Module 8 Exam Answers 2019

Question 1 (13 points)
Suppose we perform logistic regression to predict, from a number of measurements on a device, whether this device is in good or poor working condition. We code "good working condition" as 1 and "poor working condition" as 0. We assume that we have a training set of m devices $\left(\left(x^{(i)}\right),\left(y^{(i)}\right)\right)$ where, for each device i the vector $x^{(i)}$ contains the measurements and $y^{(i)}$ is the known working condition (equal to 0 or 1). We call $h_\theta\left(x^{(i)}\right)$ the hypothesis produced by logistic regression for device i. A sigmoid activation function Is used.
The expression of the cross-entropy loss function over the m devices of the training set is:

$$-\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)+\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]$$

Questions :

- **Explain how $h_\theta\left(x^{(i)}\right)$ is calculated and what is its range of variation. How can $h_\theta\left(x^{(i)}\right)$ be interpreted (5 points) ?**
- **Is the cross-entropy negative or positive (2 points)?**
- **Explain how the cross-entropy treats high and low values of $h_\theta\left(x^{(i)}\right)$ in the case of a device i in good working condition (3 points).**
- **Explain how the cross-entropy treats high and low values of $h_\theta\left(x^{(i)}\right)$ in the case of a device i in poor working condition (3 points).**

Answer:

- $h_\theta\left(x^{(i)}\right)$ is equal to $\sigma\left(\theta^T x^{(i)}\right)$, where σ is the sigmoid function, and θ is the vector of parameters to be trained. $h_\theta\left(x^{(i)}\right)$ varies between 0 and 1 and can be interpreted as a probability.
- The cross-entropy is always positive because the two logarithm terms are negative and $y^{(i)}$is either 0 or 1.
- The first term of the cross-entropy quantifies the discrepancies between the samples labelled as $y^{(i)}$=1 and their predicted probabilities. If $h_\theta\left(x^{(i)}\right)$ is close to 1, $y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)$ is close to 0. And the closer $h_\theta\left(x^{(i)}\right)$ is to 0, the closer to minus infinity is $y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)$, and hence the closer to plus infinity is the loss function.
- The second term of the cross-entropy quantifies the discrepancies between the samples labelled as $y^{(i)}$=0 and their predictions. If $h_\theta\left(x^{(i)}\right)$ is close to 0, $\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)$ is close to 0. And the closer $h_\theta\left(x^{(i)}\right)$ is to 1, the closer to minus infinity is $y^{(i)}\log\left(1-h_\theta\left(x^{(i)}\right)\right)$, and hence the closer to plus infinity is the loss function.

Question 2 (12 points)
Suppose that the last layer of a neural network includes 10 neurons, and that the values of the 10 neurons are: (-3, 2, 4, 1, 7, -5, -1, -3, 3, 5).
For each coordinate a_i of a vector $(a_1, a_2, ., a_9, a_{10})$, the Softmax function is defined as:

$$\textbf{Softmax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{j=10} e^{a_j}}$$

Questions:

- **What are the values obtained by Softmax for the 10 output neurons (calculations up to two decimal places) (4 points)?**
- **In less than 100 words, interpret the results of the Softmax function and explain how it is used in practice. (8 points)**

Answer:

- Let us first calculate the exponential of each term. We obtain: 0.05, 7.39, 54.60, 2.72, 1096.63, 0.01, 0.37, 0.05, 20.09, 148.41.
  The sum of these values is: 1330.31
  We now divide each exponential by this sum and obtain:
  (0.00, 0.01, 0.04, 0.00, 0.82, 0.00, 0.00, 0.00, 0.02, 0.11)
- These numbers can now be interpreted as probabilities (they are positive and add up to one) and we see how Softmax magnifies the differences between the 10 different input numbers and gives a very high probability (0.82) to the value (7) that was initially slightly higher than the 9 others. The Softmax function transforms its 10-dimensional input vector into a 10-dimensional probability vector. The maximum coordinate of this probability vector is interpreted as the coordinate corresponding to the predicted class.

Question 3 (18 points)
You have a training set of 1000 labelled images that you want to use to build a binary classification neural network. Each image is a color image of 128x128 pixels. The first class is that of images containing a car and the second class is that of images containing a plane. 500 images contain a car and 500 contain a plane.
In addition, you have a test set of 200 labelled images which will be used to evaluate the generalization performance of your trained model. 100 images of the test set contain cars and 100 contain planes.
Questions:

- **What is the benefit of using transfer learning to train your neural network? (5 points)**
- **In this example, would it be more appropriate to start from a network trained on MNIST or trained on Imagenet? Explain your answer. (5 points)**
- **Give an example of three hyperparameters to optimize. (3 points)**
- **Which approach do you recommend to choose the optimal values of your hyperparameters? (5 points)**

Answer:

- Because the Training Set is small, it is better to use as a starting point a network already trained to do classification on a similar but much larger dataset. Transfer

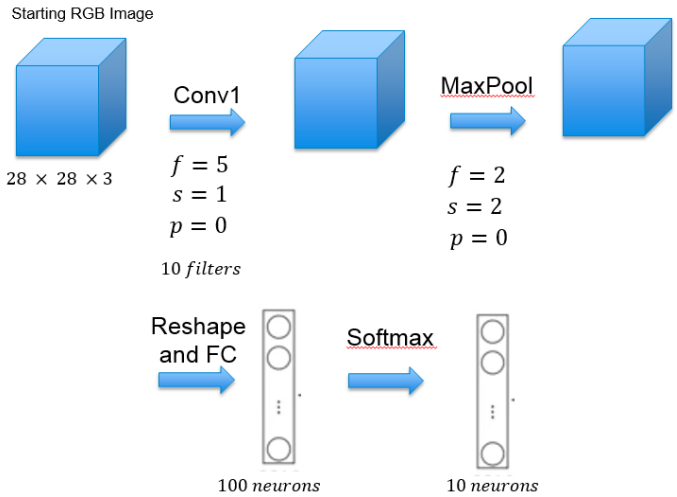
Learning will then consist of simply re-training the output layer or possibly two or three of the last layers.

- ImageNet contains 1000 different classes of images with their labels, including a class for cars and one for planes. MNIST contains images of hand-written digits with their labels. Therefore the ImageNet database is much more appropriate for our purpose.
- The hyper-parameters to fit could be for example the number of layers to re-train, the learning rate, or the size of the mini-batches to use for the re-training.
- The approach for optimizing the hyper-parameters can consist of creating a Validation Set from the Training Set. Based on the distribution of classes in the Training Set, a Validation Set of 100 images containing cars and 100 images containing planes can be used, leaving 800 data points in the Training Set. Then the hyper-parameters are chosen as those producing the best predictions on the Validation Set. Another approach would be to use (for example) 4-fold validation, where we would in rotation re-estimate 250 images from the 750 other images of the Training Set, and again pick the hyperparameter producing the best average result.

Question 4 (30 points)

The figure below describes a simple convolutional network. We assume that each convolution operation is performed with a bias term.

- **Calculate the number of parameters associated with each of the three layers (Conv1, Reshape and FC, and Softmax). (20 points)**
- **Calculate the number of output neurons in the Conv1 and MaxPool layers. (10 points)**



Answer:

	Size of input image n	Number of input channels	Convolution filter f	Padding p	Stride s	Size of output image (n+2p-f)/s + 1	Number of output channels or filters	Number of output neurons	Size of Filter + 1	Number of Parameters
Conv1	28	3	5	0	1	24	10	5760	76	760
MaxPool	24	10	2	0	2	12	10	1440		
	Size of input							Number of output neurons		
Reshape and FC	1440							100	1441	144100
Softmax	100							10	101	1010
							Total Neurons	7310	Total Parameters	145870

Question 5 (12 points)

- **Suppose the random variable X follows a uniform distribution between 0 and 1. λ is a positive parameter. What is the cumulative density function (cdf) of the random variable Y defined as $Y = -\frac{log(1-X)}{\lambda}$ (7 points)?**
- **What is the probability density function (pdf) of Y (5 points)?**
  Note that this transformation is used to generate samples of an exponential distribution, which is used frequently in the study of time-dependent stochastic processes.

Answer:

- We can write the cdf of Y as:

$$P(Y < y) = P\left(-\frac{log(1-X)}{\lambda} < y\right) = P\left((1-X) > e^{-\lambda y}\right) = P\left(X < \left(1 - e^{-\lambda y}\right)\right).$$

  But, since X is a uniform distribution and since $\left(1 - e^{-\lambda y}\right)$ is between 0 and 1, we have:

$$P\left(X < \left(1 - e^{-\lambda y}\right)\right) = 1 - e^{-\lambda y}$$

  This is the cdf of an exponential distribution.
- The corresponding pdf is the derivative of this cdf, that is: $g(y) = \lambda e^{-\lambda y}$

Question 6 (15 points)
This question is about the similarities and the differences between Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs).

- **Explain how each technique treats the latent vector to obtain a sample in the model space (9 points).**
- **Define what is an encoder and a decoder, and explain how VAEs and GANs address encoding and decoding (6 points).**

Answer:

- GANs apply a generator neural network G to a latent vector z, which is usually Gaussian. Once $G(z)$ has been trained jointly with the discriminator $D(G(z))$, each $x = G(z)$ in the model space, for every random value of z , can be considered a sample of p_{data}.

VAEs associate to each latent vector z the mean $\mu(z)$ and the variance-covariance matrix $\Sigma(z)$ of a multiGaussian variable in the model space, and then sample x in the model space from this multi-Gaussian pdf $N\big(x; \mu(z), \Sigma(z)\big)$.

- A decoder associates a sample x in the model space to a latent vector z. An encoder associates a latent vector z to a data sample x.
  A GAN does not automatically calculate the latent vector associated with each data point from the training dataset. Hence a GAN is a decoder, but it is not an encoder. VAEs provide both a decoder and an encoder. They associate to every point $x^{(i)}$ of the training dataset a distribution of random latent vectors. This is provided by the trained distribution $N\left(z^{(i)}; \mu\big(x^{(i)}\big), \Sigma\big(x^{(i)}\big)\right)$. They also provide, as GANs, a decoder which is also a multi-Gaussian pdf $N\big(x; \mu(z), \Sigma(z)\big)$.