Introduction to Classification Methods

Rebecca C. Steorts, Duke University

STA 325, Chapter 4 ISL

Agenda

- Intro to Prototype Classification
- Intro to KNN Classification

Classification

Classification is a predictive task in which the response takes values across discrete categories (i.e., not continuous), and in the most fundamental case, two categories.

Examples:

- Predicting whether a patient will develop breast cancer or remain healthy, given genetic information
- Predicting whether or not a user will like a new product, based on user covariates and a history of his/her previous ratings
- Predicting the region of Italy in which a brand of olive oil was made, based on its chemical composition
- Predicting the next elected president, based on various social, political, and historical measurements

Classification

The point of classification methods is to accurately assign new, unlabeled examples, from the test data} to these classes.

This is “supervised” learning because we can check the performance on the labeled training data.

The point of calculating information was to select features which made classification easier.

Classification versus Clustering

Similar to our usual setup, we observe pairs (x_i, y_i), $i = 1, \dots n$, where y_i gives the class of the ith observation, and $x_i \in \mathbb{R}^p$ are the measurements of p predictor variables

Though the class labels may actually be $y_i \in \{\text{healthy}, \text{sick}\}$ or $y_i \in \{\text{Sardinia}, \text{Sicily}, ...\}$, but we can always encode them as

$$y_i \in \{1, 2, \dots K\}$$

where K is the total number of classes

Note that there is a big difference between classification and clustering; in the latter, there is not a pre-defined notion of class membership (and sometimes, not even K), and we are not given labeled examples (x_i, y_i), $i = 1, \dots n$, but only x_i, $i = 1, \dots n$

Classification versus clustering

Constructed from training data (x_i, y_i), $i = 1, \dots n$, we denote our classification rule by $\hat{f}(x)$; given any $x \in \mathbb{R}^p$, this returns a class label $\hat{f}(x) \in \{1, \dots K\}$

As before, we will see that there are two different ways of assessing the quality of $\hat{f}$: its predictive ability and interpretative ability

Binary classification and linear regression

Classification: K-nearest neighbor versus Prototype Classification

1. In **nearest neighbor** classification, we assign each new data point to the same class as the closest labeled vector, or **exemplar** (or **example**, to be slightly less fancy). This uses lots of memory (because we need to keep track of many vectors) and time (because we need to calculate lots of distances), but assumes next to nothing about the geometry of the classes.
2. In **prototype classification**, we represent each class by a single vector, its *prototype*, and assign new data to the class whose prototype is closest. This uses little memory or computation time, but implicitly assumes that each class forms a compact (in fact, convex) region in the feature space.

Prototype Method

K-Nearest Neighbors vs Linear Regression

Recall that linear regression is an example of a **parametric approach** because it assumes a linear functional form for f(X).

In this module, we introduce K-Nearest Neighbors (KNN), which is a **non-parametric method**

Parametric methods

1. Advantages

- Easy to fit. One needs to estimate a small number of coefficients.
- Often easy to interpret.

2. Disadvantages

- They make strong assumptions about the form of $f(X)$.
- Suppose we assume a linear relationship between X and Y but the true relationship is far from linear, then the resulting model will provide a poor fit to the data, and any conclusions drawn from it will be suspect.

Non-parametric models

1. Advantages

- They do not assume an explicit form for $f(X)$, providing a more flexible approach.

2. Disadvantages

- They can be often more complex to understand and interpret
- If there is a small number of observations per predictor, then parametric methods then to work better.

K-Nearest Neighbors (KNN)

We introduce one of the simplest and best-known non-parametric methods, K-nearest neighbors regression (KNN).

It is closely related to the KNN classifier from Chapter 2 (see ISL and read on your own for more details).

KNN method

1. Assume a value for the number of nearest neighbors K and a prediction point x_o.
2. KNN identifies the training observations N_o closest to the prediction point x_o.
3. KNN estimates $f(x_o)$ using the average of all the reponses in N_o, i.e.
$$\hat{f}(x_o) = \frac{1}{K} \sum_{x_i \in N_o} y_i.$$

Choosing K

- In general, the optimal value for K will depend on the **bias-variance** tradeoff.
- A small value for K provides the most flexible fit, which will have low bias but high variance.
- This variance is due to the fact that the prediction in a given region is entirely dependent on just one observation.
- In contrast, larger values of K provide a smoother and less variable fit; the prediction in a region is an average of several points, and so changing one observation has a smaller effect.

Application of KNN (Chapter 4.6.5 of ISL)

Perform KNN using the knn() function, which is part of the class library.

We call knn() and it forms forms predictions using a single command, with four inputs:

1. A matrix containing the predictors associated with the training data (train.X).
2. A matrix containing the predictors associated with the data for which we wish to make predictions (test.X).
3. A vector containing the class labels for the training observations train.Direction.
4. A value for K, the number of nearest neighbors to be used by the classifier.

Smarket data

- We apply KNN to the Smarket data of the ISLR library.
- We will begin by examining some numerical and graphical summaries.
- This data set consists of percentage returns for the S&P 500 stock index over 1, 250 days, from the beginning of 2001 until the end of 2005.
- For each date, we have recorded the percentage returns for each of the five previous trading days, Lag1 through Lag5. We have also recorded Volume (the number of shares traded

Smarket data

```
library(ISLR)
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
dim(Smarket)
```

```
## [1] 1250    9
```

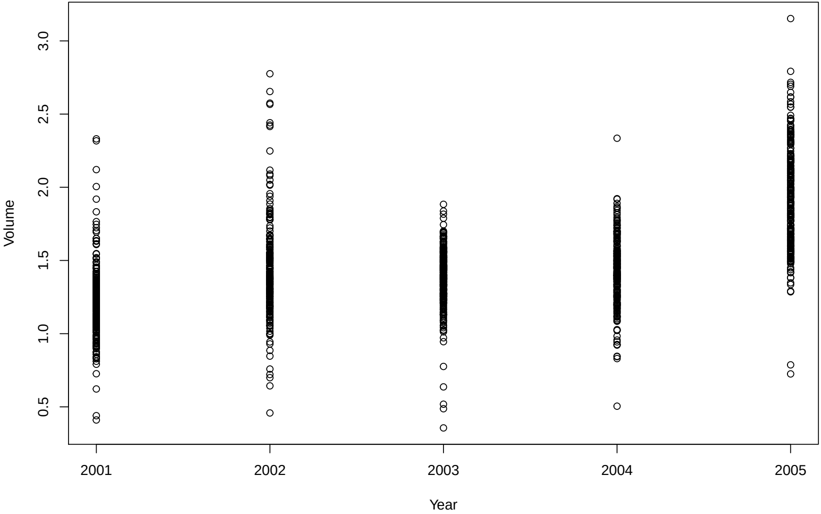
Smarket data

```r
cor(Smarket[,-9])
```

```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

- As one would expect, the correlations between the lag variables and today's returns are close to zero.
- In other words, there appears to be little correlation between today's returns and previous days' returns. The only substantial correlation is between Year and Volume.

Correlation



By plotting the data we see that Volume is increasing over time. In other words, the average number of shares traded daily increased from 2001 to 2005.

KKN applied to the Smarket Data

We now fit a KNN to the Smarket data.

We use the cbind() function, short for column bind, to bind the Lag1 and Lag2 variables together into two matrices, one for the training set and the other for the test set.

KKN applied to the Smarket Data

```r
library(class)
train = (Year < 2005)
Smarket.2005= Smarket[!train ,]
dim(Smarket.2005)
```

```
## [1] 252   9
```

```r
Direction.2005=Direction[!train]
```

The object train is a vector of 1,250 elements, corresponding to the observations in our data set.

The elements of the vector that correspond to observations that occurred before 2005 are set to TRUE, whereas those that correspond to observations in 2005 are set to FALSE.

The object train is a Boolean vector, since its elements are TRUE and FALSE.

KKN applied to the Smarket Data

```
train.X=cbind(Lag1 ,Lag2)[train ,]
test.X=cbind(Lag1,Lag2)[!train,]
train.Direction =Direction [train]
```

- Now the knn() function can be used to predict the market's movement for the dates in 2005.
- We set a seed for reproducibility.

KKN applied to the Smarket Data

```
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction ,k=1)
table(knn.pred,Direction.2005)
```

```
##           Direction.2005
## knn.pred Down Up
##      Down   43 58
##      Up     68 83
```

- The results using $K = 1$ are not very good, since only 50% of the observa- tions are correctly predicted.
- Of course, it may be that $K = 1$ results in an overly flexible fit to the data.

KKN applied to the Smarket Data

```r
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction ,k=3)
table(knn.pred,Direction.2005)
```

```
##          Direction.2005
## knn.pred Down Up
##     Down   48 55
##     Up     63 86
```

The results have improved slightly. But increasing K further turns out to provide no further improvements. It turns out the KNN does not work very well for this data set.

However, it did not take very long to make this conclusion. (If you try QDA, you will find it works quite well).