EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Machine Learning in Graphics and Vision

- SVMs & Random Forests-

SoSe 2018

Hendrik Lensch

Classification and Regression

- Classification task
  - Given sample x produce class label / category $y(x)$
  - Discrete set of possible classes
- Regression task
  - Given sample x produce function value $y(x)$
  - Continuous space of values, even vectors

Examples

cat

dog

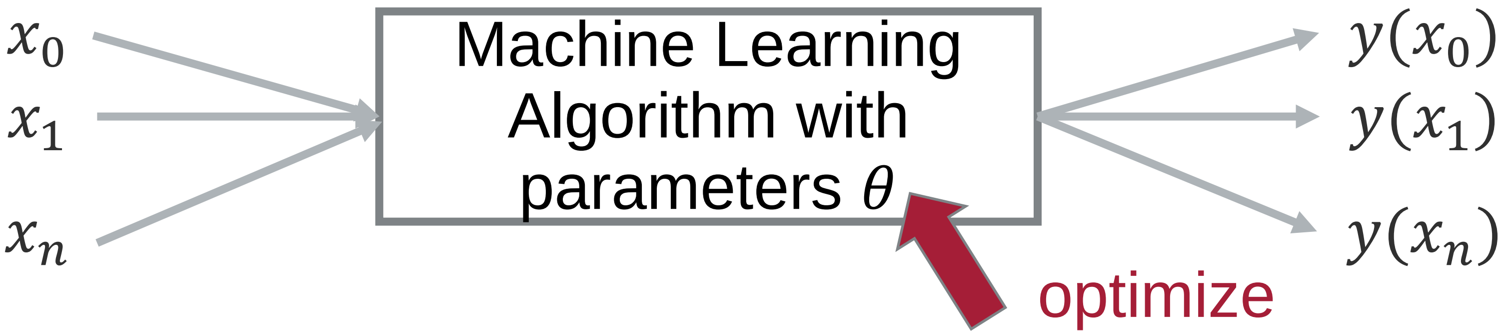
$$\begin{pmatrix} THU \\ ml \end{pmatrix} \longrightarrow true \qquad \begin{pmatrix} 0.3 \\ 1.4 \end{pmatrix} \longrightarrow 42$$

$$\begin{pmatrix} THU \\ cp \end{pmatrix} \longrightarrow false \qquad \begin{pmatrix} 1.1 \\ 0.7 \end{pmatrix} \longrightarrow 3$$

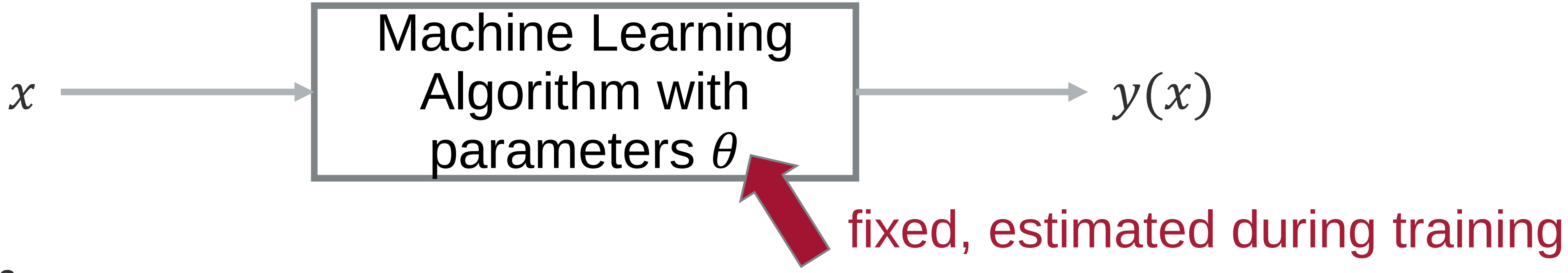
$$\begin{pmatrix} TUE \\ cp \end{pmatrix} \longrightarrow true \qquad \begin{pmatrix} 0.1 \\ 2.3 \end{pmatrix} \longrightarrow 63$$

Machine Learning

- Learning / training
  - Determine parameters θ given some training set



- Prediction
  - Assign class or value to new sample x given trained parameters θ



- Generalization
  - How well does the task perform on the new samples

(Un)Supervised Learning

- Learning is closely related to optimization

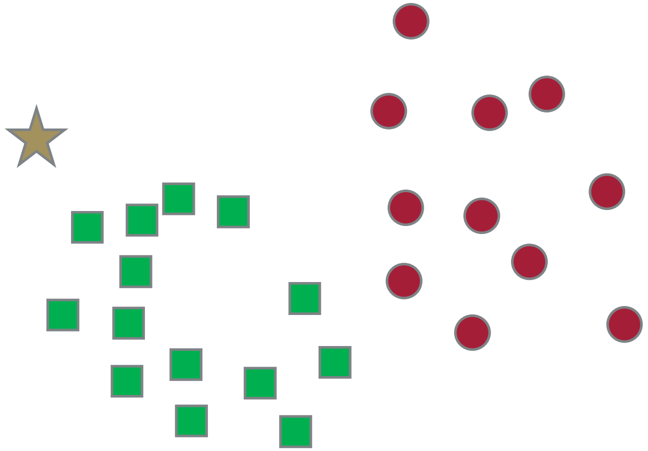
- Supervised Learning
  - Given data points and labels
  - Labels are expensive
  - Training set = { (sample, class) }

- Unsupervised Learning
  - No given labels required
  - Training set = { (samples) }

- Semi-supervised
  - Training set = some labeled samples + many unlabeled samples

Classification

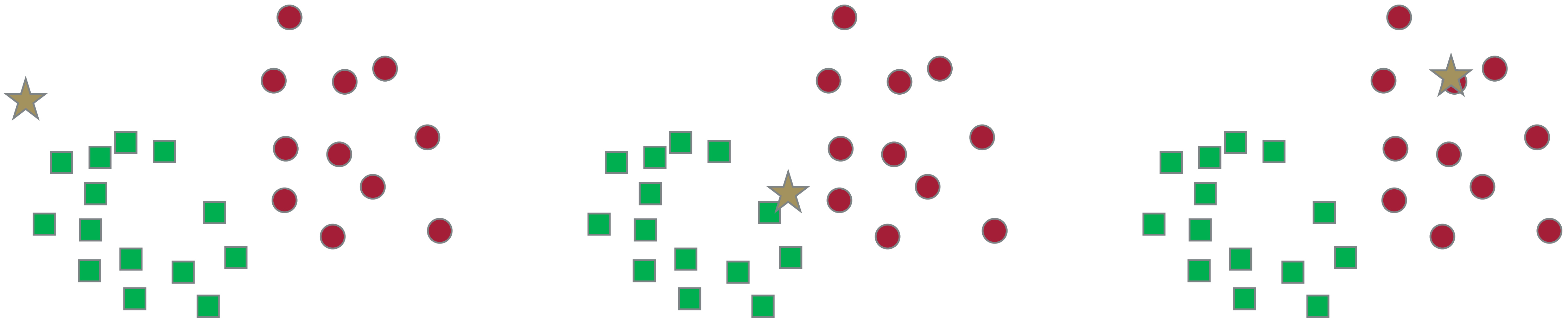
- Distribution of labeled samples



class 1
class 2
query

Classification

- How to determine the class of the query?



$y(\bigstar)$ = class 1

$y(\bigstar)$ = ?

$y(\bigstar)$ = class 2

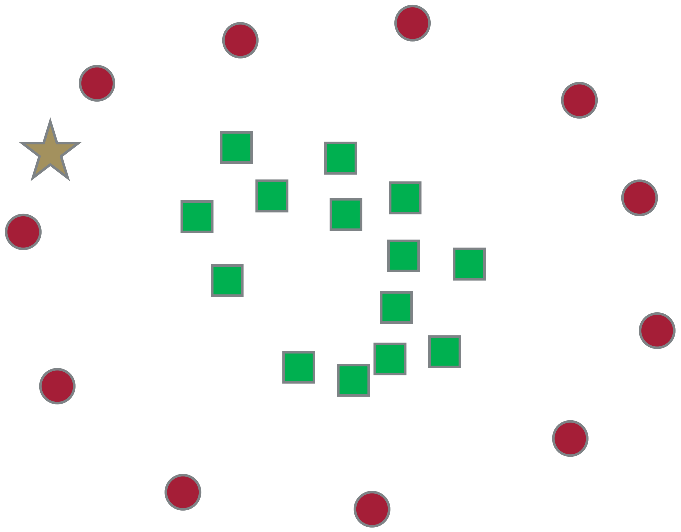
■ class 1

● class 2

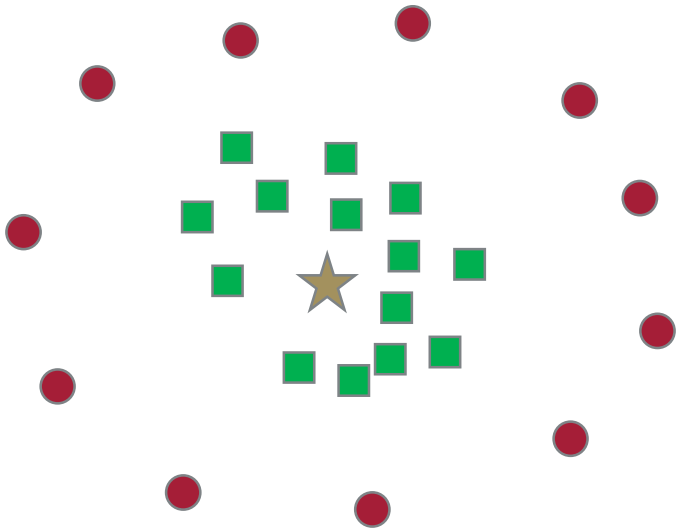
★ query

Classification

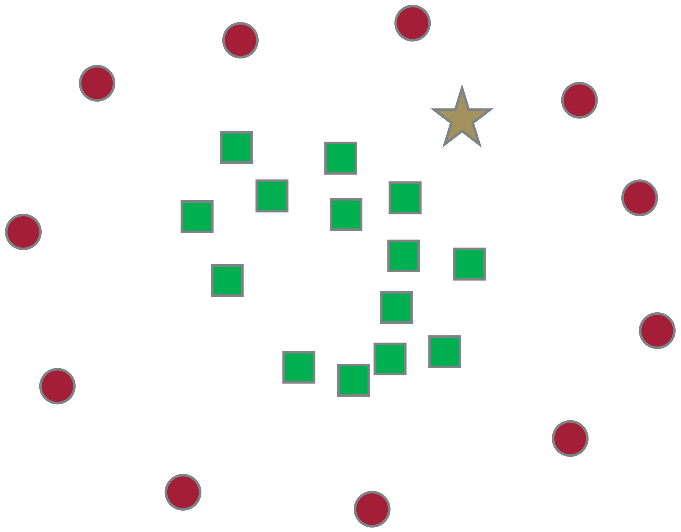
- How to determine the class of the query?



$y($★$)$ = class 2



$y($★$)$ = class 1

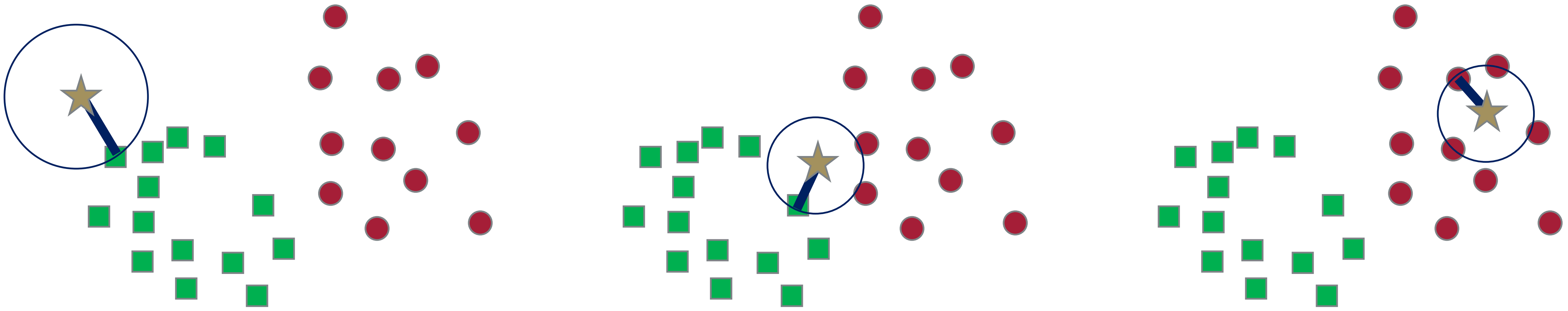


$y($★$)$ = ?

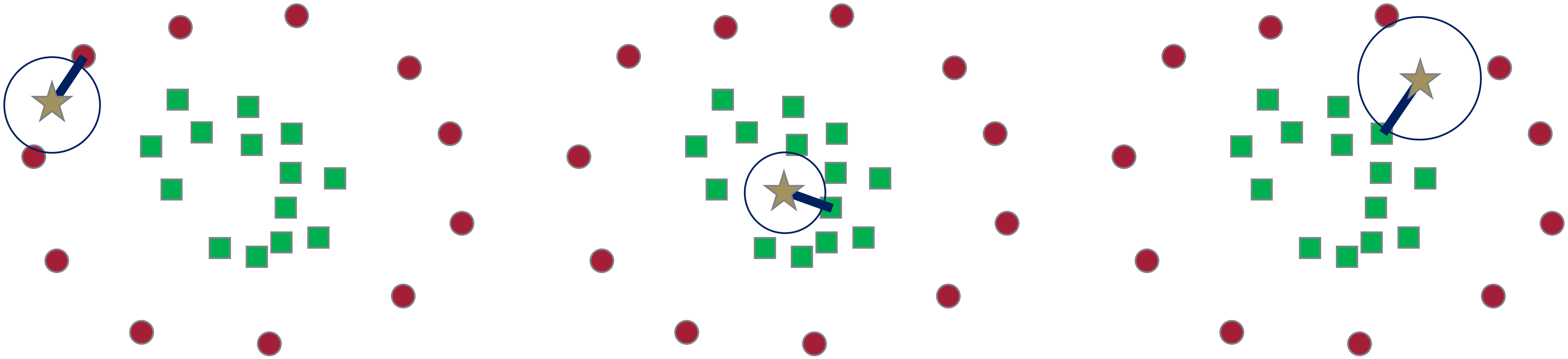
■ class 1
● class 2
★ query

kNN Classification

Nearest Neighbor Classification

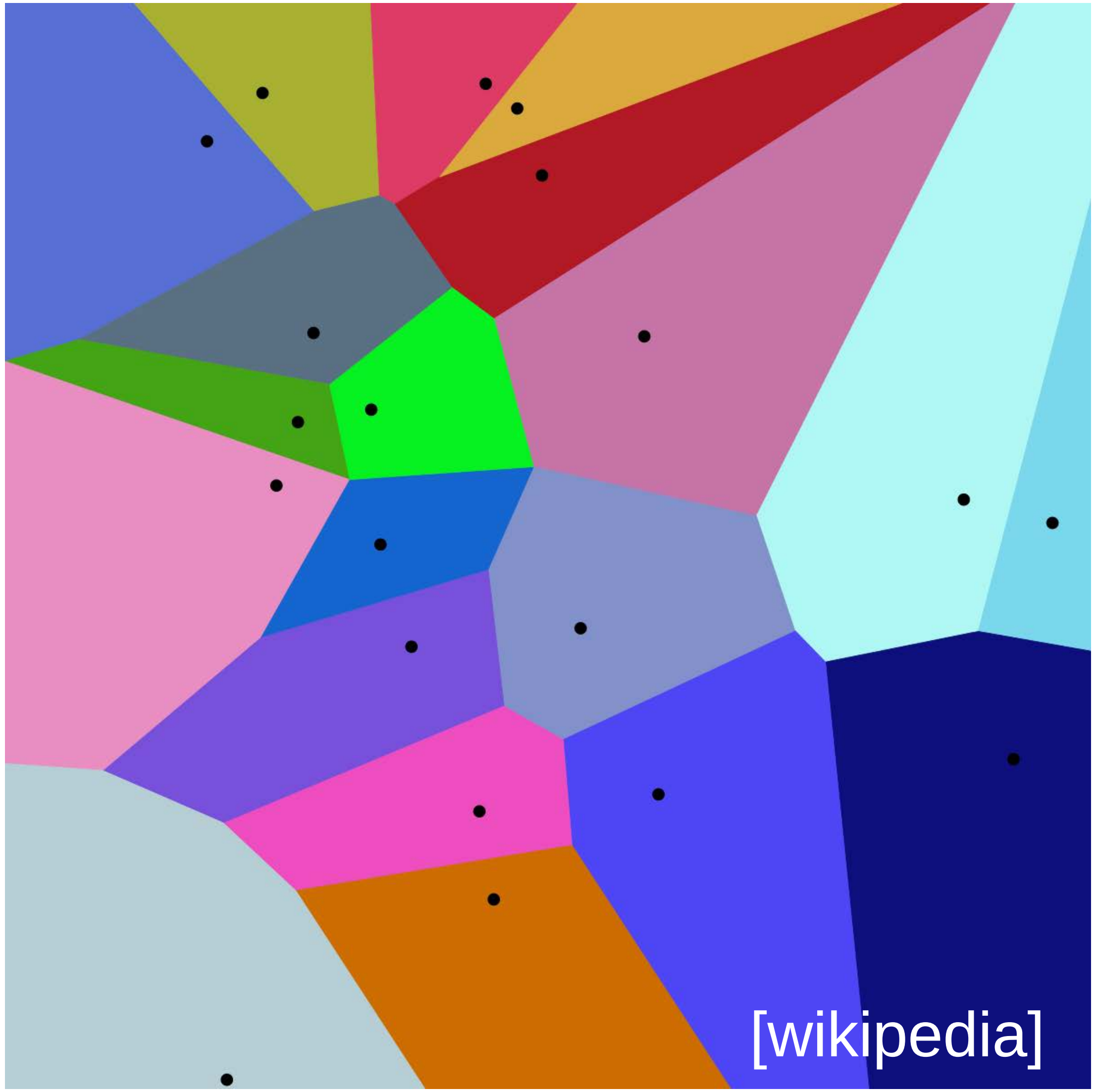


Nearest Neighbor Classification



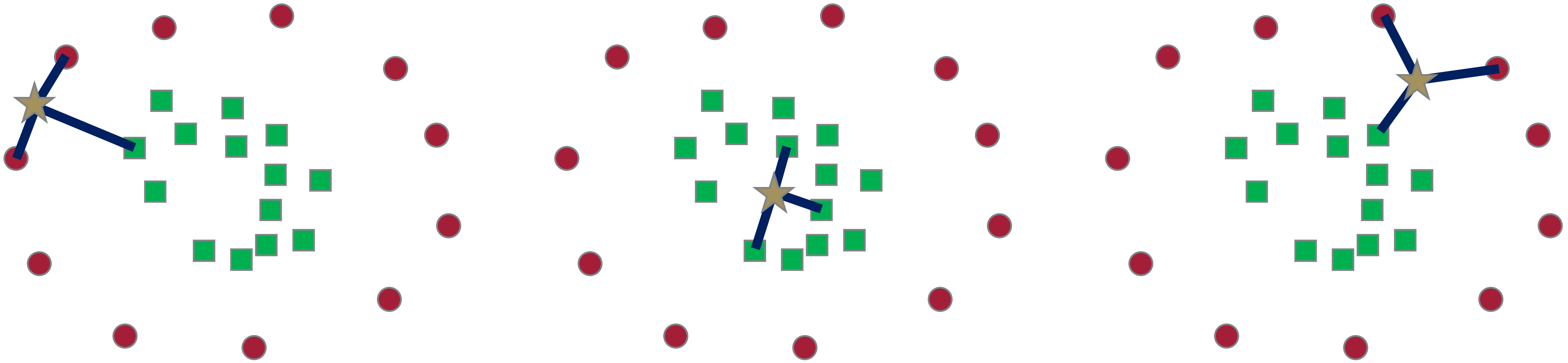
Nearest Neighbor

- NN classification assigns the label of the representative sample to the entire area of ist Voronoi cell



kNN Classification

- More robust classification by considering multiple neighbors?



- Aggregation for classification: e.g. majority voting
- Aggregation for regression

average $y(x) = \frac{1}{k} \sum_{k} y(x_i)$ distance-weighted average $y(x) = \frac{\sum_k w(x, x_i) y(x_i)}{\sum_k w(x, x_i)}$

kNN Classification

- How to chose k?
- Larger k reduces classification noise
- Larger k renders decision boundaries less distinct
- Optimal choice depends on data

Curse of Dimensionality

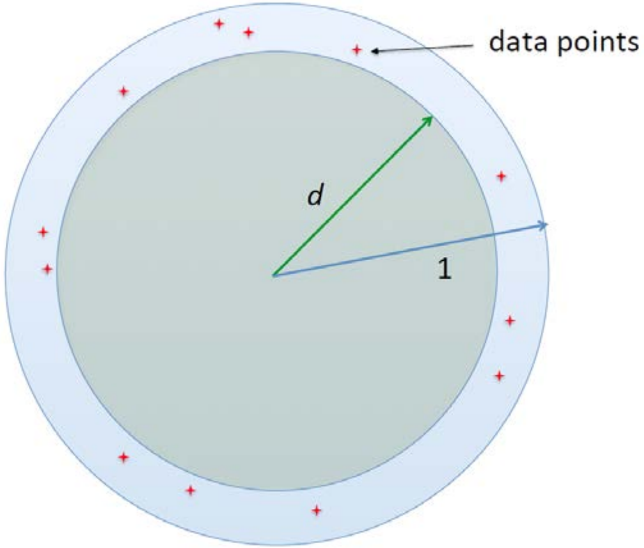
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Relative size of N Voronoi cells grow in $D-$ dimensional feature spaces

Example 1
- Volume of unit hypercube: 1
- Volume of N subcubes / partitioning: $\frac{1}{N}$
- Side length: $d = \left(\frac{1}{N}\right)^{\left(\frac{1}{D}\right)}$ $\qquad \lim_{D \to \infty} d = 1$

Example 2
- uniform sample distribution in unit sphere
- Expected median distance d
- Volume between the sphere of radius 1 and sphere of radius d

$$\frac{1}{2} = \left(\frac{k_n - k_n d^n}{k_n}\right)^N$$



In a highly dimensional space all homogeneously distributed data points seem to be near the shell of the sphere

[image by Raul Rojas]

Support Vector Machines

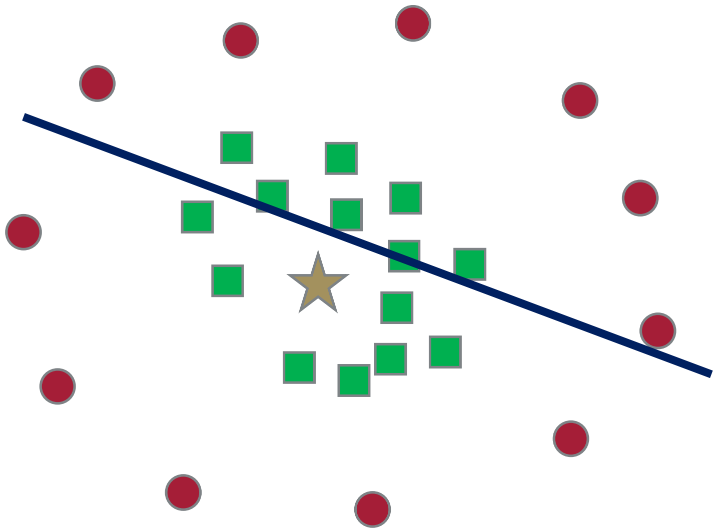
Some slides from

Support Vector Machine & Its Applications by Mingyue Tan

Überwachtes Lernen / Support Vector Machines by Rudolf Kruse

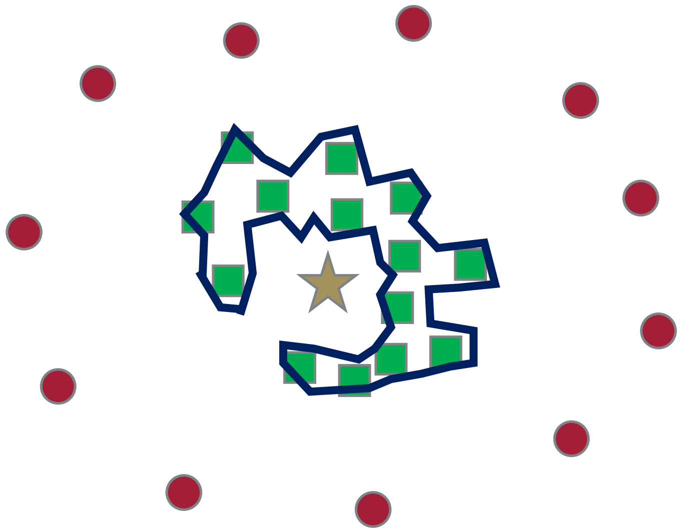
Under / Over Fitting

- Possible decision boundaries



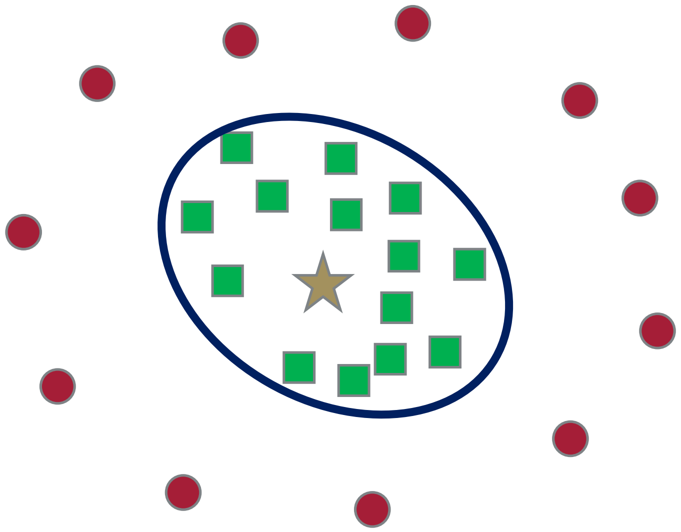
;-(too simple

$y($★$)$ = class 2



;-(too complex

$y($★$)$ = class 2



;-) tradeoff

$y($★$)$ = class 1

■ class 1

● class 2

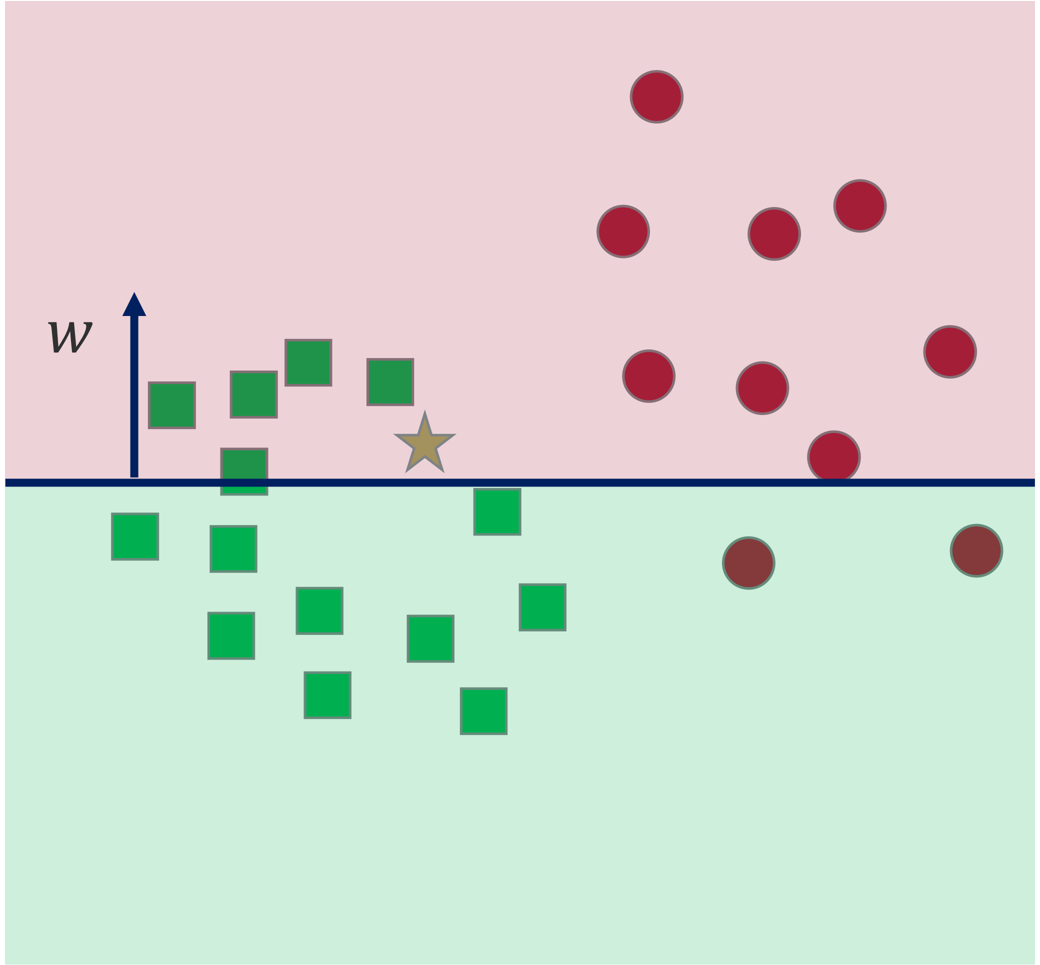
★ query

Linear Classifier

- Classification boundary given by hyperplane

$$\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$$

- Which hyperplane is a good choice?

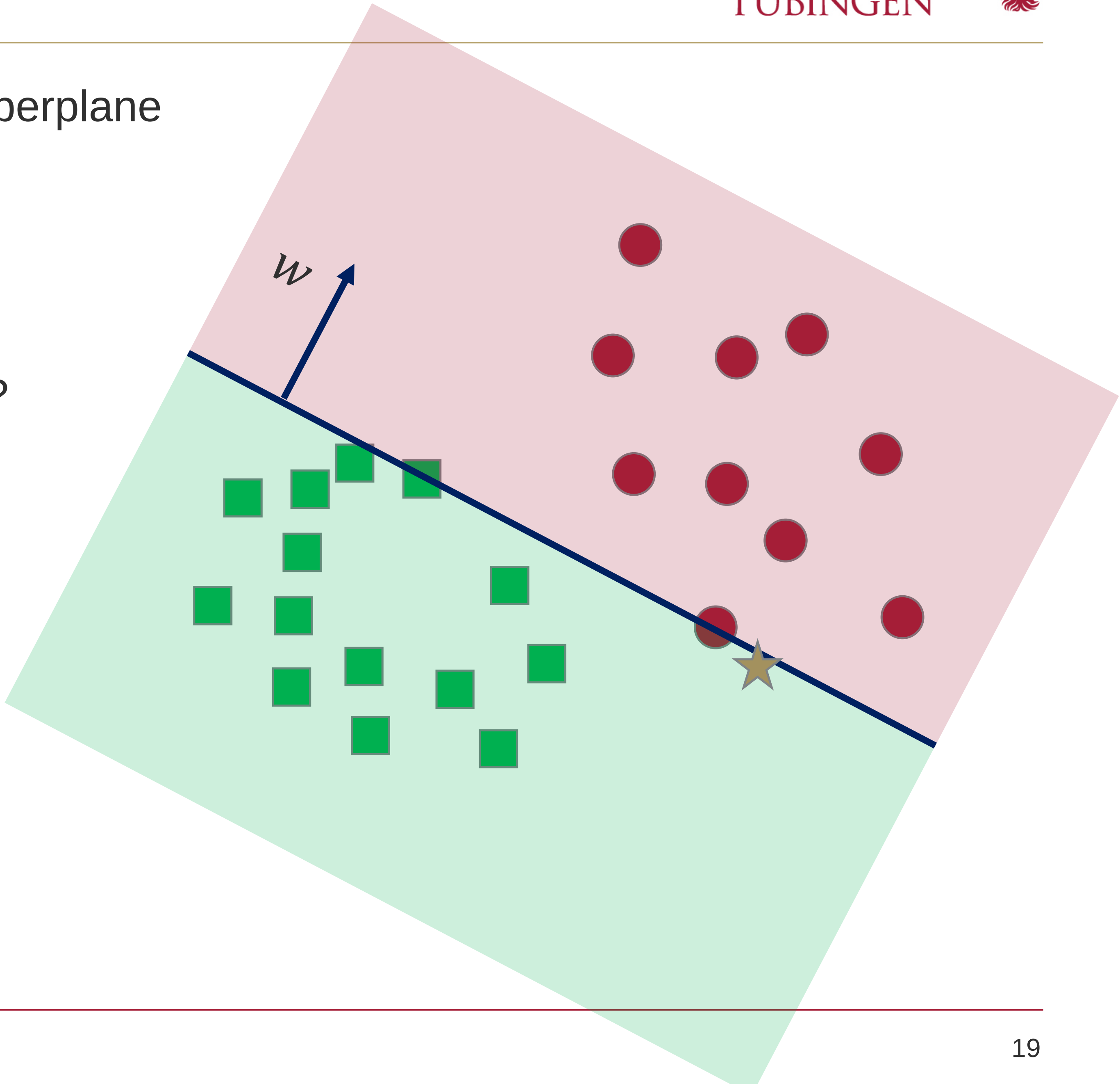


Linear Classifier

- Classification boundary given by hyperplane

$$\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$$

- Which hyperplane is a good choice?

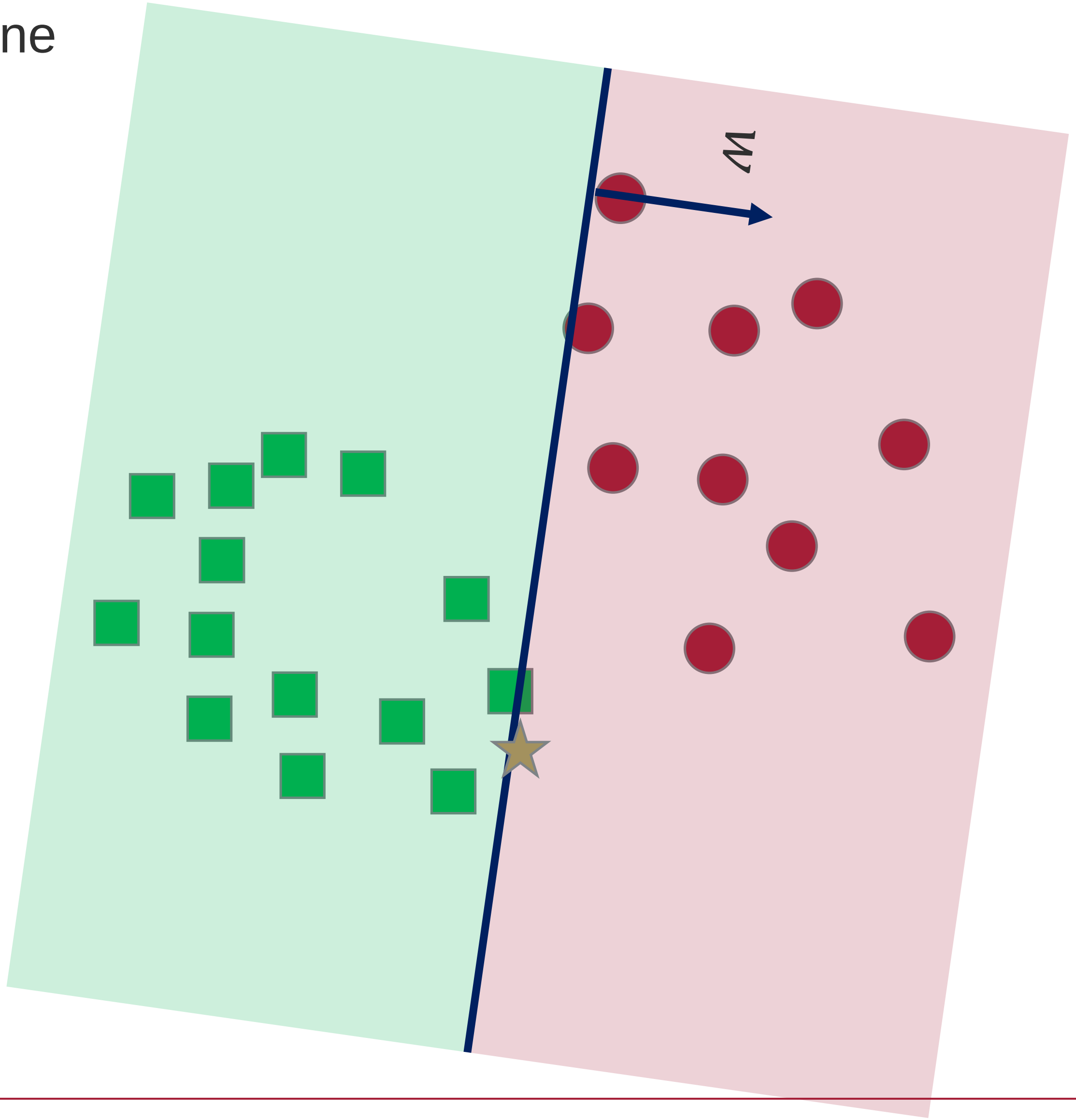


Linear Classifier

- Classification boundary given by hyperplane

$$\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$$

- Which hyperplane is a good choice?



Linear Classifier

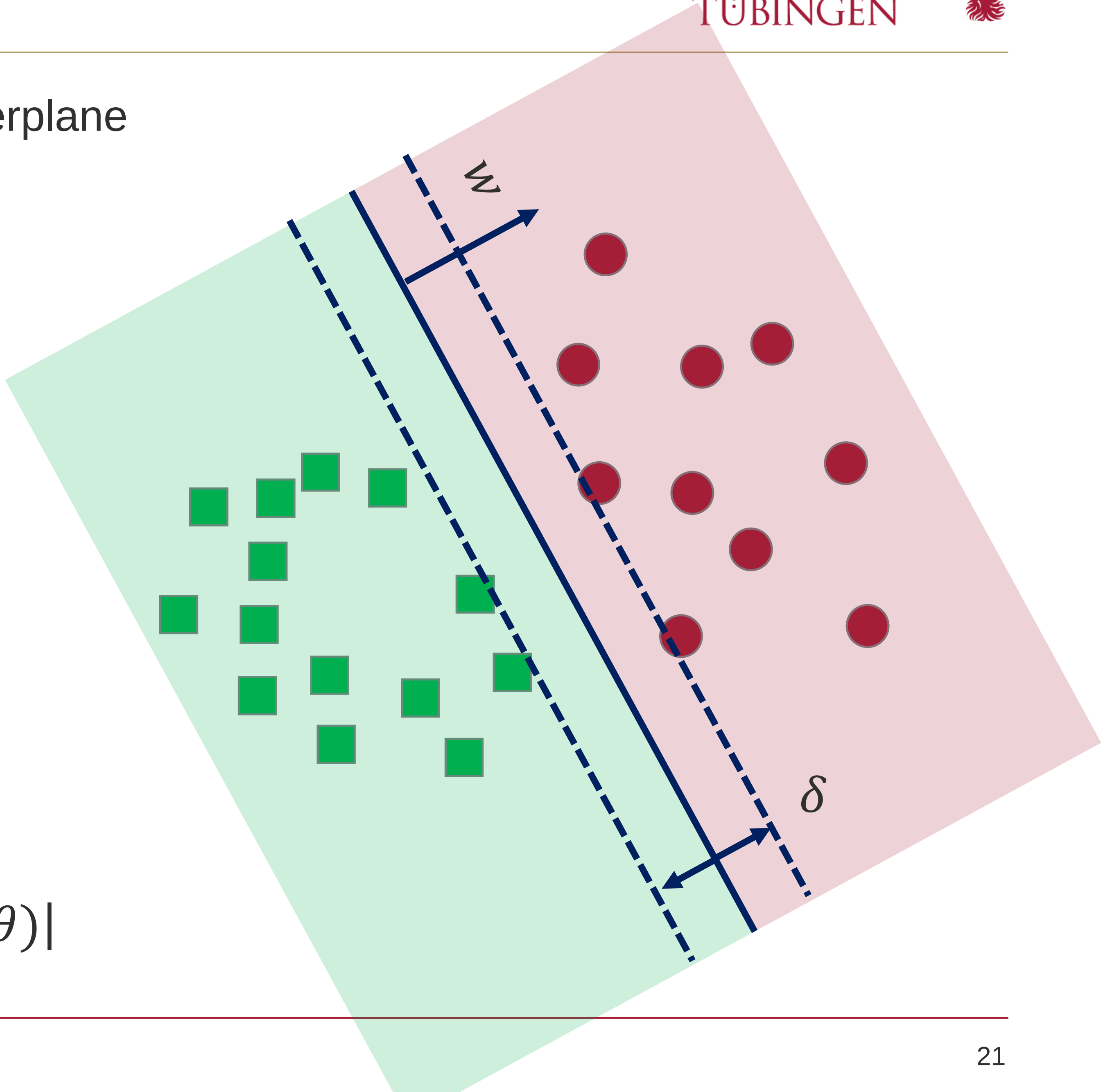
- Classification boundary given by hyperplane

$$\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$$

- Which hyperplane is a good choice?
  - Hyperplane separating classes with maximal margin δ

- Empirical Risk Minimization
  - Optimize parameter, e.g. $\theta = (w, b)$
  - Given training set of N pairs (x_i, x_i)

$$R_{emp}(\theta) = \frac{1}{2N} \sum_{i=1}^{N} |y_i - f(x_i, \theta)|$$



Linear Classifier

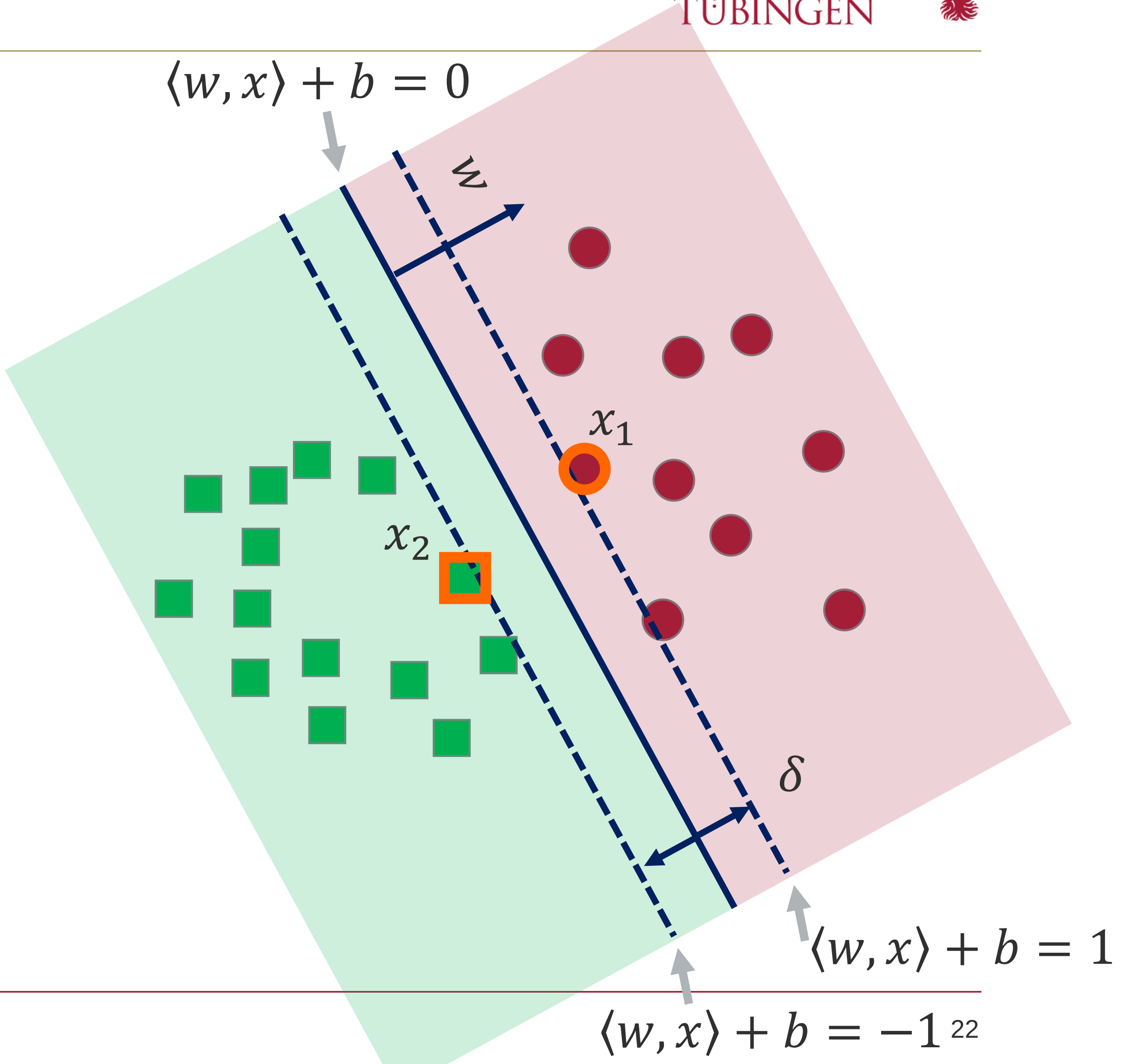
- Binary class label $Y = \{-1, 1\}$
- Correct classification if

$$y(x) = \langle w, x \rangle + b > 0$$

- Scaling pushes margin to 1
- Maximum margin as

$$2\delta = \left\langle \frac{w}{\|w\|}, (x_1 - x_2) \right\rangle = \frac{2}{\|w\|}$$

- Maximize δ $\equiv$ minimze $\|w\|^2$



SVM for linearly separable problems

- Optimal separation by minimizing quadratic cost function with linear side condition

Primary optimization problem

Mimimize $J(w, b) = \frac{1}{2} \|w\|^2$

subject to $\forall i \left[y_i\left(w^T x_i + b\right) \geq 1\right],\ i = 1,2,\ldots,n$ Type equation here.

Introduce Lagrange-function and Lagrange multiply α_i for each sample

$$\mathrm{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum^N \alpha_i \left[y_i\left(w^T w x_i + b\right) - 1\right]; \quad \alpha_i \geq 0$$

leads to dual optimization problem: maximize $\mathrm{L}(w, b, \alpha)$. wrt. α subject to

$$\frac{\mathrm{L}(w, b, \alpha)}{\partial w} = 0 \quad \Rightarrow w = \sum^N \alpha_i y_i x_i \qquad \frac{\mathrm{L}(w, b, \alpha)}{\partial b} = 0 \quad \Rightarrow \sum^N \alpha_i y_i = 0$$

Dual Optimization

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- The solution has the form:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$ $\quad$ $b = y_k - \mathbf{w}^{\mathbf{T}}\mathbf{x_k}$ for any $\mathbf{x_k}$ such that $\alpha_k \neq 0$

- Each non-zero αi indicates that corresponding x_i is a **support vector**.
- Then the classifying function will have the form:

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^{\mathbf{T}}\mathbf{x} + b$

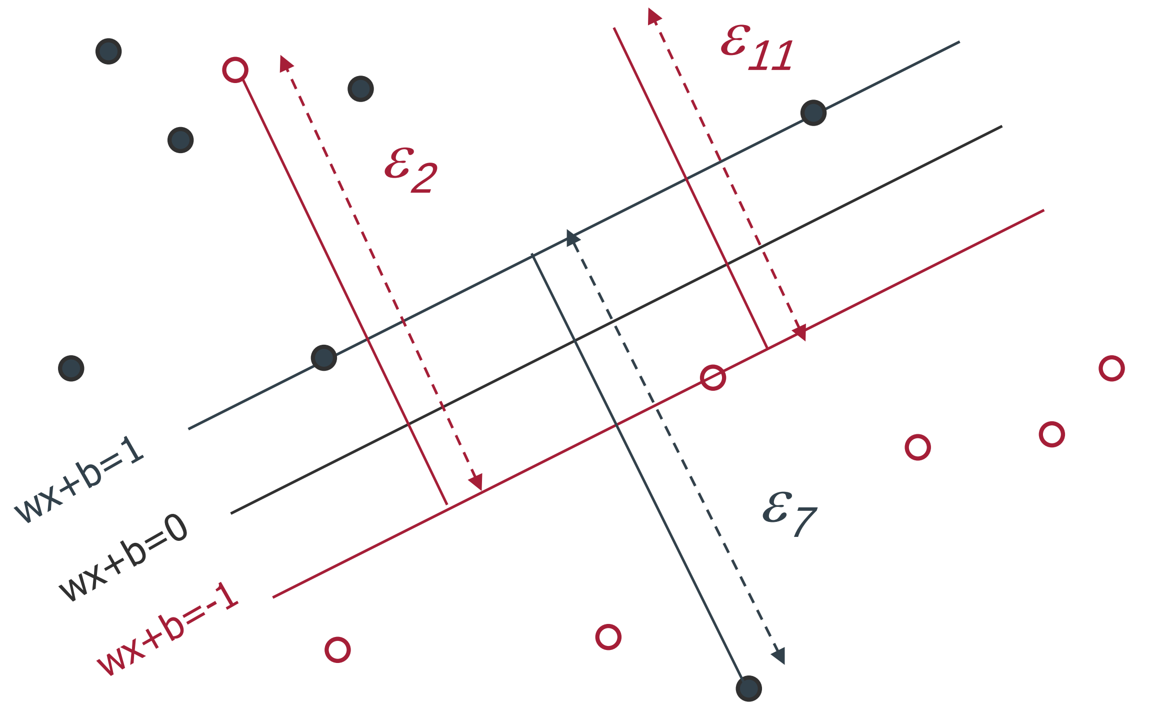
- Notice that it relies on an inner product between the test point x and the support vectors xi – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $x_i^T x_j$ between all pairs of training points.

Soft Margin Classification

- Slack variables ξi can be added to allow misclassification of difficult or noisy examples.

- Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$



[image by Mingyue Tan]

Hard Margin v.s. Soft Margin

- The old formulation:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\, (\mathbf{w^T x_i} + \mathrm{b}) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\, (\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting.

Linear SVMs: Overview

- The classifier is a separating hyperplane.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x}_i^T\mathbf{x}_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x}_i^T\mathbf{x}} + b$$

Multiclass Classification

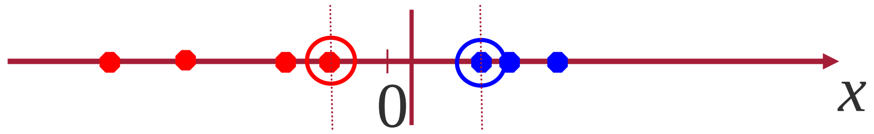
- Binary label $Y = \{-1,1\}$
- Multiclass label $Y = \{1, \ldots, n\}$
- Train SVM for each class individual
- Aggregate classification results

$$c(x) = \arg \max_{1 \le i \le n} (\langle w_i \cdot x \rangle + b_i)$$

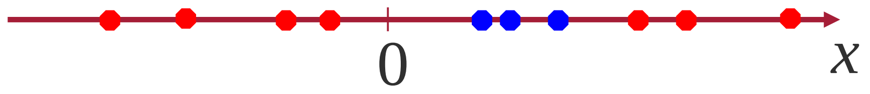
- Geometric interpretation of result: label with the largest distance to hyperplane

Non-linear SVMs

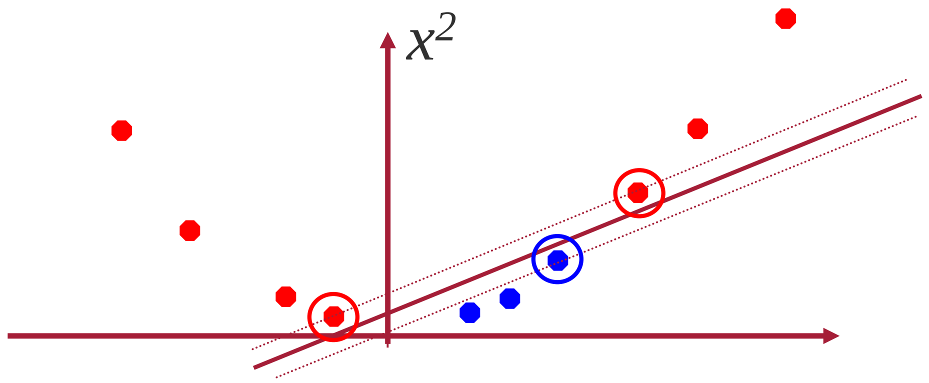
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?



- How about… mapping data to a higher-dimensional space:



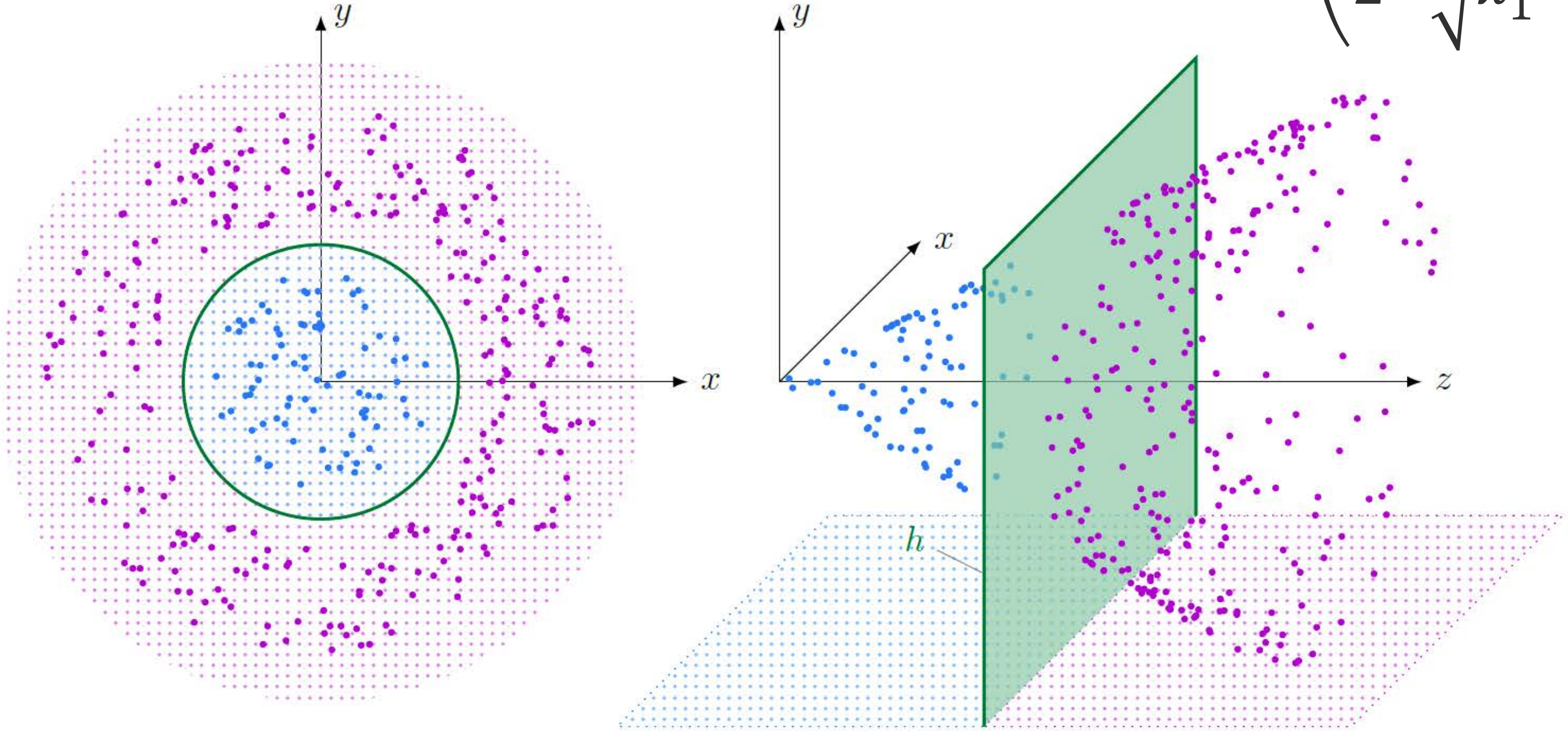
[image by Mingyue Tan]

Non-linear SVM: Transformation of Feature Space

- Embed original features into a higher dimensional space to support simpler classification

- Map $\Phi\colon\ \mathbf{x} \longrightarrow \varphi(\mathbf{x})$

$$\Phi\colon\ \mathbb{R}^2 \longrightarrow \mathbb{R}^3, \qquad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \longrightarrow \begin{pmatrix} x_1 \\ x_2 \\ 2 \cdot \sqrt{x_1^2 + x_2^2} \end{pmatrix}$$



[images by Patrick Wieschollek]

Kernel Trick

- Map samples into an inner product space $\Phi: x \longrightarrow \varphi(x)$
- Usually embedding into space of higher dimensionality than input features
- Allows for simpler classification / regression – typically linear

- Kernel $K(X, Y)$
  - Symmetric
  - Positive semi-definite (Mercer‘s condition):

$$\iint f(x)K(x,y)f(y)dxdy \geq 0$$

  - $K(X, Y) = \langle \varphi(x), \varphi(y) \rangle$
  - Result of inner product is sufficient
  - Mapping needs not to be known (might have implicit representation, e.g. Gaussian kernel)

Kernel - Examples

- Polynomial (homogeneous): $K(X,Y) = (\mathrm{x} \cdot y)^d$
- Polynomial (inhomogeneous): $K(X,Y) = (\mathrm{x} \cdot y + 1)^d$
- Hyperbolic tanget: $K(X,Y) = \tanh(\alpha \mathrm{x} \cdot y + \beta)$
- Gaussian: $K(X,Y) = \exp(-\frac{1}{\sigma^2}|x - y|^2)$

- Function of the distance between samples

Non-linear SVMs Mathematically

- Dual problem formulation:

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- The solution is:

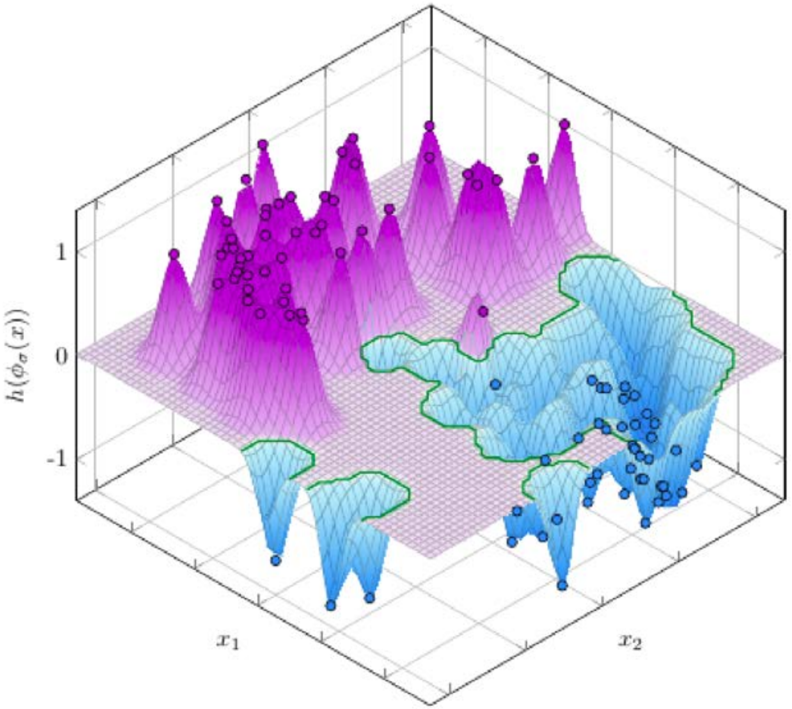
$$f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x}_i, \mathrm{x}_j) + b$$

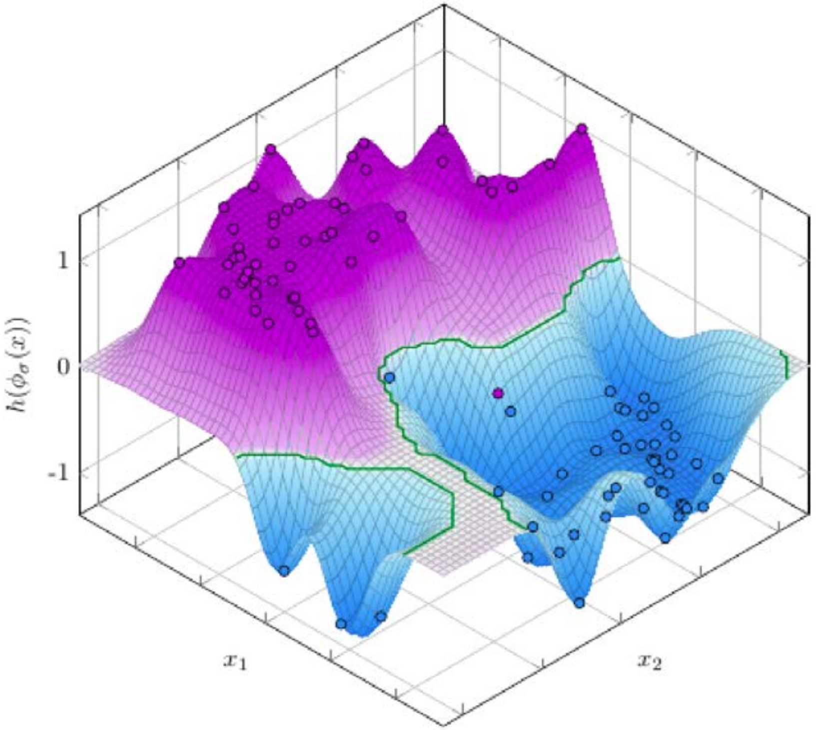
- Optimization techniques for finding αi's remain the same!

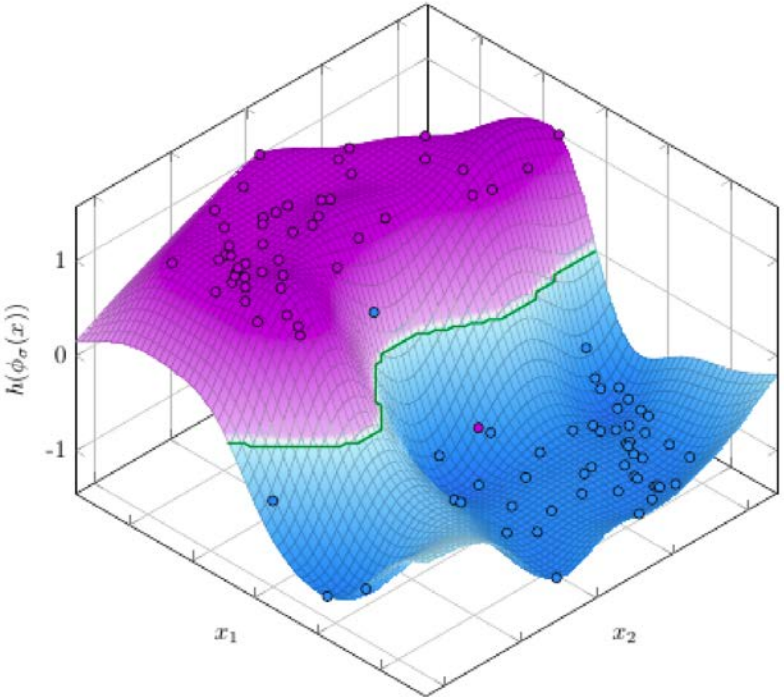
Non-linear SVM - Properties

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Classification Landscape







Gaussian kernel $\sigma = 0{,}1$
overfitting

Gaussian kernel $\sigma = 0{,}2$
good

Gaussian kernel $\sigma = 0{,}4$
underfitting

[images by Patrick Wieschollek]

SVM Summary

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection

Software

- List of SVM implementations
  - http://www.kernel-machines.org/software

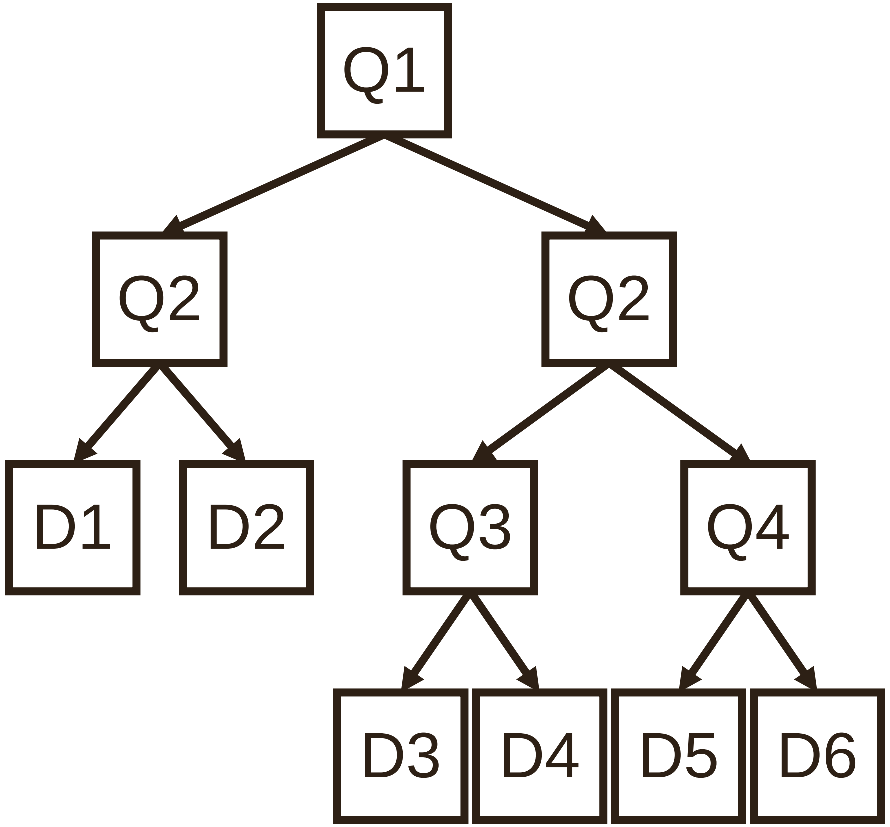
- libSVM at
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Random Forest

Decision Tree

- Inner nodes
  - A condition evaluated for each data point
  - One child per possible answer
  - Often only binary decisions

- Leaf nodes
  - Corresponding to some decision reached / predicted label



CART – Classification and Regression Tree

- Supervised training with labeled sample set
  - Classification or
  - regression

- Binary decision tree
  - Top down, greedy build
  - Inner nodes: recursive partitioning of the feature space into hyper-rectangles
  - Compare kD-tree

Building a CART

- All labeled samples are assigned to root node
- Try to find a split condition that separates the sample set into two „more meaningful“ subsets
- Recurse

Building a CART

- Assing samples S to root node N

- With node N do
  - Find feature F + threshold T
    - Split samples S assigned to N into 2 subsets S_{left} and S_{right}; $S = S_{left} \cup S_{right}$
    - Split should increase purity within subsets

  - If S_{left} or S_{right} are too small to be split again
    - Create leaf node
    - Assign label which is most present in S_{left} or S_{right}, resp.

  - Assign S_{left} to N_{left} and S_{right} to N_{right}
    - Recurse with N_{left} and N_{right}

Measure of (Im)Purity

- Quality measure for the purity of the labels in a given sample set
- p_l is the proportion of examples in S that belong to class l

Examples:

- Gini index $$G(S) = \sum_{l}^{L} p_l(1 - p_l)$$
- Entropy $$E(S) = -\sum_{l}^{L} p_l \log p_l$$
- Missclassification error $$E(S) = 1 - \max_{l \in L} p_l$$

Classification Properties

	CART	kNN	SVM
Intrinsically multiclass	🟢	🟢	🟡
Handles apple and oranges well	🟢	🔴	🔴
Robustness to outliers	🟢	🟢	🟡
Works w/ „small“ learning set	🔴	🔴	🟢
Large learning set	🟢	🔴	🔴
Prediction accuracy	🔴	🟡	🟢
Parameter tuning	🟢	🟡	🔴

Random Forest

Definition
- Collection of unpruned CARTs
- Requires rule to combine indivdual tree decisions

Goal
- Improve prediction accuracy

Principle
- Encourage diversity amoung trees

Solution: Randomness
- Bagging
- Random decsion trees (rCART)

Bagging

- Bagging: Bootstrap aggregation

- Technique of ensemble learning
  - To avoid over-fitting
  - To improve stability and accuracy

- Two steps:
  - Bootstrap sample set
  - Aggregation

Bagging - Bootstrap

- L: original learning set composed of n samples

- Generate k learning sets of L_k
  - Composed of q samples, $q \leq n$
  - Obtained by uniform sampling with replacement from L
  - In consequence, L_k may contain repeated samples

- Random Forest: often $\boldsymbol{q = n}$
  - Approx. 63% unique samples for $k = 100$
  - The remaining samples can be used for testing

Bagging - Aggregation

- Learning
  - For each L_k train one classifier (rCART) C_k

- Prediction
  - For sample x
  - Compute all results $C_k(x)$
  - Aggregate:
    - Classification: majority vote on $C_k(x)$
    - Regression: average over $C_k(x)$

Building Random CART

- **Random subset of features**
  - Random drawing repeated at each node (e.g. $\sqrt{D}$ possible dimension out of D)
  - Increase diversity amoung the rCARTs + reduce computational load

- Assing samples S to root node N
- With node N do
  - Find feature F **among random subset of features** + threshold T
    - Split samples S assigned to N into 2 subsets S_{left} and S_{right}; $S = S_{left} \cup S_{right}$
    - Split should increase purity within subsets
  - If S_{left} or S_{right} are too small to be split again
    - Create leaf node
    - Assign label which is most present in S_{left} or S_{right}, resp.
  - Assign S_{left} to N_{left} and S_{right} to N_{right}
    - Recurse with N_{left} and N_{right}

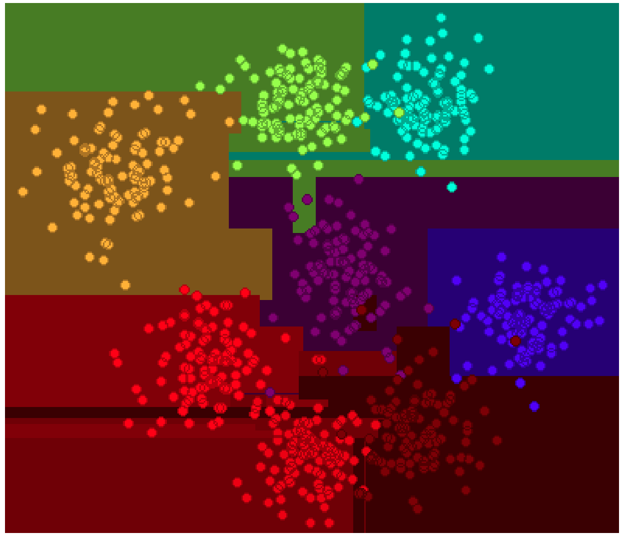
Classification Properties

	RF	CART	kNN	SVM
Intrinsically multiclass	🟢	🟢	🟢	🟡
Handles apple and oranges well	🟢	🟢	🔴	🔴
Robustness to outliers	🟢	🟢	🟢	🟡
Works w/ „small“ learning set	🔴	🔴	🔴	🟢
Large learning set	🟢	🟢	🔴	🔴
Prediction accuracy	🟢	🔴	🟡	🟢
Parameter tuning	🟢	🟢	🟡	🔴

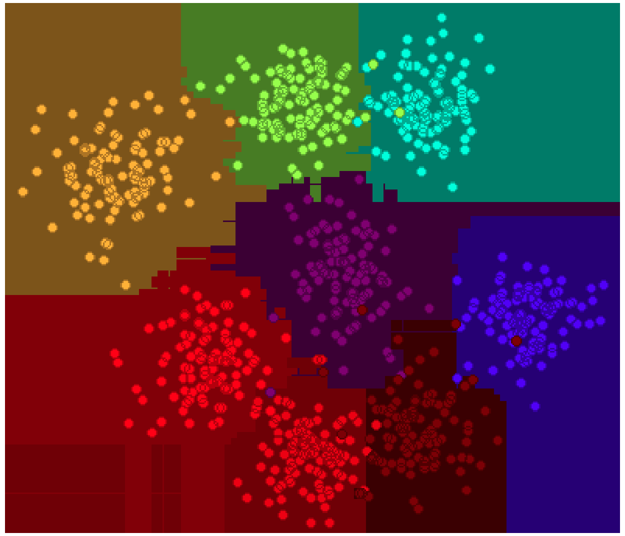
Video

- link

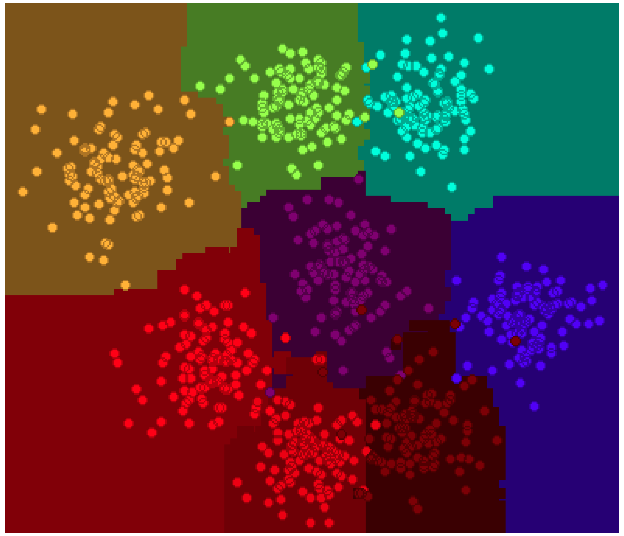
Random Forest - Example



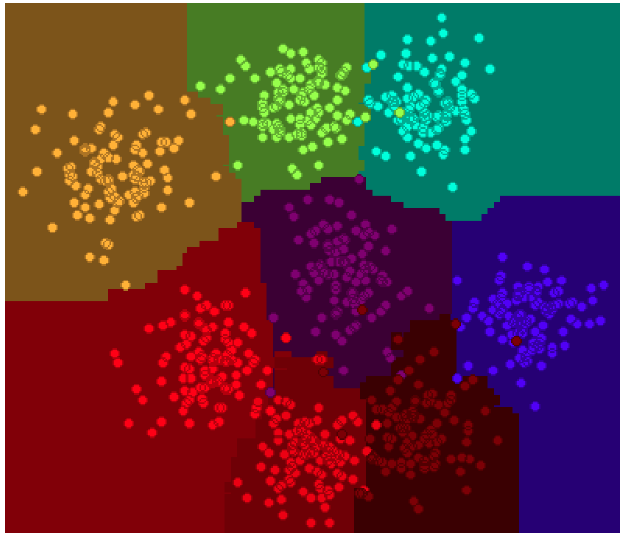
1 rCART



10 rCARTs



100 rCARTs

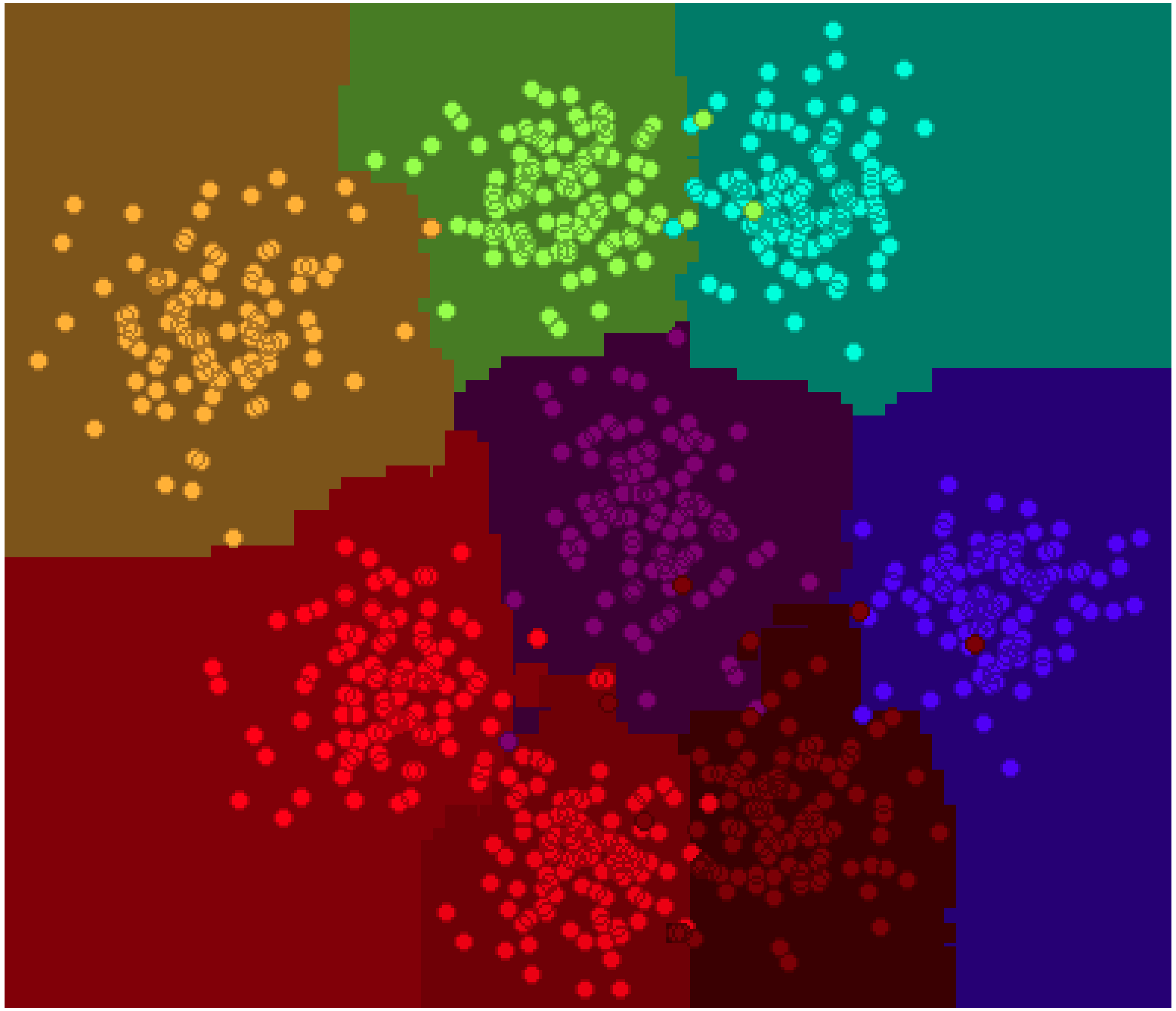


1000 rCARTs

[images by Eric Debreuve]

Random Forest - Limitation

- Oblique / curved decision boundaries
  - Staircase effect
  - Involves many orthogonal hyperplanes
- Fundamentally discreet
  - Classification of functional data?



[image by Eric Debreuve]

Kernel Trick

- Map samples into an inner product space $\Phi: x \longrightarrow \varphi(x)$
- Usually embedding into space of higher dimensionality than input features
- Allows for simpler classification / regression – typically linear

- Kernel $K(X, Y)$
  - Symmetric
  - Positive semi-definite (Mercer‘s condition):

$$\iint f(x)K(x,y)f(y)dxdy \geq 0$$

  - $K(X, Y) = \langle \varphi(x), \varphi(y) \rangle$
  - Result of inner product is sufficient
  - Mapping needs not to be known (might have implicit representation, e.g. Gaussian kernel)

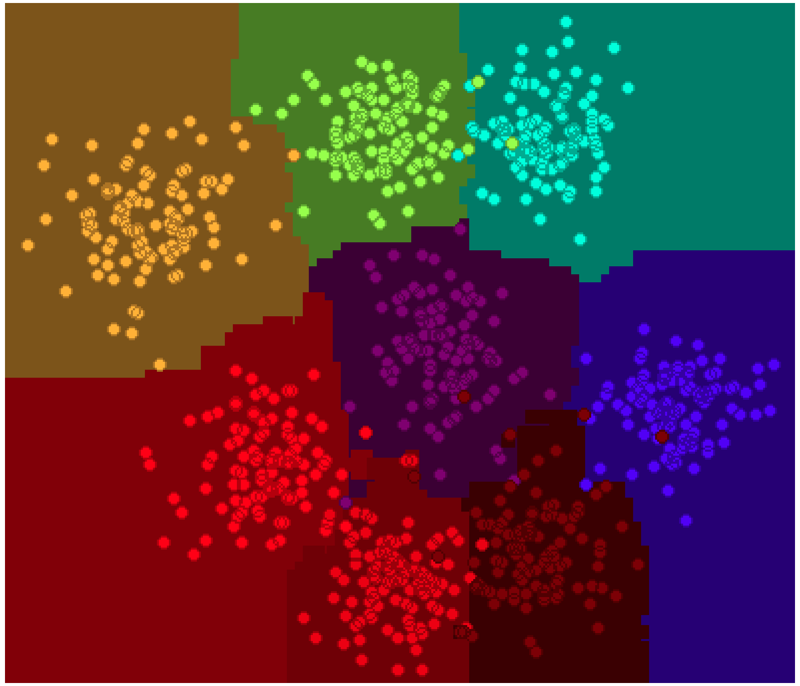
Kernel - Examples

- Polynomial (homogeneous): $K(X, Y) = (\mathrm{x} \cdot y)^d$
- Polynomial (inhomogeneous): $K(X, Y) = (\mathrm{x} \cdot y + 1)^d$
- Hyperbolic tanget: $K(X, Y) = \tanh(\alpha \mathrm{x} \cdot y + \beta)$
- Gaussian: $K(X, Y) = \exp(-\gamma |x - y|^2)$

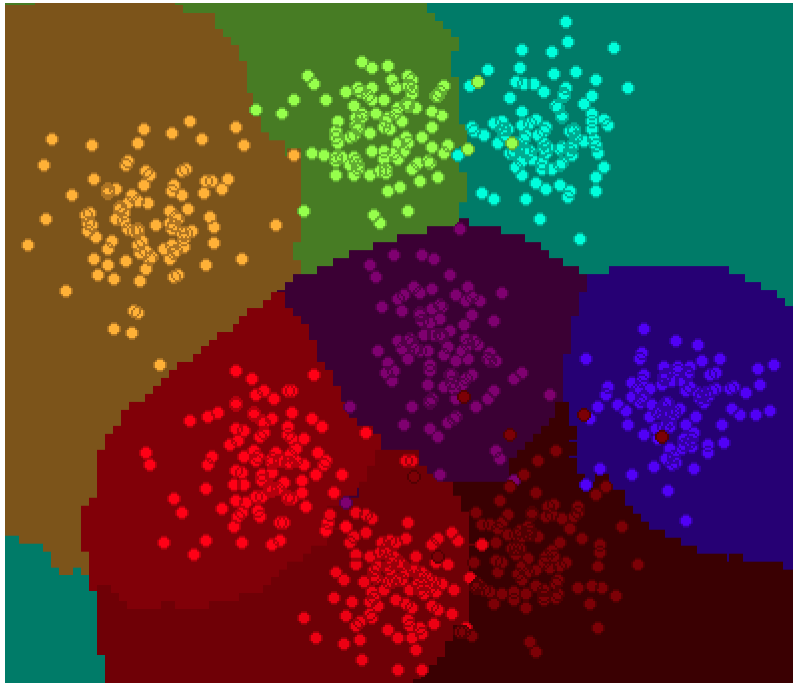
- Function of the distance between samples

Kernel-Induced Random Forest - Examples

- Using a Gaussian kernel
  - Decision boundaries typically get simpler



RF w/ 100 rCARTs



Kernel-induced
RF w/ 100 rCARTs

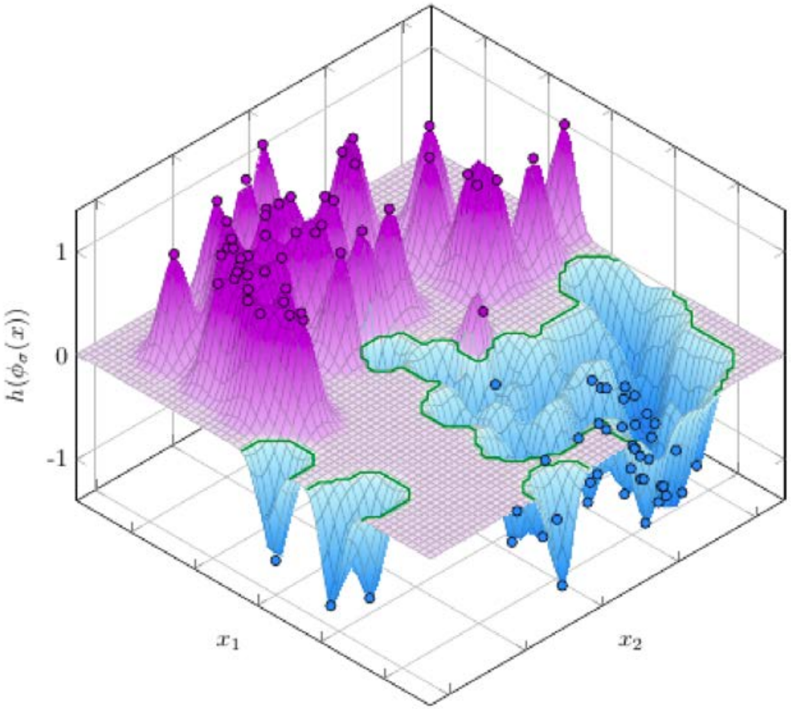
[images by Eric Debreuve]

Generalization

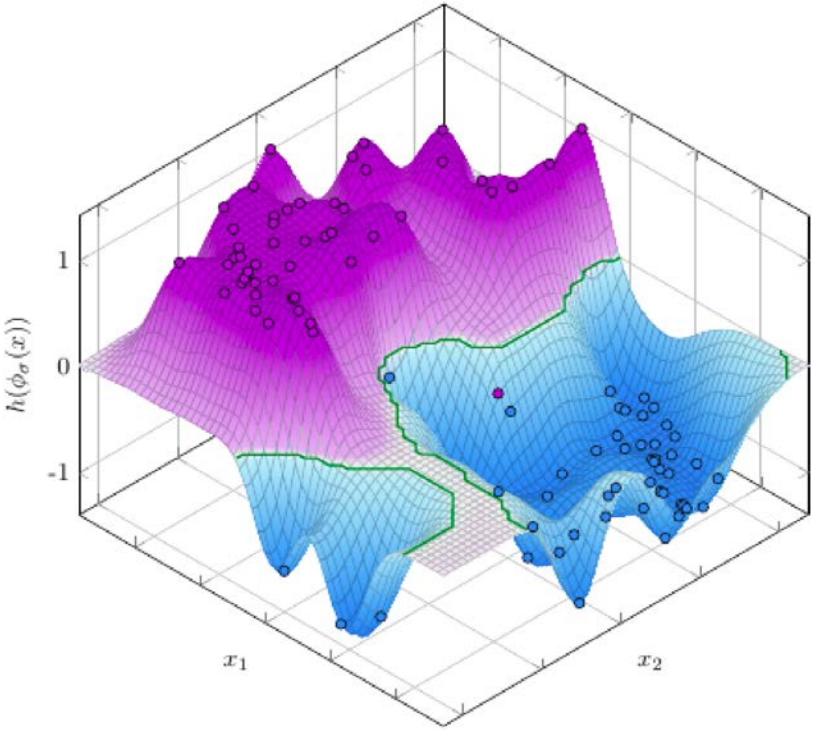
How to measure classification performance?

Generalization

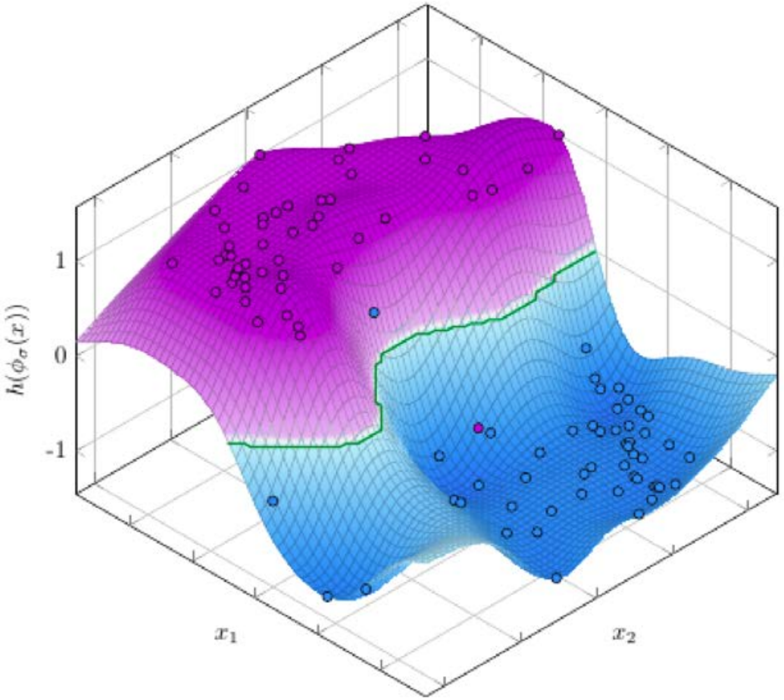
- How well does the classifier perfom on new data?



Gaussian kernel $\sigma = 0{,}1$
overfitting
Error on training set = zero



Gaussian kernel $\sigma = 0{,}2$
good
Error on training set = low



Gaussian kernel $\sigma = 0{,}4$
underfitting
Error on training set = higher

[images by Patrick Wieschollek]

Measure generalization performance

Need to provide

- **Training data** set
- **Testing data** set
- Non-overlapping

- E.g. for random forests:
  - Training data set: drawn with replication - covers approximately 63% of samples
  - Testing data set: rest (out-of-bag) as test samples

TP, TN, FP, FN

- TP – True Positive – the number of observations correctly assigned to the positive class
- TN – True Negative – the number of observations correctly assigned to the negative class
- FP – False Positive – the number of observations assigned by the model to the positive class, which in reality belong to the negative class
- FN – False Negative – the number of observations assigned by the model to the negative class, which in reality belong to the positive class

predicted → real ↓	*Class_pos*	*Class_neg*
Class_pos	TP	FN
Class_neg	FP	TN

Precision, Recall, Accuracy

- $precision = \frac{TP}{TP+FP}$
- $recall = \frac{TP}{TP+FN}$
- $true\ negative\ rate = \frac{Tn}{TN+FP}$
- $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

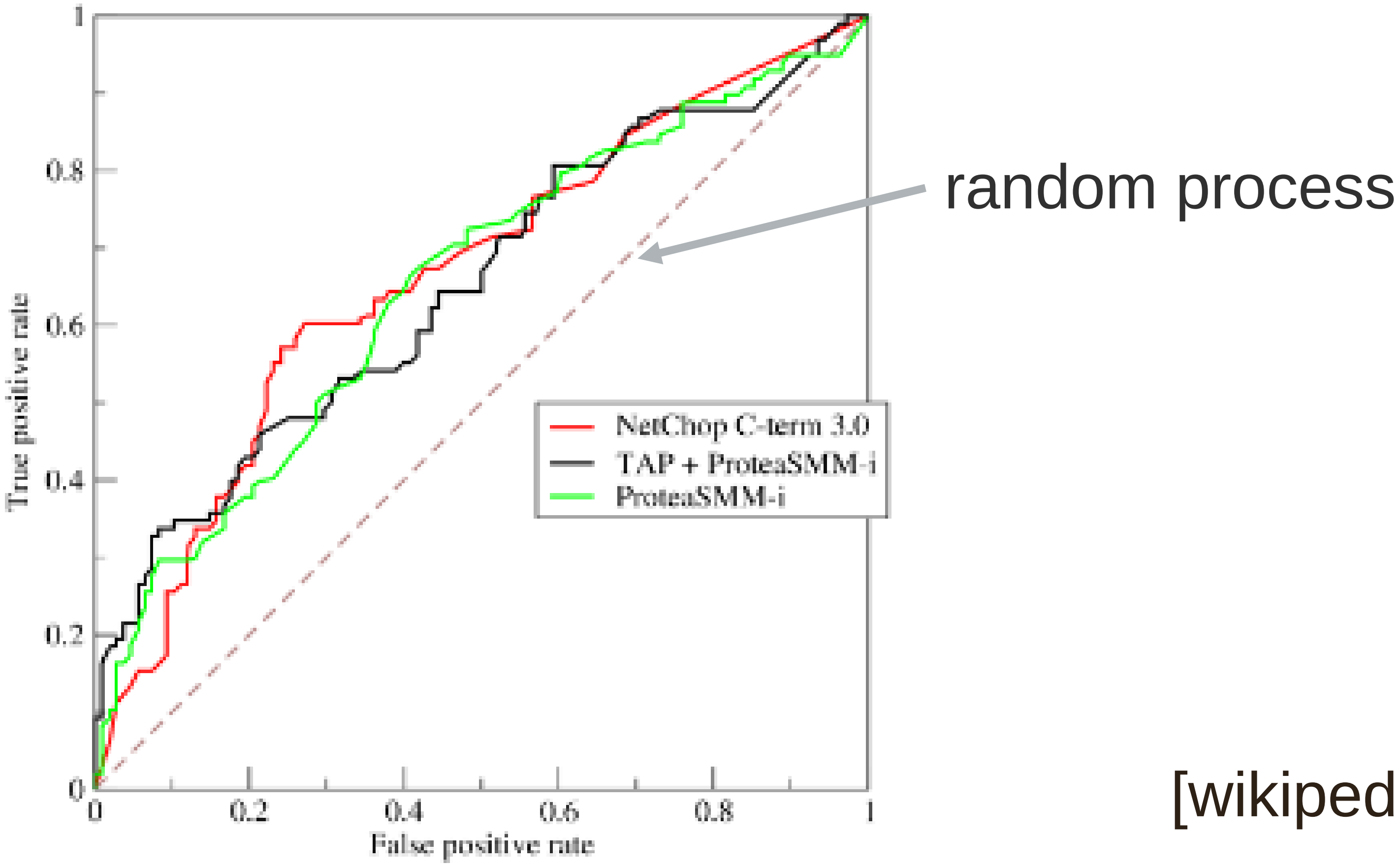
Possible Measures

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

[wikipedia]

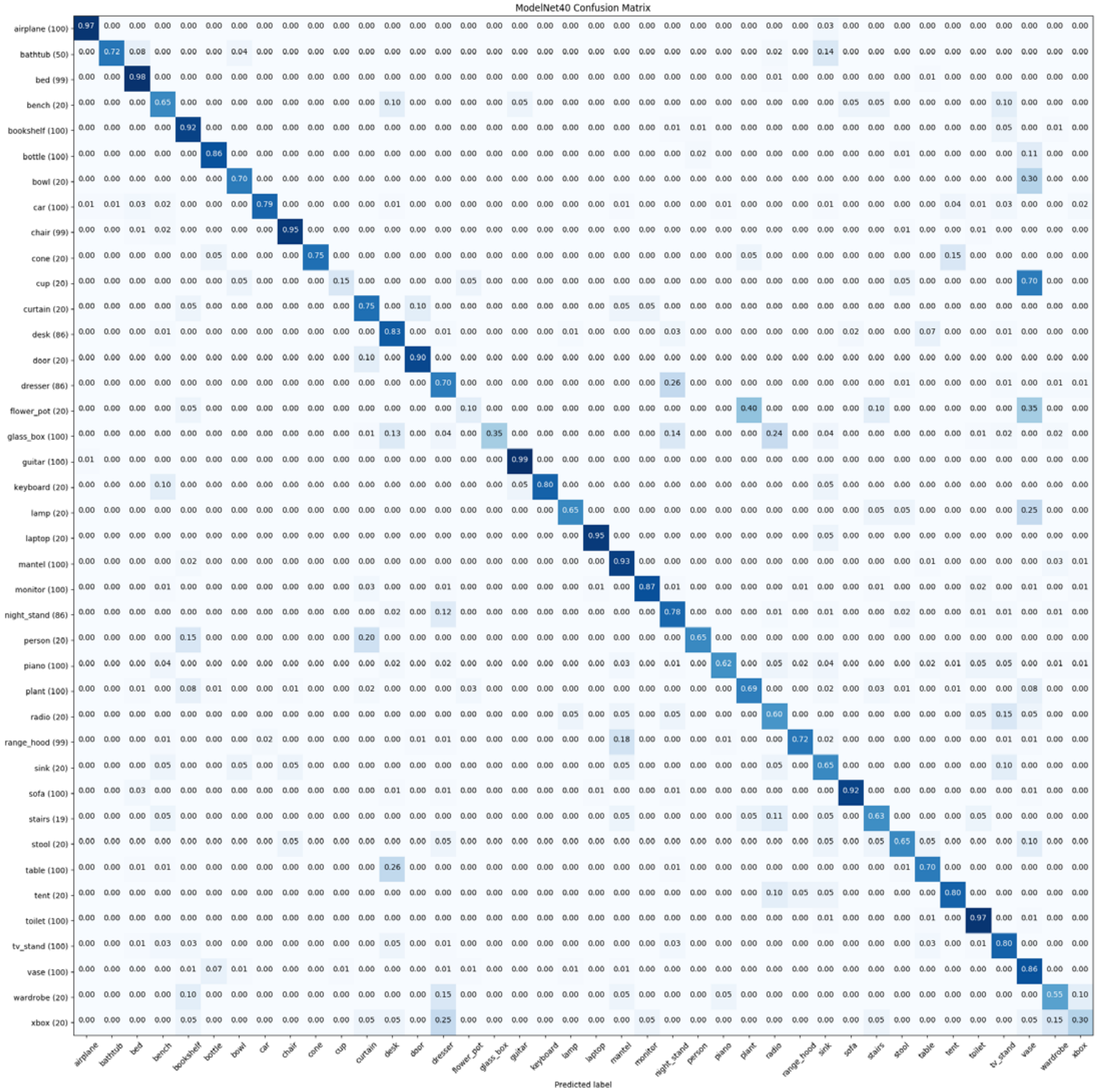
Receiver Operating Characteristic

- Depending on the hyperparameter the performance can be tuned, accepting more or less samples in one class
- True positive / False positive rate: development of TP/FP given this parameter



[wikipedia]

Confusion Matrix



References

- Bernhard Schölkopf und Alexander J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press 2001.
- V. Vapnik, Statistical Learning Theory, Wiley 1998.
- www.cs.utexas.edu/users/mooney/cs391L/svm.ppt
- Mingyue Tan, Support Vector Machine & Its Applications, UBC 2004
- Raul Rojas, The Curse of Dimensionality, 2015
- Eric Debreuve, An introduction to random forests, INRIA
- Rudolf Kruse, Neuronale Netze, Universität Oldenburg