Neural Networks: III. Regularization (Part 6)

Jan Bauer

jan.bauer@dhbw-mannheim.de

04.06.19

Interpret the Outcome

What to change to get better performance?

Accuracy & Loss

Motivation

- improve generalization (and performance) by building several NN
- take the average of the outputs as our best guess
- problem: Machine intense
- ...but what if we already have several NN?

Motivation

Consider our NN consits of M nodes. Then there exist 2^M sampled networks.

Example:

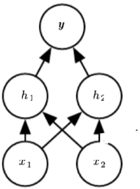

Figure: Two Layer Network with Four Nodes (Deep Learning - Goodfellow, Bengio & Courville)

Motivation

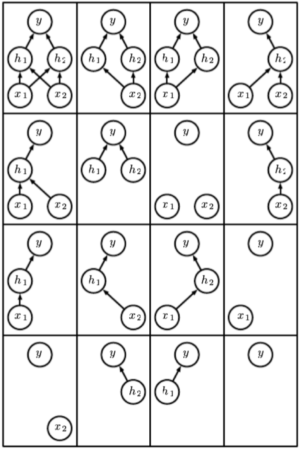

Figure: Possible Sampled Networks (Deep Learning - Goodfellow, Bengio & Courville)

Dropout

- drop out units
  - temporarily removing it from the network
  - randomly choose which units to drop
- combining exponentially many different neural network architectures
- increases generalization i.e. prevents overfitting

Dropout

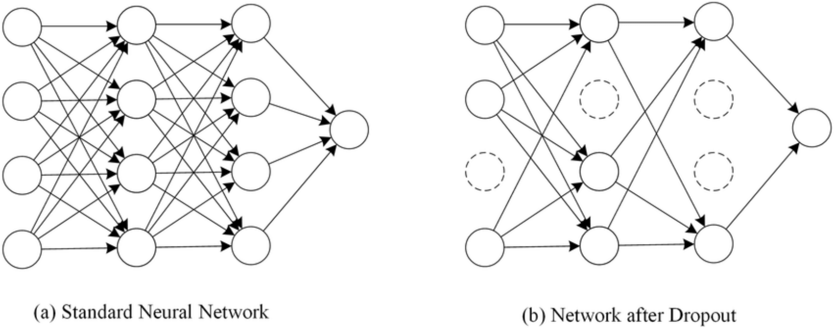

Figure: Before and after using Dropout (Dropout: A Simple Way to Prevent Neural Networks from Overfitting (2014))

Dropout

How to?

- retain each unit with a probability p independent of other units

- drop each unit with a probability $1 - p$ independent of other units

- you can choose the value of p
  - to be 0.5 (which works quite well for most of the cases), while the probability for the input layer should be close to 1
  - according to your validation results (i.e. during training)

- increases generalization i.e. prevents overfitting

Dropout

In practice:

- use dropout during training

- at test time, use the NN without dropout

- note: The weights of the NN without dropout are scaled-down versions of the trained weights:

$$\mathrm{E}(\text{neuron}) = p \cdot y + (1 - p) \cdot 0 = p \cdot y$$

- Hence, we scale the NN without dropout by p to have the same weights:

$$p \cdot \mathrm{E}(\text{neuron}) = p \cdot y$$