Alexander Bukharin

**ISYE 6416 Homework #2**

2b) Below is the code and resulting visualization of the number of iterations versus log-likelihood. Each method is described via comments.

**Code:**

# Logistic regression class

class logisticRegression:

def \_\_init\_\_(self, a = np.array([0.0, 0.0]).T, b = 0.0):

# Initialize a and b

self.a = a

self.b = b

def sigmoid(self, x):

# Sigmoid function

return 1 / (1 + np.exp(-np.matmul(self.a, x.T) - self.b))

def log\_like(self, x, y):

# Calculate log-likelihood

return np.sum(y \* np.log(self.sigmoid(x)) + (1 - y) \* np.log(1 - self.sigmoid(x)))

def hessian(self, x, y):

# calculate hessian of likelihood function with respect to weights. We calculate element-wise

h11 = np.sum(x[:, 0] \* x[:, 0] \* self.sigmoid(x) \* (1 - self.sigmoid(x)))

h12 = np.sum(x[:, 0] \* x[:, 1] \* self.sigmoid(x) \* (1 - self.sigmoid(x)))

h13 = np.sum(x[:, 0] \* self.sigmoid(x) \* (1 - self.sigmoid(x)))

h22 = np.sum(x[:, 1] \* x[:, 1] \* self.sigmoid(x) \* (1 - self.sigmoid(x)))

h23 = np.sum(x[:, 1] \* self.sigmoid(x) \* (1 - self.sigmoid(x)))

h33 = np.sum(self.sigmoid(x) \* (1 - self.sigmoid(x)))

return np.array([np.array([h11, h12, h13]), np.array([h12, h22, h23]), np.array([h13, h23, h33])])

def gradient(self, x, y):

# Calculate gradient of likelihood function

a\_gradient1 = np.sum(y \* (1 - self.sigmoid(x)) \* x[:, 0] - (1 - y) \* self.sigmoid(x) \* x[:, 0])

a\_gradient2 = np.sum(y \* (1 - self.sigmoid(x)) \* x[:, 1] - (1 - y) \* self.sigmoid(x) \* x[:, 1])

b\_gradient = np.sum(y \* (1 - self.sigmoid(x)) - (1 - y) \* self.sigmoid(x))

return -1 \* np.array([a\_gradient1, a\_gradient2, b\_gradient])

def newtons\_method(self, x, y, step\_size):

# Iteratively apply newtons method

log\_l = []

num\_iterations = 0

past\_like = -1000000000

while self.log\_like(x, y) - past\_like > 0.0000001:

num\_iterations += 1

past\_like = self.log\_like(x, y)

hess = self.hessian(x, y)

hess\_inv = np.linalg.inv(hess)

grad = self.gradient(x, y)

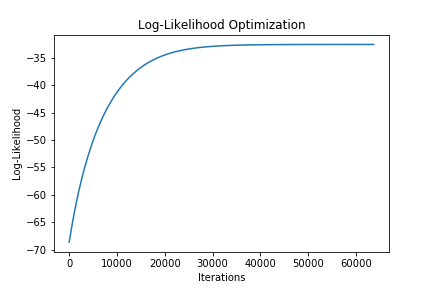
self.a[0] = self.a[0] - step\_size \* np.matmul(hess\_inv, grad)[0]

self.a[1] = self.a[1] - step\_size \* np.matmul(hess\_inv, grad)[1]

self.b = self.b - step\_size \* np.matmul(hess\_inv, grad)[2]

log\_l.append(self.log\_like(x, y))

**Image:**

****

As the number of iterations increases, the log-likelihood function increases. However, at a certain point a maximizer is found and the increase in log-likelihood is very small, so we have converged.

**Coefficients:** a = [0.75627727, 1.16495963], b = -2.606358864784067

3c)

**Code for weighted Linear regression (for linear regression set w\_i = 1):**

class weighted\_lin\_reg:

def \_\_init\_\_(self, num\_dim):

self.num\_dim = num\_dim

self.coeff = np.ones(num\_dim)

def predict(self, x):

# Calculate y-hat

return np.matmul(self.coeff, x)

def loss(self, x, y, weights):

# Calculate loss

return (1/2) \* np.sum(weights \* (self.predict(x) - y) \*\* 2)

def gradient\_descent(self, x, y, weights, step\_size):

C#alculate gradient and run gradient descent

jl = []

for j in range(1000):

# Calculate gradient

grad = np.zeros(self.num\_dim)

for i in range(self.num\_dim):

grad[i] = np.sum(weights \* (self.predict(x) - y) \* x[i, :])

#Adjust weights accordingly

self.coeff = self.coeff - step\_size \* grad

**Linear regression plot:**

A close up of a map

Description automatically generated

The resulting linear regression has coefficients: [0.32945371, 0.1751245 ]

3d) Using weighted linear regression with the instructed weights, we get the following line:

A close up of a map

Description automatically generated

This line has the following coefficient: [0.75992091, 0.39905379]

**Plot of Loss versus the number of iterations:**

**A screenshot of a cell phone

Description automatically generated**

We can see that that the loss decreases as the number of iterations increa-ses, until the loss converges at a minimum.

5) We use the sklearn.LinearModel class in python to fit the regression model. However, before we fit the regression, we format the data differently. Because location is a categorical variable, for each location we add a dummy variable that is equal to 1 if the location matches and 0 otherwise. However, we do this only for n - 1 locations, because you can encode which location it is with only n – 1 binary variables. Then we divide the data into 3 subsets: “foreclosure”, “short sale” and “regular” sales. We then fit the linear model for each subset. Here are the resulting coefficients:

| **Short Sale** | **Regular** | **Foreclosure** |
| --- | --- | --- |
| **Arroyo Grande** | -1.455192e-11 | 3.018670e+04 | 0.000000e+00 |
| **Atascadero** | 5.820766e-11 | -1.593769e+05 | 0.000000e+00 |
| **Bradley** | -1.273293e-10 | -2.725056e+05 | 1.091394e-10 |
| **Cambria** | 0.000000e+00 | -2.451125e+05 | -1.818989e-12 |
| **Cayucos** | 0.000000e+00 | -2.035427e+05 | 0.000000e+00 |
| **Creston** | 0.000000e+00 | -8.781725e+04 | 0.000000e+00 |
| **Grover Beach** | 6.402843e-10 | -2.379382e+05 | -4.365575e-11 |
| **Lompoc** | 0.000000e+00 | 3.441742e+05 | 0.000000e+00 |
| **Los Osos** | 0.000000e+00 | -2.706757e+04 | 1.115552e-12 |
| **Morro Bay** | 5.820766e-11 | -1.508408e+05 | 0.000000e+00 |
| **Nipomo** | 2.546585e-11 | 6.194220e+04 | -8.731149e-11 |
| **Oceano** | -5.820766e-11 | -1.300945e+06 | 0.000000e+00 |
| **Out Of Area** | -2.546585e-11 | -3.501632e+05 | -2.364686e-11 |
| **Paso Robles** | 0.000000e+00 | -5.284550e+04 | 0.000000e+00 |
| **Pismo Beach** | 0.000000e+00 | -2.102207e+05 | 0.000000e+00 |
| **San Luis Obispo** | 0.000000e+00 | -1.478397e+05 | 0.000000e+00 |
| **San Miguel** | 0.000000e+00 | -2.097723e+05 | 0.000000e+00 |
| **Santa Maria-Orcutt** | 0.000000e+00 | 4.186655e+04 | 0.000000e+00 |
| **Templeton** | -7.585186e-10 | -7.646144e+04 | -9.167707e-10 |
| **Arroyo Grande** | -1.681077e+04 | 0.000000e+00 | -2.820499e+03 |
| **Atascadero** | -6.215101e+04 | 0.000000e+00 | 1.278523e+04 |
| **Avila Beach** | 6.088038e+05 | 1.164153e-10 | -2.517941e+05 |
| **Bakersfield** | -3.367942e+04 | 0.000000e+00 | 0.000000e+00 |
| **Bathrooms** | 1.629114e+04 | 2.378297e+04 | 1.957310e+04 |
| **Bedrooms** | -1.875478e+04 | -5.638246e+04 | -7.326006e+03 |
| **Bradley** | -1.444307e+05 | 0.000000e+00 | 5.936287e+03 |
| **Buellton** | -4.743851e+04 | 0.000000e+00 | -2.362668e+04 |
| **Cambria** | -1.789769e+05 | 0.000000e+00 | -1.277098e+04 |
| **Cayucos** | 3.850595e+05 | 0.000000e+00 | 0.000000e+00 |
| **Coalinga** | -4.847648e+04 | 5.820766e-10 | 2.391971e-10 |
| **Creston** | -5.820766e-11 | 0.000000e+00 | -6.046243e+03 |
| **Greenfield** | 0.000000e+00 | 0.000000e+00 | -2.771641e+04 |
| **Grover Beach** | -5.835280e+04 | -1.644366e-09 | -2.299085e+04 |
| **Guadalupe** | -1.613664e+04 | 0.000000e+00 | 6.114371e+04 |
| **King City** | 0.000000e+00 | 0.000000e+00 | -5.557550e+04 |
| **Lockwood** | -3.359902e+04 | 5.784386e-10 | 9.349606e-10 |
| **Lompoc** | -5.218000e+04 | -8.003553e-11 | 9.468998e+03 |
| **Los Alamos** | -7.271770e+04 | 1.455192e-11 | -2.910383e-11 |
| **Los Osos** | -4.566427e+04 | 0.000000e+00 | -2.869267e+04 |
| **MLS** | -1.343421e+00 | 2.384657e+02 | -3.358191e-01 |
| **Morro Bay** | -9.449830e+02 | 1.501438e-12 | 2.176340e+03 |
| **New Cuyama** | 3.510720e+03 | 0.000000e+00 | 0.000000e+00 |
| **Nipomo** | -2.295845e+04 | 0.000000e+00 | 3.289436e+03 |
| **Oceano** | 1.071727e+04 | 0.000000e+00 | 1.602975e+04 |
| **Out Of Area** | 0.000000e+00 | 1.661063e+05 | 0.000000e+00 |
| **Paso Robles** | -5.893907e+04 | 0.000000e+00 | -1.455344e+04 |
| **Pismo Beach** | -7.208198e+04 | 0.000000e+00 | 2.470532e+05 |
| **Price/SQ.Ft** | 1.355521e+03 | 2.586250e+03 | 1.735858e+03 |
| **San Luis Obispo** | -3.675891e+04 | 0.000000e+00 | -6.212733e+04 |
| **San Miguel** | -4.697347e+04 | 0.000000e+00 | 1.321772e+04 |
| **San Simeon** | -2.182787e-10 | 2.910383e-10 | -2.641362e+04 |
| **Santa Margarita** | 0.000000e+00 | 0.000000e+00 | 1.136507e+05 |
| **Santa Maria-Orcutt** | -6.546226e+04 | 1.882249e+05 | 1.317745e+04 |
| **Santa Ynez** | 1.549332e+05 | -4.365575e-11 | 4.473043e+04 |
| **Size** | 2.045788e+02 | 4.816276e+02 | 1.979149e+02 |
| **Soledad** | -2.546585e-11 | 0.000000e+00 | -1.671636e+04 |
| **Solvang** | -2.884171e+04 | 0.000000e+00 | -2.381116e+04 |
| **Templeton** | -1.944952e+04 | 0.000000e+00 | 3.299671e+04 |

The corresponding R-squared values are 0.9636038261605505, 0.9035225203628774, and 0.9008663240827933 for foreclosure, regular, and short sale respectively.