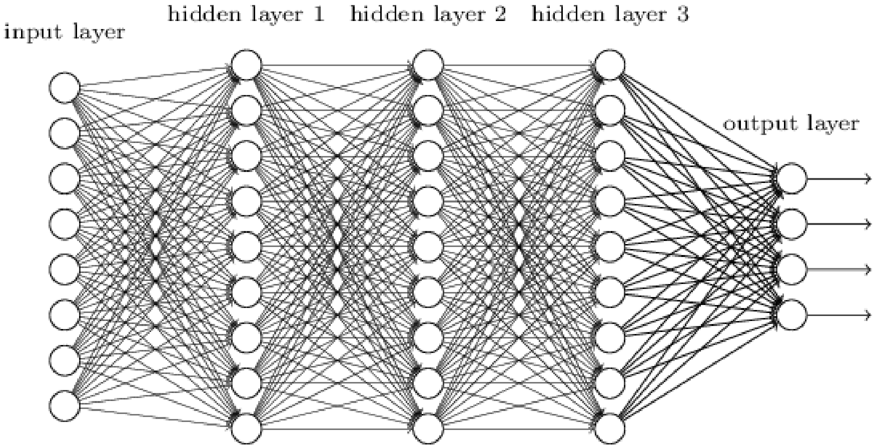
Improving Neural Networks, Overfitting and Regularization techniques

Michael Nielsen's Deep Learning book

Winter School on Quantitative Biology Learning and Artificial Intelligence

Alessio Ansuini
ICTP - November 2018

Where we are : Feed-forward NN



Where we are : Backpropagation Algorithm

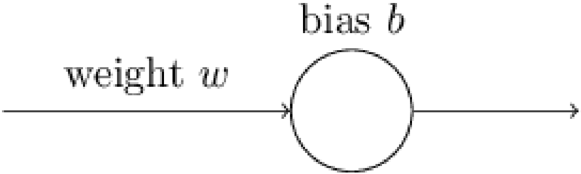
Quadratic loss $C = \frac{(y-a)^2}{2}$
Backpropagation : $\nabla_w C \quad \nabla_w C$
Learning rules : $w \leftarrow w - \eta \nabla_w C$

More appropriate loss function C' (faster convergence) and techniques to reduce overfitting:

- Regularization
- Data augmentation
- Dropout

A toy model

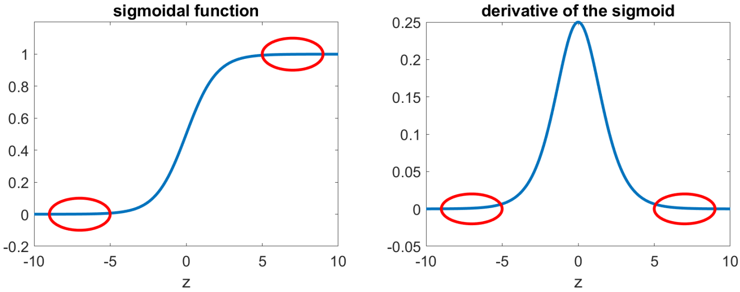
$$x = 1 \to y = 0$$



$$z = wx + b \qquad a = \sigma(z) = \sigma(wx + b)$$

A toy model

$$C = \frac{1}{2}(y-a)^2$$
$$\frac{\partial C}{\partial w} \sim \sigma'$$
$$\frac{\partial C}{\partial b} \sim \sigma'$$



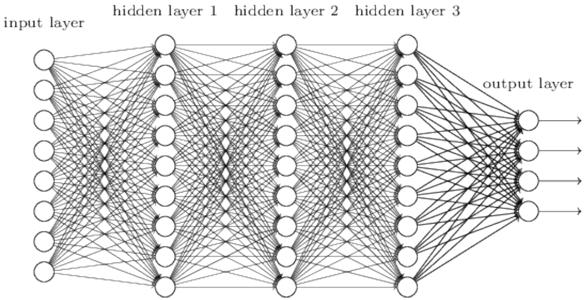
Cross-entropy loss

$$C = -\left[\; y \ln a + (1-y)\ln(1-a) \;\right]$$

$$\begin{aligned}
y = 0 \quad a = 0.1 &\rightarrow C = 0.1054 \\
y = 0 \quad a = 0.9 &\rightarrow C = \color{red}{2.3026} \\
y = 1 \quad a = 0.9 &\rightarrow C = 0.1054 \\
y = 1 \quad a = 0.001 &\rightarrow C = \color{red}{6.9078}
\end{aligned}$$

Measures the *suprise* to observe the true value once we make our prediction

General cross-entropy formula



$$C = -\frac{1}{n}\sum_{x}\sum_{j}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] \quad j = 0, 1, \ldots 9$$

Softmax Layer

Output interpretable as a probability distribution on categories (conditioned to the data) : $a_j^L = p(y = j|x)$

$$z_j^L = \sum_k w_{jk}^L a_k^{L-1} + b_j^L \quad \rightarrow \quad a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$$

$$a_j^L \geq 0 \quad \sum_j a_j^L = 1$$

- Look at the interactive sliders in Nielsen's book
- It can be shown that softmax output in combination with log-likelihood loss $C = -\ln a_y^L$ is $\sim$ to sigmoidal output in combination with cross-entropy loss.

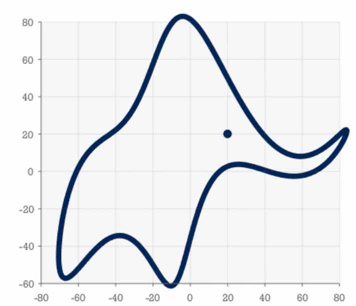
Overfitting

"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk"

John von Neumann

The Elephant

"Jens Morten Hansen (JMH) and co-authors have recently published a study where they use sine-regression to fit 5 oscillations plus a linear trend to a 160-year sea level [...] This result is clearly not significant in any meaningful sense of the word."



The animation

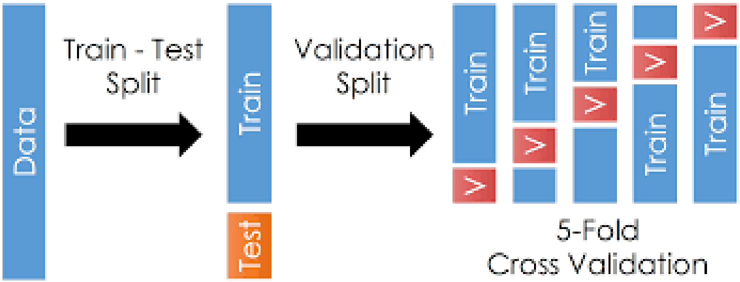
Number of trainable parameters in Neural Networks

- Fully connected $[784, 30, 10]$: $\sim 24K$
- Fully connected $[784, 100, 10]$: $\sim 80K$
- LeNet (1998, convolutions, 8 layers) : $\sim 60K$
- AlexNet (2012) : $\sim 60M$
- VGG-16 (2014) : $\sim 138M$
- $\dots$

The overfitting problem in neural networks is (potentially) very serious. But before addressing networks $\dots$

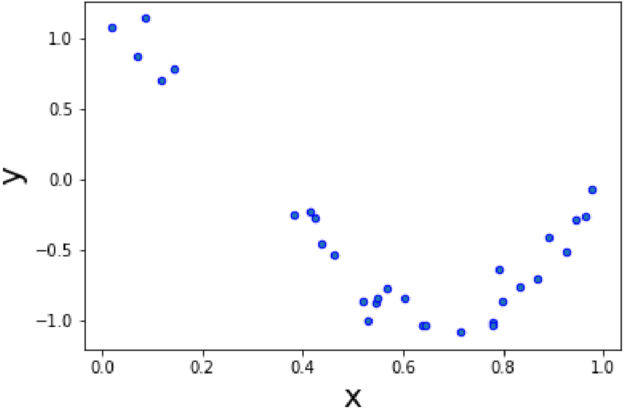
How to split the data

General approach in machine learning (clustering, regression, networks . . .)



Polynomial approximation as a regression problem : what is the best degree (hyperparameter) of the interpolating polinomial ?

Example : $y = cos(x) + \varepsilon$



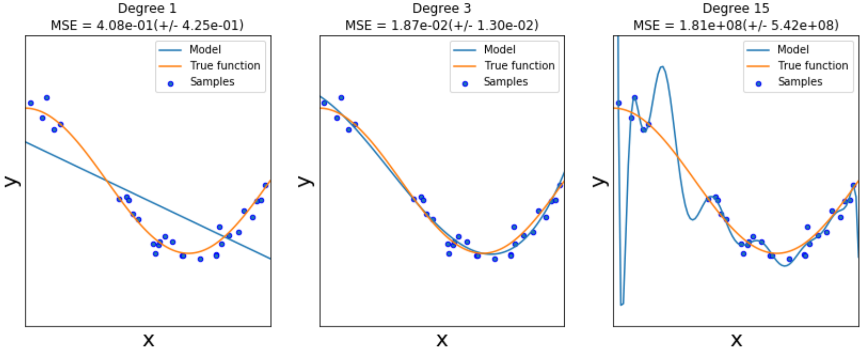
Models : polynomials of growing degree

Number of parameters is degree + 1

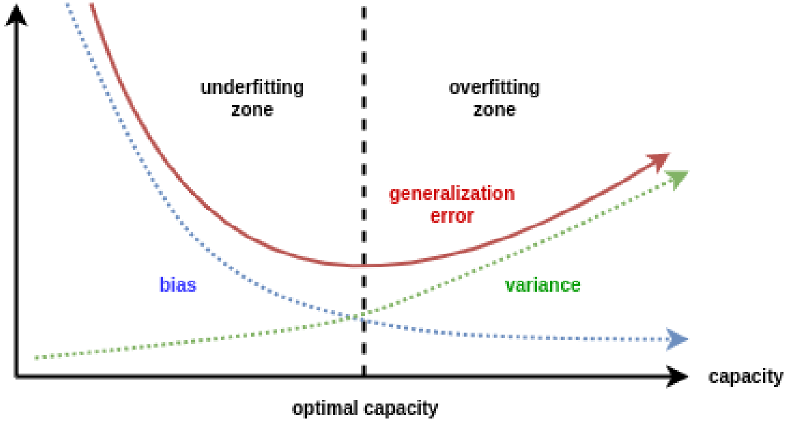
- $y = \theta_0 + \theta_1 x$
- $\dots$
- $y = \theta_0 + \theta_1 x + \cdots + \cdots + \theta_{15} x^{15}$

Look for the best model (the one with lowest MSE)
We will use (10-fold) cross-validation

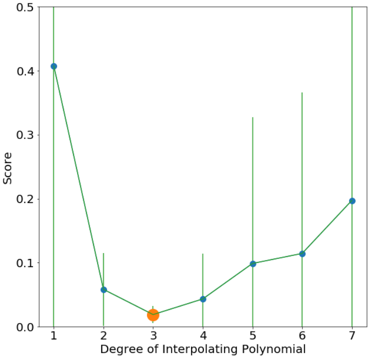
Which one is the best ?



Bias-Variance tradeoff



Hyper-parameters search



The best degree is 3

Find out more here : scikit-learn

Regularization

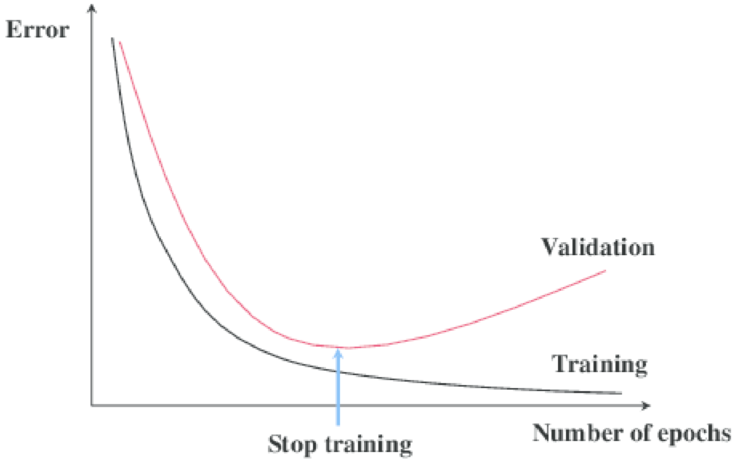
The general approach with Neural Networks

- Choose a model complex enough to ensure overfitting
- Regularize to reduce its generalization error

Regularization techniques

- Early stopping $\rightarrow$ optimal number of epochs
- Weight decay
- Data augmentation (rotations, translations, etc.)
- Dropout
- ...

Early stopping



Regularization by weight decay

Reduce *effective* model complexity[1]

$$C = C_0 + \frac{\lambda}{2n}||w||_2^2 \qquad ||w||_2^2 = \sum_w w^2$$

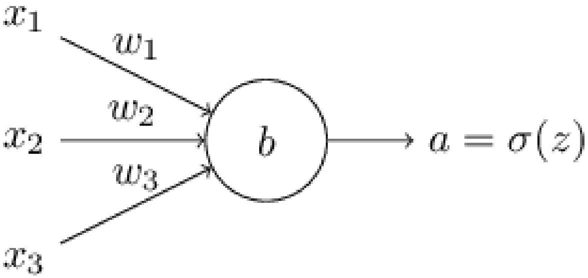
New learning rule

$$\begin{aligned} w &\rightarrow w - \eta\frac{\partial C_0}{\partial w} - \frac{\eta\lambda}{n}w \\ &= \left(1 - \frac{\eta\lambda}{n}\right)w - \eta\frac{\partial C_0}{\partial w}. \end{aligned}$$

Exercise !

[1] Not applied on the biases

Why regularization should help ?



L1 regularization

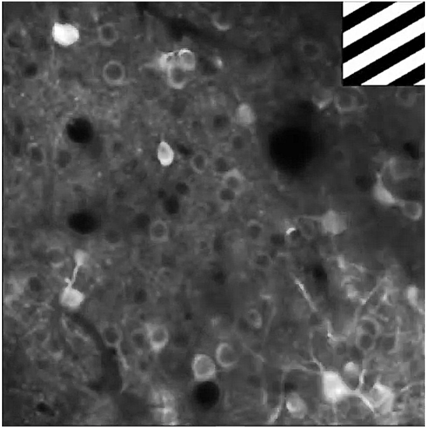
$$C = C_0 + \frac{\lambda}{n}||w||_1 \qquad ||w||_1 = \sum_w |w|$$

New learning rule

$$w \quad \rightarrow \quad w - \eta \frac{\partial C_0}{\partial w} - \ldots$$

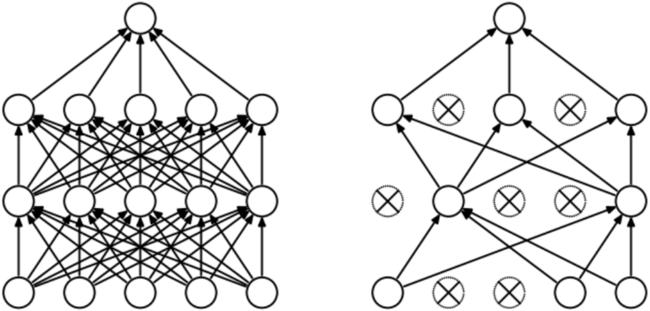
Exercise !

Calcium imaging visual cortex (V1) of the mouse (Mriganka Sur Lab, MIT)



Dropout

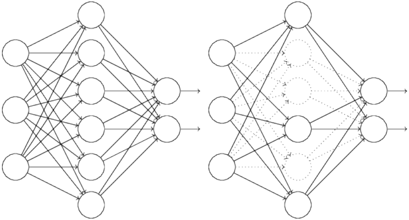
Does not modify C but the network's way to operate



(a) Standard Neural Net (b) After applying dropout.

Dropout

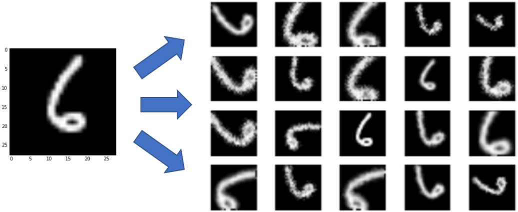
In each mini-batch we randomly deactivate a fraction p of units



$$p = 0.5 \quad w \to pw = \frac{1}{2}w$$

"[...] imagine that we are training different neural networks [...] the different networks will overfit in different ways"
"[...] reduces complex co-adaptations of neurons [...]"

Data augmentation



Data augmentation on MNIST

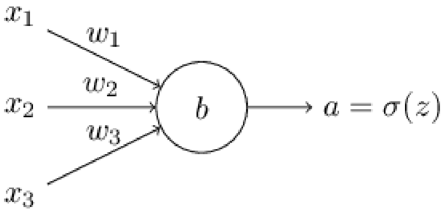
- small rotations
- small translations
- elastic distortions (emulate the random oscillations found in hand muscles)
- other ...

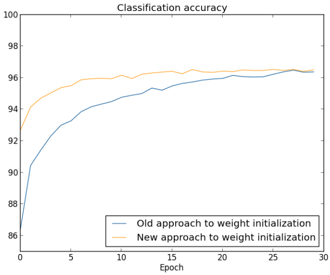
More data means more chance for the network to *make experience* of the natural variability in the data. The data augmentation strategy should reflect this variability (simple example - with implementation - in the Lecture on convnets).

Exercise ! We will give progressively more training data to the network and plot its performance after 10 epochs.

Weights initialization

$$\mathcal{N}(0,1) \quad \text{(large weights)} \rightarrow \mathcal{N}\left(0, \frac{1}{\sqrt{\text{n.of inputs}}}\right) \quad \text{(improved weights)}$$





Bibliography

- "Pattern Recognition and Machine Learning", C. Bishop, 2006
- "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Hastie, Tibshirani, Friedman, (2nd ed. 2009) download link
- "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis", Simard, Steinkraus and Platt, (2003)
- "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Srivastava et al., (2014)
- "Improving neural networks by preventing co-adaptation of feature detectors", Hinton et al. (2012).

Sitography

- http://neuralnetworksanddeeplearning.com/
- https://www.deeplearningbook.org/
- http://cs231n.stanford.edu/
- https://scikit-learn.org/stable/