Version v18.1

Lecture notes in data mining and multidimensional statistical analysis

michael.blum@imag.fr, jean-baptiste.durand@imag.fr, massih-reza.amini@imag.fr

1 Multiple Regression

1.1 Introduction

We assume that we have the p-dimensional input vectors $x_{i\cdot} = (x_{i1}, x_{i2}, \ldots, x_{ip})$, and we want to predict the real-valued output Y_i's for $i = 1, \ldots, n$ where n is the number of datapoints. The linear regression model has the form

$$Y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i, \tag{1}$$

where ε_i is a residual term with $E[\varepsilon_i] = 0$ and $\mathrm{var}(\varepsilon_i) = \sigma^2$. Here the β_j's are unknown coefficients, and the input variables can come from different sources: quantitative inputs; transformations of quantitative inputs, such as log, square-root or square; interactions between variables, for example, $x_{i3} = x_{i1}x_{i2}$.

1.2 Least Squares

1.2.1 Solution of least squares

We assume that we have a set of training data $((x_{1\cdot}, y1), \ldots, (x_{n\cdot}, y_n))$ from which to estimate the parameters β of regression equation (1). Each $x_{i\cdot} = (x_{i1}, x_{i2}, \ldots, x_{ip})$ is a vector of measurements for the i^{th} individual and y_i is the i^{th} output. The most popular estimation method is least squares, in which we choose the coefficients $\beta = (\beta_0, \ldots, \beta_p)$ that minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 . \tag{2}$$

We can show that the unique solution is

$$\hat{\beta} = (X^{\,T}X)^{-1}X^{\,T}y, \tag{3}$$

where X is the *design* matrix

$$X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \text{ and } y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

The regression model can also be expressed in matrix form as

$$y = X\beta + \varepsilon, \tag{4}$$

where $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$.

1.2.2 Coefficient of determination

We denote by y_i^* the fitted values from regression analysis

$$y_i^* = \hat{\beta}_0 + \sum_{j=1}^{p} x_{ij}\hat{\beta}_j$$

and by y^* the vector $(y_i^*)_{i=1,\ldots,n}$. The coefficient of determination is defined as

$$R^2 = \widehat{\text{Cor}}^2(y^*, y) = \frac{\widehat{\text{Cov}}^2(y^*, y)}{\sqrt{\widehat{\text{Var}}(y^*)\widehat{\text{Var}}(y)}}$$

where $\widehat{\text{Var}}$ denotes the sample variance,

$$\widehat{\text{Cor}}(y^*, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i^* - \overline{y_n^*})(y_i - \bar{y}_n)$$

the empirical correlation coefficient, $\overline{y_n^*}$ and $\bar{y}_n$ the sample mean of y^* and y_n, respectively. We can show that

$$R^2 = \frac{\widehat{\text{Var}}(y^*)}{\widehat{\text{Var}}(y)} = \frac{\widehat{\text{Var}}(y) - \widehat{\text{Var}}(\hat{\varepsilon})}{\widehat{\text{Var}}(y)}, \tag{5}$$

where $\hat{\varepsilon} = y - y^*$ is the vector of residuals from the regression. The numerator of equation (5) is called the *explained* variance. The coefficient of determination $R^2 \in [0, 1]$ is therefore interpreted as the proportion of variance that is explained by the p predictors of the regression. The residual variance is a biased estimator of σ^2; however, $\hat{\sigma}^2 = n\widehat{\text{Var}}(\hat{\varepsilon})/(n - p - 1)$ is an unbiased estimator.

Remark: regression as conditional expectation For this paragraph only, we assume that x_{ip} is not fixed but rather a realization of a random variable X_{ip} for each i and p. Assuming that Y_i has finite variance, it can be shown that the best approximation

$$\arg\min_f E[(Y_i - f(X_{i1}, \ldots, X_{ip}))^2]$$

of Y_i by a function $f(X_{i1}, \ldots, X_{ip})$ with finite variance is the function $E[Y_i|X_{i1}, \ldots, X_{ip}]$. Since the functional form of the conditional expectation of Y_i given $X_{i1}, \ldots, X_{ip}$ is unknown, we have to assume a particular parametric form for f. When f belongs to a parametric family $\{f_\beta\}_{\beta \in B}$, the regression model has the general form

$$Y_i = f_\beta(x_{i1}, \ldots, x_{i1}) + \varepsilon_i,$$

where ε_i is the residual term with $E[\varepsilon_i] = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma^2$. If f is assumed to depend linearly on its parameters β, we obtain the linear regression model of equation (1).

1.3 Gaussian model

So far, we did not assume any parametric model for the residuals. However, if we want to perform statistical hypothesis-testing, we have to resort to parametric approximations. The most standard parametric approach assumes the following Gaussian model

$$\varepsilon \rightsquigarrow \mathcal{N}(0, \sigma^2).$$

1.3.1 Multivariate Gaussian distributions

Some random vector $Z = (Z_1, \ldots, Z_n)$ has a multivariate Gaussian distribution iff its probability density function (PDF) has the form

$$f_{m,\Sigma}(z) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}} \exp\left[-\frac{1}{2}(z-m)^T\Sigma^{-1}(z-m)\right]$$

where m (n-dimensional vector) and Σ ($n \times n$ symetric positive definite matrix) are parameters. This is denoted by $Z \rightsquigarrow \mathcal{N}(m, \Sigma)$. In this case, $E[Z] = m$, $\forall (i,j)$ $\mathrm{cov}(Z_i, Z_j) = \Sigma_{i,j}$ and $Z_i \rightsquigarrow \mathcal{N}(m_i, \Sigma_{i,i})$. Moreover: (i) Z_i and Z_j are independent iff $\Sigma_{i,j} = 0$ (ii) for any deterministic matrix A and vector b, $AZ + b \rightsquigarrow \mathcal{N}(Am + b, A\Sigma A^T)$ (some components of $AZ + b$ may be degenerate Gaussian – in other words deterministic).

In the Gaussian regression model, the least square estimators have the following properties:

- $\hat{\beta} \rightsquigarrow \mathcal{N}(\beta, \sigma^2(X^TX)^{-1})$
- $\frac{n-p-1}{\sigma^2}\hat{\sigma}^2 \rightsquigarrow \chi^2_{n-p-1}$
- both statistics are independent.

1.3.2 F-test

F-test for a group of variables. With the Gaussian hypothesis, we can use the F-test to test if a group of variables, let us say the p'^{th} to the p^{th} variable ($p \geq p'$), significantly improves the fit of the regression. The null hypothesis is $H_0 : (\forall j \in [p', p], \beta_j = 0)$ against $H_1 : (\exists j \in [p', p], \beta_j \neq 0)$. The test statistic is the F statistic,

$$F = \frac{(RSS_{p'} - RSS_p)/(p - p')}{RSS_p/(n - p - 1)} \tag{6}$$

where RSS_p is the residual sum-of-squares of the larger model with $p+1$ parameters, and $RSS_{p'}$ is the same for the *nested* model with $p'+1$ parameters. The F statistic measures the decrease in residual sum-of-squares per additional parameter (the numerator of equation (6)), and it is normalized by an estimate of σ^2 (the denominator of equation (6)). Under H_0, the F statistic follows a Fisher distribution $F_{p-p',n-p-1}$ with $p-p'$ and $n-p-1$ degrees of freedom .

The result of a statistical test is nowadays reported using P-values. The P-value is the probability under the null hypothesis to have a test statistic more atypical—in the direction of the alternative hypothesis—than the observed one. Here, we will reject H_0 for large enough values of F

$$P\text{-value} = P(F \geqslant f),$$

where f is the F-statistic computed for the data and $F \rightsquigarrow F_{p-p',n-p-1}$. The null hypothesis is rejected when the P-value is small enough ($P < 5\%$, $P < 1\%$, $P < 0.1\%$, ..., depending on the context).

This test can be generalized to the null hypothesis that β belongs to an affine subspace E of $\mathcal{R}^{p+1}$ of dimension $p' + 1$, i.e. when β satisfies a given set of linear constraints. As an example, the null hypothesis could be $H_0 : \beta_1 = 2\beta_3$ against $H_1 : \beta_1 \neq 2\beta_3$. The test statistic is still given by equation (6), where $RSS_{p'}$ now refers to the residual sum-of-squares for the model estimated under the linear constraints. Note that if A is a $(p+1) \times (p'+1)$ matrix whose columns represent a basis of E, the least-square estimates that satisfy the linear constraints are provided by equation (3) using XA as design matrix .

F-test of the regression. There is an instance of the F-test that is of particular interest. It is the situation where the bigger model is the complete model that includes all explanatory variables whereas the nested model is the model with parameter β_0 only. This test is called the F-test of the regression and it asseses whether there is an effect of the predictors on the variable to predict Y. Formally, the null hypothesis is $H_0 : (\forall j \geqslant 1, \beta_j = 0)$ against $H_1 : (\exists j \geqslant 1, \beta_j \neq 0)$. We can show that the F statistic of the regression is

$$F = \frac{n-p-1}{p} \frac{R^2}{1-R^2}, \tag{7}$$

and it follows a Fisher distribution $F_{p,n-p-1}$ with p and $n-p-1$ degrees of freedom under H_0. The formula of the F statistic given in equation (7) can be derived using equations (6) and (5) (an excercise for you!) .

Here, we will reject H_0 for large enough values of R^2, i.e. large enough values of F so that the P-value is equal to $P(F \geqslant f)$, where f is the F-statistic computed for the data and $F \rightsquigarrow F_{p,n-p-1}$.

1.3.3 t-tests of the regression

We test the hypothesis that a particular coefficient is null. The null hypothesis is $H_0 : \beta_j = 0$, and the alternative one is $H_1 : \beta_j \neq 0$. The T statistic is

$$T = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}},$$

where v_j is the jth diagonal element of $(X^TX)^{-1}$. Because of the distributions of the maximum likelihood estimators, the T statistic follows under H_0 a Student's t distribution with $n-p-1$ degrees of freedom.

1.3.4 R Example

In this example, we regress the murder rate, obtained for each of the 50 US states in 1973, by two potential explicative variables: the assault rate and the percentage of urban populations.

```
Violent Crime Rates by US State

     This data set contains statistics, in arrests per 100,000 residents for assault,
     murder, and rape in each of the 50 US states in 1973.  Also given is the percent
     of the population living in urban areas.

Format:

     A data frame with 50 observations on 4 variables.

       [,1]  Murder    numeric  Murder arrests (per 100,000)
       [,2]  Assault   numeric  Assault arrests (per 100,000)
       [,3]  UrbanPop  numeric  Percent urban population
       [,4]  Rape      numeric  Rape arrests (per 100,000)
> mylm<-lm(Murder~UrbanPop+Assault,data=USArrests)
>summary(mylm)
lm(formula = Murder ~ UrbanPop + Assault, data = USArrests)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.207153   1.740790   1.842   0.0717 .
UrbanPop    -0.044510   0.026363  -1.688   0.0980 .
Assault      0.043910   0.004579   9.590 1.22e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 2.58 on 47 degrees of freedom
Multiple R-squared: 0.6634,Adjusted R-squared: 0.6491
F-statistic: 46.32 on 2 and 47 DF,  p-value: 7.704e-12
```

There is a significative effect of the explicative variables on the murder rate because the P-value of the F-test is 7.704×10^{-12}. Once the percentage of urban population has been accounted for, the assault rate has a significative effect on the murder rate because the P-value of the t-test is 1.22×10^{-12}. However, once the assault rate is accounted for, there is no additional effect of the percentage of urban population on the murder rate ($P = 0.0980$) unless we accept a large enough type I error (e.g. $\alpha = 10\%$).

1.4 Variable selection

Variable selection or *feature* selection consists of selecting a subset of relevant variables (or features) among the p initial variables. There are two main reasons to perform variable selection.

- The first is *prediction accuracy*: the least squares estimates often have low bias but large variance (see subsection 1.4.1) especially when the number of variables p is large, of the same order as the sample size n. Prediction accuracy can be improved by variable selection, which consists of choosing a subsample of the initial set of predictive variables.

- The second reason is *interpretation*. With a large number of predictors, we want to sacrifice some of the small details in order to get the 'big picture' carried by a smaller subset of variables that exhibit the strongest effects.

1.4.1 Bias-variance decomposition

Assume that we want to make a prediction for an input $x_0 = (x_{01}, \dots, x_{0p})$. Denoting by $\hat{f}$ the regression function $\hat{f}(x_0) = \hat{\beta}_0 + \sum_{j=1}^{p} x_{0j}\hat{\beta}_j$, the prediction error is given by

$$\begin{aligned} E_{\text{pred}}(x_0) &= E[(Y - \hat{f}(x_0))^2] & (8) \\ &= E[(f(x_0) + \varepsilon - \hat{f}(x_0))^2] & (9) \\ &= \sigma^2 + \text{bias}^2\left(\hat{f}(x_0)\right) + \text{Var}\left(\hat{f}(x_0)\right) & (10) \\ &= \text{irreducible error} + \underbrace{\text{bias}^2 + \text{variance}}_{\text{mean square error of } \hat{f}(x_0)} \end{aligned}$$

When p is large (e.g. large number of predictors, large-degree polynomial regression...), the model can accommodate complex regression functions (small bias) but because there are many β parameters to estimate, the variance of the estimates may be large. For small values of p, the variance of the estimates gets smaller but the bias increases. There is therefore an optimal value to find for the number of predictors that reaches the so-called *bias-variance tradeoff*.

To give a concrete example, assume that we make a polynomial regression $f(x) = \beta_0 + \beta_1 x + \cdots + \beta_p x^p$ with $p = n - 1$. Since $p = n - 1$, the regression function will interpolate all of the n points. This is not a desirable feature and it suffers from *overfitting*. Rather than fitting all the points in the training sample, the regression function should rather provides a good *generalization*, i.e. be able to make good predictions for new points that have not been used during the fit. This is the reason why choosing the model that minimizes the residual squared error computed from the whole dataset is not a reasonable strategy. In subsections 1.4.2 and 1.4.3, we present different methods to compute the prediction error so that we can choose a model that achieves optimal generalization.

1.4.2 Split-validation

If we are in a data-rich situation, a standard approach to estimate the prediction error is to randomly divide the dataset into two parts: a *training* set with n_1 datapoints, and a *validation* set with n_2 datapoints ($n_1 + n_2 = n$). The training set is used to fit the models, i.e. to provide estimates of the regression coefficients ($\hat{\beta}_0, \ldots, \hat{\beta}_p$) and the validation set is used to estimate the prediction error. The prediction error is estimated by averaging the squared errors over the points of the validation set

$$E_{\text{pred}} = \frac{1}{n_2} \sum_{(x_i, y_i) \in \text{Validation Set}} \left(y_i - (\hat{\beta}_0 + \sum_{j=1}^{p} x_{ij} \hat{\beta}_j) \right)^2 . \tag{11}$$

We typically choose the model that generates the smallest prediction error.

1.4.3 Cross-validation

The caveat of the split-validation approach is that the prediction error may severely depend on a specific choice of the training and validation sets. To overcome this problem, the following cross-validation approaches consider a rotation of the validation and training sets.

- **K-fold cross-validation.** The idea is to split the data into K parts. To start with, $K - 1$ subsets are used as the training sets and the remaining subset is used as a validation subset to compute a prediction error E^1_{pred} as given by equation (11). This procedure is repeated K times by considering each subset as the validation subset and the remaining subsets as the training sets. The average prediction error is defined as

$$E_{\text{pred}} = \frac{1}{K} \sum_{k=1}^{K} E^k_{\text{pred}}$$

- **Leave-one-out.** It is the same idea as K-fold cross validation with $K = n$. The prediction error is computed as

$$E_{\text{pred}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}_{-i}(x_{i\cdot}))^2, \tag{12}$$

where $\hat{f}_{-i}$ denotes the estimated regression function when the i^{th} point $x_{i\cdot}$ has been removed from the training dataset.

To show the potential of cross-validation, we consider the following polynomial example

$$\begin{aligned} Y &= X - X^2 + X^3 + \varepsilon, \qquad (13) \\ \varepsilon &\rightsquigarrow \mathcal{N}(0, \sigma^2 = 10^2). \end{aligned}$$

We simulate a dataset with $n = 100$ using the polynomial model of equation (13) (see the left panel of Figure 1). We then consider a polynomial regression to smooth the data

$$Y = \beta_0 + \beta_1 x + \cdots + \beta_p x^p + \varepsilon$$

We consider a range of polynomial degrees $p = 1, \ldots, 20$ for polynomial regression. In the right panel of Figure 1, we show the leave-one-out prediction error (equation (12)) and the residual squared error for the entire data set ($RSS(\hat{\beta})$ in equation (2)) as a function of the number of degrees. The residual squared error is a decreasing function of the number of degrees so that minimizing naively the residual squared error would lead to the selection of the most complex model. The prediction error has a nonmonotonous behavior and points to a polynomial regression of degree 3, which is consistent with the generating mechanism given in equation (13). We provide below the R code that we used to generate Figure 1.

```
###########3 Polynomial regressions with different degrees

######Scatterplot and plot of the 3 polynomial regressions (deg=1,3,20)

xx<-seq(-5,5,length=100)

yy<-xx-xx^2+xx^3+rnorm(100,mean=0,sd=20)

plot(xx,yy,xlim=c(-7,7),ylim=c(-200,200))

###########################

# To save the figure into "fig.pdf", the command would be

# pdf('fig.pdf'), plot(xx,yy,xlim=c(-7,7),ylim=c(-200,200)), dev.off()

# pdf('fig.pdf') creates and opens the file, dev.off() closes it.

###########################

zz<-seq(-7,7,length=100)

degree<-c(1,3,20)

for(i in 1:3)
```

```
{ fm <- lm(yy ~ poly(xx, degree=degree[i],raw=T))
lines(zz, predict(fm, data.frame(xx=zz)), col = i,lwd=2) }
legend("top",col=1:3,legend=c("degree=1","degree=3","degree=20"),lwd=c(2,2,2))
###########Choice of the degree with cross-validation
#####First, compute the residual squared errors
####It is not the right criterion for model selection because the more complex model is chosen
maxdeg<-20
res<-vector(mode="numeric",length=maxdeg)
for (i in 1:20)
{ fm <- lm(yy ~ poly(xx, degree=i,raw=T))
res[i]<-mean( (fitted(fm)-yy)^2 ) }
############Compute the prediction errors with cross validation (10-fold cross validation)
mycross_val<-function(yy,xx,degree)
{res<-NULL
for (i in (1:10))
{ torm<-i+seq(0,90,by=10);aux<-xx[-torm]
fm <- lm(yy[-torm] ~ poly(aux, degree=degree,raw=T))
   res<-c(res,mean((predict(fm, data.frame(aux=xx))[torm] - yy[torm] )^2)) }
return (c(mean(res))) }
res2<-vector(mode="numeric",length=maxdeg)
for (i in 1:maxdeg)
{ aux<-mycross_val(yy,xx,i);res2[i]<-aux[1] }
###########Display the 2 errors as a function of the degree used in the polynomial regression
plot(1:maxdeg,res,pch=19,ylim=c(min(res,res2)-100,max(res,res2)+100),xlab="Degree",ylab="Error")
points(1:maxdeg,res2,pch=22,col=2,bg=2)
legend("top",c("Residual squared error","Prediction error"),pch=c(19,22),col=1:2,pt.bg=1:2,cex=1.5)
```

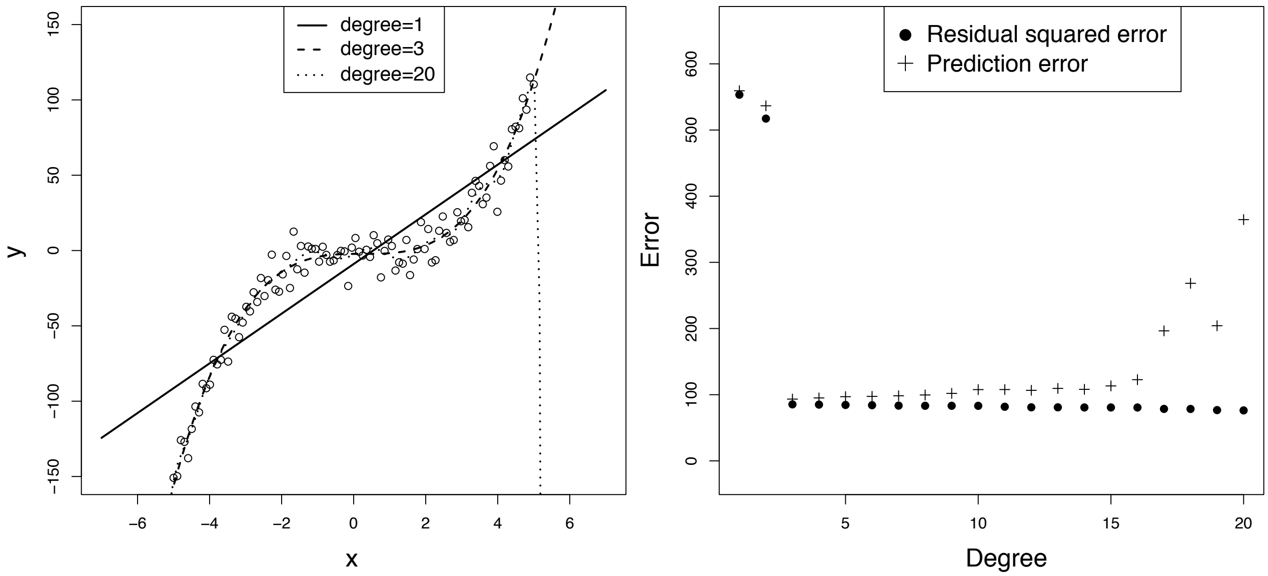

Figure 1: Polynomial regression for artificial data points. The data points are generated with the model given in equation (13). The prediction error is estimated with a leave-one-out cross-validation method.

1.4.4 Akaike information criterion (AIC)

The computation of a prediction error based on cross-validation is time consuming because the regression function must be fitted several times. Alternative criteria, such as the Akaike Information Criterion (AIC), do not require several trainings of the regression function. Model selection by AIC picks the model that minimizes

$$\text{AIC} = n \log \frac{RSS(\hat{\beta})}{n} + 2p, \tag{14}$$

where $RSS(\cdot)$ is the residual sum of squares given in equations (2). The AIC is obtained as an approximation of an information-theoretic criterion. It achieves a tradeoff between model fit (1st term in equation (14)) and model complexity (2nd term in equation (14)).

1.4.5 Optimization algorithms

If there are p predictors, the number of potential subsets (i.e. regression models) to explore is 2^p and estimating a prediction error for all models might be impossible. Instead, greedy algorithms are usually considered. *Forward* algorithms start with one predictor, keep the best predictor according to one of the criteria proposed above, and add a second predictor if it decreases the prediction error. The procedure is continued until 1) it is not possible to decrease the prediction error or until 2) all predictors have been included. *Backward* algorithms work similarly but start with the complete regression model that includes all predictors $(X_1, \ldots, X_p)$ and remove the predictors one after another. Using the `USArrests` example, we show how to perform in `R` the backward algorithm with AIC.

```
> mylm<-lm(Murder~.,data=USArrests)
#####By default, the step function proceeds backward
> stepAIC(mylm)
Start:  AIC=98.39
Murder ~ Assault + UrbanPop + Rape
           Df Sum of Sq    RSS     AIC
- Rape      1     8.041 312.87  97.689
<none>                  304.83  98.387
- UrbanPop  1    25.503 330.33 100.404
- Assault   1   300.020 604.85 130.648
Step:  AIC=97.69
Murder ~ Assault + UrbanPop
           Df Sum of Sq    RSS     AIC
<none>                  312.87  97.689
- UrbanPop  1     18.98 331.85  98.633
- Assault   1    612.18 925.05 149.891
Call:
lm(formula = Murder ~ Assault + UrbanPop, data = USArrests)
Coefficients:
(Intercept)      Assault     UrbanPop
    3.20715      0.04391     -0.04451
```

2 Analysis of variance (ANOVA)

Quoting Wikipedia (`http://en.wikipedia.org/wiki/Analysis_of_variance`),

> Analysis of variance (ANOVA) is a collection of statistical models, and their associated procedures, in which the observed variance in a particular variable is partitioned into components attributable to different sources of variation. In its simplest form ANOVA provides a statistical test of whether or not the means of several groups are all equal, and therefore generalizes t-test to more than two groups.

2.1 Introduction to one-way ANOVA

In one-way ANOVA, we test if a qualitative variable called a *factor* has a significant effect on a quantitative variable. Generalizing this approach to two qualitative explicative variables is possible with two-way ANOVA as shown in subsection 2.5.

To give a concrete example, one-way ANOVA will provide a statistical framework to test if the Ensimag specialization (mathematical finance, telecommunications,...) influences the marks of the students in the course of introductory statistics (PMS, Principes et Méthodes Statistiques). The factor corresponding to the students' specialization can take $K = 7$ different *levels* (in 2009/2010, the levels were MMIS, TEL, ISI, MIF, SIF, SLE, m1rmosig). In ANOVA, we assume that the data—the marks in the example—are Gaussian

$$Y_k^i \rightsquigarrow \mathcal{N}(m_k, \sigma^2), i = 1, \ldots, n_k$$

where m_k and n_k denotes the mean and the sample size for the data in the k^{th} level. We test the null hypothesis $H_0 : m_1 = \ldots = m_K$ against the alternative one $H_1 : \exists k, l \quad m_k \neq m_l$.

2.2 The F-test for one-way ANOVA

We denote by S_W^2 the residual variance or the variance *within* groups. It is defined by

$$S_W^2 = \frac{1}{n} \sum_{k=1}^{K} \sum_{i=1}^{n_k} (Y_k^i - \bar{Y}_k)^2,$$

where $\bar{Y}_k$ denotes the empirical mean for the data in the k^{th} level. We denote by S_B^2 the variance that is attributable to the factor or the variance *between* groups. It is defined by

$$S_B^2 = \frac{1}{n}\sum_{k=1}^{K} n_k(\bar{Y} - \bar{Y}_k)^2.$$

We finally denote by S^2 the total variance

$$S^2 = \frac{1}{n}\sum_{k=1}^{K}\sum_{i=1}^{n_k}(Y_k^i - \bar{Y})^2.$$

The total variance can be partitioned as

$$S^2 = S_B^2 + S_W^2. \tag{15}$$

Using equation (15), it can be shown that under the null hypothesis, the F-statistic

$$\frac{S_B^2/(K-1)}{S_W^2/(n-K)} \rightsquigarrow F_{K-1,n-K},$$

where F_{df_1,df_2} is the Fisher distribution with df_1 and df_2 degrees of freedom. The P-value of the test equals

$$P\text{-value} = \Pr(F \geqslant f), \text{ where } F \rightsquigarrow F_{K-1,n-K},$$

and f is the F-statistic computed for the data.

2.3 ANOVA in R

In R, the one way ANOVA to test if the students' specialization influences their mark in introductory statistics (PMS) is performed as follows

```
#Read the data
> alldata<-read.table("pms_anova.data",header=TRUE)
> #What is in the data
> names(alldata)
[1] "marks"          "corrector"      "specialization"

> #E.g. first mark in the list
> alldata[1,]
```

```
marks corrector specialization
1  14.5         1           MMIS
#convert the column corresponding to the specialization into a factor
> alldata[,3]<-factor(alldata[,3])
#Perform anova
> anova(lm(marks~specialization,data=alldata))
Analysis of Variance Table
Response: marks
                Df  Sum Sq Mean Sq F value   Pr(>F)
specialization   6  245.84  40.973   3.601 0.002097 **
Residuals      190 2161.83  11.378
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Since $P = 0.2\%$, there is a significant effect of the student's specialization on the marks in introductory statistics. To interpret the results of ANOVA, it is often useful to compute empirical means (together with the n_k's, referred to as `rep`) for each level of the factor, which is obtained by:

```
> model.tables(aov(marks~specialization,data=alldata), type="means")
Tables of means
Grand mean
11.30711
 specialization
      ISI m1rmosig   MIF  MMIS   SIF    SLE   TEL
    10.79        9 12.85 11.67 10.25  8.731 10.21
rep 50.00        1 43.00 64.00 14.00 13.000 12.00
```

2.4 Estimation of the effects

The model of one-way ANOVA can be written with a linear model as follows

$$\begin{aligned} Y_k^i &= \mu + \alpha_k + \varepsilon_k^i \qquad (16) \\ \varepsilon_k^i &\rightsquigarrow \mathcal{N}(0, \sigma^2). \end{aligned}$$

The model is not identifiable because we clearly have $\mu + \alpha_k = \bar{y}_k$, $k = 1, \ldots, K$, so that there is an infinite number of solutions for the $(K+1)$ parameters $(\mu, \alpha_1, \ldots, \alpha_K)$. To solve this problem, we usually add the constraint (also called contrast) that the mean of the effects is null $\sum_{k=1}^{K} n_k \alpha_k = 0$, so that the least squares solution is

$$\begin{aligned} \hat{\mu} &= \bar{y} \\ \hat{\alpha}_k &= \bar{y}_k - \bar{y}. \end{aligned}$$

With this constraint, the α_k's measure the difference between the mean for the data in the k^{th} level and the global mean. By default in R, fullpagethe contrast is $\alpha_1 = 0$ rather than $\sum_{k=1}^{K} n_k \alpha_k = 0$, which leads to a different interpretation of the other α_k's.

Using matrix notations, the model of one-way ANOVA can be described with a regression model of the same form as equation (4)

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ \vdots & & & & \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \mu \\ \alpha_1 \\ \vdots \\ \alpha_K \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

The matrix that contains the 0's and 1's is the design matrix X of the regression. In the i^{th} raw and $(j+1)^{\text{th}}$ column of the matrix, there is a 1 if the factor of the i^{th} individual corresponds to the j^{th} level and a 0 otherwise. The 2nd to the $(p+1)^{\text{th}}$ columns of X correspond to the so-called *dummy* variables. To account for the constraint $\sum_{k=1}^{K} n_k \alpha_k = 0$, we can suppress one of the column of X and replace all the remaining columns by their difference with the removed column. The results established for multiple regression now hold with the design matrix X, which contains the 0's and 1's. For instance, the least-squares solution of ANOVA is given by equation (3) and testing if the α_k's are significantly different from 0 can be done with the t-test.

Analyzing in R the effects of the different specializations produces the following output

```
#Read the data file

>load("pms_anova.data")

#We set the constraints/contrasts

> options(contrasts=c("contr.sum","contr.sum"))

#What are the different levels

> levels(alldata$specialization)

[1] "ISI"      "MIF"      "MMIS"     "SIF"      "SLE"      "TEL"      "m1rmosig"

#Estimation of the effects

> summary(lm(marks~specialization,data=alldata))

Call:

lm(formula = marks ~ specialization, data = alldata)

Residuals:

Min     1Q Median     3Q    Max

-9.790 -2.231  0.000  2.250  8.328

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept)      10.5000     0.5474  19.181  < 2e-16 ***

specialization1   0.2900     0.6799   0.427 0.670159

specialization2  -1.5000     2.9029  -0.517 0.605957

specialization3   2.3489     0.6991   3.360 0.000942 ***

specialization4   1.1719     0.6532   1.794 0.074386 .

specialization5  -0.2500     0.9382  -0.266 0.790186

specialization6  -1.7692     0.9617  -1.840 0.067375 .

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 3.373 on 190 degrees of freedom
Multiple R-squared: 0.1021,     Adjusted R-squared: 0.07375
F-statistic: 3.601 on 6 and 190 DF,  p-value: 0.002097
```

The two values of the linear model that are significantly different from 0 ($P < 5\%$) are

- The intercept: the mean μ of the marks is unsurprisingly significantly different from 0 ($P < 2e-16$).
- The parameter α_2 : the mean of the marks for the students with the MIF specialization (mathematical finance) is significantly larger than the average marks ($P = 0.1\%$).

Note that the variance σ^2 is assumed to be constant across groups in ANOVA (equation (16)). Although this assumption should be in principle tested before performing ANOVA (using for instance the Bartlett test), it is actually rarely tested.

2.5 Two-way ANOVA

In two-way ANOVA, we test simultaneously the effect of two qualitative variables on a quantitative variable. We will not give the mathematical details of two-way ANOVA. The model is simply defined as

$$Y_{k,l}^i \rightsquigarrow \mathcal{N}(m_{k,l};\sigma^2), \quad k = 1,\dots,K,\ l = 1,\dots,L,\ i = 1,\dots,n_{k,l},$$

where $m_{k,l} = \mu + \alpha_k + \beta_l + \gamma_{k,l}$, K and L are the number of levels of the two factors, and $n_{k,l}$ is the number of measurements for the couple of levels (k,l). We show that it can be easily performed using the anova function in R. We consider the same example of the marks in introductory statistics. In addition to the specialization of the students, we now consider the person who corrected the copies as a second factor. To test simultaneously the effect of the two factors we use the following command line. The command as.factor(alldata$corrector) ensures that the number indicating the professor is considered as a categorical (as opposed to numerical) variable. We first consider the model with additive effect of specialization and corrector: $m_{k,l} = \mu + \alpha_k + \beta_l$.

```
> alldata$corrector <- as.factor(alldata$corrector)
> anova(lm(marks~specialization+corrector,data=alldata))
Analysis of Variance Table
```

```
Response: marks
                Df  Sum Sq Mean Sq F value  Pr(>F)
specialization   6  245.84  40.973  3.6913 0.00173 **
corrector         5  108.36  21.672  1.9525 0.08772 .
Residuals      185 2053.47  11.100
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

The p-values result from a F-test for testing linear hypotheses (see Section 1.3.2). For instance, when testing the effect of specialization (the first factor), the null hypothesis is $H_0 : \alpha_1 = \cdots = \alpha_K = 0$. This two-way ANOVA analysis confirms that there is a significative effect of the specialization on the marks ($P = 0.2\%$). By contrast, we do not find a clear evidence that the identity of the corrector impacts the marks once the specialization has been accounted for ($P = 9\%$).

Interaction effects can be tested, the null hypothesis being $H_0 : \gamma_{1,1} = \cdots = \gamma_{K,L} = 0$.

```
> anova(lm(marks~specialization*corrector,data=alldata))
Analysis of Variance Table
Response: marks
Df  Sum Sq Mean Sq F value   Pr(>F)
specialization            6  245.84  40.973  3.6225 0.002043 **
corrector                 5  108.36  21.672  1.9161 0.093765 .
specialization:corrector  6   28.88   4.814  0.4256 0.861259
Residuals               179 2024.59  11.311
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here again, the test does not highlight any evidence in favour of interactions ($P = 86\%$). The average marks by pairs of factors can be obtained by

```
> model.tables(aov(marks~specialization*corrector,data=alldata), type="means")
Tables of means
Grand mean

11.30711

specialization
ISI m1rmosig   MIF  MMIS   SIF    SLE   TEL
10.79        9 12.85 11.67 10.25  8.731 10.21
rep 50.00        1 43.00 64.00 14.00 13.000 12.00

corrector
1     2     3    4     5     6
11.78 10.92 11.36 10.6 11.64 11.58
rep 32.00 36.00 32.00 32.0 33.00 32.00

specialization:corrector
corrector
specialization 1     2     3     4     5     6
ISI                               10.85 10.73
rep       0.00  0.00  0.00  0.00 26.00 24.00
m1rmosig  9.00
rep       1.00  0.00  0.00  0.00  0.00  0.00
MIF            18.00 13.14 11.81
rep       0.00  1.00 29.00 13.00  0.00  0.00
MMIS     12.33 11.13
rep      29.00 35.00  0.00  0.00  0.00  0.00
```

```
SIF                        10.25

rep       0.00  0.00  0.00 14.00  0.00  0.00

SLE       6.75        6.50  5.00  9.50 10.00

rep       2.00  0.00  1.00  1.00  3.00  6.00

TEL                   8.00  8.87 12.00 11.50

rep       0.00  0.00  2.00  4.00  4.00  2.00
```

3 Principal Component Analysis (PCA)

The data consist of n individuals and for each of these individuals, we have p measurements (or variables). The data are given in the following $n \times p$ design matrix

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}.$$

Quoting once again Wikipedia (`http://en.wikipedia.org/wiki/Principal_component_analysis`)

> Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated p variables into a set of values of d uncorrelated variables called principal components ($d \leq p$). This transformation is defined in such a way that the first principal component has as high a variance as possible (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (uncorrelated with) the preceding components.

3.1 Preliminary notations

3.1.1 Covariance and correlation matrices

We denote by c_{kl} the empirical covariance between the k^{th} and l^{th} variable

$$c_{kl} = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ik} - \bar{x}_{\cdot k})(x_{il} - \bar{x}_{\cdot l}).$$

The covariance matrix of the design matrix X is

$$\Sigma = \begin{pmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{p1} & \cdots & c_{pp} \end{pmatrix}.$$

Note that the diagonal of the covariance matrix contains the vector of the variances. Next, we denote by X' the $n \times p$ matrix obtained from the design matrix X after we have centered all of the columns (i.e. all of the columns of X' have a mean of 0), then we have

$$\Sigma = \frac{1}{n-1}X'^{\,T}X'.$$

Next, we define the empirical correlation between the k^{th} and l^{th} variable

$$r_{kl} = \frac{c_{kl}}{s_k s_l}, \text{ where } s_k = \sqrt{c_{kk}}\ .$$

Accordingly the correlation matrix can be written as

$$R = D^{-1}\,\Sigma\,D^{-1},$$

where

$$D = \begin{pmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & s_p \end{pmatrix}.$$

3.1.2 Eigenvectors and eigenvalues

We rank the p eigenvectors of the covariance matrix Σ by decreasing order of their corresponding eigenvalues. The first eigenvector, the column vector a_1, has the largest eigenvalue λ_1, the second eigenvector, the column vector a_2, has the second largest eigenvalue λ_2, etc.

The $(p \times p)$ matrix A of the eigenvectors is

$$A = [a_1\ a_2\ \dots\ a_p],$$

and the matrix of the eigenvalues is the diagonal matrix

$$\Sigma_Y = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_p \end{pmatrix}.$$

By definition of the matrix of the eigenvectors, we have

$$\Sigma = A\,\Sigma_Y\,A^T.$$

3.2 Solution of PCA

We can show that PCA can be done by eigenvalue decomposition of the data covariance matrix Σ. The first principal component corresponds to the eigenvector a_1. It means that a_1 defines the one-dimensional projection that maximizes the variance of the projected values. The second eigenvector a_2 defines the one-dimensional projection that maximizes the variance of the projected values among the vectors orthogonal to a_1, and so on.

The $(n \times p)$ matrix Y of the projected values on the principal component axes is

$$Y = X'A.$$

In the first column of Y, we read the projections of the data on the first principal component, in the second column of Y, we read the projections of the data on the second principal component, and so on.

3.3 Variance captured by the principal components

We can easily show that 1) the variance-covariance matrix of the projected values Y is given by Σ_Y, and that 2) the sum of the variances of the original data $\sum_{j=1}^{p} s_j^2$ is equal to the sum of the eigenvalues $\sum_{j=1}^{p} \lambda_j$. This means that the k^{th} principal component captures a fraction $\lambda_k / \sum_{j=1}^{p} \lambda_j$ of the total variance, and that the k first principal components capture a fraction $\sum_{j=1}^{k} \lambda_j / \sum_{j=1}^{p} \lambda_j$ of the total variance.

3.4 Scale matters in PCA

Assume that we record the following information for n sampled individuals: their weight in grams, their height in centimeters, their percentage of body fat and their age (in years). The corresponding design matrix X has n lines and $p = 4$ columns. If we perform PCA naively using the variance-covariance matrix Σ, it is quite clear that the first principal component will be almost equal to the first column (weight in grams). Indeed the choice of the unity of measure, gram here, is such that the variance of the first column will be extremely large compared to the variances of the remaining columns. The fact that the result of PCA depends on the choice of scale, or unity of measure, might be unsatisfactory in several settings. To overcome this problem, PCA may be performed using the correlation matrix R rather than using the variance-covariance matrix Σ. It is the same as performing PCA after having standardized all of the columns of X (i.e. the columns of the standardized matrix have an empirical variance equal to 1).

3.5 PCA in R

We consider the same `USArrests` dataset than in the section about multiple regression. This dataset contains four variables measuring three criminality measures and the percentage of urban population in each of the 50 US states in 1973. Using PCA, we can display the dataset in two dimensions (Figure 2).

```
## The variances of the variables in the
## USArrests data vary by orders of magnitude, so scaling is appropriate
res<-prcomp(USArrests)  # inappropriate
res<-prcomp(USArrests, scale = TRUE) # appropriate
## The matrix A of the eigenvectors
> res$rotation
               PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
> summary(res)
Importance of components:
                        PC1   PC2    PC3    PC4
Standard deviation     1.57 0.995 0.5971 0.4164
Proportion of Variance 0.62 0.247 0.0891 0.0434
Cumulative Proportion  0.62 0.868 0.9566 1.0000
##The following function displays the observations on the PC1, PC2 axes
## but also the original variables on the same axis (see Figure 2)
>biplot(res,xlim=c(-.3,.3),ylim=c(-.3,.3),cex=.7)
# To obtain a view of variables with appropriate metrics (correlation):
P = res$rotation
for(i in 1:4) P[,i] <- P[,i] * res$sdev[i]
```

```
plot(P[,1],P[,2], xlab=paste("Axis", 1), ylab = paste("Axis", 2), xlim=c(-1,1), ylim=c(-1,1))

text(P[, i], P[, j]-0.05, names(USArrests))

abline(h=0, v=0)

x = (-100:100)/100; y = sqrt(1-x^2)

lines(x, y); lines(x, -y) # draw a circle
```

We see that the proportion of the total variance captured by the first 2 principal components is 0.86%. In Figure 2, it is tempting to interpret the first axis as a 'violence' axis and the second one as the 'Urban versus Rural' axis although this kind of interpretation should be done with caution.

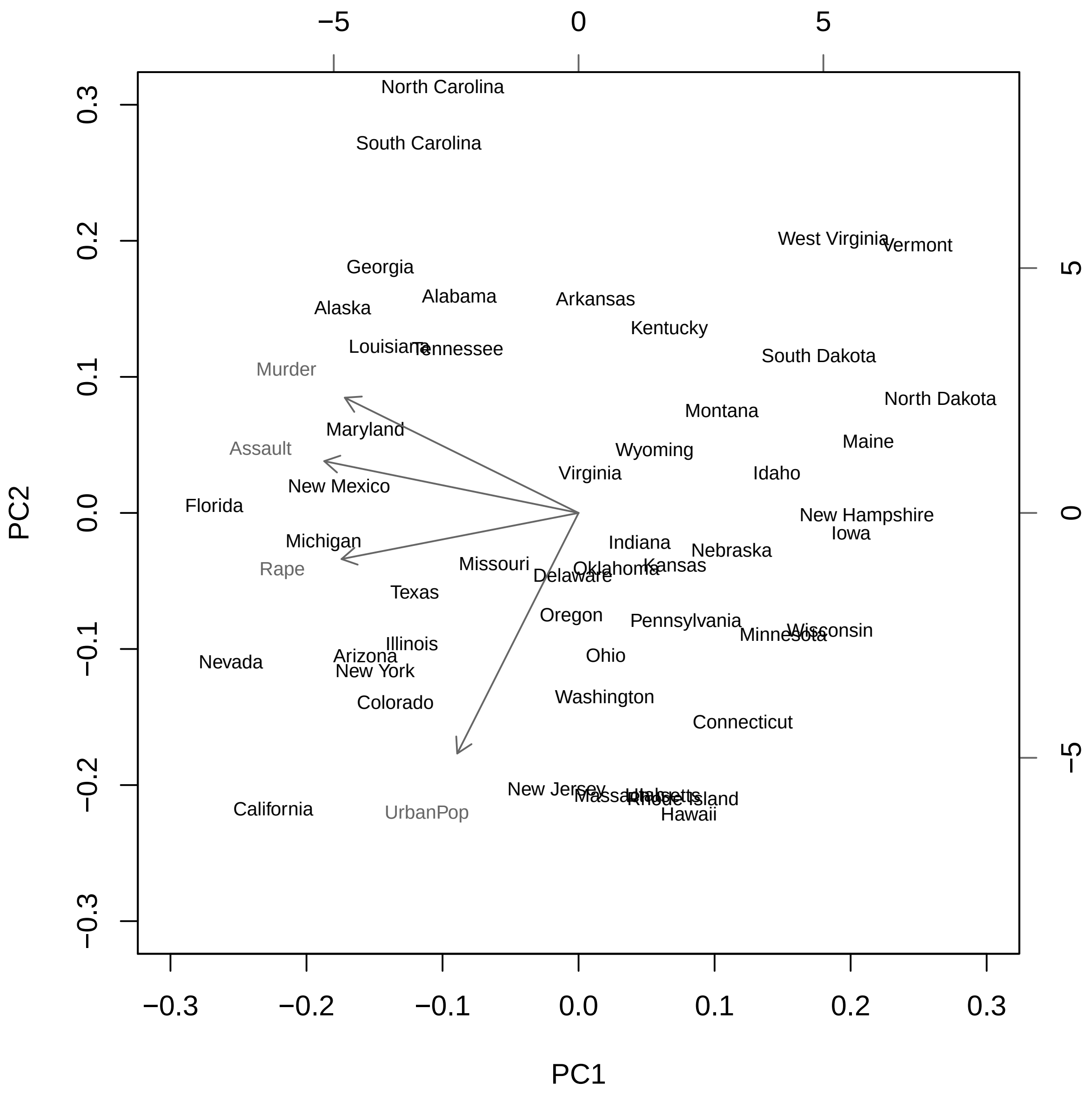

Figure 2: The 50 US states projected into the first two principal components. Here we use the biplot function in order to display additionally the original 4 variables to the PC1-PC2 graph.

4 Classification

4.1 Principles

4.1.1 The Classification problem

We introduce the classification problem with the classical example of handwritten digit recognition. The data from this example come from the handwritten ZIP codes on envelopes from U.S. postal mail. Each image is a segment from a five digit ZIP code, isolating a single digit. The images are 16×16 eight-bit greyscale maps, with each pixel ranging in intensity from 0 to 255. Some sample images are shown in Figure 3. The task is to predict, from the 16×16 matrix of pixel intensities, the identity of each image in $(0, 1, \dots, 9)$ (for sake of simplicity we consider only the 3 digits $(1, 2, 3)$ in the numerical example). If it is accurate enough, the resulting algorithm would be used as part of an automatic sorting procedure for envelopes.

Mathematically, the classifier is a function $\hat{G}$ that takes as input an image $X \in \mathcal{R}^p$ $(p = 16 \times 16)$ and returns as an output Y a digit in $\mathcal{G} = \{0, 1, \dots, 9\}$. In this example of image classification, each pixel corresponds to a variable so that an image is coded as a vector with 16×16 variables. To build the classifier, we have a training set $((x_{1\cdot}, y1), \dots, (x_{n\cdot}, y_n))$, where the data $x_{i\cdot} \in \mathcal{R}^p$ and the *label* $y_i \in \mathcal{G}$. We refer to binary classification when $\mathcal{G} = \{0, 1\}$ and more generally we have $\mathcal{G} = \{0, K\}$.

We do not address the related problem of clustering. Clustering occurs when the data—the x_i's — are not labelled in the training set. Classification is also referred as supervised learning and clustering as unsupervised learning.

4.1.2 Optimal/Bayes classifier

This subsection is quite technical and can be skipped. We assume to have a loss function L that measures the cost of replacing the true and unknown label Y with the label given by the classifier $\hat{G}(x)$. We want to minimize the expected loss function

$$C(x) = E[L(Y, \hat{G}(x))],$$

where E refers to the expectation with respect to the conditional distribution of Y given x. Using the law of total probability, we have

$$C(x) = \sum_{k=1}^{K} L(k, \hat{G}(x)) \Pr(Y = k | X = x).$$

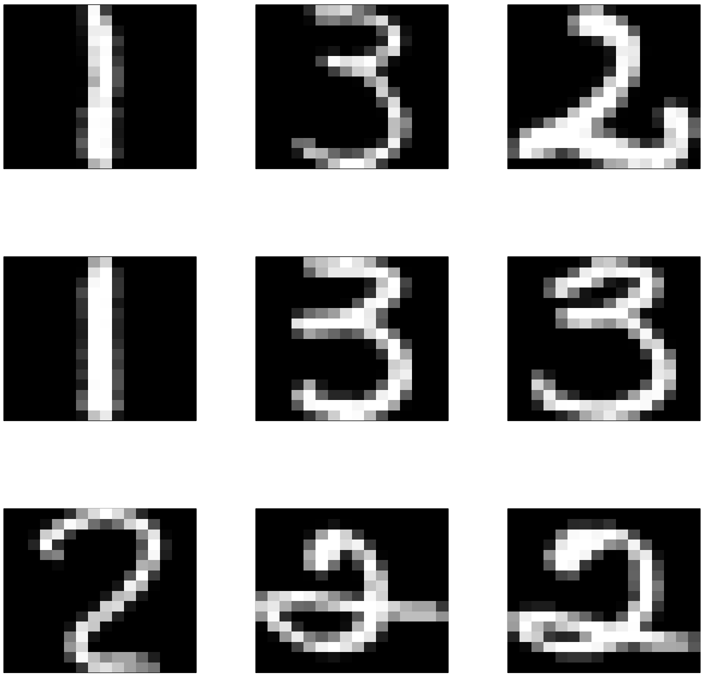

Figure 3: Examples of handwritten digits from U.S. postal envelopes. We consider only the 1, 2 and 3.

If we take the $0-1$ loss function that is equal to 0 when the classifier is right and 1 otherwise, we have

$$C(x) = 1 - \Pr(Y = \hat{G}(x)|X = x).$$

Minimizing the expected loss $C(x)$ amounts at choosing the class

$$\hat{G}(x) = \arg\max_{k \in \mathcal{G}} \Pr(Y = k|X = x)$$

This solution is known as the *Bayes classifier*, and says that we classify to the most probable class, using the conditional (discrete) distribution $\Pr(Y|X)$. Using the Bayes theorem

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k)\Pr(Y = k)}{\Pr(X = x)},$$

we find that the Bayes classifier maximizes the product of the conditional probability of the data given the class and the *prior* for the class

$$\hat{G}(x) = \arg\max_{k \in \mathcal{G}} \Pr(X = x|Y = k)\Pr(Y = k). \tag{17}$$

If we assume an uniform prior, the Bayes classifier proceeds as maximum likelihood that chooses the parameter—here the class— that makes the data x the most likely.

4.1.3 Evaluating the classification error

Evaluating the classification error is an important task to compare different classifiers and to asses the overall performance of the classifier. We can use the same methods as for regression: split-validation, K-fold cross-validation, leave-one-out cross-validation. For split-validation, we split the data into two parts, use one part of the data to build the classifier and use the other part to estimate the error. A standard criterion for evaluating the classification error is the misclassification rate.

4.2 Classic classifiers

4.2.1 k-nearest neighbor classifier

The k-nearest neighbor classifier is based on a simple counting procedure. A new individual x is classified by a majority vote of its neighbors, with the individual being assigned to the most common class amongst its k nearest neighbors ($k \geqslant 1$). A distance has to be defined in order to find the k-nearest neighbors. For image classification, a simple Euclidean distance between two images can be used although more elaborate versions of distances that are robust to deformations may be appropriate. If $k = 1$, then the individual is simply assigned to the class of its nearest neighbor. Using the methodology of subsection 4.1.3, we can choose the value of k that minimizes a misclassification rate estimated with a validation technique.

4.2.2 Linear discriminant analysis (LDA)

In LDA, we assume that the probability distribution of X given the class k is a Gaussian distribution $\mathcal{N}(\mu_k, \Sigma)$. We denote by f_k the p.d.f. of the multivariate Gaussian $\mathcal{N}(\mu_k, \Sigma)$

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)},$$

where $|\Sigma|$ denotes the determinant of Σ. What is particularly important in LDA is that we assume that the variance-covariance matrices are the same for all classes. Using the Bayes classifier given by equation (17), we assign an individual to the class k so that

$$k = \arg\max_{k \in \mathcal{G}} f_k(x)\pi_k,$$

where π_k is the prior for the class k. With standard algebra, we can show that it is equivalent to assigning x to the class k that maximizes the *linear discriminant functions*

$$\delta_k(x) = {}^T x \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k {}^T \Sigma^{-1} \mu_k + \log \pi_k.$$

To compute the linear discriminant functions, the μ_k's are estimated using the empirical means in each class and Σ is estimated as the (weighted) average of the covariance matrices estimated in each class.

In the following, we show how to perform LDA in R using the example of handwritten digit recognition.

```
####################Classification for the ZIP code data

require(MASS)

require(ElemStatLearn)

######Read the data

zippict<-zip.train

#######We keep only the 1, 2, and 3

bool<-(zippict[,1]==1 |zippict[,1]==2) | zippict[,1]==3

#bool<-(zippict[,1]==1 |zippict[,1]==2)

zipaux<-zippict[bool,]

#######Dimension of the data

n<-dim(zipaux)[1]

p<-dim(zipaux)[2]

######Display 9 images

par(mfrow=c(3,3),mar=c(2, 2, 4, 2)+0.1)

for (i in 10+(1:9)){mm<-(matrix(as.numeric((zipaux[i,2:257])),nrow=16,byrow=F))

image(z=mm[,16:1],col=grey((0:256)/256),xaxt="n",yaxt="n")}

####Definition of a data frame

###First column is the digit

data.zip<-data.frame(cbind(as.factor(zipaux[,1]),zipaux[,-c(1,16)]))

names(data.zip)[1]<-"number"

########We remove the pixels with a constant variance within each group
```

```
########Otherwise LDA returns an error message because the covariance matrix is not invertible
torm<-NULL
for (var in 1:3)
{thevar<-(zipaux[,1]==var)
for (i in 2:p)
{if(length(unique(zipaux[thevar,i]))==1 )
torm<-c(torm,i)}}
####Definition of a data frame
data.zip<-data.frame(cbind(as.factor(zipaux[,1]),zipaux[,-c(1,torm)]))
names(data.zip)[1]<-"number"
####Training set
train <- sample(1:n,floor(n/2))
table(data.zip$number[train])
###Linear discriminant analysis
z <- lda(number ˜ ., data.zip, prior = c(1,1,1)/3, subset = train)
####Prediction
myp<-predict(z, data.zip[-train, ])$class
####Counts of the images to classify
table(data.zip[-train,1])
  1     2      3
486    380   331
###Confusion matrix
myconf<-table(data.zip[-train,1],myp)
      1   2   3
  1 484   2   0
  2   0 372   8
  3   0   9 322
```

```
###Display of the confusion matrix (bar plot)

mycol<-c("black","grey40","lightgrey")

barplot(t(myconf),col=mycol)

legend("topright",paste(1:3),col=mycol,lwd=2,cex=2)

####Misclassification rate

1-sum(diag(myconf))/sum(myconf)

[1] 0.01587302
```

We consider a split-sample validation approach and find a misclassification rate of 1.6%. In addition to the misclassification rate, we compute in the variable `myconf` a confusion matrix, where each row of the matrix represents the instances in an actual class and each column represents the instances in a predicted class. For intense, there is a total of 486 images containing the digit 1 and 484 are correctly classified. One benefit of a confusion matrix is that it is easy to see if the classifier is confusing two classes, e.g. the digits 2 and 3 here. Figure 4 displays the confusion matrix using the `barplot` function.

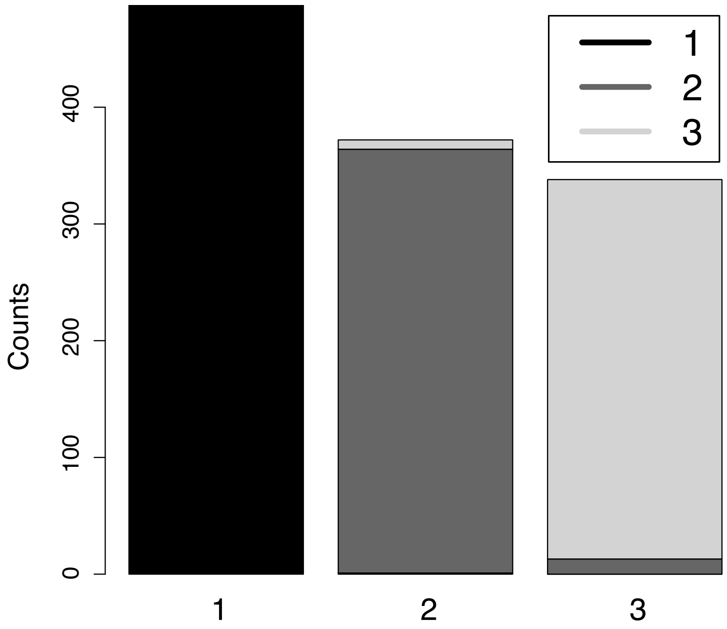

Figure 4: Confusion matrix for the LDA classifier. In each bar there are the instances in an actual class.

4.2.3 Quadratic discriminant analysis (QDA)

In QDA, we no longer assume that the variance-covariance matrices are the same for all classes. The densities f_k are now the p.d.f. of multivariate Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$, $k = 1, \ldots, K$. Using the Bayes classifier, we can show that we assign an individual x to the class k that maximizes the quadratic discriminant functions

$$\delta_k(x) = -\frac{1}{2}\log \Sigma_k - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k.$$

4.3 Logistic regression

Logistic regression is a useful way of describing the relationship between quantitative variables (e.g., age, weight, etc.) and a binary response variable that has only two possible values. Logistic regression models are used mostly as a data analysis and inference tool, where the goal is to understand the role of the input variables in explaining the outcome. It is widely used in biostatistical applications where binary responses (two classes) occur quite frequently. For example, patients survive or die, have heart disease or not, pass a test or not... Compared to discriminant analysis, logistic regression does not assign deterministically an individual x to the classes 0 or 1 but rather provides a *probabilistic* classifier because it computes $\mathrm{P}(Y = 0|X = x)$ and $\mathrm{P}(Y = 1|X = x)$.

Unsurprisingly, logistic regression relies on the logistic function (see Figure 5)

$$f(z) = \frac{1}{1 + e^{-z}},\ z \in \mathcal{R},$$

which maps an input value in $\mathcal{R}$ to an output confined between 0 and 1. The inverse of the logistic function is the *logit* function

$$g(p) = \log\frac{p}{1-p},\ p \in [0, 1],$$

and can therefore map a probability to a value between $-\infty$ and $+\infty$. In logistic regression, we use the logit function to link the probability $\mathrm{P}(Y = 1|X = x)$ to a linear combination of the explanatory variables

$$\log\frac{\mathrm{P}(Y = 1|X = x)}{1 - \mathrm{P}(Y = 1|X = x)} = \beta_0 + \sum_{j=1}^{p} x_j\beta_j, \tag{18}$$

and we assume a Bernoulli distribution for the response variables

$$Y_i \rightsquigarrow \mathrm{Ber}(\mathrm{P}(Y = 1|X = x_i)).$$

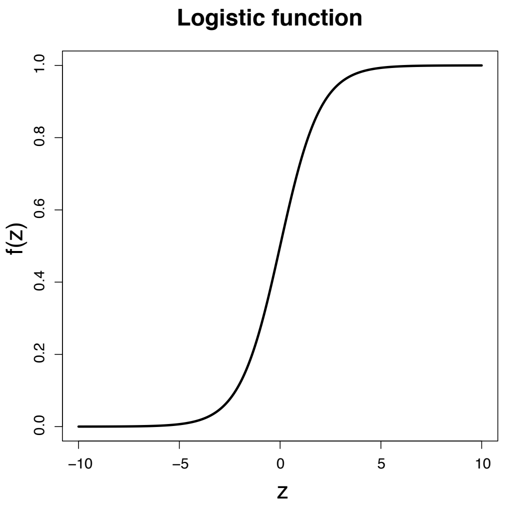

Figure 5: Graph of the logistic function.

Estimation of the β parameters can be obtained with maximum likelihood. Although there is no closed-form formula for finding the optimal β values, an optimization algorithm—derived from a Newton's method—called the *Fisher scoring algorithm* is usually used. Logistic regression can be extended to handle responses that are polytomous (more than two alternative categories).

In the following, we show how to use R to perform logistic regression. We consider a dataset where the response variable is the presence or absence of myocardial infarction (MI) at the time of the survey. There are 160 cases in the data set, and a sample of 302 controls. The aim of the study was to establish the intensity of some heart disease risk factors.

```
#Load the library where there are the data
#The library contains the examples from the book of Hastie et al. (2009)
> library(ElemStatLearn)
> ?SAheart
SAheart             package:ElemStatLearn              R Documentation
South African Hearth Disease Data
Description:
     A retrospective sample of males in a heart-disease high-risk
     region of the Western Cape, South Africa.

Usage:

     data(SAheart)

Format:

     A data frame with 462 observations on the following 10 variables.

     sbp systolic blood pressure

     tobacco cumulative tobacco (kg)

     ldl low density lipoprotein cholesterol

     adiposity a numeric vector

     famhist family history of heart disease, a factor with levels  ‘Absent’ ‘Present’

     typea type-A behavior

     obesity a numeric vector

     alcohol current alcohol consumption

     age age at onset

     chd response, coronary heart disease

Details:

     A retrospective sample of males in a heart-disease high-risk

     region of the Western Cape, South Africa. There are roughly two

     controls per case of CHD. Many of the CHD positive men have

     undergone blood pressure reduction treatment and other programs to

     reduce their risk factors after their CHD event. In some cases the

     measurements were made after these treatments. These data are

     taken from a larger dataset, described in Rousseauw et al, 1983,

     South African Medical Journal.

#Perform logistic regression with the glm function

# For more details about glm, look in Venables and Ripley (2002)

>myfit<-glm(chd˜.,data=SAheart,family=binomial())
```

```
> summary(myfit)
Call:
glm(formula = chd ~ ., family = binomial(), data = SAheart)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7781  -0.8213  -0.4387   0.8889   2.5435
Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)    -6.1507209  1.3082600  -4.701 2.58e-06 ***
sbp             0.0065040  0.0057304   1.135 0.256374
tobacco         0.0793764  0.0266028   2.984 0.002847 **
ldl             0.1739239  0.0596617   2.915 0.003555 **
adiposity       0.0185866  0.0292894   0.635 0.525700
famhistPresent  0.9253704  0.2278940   4.061 4.90e-05 ***
typea           0.0395950  0.0123202   3.214 0.001310 **
obesity        -0.0629099  0.0442477  -1.422 0.155095
alcohol         0.0001217  0.0044832   0.027 0.978350
age             0.0452253  0.0121298   3.728 0.000193 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)
Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 472.14  on 452  degrees of freedom
AIC: 492.14
Number of Fisher Scoring iterations: 5
```

The summary provided by R shows the Z-scores associated with the different variables. A Z-score is equal to the value of the β coefficient divided by its standard error and follows a standardized centered Gaussian variable under the null hypothesis ($\beta = 0$). Quoting Hastie et al. (2009),

> there are some surprises in this table of coefficients, which must be interpreted with caution. Systolic blood pressure (`sbp`) is not significant! Nor is obesity, and its sign is negative. This confusion is a result of the correlation between the set of predictors. On their own (i.e. simple linear regression), both `sbp` and `obesity` are significant, and with positive sign. However, in the presence of many other correlated variables, they are no longer needed (and can even get a negative sign). At this stage the analyst might do some model (variable) selection.

Let us do that in `R`

```
> stepAIC(myfit)->newfit

> summary(newfit)

Call:

glm(formula = chd ~ tobacco + ldl + famhist + typea + age, family = binomial(),

    data = SAheart)

Deviance Residuals:

    Min       1Q   Median       3Q      Max

-1.9165  -0.8054  -0.4429   0.9329   2.6139

Coefficients:

               Estimate Std. Error z value Pr(>|z|)

(Intercept)    -6.44644    0.92087  -7.000 2.55e-12 ***

tobacco         0.08038    0.02588   3.106  0.00190 **

ldl             0.16199    0.05497   2.947  0.00321 **

famhistPresent  0.90818    0.22576   4.023 5.75e-05 ***

typea           0.03712    0.01217   3.051  0.00228 **

age             0.05046    0.01021   4.944 7.65e-07 ***

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 475.69  on 456  degrees of freedom
AIC: 487.69

Number of Fisher Scoring iterations: 5
```

Here we see that the retained variables are: tobacco consumption, cholesterol, family history, type A behavior and age.

To understand what the coefficient means in logistic regression, we give the last words to Hastie and colleagues (2009).

> How does one interpret a coeffcient of 0.080 (Std. Error = 0.026) for tobacco, for example? Tobacco is measured in total lifetime usage in kilograms. Thus an increase of 1kg in lifetime tobacco usage accounts for an increase in the odds of coronary heart disease (the odds are $p/(1-p)$) of $e^{0.080} = 1.083$ or 8.4% (see equation (18)). Incorporating the standard error we get an approximate 95% confidence interval of $e^{0.080 \pm 2 \times 0.026} = (1.03, 1.14)$.

In the case of multi-class classification, logistic regression has an extension referred to as multinomial regression. The model is defined by equations

$$\log \frac{\mathrm{P}(Y = k|X = x)}{\mathrm{P}(Y = K|X = x)} = \beta_{0,k} + \sum_{j=1}^{p} x_j \beta_{j,k}, \tag{19}$$

where K is some arbitrary value of the class taken as a reference and $k \neq K$ is some other possible value for the class.

The `R` software contains an implementation of multinomial regression models in the `nnet` library (the models are recast as neural networks with no hidden layer). Training is achieved using the `multinom` command (same syntax as `lm`).

References

Chatfield C. and Collins AJ (1980) Introduction to multivariate analysis. Science paperbacks
Section 3 about PCA

Dalgaard, P (2003) Introductory statistics with R. 2nd ed. (English). Statistics and Computing. NY: Springer
Section 2 about ANOVA

Hastie T, Tibsirani R and Friedman J (2009) The Elements of Statistical Learning, 2nd ed, Springer. Available in electronic format at `http://www-stat.stanford.edu/~tibs/ElemStatLearn`
The holy book, sections 1, 3, and 4 mostly come from this book

Saporta G (1990) Probabilités, analyse des données et statistique, 1ère édition, Technip
Section 2 about ANOVA

Venables WN and Ripley BD (2003). Modern Applied Statistics with S, 4th Edition, Springer-Verlag, New York
The reference book for modern statistics in `S` *(or* `R` *that is almost the same)*

5 Frequent Itemset Mining

NB: For the next two sections the notes are succinct and should be only used for reference of main definitions and algorithms. They are only a complement to your class notes.

Goal: Find regularities, called *patterns*, in the data.

Applications: Market basket analysis,

5.1 Definitions

Let $I = \{i_1, ..., i_n\}$ be a set of elements called **items**. An **itemset** P is a set of items: $P \subseteq I$. A **dataset** $\mathcal{D}$ is a multiset of **transactions**, where a transaction is an itemset. More formally, $\mathcal{D} = \{t_1, ..., t_m\}$, where $\forall i \;\; t_i \subseteq I$. i is called the *transaction identifier* (**tid** for short) of transaction t_i.

Definition 1 (Tidlist) *Let $P \subseteq I$ be an itemset and $\mathcal{D}$ a dataset. The* ***tidlist*** *of P is the set of transactions of $\mathcal{D}$ containing P:*

$$tidlist_{\mathcal{D}}(P) = \{t \mid t \in \mathcal{D}, P \subseteq t\}$$

In the following, the tidlist of P will be denoted $tidlist(P)$ when $\mathcal{D}$ is clear from context.

Definition 2 (Support) *Let $P \subseteq I$ be an itemset and $\mathcal{D}$ a dataset. The support of P in $\mathcal{D}$, denoted $support_{\mathcal{D}}(P)$, is the number of transactions of $\mathcal{D}$ in which P appears:*

$$support_{\mathcal{D}}(P) = |tidlist_{\mathcal{D}}(P)|$$

In the following, the support of P will be denoted $support(P)$ when $\mathcal{D}$ is clear from context.

Definition 3 (Frequent itemset) *Let $P \subseteq I$ be an itemset and $\mathcal{D}$ a dataset. Given a* frequency threshold *$\varepsilon \in [0, m]$, P is* ***frequent*** *if $support(P) \geq \varepsilon$.*

From a frequent itemset P such as $P = X \cup Y$, one can derive an **association rule** $X \rightarrow Y$ that can lead to further insights on the data. The quality of such association rule is estimated with the measure of **confidence** :

Definition 4 (Confidence) *Let $X \subseteq I$, $Y \subseteq I$ such as $X \cap Y = \emptyset$ and $X \cup Y$ is a frequent itemset. Then $X \rightarrow Y$ is an association rule, its confidence is:*

$$conf(X \rightarrow Y) = \frac{support(X \cup Y)}{support(X)}$$

Similarly to frequency, it is possible to filter association rules with a minimum confidence threshold.

5.2 Apriori algorithm

Apriori is, historically, the first algorithm that allowed to efficiently mine frequent itemsets (1993). It performs a *levelwise* search in a breadth-first manner, and exploits an **anti-monotonicity** property of support:

Property 1 (Anti-monotonicity) *Any super-itemset of an infrequent itemset is infrequent.*

Algorithm 1: Apriori

```
   Data: Dataset $\mathcal{D}$, minimum support threshold $\varepsilon$
   Result: Output all frequent itemsets in $\mathcal{D}$
1  begin
2      $F_1 \leftarrow$ {Frequent 1-itemsets}
3      for $(k = 2\ ;\ F_{k-1} \neq \emptyset\ ;\ k++)$ do
4          $C_k \leftarrow apriori\_gen(F_{k-1})$
5          foreach transaction $d \in \mathcal{D}$ do
6              $c_d \leftarrow subset(C_k, d)$
7              foreach candidate $c \in c_d$ do
8                  $c.count++$
9          $F_k = \{c \in C_k \mid c.count \geq \varepsilon\}$
10     return $\cup_k F_k$

11 Function $apriori\_gen(F_{k-1})$
       Data: $F_{k-1}$: frequent $(k-1)$-itemsets
       Result: $C_k$: candidate $k$-itemsets
12     begin
13         insert into $C_k$
14         select $p[1], ...p[k-1], q[k-1]$
15         from $p, q \in F_{k-1}$
16         where $p[1] = q[1], ..., p[k-2] = q[k-2], p[k-1] < q[k-1]$
17         foreach itemset $c \in C_k$ do
18             foreach $(k-1)$-subset $s$ of $c$ do
19                 if $s \notin F_{k-1}$ then
20                     delete $c$ from $C_k$
```

In Algorithm 1, $subset(C_k, d)$ in line 6 returns all subsets of d that occur in C_k. Lines 17-20 exploit the anti-monotonicity property for pruning the set of candidates.

5.3 Backtracking algorithm

Due to its breadth-first search, Apriori's space complexity is exponential, which prevents it to handle large real-world datasets due to memory saturation. Many depth-first algorithms with polynomial space complexity have been proposed, to allow mining large datasets. They are all based on the schema of *backtracking algorithms*, presented in Algorithm 2. For this algorithm, we assume that there is a total order on I (such order can always be arbitrarily imposed). For any $P \subseteq I$, $max\_item(P)$ is the maximum item of P according to the order on I.

Algorithm 2: Backtracking

```
Data: Dataset $\mathcal{D}$, minimum support threshold $\varepsilon$
Result: Output all frequent itemsets in $\mathcal{D}$
begin
    $backtracking(\emptyset)$

Function $backtracking(P)$
    Data: Frequent itemset $P$
    Result: Output all frequent itemsets having $P$ has a prefix
    begin
        print $P$
        foreach $i \in I$, $i > max\_item(P)$ do
            if $support(P \cup \{i\}) \geq \varepsilon$ then
                $backtracking(P \cup \{i\})$
```

5.4 Closed frequent itemsets and the LCM algorithm

A problem of frequent itemsets is that they contain a high amount of redundancy. It is possible to reduce such redundancy *without loosing information* by considering **closed frequent itemsets**.

Definition 5 (Closed frequent itemsets) *Let $\mathcal{D}$ be a dataset, ε a minimum support threshold, and $\mathcal{F}$ the set of frequent itemsets on $\mathcal{D}$ with a support greater or equal to ε. A frequent itemset $P \in \mathcal{F}$ is* **closed** *if $\nexists Q \in \mathcal{F}$ s.t.: $P \subset Q$ and $support(P) = support(Q)$.*

Let $P \in \mathcal{F}$ be a frequent itemset. The **closure** of P, denoted $clo(P)$, is computed as:

$$clo(P) = \bigcap_{\{d \mid d \in \mathcal{D},\ P \subseteq d\}} d$$

$clo(P)$ is a closed frequent itemset.

The LCM algorithm is, to date, one of the most efficient algorithms to compute closed frequent itemsets. It's simplified pseudo-code is presented in Algorithm 3. It is based on the backtracking algorithm.

For further efficiency, this algorithm uses the concept of **dataset reduction**: for any pattern Q constructed at line 8, it is guaranteed to contain P. Thus to compute it's closure, one only needs the transactions of $\mathcal{D}$ that contain P. Such set of transaction is called a **reduced dataset** for itemset P and denoted $\mathcal{D}_P$. The construction of the reduced dataset of Q is shown on line 11.

6 Clustering

The goal of clustering is, given a dataset and a distance (or similarity) measure, to constitute groups of elements of the dataset where the distance between the elements within a group (intra-cluster distance) is small and the distance between elements of different groups (inter-cluster distance) is high.

A very well-known algorithm for this task is the **k-means** algorithm.

The input data $\mathcal{D}$ is a set of p-dimensional vectors $x_{i\cdot} = (x_{i1}, ..., x_{ip})$ with n data points. We suppose that we have a distance measure $dist(x_{i\cdot}, x_{j\cdot}) \in [0, 1]$ between these vectors.

The algorithm takes as parameter an integer $k \in [1, n]$ which indicates the expected number of clusters. It's simplified pseudo-code is given in Algorithm 4.

The algorithm is based on the notion of **centroid** of a group of points, which is the geometric center of these points.

Algorithm 3: LCM

```
Data: dataset $\mathcal{D}$, minimum support threshold $\varepsilon$
Result: Output all closed frequent itemsets in $\mathcal{D}$
begin
    $\emptyset_{clo} \leftarrow clo(\emptyset) = \bigcap_{t \in \mathcal{D}} t$
    print $\emptyset_{clo}$
    foreach $e \in I$ s.t. $support(e, \mathcal{D}) \geq \varepsilon$ do
        $expand(\emptyset_{clo}, e, \mathcal{D}, \varepsilon)$

Function $expand(P, e, \mathcal{D}_P, \varepsilon)$
    Data: Closed frequent itemset $P$, frequent item $e$, reduced dataset $D_P$, frequency threshold $\varepsilon$
    Result: Output all closed frequent itemsets containing $\{e\} \cup P$
    begin
        $Q \leftarrow clo(\{e\} \cup P)$
        if $max(Q \setminus P) = e$ then
            print $Q$
            $D_Q = \{t \mid t \in \mathcal{D}_P,\ Q \subseteq t\}$
            foreach $i \in I$ s.t. $support(i, D_Q) \geq \varepsilon$ and $i < e$ do
                $expand(Q, i, \mathcal{D}_Q, \varepsilon)$
```

Algorithm 4: k-means

```
Data: Dataset $\mathcal{D}$, $k \in [1, n]$
Result: A partitioning of $\mathcal{D}$ in $k$ clusters
begin
    Select $k$ points as the initial centroids
    repeat
        Form $k$ clusters by a assigning each point of $\mathcal{D}$ to its closest centroid
        Recompute the centroid of each cluster
    until the centroids don't change;
```

References

Tan P.N., Steinbach M., Kumar V. (2005) Introduction to Data Mining, Addison-Wesley.

An excellent reference book for all data mining tasks.

LCM algorithm: c.f. Takeaki Uno's web page: `http://research.nii.ac.jp/~uno/codes.htm`.

An efficient parallel version of LCM for multicore processors, made at LIG lab, is available here (in Java): `https://github.com/martinkirch/jlcm`

7 Document classification

In various practical cases of information search (IS), the aim of the analysis is to discriminate documents with respect to some determined set of information-related requirements. The most typical examples of such situations is spam detection, filtering of positive comments written on products or more generally, categorization of document collections (e.g., journal articles, emails, blog messages, internal documents in companies) using predefined categories. Up to the 80's, the main approaches to perform these tasks were *manual assignment* of class labels to documents and application of *simple human-defined rules* based on keywords characterizing specific topics and subjects. Both approaches, however efficient in terms of error rates on small-size collections, cannot be scaled and applied in real time on current text collections, which are now far larger. In this chapter, automatic document classification techniques are presented. These are issued from statistical learning theories, which have now become largely used in IS systems.

7.1 Generative models

Generative models were the first statistical learning models developed in IS for categorization tasks. Their fundamental assumption is that each vector **d** representing a document is the realization of some multivariate random variable, issued from a mixture of K PDFs with proportions (or *a priori* proportions) $\pi_1, ..., \pi_K$ satisfying:

$$\sum_{k=1} \pi_k = 1 \qquad \text{and} \qquad \forall k = 1, ..., K,\, \pi_k \geq 0.$$

Each PDF in this mixture of distributions is some parametric function interpreted as the conditional PDF of **d** given its class $k \in \mathcal{Y}$:

$$\forall k \in \mathcal{Y}, P(\mathbf{d} \mid C_k = 1) = f_k(\mathbf{d}, \theta_k),$$

where C_k is a binary random variable indicating membership ($C_k = 1$) or not ($C_k = 0$) of a document to class k. Thus, the mixture PDF results from generation of **d** by one of the K PDFs and is written as a convex combination of those:

$$P(\mathbf{d}, \Theta) = \sum_{k=1}^{K} \pi_k f_k(\mathbf{d}, \theta_k), \tag{20}$$

where Θ is the set of proportions π_k and of every parameter defining the f_k PDFs:

$$\Theta = \{\theta_k, \pi_k : k \in \{1, ..., K\}\}.$$

The aim of the learning step is thus to estimate parameter Θ so that the mixture models explains the training sample as well as possible. Once parameter estimation achieved, some new document d' is assigned to a class among the set $\mathcal{Y}$ using the Bayesian decision rule:

$$d' \text{ belongs to class } k \;\text{ iff }\; k = \arg\max_{h \in \mathcal{Y}} P(C_h = 1 \mid \mathbf{d}')$$

or equivalently, from Bayes' rule:

$$d' \text{ belongs to class } k \;\text{ iff }\; k = \arg\max_{h \in \mathcal{Y}} \pi_h f_h(\mathbf{d}', \theta_h). \tag{21}$$

In what follows, two generative models are introduced. These are the most frequently used and best known models in automatic document categorization. Both models assume conditional independence of descriptors given the class (Naive Bayes assumption) and thus, are simply referred to as *Naive Bayes* in the literature. Both models differ in the expression of their conditional PDFs, which determine their names.

7.1.1 Univariate Bernoulli model

In the univariate Bernoulli model, it is assumed that each document d has some binary vector representation $\mathbf{d} = (w_{id})_{i \in \{1,...,V\}}$, where each feature w_{id} is either 0 or 1, depending on whether the term with index i in the vocabulary is present or not in document d. From the *Naive Bayes* assumption, the conditional PDFs for document d write as:

$$\forall k \in \mathcal{Y}, f_k(\mathbf{d}, \theta_k) = P(\mathbf{d} = (w_{1d}, ..., w_{Vd}) \mid C_k = 1) = \prod_{i=1}^{V} P(w_{id} \mid C_k = 1).$$

This model is referred to as univariate Bernoulli since the events that represent absence of presence of every vocabulary term in the different classes follow some univariate Bernoulli distribution whose parameter is specific of the term and the class. Denoting as $\theta_{t_i|k} = P(w_{id} = 1 \mid C_k = 1)$ the probability for term i in class k to be present in d, the conditional PDFs write as:

$$\forall d; \forall k \in \mathcal{Y}, f_k(\mathbf{d}, \theta_k) = \prod_{i=1}^{V} \theta_{t_i|k}^{w_{id}} (1 - \theta_{t_i|k})^{1-w_{id}} \tag{22}$$

Here, parameter θ_k represents the set of probabilities for each term to be present, given the document being in class k (so-called mixture components). Parameter Θ of the Bernoulli model is defined as:

$$\Theta = \{\theta_{t_i|k} : i \in \{1, ..., V\}, k \in \{1, ..., K\}; \pi_k : k \in \{1, ..., K\}\}$$

Denoting as $\mathrm{df}_t(k)$ the number of documents in class k containing term t of vocabulary, as S_k the set of documents in class k within the training set and as $N_k(S) = |S_k|$ the cardinal number of S_k, the maximum likelihood equations for estimating parameters Θ of the Bernoulli model write as:

$$\forall i \in \{1, ..., V\}, \forall k \in \mathcal{Y}, \quad \hat{\theta}_{t_i|k} = \tfrac{\mathrm{df}_{t_i}(k)}{N_k(S)} \tag{23}$$

$$\forall k \in \mathcal{Y}, \quad \hat{\pi}_k = \tfrac{N_k(S)}{m}. \tag{24}$$

To apply the Bayesian decision rule (equation 21), the logarithm of $\pi_h f_h(\mathbf{d}', \theta_h)$ quantities is actually used. The main reason for this is that from the additive property of logarithms, computation of the probability product in equations 21 and 22 is transformed into computation of a sum of probability logarithms. This prevents underflow, since products of a high number of probabilities usually yield quantities that are to close to 0 to be stored with double precision. Moreover, the logarithm function is an increasing one-to-one mapping and thus, the order of scores computed in equation 21 is not changed by the transformation. Eventually in practice, to handle the case of rare terms that never appear in any document of a given class and lead to null presence probabilities (and thus to numerical errors when dealing with logarithms), probabilities are smoothed. This is most frequently achieved through Laplace smoothing, which consists in adding 1 to the number of documents of a class containing a given term, thus yielding:

$$\forall i \in \{1, ..., V\}, \forall k \in \mathcal{Y}, \hat{\theta}_{t_i|k} = \frac{\mathrm{df}_{t_i}(k) + 1}{N_k(S) + 2}. \tag{25}$$

Algorithm 5 presents the pseudo-code for parameter estimation in the Bernoulli model on some given training set S. Computation of the *a priori* and conditional probabilities are easily obtained by counting the number of documents per class, as well as the number of times each vocabulary term is present in a given class – using equations 23 and 25.

Once the parameters estimated, the vocabulary extracted from the training set is used to obtain the vector representation of a new document d', which class is to be predicted. Assignment of d' to some given class satisfies the rule associated with equation 21. From the conditional PDFs in equation 22 and using the logarithm transformation, the rule is rewritten as:

$$d' \in \text{class } k \ \text{ iff } \ k = \arg\max_{h \in \mathcal{Y}} \ln \hat{\pi}_h + \sum_{t \in \mathcal{V}, t \in d'} \ln \hat{\theta}_{t|h} + \sum_{t \in \mathcal{V}, t \notin d'} \ln(1 - \hat{\theta}_{t|h}).$$

This classification rule is used in algorithm 6. We now present the multinomial model, which is the other well-known model for automatic document categorization based on *Naive Bayes* assumptions.

7.1.2 Multinomial model

In the multinomial model, documents $d = \langle t_1, \ldots, t_{l_d} \rangle$ are considered as ordered sequences of terms generated from a given vocabulary $\mathcal{V}$. From the *Naive Bayes* assumptions (which state that term probabilities only

Algorithm 5: Univariate Bernoulli model, learning step

```
Input          :
    • some training set $S = ((\mathbf{d}_1, c_1), ..., (\mathbf{d}_m, c_m))$.
Initialization:
    • extract vocabulary $\mathcal{V} = \{t_1, \ldots, t_V\}$ from $S$;
    • initialize $N[.] \leftarrow 0$,
    • $df[.][.] \leftarrow 0$.
For each       : $k \in \{1, \ldots, K\}$
    $N[k] \leftarrow \sum_{d \in S_k} 1$; // $S_k = \{d \in S \mid d$ belongs to class $k\}$
    $Pi[k] \leftarrow \frac{N[k]}{m}$; // equation 23
    For each       : $i \in \{1, \ldots, V\}$
        $df[k][i] \leftarrow \sum_{d \in \text{class } k} w_{id}$ ; // $\mathbf{d} = (w_{id})_{i \in \{1,\ldots,V\}}, \forall i, w_{id} \in \{0,1\}$
For each       : $k \in \{1, \ldots, K\}$
    For each       : $i \in \{1, \ldots, V\}$
        $PC[k][i] \leftarrow \frac{df[k][i]+1}{N[k]+2}$ ; // equation 25
Output         : $Pi[.]$ and $PC[.][.]$ tables
```

Algorithm 6: Univariate Bernoulli model, test step

```
Input          :
    • some document $d'$, some vocabulary $\mathcal{V} = \{t_1, \ldots, t_V\}$;
    • $Pi[.]$ and $PC[.][.]$ tables obtained from algorithm 5.
Initialization:
    • get vector representation $\mathbf{d}' = (w'_{id})_{i \in \{1,\ldots,V\}}$ with $\mathcal{V}$;
    • initialize $PiF[.] \leftarrow 0$.
For each       : $k \in \{1, \ldots, K\}$
    $PiF[k] \leftarrow \ln(Pi[k])$ ;
    For each       : $i \in \{1, \ldots, V\}$
        If             : $w'_{id} = 1$
            $PiF[k] \leftarrow PiF[k] + \ln(PC[k][i])$;
        Else           :  $PiF[k] \leftarrow PiF[k] + \ln(1 - PC[k][i])$;
Output         : class of $d' = argmax_{h \in \mathcal{Y}} \, PiF[h]$
```

depend on the class – or mixture component – but not on the context nor position of that term within its document), the PDFs write as:

$$\forall k \in \mathcal{Y}, P(d = \langle t_1, \dots, t_{l_d} \rangle \mid C_k = 1) = \prod_{t_h \in d} \theta_{t_h|k}, \tag{26}$$

where $\forall h \in \{1, \dots, V\}, \theta_{t_h|k} = P(t_h \mid C_k = 1)$ represents the probability that term t_h in vocabulary is generated by mixture component k. These probabilities satisfy: $0 \leq \theta_{t_h|k} \leq 1$ and $\sum_{h=1}^{V} \theta_{t_h|k} = 1$. Although the parameters of this generative models are represented, as in the previous case, by the set:

$$\Theta = \{\theta_{t_i|k} : i \in \{1, ..., V\}, k \in \{1, ..., K\}; \pi_k : k \in \{1, ..., K\}\},$$

the multinomial model captures the information provided by term frequencies within documents and not only by some binary presence/absence information. Using the same notations as in the previous section, maximum likelihood parameter estimation with Laplace smoothing yields:

$$\forall i \in \{1, ..., V\}, \forall k \in \mathcal{Y}, \quad \hat{\theta}_{t_i|k} = \frac{\displaystyle\sum_{d \in S_k} \mathrm{tf}_{t_i,d} + 1}{\displaystyle\sum_{i=1}^{V} \sum_{d \in S_k} \mathrm{tf}_{t_i,d} + V} \tag{27}$$

$$\forall k \in \mathcal{Y}, \quad \hat{\pi}_k = \frac{N_k(S)}{m}. \tag{28}$$

To estimate conditional probabilities of the terms of some given class k (equation 27), the number of occurrences of every vocabulary term is counted from the documents of that class. In practice, it is more convenient to concatenate every document of a given class to obtain such counts. Algorithm 7 depicts the parameter estimation procedure in the multinomial model for the training step. Algorithm 8 depicts the test step.

Algorithm 7: Multinomial model, training step

```
Input :
    • some training set $S = ((\mathbf{d}_1, c_1), ..., (\mathbf{d}_m, c_m))$.
Initialization:
    • extract vocabulary $\mathcal{V} = \{t_1, \dots, t_V\}$ from $S$;
    • initialize $N[.] \leftarrow 0$, $tf[.][.] \leftarrow 0$, $D[.] \leftarrow 0$.
For each : $k \in \{1, \dots, K\}$
    $N[k] \leftarrow \sum_{d \in S_k} 1$; // $S_k = \{d \in S \mid d$ belongs to class $k\}$
    $Pi[k] \leftarrow \frac{N[k]}{m}$; // equation 28
    For each : $i \in \{1, \dots, V\}$
        $tf[k][i] \leftarrow \sum_{d \in S_k} \mathrm{tf}_{id}$ ;
        $D[k] \leftarrow D[k] + tf[k][i]$ ;
For each : $k \in \{1, \dots, K\}$
    For each : $i \in \{1, \dots, V\}$
        $PC[k][i] \leftarrow \frac{tf[k][i]+1}{D[k]+V}$ ; // equation 27
Output : $Pi[.]$ and $PC[.][.]$ tables
```

The R software contains an implementation of Naive Bayes models in the `naivebayes` library. Training is achieved using the `naive_bayes` command (same syntax as `lm`).

```
Algorithm 8: Multinomial model, test step
Input          :
    • some document $d'$, some vocabulary $\mathcal{V} = \{t_1, \ldots, t_V\}$;
    • $Pi[.]$ and $PC[.][.]$ tables obtained from algorithm 7.
Initialization:
    • get vector representation $\mathbf{d}' = (w'_{id})_{i \in \{1,\ldots,V\}}$ with $\mathcal{V}$;
    • initialize $PiF[.] \leftarrow 0$.
For each       : $k \in \{1, \ldots, K\}$
    $PiF[k] \leftarrow \ln(Pi[k])$ ;
    For each       : $i \in \{1, \ldots, V\}$
        $PiF[k] \leftarrow PiF[k] + w'_{id} \times \ln(PC[k][i])$;
Output         : class of $d' = \arg\max_{h \in \mathcal{Y}} PiF[h]$
```

Reference

Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008