# CS60020: Foundations of Algorithm Design and Machine Learning

Sourangshu Bhattacharya

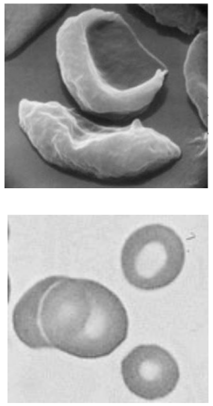
# Discrete and Continuous Labels

## Classification

X = Document

Sports
Science
News

Y = Topic



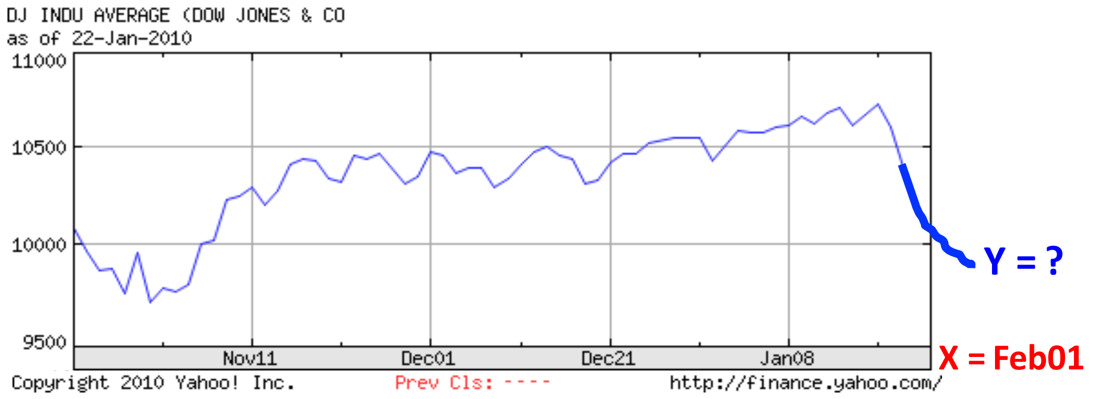
X = Cell Image

Anemic cell
Healthy cell

Y = Diagnosis

## Regression

Stock Market Prediction



## An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- Problem: to predict high-risk patients and discriminate them from low-risk patients.

## Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

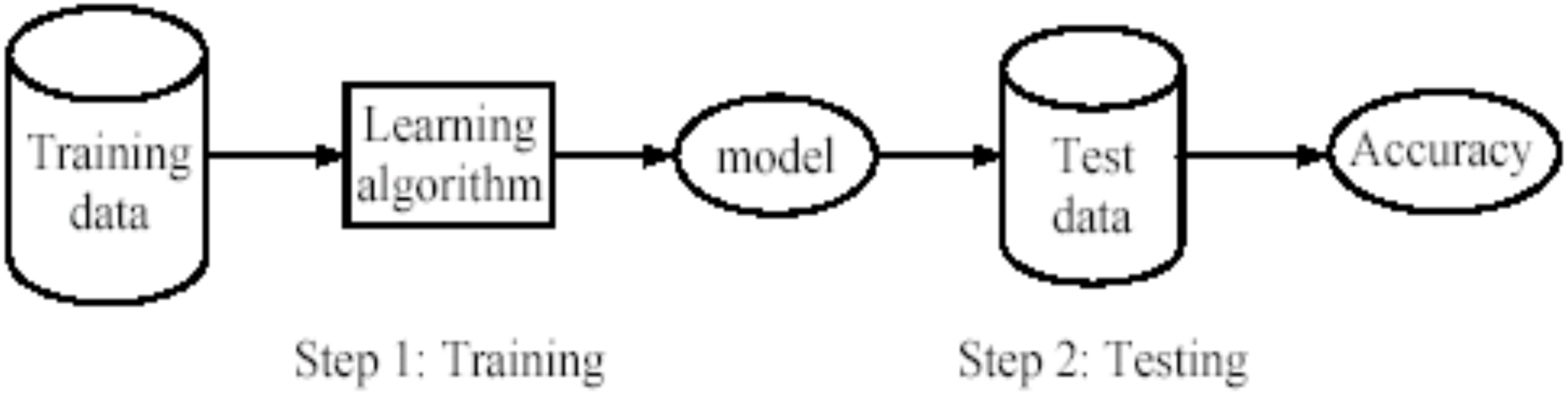
# The data and the goal

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: $A_1$, $A_2$, … $A_k$.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

# Supervised learning process: two steps

- Learning (training): Learn a model using the training data
- Testing: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# Least squares classification

- Binary classification.
- Each class is described by it's own linear model:
$$y(x) = w^T x + w_0$$
- Compactly written as:
$$\mathrm{y}(\boldsymbol{x}) = \boldsymbol{W}^T \boldsymbol{x}$$
- **W** is $[w\ w_0]$.
- $E_D(\boldsymbol{W}) = {}^1\!/_2\,(\boldsymbol{XW} - \boldsymbol{t})^T(\boldsymbol{XW} - \boldsymbol{t})$
- $n^{th}$ row of $\boldsymbol{X}$ is $x_n$**,** the $n^{th}$ datapoint.
- $\boldsymbol{t}$ is vector of +1, -1.

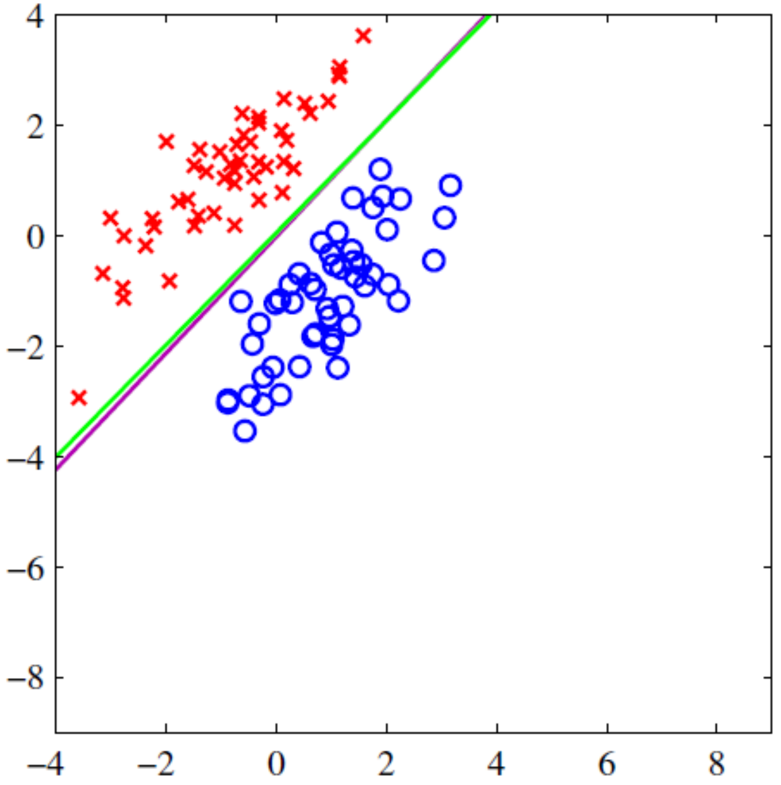
## Least squares classification

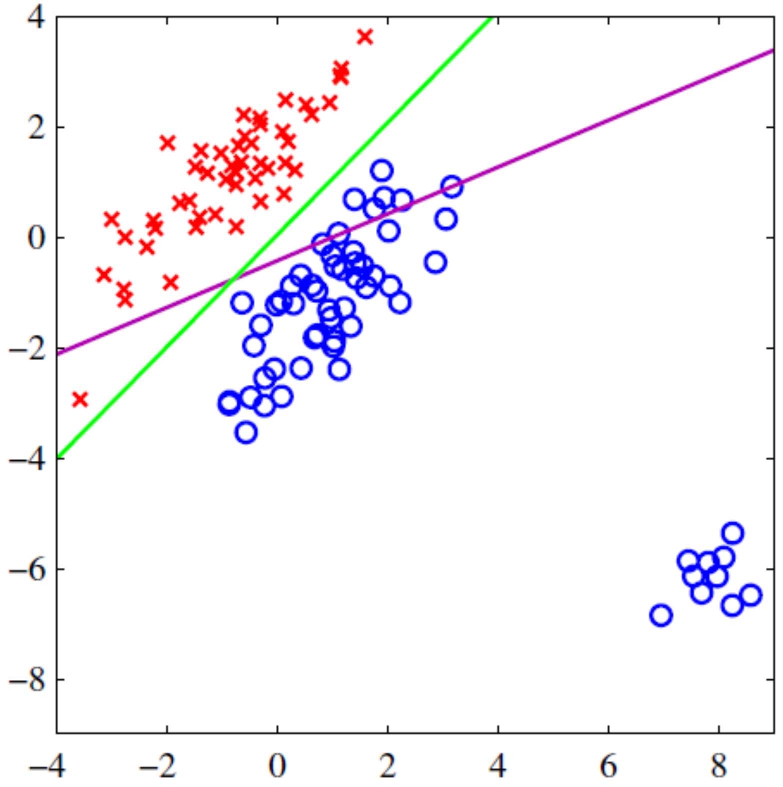
- Least squares $\boldsymbol{W}$ is:

$$\boldsymbol{W} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^T\boldsymbol{t}$$

- Problem is affected by outliers.

# Least squares classification





# Fisher's linear discriminant

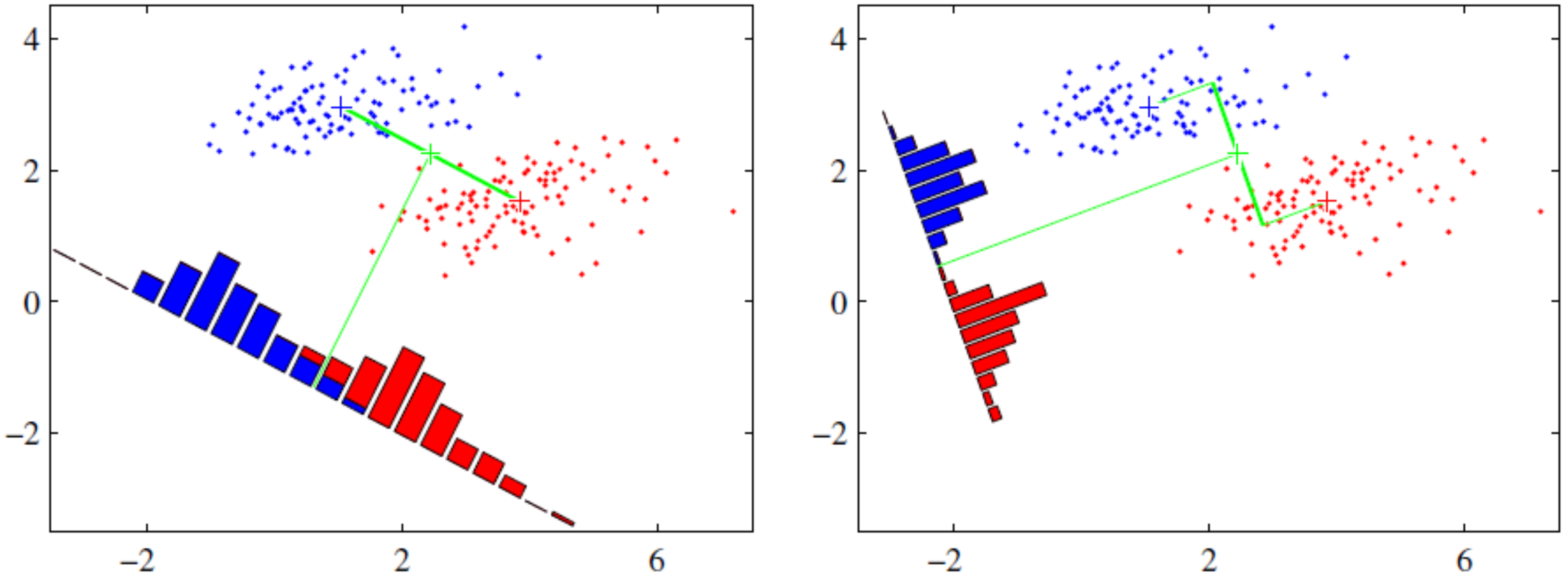
- Predictor: $y = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}$.
- If $y \succ \boldsymbol{w_0}$ predict $C_1$ else $C_2$.
- Training dataset has $N_1$points from $C_1$ and $N_2$ points from $C_2$.
- $\boldsymbol{m_1} = \frac{1}{N_1}\sum_{n \in \mathrm{C}_1} \boldsymbol{x_n}$ and $\boldsymbol{m_2} = \frac{1}{N_2}\sum_{n \in \mathrm{C}_2} \boldsymbol{x_n}$
- Maximize separation of projected means:

$$m_2 - m_1 = \boldsymbol{w}^{\boldsymbol{T}}(\boldsymbol{m_2} - \boldsymbol{m_1})$$

# Fisher's linear discriminant

- This measure can increase arbitrarily by increasing $||\boldsymbol{w}||$.
- Constrain: $||\boldsymbol{w}||^2 = 1$
- Lagrangian: $L(\boldsymbol{w}, \boldsymbol{\lambda}) = \boldsymbol{w}^{\boldsymbol{T}}(\boldsymbol{m}_{\boldsymbol{2}} - \boldsymbol{m}_{\boldsymbol{1}}) + \boldsymbol{\lambda}(||\boldsymbol{w}||^{\boldsymbol{2}} - \boldsymbol{1})$.
- Solution: $\boldsymbol{w} \propto (\boldsymbol{m}_{\boldsymbol{2}} - \boldsymbol{m}_{\boldsymbol{1}})$.

# Fisher linear discriminant



# Fisher’s linear discriminant

- Maximize separation between means while minimizing within class variance.
- Within class variance:

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- Objective:

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

# Fisher's linear Discriminant

- Same as:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S_B} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S_w} W}$$

- Between class variance:

$$\boldsymbol{S_B} = (\boldsymbol{m_2} - \boldsymbol{m_1})(\boldsymbol{m_2} - \boldsymbol{m_1})^{\boldsymbol{T}}$$

- Within class variance:

$$\boldsymbol{S_W} = \sum_{\boldsymbol{n} \in \boldsymbol{C_1}} (\boldsymbol{x_n} - \boldsymbol{m_1})(\boldsymbol{x_n} - \boldsymbol{m_1})^{\boldsymbol{T}} + \sum_{\boldsymbol{n} \in \boldsymbol{C_2}} (\boldsymbol{x_n} - \boldsymbol{m_2})(\boldsymbol{x_n} - \boldsymbol{m_2})^{\boldsymbol{T}}$$

# Fisher's linear discriminant

- Same as:

$$\max_{\boldsymbol{w}} \ \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}$$
$$s.t. \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w} = \mathbf{1}$$

- Solution given by generalized eigenvalue problem:

$$\boldsymbol{S}_B \boldsymbol{w} = \lambda \boldsymbol{S}_w \boldsymbol{w}$$

- Or

$$(\boldsymbol{S}_W)^{-1} \boldsymbol{S}_B \boldsymbol{w} = \lambda \boldsymbol{w}$$

- Solution:

$$\boldsymbol{w} \propto (\boldsymbol{S}_W)^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

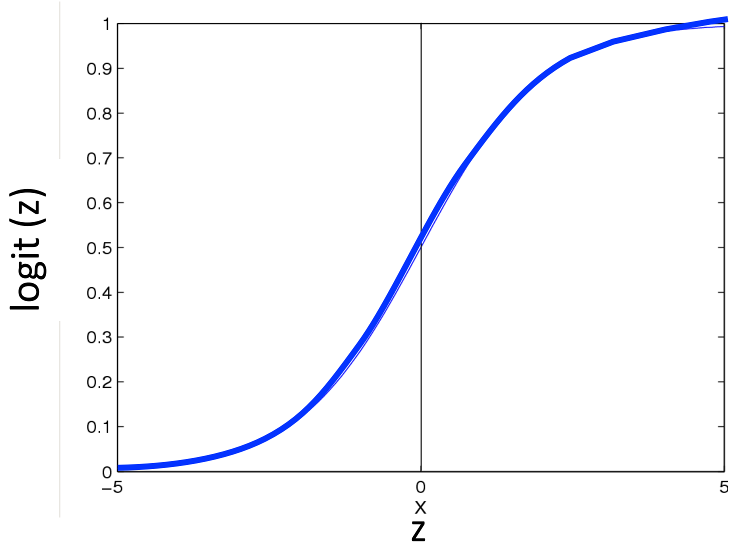
# From Linear to Logistic Regression

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Logistic function applied to a linear function of the data

**Logistic function (or Sigmoid):** $\frac{1}{1+exp(-z)}$



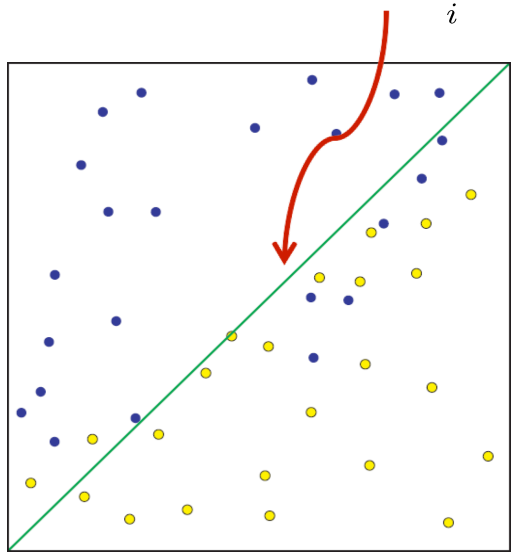
**Features can be discrete or continuous!**

# Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$w_0 + \sum_i w_i X_i = 0$$



Decision boundary:

$$P(Y=0|X) \underset{1}{\overset{0}{\gtrless}} P(Y=1|X)$$

$$w_0 + \sum_i w_i X_i \underset{1}{\overset{0}{\gtrless}} 0$$

**(Linear Decision Boundary)**

# Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$\Rightarrow P(Y=0|X) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$\Rightarrow \frac{P(Y=0|X)}{P(Y=1|X)} = \exp(w_0 + \sum_i w_i X_i) \quad \overset{0}{\underset{1}{\gtrless}} \; 1$$

$$\Rightarrow \boxed{w_0 + \sum_i w_i X_i \quad \overset{0}{\underset{1}{\gtrless}} \; 0}$$

# Logistic Regression

- Label t $\in \{+1, -1\}$modeled as:
$$P(t = 1|x, w) = \sigma(w^T x)$$
- $P(y|x, w) = \sigma(yw^T x), y \in \{+1, -1\}$
- Given a set of parameters w, the probability or likelihood of a datapoint *(x,t):*
$$P(t|x, w) = \sigma(tw^T x)$$

# Logistic Regression

- Given a training dataset $\{(x_1, t_1), \ldots, (x_N, t_N)\}$, log likelihood of a model w is given by:

$$L(w) = \sum_n \ln(P(t_n | x_n, w))$$

- Using principle of maximum likelihood, the best w is given by:

$$w^* = \arg\max_w L(w)$$

## Logistic Regression

- Final Problem:

$$\max_{w} \sum_{i=1}^{n} -\log(1 + \exp(-t_n w^T x_n))$$

Or, $\min_{w} \sum_{i=1}^{n} \log(1 + \exp(-t_n w^T x_n))$

- Error function:

$$E(w) = \sum_{i=1}^{n} \log(1 + \exp(-t_n w^T x_n))$$

- $E(w)$ is convex.

# Logistic Regression

- Final Problem:

$$\max_{w} \sum_{i=1}^{n} -\log(1 + \exp(-t_n w^T x_n))$$

- Regularized Version:

$$max \sum_{i=1}^{n} -\log(1 + \exp(-t_n w^T x_n)) - \lambda w^T w$$

Or, $\min\limits_{w} \sum_{i=1}^{n} \log(1 + \exp(-t_n w^T x_n)) + \lambda\left|\left|w\right|\right|^2$

## Properties of Error function

- Derivatives:

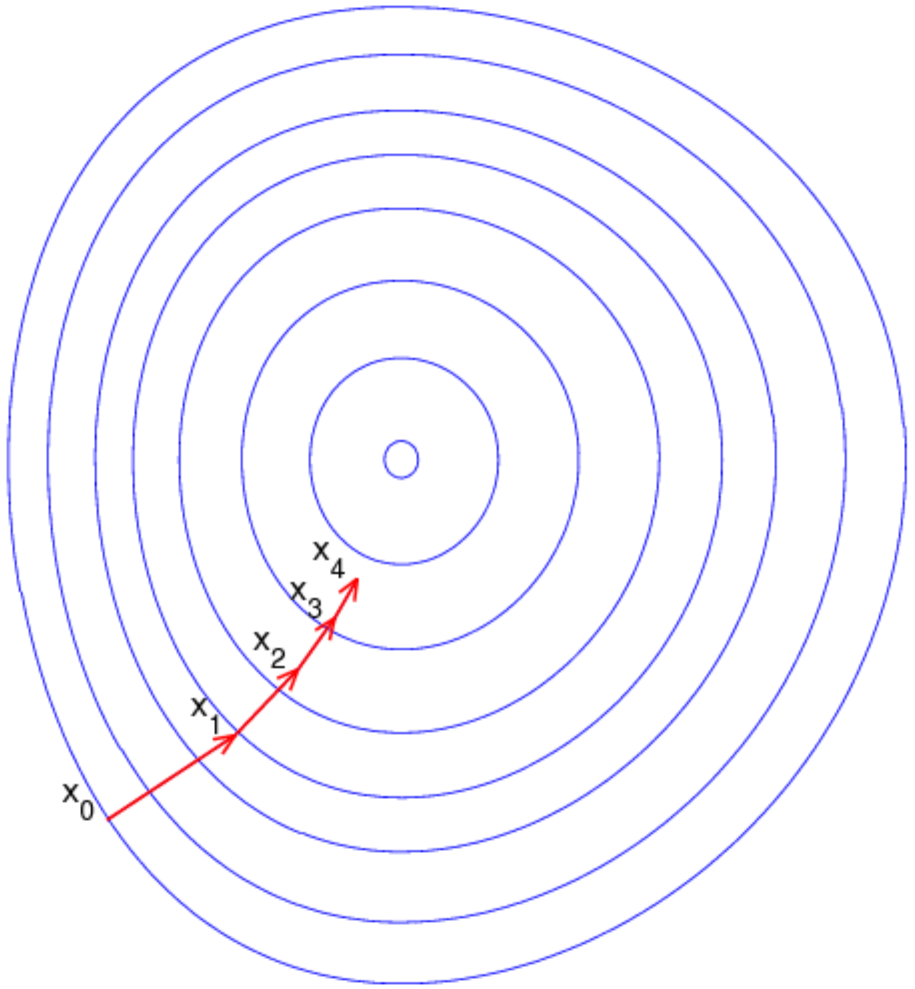
$$\nabla E(w) = \sum_{i=1}^{n} -\big(1 - \sigma(t_i w^T x_i)\big)(t_i x_i)$$

$$\nabla E(w) = \sum_{i=1}^{n} (\sigma(w^T x_i) - t_i) x_i$$

$$\nabla^2 E(w) = \sum_{i=1}^{n} \sigma(t_i w^T x_i)\big(1 - \sigma(t_i w^T x_i)\big) x_i x_i^T$$

# Gradient Descent

- Problem: min f(x)
- f(x): differentiable
- g(x): gradient of f(x)
- Negative gradient is steepest descent direction.
- At each step move in the gradient directio so that there is "sufficient decrease".



# Gradient Descent

```
input  : Function $f$, Gradient $\nabla f$
output: Optimal solution $w^*$

Initialize $w_0 \leftarrow 0$, $k \leftarrow 0$
while $|\nabla f_k| > \epsilon$ do
    Compute $\alpha_k \leftarrow \text{linesearch}(f, -\nabla f_k, w_k)$
    Set $w_{k+1} \leftarrow w_k - \alpha_k \nabla f_k$
    Evaluate $\nabla f_{k+1}$
    $k \leftarrow k + 1$
end
$w^* \leftarrow w_k$
```

# Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$\Rightarrow P(Y=0|X) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$\Rightarrow \frac{P(Y=0|X)}{P(Y=1|X)} = \exp(w_0 + \sum_i w_i X_i) \quad \overset{0}{\underset{1}{\gtrless}} \; 1$$

$$\Rightarrow \boxed{w_0 + \sum_i w_i X_i \quad \overset{0}{\underset{1}{\gtrless}} \; 0}$$

# Logistic Regression for more than 2 classes

- Logistic regression in more general case, where *Y* $\{y_1,\ldots,y_K\}$

for *k<K*

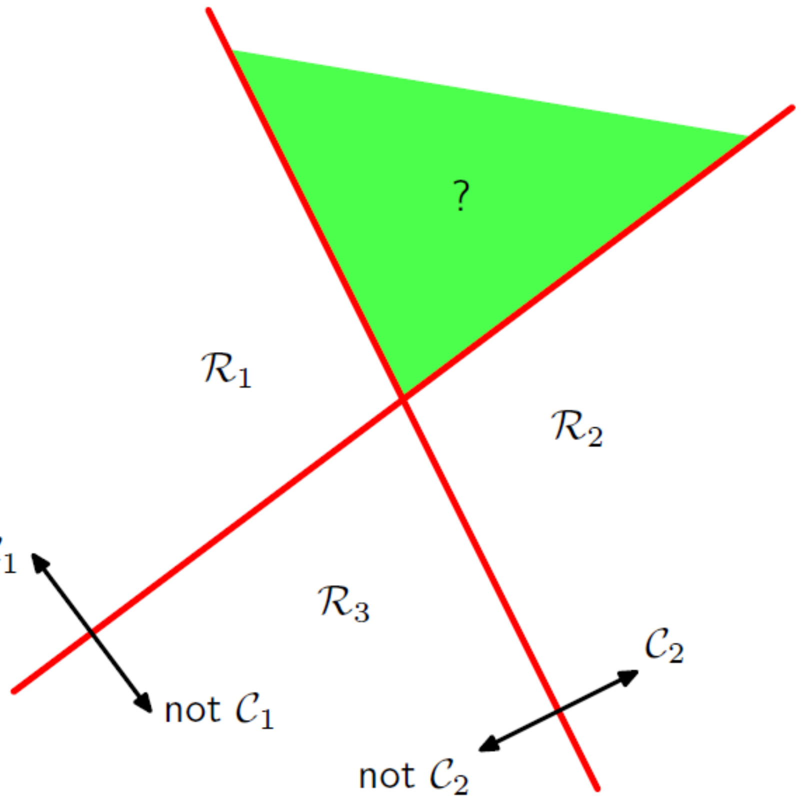
$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^{d} w_{ki} X_i)}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^{d} w_{ji} X_i)}$$

for *k=K* (normalization, so no weights for this class)

$$P(Y = y_K|X) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^{d} w_{ji} X_i)}$$

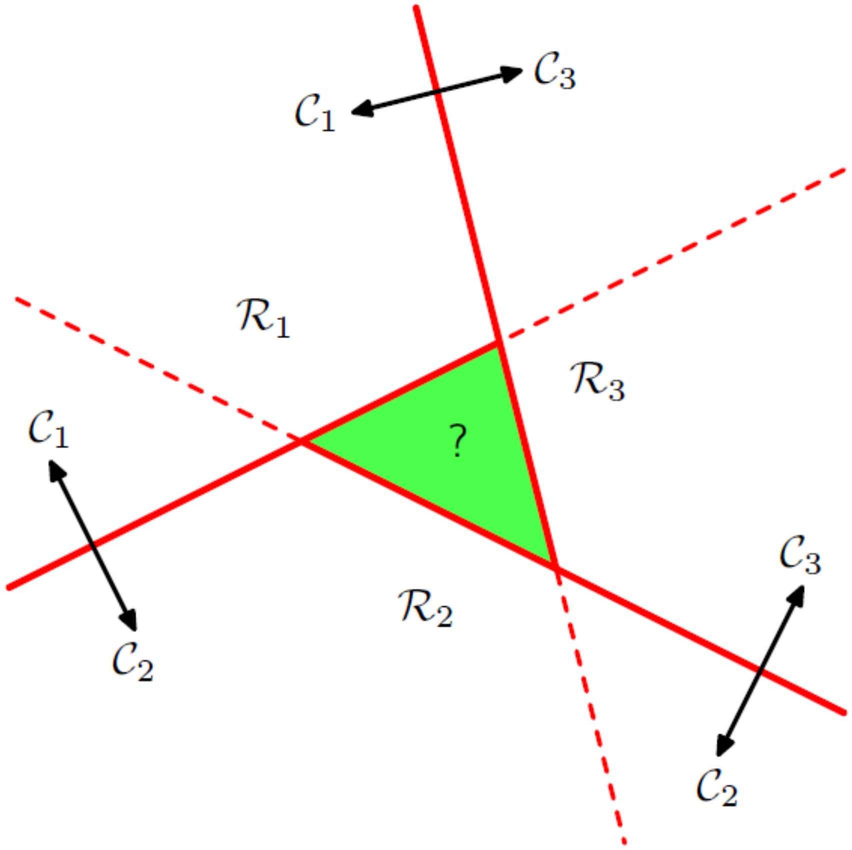
# Multiple classes

- One-vs-all: $K - 1$ hyperplanes each separating $C_1, \dots, C_{K-1}$ classes from rest.
- Otherwise $C_K$
- Low number of classifiers.



# Multiple classes

- One-vs-one: Every pair $C_i - C_j$ get a boundary.
- Final by majority vote.
- High number of classifiers.



# Multiple classes

- K-linear discriminant functions:

$$y_k(x) = w_k^T \boldsymbol{x} + w_{k0}$$

- Assign $x$ to $C_k$ if $y_k(x) \geq y_j(x)$ for all $j \neq k$
- Decision boundary:

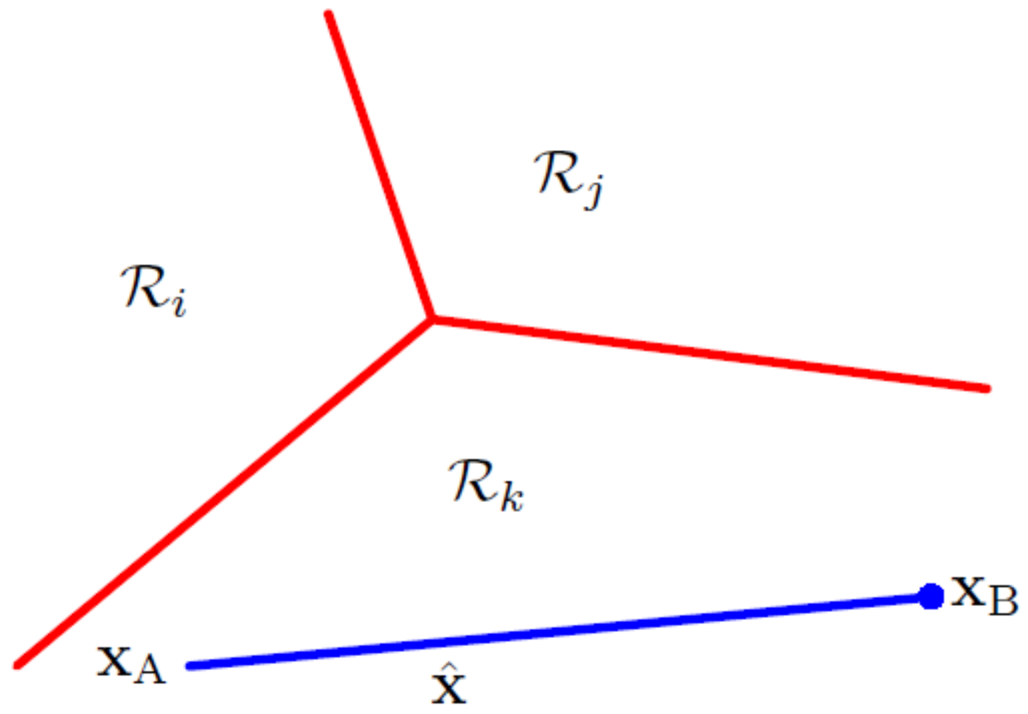
$$\left(w_k - w_j\right)^T \boldsymbol{x} + \left(w_{k0} - w_{j0}\right) = 0$$

- Decision region is singly connected:

$$x = \lambda x_A + (1 - \lambda) x_B$$

- If $x_A$ and $x_B$ have same label, so does $x$.

# Multiple Classes



## NAÏVE BAYES

# Generative vs. Discriminative Classifiers

Discriminative classifiers (e.g. Logistic Regression)

- Assume some functional form for P(Y|X) or for the decision boundary
- Estimate parameters of P(Y|X) directly from training data

Generative classifiers (e.g. Naïve Bayes)

- Assume some functional form for P(X,Y) (or P(X|Y) and P(Y))
- Estimate parameters of P(X|Y), P(Y) directly from training data

arg max_Y P(Y|X) = arg max_Y P(X|Y) P(Y)

## A text classification task: Email spam filtering

```
From: ``'' <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay
Anyone can buy real estate with no money down
Stop paying rent TODAY !
There is no need to spend hundreds or even thousands for
similar courses
I am 22 years old and I have already purchased 6 properties
using the
methods outlined in this truly INCREDIBLE ebook.
Change your life NOW !
=================================================
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
=================================================
```

How would you write a program that would automatically detect and delete this type of message?

## Formal definition of TC: Training

Given:

- A document set X
  - Documents are represented typically in some type of high-dimensional space.
- A fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., relevant vs. nonrelevant).
- A training set D of labeled documents with each labeled document $\langle d, c \rangle \in X \times C$

Using a learning method or learning algorithm, we then wish to

learn a classifier ϒ that maps documents to classes:

$$ϒ : X \rightarrow C$$

## Formal definition of TC: Application/Testing

Given: a description $d \in X$ of a document Determine: $\Upsilon(d) \in C$,

that is, the class that is most appropriate for *d*

## Examples of how search engines use classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. nonspam)
- Topic-specific or *vertical* search – restrict search to a “vertical” like “related to health” (relevant to vertical vs. not)

## Derivation of Naive Bayes rule

We want to find the class that is most likely given the document:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} P(c|d)$$

Apply Bayes rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}:$$

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \frac{P(d|c)P(c)}{P(d)}$$

Drop denominator since P(d) is the same for all classes:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} P(d|c)P(c)$$

## Too many parameters / sparseness

$$\begin{aligned} c_{\text{map}} &= \underset{c \in \mathbb{C}}{\arg\max}\ P(d|c)P(c) \\ &= \underset{c \in \mathbb{C}}{\arg\max}\, P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)P(c) \end{aligned}$$

- There are too many parameters $P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)$ , one for each unique combination of a class and a sequence of words.
- We would need a very, very large number of training examples to estimate that many parameters.
- This is the problem of data sparseness.

## Naive Bayes conditional independence assumption

To reduce the number of parameters to a manageable size, we make the Naive Bayes conditional independence assumption:

$$P(d|c) = P(\langle t_1, \ldots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k | c)$$

We assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(X_k = t_k \mid c)$.

## The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document *d* being in a class *c* as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- $n_d$ is the length of the document. (number of tokens)
- $P(t_k\,|c)$ is the conditional probability of term $t_k$ occurring in a document of class *c*
- $P(t_k\,|c)$ is a measure of how much evidence $t_k$ contributes that *c* is the correct class.
- $P(c)$ is the prior probability of *c*.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$.

## Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the “best” class.
- The best class is the most likely or maximum a posteriori (MAP) class $c_{map}$:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

## Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since log(*xy*) = log(*x*) + log(*y*), we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.

- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)\right]$$

## Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
    - Each conditional parameter log $\hat{P}(t_k|c)$ is a weight that indicates how good an indicator $t_k$ is for $c$.
    - The prior log $\hat{P}(c)$ is a weight that indicates the relative frequency of $c$.
    - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
    - We select the class with the most evidence.

## Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

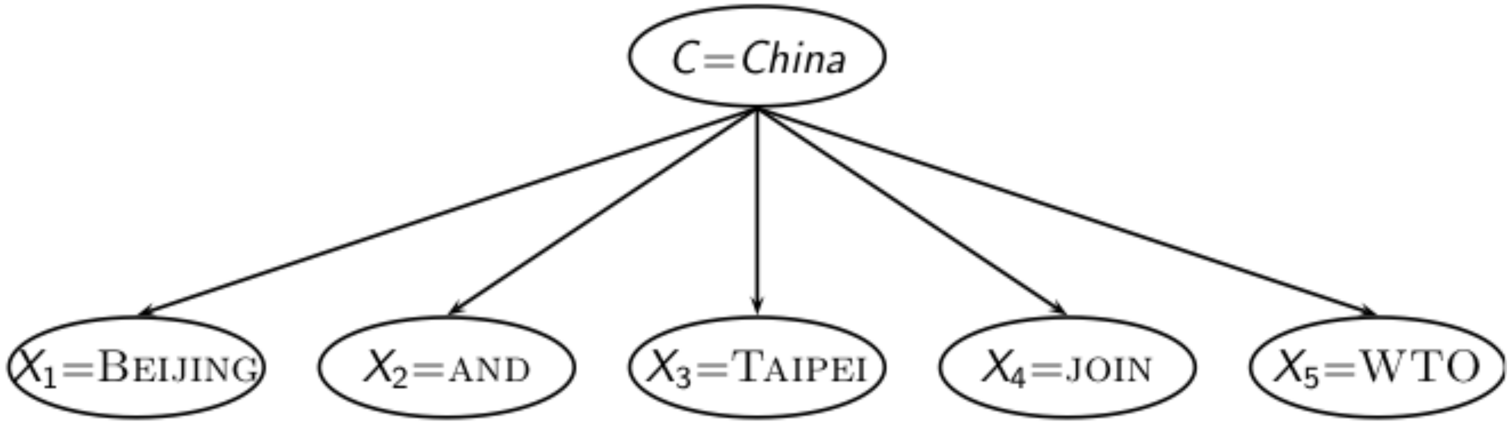
$$\hat{P}(c) = \frac{N_c}{N}$$

- $N_c$ : number of docs in class *c*; *N*: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- $T_{ct}$ is the number of tokens of *t* in training documents from class *c* (includes multiple occurrences)
- We've made a Naive Bayes independence assumption here:

## The problem with maximum likelihood estimates: Zeros



$$P(China|d) \propto P(China) \cdot P(\texttt{BEIJING}|China) \cdot P(\texttt{AND}|China) \cdot P(\texttt{TAIPEI}|China) \cdot P(\texttt{JOIN}|China) \cdot P(\text{WTO}|China)$$

- If WTO never occurs in class China in the train set:

$$\hat{P}(\text{WTO}|China) = \frac{T_{China,\text{WTO}}}{\sum_{t' \in V} T_{China,t'}} = \frac{0}{\sum_{t' \in V} T_{China,t'}} = 0$$

## The problem with maximum likelihood estimates: Zeros (cont)

- If there were no occurrences of WTO in documents in class China, we'd get a zero estimate:

$$\hat{P}(\mathrm{WTO}|China) = \frac{T_{China,\mathrm{WTO}}}{\sum_{t' \in V} T_{China,t'}} = 0$$

- → We will get P(China|d) = 0 for any document that contains WTO!
- Zero probabilities cannot be conditioned away.

## To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

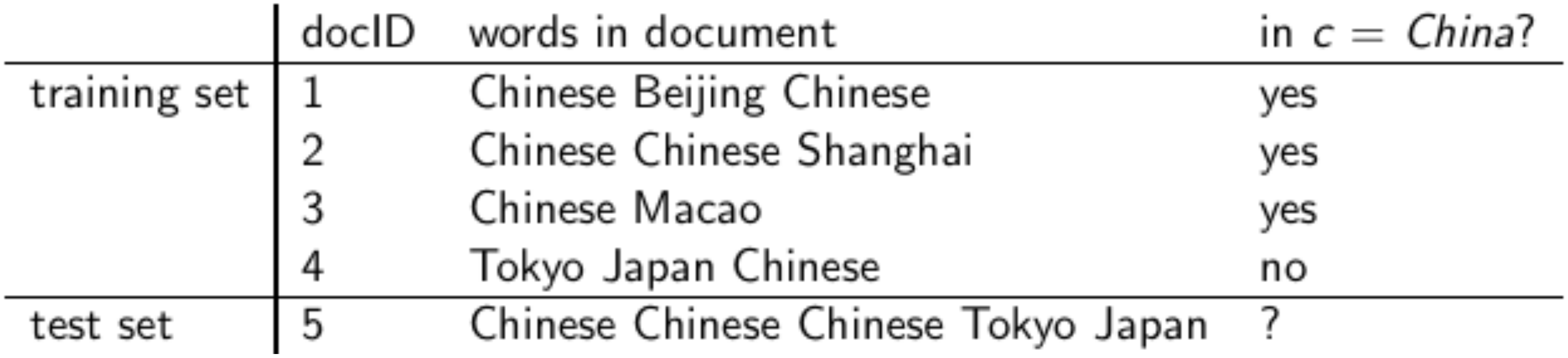
- B is the number of different words (in this case the size of the vocabulary: |*V* | = *B*)

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

## To avoid zeros: Add-one smoothing

- Estimate parameters from the training corpus using add-one smoothing
- For a new document, for each class, compute sum of (i) log of prior and (ii) logs of conditional probabilities of the terms
- Assign the document to the class with the largest score

# Exercise



| | docID | words in document | in $c = China$? |
|---|---|---|---|
| training set | 1 | Chinese Beijing Chinese | yes |
| | 2 | Chinese Chinese Shanghai | yes |
| | 3 | Chinese Macao | yes |
| | 4 | Tokyo Japan Chinese | no |
| test set | 5 | Chinese Chinese Chinese Tokyo Japan | ? |

- Estimate parameters of Naive Bayes classifier
- Classify test document

## Example: Parameter estimates

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$\begin{aligned}
\hat{P}(\textsc{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\textsc{Tokyo}|c) = \hat{P}(\textsc{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\textsc{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textsc{Tokyo}|\overline{c}) = \hat{P}(\textsc{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant $B$ is 6 as the vocabulary consists of six terms.

## Example: Classification

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$
$$\hat{P}(\overline{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Thus, the classifier assigns the test document to *c* = *China*. The reason for this classification decision is that the three occurrences of the positive indicator `CHINESE` in $d_5$ outweigh the occurrences of the two negative indicators `JAPAN` and `TOKYO`.

# Class Conditional Probabilities

To compute, $P(x_k|C_i)$

- $A_k$ is categorical:

$$P(x_k|C_i) = \frac{\textit{the number of tuples of class } C_i \textit{ in D having the value } x_k \textit{ for } A_k}{\textit{the number of tuples of class } C_i \textit{ in D.}}$$

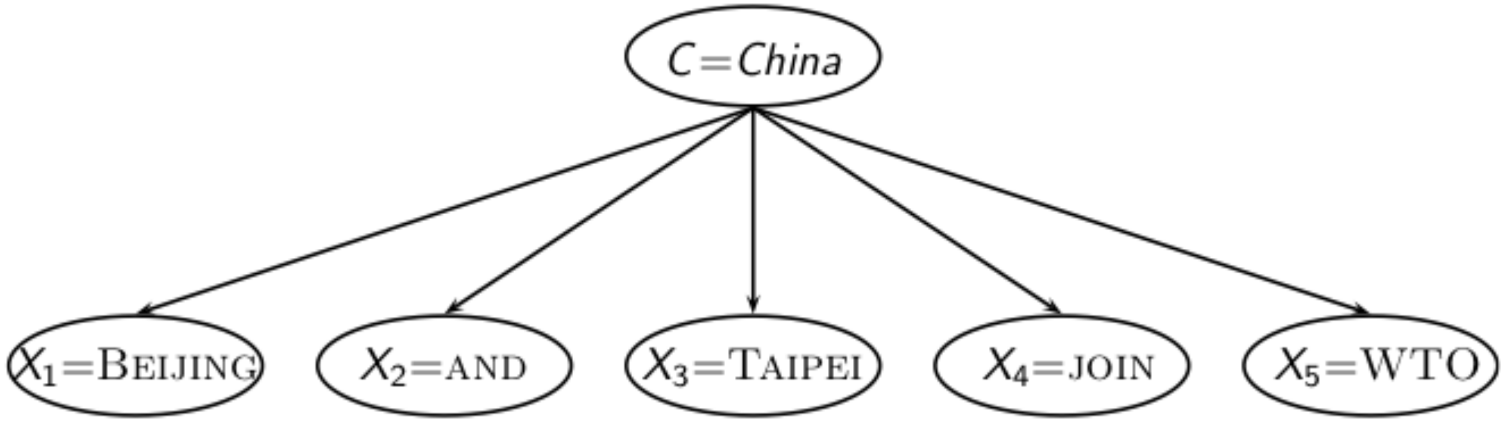
- $A_k$ is continuous:

  A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean $\mu$ and standard deviation $\sigma$

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

## Generative model



$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- Generate a class with probability $P(c)$
- Generate each of the words (in their respective positions), conditional on the class, but independent of each other, with probability $P(t_k \,|c)$
- To classify docs, we "reengineer" this process and find the class that is most likely to have generated the doc.

## On naïve Bayesian classifier

- Advantages:
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)