Introduction to Machine Learning **Spring Semester**

Homework 4: July 22, 2024

Due: August 8, 2024

Theory Questions

1. **(15 points) Sparsity of Lasso estimator.** Consider the solution for LLS with ℓ_1 regularization, also known as Lasso:

$$\hat{\mathbf{a}}^{\text{lasso}} = \arg\min_{\mathbf{a}\in\mathbb{R}^d} \left\{ \frac{1}{2}\|\mathbf{y} - X\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1 \right\}.$$

Show that if $X^T X = \text{diag}(\sigma_1, ..., \sigma_d)$ where $\sigma_j > 0$ then,

$$\hat{a}_j^{\text{lasso}} = \frac{\text{sign}(z_j)}{\sigma_j} \max(0, |z_j| - \lambda),$$

where $z_j = \sum_{i=1}^n y_i X_{ij}$.
(**Hint:** First, rewrite the objective as $\sum_{j=1}^d \left(-z_j a_j + \frac{1}{2}\sigma_j a_j^2 + \lambda|a_j| \right)$. Then, for each j divide the optimization to cases according to the sign of z_j).

2. **(15 points) Suboptimality of ID3.** (Exercise 2 in chapter 18 in the course book: Understanding Machine Learning: From Theory to Algorithms). Consider the following training set where $\mathcal{X} = \{0,1\}^3$ and $\mathcal{Y} = \{0,1\}$:

$$(\mathbf{x}_1, y_1) = ((1,1,1), 1),$$
$$(\mathbf{x}_2, y_2) = ((1,0,0), 1),$$
$$(\mathbf{x}_3, y_3) = ((1,1,0), 0),$$
$$(\mathbf{x}_4, y_4) = ((0,0,1), 0).$$

We learn a classifier using ID3 by building a tree of depth 2, where for each internal node we are only allowed to use predicates of the form $(X_i = 0)$, and at depth 2 we stop and choose leaves according to the majority label of the examples at this leaf. Assume that we choose $C(\cdot)$ as the entropy function to define the information gain, and if two feature have the same gain we break ties arbitrarily.

   (a) Prove that the training error of the resulting tree constructed by ID3 is at least $\frac{1}{4}$.
   (b) Find a tree of depth 2 that achieves zero training error.

3. **(15 points) Step-size Perceptron.** Consider the modification of Perceptron algorithm with the following update rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t y_t \mathbf{x}_t$$

whenever $\hat{y}_t \neq y_t$ ($\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$ otherwise). Assume that data is separable with margin $\gamma > 0$ and that $\|\mathbf{x}_t\| = 1$ for all t. For simplicity assume that the algorithm makes M mistakes at the first M rounds, after which it makes no mistakes. For $\eta_t = \frac{1}{\sqrt{t}}$, show that the number of mistakes step-size Perceptron makes is at most $\frac{4}{\gamma^2}\log(\frac{1}{\gamma})$. (Hint: use the fact that if $x \leq a\log(x)$ then $x \leq 2a\log(a)$). It's okay if you obtain a bound with slightly different constants, but the asymptotic dependence on γ should be tight.

4. **(20 Points) Kernel k-means.** Consider a problem of clustering n *distinct* data points $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n) \in \mathbb{R}^{d'}$ into $k \leq n$ clusters, where $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and $d \ll d'$. Recall that in the k-means algorithm, we initialize centroids $\mu_1^{(1)}, \ldots, \mu_k^{(1)} \in \mathbb{R}^{d'}$ and iteratively perform the following two steps:

   1. *Assignment:* Each data point is assigned to its nearest centroid (with respect to the L_2 norm).
   2. *Re-estimation:* Each centroid μ_j becomes the average of the data points assigned to it.

   Assume that the entries of the kernel matrix $K_{i,j} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ can be computed in time that doesn't depend on d'. Prove that for *every* initialization for which the centroids equal k distinct data points, the algorithm can be implemented in time complexity that doesn't depend on d'. In particular, show how the centroids can be represented and how the assignment and re-estimation steps can be implemented efficiently throughout the runtime of the algorithm.

5. **(15 points) Maximum Likelihood.** Consider a collection of points $x_1, \ldots x_n \in \mathbb{R}$ sampled i.i.d. according to an exponential distribution with an unknown parameter λ. (Recall that an exponential distribution with a parameter λ (denoted by $\mathsf{exp}(\lambda)$) has the density of $\lambda e^{-\lambda x}$ for non-negative x.)

   (a) What is the Maximum Likelihood (ML) estimate of λ?

   (b) In the Bayesian setting, What is the Maximum A Posteriori (MAP) value of λ given that its prior distribution is exponential with parameter 1?

   When maximizing, justify why this is indeed a maximum.

Programming Assignment

Submission guidelines:

- Download the supplied files from Moodle. Written solutions, plots and any other non-code parts should be included in the written solution submission.
- Your code should be written in Python 3.
- Your code submission should include a single python file: `pca.py`.

1. **(20 points) PCA.** In this exercise you will use PCA on pictures of faces. Specific details on how to load and use the data are provided in the skeleton file. As always you must submit all plots and tables together with the theoretical section.

   (a) As a necessary first step you are asked to implement PCA without the use of the Sklearn PCA method (or any other package that implements PCA). You can use Numpy, and in particular the SVD method in Numpy. The signature for the PCA function is located in the file `skeleton_pca.py`.

   (b) Select one person with enough pictures (> 50) from the pool of people. You have a utility function that will help you load pictures of a specific person in `skeleton_pca.py`. Construct a matrix X whose rows are the flattened images of this person. Run PCA on X with $k = 10$ and plot each of the 10 eigenvectors as pictures. In the file `skeleton_pca.py` you have a utility function that will help you plot vectors as pictures. Can you give an interpretation to some of the vectors?

   (c) Use the same matrix X as the previous section. For $k = 1, 5, 10, 30, 50, 100$ reduce the dimension using PCA to dimension k. Select at random 5 pictures (the same 5 pictures for all values of k) and plot the each of the 5 original pictures next to the pictures obtained by transforming the reduced pictures back to the original dimension. Also plot, as a function of k, the sum (over the entire dataset) of the ℓ_2 distances between the two (recall that the objective is closely related to the eigenvalues of the covariance matrix).