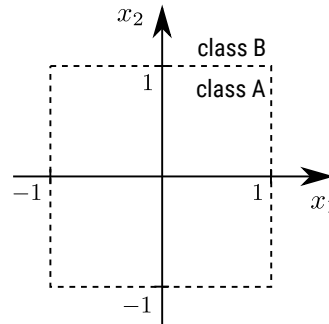


Exercise Sheet 6

Exercise 1: Designing a Neural Network (25 P)

We would like to implement a neural network that classifies data points in \mathbb{R}^2 according to decision boundary given in the figure below.



We consider as an elementary computation the *threshold neuron* whose relation between inputs $(a_i)_i$ and output a_j is given by

$$z_j = \sum_i a_i w_{ij} + b_j \quad a_j = 1_{z_j > 0}.$$

- (a) *Design* at hand a neural network that takes x_1 and x_2 as input and produces the output “1” if the input belongs to class A, and “0” if the input belongs to class B. *Draw* the neural network model and *write down* the weights w_{ij} and bias b_j of each neuron.

Exercise 2: Backward Propagation (5 + 20 P)

We consider a neural network that takes two inputs x_1 and x_2 and produces an output y based on the following set of computations:

$$z_3 = x_1 \cdot w_{13} + x_2 \cdot w_{23}$$

$$a_3 = \tanh(z_3)$$

$$z_4 = x_1 \cdot w_{14} + x_2 \cdot w_{24}$$

$$a_4 = \tanh(z_4)$$

$$z_5 = a_3 \cdot w_{35} + a_4 \cdot w_{45}$$

$$a_5 = \tanh(z_5)$$

$$z_6 = a_3 \cdot w_{36} + a_4 \cdot w_{46}$$

$$a_6 = \tanh(z_6)$$

$$y = a_5 + a_6$$

- (a) *Draw* the neural network graph associated to this set of computations.
- (b) *Write* the set of backward computations that leads to the evaluation of the partial derivative $\partial y / \partial w_{13}$. Your answer should avoid redundant computations. Hint: $\tanh'(t) = 1 - (\tanh(t))^2$.

Exercise 3: Programming (50 P)

Download the programming files on ISIS and follow the instructions.

Training a Neural Network

In this homework, our objective is to implement a simple neural network from scratch, in particular, error backpropagation and the gradient descent optimization procedure. We first import some useful libraries.

In [1]:

```
import numpy
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
na = numpy.newaxis
numpy.random.seed(0)
```

We consider a two-dimensional moon dataset on which to train the network. We also create a grid dataset which we will use to visualize the decision function in two dimensions. We denote our two inputs as x_1 and x_2 and use the suffix `d` and `g` to designate the actual dataset and the grid dataset.

In [2]:

```
# Create a moon dataset on which to train the neural network
import sklearn,sklearn.datasets
Xd,Td = sklearn.datasets.make_moons(n_samples=100)
Xd = Xd*2-1
Td = Td * 2 - 1
X1d = Xd[:,0]
X2d = Xd[:,1]

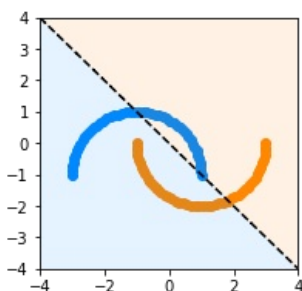
# Creates a grid dataset on which to inspect the decision function
l = numpy.linspace(-4,4,100)
X1g,X2g = numpy.meshgrid(l,l)
```

The moon dataset is plotted below along with some dummy decision function $x_1 + x_2 = 0$.

In [3]:

```
def plot(Yg,title=None):
    plt.figure(figsize=(3,3))
    plt.scatter(*Xd[Td==-1].T,color='#0088FF')
    plt.scatter(*Xd[Td==1].T,color='#FF8800')
    plt.contour(X1g,X2g,Yg,levels=[0],colors='black',linestyles='dashed')
    plt.contourf(X1g,X2g,Yg,levels=[-100,0,100],colors=['#0088FF','#FF8800'],alpha=0.1)
    if title is not None: plt.title(title)
    plt.show()

plot(X1g+X2g) # plot the dummy decision function
```



Part 1: Implementing Error Backpropagation (30 P)

$$\mathbf{i}_{j=1}^{25} : z_j = x_1 w_{1j} + x_2 w_{2j} + b_j$$

We would like to implement the neural network with the equations: $\mathbf{i}_{j=1}^{25} : a_j = \max$ where x_1, x_2 are the two input variables and y is the output of the network. The parameters of the neural network are initialized randomly using the normal distributions $w_{\{ij\}} \sim \mathcal{N}(\mu=0, \sigma^2=1/2)$, $b_{\{j\}} \sim \mathcal{N}(\mu=0, \sigma^2=1)$, $v_{\{j\}} \sim \mathcal{N}(\mu=0, \sigma^2=1/25)$. The following code initializes the parameters of the network and implements the forward pass defined above. The neural network is composed of 50 neurons.

In [4]:

```
import numpy

NH = 50

W = numpy.random.normal(0,1/2.0**.5,[2,NH])
B = numpy.random.normal(0,1,[NH])
V = numpy.random.normal(0,1/NH**.5,[NH])

def forward(X1,X2):
    X = numpy.array([X1.flatten(),X2.flatten()]).T # Convert meshgrid into dataset
    Z = X.dot(W)+B
    A = numpy.maximum(0,Z)
    Y = A.dot(V)
    return Y.reshape(X1.shape) # Reshape output into meshgrid
```

We now consider the task of training the neural network to classify the data. For this, we define the error function: $\mathcal{E}(\theta) = \sum_{k=1}^N \max(0, -y^{(k)} t^{(k)})$ where N is the number of data points, y is the output of the network and t is the label.

Task:

- Complete the function below so that it returns the gradient of the error w.r.t. the parameters of the model.

In [5]:

```
def backprop(X1,X2,T):
    X = numpy.array([X1.flatten(),X2.flatten()]).T

    # Compute activations
    Z = X.dot(W)+B
    A = numpy.maximum(0,Z)
    Y = A.dot(V)

    # Compute backward pass
    DY = (-Y*T>0)*(-T)
    DZ = numpy.outer(DY,V)*(Z>0)

    # Compute parameter gradients (averaged over the whole dataset)

    # -----
    # TODO: replace by your code
    # -----
    import solution
    DW,DB,DV = solution.gradients(X,Z,A,Y,DY,DZ)
    # -----

    return DW,DB,DV
```

Exercise 2: Training with Gradient Descent (20 P)

We would like to use error backpropagation to optimize the parameters of the neural network. The code below optimizes the network for 128 iterations and at some chosen iterations plots the decision function along with the current error.

Task:

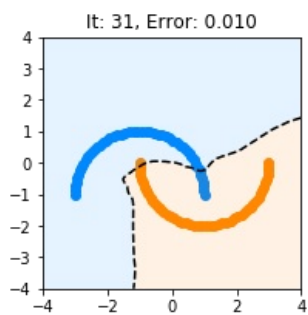
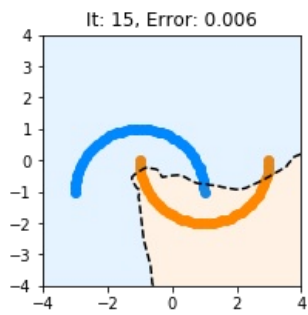
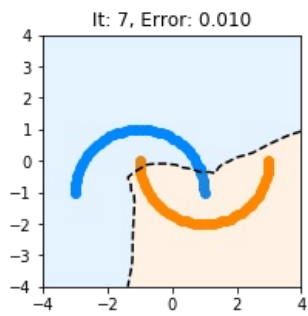
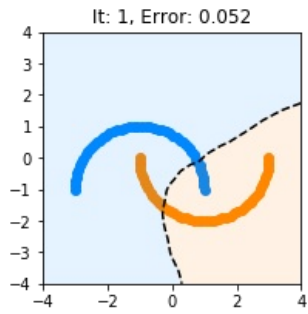
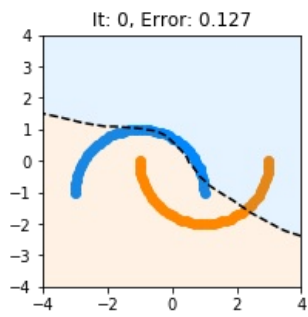
- Complete the procedure above to perform at each iteration a step along the gradient in the parameter space. A good choice of learning rate is $\eta=0.1$.

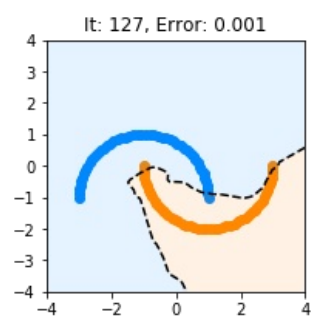
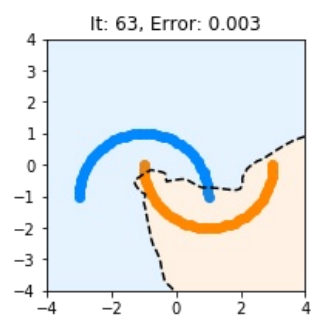
In [6]:

```
for i in range(128):

    if i in [0,1,3,7,15,31,63,127]:
        Yg = forward(X1g,X2g)
        Yd = forward(X1d,X2d)
        Ed = numpy.maximum(0,-Yd*Td).mean()
        plot(Yg,title="It: %d, Error: %.3f"%(i,Ed))

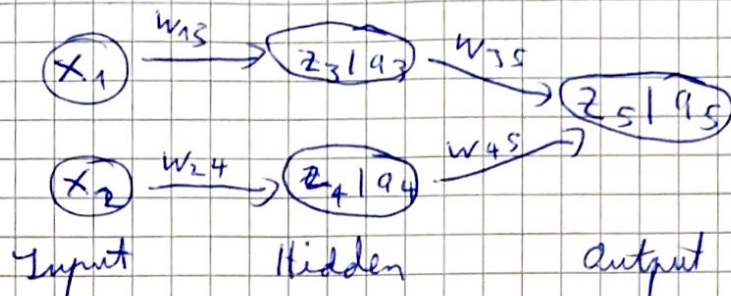
    # -----
    # TODO: replace by your code
    # -----
    import solution
    W,B,V = solution.descent(X1d,X2d,Td,W,B,V,backprop)
    # -----
```





Sheet 6

We design a neural network with three layers and three units:



We set $w_{13} = w_{24} = 1$, we set $w_{35} = w_{45} = 0,5$,
we set all bias terms $b_1 = b_2 = b_3 = b_4 = b_5 = 0$

and we define a threshold function as activation function
with
$$g(x) = \begin{cases} 1 & \text{if } |x| < 1 \\ 0 & \text{if } |x| \geq 1 \end{cases}$$

Therefore

$$a_3 = g(z_3) = g(w_{13} \cdot x_1 + b_1) = g(x_1) = \begin{cases} 1 & \text{if } |x_1| < 1 \\ 0 & \text{if } |x_1| \geq 1 \end{cases}$$

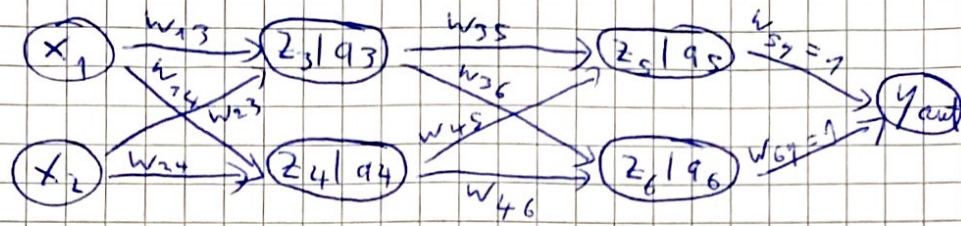
$$a_4 = g(z_4) = g(w_{24} \cdot x_2 + b_2) = g(x_2) = \begin{cases} 1 & \text{if } |x_2| < 1 \\ 0 & \text{if } |x_2| \geq 1 \end{cases}$$

$$\begin{aligned} a_5 &= g(z_5) = g(a_3 \cdot w_{35} + a_4 \cdot w_{45} + b_5) = \\ &= g(a_3 \cdot 0,5 + a_4 \cdot 0,5) = \begin{cases} 1 & \text{if } |x_1| \wedge |x_2| < 1 \\ 0 & \text{if } |x_1| \vee |x_2| \geq 1 \end{cases} \end{aligned}$$

and
 \wedge
 \vee
or

2

a)



~~Handwritten scribbles and crossed-out text.~~

$$\begin{aligned}
 b) \frac{\partial y}{\partial w_{13}} &= \frac{\partial y}{\partial a_5} \frac{\partial a_5}{\partial z_5} \frac{\partial z_5}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial w_{13}} \\
 &+ \frac{\partial y}{\partial a_6} \frac{\partial a_6}{\partial z_6} \frac{\partial z_6}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial w_{13}} \\
 &= \left(\frac{\partial y}{\partial a_5} \frac{\partial a_5}{\partial z_5} \frac{\partial z_5}{\partial a_3} + \frac{\partial y}{\partial a_6} \frac{\partial a_6}{\partial z_6} \frac{\partial z_6}{\partial a_3} \right) \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial w_{13}} \\
 &= \left[1 \cdot (1 - \tanh^2(z_5)) \cdot w_{35} + 1 \cdot (1 - \tanh^2(z_6)) \cdot w_{36} \right] \cdot (1 - \tanh^2(z_3)) \cdot x_1 \\
 &= \left[(1 - a_5^2) w_{35} + (1 - a_6^2) w_{36} \right] (1 - a_3^2) x_1
 \end{aligned}$$