Day 6: Machine Learning

Kenneth Benoit and Slava Mikhaylov

Introduction to Data Science and Big Data Analytics

25 August 2015

Day 7 Outline

- Association rules
- Model evaluation in machine learning
  - Fitting v. overfitting
  - Precision, recall, and accuracy
- Naive Bayes
- k-Nearest Neighbour

Association rules

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:
  - transaction: a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis
  - itemset: the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.
- Many different algorithms, but we will focus on one: the *a priori* algorithm

The *apriori* algorithm

- Two core notions:
  - support: the support of an item X is the number of transactions that contain X divided by the total number of transactions
  - confidence: expresses our "confidence" in the relation *if* X*, then* Y
    - formally:

$$\text{support}(\text{union}(X, Y))/\text{support}(X)$$

- The goal is to discover the interesting rules in a dataset above some pre-defined thresholds of support and confidence, such as 10% and 60%)

Model evaluation in machine learning

Generalization and overfitting

- Generalization: A classifier or a regression algorithm learns to correctly predict output from given inputs not only in previously seen samples but also in previously unseen samples
- Overfitting: A classifier or a regression algorithm learns to correctly predict output from given inputs in previously seen samples but fails to do so in previously unseen samples. This causes poor prediction/generalization

How model fit is evaluated

- For discretely-valued outcomes (class prediction): Goal is to maximize the frontier of precise identification of true condition with accurate recall, defined in terms of false positives and false negatives
  - will define formally later
- For continuously-valued outcomes: minimize Root Mean Squared Error (RMSE)

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		

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

Naive Bayes

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

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

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

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

```r
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)
```

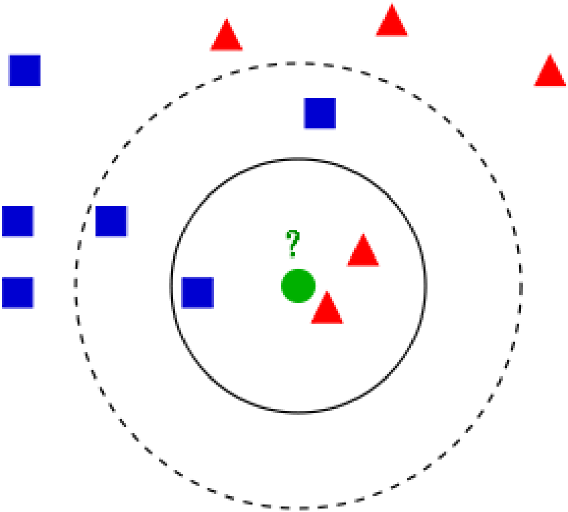
```
##
##                democrat republican
##   democrat   0.95180723 0.04819277
##   republican 0.16129032 0.83870968
```

k-Nearest Neighbour

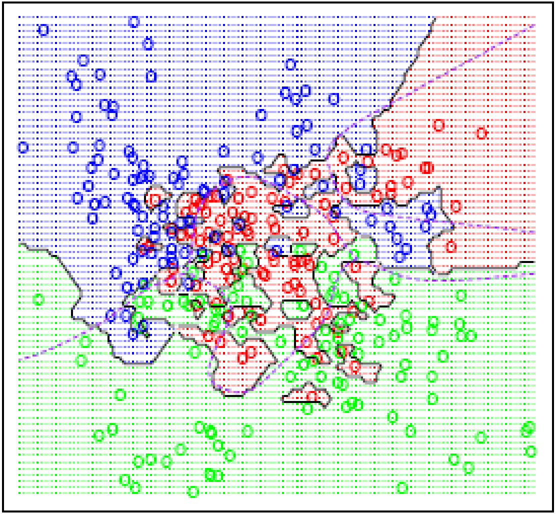
k-nearest neighbour

- 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

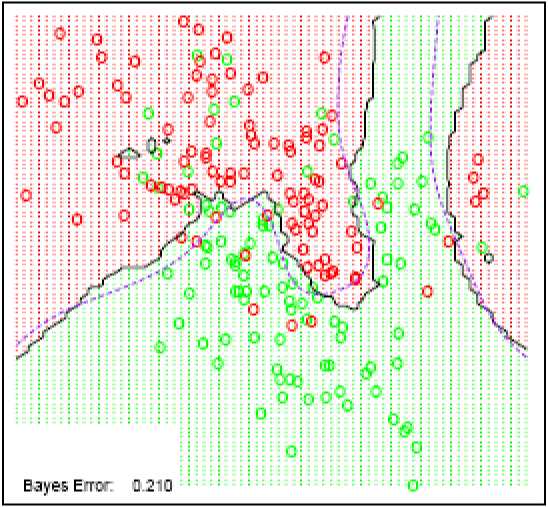
k-NN Example: Red or Blue?



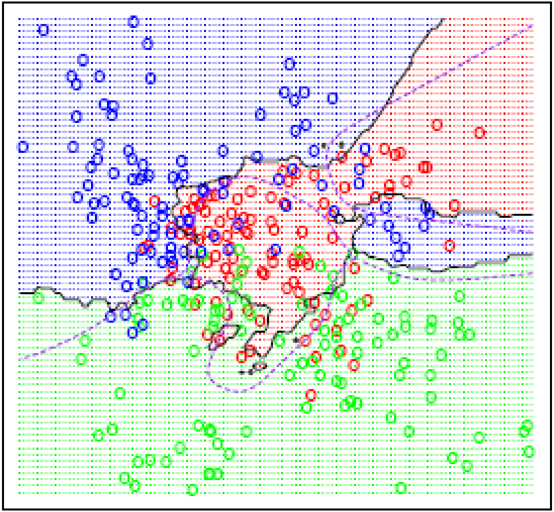
$k = 1$



$k = 7$



$k = 15$



Classifying amicus curiae briefs (Evans et al 2007)

```
## kNN classification
require(class)

## Loading required package:  class

require(quantedaData)

## Loading required package:  quantedaData
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
logical.return = TRUE, :  there is no package called 'quantedaData'

data(amicusCorpus)

## Warning in data(amicusCorpus):  data set 'amicusCorpus' not found

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

## Error in eval(expr, envir, enclos):  could not find function "dfm"

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

## Error in eval(expr, envir, enclos):  could not find function "trim"
```

Classifying amicus curiae briefs (Evans et al 2007)

```
# tf-idf weighting
amicusDfm <- weight(amicusDfm, "tfidf")

## Error in eval(expr, envir, enclos):  could not find function "weight"

# partition the training and test sets
train <- amicusDfm[!is.na(docvars(amicusCorpus, "trainclass")), ]

## Error in eval(expr, envir, enclos):  object 'amicusDfm' not found

test  <- amicusDfm[!is.na(docvars(amicusCorpus, "testclass")), ]

## Error in eval(expr, envir, enclos):  object 'amicusDfm' not found

trainclass <- docvars(amicusCorpus, "trainclass")[1:4]

## Error in eval(expr, envir, enclos):  could not find function
"docvars"
```

Classifying amicus curiae briefs (Evans et al 2007)

```
# classifier with k=1
classified <- knn(train, test, trainclass, k=1)

## Error in as.matrix(train):  object 'train' not found

table(classified, docvars(amicusCorpus, "testclass")[-c(1:4)])

## Error in table(classified, docvars(amicusCorpus,
"testclass")[-c(1:4)]):  object 'classified' not found
```

Classifying amicus curiae briefs (Evans et al 2007)

```
# classifier with k=2
classified <- knn(train, test, trainclass, k=2)

## Error in as.matrix(train):  object 'train' not found

table(classified, docvars(amicusCorpus, "testclass")[-c(1:4)])

## Error in table(classified, docvars(amicusCorpus,
"testclass")[-c(1:4)]):  object 'classified' not found
```

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