Exercises 1

Bas Machielsen

2022-07-11

Exercise 1 - Logistic Regression

Use the Boston data (e.g. from the package MASS. Create a dependent dummy variable for house price which is is higher than 200K.

```
library(MASS); library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
boston <- MASS::Boston

boston <- boston %>%
  mutate(dummy = if_else(medv > 20, 1, 0))
```

Split the data (use set.seed(55)) into a train and a test set.

- I do this through Python

```
import random
import numpy as np
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import pandas as pd

r.boston.iloc[:,0:13]
```

```
##          crim    zn  indus  chas    nox  ...  rad    tax  ptratio   black  lstat
## 0     0.00632  18.0   2.31     0  0.538  ...    1  296.0     15.3  396.90   4.98
## 1     0.02731   0.0   7.07     0  0.469  ...    2  242.0     17.8  396.90   9.14
## 2     0.02729   0.0   7.07     0  0.469  ...    2  242.0     17.8  392.83   4.03
## 3     0.03237   0.0   2.18     0  0.458  ...    3  222.0     18.7  394.63   2.94
## 4     0.06905   0.0   2.18     0  0.458  ...    3  222.0     18.7  396.90   5.33
## ..        ...   ...    ...   ...    ...  ...  ...    ...      ...     ...    ...
## 501   0.06263   0.0  11.93     0  0.573  ...    1  273.0     21.0  391.99   9.67
## 502   0.04527   0.0  11.93     0  0.573  ...    1  273.0     21.0  396.90   9.08
## 503   0.06076   0.0  11.93     0  0.573  ...    1  273.0     21.0  396.90   5.64
## 504   0.10959   0.0  11.93     0  0.573  ...    1  273.0     21.0  393.45   6.48
```

```
## 505  0.04741   0.0  11.93     0  0.573  ...    1  273.0     21.0  396.90   7.88
##
## [506 rows x 13 columns]
```

```
X_train, X_test, y_train, y_test = train_test_split(
  r.boston.iloc[:,0:13], r.boston.iloc[:,14], test_size=0.5,
  random_state=55)
```

Based on the other variables in the data, predict whether a house would be priced over 200K or below.

```
clf = sm.Logit(y_train, X_train).fit()
```

```
## Optimization terminated successfully.
##          Current function value: 0.202906
##          Iterations 10
```

```
clf.summary()
```

```
## <class 'statsmodels.iolib.summary.Summary'>
## """
##                            Logit Regression Results
## ==============================================================================
## Dep. Variable:                  dummy   No. Observations:                  253
## Model:                          Logit   Df Residuals:                      240
## Method:                           MLE   Df Model:                           12
## Date:                 ma, 11 jul 2022   Pseudo R-squ.:                  0.7063
## Time:                        14:49:17   Log-Likelihood:                -51.335
## converged:                       True   LL-Null:                       -174.79
## Covariance Type:            nonrobust   LLR p-value:                 6.005e-46
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## crim          -0.3882      0.208     -1.869      0.062      -0.795       0.019
## zn             0.0368      0.027      1.349      0.177      -0.017       0.090
## indus          0.2023      0.070      2.907      0.004       0.066       0.339
## chas           3.1655      1.216      2.603      0.009       0.782       5.549
## nox           -7.6773      4.103     -1.871      0.061     -15.720       0.365
## rm             1.9573      0.542      3.609      0.000       0.894       3.020
## age           -0.0238      0.016     -1.446      0.148      -0.056       0.008
## dis           -0.4638      0.252     -1.837      0.066      -0.959       0.031
## rad            0.3629      0.101      3.580      0.000       0.164       0.562
## tax           -0.0137      0.005     -2.799      0.005      -0.023      -0.004
## ptratio       -0.1928      0.148     -1.302      0.193      -0.483       0.098
## black          0.0135      0.007      2.000      0.046       0.000       0.027
## lstat         -0.4401      0.090     -4.864      0.000      -0.617      -0.263
## ==============================================================================
##
## Possibly complete quasi-separation: A fraction 0.16 of observations can be
## perfectly predicted. This might indicate that there is complete
## quasi-separation. In this case some parameters will not be identified.
## """
```

Create a confusion matrix to evaluate the accuracy

```
def confusion_matrix(y_test,x_test, logit_model, threshold=0.5):
```

```python
predictions = logit_model.predict(x_test)

data = (pd.DataFrame(np.array([y_test, predictions]).
transpose(),
columns=['real','predict']))

data['predict'] = data['predict'] > threshold

predicted_no_actual_yes, predicted_yes_actual_yes = (
  data.groupby('predict')['real'].apply(lambda x:
    (x==1).sum()).reset_index(name='count').iloc[:,1])
predicted_no_actual_no, predicted_yes_actual_no = (
  data.groupby('predict')['real'].apply(lambda x:
    (x==0).sum()).reset_index(name='count').iloc[:,1] )


out = np.array([[predicted_yes_actual_yes, predicted_no_actual_yes],
[predicted_yes_actual_no, predicted_no_actual_no]])

b_out = (pd.DataFrame(out, index=['actual_yes', 'actual_no'],
columns=['predicted_yes', 'predicted_no']))
return b_out
```

```python
confusion_matrix(y_test, X_test, clf)
```

```
##             predicted_yes  predicted_no
## actual_yes            131            25
## actual_no               9            88
```

Activation Functions

- What is the role of the activation function?

To scale down the predicted values towards a reasonable range (e.g. $(0, 1)$).

- Code yourself the sigmoid function, plot it (say for the [-5,5] range)
- Code yourself the relu function, plot it.
- Code yourself the leaky relu function, plot it.
- Code yourself the swish function, plot it.
- Compute and plot the derivatives of those functions2.
- Recreate the softmax chart presented during the lecture.