CS 189 Introduction to Machine Learning
Summer 2019 Marc Khoury & Brijen Thananjeyan

HW3

Due: Monday, July 15 at 11:59 pm
Deliverables:

1. Submit your predictions for the test sets to Kaggle as early as possible. Include your Kaggle scores in your write-up (see below). The Kaggle competition for this assignment can be found at:

   - MNIST: `https://www.kaggle.com/t/403584fc1c7b47b480122f21ceb722c4`
   - SPAM: `https://www.kaggle.com/t/5c74291c27bd4f478dc82337e415a900`

2. Submit a PDF of your homework, **with an appendix listing all your code**, to the Gradescope assignment entitled "HW3 Write-Up". You may typeset your homework in LaTeX or Word (submit PDF format, **not** .doc/.docx format) or submit neatly handwritten and scanned solutions. **Please start each question on a new page.** If there are graphs, include those graphs in the correct sections. **Do not** put them in an appendix. We need each solution to be self-contained on pages of its own.

   - In your write-up, please state with whom you worked on the homework.
   - In your write-up, please copy the following statement and sign your signature next to it. (Mac Preview and FoxIt PDF Reader, among others, have tools to let you sign a PDF file.) We want to make it *extra* clear so that no one inadverdently cheats.
     "I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."

3. Submit all the code needed to reproduce your results to the Gradescope assignment entitled "HW3 Code". Yes, you must submit your code twice: once in your PDF write-up (above) so the readers can easily read it, and once in compilable/interpretable form so the readers can easily run it. Do **NOT** include any data files we provided. Please include a short file named README listing your name, student ID, and instructions on how to reproduce your results. Please take care that your code doesn't take up inordinate amounts of time or memory. If your code cannot be executed, your solution cannot be verified.

4. The assignment covers concepts on Gaussian distributions and classifiers. Some of the material may not have been covered in lecture; you are responsible for finding resources to understand it.

1 Gaussian Classification

Let $P(x \mid C_i) \sim \mathcal{N}(\mu_i, \sigma^2)$ for a two-category, one-dimensional classification problem with classes C_1 and C_2, $P(C_1) = P(C_2) = 1/2$, and $\mu_2 > \mu_1$.

(a) Find the Bayes optimal decision boundary and the corresponding Bayes decision rule.

(b) The Bayes error is the probability of misclassification,

$$P_e = P((\text{misclassified as } C_1) \mid C_2)\, P(C_2) + P((\text{misclassified as } C_2) \mid C_1)\, P(C_1).$$

Show that the Bayes error associated with this decision rule is

$$P_e = \frac{1}{\sqrt{2\pi}} \int_a^\infty e^{-z^2/2} dz$$

where $a = \dfrac{\mu_2 - \mu_1}{2\sigma}$.

2 Isocontours of Normal Distributions

Let $f(\mu, \Sigma)$ be the probability density function of a normally distributed random variable in $\mathbb{R}^2$. Write code to plot the isocontours of the following functions, each on its own separate figure. You're free to use any plotting libraries available in your programming language; for instance, in Python you can use Matplotlib.

(a) $f(\mu, \Sigma)$, where $\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$.

(b) $f(\mu, \Sigma)$, where $\mu = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$.

(c) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ and $\Sigma_1 = \Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$.

(d) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$.

(e) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

3 Eigenvectors of the Gaussian Covariance Matrix

Consider two one-dimensional random variables $X_1 \sim \mathcal{N}(3, 9)$ and $X_2 \sim \frac{1}{2}X_1 + \mathcal{N}(4, 4)$, where $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2. Write a program that draws $n = 100$ random two-dimensional sample points from (X_1, X_2) such that the ith value sampled

from X_2 is calculated based on the ith value sampled from X_1. In your code, make sure to specify the Random Number Generator seed that was used so your simulation is reproducible. For each of the following parts, include the corresponding output of your program.

(a) Compute the mean (in $\mathbb{R}^2$) of the sample.

(b) Compute the 2×2 covariance matrix of the sample.

(c) Compute the eigenvectors and eigenvalues of this covariance matrix.

(d) On a two-dimensional grid with a horizonal axis for X_1 with range $[-15, 15]$ and a vertical axis for X_2 with range $[-15, 15]$, plot

  (i) all $n = 100$ data points, and

  (ii) arrows representing both covariance eigenvectors. The eigenvector arrows should originate at the mean and have magnitudes equal to their corresponding eigenvalues.

(e) Let $U = [v_1 \quad v_2]$ be a 2×2 matrix whose columns are the eigenvectors of the covariance matrix, where v_1 is the eigenvector with the larger eigenvalue. We use $U^\top$ as a rotation matrix to rotate each sample point from the (X_1, X_2) coordinate system to a coordinate system aligned with the eigenvectors. (As $U^\top = U^{-1}$, the matrix U reverses this rotation, moving back from the eigenvector coordinate system to the original coordinate system). *Center* your sample points by subtracting the mean μ from each point; then rotate each point by $U^\top$, giving $x_{\text{rotated}} = U^\top(x - \mu)$. Plot these rotated points on a new two dimensional-grid, again with both axes having range $[-15, 15]$.

4 Classification

Suppose we have a classification problem with classes labeled $1, \ldots, c$ and an additional "doubt" category labeled $c + 1$. Let $r : \mathbb{R}^d \to \{1, \ldots, c + 1\}$ be a decision rule. Define the loss function

$$L(r(x) = i, y = j) = \begin{cases} 0 & \text{if } i = j \quad i, j \in \{1, \ldots, c\}, \\ \lambda_r & \text{if } i = c + 1, \\ \lambda_s & \text{otherwise,} \end{cases}$$

where $\lambda_r \geq 0$ is the loss incurred for choosing doubt and $\lambda_s \geq 0$ is the loss incurred for making a misclassification. Hence the risk of classifying a new data point x as class $i \in \{1, 2, \ldots, c + 1\}$ is

$$R(r(x) = i|x) = \sum_{j=1}^{c} L(r(x) = i, y = j)\, P(Y = j|x).$$

(a) Show that the following policy obtains the minimum risk when $\lambda_r \leq \lambda_s$.

  (1) Choose class i if $P(Y = i|x) \geq P(Y = j|x)$ for all j and $P(Y = i|x) \geq 1 - \lambda_r/\lambda_s$;

  (2) Choose doubt otherwise.

(b) What happens if $\lambda_r = 0$? What happens if $\lambda_r > \lambda_s$? Explain why this is consistent with what one would expect intuitively.

5 Maximum Likelihood Estimation

Let $X_1, \ldots, X_n \in \mathrm{R}^d$ be n sample points drawn independently from a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$.

(a) Suppose the normal distribution has an unknown diagonal covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & \\ & & \sigma_3^2 & & \\ & & & \ddots & \\ & & & & \sigma_d^2 \end{bmatrix}$$

and an unknown mean μ. Derive the maximum likelihood estimates, denoted $\hat{\mu}$ and $\hat{\sigma}_i$, for μ and σ_i. Show all your work.

(b) Suppose the normal distribution has a known covariance matrix Σ and an unknown mean $A\mu$, where Σ and A are known $d \times d$ matrices, Σ is positive definite, and A is invertible. Derive the maximum likelihood estimate, denoted $\hat{\mu}$, for μ.

6 Covariance Matrices and Decompositions

As described in lecture, the covariance matrix $\mathrm{Var}(R) \in \mathbb{R}^{d \times d}$ for a random variable $R \in \mathbb{R}^d$ with mean μ is

$$\mathrm{Var}(R) = \mathrm{Cov}(R, R) = \mathbb{E}[(R - \mu)(R - \mu)^\top] = \begin{bmatrix} \mathrm{Var}(R_1) & \mathrm{Cov}(R_1, R_2) & \ldots & \mathrm{Cov}(R_1, R_d) \\ \mathrm{Cov}(R_2, R_1) & \mathrm{Var}(R_2) & & \mathrm{Cov}(R_2, R_d) \\ \vdots & & \ddots & \vdots \\ \mathrm{Cov}(R_d, R_1) & \mathrm{Cov}(R_d, R_2) & \ldots & \mathrm{Var}(R_d) \end{bmatrix},$$

where $\mathrm{Cov}(R_i, R_j) = \mathbb{E}[(R_i - \mu_i)(R_j - \mu_j)]$ and $\mathrm{Var}(R_i) = \mathrm{Cov}(R_i, R_i)$.

If the random variable R is sampled from the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ with the PDF

$$f(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-((x-\mu)^\top \Sigma^{-1}(x-\mu))/2},$$

then $\mathrm{Var}(R) = \Sigma$.

Given n points $X_1, X_2, \ldots, X_n$ sampled from $\mathcal{N}(\mu, \Sigma)$, we can estimate Σ with the maximum likelihood estimator

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (X_i - \mu)(X_i - \mu)^\top,$$

which is also known as the covariance matrix of the sample.

(a) The estimate $\hat{\Sigma}$ makes sense as an approximation of Σ only if $\hat{\Sigma}$ is invertible. Under what circumstances is $\hat{\Sigma}$ not invertible? Make sure your answer is complete; i.e., it includes all cases in which the covariance matrix of the sample is singular. Express your answer in terms of the geometric arrangement of the sample points X_i.

(b) Suggest a way to fix a singular covariance matrix estimator $\hat{\Sigma}$ by replacing it with a similar but invertible matrix. Your suggestion may be a kludge, but it should not change the covariance matrix too much. Note that infinitesimal numbers do not exist; if your solution uses a very small number, explain how to calculate a number that is sufficiently small for your purposes.

(c) Consider the normal distribution $\mathcal{N}(0, \Sigma)$ with mean $\mu = 0$. Consider all vectors of length 1; i.e., any vector x for which $|x| = 1$. Which vector(s) x of length 1 maximizes the PDF $f(x)$? Which vector(s) x of length 1 minimizes $f(x)$? (Your answers should depend on the properties of Σ.) Explain your answer.

7 Gaussian Classifiers for Digits and Spam

In this problem, you will build classifiers based on Gaussian discriminant analysis. Unlike Homework 1, you are NOT allowed to use any libraries for out-of-the-box classification (e.g. `sklearn`). You may use anything in `numpy` and `scipy`.

The training and test data can be found on in the post corresponding to this homework. Don't use the training/test data from Homework 1, as they have changed for this homework. Submit your predicted class labels for the test data on the Kaggle competition website and be sure to include your Kaggle display name and scores in your writeup. Also be sure to include an appendix of your code at the end of your writeup.

(a) Taking pixel values as features (no new features yet, please), fit a Gaussian distribution to each digit class using maximum likelihood estimation. This involves computing a mean and a covariance matrix for each digit class, as discussed in lecture.

Hint: You may, and probably should, contrast-normalize the images before using their pixel values. One way to normalize is to divide the pixel values of an image by the l_2-norm of its pixel values.

(b) (Written answer) Visualize the covariance matrix for a particular class (digit). How do the diagonal terms compare with the off-diagonal terms? What do you conclude from this?

(c) Classify the digits in the test set on the basis of posterior probabilities with two different approaches.

(1) Linear discriminant analysis (LDA). Model the class conditional probabilities as Gaussians $\mathcal{N}(\mu_C, \Sigma)$ with different means μ_C (for class C) and the same covariance matrix Σ, which you compute by averaging the 10 covariance matrices from the 10 classes.

To implement LDA, you will sometimes need to compute a matrix-vector product of the form $\Sigma^{-1}x$ for some vector x. You should **not** try to compute the inverse of Σ (nor the

determinant of Σ). Instead, you should find a way to solve the implied linear system without computing the inverse.

Hold out 10,000 randomly chosen training points for a validation set. Classify each image in the validation set into one of the 10 classes (with a 0-1 loss function). Compute the error rate and plot it over the following numbers of randomly chosen training points: [100, 200, 500, 1,000, 2,000, 5,000, 10,000, 30,000, 50,000]. (Expect some variance in your error rate when few training points are used.)

(2) Quadratic discriminant analysis (QDA). Model the class conditionals as Gaussians $\mathcal{N}(\mu_C, \Sigma_C)$, where Σ_C is the estimated covariance matrix for class C. (If any of these covariance matrices turn out singular, implement the trick you described in Q6.(b). You are welcome to use k-fold cross validation to choose the right constant(s) for that trick.) Repeat the same tests and error rate calculations you did for LDA.

(3) (Written answer.) Which of LDA and QDA performed better? Why?

(4) Using the `mnist_data.mat`, train your best classifier for the `training_data` and classify the images in the `test_data`. Submit your labels to the online Kaggle competition. Record your optimum prediction rate in your submission. You are welcome to compute extra features for the Kaggle competition. If you do so, please describe your implementation in your assignment. Please use extra features **only** for the Kaggle portion of the assignment.

In your submission, include plots of error rate versus number of training examples for both LDA and QDA. Similarly, include a plot of validation error for each digit across the training points. Create one graph with the plot for each digit. Which digit is easiest to classify? Include written answers where indicated.

(d) Next, apply LDA or QDA (your choice) to spam. Submit your test results to the online Kaggle competition. Record your optimum prediction rate in your submission. If you use additional features (or omit features), please describe them.

Optional: If you use the defaults, expect relatively low classification rates. The TAs suggest using a bag-of-words model. You may use third-party packages to implement that if you wish. Also, normalizing your vectors might help.