

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 $\mathbf{x} = (x_1, x_2)$ that is jointly Gaussian with parameters

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

where

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

Compute

- (a) The marginal distribution $p(x_1)$.
- (b) The marginal distribution $p(x_2)$.
- (c) The conditional distribution $p(x_1|x_2)$
- (d) The conditional distribution $p(x_2|x_1)$

First note that \vec{x}_1 is a two-dimensional random variable and \vec{x}_2 is a one-dimensional random variable.

a) We get $p(\vec{x}_1)$ by projecting onto the \vec{x}_1 plane:

$$p(\vec{x}_1) = N(\vec{x}_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) = \frac{1}{2\pi|\boldsymbol{\Sigma}_{11}|^{1/2}} \exp\left[-\frac{1}{2}(\vec{x}_1 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_{11}^{-1} (\vec{x}_1 - \boldsymbol{\mu}_1)\right]$$

$$= \frac{1}{2\pi\sqrt{14}} \exp\left(-\frac{1}{2}\vec{x}_1^T \overset{\curvearrowleft}{\boldsymbol{\Sigma}_{11}^{-1}} \vec{x}_1\right) \quad \boxed{\begin{bmatrix} \frac{13}{14} & -\frac{4}{7} \\ -\frac{4}{7} & \frac{3}{7} \end{bmatrix}}$$

$$= \boxed{\frac{1}{2\pi\sqrt{14}} \exp\left(-\frac{1}{2}\left[\frac{13}{14}x_1^2 + \frac{3}{7}x_2^2 - \frac{8}{7}x_1x_2\right]\right)}$$

b) We get $p(\vec{x}_2)$ by projecting onto the x_2 line:

$$p(\vec{x}_2) = N(\vec{x} | \mu_2, \Sigma_{22}) = \frac{1}{\sqrt{2\pi} |\Sigma_{22}|^{1/2}} \exp\left(-\frac{1}{2} (\vec{x} - \mu)^T \Sigma^{-1} (\vec{x} - \mu)\right)$$

$$= \boxed{\frac{1}{\sqrt{2\pi} \sqrt{14}} \exp\left(-\frac{1}{2} (\vec{x} - 5)^2 / 14\right)}$$

c) $p(\vec{x}_1 | \vec{x}_2)$ is given by $N(\vec{x}_1 | \mu_{1|2}, \Sigma_{1|2})$, where

$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} = \begin{bmatrix} 6 & 8 \\ 8 & 13 \end{bmatrix} - \begin{bmatrix} 25 & 55 \\ 55 & 121 \end{bmatrix} \frac{1}{14} = \begin{bmatrix} 59 & 57 \\ 57 & 61 \end{bmatrix} \frac{1}{14}$$

$$\mu_{1|2} = \mu_1 + \Sigma_{12} \Sigma_{22}^{-1} (\vec{x}_2 - \mu_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{(\vec{x}_2 - 5)}{14} \begin{bmatrix} 5 \\ 11 \end{bmatrix} = \frac{(\vec{x}_2 - 5)}{14} \begin{bmatrix} 5 \\ 11 \end{bmatrix}$$

$$\text{So } p(\vec{x}_1 | \vec{x}_2) = \boxed{N\left(\frac{(\vec{x}_2 - 5)}{14} \begin{bmatrix} 5 \\ 11 \end{bmatrix}, \begin{bmatrix} 59 & 57 \\ 57 & 61 \end{bmatrix} \frac{1}{14}\right)}$$

d) Similarly, $p(\vec{x}_2 | \vec{x}_1)$ is given by $N(\vec{x} | \mu_{2|1}, \Sigma_{2|1})$, where

$$\Sigma_{2|1} = \Sigma_{22} - \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12} = 14 - [5 \ 11] \begin{bmatrix} \frac{13}{14} & \frac{-4}{7} \\ -\frac{4}{7} & \frac{3}{7} \end{bmatrix} \begin{bmatrix} 5 \\ 11 \end{bmatrix} = \frac{25}{14}$$

$$\mu_{2|1} = \mu_2 + \Sigma_{21} \Sigma_{11}^{-1} (\vec{x}_1 - \vec{\mu}_1) = 5 + [5 \ 11] \begin{bmatrix} \frac{13}{14} & \frac{-4}{7} \\ -\frac{4}{7} & \frac{3}{7} \end{bmatrix} \vec{x}_1$$

$$= 5 + \left[-\frac{23}{14} \ \frac{13}{7} \right] \vec{x}_1$$

$$\text{So } p(\vec{x}_2 | \vec{x}_1) = \boxed{N\left(5 + \left[-\frac{23}{14} \ \frac{13}{7} \right] \vec{x}_1, \frac{25}{14}\right)}$$

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 $0 \leq \text{label} \leq 9$ 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.

- (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 $\mathbb{P}(y=1|x)$ 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%.
- (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 its learning curve. Your accuracy should be over 90%.

$$a) NLL = -\sum (y_i \log \mu_i + (1-y_i) \log(1-\mu_i)) + \lambda \|W\|^2$$

$$\sigma(x^T w)$$

$$\nabla_{NLL} = \vec{x}^T (\vec{\mu} - \vec{y}) + \lambda w$$

From HW2 pr 1
p. 252

$$H = \vec{x}^T S \vec{x} + \lambda I, \text{ where } S = \text{diag}(\mu_i(1-\mu_i))$$

b) The probability for multinomial logistic regression is

$$P(y=c|x, W) = \frac{1}{Z(x, w)} \exp(w_c^T x) = \frac{\exp(w_c^T x)}{\sum_c \exp(w_c^T x)}$$

where w_c is a weight vector and $W = (w_0, w_1, \dots)$

$$\rightarrow NLL = -\log \prod_i \prod_c (w_c)_c^{(y_i)_c} = -\sum_i \sum_c (y_i)_c \log(w_c)_c + \sum_c \lambda \|w_c\|^2 \text{ for regularization}$$

$$\nabla_{w_c} NLL = \left(\frac{\partial NLL(w)}{\partial w_{c,0}}, \frac{\partial NLL(w)}{\partial w_{c,1}}, \dots \right) = - \sum_{i=1}^N [1(y_i=c) - p(y_i=c|x_i; w)] x_i$$

$$= X^T ((\mu)_c - \mu_c) \rightarrow \nabla_w NLL = X^T (\mu - y)$$

Each row is a $(y_i)_c$, + λW for regularization

To actually construct this on Python, we can get the μ matrix by taking

$$\exp(XW) \underbrace{\begin{bmatrix} d \\ m \end{bmatrix}}_X \underbrace{\begin{bmatrix} c \\ d \end{bmatrix}}_W = \underbrace{\begin{bmatrix} \mu_{11} & \mu_{12} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}}_m = \begin{bmatrix} \exp(x^{(1)}w_1) & \exp(x^{(1)}w_2) & \dots \\ \exp(x^{(2)}w_1) & \exp(x^{(2)}w_2) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Then we divide each row by the sum of the row. Each row will then be

$$(\mu_i)_c = \left[\frac{\exp(x^{(1)}w_c)}{\sum_c \exp(x^{(1)}w_c)} \quad \dots \quad \frac{\exp(x^{(c)}w_c)}{\sum_c \exp(x^{(c)}w_c)} \right]$$

We can then get

$$-\sum_i \sum_c (y_i)_c \log((\mu_i)_c) - \text{by taking } \begin{matrix} y \log(\mu) \\ \text{element wise} \end{matrix}$$

$$\begin{bmatrix} c \\ m \end{bmatrix} \quad \begin{bmatrix} c \\ m \end{bmatrix}$$

Only one entry in each column will remain non-zero, and we sum that all up

For predictions, we take $\underset{\text{each}}{(\mu_i)_c}$ and calculate the argmax