

IV. Deep Neural Networks

I) Algoritmi di Training efficienti

- I.1) Implementazione di backpropagation (Mathematica)
- I.2) DNNs con TensorFlow/Keras
- I.3) Data pre-processing

1.1)

Implementing backpropagation

$$1) \quad a_i^{(1)} = x_i$$

$$2) \quad z_i^{(l)} = \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)} \quad a_i^{(l)} = \sigma(z_i^{(l)})$$

$$3) \quad \Delta_i^{(L)} = \frac{\partial E}{\partial a_i^{(L)}} \sigma'(z_i^{(L)})$$

$$4) \quad \Delta_i^{(l)} = \sigma'(z_i^{(l)}) \sum_{j=1}^{n_{l+1}} \Delta_j^{(l+1)} w_{ji}^{(l+1)}$$

$$5) \quad \frac{\partial E}{\partial b_i^{(l)}} = \Delta_i^{(l)} \quad \frac{\partial E}{\partial w_{ij}^{(l)}} = \Delta_i^{(l)} a_j^{(l-1)}$$

1.2)

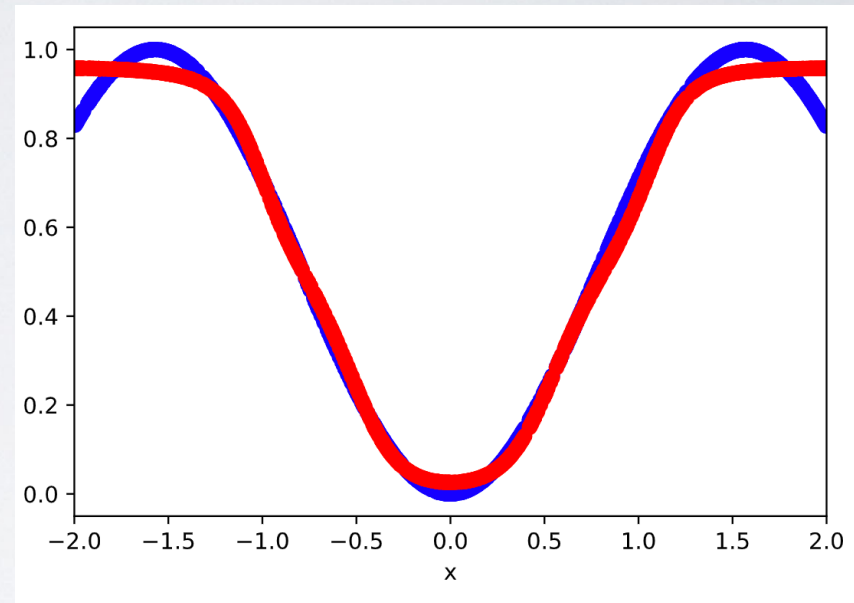
DNNs with Tensorflow/Kera

```
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import activations

from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import Adam
# for 2nd attempt
from keras.initializers import glorot_uniform
from keras.callbacks import EarlyStopping

import keras.backend as K
```



see example notebook `NNbasic.ipynb`

1.2)

Efficient Training Methods (Regularisation)

Parameter initialisation e.g. Glorot-Bengio initialisation

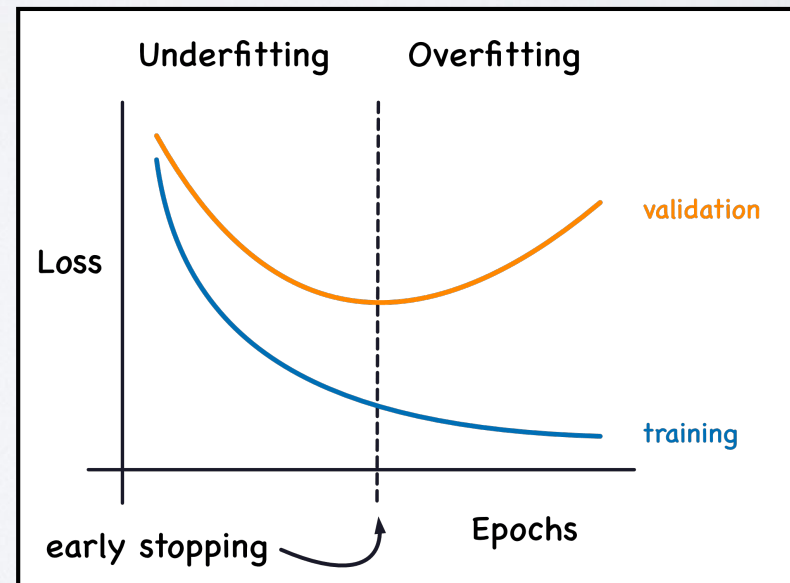
We initialized the biases to be 0 and the weights W_{ij} at each layer with the following commonly used heuristic:

$$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right], \quad (1)$$

where $U[-a, a]$ is the uniform distribution in the interval $(-a, a)$ and n is the size of the previous layer (the number of columns of W).

<http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

Cross validation of loss function on testing and training sets: **Early stopping**

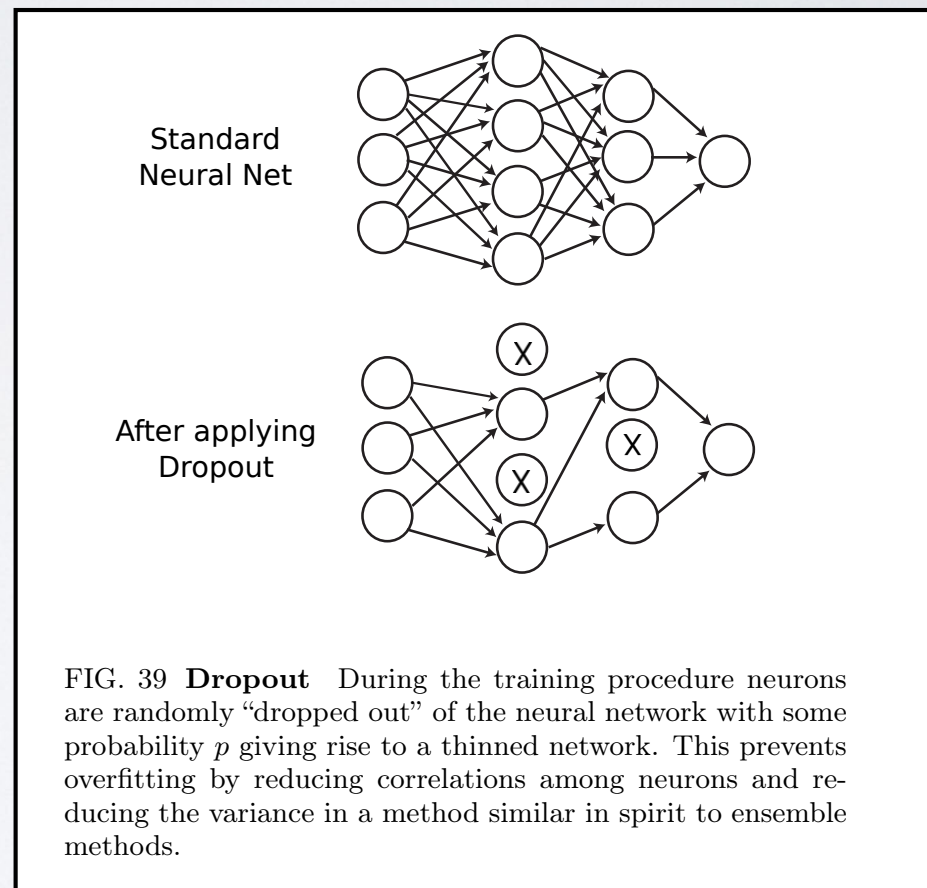


1.2)

Efficient Training Methods (Regularisation)

Dropout

see <https://arxiv.org/pdf/1803.08823.pdf>



1.2)

Efficient Training Methods (Regularisation)

mini-batch: update model parameters
using small subsets of the data

controlled split of test training datasets:
e.g. `sklearn.model_selection.train_test_split`

Hyperparameter tuning: learning rate, layers and depths, ...
see KerasTuner: https://keras.io/keras_tuner/

1.3)

Data pre-processing

So far we have made sure to choose functions that lie between 0 and 1 so the sigmoid (logistic) activation function was appropriate

In general we should organise our data to make sure the choice of network architecture is appropriate

$$x'(x, \langle x \rangle, \sigma_x) = \frac{x - \langle x \rangle}{\sigma_x} \quad \text{'standardise'}$$

$$f'(f(x), f_{\min}, f_{\max}) = \frac{f(x) - f_{\min}}{f_{\max} - f_{\min}} \quad \text{'normalise'}$$

After calling the trained network, de-standardisation/de-normalisation should be applied e.g.

$$f(x) = f'(x') \times (f_{\max} - f_{\min}) + f_{\min}$$