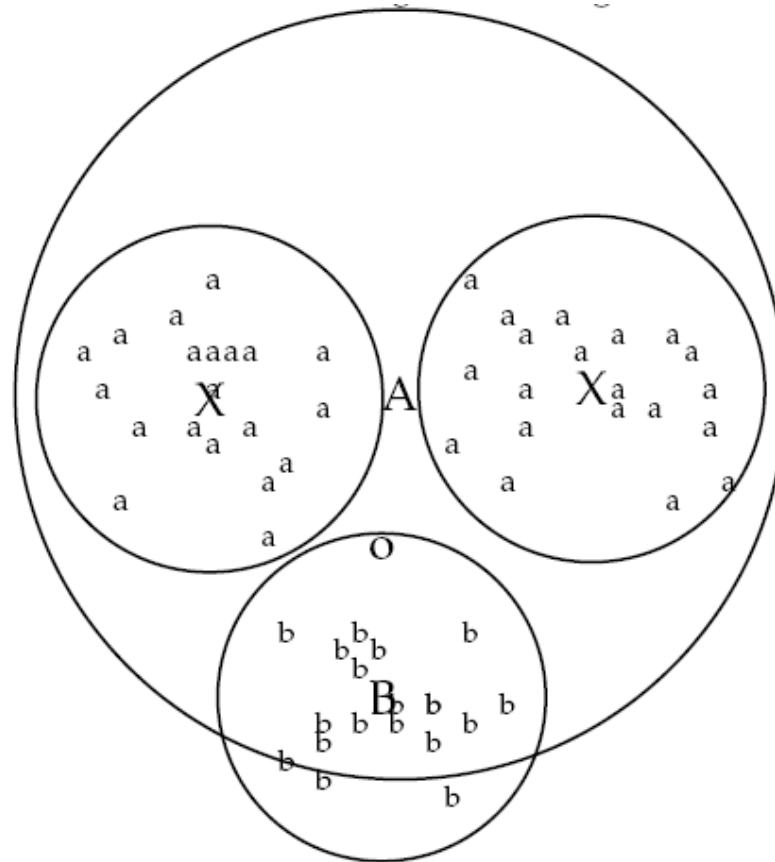


Rocchio example II



How would be classified a point here?
Is it a good idea?

Rocchio: Multimodal classes



► **Figure 14.5** The multimodal class “a” consists of two different clusters (small upper circles centered on X’s). Rocchio classification will misclassify “o” as “a” because it is closer to the centroid A of the “a” class than to the centroid B of the “b” class.

Two-class Rocchio as a linear classifier

- Line or hyperplane defined by:

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

Vector orthogonal to the hyperplane

Distance from the origin

- For Rocchio, set:

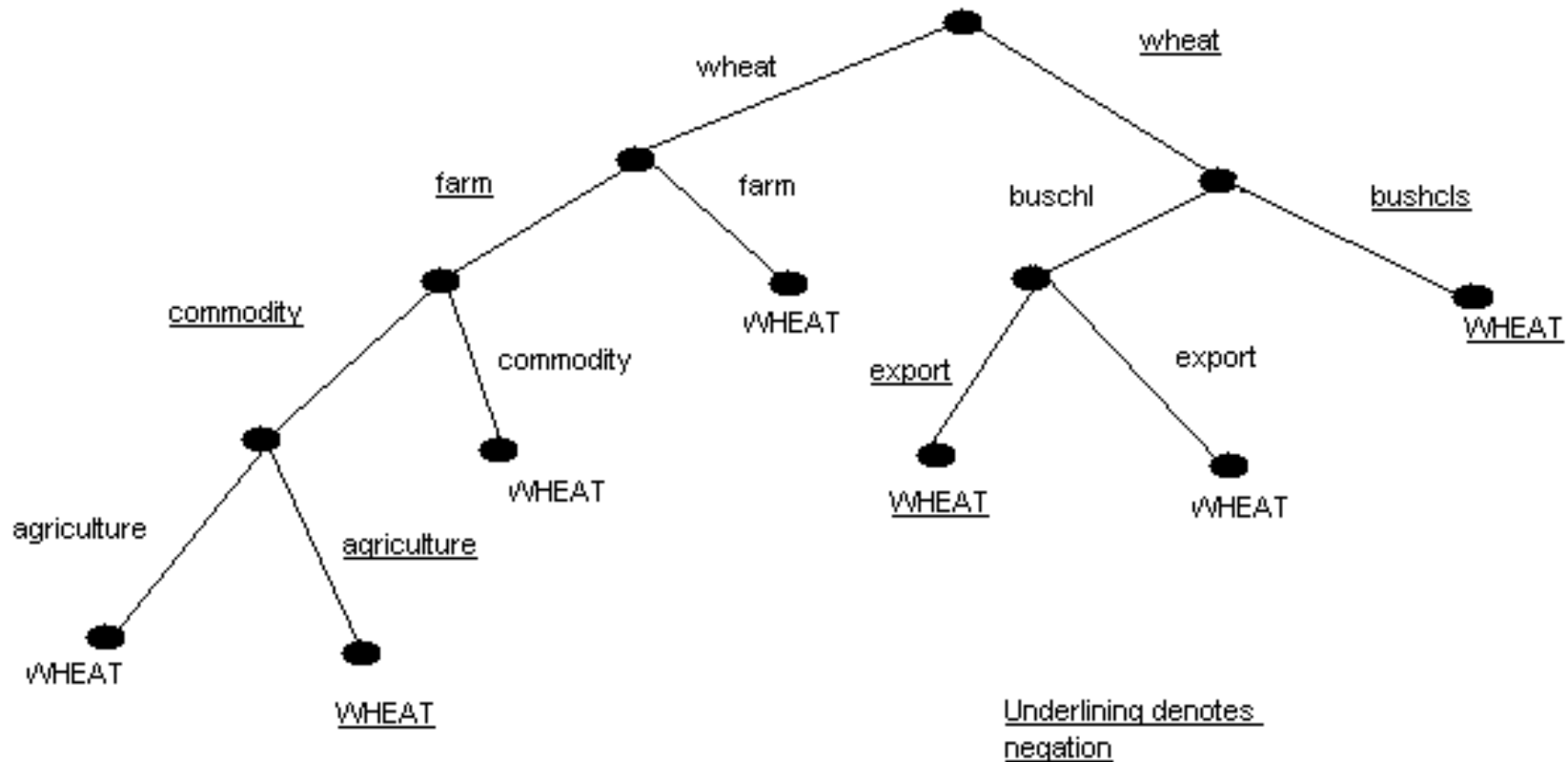
$$\vec{w} = \vec{\mu}(c_1) - \vec{\mu}(c_2)$$

$$b = 0.5 \times (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$$

Decision Tree Classification

- Tree with **internal nodes** labeled by **terms**
- **Branches** are **labeled by tests** on the **weight** that the term has (e.g. present/absent)
- **Leaves** are labeled by **categories**
- Classifier **categorizes** document by **descending tree following tests to leaf**
- The label of the leaf node is then assigned to the document
- Most decision trees are binary trees (never disadvantageous; may require extra internal nodes)
- DT make good use of a few high-leverage features.

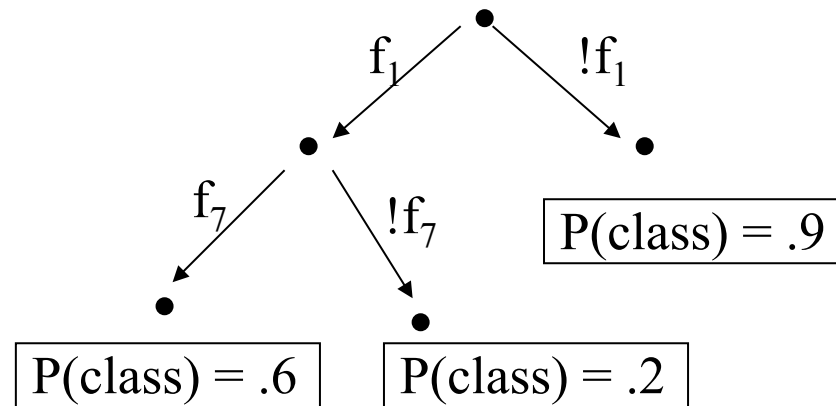
DT Categorization: Example



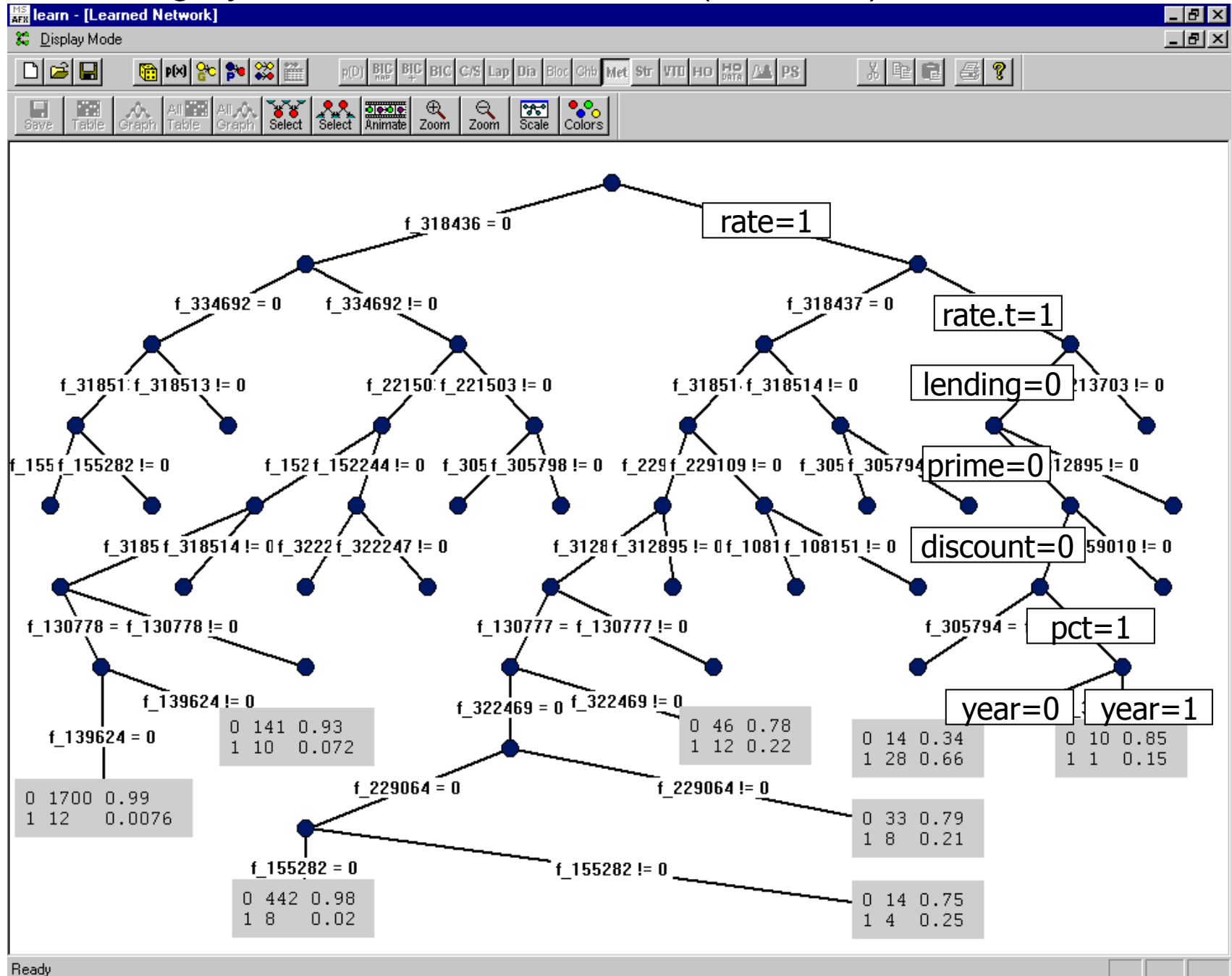
Geometric interpretation of DT?

Decision Tree Learning

- Learn a sequence of tests on features, typically using top-down, greedy search
 - At each stage choose the unused feature with highest **Information Gain**
 - That is, the split that produces the highest reduction of the entropy in the data
- Binary (yes/no) or continuous decisions



Category: "interest" – Dumais et al. (Microsoft) Decision Tree

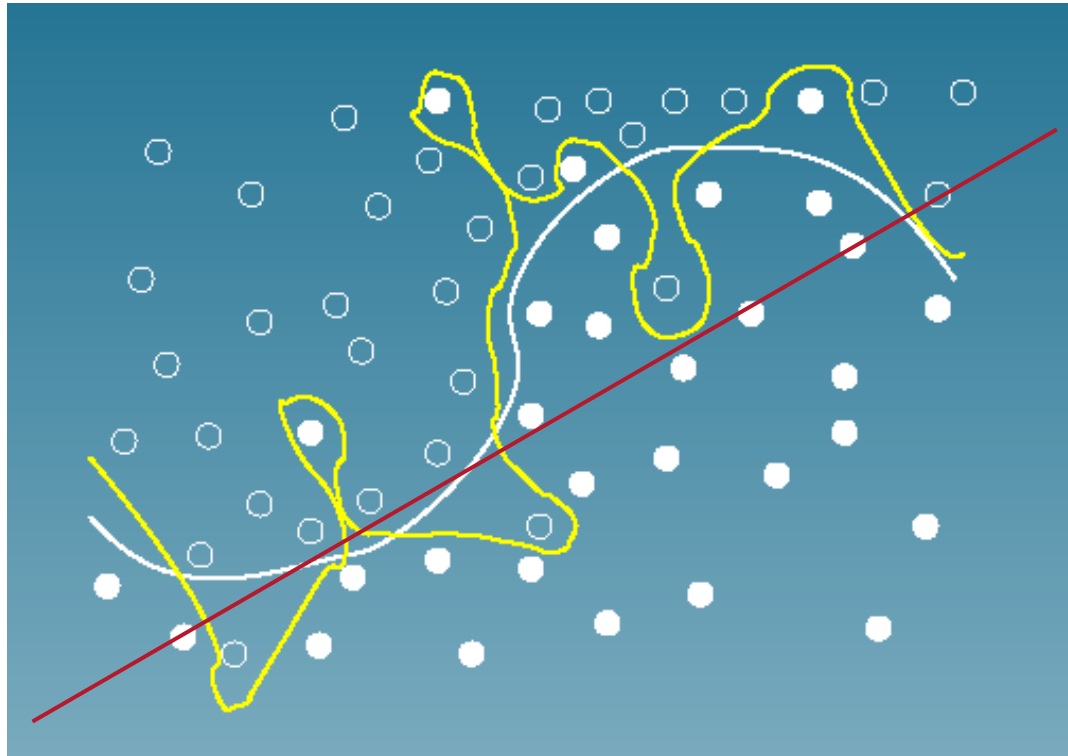


kNN vs. Naive Bayes

- Bias/Variance tradeoff
 - Variance \approx Capacity
- **kNN** has **high variance** and **low bias**
 - **Infinite memory to adapt to training data**
- **NB** has **low variance** and **high bias**
 - Decision surface has to be linear (hyperplane – see later)
- Consider asking a botanist: **Is an object a tree?**
 - Case 1: too much capacity/variance, low bias
 - Botanist who memorizes all the trees he has seen
 - Will always say “no” to new object (e.g., different # of leaves)
 - Case 2: not enough capacity/variance, high bias
 - Lazy botanist
 - Says “yes” if the object is green
 - You want the middle ground

(Example due to C. Burges)

Bias vs. variance: Choosing the correct model capacity



Bias-Variance decomposition of MSE

- Assume that our goal is to find a classifier γ s.t. the predicted probability of d to be in class c , $\gamma(d)$, is as close as possible to the true probability $P(c|d)$
 - $MSE(\gamma) = E_d[\gamma(d) - P(c|d)]^2$
- A classifier γ is **optimal** if it minimizes $MSE(\gamma)$
- Imagine now that Γ is a learning method that produces a classifier γ for each training set D
- Γ is a good method if averaged over all D the error of Γ_D – the classifier built using D – is minimal
 - $Learning-error(\Gamma) = E_D[MSE(\Gamma_D)]$

Bias-Variance decomposition

$P(c|d)$
predicted by Γ_D

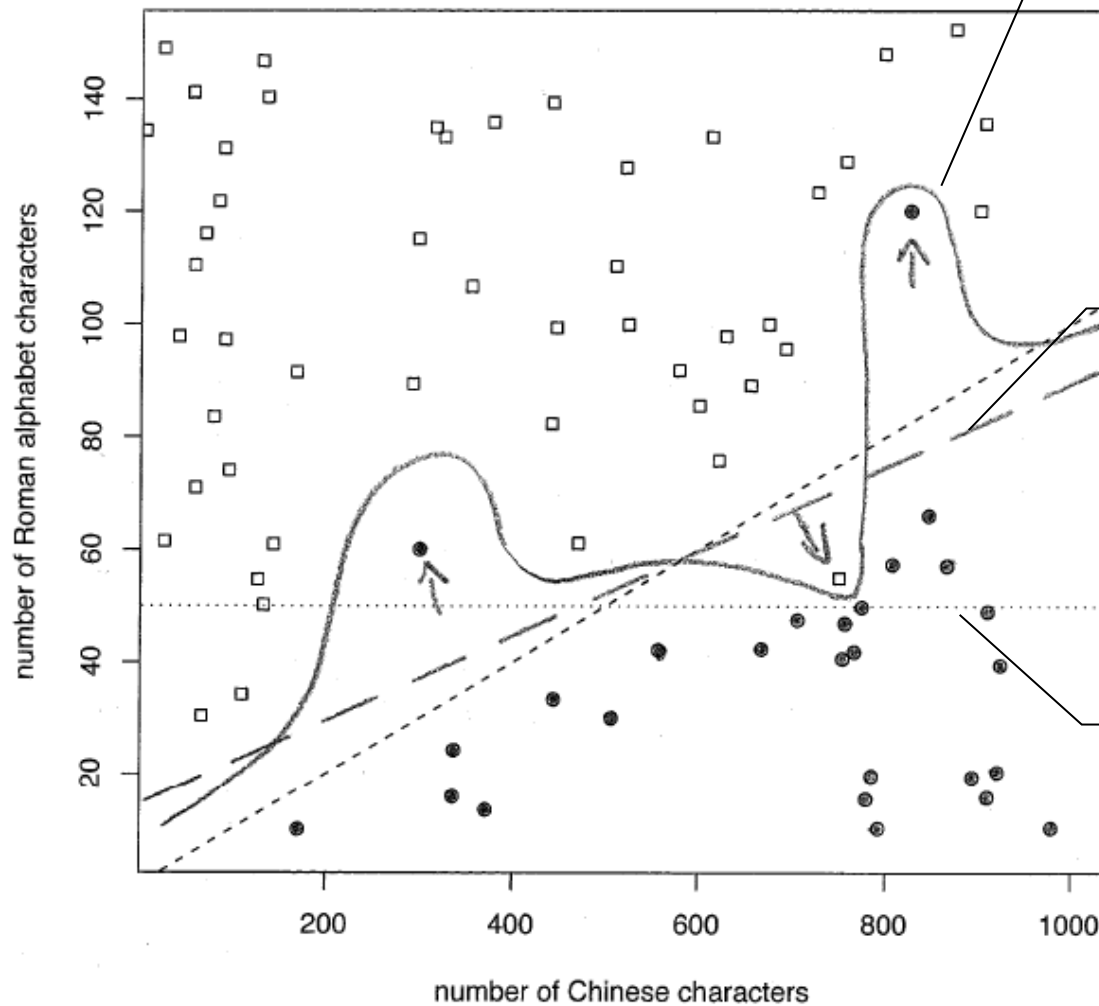
- Learning-error(Γ) = $E_D[\text{MSE}(\Gamma_D)]$
= $E_D E_d[\Gamma_D(d) - P(c|d)]^2$
= $E_d[\text{bias}(\Gamma, d) + \text{variance}(\Gamma, d)]$
- *Math derivation is shown in the book ...*
 - **Bias**(Γ, d) = $[P(c|d) - E_D \Gamma_D(d)]^2$
 - **Variance**(Γ, d) = $E_D[\Gamma_D(d) - E_D \Gamma_D(d)]^2$
- Bias (for a document d) is **small** if the average, over different D , of the predicted probability is close to the true probability (KNN)
- Bias is **large** if on average the classifiers Γ_D are predicting a wrong $P(c|d)$ (Linear)

Bias-Variance decomposition

- **Bias**(Γ, \mathbf{d}) = $[\mathbf{P}(\mathbf{c}|\mathbf{d}) - \mathbf{E}_D \Gamma_D(\mathbf{d})]^2$
- **Variance**(Γ, \mathbf{d}) = $\mathbf{E}_D [\Gamma_D(\mathbf{d}) - \mathbf{E}_D \Gamma_D(\mathbf{d})]^2$

- Variance is **low** if $\Gamma_D(\mathbf{d})$ is rather stable, by varying D , and is close to the average $\mathbf{E}_D \Gamma_D(\mathbf{d})$ (linear)
- Variance is **high** if the prediction is strongly influenced by the training set D (KNN).

Example



A "fit training set perfectly" model:
low bias – high
variance

A linear model:
medium bias –
low variance

A simple model
using only one
feature: high bias
– low variance

Discussion

- **Linear models** s.a. Rocchio and NB have **high bias** (for non linear problems) because they can only model one type of class boundary – a linear hyperplane
- We should choose a linear model if we know that the problem is **linearly separable**
- **Non linear models** as KNN have **low bias** – depending of the training set they can learn complex concepts
- **Linear models** have **low variance** because most randomly chosen training set will produce the same model (stable)
- **Non linear models** as KNN can model any decision boundary but are sensitive to noise (will fit them)
- **High variance** models are **prone to overfitting** the training data
 - *The goal of classification is to correctly predict the instances not yet considered!*

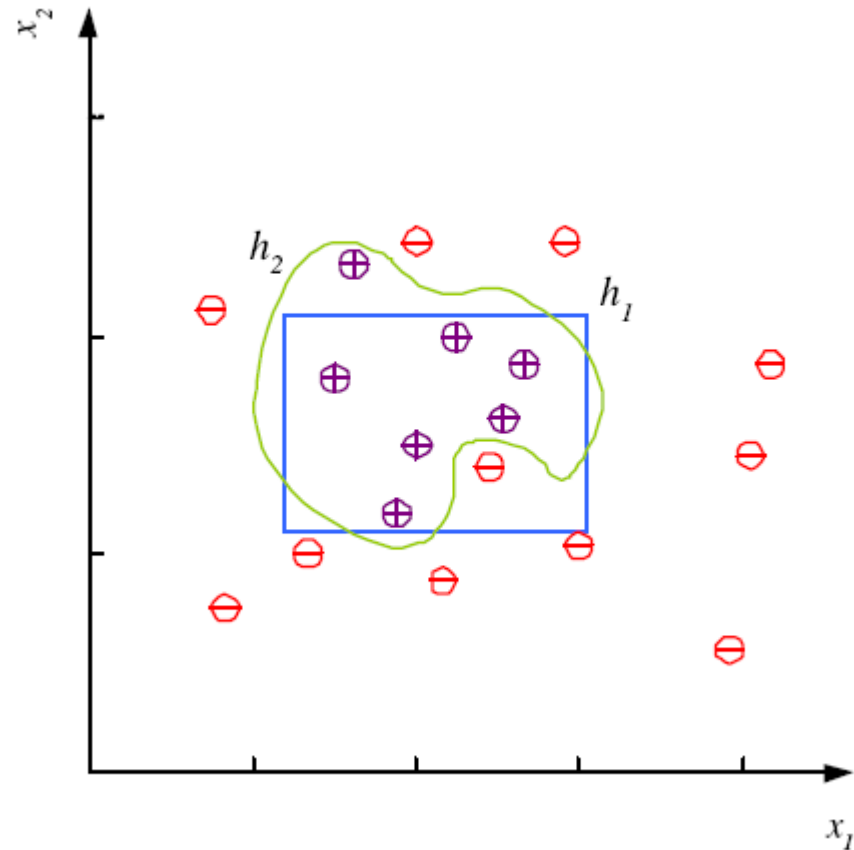
Bias Variance Tradeoff

- Learning-error(Γ) = $E_d[\text{bias}(\Gamma, d) + \text{variance}(\Gamma, d)]$
- If we want to minimize the error we can either try to reduce the bias or the variance
- **In general both of them cannot be reduced**
- Given an application we should evaluate the respective merits of the possible methods
- And choose according to the application goals.

Noise and Model Complexity

Use the simpler model because

- Simpler to use
(lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain
(more interpretable)
- Generalizes better (lower variance - Occam's razor)

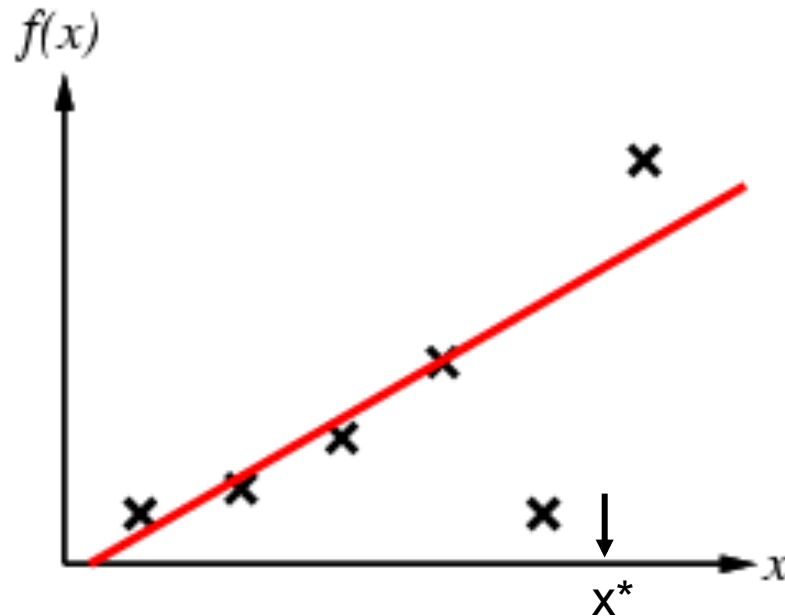


Model Selection & Generalization

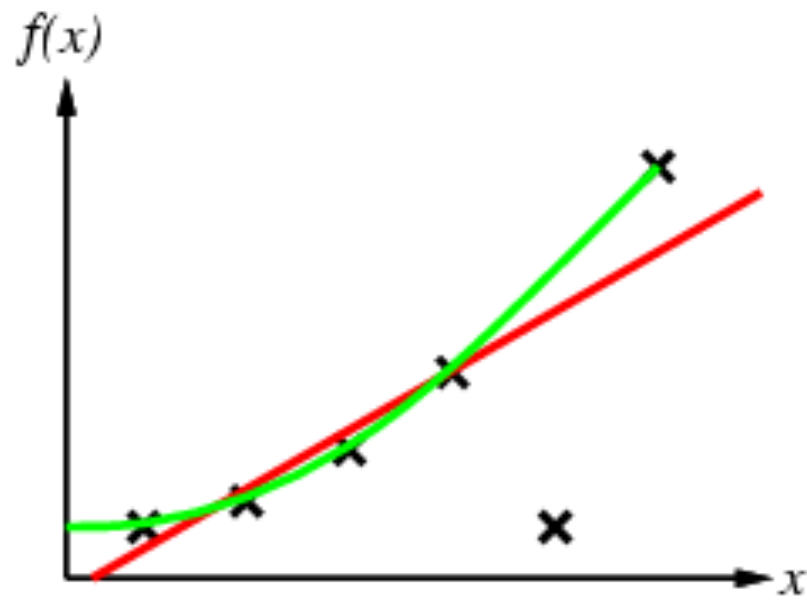
- Learning is an **ill-posed problem**
 - data is not sufficient to find a unique solution!
- The need for **inductive bias**, assumptions about H (the space all possible hypothesis)
- **Generalization**: How well a model performs on new data
- Overfitting: H more complex than C or f
- Underfitting: H less complex than C or f

Generalization

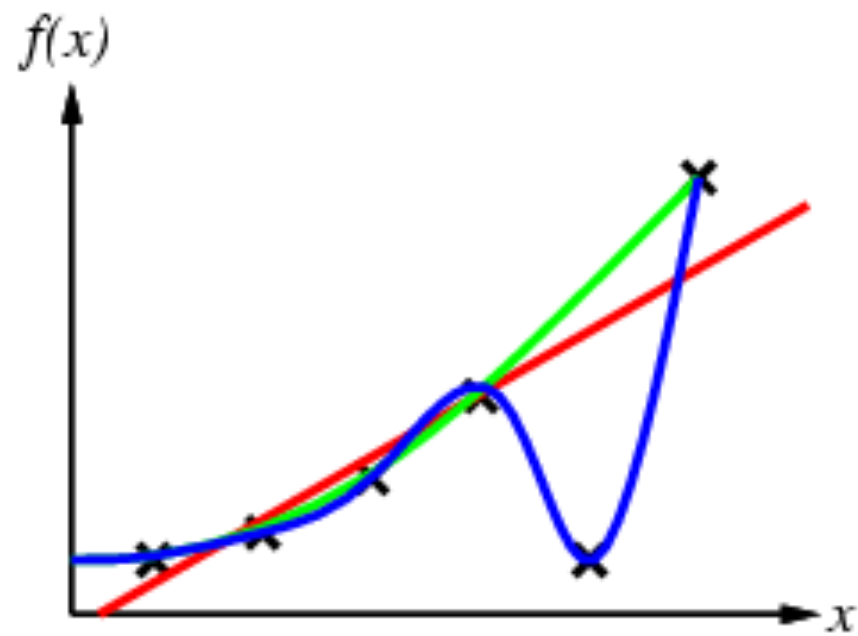
- ❑ Consider the following *regression* problem:
- ❑ Predict the real value on the y -axis from the real value on the x -axis.
- ❑ You are given 6 examples: $\{x_i, y_i\}$.
- ❑ What is the y -value for a new query point x^* ?



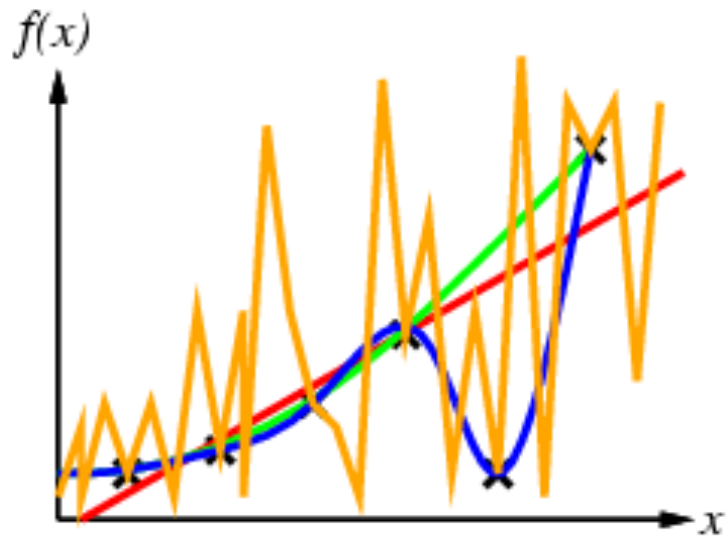
Generalization



Generalization

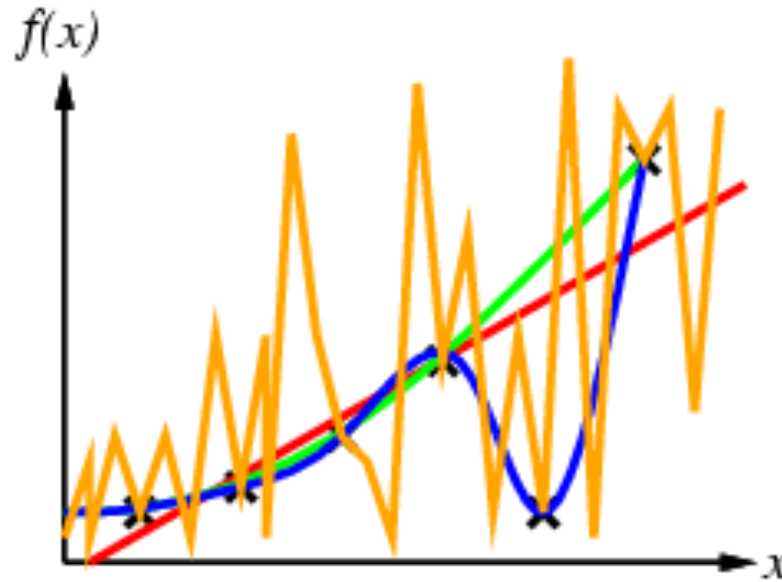


Generalization



which curve is best?

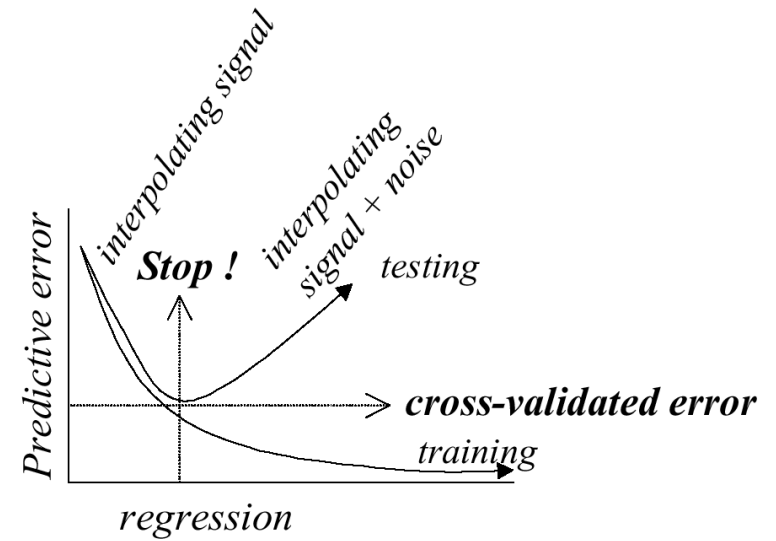
Generalization



- ❑ **Ockham's razor**: prefer the simplest hypothesis consistent with data
- ❑ Learning is concerned with accurate prediction of future data, *not* accurate prediction of training data.

Cross-validation

How do we ensure good generalization, i.e. avoid “over-fitting” on our particular data sample.



- ❑ You are ultimately interested in good performance on new (unseen) test data
- ❑ To estimate that, split off a (smallish) subset of the training data (called validation set)
- ❑ Train without validation data and “test” on validation data
- ❑ Repeat this over multiple splits of the data and average results
- ❑ Reasonable split: 90% train, 10% test, average over the 10 splits.

Summary: Representation of Text Categorization Attributes

- Representations of text are usually very high dimensional (one feature for each word)
- **High-bias** algorithms that prevent **overfitting** in high-dimensional space generally work best
- For most text categorization tasks, there are many relevant features and many irrelevant ones
- Methods that combine evidence from many or all features (e.g. naive Bayes, kNN, neural-nets) often tend to work better than ones that try to isolate just a few relevant features (standard decision-tree or rule induction)*
 - *Although the results are a bit more mixed than often thought.

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 problem?
 - How stable is the problem over time?
 - ❑ For an unstable problem, it's better to use a simple and robust classifier.

References

- IIR 14
- Tom Mitchell, Machine Learning. McGraw-Hill, 1997.
- Weka: A data mining software package that includes an implementation of many ML algorithms
- R. Duda, P. Hart, and D. Stork, Pattern Classification (2nd Edition). Wiley, 2000.