Homework 3

AI 539 - Machine Learning for Non-AI Majors
Instructor: Alireza Aghasi
Due date: See Canvas

April 25, 2024

Please revise the homework guidelines reviewed in the first lecture. Specifically note that:

- Start working on the homework early
- Late homework **is not accepted** and will receive zero credit.
- Each student must write up and turn in their own solutions
- **(IMPORTANT)** If you solve a question together with another colleague, each need to write up your own solution and **need to list the name of people who you discussed the problem with on the first page of the material turned in**
- The homework should be manageable given the material lectured in the class. The long questions are to help clarifying the problem.

Q1. The goal of this problem is to implement your own version of logistic regression, and compare it to the output of the Python package.

(a) Load the data file `BinaryData.csv` and perform a simple logistic regression in the programming language of your choice, predicting the class y based on x. Report the values of β_0 and β_1.

```
 import numpy as np
 import statsmodels.api as sm
 import pandas as pd
 from matplotlib.pyplot import subplots
 import statsmodels.api as sm
 from ISLP import load_data
 from ISLP.models import (ModelSpec as MS,
 summarize)

data = open("BinaryData.csv")
myData = np.loadtxt(data, delimiter=",", skiprows=1)

# constructing the data matrix with a column of all ones for the intercept
X = pd.DataFrame({'intercept': np.ones(myData.shape[0]),
'x': myData[:,0]})
y = myData[:, 1]
glm_train = sm.GLM(y, X,
family=sm.families.Binomial())
results = glm_train.fit()
print(results.summary())
```

Program Output. .

```
Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   60
Model:                            GLM   Df Residuals:                       58
Model Family:                Binomial   Df Model:                            1
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -21.412
Date:                Thu, 25 Apr 2024   Deviance:                       42.824
Time:                        11:40:00   Pearson chi2:                     42.4
No. Iterations:                     6   Pseudo R-squ. (CS):             0.4284
Covariance Type:            nonrobust
==============================================================================
coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -0.7776      0.454     -1.711      0.087      -1.668       0.113
x              1.2088      0.318      3.807      0.000       0.586       1.831
==============================================================================
# Baased on which the value of beta_0 and beta_1 are
# beta0 = -0.7775995
# beta1 = 1.2088080
```

(b) Now lets work on implementing our own version of logistic regression, and understand its basics. To start, consider the function

$$f(z) = \alpha \log\left(1 + e^{-z}\right) + (1 - \alpha) \log\left(1 + e^{z}\right), \qquad 0 \leq \alpha \leq 1,$$

where α is a known coefficient between 0 and 1. Show that $z = \log(\frac{\alpha}{1-\alpha})$ is a stationary point (point of zero derivative) for $f(z)$.

Taking a derivative we have

$$\begin{aligned} f'(z) &= -\alpha \frac{e^{-z}}{1+e^{-z}} + (1-\alpha)\frac{e^z}{1+e^z} \\ &= -\alpha \frac{1}{1+e^z} + (1-\alpha)\frac{e^z}{1+e^z} \\ &= \frac{e^z}{1+e^z} - \alpha \end{aligned}$$

Setting $f'(z) = 0$ yields $e^z(1-\alpha) = \alpha$, which gives $z = \log(\frac{\alpha}{1-\alpha})$.

(b) Show that $f(z)$ is convex (the second derivative test might be the easiest).

Taking a derivative from $f'(z)$ gives us the second derivative as

$$\begin{aligned} f''(z) &= \frac{d}{dz}\left(\frac{e^z}{1+e^z} - \alpha\right) \\ &= \frac{e^z(1+e^z) - e^{2z}}{(1+e^z)^2} \\ &= \frac{e^z}{(1+e^z)^2} \end{aligned}$$

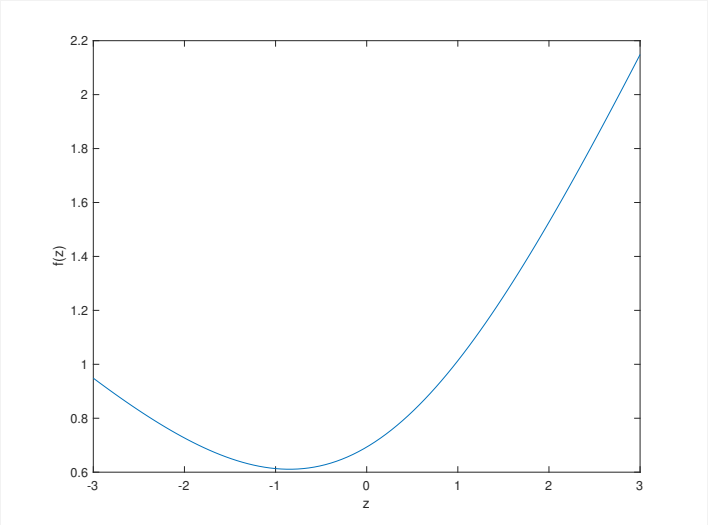
Since both e^z and the denominator are positive functions, we have $f''(z) > 0$, which implies that f is convex everywhere.

(c) Now that you know $f(z)$ is convex, is $z = \log(\frac{\alpha}{1-\alpha})$ a minimizer or a maximizer? Why?

Stationary points of smooth convex functions are minimizers.

(d) Plot $f(z)$ for $\alpha = 0.3$, and the values of z between -3 and 3.

See the plot below:



(e) In the class we learned that sum of convex functions is convex. Furthermore, we showed that if $f(z)$ is convex, $f(\beta_0 + \beta_1 x)$ is also convex. This is an indication that the logistic loss

$$L(\beta_0, \beta_1) = \sum_{i=1}^{n} y_i \log\left(1 + e^{-\beta_0 - \beta_1 x_i}\right) + (1 - y_i) \log\left(1 + e^{\beta_0 + \beta_1 x_i}\right).$$

is convex. Now, derive an expression for

$$\frac{\partial L}{\partial \beta_0} = \ldots, \qquad , \frac{\partial L}{\partial \beta_1} = \ldots.$$

Simplify the expressions in a way that the end results only involve `sigmoid` functions and not the log or exp functions. Expressions like $\sum_{i=1}^{n} w_i \texttt{sigmoid}(w_i') + w_i''$, where w_i, w_i', w_i'' are expressions in terms of the problem parameters.

Solution. We have

$$\begin{aligned}
\frac{\partial L}{\partial \beta_0} &= \sum_{i=1}^{n} -y_i \frac{e^{-\beta_0 - \beta_1 x_i}}{1 + e^{-\beta_0 - \beta_1 x_i}} + (1 - y_i) \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \\
&= \sum_{i=1}^{n} -y_i \frac{1}{1 + e^{\beta_0 + \beta_1 x_i}} + (1 - y_i) \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \\
&= \sum_{i=1}^{n} \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} - y_i \\
&= \sum_{i=1}^{n} \texttt{sigmoid}(\beta_0 + \beta_1 x_i) - y_i.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\frac{\partial L}{\partial \beta_1} &= \sum_{i=1}^{n} -x_i y_i \frac{e^{-\beta_0 - \beta_1 x_i}}{1 + e^{-\beta_0 - \beta_1 x_i}} + x_i (1 - y_i) \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \\
&= \sum_{i=1}^{n} -x_i y_i \frac{1}{1 + e^{\beta_0 + \beta_1 x_i}} + x_i (1 - y_i) \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \\
&= \sum_{i=1}^{n} x_i \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} - x_i y_i \\
&= \sum_{i=1}^{n} x_i \texttt{sigmoid}(\beta_0 + \beta_1 x_i) - x_i y_i.
\end{aligned}$$

(f) Use the data file `BinaryData.csv` in part (a) and set up $L(\beta_0, \beta_1)$ for the x_i and y_i in the dataset. Write a gradient descent (GD) scheme to minimize $L(\beta_0, \beta_1)$ in Matlab or Python. For your scheme use a learning rate of $\eta = 0.01$, and run the GD for 500 iterations. As the initial values for β_0 and β_1 you can use zero (clearly, since the problem is convex, the initialization does not matter and we will converge to the global minimizer no matter where we start). Attach all your code and results.

```
 import numpy as np
import statsmodels.api as sm
import pandas as pd
from matplotlib.pyplot import subplots
import statsmodels.api as sm
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
summarize)

data = open("BinaryData.csv")
myData = np.loadtxt(data, delimiter=",", skiprows=1)
y = myData[:, 1]

# Running gradient descent:
x = myData[:,0]
# initializing the variables
beta0 = 0
beta1 = 0
numIter = 300
eta = .01
# defining the sigmoid function
def sigmoid(x):
return 1/(1 + np.exp(-x))

# Runnign the gradient descent loop:
for i in range(numIter):
beta0 = beta0 - eta*np.sum(sigmoid(beta0+beta1*x)-y)
beta1 = beta1 - eta*np.sum(x*(sigmoid(beta0+beta1*x)-y))

print(beta0)
print(beta1)
```

Program Output.

```
The program output is
-0.7775971269224761
 1.2088067621992857
```

Q2. For a classification problem with two features (that is, our feature vector $\boldsymbol{x}$ has two components, i.e., $\boldsymbol{x} = (x_1, x_2)^\top$) the label value is binary (the values of the response only take two values). We use an LDA approach and after fitting an LDA model we obtain the following parameters:

$$\pi_1 = \frac{1}{3},\ \pi_2 = \frac{2}{3},\ \boldsymbol{\mu}_1 = \begin{pmatrix} -3 \\ 2 \end{pmatrix},\ \boldsymbol{\mu}_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \text{ and } \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} 5 & -2 \\ -2 & 2 \end{pmatrix}.$$

Show that the line that separates the two classes (the decision boundary) is the following line:

$$12x_2 - 27x_1 = 31.5 + \log(2) \tag{1}$$

In our solution we make use of the following simple facts:

- Simple matrix multiplication reveals that for a vector $\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ and a matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & \gamma \\ \gamma & \sigma_2 \end{pmatrix}$:

$$\boldsymbol{z}^\top \boldsymbol{\Sigma} \boldsymbol{z} = \sigma_1 z_1^2 + 2\gamma z_1 z_2 + \sigma_2 z_2^2. \tag{2}$$

- You may need to use the simple equation

$$\pi \exp(g) = \exp(\log(\pi) + g). \tag{3}$$

We first formulate $f_\ell(\boldsymbol{x})$ for our labels $\ell = 1, 2$, which using the proposed multivariate Gaussian distribution become

$$\begin{aligned} f_1(\boldsymbol{x}) &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_1)\right) \\ &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_1 + 3 \\ x_2 - 2 \end{pmatrix}^\top \boldsymbol{\Sigma}^{-1} \begin{pmatrix} x_1 + 3 \\ x_2 - 2 \end{pmatrix}\right) \\ &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{5}{2}(x_1 + 3)^2 + 2(x_1 + 3)(x_2 - 2) - (x_2 - 2)^2\right) \end{aligned}$$

where in the third equality we used 2. Similarly we derive

$$\begin{aligned} f_2(\boldsymbol{x}) &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_2)\right) \\ &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_1 - 2 \\ x_2 - 1 \end{pmatrix}^\top \boldsymbol{\Sigma}^{-1} \begin{pmatrix} x_1 - 2 \\ x_2 - 1 \end{pmatrix}\right) \\ &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{5}{2}(x_1 - 2)^2 + 2(x_1 - 2)(x_2 - 1) - (x_2 - 1)^2\right). \end{aligned}$$

If we set $\pi_1 f_1(\boldsymbol{x}) = \pi_2 f_2(\boldsymbol{x})$ the term $2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}$ cancels from both sides and using 3 we obtain

$$\begin{aligned} \exp\left(\log\left(\frac{1}{3}\right) - \frac{5}{2}(x_1 + 3)^2 + 2(x_1 + 3)(x_2 - 2) - (x_2 - 2)^2\right) = \\ \exp\left(\log\left(\frac{2}{3}\right) - \frac{5}{2}(x_1 - 2)^2 + 2(x_1 - 2)(x_2 - 1) - (x_2 - 1)^2\right) \end{aligned}$$

which requires the exp arguments to be equal, i.e.,

$$\begin{aligned} \log\left(\frac{1}{3}\right) - \frac{5}{2}(x_1 + 3)^2 + 2(x_1 + 3)(x_2 - 2) - (x_2 - 2)^2 = \\ \log\left(\frac{2}{3}\right) - \frac{5}{2}(x_1 - 2)^2 + 2(x_1 - 2)(x_2 - 1) - (x_2 - 1)^2. \end{aligned}$$

We expand the quadratic terms and after basic simplification we get

$$12x_2 - 27x_1 - 31.5 + \log\left(\frac{1}{3}\right) - \log\left(\frac{2}{3}\right) = 0$$

Since $\log(1/3) - \log(2/3) = -\log(2)$ we end up with 1.

Q3. The goal of this question is predicting the heart health of patients in a hospital. In the homework package, you can access the data file "HeartData.csv", which consists of 13 features and one response variable (num). The features represent some measurements of the patients' health atributes and num is an indication of the heart health. If num = 0, the heart is healthy, and if num = 1, it reports an issue. Below we summarize a brief description of each feature:
age: Age of patient
sex: Sex, 1 for male
cp: chest pain
trestbps: resting blood pressure
chol : serum cholesterol
fbs: fasting blood sugar larger 120mg/dl (1 true)
restecg: resting electroc. result (1 anomality)
thalach: maximum heart rate achieved
exang: exercise induced angina (1 yes)
oldpeak: ST depression induc. ex.
slope: slope of peak exercise ST
ca: number of major vessel
thal: no explanation provided, but probably thalassemia (3 normal; 6 fixed defect; 7 reversable defect)
num: diagnosis of heart disease (angiographic disease status)

Consider splitting the data into a a training and test set. Samples 1 to 200 form the training set and samples 201 to 297 form the test set. Try the following classification models to predict "num" in terms of the other features in the dataset:

- Use logistic regression for your classification. Report the p-values associated with the intercept and all the features. Which features have large p-values? Use the test data to estimate the accuracy of your model.
- Apply LDA and QDA, and again report your model accuracies using the test data.
- Among logistic regression, LDA, and QDA which model(s) seemed the most accurate one(s)?

Solution.

```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
import statsmodels.api as sm
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
summarize)
from ISLP import confusion_table
from ISLP.models import contrast
from sklearn.discriminant_analysis import \
(LinearDiscriminantAnalysis as LDA,
QuadraticDiscriminantAnalysis as QDA)
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Logistic Regression
myData = pd.read_csv('HeartData.csv')
allvars = myData.columns.drop(['num'])
design = MS(allvars)
X = design.fit_transform(myData)
y = myData.num == 1
X_train, X_test = X.loc[0:199,], X.loc[200:296]
y_train, y_test = y.loc[0:199,], y.loc[200:296]

glm_train = sm.GLM(y_train,
X_train,
family=sm.families.Binomial())
results = glm_train.fit()
print(results.summary())
probs = results.predict(exog=X_test) # to apply the model to the test data

labels = np.array([0]*X_test.shape[0])
labels[probs>0.5] = 1
print('================================================================================')
print(np.mean(labels == y_test))
```

```
# the output is
Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    num   No. Observations:                  200
Model:                            GLM   Df Residuals:                      186
Model Family:                Binomial   Df Model:                           13
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -63.478
Date:                Thu, 25 Apr 2024   Deviance:                       126.96
Time:                        14:46:56   Pearson chi2:                     174.
No. Iterations:                     6   Pseudo R-squ. (CS):             0.5236
Covariance Type:            nonrobust
==============================================================================
coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept    -10.3711      3.716     -2.791      0.005     -17.655      -3.087
age           -0.0073      0.031     -0.236      0.813      -0.068       0.054
sex            1.8161      0.703      2.582      0.010       0.438       3.195
cp             0.9642      0.290      3.324      0.001       0.396       1.533
trestbps       0.0341      0.014      2.432      0.015       0.007       0.062
chol           0.0075      0.005      1.583      0.113      -0.002       0.017
fbs           -1.0563      0.654     -1.616      0.106      -2.337       0.225
restecg        0.4627      0.244      1.894      0.058      -0.016       0.942
thalach       -0.0285      0.014     -1.967      0.049      -0.057   -9.99e-05
exang          0.6358      0.524      1.214      0.225      -0.390       1.662
oldpeak        0.2416      0.260      0.928      0.353      -0.268       0.752
slope          0.5692      0.454      1.252      0.210      -0.322       1.460
ca             0.9591      0.316      3.031      0.002       0.339       1.579
thal           0.3448      0.128      2.689      0.007       0.093       0.596
==============================================================================
==========================================================================
0.8041237113402062
```

```
# LDA
lda = LDA(store_covariance=True)
# Since the LDA estimator automatically adds an intercept, we should remove it.
#We can also directly use the labels rather than the Boolean
# vectors y_train.
X_train, X_test = [M.drop(columns=['intercept'])
for M in [X_train, X_test]]
L_train = myData.num.loc[0:199]
L_test = myData.num.loc[200:296]
lda.fit(X_train, L_train)
lda_pred = lda.predict(X_test)
print(confusion_table(lda_pred, L_test))
print(np.mean(lda_pred == L_test))

# Output:
#Truth       0   1
#Predicted
#0          46  14
#1           4  33
#0.8144329896907216

# QDA
qda = QDA(store_covariance=True)
qda.fit(X_train, L_train)
qda_pred = qda.predict(X_test)
print(confusion_table(qda_pred, L_test))
print(np.mean(qda_pred == L_test))

# Output
#Truth       0   1
#Predicted
#0          46  15
#1           4  32
#0.8041237113402062
```

Q4. A study analyzes the data on law school admission, and the goal is to examine the correlation between LSAT score and the first year GPA. For each of 15 law schools, we have the pair of data points (LSAT, GPA) as

$$\begin{array}{lll} (576, 3.93), & (580, 3.07), & (653, 3.12) \\ (635, 3.30), & (555, 3.00), & (575, 2.74) \\ (558, 2.81), & (661, 3.43), & (545, 2.76) \\ (578, 3.03), & (651, 3.36), & (572, 2.88) \\ (666, 3.44), & (605, 3.13), & (594, 2.96). \end{array}$$

(a) Calculate the correlation coefficient between LSAT and GPA.

(b) Pick the programming language of your choice, and use bootstrapping to estimate the standard deviation of the correlation coefficient. Use $B = 1000$ bootstrap resamples. Also plot a histogram of the results (use 20 bins).

```
import numpy as np
import matplotlib.pyplot as plt
from random import choices

LSAT = np.array([576, 580, 653, 635, 555, 575, 558, 661, 545, 578, 651, 572, 666, 605, 594]
GPA = np.array([3.93, 3.07, 3.12, 3.30, 3.00, 2.74, 2.81, 3.43, 2.76, 3.03, 3.36, 2.88, 3.4
# calculating the correlation coefficient by extracting the off diagonal element
# of the correlation matrix
CC = np.corrcoef(LSAT,GPA)[0,1]
print(CC)

# Set bootstrap size
n = 10000

# define a vector of zeros
ccB = np.zeros(n)

for i in np.arange(0, n):
BootInd = choices(range(0, LSAT.size), k=LSAT.size)
ccB[i] = np.corrcoef(LSAT[BootInd],GPA[BootInd])[0,1]

plt.hist(ccB,bins=15)
plt.show()
```