Master MVA
Object Recognition and Computer Vision
Assignment 3: Neural Network - 22/11/2016
ACHARI BERRADA YOUSSEF

1 Training a neural network by back-propagation

QA- Computing gradients of the loss with respect to network parameters

It is useful to write down the dimensions of all the variable and the parameters:

Variables	Parameters
$X \in \mathbb{R}^{2\times 1}$	$W_i \in \mathbb{R}^{5\times 2}$
$H \in \mathbb{R}^{5\times 1}$	$B_i \in \mathbb{R}^{5\times 1}$
$Y \in \mathbb{R}^{1}$	$W_o \in \mathbb{R}^{1\times 5}$
	$B_o \in \mathbb{R}^{1}$

Also the equations are :

$$\begin{aligned} H &= \mathrm{ReLU}(W_i X + B_i) \\ Y(X) &= W_o H + B_o \\ s(Y(X), Y) &= \log\left[1 + \exp(-Y.Y(X))\right] \end{aligned}$$

Chain Rule:

$$\frac{\partial s}{\partial Y(X)} = -Y\sigma(-Y.Y(X))$$

- Backpropagation Output:

$$\frac{\partial Y(X)}{\partial W_o} = H^T \quad , \quad \frac{\partial Y(X)}{\partial B_o} = 1$$

$$\Rightarrow \quad \frac{\partial s}{\partial W_o} = \frac{\partial s}{\partial Y(X)}.\frac{\partial Y(X)}{\partial W_o} = -Y\sigma(-Y.Y(X))H^T \quad , \quad \frac{\partial s}{\partial B_o} = \frac{\partial s}{\partial Y(X)}.\frac{\partial Y(X)}{\partial B_o} = -Y\sigma(-Y.Y(X))$$

- Backpropagation Input:

$$\frac{\partial H}{\partial W_i} = \mathbb{1}_{W_i X + B_i > 0} X^T \quad , \quad \frac{\partial H}{\partial B_i} = \mathbb{1}_{W_i X + B_i > 0}$$

Hence we have:

$$\begin{aligned} \frac{\partial s}{\partial W_i} &= \frac{\partial s}{\partial Y(X)}.\frac{\partial Y(X)}{\partial H}.\frac{\partial H}{\partial W_i} \\ &= -Y\sigma(-Y.Y(X)).W_o^T \odot \mathbb{1}_{W_i X + B_i > 0} X^T \end{aligned} \quad , \quad \begin{aligned} \frac{\partial s}{\partial B_i} &= \frac{\partial s}{\partial Y(X)}.\frac{\partial Y(X)}{\partial H}.\frac{\partial H}{\partial B_i} \\ &= -Y\sigma(-Y.Y(X)).W_o^T \odot \mathbb{1}_{W_i X + B_i > 0} \end{aligned}$$

QB- Numerically verify the gradients

1-

$$\frac{s(\Theta + \Delta\Theta) - s(\Theta)}{\Delta\Theta} = \frac{\partial s}{\partial \Theta}$$

2- See figure 1.

QC- Training the network using backpropagation and experimenting with different parameters.

1- The error converge to 0.0% in the training and validation set after $30,000$ iterations. See figure 2.

2- **Random Initialization:** Due to a bad initialization, the error may not converge to zero in the training and validation set.
See figure 3.

3- **Learning Rate:** When the learning rate is 2, the error does not converge to zero. When it is 0.002, it converge but slowly. See figure 4.

4- **The number of hidden units:** We notice that there is a minimum number of hidden units in order to have convergence. Also when we have a big number of hidden units, we have convergence in few iterations but computational time is long. So we have to make a compromise on efficiency and robustness in one hand and computational cost in another hand. See table 1.

Table 1: Number of hidden units sensibility.

h	1			2			5			7		
Training Error	7.8	9.2	9.2	7.2	7.0	8.0	9.0	6.6	6.9	7.6	6.1	0
Validation Error	10.7	11.2	11.8	8.6	7.9	10.3	8.5	7.5	7.9	9.7	7.7	0
Iteration to Convergence	No	No	No	No	No	No	No	No	No	No	No	15K

h	10			100		
Training Error	0	0	0	0	0	0
Validation Error	0	0	0	0	0	0
Iteration to Convergence	20K	15K	10K	6K	5K	7K

2 Image classification with CNN features

Q2A- Computing CNN features

1- The code normalize each image in format to fit the input of the convolutional neural network stored in the net structure we downloaded.

2- CNN :

- net: Is the convolutional neural network (CNN) that has been pre-trained on a large-scale ImageNet classification task. So it is composed by 20 hidden layers and a last 'prob' layer of size 1000 who output the probability of the classification. So it encodes the weights and the properties and type of each layer `net.layers(k)` .
- res: stores the features on each layer. `res(k).x` stores the feature propagated in the layer k of a given image.

Q2B- Image classification using CNN features and linear SVM

1- See figure 6.

2- CNN have better result in classification that the other methods seen in Assignment 2. And there is no need to normalize features.

Table 2: CNN performance with multiple layers.

		Aeroplane	Motorbike	Person
CNN	Final Layer	0.94	0.89	0.90
		36/36	35/36	36/36
	Layer 8	0.78	0.67	0.81
		36/36	32/36	36/36
	Layer 7	0.76	0.64	0.78
		34/36	32/36	35/36

See figures 7,9 and 11 for false positive classification in each classe.
And See figures 8,10 and 12 for false negative classification in each classe.

Figures:

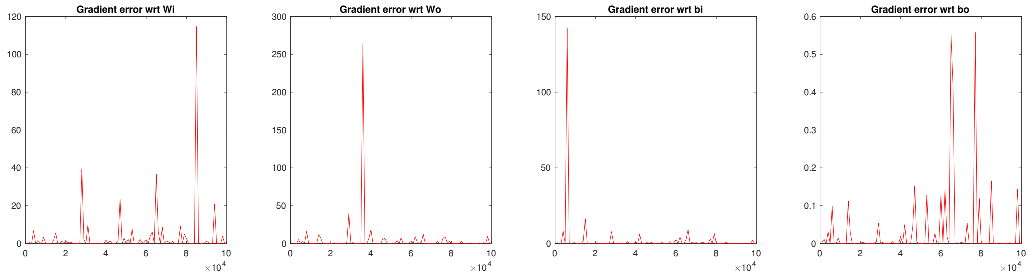


Figure 1: Gradient Error for $h = 3$.

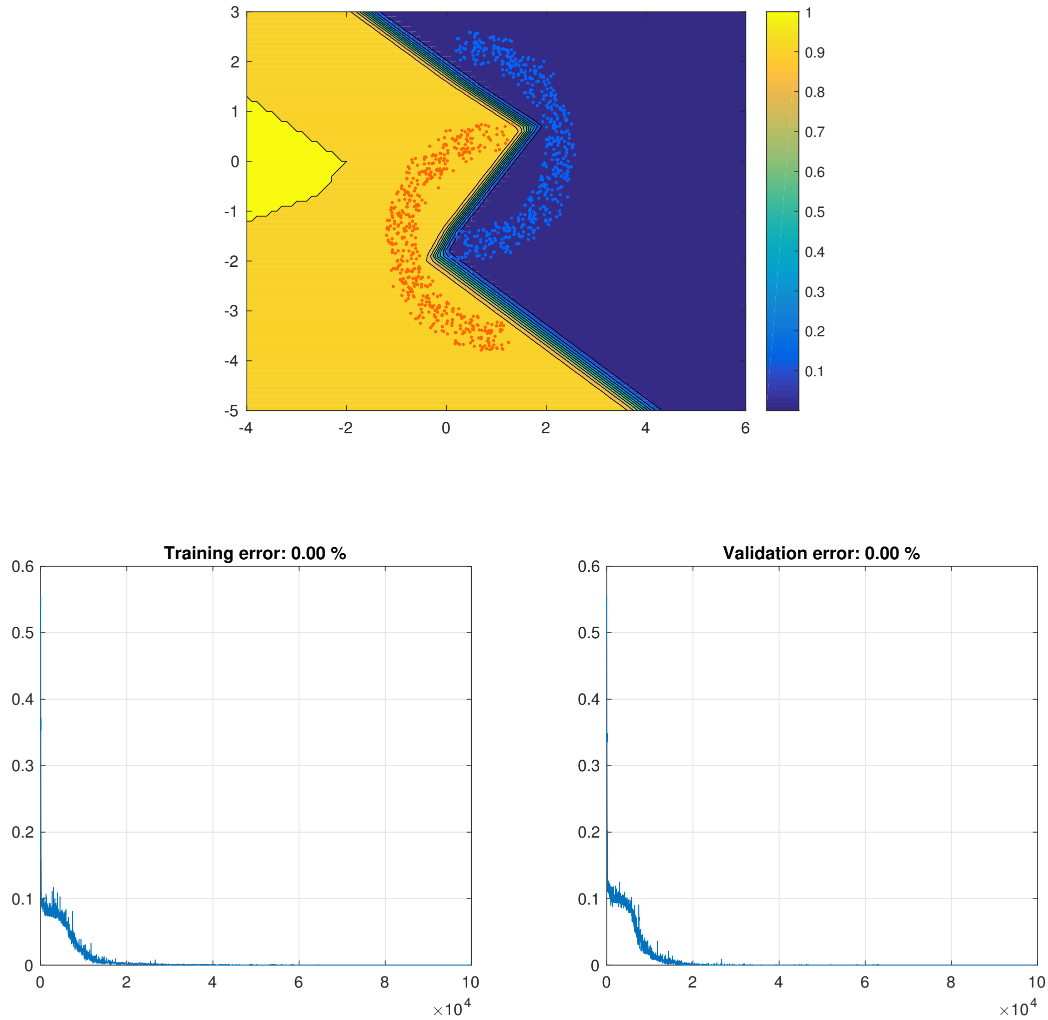


Figure 2: UP: Decision Hyperplan. Down: Training error and Validation Error. Number of hidden units: 7.

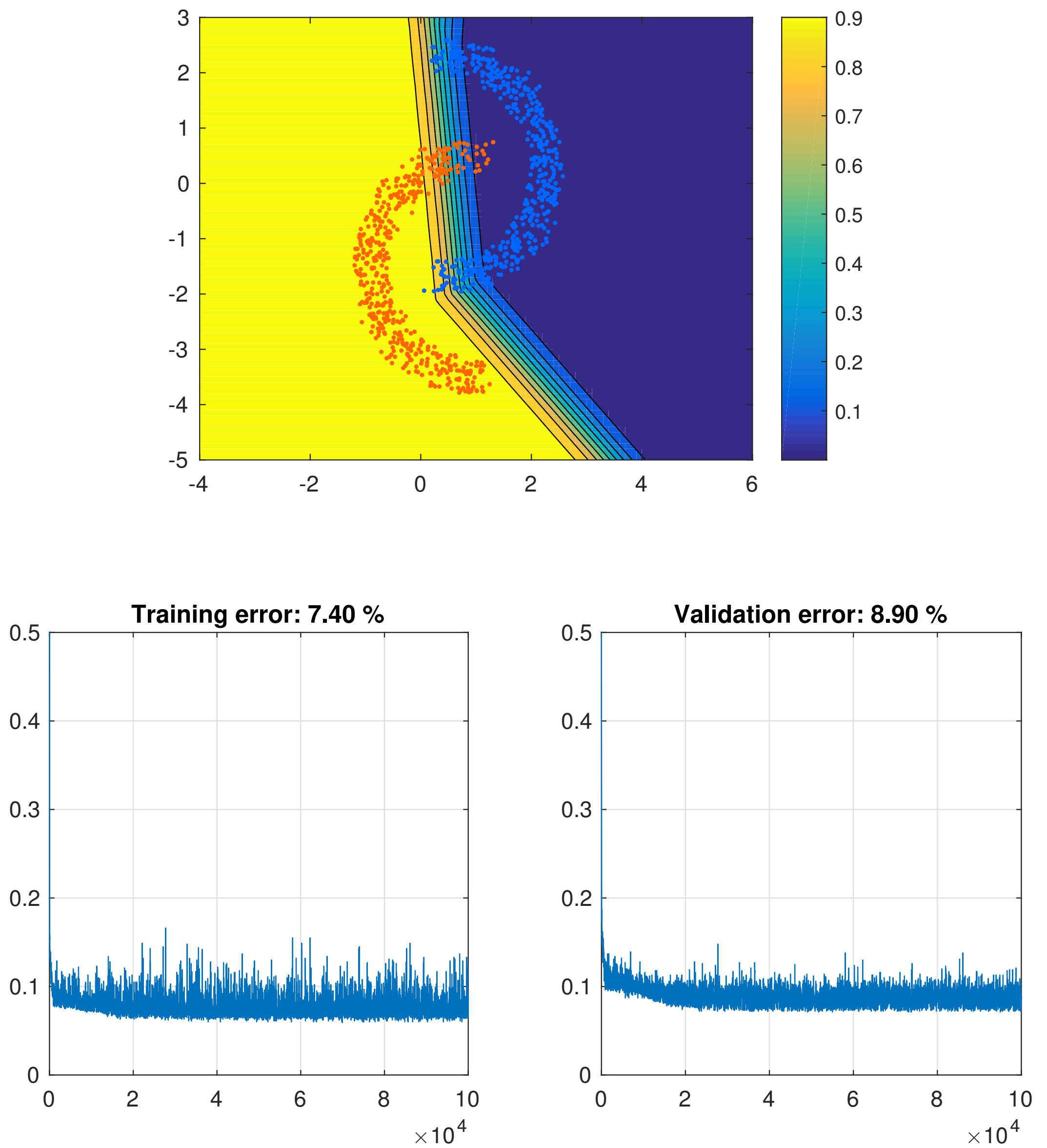


Figure 3: Non convergence to zero of the training and validation error due to bad initialization. Number of hidden units: 7.

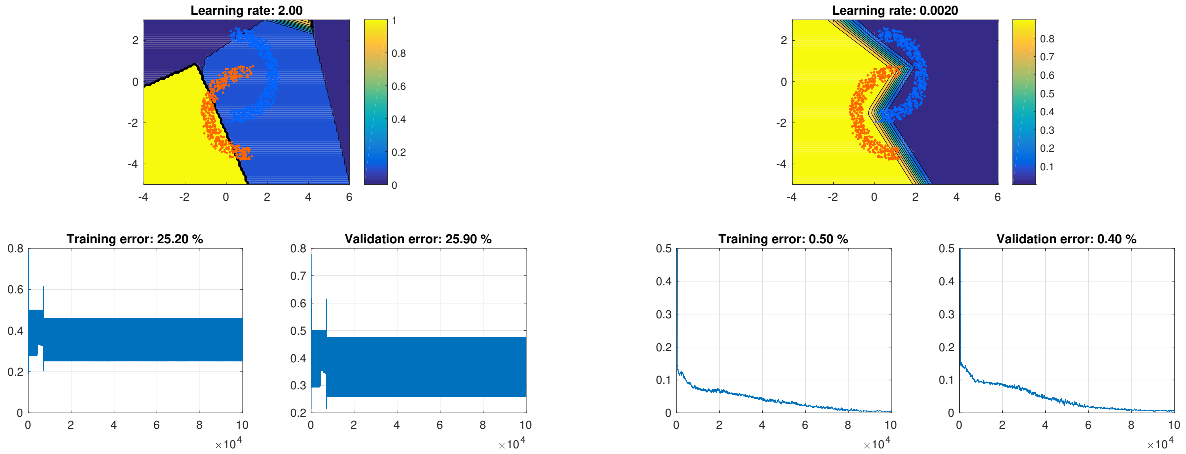
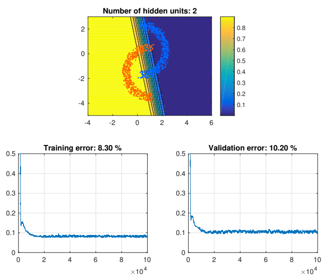
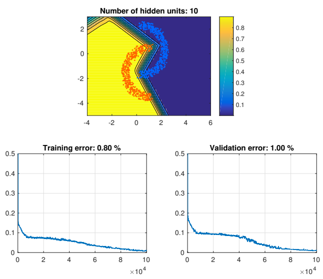


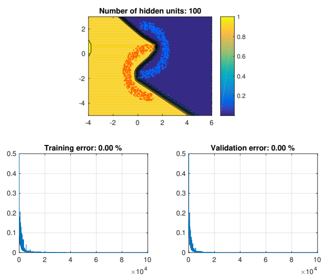
Figure 4: Learning rate sensibility.



(a) h=2

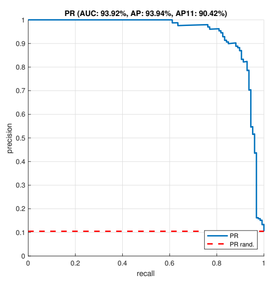


(b) h=10

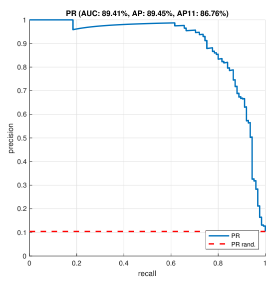


(c) h=100

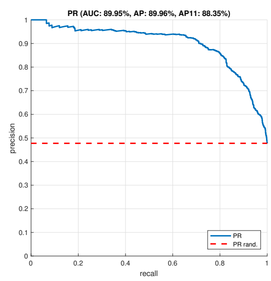
Figure 5: Number of hidden units sensibility.



(a) Aeroplane AP = 0.94



(b) Motorbike AP = 0.89



(c) Person AP = 0.90

Figure 6: Precision-Recall curves on the test images and their AP scores with CNN last layer representation.

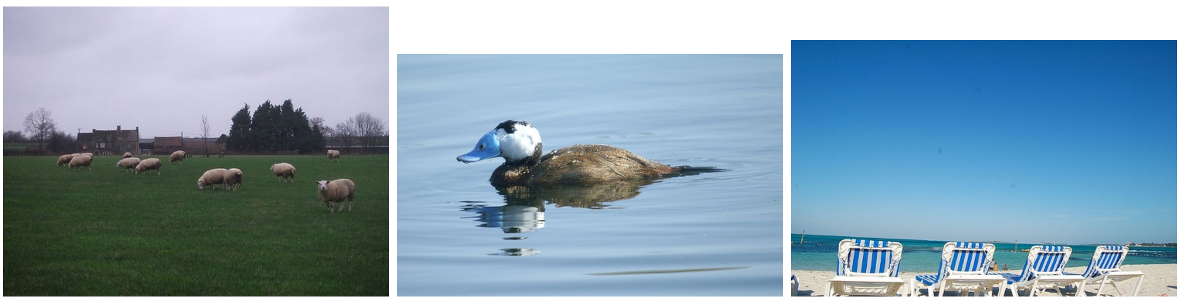


Figure 7: False positive images for Aeroplanes.

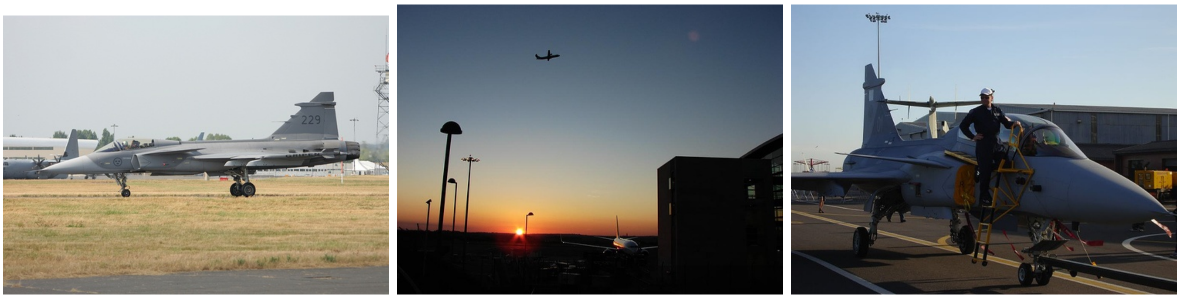


Figure 8: False negative images for Aeroplanes.

Figure 9: False positive images for Motorbikes.

Figure 10: False negative images for Motorbikes.

Figure 11: False positive images for Persons.

Figure 12: False negative images for Persons.