Lecture 4 - Supervised Learning: Classification

Sondre Elstad

2022-06-25

Regression Continued

Before we proceed to the next supervised learning problem, there are some final aspects of linear regression which I would like to make. In the end of the last lecture, we did consider multiple regression. However, we only considered it in its most stylized form. All linear quantitative predictors. By now, you might be able to guess what comes next. Indeed, you are right: we are going to expand the multiple regression framework with nonlinear terms and qualitative predictors. First, note that there is a very crucial difference between these two models:

$$Y = \beta_0 + \beta_1 X + \beta_1^2 X \quad (A)$$

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 \quad (B)$$

Model (A) is non-linear in the regression coefficients. We can show that we cannot solve this model using OLS (ordinary least squares). Model (B), however, is nonlinear in the predictors. Here, there are no issues in the estimation. As such, whenever we talk about nonlinearities for these models, we are referring to the case in which there is nonlinearies in the predictors. We will not consider models that are nonlinear in the regression coefficients.

We might have qualitative predictors. In the stock market case, for example, we might consider another application in witch the response variable ('Up'/'Down') and one of the predictors switch places. In that case, we would have a qualitative predictor, and the way to deal with a qualitative predictor is to encode it. A dummy variable is a variable which takes one value if a certain condition is true, and another value if the condition is false. For instance, we could define the following dummy variable:

$$x_t = \begin{cases} 1 & \text{if stock market prices increases on day } t \\ 0 & \text{if stock market prices decreases on day } t \end{cases}$$

This predictor might readily be used in a regression. Doing so makes for the following model

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if stock market prices increases on day } t \\ \beta_0 + \epsilon_i & \text{if stock market prices decreases on day } t \end{cases}$$

Strictly speaking, this is a time series models. This closely resembles the cross-sectional regression models we have been looking at. However, there is a difference in data used. Time series data are observations which occurs in a recursive fashion. E.g., GDP measured at different time periods. Cross-sectional data is data for a range of different units measured at the same period of time. The main point here is that the dummy variable affects the value of the response variable. We can generalize this to the case where there are d different classes, by creating $d-1$ different dummies. An example illustrates this:

$$x_{i1} = \begin{cases} 1 & \text{if } i\text{th person is from the South} \\ 0 & \text{if } i\text{th person is not from the South,} \end{cases}$$

and

$$x_{i2} = \begin{cases} 1 & \text{if } i\text{th person is from the West} \\ 0 & \text{if } i\text{th person is not from the West,} \end{cases}$$

The resulting model is given by

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_1 & \text{if } i\text{th person is from the South} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i\text{th person is from the West} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is from the East.} \end{cases}$$

We have considered models that are nonlinear in the predictor itself. We have considered models with qualitative predictors. We can also consider models with multiplicative predictors - interaction terms. They allow us to incorporate effects that are dependent on the levels of covariates. For example:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon.$$

This can be rewritten as

$$Y = \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon = \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon$$

where $\tilde{\beta}_1 = \beta_1 + \beta_3 X_2$. A partial effects interpretation of each constant allows us to redefine regression coefficients in this way. The idea is that we keep X_2 constant when we change the level of X_1, and vice versa. Since X_2 is constant when we change X_1, $\tilde{\beta}_1$ will indeed be a constant when we change X_1 even though the coefficient depends on the level of X_2.

The final note I would like about linear regressions is that there are a number of threats we must contain in working with such models. In fact, these are symptomatic for more general threats which we must consider when working with any type of statistical model. Common problems which may occur when working with linear regression models are 1) non-linearity of the response-predictor relationships, 2) correlation of error terms, 3) non-constant variance of error terms, 4) outliers; 5) high-leverage points and 6) collinearity. We have already briefly talked about 1). If there are reasons to believe that the effect of some predictor on a response variable depends on the level of the predictor or the level of some other predictor, we should consider polynomial terms and interaction terms as necessary. We usually consider i.i.d. (identically and independently distributed) observations. If error terms are correlated, some observations might help predict other observations. If the variance of the error terms are non-constant, they are dependent on the observations. If so, there might be other statistical models which yields lower prediction errors than linear regression. Outliers are observations for which y_i is unusual given x_i and high-leverage points are observations for which x_i is unusual given y_i. Since linear regression models involve linear interpolation between observations, outliers and high-leverage points might severely affect the gradient of the linear model. Collinearity implies a very high correlation between some of the predictors. This can severely affect the estimated standard errors of the regression coefficients, and predictors who are in fact significant might come off as insignificant.

Classification - Logistic Regression

In the previous section, we considered dummy variables. We saw that the solution to qualitative predictors is to encode them (i.e., assigning a numerical value to each of its state). This is in fact the starting point for classification. The only difference is that we encode the response variable. Let's say that we encode

'Up' as 1 and 'Down' as 0. This 0-1 encoding gives a probabilistic interpretation to the statistical model. Given the set of predictors, we estimate the probability that y_i equals 1: $P(y_i = 1|X)$. Conceptually, there is nothing wrong with estimating this probability using a linear regression. From a mathematical point of view, however, there is one major concern: we might end up getting probabilities lower than zero or higher than one. In order to avoid this problem, we must consider a statistical model which only takes values between 0 and 1. This gives the statistical model a valid probabilistic interpretation. Logistic regression is such a model:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

$$\Rightarrow \frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}.$$

$$\Rightarrow \log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X.$$

This shows that the logistic regression model has a linear log-odds or logit. The logistic regression model cannot be solved analytically. This means that we cannot derive closed-form solutions for $\hat{\beta}_0$ and $\hat{\beta}_1$ as we did in the linear regression case. Rather, we must solve numerically for these estimators. We apply an algorithm based on the maximum likelihood principle for this purpose, although we are not going deeper into this now. I would only say that the maximum likelihood principle entails that we choose the estimator values that maximizes a probability distribution over observations which depends on these estimators. Put differently, we choose the values of $\hat{\beta}_0$ and $\hat{\beta}_1$ which maximizes the likelihood of observing the observations that we actually observe. However, all of the necessary details of this estimation is already implemented in R and so we do not need to concern ourselves with it.

We can extend this framework to the multiple regression case:

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p,$$

where $X = (X_1, \ldots, X_p)$ are p predictors. The probability of $Y = 1$ is then given by

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}.$$

The same maximum likelihood estimation method applies. A further problem with using the linear regression model for classification is that it is difficult to extend it to the case in which we have more than two classes. We can extend the logistic regression model to this case - this is called multinomial logistic regression.

The multinomial logistic regression framework typically implies choosing a baseline class (say, the Kth class). We estimate the following model:

$$Pr(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+...+\beta_{kp}x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+...+\beta_{lp}x_p}}$$

for $k = 1, \ldots, K - 1$, and

$$Pr(Y = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+...+\beta_{lp}x_p}}$$

The log-odds between two classes k and K is then given by

$$\log\left(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\right) = \beta_{k0} + \beta_{k1}x_1 + ... + \beta_{kp}x_p$$

Absolute estimates differ depending on the choice of baseline. However, the relative estimates remain the same and the fitted values (i.e., predictions) remain the same.

There is an alternative approach, called the softmax coding. Rather than selecting a baseline class, all K classes are treated symmetrically and we write

$$Pr(Y=k|X=x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+...+\beta_{kp}x_p}}{\sum_{l=1}^{K} e^{\beta_{l0}+\beta_{l1}x_1+...+\beta_{lp}x_p}}.$$

The log-odds between two classes is then given by

$$\log\left(\frac{Pr(Y=k|X=x)}{Pr(Y=k'|X=x)}\right) = (\beta_{k0}-\beta_{k'0}) + (\beta_{k1}-\beta_{k'1})x_1 + ... + (\beta_{kp}-\beta_{k'p})x_p.$$

R Lab (ISL 2017)

We will begin by examining some numerical and graphical summaries of the Smarket data, which is part of the ISLR2 library.

```
library(ISLR2)
names(Smarket)
```

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

```
dim(Smarket)
```

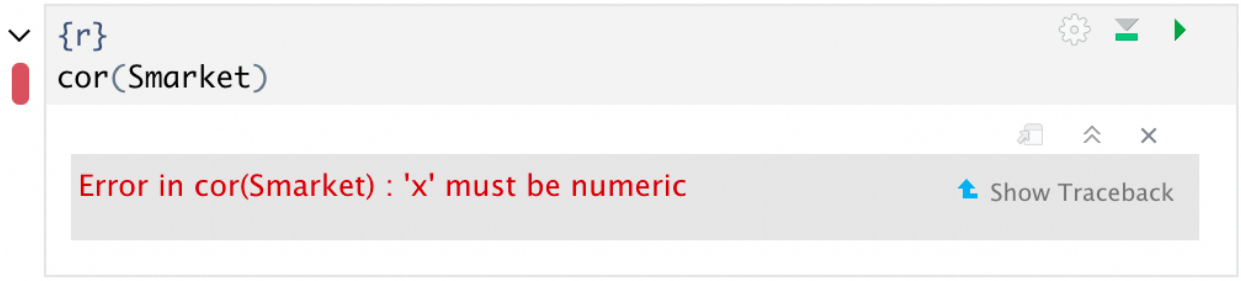
```
## [1] 1250    9
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2                Lag3
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000
##       Lag4                Lag5              Volume           Today
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000
##  Direction
##  Down:602
```

```
##  Up   :648
##
##
##
##
```

The cor() function produces a matrix that contains all of the pairwise correlations among the predictors in a data set. The first command below gives an error message because the Direction variable is qualitative.

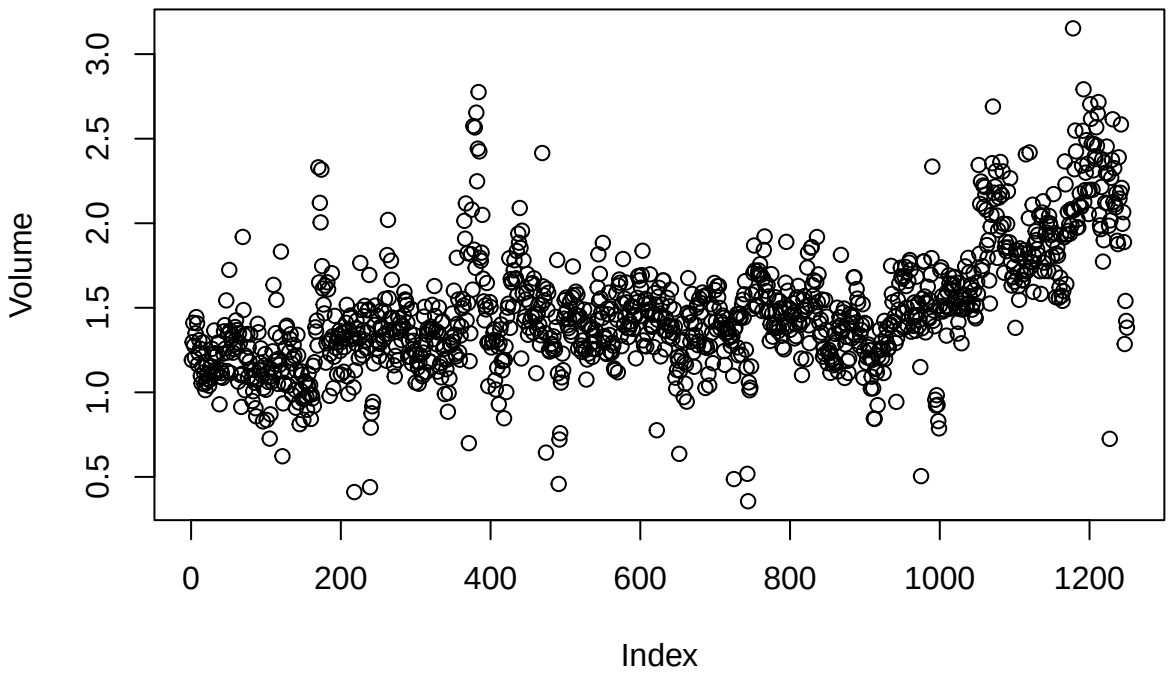



```
cor(Smarket[, -9])
```

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

The only substantial correlation is between Year and Volume. By plotting the data, which is ordered chronologically, we see that Volume is increasing over time.

```
attach(Smarket)
plot(Volume)
```



The glm() function can be used to fit many types of generalized linear models, including logistic regression. The syntax of the glm() function is similar to that of lm(), except that we must pass in the argument family=binomial in order to tell R to run a logistic regression rather than some other type of generalized linear model.

```
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket, family = binomial
)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
```

```
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

We use the coef() function in order to access just the coefficients for this fitted model.

```
coef(glm.fits)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
##       Volume
##  0.135440659
```

```
summary(glm.fits)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
summary(glm.fits)$coef[, 4]
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##   0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974
##      Volume
##   0.3924004
```

The predict() function can be used to predict the probability that the market will go up, given values of the predictors. The type = "response" option tells R to output probabilities of the form $P(Y = 1|X)$, as opposed to other information such as the logit. If no data set is supplied to the predict() function, then the probabilities are computed for the training data that was used to fit the logistic regression model. Here we have printed only the first ten probabilities.

```
glm.probs <- predict(glm.fits, type = "response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7         8
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
##         9        10
## 0.5176135 0.4888378
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

The following two commands create a vector of class predictors based on whether the predicted probability of a market increase is greater than or less than 0.5.

```
glm.pred <- rep("Down", 1250)
glm.pred[glm.probs > .5] = "Up"
```

Given these predictions, the table() function can be used to produce a confusion matrix in order to determine how many observations were correctly or incorrectly classified.

```
table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

```
mean(glm.pred == Direction)
```

```
## [1] 0.5216
```