Problem Set 3
ECE 590 Fall 2019

Instructor: Vahid Tarokh
ECE Department, Duke University

Due: 8:59:59 a.m. EST on Sept. 30, 2019

Problem 1: Maximum Likelihood Estimation for Binary Gaussian Class-Conditional Density

Consider a binary classification problem where each class has a Gaussian class conditional density with shared covariance matrix Σ. The data set is given by $\{\mathbf{x}_n, t_n\}$, $n = 1, \ldots, N$ where $t_n = 1$ denotes class $\mathcal{C}_1$ and $t_n = 0$ denotes class $\mathcal{C}_2$. Recall from the lecture notes that maximizing the likelihood function with respect to Σ is equivalent to maximizing the following function:

$$-\frac{N}{2}\log\det(\Sigma) - \frac{N}{2}\mathrm{Tr}(\Sigma^{-1}S),$$

where $S = \frac{N_1}{N}S_1 + \frac{N_2}{N}S_2$ with S_1 and S_2 denote the emprical covariance matrix for each class respectively and $N_1 = \sum_{\mathbf{x}_n \in \mathcal{C}_1} t_n$ and $N_2 = N - N_1$.
Derive the expression of the optimal covariance matrix Σ that maximizes the above function.

Problem 2: Step-size in Full Gradient Descent

In this problem, we will study the properties of step size that guarantee convergence of Gradient methods for convex objective functions. Consider the objective function:

$$L(x_1, x_2, x_3, x_4, x_5) = \sum_{i=1}^{5} x_i^2 + \sum_{1 \le i < j \le 5} x_i x_j.$$

Starting from the point $\mathbf{x} = (2, 2, 2, 2, 2)$, write a Python program that implements gradient descent to minimize L with a step size $\eta_t = 1/t$ at iteration t, $t \ge 1$ (recall that $\eta_t \to 0$ and $\sum_{t \ge 1} \eta_t \to \infty$). Determine the minimum number of iterations to guarantee that the objective function $L \le 0.1$.

Problem 3: Bayes Decision Rule

Consider a binary classification problem where we are given a dataset $\{x_n, t_n\}_{n=1}^{N}$ i.i.d drawn from some joint distributin $p(x, t)$ with $t_n \in \{-1, 1\}$ (we do not assume any specifc distribution for our dataset). If we use the indicator loss (or the $0-1$ loss), i.e., $L(f(x), t) = \mathbb{I}_{\{x: f(x) \neq t\}}(x)$ for some decisoin rule f and true label t, show that the population risk, $\mathbb{E}(L(f(x), t)$ is minimized by the following dicision rule:

$$f^*(x) = \begin{cases} 1 & \mathbb{P}(t = 1|x) \ge \frac{1}{2}, \\ -1 & \text{otherwise.} \end{cases}$$

where $\mathbb{P}(t = 1|x)$ denotes the conditional probablity. In the litrature, f^* is called the *Bayes decision rule*, and the corresponding population risk, $\mathbb{E}(L(f^*(x), t)$ is called the *Bayes risk.*

Note: The indicator function is defined as follows:

$$\mathbb{I}_A(x) = \begin{cases} 1 & x \in A, \\ 0 & x \notin A. \end{cases}$$

Problem 4: Binary Classification with Pytorch

The goal of this problem is to predict Breast Cancer using features extracted from cell images. For this purpose, you are required to build a Logistic Regression model with Pytorch (see Figure 1). Recall that the model consists of one fully connected layer with a sigmoid output activation function.

- Train the model on the Breast Cancer data set and plot the cross-entropy training loss during the iterations of Gradient Descent and finally report the accuracy on both the training and the test data set (More informations about the data set are given in the Jupyter notebook template that will be posted on Sakai).

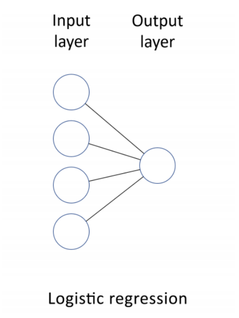

Figure 1: Logistic regression model.

- Now, build another model with two fully connected hidden layers with ReLu activations (see Figure 2) where the first hidden layer has 32 neurons and the second hidden layer has 16 neurons and a sigmoid activation function at the output layer. Train the model on the Breast Cancer data set and report the cross-entropy training loss during the iterations of Gradient Descent and report the accuracy on both the training and the test data set.

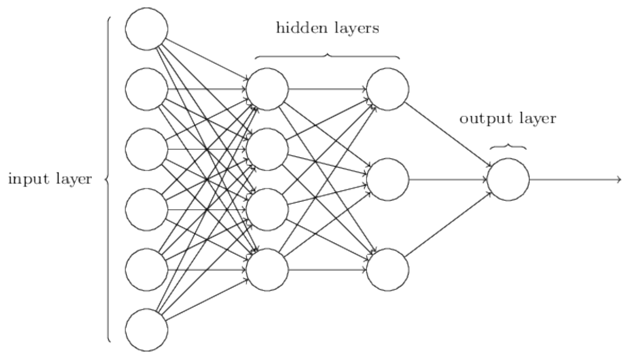

Figure 2: Neural network model with two hidden fully connected layers.

Note that you are not required to implement Gradient Descent from scratch and you are free to use the Optim package from Pytorch.

Problem 5: Implementing a Two-Layer Neural Network

Recall that when training your model you are required to compute the gradient of the loss with respect to the model parameters. Repeat the tasks required in part 2 (See Figure 2) of Problem 3 without using the Optim package from Pytorch and by implementing full Gradient Descent from scratch (Autograd is not allowed). In other words, you have to implemnt the backward propagation and the full gradient descent algorithms by yourself.