Supervised learning: classification

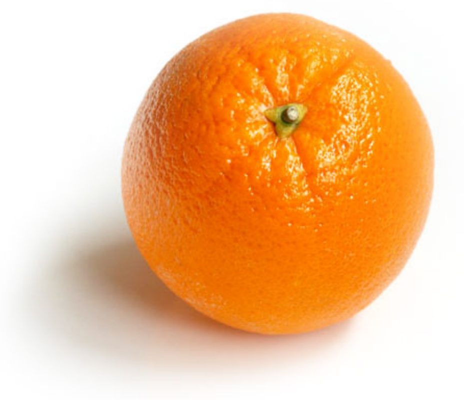
Fons van der Sommen

Eindhoven university of technology

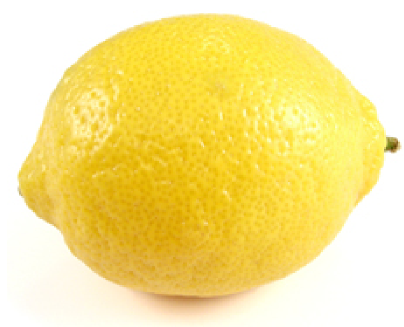
(Part of 5XSA0: Introduction to Medical Imaging)

Supervised learning: example (1/5)

* Separate lemons from oranges



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

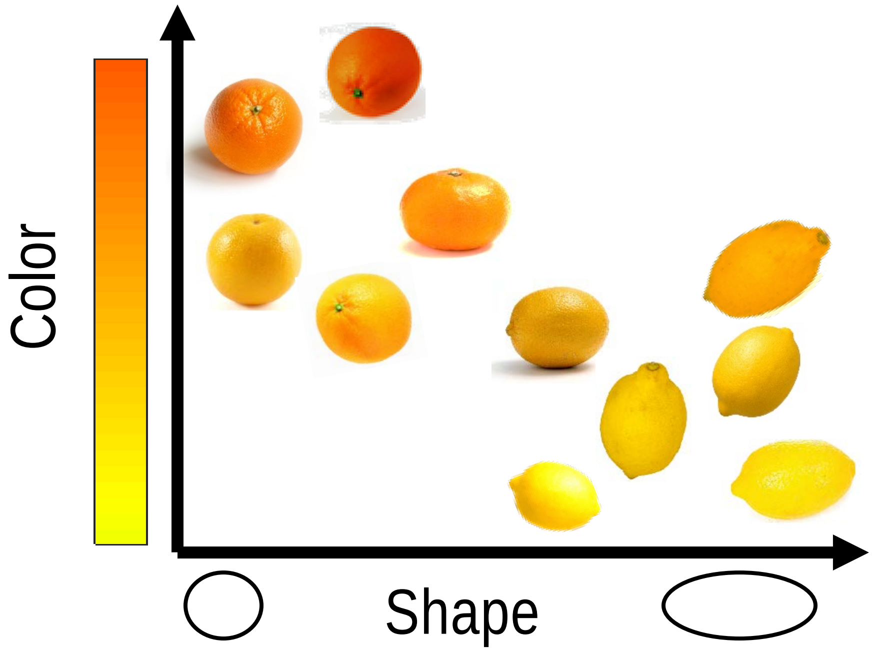


Color: yellow
Shape: elipsoid
Ø: ± 8 cm
Weigth: ±0.1 kg

* Use “color” and “shape” as features

Supervised learning: example (2/5)

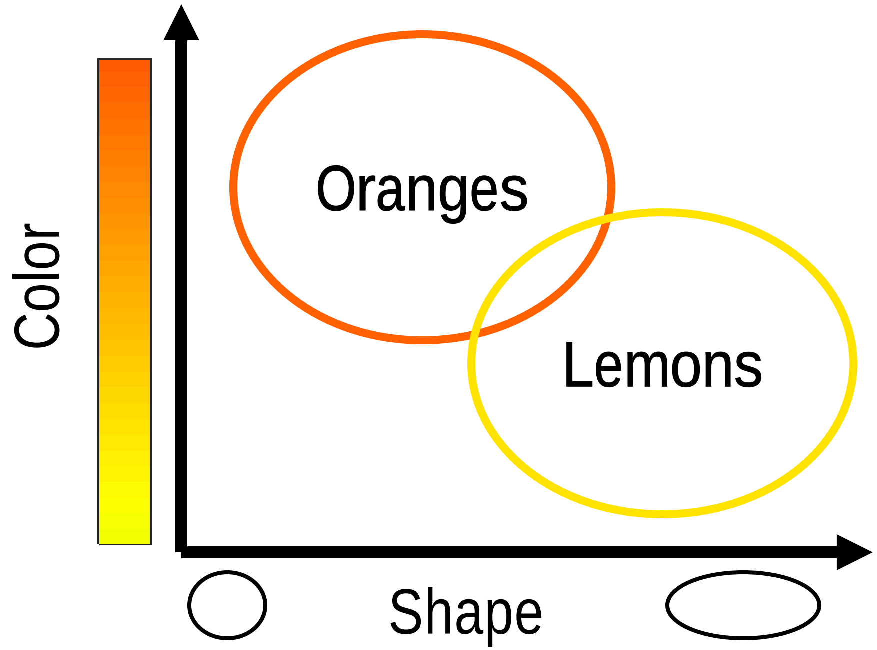
- Separate lemons from oranges



Supervised learning: example (3/5)

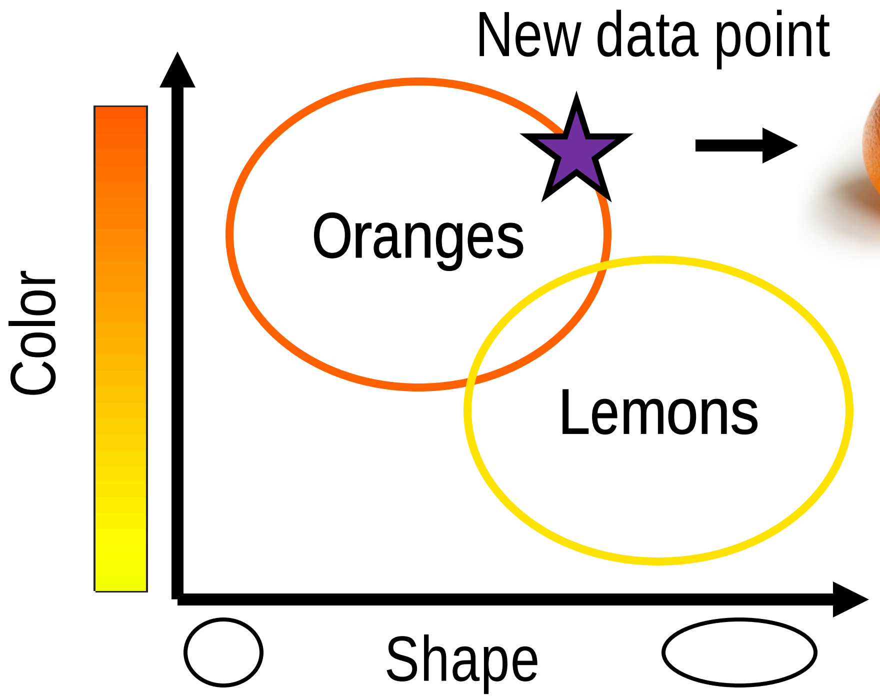
- Separate lemons from oranges

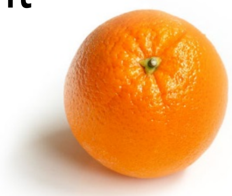
Model the given - *training* - data



Supervised learning: example (4/5)

* Separate lemons from oranges

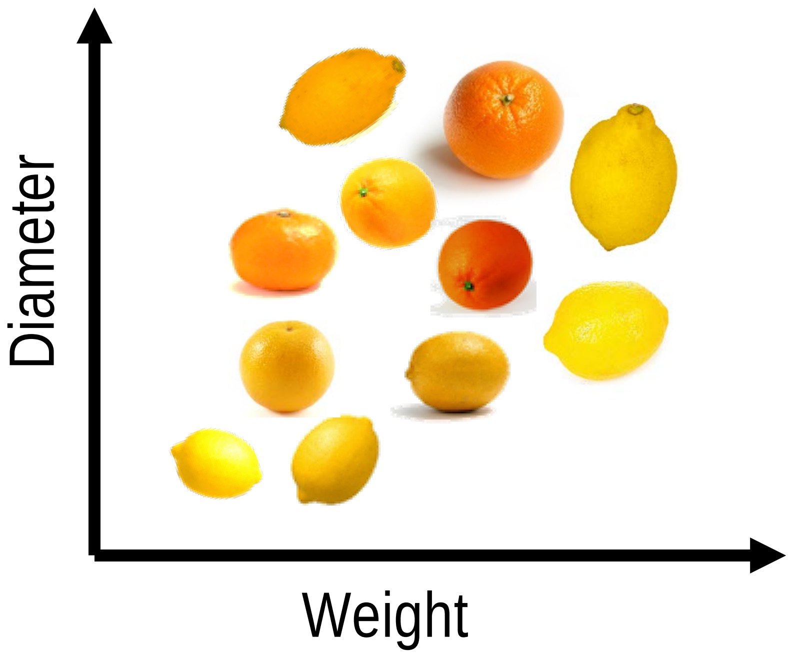


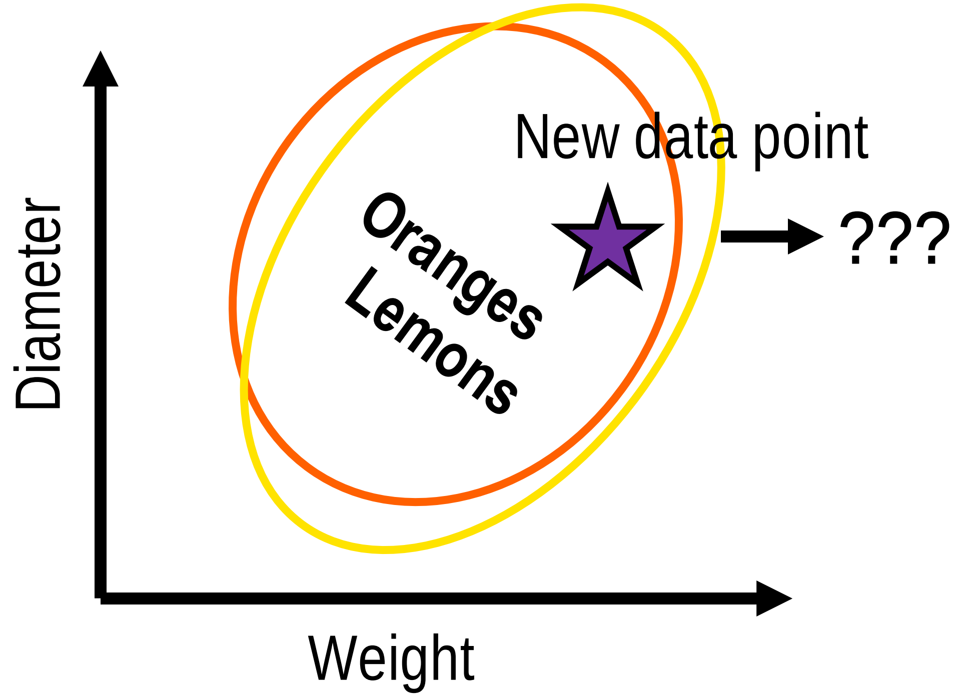


Classifier:
"It's an orange!"

Supervised learning: example (5/5)

* What if we had chosen the wrong features?





Supervised learning

Summary

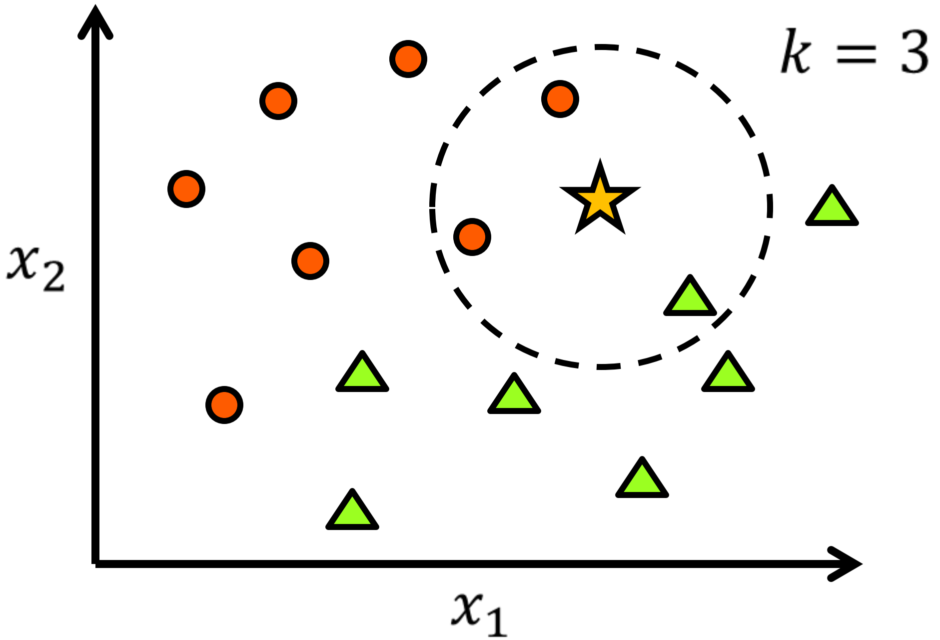
* Choose distinctive features
* Make a model based on labeled data (a.k.a. supervised learning)
* Use the *learned* model to predict the class of new, unseen data points

Models for classification

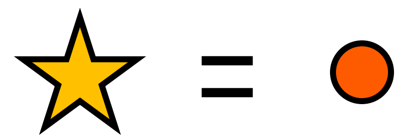
- Support Vector Machine (SVM)
- k Nearest Neighbours (k-NN)
- Random Forests
- Boosting
- Neural Networks
- …

k Nearest Neighbours (1)

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

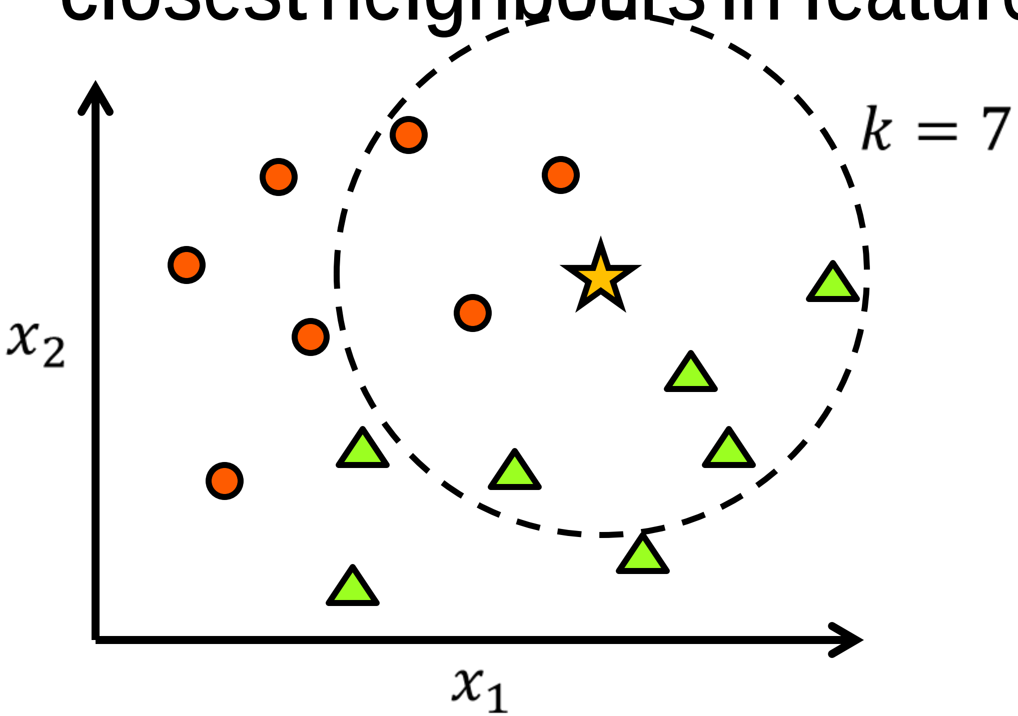


For 3 nearest neighbours:

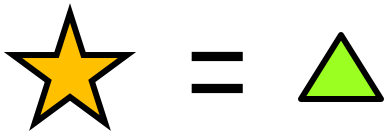


k Nearest Neighbours (2)

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



For 7 nearest neighbours:



k Nearest Neighbours (3)

* 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

k Nearest Neighbours (4)

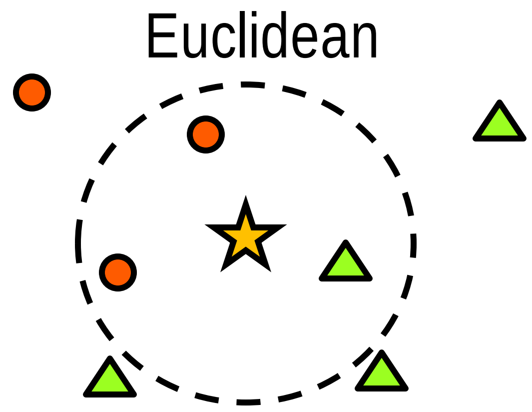
* Distance metrics
  – Euclidean distance (L^2-norm)

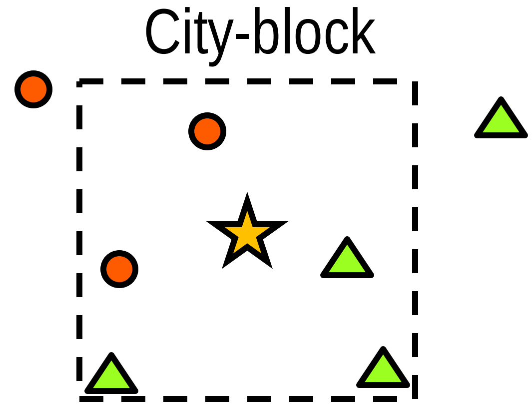
$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{D} (p_i - q_i)^2}$$

  – City block distance (L^1-norm)

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

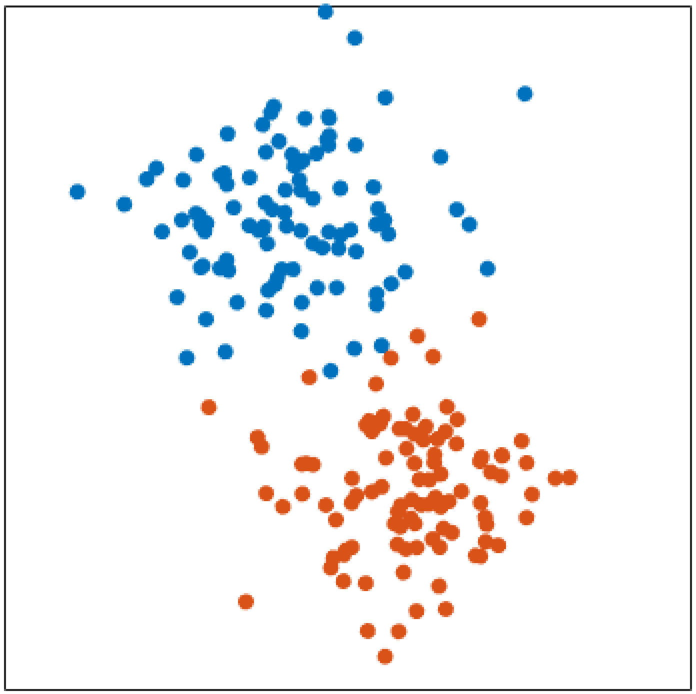
  – Many more options!



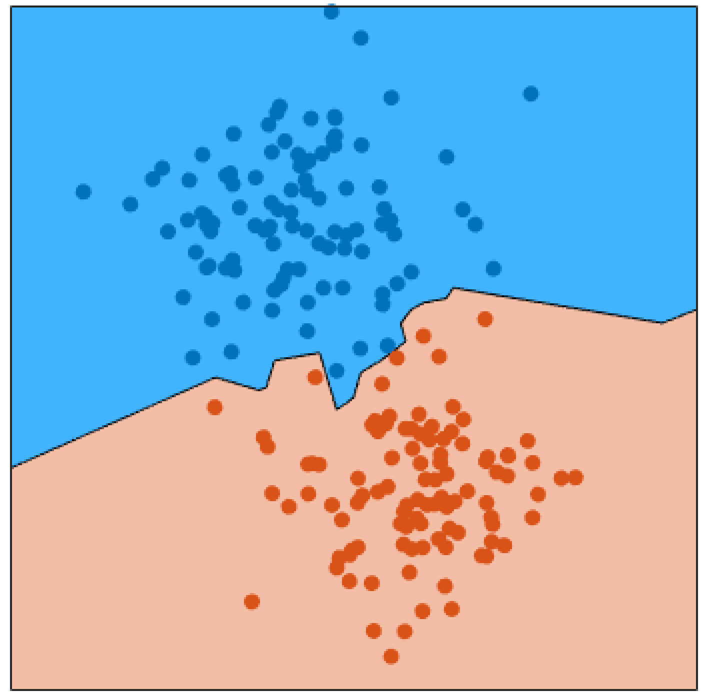


k Nearest Neighbours
neigbours & generalization (1)

2 classes in feature space

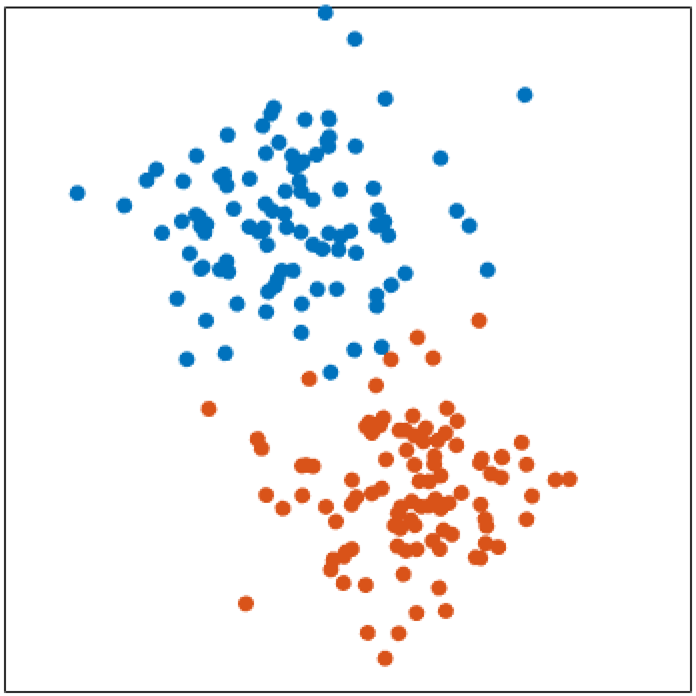


k-NN decision for k=1

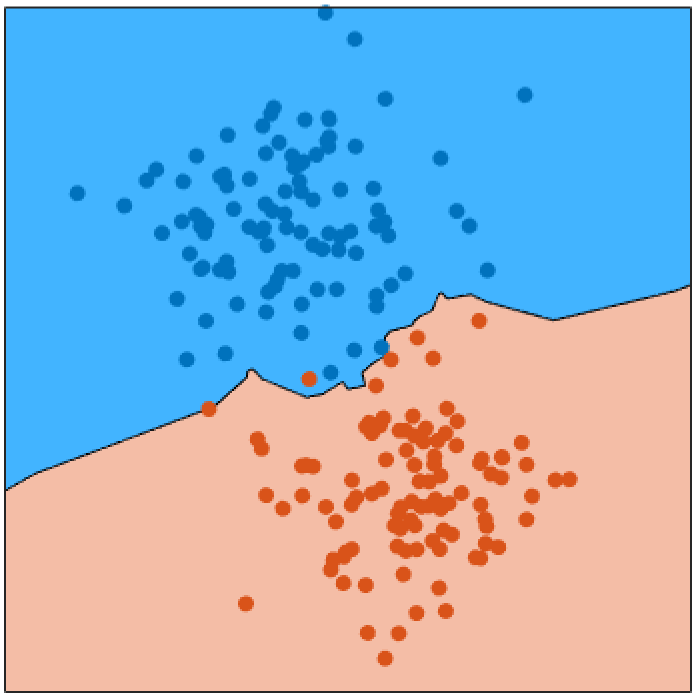


k Nearest Neighbours
neigbours & generalization (2)

2 classes in feature space

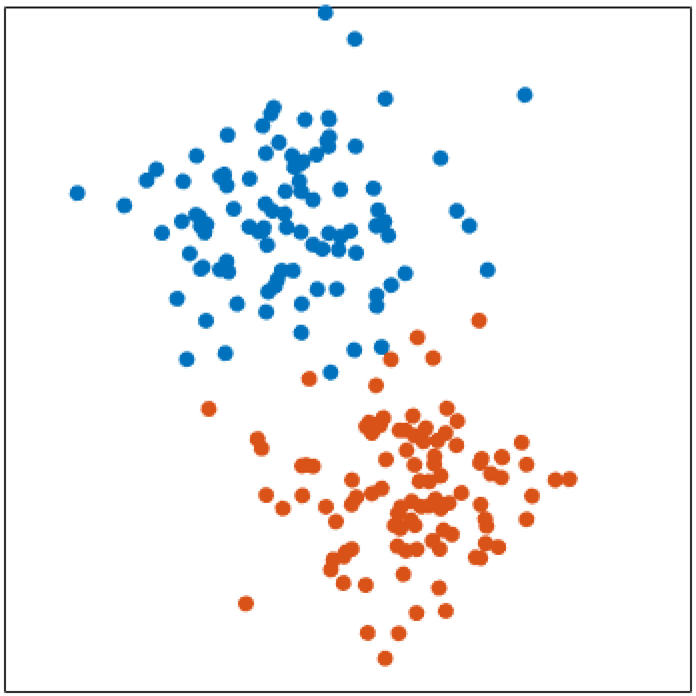


k-NN decision for k=3

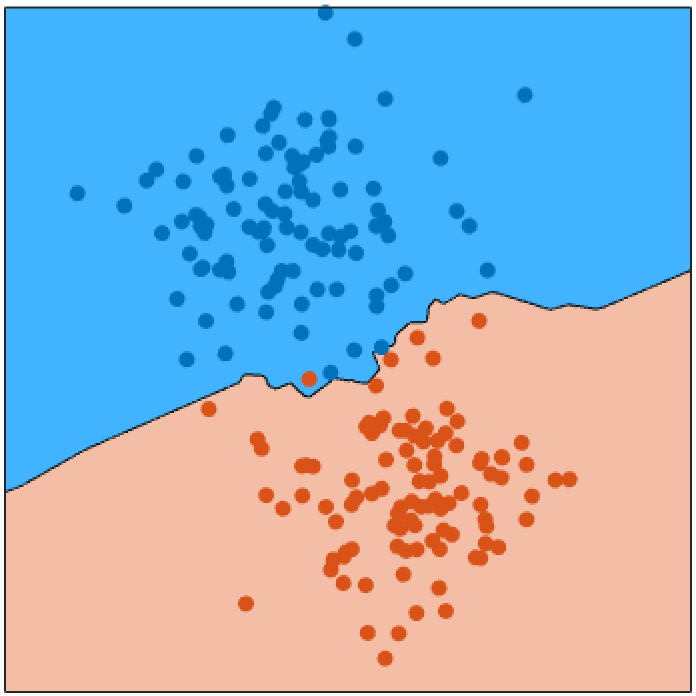


k Nearest Neighbours
neigbours & generalization (3)

2 classes in feature space

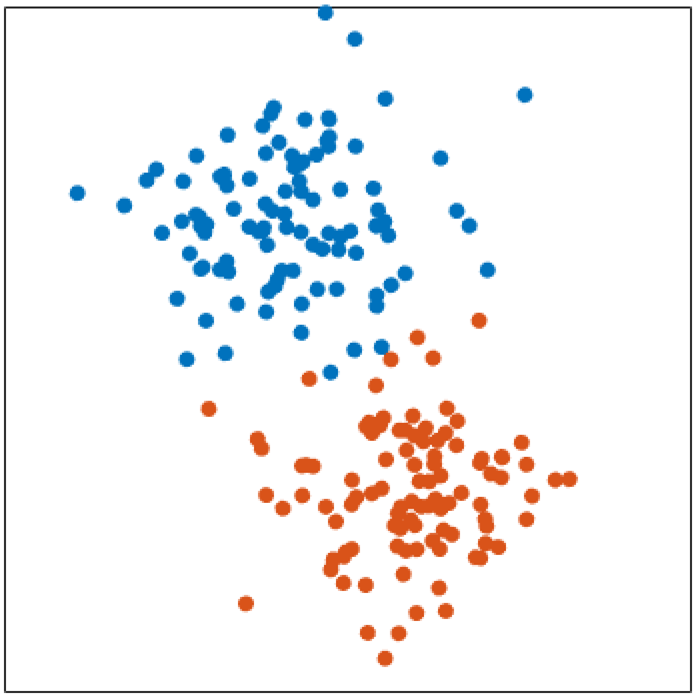


k-NN decision for k=5

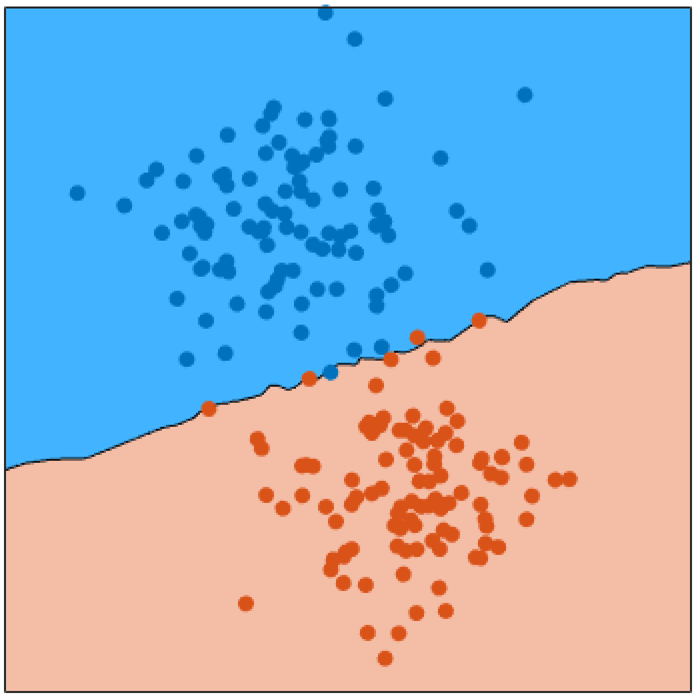


k Nearest Neighbours
neigbours & generalization (4)

2 classes in feature space

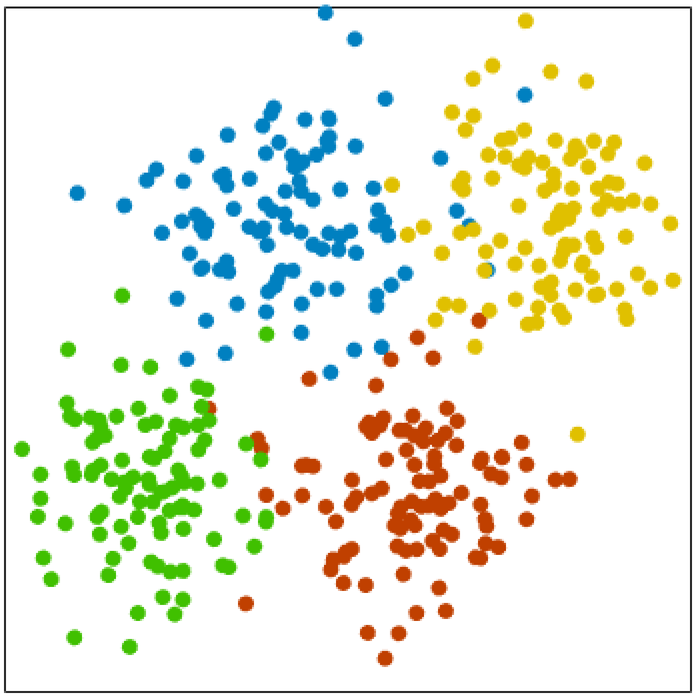


k-NN decision for k=10

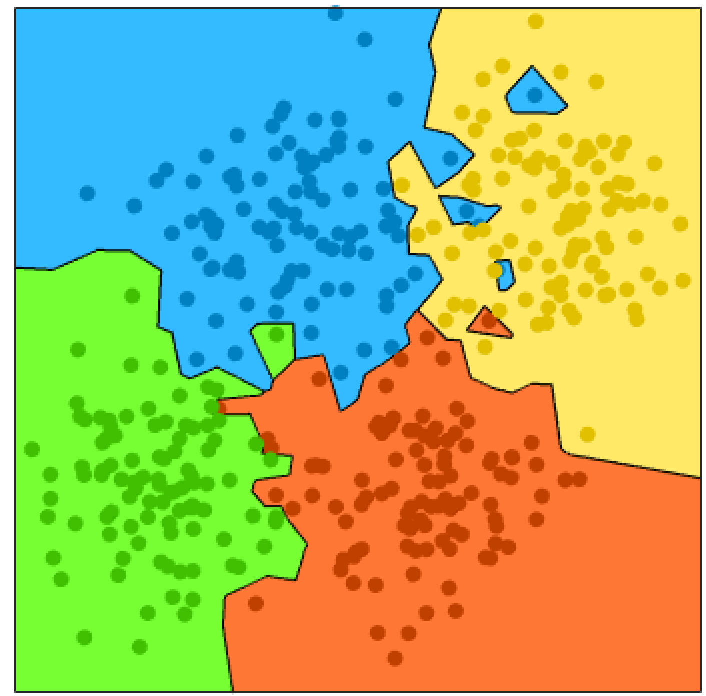


k Nearest Neighbours
neigbours & generalization (5)

4 classes in feature space

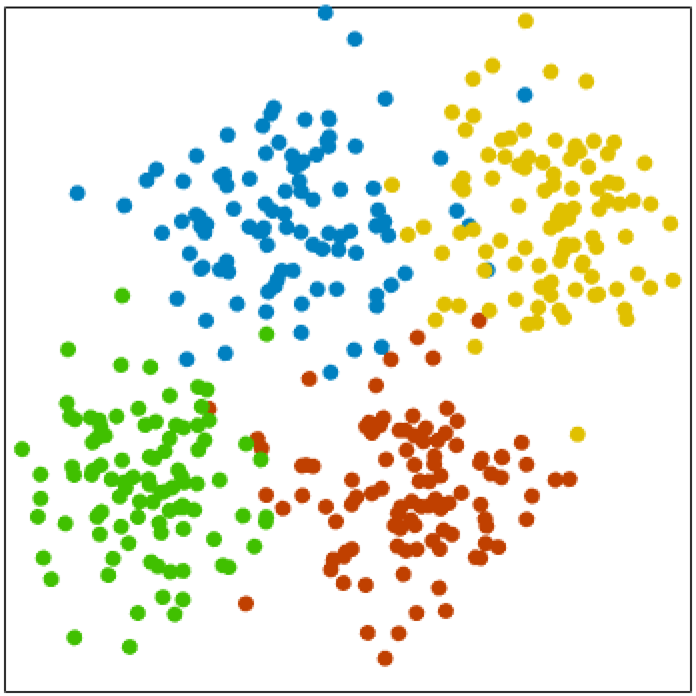


k-NN decision for k=1

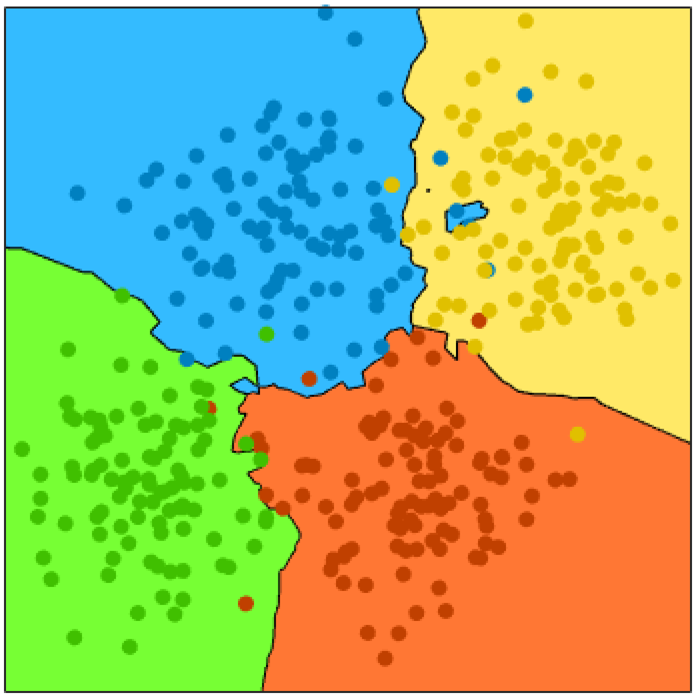


k Nearest Neighbours
neigbours & generalization (6)

4 classes in feature space

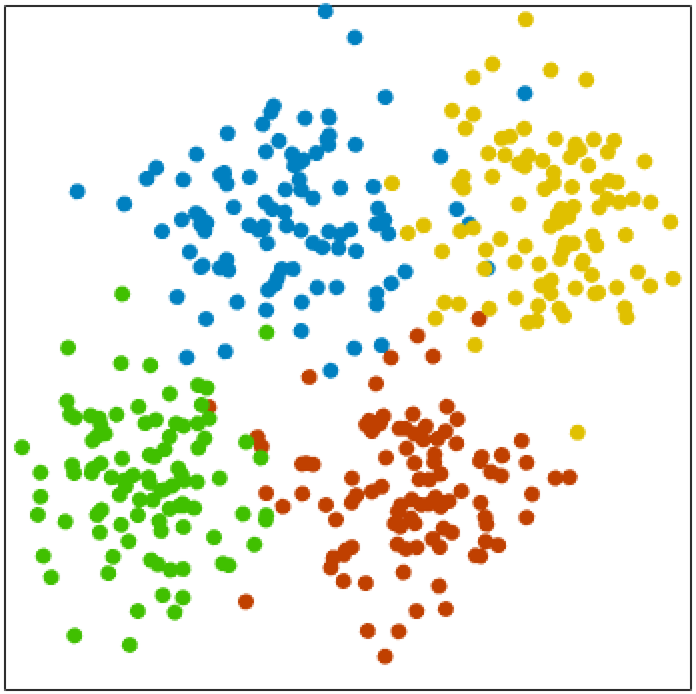


k-NN decision for k=3

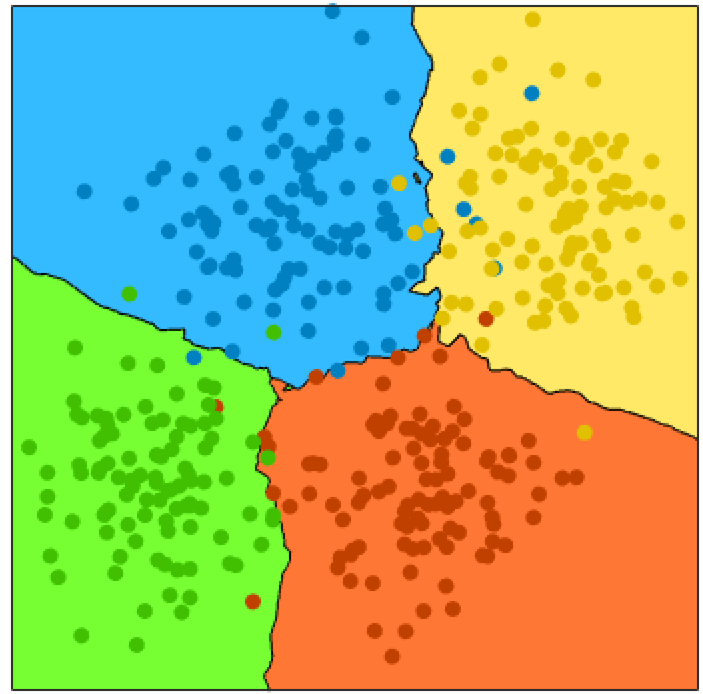


k Nearest Neighbours
neigbours & generalization (7)

4 classes in feature space

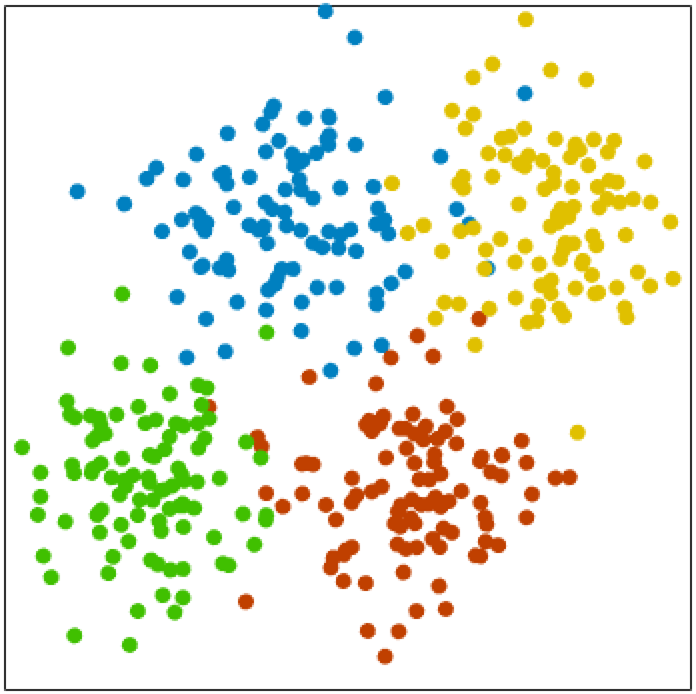


k-NN decision for k=10

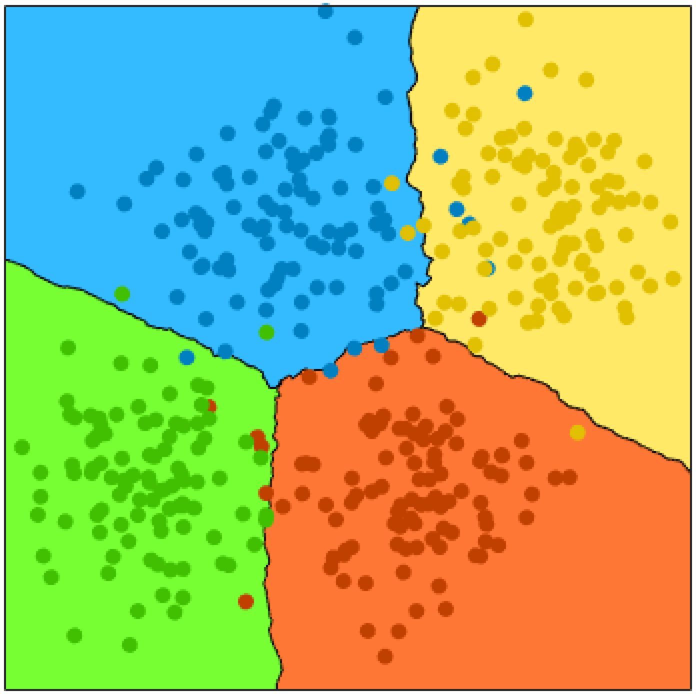


k Nearest Neighbours
neigbours & generalization (8)

4 classes in feature space



k-NN decision for k=25

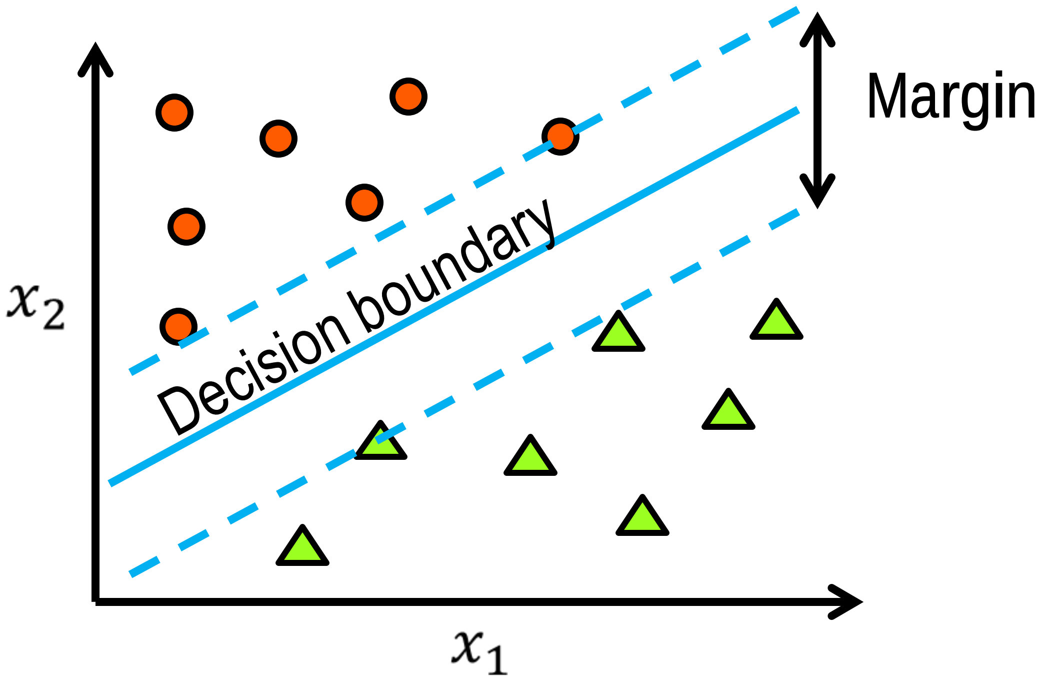


k Nearest Neighbours

* Summary
  – Instance learning: no explicit modeling
  – Memory heavy: all training samples are stored
  – Two important parameters
    1. *Number of nearest neighbours k*
    2. *Distance metric d*
  – Different parameters choices can lead to different results!
  – Higher k leads to better generalization, but also makes classification of a new sample a lot slower!

Support Vector Machine (SVM) (1)

* Find a hyperplane that separates the classes with a maximum margin

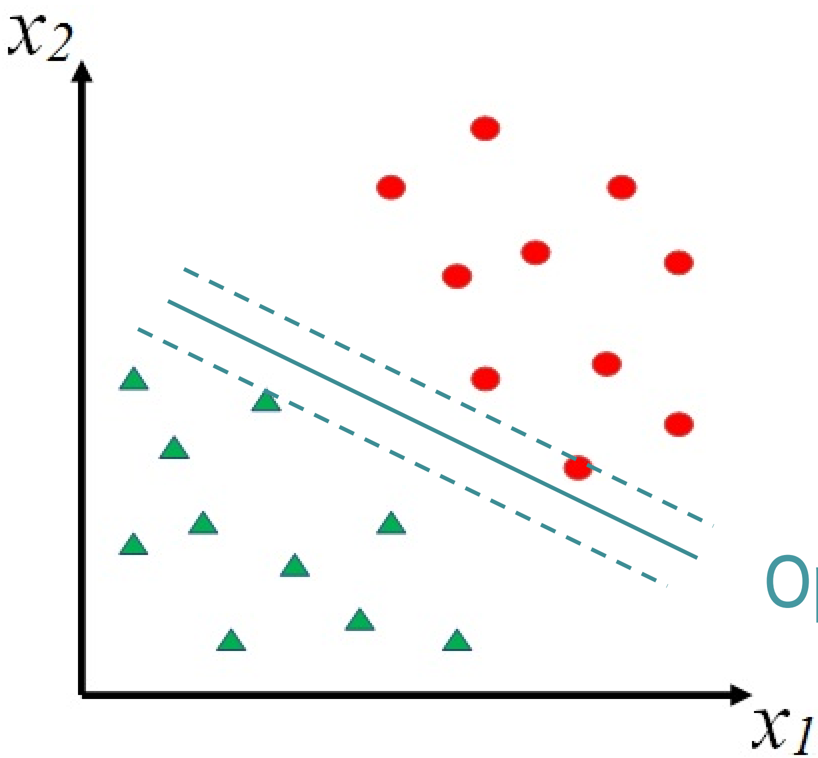


Support Vector Machine (SVM) (2)

- Based on emperical risk minimization (1960s)
  - Non-linearity added in 1992 (Boser, Guyon & Vapnik)
  - Soft-margin SVM introduced in 1995 (Cortes & Vapnik)
- Has become very popular since then
  - Easy to use, a lot of open libraries available
  - Fast learning and very fast classification
  - Good generalization properties

Support Vector Machine (SVM) (3)

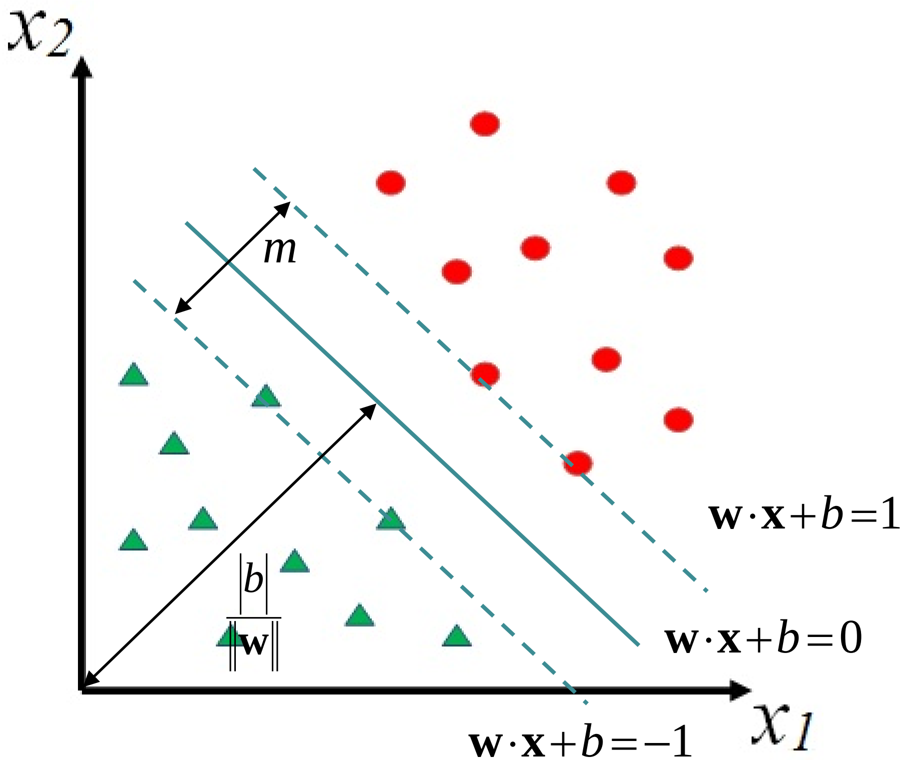
* How to find the optimal hyperplane?



Optimal? No!

Support Vector Machine (SVM) (4)

* How to find the optimal hyperplane?



Width of the margin:

$$m = \frac{|b|+1}{\|\mathbf{w}\|} - \frac{|b|-1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Maximize margin:

$$\max_{\mathbf{w},b} \quad \frac{2}{\|\mathbf{w}\|}$$

$$\text{subject to} \quad \mathbf{w}^T x_i + b \begin{cases} \geq 1 & y_i = 1 \\ \leq -1 & y_i = -1 \end{cases}$$

Support Vector Machine (SVM) (5)

* We can rewrite this to

$$\min_{\mathbf{w},b} ||\mathbf{w}||$$

$$\text{subject to } y_i(\mathbf{w}^T x_i + b) \geq 1$$

* Formulate as a Quadratic Programming problem:

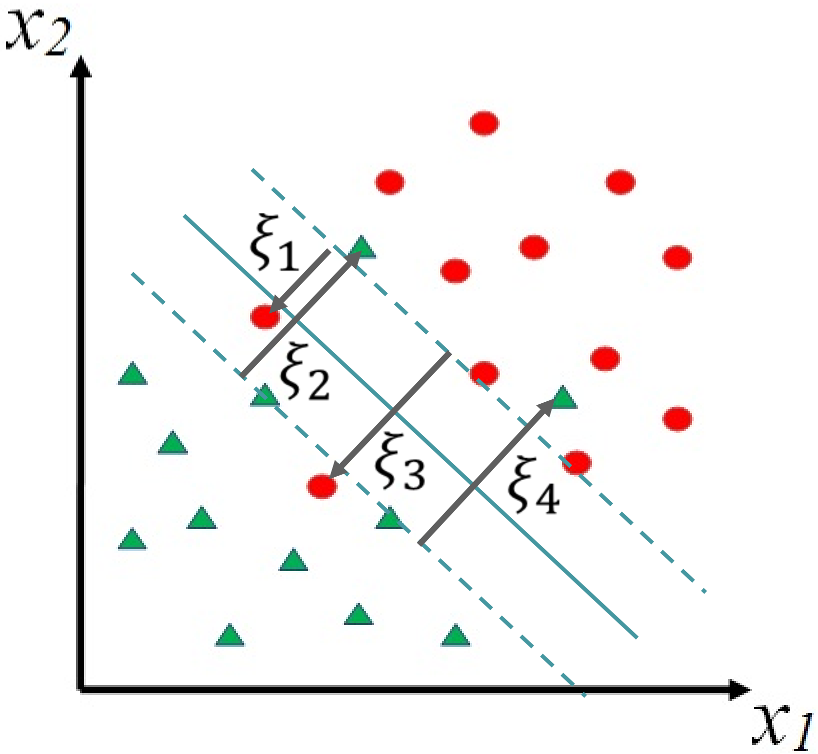
$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{subject to } y_i(\mathbf{w}^T x_i + b) \geq 1$$

Efficient methods available to solve this problem!

Support Vector Machine (SVM) (6)

* The data is usually not linearly separable…



Introduce slack variables ξ_i

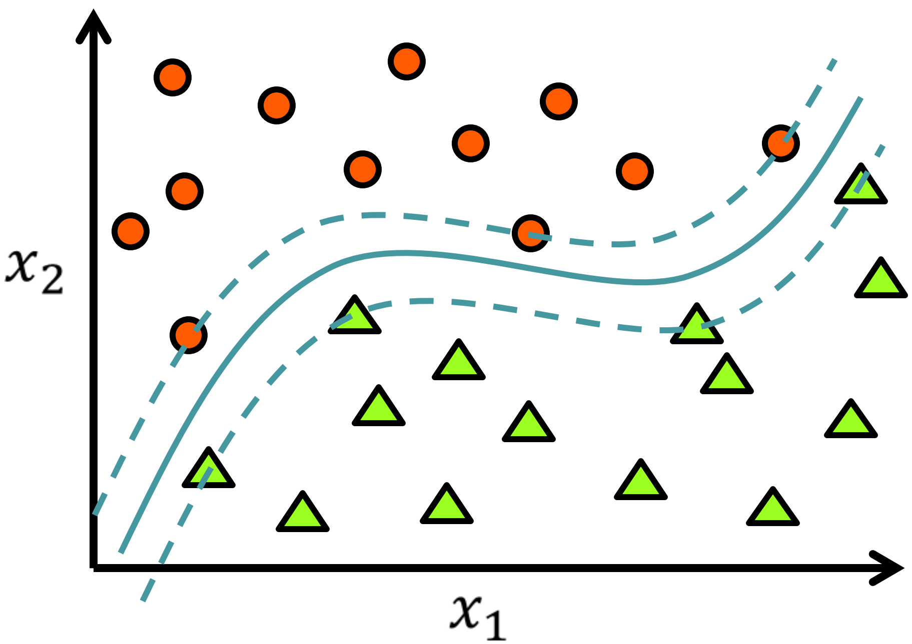
Put a cost *C* on crossing the margin, so the optimization problem becomes:

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$

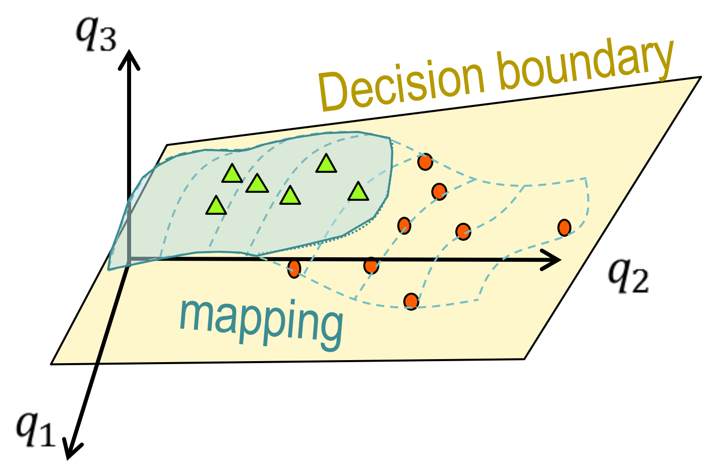
$$\text{subject to } y_i(\mathbf{w}^T + b) \geq 1$$

Support Vector Machine (SVM) (7)

* A more complex extension: non-linear SVMs

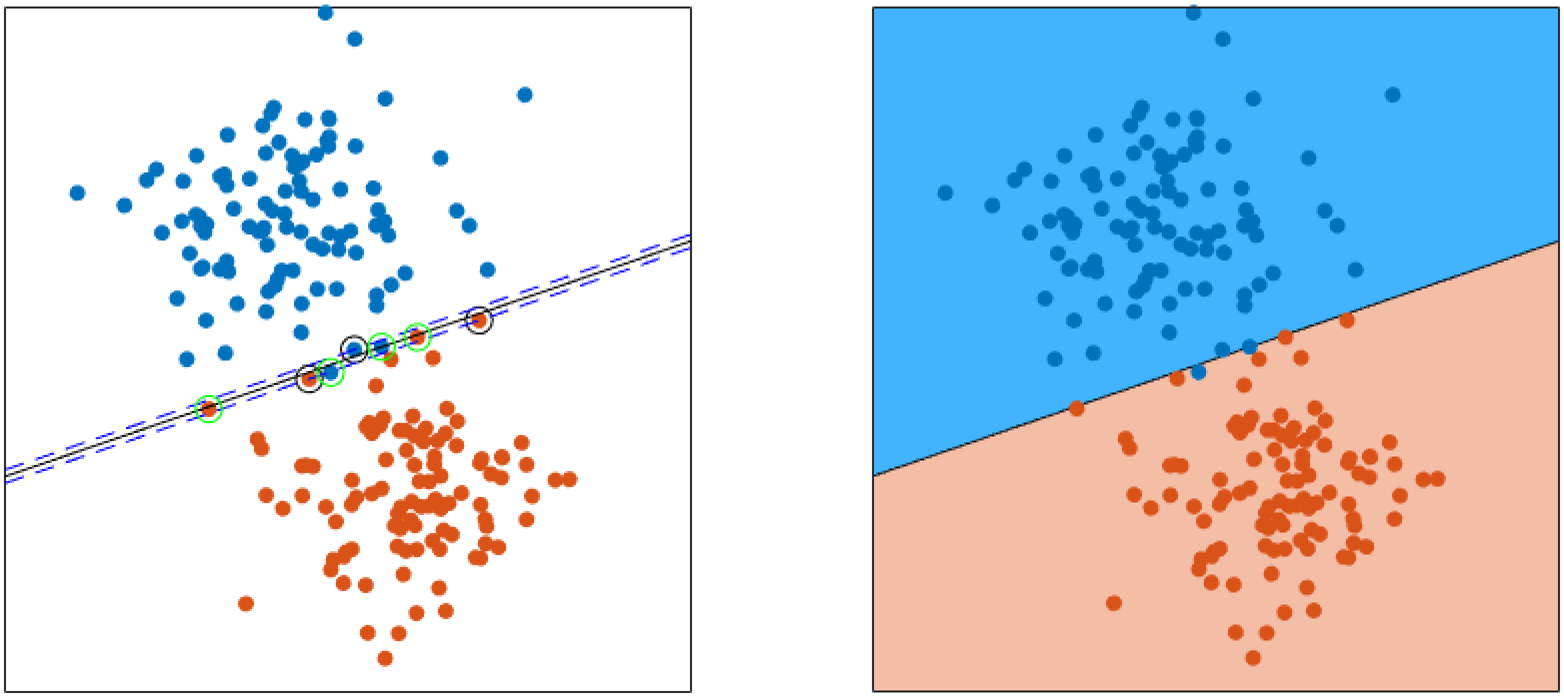


Basic idea: map the data to a higher-dimensional space, in which we can apply a linear SVM



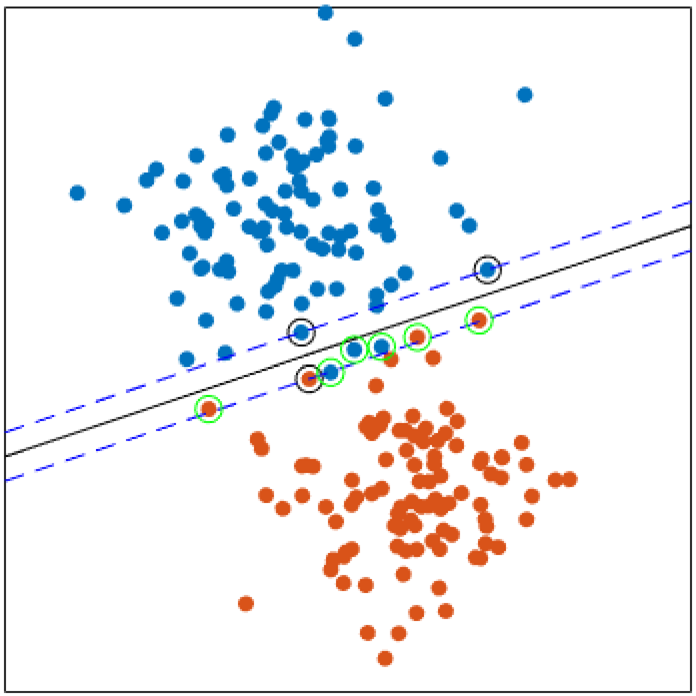
Support Vector Machine (SVM)
Cost parameter & generalization (1)

Optimal hyperplane for C=100 | SVM decision for C=100

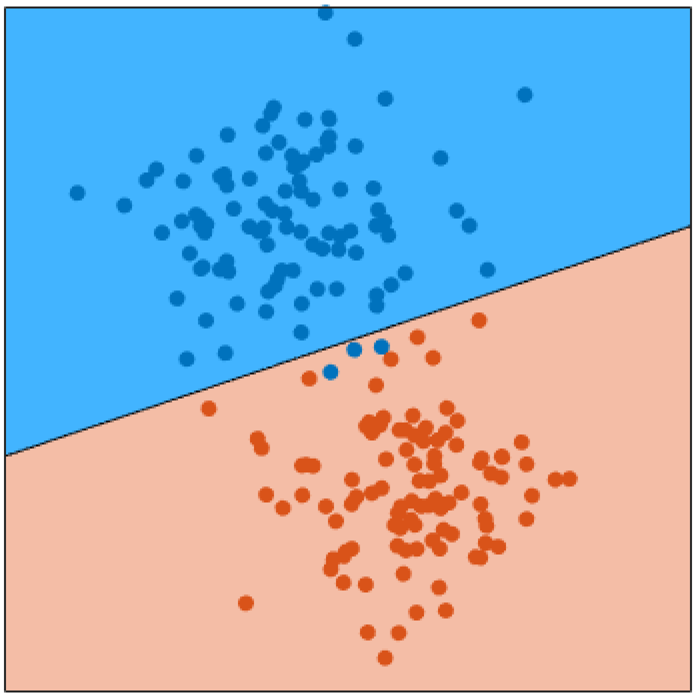


Support Vector Machine (SVM)
Cost parameter & generalization (2)

Optimal hyperplane for C=10

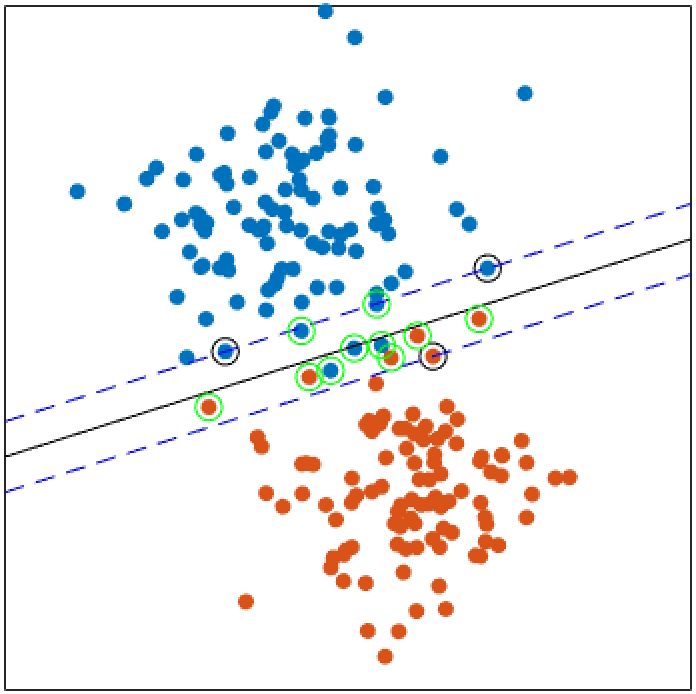


SVM decision for C=10

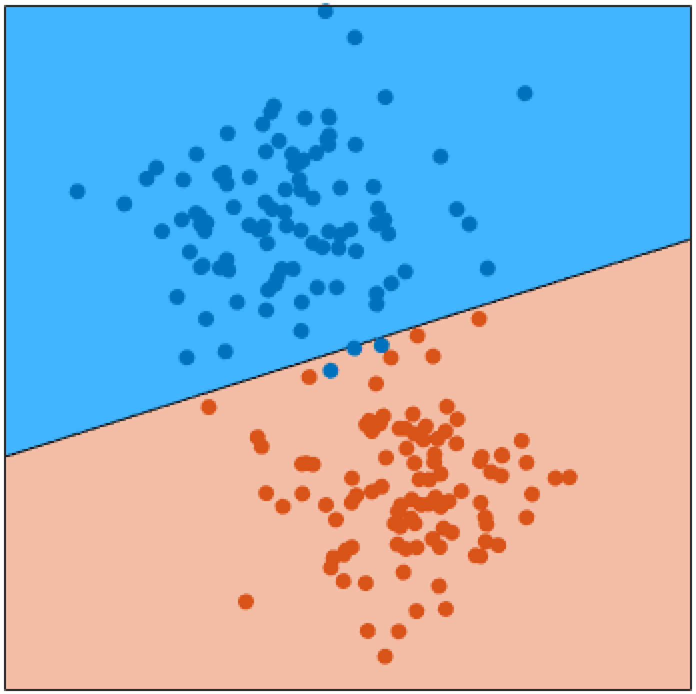


Support Vector Machine (SVM)
Cost parameter & generalization (3)

Optimal hyperplane for C=1

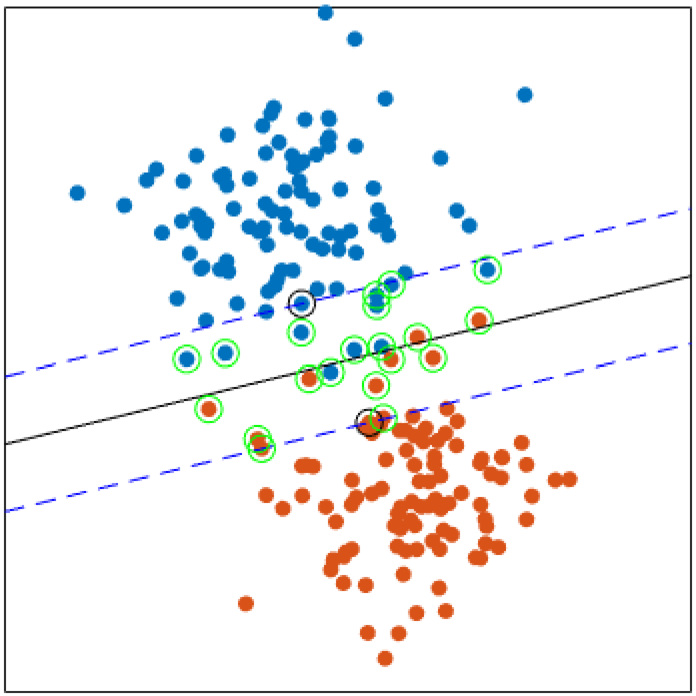


SVM decision for C=1

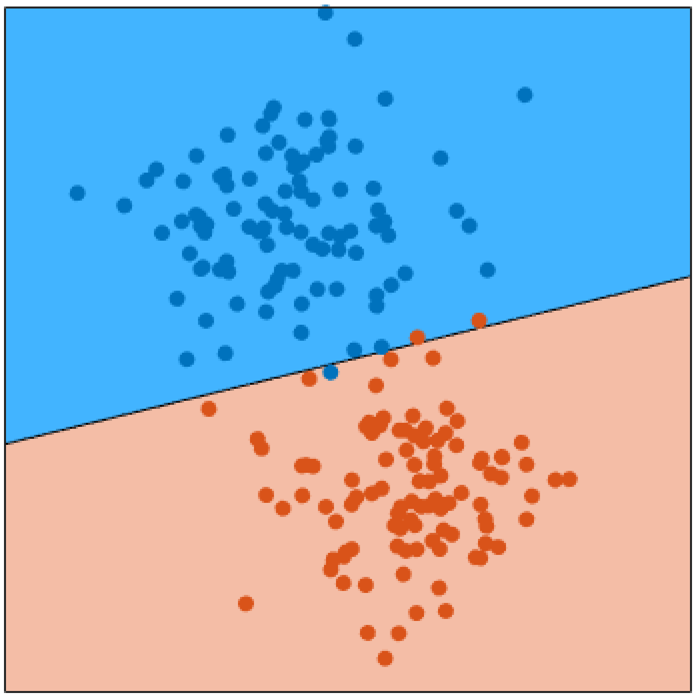


Support Vector Machine (SVM)
Cost parameter & generalization (4)

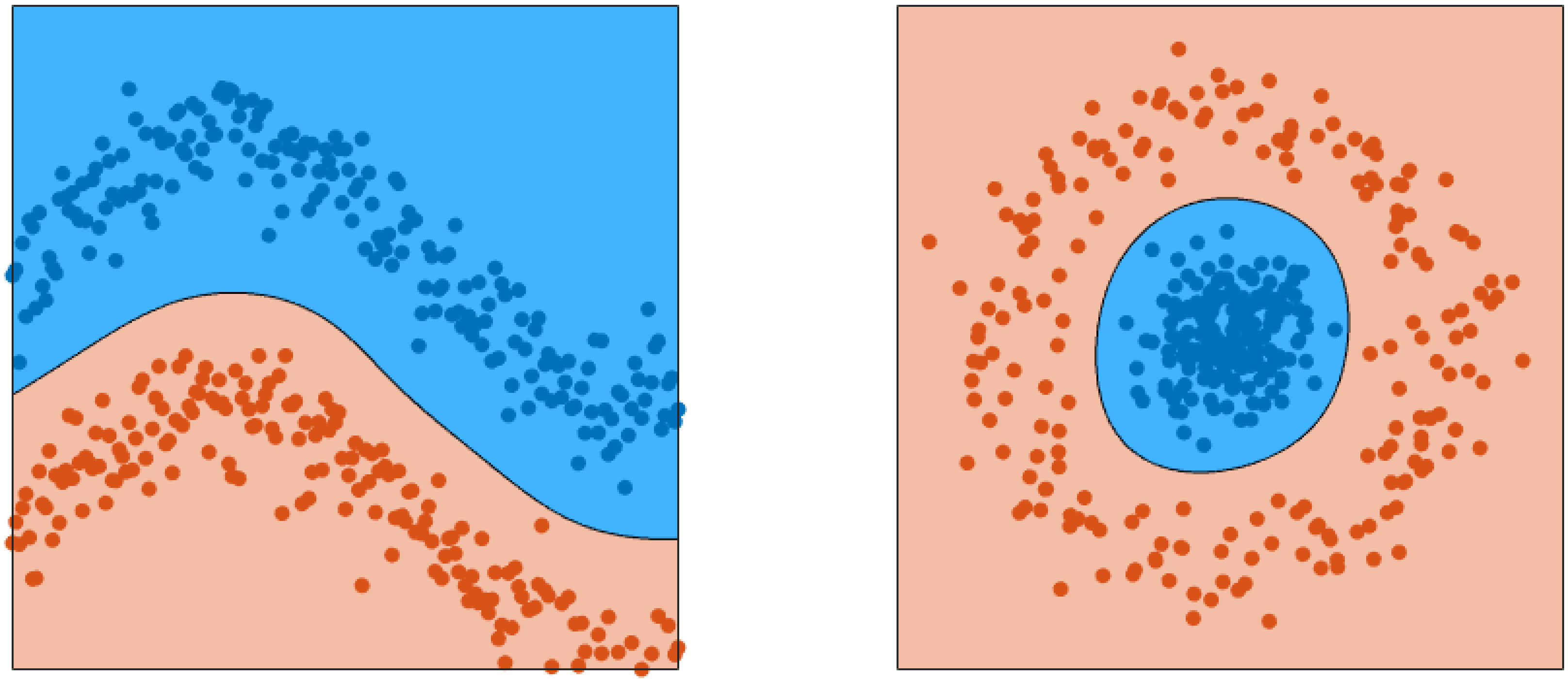
Optimal hyperplane for C=0.1



SVM decision for C=0.1



Support Vector Machine (SVM)
Non-linear SVM examples



Support Vector Machine (SVM)

* Summary
  – Fast and efficient method for <u>binary</u> classification
  – Splits the classes based on maximizing the margin
  – Optimal hyperplane can be computed using Quadratic Programming
  – Cost-parameter for points crossing the margin
  – Non-linear SVM can also handle more complex class distributions by mapping the data to another space

Random Forest (1)

- Build decision trees on subsets of the data
- Let the trees vote on the class of a new sample

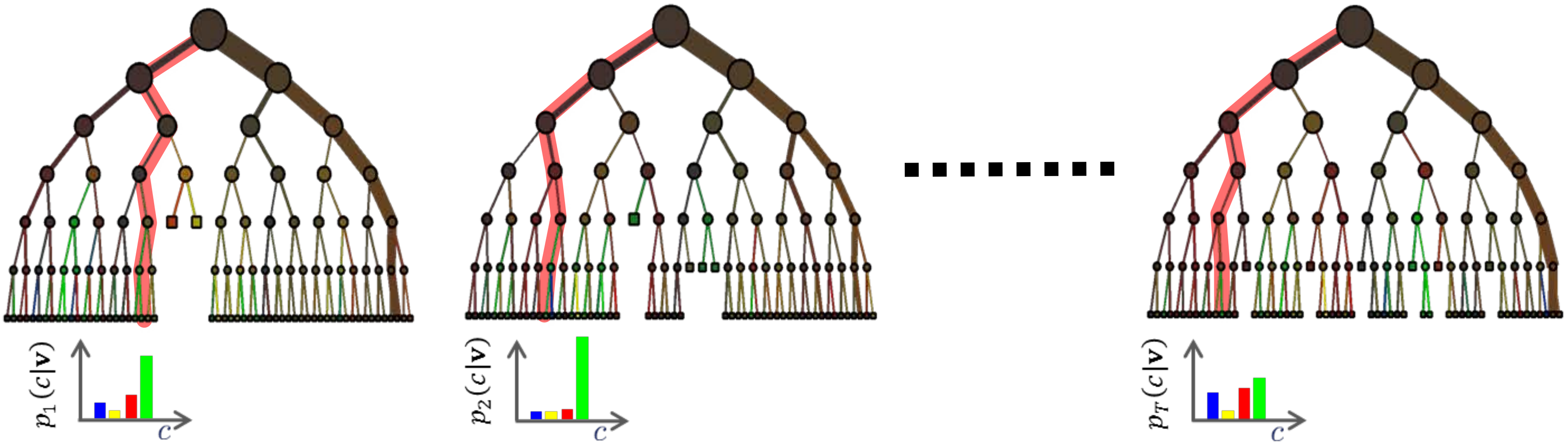


Image from a tutorial of Antonio Criminisi download at: http://research.microsoft.com/en-us/projects/decisionforests/

Random Forest (2)

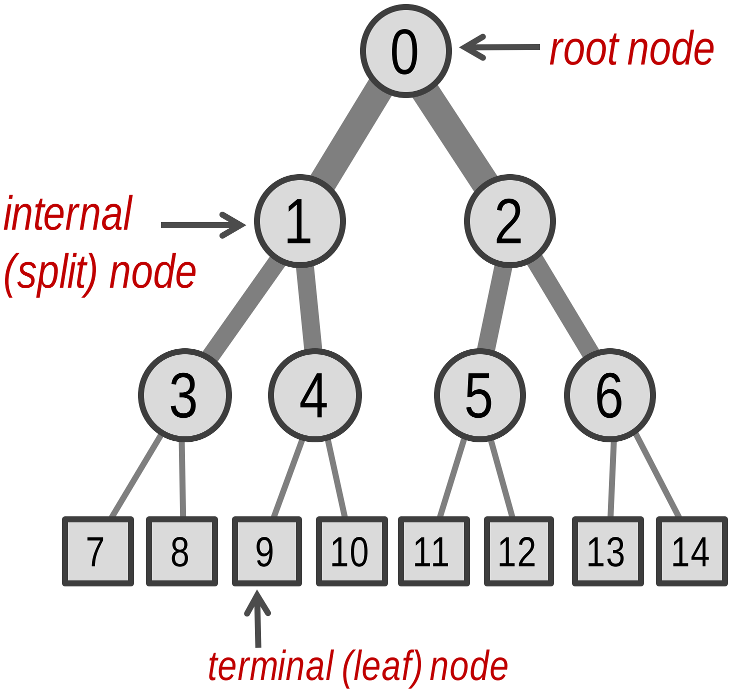
* Robustness through randomness
  - A random subset is used to train each tree
  - For training a tree, each node receives a random set of split options
* Intrinsically probabilistic output
  - Measure of confidence / uncertainty
* Automatic feature selection
* Naturally multi-class
* Runs efficiently – trees can run in parallel

Random Forest
Decision trees (1)

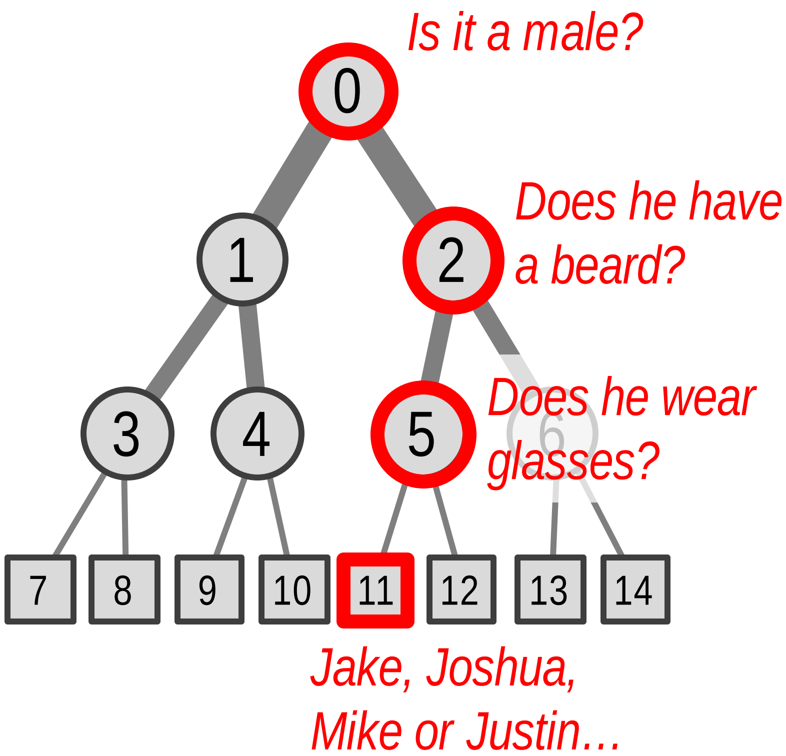
A forest consists of trees

A general tree structure

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



Random Forests
Decision trees (2)



*Example: GUESS WHO**

**Credits to Mark Janse*

Randomized Node Optimization (RNO)

Random Forests

Decision trees (3)

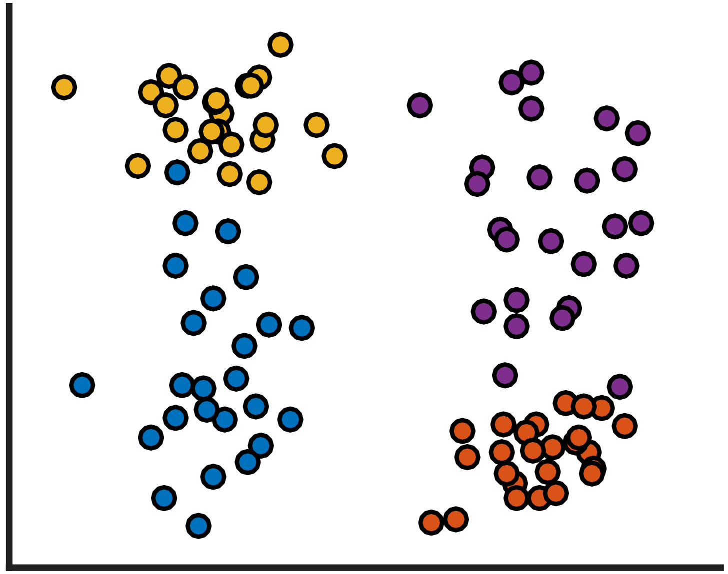
Bagging

* How to train a decision tree?
  - Start with a subset of all the training data at the root node
  - From a set of randomly chosen split options $\boldsymbol{\theta}$, select the one that maximizes some split metric (e.g. information gain)
  - Repeat this for all the 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)
    - All trianing samples in the node are from the same class

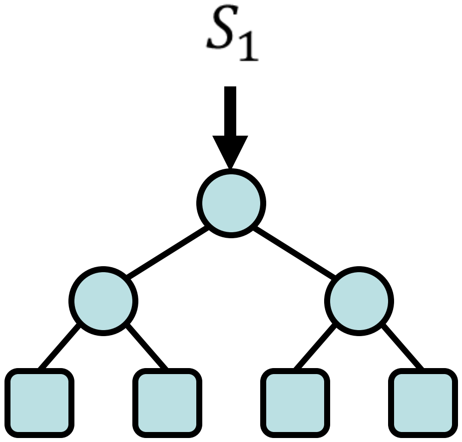
Random Forests
How to grow a tree? (1)

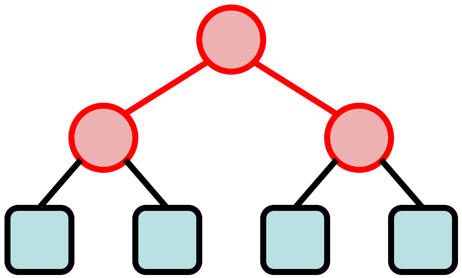
* Let's grow a tree with depth D = 2:

Subset S_1 of all availabe data S

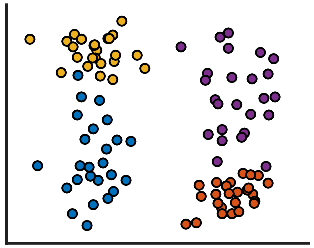


Start at the root node

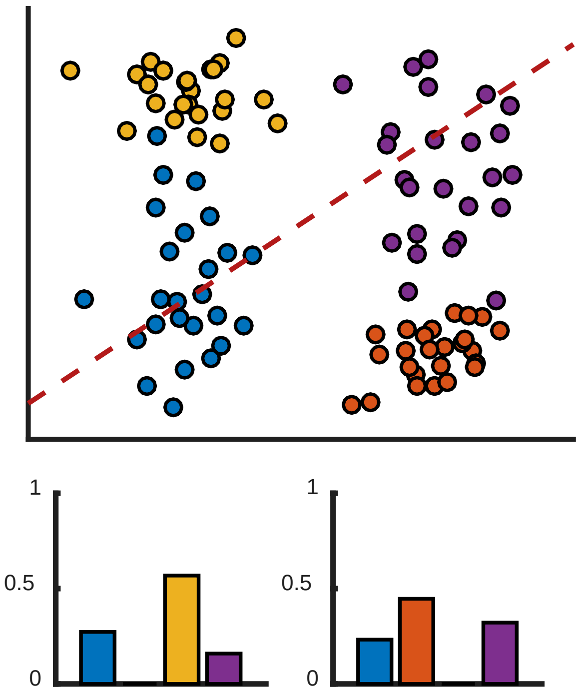




Random Forests
How to grow a tree? (2)

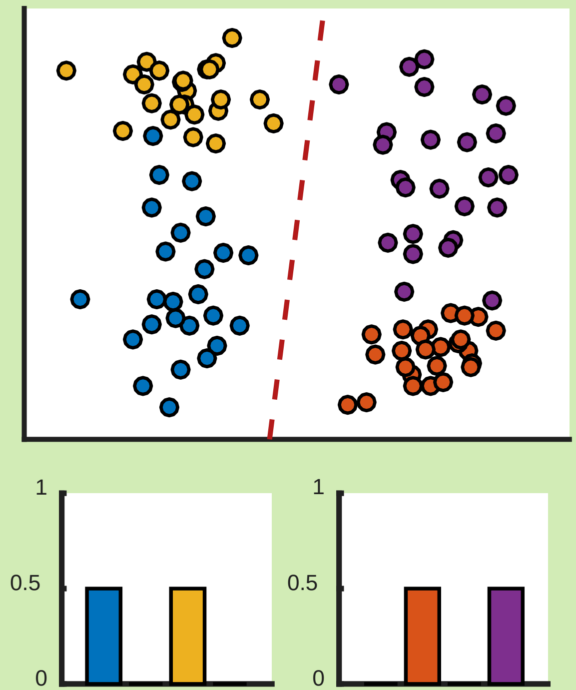


Option 1



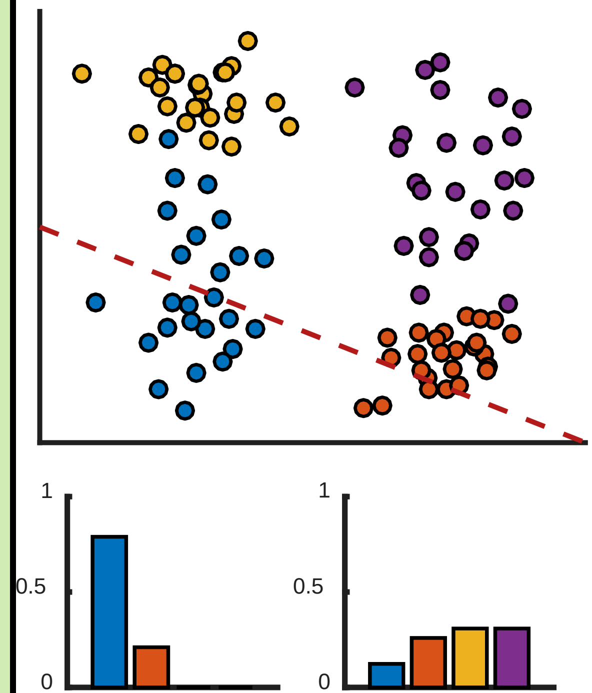
$I(\mathcal{S}_1, \theta_1) = 0.364$

Option 2

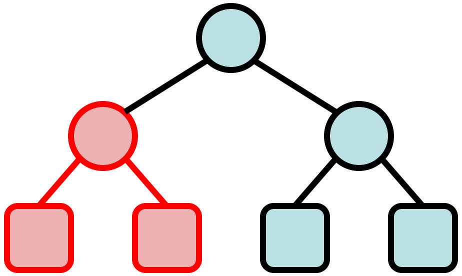


$I(\mathcal{S}_1, \theta_2) = 0.693$

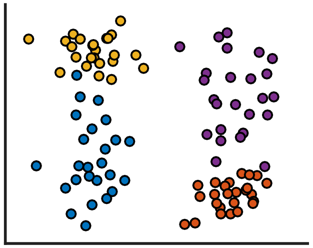
Option 3



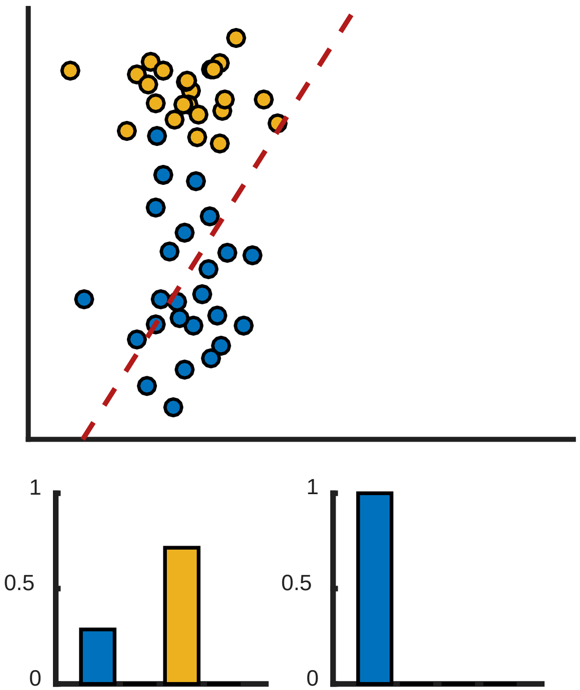
$I(\mathcal{S}_1, \theta_3) = 0.208$



Random Forests
How to grow a tree? (3)

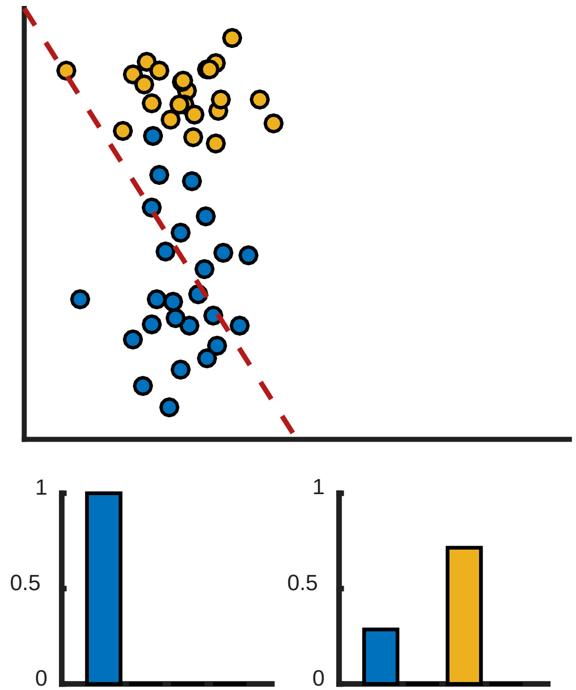


Option 1



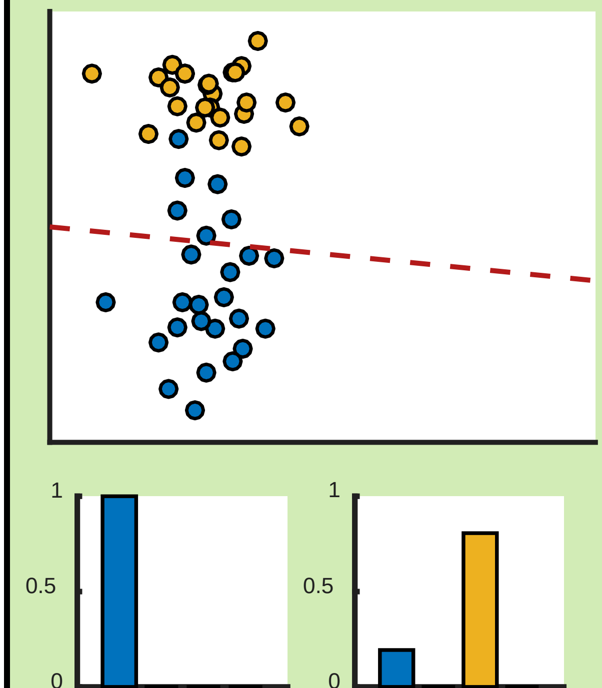
$I(\mathcal{S}_{1,L}, \theta_1) = 0.274$

Option 2

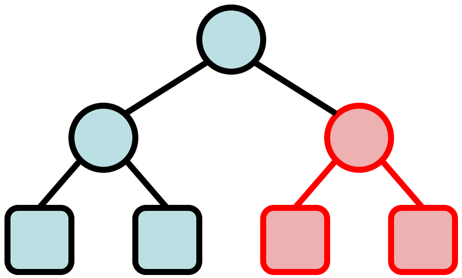


$I(\mathcal{S}_{1,L}, \theta_2) = 0.274$

Option 3

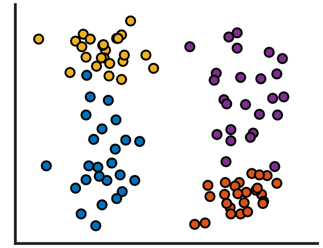


$I(\mathcal{S}_{1,L}, \theta_3) = 0.389$

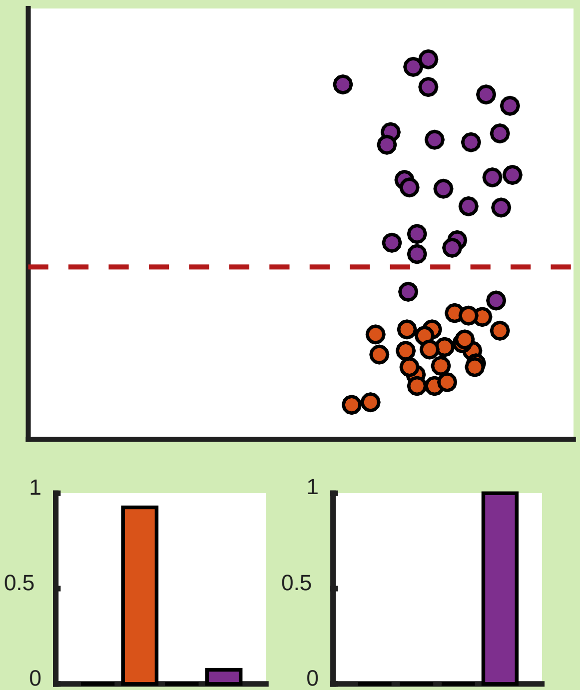


Random Forests

How to grow a tree? (4)

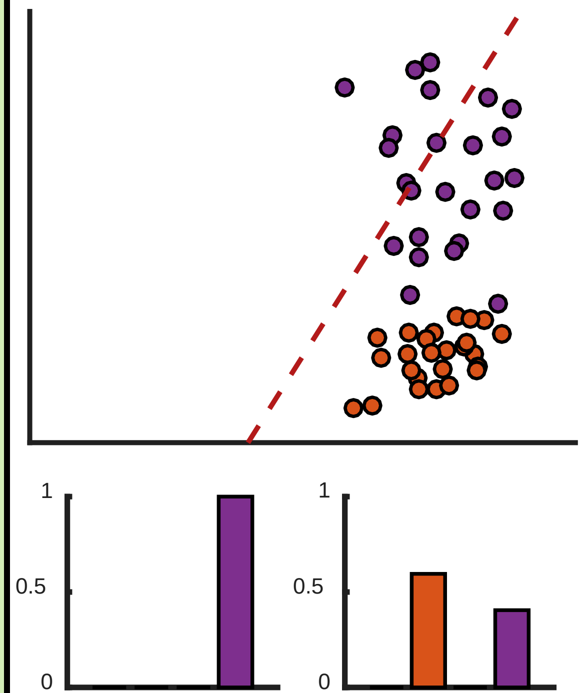


Option 1



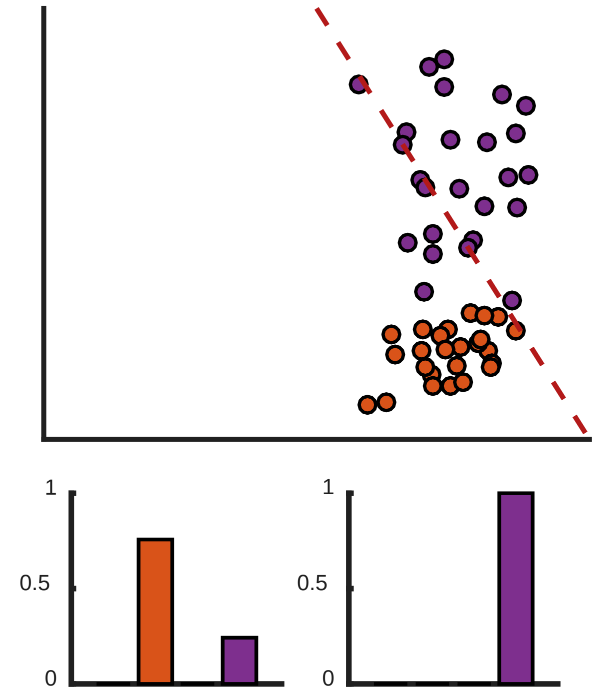
$$I\left(\mathcal{S}_{1,R}, \theta_1\right) = 0.551$$

Option 2



$$I\left(\mathcal{S}_{1,R}, 2\right) = 0.126$$

Option 3

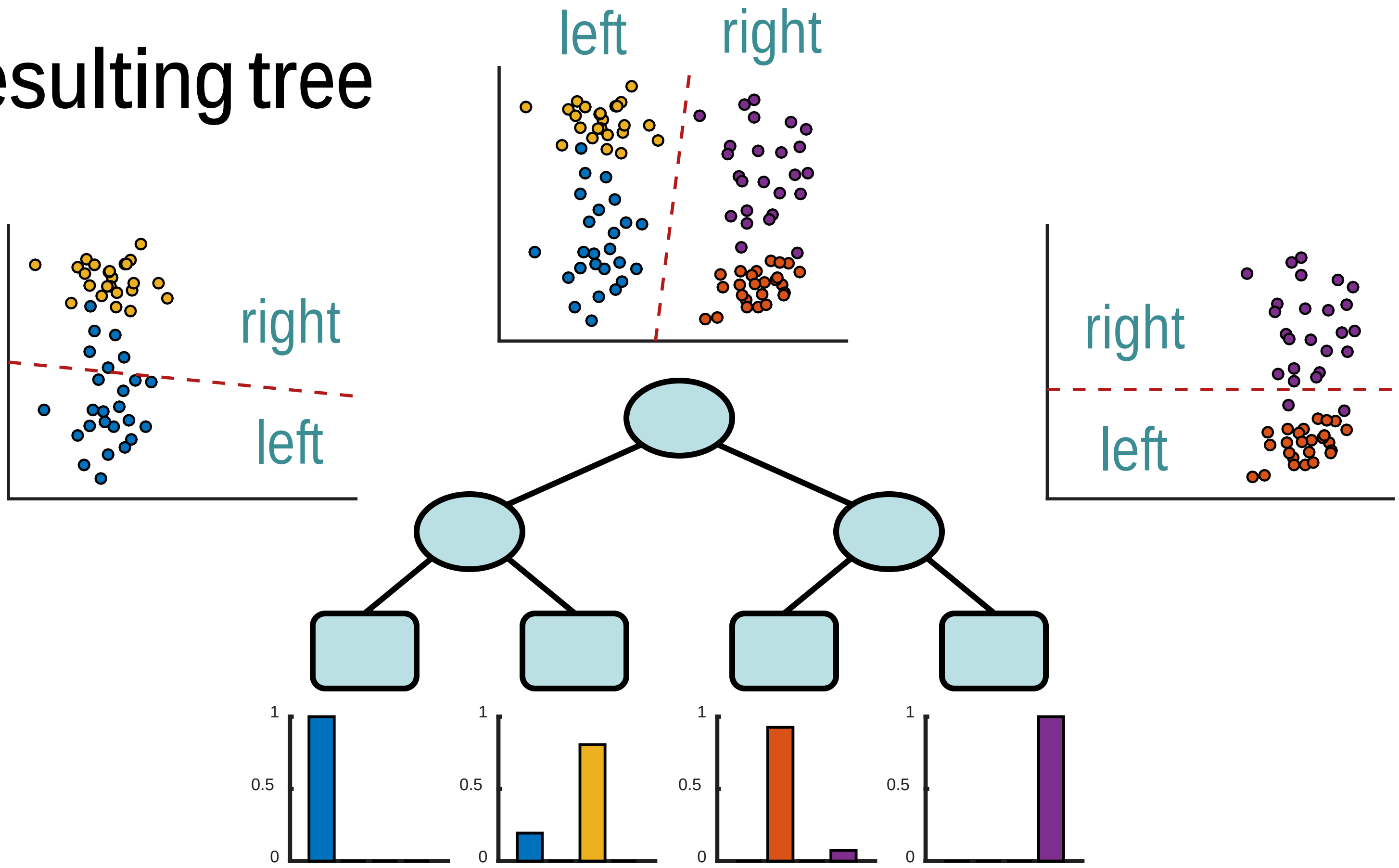


$$I\left(\mathcal{S}_{1,R}, \theta_3\right) = 0.328$$

Random Forests

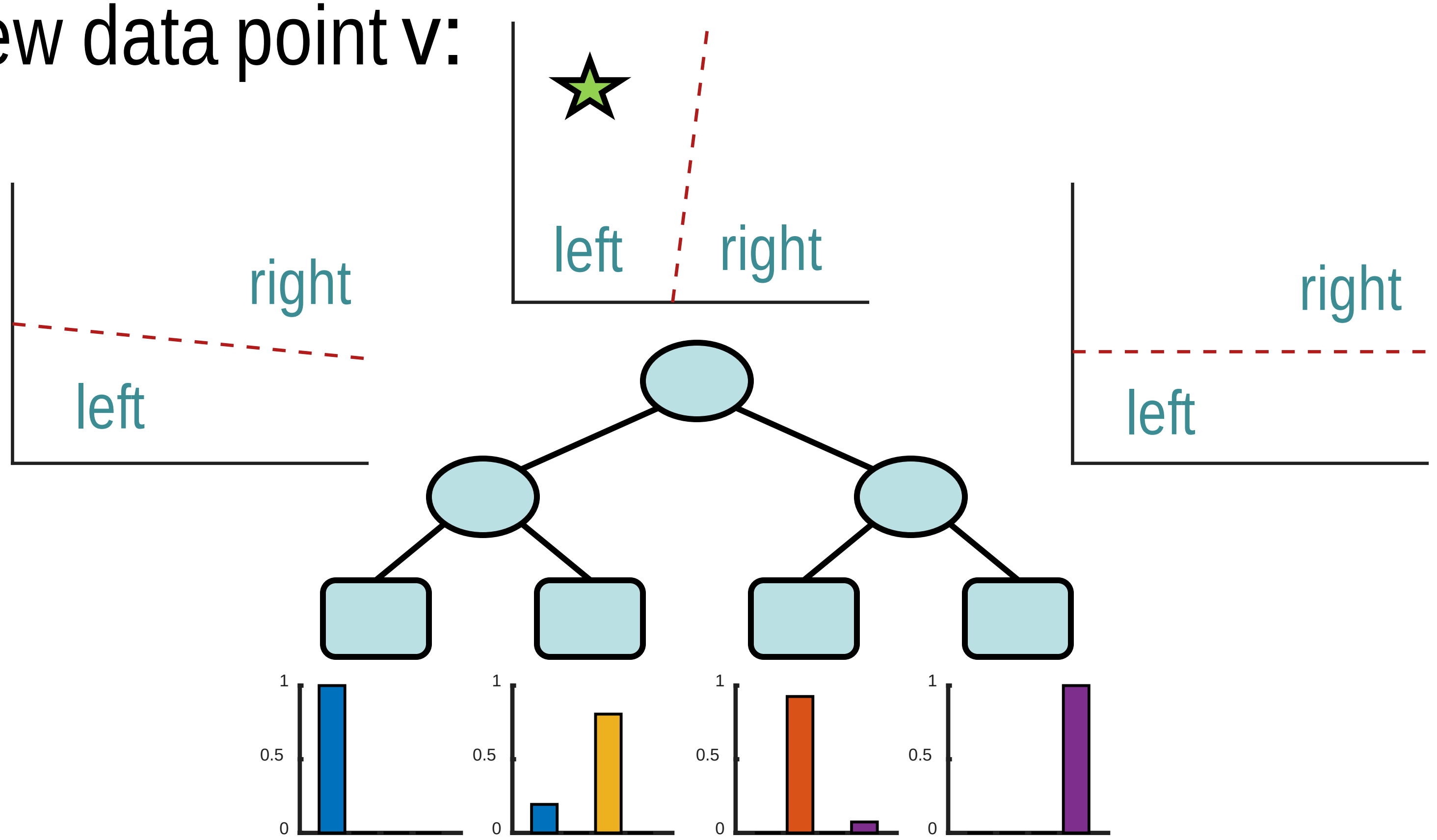
How to grow a tree? (5)

* Resulting tree



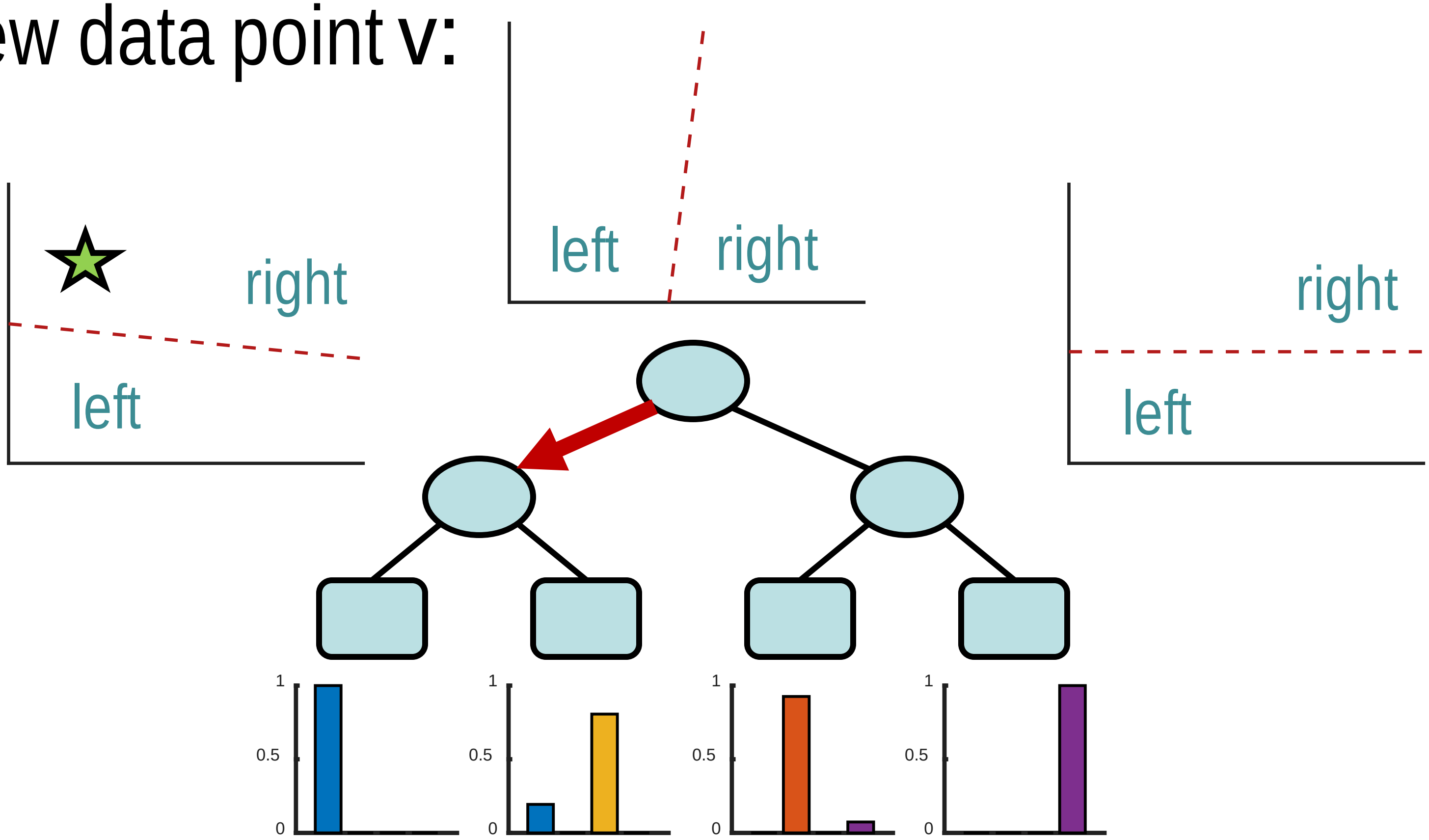
Random Forests
Classify a new data point (1)

* New data point v:



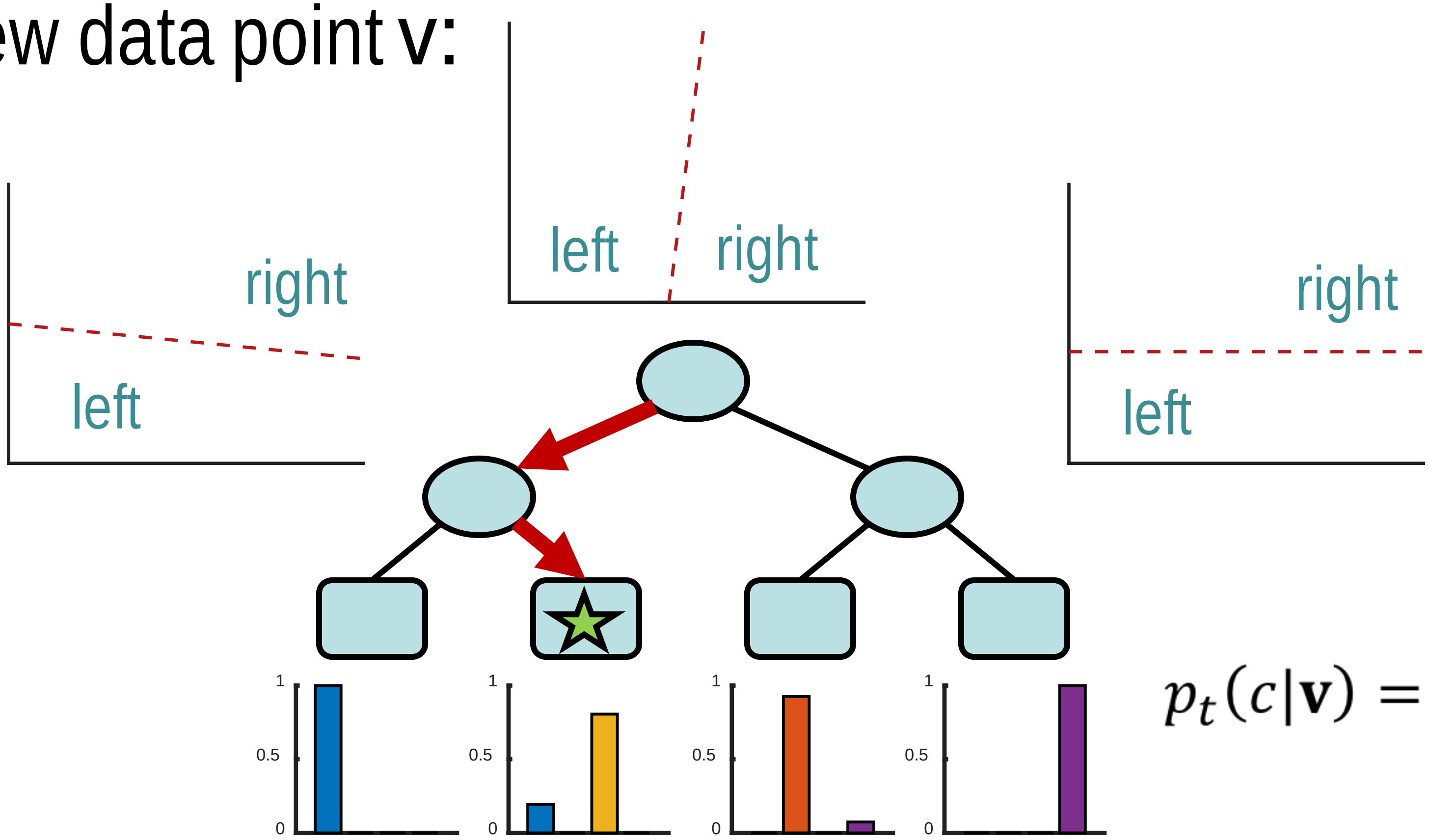
Random Forests
Classify a new data point (2)

* New data point v:



Random Forests
Classify a new data point (3)

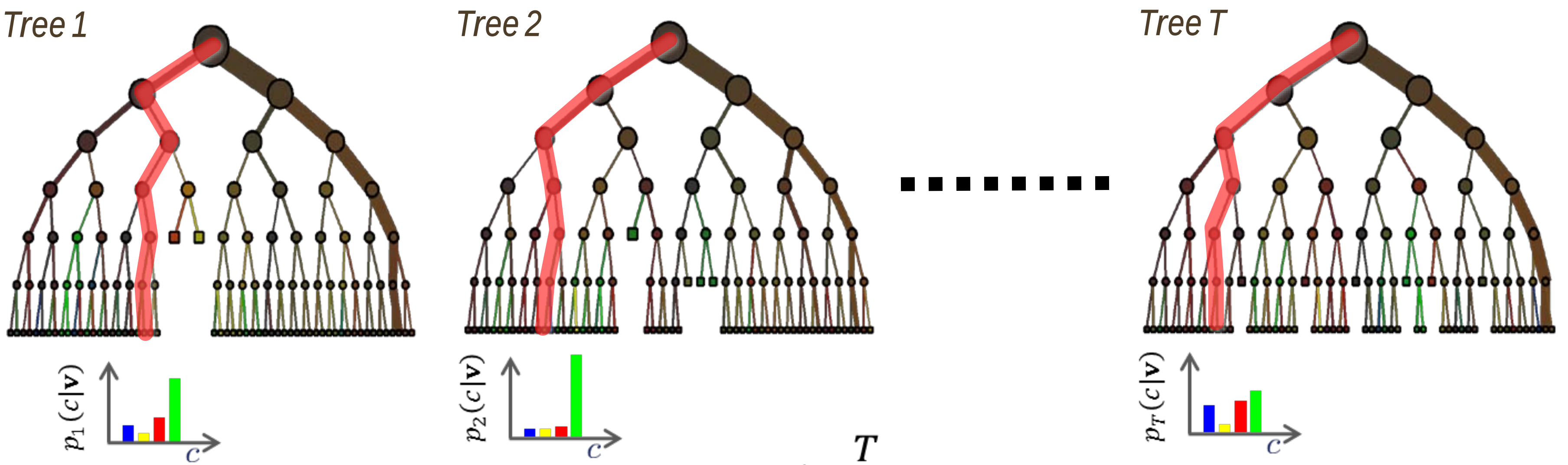
* New data point **v**:



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

Random Forests
Classification examples

* How to combine tree output?

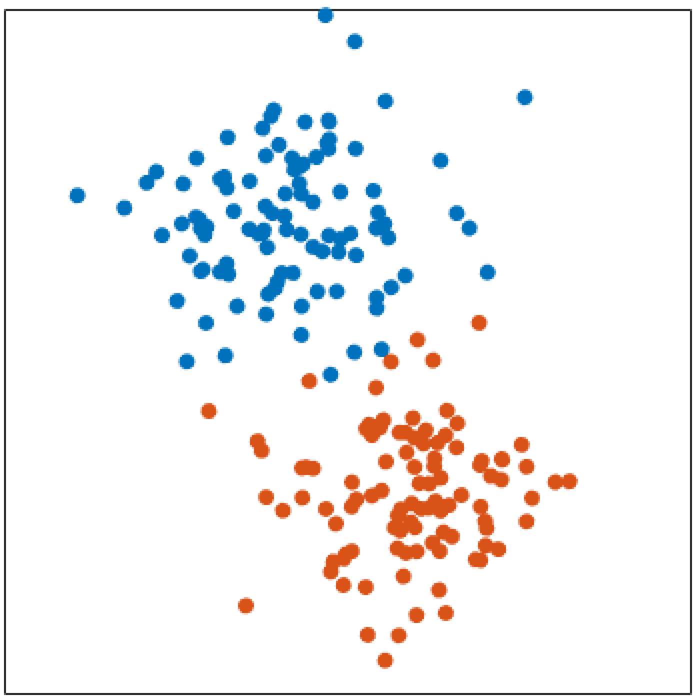


- Averaging: $$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$$

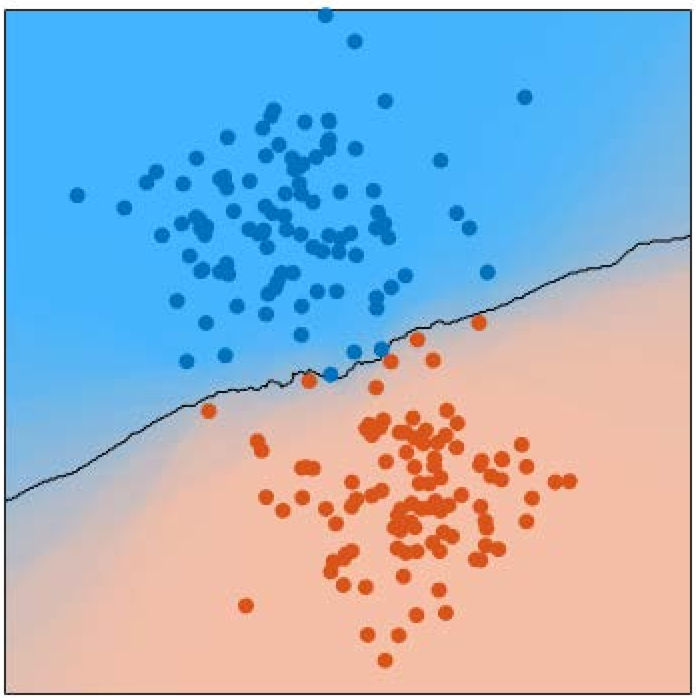
Image from a tutorial of Antonio Criminisi download at: http://research.microsoft.com/en-us/projects/decisionforests/

Random Forests
Classification example (1)

2 classes in feature space



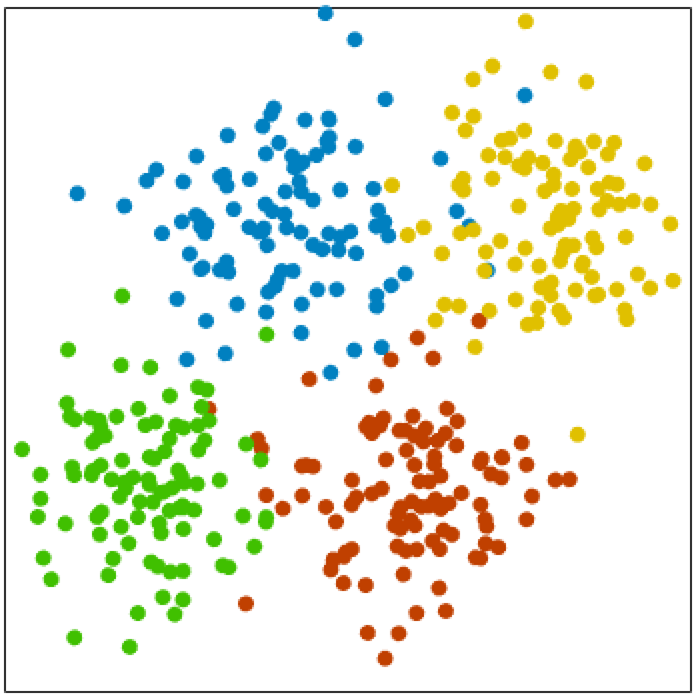
Random forest decision



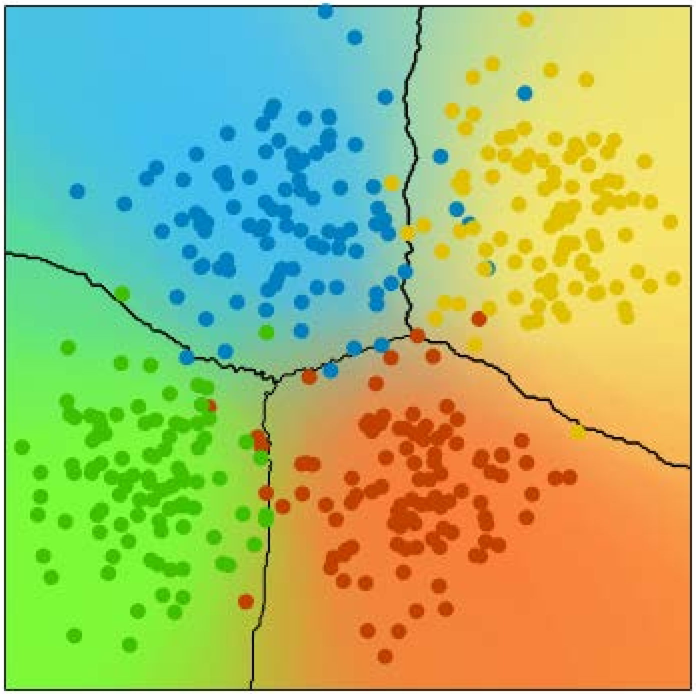
N = 100 trees, max number of nodes = 5, # candidate splits per node = 3

Random Forests
Classification example (2)

4 classes in feature space



Random forest decision

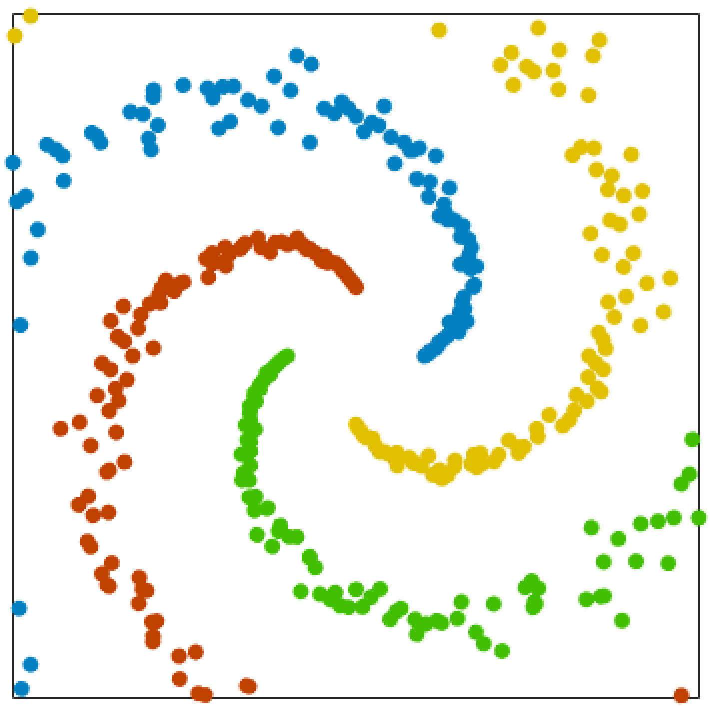


N = 100 trees, max number of nodes = 4, # candidate splits per node = 3

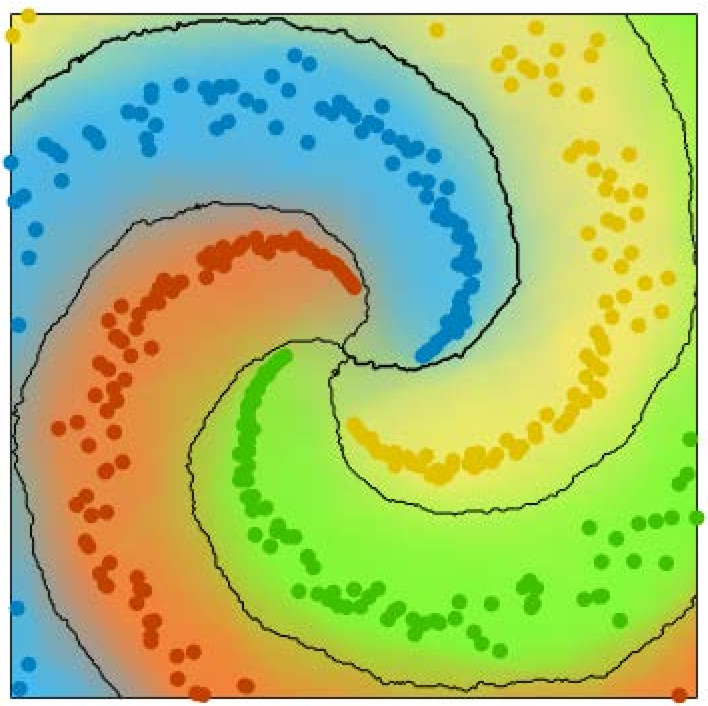
Random Forests

Classification example (3)

4 classes in feature space



Random forest decision



N = 100 trees, max number of nodes = 10, # candidate splits per node = 8

Random Forests

* Summary
  - Good generalization due to randomness model
    - Bagging
      - Each tree is trained on a randomly selected subset of the data
    - Randomized Node Optimization (RNO)
      - Each node receives a randomly selected subset of all possible split options.
  - Multi-class classification with probablistic output
  - Suboptimal splits lead to a robust model
  - Result depends heavily on the forest parameters

Performance evaluation

* So, now we have model, how good is it?
  - We have labeled data (ground truth), so we can validate!
* Model validation:
  - Separate sets for training and testing the model
    - Train the model using the training set
    - Use the test set to evaluate the performance
  - Compute figures of merit, which indicate the performance
  - What is a good performance metric? And how should we split the data?

Performance evaluation

Number of samples

* Some popular figures of merit:
  - Accuracy (#TP + #TN) / $\underbrace{\text{(\#TP + \#FN +\#TN + \#FP)}}_{\text{Number of samples}}$
  - Sensitivitiy (#TP) / (#TP + #FN) *a.k.a. True Positive Rate*
  - Specificity (#TN) / (#TN + #FP) *a.k.a. True Negative Rate*

Where

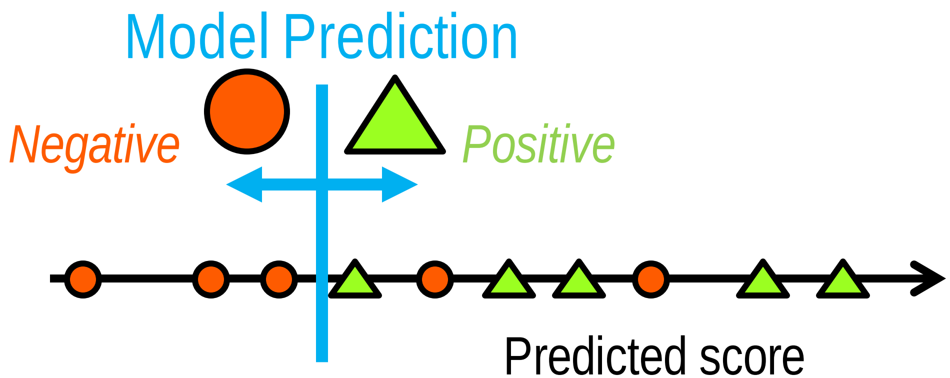
True Positive (TP): positive sample classified as positive
True Negative (TN): negative sample classified as negative
False Positive (FP): negative sample classified as positive
False Negative (FN): positive sample classified as negative

Performance evaluation

* Receiver Operating Characteristic (ROC)
  - Sensitivity / specificity give the performance for just one possible setting (i.e. decition threshold) of the model
  - We can vary this threshold and recompute these performance metrics
  - This yields a curve of possible combinations of sensitivity and specificity, called the ROC curve
  - Generally true: ↑ sensitivity ↓ specificity and vice versa

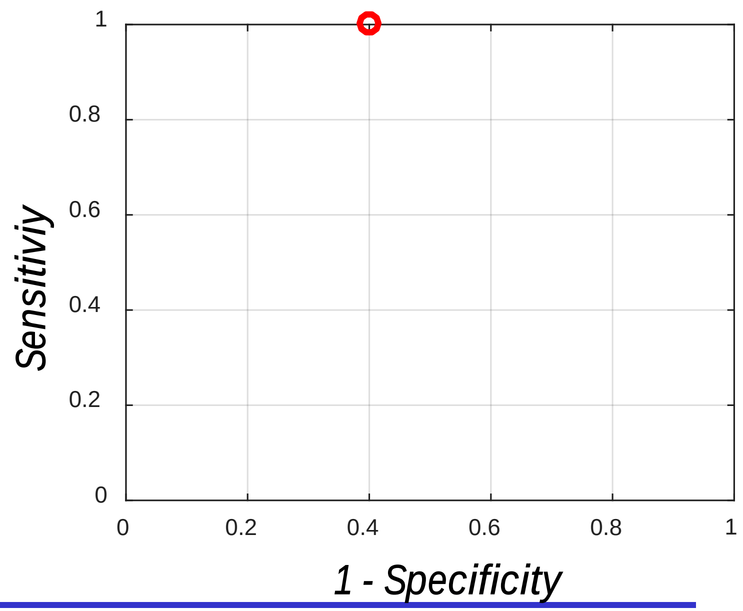
Performance evaluation

- How to compute the ROC curve?
  - For each sample we have a predicted class and a score
  - Sort the samples according to score and move the threshold



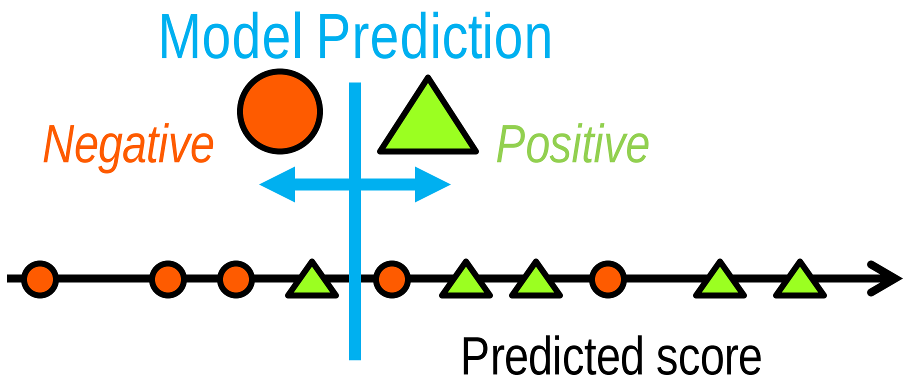
Sensitivitiy = 5 / (5+0) = 1.00

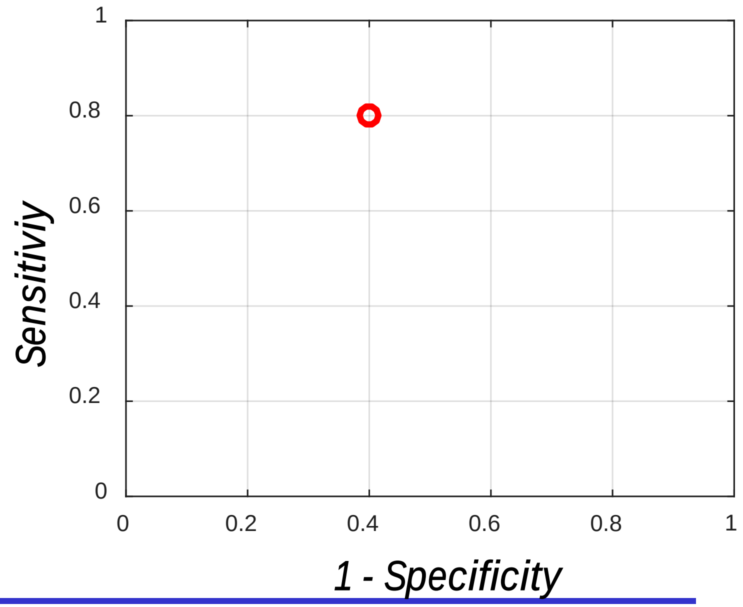
Specificity = 3 / (3+2) = 0.60



Performance evaluation

* How to compute the ROC curve?
  – For each sample we have a predicted class and a score
  – Sort the samples according to score and move the threshold



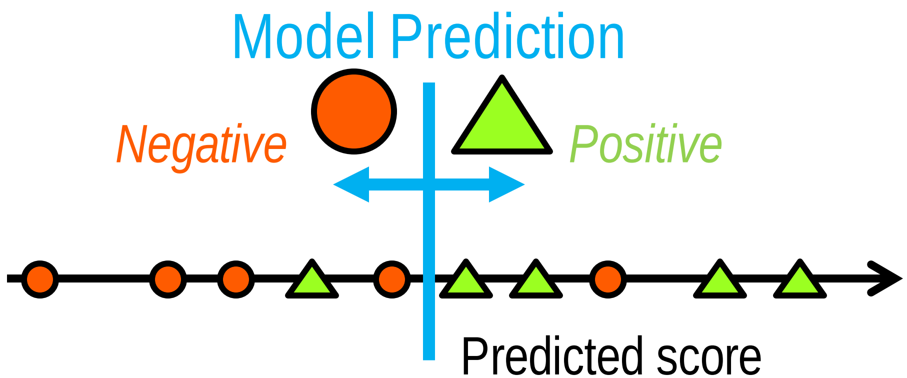


Sensitivitiy = 4 / (4+1) = 0.80

Specificity = 3 / (3+2) = 0.60

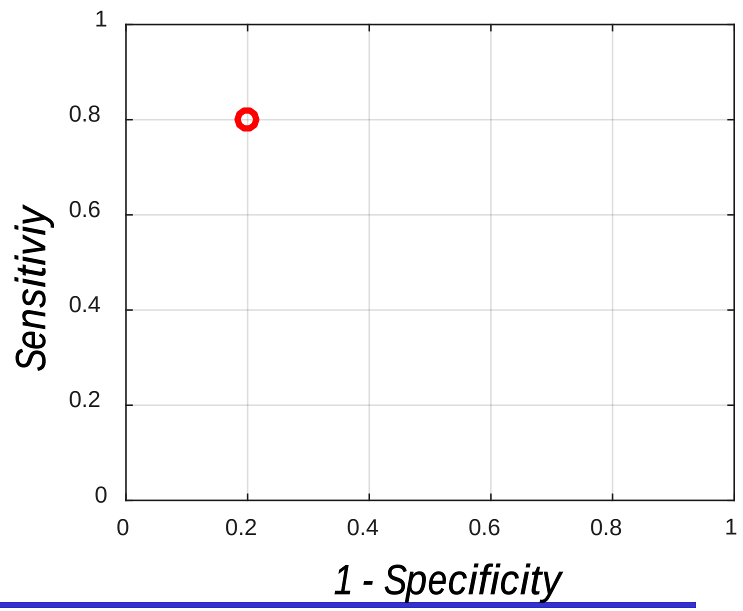
Performance evaluation

- How to compute the ROC curve?
  - For each sample we have a predicted class and a score
  - Sort the samples according to score and move the threshold



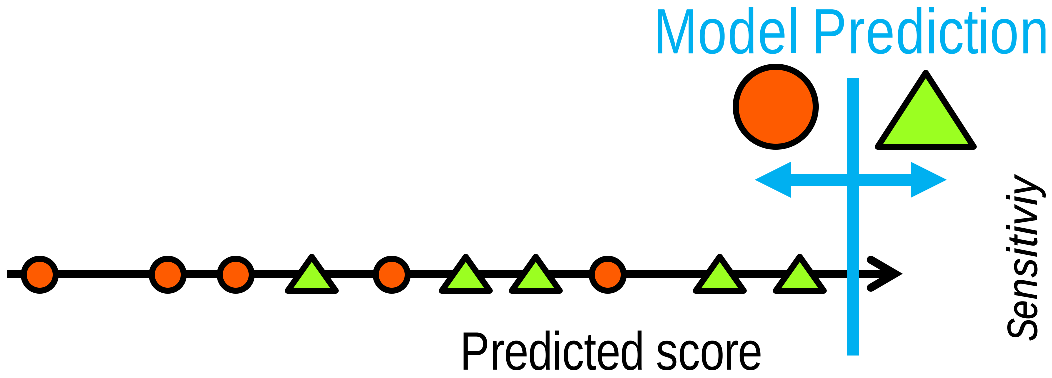
Sensitivitiy = 4 / (4+1) = 0.80

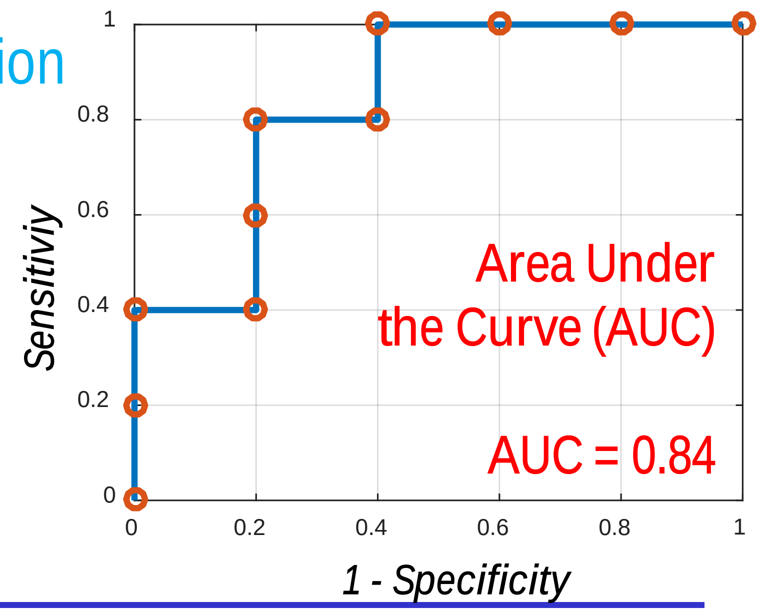
Specificity = 4 / (4+1) = 0.80



Performance evaluation

* How to compute the ROC curve?
  – For each sample we have a predicted class and a score
  – Sort the samples according to score and move the threshold



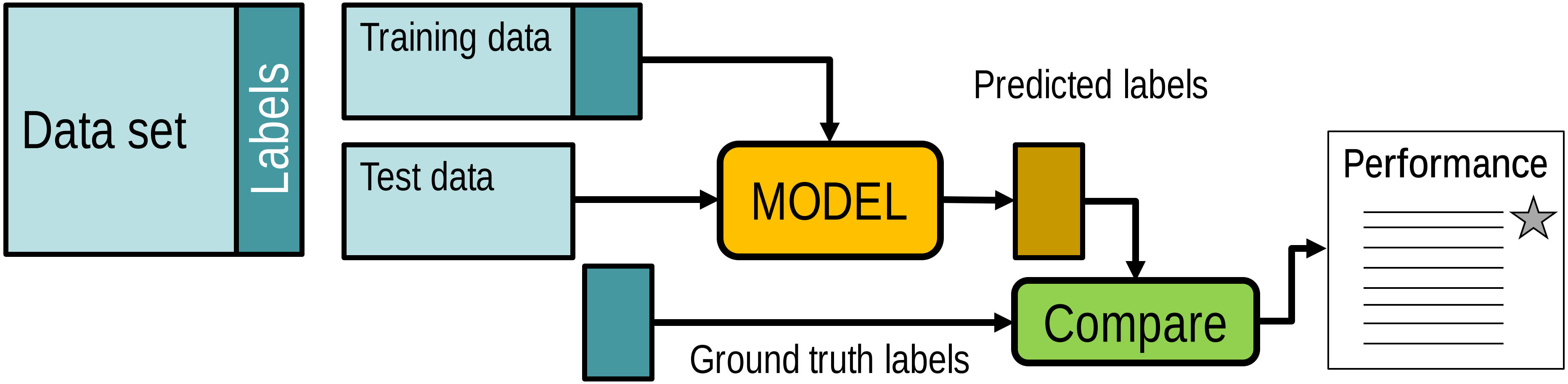


Sensitivitiy = 0 / (0+5) = 0.00

Specificity = 5 / (5+0) = 1.00

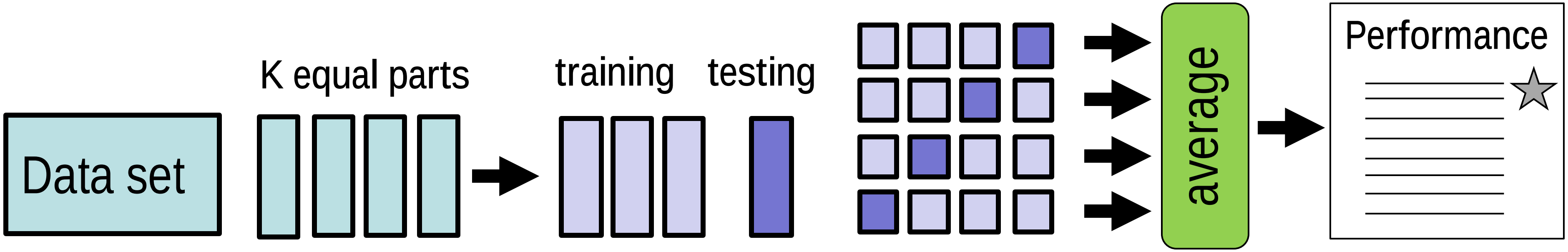
Performance evaluation

- Large data set: randomly sample half the samples for training and half for testing
  - Training and testing is time consuming for large datasets
  - The test set is probably a good reflection of the training set



Performance evaluation

- How should we split the data?
  - Different choices might lead to different results…
- K-fold cross-validation
  - Split the data in K equally sized parts
  - Use K-1 parts for training and use the left-out part of the data for testing, repeat this for each part and average:

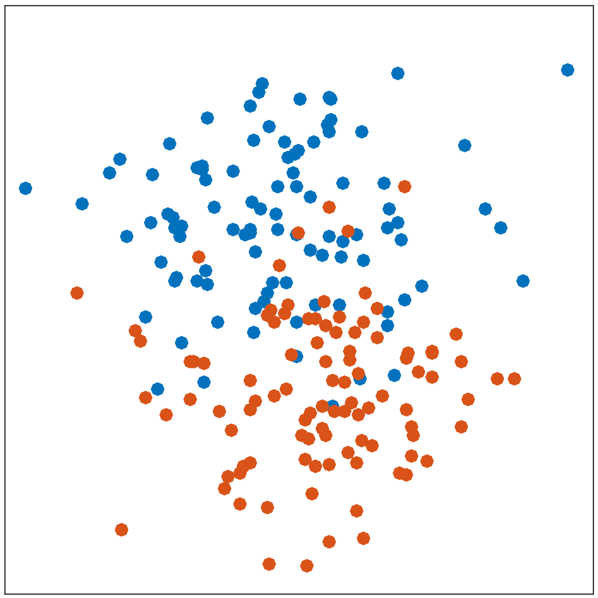


Performance evaluation

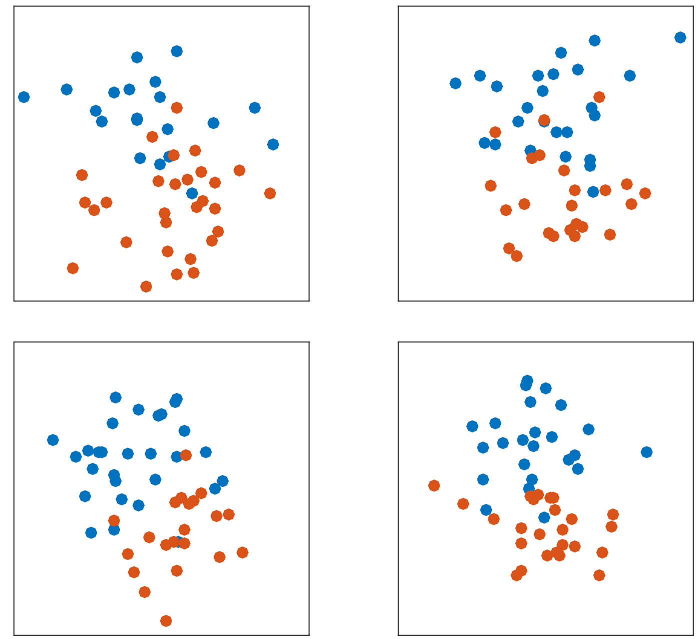
- Leave-One-Out Cross-Validation
  - Leave one sample out of the complete set and use the remaining set to train the model
  - Test the model on the left-out sample
  - Repeat this for all samples.
- Best performance indication for small data set
  - You want to use as much of the little data you have for training the model

Performance evaluation

EXAMPLE: 4-fold cross validation (1)



Split in 4
equally-sized
partitions

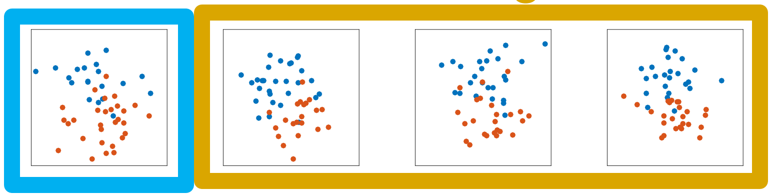


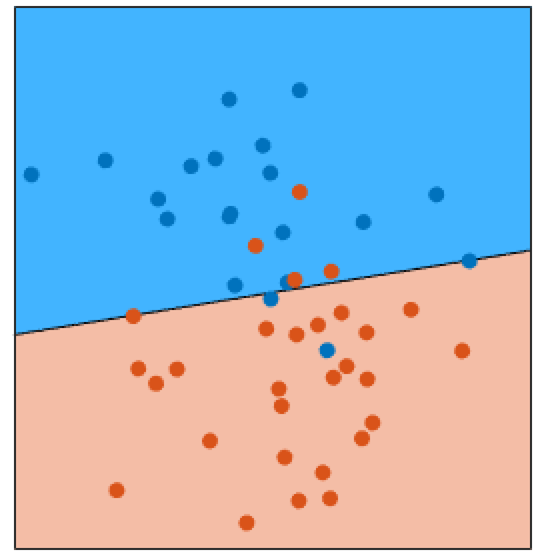
Performance evaluation

EXAMPLE: 4-fold cross validation (2)

Test set Training set

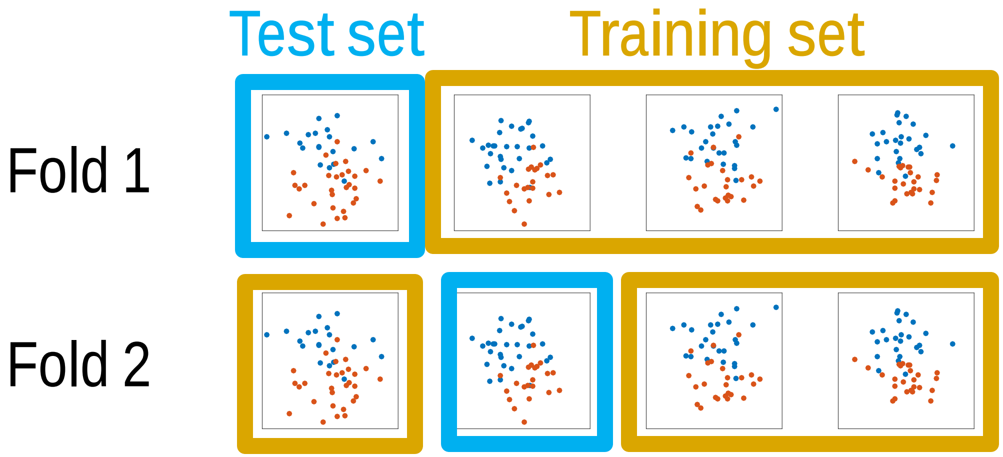
Fold 1

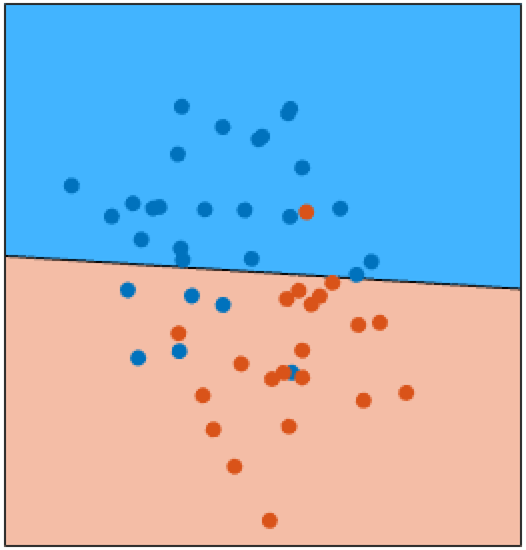




Fold 1: Accuracy = 0.86

Performance evaluation
EXAMPLE: 4-fold cross validation (3)

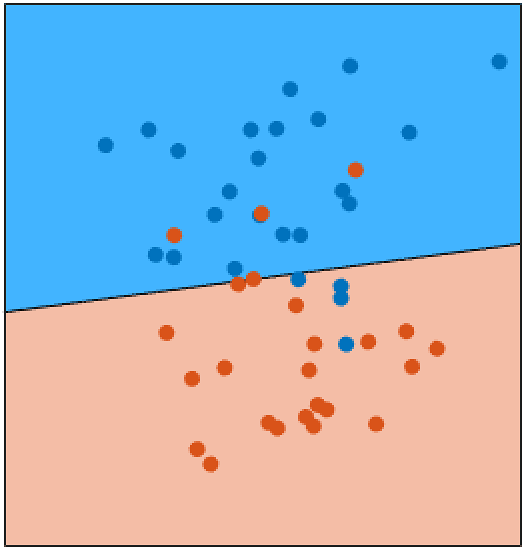




Fold 2: Accuracy = 0.86

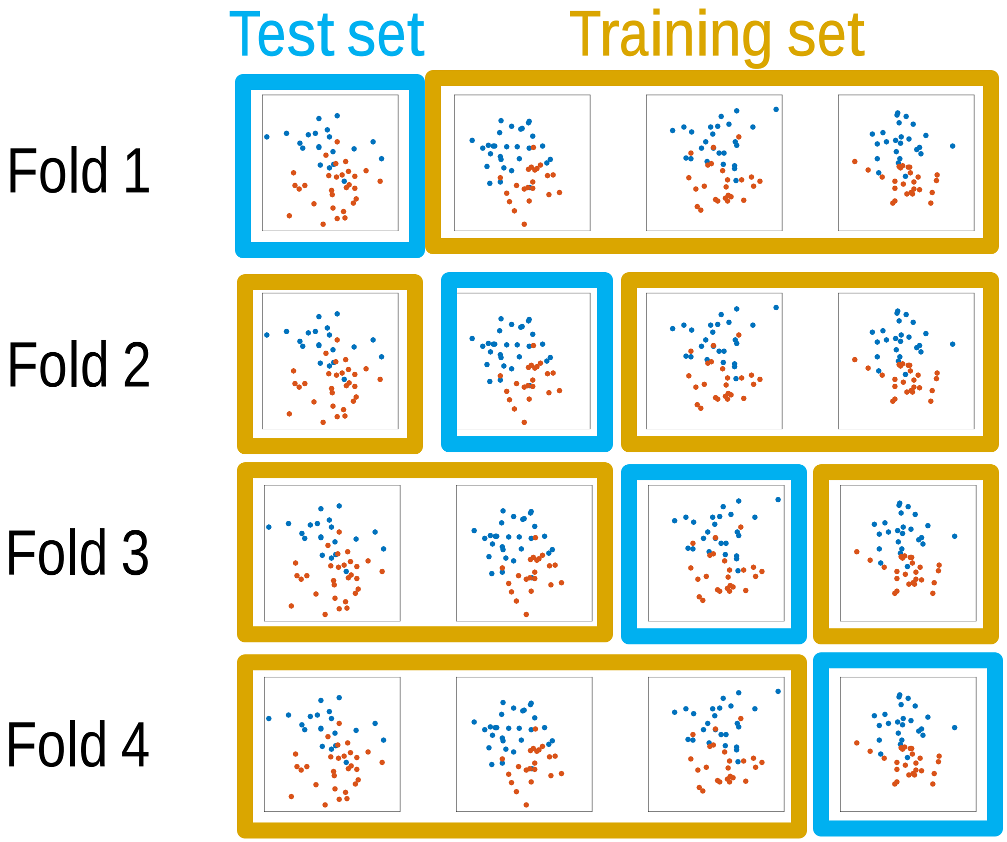
Performance evaluation

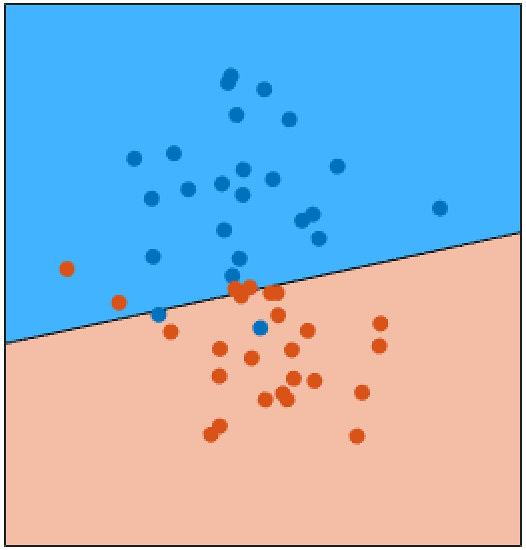
EXAMPLE: 4-fold cross validation (4)



Fold 3: Accuracy = 0.84

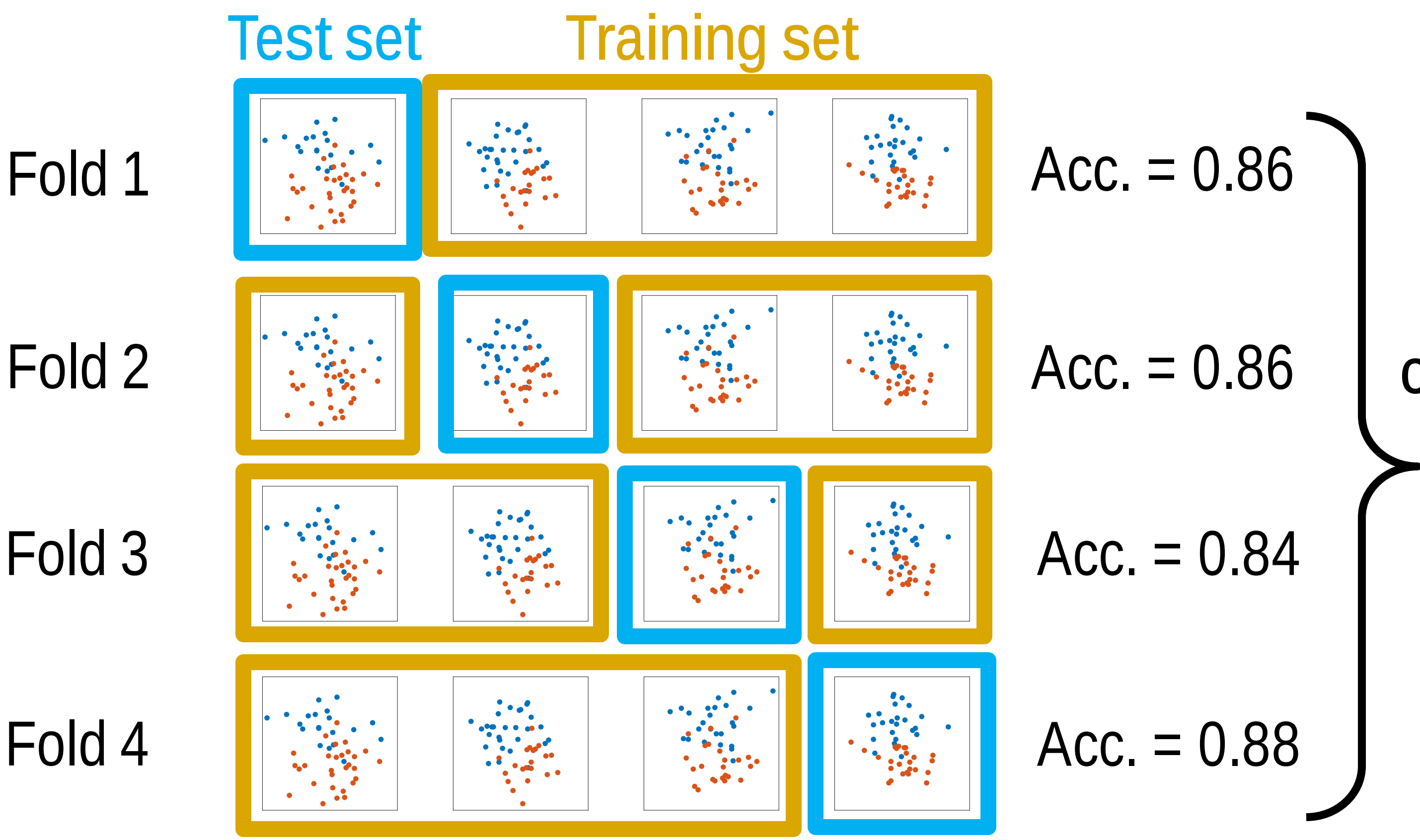
Performance evaluation
EXAMPLE: 4-fold cross validation (5)





Fold 4: Accuracy = 0.88

Performance evaluation
EXAMPLE: 4-fold cross validation (6)

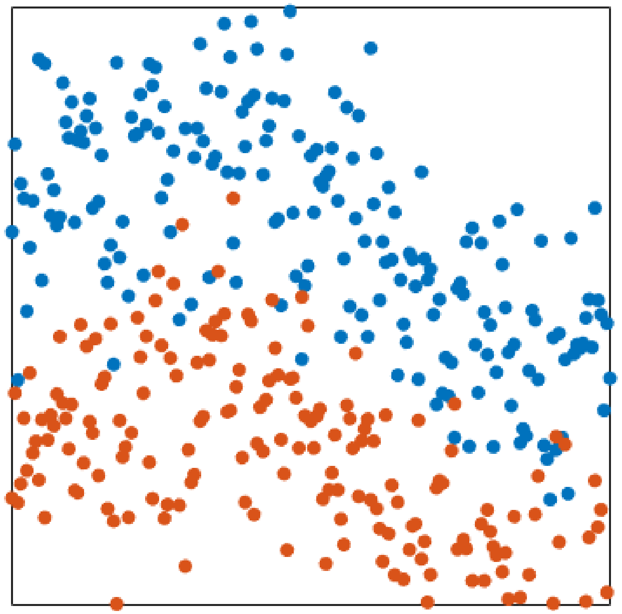


4-fold cross-validation accuracy = 0.86 ± 0.016

(mean ± stdev)

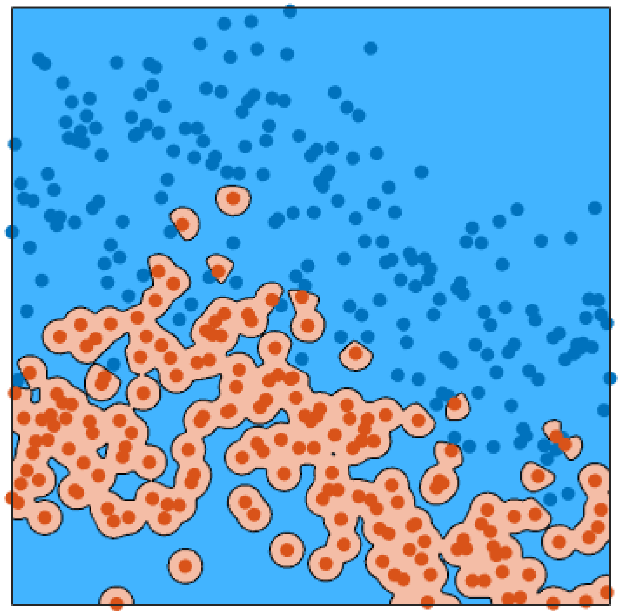
Performance evaluation
Generalization: under- and overfitting

* Why don't we evaluate on the training set?
  – Example:



Performance evaluation
Generalization: under- and overfitting

* Why don’t we evaluate on the training set?
  – Example:



Is this a good classifier?

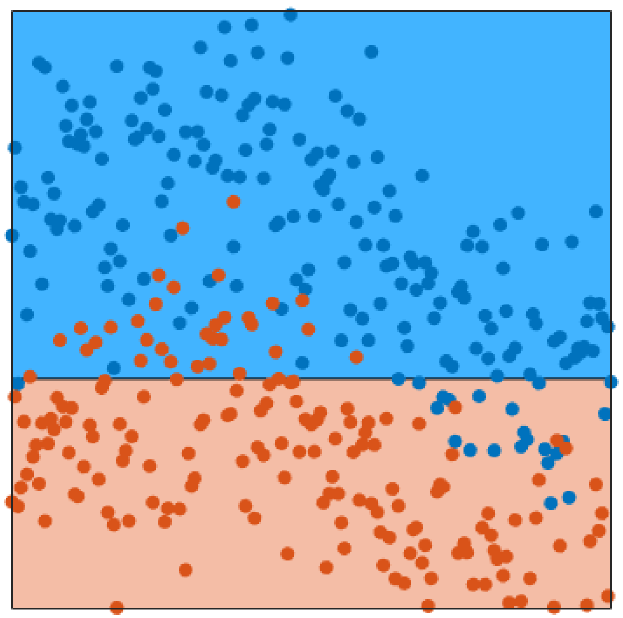
- No errors on the training set!!!
- 100% accuracy

NO!

- Very poor generalization
- On new, identically distributed data:
  - 81% accuracy…
- Overfitting!

Performance evaluation
Generalization: under- and overfitting

* Why don't we evaluate on the training set?
  – Example:



Is this a good classifier?

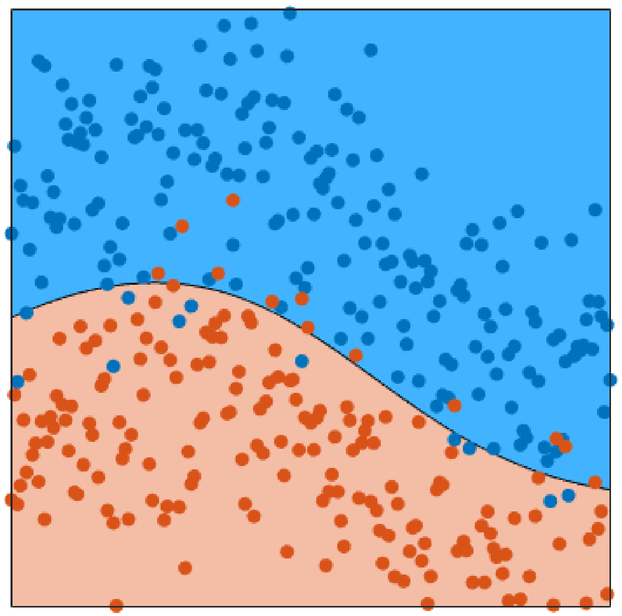
- Many errors on the training set…
- 86% accuracy

NO!

- Model complexity too low!
  - Underfitting!
- On new, identically distributed data:
  - 84% accuracy… (≈ train acc. !)

Performance evaluation
Generalization: under- and overfitting

* Why don't we evaluate on the training set?
  – Example:



Is this a good classifier?

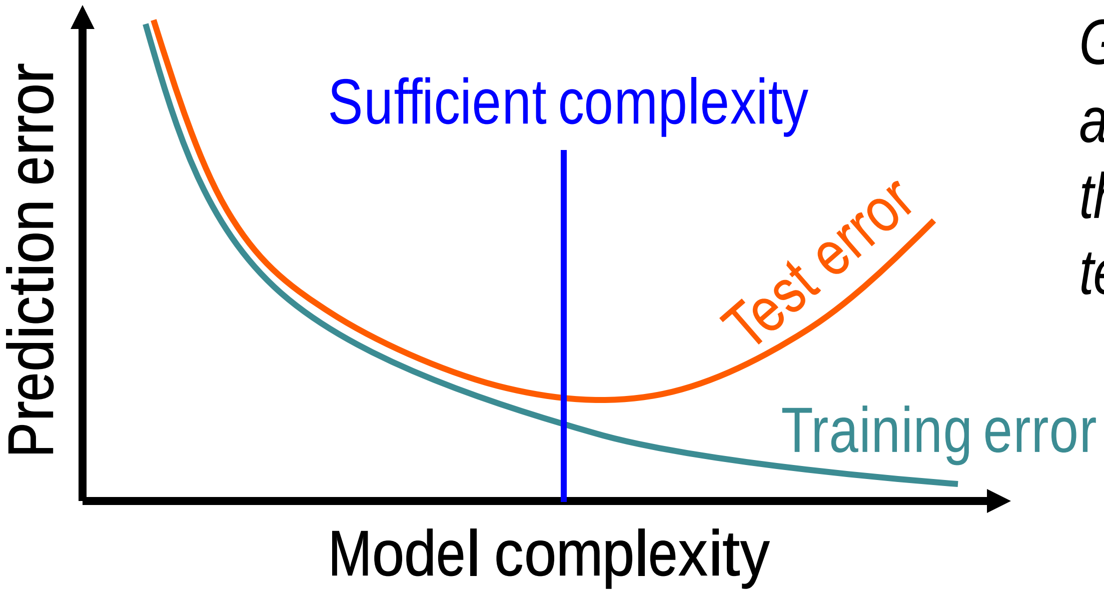
- Accuracy on trianing set: 94%
- Accuracy on test set: 95%
- Approximately equal train and test error
  - Good generalization!

≈

YES!

Performance evaluation
Generalization: under- and overfitting

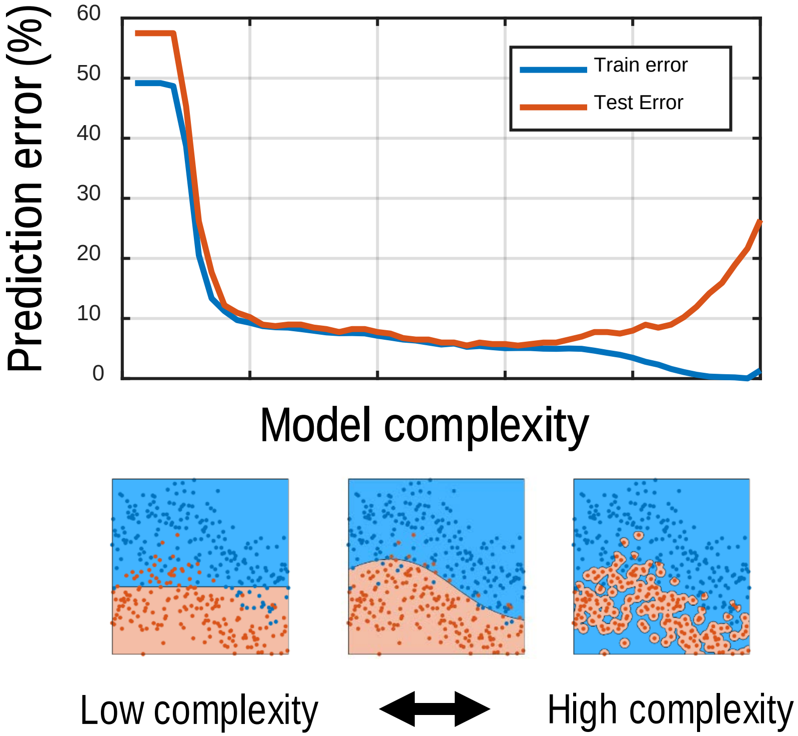
- Model complexity: what is a good model?
  - A model with good generalization!



Good prediciton accuracy on both the training and the test set!

Performance evaluation
Generalization: under- and overfitting

* Model complexity: what is a good model?
  – Example:
    • Non-linear SVM
    • Fixed cost parameter C
    • Complexity increases with reducing the size of the kernel scale (flexibility)
    • 10-fold cross validation to estimate the test error
    • Validate on training set for computing the train error



Performance evaluation

- Summary:
  - In supervised learning the “ground truth” is available, so we can evaluate the prediction performance of the model.
  - Split the data in two sets (training set and test set).
  - Use figures of merit for measuring the performance:
    - *Accuracy, Sensitivity, Specificity, AUC,…*
  - Use K-fold cross-validation for reliable evaluation.
  - Increasing the model complexity may lead to overfitting!
    - *Poor generalization: Low training set error, high test set error.*