Data-Driven Techniques: 1st Homework

Problem 1: Consider the random vector $X = [x_1, x_2]$ of length 2 for which we have the following two hypotheses:

H_0 : x_1, x_2 are independent with probability density $\mathsf{f}_0(x_1, x_2) = \mathsf{f}_0(x_1)\mathsf{f}_0(x_2)$ where $\mathsf{f}_0(x) \sim \mathcal{N}(0, 1)$.

H_1 : x_1, x_2 are independent with probability density $\mathsf{f}_1(x_1, x_2) = \mathsf{f}_1(x_1)\mathsf{f}_1(x_2)$ where $\mathsf{f}_1(x) \sim 0.5\{\mathcal{N}(-1, 1) + \mathcal{N}(1, 1)\}$.

Suppose that the two prior probabilities are equal to $\mathsf{P}(\mathsf{H}_0) = \mathsf{P}(\mathsf{H}_1) = 0.5$.

a) What is the optimum Bayes test that minimizes the decision error probability?

b) You will need to compute the error probability with the help of simulations. Generate 10^6 **pairs** $[x_1, x_2]$ from $\mathsf{f}_0(x_1, x_2)$ and another 10^6 pairs from $\mathsf{f}_1(x_1, x_2)$. Regarding $\mathsf{f}_1(x)$ with probability 0.5 the sample has mean equal to -1 and with probability 0.5 mean equal to 1. After you generate the two sets of data you must apply the Bayes rule and count the **percentage** of wrong decisions for each data set. From these two error probabilities compute the total probability of error. This is the optimum percentage that cannot be improved by any other decision method.

c) The pairs that you generated in the previous question you need to keep them to answer this question. Generate an **additional** 200 pairs for each hypothesis to use them for the training of neural networks. We are interested in a fully connected network of dimensions $2 \times 20 \times 1$ (input size 2, hidden layer of size 20 and a single output). Apply the method that estimates the likelihood ratio function $r(x_1, x_2) = \frac{\mathsf{f}_1(x_1)}{\mathsf{f}_0(x_1)} \frac{\mathsf{f}_1(x_2)}{\mathsf{f}_0(x_2)}$ using some strictly increasing transformation. In particular apply 1) the cross-entropy method with $\phi(z) = -\log(1 - z)$, $\psi(z) = -\log(z)$ that estimates the posterior probability $\frac{r(x_1,x_2)}{1+r(x_1,x_2)}$ and 2) the exponential with $\phi(z) = e^{0.5z}$, $\psi(z) = e^{-0.5z}$ that estimates the log-likelihood ratio $\log r(x_1, x_2)$. Use the 200+200 training data to train two neural networks by applying the stochastic gradient descent. Every time the data are exhausted (epoch) continue re-using the same data starting from the beginning. You may verify convergence by following the cost value $\phi(u(x^0_{1,t}, x^0_{2,t}, \theta_t)) + \psi(u(x^1_{1,t}, x^1_{2,t}, \theta_t))$ after smoothing it by averaging the current and 19 previous cost values. In the case of cross-entropy recall that we need to apply at the output a nonlinearity so that the final output is in the interval [0,1] (e.g. sigmoid). In the case of the exponential method this is not necessary since the log-likelihood ratio takes any (positive or negative) real value. After convergence you must take the two neural networks and apply them on the $10^6 + 10^6$ samples you have from Question b) in order to compute the error percentage of the two neural networks. Recall that the likelihood ratio test decides in favor of H_1 when $\omega\big(r(x_1, x_2)\big) > \omega(1)$ and H_0 when $\omega\big(r(x_1, x_2)\big) < \omega(1)$. If you apply the log-likelihood ratio then we compare $\log r(x_1, x_2)$ with 0, while when using the posterior probability the test becomes comparison of $\frac{r(x_1,x_2)}{1+r(x_1,x_2)}$ with 0.5. The $\log r(x_1, x_2)$ is approximated by the neural network of the exponential method and $\frac{r(x_1,x_2)}{1+r(x_1,x_2)}$ by the neural network (followed by the sigmoid) of the cross-entropy method. How do the two neural network based methods compare with the optimal Bayes?

Problem 2: Apply the idea of Question c) of Problem 1 to the data of the database MNIST. Isolate the numerals 0 and 8 and design a classifier that distinguishes between the two numerals by employing a neural network that you need to train using a) Cross-entropy and b) Exponential. MNIST has 5500 images of dimension 28×28 with pixels in gray intensity for each case. Use a full neural network with dimensions $784 \times 300 \times 1$ ($784 = 28 \cdot 28$ inputs (we transform the images into vectors of length 784), 300 neurons for the hidden layer and 1 output). Make sure the pixel

intensity is normalized to values in the interval [0,1]. If the gray levels are in [0,255] simply divide each value by 255.

MNIST also contains testing images in addition to the ones for training. For every numeral there is a different number of testing images. Once you design the neural network using the training data apply it to the testing data to compute the percentage of errors. Make a table for each method where we can see the error percentage per numeral and the total error percentage.

Hint: The initialization of the parameters of a neural network must be made as follows: Offset must be 0. Each layer has a matrix of weights say of dimensions $m \times n$. Then the elements of the matrix are selected to be realizations of a Gaussian with mean 0 and variance $\frac{1}{n+m}$. To apply the Gradient Descent you will need to compute the gradients with respect to the network parameters. For your convenience there is the file *derivative.pdf* that explains how to compute these derivatives. Also it explains how to normalize the derivatives using the ADAM method which assures a relatively uniform convergence during training.

Attention!

- Please submit your report **ONLY if you are taking the class for credit and your name appears in Progress**. Reports with names not appearing in Progress will not be evaluated.
- You must upload your report to the e-class by Sunday 17 November, before 12 midnight. There will be no extension to this deadline.
- The ONLY acceptable file type is PDF. Name your file as follows: firstname-lastname-1.pdf
- In the first page of your report write your full name and university code (Arithmos Mitroou) legibly.
- DO NOT send Python or Matlab code in separate files. Please include your code into the pdf file as text AFTER the presentation of your results. Code only, is not considered presentation of results and if your report contains only code then your grade will be 0.
- Do not simply report final probabilities. Since you will be using gradient descent algorithms present the learning curves where we can see how the algorithms converge.
- Please make sure your files are small to avoid communication problems because of excessive size.