Computer Vision Systems Programming VO

Deep Learning

Christopher Pramerdorfer

Computer Vision Lab, Vienna University of Technology

Topics

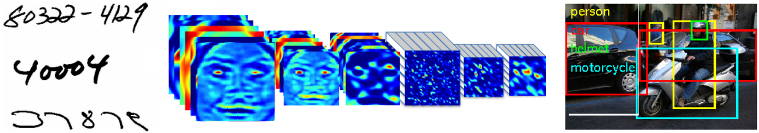
Deep learning motivation

Multilayer perceptrons

Pylearn2 library

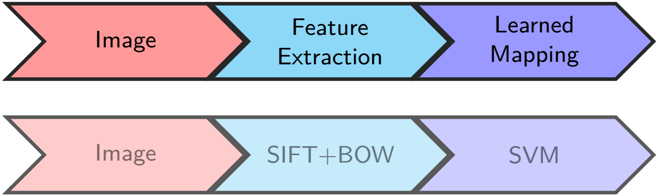
Convolutional neural networks

Deep learning applications



Images from LeCun et al. 1989, Taigman et al. 2013, `image-net.org`

Object Recognition
Traditional Approach



Object Recognition
Traditional Approach

Problem: how to choose the representation/features?

"General" features not optimal

- Not tuned to task at hand, low-level

Designing task-specific features is complex

- Virtually impossible to do optimally

Solution: learn representation as well

Learning high-level representations directly is difficult

Deep Learning (DL) solves this

- By learning a hierarchy of representations
- Layers in hierarchy build upon each other

Object Recognition
Deep Learning

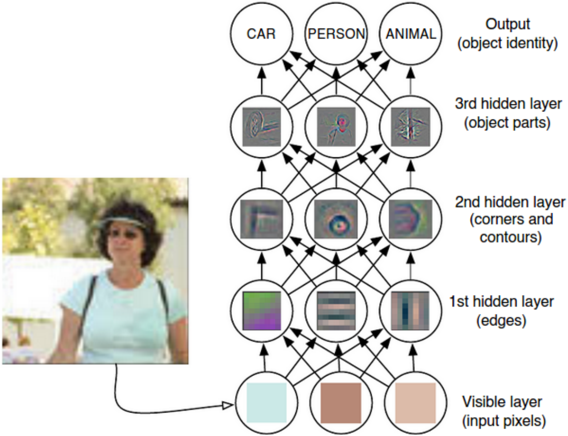
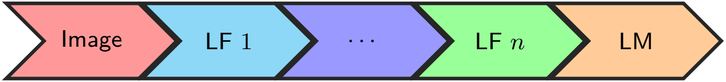


Image from Bengio, Goodfellow, and Courville 2014

Object Recognition
Deep Learning

n levels of features/representations

Learned jointly with the output mapping



Multilayer Perceptrons

DL is usually realized using MultiLayer Perceptrons (MLPs)

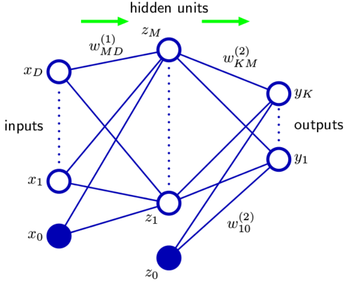


Image from Bishop 2006

Binary linear classifier

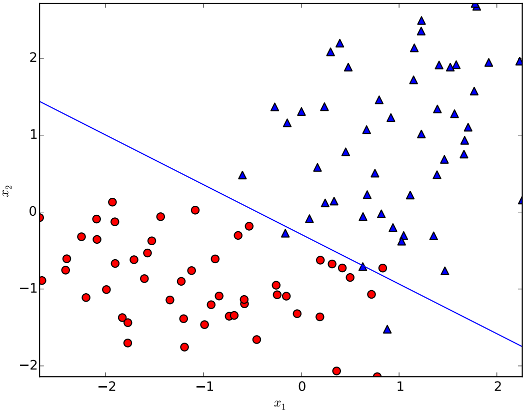
Feature vectors $\mathbf{x}$ classified as $f(\mathbf{w}^\top \mathbf{x} + b) \in \{-1, +1\}$

f is a discontinuous step function

$$f(v) = \begin{cases} +1 & \text{if } v > 0 \\ -1 & \text{otherwise} \end{cases}$$

$\mathbf{w}, b$ learned from training data

Multilayer Perceptrons
The Perceptron



- Only two classes
- Linear decision boundaries
- Learning never converges for non-separable data

Multilayer Perceptrons

Two-Layer Architecture

Replace f with continuous nonlinearity (e.g. $\tanh(\cdot)$)

Introduce layer of M such "Perceptrons" (hidden units)

Hidden units connected to layer of K output units

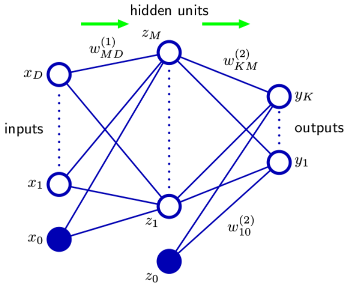


Image from Bishop 2006

Multilayer Perceptrons
Two-Layer Architecture

Output of mth hidden unit is $z_m(\mathbf{x}) = f(\mathbf{w}_m^\top \mathbf{x})$

Bias b included in $\mathbf{w}$ and $\mathbf{x}$, $w_0 = b$, $x_0 = 1$

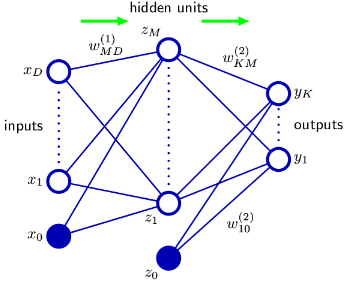


Image from Bishop 2006

Multilayer Perceptrons
Two-Layer Architecture

Output of kth output unit is $y_k(\mathbf{z}) = g(\mathbf{w}_k^\top \mathbf{z})$

Choice of g depends on problem (regression, classification)

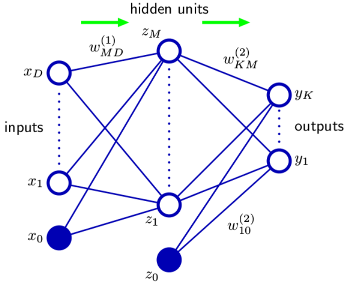


Image from Bishop 2006

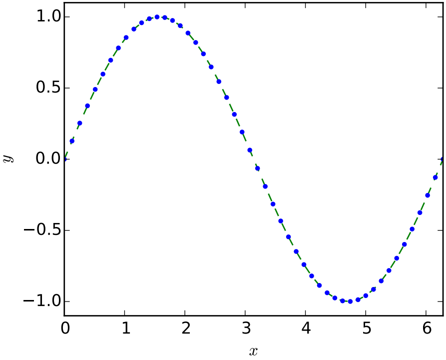
Both f and g are differentiable

- Learn $\mathbf{w}$ using gradient descent
- Gradients evaluated via error backpropagation

The Pylearn2 Library

- Machine learning library with focus on DL
- Written in Python, but interaction mostly in YAML
- Open-source: `https://github.com/lisa-lab/pylearn2`

We will use pylearn2 to train a MLP for regression

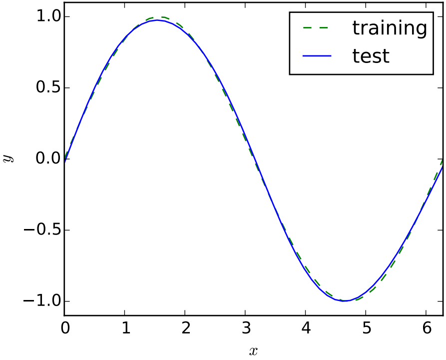


```
model: !obj:pylearn2.models.mlp.MLP {
    nvis: 1, # one input unit x
    layers: [ # two layers
        !obj:pylearn2.models.mlp.Tanh { # tanh activations for hidden units
            dim: 3, # use M=3 hidden units
            layer_name: 'hidden',
            irange: 1
        },
        !obj:pylearn2.models.mlp.Linear { # linear output layer for regression
            dim: 1, # one output unit, K=1
            layer_name: 'out',
            irange: 1
        }
    ]
}
```

The Pylearn2 Library
MLP Regression Example

```
# dataset contains the (x,y) pairs from the previous figure
dataset: &train !pkl: 'mlp_data_regression.pkl',
# train using batch gradient descent
algorithm: !obj:pylearn2.training_algorithms.bgd.BGD {
    conjugate: 1,
    batch_size: 50,
    line_search_mode: 'exhaustive',
    termination_criterion: !obj:pylearn2.termination_criteria.EpochCounter {
        max_epochs: 100 # train for 100 epochs
    }
}
```

Full example: `https://github.com/cpra/cvsp-vo-slides`



Bibliography I

Bengio, Yoshua, Ian Goodfellow, and Aaron Courville (2014). **Deep Learning (draft)**. MIT Press.

Bishop, Christopher (2006). **Pattern recognition and machine learning**. Springer.

LeCun, Yann et al. (1989). **Backpropagation applied to handwritten zip code recognition**. Neural computation.

Taigman, Yaniv et al. (2013). **Deepface: Closing the gap to human-level performance in face verification**. CVPR.