Garrett Merz

① Let $\ell(w) = \sum_{i=1}^{n} \ell_i(w) = \sum_{i=1}^{n} -y_i \log h(x_i) - (1-y_i)\log(1-h(x_i))$

where $h(x) = g(w^T x) = \frac{1}{1+e^{-w^T x}}$.

a) Find $\nabla \ell(w)$.

$$\nabla \ell(w) = \left(\frac{\partial \ell(w)}{\partial w_1}, \frac{\partial \ell(w)}{\partial w_2}, \cdots \frac{\partial \ell(w)}{\partial w_n} \right).$$

So $$\frac{\partial \ell(w)}{\partial w_i} = \frac{\partial \ell_i(w)}{\partial w_i} = \sum_{i=1}^{n} \frac{-y_i}{h(x_i)} \frac{\partial h(x_i)}{\partial w_i} - \frac{(1-y_i)}{1-h(x_i)} \left(-\frac{\partial h(x_i)}{\partial w_i}\right)$$

$$= \sum_{i=1}^{n} \left(\frac{-y_i}{h(x_i)} + \frac{(1-y_i)}{1-h(x_i)} \right) \left(\frac{\partial h(x_i)}{\partial w_i} \right)$$

$$= \sum_{i=1}^{n} \frac{(h(x_i)-1)y_i + (1-y_i)h(x_i)}{h(x_i)(1-h(x_i))} \left(\frac{\partial h(x_i)}{\partial w_i} \right)$$

$$= \sum_{i=1}^{n} \frac{h(x_i) - y_i}{h(x_i)(1-h(x_i))} \left(\frac{\partial h(x_i)}{\partial w_i} \right)$$

Let $x_i = (x_1^{(i)}, x_2^{(i)}, \cdots x_n^{(i)}) = x^{(i)}$.

Then we find $\frac{\partial h(x_i)}{\partial w_j} = \frac{\partial}{\partial w_j} \left((1 + e^{-w_1 x_1^{(i)}} e^{-w_2 x_2^{(i)}} \cdots) \right)^{-1}$.

$$= \frac{-1}{(1+e^{-w^T x^{(i)}})^2} \cdot -x_j^{(i)} e^{-w^T x^{(i)}} = \frac{x_j^{(i)} e^{-w^T x^{(i)}}}{(1+e^{-w^T x^{(i)}})^2} = x_j^{(i)} (h(x^{(i)}))(1-h(x^{(i)}))$$

So $$\nabla \ell(w) = \sum_{i=1}^{n} x_j^{(i)} (h(x^{(i)}) - y^{(i)}).$$ □

b) Find H for $\ell(w)$, and show that it is PSD, and thus that $\ell(w)$ is convex.

$$H_{kj} = \frac{\partial^2 \ell}{\partial w_k \partial w_j} = \frac{\partial}{\partial w_k}\left(\sum_{i=1}^{n} x_j^{(i)}\left(h(x^{(i)}) - y^{(i)}\right)\right)$$

$$= \sum_{i=1}^{n} x_j^{(i)} \frac{\partial h(x)}{\partial w_k} = \sum_{i=1}^{n} x_j^{(i)} x_k^{(i)} \left(h(x^{(i)})\right)\left(1 - h(x^{(i)})\right)$$

So $H = \sum_{i=1}^{n} x^{(i)} \left(x^{(i)}\right)^T h(x^{(i)})\left(1 - h(x^{(i)})\right)$

(if we assume $x^{(i)}$ is a column vector).

We show H is PSD. Let A be a column vector in $\mathbb{R}^n$ $(\neq 0)$.

Then

$$A^T H A = \sum_{i=1}^{n} A^T x^{(i)} \left(x^{(i)}\right)^T A \; h(x^{(i)})\left(1 - h(x^{(i)})\right)$$

$$= \sum_{i=1}^{n} \left(A \cdot x^{(i)}\right)^2 h(x^{(i)})\left(1 - h(x^{(i)})\right)$$

$$= \sum_{i=1}^{n} \frac{\left(A \cdot x^{(i)}\right)^2}{\left(1 + e^{-w^T x^{(i)}}\right)^2}$$

This term is $\geq 0 \;\; \forall \; A \in \mathbb{R}^n \; (A \neq 0)$

So H is positive semidefinite, and thus, $\ell(w)$ is convex.

□

c) Use this on the Classification data set:

See attached code.

---

(2) Let $L(\theta) = \prod_{i=1}^{n} f(x_i;\theta)$ be a likelihood, and

$$\ell(\theta) = \sum_{i=1}^{n} \log f(x_i;\theta).$$

a) Suppose $f(x;\alpha,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\alpha)^2}{2\sigma^2}\right)$.

Find the MLE for α, σ^2:

$$\log(f(x_i;\theta)) = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(x_i-\alpha)^2}{2\sigma^2}$$

$$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_i-\alpha)^2}{2\sigma^2}$$

$$= -\frac{1}{2}\log(2\pi) - \log(\sigma) - \frac{(x_i-\alpha)^2}{2\sigma^2}$$

So $\ell(\theta) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \sum_{i=1}^{n} \frac{(x_i-\alpha)^2}{2\sigma^2}$

Maximize: $\frac{d\ell}{d\sigma} = -\frac{n}{\sigma} + \sum_{i=1}^{n} \frac{2(x_i-\alpha)^2}{2\sigma^3} = \frac{1}{\sigma}\left(-n + \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\alpha)^2\right) = 0$

So $\frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\alpha)^2 = n$, i.e.,

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i-\alpha)^2}{n}.$$

Similarly, $\frac{\partial l}{\partial \alpha} = -\sum_{i=1}^{n} \frac{2(x_i - \alpha)}{2\sigma^2} = -\sum_{i=1}^{n} \frac{(x_i - \alpha)}{2\sigma^2}$.

So $\frac{\partial l}{\partial \alpha} = 0$ gives $(\sum_{i=1}^{n} x_i) - n\alpha = 0$ i.e.,

$\alpha = \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}$.

So putting it all together,

$\alpha = \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}$, $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$. □

b) Let $f(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu))$. Find MLE for μ, Σ.

Then. $l(\theta) = \sum_{i=1}^{n} -\frac{1}{2}\log((2\pi)^d |\Sigma|) - \frac{1}{2}(x^{(i)} - \mu)^T \Sigma^{-1} (x^{(i)} - \mu)$

$= \sum_{i=1}^{n} -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x^{(i)} - \mu)^T \Sigma^{-1} (x^{(i)} - \mu)$.

$= \sum_{i=1}^{n} \left(\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2} \sum_{j=1}^{m} \sum_{a=1}^{m} (x_a^{(i)} - \mu_a) \Sigma_{aj}^{-1} (x_j^{(i)} - \mu_j) \right)$

So $\frac{\partial l}{\partial \mu_k} = \sum_{i=1}^{n} \left(\frac{1}{2} \sum_{j \neq k} \Sigma_{kj}^{-1} (x_j^{(i)} - \mu_j) + \sum_{a \neq k} (x_a^{(i)} - \mu_a) \Sigma_{ak}^{-1} + 2\Sigma_{kk}^{-1} (x_k^{(i)} - \mu_k) \right)$. But Σ is symmetric, so Σ^{-1} is too, so

$\frac{\partial l}{\partial \mu_k} = \sum_{i=1}^{n} \sum_{a=1}^{m} \Sigma_{ak}^{-1} (x_a^{(i)} - \mu_a) = \sum_{i=1}^{n} (\Sigma^{-1}(x^{(i)} - \mu))_k =$

Setting this equal to zero $\forall k$, we get

$\Sigma^{-1} (\sum_{i=1}^{n} x^{(i)}) = n\Sigma^{-1}\mu$. Multiply by Σ:

$\sum_{i=1}^{n} x^{(i)} = n\mu$, so $\mu = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}$. □

(3) a) Show that $I(X,Y) = H(x) - H(x|y) = H(y) - H(y|x)$

where $I(x,y) = \int p(x,y) \ln\left(\frac{p(x,y)}{p(x)\,p(y)}\right) dx\,dy$

$$H(x) = -\int p(x) \ln(p(x))\, dx$$

and $H(x|y) = -\int p(x|y) \ln(p(x|y))\, dx\,dy$.

we have

$$I(x,y) = \int p(x,y) \ln\left(\frac{p(x,y)}{p(y)}\right) dx\,dy - \int p(x,y) \ln(p(x))\, dx\,dy$$

But $p(x|y) = \frac{p(x,y)}{p(y)}$, so this is

$$I(x,y) = \int p(x|y)\,p(y) \ln(p(x|y))\, dx\,dy - \int p(x,y) \ln(p(x))\, dx\,dy$$

$$= -\int dy\, p(y) \left(\int p(x|y) \ln(p(x|y))\, dx\right) - \int dx \ln(p(x)) \left(\int dy\, p(x,y)\right)$$

$$-\int dy\, p(y)\, H(x|Y=y) - \int p(x) \ln(p(x))\, dx$$

$$= -H(x|y) + H(x) = H(x) - H(x|y).$$

Because I is symmetric in X and Y we see that we can follow the same logic to get

$$I(x,y) = H(y) - H(y|x).$$

□

b) Prove that if X and Y are related by a bijection (i.e., if $X = f(Y)$ and $Y = f^{-1}(x)$, $I(x,y) = H(x) = H(y)$.

If $X = f(Y)$, then $p(x|y) = 1$ and $p(y|x) = 1$ (i.e., knowing X means we know Y and vice versa).

Thus, $H(x|y) = -\int 1 \ln(1)\, dy\, dx = -\int 0 = 0$

So $I(x,y) = H(x) = H(y)$.

c) Suppose we observe N samples $D = (x_1, \ldots x_N)$ from some unknown distribution. Define $\hat{p}(x)$ to be the empirical probability density estimate:

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} I(x = x_i)$$

Let $q(x|\theta)$ be the prob. density corresponding to some known model with parameter θ.

Show that the min K-L divergence

$$\min_{\theta} (D_{KL}(p||q)) \doteq \min_{\theta} \left(-\int \hat{p}(x) \ln \left(\frac{q(x|\theta)}{\hat{p}(x)} \right) dx \right)$$

is obtained by θ_{ML} given the data D.

we see that $D_{KL}(p||q) = \int \hat{p}(x) \ln \left(\frac{\hat{p}(x)}{q(x|\theta)} \right) dx$

$$= \int \hat{p}(x) \ln(\hat{p}(x))\, dx - \int \hat{p}(x) \ln(q(x|\theta))\, dx$$

$$= H(x) - \int \hat{p}(x) \ln(q(x|\theta))\, dx$$

$$= H(x) - \frac{1}{N} \int \sum_{i=1}^{N} \ln(q(x|\theta)) \cdot I(x = x_i)$$

$$= H(x) - 1/N \ln(q(x_i|\theta))$$

Thus we see that

$$\arg\min_\theta (D_{KL}(p\|q)) = \arg\min_\theta (H(x) - \tfrac{1}{N} q(x_i|\theta))$$

$$= \arg\max_\theta (\tfrac{1}{N} q(x_i|\theta))$$

But this is just maximizing the likelihood for q, so we are done!

d) Let $p = N(\mu, \sigma^2)$ be a Gaussian distribution and let q be any probability density with mean μ and variance σ^2. Prove that $H(q) \leq H(p)$, that is, the Gaussian has maximum entropy among all equivariant distributions.

We maximize a new function:

$$J(p) = -\int p(x) \ln(p(x))\,dx + \lambda_0\left(\int p(x)\,dx - 1\right)$$

$$+ \lambda_1\left(\int x\,p(x)\,dx - \mu\right) + \lambda_2\left(\int (x-\mu)^2 p(x)\,dx - \sigma^2\right)$$

(i.e, maximize H subject to the constraints that p is a pdf with mean μ and variance σ^2).

Then $\frac{dJ}{dp} = -\frac{p(x)}{p(x)} - \ln(p(x)) + \lambda_0 + \lambda_1 x + \lambda_2 (x-\mu)^2 = 0$

i.e, $p(x) = e^{-1+\lambda_0+\lambda_1 x + \lambda_2 (x-\mu)^2}$.

Then the constraint equations:

$\int p(x)\,dx = 1$, $\int x\,p(x)\,dx = \mu$, and

$\int (x-\mu)^2 p(x)\,dx = \sigma^2$

give:

(solving with Mathematica)

$\lambda_1 = 0$, $\lambda_0 = 1 + \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)$ and $\lambda_2 = -\frac{1}{2\sigma^2}$.

Thus,

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ as expected. □

(4) Weighted least squares

a) Let $w = [b \ \beta^T]^T = \underset{\beta,b}{\operatorname{argmin}} L(\beta, b) = \underset{\beta,b}{\operatorname{argmin}} \sum_{i=1}^{n} c_i (y_i - \beta^T x_i - b)^2$

Find a closed-form solution for w in terms of X and C.

Let $X = \begin{pmatrix} 1 \\ \vec{x}_i \end{pmatrix}$, $w^T = (b, \beta^T)$, and let x_i, C' Let C be the square matrices with y_i, c_i on the diagonal.

Then $L(\beta, b) = \sum_{i=1} c_i (y_i - w^T x_i)^2 = (y - w^T X)^T C (y - w^T X)$

$= (y^T - (X)^T w) C (y - w^T X) = (y^T C y - x^T w C y - y^T C w^T X + x^T w C (w^T)^T x)$

So $\frac{dL}{dw} = 2(-(X)^T C y + (X^T) C X w)$

So $\frac{dL}{dw} = 0$ gives

$(X^T) C X w = (X^T) C y$ i.e.,

$w = ((X^T) C X)^{-1} (X^T) C y$.

Next, model $y_i = \beta^T x_i + b + \varepsilon_i \ \forall i$

Then $y_i - \beta^T x_i - b$ is normally distributed $\forall i$ about 0 with variance σ^2.

Suppose now we wish to do a likelihood fit. Because we assume equal variance,

$$L(\beta, b) = e^{\frac{-(y_1 - \beta^T x_1 - b)^2}{2\sigma^2}} e^{\frac{-(y_2 - \beta^T x_2 - b)^2}{2\sigma^2}} \cdots$$

$$\text{so } \log L = -\sum_{i=1}^{n} \frac{(y_i - \beta^T x_i - b)^2}{2\sigma^2} = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \beta^T x_i - b)^2,$$

and minimizing $\log L$ is the same as maximizing $\sum_{i=1}^{n} (y_i - \beta^T x_i - b)^2$, that is, performing a least-squares for $C_i = 1 \; \forall i$.

b) Repeat part a), but with $y_i = \beta^T x_i + b + \varepsilon_i$, where $\varepsilon_i | x_i \sim N(0, \sigma_i^2)$.

We see that our value of w remains the same, but we must redefine C_i: As in (a),

$$L(\beta, b) = e^{\frac{-(y_1 - \beta^T x_1 - b)^2}{2\sigma_1^2}} e^{\frac{-(y_2 - \beta^T x_2 - b)^2}{2\sigma_2^2}} \cdots \text{ etc}$$

$$\text{Then } \log L = -\sum_{i=1}^{n} \frac{(y_i - \beta^T x_i - b)^2}{2\sigma_i^2} = -\sum_{i=1}^{n} C_i (y_i - \beta^T x_i - b)^2$$

$$\text{where } C_i = 1/2\sigma_i^2.$$

Then, as in (a), maximizing $\log L$ is the same as minimizing $\sum_{i=1}^{n} C_i (y_i - \beta^T x_i - b)^2$.

□

⑤ a) Show that the minimization problems for a soft-margin SVM:

$$\left\{ \begin{array}{l} \min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to } t^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \quad (i = 1 \ldots N) \end{array} \right\}$$

and

$$\left\{ \min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \max(0, 1 - t^{(i)}(w^T x^{(i)} + b)) \right\}$$

We note that $0 \leq \xi_i$ and $(1 - t^{(i)}(w^T x^{(i)} + b)) \leq \xi_i$.

Since we are minimizing the ξ_i, we see that at the solution, we have equality: either $\xi_i = (1 - t^{(i)}(w^T x^{(i)} + b)) \geq 0$ or $\xi_i = 0$ and $(1 - t^{(i)}(w^T x^{(i)} + b)) \leq 0$.

Thus, $\xi_i = \max(0, 1 - t^{(i)}(w^T x^{(i)} + b))$

So the problem becomes

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \max(0, 1 - t^{(i)}(w^T x^{(i)} + b)).$$

b) Let (w^*, b^*, ξ^*) be the solution of the second Problem. Show that if $\xi_i^* \geq 0$, then the distance from $x^{(i)}$ to $t^{(i)}((w^*)^T x + b^*) = 1$ is proportional to $-\xi_i^*$.

If $\xi_i^* > 0$, $\xi_i^* = (1 - t^{(i)}((w^*)^T x^{(i)} + b^*))$

Lemma: The distance from $x^{(i)}$ to the hyperplane $w^T x + b = 0$ is found by minimizing the Lagrangian

$$L = \|x - x^{(i)}\|^2 + 2\lambda(w^T x + b)$$

(let us use a multiplier of 2λ)

$$= |x|^2 - 2(x^{(i)} - \lambda w)^T x + 2\lambda b + |x^{(i)}|^2$$

So $\frac{\partial \mathcal{L}}{\partial x} = -2(x^{(i)} - \lambda w) + 2x$

and $\frac{\partial \mathcal{L}}{\partial x} = 0$ gives

$$x = (x^{(i)} - \lambda w)$$

So we have $\{w^T(x^{(i)} - \lambda w) + b = 0\}$

i.e., $\{w^T x^{(i)} + b - \lambda \|w\|^2 = 0\}$

So $$\lambda = \frac{w^T x^{(i)} + b}{\|w\|^2}$$

So the shortest distance is

$$\|x - x^{(i)}\| = \|\lambda w\| = \frac{\|w^T x^{(i)} + b\|}{\|w\|^2}. \quad \square$$

With this in mind, the distance from $x^{(i)}$ to $1 - t^{(i)}((w^*)^T x + b^*) = 0$ is

$$d = \frac{\|t^{(i)} w^T x^{(i)} + 1 - t^{(i)} b^*\|}{\|t^i w\|^2} =$$

$$= \frac{\|\xi_i^*\|}{\|t^{(i)} w\|^2}, \text{ i.e., } d \text{ is proportional to}$$

ξ_i.

c) Show that taking $C \to \infty$ gives the hard-margin SVM.

We see that taking $C = \infty$ enforces $\xi_i = 0 \ \forall i$, (otherwise, we cannot minimize anything). Thus,

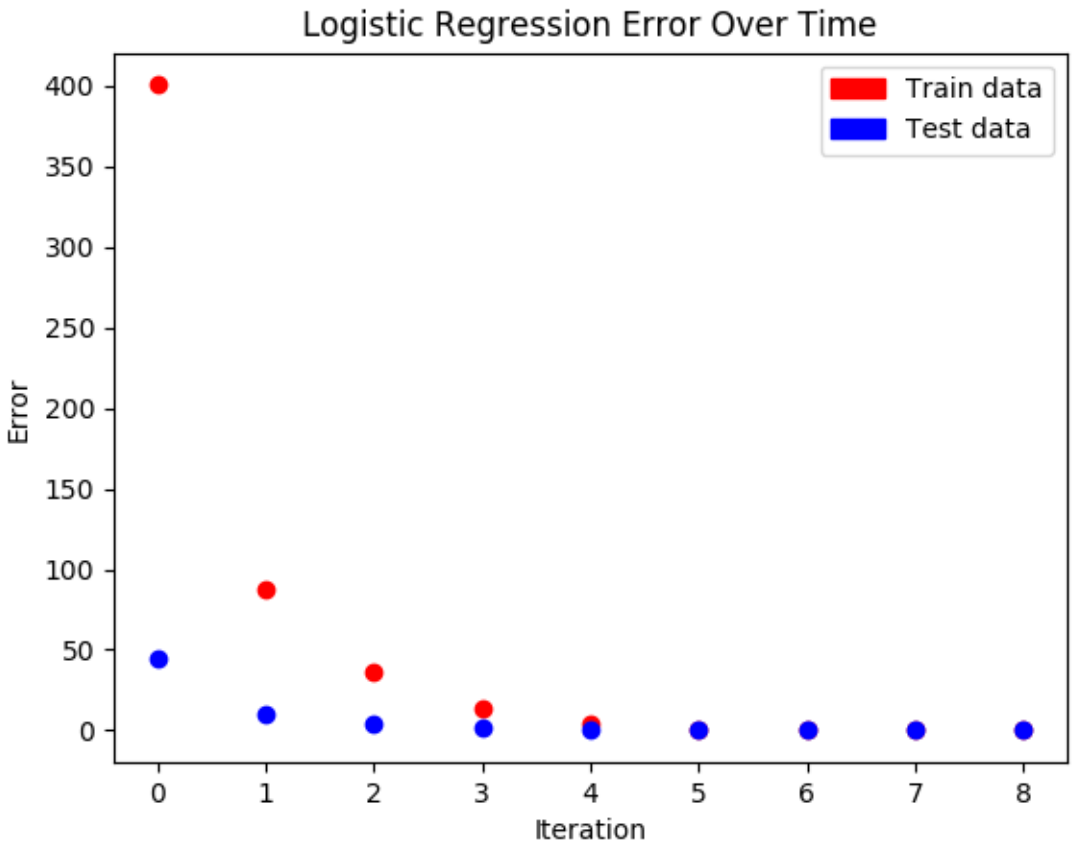
$1 - t^{(i)}(w^T x^{(i)} + b) \leq 0$, i.e.,

$t^{(i)}(w^T x^{(i)} + b) \geq 1$. This, however, is the hard-margin constraint, so we have shown that the hard-margin problem is equivalent to taking $C \to \infty$. $\square$

With the aid of Newton's Method, we perform logistic regression on a labeled dataset and plot our classifer error as a function of iterations. The algorithm converges after eight steps. Our code and plot is below:

```
import numpy
from numpy import *
from scipy.special import expit as expit
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

err = 10**8
epsilon=10**-8
w = array([0.0,0.0,0.0])

train_features_load = array(loadtxt('train_features.dat',unpack=True))
train_labels = array(loadtxt('train_labels.dat',unpack=True))
test_features_load = array(loadtxt('test_features.dat',unpack=True))
test_labels = array(loadtxt('test_labels.dat',unpack=True))

train_features = numpy.concatenate((numpy.ones((1,900),dtype=numpy.float64), train_features_load),
axis = 0)
test_features = numpy.concatenate((numpy.ones((1,100),dtype=numpy.float64), test_features_load),
axis = 0)

def sigmoid(xval,wval):
 z= xval @ wval
 return expit(-z)

def getlvalTrain(wval):
 lval = 0.00000000
 for i in range(train_features.shape[1]):
  lsum = -train_labels[i]*numpy.log(sigmoid((train_features[:,i]),wval))-(1-
train_labels[i])*numpy.log(1-sigmoid((train_features[:,i]),wval))
  lval = lval + lsum
 return lval

def getlvalTest(wval):
 lval = 0.00000000
 for i in range(test_features.shape[1]):
  lsum = -test_labels[i]*numpy.log(sigmoid((test_features[:,i]),wval))-(1-test_labels[i])*numpy.log(1-
sigmoid((test_features[:,i]),wval))
  lval = lval + lsum
 return lval

def getHval(wval):
 Hval = numpy.zeros((3,3),dtype=numpy.float64)
 for i in range(train_features.shape[1]):
```

```
  Hsum = numpy.outer((train_features[:,i]),(train_features[:,i]))*sigmoid((train_features[:,i]),wval)*(1-
sigmoid((train_features[:,i]),wval))
  Hval = Hval + Hsum
 return Hval

def getGradval(wval):
 Gradval = numpy.zeros((3),dtype=numpy.float64)
 for i in range(train_features.shape[1]):
  Gradsum = (train_features[:,i])*(sigmoid((train_features[:,i]),wval)-(train_labels[i]))
  Gradval = Gradval + Gradsum
 return Gradval

trainErrs = []
testErrs = []

steps = 0
while abs(err) > epsilon:
 steps = steps + 1
 wOld = w
 lvalOldTrain = getlvalTrain(wOld)
 lvalOldTest = getlvalTest(wOld)
 GradvalOld = getGradval(wOld)
 HvalOld = getHval(wOld)
 Hessinv = numpy.linalg.inv(HvalOld)
 w = w + Hessinv @ GradvalOld
 print(w)
 err = abs(lvalOldTrain - getlvalTrain(w))
 errTest = abs(lvalOldTest - getlvalTest(w))
 print('errTrain is')
 print(err)
 trainErrs.append(err)
 print('errTest is')
 print(errTest)
 testErrs.append(errTest)

plt.plot(range(steps),trainErrs,'ro', range(steps),testErrs,'bo')
plt.title('Logistic Regression Error Over Time')
red_patch = mpatches.Patch(color='red', label='Train data')
blue_patch = mpatches.Patch(color='blue', label='Test data')
plt.legend(handles=[red_patch,blue_patch])

plt.xlabel('Iteration')
plt.ylabel('Error')
plt.show()
```


Logistic Regression Error Over Time
Train data
Test data
Error
Iteration
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8

We implement hard- and soft-margin SVMs to classify whether or not a person with given characteristics has diabetes. We attach our code and output.

Code:

GNU nano 2.5.3 File: SVM.py

```
import numpy as np
import pandas as pd
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV
from sklearn import metrics

param_grid = dict(C=np.linspace(0.1, 2, 20))

df=pd.read_csv('diabetes_scale.csv', sep=',',header=None)
datavals=df.values
data=datavals.astype(np.float)

trainlabels = data[0:499,0]
trainfeatures = data[0:499,1:]
testlabels = data[500:767,0]
testfeatures = data[500:767,1:]

print("Soft-Margin SVM:")
SoftClf =
GridSearchCV(LinearSVC(loss='hinge',penalty='l2',random_state=42),param_grid=param_grid, cv=5)
SoftClf.fit(trainfeatures,trainlabels)
print("The best parameter is %s with a training accuracy of %f"
 % (SoftClf.best_params_, SoftClf.best_score_))
SoftTestGuess=SoftClf.predict(testfeatures)
errs = metrics.accuracy_score(testlabels,SoftTestGuess)
print("The test accuracy is %f"
 % (errs))

print("Hard-Margin SVM:")
HardClf = LinearSVC(C=1000000,loss='hinge',penalty='l2',random_state=42)
HardClf.fit(trainfeatures,trainlabels)
HardTrainGuess=HardClf.predict(trainfeatures)
TrainErrs = metrics.accuracy_score(trainlabels,HardTrainGuess)
print("The train accuracy is %f"
 % (TrainErrs))
HardTestGuess=HardClf.predict(testfeatures)
TestErrs = metrics.accuracy_score(testlabels,HardTestGuess)
print("The test accuracy is %f"
 % (TestErrs))
```

Output:

Soft-Margin SVM:
The best parameter is {'C': 0.5} with a training accuracy of 0.769539
The test accuracy is 0.775281
Hard-Margin SVM:
The train accuracy is 0.739479
The test accuracy is 0.734082

We note that the soft-margin SVM performed better on both the training and testing set. This is likely because the soft-margin SVM is more robust, and is able to ignore noise that the hard-margin SVM is forced to consider. Thus, the hard-margin SVM will fit to the noise and select a less-than ideal decision boundary, while the soft-margin SVM is able to ignore noise when defining its margin.