Signal Processing Systems

Machine learning for signal processing *[5LSL0]*

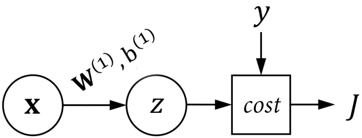
Ruud van Sloun, Rik Vullings

Technische Universiteit
Eindhoven
University of Technology

May 24, 2018

Where innovation starts

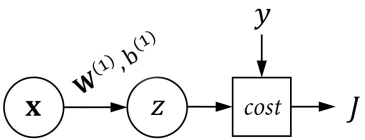
Recap Wiener filter



Differences in notation and assumption with respect to week 1 and 2:

- $\mathbf{x}$ no longer zero mean. The bias b accounts for non-zero means
- $\mathbf{z}$ is the same as $\hat{y}$

Recap Wiener filter

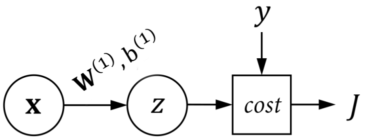


Differences in notation and assumption with respect to week 1 and 2:

- $\mathbf{x}$ no longer zero mean. The bias b accounts for non-zero means
- $\mathbf{z}$ is the same as $\hat{y}$

Cost function MSE.

Recap Wiener filter



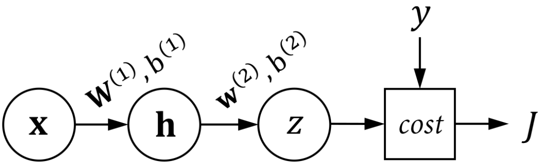
Differences in notation and assumption with respect to week 1 and 2:

- $\mathbf{x}$ no longer zero mean. The bias b accounts for non-zero means
- $\mathbf{z}$ is the same as $\hat{y}$

Cost function MSE.

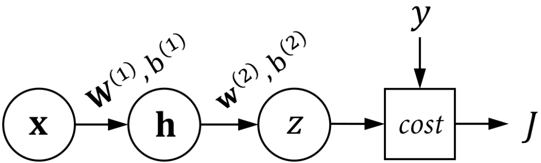
Optimization of weights W and b via gradient descent.

Recap Activation functions



We have added a non-linearity **h**.

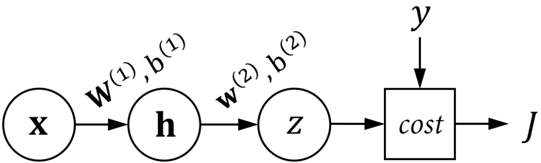
Recap Activation functions



We have added a non-linearity **h**.

Cost function MSE or other, possibly extended with regularization. E.g. MSE + L^2 norm penalty

Recap Activation functions



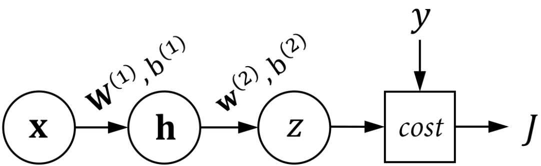
We have added a non-linearity **h**.

Cost function MSE or other, possibly extended with regularization. E.g. MSE + L^2 norm penalty

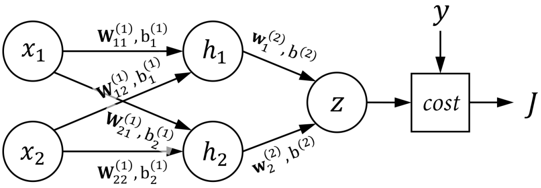
Optimization via (stochastic) gradient descent

Technische Universiteit Eindhoven University of Technology

Recap Activation functions



Fully connected layers



Sparse connectivity

Fully connected layers require many parameters:

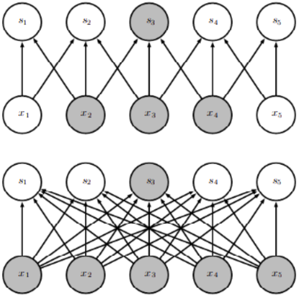
- Memory: we need more memory to store all parameters
- Statistical inefficient: we need more data to train the network

Sparse connectivity

Fully connected layers require many parameters:

- Memory: we need more memory to store all parameters
- Statistical inefficient: we need more data to train the network

Instead, consider sparse connections to save memory and increase statistical efficiency.



from deeplearningbook.org

Sparse connectivity

Fully connected layers require many parameters:

- Memory: we need more memory to store all parameters
- Statistical inefficient: we need more data to train the network

Instead, consider sparse connections to save memory and increase statistical efficiency.

In deep networks, output still indirectly connected to most inputs

Parameter sharing

Recap from regularization

Parameter sharing Some model parameters are forced to be identical

Two main reasons:

- Similar (parts of) models should often have similar behavior
- Reduce amount of parameters (memory and statistical efficiency)

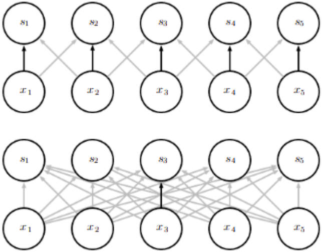
Parameter sharing

Recap from regularization

Parameter sharing Some model parameters are forced to be identical

Two main reasons:

- Similar (parts of) models should often have similar behavior
- Reduce amount of parameters (memory and statistical efficiency)



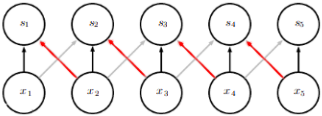
from deeplearningbook.org

Here, the black arrows represent shared parameters.

Technische Universiteit Eindhoven University of Technology

Parameter sharing

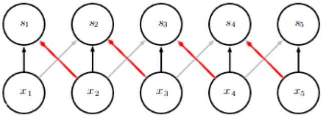
Let's take this one step further, by also sharing parameters indicated by red and grey arrows:



modified from deeplearningbook.org

Parameter sharing

Let's take this one step further, by also sharing parameters indicated by red and grey arrows:

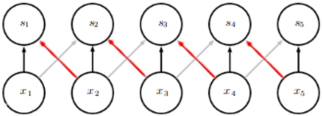


modified from deeplearningbook.org

Similar to moving a *kernel* of weights over the inputs **x** to yield output **s**.

Parameter sharing

Let's take this one step further, by also sharing parameters indicated by red and grey arrows:



modified from deeplearningbook.org

Similar to moving a *kernel* of weights over the inputs **x** to yield output **s**.

This concept is known as *convolution*

Convolution neural networks

In short:

- (Sparse) network of linear weights and non-linear activations
- Weights optimized via (modified versions of) stochastic gradient descent
- Typically, optimization regularized to enable generalization

Highly similar to the "simple" filters that we started with!

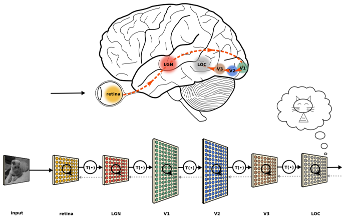
Convolution neural networks

In short:

- (Sparse) network of linear weights and non-linear activations
- Weights optimized via (modified versions of) stochastic gradient descent
- Typically, optimization regularized to enable generalization

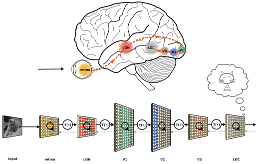
Highly similar to the "simple" filters that we started with!

Inspired by visual system in mammals



https://figshare.com/articles/Ventral_visual_stream/106794

Visual system



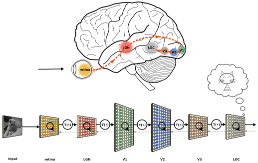
https://figshare.com/articles/Ventral_visual_stream/106794

Main processing of visual images in primary visual cortex (V1):

- V1 arranged in 2D spatial map: light on lower half of retina, only affects lower half of V1 $\rightarrow$ Conv nets: 2D maps

Technische Universiteit
Eindhoven
University of Technology

Visual system

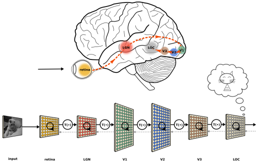


https://figshare.com/articles/Ventral_visual_stream/106794

Main processing of visual images in primary visual cortex (V1):

- V1 arranged in 2D spatial map: light on lower half of retina, only affects lower half of V1 $\rightarrow$ Conv nets: 2D maps
- V1 contains simple cells: activity characterized by linear functions of image in small, spatially localized field $\rightarrow$ Conv nets: feature detector units

Visual system



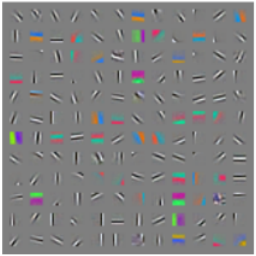
Main processing of visual images in primary visual cortex (V1):

- V1 arranged in 2D spatial map: light on lower half of retina, only affects lower half of V1 $\rightarrow$ Conv nets: 2D maps
- V1 contains simple cells: activity characterized by linear functions of image in small, spatially localized field $\rightarrow$ Conv nets: feature detector units
- V1 contains complex cells: invariant to small shifts in position of feature $\rightarrow$ Conv nets: pooling

Technische Universiteit Eindhoven University of Technology

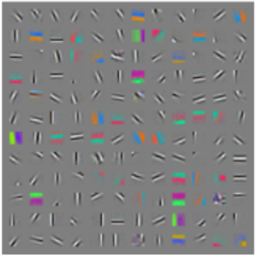
Visual system

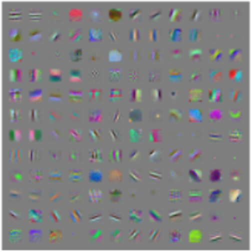
V1 cells are known to have weights described by Gabor functions. Extensive studies show that V1 learns mostly the edges or specific colors of edges when applied to natural images



Visual system

V1 cells are known to have weights described by Gabor functions. Extensive studies show that V1 learns mostly the edges or specific colors of edges when applied to natural images





Supervised deep learning approaches have been shown to typically also learn these features already in their first layer

Convolution

Convolution is a mathematical operator:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\mathrm{d}a$$

Convolution

Convolution is a mathematical operator:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\mathrm{d}a$$

In discrete-time signals

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a)$$

Convolution

Convolution is a mathematical operator:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\mathrm{d}a$$

In discrete-time signals

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a)$$

Note, although we refer to them as convolutional neural networks, often we do not apply convolutions, but cross-correlations:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t+a)$$

Technische Universiteit Eindhoven University of Technology

Convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t+a)$$

Terminology:
kernel: w (think of filter coefficients)
stride: step-size of a (in example above: stride = 1)

Convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t+a)$$

Terminology:
kernel: w (think of filter coefficients)
stride: step-size of a (in example above: stride = 1)

Question: why stride > 1 can be regarded as downsampling of output $s(t)$?

Properties of convnets

Convnets leverage four important concepts:

- Sparse connections
- Parameter sharing
- Equivariant representations
- Tolerance to variable input dimensions

Technische Universiteit Eindhoven University of Technology

Properties of convnets

Convnets leverage four important concepts:

- Sparse connections
- Parameter sharing
- Equivariant representations
- Tolerance to variable input dimensions

Equivariance functions:

$$f(g(x)) = g(f(x))$$

Technische Universiteit Eindhoven University of Technology

Properties of convnets

Convnets leverage four important concepts:

- Sparse connections
- Parameter sharing
- Equivariant representations
- Tolerance to variable input dimensions

Equivariance functions:

$$f(g(x)) = g(f(x))$$

Consider $g(x)$ to be a shift operator, applied to image I and $f(x)$ convolution:

$$I'(x, y) = g(I(x, y)) = I(x-1, y)$$

Prove yourself that

$$f(g(x)) = g(f(x))$$

Technische Universiteit Eindhoven University of Technology

Pooling

Convolution is equivariant to translation, but NOT equivariant to *scaling* and *rotation*

A simple convnet trained to detect the cat in the left image, would succeed in detecting the cat in the second image, but not in the right image.

Pooling

Convolution is equivariant to translation, but NOT equivariant to *scaling* and *rotation*

A simple convnet trained to detect the cat in the left image, would succeed in detecting the cat in the second image, but not in the right image.

By pooling, the convnet can learn invariance to scaling and rotation

Pooling

Typical layer of a convolution network consists of three stages:

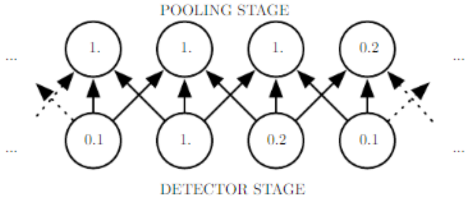
1. Convolutions
2. Nonlinear activations (e.g. ReLU) (detector stage)
3. Pooling

Pooling

Typical layer of a convolution network consists of three stages:

1. Convolutions
2. Nonlinear activations (e.g. ReLU) (detector stage)
3. Pooling

Max pooling: report maximum output within a rectangular neighbourhood.

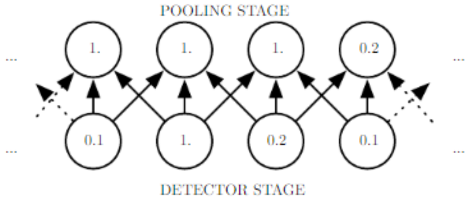


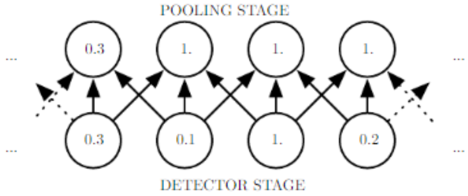
from deeplearningbook.org

Technische Universiteit Eindhoven University of Technology

Pooling

Pooling is equivariant to translations...

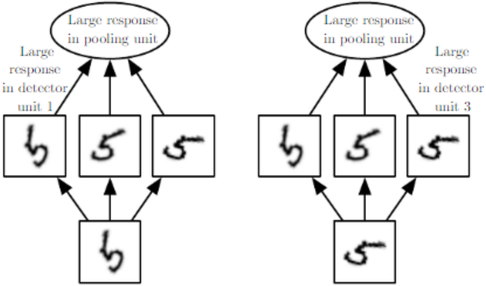




from deeplearningbook.org

Pooling

... but also rotations



from deeplearningbook.org

Pooling

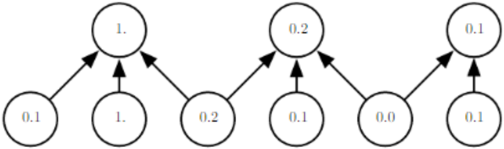
Recap definition:
Max pooling: report maximum output within a rectangular neighbourhood.

Technische Universiteit
Eindhoven
University of Technology

Pooling

Recap definition:
Max pooling: report maximum output within a rectangular neighbourhood.

This enables the use of max pooling for downsampling (i.e. use fewer pooling units than detector units)

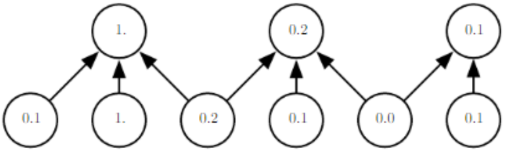


from deeplearningbook.org

Pooling

Recap definition:
Max pooling: report maximum output within a rectangular neighbourhood.

This enables the use of max pooling for downsampling (i.e. use fewer pooling units than detector units)



from deeplearningbook.org

This enables:

- Computational efficiency
- Handling variable size inputs (by varying size of pooling regions to match the targeted output dimensions)

Technische Universiteit Eindhoven University of Technology

Convolution + pooling as infinitely strong prior

Consider an example of classifying an image X in one (out of multiple) classes y.

Convolution + pooling as infinitely strong prior

Consider an example of classifying an image X in one (out of multiple) classes y.
Bayes rule:

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)}$$

with

- $p(y|X)$ the posterior probability that X belongs to class y, given the image data X
- $p(y)$ our prior beliefs about the distribution of classes (e.g. we might assume that there will be far more cats than dogs in an image set, without having seen any images)
- $p(X|y)$ the probability (likelihood) of X, given that it belongs in class y (e.g. how much does it look like a cat, or how much does it look like a dog)

Bayes rule says that the posterior is proportional to the product of prior and likelihood.

Convolution + pooling as infinitely strong prior

The entropy of the prior probability distribution (for Gaussian distribution: the variance) expresses the strength of our *a priori* beliefs. A infinitely low entropy, yields an infinitely strong prior. In other words, the support by the data does not matter anymore; our initial beliefs determine the class

Convolution + pooling as infinitely strong prior

The entropy of the prior probability distribution (for Gaussian distribution: the variance) expresses the strength of our *a priori* beliefs. A infinitely low entropy, yields an infinitely strong prior. In other words, the support by the data does not matter anymore; our initial beliefs determine the class

Convnets can be regarded as fully connected layers with infinitely strong priors:

- that weights for hidden unit are identical to that of its neighbour, but shifted in space
- that weights are zero, except for receptive field (kernel) assigned to hidden unit
- that the layer should learn based on only local interactions and is invariant to translation

Convolution + pooling as infinitely strong prior

Based on this insight, one can see that

- Convolution and pooling can cause underfitting: the priors can be wrong, yielding a (too) large training loss

Practical implementations

For implementing Convolutional Neural Networks, you are strongly encouraged to use available repositories like

- TensorFlow (www.tensorflow.org)
- Keras (keras.io)

Also, in Chapter 11 of "Deep Learning" (Goodfellow et al.), suggestions for practical implementations and selection strategies for hyperparameters are provided.