Introduction to Machine Learning	**Fall Semester**

Homework 6: June 2, 2021

Due: June 20, 2021

Theory Questions

1. **(10 points) Linear regression with dependent variables.** Consider the regression problem where X is a $n \times d$ data matrix, $\boldsymbol{y}$ is a column vector of size n, and $\boldsymbol{a}$ is a column vector of size d of coefficients. As we discussed in the lecture, if there are dependent variables there are infinite possible solutions that achieve this minimum. One sensible criterion to choose one among all possible solutions, is to prefer a solution with a minimal ℓ_2 norm. That is, we search for $\boldsymbol{a}$ that solves the following problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{a}} \|a\|^2 \\ \text{s.t} \quad & X\boldsymbol{a} = y \end{aligned}$$

Assume that $d > n$ and that the matrix X has rank n (note that it in principle the rank can be smaller than n). And denote by $\boldsymbol{a}^\star$ the optimum of the above problem.

   (a) (No need to submit) Convince yourself that there exists a $\boldsymbol{a}$ such that $X\boldsymbol{a} = y$. Namely, the above problem has at least one feasible solution.

   (b) Show that the optimal $\boldsymbol{a}$ can be written as a linear combination of the data point. Namely, there exists a vector $\alpha \in \mathbb{R}^n$ such that the solution is given by $\boldsymbol{a}^\star = X^T\alpha$.

   (c) Show that you can calculate $\boldsymbol{x}^T\boldsymbol{a}^\star$ for all x by using only dot products between $\boldsymbol{x} \in \mathbb{R}^d$ (hint: express the solution using the kernel matrix $K_S = XX^T$).

Note that the above implies that you can use the "kernel trick" in this case. Namely, you can also work with features $\phi(\boldsymbol{x})$ as long as you can calculate the corresponding kernel.

2. **(15 Points) Kernel PCA.** In the PCA algorithm, we are given a sample $\boldsymbol{x}_i \in \mathbb{R}^d$, for $i = 1, \ldots, n$. We would like to extend it to Kernel PCA, as follows. We are given a mapping function $\phi : \mathbb{R}^d \to H$, where H is a space to which points are mapped. We would like to perform PCA on the mapped points, $\phi(\boldsymbol{x}_i)$. For the Kernel PCA algorithm, we will use the matrix $\bar{K}$ defined as

$$\bar{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where as usual $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$. The algorithm should only use K and not $\phi(\boldsymbol{x})$.

   (a) Recall that in PCA, we require the sample to be mean-centered. Namely: $\frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i = 0$. In regular PCA, we can achieve this by subtracting the mean. For kernel PCA, we would like to achieve this by using the kernel function alone. Denote the following, mean-centered version of $\phi(\boldsymbol{x})$:

$$\mathbf{v}_i = \phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_{t=1}^n \phi(\boldsymbol{x}_t).$$

We would like to calculate the kernel matrix for the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Namely, define the matrix $\bar{K}' \in \mathbb{R}^{m \times m}$:

$$\bar{K}'_{i,j} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle.$$

Show how $\bar{K}'$ can be calculated using only the original kernel matrix $\bar{K}$.

(b) For the rest of the question, you may assume that $\sum_{i=1}^{n} \phi(\boldsymbol{x}_i) = 0$. We would like to apply PCA to the vectors $\phi(\boldsymbol{x}_i)$. Denote by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ the first k principal components in H, corresponding to the sample $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)$. Show that $\boldsymbol{u}_j$ (for $j = 1, \ldots, k$) is a linear combination of $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)$. How can its coefficients be calculated?

(c) Since H can be infinite-dimensional or with a high dimension, we will not look for the principal components themselves, but instead will be satisfied with the ability to perform a dot product of each principal component with the mapping $\phi(\boldsymbol{x})$ of a new point, $\boldsymbol{x}$. More explicitly, let x be a new point. Show how we can calculate

$$\langle \boldsymbol{u}_j, \phi(\boldsymbol{x}) \rangle$$

for $j = 1, \ldots, k$. What is the complexity of the solution?

3. **(15 points) Maximum Likelihood.** Consider a collection of points $x_1, \ldots x_n$ sampled i.i.d. according to an exponential distribution. (Recall that an exponential distribution has a parameter λ and the density of x is $\lambda e^{-\lambda x}$.)

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

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

(c) Consider a mixture of exponential distributions with density: $\frac{1}{K} \sum_{k=1}^{K} \lambda_k e^{-\lambda_k x}$. Derive the EM updates for the parameters λ_k.

When maximizing, justify why this is indeed a maximum.

4. **(15 points) Maximum Likelihood and EM.** Consider the following generative process. First, a random variable $Z \in \{0, \ldots, 31\}$ is generated with probability $\Pr[Z = r] = \theta_r$ for parameters $\theta_0, \ldots, \theta_{31}$. Denote by Z_b the binary representation of Z. For example, if $Z = 18$ then $Z_b = 10010$. Next, two random bits of Z_b are replaced with the digit "2" to create the random variable X. For example, the result of this replacement for $Z = 18$ may be one of "22010", "20210", "20020", etc., each with probability $\binom{5}{2}^{-1} = 0.1$. The sample $x_1, \ldots, x_n$ are n IID samples of the variable X.

(a) What is the log-likelihood of the parameters $\theta_0, \ldots, \theta_{31}$ given the data $x_1, \ldots, x_n$?

(b) Explain why this can be viewed as a model with latent variables, and derive the EM updates.

5. **(15 points) Sparsity of LASSO estimator.** Consider the solution for LLS with $\ell 1$ regularization, also known as LASSO:

$$\hat{\boldsymbol{a}}^{\text{lasso}} = \arg\min_{\boldsymbol{a}} \frac{1}{2} \|\boldsymbol{y} - X\boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1.$$

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

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

where $z_j = \sum_{i=1}^{n} y_i X_{ij}$.

Programming Assignment

1. **(15 points)** 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 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`. Run PCA on this matrix and plot the first 10 eigen-vectors as pictures. In the file `skeleton_pca.py` you have a utility function that will help you plot vectors as pictures. Can you give interpretation to some of the vectors?

   (c) Use the same matrix as the previous section. For $k = 1, 5, 10, 30, 50, 100$ reduce the dimension using PCA of the picture to dimension k and then transform it back to the original dimension. Select at random 5 pictures and plot the original picture besides the transformed picture. Also plot the sum of the $\ell 2$ distances between the two (recall that the objective is closely related to the eigenvalues of the covariance matrix).

2. **(15 points)** Consider Poisson Mixture Model (PMM) with $K = 3$: Given $c_1, c_2, c_3 \geq 0$ such that $c_1 + c_2 + c_3 = 1$ and $\lambda_1, \lambda_2, \lambda_3 > 0$,

$$z_i \sim P \text{ where } P(z_i = z) = c_z \quad ; \quad x_i \sim \text{Pois}(\lambda_{z_i}),$$

   for each $1 \leq i \leq n$.

   (a) Generate data $x_1, ..., x_n$ of size $n = 1000$ using the parameters $\lambda_1 = 5$, $\lambda_2 = 10$, $\lambda_3 = 11$, $c_1 = 0.4$, $c_2 = 0.4$ and $c_3 = 0.2$. You can use `numpy.random.poisson()` for generating a sample from Poisson distribution.

   (b) Derive EM update for PMM, and run EM for $T = 1000$ iterations. Initialize the parameters as $\hat{\lambda}_1 = 1$, $\hat{\lambda}_2 = 2$, $\hat{\lambda}_3 = 3$, and $\hat{c}_z = 1/3$. You can use `scipy.stats.poisson.pmf()` to get the PMF value of Poisson distribution.

   (c) For each $t \in \{5, 50, 100, 500, 1000\}$, plot the true mixture distribution and the estimated mixture distribution (use a different plot for each t).

   Submit you file as `PMM.py`.