Basics of Machine Learning and Neural Networks

Turku AI Week 02/2018

VINCIT

Your lecturer

- Ilmari Ahonen, (PhD 3.3.2018)
- Data analyst at Vincit
- Prior work experience: Avoltus, University of Turku, VTT and Roche
- Statistics, machine learning and mathematics
- R + Python

Machine Learning Basics

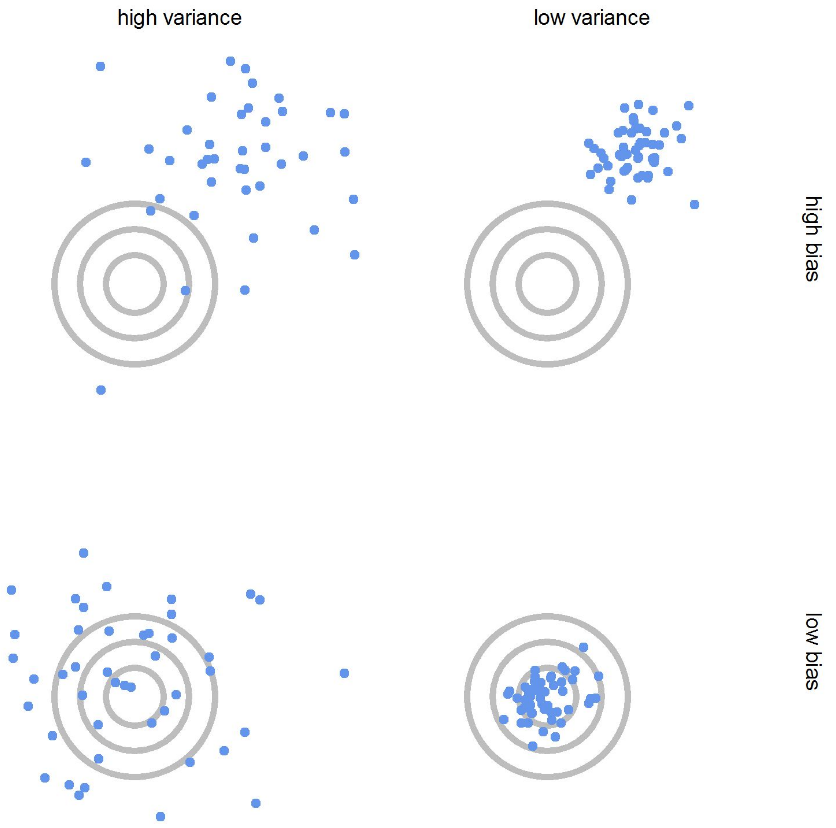
Data-based Methods

The methods discussed today are characterized by two things:

- The patterns/rules/decisions/estimates are learned from data
- The used models are not deterministic in nature but contain an aspect of uncertainty

Bias and Variance

- **Bias:** systematic error
- **Variance:** random error



Statistical learning

Accept some bias for simpler models

Try to minimize bias for accurate predictions

Inference

Relationships between variables

Theoretical foundations

Modeling

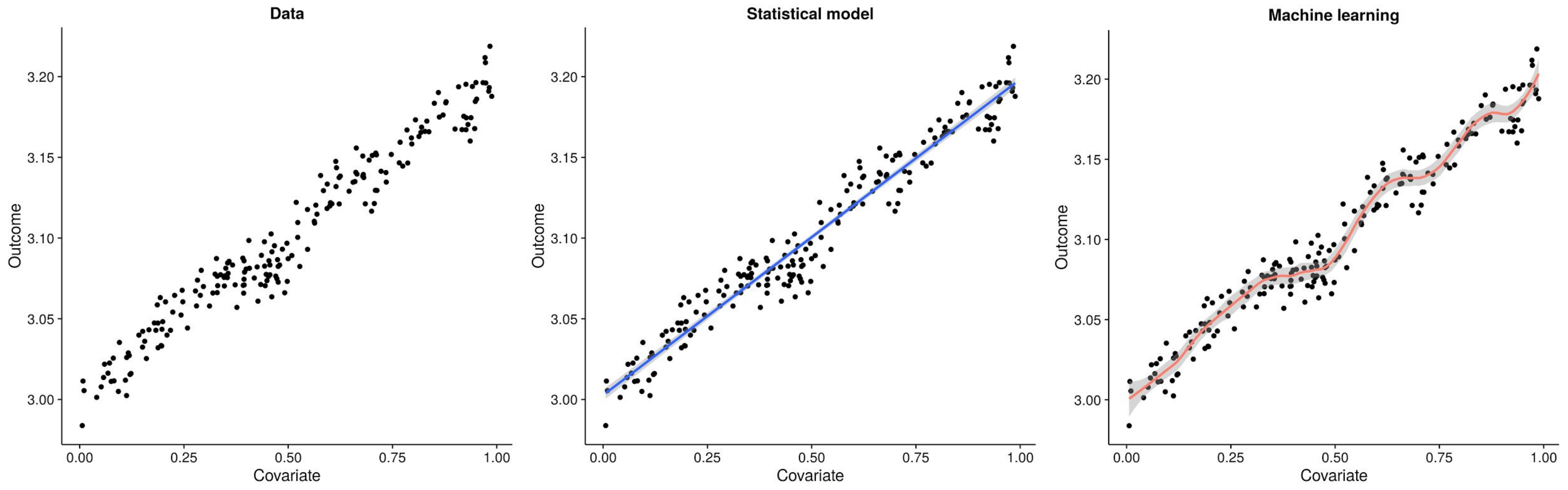
Feature extraction

Prediction

Cross-validation

Black box -models

Statistics ⟷ *Machine Learning*


Data
Statistical model
Machine learning
Outcome
Covariate
3.00
3.05
3.10
3.15
3.20
0.00
0.25
0.50
0.75
1.00

Supervised vs unsupervised

Supervised:

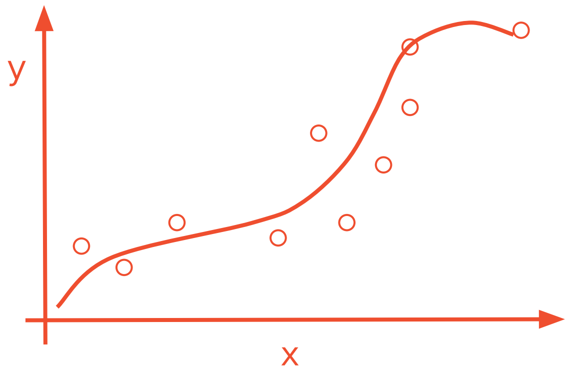
- Learning by examples
- Find prediction for Y based on explanatory variables X_1 , X_2, …, X_N
- Y: response, outcome, output
- X: inputs
- Most of machine learning is supervised

Unsupervised:

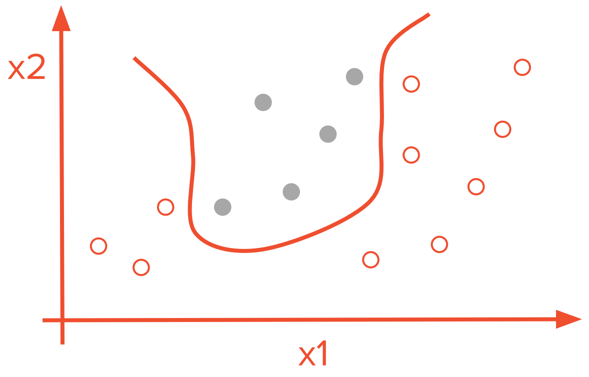
- All variables are outcomes
- Some common goals:
    - Grouping similar variables
    - Grouping similar observations
    - Reducing the data dimension

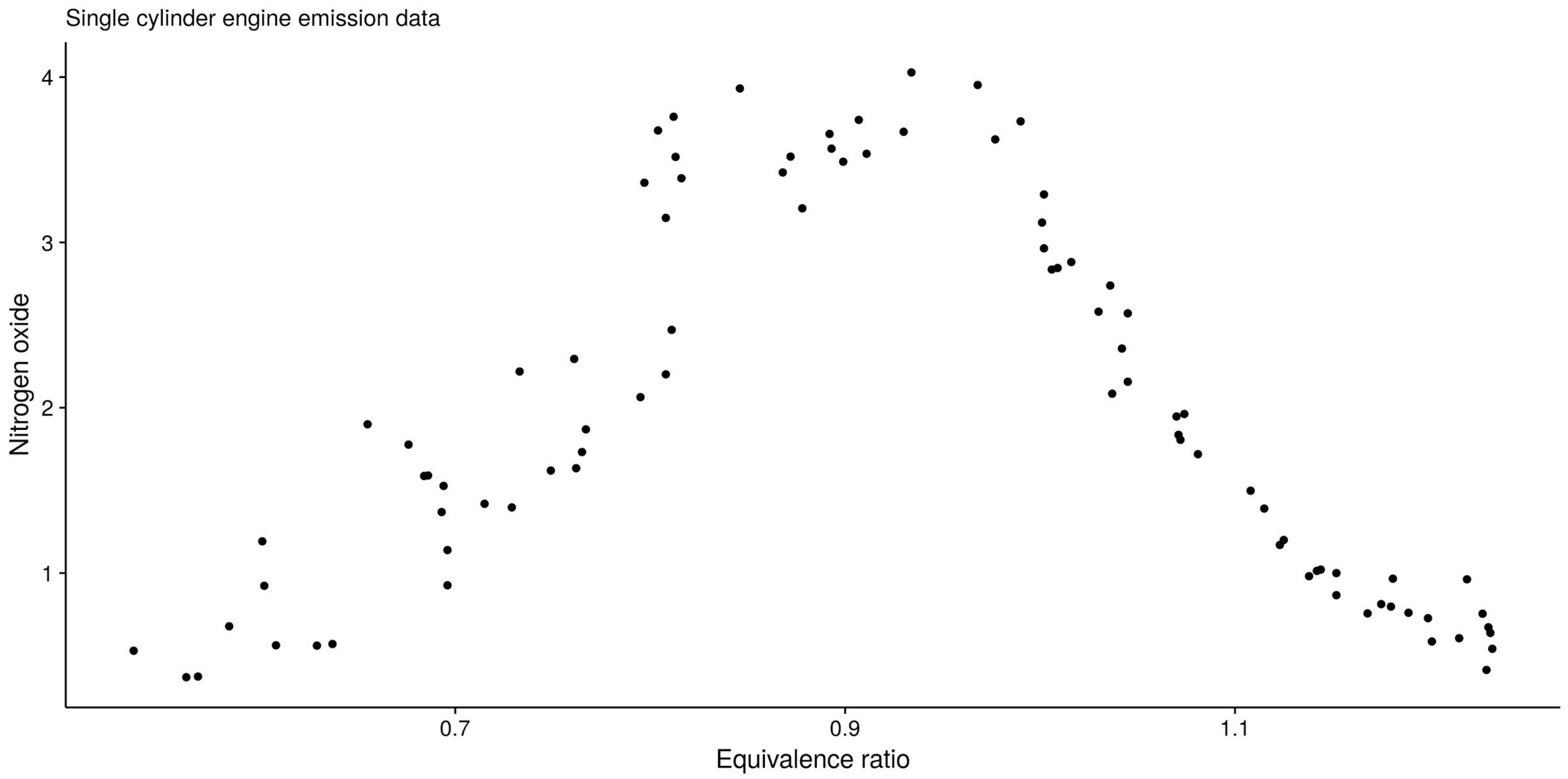
Regression vs classification

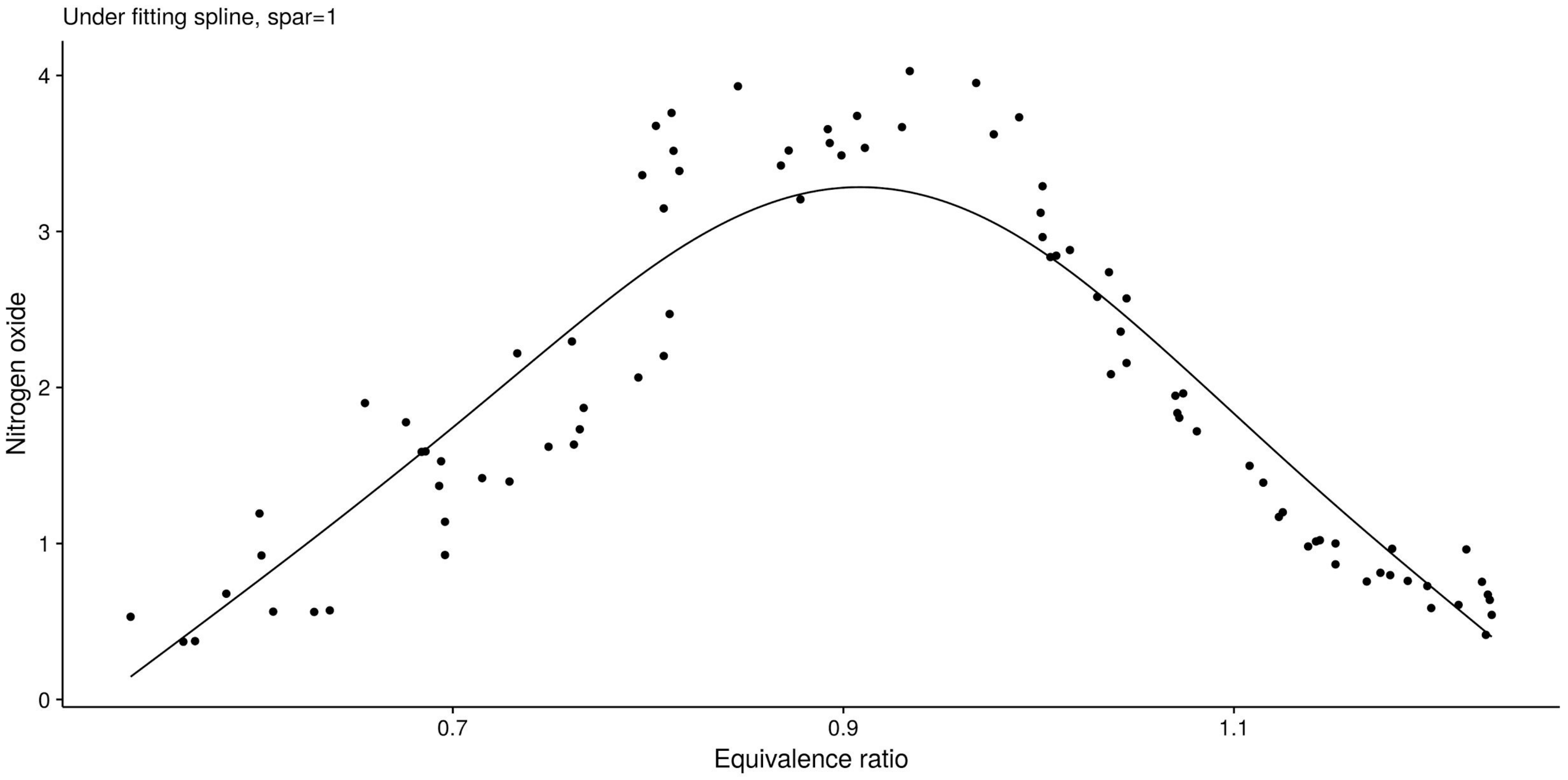
Regression

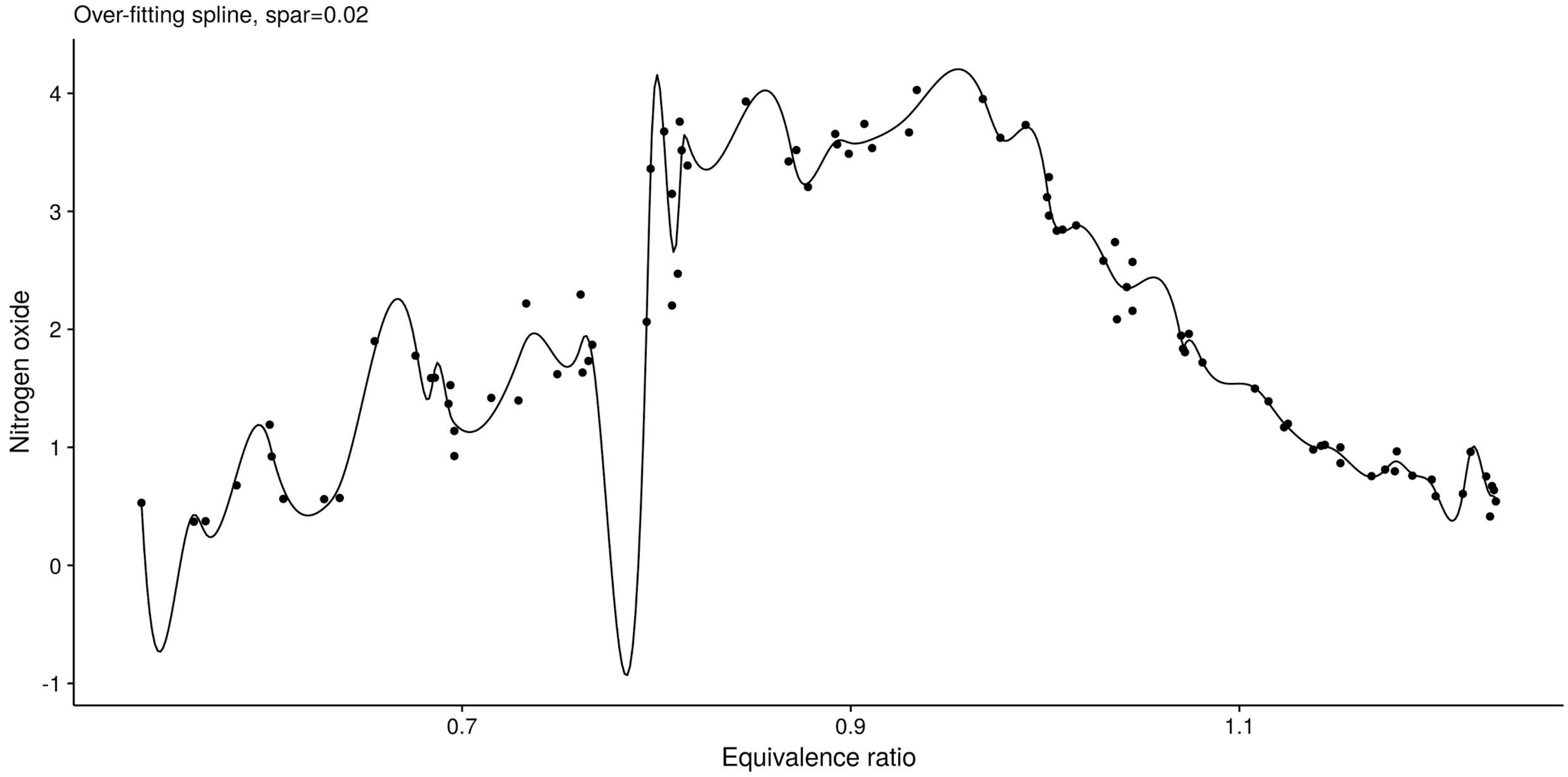


Classification



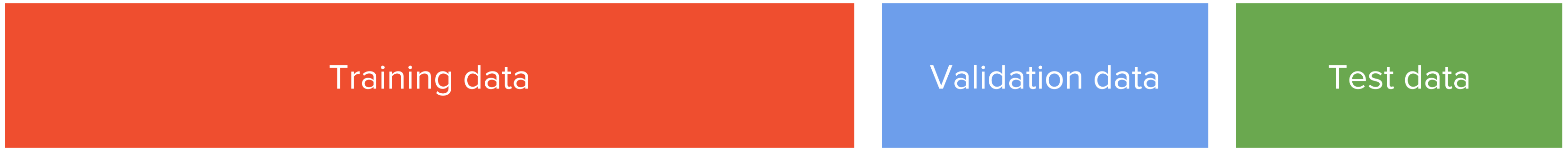

Single cylinder engine emission data
Nitrogen oxide
4
3
2
1
0.7
0.9
1.1
Equivalence ratio


Under fitting spline, spar=1
Nitrogen oxide
0
1
2
3
4
0.7
0.9
1.1
Equivalence ratio

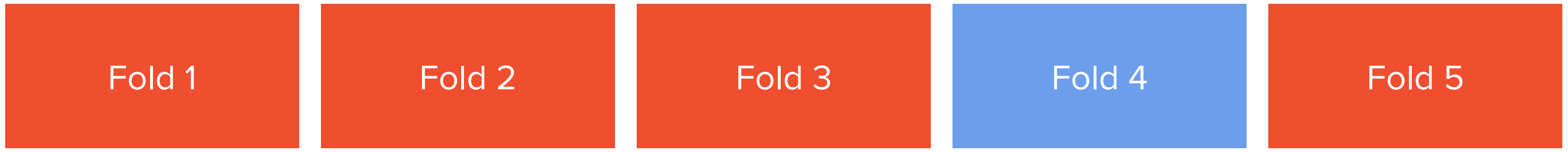

Over-fitting spline, spar=0.02
Nitrogen oxide
4
3
2
1
0
-1
0.7
0.9
1.1
Equivalence ratio

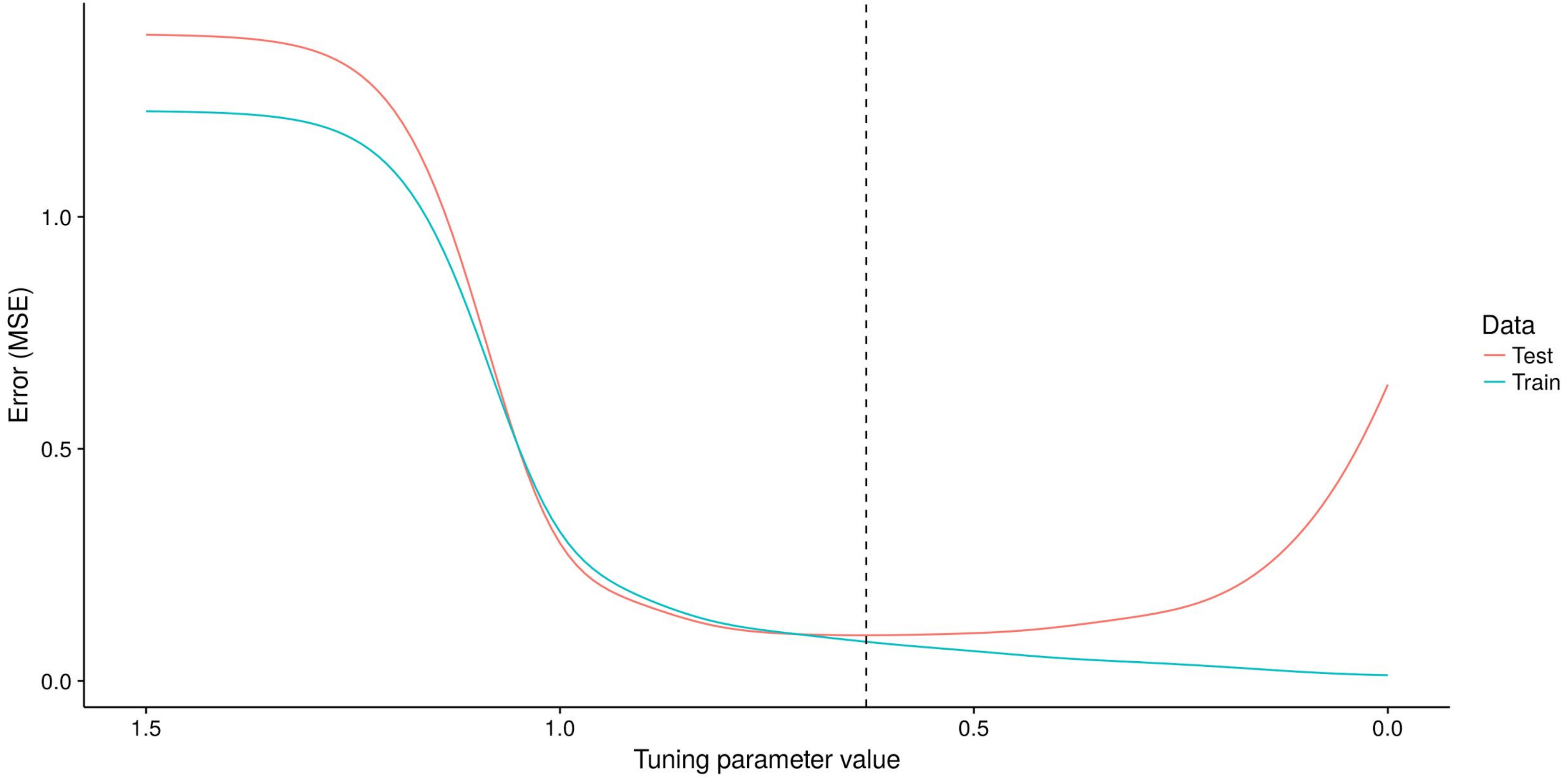
Cross-validation

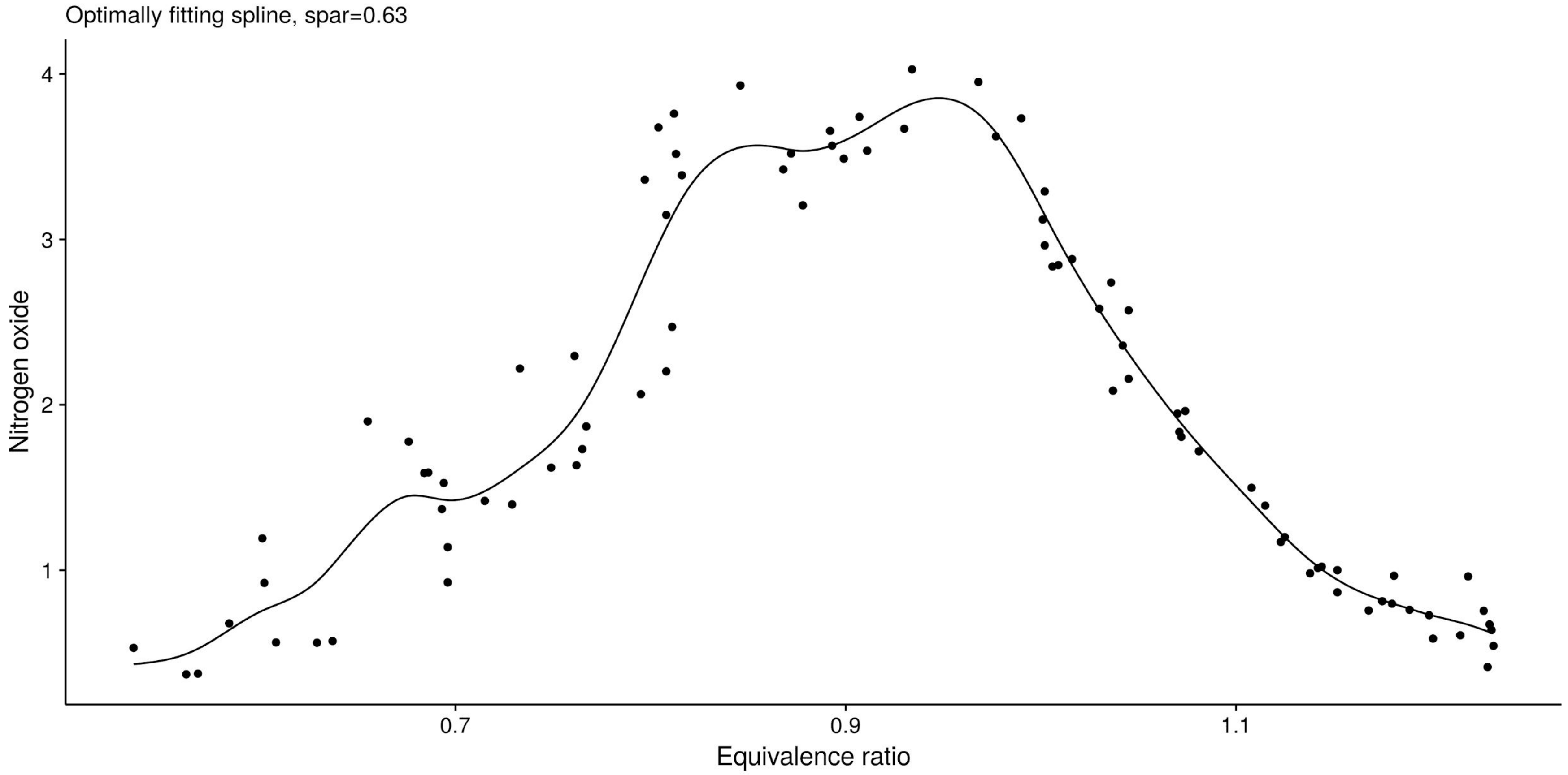
Traditional cross-validation scheme



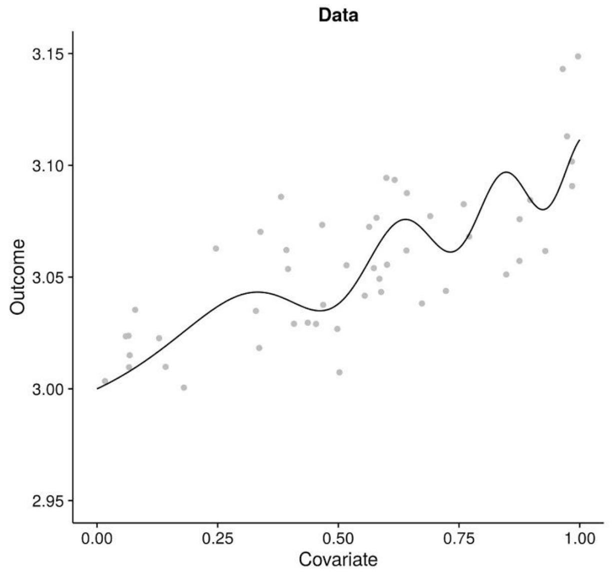
5-fold cross-validation

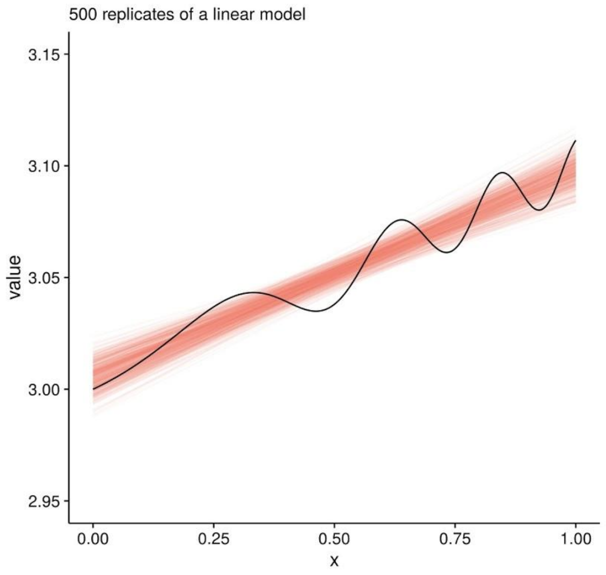


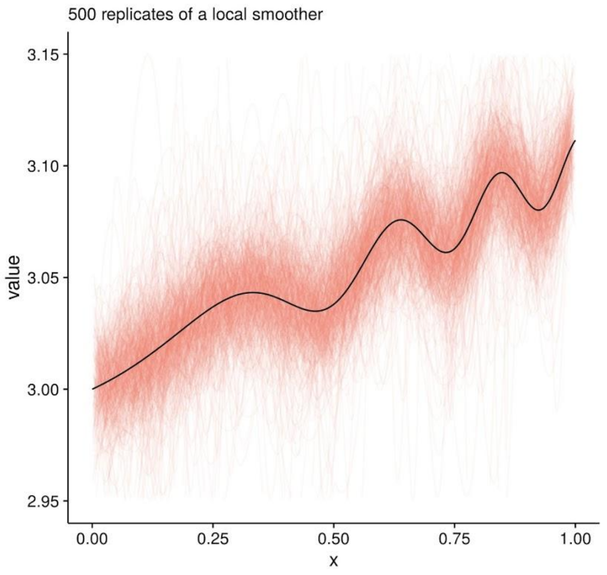

Error (MSE)
1.0
0.5
0.0
1.5
1.0
0.5
0.0
Tuning parameter value
Data
Test
Train


Optimally fitting spline, spar=0.63
Nitrogen oxide
4
3
2
1
0.7
0.9
1.1
Equivalence ratio

Local vs global model







Why not always use local methods?

The answer: curse of dimensionality. When $p >> 0$:

- Local data are no longer in local regions:
  - Consider uniformly sampled data within a 10 dimensional hypercube. By splitting each dimension in half, one catches only $(1/2)^{10} < 0.1\%$ of the data
- All data tend to be located near the border of the variable space:
  - Consider 500 observations uniformly sampled within a 10-dimensional unit ball. Most observations are likely to be closer to the balls boundary than any other data point -> a prediction model needs to extrapolate instead of interpolating
- Required sample size grows exponentially:
  - If 100 observations is deemed sufficient for 1 variable, one needs 100^{10} observations for the same sampling density with 10 variables

Summary

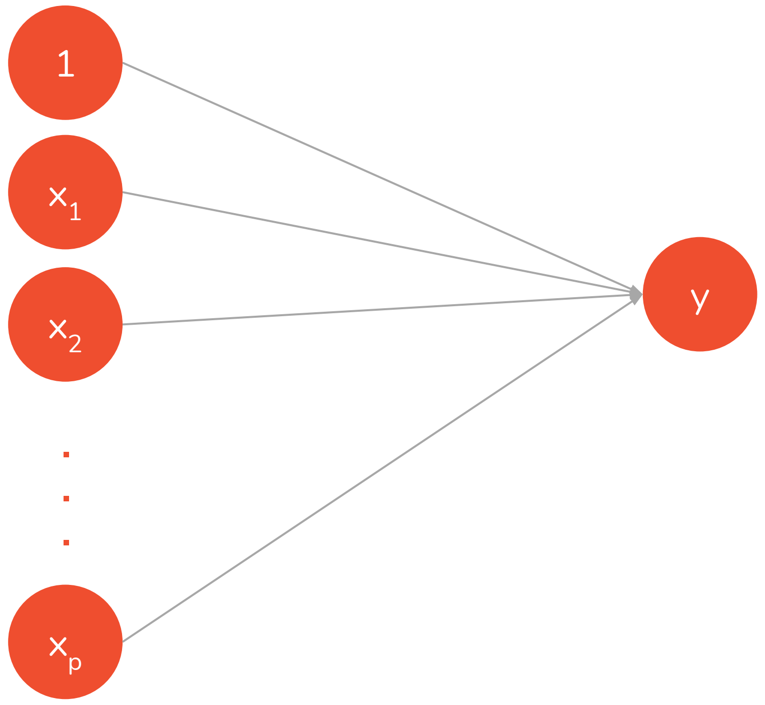
- Balancing between bias and variance
- Supervised methods = learning by example
- Cross-validation is used to avoid overfitting
- Curse of dimensionality breaks local methods

Neural Networks and Deep Learning

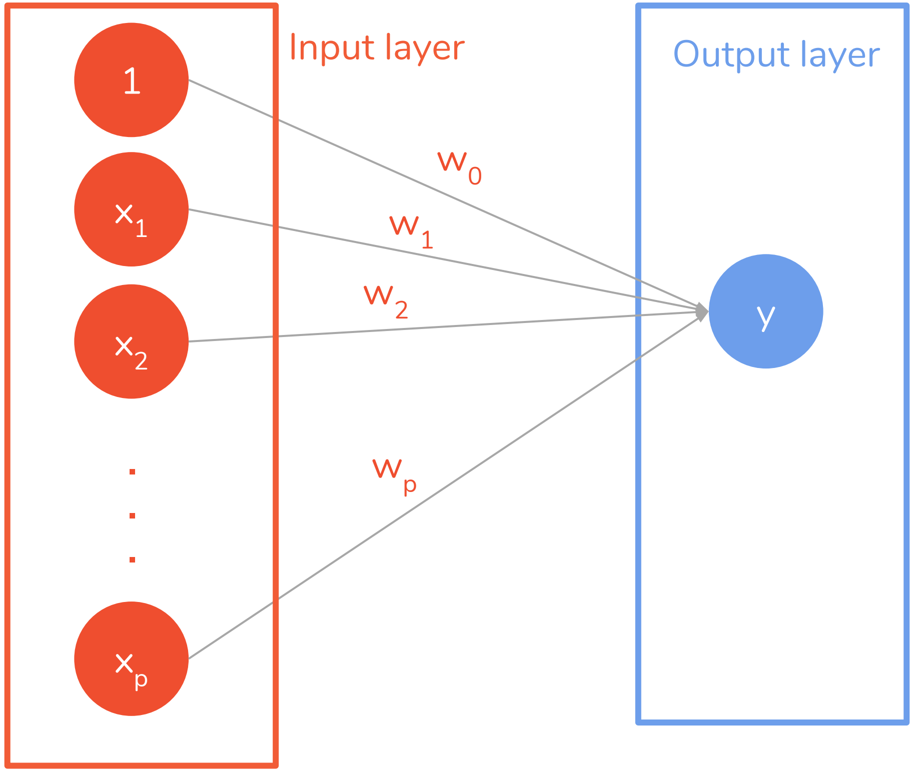
Neural networks

- "The" machine learning method. Was much hyped in the 80s but was later slightly shadowed by ensemble methods and SVM. Is now experiencing a renaissance along with the breakthroughs in computational capabilities and deep learning
- The terminology originates from the attempt at mimicking the human brain
- Neural networks are essentially nonlinear statistical models
- The network can be represented as a directed graph

The simple case



The simple case

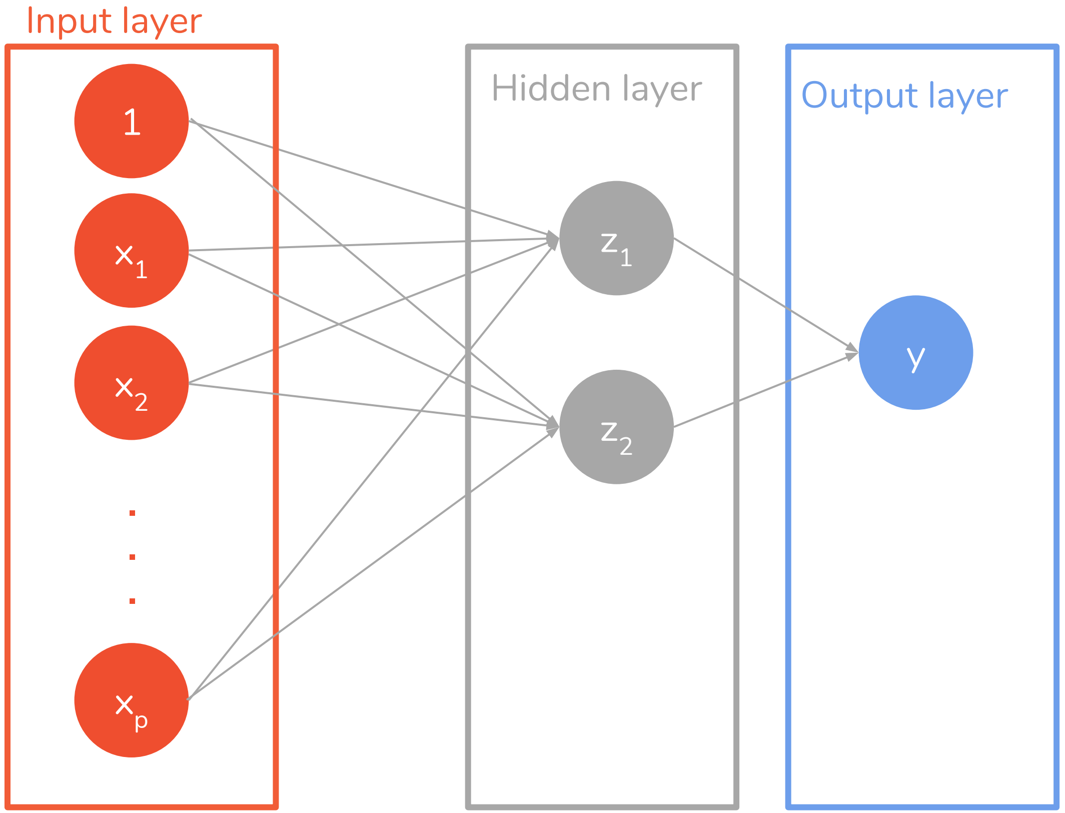


The prediction for y is formed as a function of the linear combination of the inputs:

$$\hat{y}=g(w_0+\sum_i w_i x_i),$$

where w_i are the model **weights** and g is often referred to as an **output function**

The basic idea

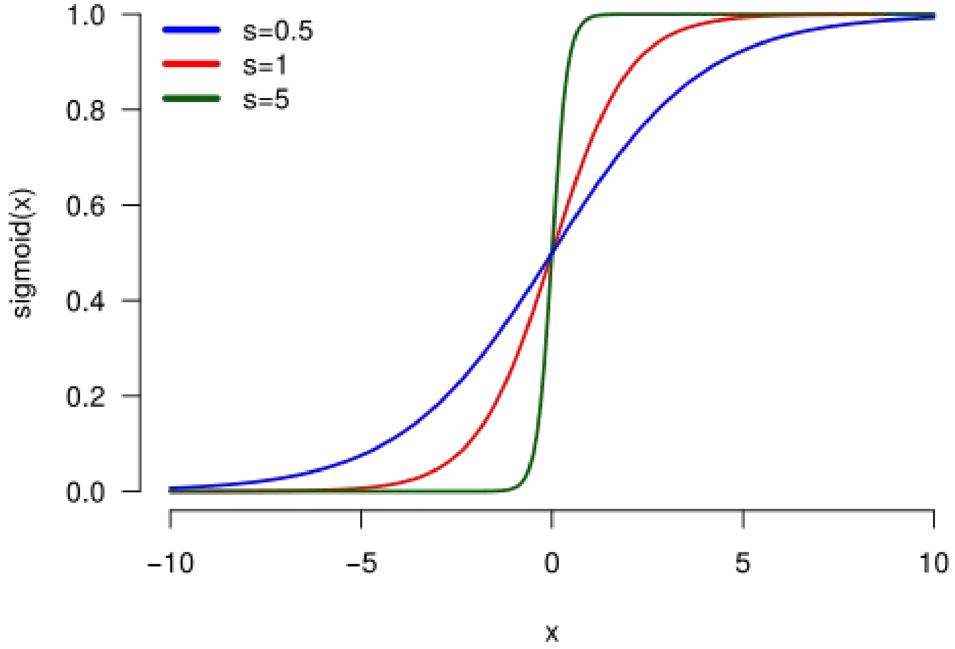


The flexibility of the neural network is greatly increased by adding a **hidden layer** of nodes between the inputs and outputs

Activation function

The weighted signals coming to a hidden node are passed through an activation function (similar to the output function in the output node). A sigmoid function is the classical choice:

$$\sigma_s(x) = 1/(1 + \exp(-sx))$$



Training neural networks

Whatever model we choose, we aim at minimizing the prediction error i.e. the loss function. The measure of loss again depends on the type of the outcome. The usual suspects are:

- Squared errors for regression
- log-loss, deviance or cross-entropy for classification

The goal is to find the set of values for the weights that minimize the loss.

Training neural networks

- Since the network can be expressed as one compact, non-linear function of the parameters, we can base the minimization on the usual multivariate function analysis using gradients. However, the related Hessian matrix is often very large, which renders, for instance, the Newton's method infeasible.
- The traditional way to solve this problem is the gradient descent algorithm often referred to as back-propagation.
- For unregularized networks, the algorithm can be stopped before the global minimum is found in order to avoid overfitting. This is called early stopping.

Back-propagation algorithm

In a nutshell:

- essentially a gradient descent algorithm
- update the weights towards the minimum loss based on a learning parameter
- The algorithm:
    - Calculate predicted value using current weights (forward pass)
    - Calculate “errors” in the input layer and use these to calculate the errors in the hidden layer (backward pass)
    - Update the weights based on the gradient and repeat

The initial weights are usually randomly assigned, unless some context knowledge exists

Weight decay

Regularization method similar to ridge regression:

Let R(**w**) be the loss associated with a vector of weights **w**. Instead of minimizing just R(**w**), a penalty term J(**w**) = $\Sigma_i w_i^2$ is introduced and the full loss function becomes

$$R(\mathbf{w}) + ƛ\, J(\mathbf{w})$$

Here, ƛ becomes a tuning parameter that controls the flexibility of the model

Things to consider

The starting weights are usually set to be random values close to zero. Large starting weights often produce poor results.

The weights are directly related to the scale of the inputs in the first layer. It is a good idea to standardize the inputs prior fitting

Online demo:

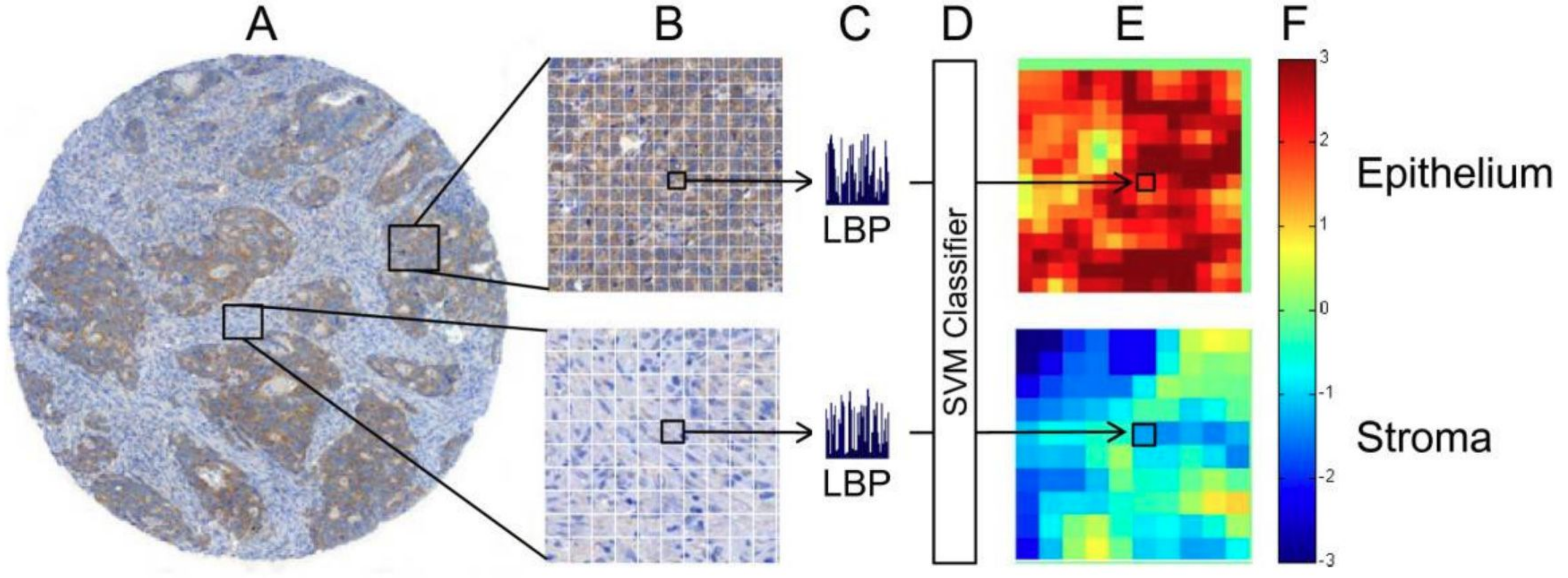
http://playground.tensorflow.org/

On the way to deep learning

Two things characterize deep learning:

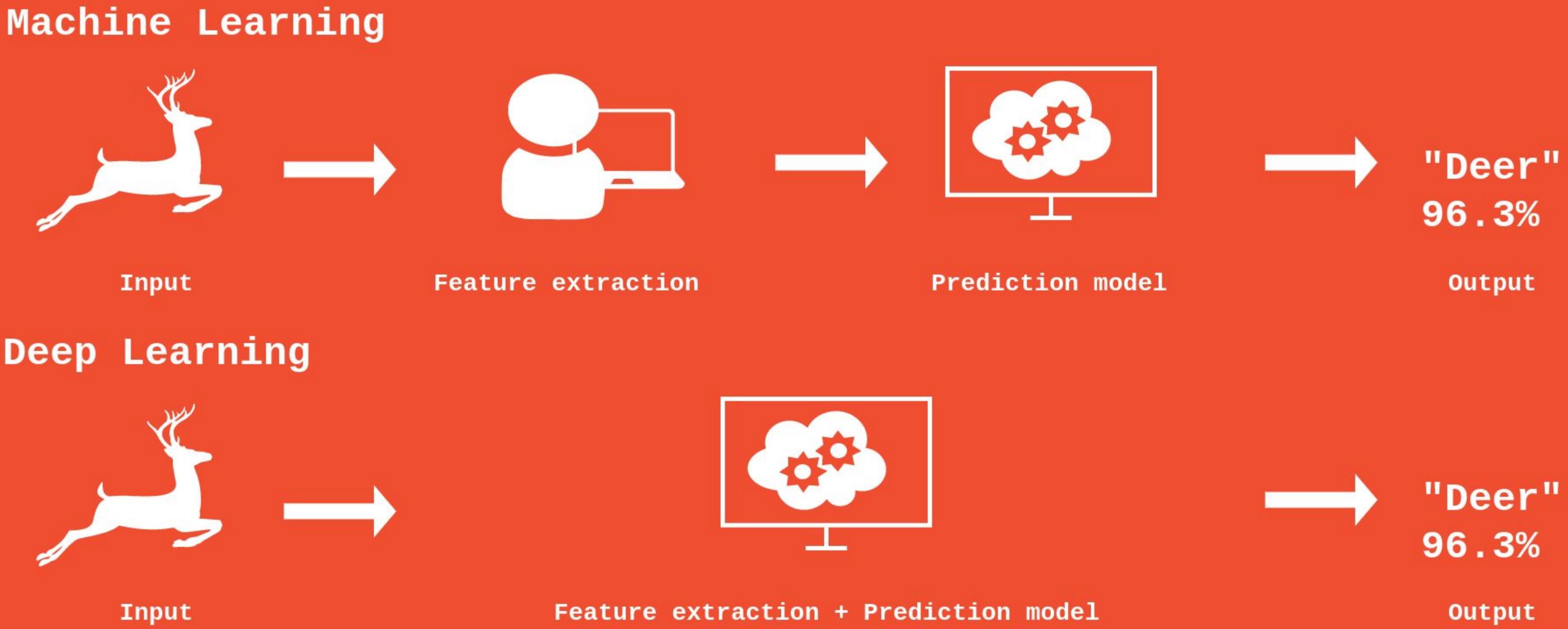
- Deep neural networks, meaning multiple layers of nodes
- Raw data as input; features are learned based on data

Feature extraction



Linder et al. (2012)

Deep Learning vs Machine Learning



Examples of layers

Fully connected: All inputs from previous layer connect to all nodes in the current layer

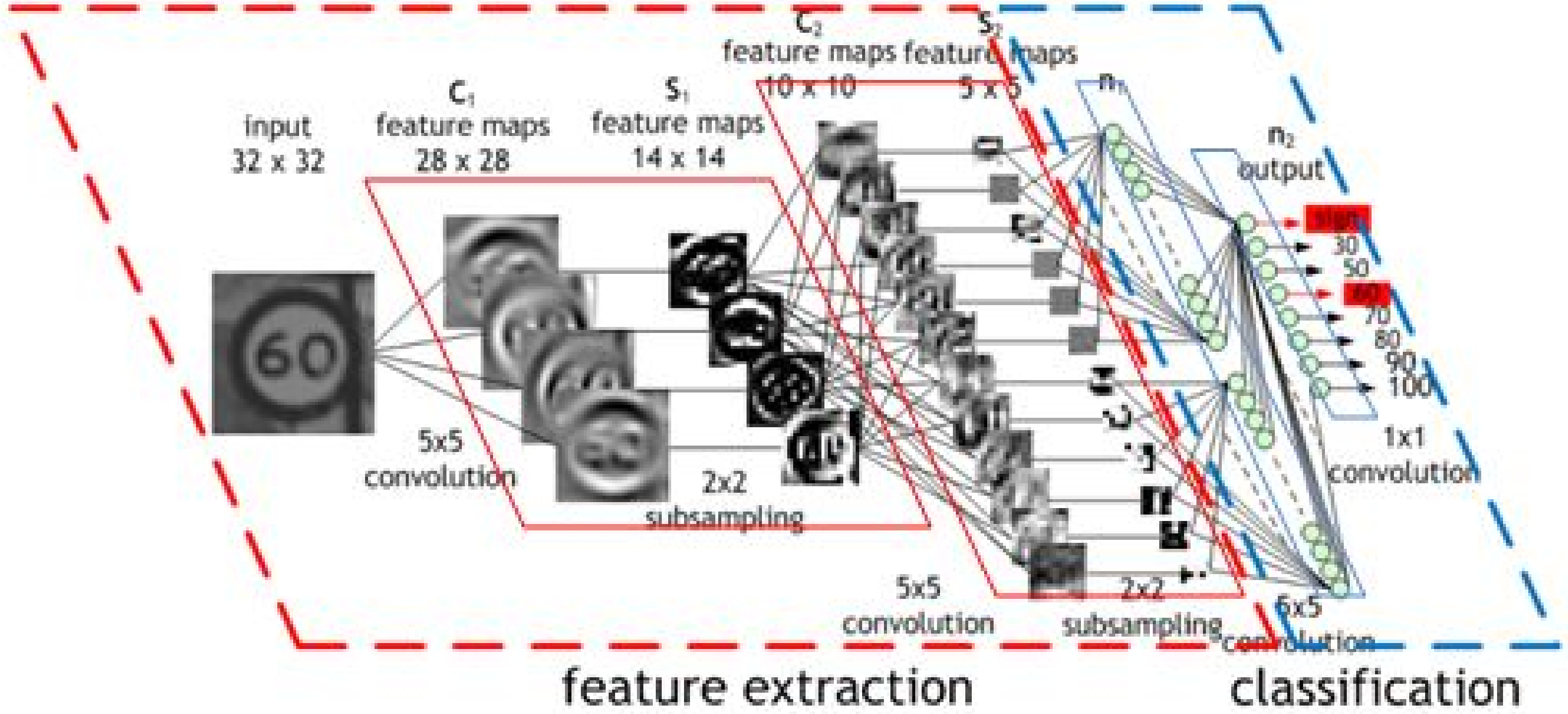
Convolution: https://github.com/vdumoulin/conv_arithmetic

Embedding: Maps a categorical input to R^p

Max-pooling: Selects the maximum from a set of inputs

Subsampling: Pools sets of inputs together

Recurrent layers: Previous observations are fed as inputs to the next observation

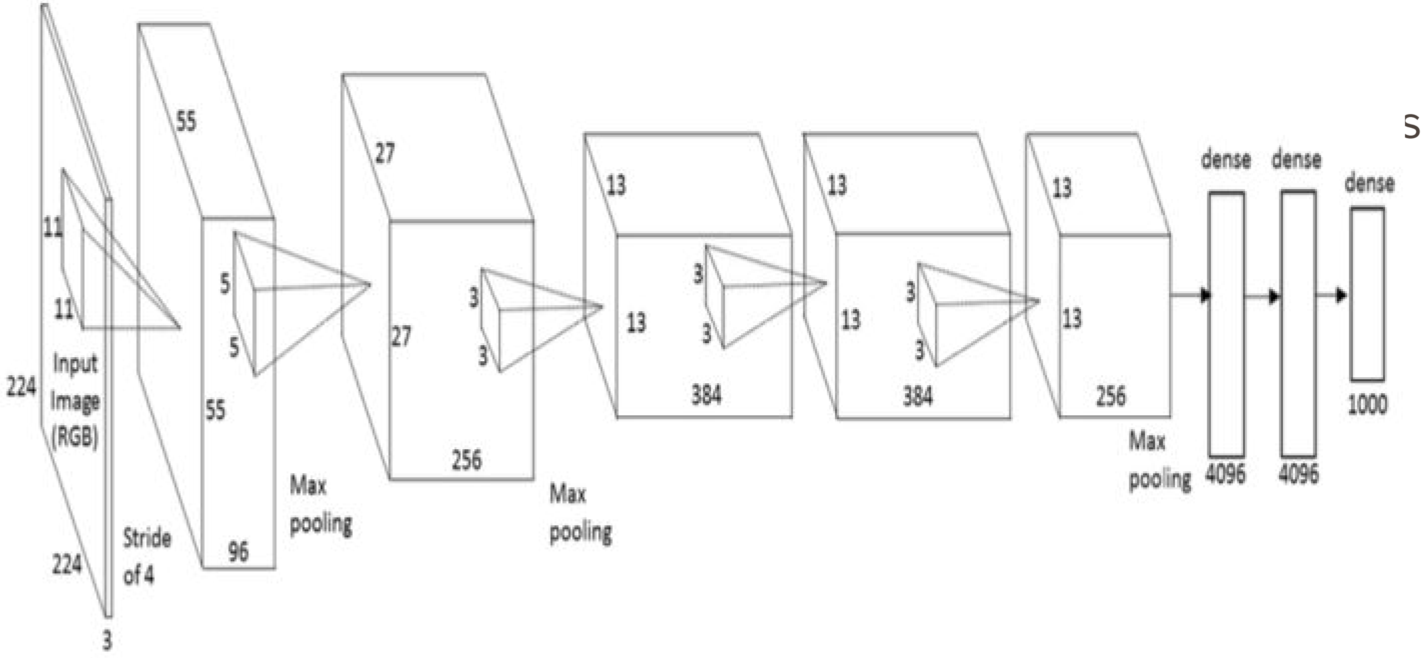


Credits to [Maurice Peemen]

Deep learning practices

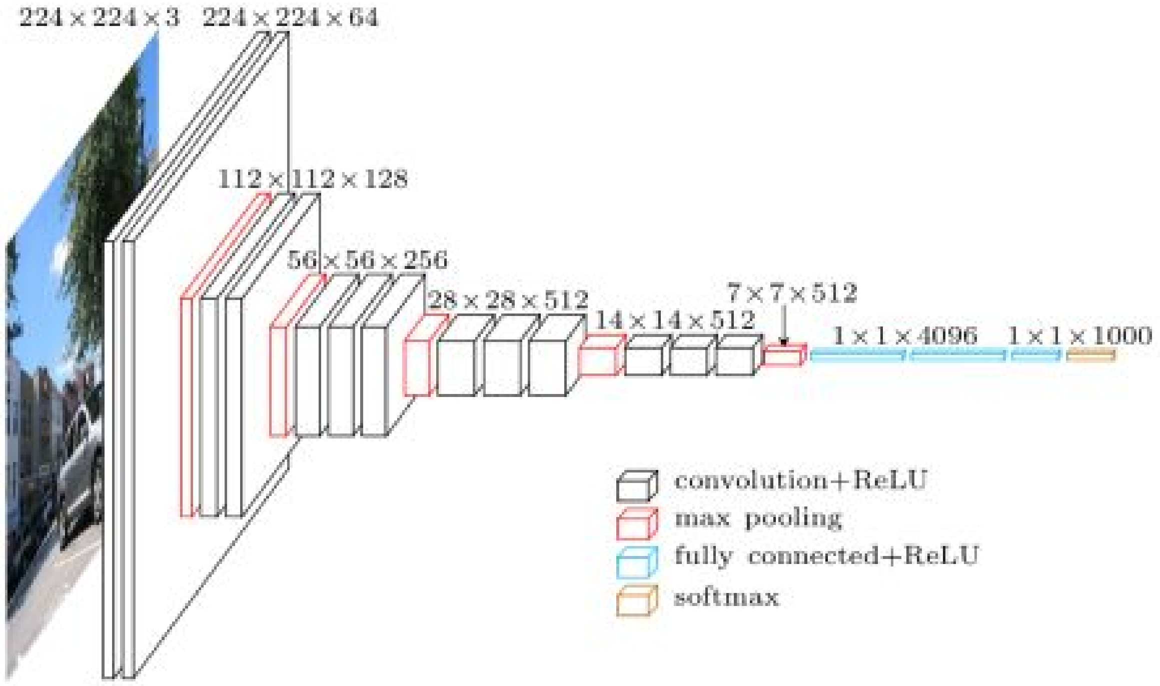
- Rectified linear unit (**ReLU**) is the preferred choice for activation function in deep networks
- Trained using stochastic gradient descent (SGD) in batches of data
- **Dropout** is used for regularization and to mimic ensemble methods. Can also be used for estimating prediction uncertainty.
- Models are designed on a higher level of abstraction, where instead of individual nodes, one considers different layers
- Different types of layers are utilized depending on the problem at hand

Examples of Deep Learning



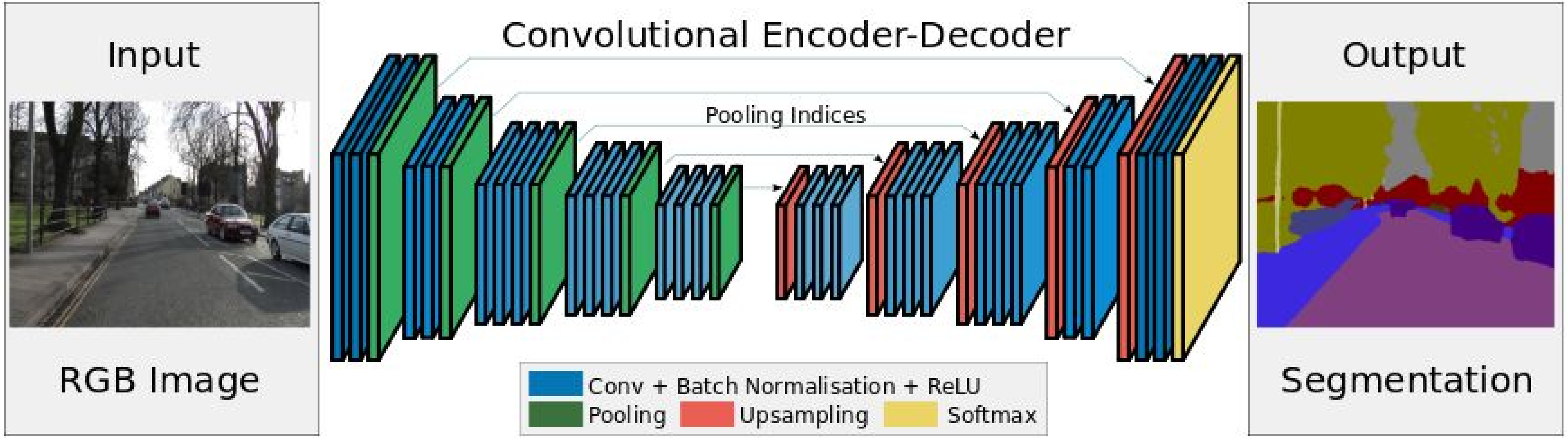
AlexNet (Krizhevsky, Hinton, Sutskever 2012)

Examples of Deep Learning



VGGNet (Simonyan and Zisserman 2014)

Examples of Deep Learning



SegNet (Badrinarayanan, Handa, Cipolla 2015)

Online demo:

http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

Other points

- How to quantify uncertainty? Bayesian paradigm to the rescue?
- How to handle missing values?
- Most computation in deep learning involves linear algebra. GPUs are specialized for such tasks

Practical issues

- How to manage continuous learning?
- How to limit technical debt?
- How to account for sudden changes in the data behavior?
- How to avoid bias when the model's decisions influence the future data?

Thanks for listening!

Feel free to contact me at
ilmari.ahonen@vincit.fi

VINCIT