Artificial Intelligence

Neural Networks

Lesson 9: Deep Learning

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Motivation
- Main problems
- Overfitting
- Vanishing gradient
- Auto-encoders
- Stacked auto-encoders
- Convolutional neural networks (CNNs)

Motivation (1)

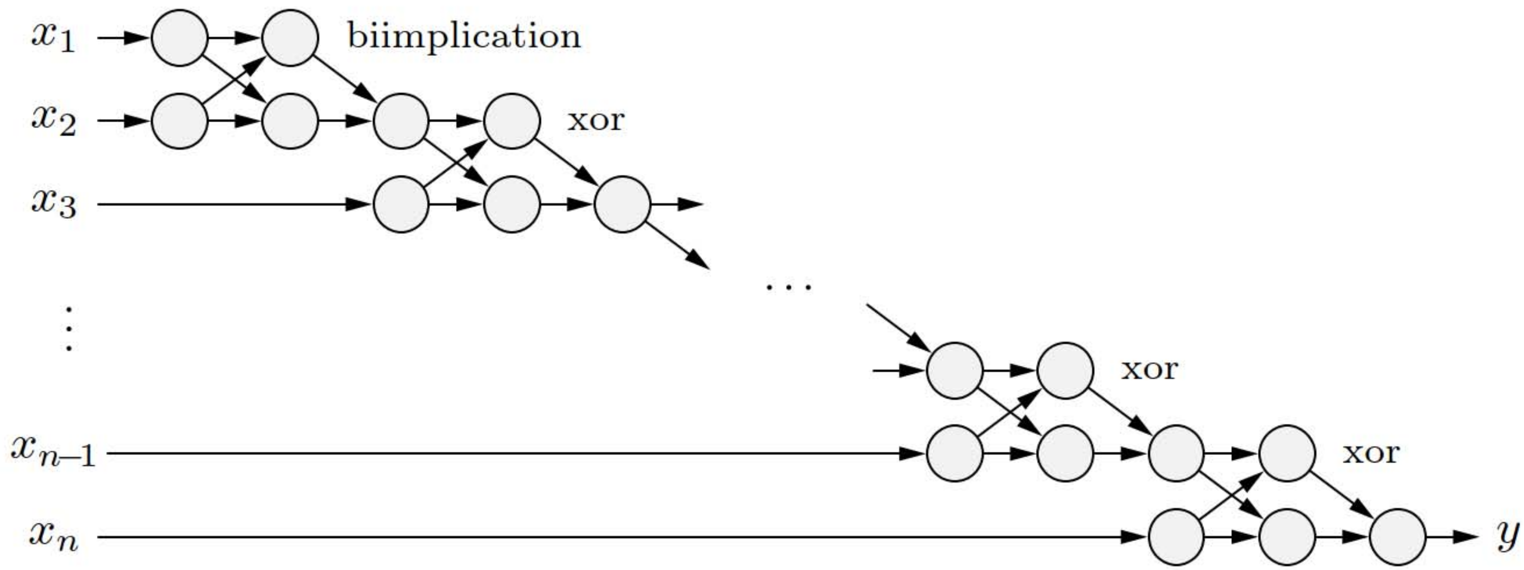
- Universal Approximation Theorem
  - Any continuous function can be approximated arbitrarily well with a three-layer perceptron
  - We can consider multi-layer perceptrons with only one hidden layer?
- However
  - No information about the number of hidden neurons to achieve a desired approximation accuracy
  - More hidden layers may enable to achieve the same approximation quality with a significantly lower number of neurons

Motivation (2)

- n-bit parity function
  - Output is 1 if an even number of inputs is 1; output is 0 otherwise
  - Multi-layer perceptron with only one hidden layer
    - One hidden layer with 2^{n-1} neurons
    - Number of hidden neurons grows **exponentially** with the number of inputs
  - Multiple hidden layers
    - If more hidden layers are admissible, **linear growth** is possible

Motivation (3)

- n-bit parity function



  - The number of neurons increases to $n(n+1)-1$

Motivation (4)

- Training data sets are necessarily limited in size
  - Complete training data for an n-ary Boolean function has 2^n training examples
  - Data sets for practical problems usually contain much fewer sample cases
  - Using more than one hidden layer promises in many cases to reduce the number of needed neurons

Motivation (5)

- Deep Learning
  - More than one hidden layer
  - **Depth**: the length of the longest path in the network graph

Main Problems

- Overfitting
  - Increased number of adaptable parameters
- Vanishing gradient
  - The gradient tends to vanish if many layers are backpropagated through
  - Learning in the early hidden layers can become very slow

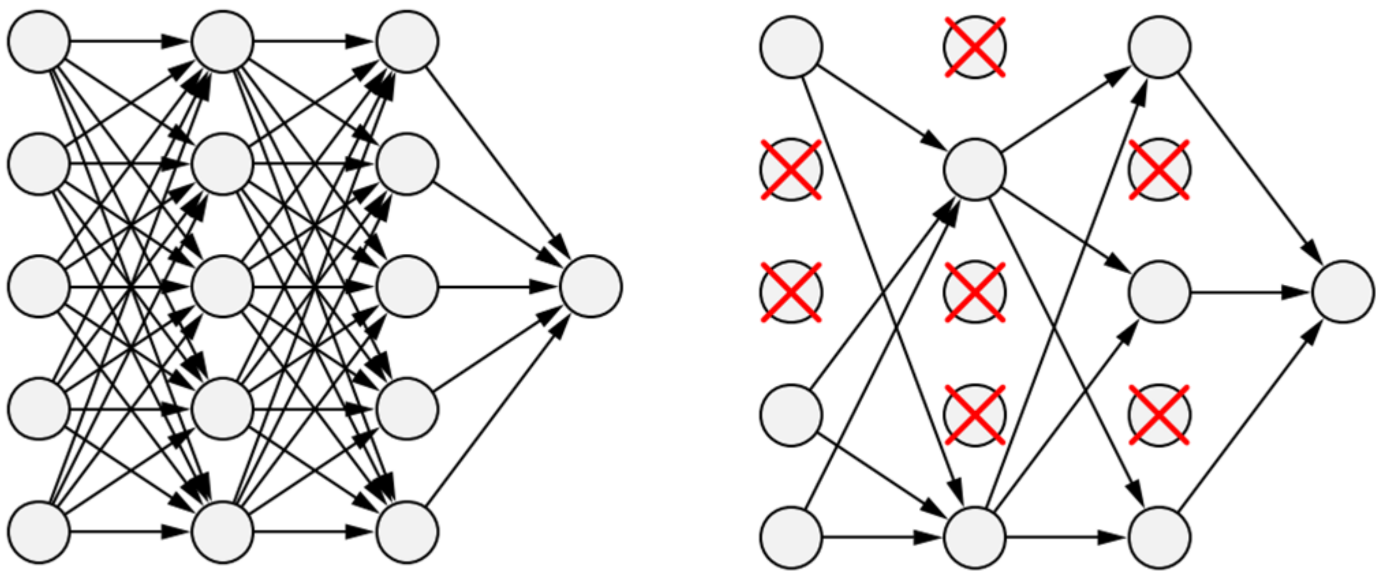
Overfitting (1)

- Overfitting: possible solutions
  - Weight decay
    - Prevents large weights and thus an overly precise adaptation
  - Sparsity constraints to avoid overfitting
    - Restricted number of neurons in the hidden layers
    - Only few hidden neurons should be active (on average)

Overfitting (2)

– Dropout training

  - Some units are randomly omitted from the input/hidden layers during training



Vanishing Gradient (1)

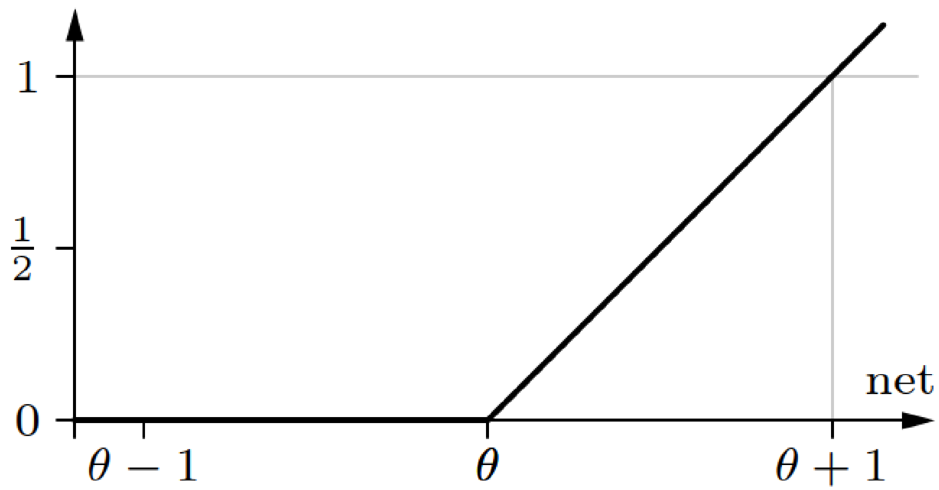
- Vanishing gradient: possible solutions
  - In principle, a small gradient may be counteracted by a large weight
  - However a large weight saturates the activation functions
  - In most cases, the gradient in the first hidden layer (were the inputs are processed) becomes the smaller

Vanishing Gradient (2)

- Other activation functions limit the problem of vanishing gradient
  - Rectified linear units (ReLUs)

rectified maximum/ramp function:

$$f_{\text{act}}(\text{net}, \theta) = \max\{0, \text{net} - \theta\}$$



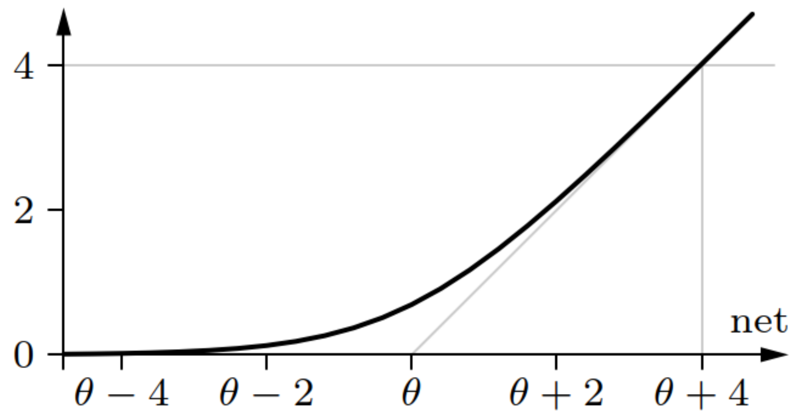
Vanishing Gradient (3)

- Softplus function (more complex)

softplus function:

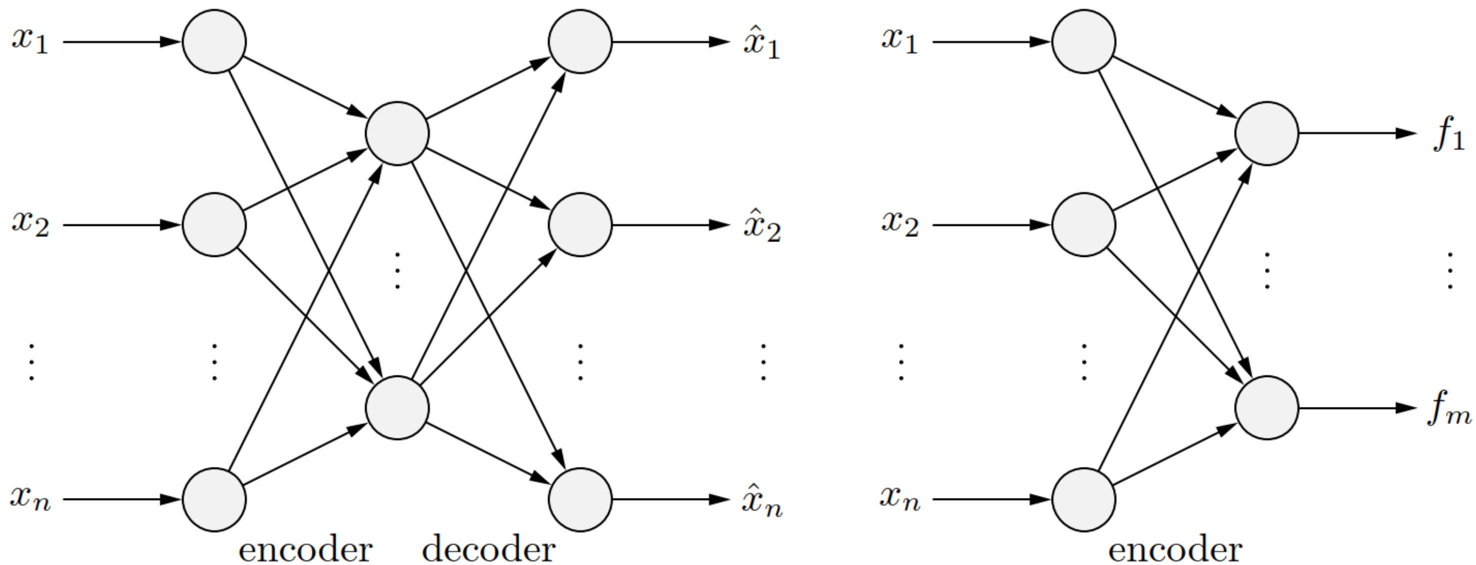
Note the scale!

$$f_{\text{act}}(\text{net}, \theta) = \ln(1 + e^{\text{net} - \theta})$$



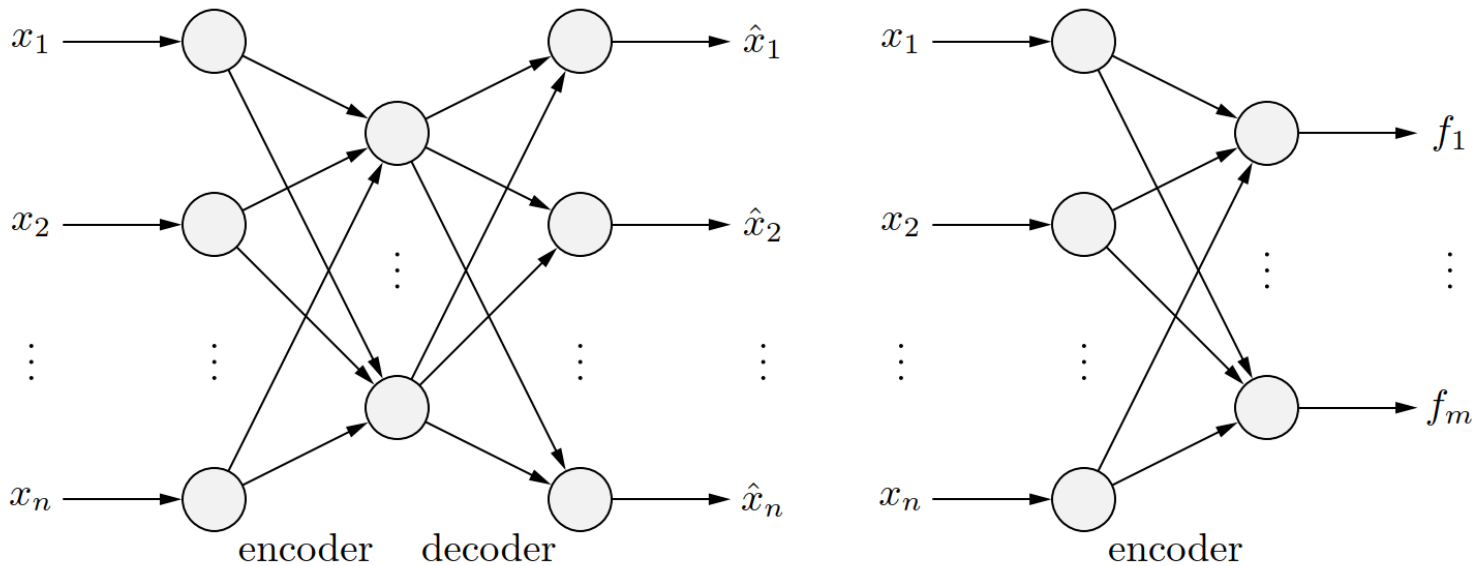
- Alternatives
  - Leaky Rectified Linear Units: parametric slope
  - Noisy Rectified Linear Units: adding Gaussian noise

Auto-Encoders (1)



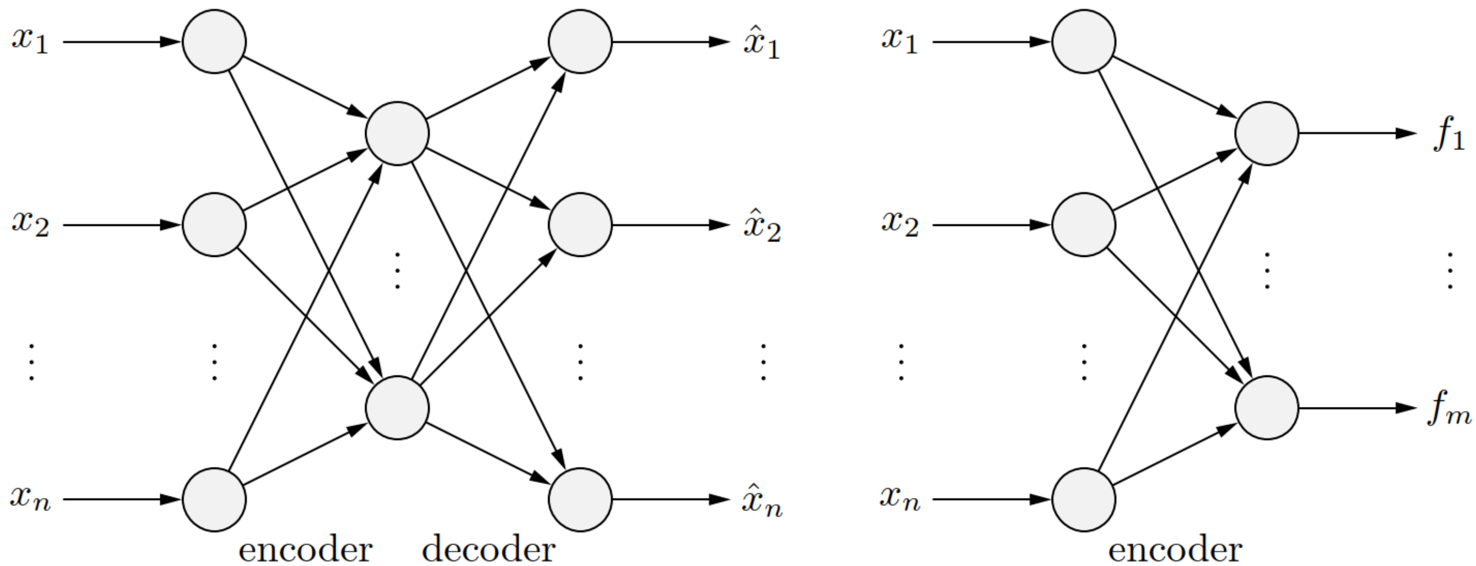
- An auto-encoder is a 3-layer perceptron that maps its inputs to approximations of these inputs
  - The hidden layer forms an encoder into some form of internal representation
  - The output layer forms a decoder that (approximately) reconstructs the inputs

Auto-Encoders (2)



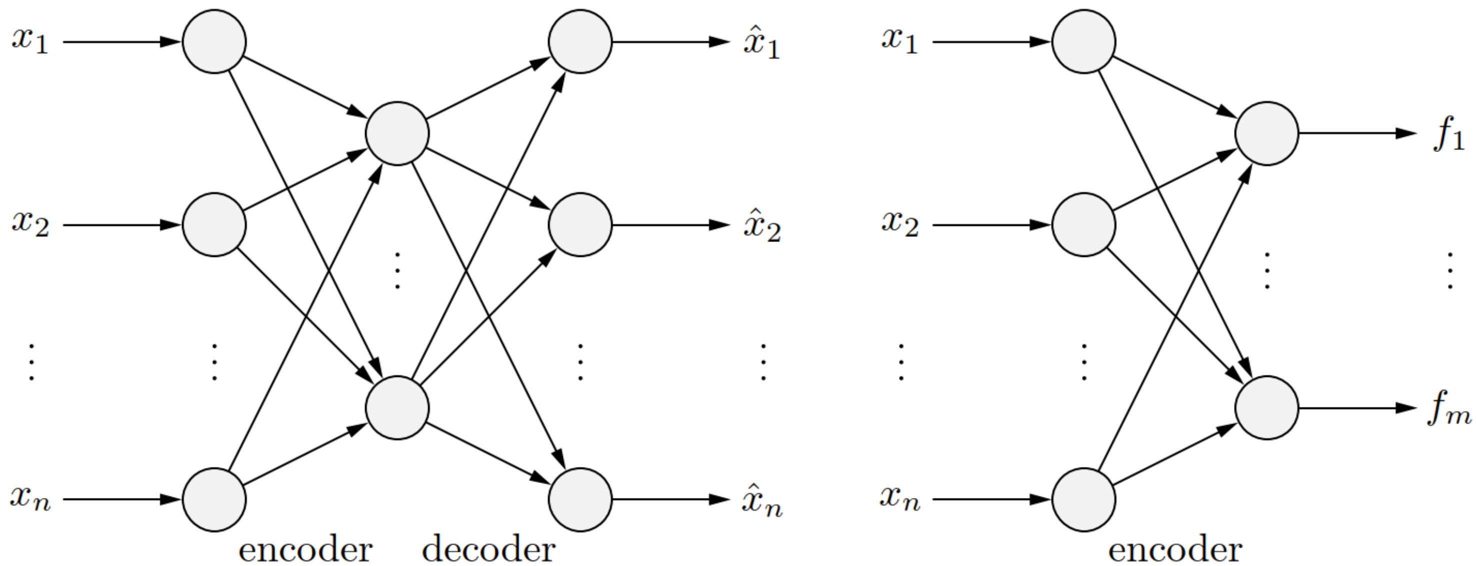
- An auto-encoder/decoder (left), of which only the encoder part (right) is later used
  - x_i are the given inputs, $\hat{x}_i$ are the reconstructed inputs, f_i are the constructed features
  - Training is conducted with error backpropagation

Auto-Encoders (3)



- The hidden layer is expected to construct features
  - Features capture the information contained in the input in a compressed form (encoder), so that the input can be well reconstructed from it (decoder)

Auto-Encoders (4)



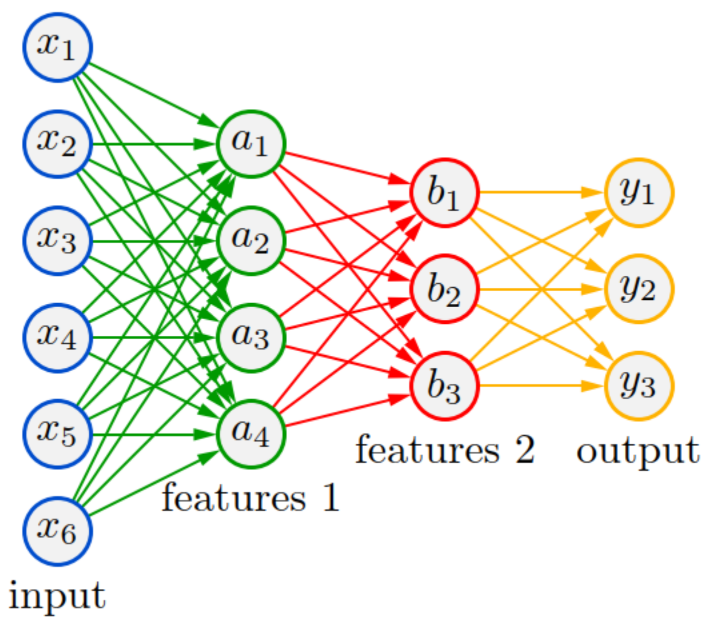
- Problem
  - If there are as many (or even more) hidden units as there are inputs, it is likely that it will merely pass through its inputs to the output layer

Auto-Encoders (5)

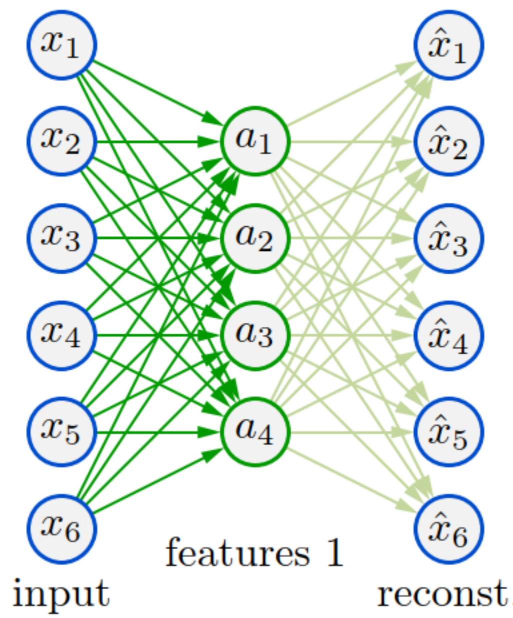
- Solutions
  - **Sparse auto-encoders**
    - There should be (considerably) fewer hidden neurons than there are inputs
    - Few hidden neurons force the auto-encoder to learn relevant features
  - **Sparse activation scheme**
    - The number of active neurons in the hidden layer is restricted to a small number
  - **Denoising auto-encoders**
    - Add noise (random variations) to the input

Stacked Auto-Encoders (1)

- Stacked auto-encoders
  - Build network layer by layer,
    train only newly added layer in each step

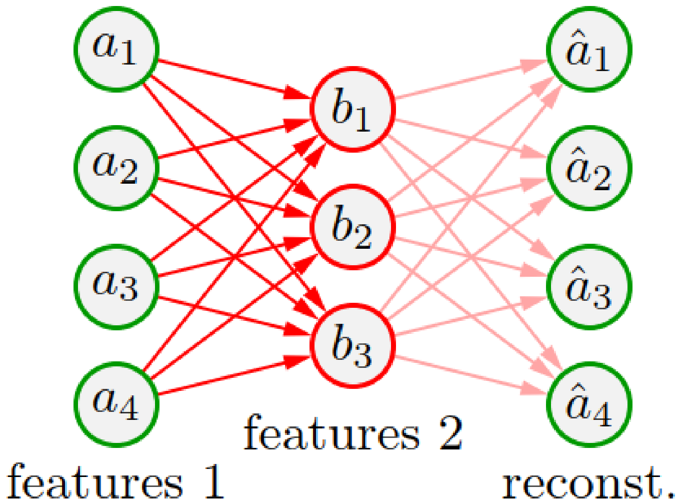


Stacked Auto-Encoders (2)



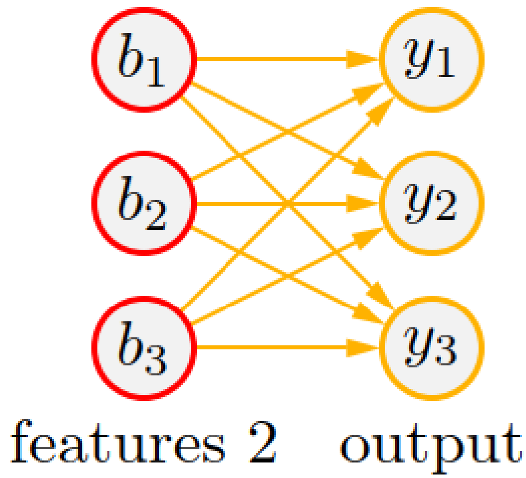
- In the first step, auto-encoder is trained for the raw input features. The hidden layer constructs **primary features** useful for reconstruction

Stacked Auto-Encoders (3)



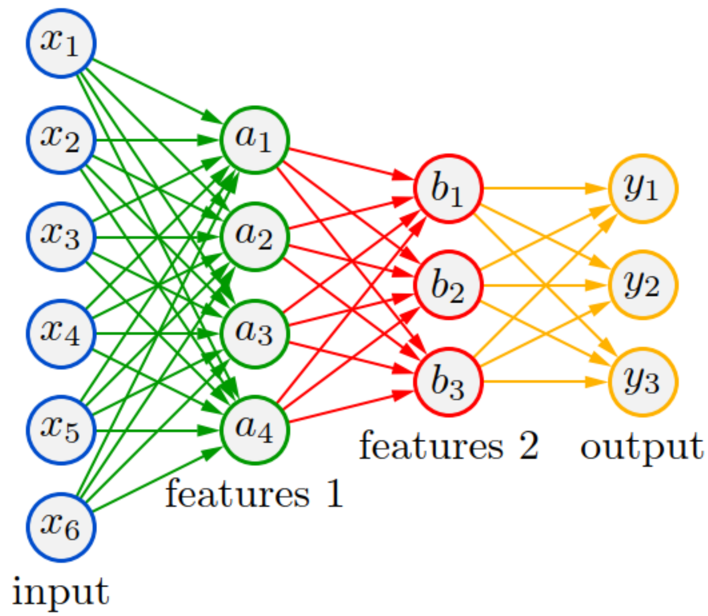
- In the second step, auto-encoder is trained for the obtained feature data set. The hidden layer constructs **secondary features** useful for reconstruction

Stacked Auto-Encoders (4)



- A classifier/predictor for the output is trained from the secondary feature set

Stacked Auto-Encoders (5)



- Encoder parts of the trained auto-encoders are **stacked** and the resulting network is fine-tuned with error backpropagation

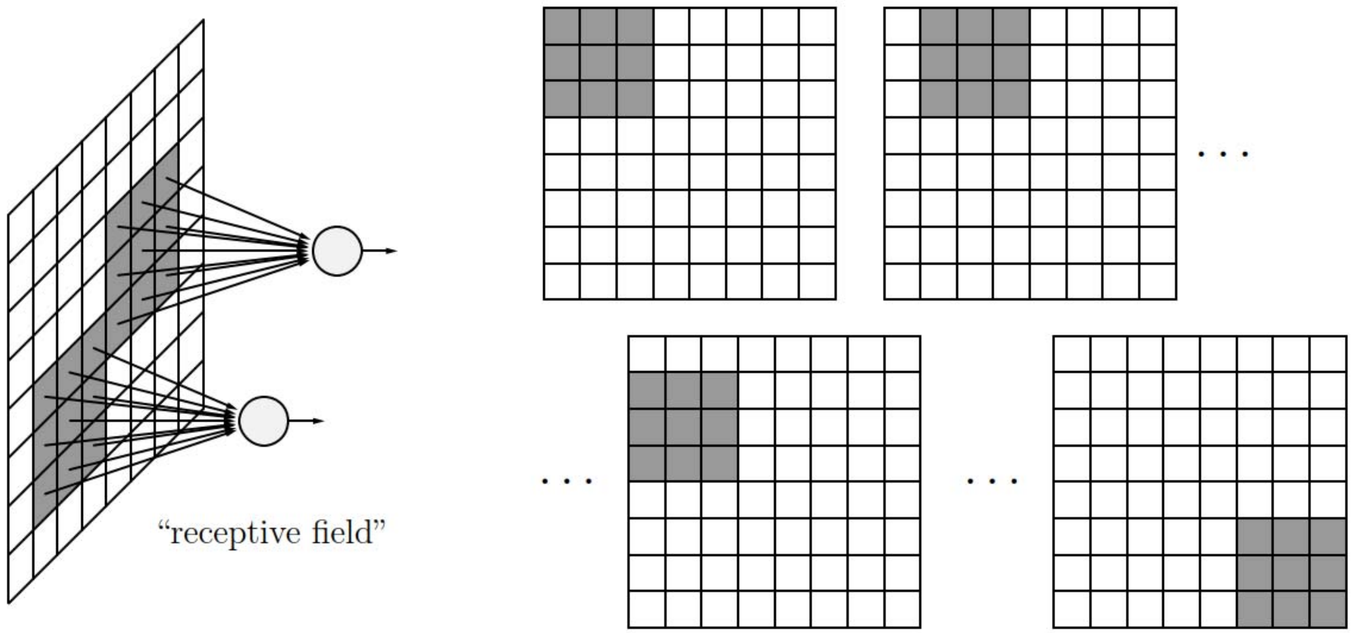
Convolutional Neural Networks (1)

- Multilayer perceptrons with several hidden layers built in the way described have been applied very successfully for handwritten digit recognition
  - The handwriting has already been preprocessed in order to separate the digits
  - Features are localized in the image

Convolutional Neural Networks (2)

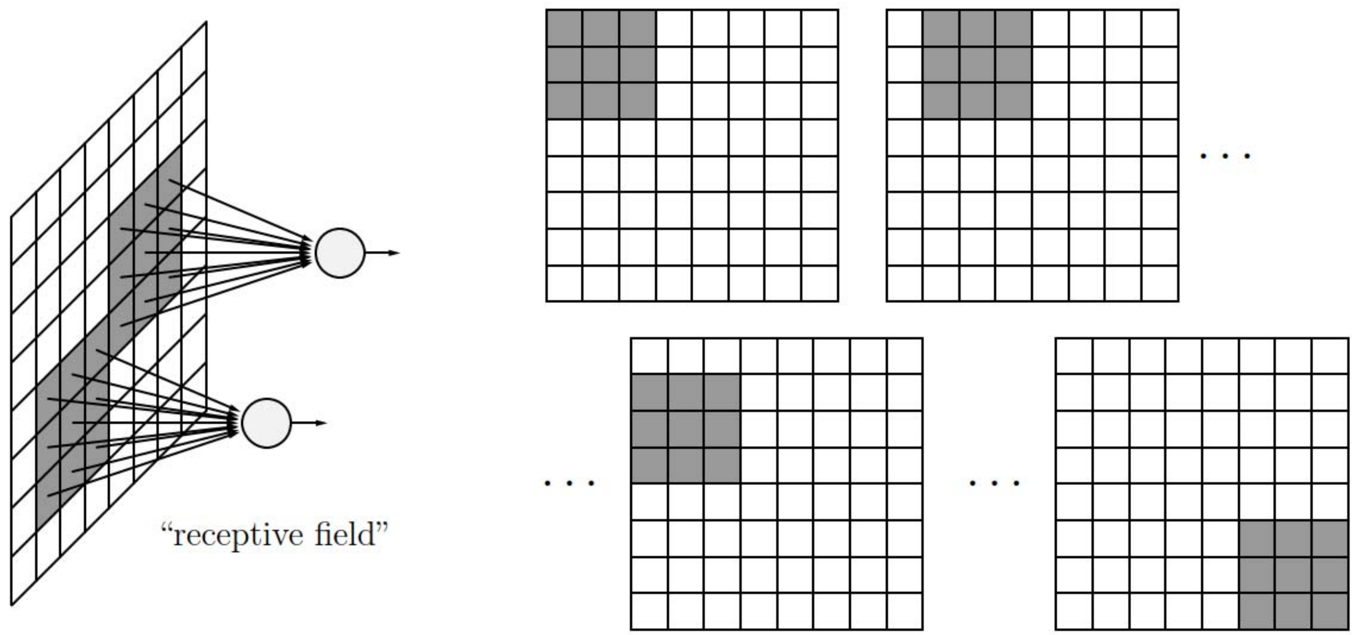
- How to use similar networks also for more general applications
  - Recognizing whole lines of handwriting
  - Analyzing photos to identify their parts as sky, landscape, house, pavement, tree, human being etc.
  - It is advantageous that the features constructed in hidden layers are not localized to a specific part of the image

Convolutional Neural Networks (3)



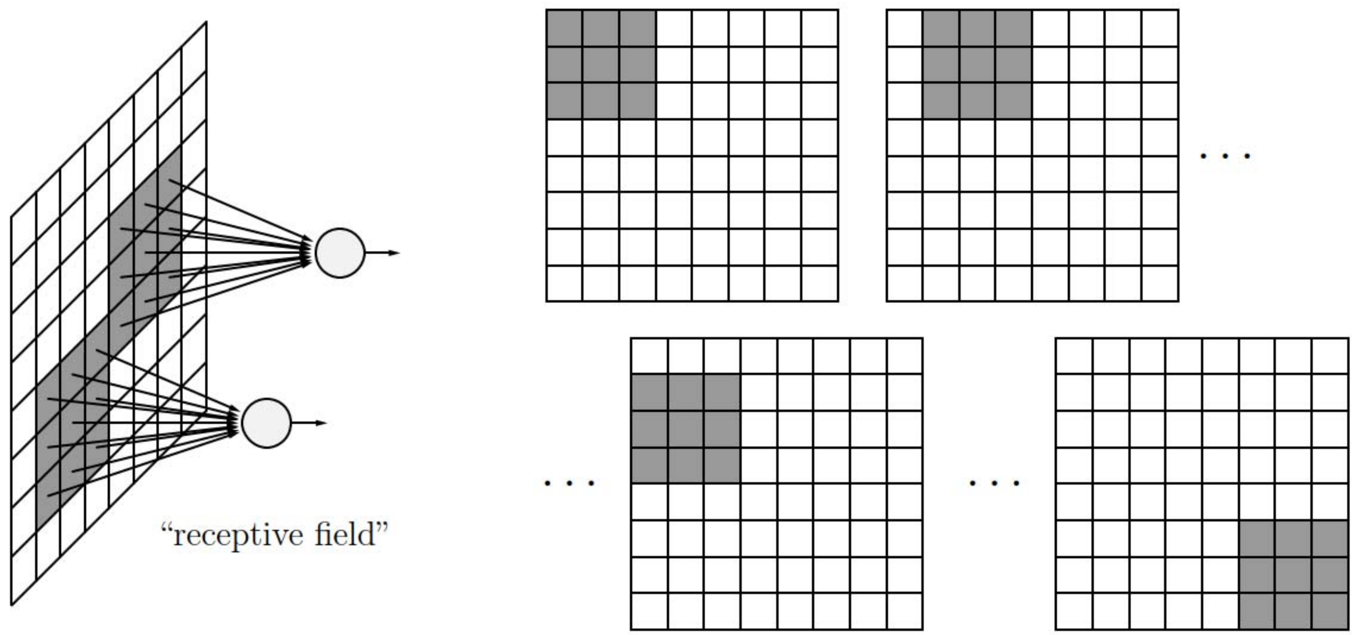
- Convolutional Neural Nets (CNNs) are a special form of deep learning multi-layer perceptron
  - Inspired by the human retina
  - Sensory neurons have a receptive field: a limited region in which they respond to a (visual) stimulus.

Convolutional Neural Networks (4)



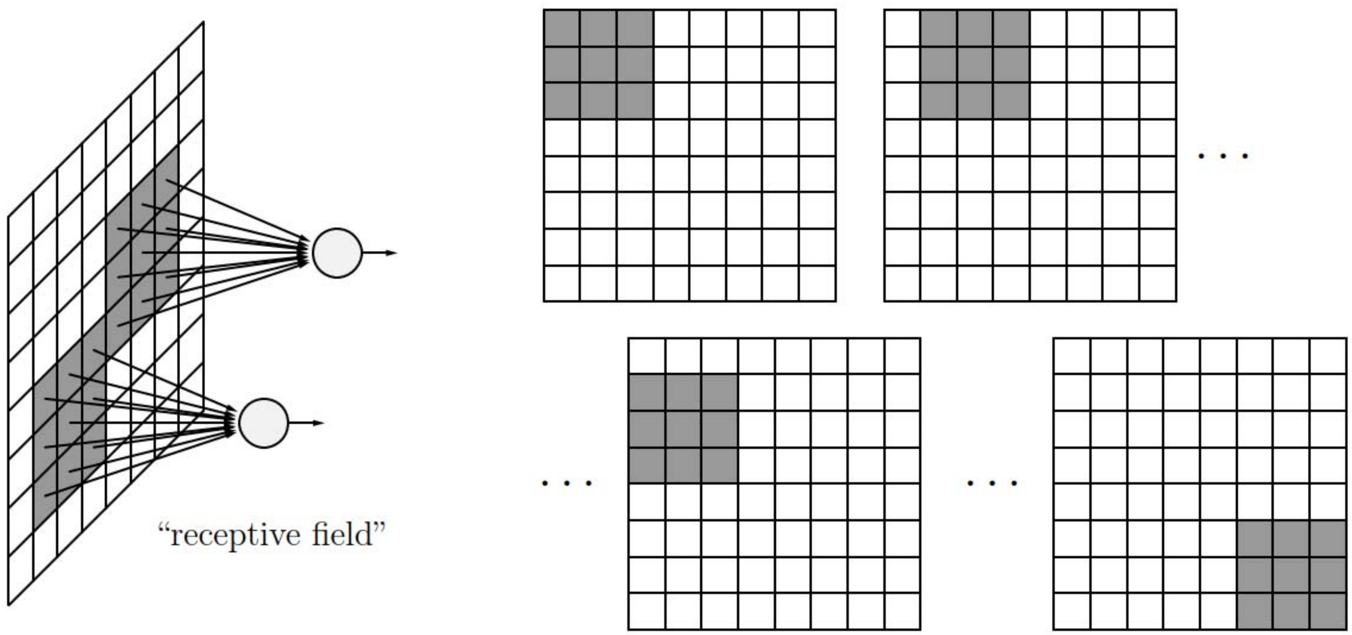
- Each neuron of the (first) hidden layer is connected to a small number of input neurons that refer to a contiguous region of the input image (left).

Convolutional Neural Networks (5)



- Connection weights are shared
- The input field is "moved" step by step over the whole image (right)
- Equivalent to a **convolution** with a small size kernel

Convolutional Neural Networks (6)



- Neurons in the successor layer apply maximum pooling over small regions
- Pooling maintains knowledge of the features, not of their location in the image
- Further layers allow for high-level features