Problem Set 3
ECE 685D Fall 2020

Instructor: Vahid Tarokh
ECE Department, Duke University

Due: 8:59:59 a.m. EST on Oct. 6, 2020

Problem 1: Supervised Learning of a Probabilistic Model

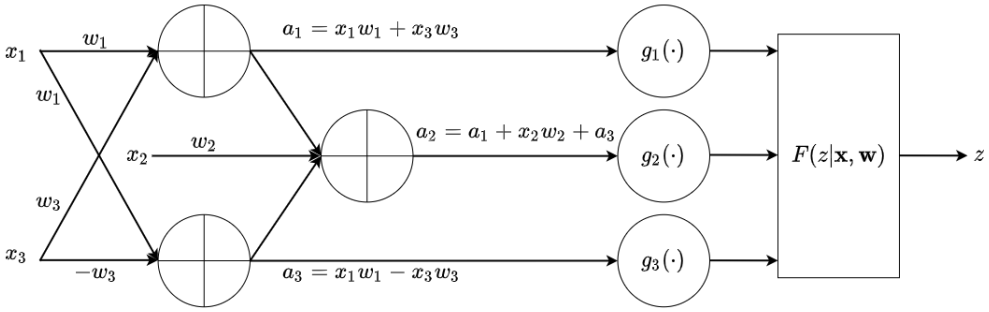

Figure 1: Probabilistic model.

We consider the probabilistic model given in Fig. 1 where $F(z|\mathbf{x}, \mathbf{w})$ denotes the cumulative distribution function (CDF) of the random variable z and defined as:

$$F(z|\mathbf{x}, \mathbf{w}) = \exp\{-\exp\{\mu(\mathbf{x}, \mathbf{w}) - z\}\}$$

where $\mu(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{3} g_i(a_i)$, $\mathbf{x} = (x_1, x_2, x_3)$, $\mathbf{w} = (w_1, w_2, w_3)$, $a_1 = x_1w_1 + x_3w_3$, $a_3 = x_1w_1 - x_3w_3$ and $a_2 = a_1 + x_2w_2 + a_3$, see Fig. 1. In order to determine $\mathbf{w}$, we have generated a training data set of N i.i.d data samples, defined as $\mathcal{D} = \{(\mathbf{x}^{(i)}, z^{(i)})\}, i = 1, \ldots, N$.

Let $g_1(t) = g_3(t) = t$ and $g_2(t) = \sigma(t)$ where $\sigma(t) = \frac{1}{1-\exp(-t)}$ is the sigmoid function.

1. Write an equation for a cost function $\mathcal{L}(\mathbf{w}|\mathcal{D})$ whose minimization gives the maximum likelihood estimate for $\mathbf{w}$.
2. Compute the gradient of $\mathcal{L}(\mathbf{w}|\mathcal{D})$ with respect to the parameter vector $\mathbf{w}$.
3. Write a pseudo-code for minimizing $\mathcal{L}(\mathbf{w}|\mathcal{D})$ using Stochastic Gradient Descent with learning rate $\eta > 0$ and minibatch size $B < N$
4. Compute the Hessian of $\mathcal{L}(\mathbf{w}|\mathcal{D})$ with respect to the parameter vector $\mathbf{w}$.
5. Write a pseudo-code for minimizing $\mathcal{L}(\mathbf{w}|\mathcal{D})$ using Newton's method.

Problem 2: Bayes Decision Rule

Consider a binary classification problem where we are given a data set $\{x_n, t_n\}_{n=1}^{N}$, i.i.d drawn from some joint distribution $p(x, t)$ with $t_n \in \{-1, 1\}$ (we do not assume any specific distribution for our data). 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

decision rule f and true label t, show that the population risk, $\mathbb{E}(L(f(x), t))$ is minimized by the following decision rule:

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

where $\mathbb{P}(t = 1|x)$ denotes the conditional probability. In the literature, f^* is known as 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 3: Binary Classification with Generalized Linear Models

The goal of this problem is to predict Breast Cancer using features extracted from cell images; more information about this data set is provided in the Jupyter notebook template.

1. You are first required to build a Linear Discriminant Analysis (LDA) model for binary classification. Implement and train the LDA model on the training data set and report the training/test loss and classification accuracy. Note that you are required to implement the LDA from scratch, without using the built-in function in the sklearn package; to refresh your memory on the training and classification using LDA models, consult the lecture notes (Lecture notes Part 3, slides 24-34).
2. Next, build a Logistic Regression (LR) model in PyTorch; recall that the LR model consists of set of connections that connect the input with the output through a sigmoid activation function (you may again consult the Lecture notes Part 3). Train the LR model using Gradient Descent on the training data set and plot *i)* the evolution of the training and test cross-entropy loss, and *ii)* the evolution of the training and test classification accuracy. For this part, you may use the Optim package from PyTorch.
3. Write the log-odds (also known as logit function) for both LDA and LR models for binary classification. What is the main difference between these two models?

Problem 4: Binary Classification with Neural Networks

In this problem, you are required to build non-linear binary classification models based on fully-connected feedforward neural networks and train them on the same data set from Problem 3.

1. Consider a neural network for classification with K outcomes. Assume that the neural network uses cross-entropy loss. If the network has no hidden layer, show that the model is equivalent to the multinomial LR model.
2. Build a fully-connected Multilayer Perceptron (MLP) with N hidden layers, M neurons per hidden layer, ReLU activations and a sigmoid activation for the output neuron.
   a Fix $N = 2, M = 16$, train the MLP model using Gradient Descent on the Breast Cancer training data set and plot *i)* the evolution of the training and test cross-entropy loss, and *ii)* the evolution of the training and test classification accuracy.
   b Let $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $\mathcal{M} = \{2, 4, 8, 16, 32, 64, 128\}$; train the MLP model for each pair $(N, M), N \in \mathcal{N}, M \in \mathcal{M}$. Plot the final test classification accuracy against (M, N). Plot the magnitude of the gradient of the cross-entropy loss with respect to the weights connecting the input layer with the first hidden layer, against (M, N). What do you notice?

   Note that for part 2 you are free to use the Optim package from PyTorch.
3. Repeat the tasks from part 2a without using the Optim package and by implementing Gradient Descent from scratch (Autograd is not allowed for this part). In other words, you are required to implement the backward propagation and the full gradient descent algorithms from scratch.

Problem 5: First-order Optimization Methods

In this problem, we want to fit a mutinomial Logistic Regression (LR) model (see Fig. 2) to the MNIST data set using $\mathcal{L}_2$-regularized cross-entropy loss function.

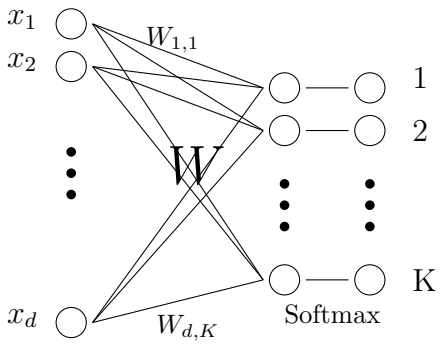

Figure 2: Logistic Regression model

Let K be the number of classes (for MNIST $K = 10$) and let $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}, n = 1, \ldots, N$ denote the data set where $\mathbf{y}_n$ is the label of the n-th data point, represented as a K-dimensional vector such that its k-th entry is 1 if the data point belongs to class k and 0 otherwise (i.e., 1-of-K encoding) whereas $\mathbf{x}_n$ is the corresponding d-dimensional feature vector. Then, the cross-entropy loss can be written as:

$$\mathcal{L}(\mathbf{W}|\mathcal{D}) = \sum_{n=1}^{N} -\mathbf{y}_n^T \log \text{Softmax}(\mathbf{W}\mathbf{x}_n) + \lambda \|\mathbf{W}\|_F^1, \quad [\text{Softmax}(\mathbf{a})]_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)}.$$

Write a Python program that fits the model using the following optimization methods (see lecture notes):

1. Momentum method with parameter $\beta = .9$
2. Nesterov's accelerated gradient (NAG) method with parameter $\beta = .95$
3. RMSprop with parameters $\beta = .9$, $\gamma = 1$ and $\epsilon = 10^{-8}$
4. Adam with parameters $\beta_1 = .9$, $\beta_2 = .999$ and $\epsilon = 10^{-8}$

with learning rate $\eta = .001$ and batch sizes $\{1, 500, 6 \times 10^4\}$. Report the classification accuracy on the test data set for $\lambda \in \{0.01, 0.1, 1\}$.

Note: You can use the Autograd package from Pytorch to compute the gradient. However, you are **not allowed** to use any built-in optimizers in Python. Instructions on how to download and prepare the data will be given in a Jupyter notebook that will be posted on Sakai.