

# Other Nonlinear Supervised Learning Methods

Johanni Brea

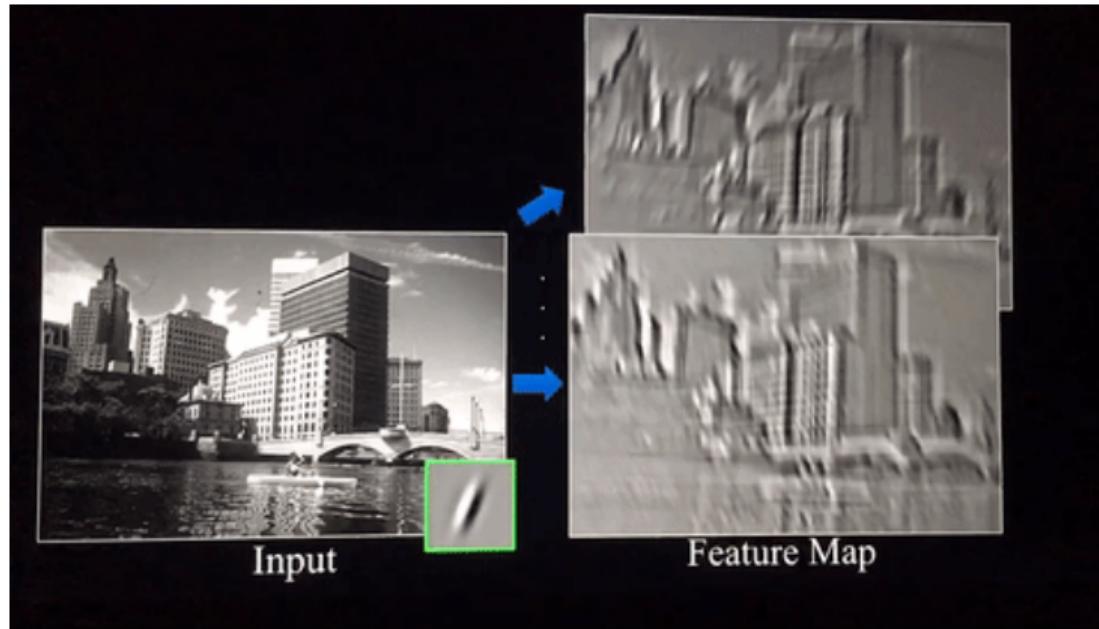
Einführung Machine Learning

GYMINF 2022

## Table of Contents

- 1. Convolutional Neural Networks
  - 2. Transfer Learning
  - 3. Recurrent Neural Networks
  - 4. Tree-Based Methods: Random Forests and Gradient Boosting Trees
  - 5. Support Vector Machines
  - 6. Supervised Learning: the Big Picture

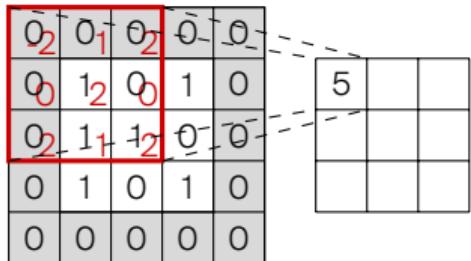
# Reusing Local Features: Convolutions



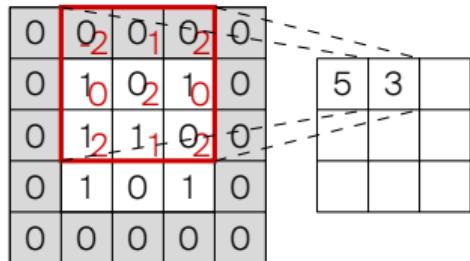
# Convolution

**filter size =  $3 \times 3$ , stride = 1, padding = 1**

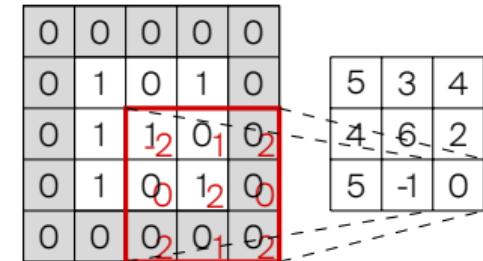
padded image      filtered image



padded image      filtered image

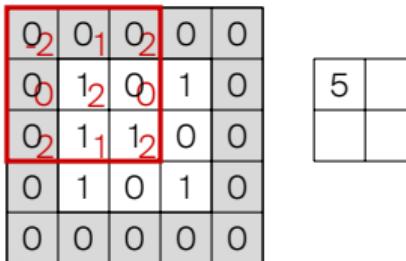


padded image      filtered image



**filter size =  $3 \times 3$ , stride = 2, padding = 1**

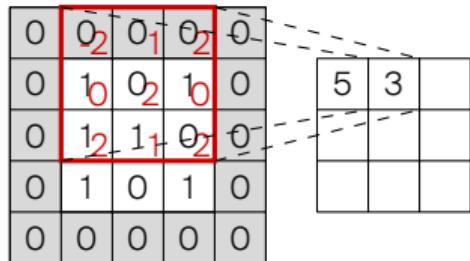
padded image



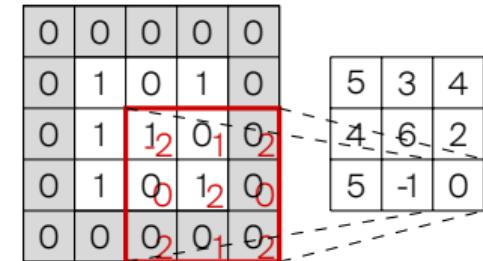
filtered image



padded image      filtered image



padded image      filtered image



# Learning MNIST Features with a Convolutional Neural Network

## Convolutional Neural Network

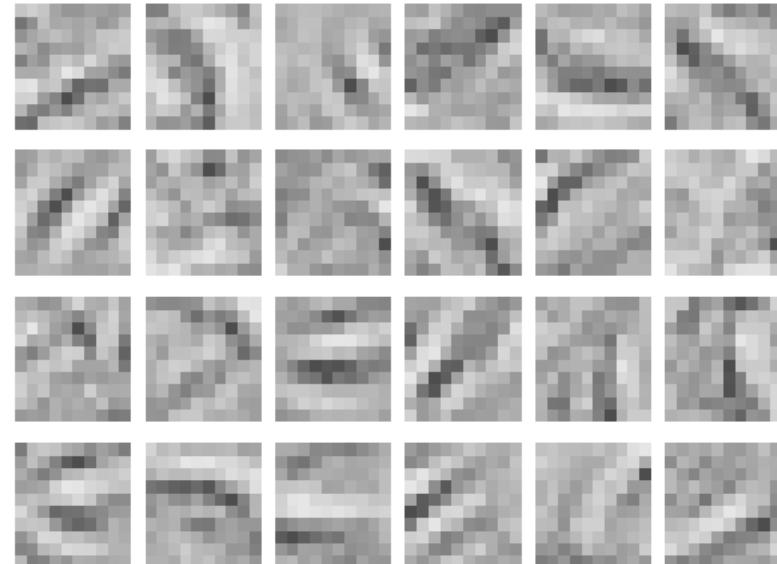
28×28 input neurons

1 hidden layer with 200 relu conv. filters

10 softmax neurons

Test Misclassification Rate: < 1%

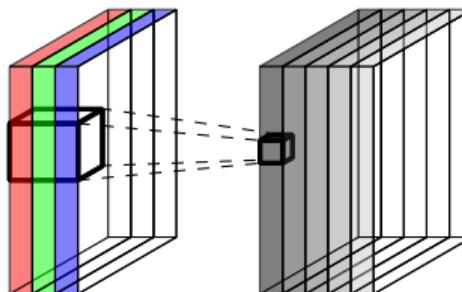
Examples of learned features



# Convolution Volumes

From 3 (colour) channels to 5 (filter) channels

colour image multi-filtered image



Input volume  $n \times n \times c$  leads to output volume

$$\left( \frac{n + 2p - f}{s} + 1 \right) \times \left( \frac{n + 2p - f}{s} + 1 \right) \times k.$$

padding  $p$ , filter size  $(f, f)$ , stride  $s$ , number of filters  $k$

# Convolutional Layer

A convolutional layer  $I$  with  $K^{(I)}$  filters of size  $(2f + 1, 2f + 1)$ , padding  $f$  and stride 1 computes

$$a_{xy,k}^{(I)} = g^{(I)} \left( w_{k0}^{(I)} + \sum_{q=-f}^f \sum_{r=-f}^f \sum_{s=1}^{K^{(I-1)}} w_{qrs,k}^{(I)} a_{x+q,y+r,s}^{(I-1)} \right),$$

where  $k = 1, \dots, K^{(I)}$ ,  $a_{x+q,y+r,s}^{(I-1)}$  are the activities in the previous layer,  $w_{k0}^{(I)}$  is the bias of filter  $k$ ,  $w_{qrs,k}^{(I)}$  are the weights of filter  $k$ .

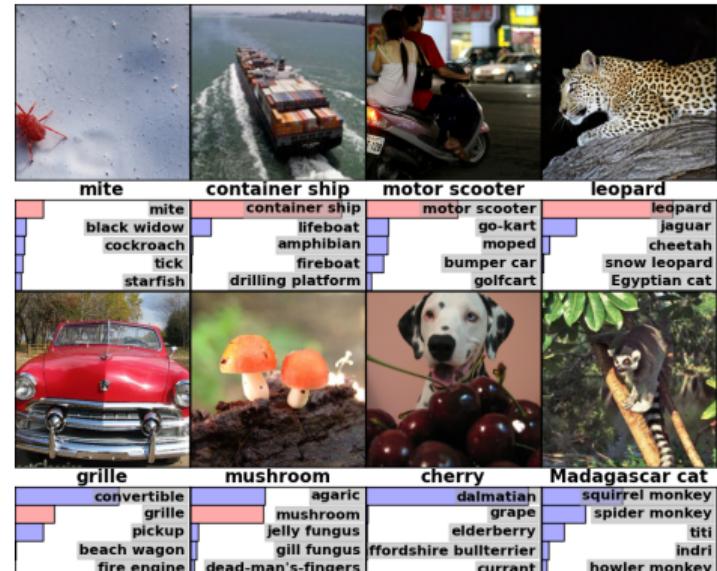
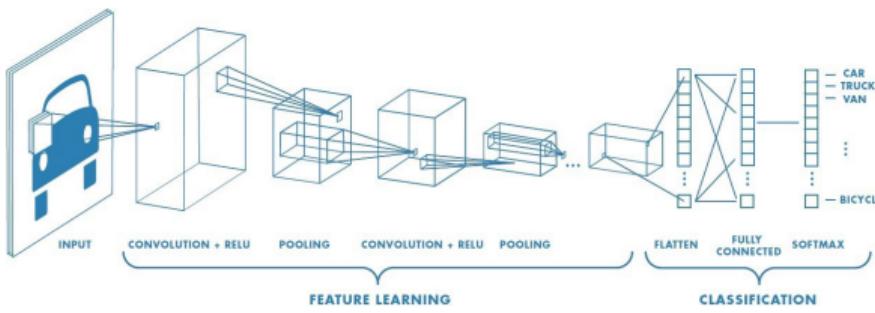
# Convolutional Networks

winner of imagenet competition 2012

input: colour images  $224 \times 224 \times 3$

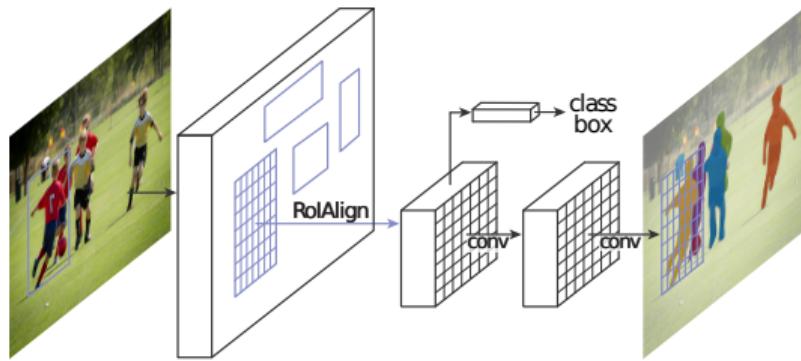
output: 1000 classes

hidden layers: 5 convolutional layers,  
3 dense layers



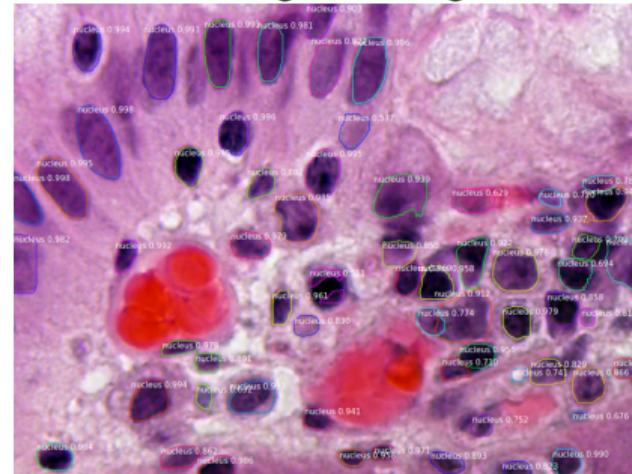
ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky, Sutskever, Hinton, 2012

# Convolutional Networks: Mask R-CNN



$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{class}} + \mathcal{L}^{\text{box}} + \mathcal{L}^{\text{mask}}$$

## Nuclei Counting and Segmentation



# Table of Contents

1. Convolutional Neural Networks

2. Transfer Learning

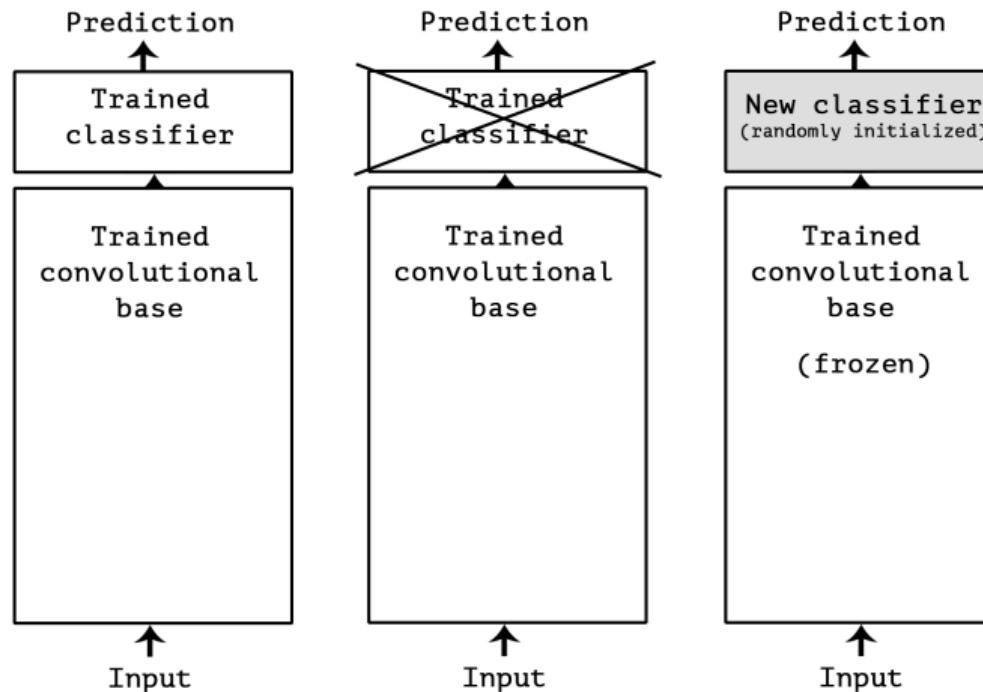
3. Recurrent Neural Networks

4. Tree-Based Methods: Random Forests and Gradient Boosting Trees

5. Support Vector Machines

6. Supervised Learning: the Big Picture

# Transfer Learning: Reusing Learned Features



# Transfer Learning: Reusing Learned Features

## Training Set for Features

Imagenet Dataset: more than 1 million images 1000 categories



## Training Set for Classifier

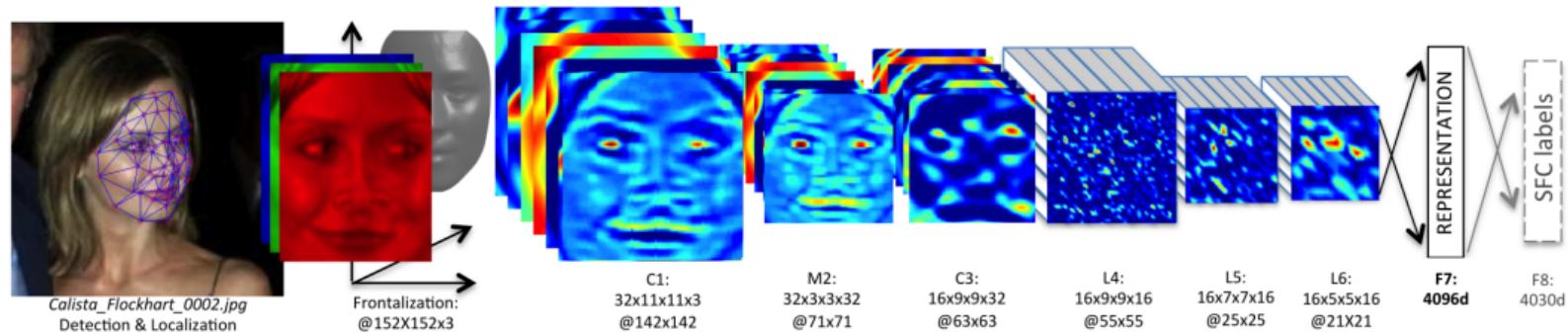
102 Category Flower Dataset ( $\approx 10'000$  images)

### 102 Category Flower Dataset

Category	#ims	Category	#ims	Category	#ims
	43		71		40
	105		102		162
	78		91		166
	96		82		91

# Transfer Learning: Reusing Learned Features

DeepFace: Closing the Gap to Human-Level Performance in Face Verification (2014)



- ▶ 4 million user-labeled faces on FaceBook images of 4000 individuals
- ▶ Retrain fully-connected layers at the top on Labeled Faces in the Wild (LFW) dataset reaching (human level) accuracy of 97.35%

# Table of Contents

1. Convolutional Neural Networks

2. Transfer Learning

3. Recurrent Neural Networks

4. Tree-Based Methods: Random Forests and Gradient Boosting Trees

5. Support Vector Machines

6. Supervised Learning: the Big Picture

# Recurrent Neural Networks(RNN)

Let us assume time dependent inputs  $x(t) = (x(0), x(1), \dots, x(T))$ .

In a recurrent neural network the activity  $a^{(l)}(t)$  of the hidden neurons at time  $t$  does not only depend on the input  $x(t)$  at  $t$ , but also on the previous hidden activity  $a^{(l)}(t-1)$ , e.g.

$$a^{(l)}(t) = g^{(l)}(w^{(l)} a^{(l-1)}(t) + w^{(l, \text{rec})} a^{(l)}(t-1) + b^{(l)})$$

# Application of a Recurrent Neural Network for Language Detection

inputs: sentences, output: language label

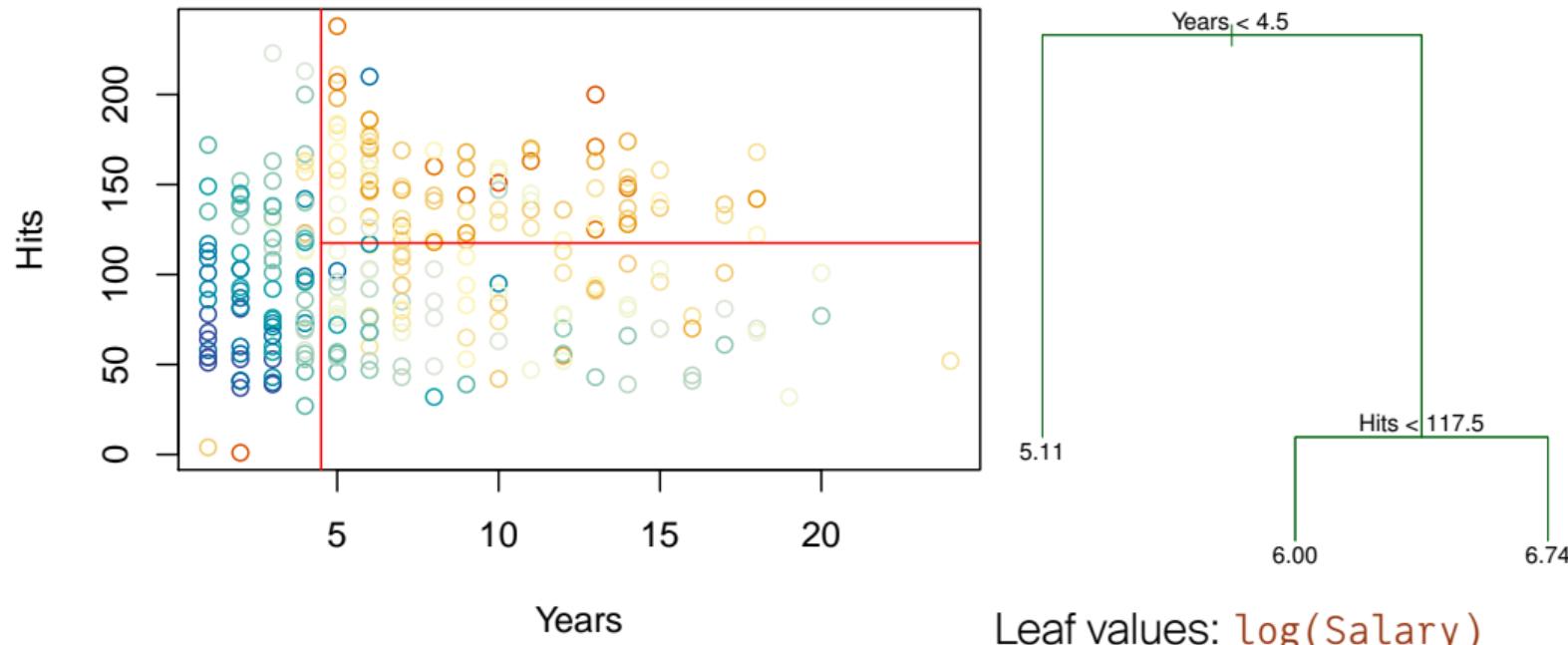
time step	1	2	3	4	5	6	7	8	9	10	11	12
character	t	o	b	e		o	r		n	o	t	
	0	1	0	0	0	0	1	0	0	0	1	0
1-hot input	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	1	0	0	0	0	0	0	0	0	0	0	1
	0.1	0.4	0.0	0.8	0.7	0.1	0.6	0.0	0.0	0.0	0.0	0.0
hidden layer	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	0.3	0.7	1.0	0.3	0.0	0.8	0.3	0.3	1.1	1.3	1.3	1.4

Final hidden activity depends on whole phrase; can be used to predict the language.  
All weights are jointly trained with gradient descent to maximize the likelihood function.

# Table of Contents

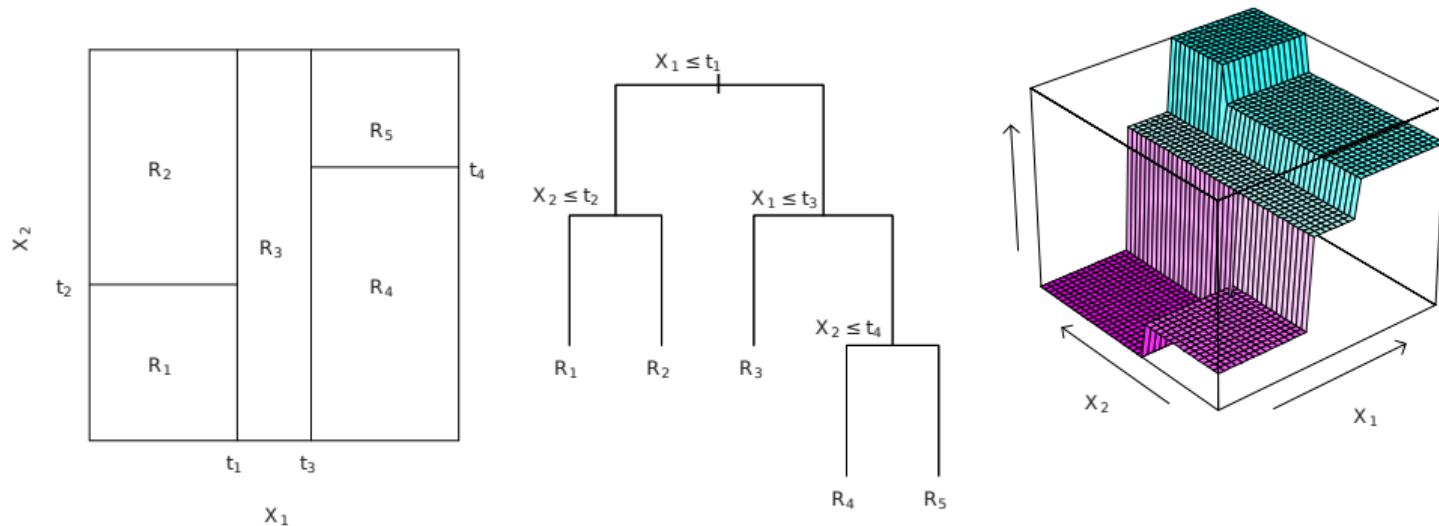
1. Convolutional Neural Networks
2. Transfer Learning
3. Recurrent Neural Networks
- 4. Tree-Based Methods: Random Forests and Gradient Boosting Trees**
5. Support Vector Machines
6. Supervised Learning: the Big Picture

# Example: Baseball Player Salary Prediction with Regression Trees



Prediction: Players with  $> 4.5$  years of experience and  $> 117.5$  hits have an average salary of  $\exp(6.74) \approx 845$  thousand USD.

# Regression Trees



Tree with 4 **internal nodes** and 5 **terminal nodes or leafs**.

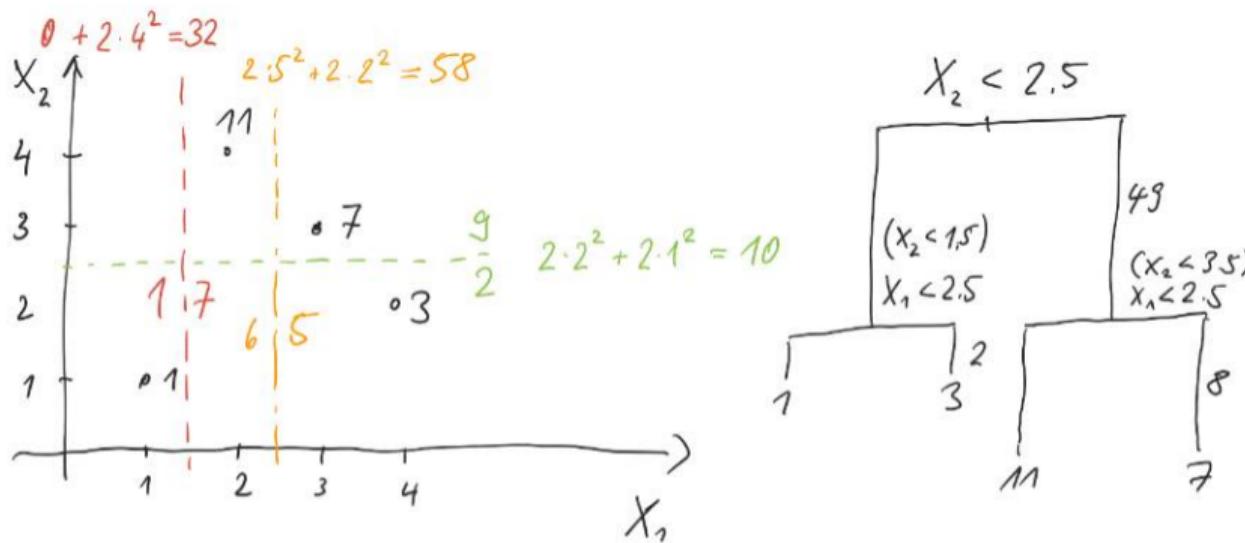
# Recursive Binary Splitting

1. Start with the root of the tree, i.e. the full predictor space.
2. Define  $R_1(j, s) = \{X | X_j < s\}$  and  $R_2(j, s) = \{X | X_j \geq s\}$  and seek  $j$  and  $s$  that minimize

$$\sum_{i: x_i \in R_1(j, s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j, s)} (y_i - \hat{y}_{R_2})^2$$

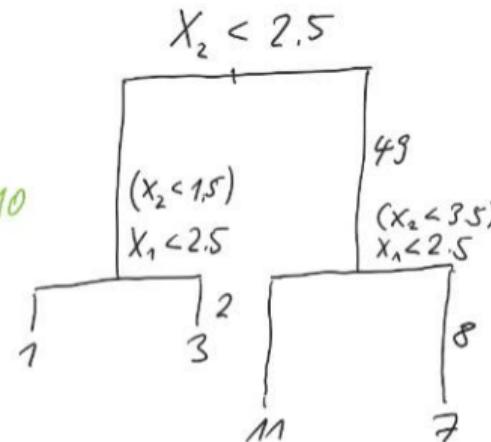
3. Repeat the splitting process on each of the two regions.
4. Iteratively split resulting regions until some stopping criterion is met; e.g. until no region contains more than 5 observations.

# Recursive Binary Splitting: Example

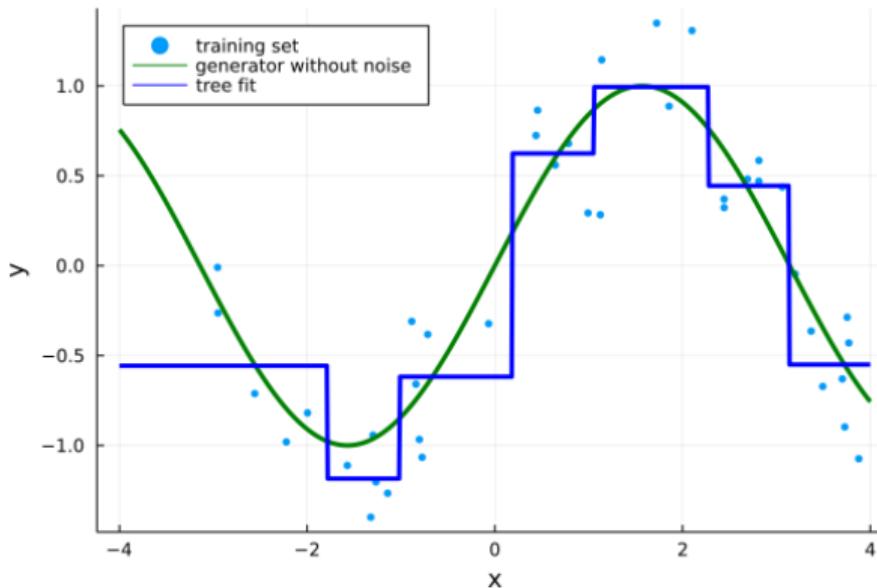


Without cut: 5.5

$$5.5^2 + 4.5^2 + 2.5^2 + 1.5^2 = 59$$

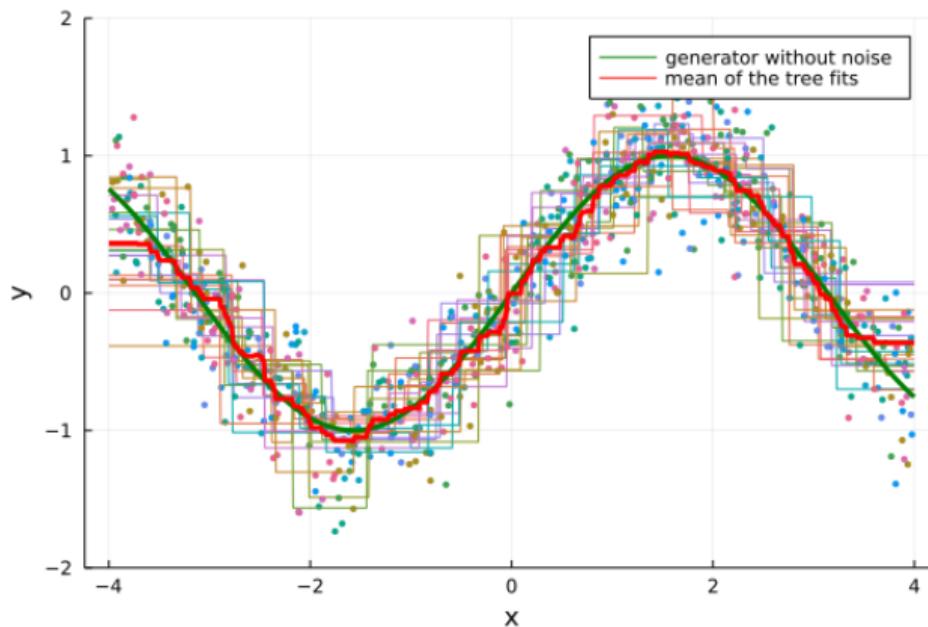


# Example: Fitting a Decision Tree



In one input dimension the result of a decision tree regression looks like the result of a zeroth order spline, where the knots are found automatically.

# Ensembling



Averaging individual fits with low bias but high variance  
can lead to an improved prediction.

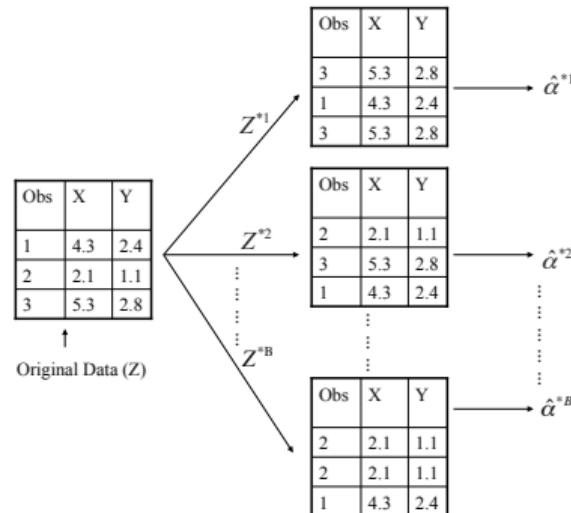
# The Bootstrap

“Pulling oneself up by one’s bootstraps”

19th century saying for impossible tasks



Subsample data with replacement



How can we get new data sets without having new data?

Original data  $Z \rightarrow B$  bootstrap data sets  
 $Z_1^*, \dots, Z_B^*$

Some of the figures in this presentation are taken from “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

# Bagging

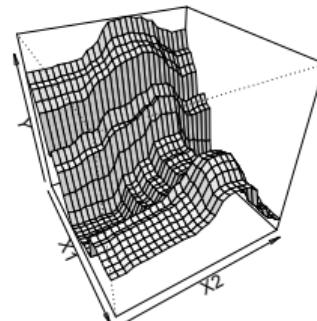
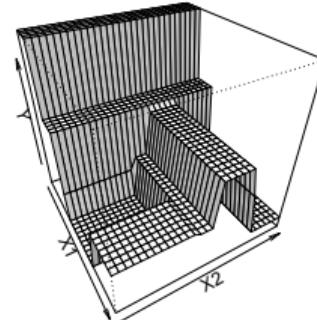
**Bagging = Bootstrap Aggregation**

B bootstrap training data sets

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$$

where  $\hat{f}^{*b}(x)$  is a function fitted to training set  $b$ .

- ▶ For flexible machine learning methods, individual fits have low bias but high variance.
- ▶ Averaging the individual fits reduces the variance.
- ▶ Majority vote for classification.



# Bagging

**Idea of Bagging** Individual fits have high variance but low bias. Average across many fits reduces the variance while keeping the bias low.

How does the variance decrease with the number of fits?

## Toy example to sharpen our intuition

$X_i$  are uncorrelated and normally distributed with mean 0 and variance  $\sigma^2$ :

$$\frac{1}{B} \sum_{i=1}^B X_i \text{ has variance } \frac{\sigma^2}{B}.$$

$X_i$  are correlated with pairwise correlation  $\rho$ :

$$\frac{1}{B} \sum_{i=1}^B X_i \text{ has variance } \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

**Problem of Bagging** The fits are highly correlated.

# Random Forests

**Decorrelation Idea** When growing a tree on a bootstrapped dataset, select  $m \leq p$  of the input variables at random as candidates for splitting before each split.

**Why does this decorrelate?** Trees differ not only because of different bootstrapped training data but also because of random subset of predictors.

**Typical Choices** Select  $m \approx \sqrt{p}$  predictors for each split.  
(If  $m = p$  we are back to simple bagging).

# Gradient Boosting Trees

Random Forests

$$f(x) = \frac{1}{K} \sum_{k=1}^K \text{tree}_k(x)$$

where  $\text{tree}_k(x)$  are learned from data and should not be too correlated.

Gradient Boosting Trees (GBT):

$$f(x) = \sum_{k=1}^K a_k \text{tree}_k(x)$$

The trees  $\text{tree}_k$  and the  $a_k$ s are learned sequentially. Idea:  $\text{tree}_i$  should boost the performance by compensating for the mistakes made by  $\sum_{k=1}^{i-1} a_k \text{tree}_k(x)$ .

# Summary

- ▶ Tree-based methods split the input space parallel to the axes.  
This is similar to spline-based methods or neural networks with the heaviside activation function and vector features parallel to the axes (only one of the weight entries is non-zero). General neural networks are more flexible.
- ▶ The Bootstrap is a method to obtain multiple training sets from a single training set. Averaging models over bootstrapped training sets (bagging) can improve performance.
- ▶ Ensembling: flexible, individual models with high variance and low bias can be averaged to obtain good fits.  
Generalization: One can also (weighted) average models of different kind, e.g. neural networks with random forests.

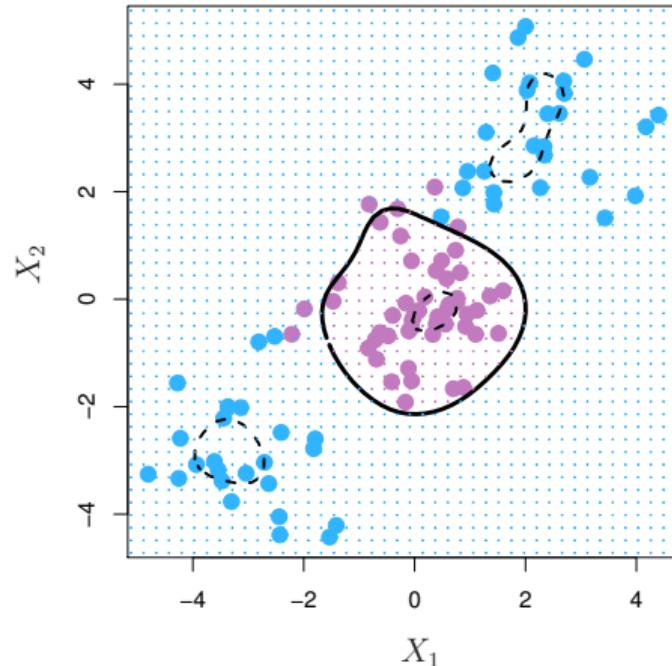
# Table of Contents

1. Convolutional Neural Networks
2. Transfer Learning
3. Recurrent Neural Networks
4. Tree-Based Methods: Random Forests and Gradient Boosting Trees
5. Support Vector Machines
6. Supervised Learning: the Big Picture

# Support Vector Machines

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \kappa(x_i, x)$$

- ▶  $\kappa(x, y)$  is called kernel; we can choose the type of kernel, e.g. radial kernel  
 $\kappa(x, y) = \exp(-\gamma \sum_{i=1}^p (x_i - y_i)^2)$ .
- ▶  $\beta_0$  and  $\alpha_i$  are learned with a special loss function (not maximum likelihood).
- ▶ Most  $\alpha_i$ 's are zero after training.
- ▶ The training points  $x_i$  with non-zero  $\alpha_i$  are called support vectors.



# Table of Contents

1. Convolutional Neural Networks
2. Transfer Learning
3. Recurrent Neural Networks
4. Tree-Based Methods: Random Forests and Gradient Boosting Trees
5. Support Vector Machines
6. Supervised Learning: the Big Picture

# Supervised Learning: the Big Picture

# Supervised Learning: the Big Picture

Lookup Table	Function Approximation
Only useful when the training set contains all inputs that will ever appear	Allows for generalization to unseen input. How good the generalization is depends on the data and the <u>inductive bias</u> , i.e. generalization is good when the function approximator comes close to the true data generator.
Non-Parametric Function Approximation	Parametric Function Approximation
The training set is explicitly stored and used to make predictions, e.g. KNN; typically fast at training, slow at prediction	The training set is implicitly stored in the parameters, e.g. linear regression, neural networks, trees; typically slow at training, fast at prediction

## Additional Considerations:

- ▶ What is a good noise model for the conditional  $p(Y|X)$ ? E.g. normal, log-normal, Poisson, ...
- ▶ Is there known structure in the data that can be exploited?

# Supervised Learning: the Big Picture

In general it is not clear which method should be chosen for which task and data set.

## Rules of Thumb

- ▶ Images as input: convolutional neural networks
- ▶ Sequences as input (text, music): transformers or recurrent neural networks
- ▶ With Gradient Boosting Trees or Random Forests one can often get decent results in little time (not much tuning is required). But well tuned neural networks tend often to be better.
- ▶ Carefully tuned regularization (L1, L2, early stopping or dropout) usually matters a lot.