TP2 : kNN, DECISION TREES AND STOCK MARKET RETURNS

Prédicteur kNN et validation croisée

Le but de cette partie est d'apprendre à utiliser le classifieur kNN avec le logiciel `R`. Pour cela, on chargera la librairie `knnflex` :

```
install.packages("knnflex")
library(knnflex)
```

On va considérer le jeu de données `iris`, devenu classique en statistique et apprentissage :

```
library(class)
data(iris)
head(iris)
summary(iris)
```

On séparera ces données en 2 partie : un échantillon d'apprentissage et un échantillon d'évaluation (ou de test) :

```
train = iris[c(1:30,51:80,101:130),1:5]
test = iris[c(31:50,81:100,131:150),1:5]
```

On veut donc déterminer l'espèce d'iris à partir des mesures prises sur le pétale et le sépale. Pour obtenir les prédictions, on utilisera la fonction `knn()` :

```
pred = knn(train[,1:4], test[,1:4], train[,5], k = 2)
# display the confusion matrix
table(pred,test[,5])
```

Ces résultat montrent que le clasifieur kNN avec $k = 2$ conduit vers une erreur de classification très faible (3.33%) dans ce problème.

Afin de choisir k, il est récommandé d'effectuer une validation croisée, comme montré ci-dessous :

```
# 5-fold cross-validation to select k
# from the set {1,...,10}
fold = sample(rep(1:5,each=18))           # creation des groupes B_v
cvpred = matrix(NA,nrow=90,ncol=10)       # initialisation de la matrice
                                          # des prédicteurs
for (k in 1:10)
  for (v in 1:5)
   {
   sample1 = train[which(fold!=v),1:4]
   sample2 = train[which(fold==v),1:4]
   class1 = train[which(fold!=v),5]
   cvpred[which(fold==v),k] = knn(sample1,sample2,class1,k=k)
   }
class = as.numeric(train[,5])
# display misclassification rates for k=1:10
apply(cvpred,2,function(x) sum(class!=x)) # calcule l'erreur de classif.
```

Predicting stock market returns [1]

1. Problem Description and Objectives

Stock market trading is an application domain with a large potential for data mining. In effect, the existence of an enormous amount of historical data suggests that data mining can provide a competitive advantage over human inspection of these data. On the other hand, there are researchers claiming that the markets adapt so rapidly in terms of price adjustments that there is no space to obtain profits in a consistent way. This is usually known as the efficient markets hypothesis. This theory has been successively replaced by more relaxed versions that leave some space for trading opportunities due to temporary market inefficiencies.

The general goal of stock trading is to maintain a portfolio of assets based on buy and sell orders. The long-term objective is to achieve as much profit as possible from these trading actions. In the context of the present work, we will only "trade" a single security, actually a market index. Given this security and an initial capital, we will try to maximize our profit over a future testing period by means of trading actions (Buy, Sell, Hold). Our trading strategy will use as a basis for decision making the indications provided by the result of a data mining process. This process will consist of trying to predict the future evolution of the index based on a model obtained with historical quotes data. Thus our prediction model will be incorporated in a trading system that generates its decisions based on the predictions of the model. Our overall evaluation criteria will be the performance of this trading system, that is, the profit/loss resulting from the actions of the system as well as some other statistics that are of interest to investors. This means that our main evaluation criteria will be the operational results of applying the knowledge discovered by our data mining process and not the predictive accuracy of the models developed during this process.

2. The Available Data

In our case study we will concentrate on trading the S&P 500 market index. Daily data concerning the quotes of this security are freely available in many places, for example, the Yahoo finance site (`http://finance.yahoo.com`).

The data we will use is available in the package `DMwR`. In order to get the data, it is enough to issue

```
library(DMwR)
data(GSPC)
```

The first statement is only required if you have not issued it before in your `R` session. The second instruction will load an object, `GSPC`, of class `xts`. For now you can manipulate it as if it were a matrix or a data frame (try, for example, head(GSPC)).

The daily stock quotes data includes information regarding the following properties :

- Date of the stock exchange session
- Open price at the beginning of the session
- Highest price during the session
- Lowest price
- Closing price of the session
- Volume of transactions
- Adjusted close price : this is basically the closing price adjusted for stock splits, dividends/distributions, and rights offerings.

The data available for this case study depends on time. This means that each observation of our dataset has a time tag attached to it. This type of data is frequently known as time series data. The main distinguishing feature of this kind of data is that order between cases matters, due to their attached time tags.

[1] This part is borrowed from the excellent book by Luis Torgo "Data Mining with R : Learning with Case Studies." Chapman & Hall/CRC Data Mining and Knowledge Discovery Series 2010.

In the case of our stocks data, we have what is usually known as a multivariate time series, because we measure several variables at the same time tags, namely the Open, High, Low, Close, Volume, and AdjClose.

R has several packages devoted to the analysis of this type of data, and in effect it has special classes of objects that are used to store type-dependent data. Moreover, R has many functions tuned for this type of objects, like special plotting functions, etc.

3. Defining the Prediction Tasks

Generally speaking, our goal is to have good forecasts of the future price of the S&P 500 index so that profitable orders can be placed on time. This general goal should allow us to easily define what to predict with our models – it should resort to forecast the future values of the price time series. However, it is easy to see that even with this simple task we immediately face several questions, namely, (1) which of the daily quotes ? or (2) for which time in the future ? Answering these questions may not be easy and usually depends on how the predictions will be used for generating trading orders.

3.1 What to Predict ?

The trading strategies we will describe assume that we obtain a prediction of the tendency of the market in the next few days. Based on this prediction, we will place orders that will be profitable if the tendency is confirmed in the future.

Let us assume that if the prices vary more than $p\%$, we consider this worthwhile in terms of trading (e.g., covering transaction costs). In this context, we want our prediction models to forecast whether this margin is attainable in the next k days. Please note that within these k days we can actually observe prices both above and below this percentage. This means that predicting a particular quote for a specific future time $t + k$ might not be the best idea. In effect, what we want is to have a prediction of the overall dynamics of the price in the next k days, and this is not captured by a particular price at a specific time. For instance, the closing price at time $t + k$ may represent a variation much lower than $p\%$, but it could have been preceded by a period of prices representing variations much higher than $p\%$ within the window $t \dots t + k$. So, what we want in effect is to have a good prediction of the overall tendency of the prices in the next k days.

We will describe a variable, calculated with the quotes data, that can be seen as an indicator (a value) of the tendency in the next k days. The value of this indicator should be related to the confidence we have that the target margin p will be attainable in the next k days. At this stage it is important to note that when we mention a variation in $p\%$, we mean above or below the current price. The idea is that positive variations will lead us to buy, while negative variations will trigger sell actions. The indicator we are proposing resumes the tendency as a single value, positive for upward tendencies, and negative for downward price tendencies.

Let the daily average price be approximated by

$$\bar{P}_i = \frac{C_i + H_i + L_i}{3} \tag{1}$$

where C_i, H_i and L_i are the close, high, and low quotes for day i, respectively.

Let V_i be the set of k percentage variations of today's close to the following k days average prices (often called arithmetic returns) :

$$V_i = \left(\frac{\bar{P}_{i+1} - C_i}{C_i}, \frac{\bar{P}_{i+2} - C_i}{C_i}, \dots, \frac{\bar{P}_{i+k} - C_i}{C_i} \right). \tag{2}$$

Our indicator variable is the total sum of the variations whose absolute value is above our target margin $p\%$:

$$T_i = \sum_{j=1}^{k} V_i^j \mathbb{1}(|V_i^j| > p\%). \tag{3}$$

The general idea of the variable T is to signal k-days periods that have several days with average daily prices clearly above the target variation. High positive values of T mean that there are several average daily prices that are p% higher than today's close. Such situations are good indications of potential opportunities to issue a buy order, as we have good expectations that the prices will rise. On the other hand, highly negative values of T suggest sell actions, given the prices will probably decline. Values around zero can be caused by periods with "flat" prices or by conflicting positive and negative variations that cancel each other.

The following function implements this simple indicator :

```
T.ind = function(quotes, tgt.margin = 0.025, n.days = 10) {
  v = apply(HLC(quotes), 1, mean)
  r = matrix(NA, ncol = n.days, nrow = NROW(quotes))
  for (x in 1:n.days) r[, x] = Next(Delt(v, k = x), x)
  x = apply(r, 1, function(x) sum(x[x > tgt.margin | x <
  -tgt.margin]))
  if (is.xts(quotes))
  xts(x, time(quotes))
  else x
  }
```

The function starts by obtaining the average price calculated according to Equation (1). The function `HLC()` extracts the High, Low, and Close quotes from a quotes object. We then obtain the returns of the next `n.days` days with respect to the current close price. The `Next()` function allows one to shift the values of a time series in time (both forward or backward). The `Delt()` function can be used to calculate percentage or log returns of a series of prices. Finally, the `T.ind()` function sums up the large absolute returns, that is, returns above the target variation margin, which we have set by default to 2.5%.

We can get a better idea of the behavior of this indicator by looking at the figure produced with the following code :

```
candleChart(last(GSPC, "3 months"), theme = "white", TA = NULL)
avgPrice = function(p) apply(HLC(p), 1, mean)
addAvgPrice = newTA(FUN = avgPrice, col = 1, legend = "AvgPrice")
addT.ind = newTA(FUN = T.ind, col = "red", legend = "tgtRet")
addAvgPrice(on = 1)
addT.ind()
```

The function `candleChart()` draws candlestick graphs (Chandeliers japonais) of stock quotes. We have added to the candlestick graph two indicators : the average price (on the same graph as the candlesticks) and our T indicator (below). The function `newTA()` can be used to create new plotting functions for indicators that we wish to include in candlestick graphs.

◁ **Question 1** Ecrire un code qui permet d'ajouter aux chandeliers japonais la courbe des valeurs médianes de $(C_i, , H_i, L_i)$. Que se passe-t-il si l'on supprime l'argument `on=1` de la fonction `addAvgPrice` ?

In our approach to this problem we will assume that the correct trading action at time t is related to what our expectations are concerning the evolution of prices in the next k days. Moreover, we will describe this future evolution of the prices by our indicator T. The correct trading signal at time t will be "buy" if the T score is higher than a certain threshold, and will be "sell" if the score is below another threshold. In all other cases, the correct signal will be do nothing (i.e., "hold"). In summary, we want to be able to predict the correct signal for time t. On historical data we will fill in the correct signal for each day by calculating the respective T scores and using the thresholding method just outlined above.

3.2 Which Predictors ?

We are assuming that if in the past a certain behavior p was followed by another behavior f, and if that causal chain happened frequently, then it is plausible to assume that this will occur again in the future. Thus if we observe p now, we predict that we will observe f next. We are approximating the future behavior (f), by our indicator T. We now have to decide on how we will describe the recent prices pattern (p in the description above).

The simplest type of information we can use to describe the past are the recent observed prices. That is the type of approach followed in several standard time series modeling approaches. These approaches develop models that describe the relationship between future values of a time series and a window of past q observations of this time series. We will try to enrich our description of the current dynamics of the time series by adding further features to this window of recent prices.

Technical indicators are numeric summaries that reflect some properties of the price time series. The amount of technical indicators available can be overwhelming. In R we can find a very good sample of them, thanks to package TTR.

We will use a simple approach to select the features (feature = technical indicator) to include in our model. The idea is to illustrate this process with a concrete example and not to find the best possible solution to this problem. We will define an initial set of features and then use a technique to estimate the importance of each of these features. Based on these estimates we will select the most relevant features.

We will center our analysis on the Close quote, as our buy/sell decisions will be made at the end of each daily session. The initial set of features will be formed by several past returns on the Close price. The h-days (arithmetic) returns, or percentage variations, can be calculated as

$$R_{i-h} = \frac{C_i - C_{i-h}}{C_{i-h}}. \tag{4}$$

We have included in the set of candidate features ten of these returns by varying h from 1 to 10. Next, we have selected a representative set of technical indicators, from those available in package TTR – namely, the Average True Range (ATR), which is an indicator of the volatility of the series ; the Stochastic Momentum Index (SMI), which is a momentum indicator ; the Welles Wilder's Directional Movement Index (ADX) ; the Aroon indicator that tries to identify starting trends ; the Bollinger Bands that compare the volatility over a period of time ; the Chaikin Volatility ; the Close Location Value (CLV) that relates the session Close to its trading range ; the Arms' Ease of Movement Value (EMV) ; the MACD oscillator ; the Money Flow Index (MFI) ; the Parabolic Stop-and-Reverse ; and the Volatility indicator.

The following functions implement this process :

```
myATR = function(x) ATR(HLC(x))[, "atr"]
mySMI = function(x) SMI(HLC(x))[, "SMI"]
myADX = function(x) ADX(HLC(x))[, "ADX"]
myAroon = function(x) aroon(x[, c("High", "Low")])$oscillator
myBB = function(x) BBands(HLC(x))[, "pctB"]
myChaikinVol = function(x) Delt(chaikinVolatility(x[, c("High","Low")]))[, 1]
myCLV = function(x) EMA(CLV(HLC(x)))[, 1]
myEMV = function(x) EMV(x[, c("High", "Low")], x[, "Volume"])[,2]
myMACD = function(x) MACD(Cl(x))[, 2]
myMFI = function(x) MFI(x[, c("High", "Low", "Close")],x[, "Volume"])
mySAR = function(x) SAR(x[, c("High", "Close")])[, 1]
myVolat = function(x) volatility(OHLC(x), calc = "garman")[,1]
```

The variables we have just described form our initial set of predictors for the task of forecasting the future value of the T indicator. We will try to reduce this set of 22 variables using a feature selection method. Random forests can be used to estimate the importance of the variables involved in a prediction task. Informally, this importance can be estimated by calculating the percentage increase in the error of the random forest if we remove each variable in turn.

In our approach to this application, we will split the available data into two separate sets : (1) one used for constructing the trading system ; and (2) other to test it. The first set will be formed by the first 30 years of quotes of S&P 500. We will leave the remaining data (around 9 years) for the final test of our trading system. In this context, we must leave this final test set out of this feature selection process to ensure unbiased results.

We first build a random forest using the data available for training :

```
data(GSPC)
library(randomForest)
data.model = specifyModel(T.ind(GSPC) ~ Delt(Cl(GSPC),k=1:10) +
myATR(GSPC) + mySMI(GSPC) + myADX(GSPC) + myAroon(GSPC) +
myBB(GSPC) + myChaikinVol(GSPC) + myCLV(GSPC) +
CMO(Cl(GSPC)) + EMA(Delt(Cl(GSPC))) + myEMV(GSPC) +
myVolat(GSPC) + myMACD(GSPC) + myMFI(GSPC) + RSI(Cl(GSPC)) +
mySAR(GSPC) + runMean(Cl(GSPC)) + runSD(Cl(GSPC)))
set.seed(1234)
rf = buildModel(data.model,method="randomForest",
training.per=c(start(GSPC),index(GSPC["1999-12-31"])),
ntree=50, importance=T)
```

◁ **Question 2** A quoi l'option `training.per` de la fonction `buildModel` correspond-t-elle ? et l'option `importance` ?

After obtaining the model, we can check the importance of the variables as follows :

```
varImpPlot(rf@fitted.model, type = 1)
```

◁ **Question 3** Sachant que le graphique tracé représente le pourcentage d'augmentation de l'erreur quadratique due à la suppression d'une variable explicative, déterminer les 8 variables les plus pertinentes.

◁ **Question 4** En utilisant la fonction `specifyModel`, définir le nouveau modèle `data.model` qui a pour variable à expliquer `T.ind(GSPC)` et comme variables explicatives les 8 variables les plus pertinentes trouvées dans la question précédente.

The following code creates all the data structures that we will use in what follows for obtaining predictive models for the two tasks.

```
Tdata.train = as.data.frame(modelData(data.model,
      data.window=c("1970-01-02","1999-12-31")))
Tdata.eval = na.omit(as.data.frame(modelData(data.model,
      data.window=c("2000-01-01","2009-09-15"))))
```

◁ **Question 5** Que fait la fonction `na.omit` utilisée ci-dessus. Pourquoi son utilisation est plus importante (voir indispensable) dans la définition de l'échantillon de test, alors que l'on s'en passe dans la définition de l'échantillon d'entrainement.

Instead of predicting T, we will focus our attention on the prediction of

$$signal = \begin{cases} sell, & \text{if } T < -0.1, \\ hold, & \text{if } |T| \leq 0.1, \\ buy, & \text{if } T > 0.1. \end{cases}$$

The selection of the values 0.1 and -0.1 is purely heuristic and we can also use other thresholds. Still, these values mean that during the 10 day-period used to generate the T values, there were at least four average daily prices that are 2.5% above the current close ($4 \times 0.025 = 0.1$).

Function `trading.signals()`, available in the book package, can carry out this transformation of the numeric T values into a factor with three possible values : ''s'', ''h'', and ''b'', for sell, hold and buy actions.

```
Tdata.train[,1]  = trading.signals(Tdata.train[,1],0.1,-0.1)
names(Tdata.train)[1] = "signal"
summary(Tdata.train)
```

A faire pour le compte-rendu du TP2

Utilisez votre éditeur de texte préféré (Word, LaTeX, Open Office) pour rédiger le compte-rendu. Convertissez le fichier final en pdf avant de me l'envoyer à l'adresse

arnak.dalalyan@ensae.fr

1. Expliquer la commande `apply(cvpred,2,function(x) sum(class!=x))` utilisée à la page 2.
2. Pourquoi si l'on relance deux fois les commandes de la page 2, les résultats obtenus ne sont pas les mêmes ? Imaginons que nous avons lancé ces commandes 100 fois. Proposez une stratégie pour choisir k en combinant ces 100 résultats obtenus.
3. Répondre aux questions 1-5 ci-dessus an insérant le code le cas échéant.
4. Utiliser l'algorithme kNN pour prédire la variable `signal` sur l'échantillon de test `Tdata.eval`, en utilisant `Tdata.train` comme échantillon d'entraînement. Evaluer l'erreur de la prédiction. Inclure le code.
5. La même chose pour l'algorithme d'arbre de décision (cf. la commande `rpart()` vue en TP1). Est-il meilleur que kNN ? Afficher l'arbre de décision obtenu.

La date limite pour l'envoi du compte-rendu est le 8 mars.