Assignment 5

Maxime CHAMBREUIL
McGill ID: 260067572
maxime.chambreuil@mail.mcgill.ca

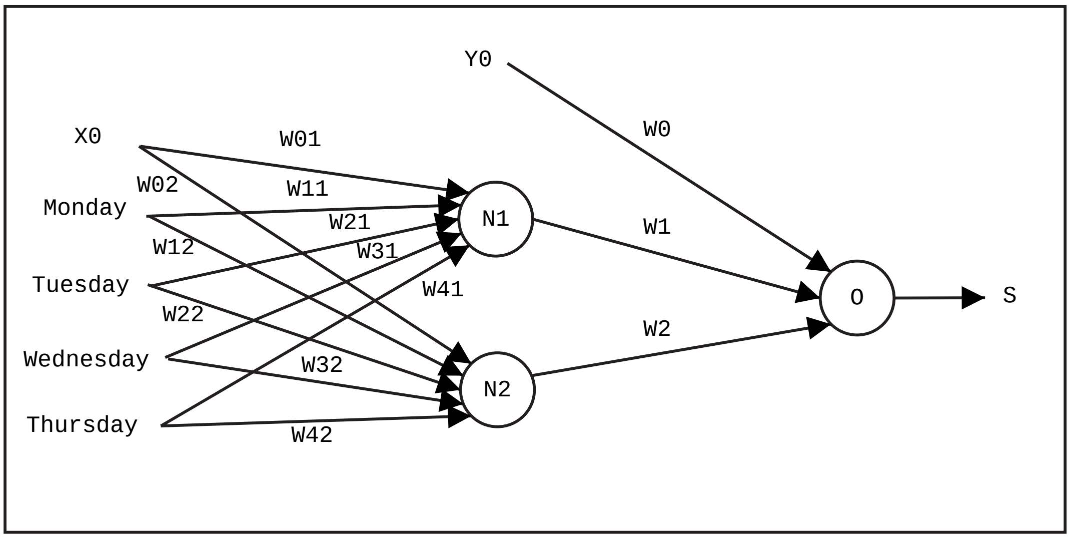
Contents

1 Building a simple neural net

1.1 Networks

1.1.1 With 2 neurons in the hidden layer

This is the annoted schema of the network :

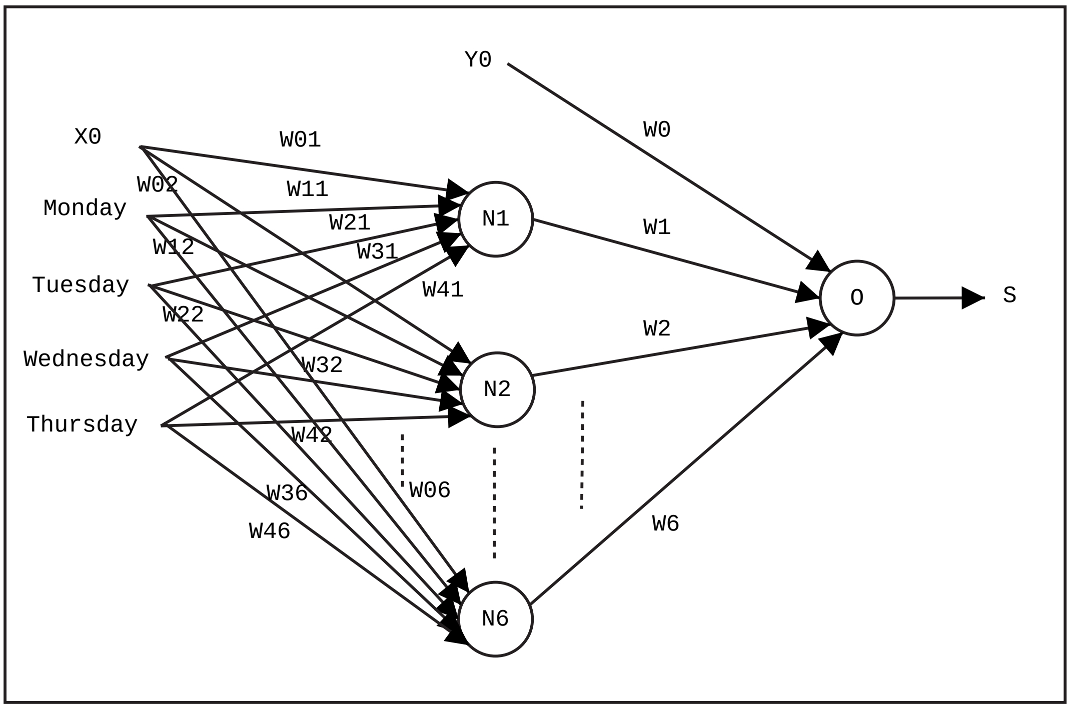


Set X_1 the boolean value of Monday, X_2 of Tuesday, X_3 of Wednesday and X_4 of Thursday. Set S_i the boolean output of the neuron N_i. We have:

$$S = W_0Y_0 + W_1S_1 + W_2S_2 = W_0Y_0 + W_1\left(\sum_{i=0}^{4} W_{i1}X_i\right) + W_2\left(\sum_{i=0}^{4} W_{i2}X_i\right)$$

1.1.2 With 6 neurons

This is the annoted schema of the network :



With 6 neurons, we have:

$$S = W_0 Y_0 + \sum_{j=1}^{6} W_j S_j = W_0 Y_0 + \sum_{j=1}^{6} W_j \left(\sum_{i=0}^{4} W_{ij} X_i \right)$$

1.2 Choice of numbers of neurons

My first answer would be the network with 2 neurons, because the output of the network is a boolean and we don't really need an accurate result: we will have a discrimination rule based on being over or under 0.5, to decide if the output is 0 or 1. But by experience, I would choos the 6-neurons network, because 2 neurons is not enough to get a reliable result.

2 Ex 20.19 p 762

The network will predict a 1 for this example, as it has been trained with more samples (80 % of the data) with a real value set to 1.

$$E = \sum_{i=1}^{100} (R_i - O_i)^2 = \sum_{i=1}^{80} (1 - O_i)^2 + \sum_{i=81}^{100} O_i^2$$

I wanted to differentiate the error function and set it zero as the hint said in the exercise, but we don't have any information of the network.

3 Gradient descent

The error function is:

$$E = \frac{1}{2} \left[(y - O_\theta)^2 + \lambda \sum_{j=1}^{m} O_j^2 \right]$$

3.1 Update rule for one unit

When we differentiate with θ_i, we obtained:

$$\begin{aligned}
\frac{\partial E}{\partial \theta_i} &= -(y - O_\theta) \frac{\partial O_\theta}{\partial \theta_i} + \lambda \theta_i = -(y - O_\theta) g'(in_\theta) \frac{\partial in_\theta}{\partial \theta_i} + \lambda \theta_i \\
\frac{\partial E}{\partial \theta_i} &= -(y - O_\theta) g'(in_\theta) \frac{\partial}{\partial \theta_i} \left(\sum_{j=1}^{m} a_j \theta_j \right) + \lambda \theta_i \\
\frac{\partial E}{\partial \theta_i} &= -(y - O_\theta) g'(in_\theta) a_i + \lambda \theta_i
\end{aligned}$$

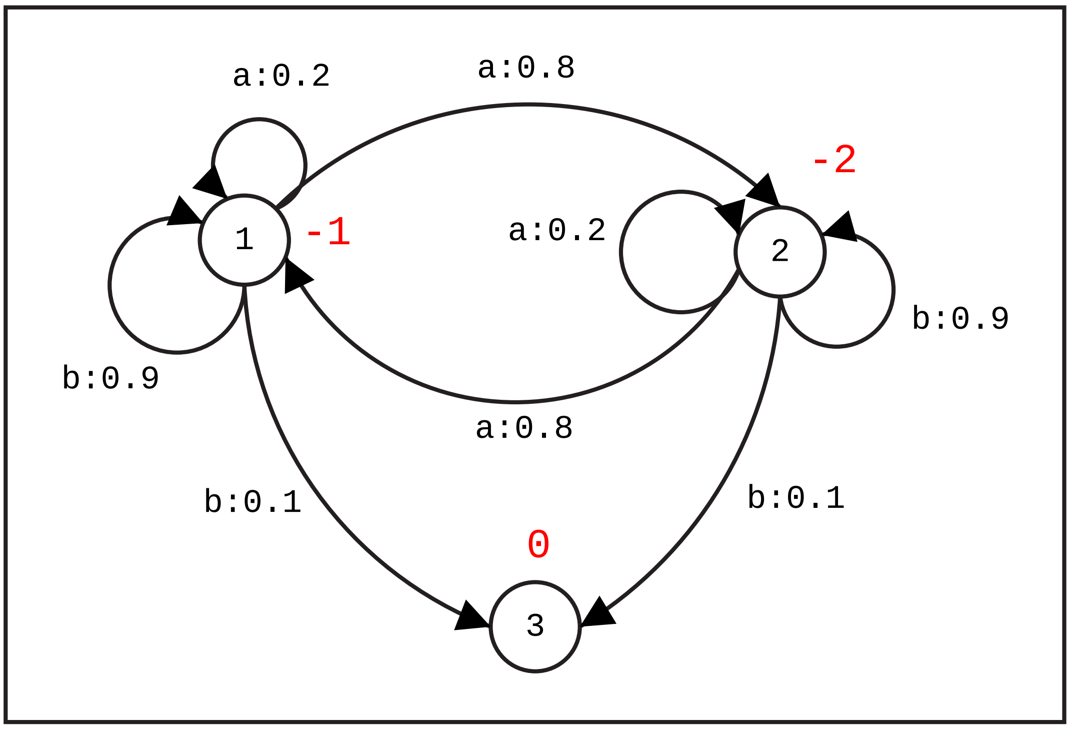
Originally, the update rule is:

$$W_j = W_j + \alpha \times Err \times g'(in) \times x_j$$

but I don't see how we can derive a new update rule from our result. I think the update rule is the same but adding the new term will intend some coefficient θ_i to be zero (cf My practical session of Neurals Networks on Models comparison and Regularisation, available on my website: http://www.maxime-chambreuil.fr.st/education/asi/asi.4.2/rna/).

3.2 Update rule for the network

4 Ex 17.4 p 646



4.1 Qualitatively determination

Qualitatively (without considering the probabilities), we can determine the choice of action: b, because of the reward. We should go as soon as possible to the state 3 and therefore lose less reward.

4.2 Action b

We know that:

$$U(S) = R(S) + \gamma \sum_{S'} T(S, a_i, S)U(S')$$

So we obtain with our exercise and $\gamma = 0,9$:

$$\left\{ \begin{array}{lcl} U(1) & = & -1 + \gamma \times 0,1 \times U(3) + \gamma \times 0,9 \times U(1) \\ U(2) & = & -2 + \gamma \times 0,1 \times U(3) + \gamma \times 0,9 \times U(2) \\ U(3) & = & 0 \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} U(1) & = & -5,263 \\ U(2) & = & -10,526 \\ U(3) & = & 0 \end{array} \right.$$

4.3 Action a

So we obtain:

$$\left\{ \begin{array}{lcl} U(1) & = & -1 + \gamma \times 0,2 \times U(1) + \gamma \times 0,8 \times U(2) \\ U(2) & = & -2 + \gamma \times 0,2 \times U(2) + \gamma \times 0,8 \times U(1) \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} U(1) & = & -2,614 \\ U(2) & = & -1,588 \end{array} \right.$$

The optimal policy would be to choose action a, and as the discount factor is applied with the same value for each action, the optimal policy does not depend on it.

5 Ex 17.5 p 647

5.1 Bellman Equation

The original Bellman Equation is:

$$\forall S, U(S) = R(S) + \gamma max \left(\sum_{S'} T(S', a, S) U(S') \right)$$

For R (S , a), we have to consider all actions, that have been taken to lead to S:

$$\forall S, U(S) = \left(\sum_{i} R(S, a_i) \right) + \gamma max \left(\sum_{S'} T(S', a, S) U(S') \right)$$

For R (S , a , S′), we have to consider all actions, that have been taken to lead to S, and all states S′ we can reach from S:

$$\forall S, U(S) = \left[\sum_{S'} \left(\sum_{i} R(S, a_i, S') \right) \right] + \gamma max \left(\sum_{S'} T(S', a, S) U(S') \right)$$

5.2 MDP Transformation

This means that considering the state S′ in the reward of S should not affect the policy. S′ does not affect the reward if the derivation of R (S , a , S′) on S′ is null. S′ would be a constant in R for all S and would not affect the policy.

6 Understanding reward structures

It should not affect the task, because the action is chosen to maximize the rewrd and if the probabilities do not change, the decision rule would be applied with the same manner.

If we use the Bellman Equation, we originally have:

$$\forall S, U(S) = R(S) + \gamma max \left(\sum_{S'} T(S', a, S) U(S') \right)$$

If we add a constant C to R (S), we obtain:

$$\forall S, U(S) = (R(S) + C) + \gamma max \left(\sum_{S'} T(S', a, S) U(S') \right) = U(S) + C$$

So I would conclude that K = C.