TU München
Fakultät für Informatik
PD Dr. Rudolph Triebel
J. Chiotellis, C. Hazirbas, P. Haeusser

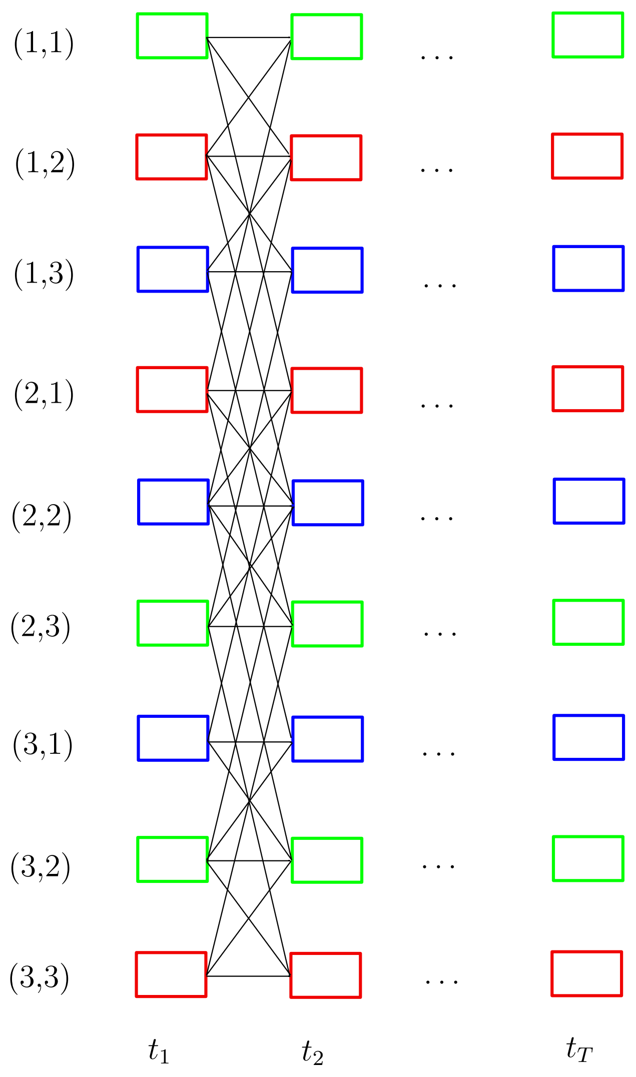
Machine Learning for Robotics and Computer Vision
Winter term 2015

Solution Sheet 3

Topic: Deep Learning
November 23th, 2015

Exercise 1: Viterbi algorithm

a) The state is the position of the robot. We have a discrete state space of 9 squares. Each state is a pair (x,y), so $x_i \in \{(1,1),(1,2)\ldots,(3,3)\}$.
The observation space is also discrete and it consists of the 3 colors the robot may observe, so $z_i \in \{R,G,B\}$. The trellis diagram would look like this:



b) The robot can only move vertically or horizontally, so there are four possible moves (up, down, left, right). Since the robot moves randomly, each of these has probability $p_{move} = 0.25$. For all states except the one in the central square, there are moves that lead out of the bounds of the room. Then the robot stays at its current position, so the probability for that move is assigned to the transition to the self-state. The transition matrix looks as follows:

x_i \ x_i	**(1,1)**	**(1,2)**	**(1,3)**	**(2,1)**	**(2,2)**	**(2,3)**	**(3,1)**	**(3,2)**	**(3,3)**
(1,1)	0.50	0.25	0.00	0.25	0.00	0.00	0.00	0.00	0.00
(1,2)	0.25	0.25	0.25	0.00	0.25	0.00	0.00	0.00	0.00
(1,3)	0.00	0.25	0.50	0.00	0.00	0.25	0.00	0.00	0.00
(2,1)	0.25	0.00	0.00	0.25	0.25	0.00	0.25	0.00	0.00
(2,2)	0.00	0.25	0.00	0.25	0.00	0.25	0.00	0.25	0.00
(2,3)	0.00	0.00	0.25	0.00	0.25	0.25	0.00	0.00	0.25
(3,1)	0.00	0.00	0.00	0.25	0.00	0.00	0.50	0.25	0.00
(3,2)	0.00	0.00	0.00	0.00	0.25	0.00	0.25	0.25	0.25
(3,3)	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.25	0.50

c) We want to use the Viterbi algorithm to estimate the most likely sequence of squares the robot followed. To do that we need to compute the transition matrix A (previous question), the initial state probabilities π_i and the observation model $p(z_i|x_i)$. The robot's initial position is unknown, therefore we have $\pi_i = \frac{1}{9} \quad \forall i \in \{1, \ldots, 9\}$.
The robot's observation model is almost given by the table. We just have to replace the actual color with each state the robot can be at. Then we obtain $p(z_i|x_i)$:

z_i \ x_i	**(1,1)**	**(1,2)**	**(1,3)**	**(2,1)**	**(2,2)**	**(2,3)**	**(3,1)**	**(3,2)**	**(3,3)**
R	0.1	0.8	0.1	0.8	0.1	0.1	0.1	0.1	0.8
G	0.6	0.1	0.2	0.1	0.2	0.6	0.2	0.6	0.1
B	0.3	0.1	0.7	0.1	0.7	0.3	0.7	0.3	0.1

We initialize with $\delta(x_0) = p(x_0)p(z_0|x_0)$. We know $p(x_0) = \pi$ and $z_0 = R$, so we have $\delta(x_0)$:

z_0 \ x_0	**(1,1)**	**(1,2)**	**(1,3)**	**(2,1)**	**(2,2)**	**(2,3)**	**(3,1)**	**(3,2)**	**(3,3)**
R	0.0111	0.0889	0.0111	0.0889	0.0111	0.0111	0.0111	0.0111	0.0889

Now we look at the second observation: $z_1 = G$. For each state we compute $\delta(x_1)$:

$$\delta(x_1) = p(z_1|x_1) \max_{x_0}\{\delta(x_0)p(x_1|x_0)\}$$

z_1 \ x_1	**(1,1)**	**(1,2)**	**(1,3)**	**(2,1)**	**(2,2)**	**(2,3)**	**(3,1)**	**(3,2)**	**(3,3)**
G	0.0133	0.0022	0.0044	0.0022	0.0044	0.0133	0.0044	0.0133	0.0044

And for the last observation $\delta(x_2)$:

z_2 \ x_2	**(1,1)**	**(1,2)**	**(1,3)**	**(2,1)**	**(2,2)**	**(2,3)**	**(3,1)**	**(3,2)**	**(3,3)**
G	0.0040	0.0003	0.0007	0.0003	0.0007	0.0020	0.0007	0.0020	0.0003

In this last step we see that state $x^* = (1,1)$ is the most probable final state. By backtracking we see that in the first time step there are three equally probable states: (1,2), (2,1) and (3,3). All of them make sense as they are red squares (first observation) and we don't have any other information.
In the second step there are again three equally probable states: (1,1), (2,3) and (3,2). The paths that lead to these states are:

(1,2) , (1,1)
(2,1) , (1,1)
(3,3) , (2,3)
(3,3) , (3,2)

In the last step state (1,1) is most probable because there are 2 paths that can lead to it. In contrast there is one path that leads to (2,3) or (3,2) so the probability for each is exactly half of the probability for (1,1). Therefore the most likely path is

(1,2) , (1,1), (1,1) or (2,1) , (1,1), (1,1)

d) See code

Exercise 2: K-Means and EM

See code

Exercise 3: Back Propagation

- Derivative of the activation function $f(\cdot)$ at the corresponding neuron should be derived. During back propagation, the error should be back-propagated with f' for the corresponding neuron.

- $$\begin{aligned} \frac{\partial C}{\partial w_5} &= \frac{\partial C}{\partial a_1^2} \cdot \frac{\partial a_1^2}{\partial z_1^2} \cdot \frac{\partial z_1^2}{\partial w_5} \\ \frac{\partial C}{\partial w_5} &= -2(t_1 - a_1^2) \cdot \sigma'(z_1^2) \cdot w_5 \\ \frac{\partial C}{\partial w_5} &= -2(t_1 - a_1^2) \cdot \sigma'\Big(w_5 a_1^1 + w_7 a_2^1 + b_1\Big) \cdot w_5 \end{aligned} \tag{1}$$

Exercise 4: Convolutional Layer Arithmetic

In general, blobs have the canonical shape $N \times C \times H \times W$.

a) The input blob will have the shape $10 \times 3 \times 80 \times 120$.

b) In order to find the output blob shape, one has to consider what happens during a convolution. If there is a non-zero padding, zeros will get virtually added at the boundary of the input. So the input size increases by $2 \times p$ where p is the padding size. Every time a kernel gets multiplied with pixels, one number for the activation map is computed. Thus, the number of times that you can "apply" a kernel along the height/width of the input will give you the dimensions of the output. Putting all this together will give you the following formula:

$$\tilde{x} = \frac{x + 2 \times p - k}{s} + 1$$

where $\tilde{x}$ is the output height or width, respectively. k is the kernel height or width and s is the stride. That means that the output blob will have the shape
$10 \times 64 \times 40 \times 62$