Exercises L3: Restricted Boltzmann Machine

Exercise 1

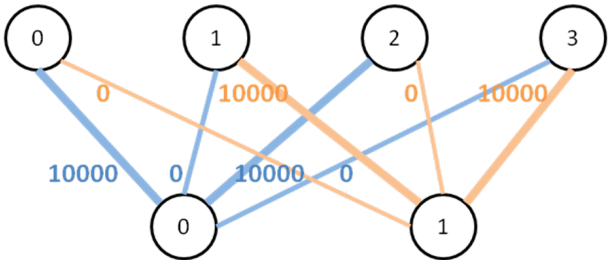
a) Write a function `sample_A_or_B(prob)` that returns `'A'` with probability `prob`, and `'B'` with probability `(1.0 - prob)`.
(Use `numpy.random.rand` to generate random numbers.)

b) Draw 1000 samples with `prob=0.3` and verify that you get approximately 300 `A`'s and 700 `B`'s

Exercise 2

Implement and test a function that samples from the hidden units of a RBM. Have a look at the files `sampling.py` and `test_sampling.py`.

a) Consider a RBM with 2 visible units and 4 hidden units. The bias terms are set to zero, and the weights are as in the figure below.



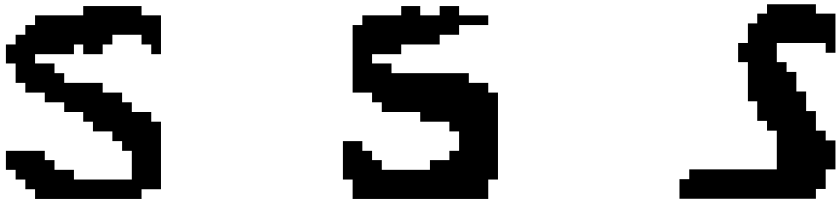
Compute the probability of each hidden unit begin 1, P(h_j = 1| v), for the following inputs:

v = [v0, v1]	[0, 0]	[1, 0]	[0, 1]	[1, 1]
h0				
h1				
h2				
h3				

b) Complete the test in `test_sampling.py` with the values computed in a)
c) Write the body of the function `sample_h` in `rbm.py` until the test passes

Exercise 3

The file `letter_S.npy` contains 39 handwritten examples of the letter "S". We want to use a RBM to learn a model of this letter. We will use the RBM implementation in the machine learning library MDP.



The module `letter.py` defines two functions:

- `load_data`: load the data from the file `letter_S.npy` .
  Each letter is a vector of length 320 (20x16 pixels)
- `show_letter`: display an activity pattern in a `matplotlib` figure

a) Load the data and use the function `show_letter` to display the handwritten letters

b) The object `mdp.nodes.RBMNode` is an implementation of a Restricted Boltzmann Machine algorithm. Read the documentation for the object: from the ipython prompt type

```
import mdp
mdp.nodes.RBMNode ?
mdp.nodes.RBMNode.train ?
mdp.nodes.RBMNode.sample_h ?
mdp.nodes.RBMNode.sample_v ?
```

c) Create a RBM with 100 hidden units:

```
rbm  =  mdp.nodes.RBMNode(100)
```

d) Train the network by presenting to it the data 100 times. Use `help(rbm.train)` to understand how to do that. You can keep the optional arguments at their default value or play around with them if you wish.

e) During training, the network has learned a model of the data. We can use it to generate new examples of the letter "S":

   a. start from a random input pattern:

```
v = np.random.randint(0, 2, size=(1, 320))
# this is the probability of v=1
pv = v
```

   b. sample from the hidden layer given the input pattern, and again the visible layer given the new hidden pattern:

```
ph, h = rbm.sample_h(pv)
pv, v = rbm.sample_v(ph)
show(v)
```

Do it a few times until the RBM begins sampling from the equilibrium distribution. Each pattern is an exemplar of what the model has learned.

c. plot the probability of the visible units being active instead:
```
show(pv)
```