# Day 3: Introduction to Machine Learning

Kenneth Benoit

Data Mining and Statistical Learning

March 2, 2015

# How do we get "true" condition?

- In some domains: through more expensive or extensive tests
- In social sciences: typically by expert annotation or coding
- A scheme should be tested and reported for its reliability

# Inter-rater reliability

Different types are distinguished by the way the reliability data is obtained.

| **Type** | **Test Design** | **Causes of Disagreements** | Strength |
|---|---|---|---|
| **Stability** | test-retest | intraobserver inconsistencies | weakest |
| **Reproducibility** | test-test | intraobserver inconsistencies + interobserver disagreements | medium |
| **Accuracy** | test-standard | intraobserver inconsistencies + interobserver disagreements + deviations from a standard | strongest |

# Measures of agreement

- Percent agreement Very simple: (number of agreeing ratings) / (total ratings) * 100%
- Correlation
  - (usually) Pearson's $r$, aka product-moment correlation
  - Formula: $r_{AB} = \frac{1}{n-1} \sum_{i=1}^{n} \left( \frac{A_i - \bar{A}}{s_A} \right) \left( \frac{B_i - \bar{B}}{s_B} \right)$
  - May also be ordinal, such as Spearman's rho or Kendall's tau-b
  - Range is [0,1]
- Agreement measures
  - Take into account not only observed agreement, but also *agreement that would have occured by chance*
  - Cohen's $\kappa$ is most common
  - Krippendorf's $\alpha$ is a generalization of Cohen's $\kappa$
  - Both range from [0,1]

# Reliability data matrixes

Example here used binary data (from Krippendorff)

| **Article:** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Coder A** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Coder B** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

- A and B agree on 60% of the articles: 60% agreement
- Correlation is (approximately) 0.10
- Observed *dis*agreement: 4
- Expected *dis*agreement (by chance): 4.4211
- Krippendorff's $\alpha = 1 - \frac{D_o}{D_e} = 1 - \frac{4}{4.4211} = 0.095$
- Cohen's $\kappa$ (nearly) identical

# Naive Bayes classification

- The following examples refer to “words” and “documents” but can be thought of as generic “features” and “cases”
- We will being with a discrete case, and then cover continuous feature values
- Objective is typically MAP: identification of the *maximum a posteriori* class probability

# Multinomial Bayes model of Class given a Word

Consider $J$ word types distributed across $I$ documents, each assigned one of $K$ classes.

*At the word level*, Bayes Theorem tells us that:

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

For two classes, this can be expressed as

$$= \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})} \qquad (1)$$

# Multinomial Bayes model of Class given a Word

## Class-conditional word likelihoods

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- The word likelihood within class
- The maximum likelihood estimate is simply the proportion of times that word $j$ occurs in class $k$, but it is more common to use Laplace smoothing by adding 1 to each oberved count within class

# Multinomial Bayes model of Class given a Word

## Word probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

- This represents the word probability from the training corpus
- Usually uninteresting, since it is constant for the training data, but needed to compute posteriors on a probability scale

# Multinomial Bayes model of Class given a Word

## Class prior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the class prior probability
- Machine learning typically takes this as the document frequency in the training set
- This approach is flawed for scaling, however, since we are scaling the latent class-ness of an unknown document, not predicting class – uniform priors are more appropriate

# Multinomial Bayes model of Class given a Word

## Class posterior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the posterior probability of membership in class $k$ for word $j$
- Under *certain conditions*, this is identical to what LBG (2003) called $P_{wr}$
- Under those conditions, the LBG "wordscore" is the linear difference between $P(c_k|w_j)$ and $P(c_{\neg k}|w_j)$

# "Certain conditions"

- The LBG approach required the identification not only of texts for each training class, but also "reference" scores attached to each training class
- Consider two "reference" scores $s_1$ and $s_2$ attached to two classes $k = 1$ and $k = 2$. Taking $P_1$ as the posterior $P(k = 1|w = j)$ and $P_2$ as $P(k = 2|w = j)$, A generalised score $s_j^*$ for the word $j$ is then

$$
\begin{aligned}
s_j^* &= s_1 P_1 + s_2 P_2 \\
&= s_1 P_1 + s_2(1 - P_1) \\
&= s_1 P_1 + s_2 - s_2 P_1) \\
&= P_1(s_1 - s_2) + s_2
\end{aligned}
$$

# Moving to the document level

- The "Naive" Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a "test" document, to produce:

$$P(c|d) = P(c) \prod_j \frac{P(w_j|c)}{P(w_j)}$$

- This is why we call it "naive": because it (wrongly) assumes:
  - *conditional independence* of word counts
  - *positional independence* of word counts

# Naive Bayes Classification Example

(From Manning, Raghavan and Schütze, *Introduction to Information Retrieval*)

► **Table 13.1** Data for parameter estimation examples.

| | 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 | ? |

# Naive Bayes Classification Example

**Example 13.1:** For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

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

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 $B$ in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

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

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.

# Naive Bayes with continuous covariates

```
library(e1071)   # has a normal distribution Naive Bayes

# Congressional Voting Records of 1984 (abstentions treated as missing)
data(HouseVotes84, package="mlbench")
model <- naiveBayes(Class ~ ., data = HouseVotes84)

# predict the first 10 Congresspeople
data.frame(Predicted = predict(model, HouseVotes84[1:10,-1]),
           Actual = HouseVotes84[1:10,1],
           postPr = predict(model, HouseVotes84[1:10, -1], type = "raw"))

##     Predicted     Actual postPr.democrat postPr.republican
## 1  republican republican    1.029209e-07      9.999999e-01
## 2  republican republican    5.820415e-08      9.999999e-01
## 3  republican   democrat    5.684937e-03      9.943151e-01
## 4    democrat   democrat    9.985798e-01      1.420152e-03
## 5    democrat   democrat    9.666720e-01      3.332802e-02
## 6    democrat   democrat    8.121430e-01      1.878570e-01
## 7  republican   democrat    1.751512e-04      9.998248e-01
## 8  republican republican    8.300100e-06      9.999917e-01
## 9  republican republican    8.277705e-08      9.999999e-01
## 10   democrat   democrat    1.000000e+00      5.029425e-11
```

# Overall prediction performance

```
# now all of them: this is the resubstitution error
(mytable <- table(predict(model, HouseVotes84[,-1]), HouseVotes84$Class))

##
##              democrat republican
##   democrat        238         13
##   republican       29        155

prop.table(mytable, margin=1)

##
##                democrat republican
##   democrat   0.94820717 0.05179283
##   republican 0.15760870 0.84239130
```

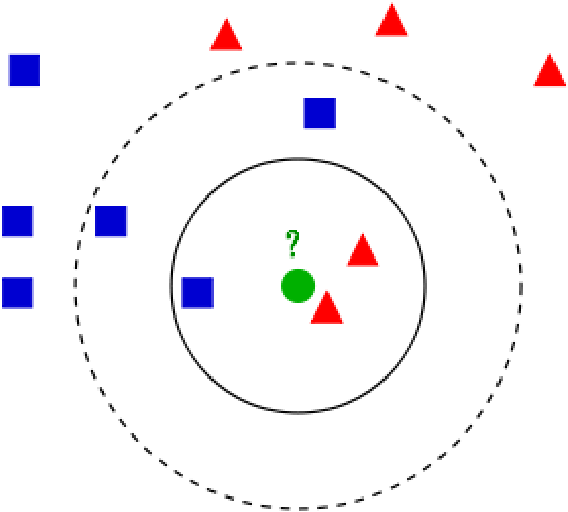
# With Laplace smoothing

```
model <- naiveBayes(Class ~ ., data = HouseVotes84, laplace = 3)
(mytable <- table(predict(model, HouseVotes84[,-1]), HouseVotes84$Class))

##
##              democrat republican
##   democrat        237         12
##   republican       30        156

prop.table(mytable, margin=1)

##
##                democrat republican
##   democrat   0.95180723 0.04819277
##   republican 0.16129032 0.83870968
```

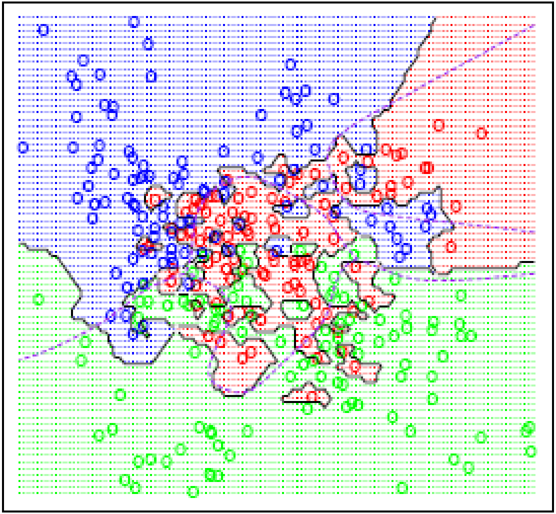
# *k*-nearest neighbour

- A non-parametric method for classifying objects based on the training examples taht are *closest* in the feature space
- A type of instance-based learning, or "lazy learning" where the function is only approximated locally and all computation is deferred until classification
- An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its $k$ nearest neighbors (where $k$ is a positive integer, usually small)
- Extremely *simple*: the only parameter that adjusts is $k$ (number of neighbors to be used) - increasing $k$ *smooths* the decision boundary

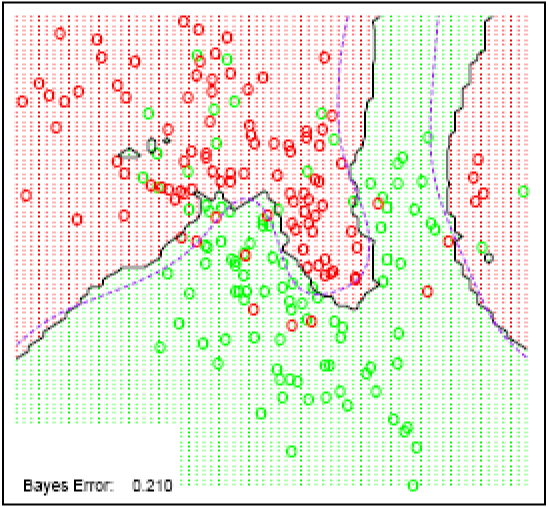
# $k$-NN Example: Red or Blue?



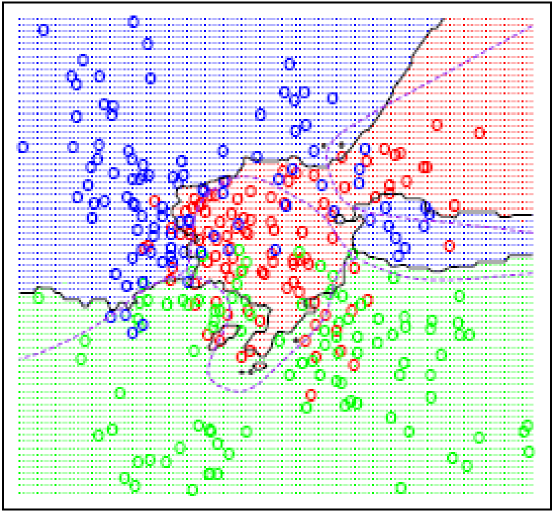
$k = 1$



$k = 7$



## $k = 15$



# Classifying amicus curiae briefs (Evans et al 2007)

```
## kNN classification
require(class)

## Loading required package:  class

require(quantedaData)

## Loading required package:  quantedaData
## Loading required package:  quanteda

data(amicusCorpus)
# create a matrix of documents and features
amicusDfm <- dfm(amicusCorpus, ignoredFeatures=stopwords("english"),
                 stem=TRUE, verbose=FALSE)

## note: using english builtin stopwords, but beware that one size may not fit

# threshold-based feature selection
amicusDfm <- trim(amicusDfm, minCount=10, minDoc=20)

## Features occurring less than 10 times: 9920
## Features occurring in fewer than 20 documents: 11381
```

# Classifying amicus curiae briefs (Evans et al 2007)

```r
# tf-idf weighting
amicusDfm <- weight(amicusDfm, "tfidf")
# partition the training and test sets
train <- amicusDfm[!is.na(docvars(amicusCorpus, "trainclass")), ]
test  <- amicusDfm[!is.na(docvars(amicusCorpus, "testclass")), ]
trainclass <- docvars(amicusCorpus, "trainclass")[1:4]
```

# Classifying amicus curiae briefs (Evans et al 2007)

```r
# classifier with k=1
classified <- knn(train, test, trainclass, k=1)
table(classified, docvars(amicusCorpus, "testclass")[-c(1:4)])

##
## classified AP AR
##          P 13  6
##          R  6 73
```

# Classifying amicus curiae briefs (Evans et al 2007)

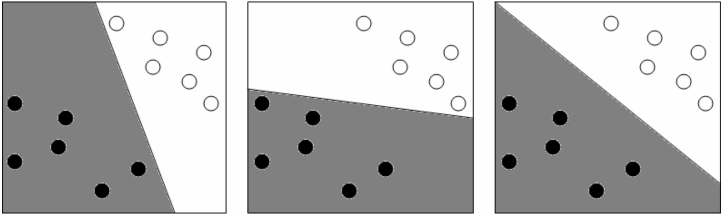
```
# classifier with k=2
classified <- knn(train, test, trainclass, k=2)
table(classified, docvars(amicusCorpus, "testclass")[-c(1:4)])

##
## classified AP AR
##          P  9 33
##          R 10 46
```

# $k$-nearest neighbour issues: Dimensionality

- Distance usually relates to all the attributes and assumes all of them have the same effects on distance
- Misclassification may results from attributes not confirming to this assumption (sometimes called the "curse of dimensionality") – solution is to reduce the dimensions
- There are (many!) different *metrics* of distance

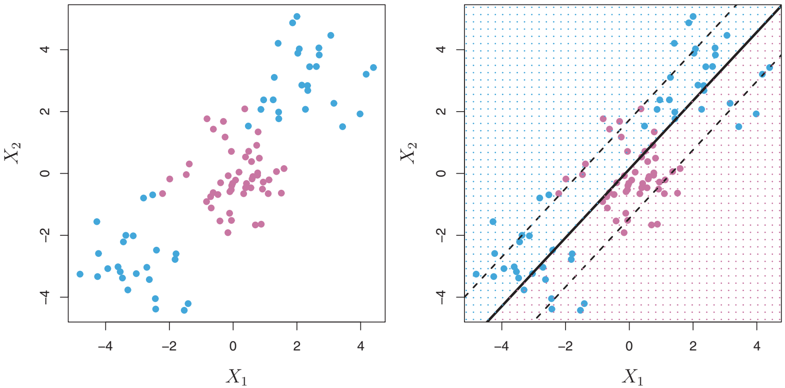
# (Very) General overview to SVMs

- Generalization of maximal margin classifier
- The idea is to find the classification boundary that maximizes the distance to the marginal points



- Unfortunately MMC does not apply to cases with non-linear decision boundaries

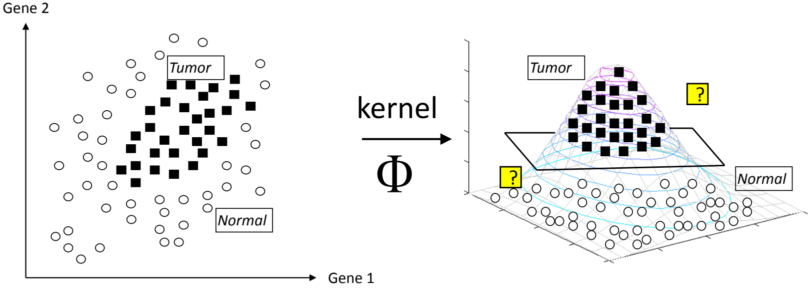
# No solution to this using support vector classifier



# One way to solve this problem

- Basic idea: If a problem is non-linear, don't fit a linear model
- Instead, map the problem from the *input space* to a new (higher-dimensional) *feature space*
- Mapping is done through a non-linear transformation using suitably chosen basis functions
  - the "kernel trick": using kernel functions to enable operations in the high-dimensional feature space without computing coordinates of that space, through computing inner products of all pairs of data in the feature space
  - different kernel choices will produce different results (polynomial, linear, radial basis, etc.)
- Makes it possible to then use a linear model in the feature space

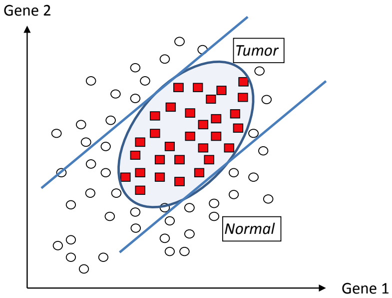
SVMs represent the data in a *higher* dimensional projection using a kernel, and bisect this using a hyperplane




Data is not linearly separable in the input space

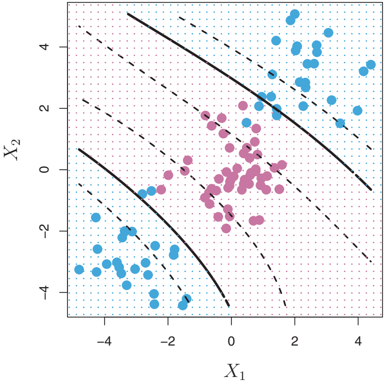
Data is linearly separable in the feature space obtained by a kernel

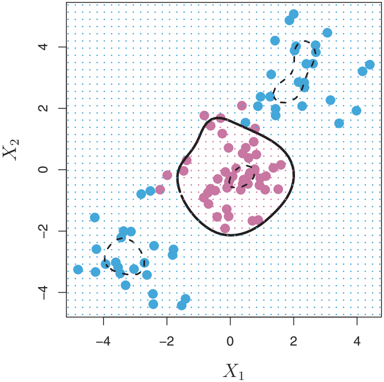
This is only needed when no linear separation plane exists - so not needed in second of these



# Different “kernels” can represent different decision boundaries

- This has to do with different projections of the data into higher-dimensional space
- The mathematics of this are complicated but solveable as forms of optimization problems - but the kernel choice is a user decision



# Precision and recall

- Illustration framework

| | | True condition | |
|---|---|---|---|
| | | Positive | Negative |
| **Prediction** | Positive | True Positive | False Positive (Type I error) |
| | Negative | False Negative (Type II error) | True Negative |

# Precision and recall and related statistics

- Precision: $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$
- Recall: $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$
- Accuracy: $\frac{\text{Correctly classified}}{\text{Total number of cases}}$
- $F1 = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
  (the harmonic mean of precision and recall)

# Example: Computing precision/recall

Assume:

- We have a sample in which 80 outcomes are really positive (as opposed to negative, as in sentiment)
- Our method declares that 60 are positive
- Of the 60 declared positive, 45 are actually positive

Solution:

$$\text{Precision} = (45/(45+15)) = 45/60 = 0.75$$
$$\text{Recall} = (45/(45+35)) = 45/80 = 0.56$$

# Accuracy?

| | | True condition | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| **Prediction** | Positive | 45 | | 60 |
| | Negative | | | |
| | | 80 | | |

# add in the cells we can compute

| | | True condition | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Prediction | Positive | 45 | *15* | 60 |
| | Negative | *35* | | |
| | | 80 | | |

# Receiver Operating Characteristic (ROC) plot