Introduction to Machine Learning **Fall Semester, 2022/23**

Homework 1: October 26, 2022

Due: November 9, 2022

Linear Algebra

1. (**15 pts**) A symmetric matrix A over $\mathbb{R}$ is called *positive semidefinite* (PSD) if for every vector $\mathbf{v}$, $\mathbf{v}^\mathsf{T} A\mathbf{v} \geq 0$.
   (a) Show that a symmetric matrix A is PSD if and only if it can be written as $A = XX^T$. Hint: a real symmetric matrix A can be decomposed as $A = QDQ^T$, where Q is an orthogonal matrix whose columns are eigenvectors of A and D is a diagonal matrix with eigenvalues of A as its diagonal elements.
   (b) Show that for all $\alpha, \beta \geq 0$ and PSD matrices $A, B \in \mathbb{R}^{n\times n}$, the matrix $\alpha A + \beta B$ is also PSD. Does this mean that the set of all $n \times n$ PSD matrices over $\mathbb{R}$ is a vector space over $\mathbb{R}$?

Calculus and Probability

1. (**15 pts**) Let $X_1, ..., X_n$ be i.i.d $U([0, 1])$ continuous random variables. Let $Y = \max(X_1, ..., X_n)$.
   (a) Compute the PDF of Y and plot it. Compute $\mathbb{E}[Y]$ and $\mathrm{Var}[Y]$ - how do they behave as a function of n as n grows large?
   (b) (**No need to submit**) Verify your answer empirically using Python.
2. (**extra credit - 5 pts**) Let X be a continuous random variable. The *median* of X is defined to be the number m which satisfies $\Pr[X \leq m] = \frac{1}{2}$ (Verify that you understand why there must exist such m. You can assume for simplicity that m is unique). Show that

$$m = \mathrm{argmin}_{a\in\mathbb{R}}\, \mathbb{E}[|X - a|]$$

Optimal Classifiers and Decision Rules

1. (**15 pts**)
   (a) Let X and Y be random variables where Y can take values in $\mathcal{Y} = \{1, \ldots, L\}$. Let ℓ_{0-1} be the 0-1 loss function defined in class. Show that $h = \arg\min_{f:\mathcal{X}\to\mathcal{Y}} \mathbb{E}\left[\ell_{0-1}(Y, f(X))\right]$ is given by

$$h(x) = \arg\max_{i\in\mathcal{Y}} \mathbb{P}[Y = i|X = x]$$

   (b) Let X and Y be random variables where Y can take values in $\mathcal{Y} = \{0, 1\}$. Let Δ be the following asymmetric loss function:

$$\Delta(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ a & y = 0, \hat{y} = 1 \\ b & y = 1, \hat{y} = 0, \end{cases}$$

where $a, b \in (0, 1]$ (note that this loss function generalizes the 0-1 loss defined in class). Compute the optimal decision rule h for the loss function Δ, i.e. the decision rule which satisfies:

$$h = \arg \min_{f:\mathcal{X}\to\mathcal{Y}} \mathbb{E}\left[\Delta(Y, f(X))\right]$$

2. (**25 pts**) Let $\mathbf{X} = (X_1, \ldots, X_d)^\mathsf{T}$ be a vector of random variables. $\mathbf{X}$ is said to have a **multivariate normal (or Gaussian) distribution** with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and a $d \times d$ positive definite[1] covariance matrix Σ, if its probability density function is given by

$$f(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

where $\mathrm{E}[X_i] = \mu_i$ and $cov(X_i, X_j) = \Sigma_{ij}$ for all $i, j = 1, \ldots, d$. We write this as $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$.

In this question, we generalize the decision rule we have seen in the recitation to more cases. Assume that the data is $\langle \mathbf{x}, y \rangle$ pairs, where $\mathbf{x} \in \mathbb{R}^d$ and $y \in \{0, 1\}$. Denote by $f_0(\mathbf{x})$ and $f_1(\mathbf{x})$ the probability density functions of $\mathbf{x}$ given each of the label values. It is known that f_0, f_1 are multivariate Gaussian:

$$f_0(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\mu}_0, \Sigma_0),$$
$$f_1(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\mu}_1, \Sigma_1).$$

Also, the probability to sample a positive sample (i.e. $y = 1$) is p. Assume throughout the question that Σ_0, Σ_1 are positive definite diagonal matrices.

(a) **(No need to submit)** Verify that if $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then $X_1, ..., X_d$ are independent normal random variables: $X_i \sim \mathcal{N}(\mu_i, \Sigma_{i,i})$ (recall that we assume Σ is a diagonal matrix). That is,

$$f(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \prod_{i=1}^{d} f(x_i; \mu_i, \Sigma_{i,i}),$$

where $f(x; \mu, \sigma^2)$ is the density of $\mathcal{N}(\mu, \sigma^2)$.

(b) Assume that $\Sigma_0 = \Sigma_1 = \Sigma$. We are given a point $\mathbf{x}$ and we need to label it with either $y = 0$ or $y = 1$. Suppose our decision rule is to decide $y = 1$ if and only if $\mathbb{P}[y = 1|\mathbf{X}] > \mathbb{P}[y = 0|\mathbf{X}]$. Find a simpler condition for $\mathbf{X}$ that is equivalent to this rule. Solve for general $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1 \in \mathbb{R}^d$ and diagonal $\Sigma \in \mathbb{R}^{d\times d}$.

(c) The decision boundary for this problem is defined as the set of points for which $\mathbb{P}[y = 1|\mathbf{X}] = \mathbb{P}[y = 0|\mathbf{X}]$. What is the shape of the decision a general $d > 1$ (think of $d = 1$ and $d = 2$ for intuition)?

(d) Assume now that $d = 1$, $\mu_0 = \mu_1 = \mu$ and $\sigma_0 \neq \sigma_1$. Find the decision rule whenever $f_0(x) = f(x; \mu, \sigma_0^2)$ and $f_1(x) = f(x; \mu, \sigma_1^2)$.

[1] A symmetric matrix A is called *positive definite* if $\mathbf{v}^T A\mathbf{v} > 0$ for all vectors $\mathbf{v}$. In particular, this means that A is invertible since all of its eigenvalues are strictly positive.

Programming Assignment

1. **Nearest Neighbor** (**30 pts**). In this question, we will study the performance of the Nearest Neighbor (NN) algorithm on the MNIST dataset. The MNIST dataset consists of images of handwritten digits, along with their labels. Each image has 28×28 pixels, where each pixel is in gray-scale, and can get an integer value from 0 to 255. Each label is a digit between 0 and 9. The dataset has 70,000 images. Although each image is square, we treat it as a vector of size 784.

   The MNIST dataset can be loaded with `sklearn` as follows:

   ```
   >>> from sklearn.datasets import fetch_openml
   >>> mnist = fetch_openml('mnist_784', as_frame=False)
   >>> data = mnist['data']
   >>> labels = mnist['target']
   ```

   Loading the dataset might take a while when first run, but will be immediate later. See `http://scikit-learn.org/stable/datasets.html` for more details. Define the training and test set of images as follows:

   ```
   >>> import numpy.random
   >>> idx = numpy.random.RandomState(0).choice(70000, 11000)
   >>> train = data[idx[:10000], :].astype(int)
   >>> train_labels = labels[idx[:10000]]
   >>> test = data[idx[10000:], :].astype(int)
   >>> test_labels = labels[idx[10000:]]
   ```

   Make sure you have version 1.0.2 or above of scikit-learn installed or the code may not work properly!

   It is recommended to use `numpy` and `scipy` where possible for speed, especially in distance computations. It is also highly recommended that you debug your code using a much smaller training set (e.g. 100 examples), and then run it on the larger training set once you're sure your code works properly.

   The k-NN algorithm is the first (and most trivial) classification algorithm we encounter in the course. In order to classify a new data point, it finds the k nearest neighbors of that point and classifies according to the majority label. More details can be found on Wikipedia.

   (a) Write a function that accepts as input: (i) a set of train images; (ii) a vector of labels, corresponding to the images; (iii) a query image; and (iv) a number k. The function will implement the k-NN algorithm to return a prediction of the query image, given the train images and labels. The function will use the k nearest neighbors, using the Euclidean L2 metric. In case of a tie between the k labels of neighbors, it will choose an arbitrary option.

   (b) Run the algorithm using the first $n = 1000$ training images, on each of the test images, using $k = 10$. What is the accuracy of the prediction (i.e. the percentage of correct classifications)? What would you expect from a completely random predictor?

   (c) Plot the prediction accuracy as a function of k, for $k = 1, \ldots, 100$ and $n = 1000$. Discuss the results. What is the best k?

(d) Using $k = 1$, run the algorithm on an increasing number of training images. Plot the prediction accuracy as a function of $n = 100, 200, \ldots, 5000$. Discuss the results.