Juliannelin

Math189R SP19
Homework 4
Monday, Feb 25, 2019

1 (Conditioning a Gaussian) Note that from Murphy page 113. "Equation 4.69 is of such importance in this book that we have put a box around it, so you can easily find it." That equation is important. Read through the proof of the result. Suppose we have a distribution over random variables $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ that is jointly Gaussian with parameters

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix},$$

where

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \mu_2 = 5, \ \Sigma_{11} = \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix}, \ \Sigma_{21}^{\mathsf{T}} = \Sigma_{12} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}, \ \Sigma_{22} = [14].$$

Compute
(a) The marginal distribution $p(\mathbf{x}_1)$.
(b) The marginal distribution $p(\mathbf{x}_2)$.
(c) The conditional distribution $p(\mathbf{x}_1|\mathbf{x}_2)$
(d) The conditional distribution $p(\mathbf{x}_2|\mathbf{x}_1)$

(a) From Murphy, we know $p(\vec{x}_1) = \mathcal{N}(\vec{x}_1 | \vec{\mu}_1, \Sigma_{11})$ so:

$$p(\vec{x}_1) = \mathcal{N}(\vec{x}_1 | \vec{\mu}_1, \Sigma_{11}) = \boxed{\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix}\right)}$$

→ checked sol to make sure this was it...

(b) Also we know $p(\vec{x}_2) = \mathcal{N}(\vec{x}_2 | \vec{\mu}_2, \Sigma_{22})$ so:

$$p(\vec{x}_2) = \mathcal{N}(\vec{x}_2 | \vec{\mu}_2, \Sigma_{22}) = \boxed{\mathcal{N}(5, 14)}$$

(c) From Murphy eq 4.69, $\boxed{p(\vec{x}_1 | \vec{x}_2) = \mathcal{N}(\vec{x}_1 | \vec{\mu}_{1|2}, \Sigma_{1|2})}$ ...

Note: $\vec{\mu}_{1|2} = \vec{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\vec{x}_2 - \vec{\mu}_2)$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 5 \\ 11 \end{bmatrix}\left(\frac{1}{14}\right)(\vec{x}_2 - 5)$$

$$\boxed{\vec{\mu}_{1|2} = \frac{1}{14}\begin{bmatrix} 5 \\ 11 \end{bmatrix}(\vec{x}_2 - 5)}$$

(★ From Murphy 4.69)

$$\star \ \Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \quad \text{since } \begin{bmatrix} 5 \\ 11 \end{bmatrix} = \Sigma_{21}^{\mathsf{T}}$$

$$= \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix} - \begin{bmatrix} 5 \\ 11 \end{bmatrix}\frac{1}{14}\begin{bmatrix} 5 & 11 \end{bmatrix}$$

$$= \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix} - \frac{1}{14}\begin{bmatrix} 25 & 55 \\ 55 & 121 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix} - \begin{bmatrix} 25/14 & 55/14 \\ 55/14 & 121/14 \end{bmatrix}$$

$$\boxed{\Sigma_{1|2} = \begin{bmatrix} 59/14 & 57/14 \\ 57/14 & 61/14 \end{bmatrix}}$$

(d) Again, from Murphy (extrapolated), $\left| p(\vec{x}_2 | \vec{x}_1) = \mathcal{N}(\vec{\mu}_{2|1}, \Sigma_{2|1}) \right|$

* $\vec{\mu}_{2|1} = \vec{\mu}_2 + \Sigma_{21} \Sigma_{11}^{-1} (\vec{x}_1 - \vec{\mu}_1) = 5 + [5\ 11] \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix}^{-1} (\vec{x}_1 - \begin{bmatrix} 0 \\ 0 \end{bmatrix})$

↳ $\begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix}^{-1} = \frac{1}{14} \begin{bmatrix} 13 & -8 \\ -8 & 6 \end{bmatrix}$

$= 5 + [5\ 11] \frac{1}{14} \begin{bmatrix} 13 & -8 \\ -8 & 6 \end{bmatrix} (\vec{x}_1 - 0)$

$= 5 + \frac{1}{14} [-23\ 26] (\vec{x}_1)$

$= 5 + \left[-\frac{23}{14}\ \frac{26}{14} \right] \vec{x}_1$

$$\boxed{\vec{\mu}_{2|1} = 5 + \left[-\frac{23}{14}\ \frac{13}{7} \right] \vec{x}_1}$$

* $\Sigma_{2|1} = \Sigma_{22} - \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12} = 14 - [5\ 11] \frac{1}{14} \begin{bmatrix} 13 & -8 \\ -8 & 6 \end{bmatrix} \begin{bmatrix} 5 \\ 11 \end{bmatrix}$ ($\begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix}^{-1}$)

$= 14 - \frac{1}{14} [-23\ 26] \begin{bmatrix} 5 \\ 11 \end{bmatrix}$

$= 14 - \frac{1}{14} (171)$

$$\boxed{\Sigma_{2|1} = \frac{25}{14}}$$

2 (**MNIST**) In this problem, we will use the MNIST dataset, a classic in the deep learning literature as a toy dataset to test algorithms on, to set up a model for logistic regression and softmax regression. In the starter code, we have already parsed the data for you. However, you might need internet connection to access the data and therefore successfully run the starter code.

The problem is this: we have images of handwritten digits with 28×28 pixels in each image, as well as the label of which digit $0 \leq$ `label` ≤ 9 the written digit corresponds to. Given a new image of a handwritten digit, we want to be able to predict which digit it is. The format of the data is `label, pix-11, pix-12, pix-13, ...` where `pix-ij` is the pixel in the `i`th row and `j`th column.

(a) (**logistic**) Restrict the dataset to only the digits with a label of 0 or 1. Implement L2 regularized logistic regression as a model to compute $\mathbb{P}(y = 1|\mathbf{x})$ for a different value of the regularization parameter λ. Plot the learning curve (objective vs. iteration) when using Newton's Method *and* gradient descent. Plot the accuracy, precision ($p = \mathbb{P}(y = 1|\hat{y} = 1)$), recall ($r = \mathbb{P}(\hat{y} = 1|y = 1)$), and F1-score ($F1 = 2pr/(p + r)$) for different values of λ (try at least 10 different values including $\lambda = 0$) on the test set and report the value of λ which maximizes the accuracy on the test set. What is your accuracy on the test set for this model? Your accuracy should definitely be over 90%.

(b) (**softmax**) Now we will use the whole dataset and predict the label of each digit using L2 regularized softmax regression (multinomial logistic regression). Implement this using gradient descent, and plot the accuracy on the test set for different values of λ, the regularization parameter. Report the test accuracy for the optimal value of λ as well as it's learning curve. Your accuracy should be over 90%.

(a) Implement L2 reg log. regression to find $\mathbb{P}(y=\vec{1}|\vec{x})$:

Recall log. model : $P(y=1|\vec{x};\vec{\theta}) = \sigma(\vec{\theta}^T\vec{x})$ &

neg. loglikelihood: $n\ell(\theta) = -\sum_i y_i \log \sigma(\vec{\theta}^T\vec{x}) + (1-y_i)\log(\vec{1} - \sigma(\vec{\theta}^T\vec{x})) + \frac{\lambda}{2}\|\theta\|_2^2$ (L2 norm)

So, $\nabla_\theta \ell = \sum_i y_i(\vec{1} - \sigma(\vec{\theta}^T\vec{x}))\vec{x} - (\vec{1}-y_i)\sigma(\vec{\theta}^T\vec{x})\vec{x} + \lambda\vec{\theta}$

$= \sum_i \left[y_i - y_i\sigma(\vec{\theta}^T\vec{x}) - \sigma(\vec{\theta}^T\vec{x}) + y_i\sigma(\vec{\theta}^T\vec{x})\right]\vec{x} + \lambda\vec{\theta}$

$= \sum_i [y_i - \sigma(\vec{\theta}^T\vec{x}_i)]\vec{x}_i + \lambda\vec{\theta}$

$= \boxed{X^T(\sigma(X\vec{\theta}) - \vec{y}) + \lambda\vec{\theta}}$

($\sum_i \vec{x}_i = X$ & $A^TB = B^TA$)

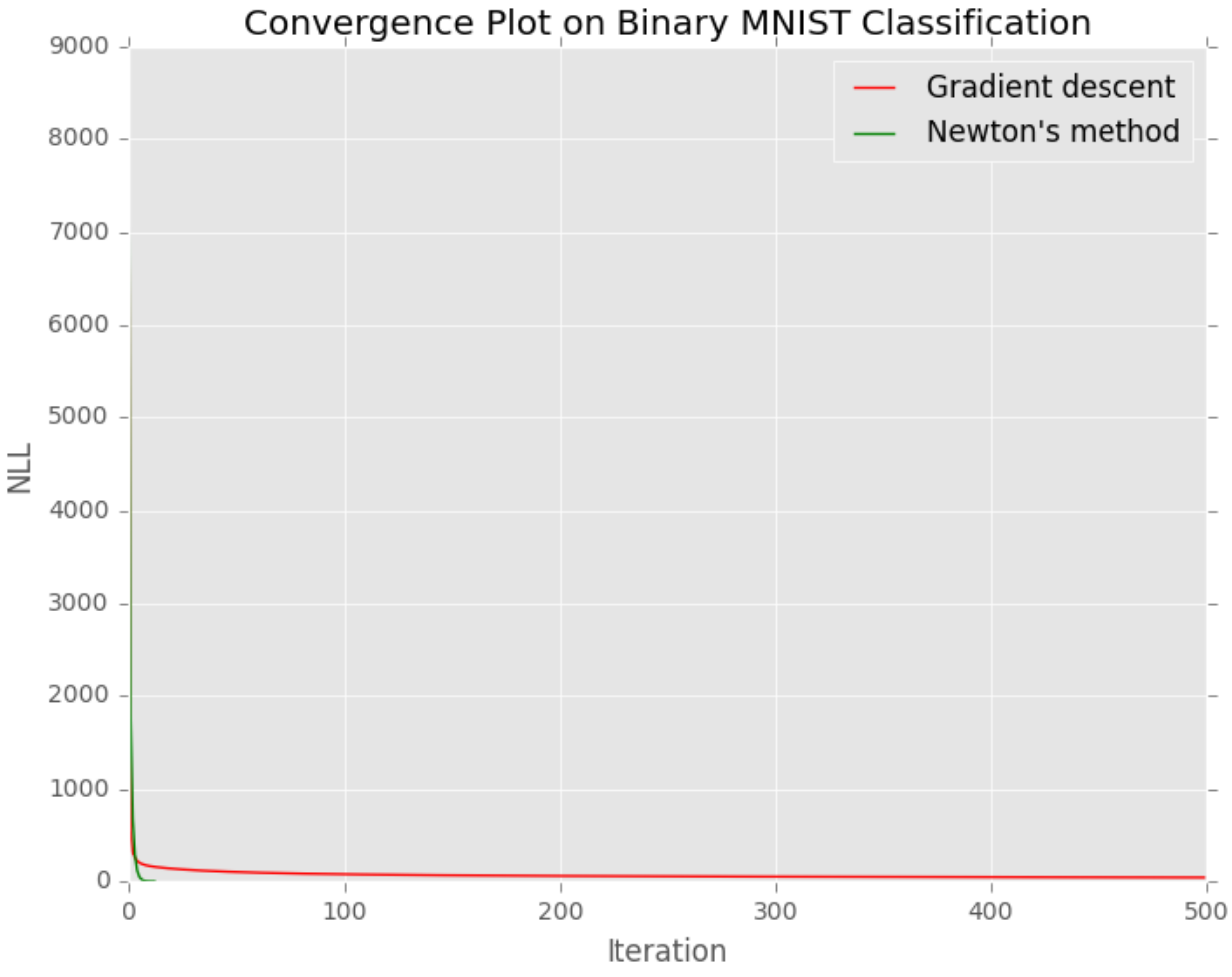
And Hessian: $\nabla^2\ell = \frac{d}{d\theta}\nabla\ell^T$

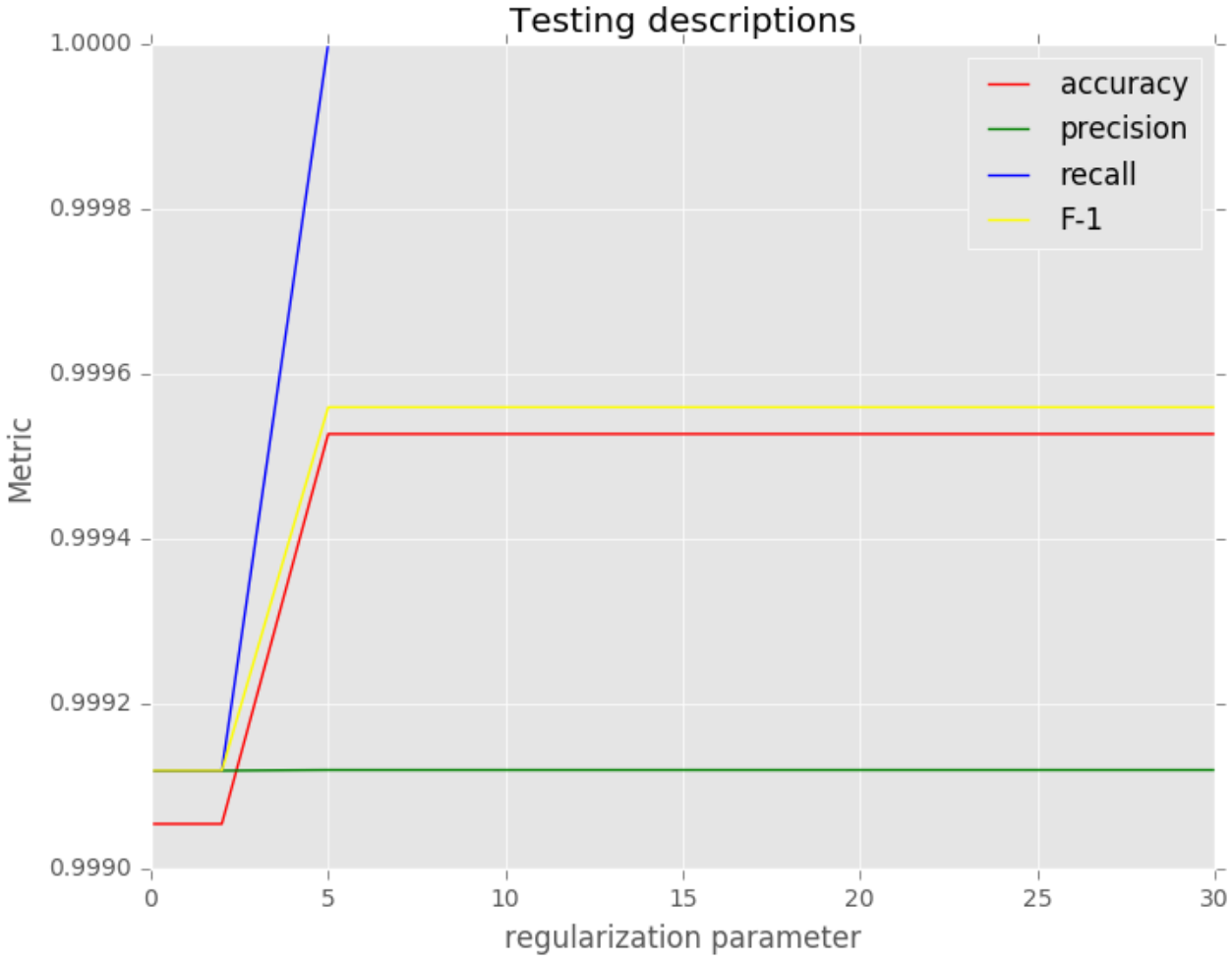
$= \sum_i \nabla_\theta \sigma(\vec{\theta}^T\vec{x})\vec{x}_i^T + \lambda I$

checked sol'n here ↓

$= \boxed{X^T \text{diag}[\sigma(X\vec{\theta})(\vec{1} - \sigma(X\vec{\theta}))]X + \lambda I}$ ■

(See plots attached)


Convergence Plot on Binary MNIST Classification
Gradient descent
Newton's method
NLL
Iteration
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500


Testing descriptions
accuracy
precision
recall
F-1
Metric
1.0000
0.9998
0.9996
0.9994
0.9992
0.9990
0
5
10
15
20
25
30
regularization parameter

(b) Softmax $p(y=c \mid \vec{x}, W) = \frac{1}{Z}\exp(\vec{w}_c^T \vec{x}) = \frac{\exp(\vec{w}_c^T \vec{x})}{\sum_i \exp(\vec{w}_i^T \vec{x})}$ from last wk.

neg log likelihood = $nl(W) = -\log \prod_i \prod_c \mu_{ic}^{y_{ic}} - \lambda tr(W^T W)$

(b/c $\log(ab) = \log a + \log b$) ↳ $= \sum_i \sum_c y_{ic} \log \mu_{ic} + \lambda tr(W^T W)$

↳ a little similar to (a)

Then $\nabla_W nl = X^T(\vec{\mu} - \vec{y}) + \lambda W$

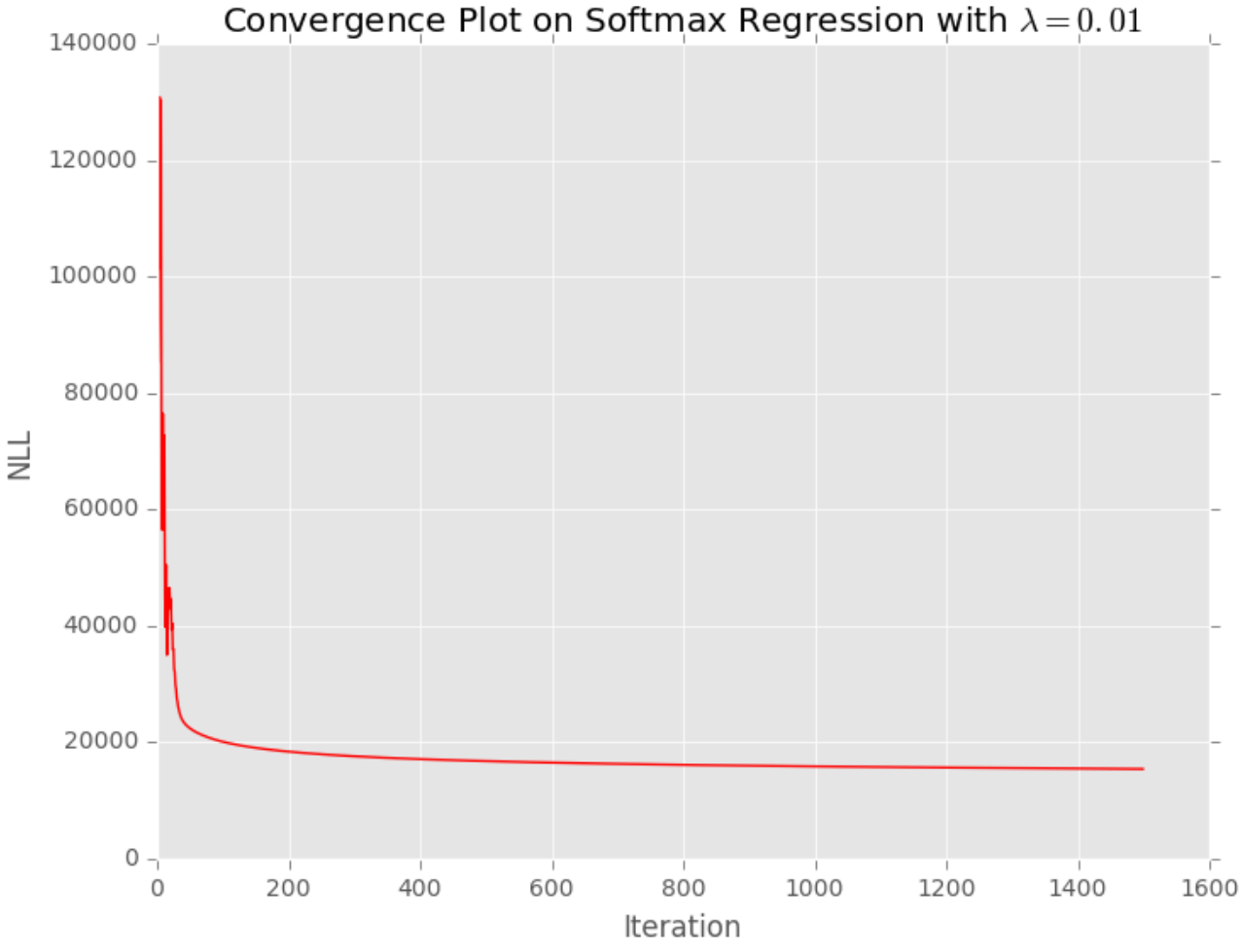
for $y \in \{0,1\}^{n \times c}$ as one-hot-encoding of outpt y, so

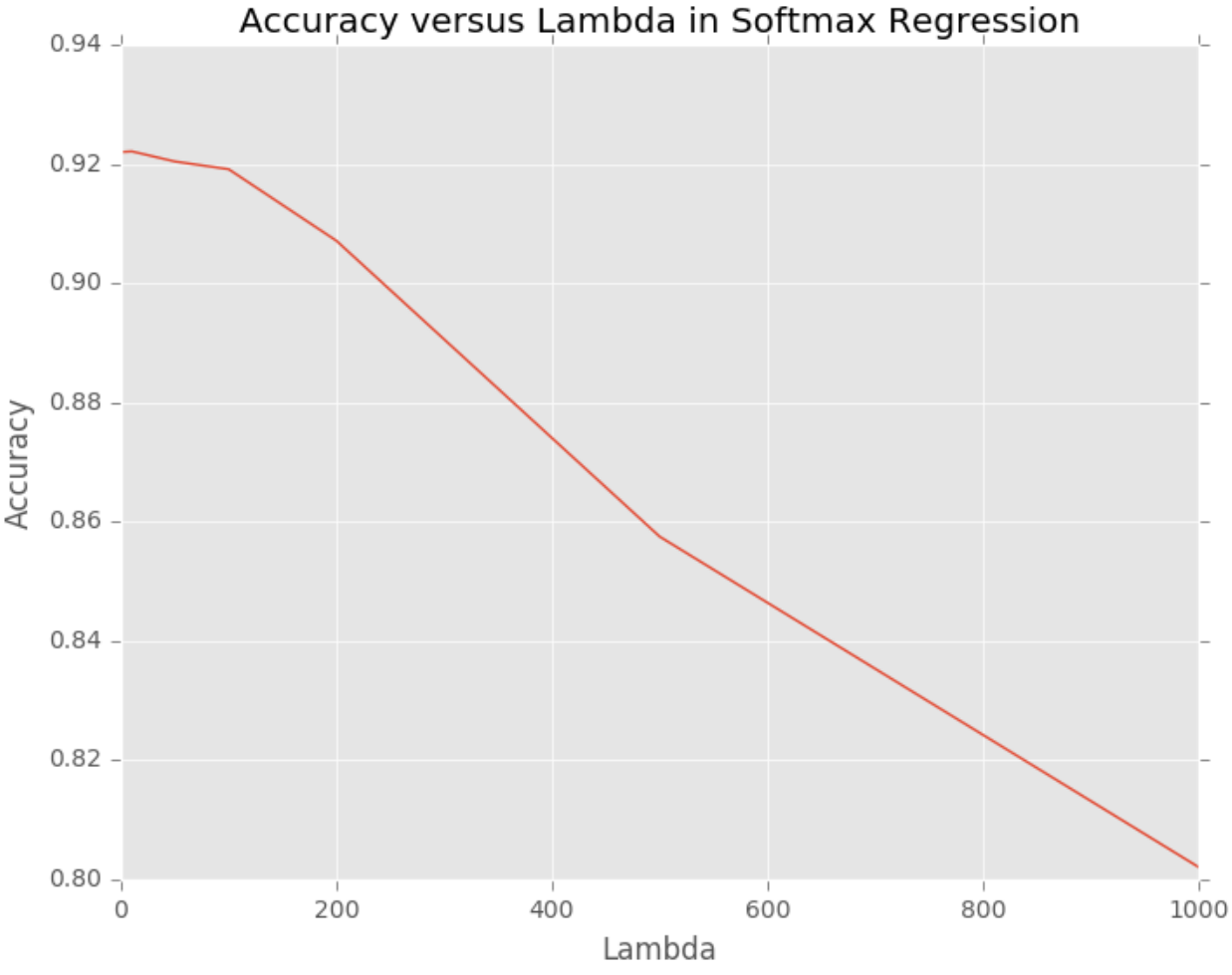
checked sol'n + Murphy ↳ $y_{ij} = \begin{cases} 1 & \text{if datapt } i \text{ is } j \\ 0 & \text{otherwise} \end{cases}$ → predicting label

and $\vec{y}\vec{1}_c = \vec{1}_n$

Also for $\vec{\mu} \in [0,\vec{1}]^{n \times c}$, $\vec{\mu}_i = S(\vec{x}_i) = \frac{\exp(W^T \vec{x})}{\vec{1}^T \exp(W^T \vec{x})}$ ↗ checked sol'n & asked prof. Gu in office hrs

→ Accuracy 0.9221 for $\lambda = 0.01$


Convergence Plot on Softmax Regression with $\lambda = 0.01$
NLL
Iteration
0
20000
40000
60000
80000
100000
120000
140000
0
200
400
600
800
1000
1200
1400
1600


Accuracy versus Lambda in Softmax Regression
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
Accuracy
0
200
400
600
800
1000
Lambda