PnS 2018

Deep Learing with Raspberry Pi

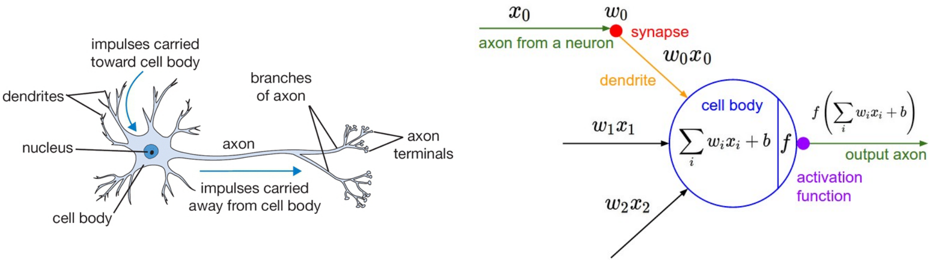
Session 3

PnS 2018 Team

Institute of Neuroinformatics
University of Zürich and ETH Zürich

Outline

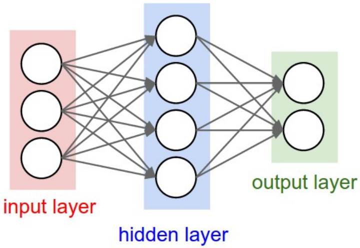
Artifical Neuron: Overview

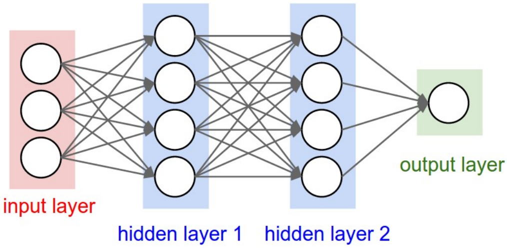


- A basic computational model of the biological model
- Single neuron as linear/logistic regression

http://cs231n.github.io/neural-networks-1/

Multi-Layer Perceptron

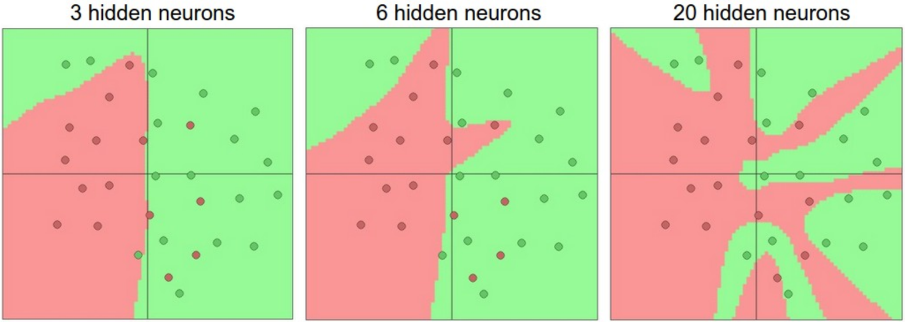




- Neurons in an acyclic feed-forward graph
- Fully connected layers
- Each fully connected layer computation is a matrix multiplication, matrix addition and an activation function

http://cs231n.github.io/neural-networks-1/

What can an MLP learn?



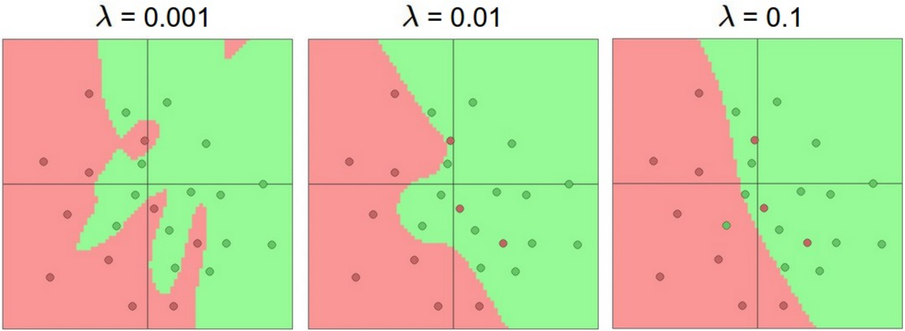
- Neural Networks with at least one hidden layer are universal approximators[1]
- More neurons are expected to approximate better

[1] Approximation by superpositions of a sigmoidal function, by Cybenko G. http://cs231n.github.io/neural-networks-1/

Regularization

- Overfitting more probable with larger models
- Could be prevented by using a regularization term in the loss function

Regularization



- Use bigger networks but take measures to prevent overfitting

http://cs231n.github.io/neural-networks-1/

Working with images

- MLPs do not work well with images
- Hierarchy of local spatial features
- Extract these local spatial features through filters

Convolution operation

$$s(t) = \int x(a)w(t-a)\,da$$

the operation is called *convolution*. The convolution operation is typically denoted with $*$:

$$s(t) = (x * w)(t)$$

In discrete form:

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

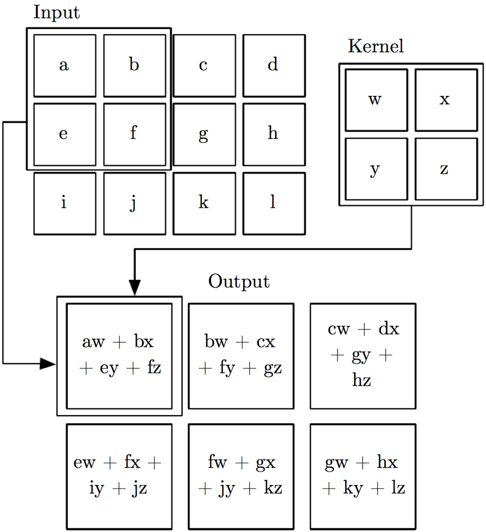
2D convolution operation

$$s[i, j] = (I * K)[i, j] = \sum_m \sum_n I[m, n] K[i - m, j - n]$$

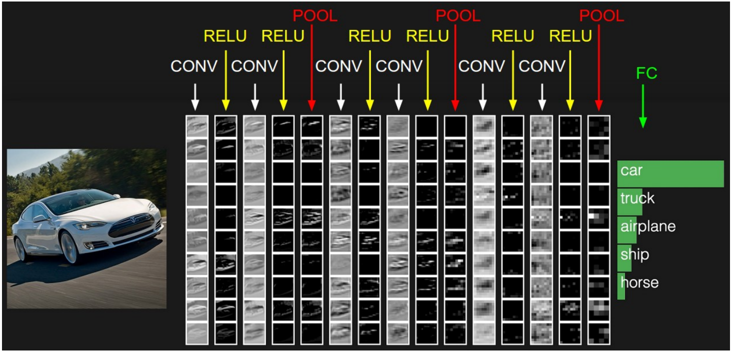
or equivalently:

$$s[i, j] = (I * K)[i, j] = \sum_m \sum_n I[i - m, j - n] K[m, n]$$

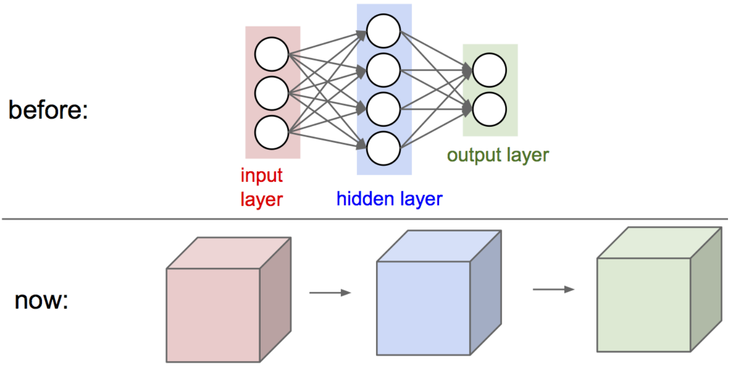
2D convolution operation



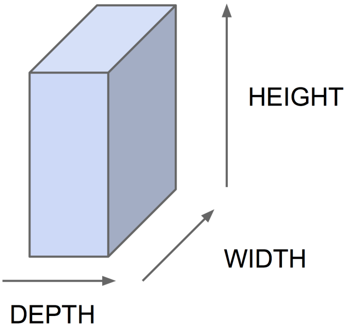
LeNet-5


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

MLP→ConvNet



Feature maps: activations of ConvNets



- Network activations in ConvNets are **feature maps**.
- All ConvNets feature maps arranged in **3 dimensions**.
- Each feature maps has size of (HEIGHT, WIDTH)
- Input image can be a special kind of feature map (*e.g.* color image is feature maps of some size with depth 3, one for each RGB channel).

Convolution Layer: simple cell

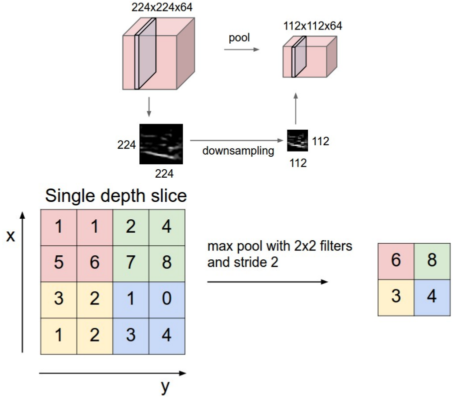
$$\mathbf{h}^{(k)} = f(\mathbf{x} * \mathbf{W}^{(k)} + b_k)$$

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Number of filters K with shape $F \times F \times D_1$, stride S, amount of zero-padding P
- Produce a volume of size $W_2 \times H_2 \times D_2$ where

$$\begin{aligned} W_2 &= (W_1 - F + 2P)/S + 1 \\ H_2 &= (H_1 - F + 2P)/S + 1 \\ D_2 &= K \end{aligned}$$

Live Demo of convolution

Pooling Layer: complex cell



Q&A