Solution
Math189R SU17
Homework 4
Wednesday, May 31, 2017

Feel free to work with other students, but make sure you write up the homework and code on your own (no copying homework *or* code; no pair programming). Feel free to ask students or instructors for help debugging code or whatever else, though.

The starter files can be found under the Resource tab on course website. The graphs for problem 2 generated by the sample solution could be found in the corresponding zipfile. These graphs only serve as references to your implementation. You should generate your own graphs for submission. Please print out all the graphs generated by your own code and submit them together with the written part, and make sure you upload the code to your Github repository.

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

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

where

$$\mu_1 = \begin{bmatrix}0\\0\end{bmatrix},\ \mu_2 = 5,\ \mathbf{\Sigma}_{11} = \begin{bmatrix}6 & 8\\ 8 & 13\end{bmatrix},\ \mathbf{\Sigma}_{21}^\top = \mathbf{\Sigma}_{12} = \begin{bmatrix}5\\11\end{bmatrix},\ \mathbf{\Sigma}_{22} = \begin{bmatrix}14\end{bmatrix}.$$

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) We know

$$p(\mathbf{x}_1) = \mathcal{N}(\mu_1, \mathbf{\Sigma}_{11}) = \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}6 & 8\\ 8 & 13\end{bmatrix}\right).$$

(b) We know

$$p(\mathbf{x}_2) = \mathcal{N}(\mu_2, \mathbf{\Sigma}_{22}) = \mathcal{N}(5, 14).$$

(c) From Equation 4.69 in Murphy we know

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}\left(\mu_{1|2}, \boldsymbol{\Sigma}_{1|2}\right)$$

where

$$\mu_{1|2} = \mu_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \mu_2) = \frac{1}{14}\begin{bmatrix}5\\11\end{bmatrix}(\mathbf{x}_2 - 5)$$

and

$$\boldsymbol{\Sigma}_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} = \begin{bmatrix}6 & 8\\8 & 13\end{bmatrix} - \frac{1}{14}\begin{bmatrix}5\\11\end{bmatrix}\begin{bmatrix}5 & 11\end{bmatrix} = \begin{bmatrix}\frac{59}{14} & \frac{57}{14}\\ \frac{57}{14} & \frac{61}{14}\end{bmatrix}.$$

(d) As above we have

$$p(\mathbf{x}_2|\mathbf{x}_1) = \mathcal{N}\left(\mu_{2|1}, \boldsymbol{\Sigma}_{2|1}\right)$$

where

$$\mu_{2|1} = \mu_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_1 - \mu_1) = 5 + \begin{bmatrix}5 & 11\end{bmatrix}\begin{bmatrix}6 & 8\\8 & 13\end{bmatrix}^{-1}(\mathbf{x}_1 - \mu_1) = 5 + \begin{bmatrix}-\frac{23}{14} & \frac{13}{7}\end{bmatrix}\mathbf{x}_1$$

and

$$\boldsymbol{\Sigma}_{2|1} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12} = 14 - \begin{bmatrix}5 & 11\end{bmatrix}\begin{bmatrix}6 & 8\\8 & 13\end{bmatrix}^{-1}\begin{bmatrix}5\\11\end{bmatrix} = \frac{25}{14}.$$

■

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 `ith` row and `jth` 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) For the logistic model we have $P(y = 1|\mathbf{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \mathbf{x})$ and, with a Gaussian prior on the weights we have the negative log likelihood

$$n\ell\ell(\boldsymbol{\theta}) = -\sum_i y_i \log \sigma(\boldsymbol{\theta}^T x) + (1 - y_i) \log \left(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x})\right) + \frac{\lambda}{2}||\boldsymbol{\theta}||_2^2.$$

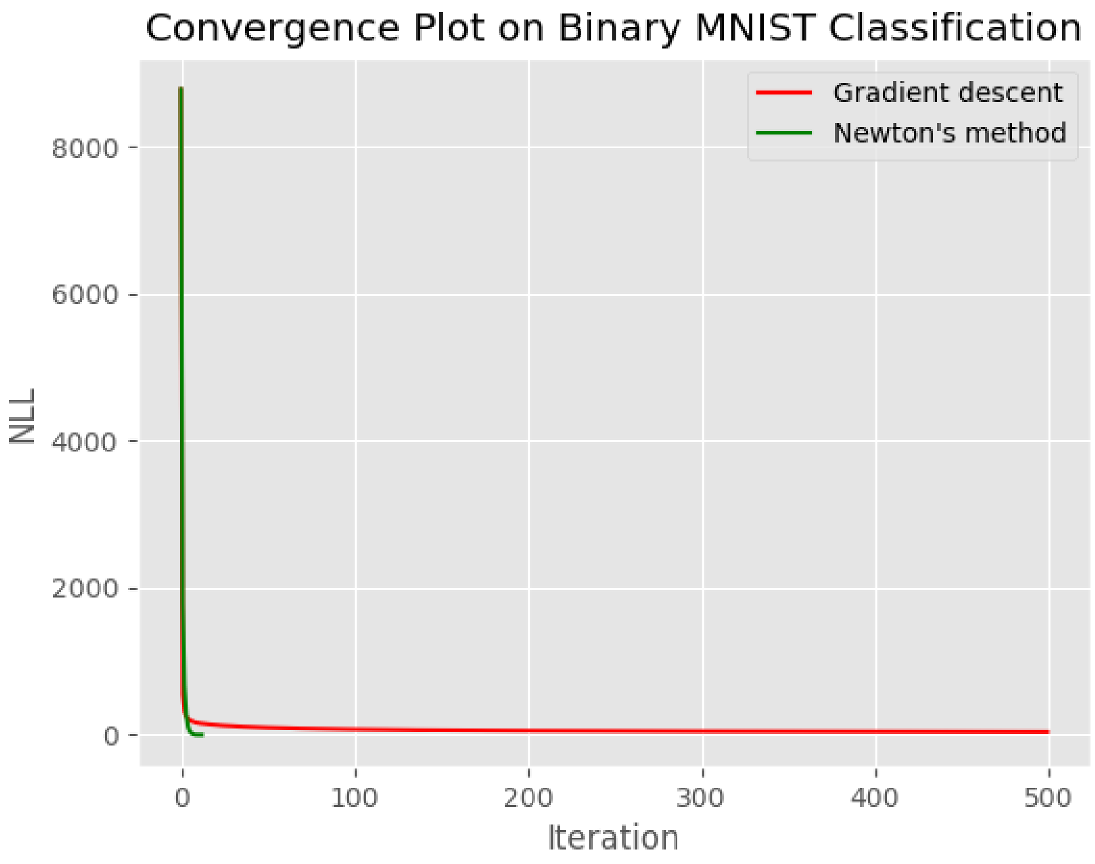
Taking gradients we find

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \ell &= \sum_i y_i \left(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x})\right) \mathbf{x} - (1 - y_i)\sigma(\boldsymbol{\theta}^\top \mathbf{x})\mathbf{x} + \lambda\boldsymbol{\theta} \\ &= \sum_i \left[y_i - \sigma(\boldsymbol{\theta}^\top \mathbf{x}_i)\right] \mathbf{x}_i + \lambda\boldsymbol{\theta} \\ &= X^\top (\sigma(X\boldsymbol{\theta}) - \mathbf{y}) + \lambda\boldsymbol{\theta}. \end{aligned}$$

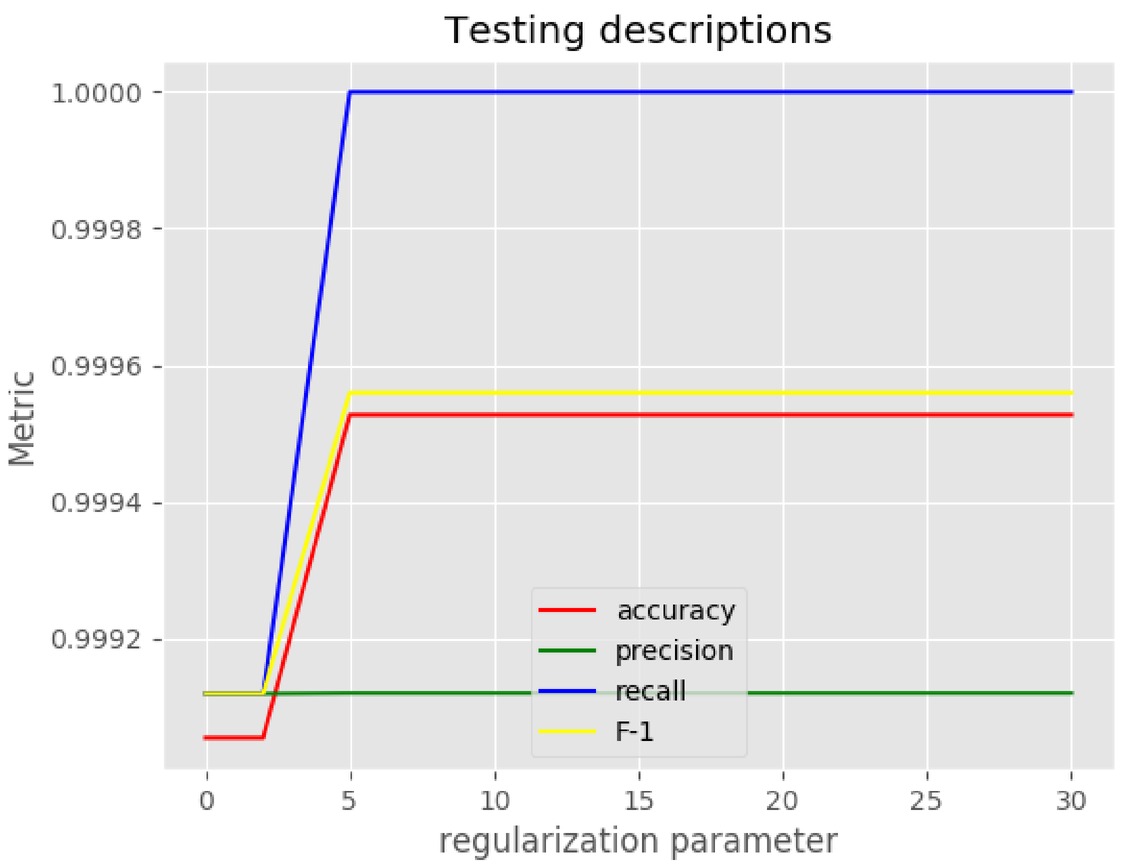
From this we can find a Hessian of

$$
\begin{aligned}
\nabla^2 \ell &= \frac{d}{d\boldsymbol{\theta}} \nabla \ell^\top \\
&= \sum_i \nabla_{\boldsymbol{\theta}} \sigma(\boldsymbol{\theta}^\top \mathbf{x}) \mathbf{x}_i^\top + \lambda I \\
&= X^T \text{diag}\left[\sigma(X\boldsymbol{\theta})\left(1 - \sigma(X\boldsymbol{\theta})\right)\right] X + \lambda I
\end{aligned}
$$

As it turns out, this problem is intrinsically easy (from a modelling point of view) as ones and zeros are easily told apart. We can see the convergence plot below:



Notice how Newton's Method is *much, much* faster than raw gradient descent by orders of magnitude. As was (or will be depending on when you read this) discussed in class/review, this speedup stems from the scale invariance of the algorithm. As was (or will be depending on when you read this) discussed in class/review, this speedup stems from the scale invariance of the algorithm. We can tell this problem is easy by looking at the plots of test metrics for different regularization parameters:



Indeed, all of them are almost perfect for any reasonable regularization parameter. This is what is known as a *easy problem*.

(b) For softmax regression we have $\mathbb{P}(y = c|\mathbf{x}, W) = \frac{1}{Z}\exp(\mathbf{w}_c^\top \mathbf{x}) = \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_i \exp(\mathbf{w}_i^\top \mathbf{x})}$. Assuming a Gaussian prior on each column of W we have the negative log likelihood

$$
\begin{aligned}
n\ell\ell(W) &= -\log \prod_i \prod_c \mu_{ic}^{y_{ic}} - \lambda \mathrm{tr}(W^\top W) \\
&= \sum_i \sum_c y_{ic} \log \mu_{ic} + \lambda \mathrm{tr}(W^\top W)
\end{aligned}
$$

We can find

$$\nabla_W n\ell\ell = X^\top(\boldsymbol{\mu} - \mathbf{y}) + \lambda W$$

where $\mathbf{y} \in \{0,1\}^{n\times c}$ is the "one-hot encoding" of the output $\mathbf{y}$ such that

$$\mathbf{y}_{ij} = \begin{cases} 1 & \text{if datum } i \text{ is digit } j \\ 0 & \text{otherwise} \end{cases}$$

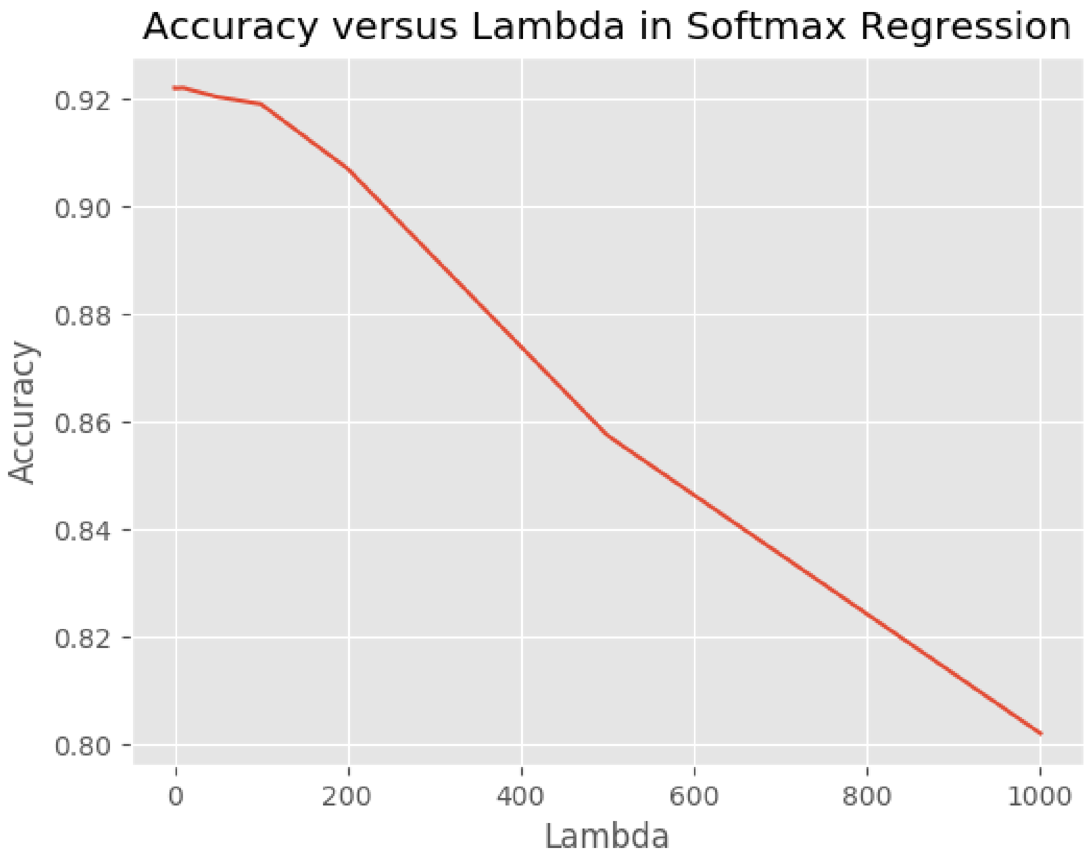
and

$$\mathbf{y}\mathbf{1}_c = \mathbf{1}_n. \tag{1}$$

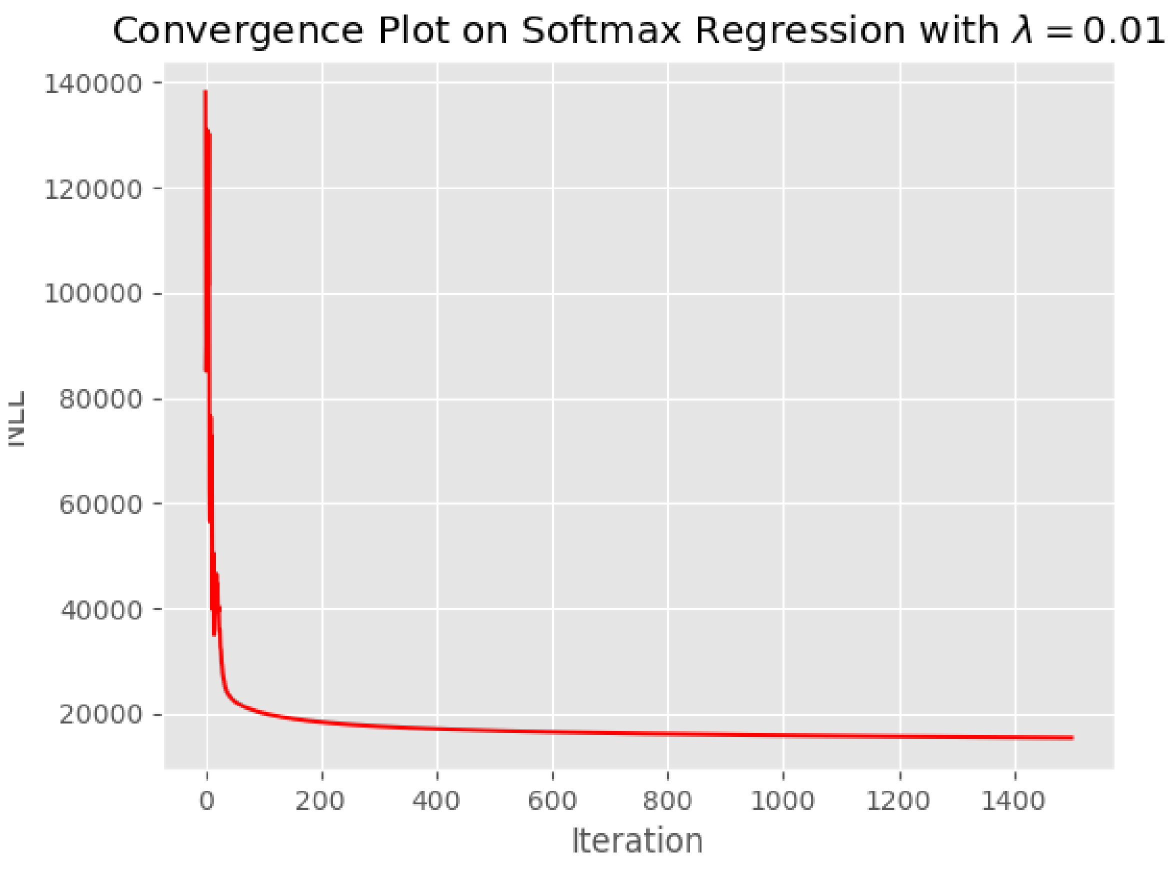
Similarly, we define $\boldsymbol{\mu} \in [0,1]^{n \times c}$ as

$$\boldsymbol{\mu}_i = \mathcal{S}(\mathbf{x}_i) = \frac{\exp(W^\top \mathbf{x})}{\mathbf{1}^\top \exp(W^\top \mathbf{x})} \tag{2}$$

and exp is applied elementwise. Using stochastic gradient descent, we have the test accuracies over different regularization parameters as follows:



where the maximum test accuracy was 0.9221 with $\lambda = 0.01$. For the optimal regularizer we also have the following convergence plot:


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

■