Stat 6910, Section 003
Statistical Learning and Data Mining II
Fall 2018

Homework 3

Jared Hansen

A-number: `A01439768`

e-mail: `jrdhansen@gmail.com`

1. **Maximum Likelihood Estimation (14 pts)**

(a) Let $X_1, \dots, X_n$ be i.i.d. sample from a Poisson distribution with parameter λ, i.e.

$$P(X = x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}.$$

i. (2 pts) Write down the likelihood function $L(\lambda)$.

$$\begin{aligned} L(\lambda) &= \prod_{i=1}^{n} f(X_i|\lambda) \\ &= \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \end{aligned}$$

ii. (2 pts) Write down the log-likelihood function $\ell(\lambda)$.

$$\begin{aligned} \ell(\lambda) &= \ln(L(\lambda)) \\ &= \sum_{i=1}^{n} \ln(\frac{\lambda^{x_i} e^{-\lambda}}{x_i!}) \\ &= \sum_{i=1}^{n} \ln(\lambda^{x_i}) + \sum_{i=1}^{n} \ln(e^{-\lambda}) - \sum_{i=1}^{n} \ln(x_i!) \\ &= \sum_{i=1}^{n} \Big(x_i \ln(\lambda) - \lambda - \ln(x_i!) \Big) \\ \ell(\lambda) &= \left(\ln(\lambda) \sum_{i=1}^{n} x_i \right) - \left(n\lambda \right) - \left(\sum_{i=1}^{n} \ln(x_i!) \right) \end{aligned}$$

iii. (3 pts) Find the maximum likelihood estimate (MLE) of the parameter λ.

To find the MLE of λ we will do the following: $\frac{d}{d\lambda}(\ell(\lambda)) = 0$, solve for $\hat{\lambda}$.

$$\begin{aligned} \frac{d}{d\lambda}(\ell(\lambda)) &= \frac{1}{\lambda} \sum_{i=1}^{n} x_i - n \\ 0 &\overset{\text{set}}{=} \frac{1}{\lambda} \sum_{i=1}^{n} x_i - n \\ n &= \frac{1}{\hat{\lambda}} \sum_{i=1}^{n} x_i \\ \hat{\lambda} &= \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}, \text{ where } \bar{x} \text{ is the sample mean} \end{aligned}$$

(b) (7 pts) Let $X_1, \ldots, X_n$ be an i.i.d. sample from an exponential distribution with the density function

$$p(x;\beta) = \frac{1}{\beta}e^{-\frac{x}{\beta}},\ 0 \le x < \infty.$$

Find the MLE of the parameter β. Given what you know about the role that β plays in the exponential distribution, does the MLE make sense? Why or why not?

Let $\lambda = (\frac{1}{\beta})$ (we'll substitute $\beta = \frac{1}{\lambda}$ back in at the end)
Now $p(x;\lambda) = \lambda e^{-\lambda x}$, $0 \le x < \infty$

$$\begin{aligned} L(\lambda) &= \prod_{i=1}^{n} e^{-\lambda x_i} \\ &= \lambda^n \prod_{i=1}^{n} \lambda e^{-\lambda x_i} \end{aligned}$$

Now we'll use log-likelihood:

$$\begin{aligned} \ell(\lambda) &= ln\left(\lambda^n \prod_{i=1}^{n} \lambda e^{-\lambda x_i}\right) \\ &= \ln(\lambda^n) + \ln\left(\sum_{i=1}^{n} e^{-\lambda x_i}\right) \\ &= n\ln(\lambda) + \sum_{i=1}^{n} \ln\left(e^{-\lambda x_i}\right) \\ &= n\ln(\lambda) + \sum_{i=1}^{n} -\lambda x_i \\ \ell(\lambda) &= n\ln(\lambda) - \lambda\sum_{i=1}^{n} x_i \end{aligned}$$

Now, taking the derivative wrt λ:

$$\begin{aligned} \frac{d}{d\lambda}(\ell(\lambda)) &= \frac{n}{\lambda} - \sum_{i=1}^{n} x_i \overset{\text{set}}{=} 0 \\ \frac{n}{\hat{\lambda}} &= \sum_{i=1}^{n} x_i \\ \hat{\lambda} &= \frac{n}{\sum_{i=1}^{n} x_i} \\ \hat{\lambda} &= (\bar{x})^{-1} \\ \hat{\lambda} &= \frac{1}{\bar{x}}, \text{ and since } \lambda = \frac{1}{\beta} \text{ we'll have:} \\ \hat{\beta} &= \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \end{aligned}$$

The exponential distribution estimates the time/space between successes, typically for continuous random variables. The quantity λ is referred to as the rate parameter. Since λ gives $\frac{time}{success}$, inverting this would logically give the expected number of successes. Since that's what $\bar{x}$ is, this MLE for β makes sense.

2. **Logistic Regression as ERM (5 pts).** Consider training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ for binary classification and assume $y_i \in \{-1, 1\}$. Show that if $L(y, t) = \log(1 + \exp(-yt))$,then

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, \boldsymbol{w}^T\boldsymbol{x}_i + b)$$

is proportional to the negative log-likelihood for logistic regression. Therefore ERM with the logistic loss is equivalent to the maximum likelihood approach to logistic regression.
Clarification: In the above expression, y is assumed to be -1 or 1. In the notes, we had $y \in \{0, 1\}$. So all you need to do is rewrite the negative log-likelihood for logistic regression using the ± 1 label convention and simplify that formula until it looks like the formula above.

$\eta(x) = P(Y = 1|X = x) = \frac{1}{1+e^{-(\boldsymbol{w}^T\boldsymbol{x}_i+b)}}$,

1 - $\eta(x) = P(Y = -1|X = x) = \frac{1}{1+e^{(\boldsymbol{w}^T\boldsymbol{x}_i+b)}}$

$\left(\frac{1}{1+e^{-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}}\right)$ for y $\in \{-1, 1\}$, where this is our conditional pmf of $Y|X$.
In other words, it's $P(Y = y|X = x)$ for y $\in \{-1, 1\}$

Let t $= \boldsymbol{w}^T\boldsymbol{x}_i + b$

So L$(\mathrm{y}_i, t) = \prod_{i=1}^{n} \frac{1}{1+e^{-y_i t}}$

$\ln(L(y_i, t)) = \ell(y_i, t) = \sum_{i=1}^{n} \ln\left(\frac{1}{1+e^{-y_i t}}\right)$

$\ell(y_i, t) = \sum_{i=1}^{n}\left(\ln(1) - \ln(1 + e^{-y_i t})\right) = -\sum_{i=1}^{n} \ln(1 + e^{-y_i t})$

So -$\ell(y_i, t) = \sum_{i=1}^{n} \ln(1 + e^{-y_i t})$, now substitute in $\left(t = \boldsymbol{w}^T\boldsymbol{x}_i + b\right)$ to $-\ell(y_i, t)$

$\ell(y_i, \boldsymbol{w}, b) = \sum_{i=1}^{n} \ln\left(1 + e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i+b}\right) = \left(L(y_i, t) = \ln\left(1 + e^{-y_i t}\right)\right)$

This quantity $L(y_i, t)$ is proportional to $\frac{1}{n}\sum_{i=1}^{n} Loss(y_i, \boldsymbol{w}^T\boldsymbol{x}_i + b)$
which is the negative log-likelihood for logistic regression.The difference comes from the $\frac{1}{n}$ term.

3. **Convexity and Optimization (24 pts).** Let $f : \mathbb{R}^d \to \mathbb{R}$.

(a) (7 pts) Show that if f is strictly convex, then f has at most one global minimizer.

This will be a proof by contradiction.

Suppose $f(x^*{}_1) = g(x^*{}_2) = m$ where m is a global minimum (meaning there are two global minima), and $x^*{}_1 \neq x^*{}_2$.

Let $x = \left[tx^*{}_1 + (1-t)x^*{}_2 \right]$ be a point between $x^*{}_1$ and $x^*{}_2$.

Using the definition of convexity,

$$\begin{aligned} \left[f(x) = f(tx^*{}_1 + (1-t)x^*{}_2) \right] &< tx^*{}_1 + (1-t)x^*{}_2 \\ &< tm + (1-t)m \\ &< tm + m - tm \\ f(x) &< m \end{aligned}$$

Since we found a value, $f(x)$, such that $f(x) < m$, we found that $\exists$ a value less than the global minimum. However, by definition, there *cannot be* something less than the global minimum. By this contradiction we can conclude that f is strictly convex when f has at most one global minimizer.

(b) (7 pts) Use the Hessian to give a simple proof that the sum of two convex functions is convex. You may assume that the two functions are twice continuously differentiable.

Let $f(x)$ and $g(x)$ both be convex functions. Also, assume $f(x)$ and $g(x)$ are twice continuously differentiable. By definition, since $f(x)$ is twice continuously differentiable we know that at some local min, call this ${x^*}_1$, $\nabla^2 f({x^*}_1)$ is a positive semi-definite matrix. The same holds true for $g(x)$ at some local minima ${x^*}_2$: $\nabla^2 g({x^*}_2)$ is a positive semi-definite matrix.

Additionally, we know that:

- $\text{result}_1 = \left[\boldsymbol{z}^T \big[\nabla^2 f({x^*}_1)\big] \boldsymbol{z} \geq 0, \forall \boldsymbol{z} \in \mathbb{R}^d \right]$
- $\text{result}_2 = \left[\boldsymbol{z}^T \big[\nabla^2 g({x^*}_2)\big] \boldsymbol{z} \geq 0, \forall \boldsymbol{z} \in \mathbb{R}^d \right]$

We know that $[result_1 + result_2] \geq 0$ since both operands are ≥ 0.

We know that $\nabla^2 f({x^*}_1)$ and $\nabla^2 g({x^*}_2)$ are both symmetric matrices $\Rightarrow \Big[[\nabla^2 f({x^*}_1)] + [\nabla^2 g({x^*}_2)]\Big]$ is also a symmetric matrix.

For shorter notation, let:

- $\boldsymbol{A} = [\nabla^2 f({x^*}_1)]$
- $\boldsymbol{B} = [\nabla^2 g({x^*}_2)]$

From properties of matrix-vector multiplication, we know that $\boldsymbol{z}^T(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{z} = \big[\boldsymbol{z}^T\boldsymbol{A}\boldsymbol{z}\big] + \big[\boldsymbol{z}^T\boldsymbol{B}\boldsymbol{z}\big]$. Since we know that $\boldsymbol{z}^T\boldsymbol{A}\boldsymbol{z} \geq 0$ and $\boldsymbol{z}^T\boldsymbol{B}\boldsymbol{z} \geq 0 \implies \boldsymbol{z}^T(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{z} \geq 0 \implies$ the matrix $\boldsymbol{A}+\boldsymbol{B}$ is positive semi-definite.
The matrix $(\boldsymbol{A}+\boldsymbol{B})$ is the Hessian for the function $\big(f(x)+g(x)\big)$ since $\nabla\big(f(x)+g(x)\big) = \nabla f(x) + \nabla g(x)$ and $\nabla^2\big(f(x)+g(x)\big) = \nabla^2 f(x) + \nabla^2 g(x) = (\boldsymbol{A}+\boldsymbol{B})$ which is positive semi-definite $\implies \Bigg(f(x)+g(x)\Bigg)$ is a convex function since it has a positive semi-definite Hessian.
Thus, we have used the Hessian to give a simple proof that the sum of two convex functions is convex.

SLDM hw 3, 3c_i

Consider the fctn $f(\underset{\sim}{x}) = \frac{1}{2}\underset{\sim}{x}^T A \underset{\sim}{x} + \underset{\sim}{b}^T\underset{\sim}{x} + c$ where $A_{d\times d}$ is symmetric. Derive the Hessian of f. Under what conditions on A is f convex? Strictly convex?

- $\nabla^2 f(x) = \frac{\partial}{\partial x}(\nabla f(x)) = \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}(f(x))\right)$ since all we have is x, no other indep. variables.

- $\nabla f(x) = \frac{\partial}{\partial x}(f(x)) = \underbrace{\frac{1}{2}\cdot\frac{\partial}{\partial x}(\underset{\sim}{x}^T A \underset{\sim}{x})} + \underbrace{\frac{\partial}{\partial x}(\underset{\sim}{b}^T\underset{\sim}{x})} + \frac{\partial}{\partial x}(c) = \boxed{A\underset{\sim}{x} + \underset{\sim}{b}}$

- First, let's look at this: $\frac{\partial}{\partial x}(\underset{\sim}{b}^T\underset{\sim}{x}) = \boxed{\underset{\sim}{b}.}$ Let's show this, where $\underset{\sim}{b}^T$ is 1xd and $\underset{\sim}{x}$ is dx1.

$\underset{\sim}{b}^T\underset{\sim}{x} = \sum_{i=1}^{d} b_i x_i \longrightarrow \frac{\partial}{\partial x}(\underset{\sim}{b}^T\underset{\sim}{x}) = \frac{\partial}{\partial x}\left(\sum_{i=1}^{d} b_i x_i\right) = \sum_{i=1}^{d} b_i\left(\frac{\partial}{\partial x}(x_i)\right) = \sum_{i=1}^{d} b_i = \underset{\sim}{b}$

- Second, let's look at this: $\frac{1}{2}\cdot\frac{\partial}{\partial x}(\underset{\sim}{x}^T A \underset{\sim}{x}) = \boxed{\frac{1}{2}\cdot\frac{2}{1}A\underset{\sim}{x} = A\underset{\sim}{x}.}$ Let's show this, where $\underset{\sim}{x}$ is dx1 and A is dxd.

Let $\alpha = \underset{\sim}{x}^T A \underset{\sim}{x} \longrightarrow$ by defn: $\alpha = \sum_{j=1}^{d}\sum_{i=1}^{d} A_{ij}x_i x_j$

$\forall k \in \{1,2,\dots,d\}$

Differentiating with respect to the kth element of $\underset{\sim}{x}$ we have $\frac{\partial\alpha}{\partial x_k} = \sum_{j=1}^{d} A_{kj}x_j + \sum_{i=1}^{d} A_{ik}x_i$

Consequently, $\frac{\partial\alpha}{\partial x} = x^T A^T + x^T A = x^T(A^T + A)$. But since A^T is symmetric $\Rightarrow A^T = A$.

Therefore, $\frac{\partial\alpha}{\partial x} = x^T(A^T + A) = x^T(A + A) = \boxed{2A\underset{\sim}{x}}$

- It is obvious that $\partial/\partial x\,(c) = 0$ since c is a scalar term with no dependence on x.

- Thus, $\nabla f(x) = \underset{\sim}{x}^T A + \underset{\sim}{b}^T$. Now, find $\nabla^2 f(x)$ by $\partial/\partial x(\underset{\sim}{x}^T A + \underset{\sim}{b}^T) = \underbrace{\frac{\partial}{\partial x}(\underset{\sim}{x}^T A)} + \frac{\partial}{\partial x}(\underset{\sim}{b}^T)$.

- Since $\underset{\sim}{b}^T$ has no dependence on x $\Rightarrow \partial/\partial x(\underset{\sim}{b}^T) = 0$.

- Now let's examine $\partial/\partial x(\underset{\sim}{x}^T A) = \boxed{A.}$ Let's show this, where $\underset{\sim}{x}^T$ is 1xd and A is dxd.

$$\underset{\sim}{x}^T A = [x_1, x_2, \dots, x_n]\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn}\end{bmatrix} = [x_1a_{11} + x_2a_{21} + \dots + x_na_{n1},\ x_1a_{12} + x_2a_{22} + \dots + x_na_{n2},\ x_1a_{1n} + \dots + x_na_{nn}]$$

$$\frac{\partial}{\partial x_1}(\underset{\sim}{x}^T A) = [\boxed{a_{11}} + x_2a_{21} + \dots + x_na_{n1},\ \boxed{a_{12}} + x_2a_{22} + \dots + x_na_{n2}, \dots, \boxed{a_{1n}} + x_2a_{2n} + \dots + x_na_{nn}]$$

elements of first row of A

Thus, $\frac{\partial}{\partial x} = \begin{bmatrix}\text{row 1 of } A \\ \text{row 2 of } A \\ \vdots \\ \text{row n of } A\end{bmatrix} = A \longrightarrow$ $\boxed{\text{Thus, the Hessian of f is the matrix A.}}$

all x

SLDM hw3, 3c ii ... Under what conditions on A is f convex? Strictly convex?

- We've shown that the Hessian of f is the matrix A.
- From the course notes, it is given that:
  - f is convex $\Leftrightarrow$ $\nabla^2 f(\underline{x})$ is positive semidefinite $\forall \underline{x} \in \mathbb{R}^d$.
  - f is strictly convex $\Leftarrow$ $\nabla^2 f(\underline{x})$ is positive definite $\forall \underline{x} \in \mathbb{R}^d$.

- Therefore, since $\nabla^2 f(\underline{x}) = A$, we know that:
  - f is convex if and only if A is positive semidefinite $\forall \underline{x} \in \mathbb{R}^d$.
  - If A is positive definite $\forall \underline{x} \in \mathbb{R}^d$ then f is strictly convex.

SLDM hw3, prob 3d. Let $J(\theta)$ be a twice continuously differentiable function. Derive the update step for Newton's method from the second-order approximation of $J(\theta)$.

- Update step (what we want at the end): $\theta_{t+1} = \theta_t - [\nabla^2 J(\theta_t)]^{-1}[\nabla J(\theta_t)]$

- 2nd order approximation: $J(\theta) = J(\theta_t) + [\nabla J(\theta_t)]^T[\theta - \theta_t] + \frac{1}{2}[\theta-\theta_t]^T[\nabla^2 J(\theta_t)][\theta - \theta_t]$
(to be minimized)

- In order to minimize this 2nd-order approximation, we'll take the gradient, $\nabla J(\theta)$, $\overset{\text{set}}{=} 0$, and solve for θ (which θ will be the θ_{t+1} in the update step).

- First, let's expand $J(\theta)$.

$$J(\theta) = \underbrace{J(\theta_t) + [[\nabla J(\theta_t)]^T[\theta]] - [[\nabla J(\theta_t)]^T[\theta_t]]}_{\text{"stuff"}} + \frac{1}{2}\left[[\theta]^T[\nabla^2 J(\theta_t)] - [\theta_t]^T[\nabla^2 J(\theta_t)]\right][\theta - \theta_t]$$

$$J(\theta) = \text{stuff} + \frac{1}{2}\left[[\theta]^T[\nabla^2 J(\theta_t)][\theta] - [\theta_t]^T[\nabla^2 J(\theta_t)][\theta] - [\theta]^T[\nabla^2 J(\theta_t)][\theta_t] + + [\theta_t]^T[\nabla^2 J(\theta_t)][\theta_t]\right]$$

- Now, take the gradient of $J(\theta)$ wrt θ. This means θ_t is a constant relative to θ_t.

$$\nabla J(\theta) = \underbrace{0 + [\nabla J(\theta_t)] - 0}_{\nabla_\theta(\text{stuff})} + \frac{1}{2}\cdot\frac{2}{1}[\nabla^2 J(\theta_t)][\theta] - \frac{1}{2}[\nabla^2 J(\theta_t)][\theta_t]^T - \frac{1}{2}[\nabla^2 J(\theta_t)][\theta_t] - \frac{1}{2}(0)$$

* Note: I took the gradient using properties given in the linear algebra resources posted on Canvas.

Now, $\nabla J(\theta) = \nabla J(\theta_t) + [\nabla^2 J(\theta_t)][\theta] - [\nabla^2 J(\theta_t)][\theta_t] \overset{\text{set}}{=} 0$, solve for θ.

$[\nabla^2 J(\theta_t)]^{-1}[\nabla^2 J(\theta_t)][\theta] = [\nabla^2 J(\theta_t)]^{-1}[\nabla^2 J(\theta_t)][\theta_t] - [\nabla^2 J(\theta_t)]^{-1}[\nabla J(\theta_t)]$, reduces to

$I[\theta] = [\theta_t] - [\nabla^2 J(\theta_t)]^{-1}[\nabla J(\theta_t)]$ showing that

θ_{t+1}

$$\boxed{\text{The update step } [\theta_{t+1}] = [\theta_t] - [\nabla^2 J(\theta_t)]^{-1}[\nabla J(\theta_t)] \text{, which was derived by taking the gradient of } J(\theta) \text{, setting equal to 0, and solving for } \theta \text{ (which is } \theta_{t+1}\text{).}}$$

4. **Logistic regression Hessian (15 pts).** Determine a formula for the gradient and the Hessian of the regularized logistic regression objective function. Argue that the objective function

$$J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||^2$$

is convex when $\lambda \geq 0$, and that for $\lambda > 0$, the objective function is strictly convex.
Hints: The following conventions and properties regarding vector differentiation may be useful. The properties can be easily verified from definitions. Try to avoid long, tedious calculations.

- If $f : \mathbb{R}^n \to \mathbb{R}$, then we adopt the convention

$$\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}} := \nabla f(\boldsymbol{z}).$$

- If $f : \mathbb{R}^n \to \mathbb{R}^m$, adopt the convention

$$\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}^T} := \left(\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)^T.$$

- Given these conventions, it follows that the Hessian H of J is

$$H = \frac{\partial}{\partial \boldsymbol{\theta}^T}\left(\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right),$$

  which is often denoted more concisely as

$$\frac{\partial^2 J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}.$$

- (One form of a multivariate chain rule): If $f(\boldsymbol{z}) = g(h(\boldsymbol{z}))$ where $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}$, then

$$\nabla f(\boldsymbol{z}) = \nabla h(\boldsymbol{z}) \cdot g'(h(\boldsymbol{z})).$$

SLDM hw3, prob4;

$+\lambda\|\theta\|^2$

- $J(\theta) = -\ell(\theta) + \lambda\|\theta\|^2$, where $\ell(\theta) = \sum_{i=1}^{n}\left[y_i \ln\left(\frac{1}{1+\exp(-\theta^T x_i)}\right) + (1-y_i)\ln\left(\frac{\exp(-\theta^T x_i)}{1+\exp(-\theta^T x_i)}\right)\right]$

- $\ell(\theta) = \sum_{i=1}^{n}\left[y_i\left(\overset{0}{\cancel{\ln(1)}} - \ln(1+\exp(-\theta^T x_i))\right) + (1-y_i)\left(\ln(\exp(-\theta^T x_i)) - \ln(1+\exp(-\theta^T x_i))\right)\right] + \lambda\|\theta\|^2$

$$\ell(\theta) = \sum_{i=1}^{n}\left[\underbrace{y_i\left(-\ln(1+\exp(-\theta^T x_i))\right)} + (-\theta^T x_i) - \ln(1+\exp(-\theta^T x_i)) + y_i(\theta^T x_i) \underbrace{+\, y_i \cdot \ln(1+\exp(-\theta^T x_i))}\right] + \lambda\|\theta\|^2$$

$$\ell(\theta) = \sum_{i=1}^{n}\left[-\ln(1+\exp(-\theta^T x_i)) + (y_i - 1)(\theta^T x_i)\right] + \lambda\|\theta\|^2$$

- $\nabla J(\theta) = \nabla(-\ell(\theta) + \lambda\|\theta\|^2) = \nabla(-\ell(\theta)) + \nabla(\lambda\|\theta\|^2)$

- First, what is $\nabla_\theta(\lambda\|\theta\|^2) = \lambda\nabla_\theta(\theta^T\theta) = 2\lambda\theta$ ← by properties in lin Alg. review from Canvas.

- Now, what is $\nabla(-\ell(\theta)) = ?$

$$\nabla(-\ell(\theta)) = \nabla_\theta\left[-\sum_{i=1}^{n}\left(-\ln(1+\exp(-\theta^T x_i)) + (y_i-1)(\theta^T x_i)\right)\right]$$

$$= -\sum_{i=1}^{n}\left[+\frac{\exp(-\theta^T x_i)(x_i)}{1+\exp(-\theta^T x_i)} + (y_i - 1)(x_i)\right]$$

- Thus, $\boxed{\nabla J(\theta) = -\sum_{i=1}^{n}\left[\frac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))}(x_i) + (y_i-1)(x_i)\right] + 2\lambda\theta}$

SLDM hw3, Prob 4ii

- Now, to find $\nabla^2 J(\theta)$: $\nabla^2 J(\theta) = \nabla_\theta(\nabla J(\theta))$, and we know that

$$\nabla J(\theta) = -\sum_{i=1}^{n}\left[-\frac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))}(x_i) + (y_i-1)(x_i)\right] + 2\lambda\theta$$

- First, what is $\nabla_\theta(2\lambda\theta) = ?$ $\longrightarrow 2\lambda \cdot \nabla_\theta(\theta) \longrightarrow \nabla_\theta(\theta_1) = [1\ 0 \cdots 0]^T$

$\longrightarrow \nabla_\theta(\theta_2) = [0\ 1\ 0 \cdots 0]^T$

$\vdots$

$\longrightarrow \nabla_\theta(\theta_n) = [0 \cdots 0\ 1]^T$

- So $$\nabla_\theta(\theta) = \begin{bmatrix} [\nabla_{\theta_1}(\theta_1)]^T \\ [\nabla_{\theta_2}(\theta_2)]^T \\ \vdots \\ [\nabla_{\theta_n}(\theta_n)]^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & & \vdots \\ 0 & & \ddots & & \vdots \\ \vdots & & & 0 & \vdots \\ \vdots & & & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} = \text{Identity matrix}$$

- Thus, $\nabla_\theta(2\lambda\theta) = 2\lambda\nabla_\theta(\theta) = 2\lambda I$

- What is $$\nabla_\theta\left[-\sum_{i=1}^{n}\frac{-\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))}(x_i)\right] + \nabla_\theta\left[\cancel{-(y_i-1)(x_i)}\right]^{0} = ?$$

What is $\nabla_\theta[\exp(-\theta^T x_i)] = -x_i\exp(-\theta^T x_i)$

So $\nabla_\theta\left[\frac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))}\right]$ by the quotient rule $=$

$$\left[\frac{(1+\exp(-\theta^T x_i))(-x_i\exp(-\theta^T x_i)) + +(\exp(-\theta^T x_i))(-x_i\exp(-\theta^T x_i))}{(1-\exp(-\theta^T x_i))^2}\right]$$

$$= \frac{-x_i\exp(-\theta^T x_i) - x_i(\exp(-\theta^T x_i))^2 + + x_i(\exp(-\theta^T x_i))^2}{(1-\exp(-\theta^T x_i))^2} = \left[\frac{-x_i\exp(-\theta^T x_i)}{(1-\exp(-\theta^T x_i))^2}\right]^T$$

SLDM hw3, Prob4iii

Now $\nabla_\theta = -\sum_{i=1}^{n} \dfrac{-x_i(x_i^T)\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))^2} + 2\lambda I$

$$= + + \sum_{i=1}^{n} \left[x_i (x_i)^T\right]\left[\frac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))^2}\right] + 2\lambda I$$

• When we sum over all n we end up with

$\nabla_\theta = X^T D X$ where the matrix $\mathbf{X}^T$ is the training data in the form $\mathbf{X}^T = [\underset{\sim}{x}_1^T \ \underset{\sim}{x}_2^T \cdots \underset{\sim}{x}_n^T]$, and the matrix

$\mathbf{X} = \begin{bmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \\ \vdots \\ \underset{\sim}{x}_n \end{bmatrix}$ and $\mathbf{D} = \text{diagonal}\left(\dfrac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))^2}\right)$.

Finally, $2\lambda I = \begin{bmatrix} 2\lambda & 0 & \cdots & 0 & 0 \\ 0 & 2\lambda & \ddots & \vdots & \vdots \\ 0 & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & 2\lambda & 0 \\ 0 & 0 & \cdots & 0 & 2\lambda \end{bmatrix}$

• $\boxed{\nabla_\theta^2 (J(\theta)) = X^T D X + 2\lambda I \text{ where all of the matrices are } n \times n}$

SLDM hw3, prob 4 iiii

- We know that a function is convex iff its Hessian is PSD.

  Examining D, it's numerator, $\exp(-\theta^T x_i)$ will always be ≥ 0 since $\lim_{x\to\infty} e^{-x} = 0$ and $e^{-x} > 0$ for all other x. Also, since the denominator is $(1 + "\geq 0")^2$, this is ≥ 1. Therefore, the diagonals of D are $\frac{"\geq 0"}{"\geq 1"} = "\geq 0"$

  Now, we know that $X^T X$ will be positive $\overset{\text{semi}}{\wedge}$ definite since all of its diagonal And are multiplied by ≥ 0 entries entries, $x_i^T x_i \geq 0$ ($=0$ when $x_i = 0$, >0 when $x_i \neq 0$). It's a known property that a matrix is PSD iff $d_i \geq 0$ $\forall i \in \{1, 2, \dots, n\}$ where the d_i are the diagonal entries.

- Since $X^T X$ is PSD $\Longrightarrow$ $\underset{\sim}{z}^T (X^T X) \underset{\sim}{z} > 0$ $\forall z \in \mathbb{R}^n$. In order for the overall Hessian to be PSD $\Longrightarrow$ $\underset{\sim}{z}^T((X^T D X) + 2\lambda I)\underset{\sim}{z} \geq 0$ $\forall z \in \mathbb{R}^n$.

  If the diagonals of $X^T D X$ are all ≥ 0 $\Longrightarrow$ the diagonals of $2\lambda I \geq 0$ in order for the Hessian to be PSD. So long as $\lambda \geq 0$, $X^T D X + 2\lambda I$ is PSD $\rightarrow$ $J(\theta)$ is convex for $\lambda \geq 0$.

- Similar logic holds for $\lambda > 0$, except that we're guaranteed for all diagonals of $X^T D X + 2\lambda I$ to be > 0 since all diagonals of $2\lambda I$ will be >0 and all diags of $X^T D X \geq 0$ $\Longrightarrow$ diags of $X^T D X + 2\lambda I$ are all >0, a known property of a PD matrix. In that case, since the Hessian is PD $\Longrightarrow$ when $\lambda > 0$ $J(\theta)$ is strictly convex.

5. **ERM and Stochastic Gradient Descent (10 pts)**. Given training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, define the empirical risk for either a regression or classification problem as

$$\hat{R}(f_{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i . f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right).$$

Write pseudocode describing how you would implement stochastic gradient descent to minimize $\hat{R}(f_{\boldsymbol{\theta}})$ with respect to θ. Assume a fixed mini-batch size of m and assume that the step size α is fixed for each epoch.

Initializing Constants

```
• maxIter = max number of iterations
• m = fixed mini batch size
• $\alpha$ = fixed step size
• n = length($\boldsymbol{x}$), aka number of observations
• nBatch = $\frac{n}{m}$, the number of mini-batches, assuming n%m = 0

• Now, initialize $\theta$ as $\theta_0$ (we haven't gone into specifics for how to do this, so let $\theta_0$ be a guess.)

• for i in range(maxIter):

    – Randomly permute the data. By rearranging the order of the $(\boldsymbol{x_i}, y_i)$ we can just step
      through the n observations in mini-batches of size m and the mini batches will be random.

    – for j in range(nBatch):
        ∗ compute gradient for each pair $(\boldsymbol{x_i}, y_i)$ in the $j^{th}$ mini batch
        ∗ sum all m of these individual gradients from the $j^{th}$ mini batch
        ∗ Updated $\theta_{t+1}$ = Current$\theta_t - (\alpha)$(sum of individual $\nabla_j$)
    – if(some stop condition is achieved before i == maxIter): break
• return $\theta \longleftarrow$ at this point we'll have the $\theta$ that minimizes our objective function
  $\hat{R}(f_{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i . f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)$
```

6. **Handwritten digit classification with logistic regression (22 pts).** Download the file `mnist_49_3000.mat` from the Homework 3 assignment. This is a Matlab data file that contains a subset of the MNIST handwritten digit dataset, which is a well-known benchmark dataset for classification. This subset contains examples of the digits 4 and 9.
   The data file contains variables $\boldsymbol{x}$ and y, with the former containing the image of the digit (reshaped into column vector form) and the latter containing the corresponding label ($y \in \{-1, 1\}$). To visualize an image, you will need to reshape the column vector into a square image. You should be able to find methods for loading the data file and for reshaping the vector in your preferred language through a Google search. If you're struggling to find something that works, you may ask for suggestions on Piazza.
   Implement Newton's method to find a minimizer of the regularized negative log likelihood for logistic regression: $J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$. Try setting $\lambda = 10$. Use the first 2000 examples as training data and the last 1000 as test data.

   (a) (6 pts) Report the test error, your termination criterion (you may choose), how you initialized $\boldsymbol{\theta}_0$, and the value of the objective function at the optimum.

   - Test error: ≈ 0.055
   - Termination criterion: maxIterations $= 2$ (this is what gave the lowest test error using selected $\lambda = 1.0$.)
   - Initialization of $\boldsymbol{\theta}_0$: $\boldsymbol{\theta}_0 = \begin{bmatrix} b = 1 \\ w_1 = 0 \\ w_2 = 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ w_d = 0 \end{bmatrix}$

     I chose this because the default for several built-in procedures (glm, for instance) is to initialize $\boldsymbol{\theta}_0$ this way. Also, I tried other initializations, and they gave awful results (singular Hessian, things like that). This one worked, so I went with it.
   - Value of the objective function at the optimum: $\min\Big(J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2\Big) \approx -223733.8$ for optimized $\boldsymbol{\theta}$ after two iterations of Newton's Method.

(b) (10 pts) Generate a figure displaying 20 images in a 4×5 array. These images should be the 20 misclassified images for which the logistic regression classifier was most confident about its prediction. You will have to define a notion of confidence in a reasonable way and explain how you define it. In the title of each subplot, indicate the true label of the image. What you should expect to see is a bunch of 4s that look kind of like 9s and vice versa.

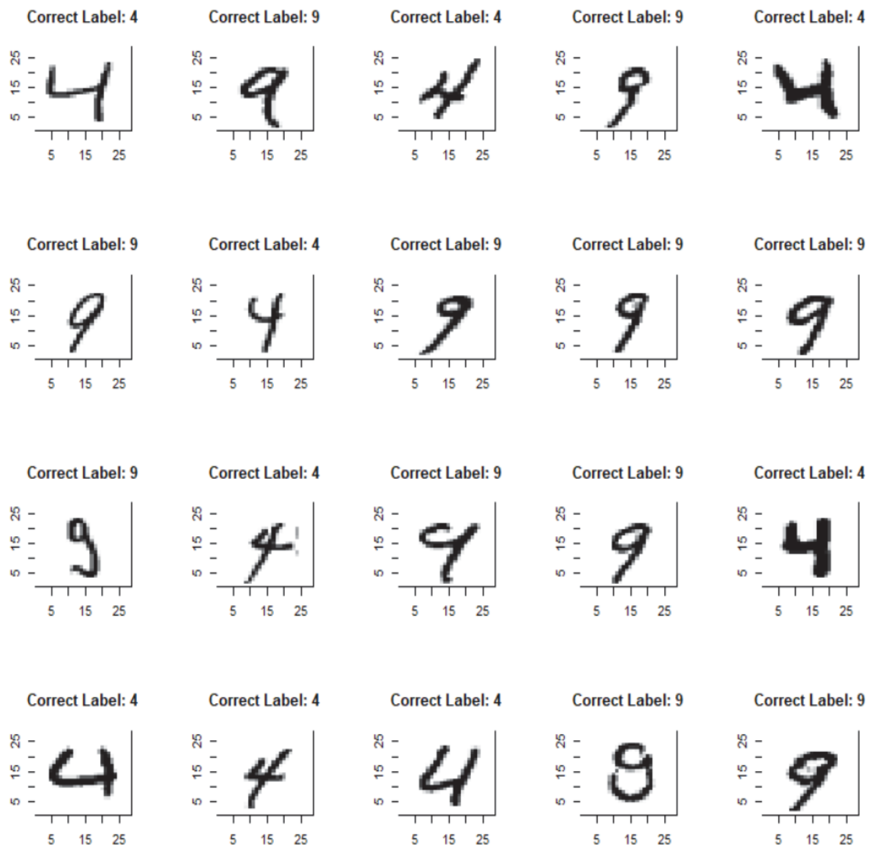

Figure 1: Panel of 20 'worst' misclassified images

Criterion I used for selecting the "worst" misclassifed instances (logRegr was confident, but wrong): The 20 misclassified images will be those whose output probability is furthest from 0.5 and ,obviously, for which the classification is incorrect. Based on examination of the data, fours are given y values of -1, and nines are given y values of 1. To see which are the "worst" misclassified, we're looking for the output probabilities that are most extreme (nearest 0 or 1) which also have a misclassification. Simply take the absValue(predProb - 0.5) to get confidence, then rank-order these. Only keep those with misclassification (correctIndVar == 0). Then arrange in order of descending 'confidenceLogRegr' (values closest to 0.5 will be at the top). Keep the top 20 closest values to 0.5. Figure 1 is these 'worst' 20 images, with highly-confident predictions that are wrong.

(c) (6 pts) Include your well-organized, clearly commented code

```r
## AUTHOR: Jared Hansen
## LAST MODIFIED: Sunday, 11/04/2018




#============================================================
#==== CODE FOR STATISTICAL LEARNING II, PROBLEM 6 ===========
#============================================================


# Necessary libraries for the script
library(dplyr)     # For data manipulation
library(R.matlab)  # For needed Matlab functionalities
library(geometry)  # For needed Matlab functionalities
library(pracma)    # For needed Matlab functionalities


# Change the working directory so that the image file can be read in.
# The line below needs to be changed if you want to run the code.
setwd("C:/Users/jrdha/OneDrive/Desktop/USU_Fa2018/Moon__SLDM2/hw3")


# Read in the MNIST data as a dataframe with response "Y" and predictors "X_1", "X_2",...
df <- R.matlab::readMat("mnist_49_3000.mat") %>% lapply(t) %>% lapply(as_tibble)
colnames(df[[1]]) <- sprintf("X_%s",seq(1:ncol(df[[1]])))
colnames(df[[2]]) <- c("Y")
df <- bind_cols(df) %>% select(Y, everything())


# This function calculates and returns the value of the objective function J(theta), where
# J(theta) = -l(theta) + lambda||theta||^2, the regularized logistic regression obj. function
# Arguments passed in are the feature vectors s (xs), response (y),
# theta (vector of parameters, constant b and weights), and lambda (scalar constant).
calcObjFctn <- function(xs, y, theta, lambda){
  indivTerms <- double(nrow(xs))
  for(i in 1:nrow(xs)){
    xi <- as.numeric(xs[i,])
    yi <- as.numeric(y$Y[i])
    t1 <- yi*log(1/(1 + exp(dot(-theta, xi))))
    t2 <- (1-yi)*log((exp(dot(-theta, xi)))/(1 + exp(dot(-theta, xi))))
    indivTerms[i] <- t1 + t2
  }
  objFctnMin <- (-1 * sum(indivTerms)) + (lambda * dot(theta, theta))
  return(objFctnMin)
}


# Used the gradient and Hessian calculated in part 4, but a resource from Carnegie Mellon
# gave a more concise, clean way of doing it. This involves calculating a value 'mu' for each
# feature vector x_i. As such, used the Carnegie Mellon formulation.
# The arguments passed are the vector theta, and the x_i.
calcMuVec <- function(theta, x_i){
  # calculates mu as defined in
  # Args:
  #   theta: vector containing intercept (b) and weights (w)
  #   x_i: vector of predictor variables (one observation ("row"))
  #
  #   theta and x_i must be vectors of same length
  #
  # Returns:
  #   The optimized value of theta
  denom <- (1 + exp(dot(-theta, x_i)))
  return(1/denom)
}


# As is mentioned in the function above, I'm using the gradient and Hessian of the objective
# function as defined in a resource from Carnegie Mellon. Here is the link:
# http://www.cs.cmu.edu/~mgormley/courses/10701-f16/slides/lecture5.pdf
# The function below calculates both the gradient and the Hessian of the regularized logistic
# regression objective function. It relies on the function that calculates the mu's from
# above. The return is the gradient and Hessian in a list.
# Also, the calculation is tuned by adjusting the value of the constant lambda as seen within t
his fctn.
# Arguments: xMatrix contains the predictor variables (with a leading column of 1s so that dims
match
#             because the first element of the theta vector is the constant b),
#             yVector contains the values of response (-1 or 1)
#             theta is still a vector containing [b, weight1, weight2, ....]^T
# Original formulas for this can be seen in the commented out code near the bottom of the file.
calcCMU_grad_Hess <- function(xMatrix, yVector, theta){

  # TUNE LAMBDA HERE
  lambda <- 1
  muVec <- double(nrow(xMatrix))

  for(i in 1:nrow(xMatrix)){
    x_i <- as.numeric(xMatrix[i,])
    muVec[i] <- calcMuVec(theta, x_i)
  }
  # This formulation of gradient is the CMU definition using the mu's in the calculation.
  gradient <- t(xMatrix) %*% (muVec - as.numeric(yVector$Y)) + 2*lambda*theta
  ds <- muVec * (1 - muVec)
  D <- diag(ds)
  XT_matrix <- t(sapply(xMatrix, as.numeric))
  X_hess_matrix <- sapply(xMatrix, as.numeric)
  twoLambIdMatrix <- diag(rep(2 * lambda, 785))
  hessian <- XT_matrix %*% D %*% X_hess_matrix + twoLambIdMatrix

  results <- list("gradient" = gradient, "hessian" = hessian)
  return(results)
}


# This function gives the initial guess for theta vector: [b=1, w1=0, w2=0,....,w784=0]^T.
# I used this because it sounds like that's what built-in functions in R do, and trial and erro
r
# suggested that it was working.
# Takes as arguments maxIterations (self-explanatory), predictors (matrix of x_i), response (ve
c of y_i)
```

```r
# Calls the newtonsMethod function (recursive function).
initialTheta <- function(maxIterations, predictors, response){

  b_0 = 1 # initial guess of b
  w_0_Vec = rep(0, 784) # initial guess of w's
  theta_0 <- c(b_0, w_0_Vec)

  theta <- newtonsMethod(theta = theta_0, current_Iter = 0, maxIterations = maxIterations, pred
_X = predictors, y = response)
  return(theta)
}


# This recursive function either returns the value of theta (after maxIterations) or calls itse
lf again
# if maxIterations hasn't yet been reached. Calls the calcCMU_grad_Hess function to calculate t
he
# gradient and the hessian
# Takes are arguments: theta (vector of b and weights, gets called in the initialTheta fctn),
#                      maxIterations (self-explanatory), current_Iter (current iteration in the
fctn),
#                      pred_X (x predictors dataframe), y (y resp values dataframe)
newtonsMethod <- function(theta, current_Iter, maxIterations, pred_X, y){

  if(current_Iter >= maxIterations){
    return(theta)
  } else {
    print(paste("Iteration Number: ", current_Iter))

    gh <- calcCMU_grad_Hess(xMatrix = pred_X, yVector = y, theta = theta)
    gradient <- gh$gradient
    hessian <- gh$hessian

    # gradient <- gradientJ(xMatrix = pred_X, yVector = y, theta = theta)
    # hessian <- hessianJ(pred_X = pred_X, theta = theta)

    current_theta <- theta - (inv(hessian) %*% gradient)
    current_Iter <- current_Iter + 1
    return(newtonsMethod(theta = current_theta, current_Iter = current_Iter, maxIterations = ma
xIterations, pred_X = pred_X, y = y))
  }
}


# This function implements regularized logistic regression to predict the class (-1, 1) of each
obs.
# It returns a dataframe that contains info on classification (correct, incorrect) as correctIn
dVar,
# the probability calculated for each x_i. Also calculates the correct classification rate (PCC
).
# Returns all of these things together as a list.
# Takes as arguments: x_df_no1s (dataframe of predictors without leading column of 1s),
#                     y_df (dataframe of response values, -1 or 1),
#                     theta (self-explanatory)
logRegr_Pred <- function(x_df_no1s, y_df, theta){

  w <- theta[2:length(theta)]
  b <- theta[1]
  probabilities <- double(nrow(x_df_no1s))
  predictedClass <- double(nrow(x_df_no1s))
  correctIndVar <- double(nrow(x_df_no1s))

  for(i in 1:nrow(x_df_no1s)){
    xi <- as.numeric(x_df_no1s[i,])
    prob_i <- 1 / (1 + exp(-dot(w, xi) + b))
    probabilities[i] <- prob_i
    if(prob_i > 0.5){
      predictedClass[i] <- 1
    } else{
      predictedClass[i] <- -1
    }

    if(as.numeric(predictedClass[i]) == as.numeric(y_df$Y[i])){
      correctIndVar[i] = 1
    } else {
      correctIndVar[i] = 0
    }
  }
  result <- data.frame(probabilities, predictedClass, Y = y_df$Y, correctIndVar)
  PCC <- sum(correctIndVar)/nrow(x_df_no1s)
  list_Results <- list("results" = result, "PCC" = PCC)

  return(list_Results)
}


# As instructed in the prompt, split the data into a training set (first 2000 obs).
# Add a column of leading 1s for correct dimensions/calculations when dealing with b in theta.
trainData <- dplyr::slice(df, 1:2000)
trainPredictors <- select(trainData, -Y)
X_0 <- rep(1, 2000)
trainPredictors <- cbind(X_0, trainPredictors)
trainResponse <- select(trainData, Y)

# As instructed in the prompt, split the data into a test set too (last 1000 obs).
# Add a column of leading 1s for correct dimensions/calculations when dealing with b in theta.
testData <- dplyr::slice(df, 2001:3000)
testPredictors <- select(testData, -Y)
X_0 <- rep(1000)
testPredictors <- cbind(X_0, testPredictors)
testResponse <- select(testData, Y)


# Call the initialTheta fctn to get the optimized theta vector from the 2000-obs training data.
optmzThetaTrainData <- initialTheta(maxIterations = 2, predictors = trainPredictors, response =
trainResponse)

# Using the optimized theta vector from trained model, predict onto the test dataset.
pred <- logRegr_Pred(x_df_no1s = select(testPredictors, -X_0), y_df = testResponse, theta = opt
mzThetaTrainData)
results <- pred$results
PCC <- pred$PCC
PCC # view PCC
```

```r
# Calculate minimum value of Objective function
minObjFctnVal <- calcObjFctn(xs = trainPredictors, y = trainResponse, theta = optmzThetaTrainDa
ta, lambda = 1)
minObjFctnVal # See what the minimum value of the obj fctn is for optimized theta.


# Criterion I used for selecting the "worst" misclassifed instances (logRegr was confident, but
wrong):
# The 20 misclassified images will be those whose output probability is furthest from 0.5
# (and obviously for which the classification is incorrect).
# Based on examination of the data, fours are given y values of -1, and nines are given y value
s of 1.
# To see which are the "worst" misclassified, we're looking for the output probabilities that a
re most
# extreme (nearest 0 or 1) which also have a misclassification.
# Simply take the absValue(predProb - 0.5) to get confidence, then rank-order these.
# Only keep those with misclassification (correctIndVar == 0).
# Then arrange in order of descending 'confLogRegridence' (values closest to 0.5 will be at the
top).
# Keep the top 20 closest values to 0.5.
results$confLogRegr <- abs(results$probabilities - 0.5000000)
results$index <- as.numeric(row.names(results))
# As of the line below, can still see the index in the dataset
allWrongObs <- results[results$correctIndVar == 0,]
allWrongObs <- arrange(allWrongObs, desc(confLogRegr))
# allWrongObs <- results[order(-results$confLogRegr),]
# Above line should do the same thing as the dplyr code, but doesn't, weird.
worst20obs <- allWrongObs[1:20,]
worst20obs.index <- as.numeric(row.names(worst20obs))


# Read in the MNIST data in order to display misclassified images
mnist <- readMat("mnist_49_3000.mat")
imgList = list()
for(i in seq(1, length(mnist$x), by = 784)) {
  img <- matrix(mnist$x[i:(i + 783)], nrow = 28, byrow = TRUE)
  img <- t(apply(img, 2, rev))
  imgList[[length(imgList)+1]] = img
}

# This will create the desired 4x5 panel with the 20 'worst' misclassified
# images (and their correct classification). Loops through the worst 20, giving
# the image and the correct label above it.
par(mfrow = c(4,5))
counter <- 1
for(i in worst20obs.index){
  print(i)
  correctLabel = "intial"
  if(worst20obs$Y[counter] == -1){
    correctLabel = "Correct Label: 4"
  } else {
    correctLabel = "Correct Label: 9"
  }
  image(1:28, 1:28, imgList[[i]], col=gray((255:0)/255), main = correctLabel,
        xlab = "", ylab = "")
  counter <- counter + 1
}
```

```
# Original Gradient & Hessian Functions (not the CMU versions: math should give same results th
o)
{
  # gradJ <- function(xmat, yvec, theta){
  #   lambda <- 10
  #   vecs <- data.frame(matrix(ncol = 3000, nrow = 785))
  #   for(i in 1:nrow(xmat)){
  #     xi <- as.numeric(xmat[i,])
  #     yi <- as.numeric(yvec$Y[i])
  #     val <- (exp(dot(-theta,xi)) * xi)/(1 + exp(dot(-theta,xi))) + ((yi*xi) - (xi))
  #     vecs[,i] <- val
  #   }
  #   gradient <- (-1 * rowSums(vecs)) + 2*lambda*theta
  #   return(gradient)
  # }

  # hessJ <- function(xs,theta){
  #   lambda <- 10
  #   twoLambIdMatrix <- diag(rep(2 * lambda, 785))
  #
  #   ds <- double(nrow(xs))
  #
  #   for(i in 1:nrow(xs)){
  #     xi <- as.numeric(xs[i,])
  #     entry <- exp(dot(-theta,xi))/((1 + exp(dot(-theta,xi)))^2)
  #     ds[i] <- entry
  #   }
  #   terms <- double(nrow(xs))
  #   for(i in 1:nrow(xs)){
  #     xi <- as.numeric(xs[i,])
  #   }
  #   D <- diag(ds)
  #   XT_matrix <- t(sapply(xs, as.numeric))
  #   X_hess_matrix <- sapply(xs, as.numeric)
  #   hess <- XT_matrix %*% D %*% X_hess_matrix
  #   hess <- hess + twoLambIdMatrix
  #   return(hess)
  # }
}
```

7. How long did this assignment take you? (5 pts)

Date	Times				Day Total
Tue, 10/30	11:00-11:30am = **0.5** HRs	4:00-5:00pm = **1** HRs	6:15-9:15pm **= 3** HRs		**4.5** HRs
Wed, 10/31	9:00 – 11:15am = **2.25** HRs	1:00 – 2:15pm = **1.25** HRs	5:15 – 10:00pm = **4.75** HRs		**8.25** HRs
Thu, 11/01	10:00 – 11:45am = **1.75** HRs	4:15 – 9:00pm = **4.75** HRs			**7.5** HRs
Fri, 11/02	8:30 – 10:45am = **2.25** HRs	11:45 – 12:15pm = **0.5** HRs	3:30 – 4:00pm = **0.5** HRs	5:00 – 11:00pm = **6** HRs	**9.25** HRs
Sat, 11/03	11:30 – 6:30pm = **7** HRs	7:15 – 11:00pm = **3.75** HRs			**10.75** HRs
Sun, 11/04	9:45 – 12:30pm = **2.75** HRs	1:00 – 2:15pm = **1.25** HRs			**4.00** HRs
					TOTAL: 44.25 HOURS