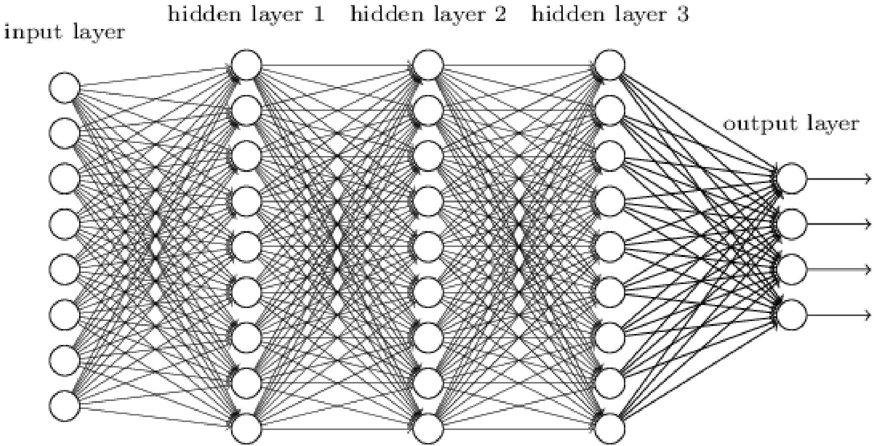
Improving Neural Networks, Overfitting and Regularization techniques

Michael Nielsen's Deep Learning book

Winter School on Quantitative Biology Learning and Artificial Intelligence

Alessio Ansuini
ICTP - November 2018

Where we are



input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

Where we are

The *Basic Swing* : Backpropagation Algorithm

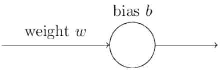
$$C = \frac{(y-a)^2}{2}$$

$$\frac{\partial C}{\partial b} \quad \frac{\partial C}{\partial w}$$

We saw how to compute (and to implement) these derivatives explicitly.

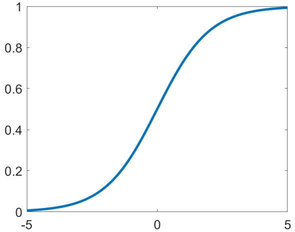
First improvement : $C \to C$ (cross-entropy loss)

The problem with quadratic loss

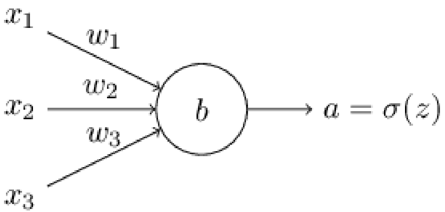


$y = 0, x = 1$

$$\frac{\partial C}{\partial w} = a\sigma'(z) \qquad \frac{\partial C}{\partial b} = a\sigma'(z)$$



Cross-entropy loss



$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right]$$

Guided exercise : 1) interpretation of the cross-entropy function 2) derivation from the maximum likelihood approach and 3) interpretation (derivation left as an exercise) of the formulas

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y) \qquad \frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y)$$

Exercise : derivation from MLE

In Nielsen's book you will find different motivations. One heuristic and the other more theoretical: check it for a comparison !

- Define a probabilistic setting : we want to model $p(y = k|x) \quad k = 0, 1, \ldots 9$
- Write the MLE problem
- Refactoring and derive the identity with the minimization of the cross-entropy
- Information theoretical approach

Demonstration : slow learning with quadratic C

MNIST T-SNE visualization, van deer Maaten, Hinton 2008 T-SNE on scikit-learn

Straightforward generalization of the loss

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

Go to Jupyter Notebook !

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.

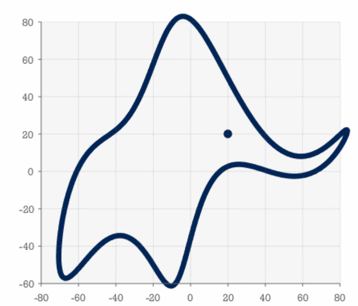
Overfitting

"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

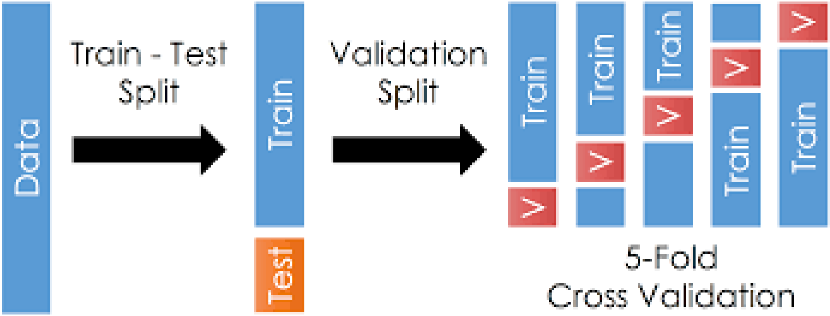
- Fully connected $[784, 30, 10]$: $\sim 24K$
- Fully connected $[784, 100, 10]$: $\sim 80K$
- LeNet (1998) : $\sim 60K$
- AlexNet (2012) : $\sim 60M$
- VGG-16 (2014) : $\sim 138M$
- $\ldots$

The overfitting problem in neural networks is potentially serious and we will address it.

But it is milder than expected, and this is yet a mystery !

Cross-validation

cross-validation on scikit-learn



Overfitting / Underfitting Example

True model that generates data points

$$y = cos(x) + \varepsilon \qquad \varepsilon : \text{stochastic perturbation}$$

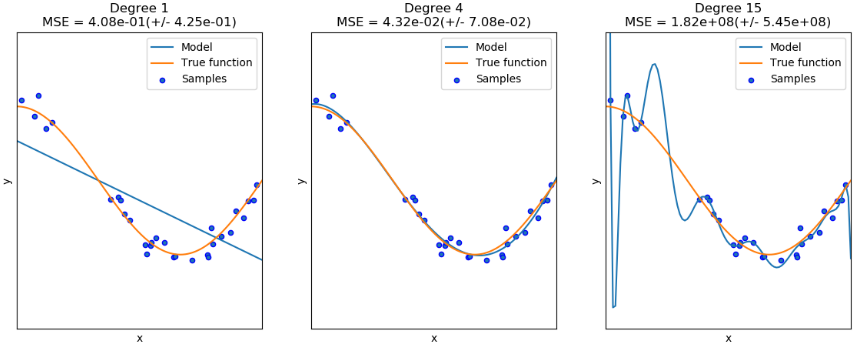
We sample randomly 30 points from the true model and then find the best fit of several models

Three models of growing complexity

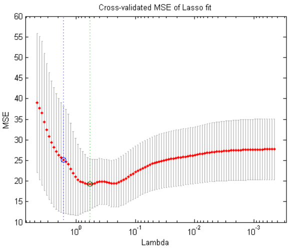
- $y = \theta_0 + \theta_1 x$ (2 parameters)
- $y = \theta_0 + \theta_1 x + \cdots + \theta_4 x^4$ (5 parameters)
- $y = \theta_0 + \theta_1 x + \cdots + \cdots + \theta_{10} x^{15}$ (16 parameters)

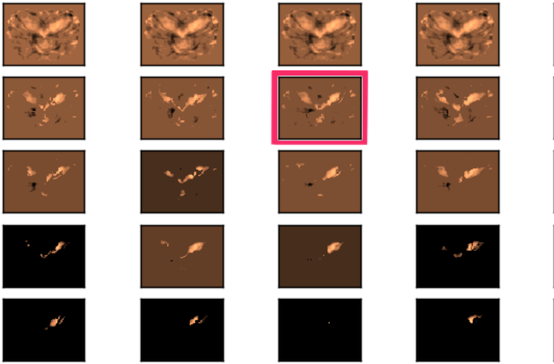
We will use (10-fold) cross-validation

Overfitting / Underfitting Example



Outcome : the Hyper-parameter(s) choice





Regularization

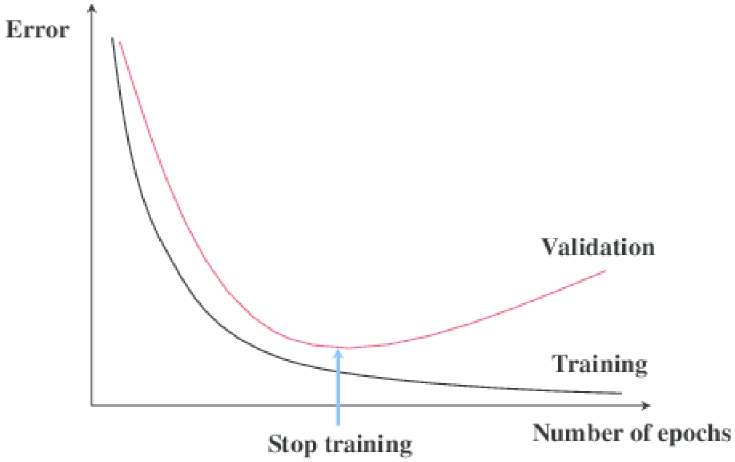
The general approach with Neural Networks

- Choose a model large enough to ensure overfitting (and with the right architectural prior, as we will see in Lecture 3)
- Regularize it in order to tame its complexity, minimizing overfitting

Regularization techniques ($\star$ typically requires hyper-parameters search)

- $\star$ Early stopping $\rightarrow$ n.of epochs
- $\star$ Weight decay (L_1 (lasso), L_2 (ridge)) $\rightarrow \lambda_1, \lambda_2$
- Data augmentation (rotations, translations, etc.)
- $\star$ Dropout $\rightarrow p$
- ...

Early stopping



Regularization by weight decay

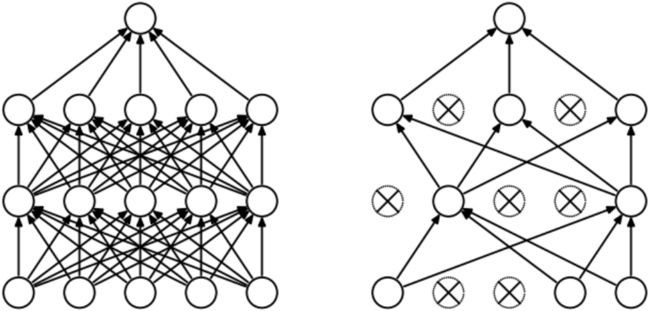
Reduce not the model complexity (number of parameters) but the *effective* model complexity i.e. the number of parameters that play an important role (not applied on the biases)

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

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

Dropout

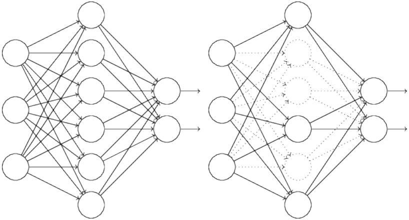
Does not modify C but the network's way to operate



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

Dropout

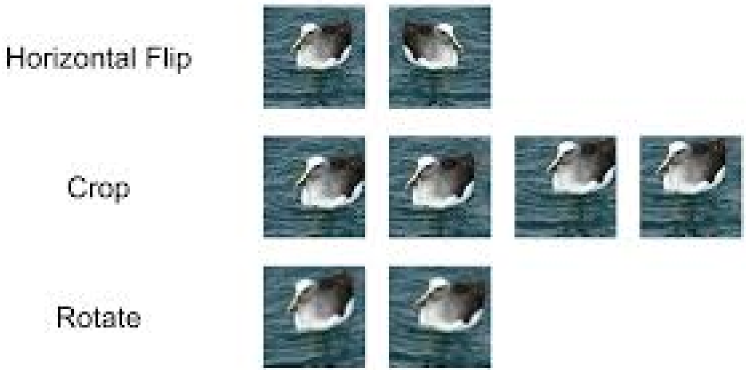
In each mini-batch we randomly deactivate units with probability p



"it's [...] like we're training different neural networks [...] the different networks will overfit in different ways and so, hopefully, the net effect of dropout will be to reduce overfitting."
"[...] reduces complex co-adaptations of neurons [...]"
(We will see Dropout at work in Lecture3)

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

Data augmentation



Data augmentation: MNIST

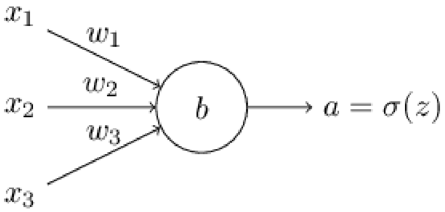
- small rotations
- small translations
- elastic distortions (emulate the random oscillations found in hand muscles) 
- other ...

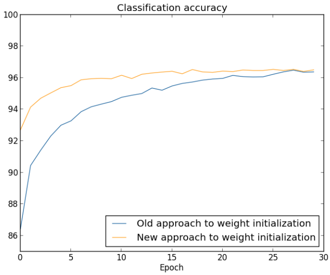
More data means more chance for the network to *make experience* of the natural variability in the data. The data augmentation strategy should reflect this variability.
(We will implement a simple data augmentation for MNIST in Lecture 3)

Question : "What about speech recognition ? What is a good data augmentation strategy ?"

Weights initialization

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





Bibliography

- "Visualizing Data using t-SNE", van deer Maaten, Hinton (2008)
- http://colah.github.io/posts/2015-01-Visualizing-Representations/
- "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).
- "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis", Simard, Steinkraus and Platt, (2003)