Homework 4: Regression

Sreesha Venkat

March 13, 2017

Collaborators: Sydney McMuldroch, Lynn Kong, Roshni Patel

Q1. Logistic Regression with Newton's Method

(1)

$$J(w) = \lambda|w|_2^2 - \sum y_i ln(s_i * (1 - s_i)) + (1 - y_i) ln((1 - (s_i * (1 - s_i))))$$

$$\nabla_w J = 2\lambda w - X^T(y - s)$$

(2)

$$\nabla_w^2 J = 2\lambda I + X^T \Omega X$$

$$\text{Where } \Omega \text{ is the matrix: } \begin{bmatrix} s_1(1-s_1) & 0 & ... \\ 0 & s_2(1-s_2) & ... \\ \vdots & \vdots & \ddots \end{bmatrix}$$

(3)

$$w \leftarrow w - (2\lambda I + X^T \Omega X)^{-1}(2\lambda w - X^T(y - s))$$

(4) $\text{s}^{(0)} = \begin{bmatrix} .9526 \\ .7311 \\ .7311 \\ .2689 \end{bmatrix}$

$w^{(1)} = \begin{bmatrix} -0.3865 & 1.404 & -2.284 \end{bmatrix}^T$

$s^{(1)} = \begin{bmatrix} .8731 \\ .8237 \\ .2932 \\ .2198 \end{bmatrix}$

$w^{(2)} = \begin{bmatrix} -0.512 & 1.453 & -2.163 \end{bmatrix}^T$

Q2. ℓ_1 and ℓ_2 Regularization

(1)

$$J(w) = |Xw - y|^2 + \lambda||w||_{\ell 1}$$
$$J(w) = y^T y + w^T X^T Xw - 2y^T Xw + \lambda||w||_{\ell 1}$$
$$J(w) = y^T y + \sum w_i^2 n - 2y^T X_i w_i + \lambda|w_i|$$

Therefore if we set $g(y) = y^T y$ and $f(X_{*i}, w_i, y, \lambda) = w_i^2 n - 2y^T X_{*i} w_i + \lambda|w_i|$, we can write $J(w)$ in the form

$$J(w) = g(y) + \sum f(X_{*i}, w_i, y, \lambda)$$

(2)

$$\nabla_w J = 2w_i n - 2X_{*i}^T y + \lambda|1| = 0$$
$$2w_i n = 2X_{*i}^T y - \lambda|1|$$
$$w_i = \frac{2X_{*i}^T y - \lambda|1|}{2n}$$

Therefore if $w_{*i} > 0$, $w_{*i} = \frac{2X_{*i}^T y - \lambda}{2n}$.

(3) Using similar logic from part 2, if $w_{*i} < 0$, $w_{*i} = \frac{2X_{*i}^T y + \lambda}{2n}$.

(4) From part 2, we can change our equations in the case that $w_{*i} = 0$ to be:

$$2X_{*i}^T y \leq \lambda$$
$$2X_{*i}^T y \geq -\lambda$$

Therefore the condition for $w_{*i} = 0$ is

$$-\lambda \leq 2X_{*i}^T y \leq \lambda$$

(5) With ridge regression, the function from part 1 is changed in that it does not take the absolute value of λ when we take the gradient of $J(w)$. To find the new condition in which $w_{*i} = 0$, we can take the gradient of $J(w)$, set it equal to 0, and solve for w_{*i}:

$$\nabla_w J = 2w_i n - 2y^T X_{*i} + 2\lambda w_i = 0$$
$$2w_i(n + \lambda) = 2y^T X_{*i}$$
$$w_i = \frac{y^T X_{*i}}{(n + \lambda)}$$

This differs from the condition we obtained in part 4 in that the former has an equality condition for w_{*i}, whereas the latter specifies a range for λ that makes the equation hold true.

Q3. Regression and Dual Solutions

(a) For $\nabla|w|^4$ we have:

$$\nabla|w|^4 = 2w * 2(w^2)$$
$$\nabla|w|^4 = 4|w|^2 w$$

And for $\nabla_w|Xw - y|^4$ we have:

$$\nabla_w|Xw - y|^4 = 4|Xw - y|^2 X^T(Xw - y)$$

(b) To show that the optimum w^* is unique, we must show that the cost function is strictly convex, or that the Hessian of w^* is positive definite. However, because $w \to Xw - y$ is an affine transformation, we know that if $\nabla^2|w|^4$ is positive semidefinite, then so is $\nabla_w^2|Xw - y|^4$. We start by first showing that the Hessian of $|w|^4$ is positive semidefinite:

$$\nabla|w|^4 = \begin{bmatrix} w_1(w_1^2 + w_2^2 + ...w_d^2) \\ \vdots \\ w_d(w_1^2 + w_2^2 + ...w_d^2) \end{bmatrix}$$

$$\nabla^2|w|^4 = \begin{bmatrix} (3w_1^2 + w_2^2 + ...) & \cdots & 2w_1w_d \\ \vdots & \ddots & \vdots \\ 2w_1w_d & ... & (3w_d^2 + w_1^2 + ...) \end{bmatrix}$$

From this we can see that the Hessian of $|w|^4$ is symmetric, which means that $\nabla^2|w|^4$ is positive semidefinite. In the regularized regression problem, we add on the regularization parameter $\lambda|w|^2$. Taking the Hessian of this parameter, we get $2\lambda I$. Because λ is strictly positive and I is positive definite, and we are adding this parameter to our positive semidefinite matrix, we are able to show that the Hessian of w^* is positive definite and therefore unique.

By setting the gradient of the objective function to zero, we can also show that w^* can be written as a linear combination $w^* = \sum a_i X_i$:

$$\nabla^2 w^* = 4|Xw - y|^2 X^T(Xw - y) + 2\lambda w = 0$$
$$w = \frac{-1}{2\lambda} 4|Xw - y|^2 X^T(Xw - y)$$
$$w = X^T a$$

Where a is the vector of dual coefficients:

$$a = \begin{bmatrix} \frac{-2}{\lambda}|Xw - y|^2(Xw - y) \\ \vdots \\ \vdots \end{bmatrix}$$

This shows that w^* can be written as a linear combination

$$w^* = \sum a_i X_i$$

.

(c)

$$\nabla w^* = X^T \nabla L(w^T X, y + 2\lambda w = 0$$
$$2\lambda w = -X^T \nabla L(w^T X, y$$

$$w = \frac{-1}{2\lambda} X^T \nabla L(w^T X, y) = X^T a$$

Where $a = \frac{-1}{2\lambda} \nabla L(w^T X, y)$. This shows that

$$w^* = \sum a_i X_i$$

If the loss function is not convex, the optimal solution does not always have the form $w^* = \sum a_i X_i$ because when $\nabla w^* = 0$ we are not necessarily finding the minimum. If the loss function is not strictly convex, the minimum value that we think we found could in fact be a maximum or saddle point. In this case the gradient would not be 0, and therefore the optimal solution would not take on the form $w^* = \sum a_i X_i$.

Q4. Franzia Classification + Logistic Regression = Party!

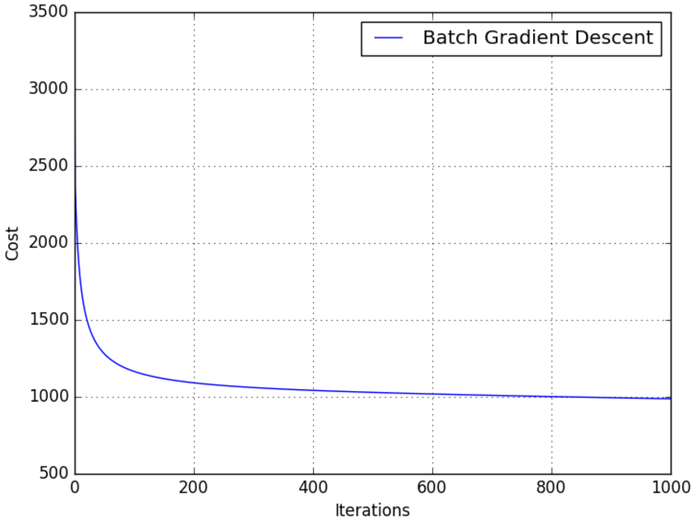
(1) Batch Gradient Descent update equation for logistic regression with ℓ_2 regularization:

$$J(w) = \lambda|w|_2^2 - \sum y_i ln(s_i * (1 - s_i)) + (1 - y_i) ln((1 - (s_i * (1 - s_i))))$$

$$\nabla_w J = 2\lambda w - X^T(y - s)$$

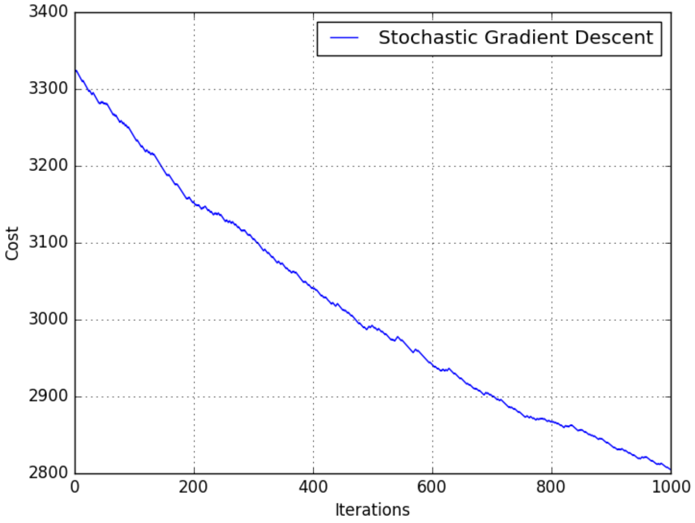
$$w \leftarrow w - \epsilon(2\lambda w - X^T(y - s(Xw)))$$

Figure 1: Batch Gradient Descent



(2) Stochastic Gradient Descent update equation for logistic regression with ℓ_2 regularization: $w \leftarrow w + \epsilon(X_i(y_i - s(X_i w)) - 2\lambda w)$

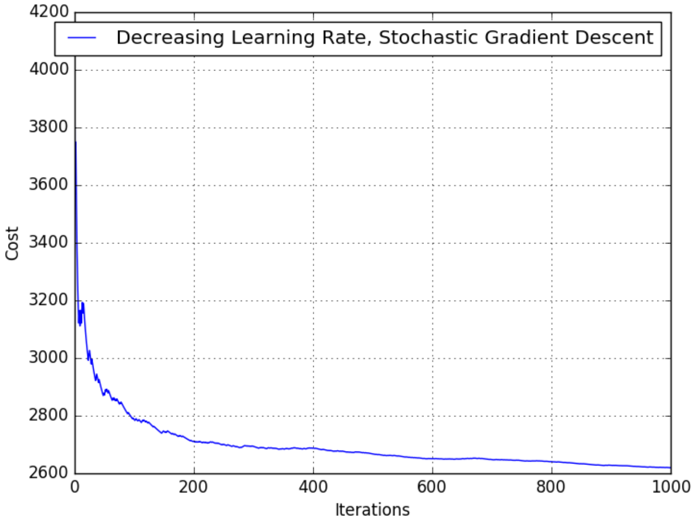
Figure 2: Stochastic Gradient Descent



From this plot we see that batch gradient descent converges more quickly than stochastic gradient descent.

(3) From the graph below, we see that this strategy is better than having a constant ϵ, as we converge more quickly.

Figure 3: Learning Rate Decreasing, Stochastic Gradient Descent



(4) Kaggle Display Name: SreeshaVenkat, Score: 0.93548. To decide on parameters for the best classifier, I used trial and error to find the learning rate and regularization parameter that resulted in lowest cost. Additionally, I ran my update equation for 5000 iterations, as I noticed that doing more iterations to converge on a better w resulted in a lower classification error rate.

Q5. Real World Spam Classification

The linear SVM can't utilize the new feature well because the timestamps before midnight and after midnight are not linearly separable. For example, if we see a spike in spam volume just before midnight at 23:59:59, and just after midnight at 00:00:01, the SVM has no way to linearly separate spam from ham.

One way to improve upon this is to change the feature to be a polynomial feature of degree 2. By doing this, we plot the timestamps of spam vs ham messages on parabola-like curve, which ultimately makes the two classes of messages linearly separable.

Alternatively, we could change the feature from a timestamp to record minutes from the closest midnight. Doing this also creates a way to linearly separate the data. For example, if spam messages spike between 23:00:00 and 01:00:00, we can draw the decision boundary at 60 minutes, where points at 60 minutes or less are spam, and points at 60 minutes or greater are ham.

```
%!PS-Adobe-3.0
%%Title: hw4_q1.py
%%For: sreeshav
%%Creator: VIM - Vi IMproved 7.4 (2013 Aug 10)
%%CreationDate: Fri Mar 10 22:54:20 2017
%%DocumentData: Clean8Bit
%%Orientation: Portrait
%%Pages: (atend)
%%PageOrder: Ascend
%%BoundingBox: 59 49 564 800
%%DocumentMedia: A4 595 842 0 () ()
%%DocumentNeededResources: font Courier
%%+ font Courier-Bold
%%+ font Courier-Oblique
%%+ font Courier-BoldOblique
%%DocumentSuppliedResources: procset VIM-Prolog 1.4 1
%%+ encoding VIM-latin1 1.0 0
%%Requirements: duplex collate
%%EndComments
%%BeginDefaults
%%PageResources: font Courier
%%+ font Courier-Bold
%%+ font Courier-Oblique
%%+ font Courier-BoldOblique
%%PageMedia: A4
%%EndDefaults
%%BeginProlog
%%BeginResource: procset VIM-Prolog
%%BeginDocument: /usr/share/vim/vim74/print/prolog.ps
%!PS-Adobe-3.0 Resource-ProcSet
%%Title: VIM-Prolog
%%Version: 1.4 1
%%EndComments
% Editing of this file is NOT RECOMMENDED.  You run a very good risk of causing
% all PostScript printing from VIM failing if you do.  PostScript is not called
% a write-only language for nothing!
/packedarray where not{userdict begin/setpacking/pop load def/currentpacking
false def end}{pop}ifelse/CP currentpacking def true setpacking
/bd{bind def}bind def/ld{load def}bd/ed{exch def}bd/d/def ld
/db{dict begin}bd/cde{currentdict end}bd
/T true d/F false d
/SO null d/sv{/SO save d}bd/re{SO restore}bd
/L2 systemdict/languagelevel 2 copy known{get exec}{pop pop 1}ifelse 2 ge d
/m/moveto ld/s/show ld /ms{m s}bd /g/setgray ld/r/setrgbcolor ld/sp{showpage}bd
/gs/gsave ld/gr/grestore ld/cp/currentpoint ld
/ul{gs UW setlinewidth cp UO add 2 copy newpath m 3 1 roll add exch lineto
stroke gr}bd
/bg{gs r cp BO add 4 -2 roll rectfill gr}bd
/sl{90 rotate 0 exch translate}bd
L2{
/sspd{mark exch{setpagedevice}stopped cleartomark}bd
/nc{1 db/NumCopies ed cde sspd}bd
/sps{3 db/Orientation ed[3 1 roll]/PageSize ed/ImagingBBox null d cde sspd}bd
/dt{2 db/Tumble ed/Duplex ed cde sspd}bd
/c{1 db/Collate ed cde sspd}bd
}{
/nc{/#copies ed}bd
/sps{statusdict/setpage get exec}bd
/dt{statusdict/settumble 2 copy known{get exec}{pop pop pop}ifelse
statusdict/setduplexmode 2 copy known{get exec}{pop pop pop}ifelse}bd
/c{pop}bd
}ifelse
/ffs{findfont exch scalefont d}bd/sf{setfont}bd
/ref{1 db findfont dup maxlength dict/NFD ed{exch dup/FID ne{exch NFD 3 1 roll
put}{pop pop}ifelse}forall/Encoding findresource dup length 256 eq{NFD/Encoding
3 -1 roll put}{pop}ifelse NFD dup/FontType get 3 ne{/CharStrings}{/CharProcs}
ifelse 2 copy known{2 copy get dup maxlength dict copy[/questiondown/space]{2
copy known{2 copy get 2 index/.notdef 3 -1 roll put pop exit}if pop}forall put
}{pop pop}ifelse dup NFD/FontName 3 -1 roll put NFD definefont pop end}bd
CP setpacking
(\004)cvn{}bd
% vim:ff=unix:
%%EOF
```

```
%%EndDocument
%%EndResource
%%BeginResource: encoding VIM-latin1
%%BeginDocument: /usr/share/vim/vim74/print/latin1.ps
%!PS-Adobe-3.0 Resource-Encoding
%%Title: VIM-latin1
%%Version: 1.0 0
%%EndComments
/VIM-latin1[
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/space /exclam /quotedbl /numbersign /dollar /percent /ampersand /quotesingle
/parenleft /parenright /asterisk /plus /comma /minus /period /slash
/zero /one /two /three /four /five /six /seven
/eight /nine /colon /semicolon /less /equal /greater /question
/at /A /B /C /D /E /F /G
/H /I /J /K /L /M /N /O
/P /Q /R /S /T /U /V /W
/X /Y /Z /bracketleft /backslash /bracketright /asciicircum /underscore
/grave /a /b /c /d /e /f /g
/h /i /j /k /l /m /n /o
/p /q /r /s /t /u /v /w
/x /y /z /braceleft /bar /braceright /asciitilde /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/space /exclamdown /cent /sterling /currency /yen /brokenbar /section
/dieresis /copyright /ordfeminine /guillemotleft /logicalnot /hyphen /registered /ma
cron
/degree /plusminus /twosuperior /threesuperior /acute /mu /paragraph /periodcentered
/cedilla /onesuperior /ordmasculine /guillemotright /onequarter /onehalf /threequart
ers /questiondown
/Agrave /Aacute /Acircumflex /Atilde /Adieresis /Aring /AE /Ccedilla
/Egrave /Eacute /Ecircumflex /Edieresis /Igrave /Iacute /Icircumflex /Idieresis
/Eth /Ntilde /Ograve /Oacute /Ocircumflex /Otilde /Odieresis /multiply
/Oslash /Ugrave /Uacute /Ucircumflex /Udieresis /Yacute /Thorn /germandbls
/agrave /aacute /acircumflex /atilde /adieresis /aring /ae /ccedilla
/egrave /eacute /ecircumflex /edieresis /igrave /iacute /icircumflex /idieresis
/eth /ntilde /ograve /oacute /ocircumflex /otilde /odieresis /divide
/oslash /ugrave /uacute /ucircumflex /udieresis /yacute /thorn /ydieresis]
/Encoding defineresource pop
% vim:ff=unix:
%%EOF
%%EndDocument
%%EndResource
%%EndProlog
%%BeginSetup
595 842 0 sps
1 nc
T F dt
T c
%%IncludeResource: font Courier
/_F0 /VIM-latin1 /Courier ref
/F0 10 /_F0 ffs
%%IncludeResource: font Courier-Bold
/_F1 /VIM-latin1 /Courier-Bold ref
/F1 10 /_F1 ffs
%%IncludeResource: font Courier-Oblique
/_F2 /VIM-latin1 /Courier-Oblique ref
/F2 10 /_F2 ffs
%%IncludeResource: font Courier-BoldOblique
/_F3 /VIM-latin1 /Courier-BoldOblique ref
/F3 10 /_F3 ffs
/UO -1 d
/UW 0.5 d
/BO -2.5 d
%%EndSetup
%%Page: 1 1
%%BeginPageSetup
sv
```

```
0 g
F0 sf
%%EndPageSetup
F1 sf
(hw4_q1.py                                                                     Page
1)59.5 792.4 ms
F0 sf
(import matplotlib)59.5 772.4 ms
(import numpy as np)59.5 762.4 ms
(import matplotlib.cm as cm)59.5 752.4 ms
(import matplotlib.mlab as mlab)59.5 742.4 ms
(import matplotlib.pyplot as plt)59.5 732.4 ms
(import math)59.5 722.4 ms
(import scipy.io as sio)59.5 712.4 ms
(import random )59.5 702.4 ms
(import csv)59.5 692.4 ms
(from scipy.stats import multivariate_normal)59.5 672.4 ms
(from sklearn import preprocessing)59.5 662.4 ms
(w = np.matrix\([[-2],[1],[0]]\))59.5 642.4 ms
(s = np.matrix\([[.9526],[.7311],[.7311],[.2689]]\))59.5 632.4 ms
(x = np.matrix\([[0, 3, 1], [1, 3, 1], [0, 1, 1], [1, 1, 1]]\))59.5 622.4 ms
(y = np.matrix\([[1], [1], [0], [0]]\))59.5 612.4 ms
(lamb = np.matrix\([[.14, 0, 0], [0, .14, 0], [0, 0, .14]]\))59.5 602.4 ms
(omega = np.matrix\([[\(s[0]*\(1-s[0]\)\), 0, 0, 0],)59.5 592.4 ms
(                                  [0, \(s[1]*\(1-s[1]\)\), 0, 0],)59.5 582.4 ms
(                                  [0, 0, \(s[2]*\(1-s[2]\)\), 0],)59.5 572.4 ms
(                                  [0, 0, 0, \(s[3]*\(1-s[3]\)\)])59.5 562.4 ms
(                                  ]\))59.5 552.4 ms
(num_1 = np.dot\(lamb, w\))59.5 532.4 ms
(y_s = np.subtract\(y.T, s.T\))59.5 522.4 ms
(num_2 = np.dot\(x.T, y_s.T\))59.5 512.4 ms
(numerator = np.subtract\(num_1, num_2\))59.5 502.4 ms
(temp = np.dot\(x.T, omega\))59.5 482.4 ms
(denominator = np.add\(lamb, np.dot\(temp, x\)\))59.5 472.4 ms
(print\("STUFF"\))59.5 462.4 ms
(print\(denominator\))59.5 452.4 ms
(denom_float = denominator.astype\(float\))59.5 442.4 ms
(inv_denom = np.linalg.inv\(denom_float\))59.5 432.4 ms
(div = np.dot\(inv_denom, numerator\))59.5 412.4 ms
(val = w - div)59.5 392.4 ms
(print\(val\))59.5 382.4 ms
(for row in x:)59.5 372.4 ms
(        print\(np.dot\(row, val\)\))59.5 362.4 ms
(a = [1.92821414, 1.54175925, -0.88009751, -1.26655239])59.5 352.4 ms
(w = val)59.5 332.4 ms
(s = np.matrix\([[.8731],[.8237],[.2932],[.2198]]\))59.5 322.4 ms
(x = np.matrix\([[0, 3, 1], [1, 3, 1], [0, 1, 1], [1, 1, 1]]\))59.5 312.4 ms
(y = np.matrix\([[1], [1], [0], [0]]\))59.5 302.4 ms
(lamb = np.matrix\([[.14, 0, 0], [0, .14, 0], [0, 0, .14]]\))59.5 292.4 ms
(omega = np.matrix\([[\(s[0]*\(1-s[0]\)\), 0, 0, 0],)59.5 282.4 ms
(                                  [0, \(s[1]*\(1-s[1]\)\), 0, 0],)59.5 272.4 ms
(                                  [0, 0, \(s[2]*\(1-s[2]\)\), 0],)59.5 262.4 ms
(                                  [0, 0, 0, \(s[3]*\(1-s[3]\)\)])59.5 252.4 ms
(                                  ]\))59.5 242.4 ms
(num_1 = np.dot\(lamb, w\))59.5 222.4 ms
(y_s = np.subtract\(y.T, s.T\))59.5 212.4 ms
(num_2 = np.dot\(x.T, y_s.T\))59.5 202.4 ms
(numerator = np.subtract\(num_1, num_2\))59.5 192.4 ms
(temp = np.dot\(x.T, omega\))59.5 172.4 ms
(denominator = np.add\(lamb, np.dot\(temp, x\)\))59.5 162.4 ms
(denom_float = denominator.astype\(float\))59.5 142.4 ms
(inv_denom = np.linalg.inv\(denom_float\))59.5 132.4 ms
(div = np.dot\(inv_denom, numerator\))59.5 112.4 ms
(val = w - div)59.5 92.4 ms
(print\("ITER 2"\))59.5 82.4 ms
(print\(val\))59.5 72.4 ms
re sp
%%PageTrailer
%%Trailer
%%Pages: 1
%%EOF
```

```
import matplotlib
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plot
import math
import scipy.io as sio
import scipy.special as special
import random
import sklearn
import csv
from sklearn import preprocessing

trainingSet = sio.loadmat('data.mat')
data = trainingSet["X"]
labels = trainingSet["y"]
test = trainingSet["X_test"]
data = sklearn.preprocessing.normalize(data, norm='l2', axis=1, copy=True)
test = sklearn.preprocessing.normalize(test, norm='l2', axis=1, copy=True)
shuffled_data = []

validation_data = data[:1200]
training_data = data[1200:]
validation_labels = labels[:1200]
training_labels = labels[1200:]

#learning_rate = 0.0023
regularization = 0.0023

def cost_fn(w, X, y):
        logistic_regression_sum = 0
        for i in range(len(X)):
                first_term = y[i] * np.log(special.expit(np.dot(X[i], w)))
                X_remainder = np.log(1 - special.expit(np.dot(X[i], w)))
                y_remainder = 1 - y[i]
                second_term = X_remainder * y_remainder
                logistic_regression_sum += (first_term + second_term)
        return logistic_regression_sum

def l2_cost_fn(w, X, y):
        l2_regularization = np.multiply(regularization, np.square(np.linalg.norm(w,
2)))
        cost = cost_fn(w, X, y)
        l2_cost = l2_regularization - cost
        return l2_cost[0]

def update(w, X, y):
        sigmoid_subtraction = np.subtract(y, special.expit(np.matmul(X, w)))
        sigmoid_mul = np.matmul(X.T, sigmoid_subtraction)
        reg = regularization * 2 * w
        update = w + (learning_rate * (reg + sigmoid_mul))
        return update

X = np.array(training_data)
y = np.array(training_labels)
w = np.zeros((len(X[0]), 1))
iterations = [i for i in range(1000)]
cost = []

for i in iterations:
        print(i)
        learning_rate = 1/(1+i)
        x_array = np.array([X[i]])
        y_array = np.array([y[i]])
        w = update(w, x_array, y_array)
        iter_cost = l2_cost_fn(w, X, y)
        cost.append(iter_cost)

plot.plot(iterations, cost, label = "Decreasing Learning Rate, Stochastic Gradient D
escent")
plot.legend()
plot.grid()
plot.xlabel("Iterations")
plot.ylabel("Cost")
plot.savefig("Learning Rate Decreasing, Stochastic Gradient Descent Cost")

X = np.array(test)
y = np.array(training_labels)

# total = 0
# error = 0
# count = 0
# for i in range(len(X)):
#       classification = special.expit(np.dot(w.T, X[i]))
#       if classification < .5:
#               value = 0
#       else:
#               value = 1
#       if value != y[i]:
#               error += 1
#       total += 1
#       count += 1

# print(cost)
# print(error/total)
```