Homework #1

CSE 546: Machine Learning
Prof. Kevin Jamieson
Due: 10/17 11:59 PM

1 Gaussians

Recall that for any vector $u \in \mathbb{R}^n$ we have $||u||_2^2 = u^T u = \sum_{i=1}^n u_i^2$ and $||u||_1 = \sum_{i=1}^n |u_i|$. For a matrix $A \in \mathbb{R}^{n \times n}$ we denote $|A|$ as the determinant of A. A multivariate Gaussian with mean $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^{n \times n}$ has a probability density function $p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$ which we denote as $\mathcal{N}(\mu, \Sigma)$.

1. *[4 points]* Let

- $\mu_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$
- $\mu_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & -1.8 \\ -1.8 & 2 \end{bmatrix}$
- $\mu_3 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$ and $\Sigma_3 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

For each $i = 1, 2, 3$ on a separate plot:

a. Draw $n = 100$ points $X_{i,1}, \ldots, X_{i,n} \sim \mathcal{N}(\mu_i, \Sigma_i)$ and plot the points as a scatter plot with each point as a triangle marker (Hint: use `numpy.random.randn` to generate a mean-zero independent Gaussian vector, then use the properties of Gaussians to generate X).

b. Compute the sample mean and covariance matrices $\widehat{\mu}_i = \frac{1}{n} \sum_{j=1}^n X_{i,j}$ and $\widehat{\Sigma}_i = \frac{1}{n-1} \sum_{j=1}^n (X_{i,j} - \widehat{\mu}_i)^2$. Compute the eigenvectors of $\widehat{\Sigma}_i$. Plot the eigenvectors as line segments originating from $\widehat{\mu}_i$ and have magnitude equal to the square root of their corresponding eigenvalues.

c. If $(u_{i,1}, \lambda_{i,1})$ and $(u_{i,2}, \lambda_{i,2})$ are the eigenvector-eigenvalue pairs of the sample covariance matrix with $\lambda_{i,1} \geq \lambda_{i,2}$ and $||u_{i,1}||_2 = ||u_{i,2}||_2 = 1$, for $j = 1, \ldots, n$ let $\widetilde{X}_{i,j} = \begin{bmatrix} \frac{1}{\sqrt{\lambda_{i,1}}} u_{i,1}^T (X_{i,j} - \widehat{\mu}_i) \\ \frac{1}{\sqrt{\lambda_{i,2}}} u_{i,2}^T (X_{i,j} - \widehat{\mu}_i) \end{bmatrix}$. Plot these new points as a scatter plot with each point as a circle marker.

2. *[1 points]* Let $X \sim \mathcal{N}(\mu_X, \Sigma_X)$ and $X' \sim \mathcal{N}(\mu_{X'}, \Sigma_{X'})$ be random n-dimensional vectors. We usually assume that Σ^{-1} exists, but in many cases it will not. Describe the conditions for which Σ_X^{-1} corresponding to random vector X will not exist (Hint: think about what happens as $\lambda_{i,2}$ goes to 0 in the last problem). Assume $\Sigma_{X'}^{-1}$ exists but Σ_X^{-1} does not; give an expression to generate random vectors $X \sim \mathcal{N}(\mu_X, \Sigma_X)$ using just random vectors $X' \sim \mathcal{N}(\mu_{X'}, \Sigma_{X'})$ and the quantities $\mu_X, \Sigma_X, \mu_{X'}, \Sigma_{X'}$ (Hint: for which $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ are X and $AX' + b$ equal in distribution).

2 MLE, MAP, and Bias-Variance Tradeoff

3. *[4 points]* Suppose we observe a random vector $X \in \mathbb{R}^n$ with likelihood $p(x|\theta) = \mathcal{N}(\theta, \sigma^2 I) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2}(x-\theta)^T(x-\theta))$.

a. Compute $\widehat{\theta}_{MLE} = \arg\max_\theta p(x|\theta)$.

b. A domain expert says that the different θ_i unknowns are highly correlated with covariance matrix $\Sigma = \frac{1}{\nu} I + \nu \mathbf{1}\mathbf{1}^T$ for some known ν, and even gives you a prior $p(\theta) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}\theta^T \Sigma^{-1} \theta)$.

(a) Show that $\widehat{\theta}_{MAP} = \arg\max_\theta p(x|\theta)p(\theta)$ is the solution to $(I + \sigma^2 \Sigma^{-1})\widehat{\theta}_{MAP} = x$.

(b) Use the Sherman-Morrison identity[1] on Σ^{-1} to show that $\widehat{\theta}_{MAP}$ is the solution to $\left((1+\sigma^2\nu)I - \frac{\nu^3\sigma^2}{1+\nu^2 n}\mathbf{1}\mathbf{1}^T\right)\widehat{\theta}_{MAP} = x$.

(c) If $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ and $\mathbf{1}$ is a vector of n ones, show that $\widehat{\theta}_{MAP}$ is a linear combination of $\bar{x}\mathbf{1}$ and x. (Hint: plug $\lambda_1 x + \lambda_2 \bar{x}\mathbf{1}$ into part (b) for $\widehat{\theta}_{MAP}$ and solve for scalars λ_1 and λ_2).

(d) Using your solution to part (c), show that $\widehat{\theta}_{MAP} \to \frac{1}{1+\sigma^2\nu}x + \frac{\sigma^2\nu}{1+\sigma^2\nu}\bar{x}\mathbf{1}$ as $n \to \infty$.

(e) In words, describe how the prior $p(\theta)$ affects the map estimate $\widehat{\theta}_{MAP}$ for very small and very large ν, and how it relates to the MLE estimate $\widehat{\theta}_{MLE}$.

4. *[4 points]* (Stein's Paradox) Let $\theta \in \mathbb{R}^n$ and $\lambda \in (0,1)$. Let $\mathbf{1}$ denote the vector of n ones. For $i = 1, \ldots, n$ let $X_i \sim \mathcal{N}(\theta_i, \sigma^2)$, $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$, $\widehat{\theta} = (1-\lambda)X + \lambda\bar{X}\mathbf{1}$, $\bar{\theta} = \frac{1}{n}\sum_{i=1}^n \theta_i$. All expectations are taken with respect to the random draws of the X_i random variables.

a. Show that $\mathbb{E}[||\widehat{\theta} - \theta||_2^2] = ||\mathbb{E}[\widehat{\theta}] - \theta||_2^2 + \mathbb{E}[||\widehat{\theta} - \mathbb{E}[\widehat{\theta}]||_2^2]$, i.e., the bias^2 plus variance.

b. Compute the variance of this estimator: $\mathbb{E}[||\widehat{\theta} - \mathbb{E}[\widehat{\theta}]||_2^2]$

c. Compute the bias^2 of this estimator: $||\mathbb{E}[\widehat{\theta}] - \theta||_2^2$

d. What value of λ minimizes the overall error $\mathbb{E}[||\widehat{\theta} - \theta||_2^2]$?

e. Describe how the optimal value of λ found in part d changes if $\frac{1}{n-1}\sum_{i=1}^n(\theta_i - \bar{\theta})^2 \gg \sigma^2$, $\frac{1}{n-1}\sum_{i=1}^n(\theta_i - \bar{\theta})^2 \approx \sigma^2$, or $\frac{1}{n-1}\sum_{i=1}^n(\theta_i - \bar{\theta})^2 \ll \sigma^2$.

3 Regularization Constants

For the following, recall that the loss function to be optimized under ridge regression is

$$\widehat{w}_{Ridge} = \sum_{i=1}^n (y_i - (w_0 + x_i^T w))^2 + \lambda\|w\|_2^2$$

where λ is our regularization constant.

The loss function to be optimized under LASSO regression is

$$\widehat{w}_{Lasso} = \sum_{i=1}^n (y_i - (w_0 + x_i^T w))^2 + \lambda\|w\|_1$$

where λ is our regularization constant.

5. *[1 points]* Discuss briefly how choosing too small a λ affects the magnitude of the following quantities. Please describe the effects for both ridge and LASSO, or state why the effects will be the same.

a. The error on the training set.

b. The error on the testing set.

c. The elements of w.

d. The number of nonzero elements of w.

6. *[1 points]* Now discuss briefly how choosing too large a λ affects the magnitude of the same quantities in the previous question. Again describe the effects for both ridge and LASSO, or state why the effects will be the same.

[1] For square invertible matrix $A \in \mathbb{R}^{n\times n}$ and vectors $u, v \in \mathbb{R}^n$, we have that $(A + uv^T)$ is invertible iff $1 + v^T A^{-1} u \neq 0$ and $(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}$.

4 Programming: Ridge Regression on MNIST

7. *[10 points]* In this problem we will implement a least squares classifier for the MNIST data set. The task is to classify handwritten images of numbers between 0 to 9.

You are **NOT** allowed to use any of the prebuilt classifiers in `sklearn`. Feel free to use any method from `numpy` or `scipy`. Remember: if you are inverting a matrix in your code, you are probably doing something wrong (Hint: look at `scipy.linalg.solve`).

Get the data from `https://pypi.python.org/pypi/python-mnist`.
Load the data as follows:

```
from mnist import MNIST

def load_dataset():
    mndata = MNIST('./data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
    X_train = X_train/255.0
    X_test = X_test/255.0
```

You can visualize a single example by reshaping it to its original 28×28 image shape.

a. In this problem we will choose a linear classifier to minimize the least squares objective:

$$\widehat{W} = \text{argmin}_{W \in \mathbb{R}^{d \times k}} \sum_{i=0}^{n} \|W^T x_i - y_i\|_2^2 + \lambda \|W\|_F^2$$

We adopt the notation where we have n data points in our training objective and each data point $x_i \in \mathbb{R}^d$. k denotes the number of classes which is in this case equal to 10. Note that $\|W\|_F$ corresponds to the Frobenius norm of W, i.e. $\|\text{vec}(W)\|_2^2$.

Derive a closed form for $\widehat{W}$.

b. As as first step we need to choose the vectors $y_i \in \mathbb{R}^k$ by converting the original labels (which are in $\{0, \ldots, 9\}$) to vectors. We will use the one-hot encoding of the labels, i.e. the original label $j \in \{0, \ldots, 9\}$ is mapped to the standard basis vector e_j. To classify a point x_i we will use the rule $\arg\max_{j=0,\ldots,9} \widehat{W}^T x_i$.

c. Code up a function called `train` that returns $\widehat{W}$ that takes as input $X \in \mathbb{R}^{n \times d}$, $y \in \{0,1\}^{n \times k}$, and $\lambda > 0$. Code up a function called `predict` that takes as input $W \in \mathbb{R}^{d \times k}$, $X' \in \mathbb{R}^{m \times d}$ and returns an m-length vector with the ith entry equal to $\arg\max_{j=0,\ldots,9} W^T x_i'$ where x_i' is a column vector representing the ith example from X'.

Train $\widehat{W}$ on the MNIST training data with $\lambda = 10^{-4}$ and make label predictions on the test data. What is the training and testing classification accuracy (they should both be about 85%)?

d. We just fit a classifier that was linear in the pixel intensities to the MNIST data. For classification of digits the raw pixel values are very, very bad features: it's pretty hard to separate digits with linear functions in pixel space. The standard solution to the this is to come up with some transform $h : \mathbb{R}^d \to \mathbb{R}^p$ of the original pixel values such that the transformed points are (more easily) linearly separable. In this problem, you'll use the feature transform:

$$h(x) = \cos(Gx + b)$$

where $G \in \mathbb{R}^{p \times d}$, $b \in \mathbb{R}^p$, and the cosine function is applied elementwise. We'll choose G to be a *random* matrix, with each entry sampled i.i.d. with mean $\mu = 0$ and variance $\sigma^2 = 0.1$, and b to be a random vector sampled i.i.d. from the uniform distribution on $[0, 2\pi]$. The big question is: *how do we choose p?*

Cross-validation, of course!

Randomly partition your training set into proportions 80/20 to use as a new training set and validation set, respectively. Using the `train` function you wrote above, train a $\widehat{W}^p$ for different values of p and plot the classification training error and validation error on a single plot with p on the x-axis. Be careful, your computer may run out of memory and slow to a crawl if p is too large ($p \leq 6000$ should fit into 4 GB of memory). You can use the same value of λ as above but feel free to study the effect of using different values of λ and σ^2 for fun.

e. Instead of reporting just the classification test error, which is an unbiased estimate of the *true* error, we would like to report a *confidence interval* around the test error that contains the true error. For any $\delta \in (0, 1)$, it follows from Hoeffding's inequality that if X_i for all $i = 1, \ldots, m$ are i.i.d. random variables with $X_i \in [a, b]$ and $\mathbb{E}[X_i] = \mu$, then with probability at least $1 - \delta$

$$\mathbb{P}\left(\left|\left(\frac{1}{m}\sum_{i=1}^{m} X_i\right) - \mu\right| \geq \sqrt{\frac{\log(2/\delta)}{2m}}\right) \leq \delta$$

We will use the above equation to construct a confidence interval around our true classification error since the test error is just the average of indicator variables taking values in 0 or 1 corresponding to the ith test example being classified correctly or not, respectively, where an error happens with probability μ, the *true* classification error.

Let $\widehat{p}$ be the value of p that approximately minimizes the validation error on the plot you just made and use $\widehat{W}^{\widehat{p}}$ to compute the classification test accuracy, which we will denote as E_{test}. Use Hoeffding's inequality, above, to compute a confidence interval that contains $\mathbb{E}[E_{test}]$ (i.e., the *true* error) with probability at least 0.95 (i.e., $\delta = 0.05$). Report E_{test} and the confidence interval.