Introduction to Machine Learning **Fall Semester, 2021/22**

Homework 1: October 13, 2021

Due: October 27, 2021

Linear Algebra

1. A matrix A over $\mathbb{R}$ is called *positive semidefinite* (PSD) if for every vector v, $v^\top A v \geq 0$. 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.

Calculus and Probability

1. *Matrix calculus* is the extension of notions from calculus to matrices and vectors. We define the derivative of a scalar y with respect to a vector $\mathbf{x}$ as the column vector which obeys:

$$\left(\frac{\partial y}{\partial \mathbf{x}}\right)_i = \frac{\partial y}{\partial x_i},$$

   for $i = 1, \ldots, n$. Let A be a $n \times n$ matrix. Prove that: $\frac{\partial \mathbf{x}^T A \mathbf{x}}{\partial \mathbf{x}} = (A + A^T)\mathbf{x}$

2. In this question we complete the proof of Hoeffding's inequality from recitation 1, and show that if $\mathbb{E}[X] = 0$ and $|X| \leq B$, then

$$\forall \lambda \quad \mathbb{E}[e^{\lambda X}] \leq e^{B^2 \frac{\lambda^2}{2}}. \tag{1}$$

   Assume that X is continuous with PDF f, and that $f(x) = 0$ for $x \notin [-B, B]$. Given a PDF h, denote

$$\mathbb{E}_h[X] = \int_{-\infty}^{\infty} x h(x) dx \quad \text{and} \quad Var_h(X) = \mathbb{E}_h[X^2] - \left(\mathbb{E}_h[X]\right)^2.$$

   (whenever we don't specify the density, we mean to f). Use the following guidance:

   (a) Define $R(\lambda) = \ln\left(\mathbb{E}[e^{\lambda X}]\right)$. Show that,

$$R''(\lambda) = \frac{\mathbb{E}[X^2 e^{\lambda X}]}{\mathbb{E}[e^{\lambda X}]} - \left(\frac{\mathbb{E}[X e^{\lambda X}]}{\mathbb{E}[e^{\lambda X}]}\right)^2.$$

   (Hint: Leibniz's rule)

   (b) Define a new PDF: $g(x) = \frac{e^{\lambda x}}{\mathbb{E}_f[e^{\lambda X}]} f(x)$. Show that,

$$R''(\lambda) = Var_g(X).$$

   Conclude that $R''(\lambda) \leq B^2$.

   (c) Show that Eq. 1 holds (Hint: use the fundamental theorem of calculus).

Decision Rules and Concentration Bounds

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

2. Let $\mathbf{X} = (X_1, \dots, X_d)^\mathsf{T}$ 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 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 $i, j = 1, \dots, d$. 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 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: $X_i \sim \mathcal{N}(\mu_i, \Sigma_{i,i})$. 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 $f_1(x) = f(x; \mu, \sigma_1^2)$.

3. **Visualizing Hoeffding bound**.
   (a) Use **numpy** to generate $N \times n$ matrix of samples from $Bernoulli(1/2)$. Calculate for each row the empirical mean, $\bar{X}_i = \frac{1}{n}\sum_{j=1}^{n} X_{i,j}$, where $N = 200000$ and $n = 20$.
   (b) Take 50 values of $\epsilon \in [0, 1]$ (**numpy.linspace(0,1, 50)**), and calculate the empirical probability that $|\bar{X}_i - 1/2| > \epsilon$. Plot the empirical probability as a function of ϵ.
   (c) Add to your plot the Hoeffding bound of that probability, as a function of ϵ. Submit your plot (no need to submit code for this question).

Programming Assignment

1. **Nearest Neighbor.** 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/mldata.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:]]
   ```

   It is recommended to use `numpy` and `scipy` where possible for speed, especially in distance computations.

   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 (measured by 0-1 loss; 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 $k = 1, \dots, 100$ 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 $n = 100, 200, \dots, 5000$. Discuss the results.