ASSIGNMENT 5

Deadline: 11:59 pm, April 29, 2024

Submit via Blackboard with VeriGuide receipt.

Please follow the course policy and the school's academic honesty policy.
Each question is worth ten points.

1. Please briefly answer the following questions.
   (a) What is the softmax function in neural networks? Please write its function formula.
   (b) What is the purpose of using the softmax function?
2. Implement GMM using Python from scratch (Please attach the original code to the submission).
   (a) Generate a set of data points composed of two Gaussians, each with 500 samples. The mean and covariance of clusters is defined as

$$\mu_1 = [0, 0]^T, \Sigma_1 = \begin{bmatrix} 1.0 & 0.6 \\ 0.6 & 1 \end{bmatrix} \tag{2}$$

$$\mu_2 = [3, 0]^T, \Sigma_2 = \begin{bmatrix} 0.5 & 0.4 \\ 0.4 & 0.5 \end{bmatrix} \tag{3}$$

   (b) Visualize the 2D-data in a scatter plot, where the first Gaussian is marked in orange and the second Gaussian is marked in blue (Please paste the figure to the answer sheet).
   (c) Implement the Expectation Maximization(EM) algorithm for GMM and write down the estimated parameters in this answer sheet.
   Step 1: Initialize the mean, covariance and weight parameters. 1) mean μ and 2) covariance matrix Σ are initialized randomly. 3) weight (mixing coefficients) π_i: fraction per class refers to the likelihood that a particular data point belongs to each class.
   Step 2: Expectation Step (E step). Compute the posterior probability given the current model, formulated as

$$r_{i,c} = \frac{\pi_c N(x_i|\mu_c, \Sigma_c)}{\sum_{j=1}^{C} \pi_j N(x_i|\mu_j, \Sigma_j)}, \tag{4}$$

   where $r_{i,c}$ the probability that i-th data point belongs to the c-th Gaussian, and C is the number of Gaussians.
   Step 3: Maximization Step (M step). Maximize the probability that it would generate the data it is currently responsible for. M-step updates the parameters as follows

$$\pi_c = \frac{\sum_{i=1}^{M} r_{i,c}}{M}, \tag{5}$$

$$\mu_c = \frac{\sum_{i=1}^{M} r_{i,c} x_i}{\sum_{i=1}^{M} r_{i,c}}, \tag{6}$$

$$\Sigma_c = \frac{\sum_{i=1}^{M} r_{i,c}(x_i - \mu_c)(x_i - \mu_c)^T}{\sum_{i=1}^{M} r_{i,c}}, \tag{7}$$

where M is the number of data points.

3. Implement Max-Margin Classifier via Python from scratch (Please attach the original code to the submission).

   (a) Generate a linearly separable dataset by yourself. There are four points in the plain, A(0,0), B(0,1) C(1,0), D(1,1). You should

   1. randomly sample 20 points from the area closed by triangle ABC and treat them as class 0.

   2. randomly sample 20 points from the area closed by triangle BCD and treat them as class 1.

   For both classes, **don't choose those points in the line!**

   (b) Finish a naive Sequential Minimum Optimization solver by yourself. Report the training time and training accuracy on your dataset and plot the training loss curve. Plot the classifier and your data in one figure, and mark the support point.

   (c) Test your algorithm with more data (50, 100 each class) and report training time, accuracy.

   There are some hints for this question.

   1. We only consider the simplest linear kernel without soft margin, which means we aim to solve the optimization

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^{\mathbf{T}} \mathbf{x_j} - \sum_{\mathbf{i=1}}^{\mathbf{N}} \alpha_{\mathbf{i}} \tag{8}$$

$$s.t. \quad 0 \leq \alpha_i, i = 1, \ldots, N \tag{9}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{10}$$

   We aim to find a solution to minimize 8 while satisfied 9 and 10. You should initialize all alpha as 0.

   2. Given an illegal solution, α, where at least one variable α_i can not satisfy the KKT condition. Then, randomly choose 2 variables α_i and α_j, and we assume α_i does not satisfy the KKT condition. Then 8 is a function of α_i and α_j. Utilize 10 and transform 8 to a function of α_i, and minimize 8 by optimize α_i. Once you have got a new α_i, clip it by 9 and then update α_j by 10.

   3. Loop 2 until KKT condition satisfied.

   4. If you don't know the idea of SMO, just refer to online documents.

   5. If you can not finish the SMO by yourself, you can use the CVXOPT python package to solve the optimization problem. You will get at most 70% score if you do so.

*** END ***