3D geoinformation
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

Lecture

Decision Trees, Random Forest, Data and Features

Shenglan Du

Today's Agenda

- Previous Lecture: Support Vector Machine

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Learning Objective

- Understand the concept of non-linear classifiers
- Understand the principles of tree classifiers
- Be familiar with the tree node impurity measurement
- Understand how to construct random forests from trees
- Be familiar with commonly used feature selection methods
- Apply Sw and Sb metrics to determine feature quality
- Apply train-test-split to evaluate the performance of a model

Today's Agenda

- Previous Lecture: Support Vector Machine

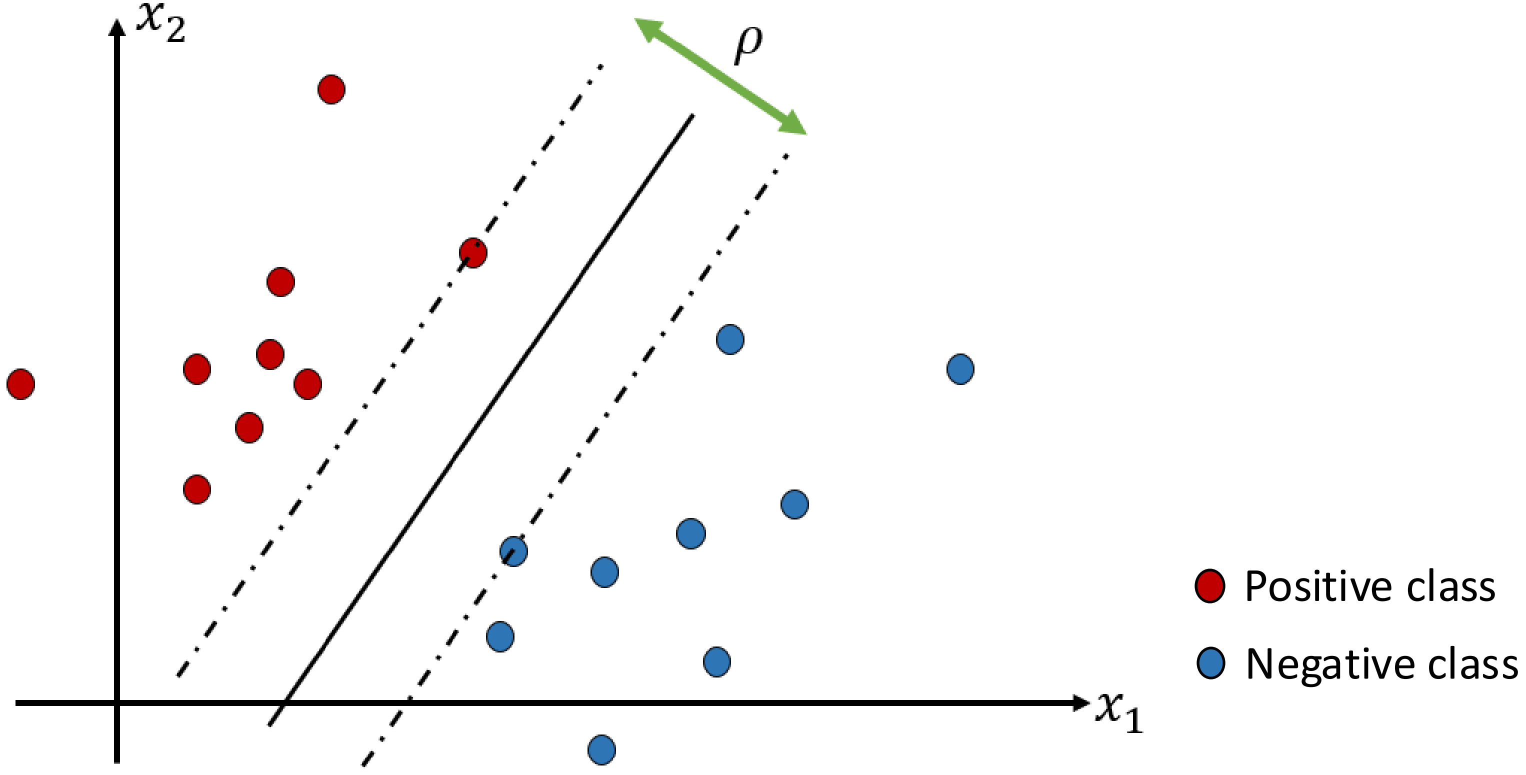
- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Support Vector Machine

- What is the overall goal?

Support Vector Machine



Support Vector Machine

- ***hypothesis***: the decision boundary is a linear model of the input vector $\boldsymbol{x}$:

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0$$

- ***loss***:

$$\min \frac{1}{2} \|\boldsymbol{w}\|^{\mathbf{2}}$$

$$\text{s.t.} \quad y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}} + b\right) - 1 \geq 0 \quad i = 1,2 \dots n$$

SVM Dual Optimization (Optional)

Support Vector Machine: Lagrangian Dual Formulation.

primal problem: (P)

$$\min f(w,b) = \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i(w^T x_i + b) - 1 \geq 0$$

Lagrangian dual: (D)

$$\max g(\lambda) = \inf_{w,b} L(w,b,\lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \lambda_i (y_i(w^T x_i + b) - 1)$$
$$\text{s.t.} \quad \lambda_i \geq 0$$
$$y_i(w^T x_i + b) - 1 \geq 0.$$

$g(\lambda)$ is the minimum attainable function value of L on the space (w, b)

For most convex optimization problems. the primal problem reaches its minimal when the dual problem reaches its maximal. This is the so called "strict duality". See Chap. 5 of "Convex Optimization" for proof details.

Strict duality implies that assume we found optimal w^* and b^* for (P), and optimal λ^* for (D). we have $g(\lambda^*) = f(w^*, b^*) = L(w^*, b^*, \lambda^*)$

Therefore, $\sum_{i=1}^{n} \lambda_i^* (y_i(w^{*T} x_i + b^*) - 1) = 0$

Due to the nonnegativity, $\lambda_i (y_i(w^T x_i + b) - 1) = 0 \quad \forall i = 1, \cdots, n$ holds for optimal λ^*, w^*, b^*. This is the so called "complementary slackness". Note that this is very important for deriving b^*!

Now let's sum up the conditions you need to meet for optimality:

$y_i(w^T x_i + b) - 1 \geq 0 \quad \forall i = 1, \cdots, n$ → original constraints

$\lambda_i \geq 0. \quad \forall i = 1, \cdots, n$ → Lagrangian assumption

$\lambda_i (y_i(w^T x_i + b) - 1) = 0 \quad \forall i = 1, \cdots, n$ → complementary slackness

$\frac{\partial L(w,b,\lambda)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \lambda_i y_i x_i$
$\frac{\partial L(w,b,\lambda)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i y_i = 0$
} → optimality assumption.

The 5 conditions above form the well-known KKT conditions.

Now, let's solve the λ. By making (D) reach its maximal we get (P) reaching its minimal. Inserting $w = \sum_{i=1}^{n} \lambda_i y_i x_i$ and $\sum_{i=1}^{n} \lambda_i y_i = 0$ back to $g(\lambda)$ we formulate (D) as:

$$\max g(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^T x_j$$
$$\text{s.t.} \quad \lambda_i \geq 0 \quad \forall i = 1, \cdots, n$$
$$\sum_{i=1}^{n} \lambda_i y_i = 0$$

This is a Quadratic programming problem with constraints. Many modern solvers can be used to solve it (e.g., Gurobi).

Let's say we get optimal λ^*. From optimality condition we can get

$$w^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i$$

From complementary slackness, we can find out the data samples x_i that has $\lambda_i^* > 0$, and derive b^* by:

$$b^* = y_i - w^{*T} x_i \quad (\text{for } x_i \text{ with } \lambda_i > 0)$$

Some follow-up notations:

1°. How do we use w^* and b^*?

We can use $w^{*T} x + b^*$ for inference. Given a new sample x with unknown label, we use $y = w^{*T} x + b^*$, if $y > 0$ x belongs to class +1, vice versa.

2°. Kernel trick.

Both the optimization objective and the inference contain the dot product $x_i^T x_j$, that's why we only care about the dot product of two feature vectors and why we can directly define the transformation outcome as kernel functions.

3°. what are the support vectors in soft margin SVM?

The Lagrangian derivation of soft Margin SVM is very similar to hard margin SVM. I highly recommend to do it yourself if you are interested.

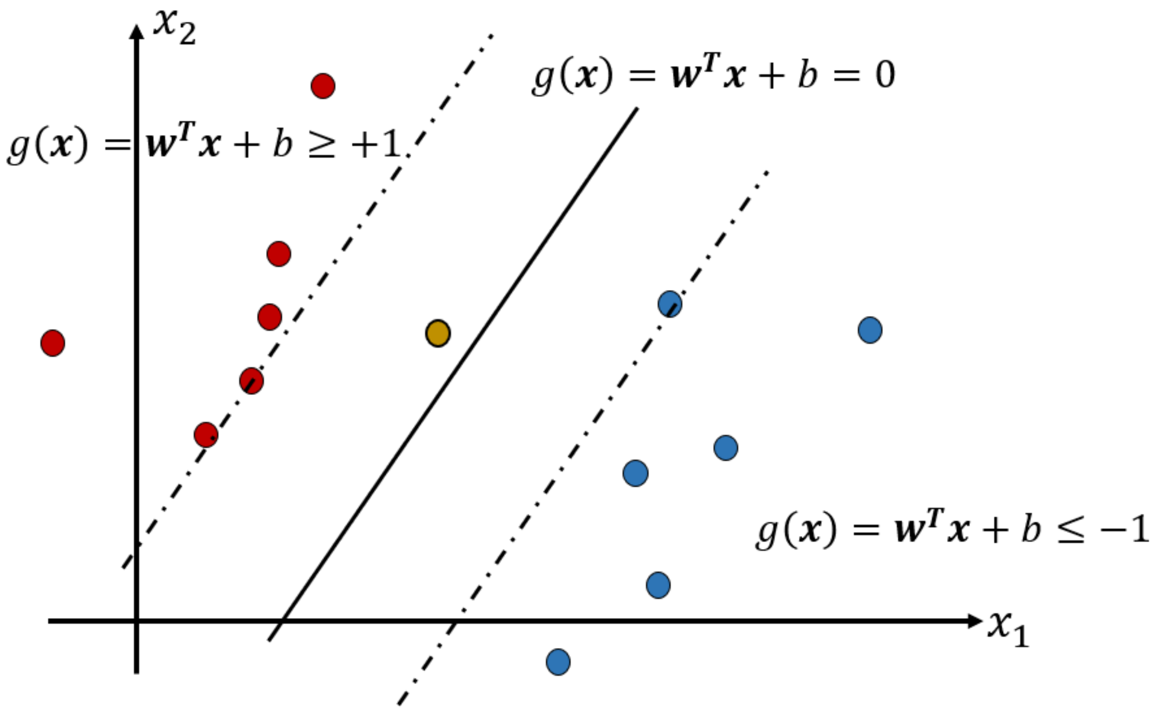
When you obtain KKT conditions for soft margin SVM, complementary slackness would tell you:

$$\lambda_i (y_i(w^T x_i + b) - 1 + \xi_i) = 0.$$ where ξ_i is the slack variable.

It means if $\lambda_i > 0$, $y_i(w^T x_i + b) = 1 - \xi_i$ equality holds. Therefore, those data samples that are both on the margin and misclassified would have influence on w^*. This means the final decision boundary is determined by both margin data samples and wrongly classified samples.

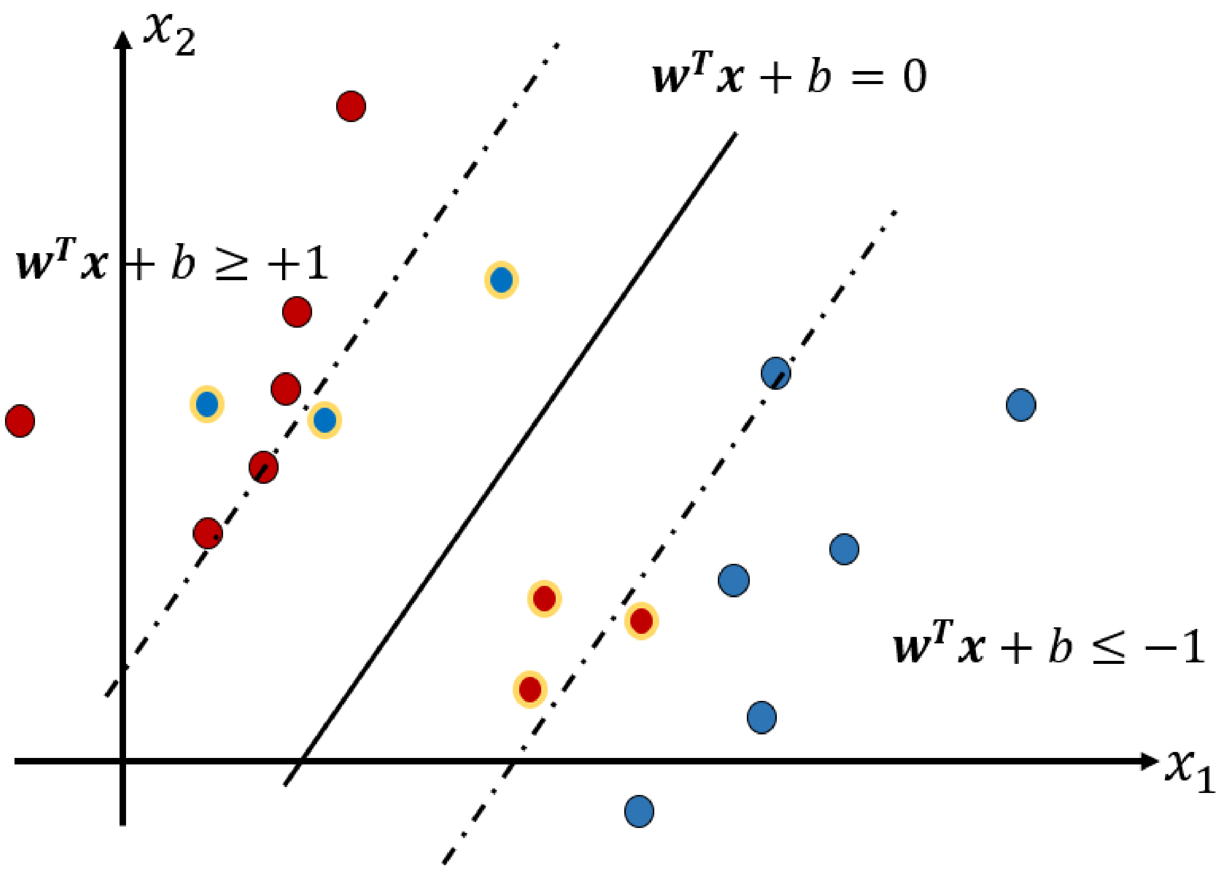
SVM Inference

- I have trained my SVM model $g(\boldsymbol{x})$ on red and blue points, what is the predicted label of the yellow point?
  - Class +1
  - Class -1
  - Class 0.5
  - We can't decide



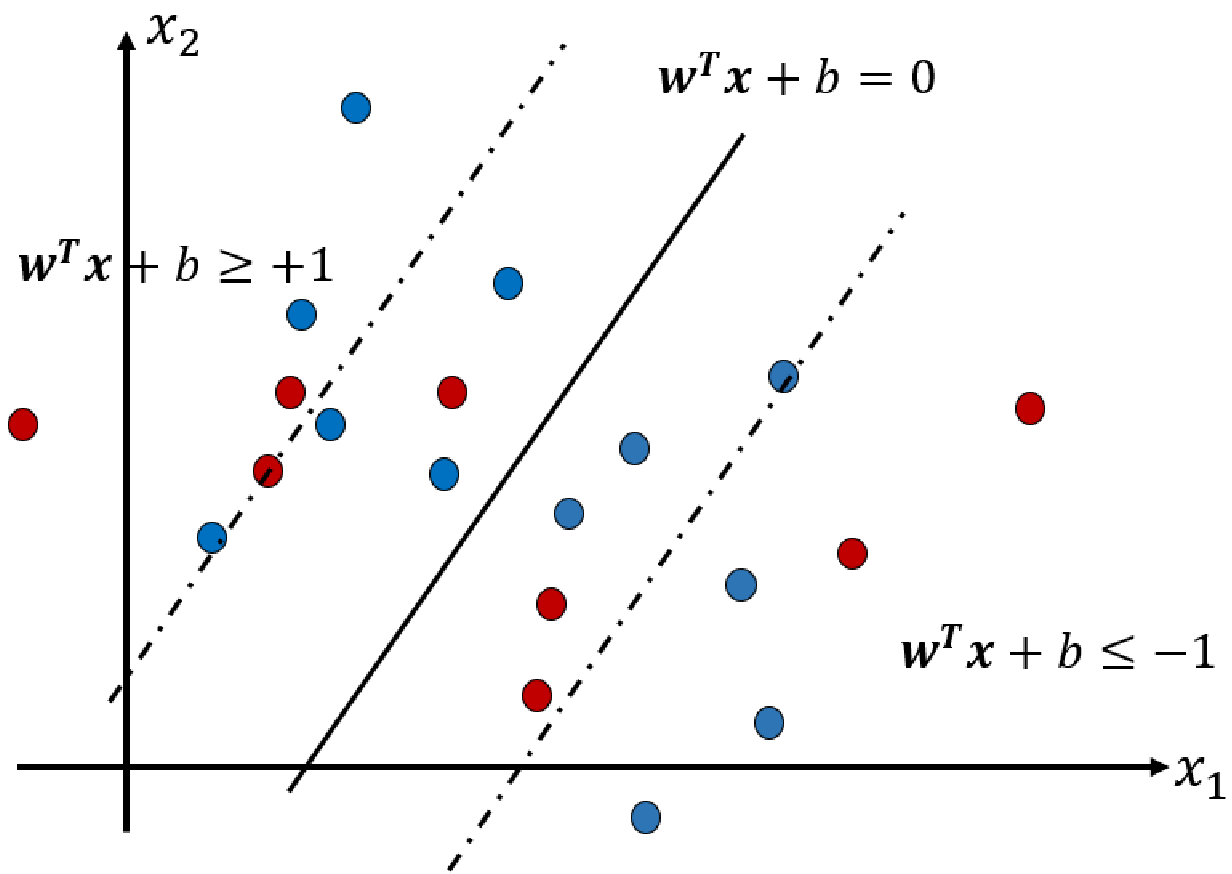
SVM with Soft Margins

- Applied when classes are slightly overlapped



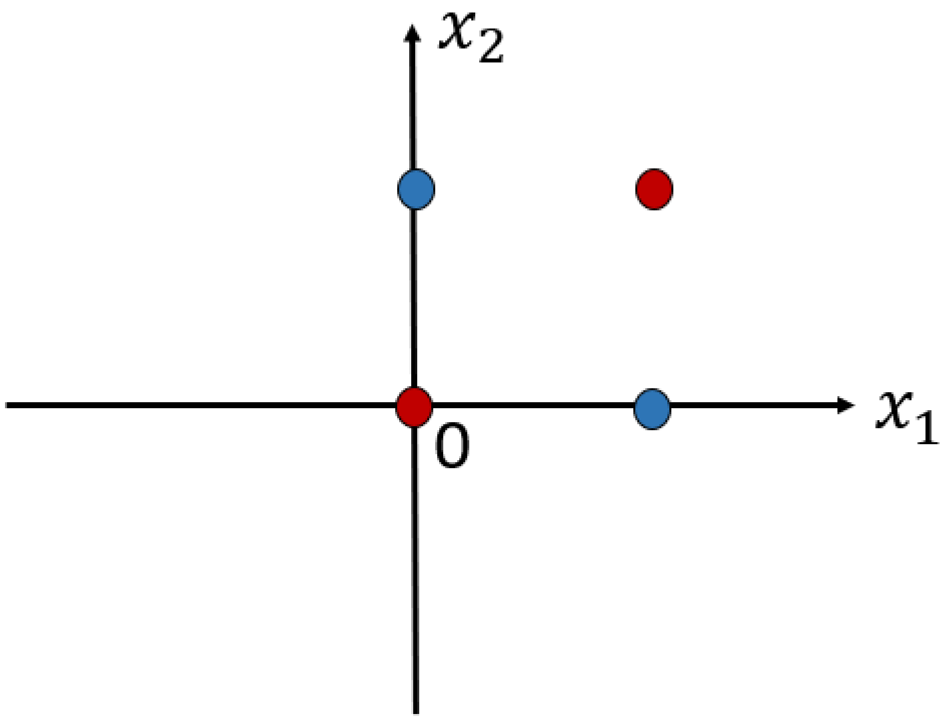
SVM with Soft Margins

- What if the classes highly overlap?



XOR Problem

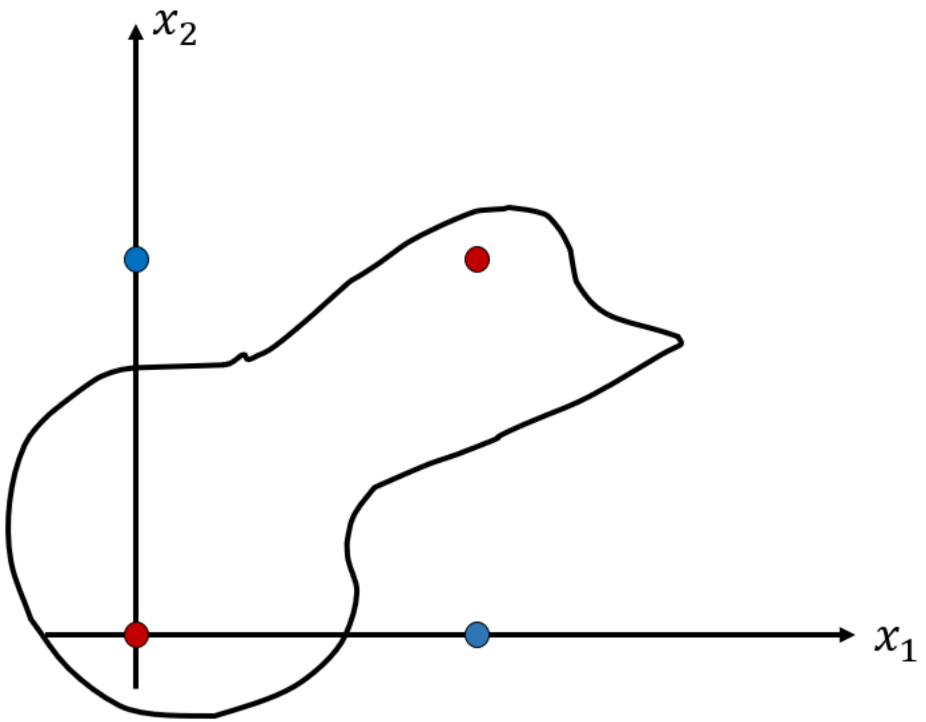
- How to train a SVM (or any models) for such data?



x1	x2	y
0	0	+1
0	1	-1
1	0	-1
1	1	+1

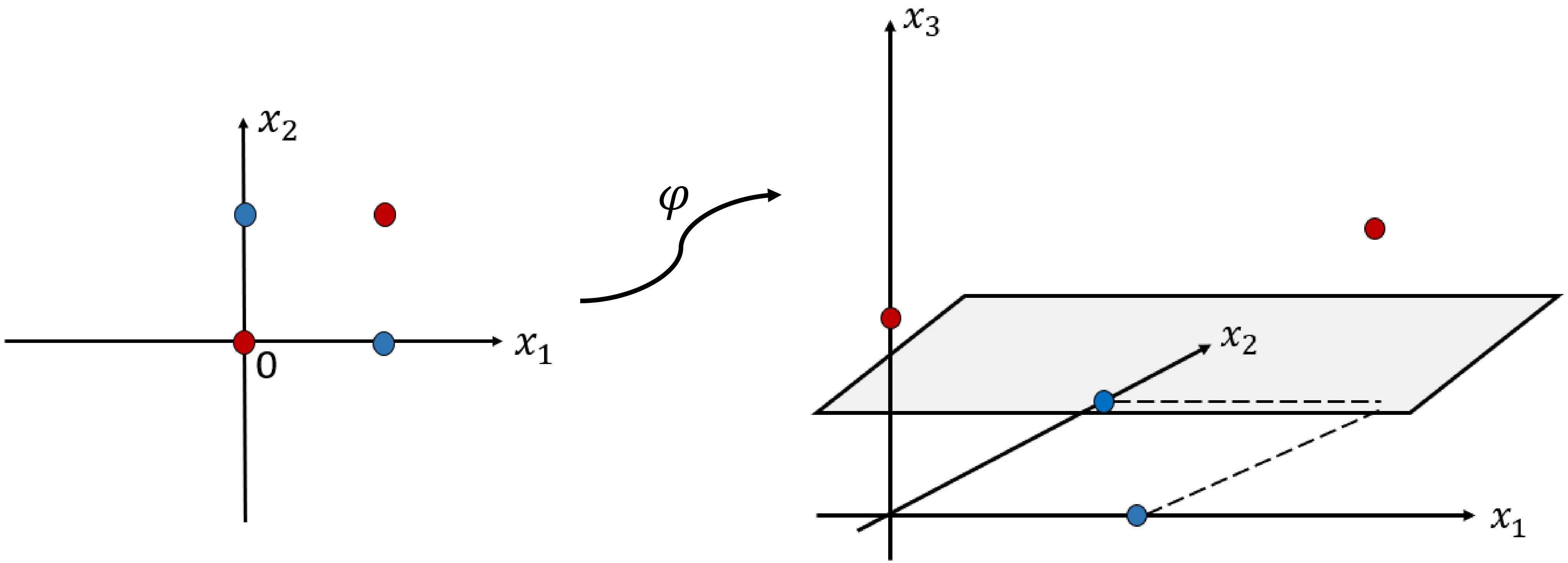
Solution 1

- Assign an arbitrarily complex decision boundary (with magic)



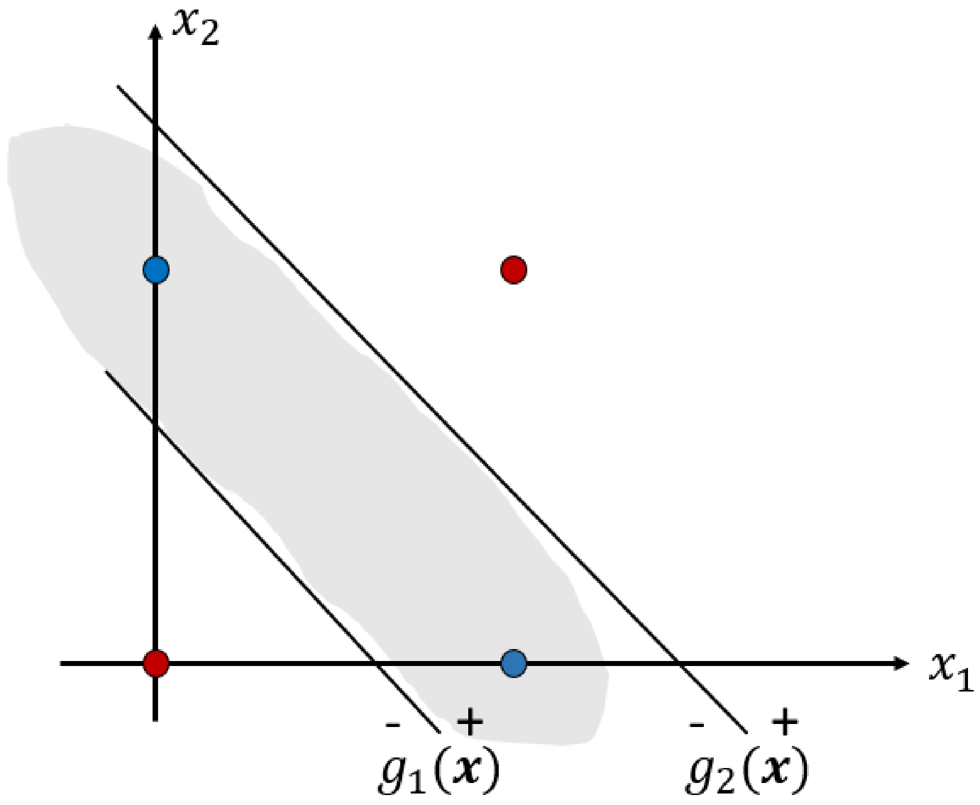
Solution 2

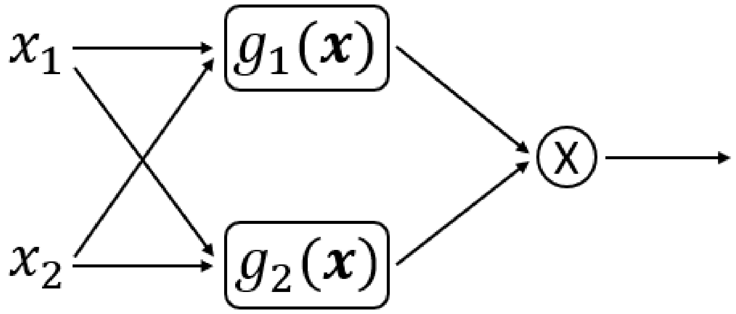
- Apply feature transformation (e.g., kernelize an SVM)



Solution 3

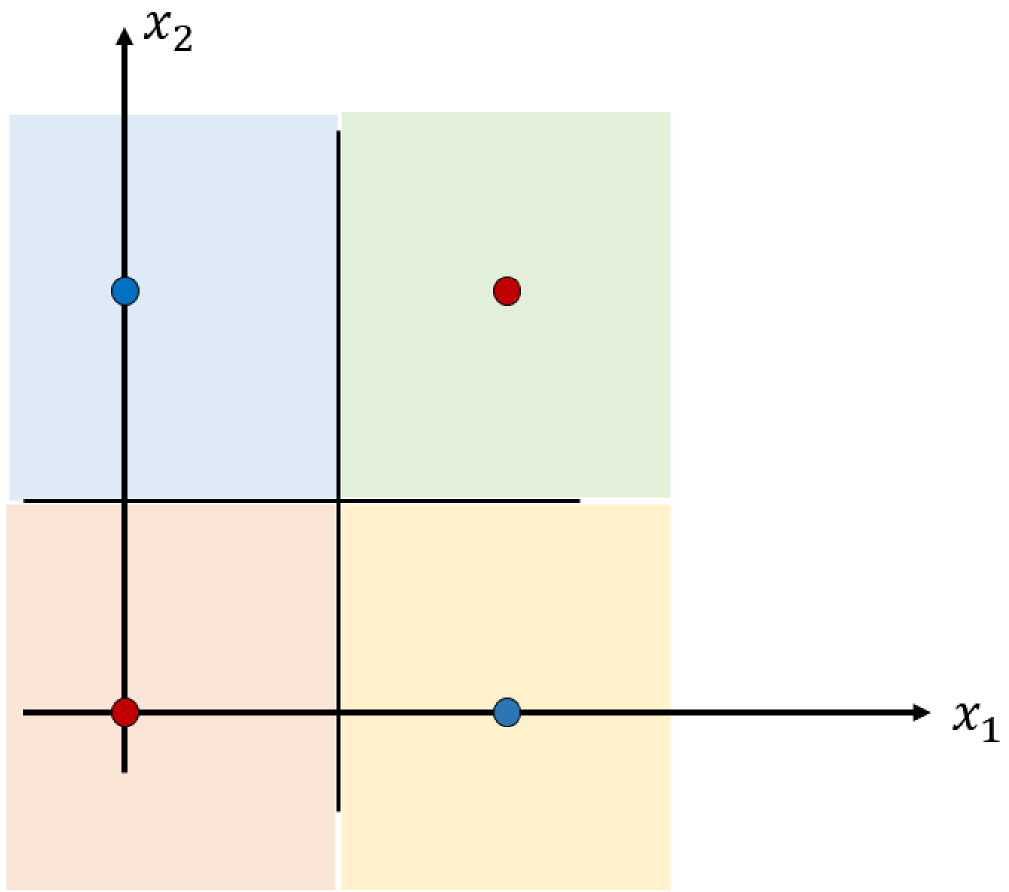
- Combine more than 1 linear models (a prototype of MLP)





Solution 4

- Partition with rectangles



Non-linear classifiers

- designed to cope with non-linearly separable classes

- Commonly used NLCs:
  - Kernelized SVMs
  - Decision trees and random forests
  - Multi-layer perceptron
  - (Deep) Neural network
  - ……

Today's Agenda

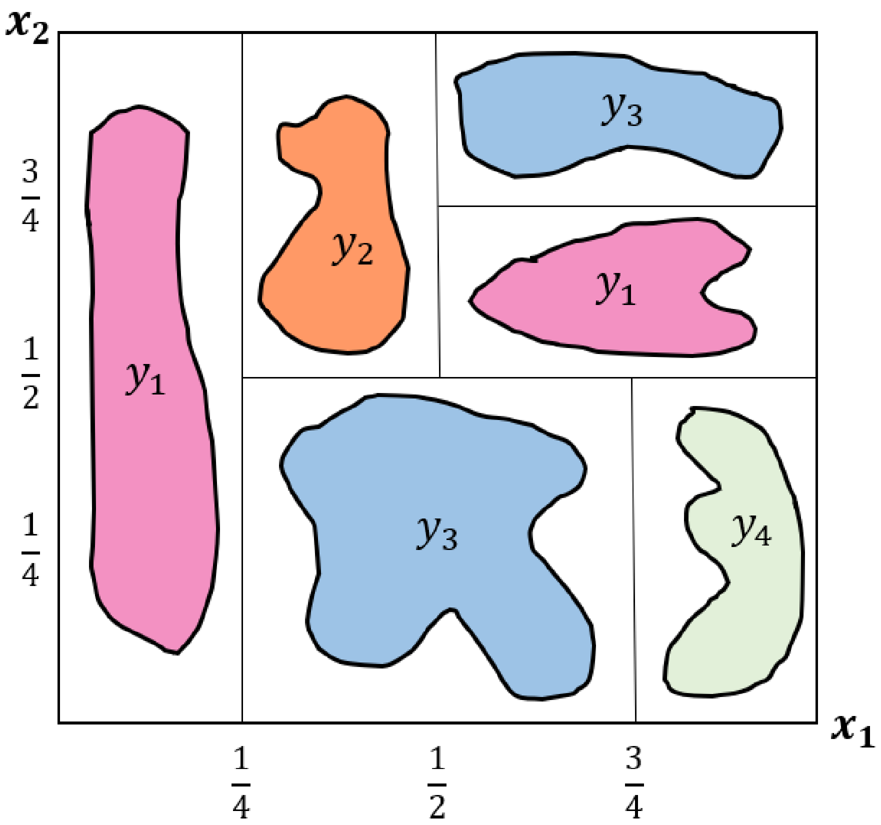
- Previous Lecture: Support Vector Machine

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

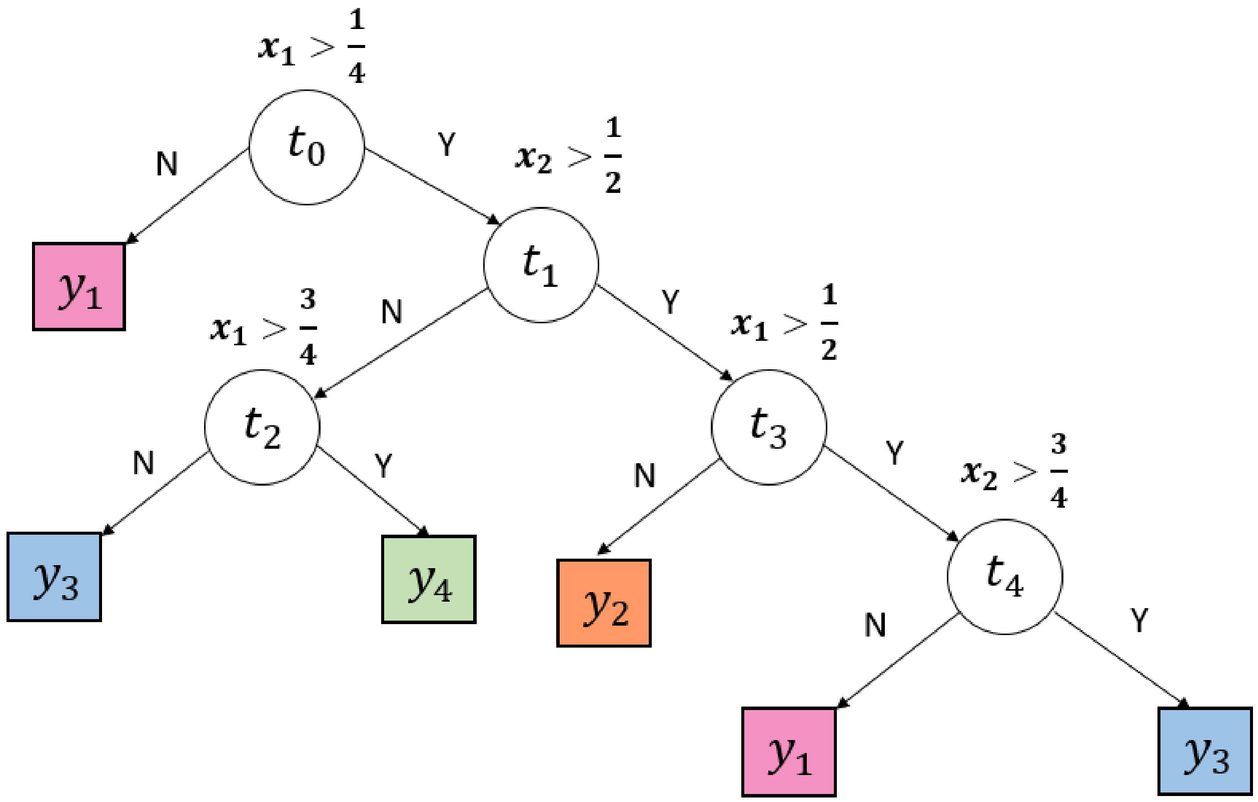
Decision Tree

- The feature space is split into unique regions, corresponding to classes, in a sequent manner



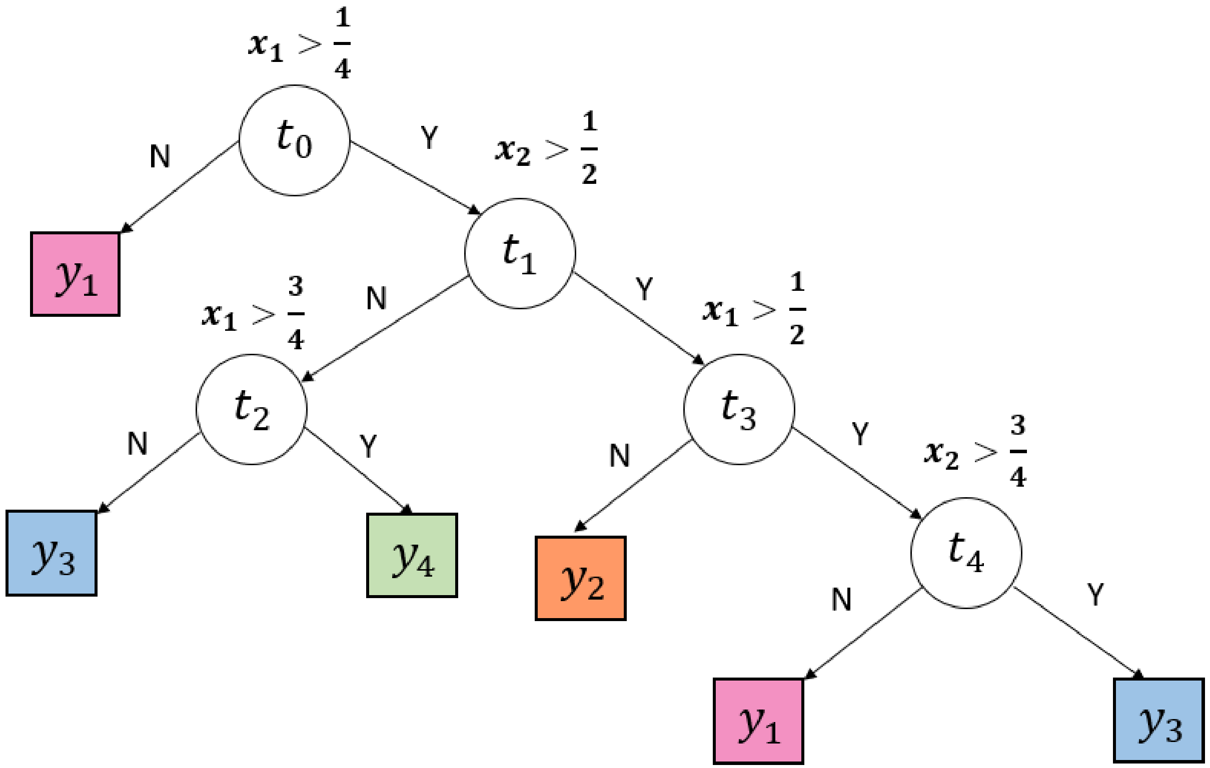
Decision Tree

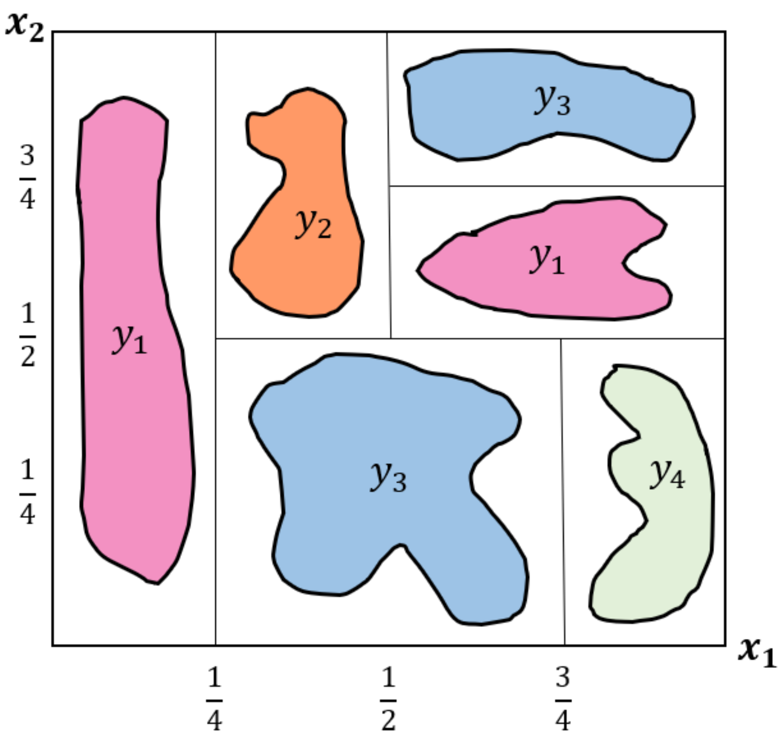
- Classifying of a data sample is done by a sequence of decisions along a path of the tree



Decision Tree

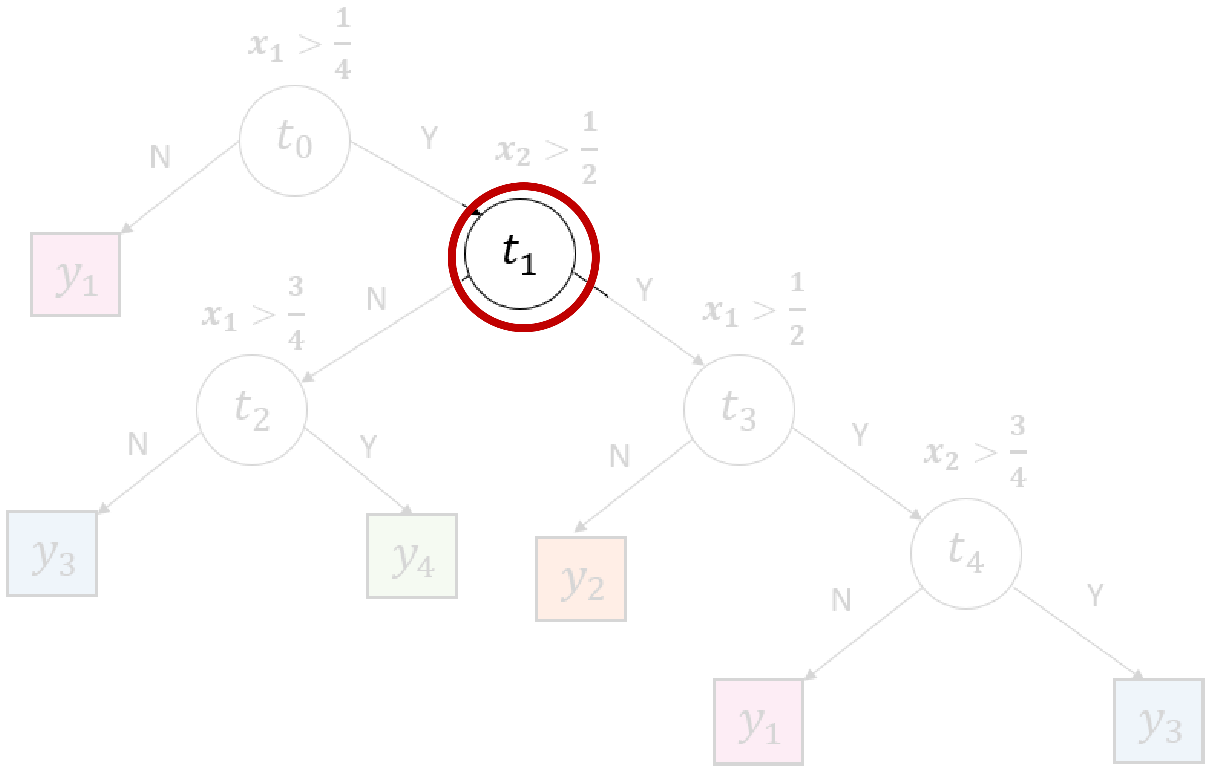
- Splitting rule: every split must generate subsets that are more class homogeneous

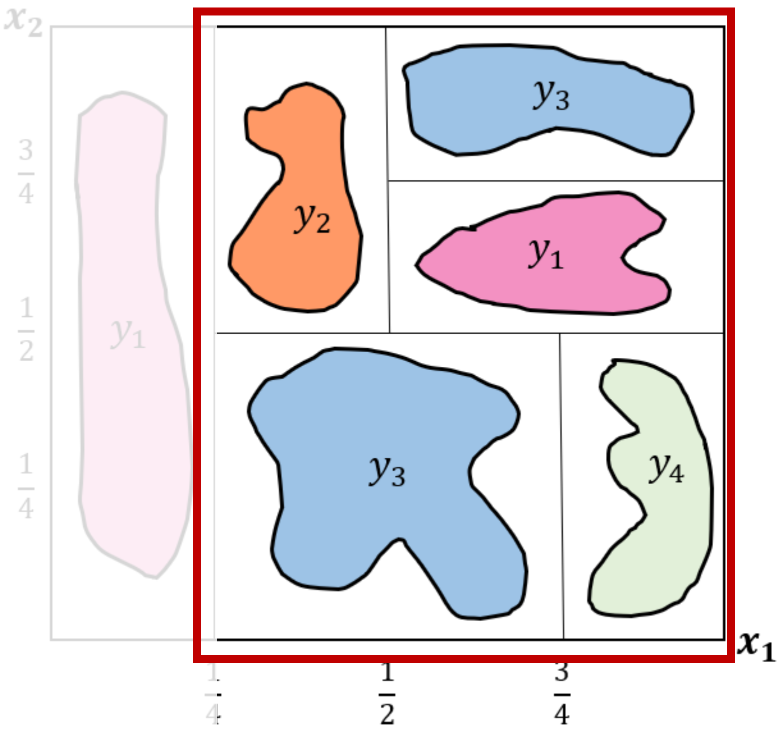




Decision Tree

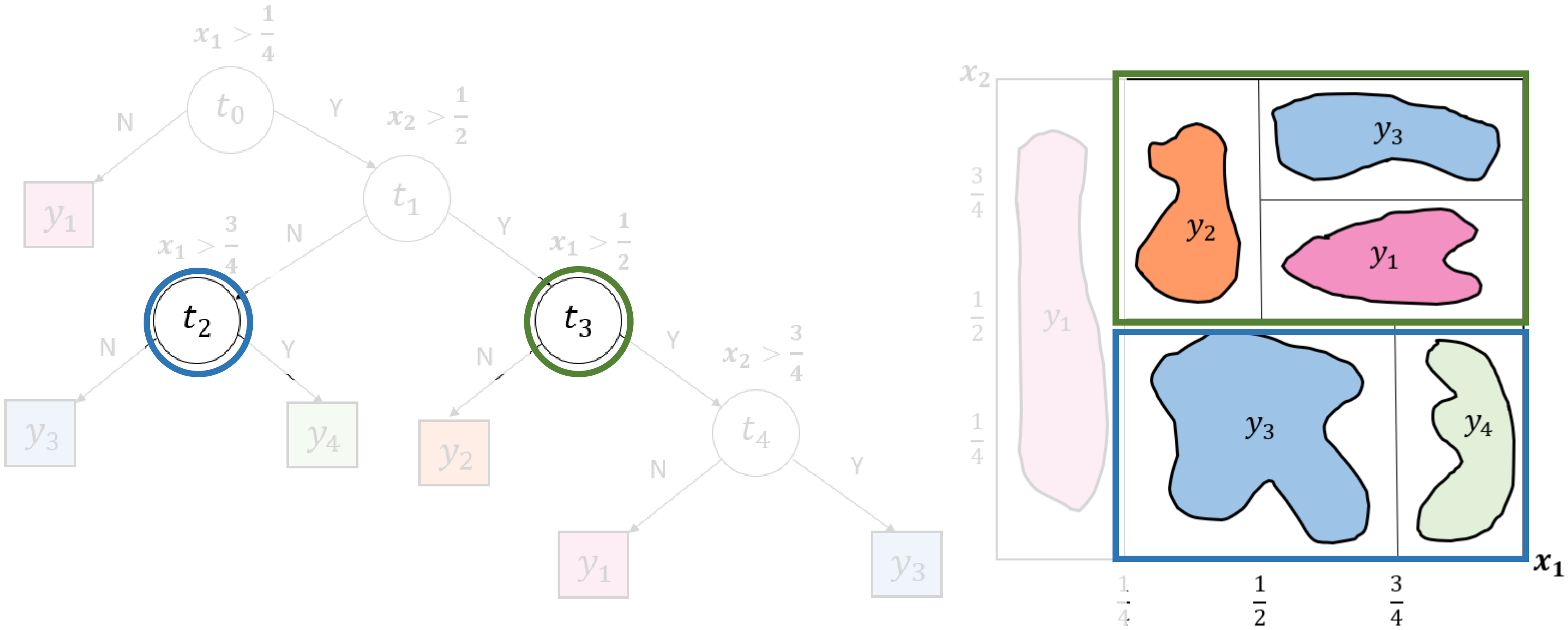
- Splitting rule: every split must generate subsets that are more class homogeneous





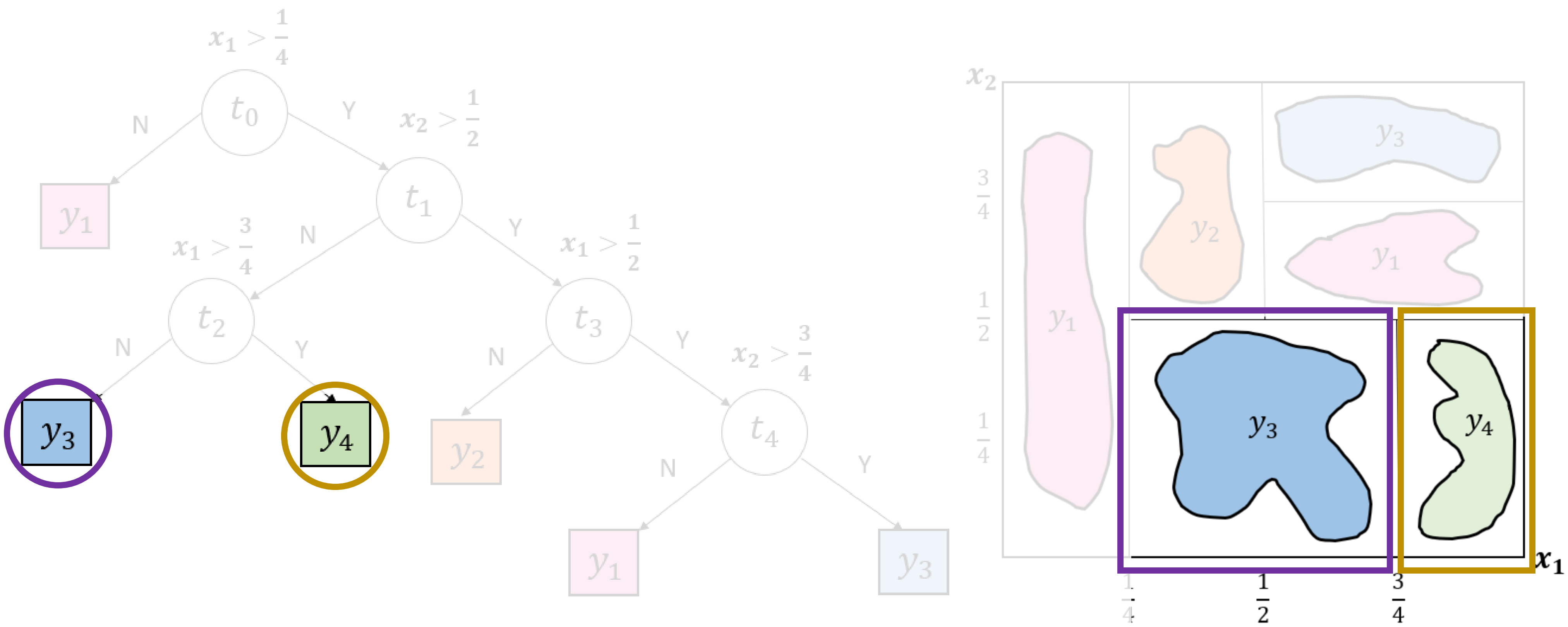
Decision Tree

- Splitting rule: every split must generate subsets that are more class homogeneous



Decision Tree

- Splitting rule: every split must generate subsets that are more class homogeneous



Decision Tree: Node Splitting

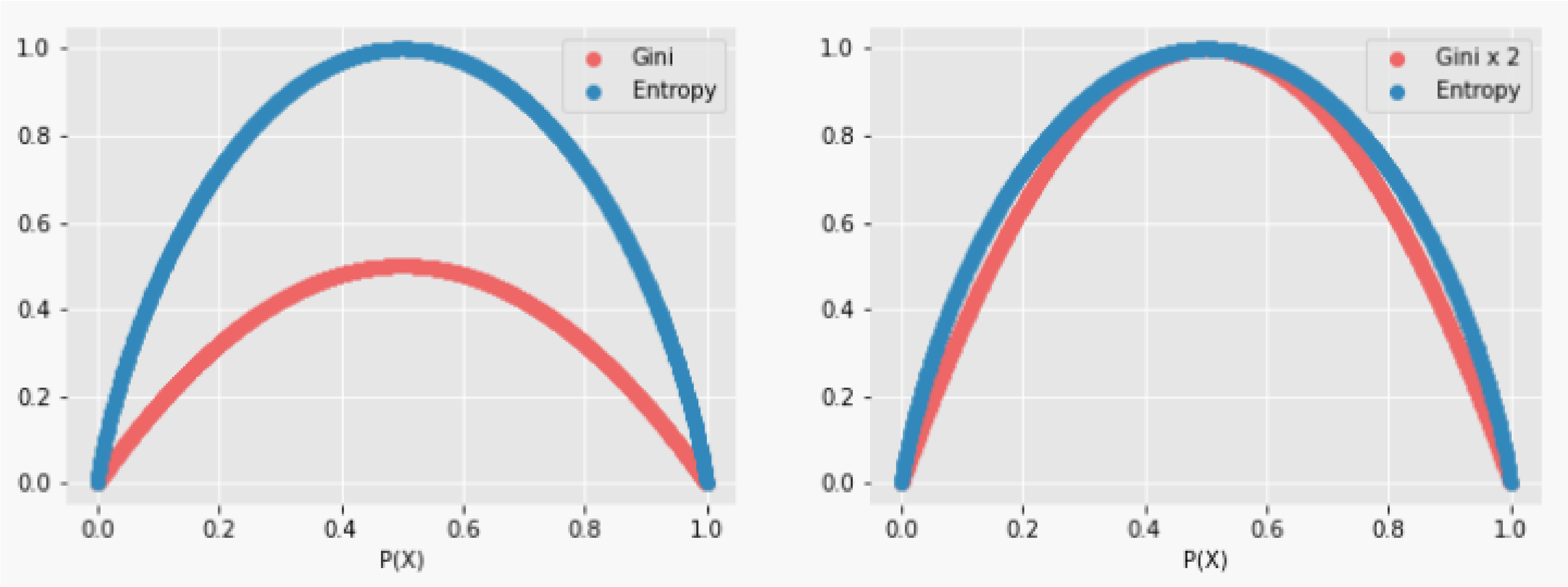
- Impurity measurements of a node t:
  - Gini impurity

$$I(t) = 1 - \sum_{k=1}^{K} p(y_k|t)^2$$

  - Entropy impurity

$$I(t) = -\sum_{k=1}^{K} p(y_k|t) log_2 p(y_k|t)$$

Decision Tree: Node Splitting



Left: original Gini compared with Entropy; Right: Gini*2 compared with Entropy

Decision Tree: Pseudo Code

- Begin with the root node t of the original dataset $X_t = X$
- For each feature x_i:
  - For each candidate value a_{in} (n=1,2,3,…,):
    - Divide the data into left node X_{tY} and right node X_{tN} by answering:

$$x_i < a_{in}$$

    - Compute the Impurity decrease

$$\Delta I = I(t) - \frac{N_{tY}}{N_t} I(tY) - \frac{N_{tN}}{N_t} I(tN)$$

- Find the feature x_i and value a_{in} that lead to the most impurity decrease
- Continue splitting……

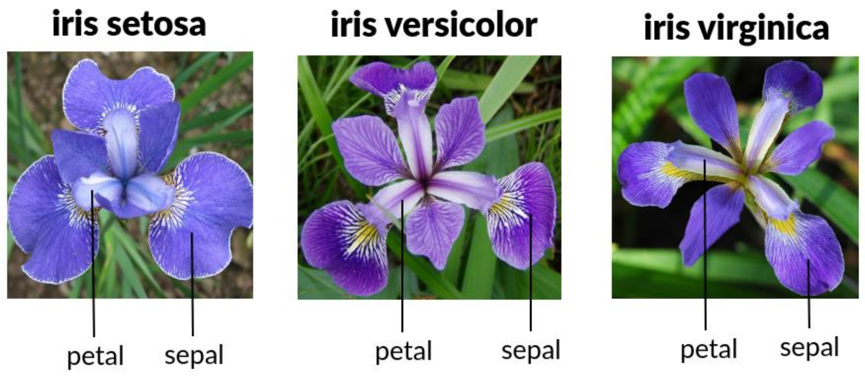
Decision Tree: Stopping Criterion

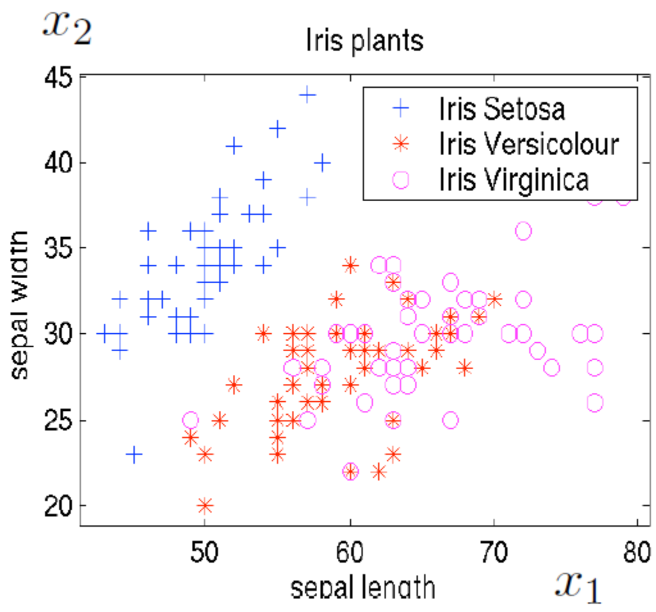
- Splitting stops until one of the following happens:
  - Using all possible splitting ways, we have:

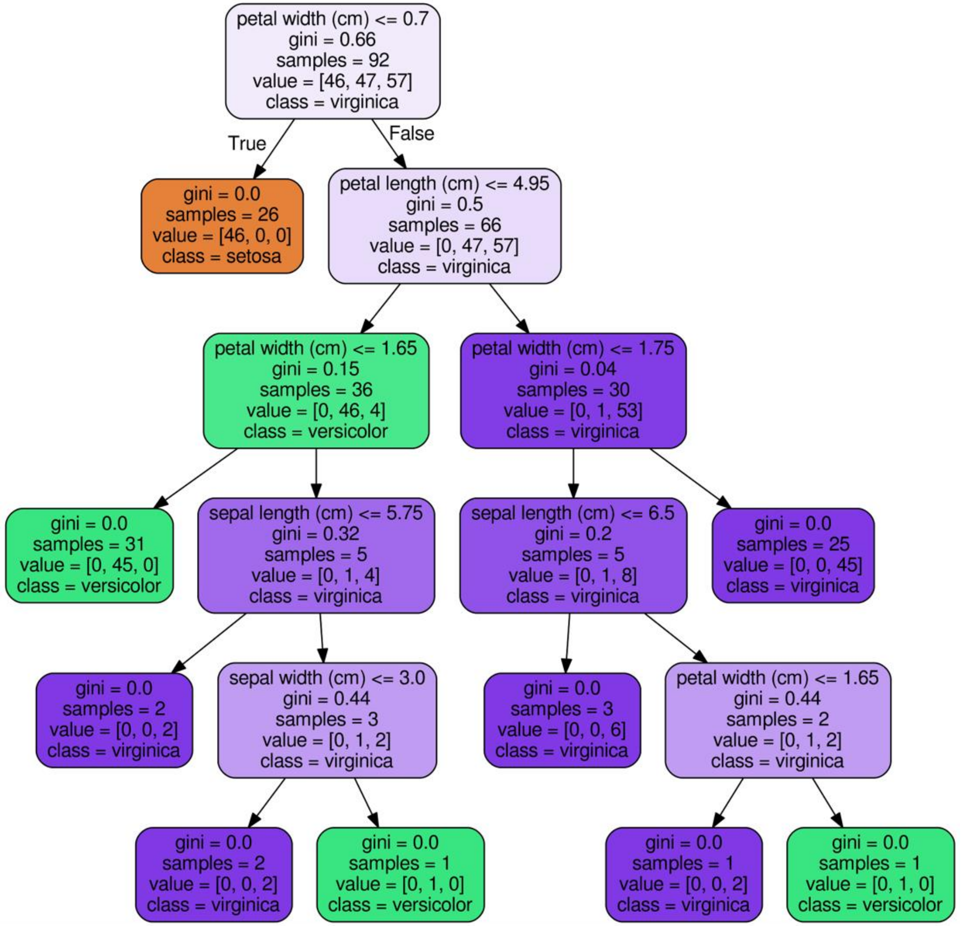
$$\Delta I < Threshold$$

  - X_t is too small
  - X_t is pure now (i.e., contains only one class)

Decision Tree: a Demo







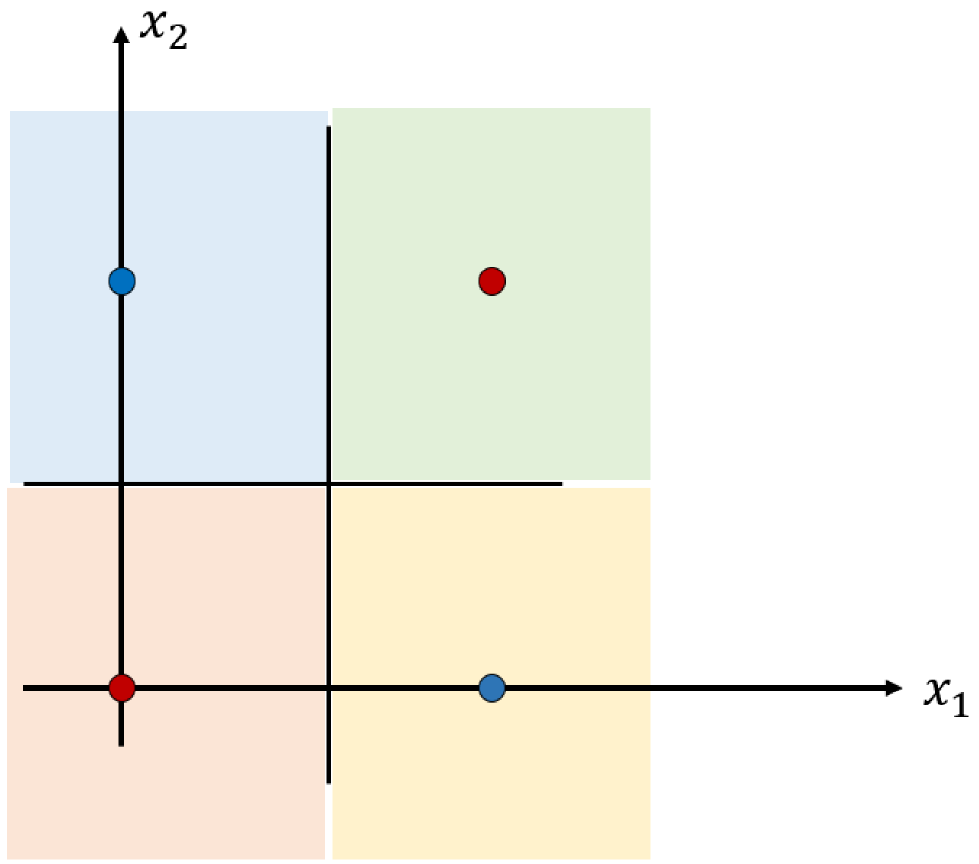
Source code: https://gist.github.com/WillKoehrsen/ff77f5f308362819805a3defd9495ffd

Decision Tree: Overview

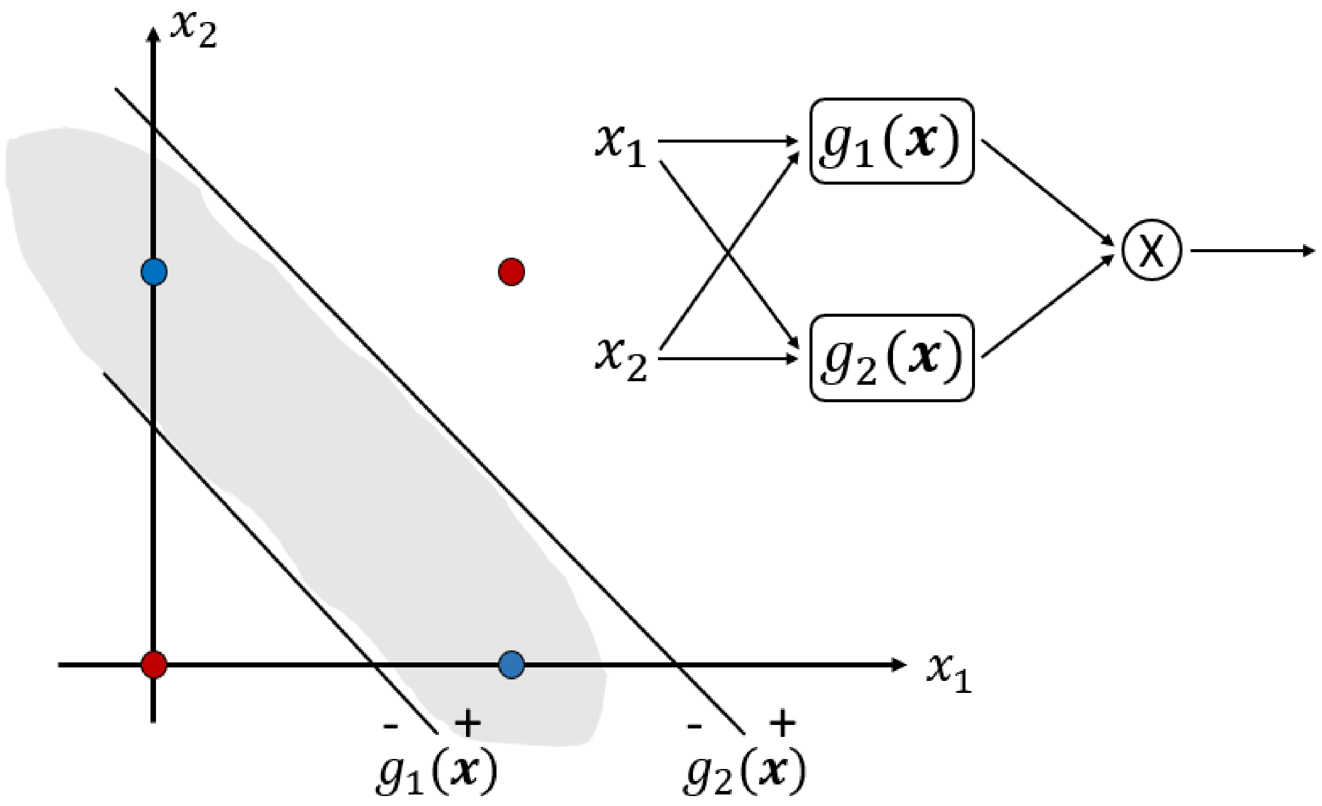
- Size of the tree must be large enough but too large. Otherwise, it overfits to particular data details

- Trees have high variance. A small change in data often leads to a very different tree

Decision Tree vs. MLP



Tree



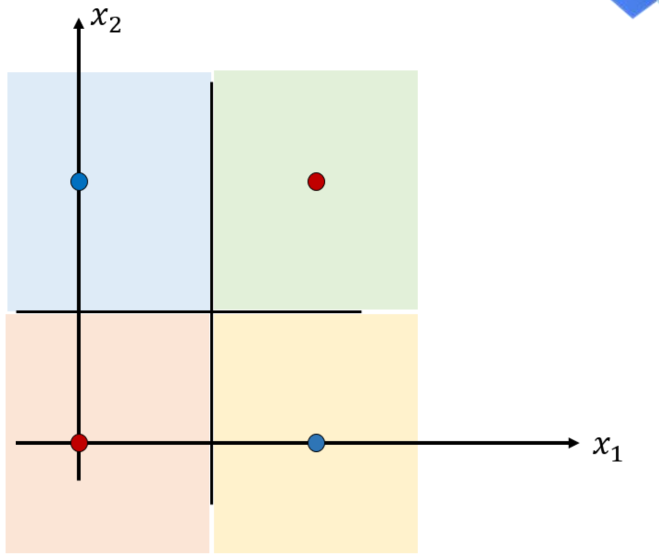
MLP

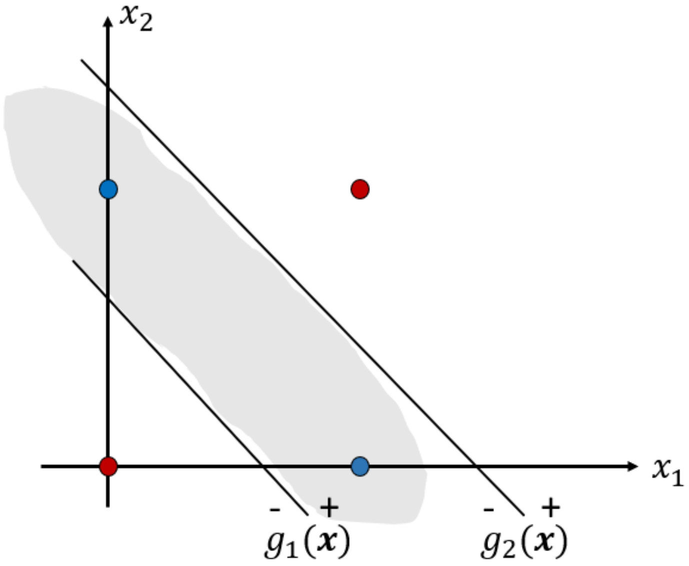
Decision Tree vs. MLP

- Simpler decision boundaries

- Single feature value involves each stage

- Higher interpretability





Interpretability

Example 7.1. Interpretability

Imagine you are constructing a classification tree for deciding whether someone is suspected of having diabetes or not, based on a number of the patient's characteristics (BMI, cholesterol, etc.). A classification tree built using simple one-feature rules such as 'is the patient's BMI higher than Z' is way more trustworthy to practitioners than a classification tree built using rules such as 'is 0.4 patient's BMI plus 0.145 patient's cholesterol level higher than Z'. This is exactly where the whole talk about interpretability of AI tools is about, if you heard about it.

Adapted from WI4635, TUDelft

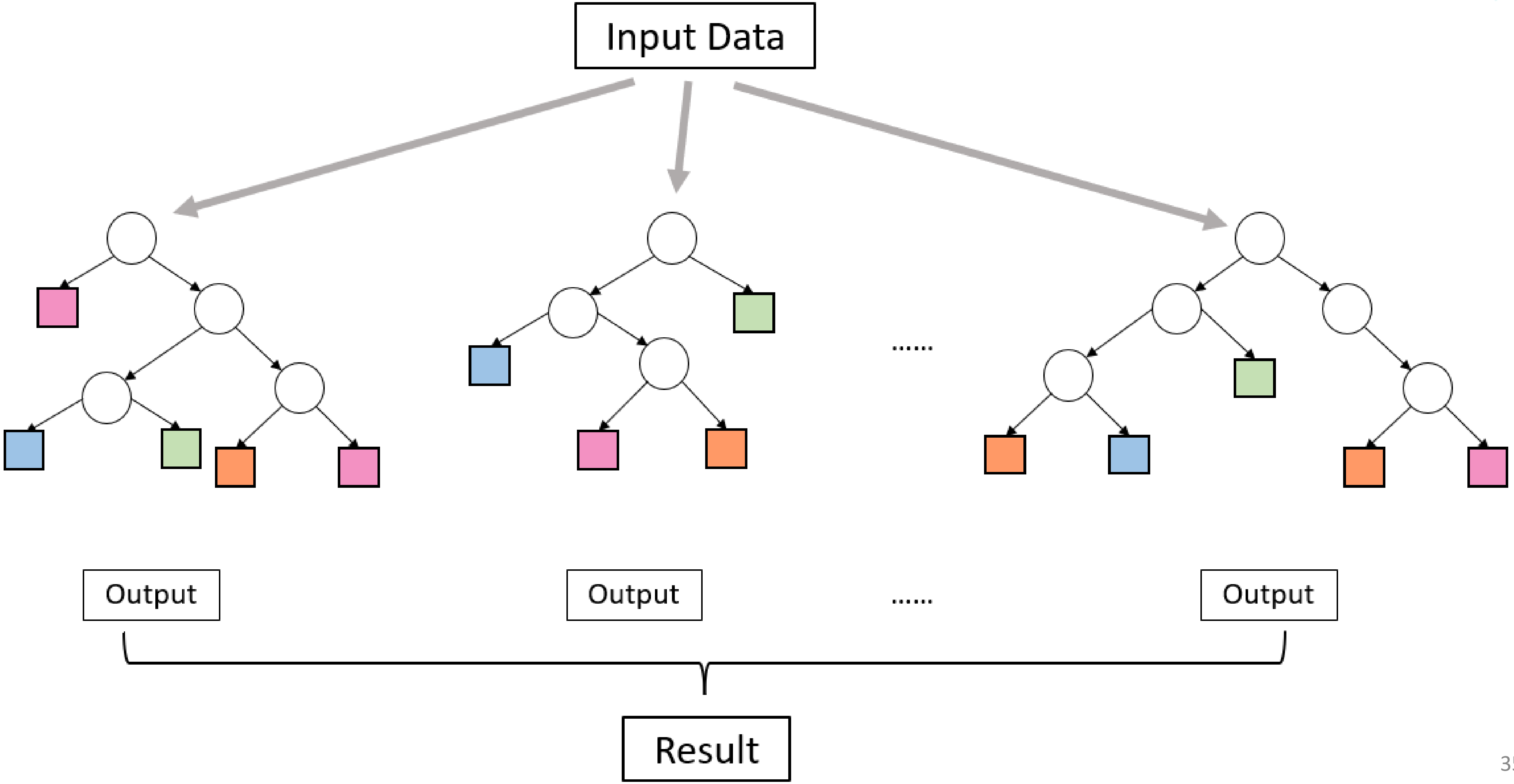
Today's Agenda

- Previous Lecture: Linear Classifiers

- Decision Trees
  - Random Forest
  - Application: SUM

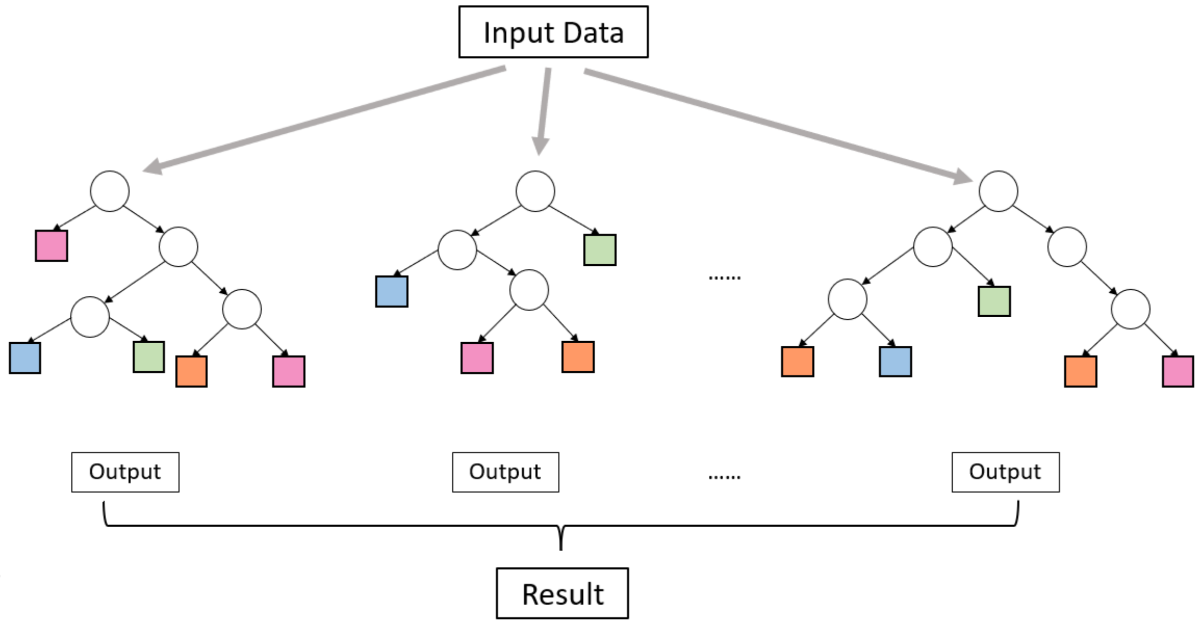
- Data and Features
  - Feature Selection
  - Classifier Evaluation

Random Forest



Random Forest: Bagging

- Sample the original dataset with replacement
  - E.g., for the original set [1,2,3,4,5], we can sample [1,3,4,4,5]
- Create multiple tree classifiers, each with bagging. Summarize the results using majority vote.



Random Forest: Random Features

- Each tree can pick only from a random subset of features
- This is to further ensure the independence

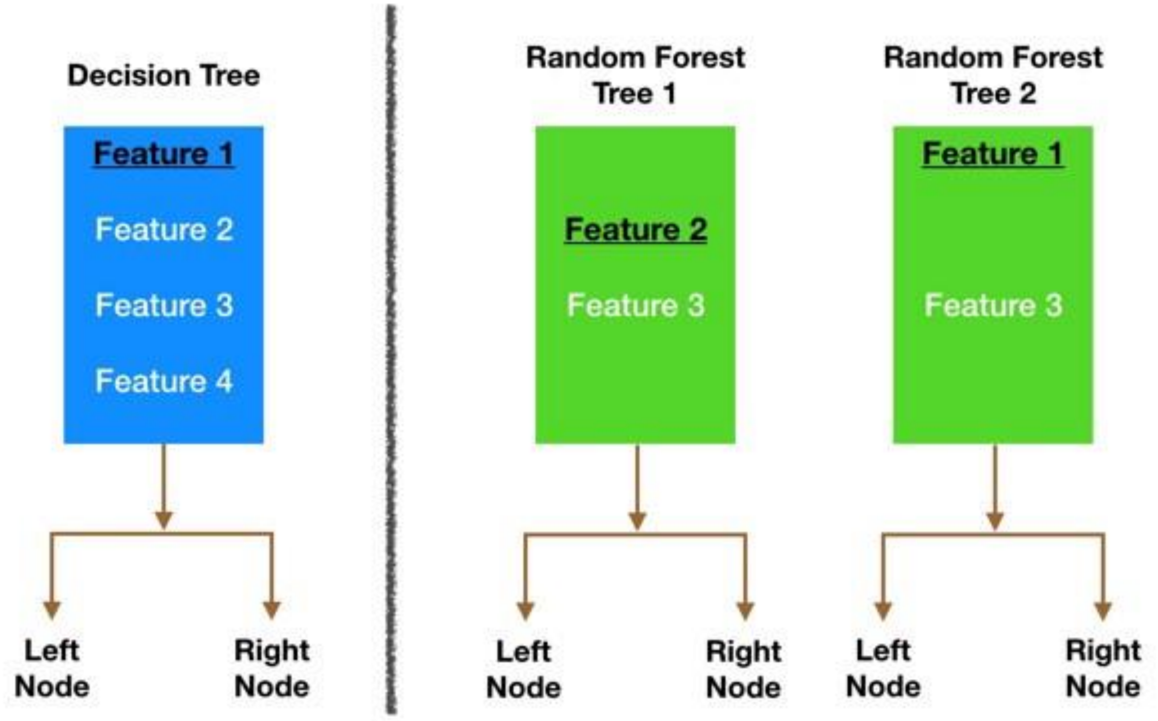


Image source: towards data science

Random Forest

- Combining relatively uncorrelated classifiers together generally outperforms a single classifier
- Combining models also helps to reduce the variance
- With sufficient trees, RF can achieve comparable performance as neural networks
- However, interpretability is gone

Today's Agenda

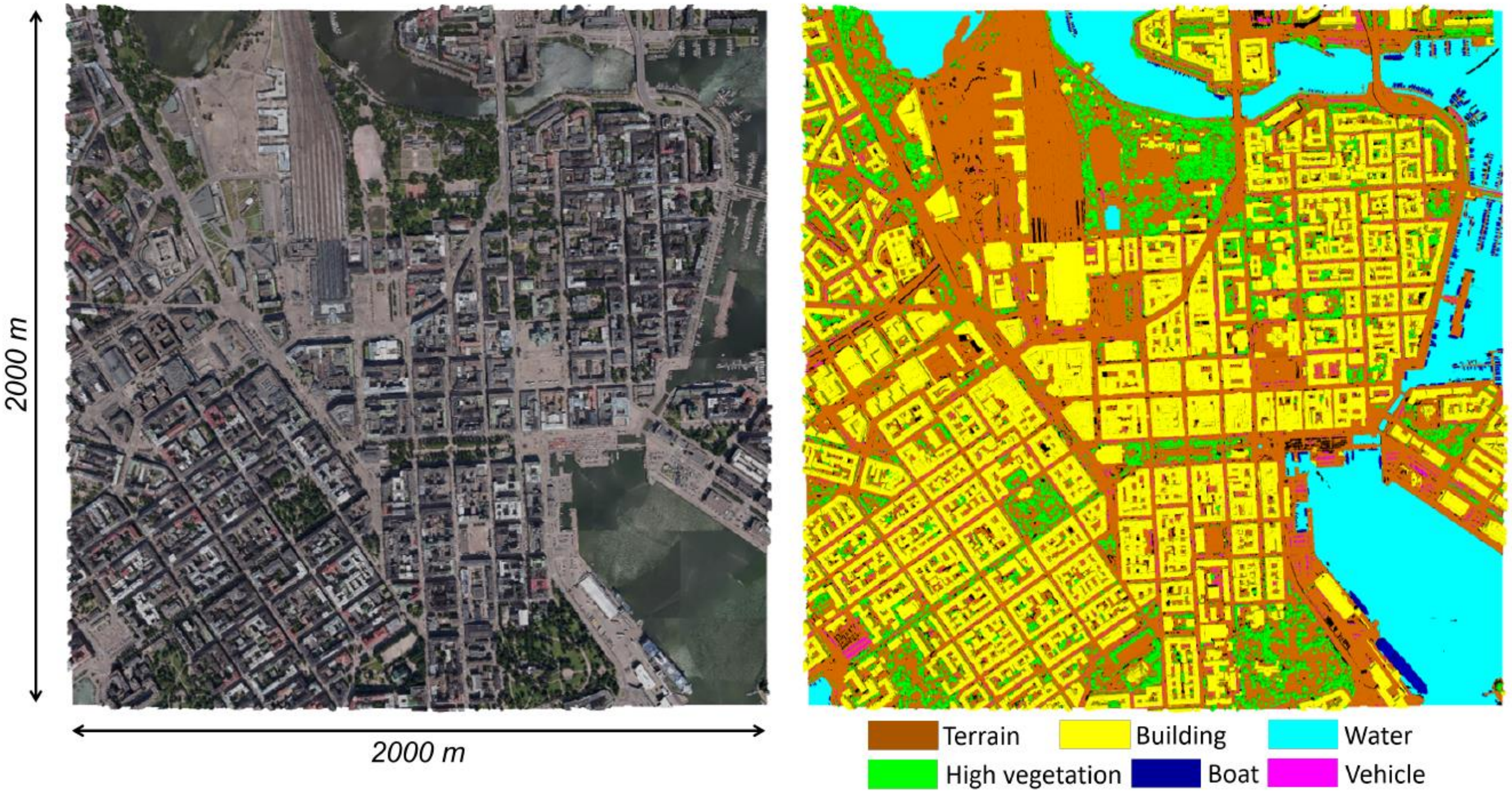
- Previous Lecture: Support Vector Machine

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Semantic Urban Meshes

SUM: A benchmark dataset of Semantic Urban Meshes, ISPRS 2021



SUM: Features

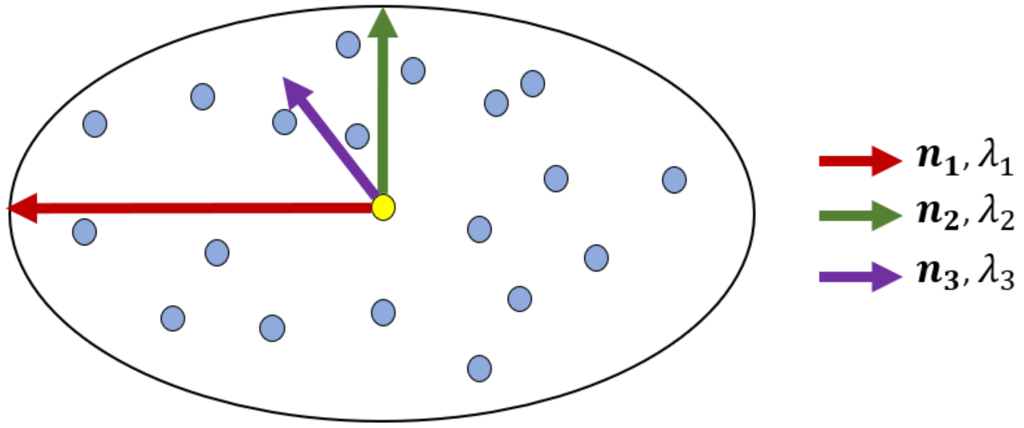
- Eigen features

  Linearity: $\frac{\lambda_1 - \lambda_2}{\lambda_1}$

  Sphericity: $\frac{\lambda_3}{\lambda_1}$

  Curvature change: $\frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$

  Verticality: $1 - |\boldsymbol{n_3} \cdot \boldsymbol{n_z}|$



- Elevation features

  Relative elevation: $z - z_{min}$

- Other features

  Colors, local color variance

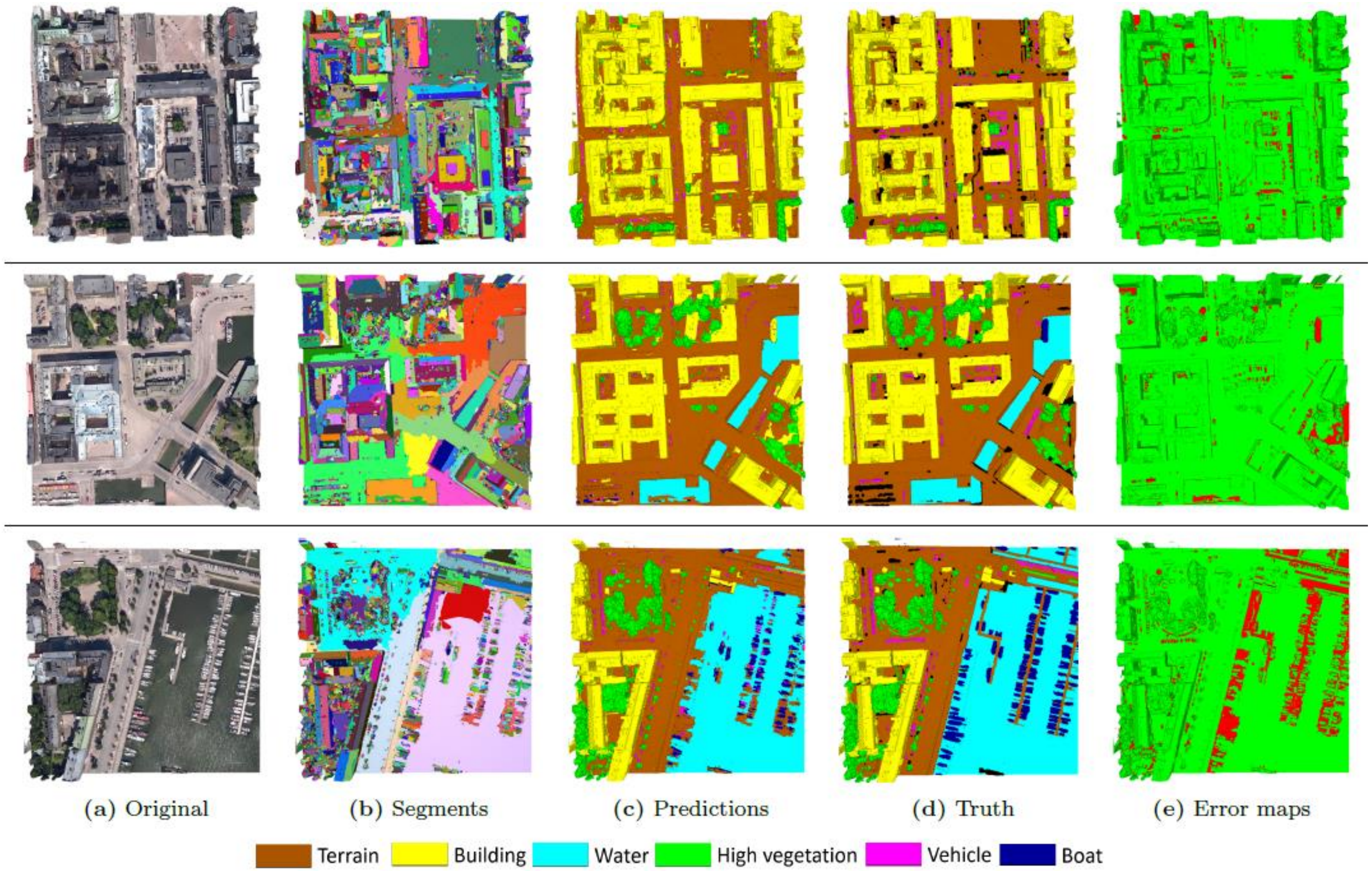
  Mesh area, triangle densities

  ……

SUM: Performance

	Terrain	High Vegeta-tion	Building	Water	Vehicle	Boat	mIoU	OA	mAcc
PointNet [14]	56.3	14.9	66.7	83.8	0.0	0.0	36.9 ± 2.3	71.4 ± 2.1	46.1 ± 2.6
RandLaNet [53]	38.9	59.6	81.5	27.7	22.0	2.1	38.6 ± 4.6	74.9 ± 3.2	53.3 ± 5.1
SPG [15]	56.4	61.8	87.4	36.5	34.4	6.2	47.1 ± 2.4	79.0 ± 2.8	64.8 ± 1.2
PointNet++ [52]	68.0	73.1	84.2	69.9	0.5	1.6	49.5 ± 2.1	85.5 ± 0.9	57.8 ± 1.8
RF-MRF [43]	77.4	87.5	91.3	83.7	23.8	1.7	60.9 ± 0.0	91.2 ± 0.0	65.9 ± 0.0
KPConv [16]	**86.5**	88.4	**92.7**	77.7	**54.3**	**13.3**	**68.8** ± 5.7	**93.3** ± 1.5	**73.7** ± 5.4
Baseline	83.3	**90.5**	92.5	**86.0**	37.3	7.4	66.2 ± 0.0	93.0 ± 0.0	70.6 ± 0.0

SUM: Visual Results



(a) Original (b) Segments (c) Predictions (d) Truth (e) Error maps

Today's Agenda

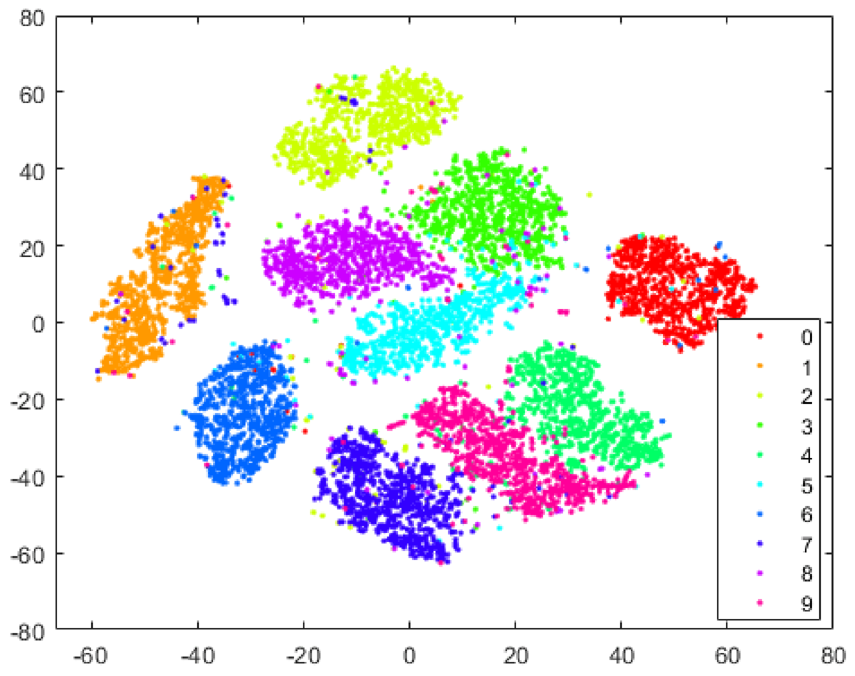
- Previous Lecture: Support Vector Machine

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Data and Features

- Will more features lead to better performance?



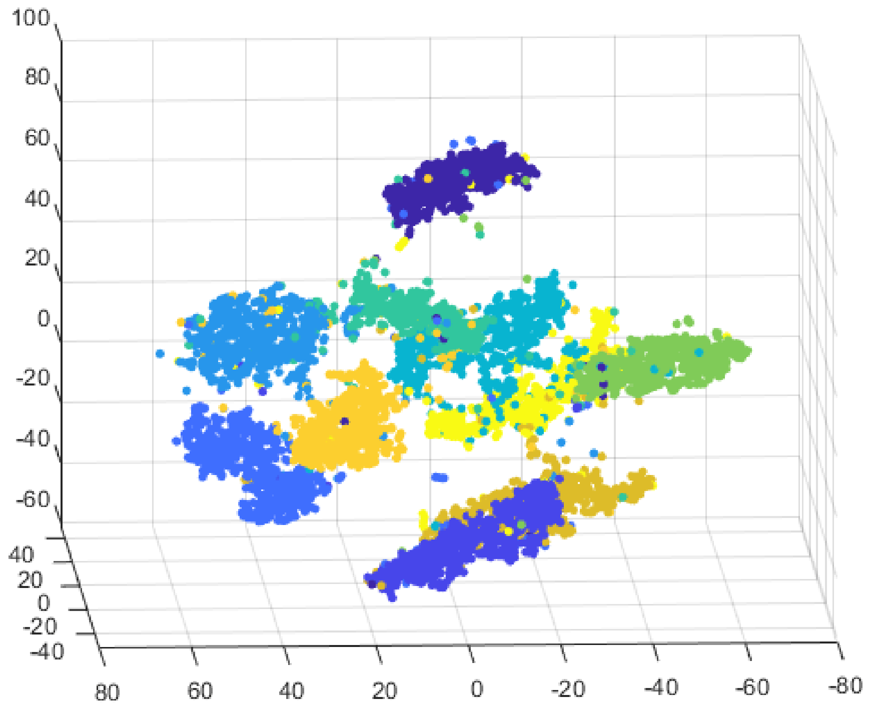


Image Source: https://www.mathworks.com/help/stats/visualize-high-dimensional-data-using-t-sne.html

Data and Features

- Curse of dimensionality
  - Too few samples in too high dimensional space

- Computation complexity

- Feature correlations
  - 1+1 is not always larger than 2

Today’s Agenda

- Previous Lecture: Support Vector Machine

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Feature Selection

- How to measure if a feature subset is good or not?
  - The best is to measure actual classification performance. However, it can be expensive

- How could we select the most important features?
  - Limit the dimensionality (i.e., number of features)
  - Retain the class discriminatory information

Feature Selection

- Scatter matrices for feature selection criterion:
  - ***Within-scatter matrix***:

$$S_w = \sum_{k=1}^{K} \frac{N_k}{N} \Sigma_k$$

  - ***Between-scatter matrix***:

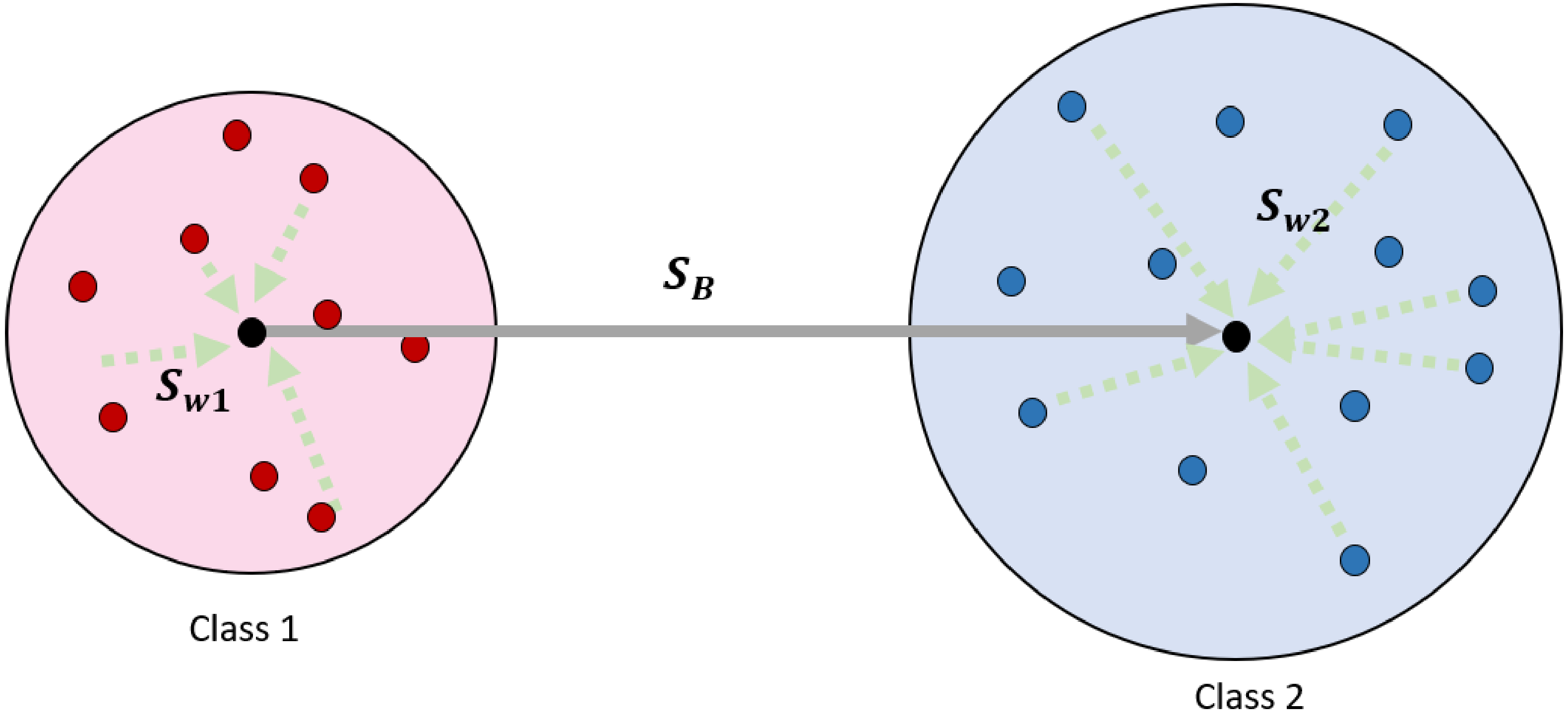
$$S_B = \sum_{k=1}^{K} \frac{N_k}{N} (\boldsymbol{\mu_k} - \boldsymbol{\mu})(\boldsymbol{\mu_k} - \boldsymbol{\mu})^T$$

K: total number of classes

$\boldsymbol{\mu}$: mean of all samples

$\boldsymbol{\mu_k}$, Σ_k: mean and covariance matrix of per-class samples

Feature Selection



- There're several ways of combining them, e.g.,

$$J = \frac{tr(S_B)}{tr(S_w)}$$

Feature Selection

- We want to select d out from p features, and choose the subset with optimal criterion value

- How many possible subsets in total?

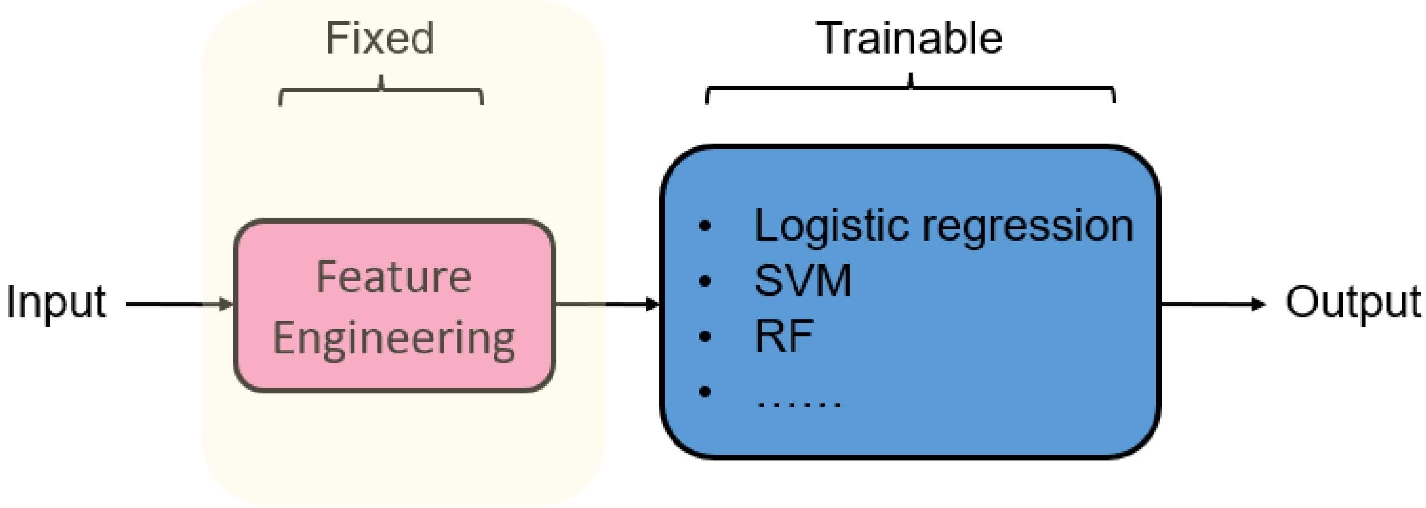
Feature Selection

- Sub-optimal searching methods
  - (1) Choose the best individual d features
  - (2) Forward search:
    - Starting with the empty set, each time add one feature that optimizes the entire chosen feature set
  - (3) Backward search:
    - Starting with the whole set, each time drop one feature that optimizes the rest of the feature set

Feature Selection

- Besides feature selection, you can also extract new features by dimension reduction methods (e.g., PCA)

- Feature engineering is the focus of most classical ML methods



Today's Agenda

- Previous Lecture: Support Vector Machine

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Classifier Evaluation

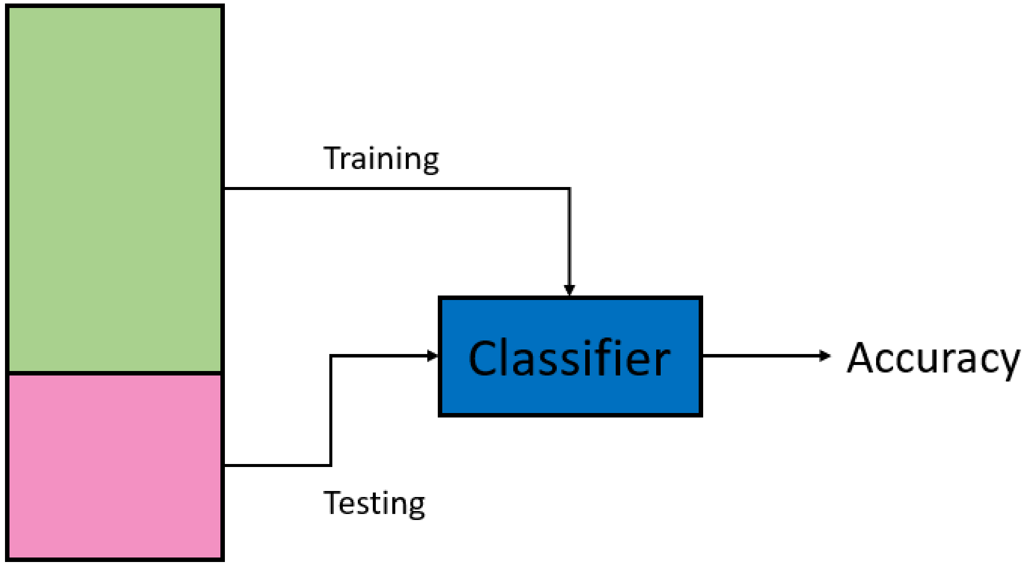
- Overall accuracy
  - Out of 500 objects, how many are correctly classified?
- Mean per-class accuracy
  - How is the accuracy of each class? Average them.
- Confusion matrix

Classifier Evaluation

- Is it good to measure the performance of the classifier in the training dataset? Why?

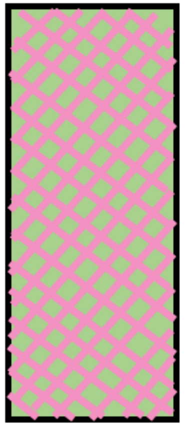
Classifier Evaluation

- Classification accuracy over training set can be biased
- We’re interested in true accuracy of the classifier

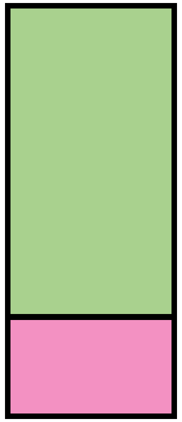


Classifier Evaluation

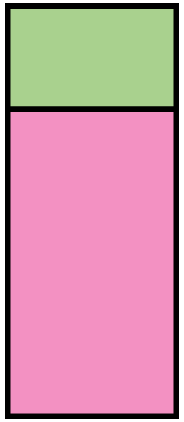
- Train-test split



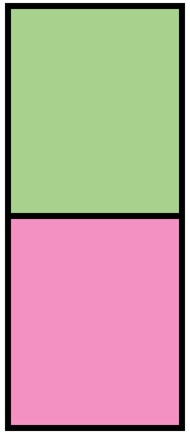
Training and testing on the same set will give a good classifier, but will yield a biased estimate of the model



A small independent test set yields an unbiased, but unreliable accuracy estimate for a well-trained classifier



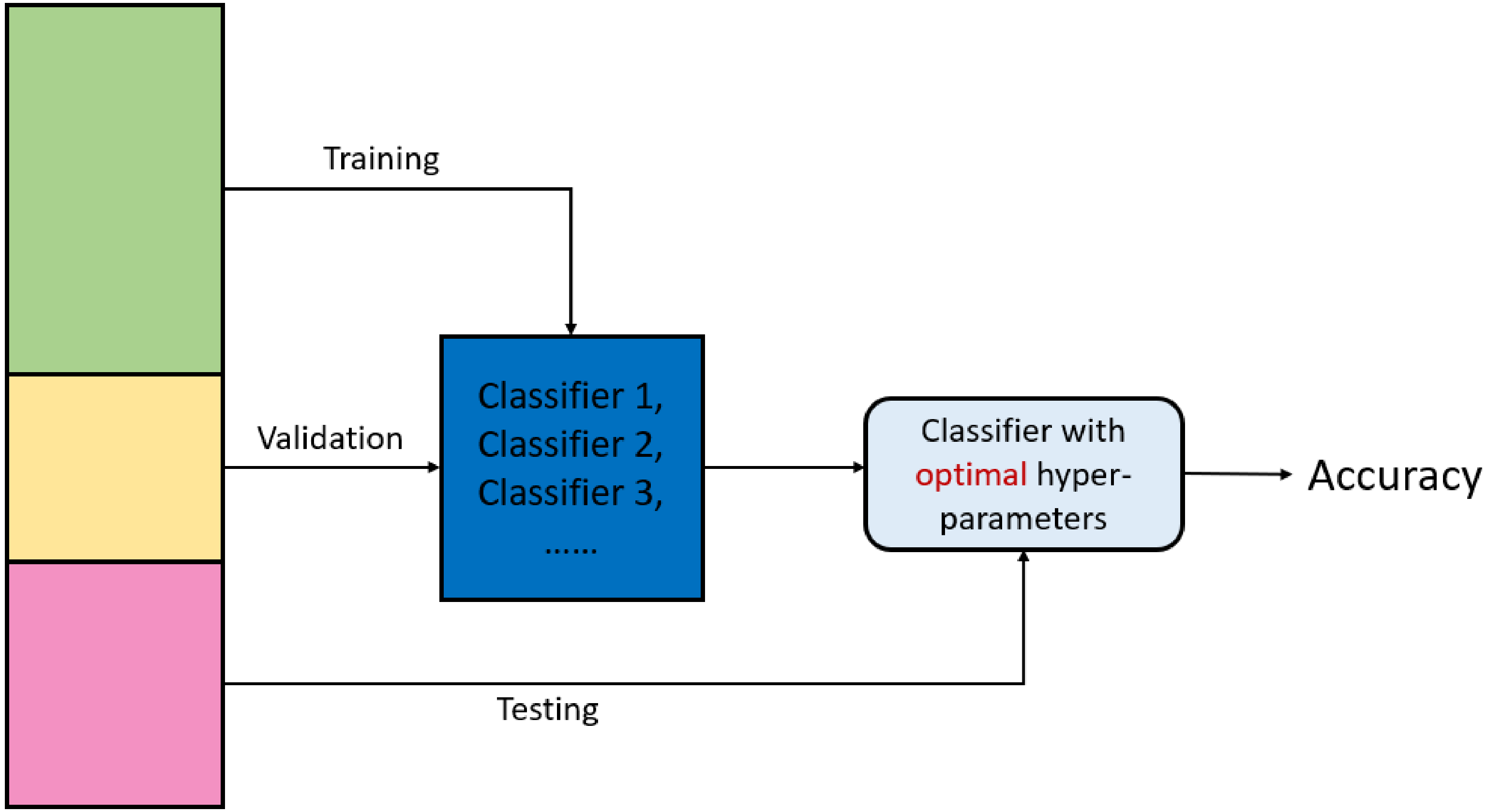
A large, independent test set yields an unbiased and reliable accuracy estimate for a badly trained classifier



7:3, 6:4, 5:5 ratios are commonly used in practice

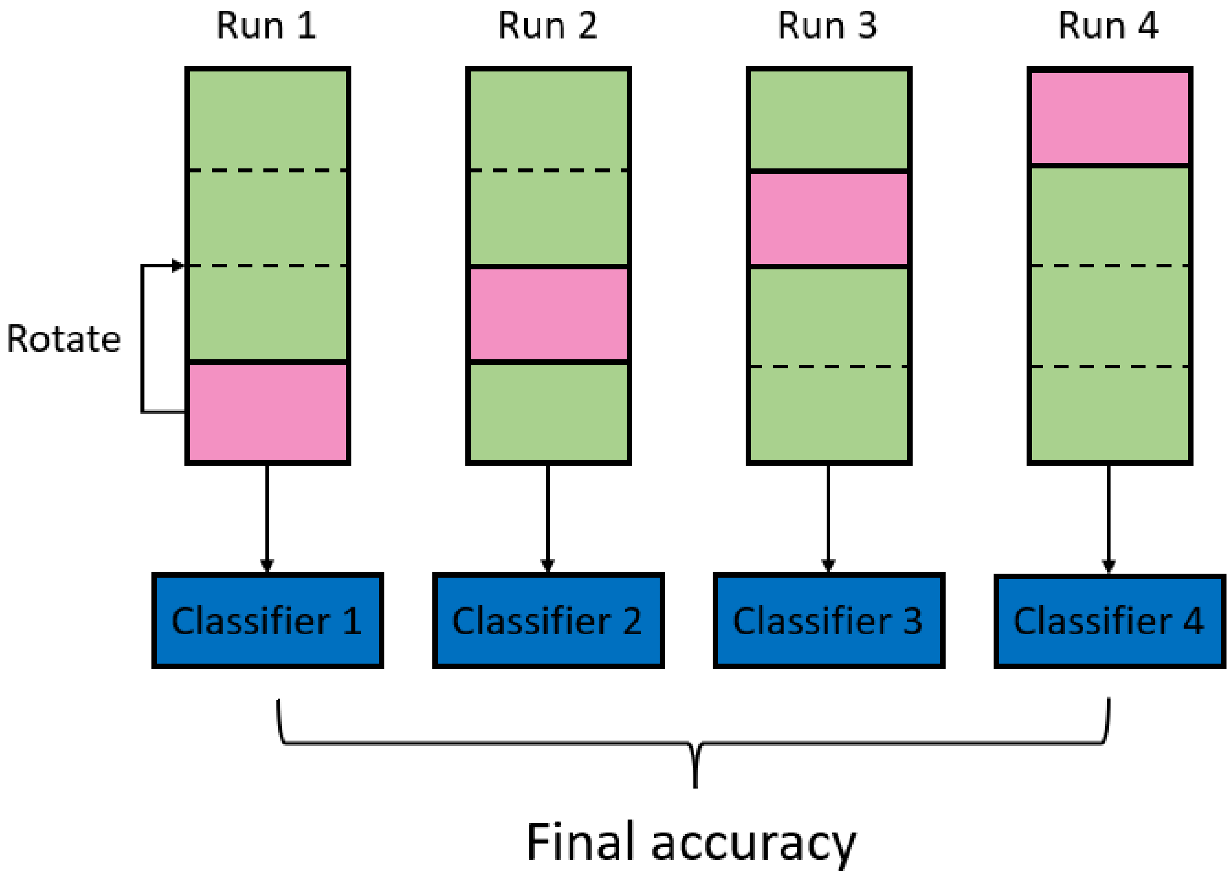
Classifier Evaluation

- Sometimes a validation set is introduced



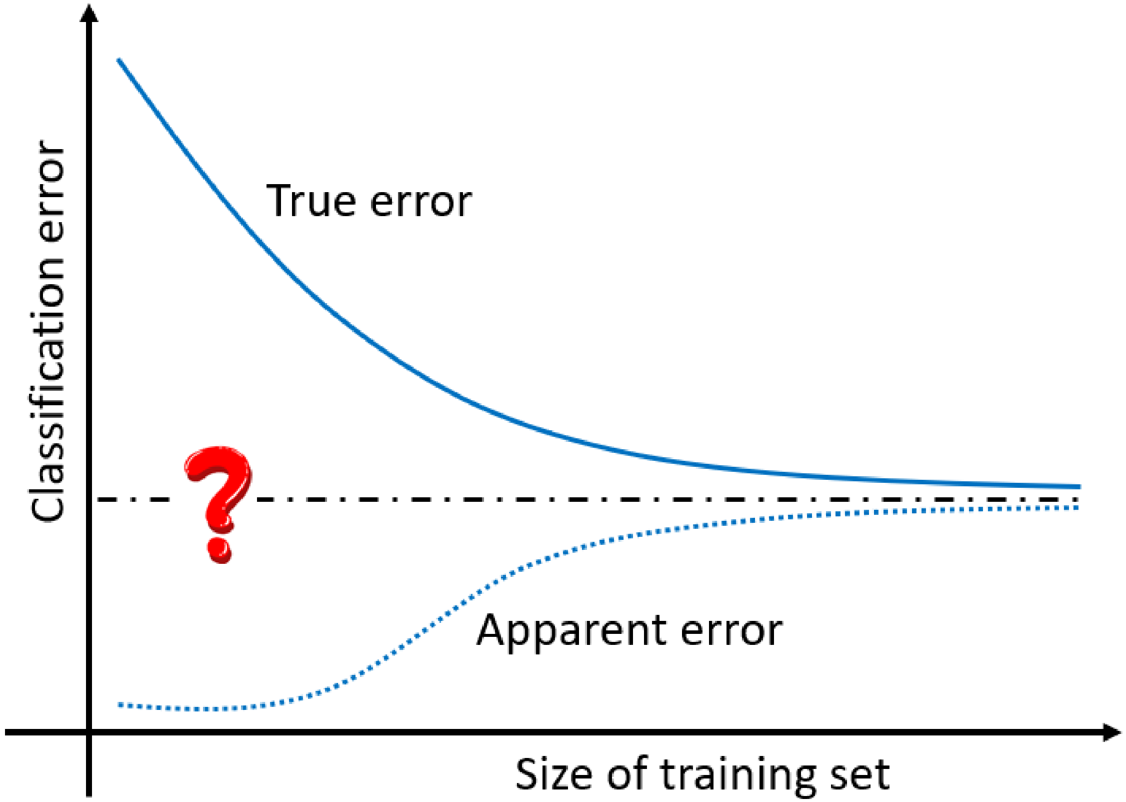
Classifier Evaluation

- Cross Validation: making full use of data



Learning Curve

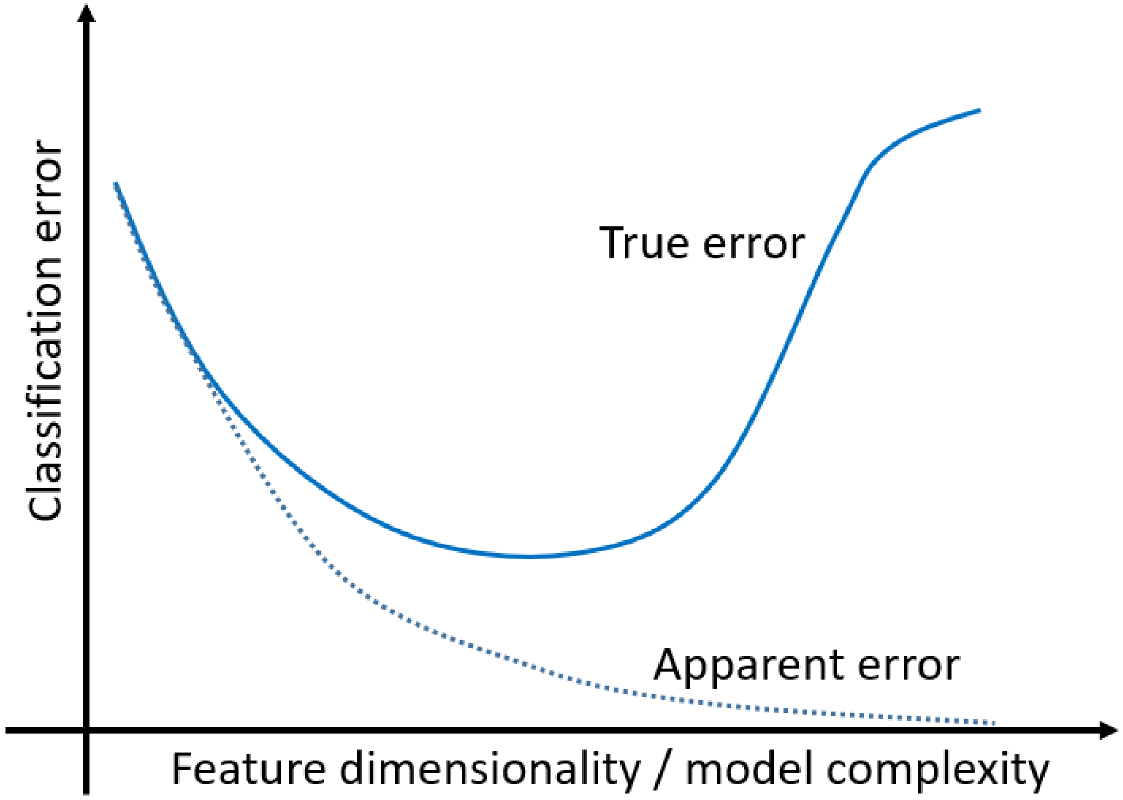
- Analysing errors w.r.t the size of the training set



Learning curve (trained using one classifier)

Feature Curve

- Analysing errors w.r.t the model complexity



Feature curve (trained using a fixed dataset)

Questions?