Labs
Machine Learning Course
Fall 2016

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
mlo.epfl.ch/page-136795.html
epfmlcourse@gmail.com

Problem Set 14, Dec 22th, 2016 (Various)

Goals. The goal of this exercise is to provide you with some further theory questions so that you can prepare better for your final exam. We might add some questions later on. So you might want to always check the latest version.

Problem 1 (Markov blanket):

In the course notes we introduced the concept of a *Markov blanket*.

Let X_i be a node in a given Bayes net and let Z be its Markov blanket. Let X_j be any other node not equal to X_i or contained in Z. Show that X_i is independent of X_j given Z by showing that X_j is D-separated from X_i by Z.

Problem 2 (Bayes Net):

Consider the example shown in Figure 1.

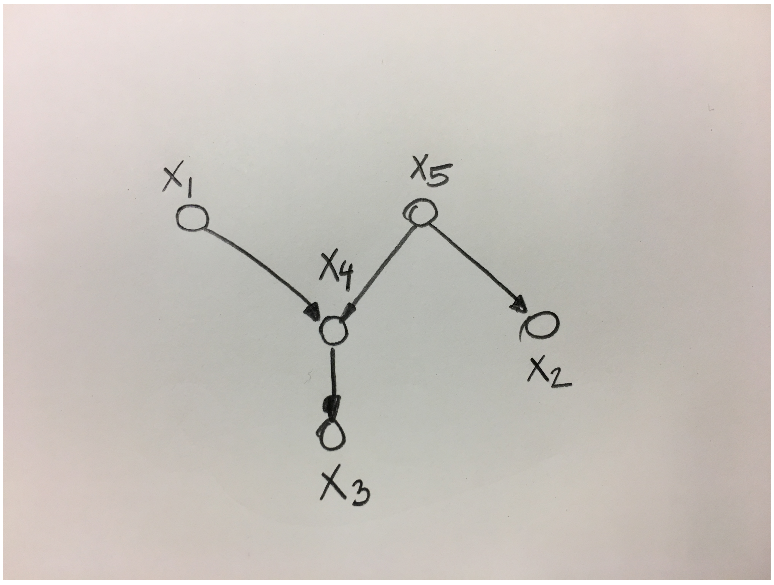

Figure 1: Bayes Net.

We will not work the answers here, but it might be a good additional exercise to work out the answer yourself.

- Is X_1 independent of X_2 given X_3?
- Is X_1 independent of X_2 given X_5?

Problem 3 (Finite differences (from ETHZ course by Hofmann)):

During lab 13 you implemented the back-propagation algorithm. It is easy to get some indices wrong and so it is important to have a good way of checking the derivatives.

1. One common way to check that the computation of a derivative is correct is to use finite differences. Considering only the i-th dimension of the parameter vector $\boldsymbol{w}$, show that finite difference yields an error

$O(\epsilon)$, i.e.

$$\nabla f(w_i) = \frac{f(w_i + \epsilon) - f(w_i)}{\epsilon} + O(\epsilon), \tag{1}$$

where $\epsilon \in \mathbb{R}^+ \leq 1$.

2. The accuracy of the finite difference method can be improved significantly by using symmetrical central differences.

$$\nabla f(w_i) \approx \frac{f(w_i + \epsilon) - f(w_i - \epsilon)}{2\epsilon} \tag{2}$$

What approximation error do we get using Eq. 2?

Problem 4 (Weight initialization (from ETHZ course by Hofmann)):

How shall we initialize the weights in a neural network. One point to consider is that we would like the nodes to work in a proper range. In more detail, assume that we have set the weights initially in such a way that the expected input to a node is around 100 and we use the $\tanh$ as an activation function. Then we will be operating in a regime where the activation function is very flat and it might need a lot of training to change the weights so that the expected input is closer to the "interesting" range $[-1, 1]$ where this activation function.

Suppose we have an input $X = (x_1, \ldots x_n)$ where $x_i \in \mathbb{R}^d$ and a linear neuron with random weights W that outputs a number Y as:

$$Y = W_1 x_1 + W_2 x_2 + \cdots + W_n x_n$$

We assume that x_i and W_i are all independent and identically distributed.

1. Show that Var$[W_i x_i] = \mathbb{E}[x_i]^2Var[W_i] + \mathbb{E}[W_i]^2Var[x_i]$ + Var$[W_i]$Var$[x_i]$.
2. Assuming that inputs and weights both have mean 0, show that Var$[Y] = n$Var$[W_i]$Var$[x_i]$.
3. Now assume that we want the variance of the output to be the same as the variance of the input. What is the condition for Var$[W_i]$.