Supervised learning

5ARE0: DATA ANALYSIS AND LEARNING METHODS

Version September 18th, 2025

Fons van der Sommen

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Outline

What is supervised learning?

Non-parametric approaches — *KNN*

Linear classification — *Logistic regression*

Non-linear classification — *Decision trees*, *Random forests*

Complexity

Introduction to supervised learning

Machine learning taxonomy

Self-supervised

Create your own labels!

Semi-supervised

Partially labeled data

Unsupervised

Only data…

Finding a “simpler” representation → dimensionality reduction

Finding similar points → clustering

Supervised

Data + labels / targets

Categorical → classification

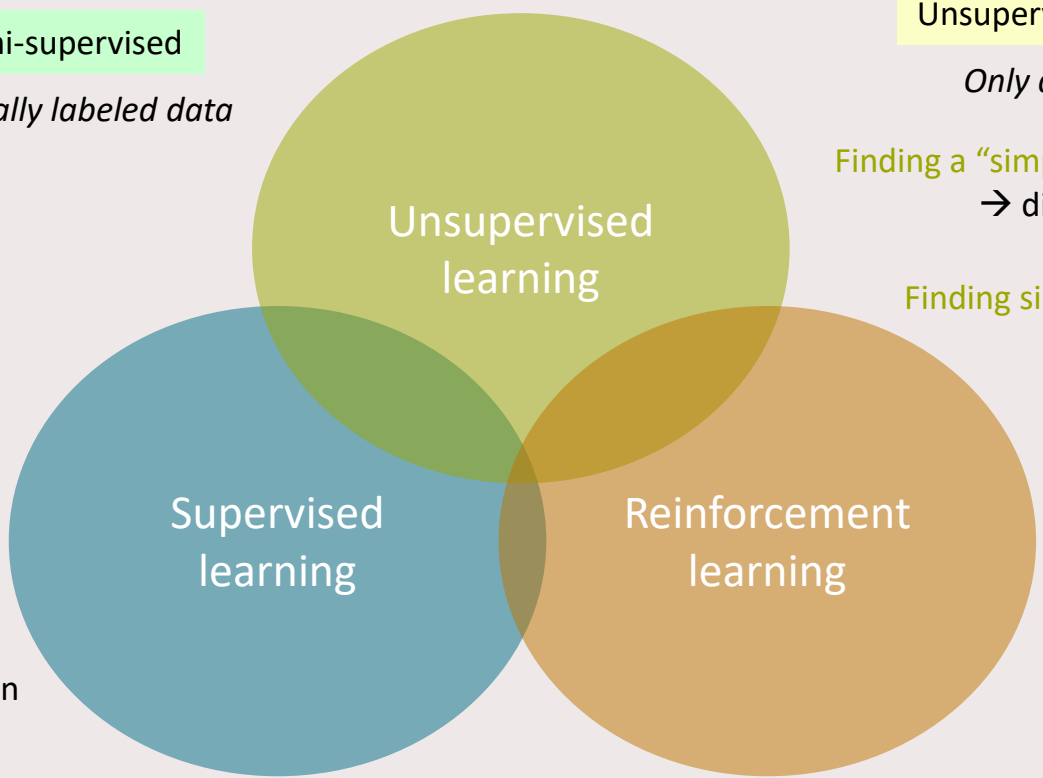
$t \in S$

Real values → regression

$t \in \mathbb{R}$

Integer masks → segmentation

$t \in \mathbb{Z}^{M \times N}$



Reinforcement

Learning while acquiring data

Outside the scope of this course…

TU/e

Machine learning taxonomy

Self-supervised

Feature learning

Semi-supervised

Exploiting larger data sets!☺

Unsupervised

Recommender systems

Targeted marketing

Supervised

Data compression

Data visualization

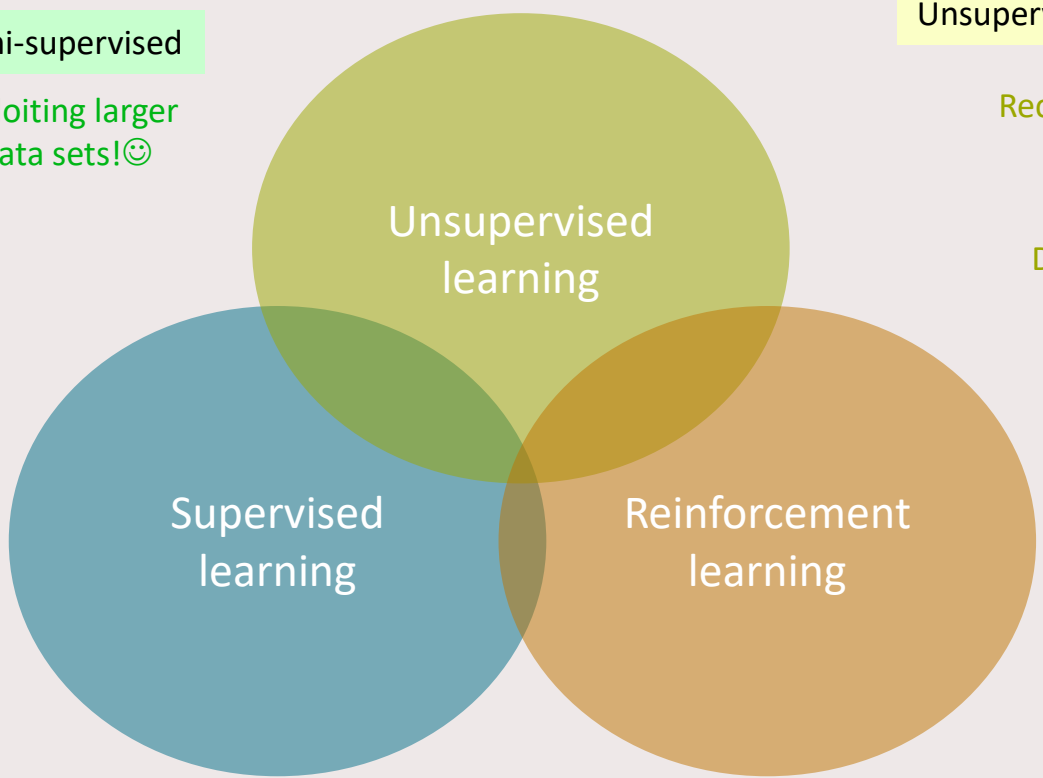


Image classification

Speech recognition

Diagnostics

Reinforcement

Game AI

Market forecasting

Robot navigation

Value estimation

Control systems

Virus spread prediction…

TU/e

Why use supervised learning?

What is meant by a heuristic approach?

Using a heuristic approach fails… badly…

- Variation in data is enormous (e.g. digits can be written in many ways!)
- ~~Sometimes~~, **Often**, the patterns cannot easily be spotted by humans (e.g. high-dimensional space)

Find these "rules" automatically!

- How can we find those rules?
  - *Learn* them from the *data* → training
- Construct a model that captures the input-output relation

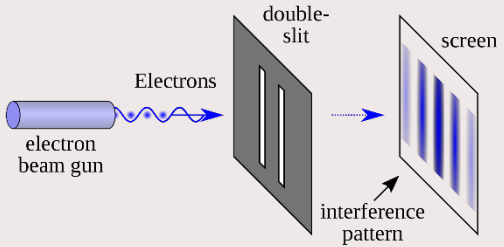
Data-driven modeling

What is supervised learning?

- Light acts as a wave!
- We get an interference pattern!
- We can derive this effect mathematically BEFORE EVEN OBSERVING IT!

You are familiar with model-driven approaches (e.g. physics)

- Use physical understanding of the problem to build a model
- Model should also give insight into the relations and data patterns
- We can explain relations and even predict them!



Wait, it acts as a particle too?

→ Let's measure each particle individually!...

Physics is just our best guess at a model that captures the world around us..

Machine learning is a data-driven approach

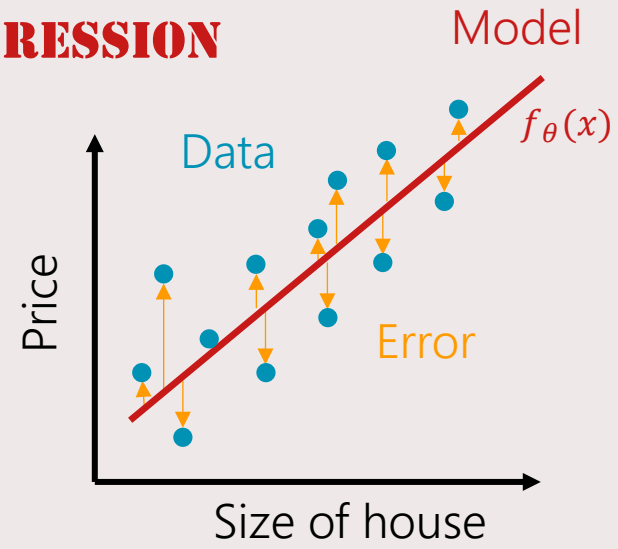
- Use the data as primary source for constructing a model
- Let go of the physical understanding of input-output relations
  - *E.g. MRI reconstruction using ML on raw data*
- Use *any* model that fits the data best!

Why could letting go of understanding be a problem?

TU/e

What is supervised learning?

REGRESSION



Basically, it's just data fitting…

- Choose a model $f_\theta(x)$ with parameters θ
- Choose or define a loss function $\mathcal{L}(y, \hat{y})$ that measures the fit with the training data $\{x, y\}_{train}$
- Minimize this loss using optimization techniques

Machine learning is a stochastic exercise

- Your training data will be a subsample of all possible data that could enter your model (data space)
- Expected mean squared error:

$$\mathbb{E}\left[\left(y - f_\theta(x)\right)^2\right] = \underbrace{\left(\mathbb{E}[f_\theta(x)] - y\right)^2}_{\text{Bias}} + \underbrace{\mathbb{E}[(\mathbb{E}[f_\theta(x)] - y)^2]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible error}}$$

"all models are wrong…
but some are useful"

George E. P. Box

TU/e

EXAMPLE

What is supervised learning?

Separate lemons from oranges

Color:	orange
Shape:	sphere
Diameter:	± 8 cm
Weigth:	±0.1 kg

Color:	yellow
Shape:	elipsoid
Diameter:	± 8 cm
Weigth:	±0.1 kg

Use color and shape as input (features) to train our model!

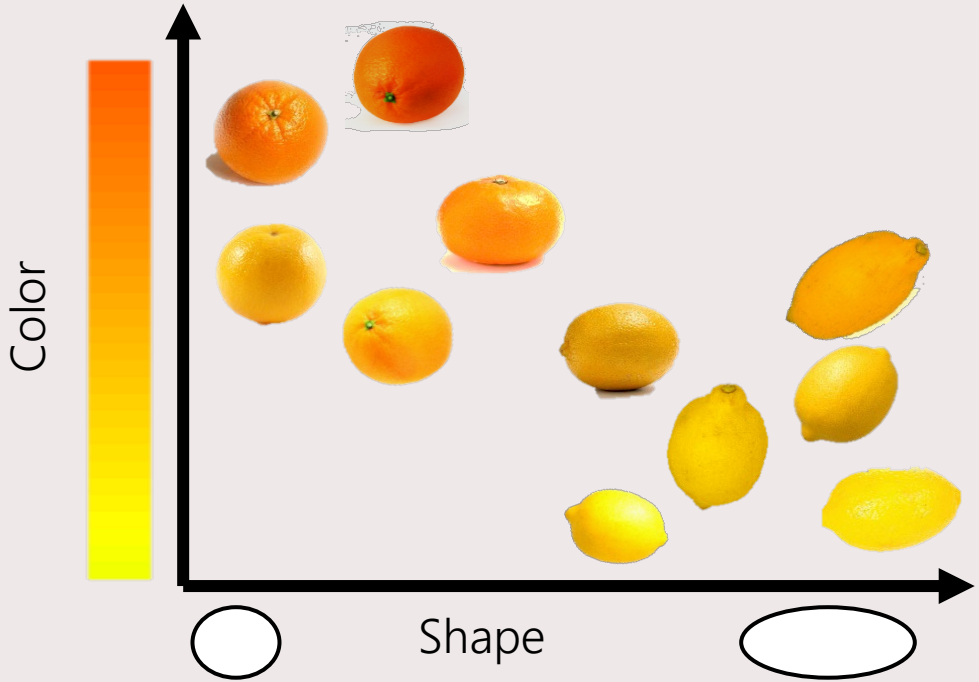
TU/e

EXAMPLE

What is supervised learning?

Separate lemons from oranges

- Use color and shape as model input
- We can actually plot the data now!
- Let's model this using two circles…



TU/e

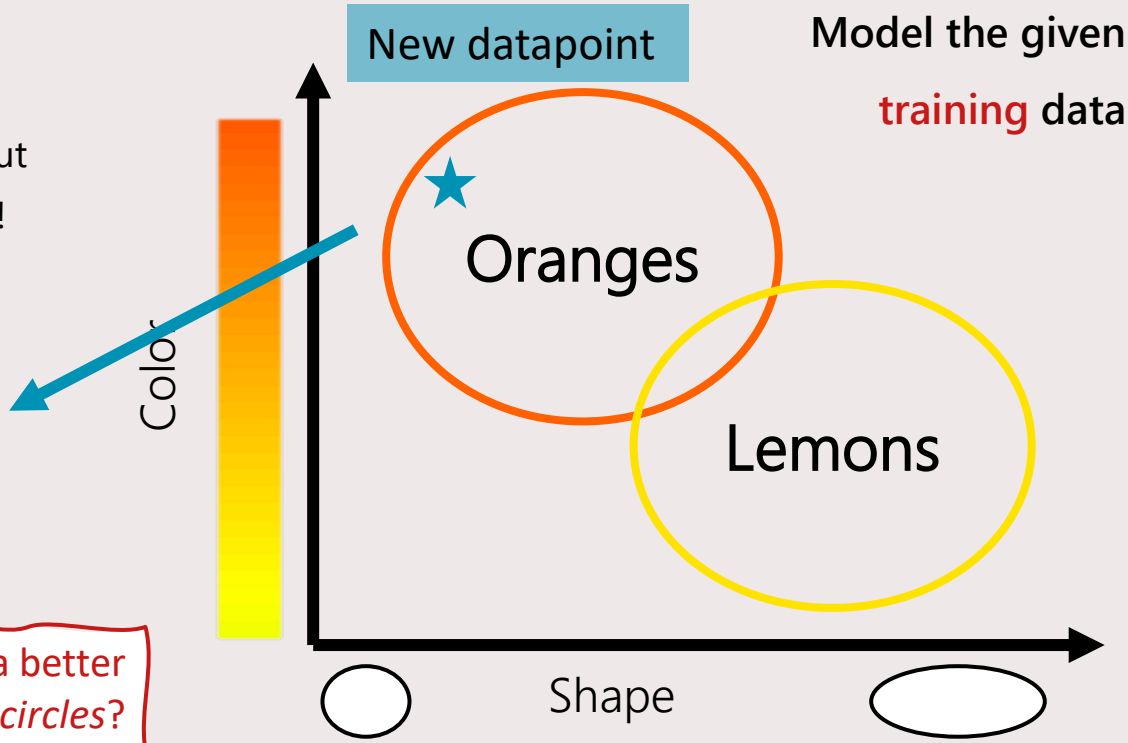
EXAMPLE

What is supervised learning?

Separate lemons from oranges

- Use color and shape as model input
- We can actually plot the data now!
- Let's model this using two circles…

Classifier:
"It's an orange!"



What is probably a better *model* than two *circles*?

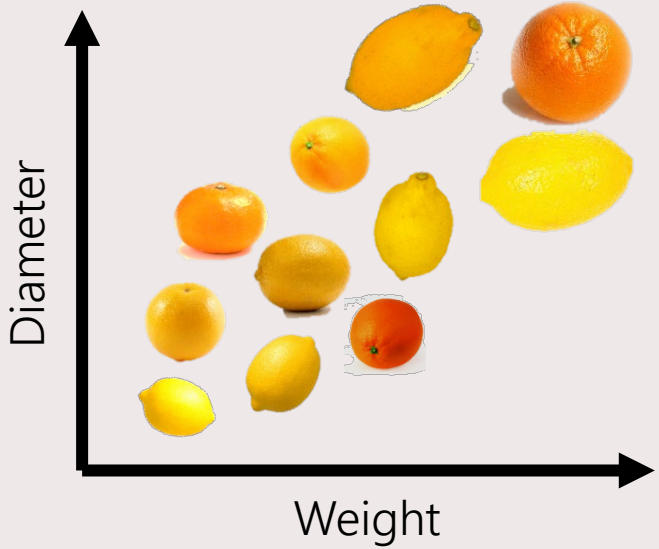
TU/e

EXAMPLE

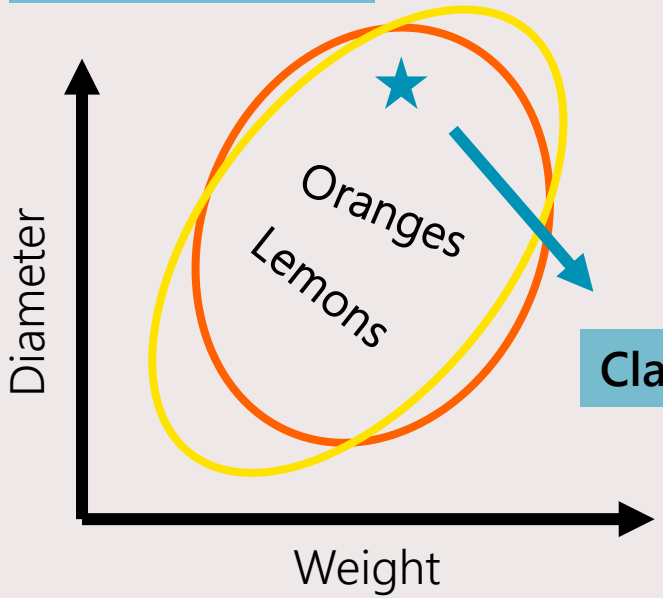
What is supervised learning?

Take-away: garbage in, garbage out!

Separate lemons from oranges



New datapoint



Classifier: *???*

Machine learning will not fix your crappy data set…

TU/e

Supervised learning

What could be parameters θ in the lemons vs oranges example?

Basic notation **CLASSIFICATION**

Data point

$$\boldsymbol{x} = \begin{bmatrix} color \\ shape \end{bmatrix}$$

WANTED

$$f(\boldsymbol{x}) = y$$

This may be a little optimistic…

Let's settle for

$$\hat{y} = f(\boldsymbol{x})$$

an ***estimate*** of y

where

$$error = d(y, \hat{y})$$

some metric measuring the distance between our estimate $\hat{y}$ and the target y

Target

$$y = \begin{cases} 0 & if \quad c = \text{lemon} \\ 1 & if \quad c = \text{orange} \end{cases}$$

Class $c \in \mathcal{C} = \{\text{orange}, \text{lemon}\}$

Number of classes $|\mathcal{C}| = 2$ is the cardinality of $\mathcal{C}$

Model

$f(x)$ is our model

Can be anything, but typically it is defined by it's parameters θ

Hence commonly you'll see $f(x|\theta)$ or $f_\theta(x)$

Parametric approach

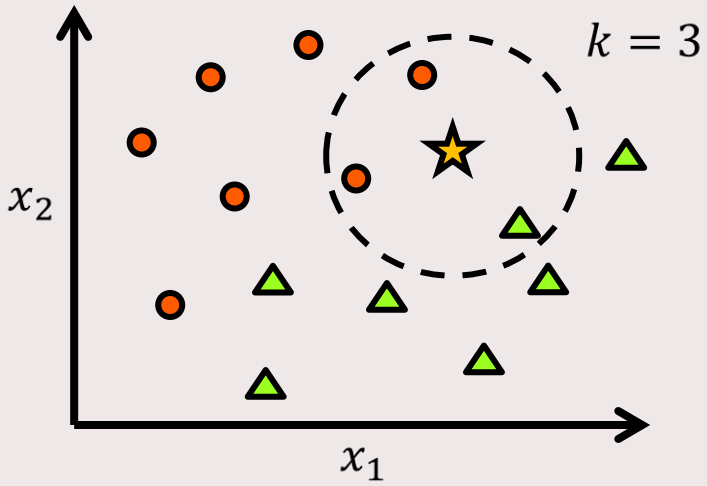
TU/e

Non-parametric classification

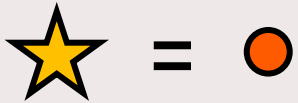
Non-parametric classification

Non-parametric approach: K Nearest Neighbours (KNN)

- Simple concept: look at the class of the k closest neighbours in feature space



For 3 nearest neighbours:



Classification

Non-parametric approach: K Nearest Neighbours (KNN)

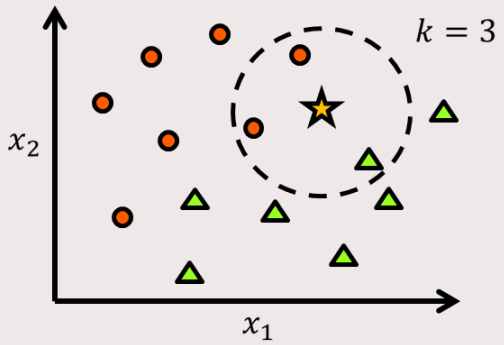
- Simple concept: look at the class of the k closest neighbours in feature space

Type of instance based learning

- A.k.a. memory based learning.
- New instance compared to training instances that are stored in memory: no explicit modelling
- Very memory-heavy classification method

Two important parameters

- Number of neighbours k
- Distance metric



Non-parametric classification

Non-parametric approach: K Nearest Neighbours (KNN)

- Simple concept: look at the class of the k closest neighbours in feature space

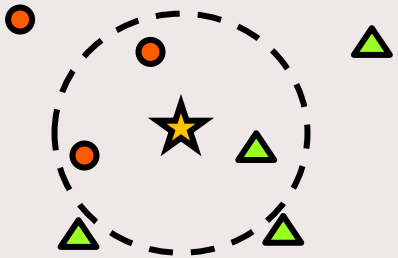
Distance metrics

- You need to select how you measure distance in feature space!
- Two commonly-used metrics:

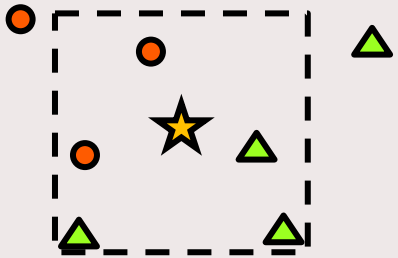
How is such a manually selected (non-trainable) parameter called?

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{D} (p_i - q_i)^2}$$

Euclidean



City-block

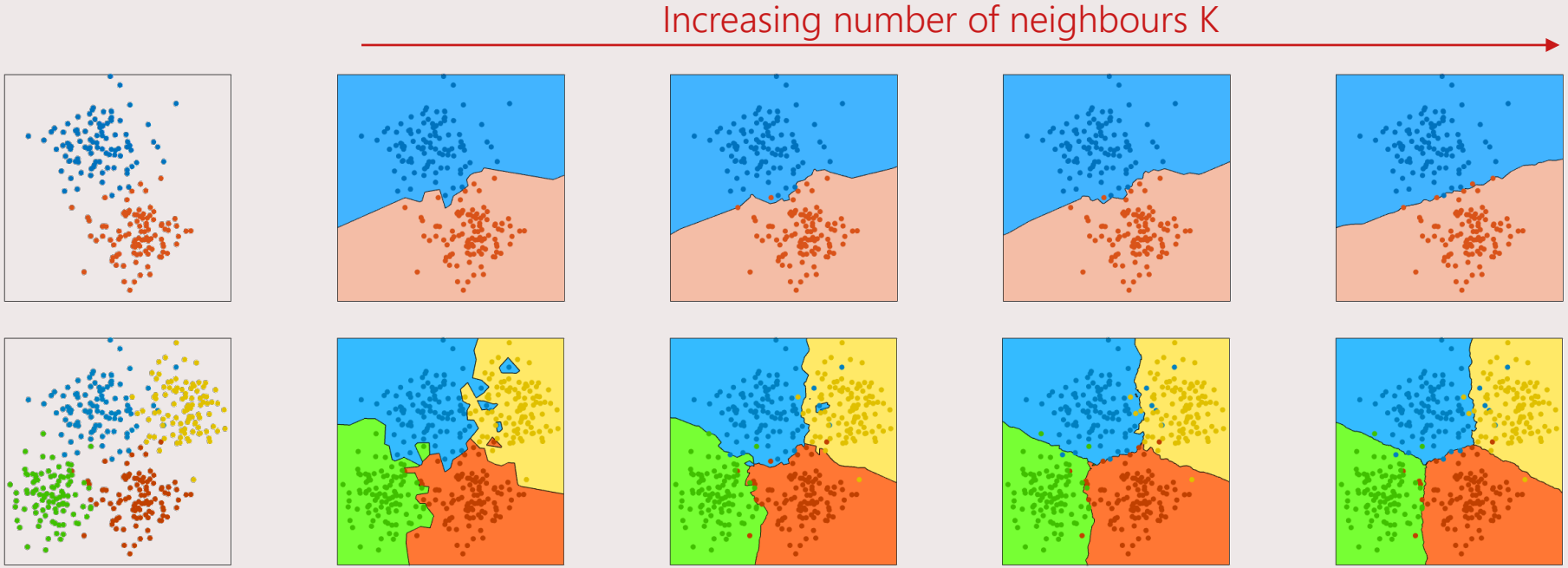


$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{D} |p_i - q_i|$$

TU/e

Non-parametric classification

Non-parametric approach: K Nearest Neighbours (KNN)



TU/e

Linear classification

Linear classification

Parametric approach

Features | Function | Prediction

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

$$f(\mathbf{x}, \mathbf{w})$$

Parameters / weights

y

Class score

High score → this is an orange
Low score → This is not an orange

Linear classification:

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}\mathbf{x} + \mathrm{b}$$

Linear classification

Parametric approach

Features

Function

Prediction

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}\mathbf{x} + \mathrm{b}$$

0.5
6.3
1.2
0.9

0.2	0.4	-0.1	0.6

0.5
6.3
1.2
0.9

$+ \quad 2.2 \quad = \quad 5.24$

Class labels in binary classification?

TU/e

Linear classification

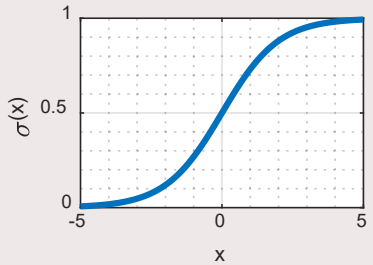
LOGISTIC REGRESSION

Matching scores with labels

- Output of $f(\mathbf{x}, \mathbf{w})$ a scalar in range $(-\infty, +\infty)$
- For classification we want labels in $\{0,1\}$
- We can map to a real number in range $[0,1]$

Logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



So our model becomes:

$$\hat{y} = \sigma(\mathbf{w}\mathbf{x} + \mathrm{b})$$

Putting a loss on wrong predictions

- Binary cross-entropy loss function:

$$\mathcal{L} = \sum_k y_k \log \hat{y}_k + (1 - y_k) \log(1 - \hat{y}_k)$$

Do we know something about one of the terms in this sum?

- Use optimization techniques to solve for **w** → more on this in coming lectures.

TU/e

Linear classification

Note that $\sigma(0) = 0.5$

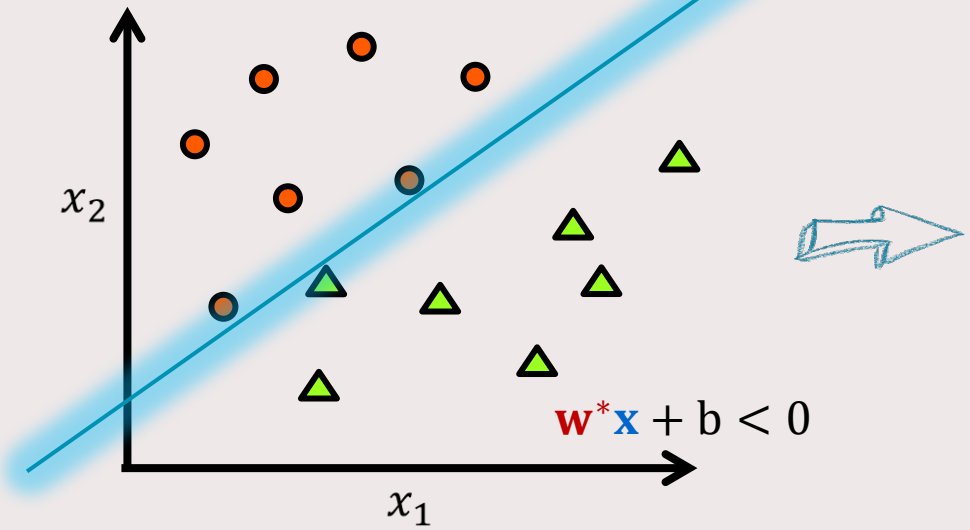
Logistic regression

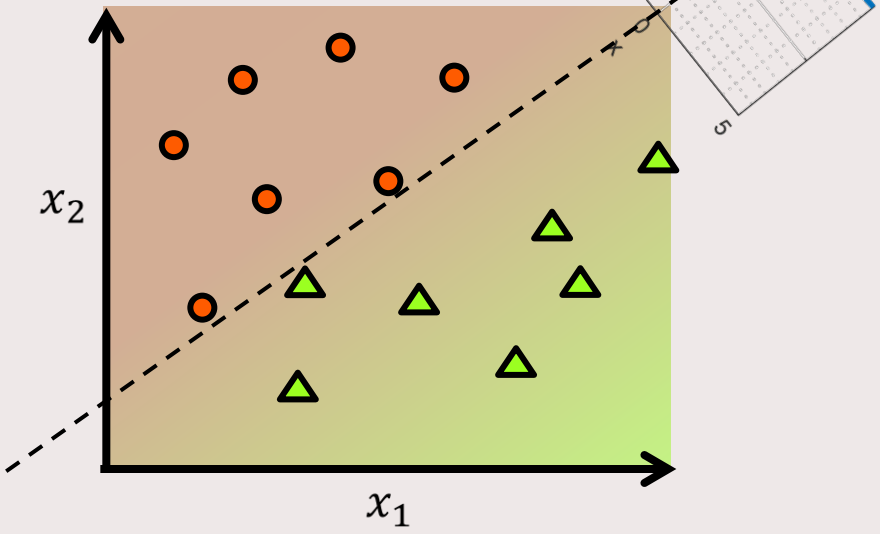
We learn this line!

$\mathbf{w}^*\mathbf{x} + b = 0$

$\mathbf{w}^*\mathbf{x} + b > 0$

$\mathbf{w}^*\mathbf{x} + b < 0$





TU/e

Linear classification

Class scores or "Logits"

What do these biases $\mathbf{b}$ represent?

Multinomial logistic regression

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

0.5
6.3
1.2
0.9

0.2	0.4	-0.1	0.6
-3.2	1.0	2.6	-1.9
2.8	-1.5	-0.2	4.7

0.5
6.3
1.2
0.9

+

2.2
-1.3
1.7

=

5.24
4.81
-2.36

Softmax

0.6
0.4
0.0

Softmax function and cross-entropy loss

$$\hat{y}_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Softmax

$$\mathcal{L}_{CE} = -\sum_k y_k \log \hat{y}_k$$

Cross-entropy loss

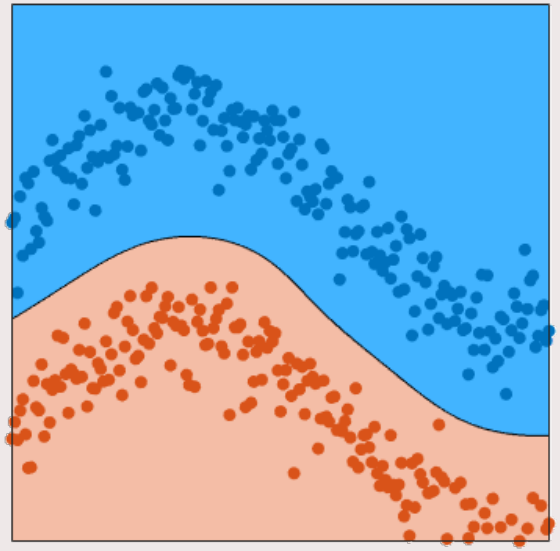
What is the range of the logits z?

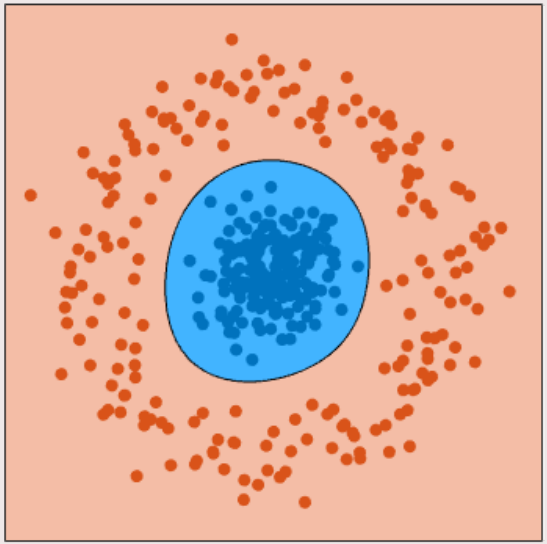
What does softmax do to the logits z?

TU/e

Linear classification

Not all data (almost no data..) is lineary separable





Non-linear classification

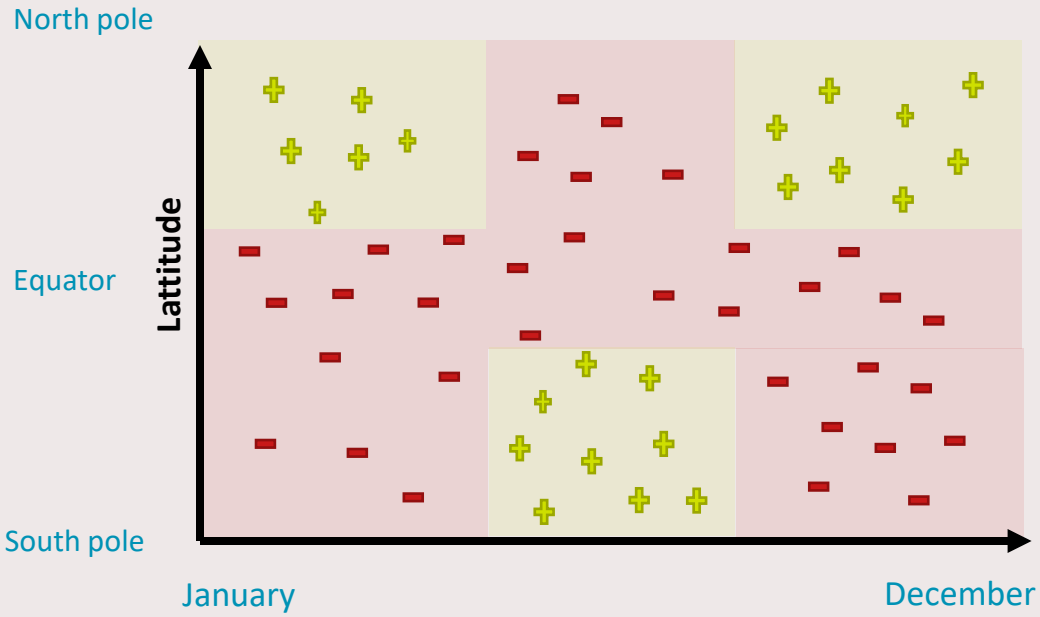
Different non-linear approachs

- Decision trees — *Today!*
- Random forests — *Today!*
- Non-linear Support Vector Machines (SVMs) — *In two weeks!*
- Artificial Neural Networks — *In two weeks!*
- Boosting
- …

Decision trees

Asking the right questions

Likelihood of being able to ice skate!



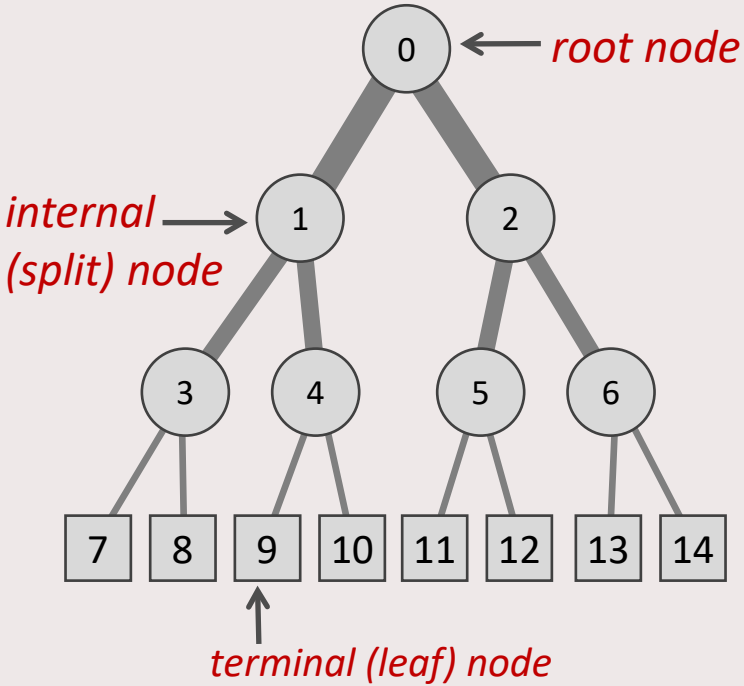
Let's build a model for this!

TU/e

The tree model

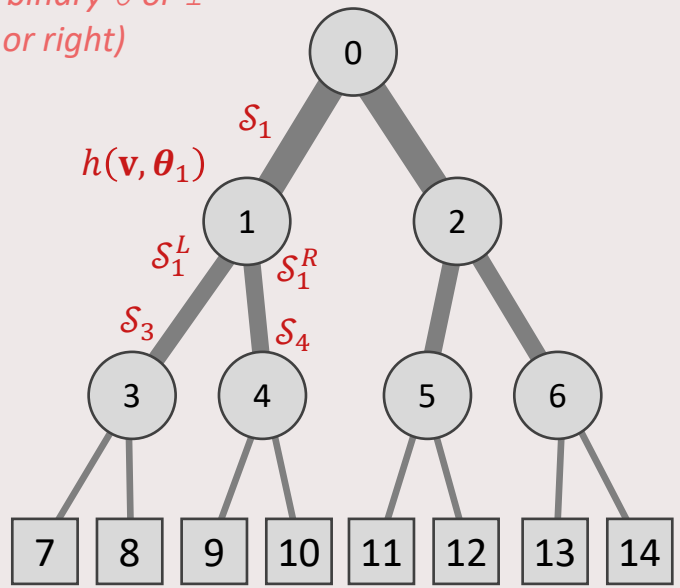
- Start at the root node
- True/false question at each split node
- Stop when a leaf node is reached: prediction

A general tree structure



The tree model

Takes data point $\mathbf{v}$ and outputs a binary 0 or 1 (left or right)



Notation

- Data point: $\mathbf{v} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$, label: y
  - Features: x_i, dimensionality: d
- Binary split function: $h(\mathbf{v}, \boldsymbol{\theta}_j): \mathbb{R}^d \times \mathcal{T} \rightarrow \{0,1\}$
  - Split parameters of node j: $\boldsymbol{\theta}_j \in \mathcal{T}$
  - Set of possible parameters $\mathcal{T}$
- Training points reaching node j: $\mathcal{S}_j = \mathcal{S}_j^L \cup \mathcal{S}_j^R$
  - Training set $\mathcal{S}_0$

Left Right

$$\mathcal{S}_j = \mathcal{S}_{2j+1} \cup \mathcal{S}_{2j+2}$$

TU/e

The tree model

How do we grow a good tree?

- Find the best data split for each node!
- What is a good split?

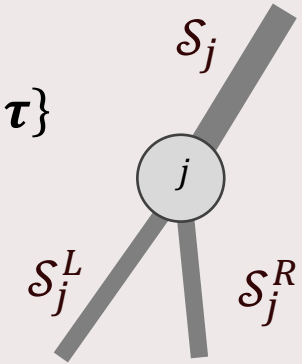
Can we **learn** to ask **the best questions**?

→ How can we find the best split function $h(\mathbf{v}, \boldsymbol{\theta})$ at each node in the tree? x

Data splitting nodes

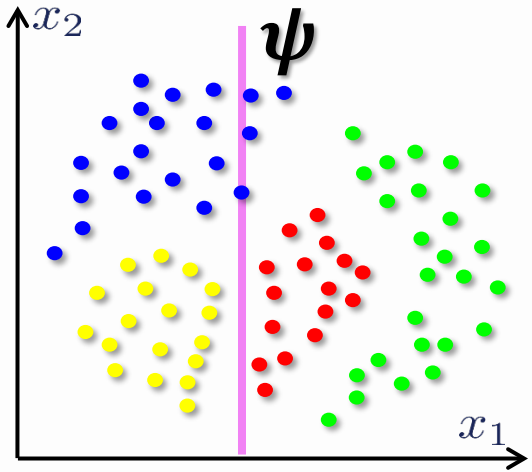
Split function $h(\mathbf{v}, \boldsymbol{\theta})$ is defined by parameters $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\tau}\}$

- Feature selection function $\boldsymbol{\phi}$
- Geometric primitive $\boldsymbol{\psi}$ (e.g. a line)
- Thresholds $\boldsymbol{\tau}$



Parameter space $\mathcal{T}$ contains the options that we have for parameters $\boldsymbol{\phi}$, $\boldsymbol{\psi}$ and $\boldsymbol{\tau}$.

Data splitting nodes

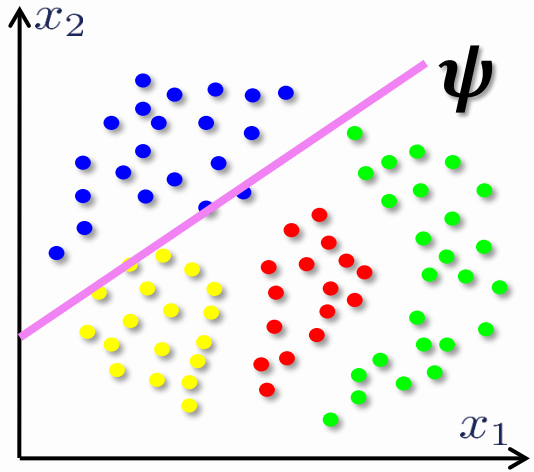


Axis aligned hyperplane

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\mathbf{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

For 2D example: $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \quad x_2 \quad 1)^T$

e.g. $\boldsymbol{\psi} = (1 \quad 0 \quad \psi_3)$

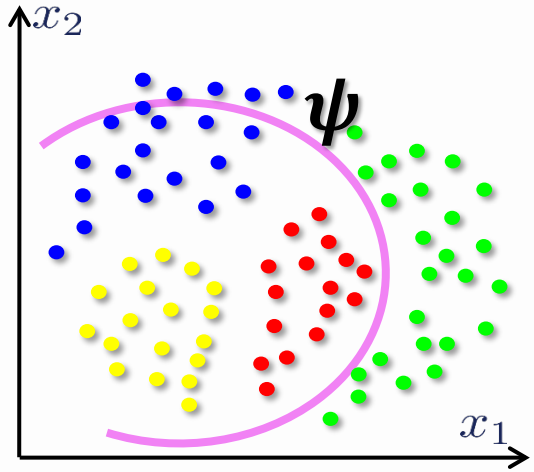


Oriented hyperplane

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\mathbf{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

$\boldsymbol{\phi}(\mathbf{v}) = (x_1 \quad x_2 \quad 1)^T$

$\boldsymbol{\psi} \in \mathbb{R}^3$, *e.g.* $\boldsymbol{\psi} = (3 \quad -2 \quad 4)$



Quadratic surface

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}^T(\mathbf{v})\, \boldsymbol{\psi}\, \boldsymbol{\phi}(\mathbf{v}) > \tau_2]$$

$\boldsymbol{\phi}(\mathbf{v}) = (x_1 \quad x_2 \quad 1)^T$

$\boldsymbol{\psi} \in \mathbb{R}^{3\times3}$ representing a conic

TU/e

Data splitting nodes

Why not just optimize $\boldsymbol{\theta}_j$ over all possible $\boldsymbol{\theta} \in \mathcal{T}$

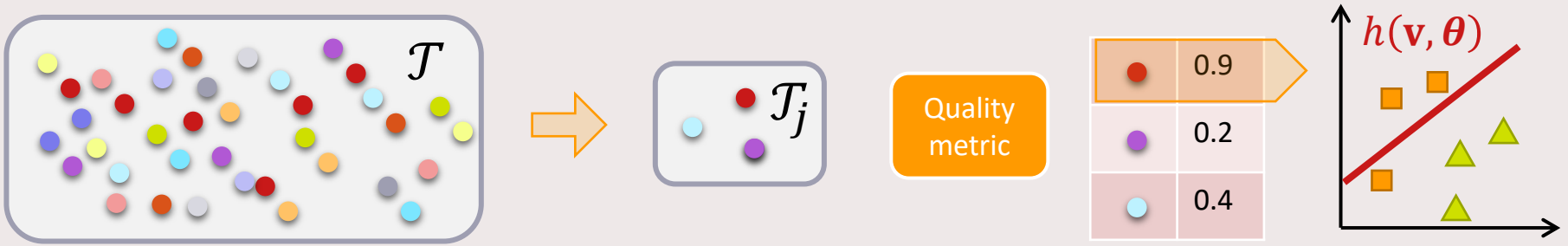
We now know how split function $h(\mathbf{v}, \boldsymbol{\theta})$ is defined by parameters $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\tau}\}$

→ But how do we find optimal $\boldsymbol{\theta}$ for each node?

Randomized node optimization

- We're gonna randomly sample options $\boldsymbol{\theta}_j$ from the total parameter space $\mathcal{T}$
- Based on some quality metric, we select the best split parameters $\boldsymbol{\theta}_j^*$

Note that this is not the best possible split..

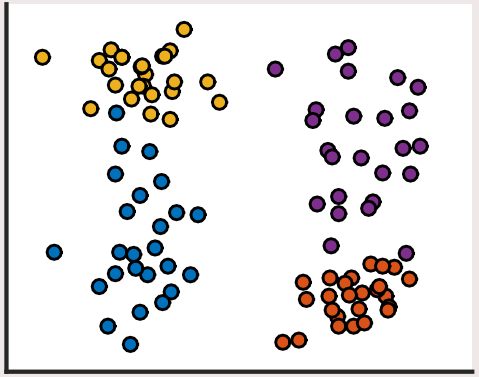


TU/e

Data splitting nodes

$$\mathcal{C} = \begin{Bmatrix} yellow, orange, \\ purple, blue \end{Bmatrix}$$

EXAMPLE



How to determine the best split?
Maximize information gain*:

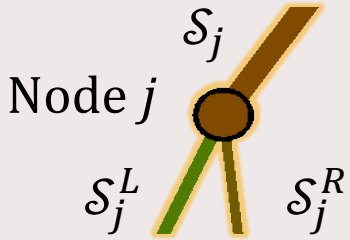
- Shannon's entropy:

$$I(\mathcal{S}, \theta) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

Node training

$$H(\mathcal{S}^i) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c))$$

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$$

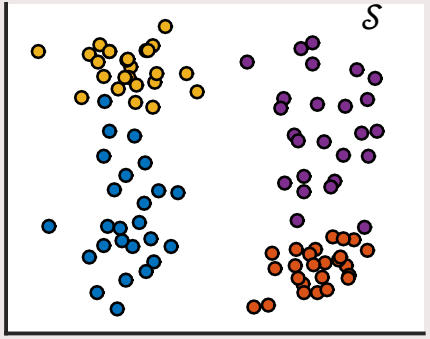


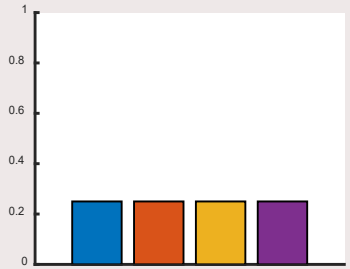
*One of many options. Other popular choices:
(1) Gini's diversity index, (2) Misclassification error*

TU/e

Data splitting nodes

EXAMPLE



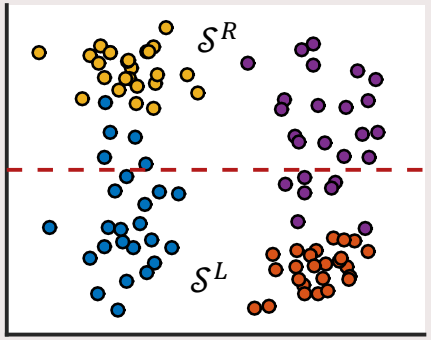


Example:

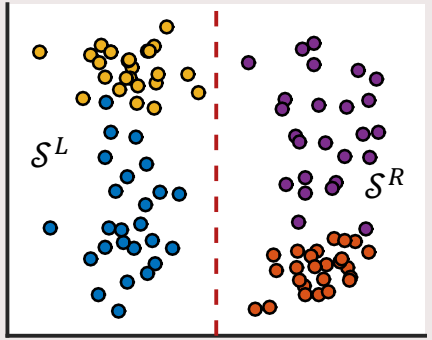
- Consider a data set with
  - Number of data points $N = 100$,
  - Number of classes$|C| = 4$,
  - Number of points per class $|S_c| = 25$ for all classes $c \in \mathcal{C}$
- We want to split this data set and we can chose from splits θ_1 and θ_2

Which would be best in your opinion? Why?

Split option θ_1



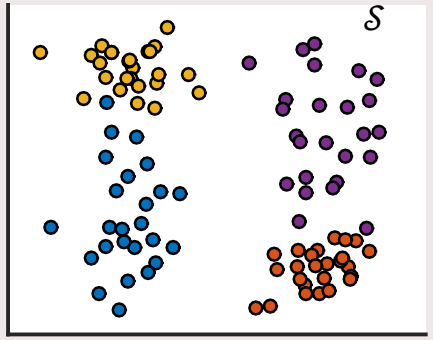
Split option θ_2

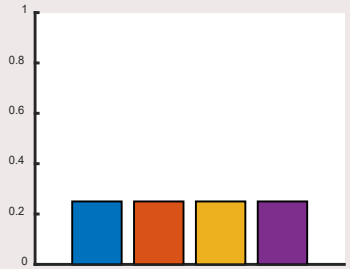


TU/e

Data splitting nodes

EXAMPLE





Example:

- Consider a data set with
  - Number of data points $N = 100$,
  - Number of classes $|C| = 4$,
  - Number of points per class $|S_c| = 25$ for all classes $c \in \mathcal{C}$
- We want to split this data set and we can chose from splits θ_1 and θ_2
- Use **information gain** to find the best split!

$$I(\mathcal{S}, \theta) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

➔Let's first compute the entropy of the input:

$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log\big(p(c)\big)$$

$$= -p(blue)\log\big(p(blue)\big) - p(orange)\log\big(p(orange)\big) - p(yellow)\log\big(p(yellow)\big) - p(purple)\log\big(p(purple)\big)$$

$$= -\frac{25}{100}\log\left(\frac{25}{100}\right) - \frac{25}{100}\log\left(\frac{25}{100}\right) - \frac{25}{100}\log\left(\frac{25}{100}\right) - \frac{25}{100}\log\left(\frac{25}{100}\right)$$

$$= 0.3466 + 0.3466 + 0.3466 + 0.3466$$

$$H(\mathcal{S}) = 1.386$$

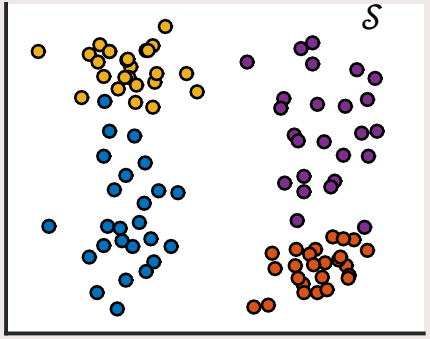
TU/e

EXAMPLE

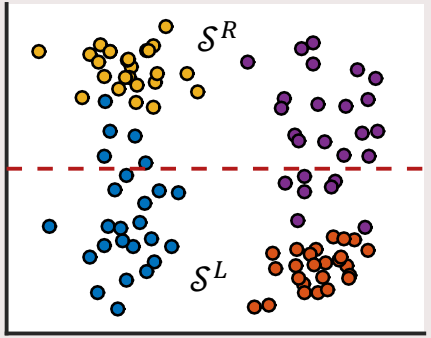
Data splitting nodes

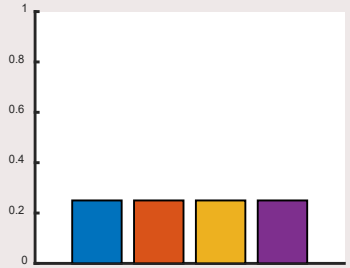
Split option θ_1

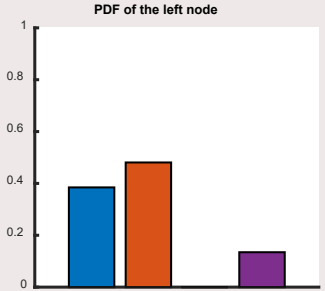
$H(\mathcal{S}) = 1.386$

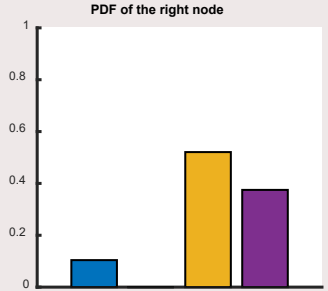


$|\mathcal{S}^L| = 52, |\mathcal{S}^R| = 48$









➔ Now let's compute the entropy of the left output:

$$H(\mathcal{S}^L) = -\sum_{c \in \mathcal{C}} p(c)\log\big(p(c)\big)$$

$$= -p(blue)\log\big(p(blue)\big) - p(orange)\log\big(p(orange)\big) - p(yellow)\log\big(p(yellow)\big) - p(purple)\log\big(p(purple)\big)$$

$$= -\frac{20}{52}\log\left(\frac{20}{52}\right) - \frac{25}{52}\log\left(\frac{25}{52}\right) - \frac{0}{52}\log\left(\frac{0}{52}\right) - \frac{7}{52}\log\left(\frac{7}{52}\right)$$

$$= 0.3675 + 0.3521 + 0.2699$$

$$H(\mathcal{S}^L) = \mathbf{0.9895}$$

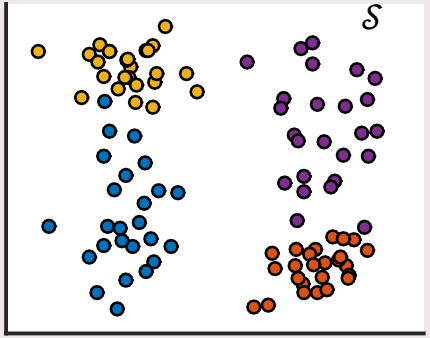
TU/e

EXAMPLE

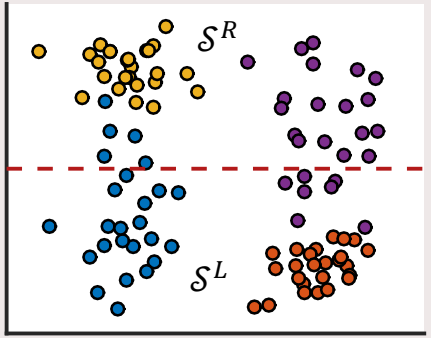
Data splitting nodes

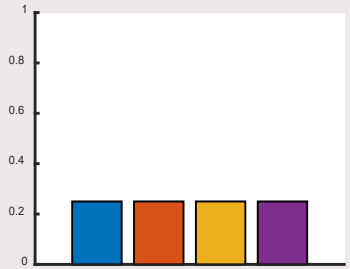
Split option θ_1

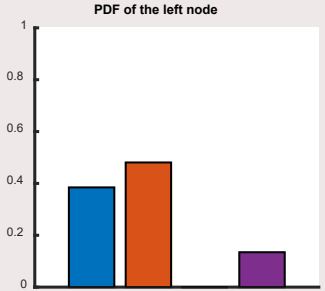
$H(\mathcal{S}) = 1.386$

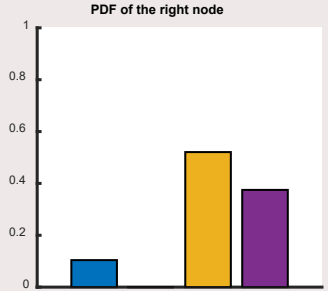


$|\mathcal{S}^L| = 52, |\mathcal{S}^R| = 48$









Do the same for $\mathcal{S}^R$

$H(\mathcal{S}^R) = \mathbf{0.9432}$

Information gain

$$I(\mathcal{S}, \theta_1) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

$$= \underbrace{1.386}_{H(\mathcal{S})} - \left(\frac{52}{100} \cdot \underbrace{0.9895}_{H(\mathcal{S}^L)} + \frac{48}{100} \cdot \underbrace{0.9432}_{H(\mathcal{S}^R)} \right)$$

$$I(\mathcal{S}, \theta_1) = 0.4187$$

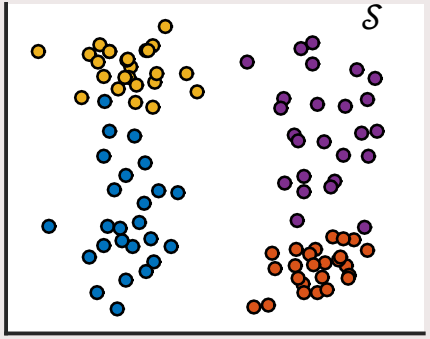
TU/e

EXAMPLE

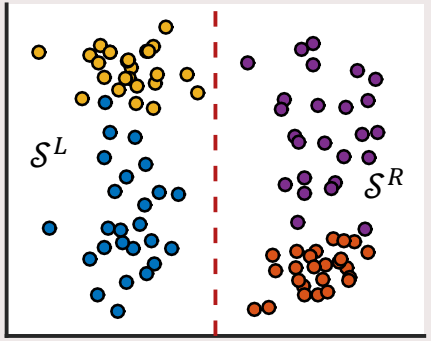
Data splitting nodes

Split option θ_2

$H(\mathcal{S}) = 1.386$



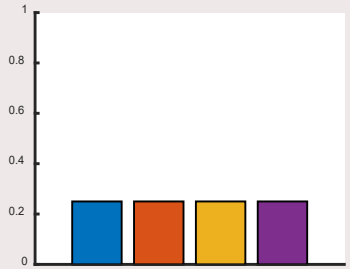
$|\mathcal{S}^L| = 50, |\mathcal{S}^R| = 50$

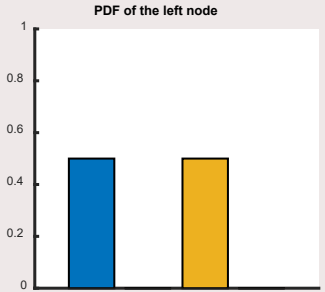


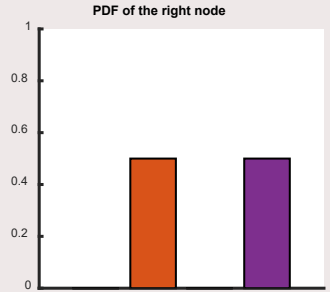
$$H(\mathcal{S}^L) = -\sum_{c \in \mathcal{C}} p(c)\log\big(p(c)\big)$$

$$= -\frac{25}{50}\log\left(\frac{25}{50}\right) - \frac{0}{50}\log\left(\frac{0}{50}\right) - \frac{25}{50}\log\left(\frac{25}{50}\right) - \frac{0}{50}\log\left(\frac{0}{50}\right)$$

$$= 0.3466 + 0.3466 \quad \boxed{= \mathbf{0.6932}}$$







$$H(\mathcal{S}^R) = \mathbf{0.6932}$$

TU/e

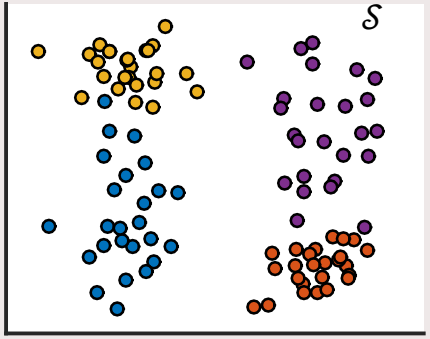
EXAMPLE

Data splitting nodes

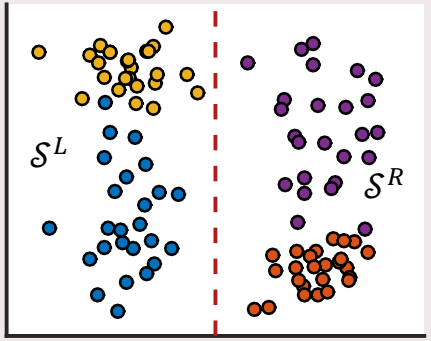
Split option θ_2

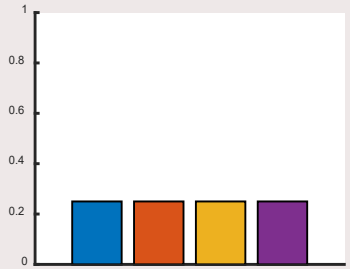
Note that in this case $H(\mathcal{S}^L) = H(\mathcal{S}^R)$ ☺

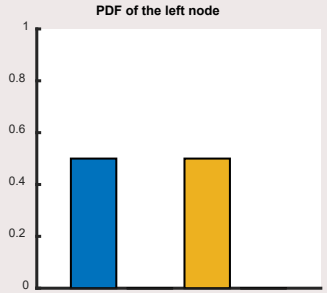
$H(\mathcal{S}) = 1.386$

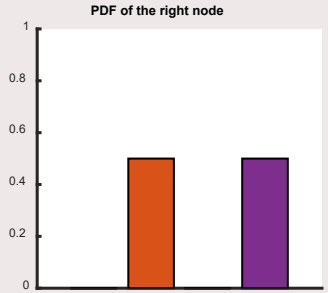


$|\mathcal{S}^L| = 50, |\mathcal{S}^R| = 50$









Information gain

$$I(\mathcal{S}, \theta_2) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

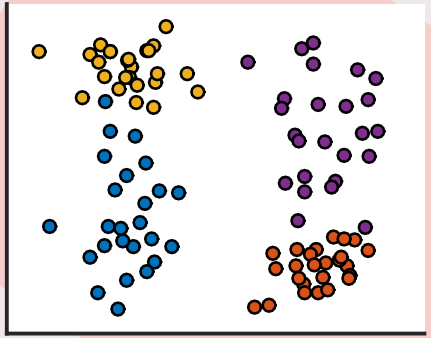
$$= 1.386 - \left(\frac{50}{100} \cdot 0.6932 + \frac{50}{100} \cdot 0.6932 \right)$$

$$I(\mathcal{S}, \theta_2) = 0.6928$$

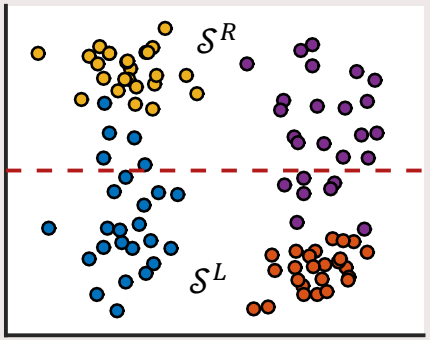
TU/e

EXAMPLE

Data splitting nodes

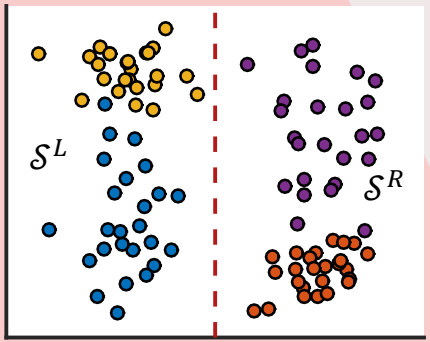


Split option θ_1



$I(\mathcal{S}, \theta_1) = 0.4187$

Split option θ_2



$I(\mathcal{S}, \theta_2) = 0.6928$

TU/e

Data splitting nodes

Where we stand:

- Information gain to measure the quality of a split:

$$I(\mathcal{S}, \theta) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

- During training, select parameters that maximize the information gain:

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$$

  - Split function parameters $\boldsymbol{\theta}$
  - Limited set of parameter settings $\mathcal{T}_j$

But how to select the set of candidate splits $\mathcal{T}$? → Next: Training trees!

Training trees

How to train a decision tree?

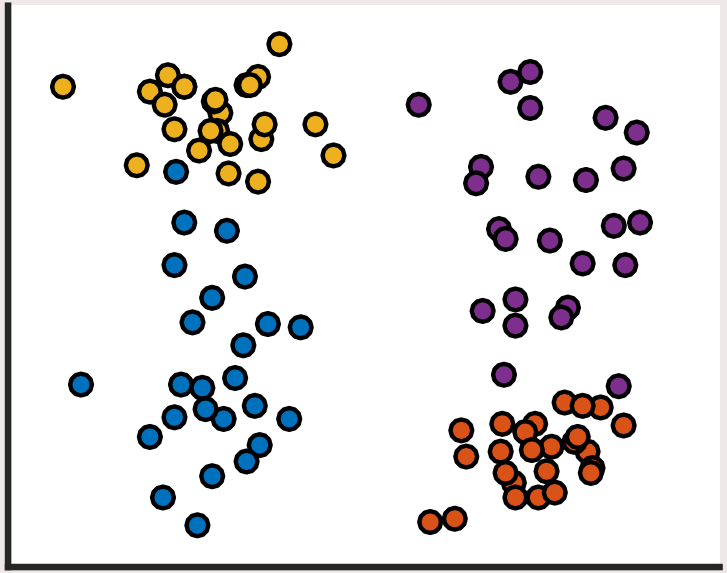
Randomized Node Optimization

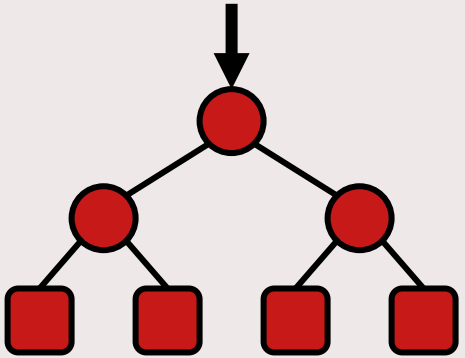
- Start with a **random** subset of all the data at the root node
- Find the split parameters $\boldsymbol{\theta}_j$ from a set of **randomly chosen** options $\mathcal{T}_j \in \mathcal{T}$ that maximize some split quality metric (e.g. information gain)
- Repeat this for the outgoing nodes and stop growing a certain branch untill one of the following two criteria holds:
  - A pre-defined tree depth D is reached (# nodes of a branch)
  - *Alternatively: untill a pre-defined total number of nodes is reached*
  - All training samples in the node are from the same class

EXAMPLE

Training trees

Let's train a tree of depth 2 for the four-class data set we've seen earlier



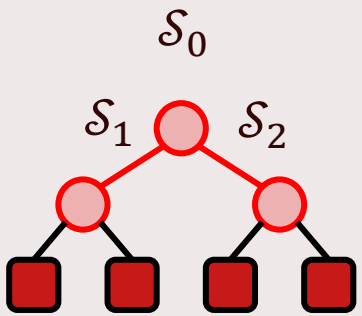


EXAMPLE

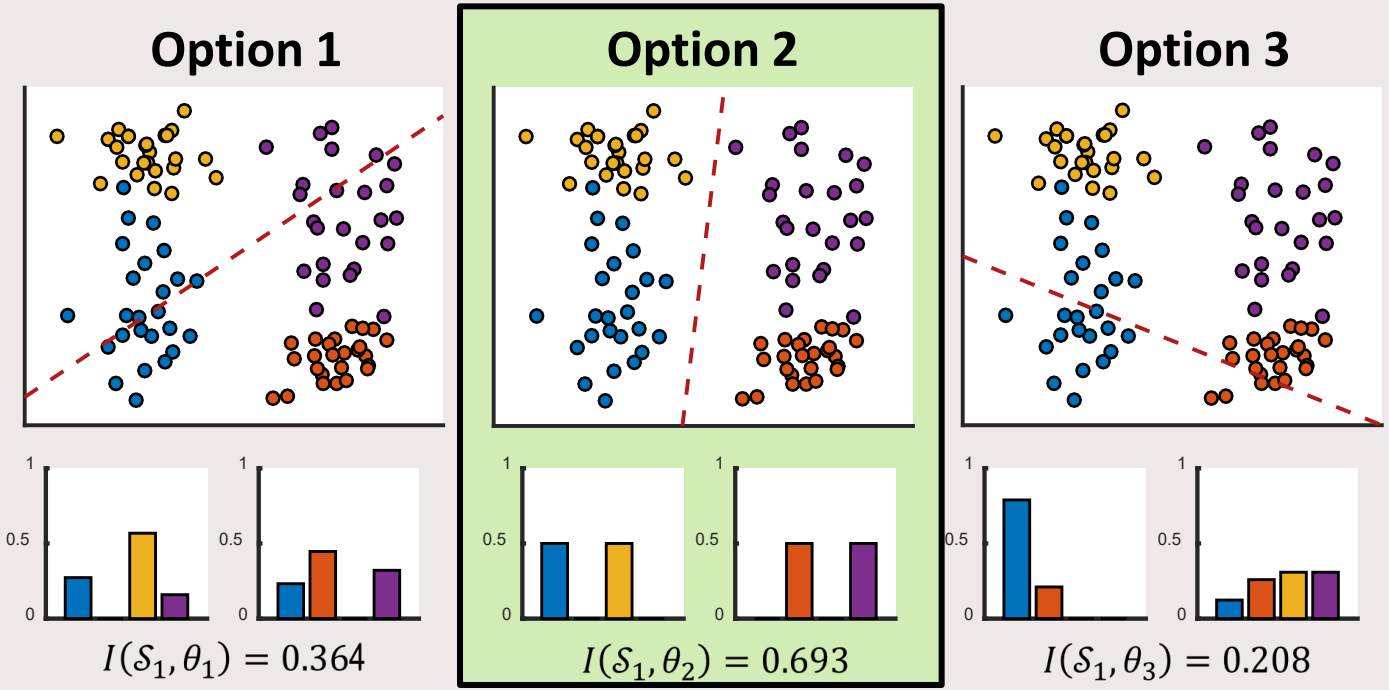
Training trees

We randomly get three options:

Training node 0



$|\mathcal{T}_j| = 3$



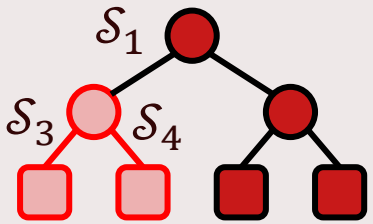
TU/e

EXAMPLE

Training trees

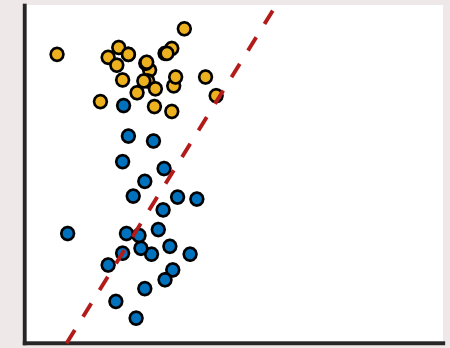
We randomly get three options:

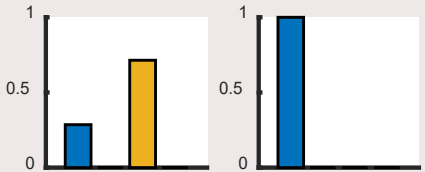
Training node 1



$|\mathcal{T}_j| = 3$

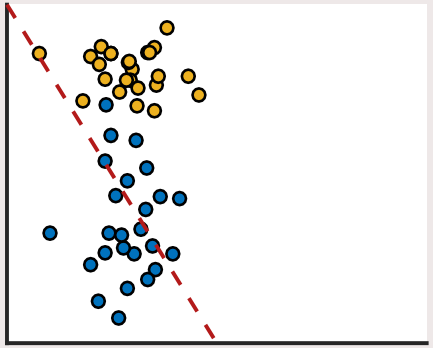
Option 1

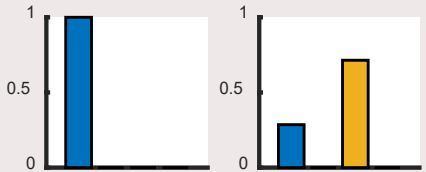




$I(\mathcal{S}_{1,L}, \theta_1) = 0.274$

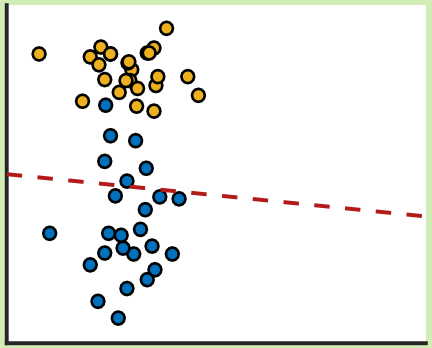
Option 2

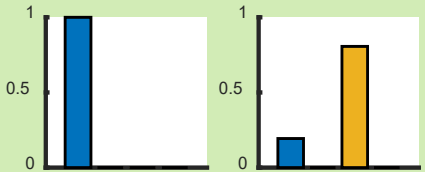




$I(\mathcal{S}_{1,L}, \theta_2) = 0.274$

Option 3





$I(\mathcal{S}_{1,L}, \theta_3) = 0.389$

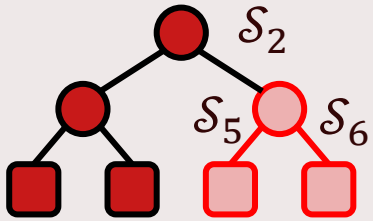
TU/e

EXAMPLE

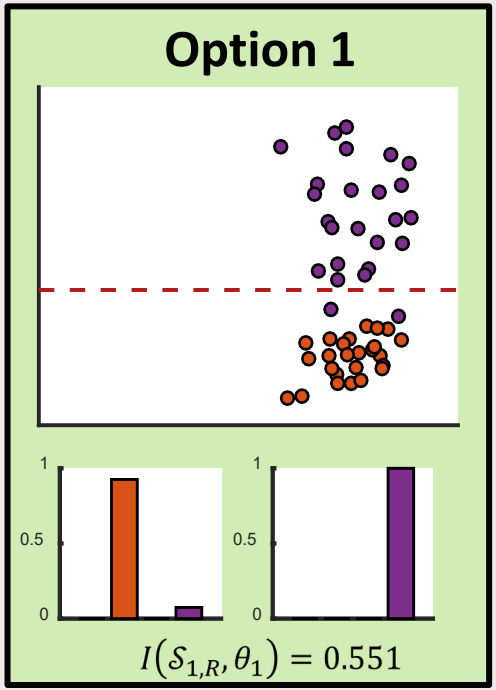
Training trees

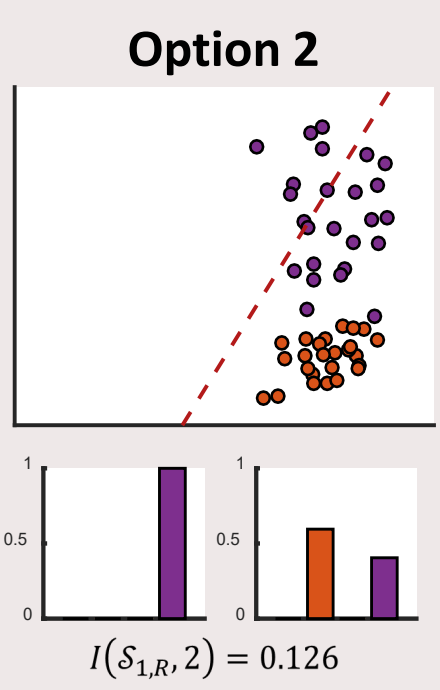
We randomly get three options:

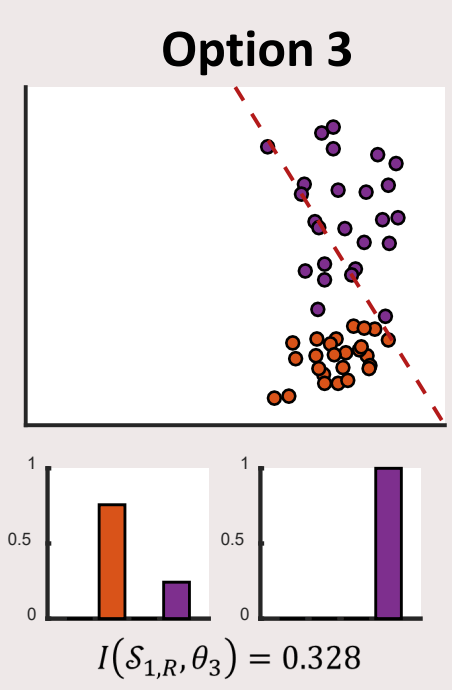
Training node 2



$|\mathcal{T}_j| = 3$



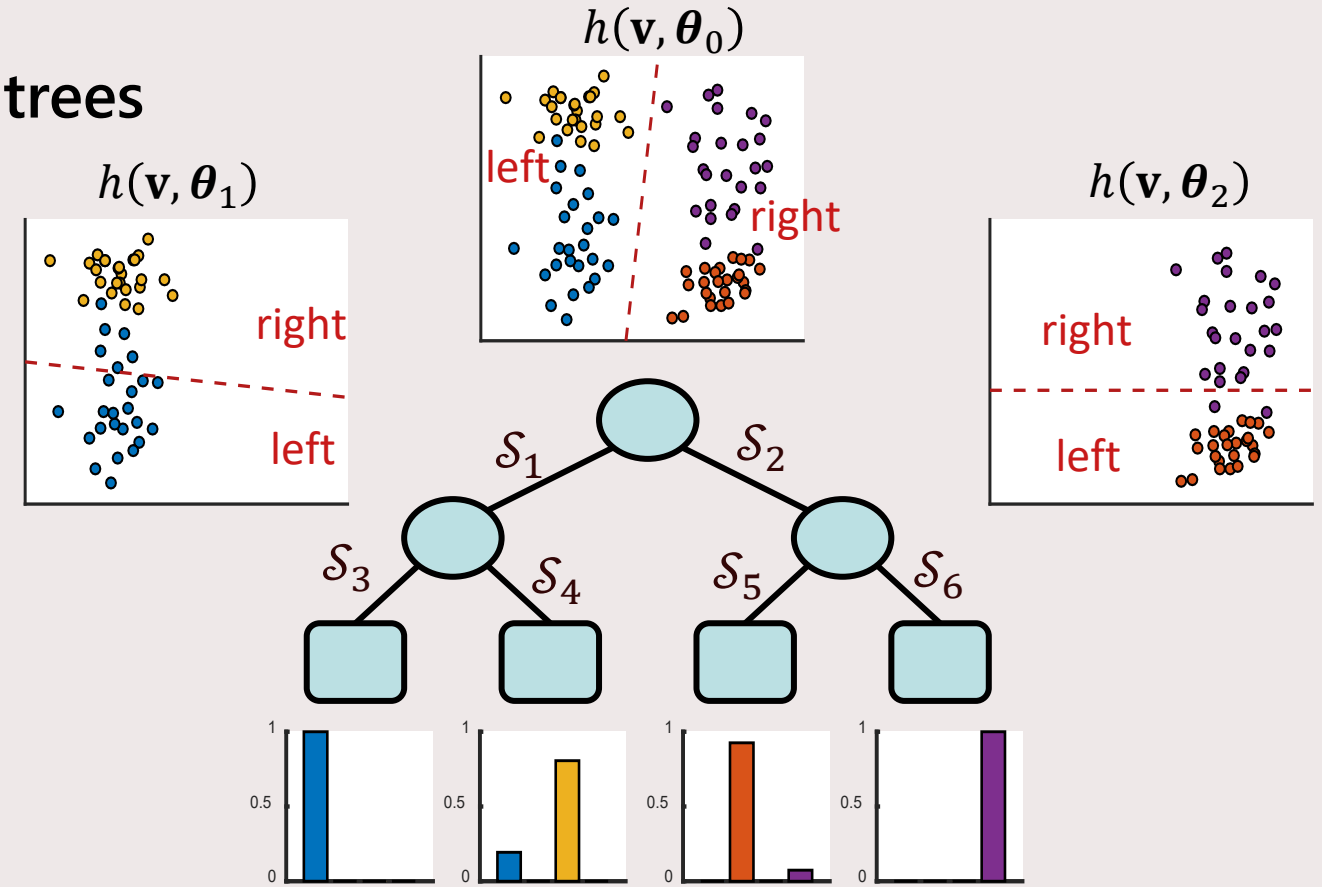




TU/e

EXAMPLE

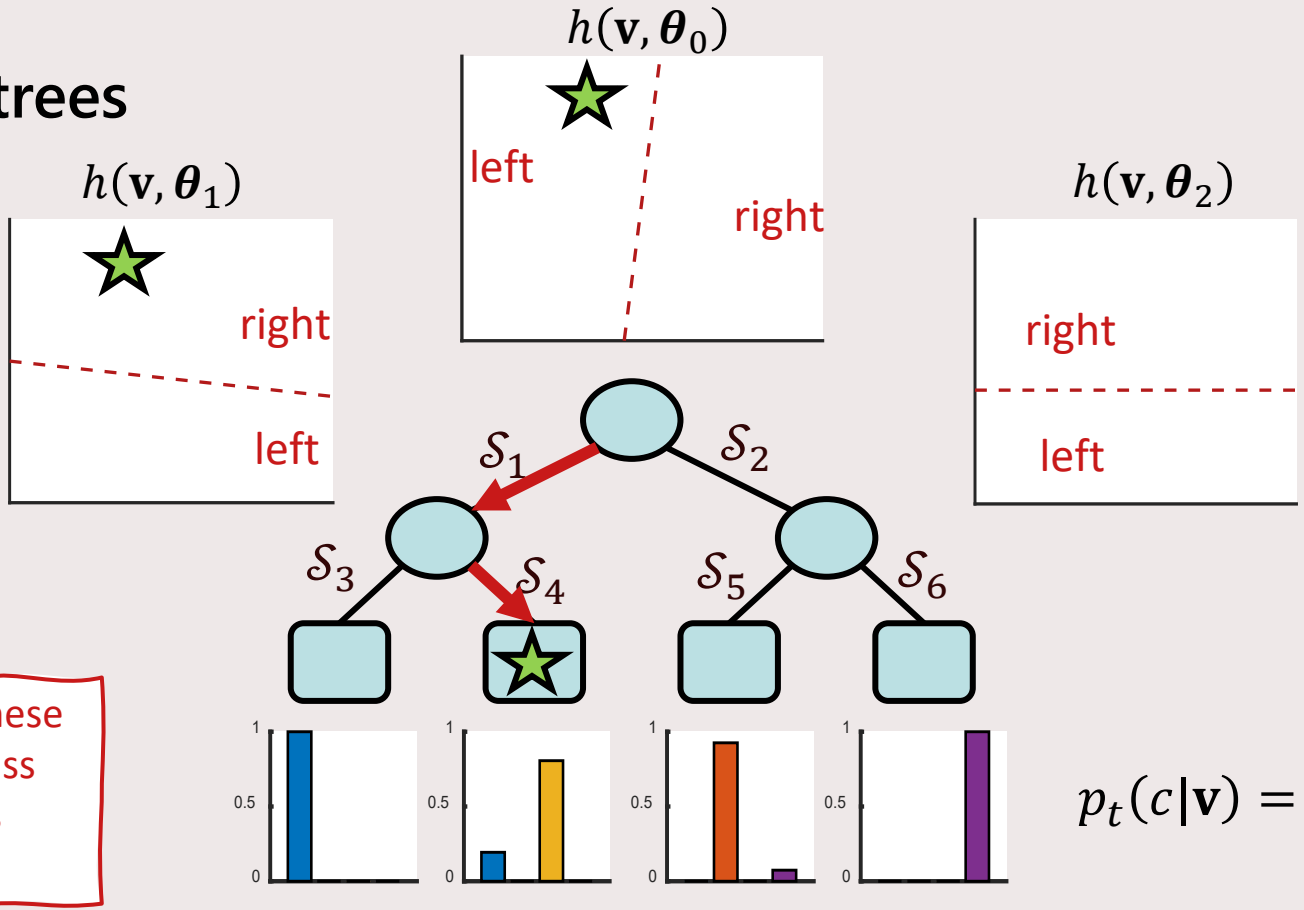
Training trees



TU/e

EXAMPLE

Training trees



Where do these posterior class probabilities come from?

$$p_t(c|\mathbf{v}) = \begin{bmatrix} 0.19 \\ 0 \\ 0.81 \\ 0 \end{bmatrix}$$

TU/e

Pros/cons of decision trees

+ Easy to explain

+ Naturally multiclass

+ Fast

+ Interpretable

- Low predictive power

- High variance

Random Forest

The sales pitch…

- *General* model for machine learning
  - Density estimation, regression, classification, …
- Robustness through randomness
  - A random subset is used to train each tree
  - For training a tree, each node receives a random set of split options
- Probabilistic output → model uncertainty!
- Automatic feature selection → interpretable!
- Trees can run in parallel → efficient!

The forest model

General idea

- Train a large number of diverse trees
- Combination of a large set of weak learners makes a strong learner
- Randomness as a powerful regularizer!

From trees to forest

- How many trees do we need?
- How do we aviod training a forest of identical trees?
- How do we combine tree predictions?

Even more hyperparameters…

How does randomness act as a regularizer? (and counters overfitting)

The forest model

How to add randomness?

- **Bagging (randomized training set)**
  - Bootstrap aggregating (Leo Breiman, 1994)
  - Subset of all data points per tree
- **Randomized Node Optimization (RNO)**
  - Features *chosen with selection function* $\boldsymbol{\phi}(\boldsymbol{v})$
  - Split function *depending on weak learner orientation* $\boldsymbol{\psi}$
  - Thresholds *given in* $\boldsymbol{\tau}$

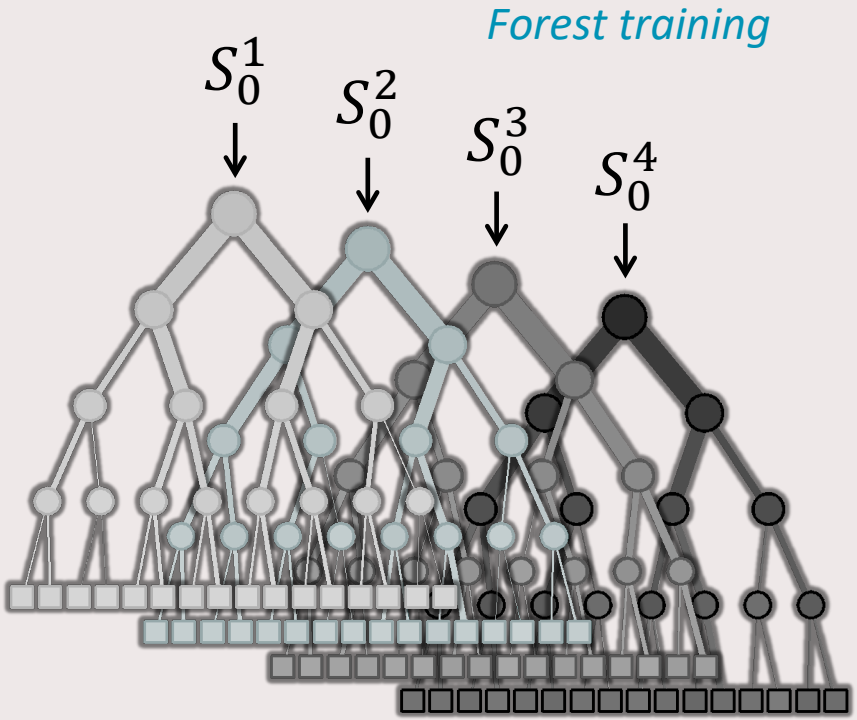
Randomness model

How to add randomness?

(1) Bagging

S_0: Full training set

$S_0^t \subset S_0$: Randomly sampled subset for training tree t



Randomness model

$$\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\tau}\} \in \mathcal{T}_j$$

How to add randomness?

(2) Randomized Node Optimization (RNO)

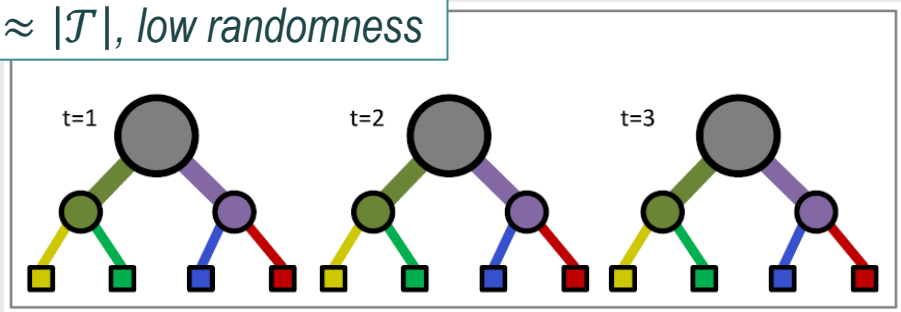
What happens if $\rho \approx |\mathcal{T}|$ and why is this bad?;)

$\mathcal{T}$: Full set of all possible node test parameter values

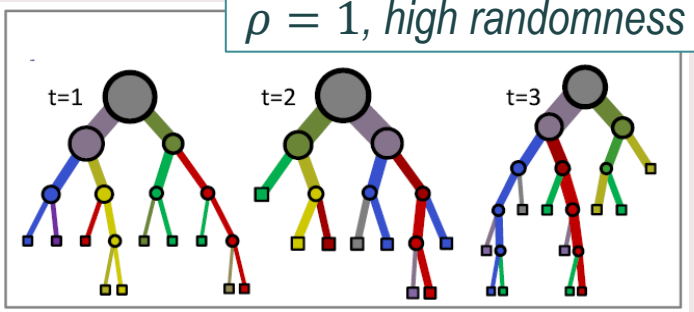
$\mathcal{T}_j \subset \mathcal{T}$: Set of randomly sampled parameter values to train node j

$\rho = |\mathcal{T}_j|$: Randomness control parameter

$\rho \approx |\mathcal{T}|$, *low randomness*



$\rho = 1$, *high randomness*



TU/e

Forest prediction

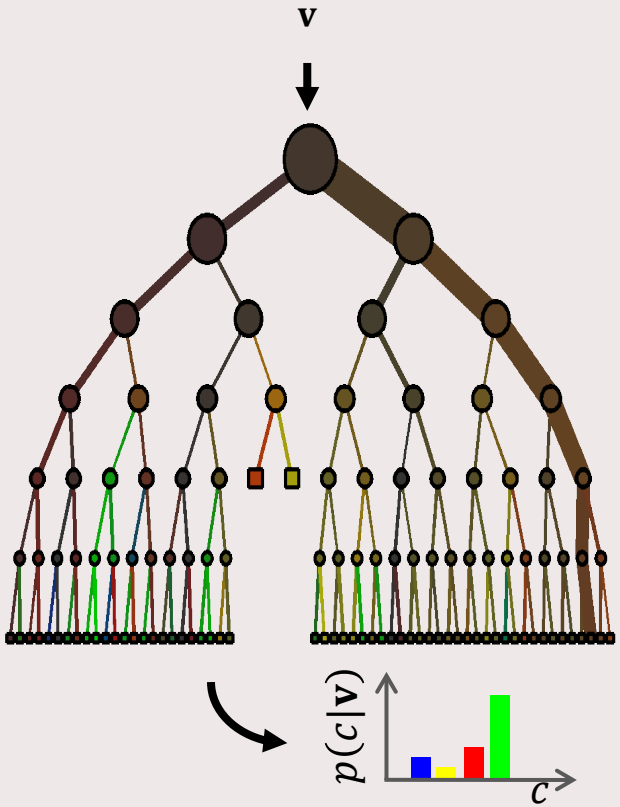
Tree prediction

- Probability distribution at leaf: $p(c|\mathbf{v})$
- Point-estimate, e.g. M.A.P.:

$$c^* = \arg\max_c p(c|\mathbf{v})$$

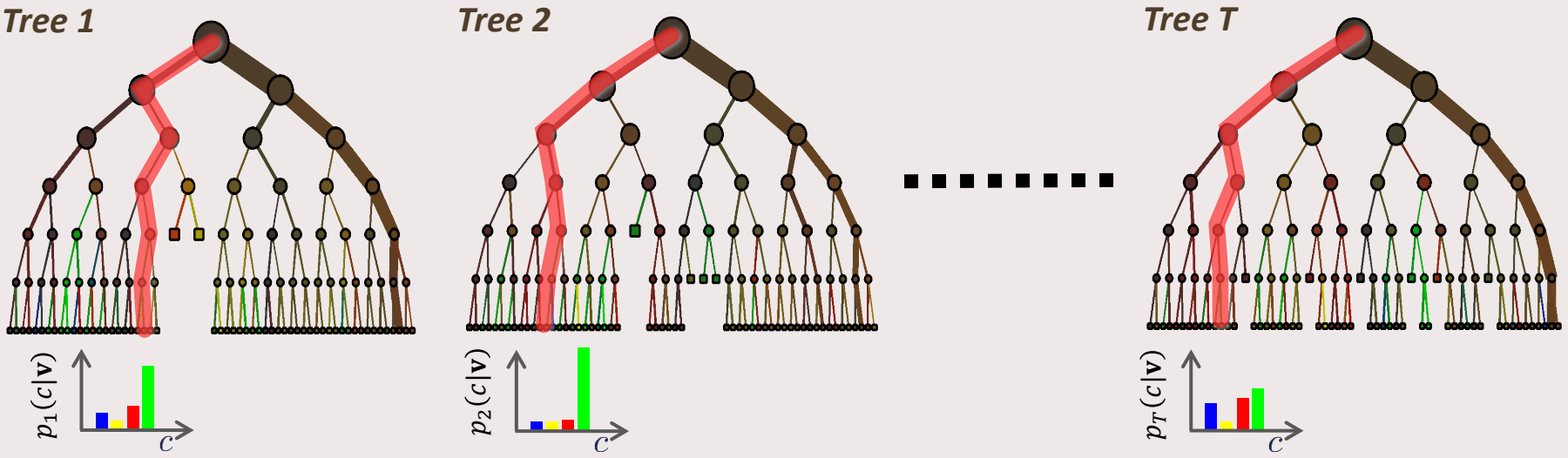
Generally the full distribution is perserved untill the decision moment

Why?



TU/e

Forest prediction



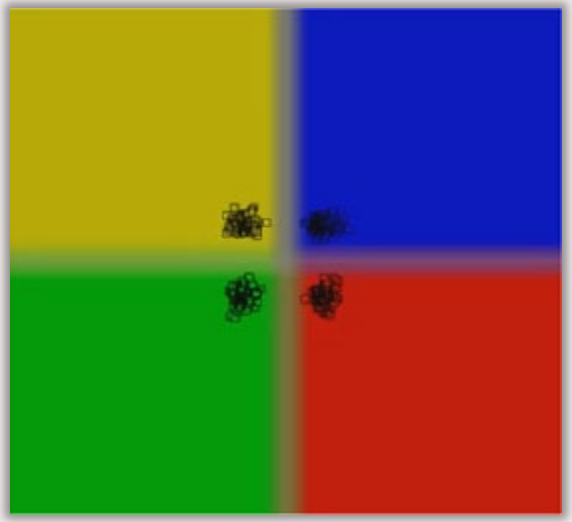
- **Averaging:** $p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$

TU/e

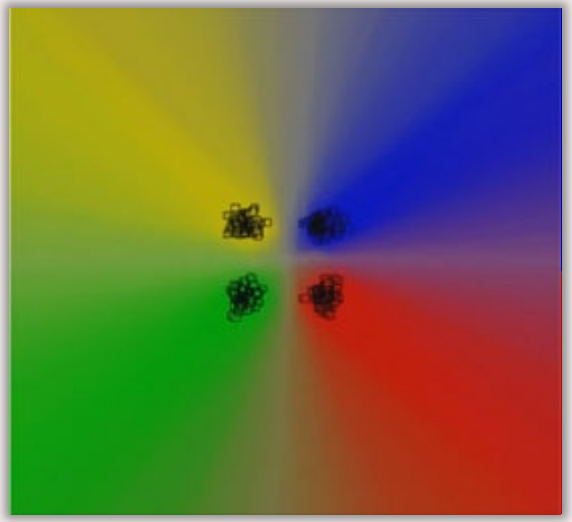
Some visual examples

What is good/bad behaviour?

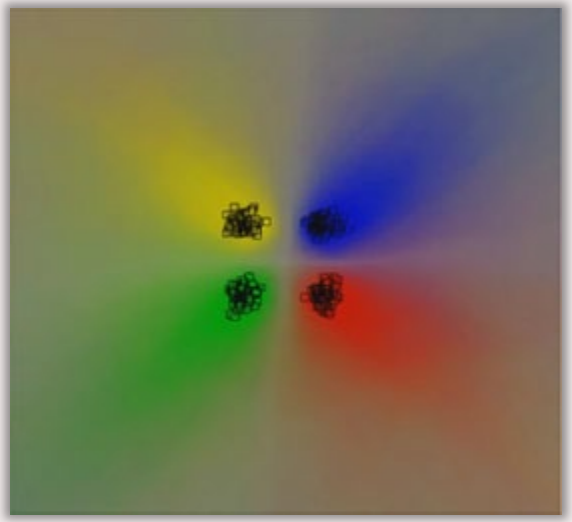
The effect of the weak learner model



Axis aligned



Oriented line



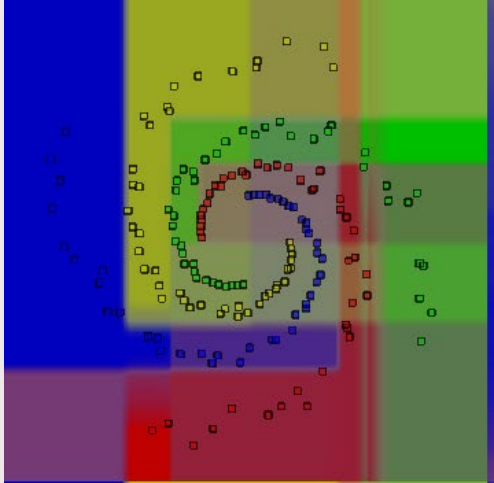
Conic section

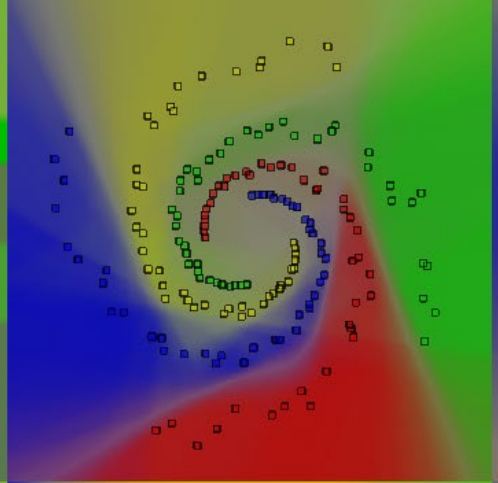
The effect of hyperparameters

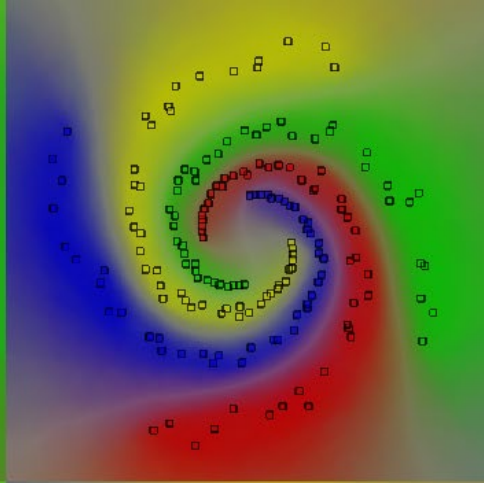
$\rho = 500$

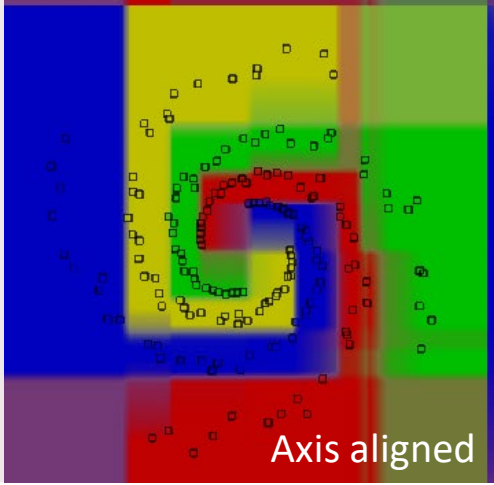
D = 5

D = 13

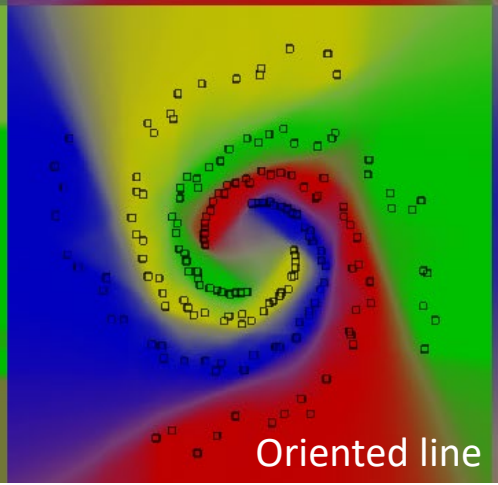




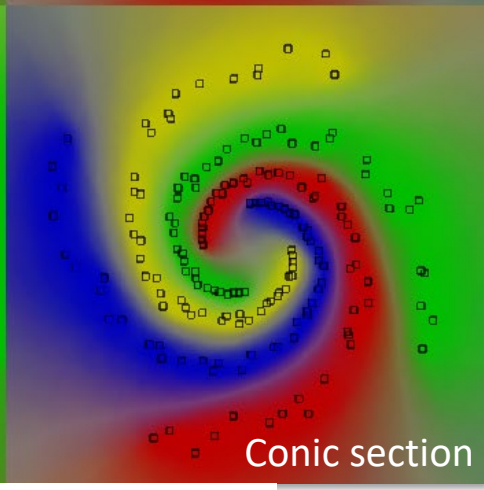




Axis aligned



Oriented line



Conic section

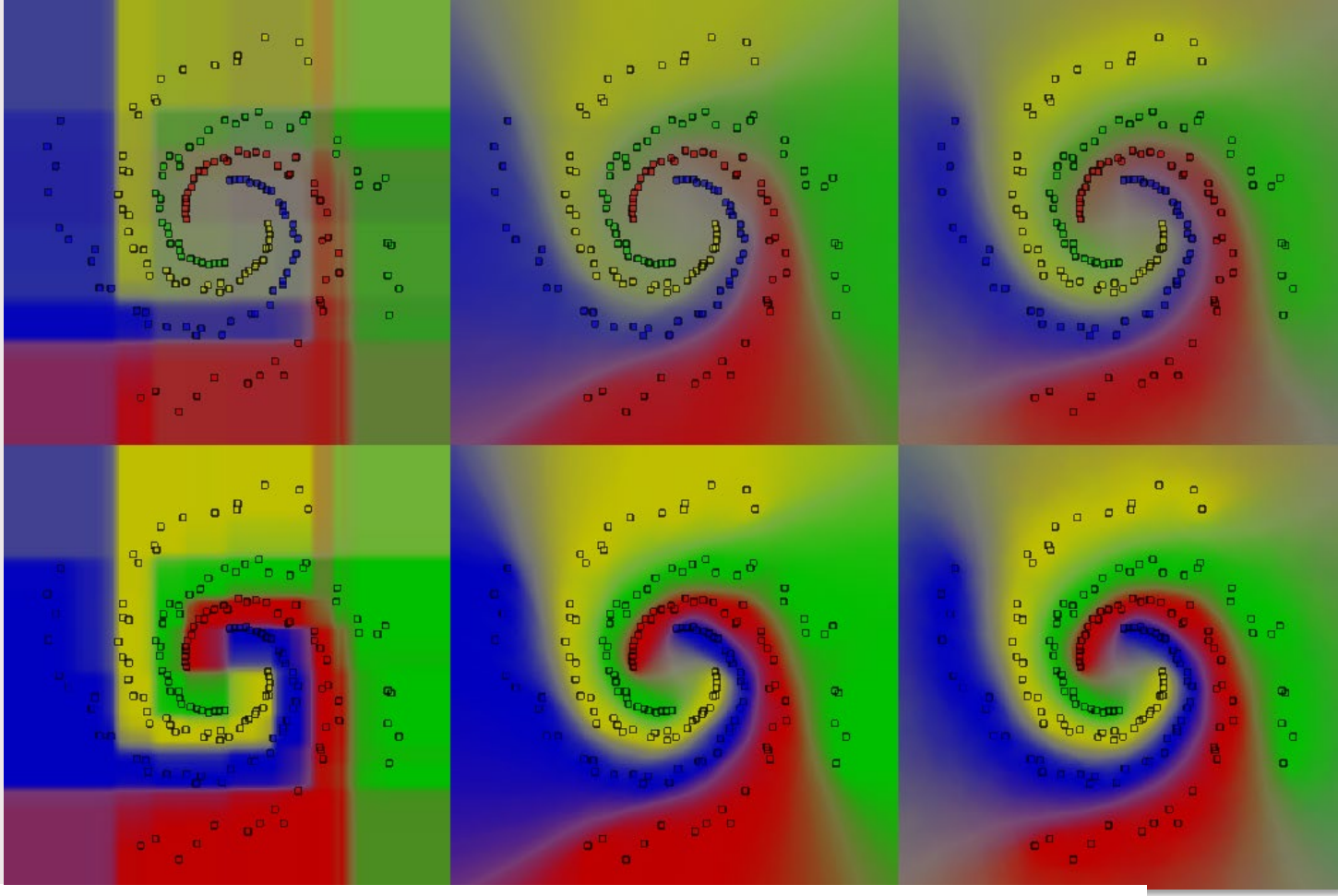
What happens when D increases?

TU/e

The effect of hyperparameters

for lower ρ

$\rho = 50$

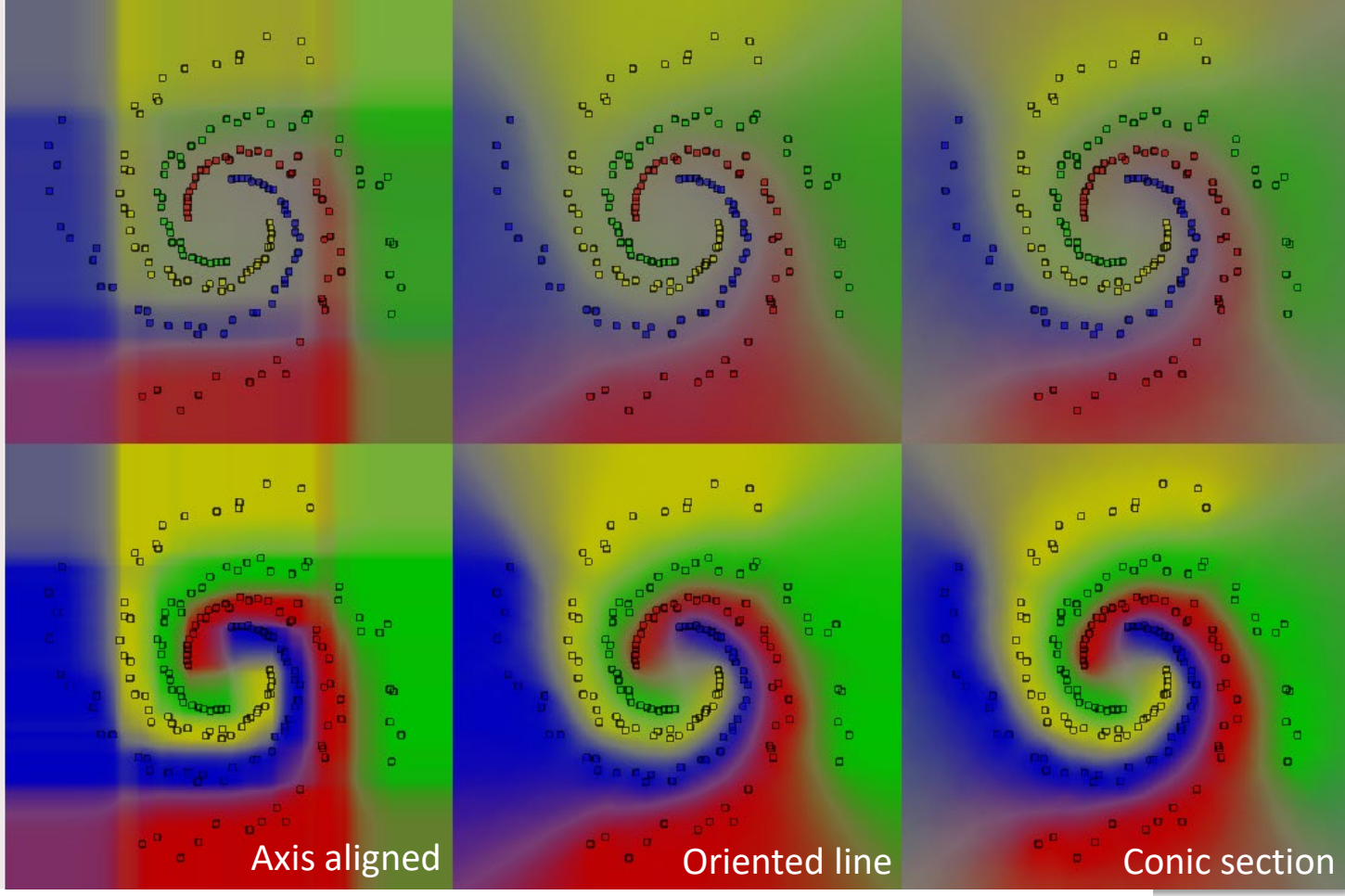


The effect of hyperparameters

For even lower ρ

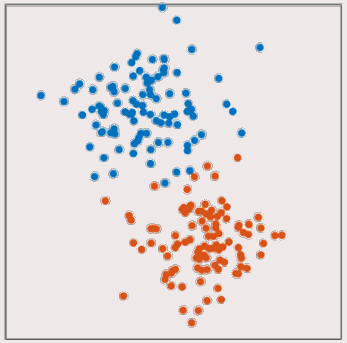
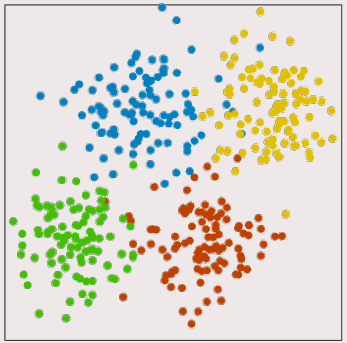
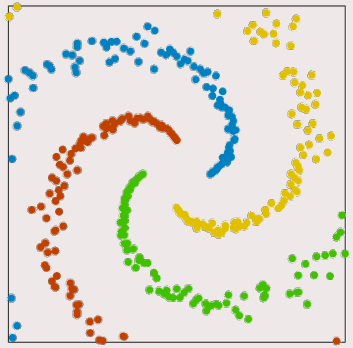
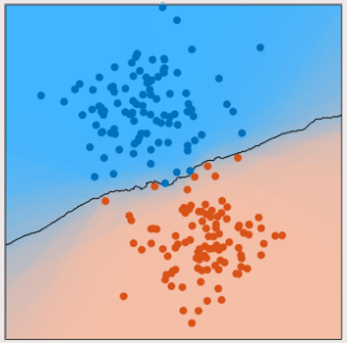
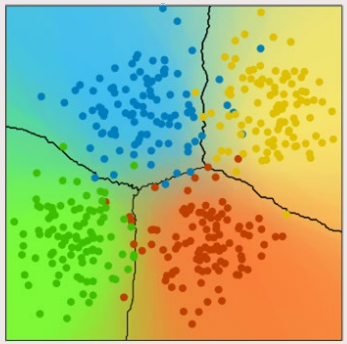
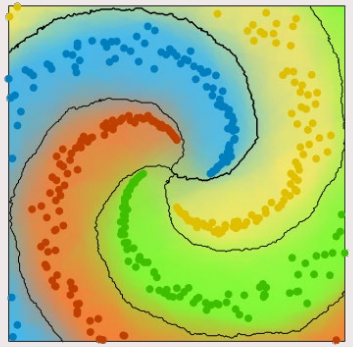
$\rho = 5$

What happens when ρ decreases?



TU/e

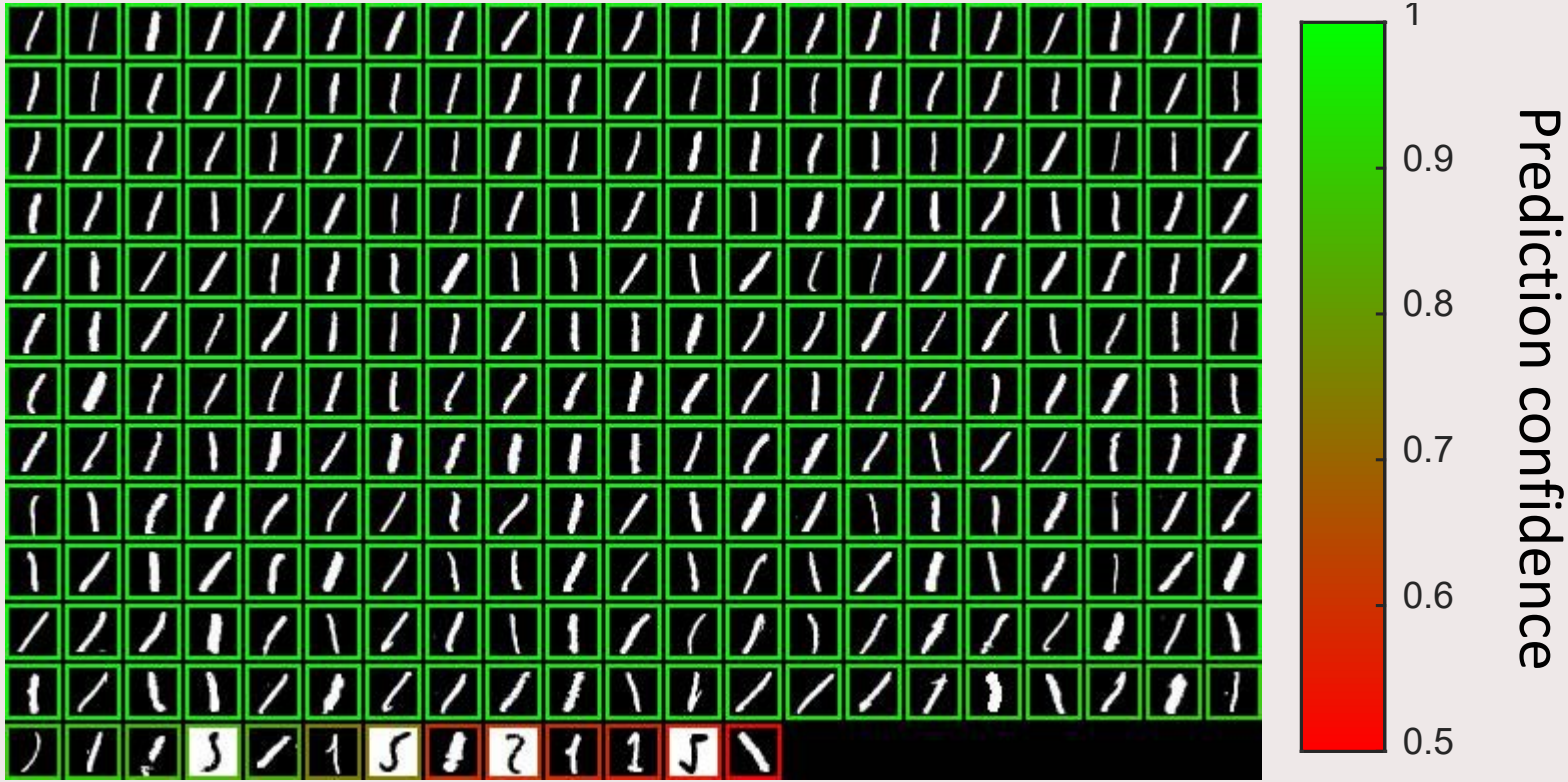
Some visual examples

TU/e

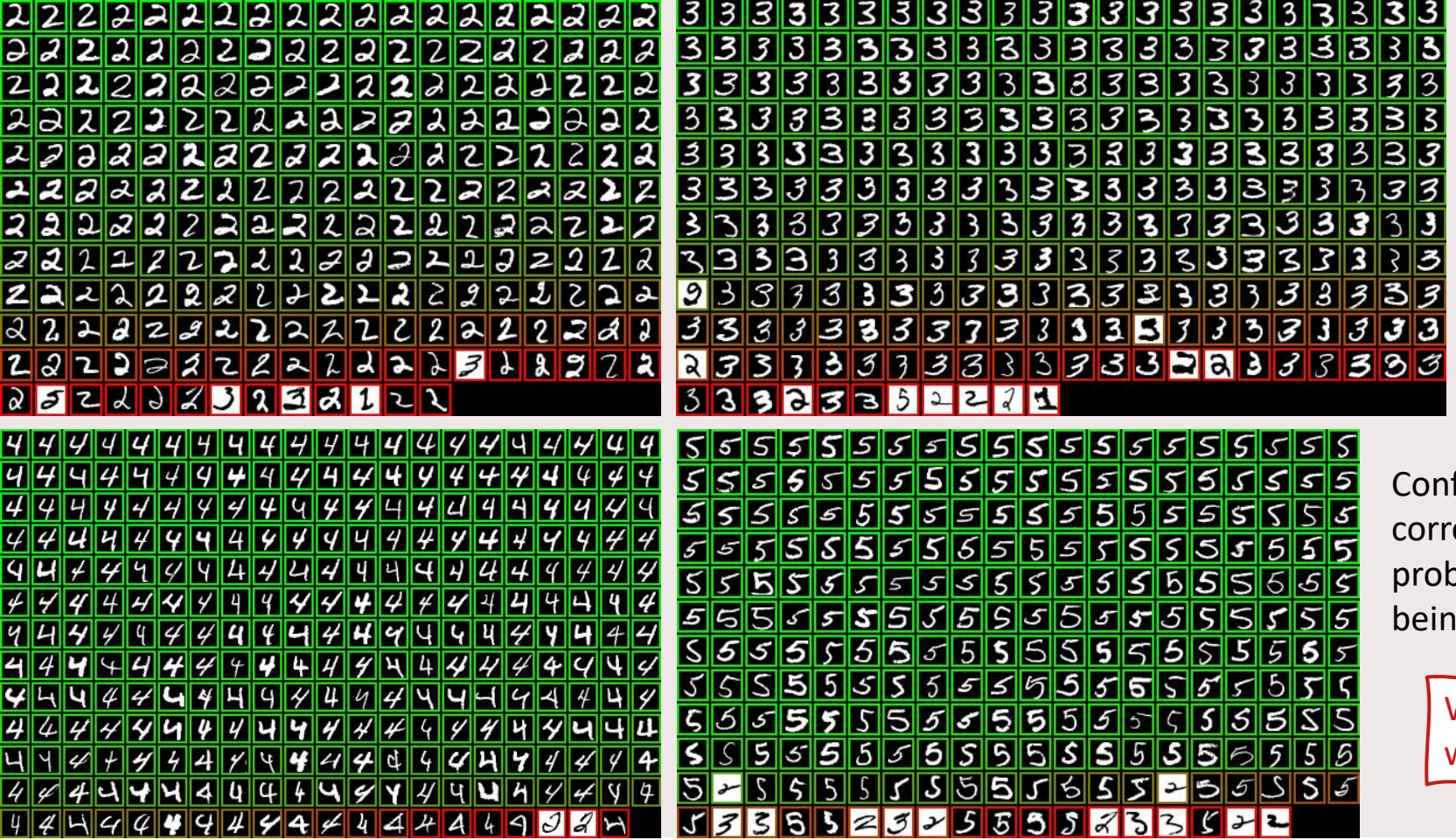
"The MNIST Database of Handwritten Digit Images for Machine Learning Research", DOI:10.1109/MSP.2012.2211477.

TU/e



Precision = 0.96 / Recall = 0.97

Forest predictions for class '1', sorted by confidence. Inverted digits are wrongly classified.



Confidence shows correlation with probability of being correct

Why do we want this?

TU/e

Summary

We can use data-driven modeling to learn input-output relations

Supervised learning!

This can be done with a simple linear function

- Use the logistic function to map class scores to [0,1]
- Also works for multi-class! Softmax function map to a probability distribution.

Logistic regression

Data is hardly ever linearly separable..

- Use non-linear supervised learning!
- The additional flexibility also comes at a cost..

Decision trees

Random forest

More methods to follow!