Photogrammetry & Robotics Lab

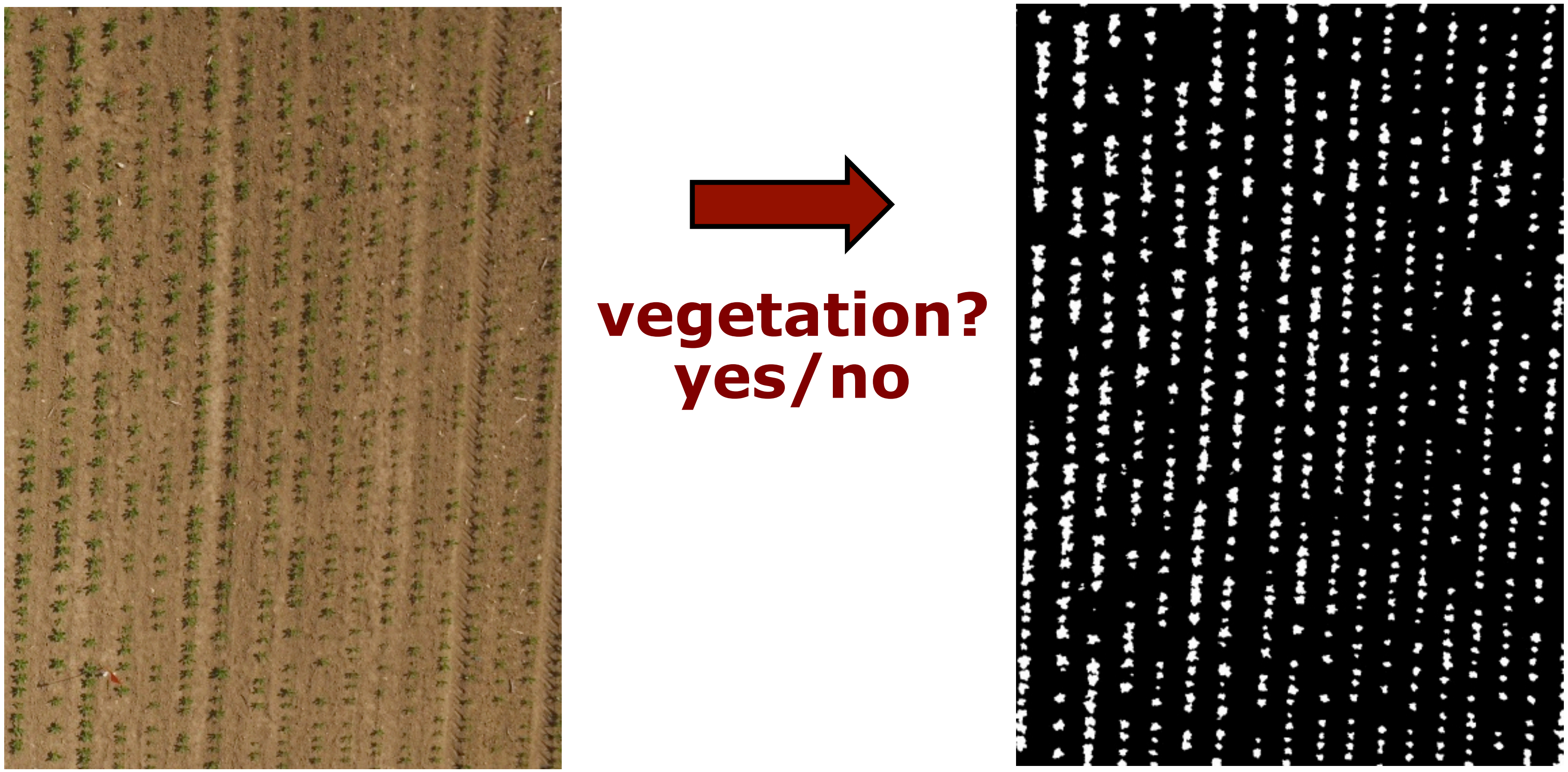
Introduction to Classification

Cyrill Stachniss

The slides have been created by Cyrill Stachniss.

Classification Example

Detecting vegetation



vegetation?
yes/no

Classification Problem

- Given a set of K classes

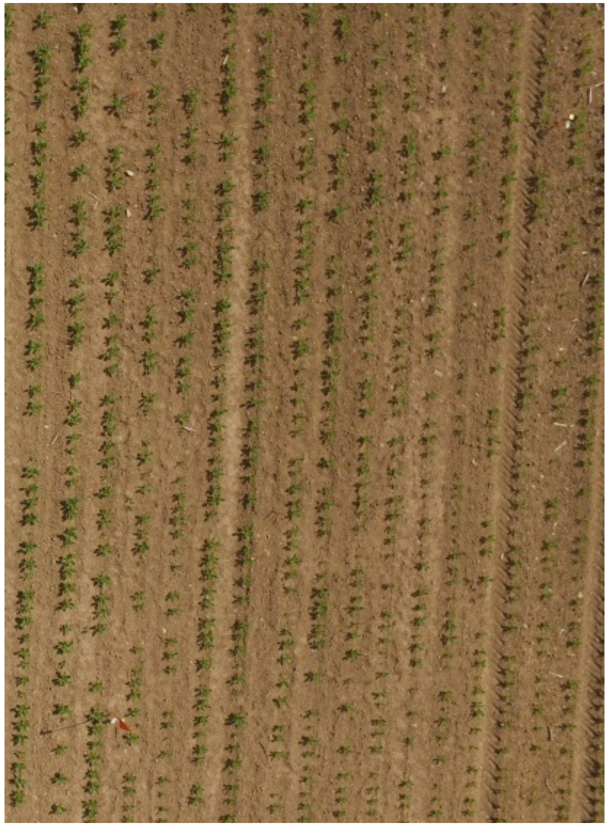
$$\Omega = \{\omega_1, \ldots, \omega_K\}$$

- and features $\boldsymbol{e}$
- learn a function f that assigns a class given feature

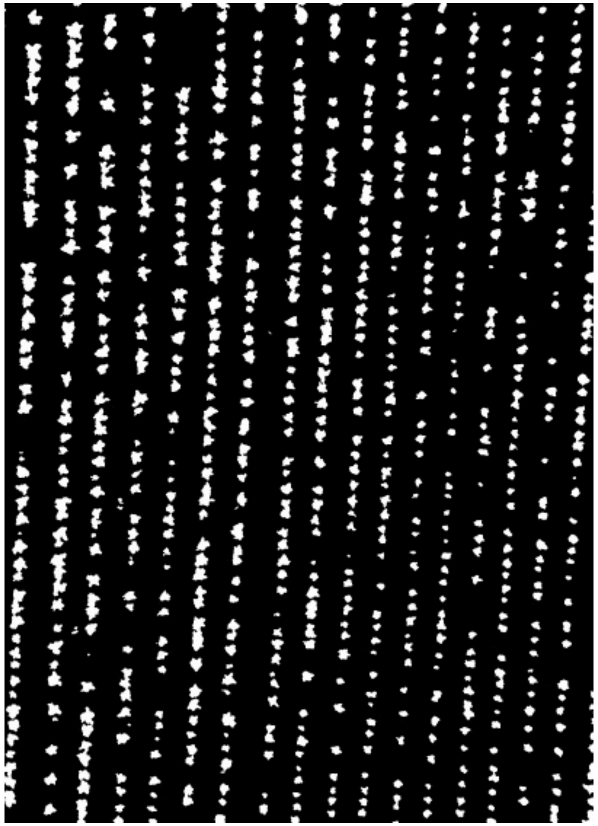
$$c = f(\boldsymbol{e}) \text{ with } c \in \Omega$$

Example

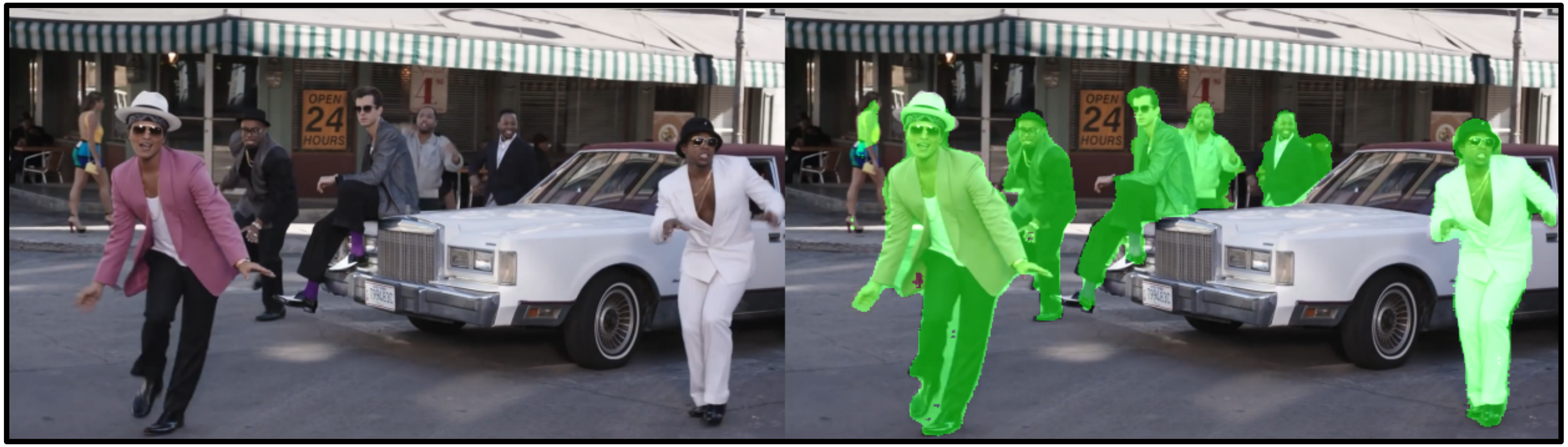
- Features: Pixel intensity values
- Classes: {“vegetation”, “no vegetation”}

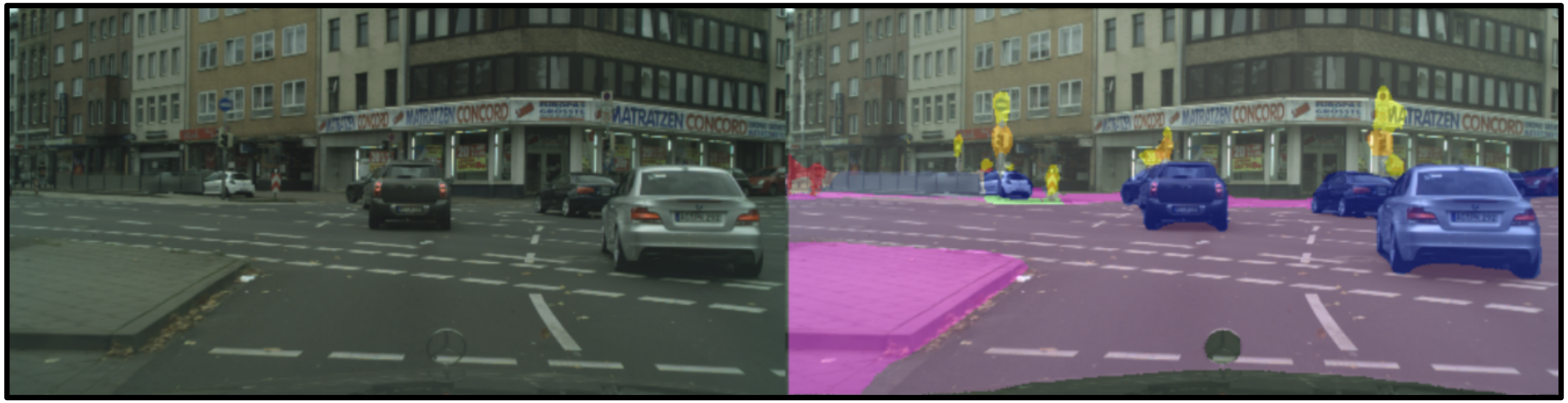


$$c = f(\boldsymbol{e})$$



State of the Art Examples





Example

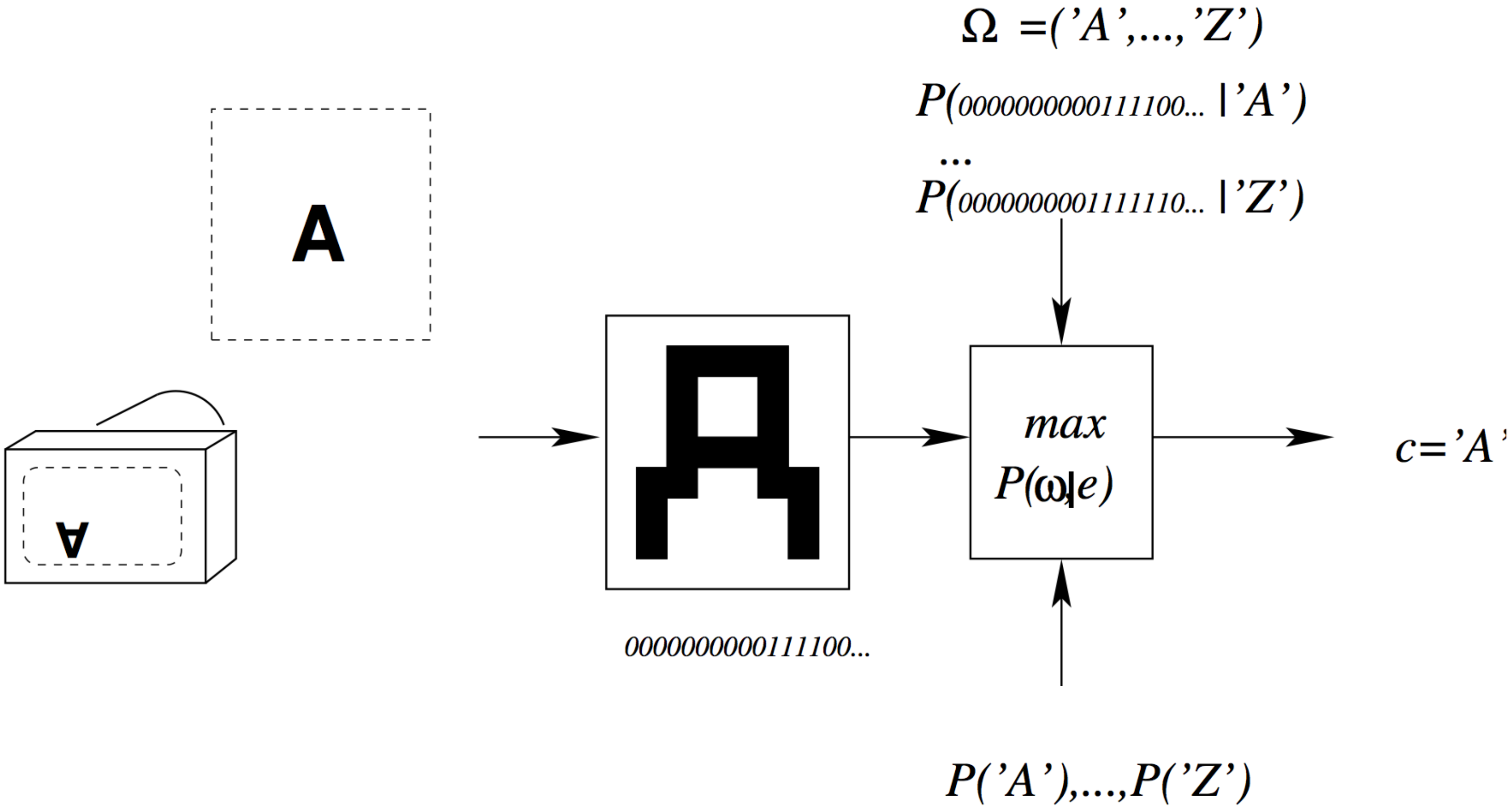


Image courtesy: Förstner

Learning Needs Information

- Learning the function f requires additional information
- This additional information/knowledge can be provided through distributions

$$P('A'), \ldots, P('Z')$$

$$P(00000\ldots|'A')$$
$$\ldots$$
$$P(11111\ldots|'Z')$$

Learning Needs Information

- Learning the function f requires additional information
- This additional information/knowledge can be provided through distributions
- Manually specifying a such knowledge is often hard and thus should be **learned from data**

Training Data

- Learning the function f requires additional information
- This additional information/knowledge is **often** provided through **training data**
- **Training data** are pairs of feature vectors and classes

$$\{(\boldsymbol{e},\omega)\}_{i=1}^{N} \xrightarrow{\text{learning}} f$$

Supervised Learning

- Training data $\{(e, \omega)\}_{i=1}^{N}$
- In case of two classes often called: **positive** and **negative** examples
- The training data is provided by a (often human) supervisor
- Learn a function f that generalizes this decision to new (unseen) data

Classification vs. Regression

Classification

- Output is always a class label
- Goal: find a separation of the input space that correspond to the classes

Regression

- Output is continuous variable
- Goal: Function that fits a set of data points

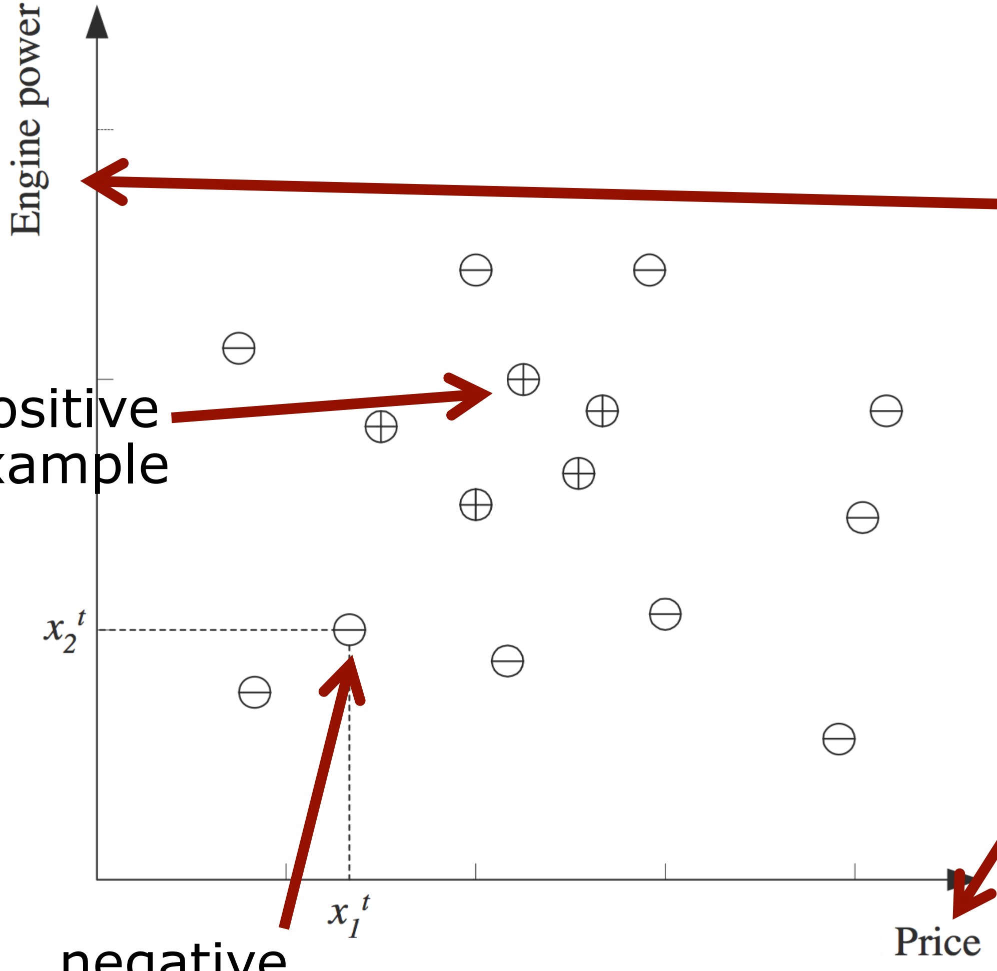
Example: Family Cars

- Features of cars are for example the **price** and **engine power**

- Example values for some cars: (15kE, 50kW), (20kE, 80kW), (7kE, 80kW), …
- Training data:(15kE, 50kW, true), (20kE, 80kW, true), (7kE, 80kW, false), …

- Classifier: “Is (12kE, 65kW) a family car?”

Example: Is it a Family Car?



feature vector

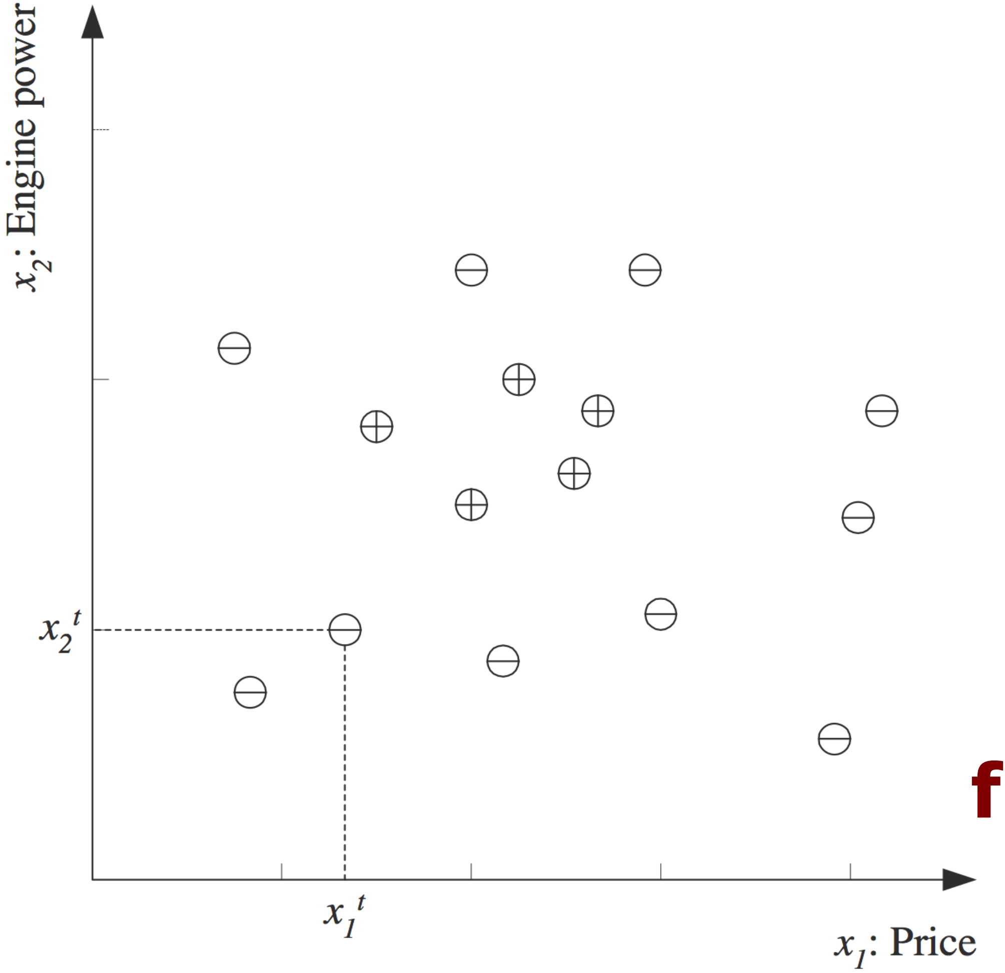
$$\boldsymbol{e} = (x_1, x_2)^\top$$

positive example

negative example

Image courtesy: Aplaydin

Example: Is it a Family Car?



feature vector

$$\boldsymbol{e} = (x_1, x_2)^\top$$

How to to make a decision for a new car if it is a family car or not?

Image courtesy: Aplaydin

Hypothesis

Are there other possible solutions?

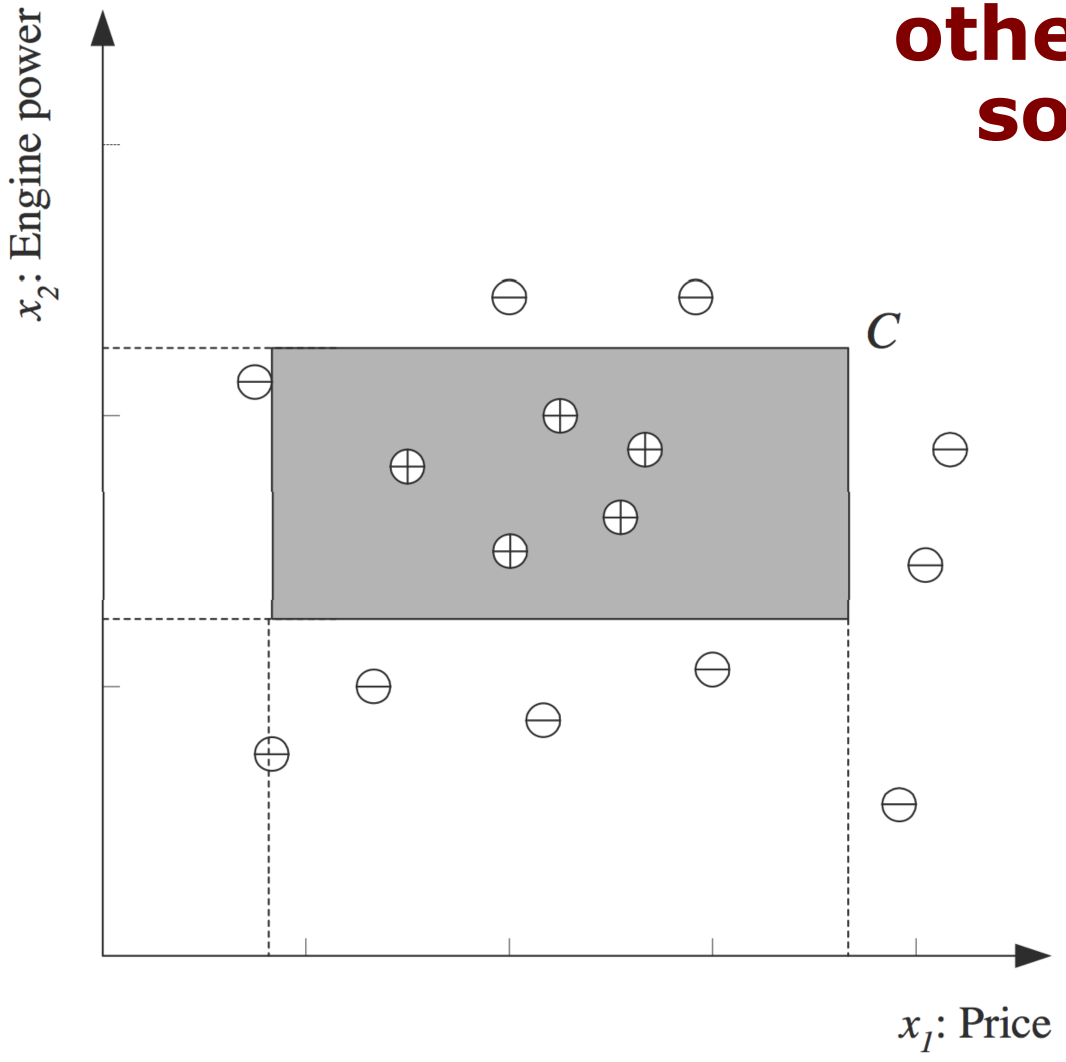


Image courtesy: Aplaydin

Multiple Consistent Hypotheses with the Training Data

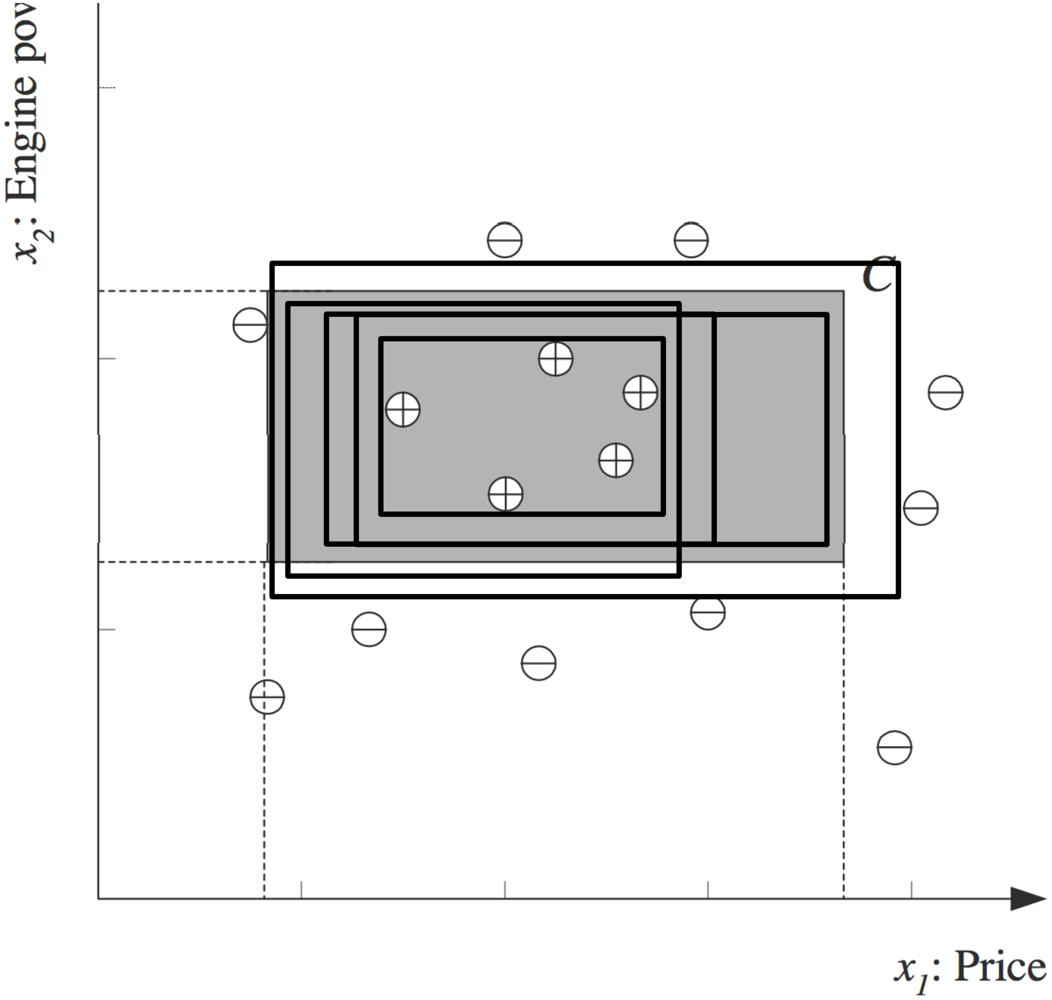
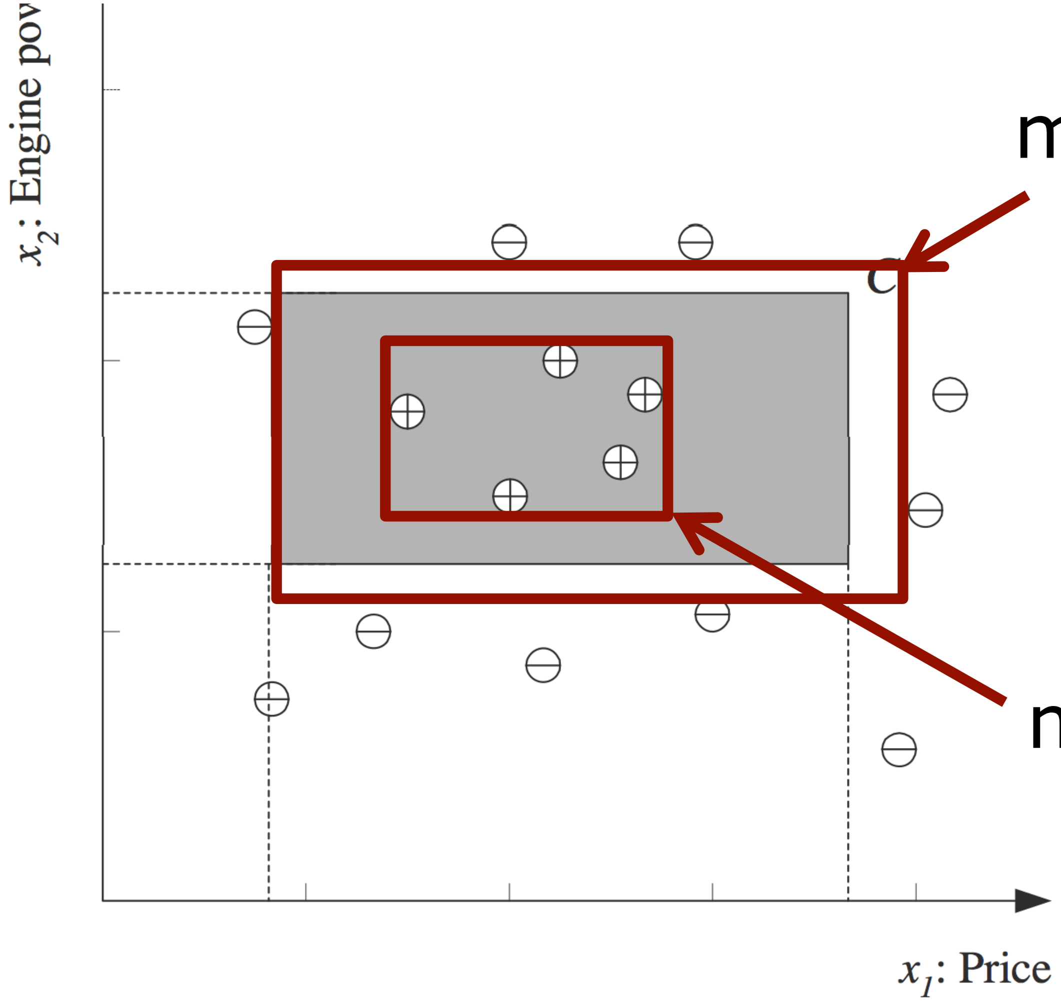


Image courtesy: Aplaydin

Multiple Consistent Hypotheses with the Training Data

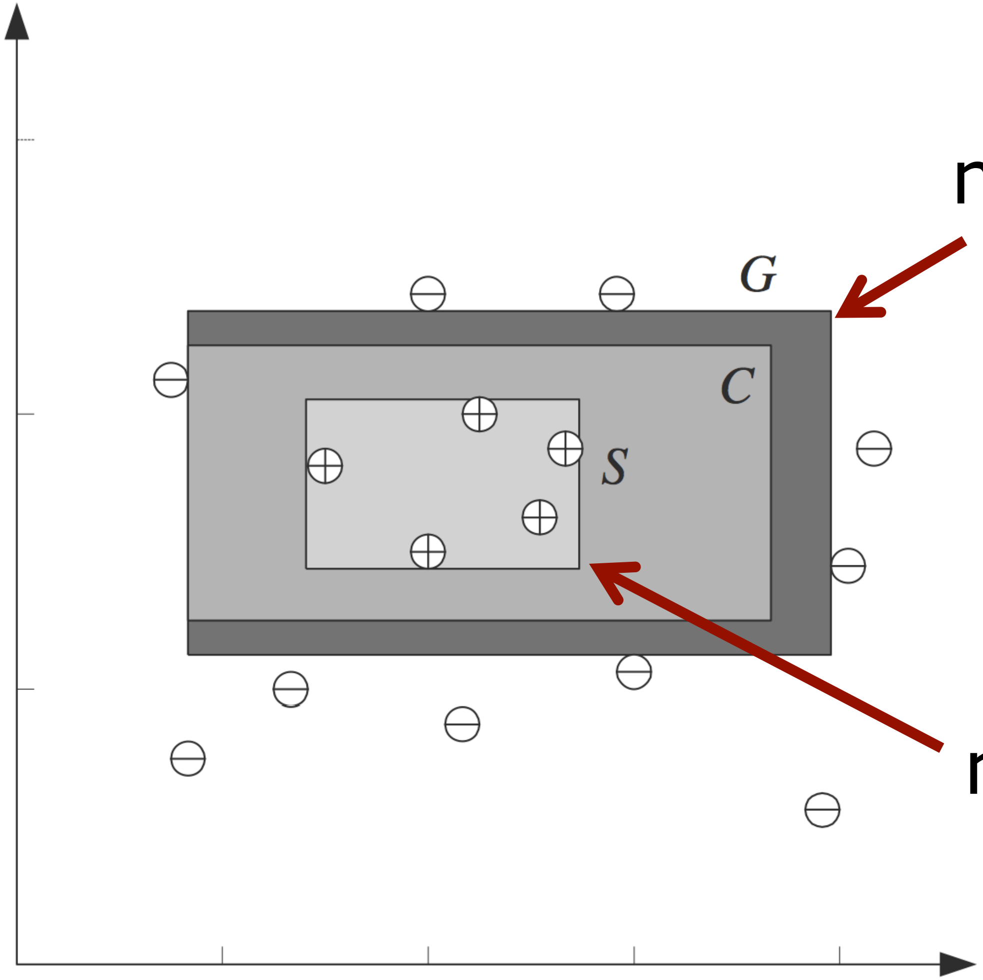


most general hypothesis

most specific hypothesis

Image courtesy: Aplaydin

Most Specific and Most General

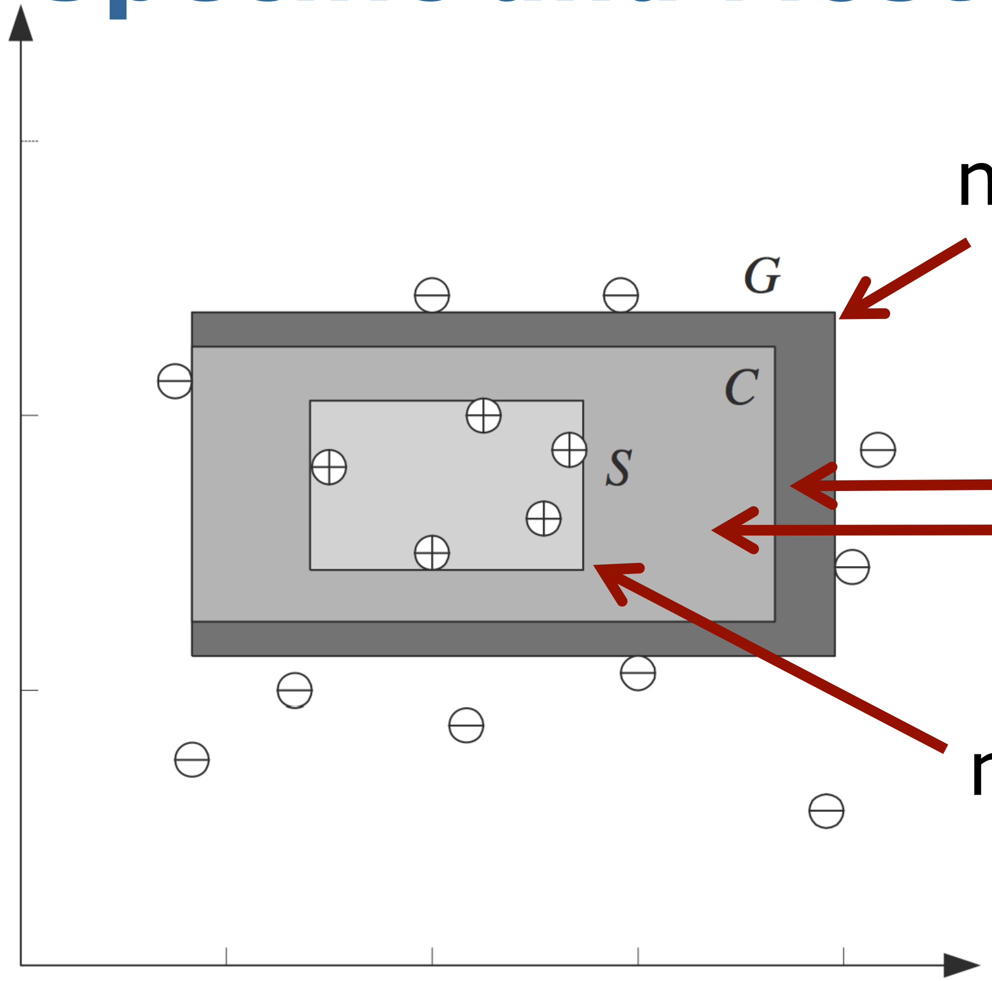


most general hypothesis

most specific hypothesis

Image courtesy: Aplaydin

Version Space is Between the Most Specific and Most General



most general hypothesis

version space

most specific hypothesis

Image courtesy: Aplaydin

Choose the Hypothesis that Maximizes the Margin to the Most Specific and General One

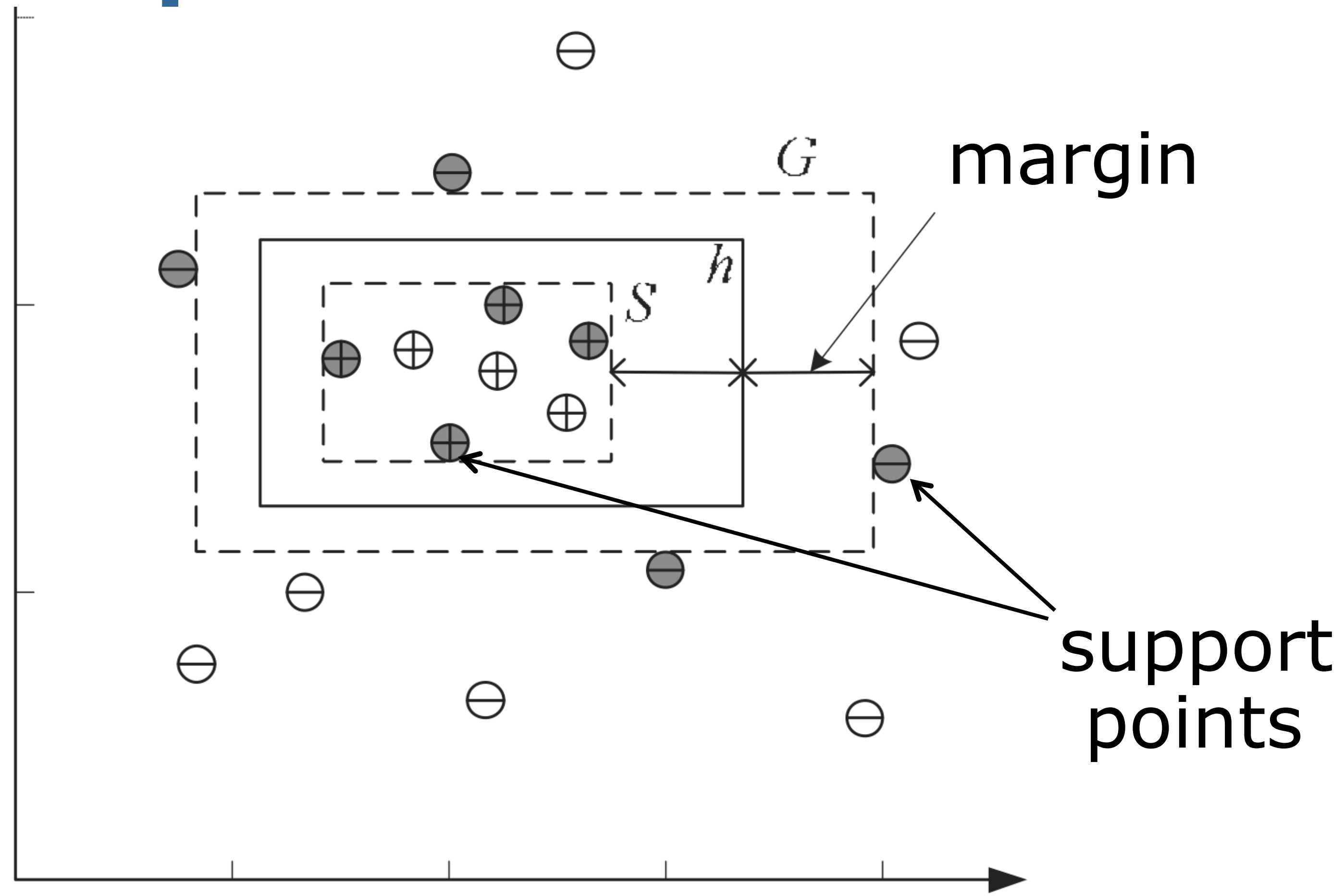
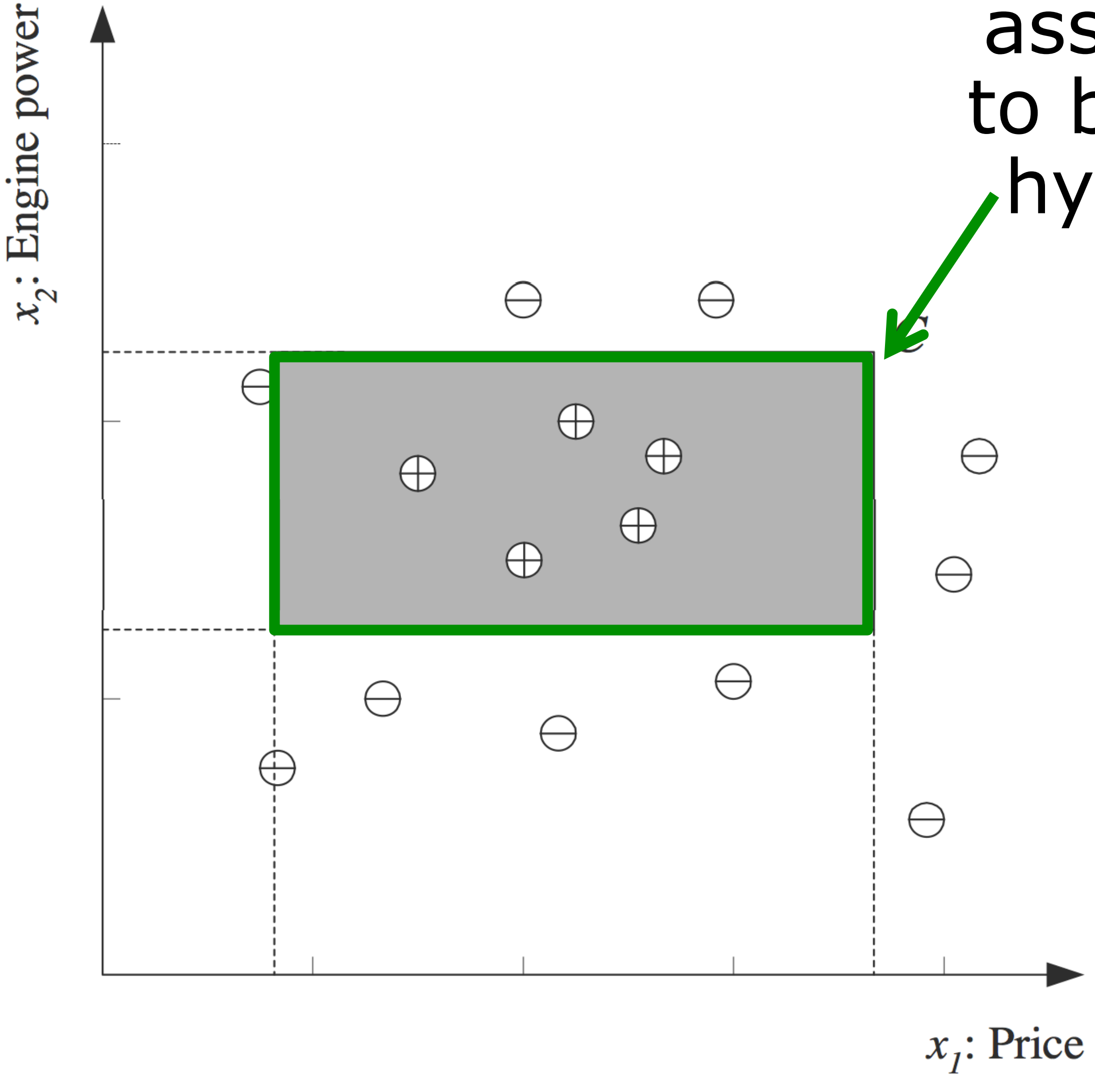


Image courtesy: Aplaydin

Classification Errors

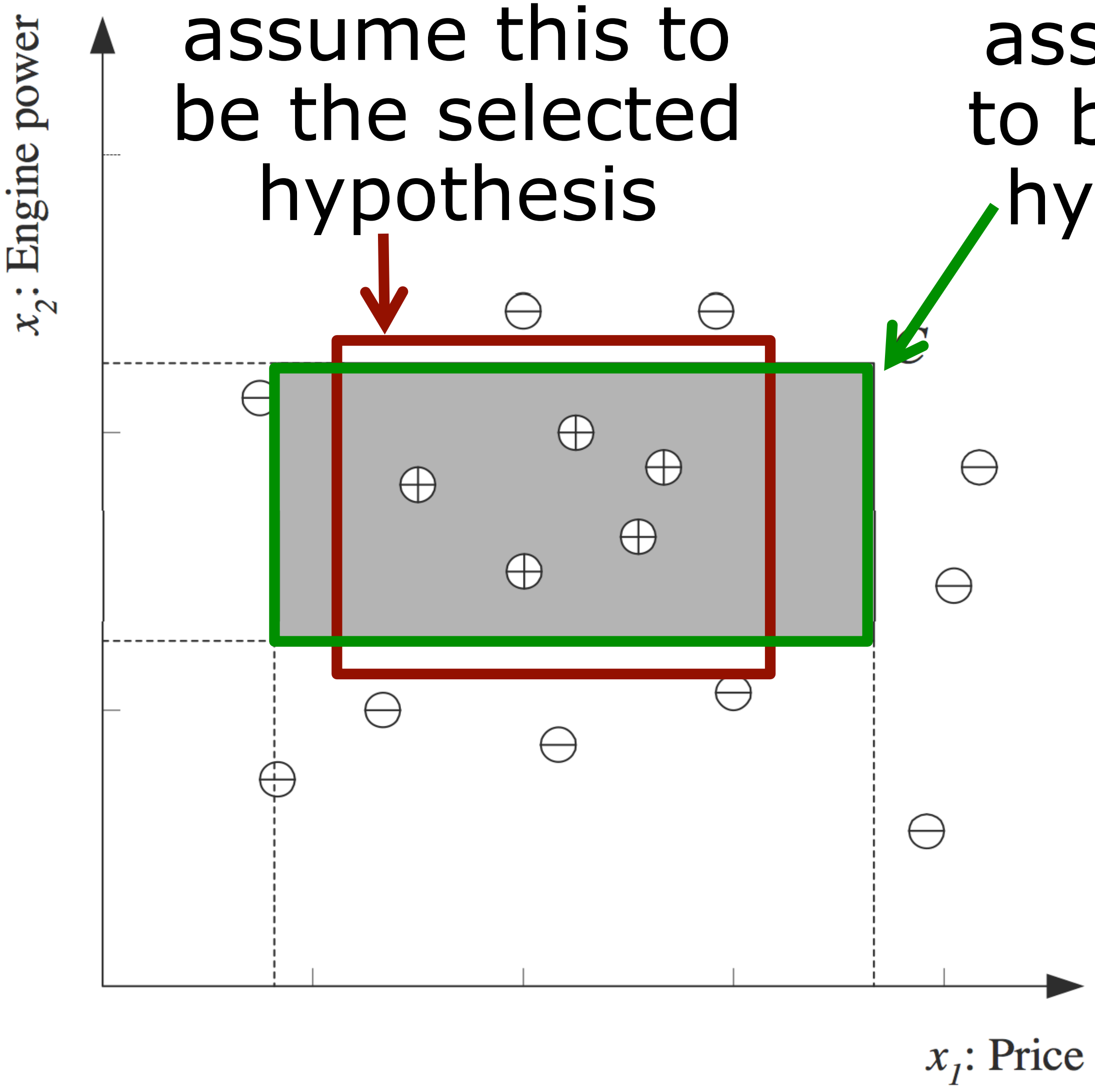
Classification Errors



assume this to be correct hypothesis

Image courtesy: Aplaydin

Classification Errors

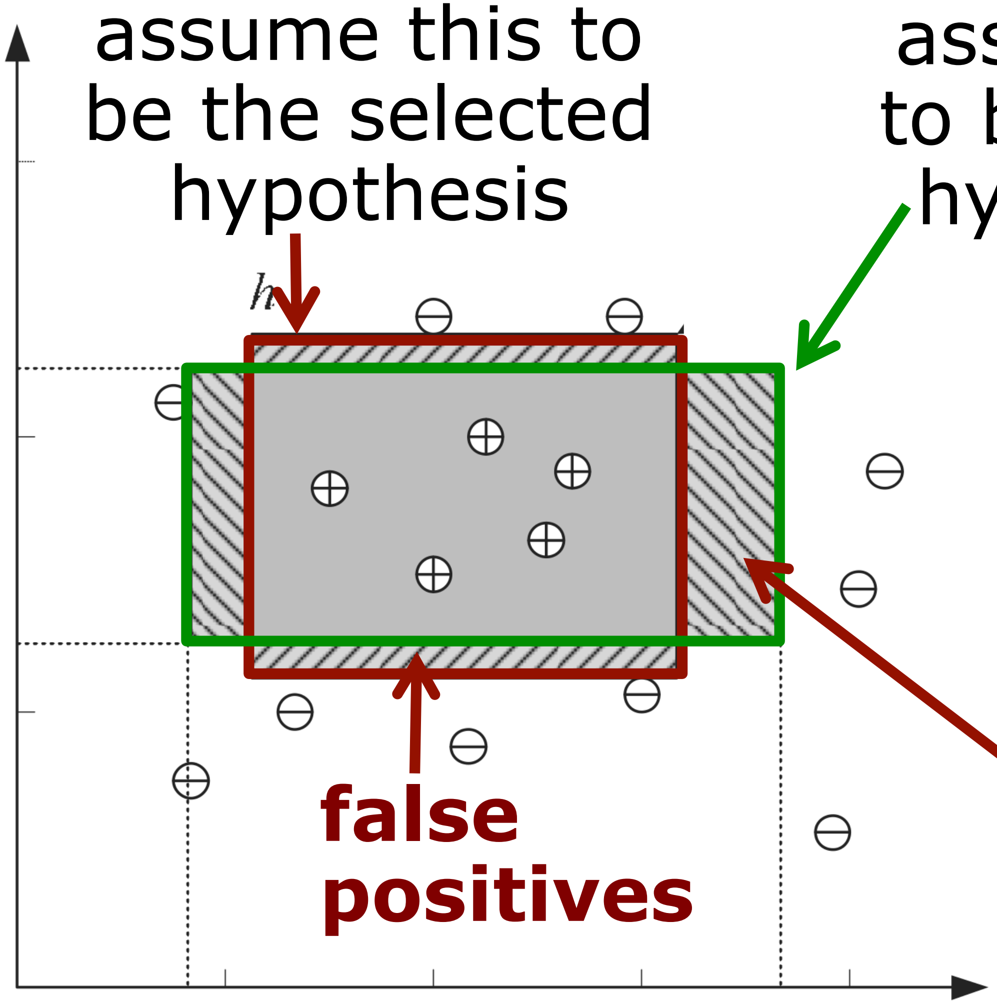


assume this to be the selected hypothesis

assume this to be correct hypothesis

Image courtesy: Aplaydin

Classification Errors



assume this to be the selected hypothesis

assume this to be correct hypothesis

false positives

false negatives

Image courtesy: Aplaydin

Possible Outcomes

- A family car is **correctly** classified as a family car (TP)
- A family car is **wrongly** classified as a non-family car (FN)
- A non-family car is **correctly** classified as a non-family car (TN)
- A non-family car is **wrongly** classified as a family car (FP)

Possible Outcomes

- **True Positives (TP)**: all positive examples classified as positives
- **False Negatives (FN)**: all positive examples classified as negatives
- **True Negatives (TN)**: all negative examples classified as negatives
- **False Positives (FP)**: all negative examples classified as positives

Possible Outcomes

		in reality	
		positive	negative
classified as	positive	TP	FP
	negative	FN	TN

- FP is also called type I error
  (In German: Fehler 1. Art oder α-Fehler)
- FN is also called type II error
  (In German: Fehler 2. Art oder β-Fehler)

Identical to the Standard Confusion Matrix for 2 Classes

in reality

classified as

	positive	negative
positive	**TP**	**FP**
negative	**FN**	**TN**

confusion matrix

in reality

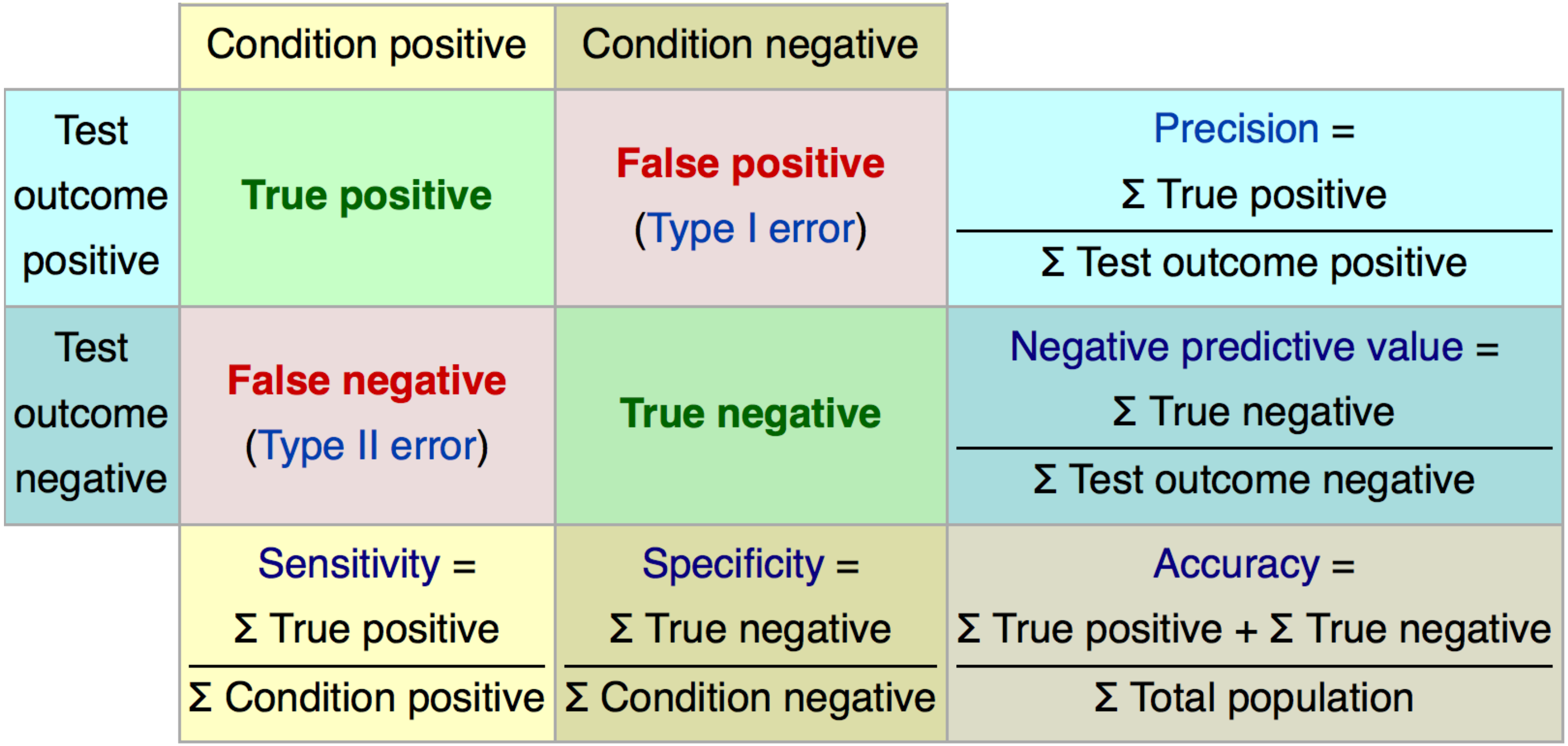
classified as

	class 1	class 2
class 1	**1 as 1**	**2 as 1**
class 2	**1 as 2**	**2 as 2**

Evaluating a Classifier

in reality

classified as



	Condition positive	Condition negative	
Test outcome positive	**True positive**	**False positive** (Type I error)	Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$
Test outcome negative	**False negative** (Type II error)	**True negative**	Negative predictive value = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Sensitivity = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	Specificity = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Accuracy = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$

(sensitivity is also called **recall** or **true positive rate**)

(specificity is also called **true negative rate**)

Image courtesy: Wikipedia.org

False and True Positive Rate

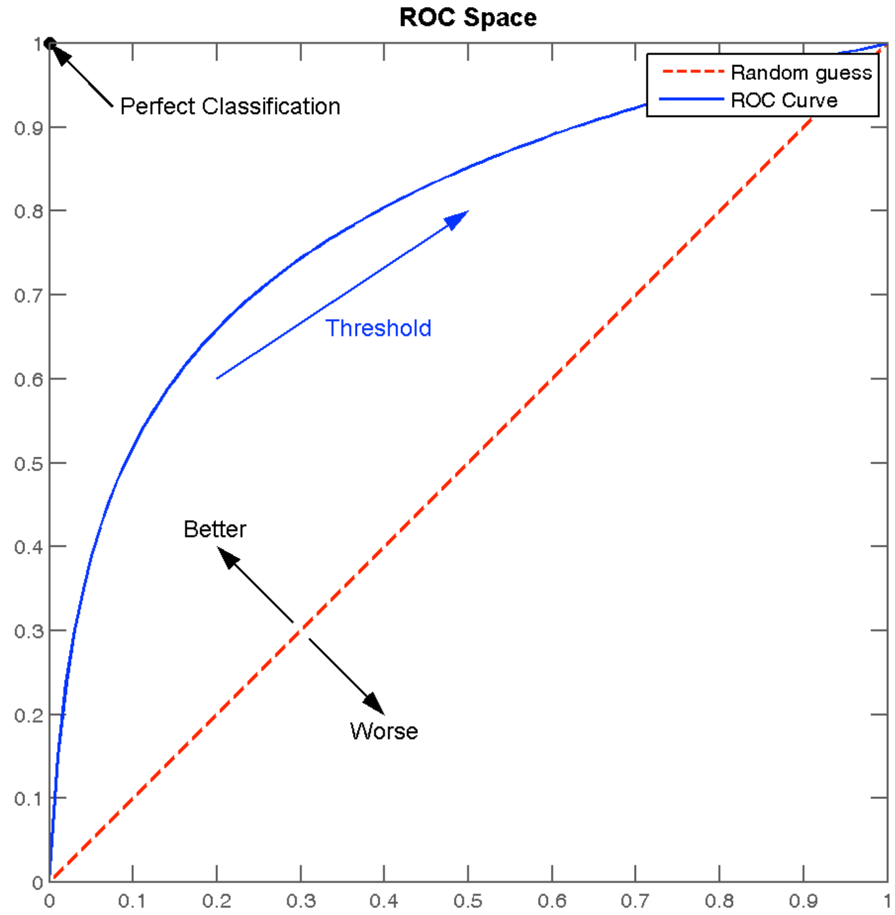
- **False positive rate** is the probability that a randomly selected and in reality negative example is classified positive

$$\text{false positive rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- **True positive rate** (=sensitivity, recall) is the probability that a randomly selected, in reality positive example is classified as positive

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Receiver Operating Characteristic (ROC Curves)



true positive rate / recall / sensitivity

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

false positive rate

$$\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$$

Image courtesy: Wikipedia

Precision and Recall

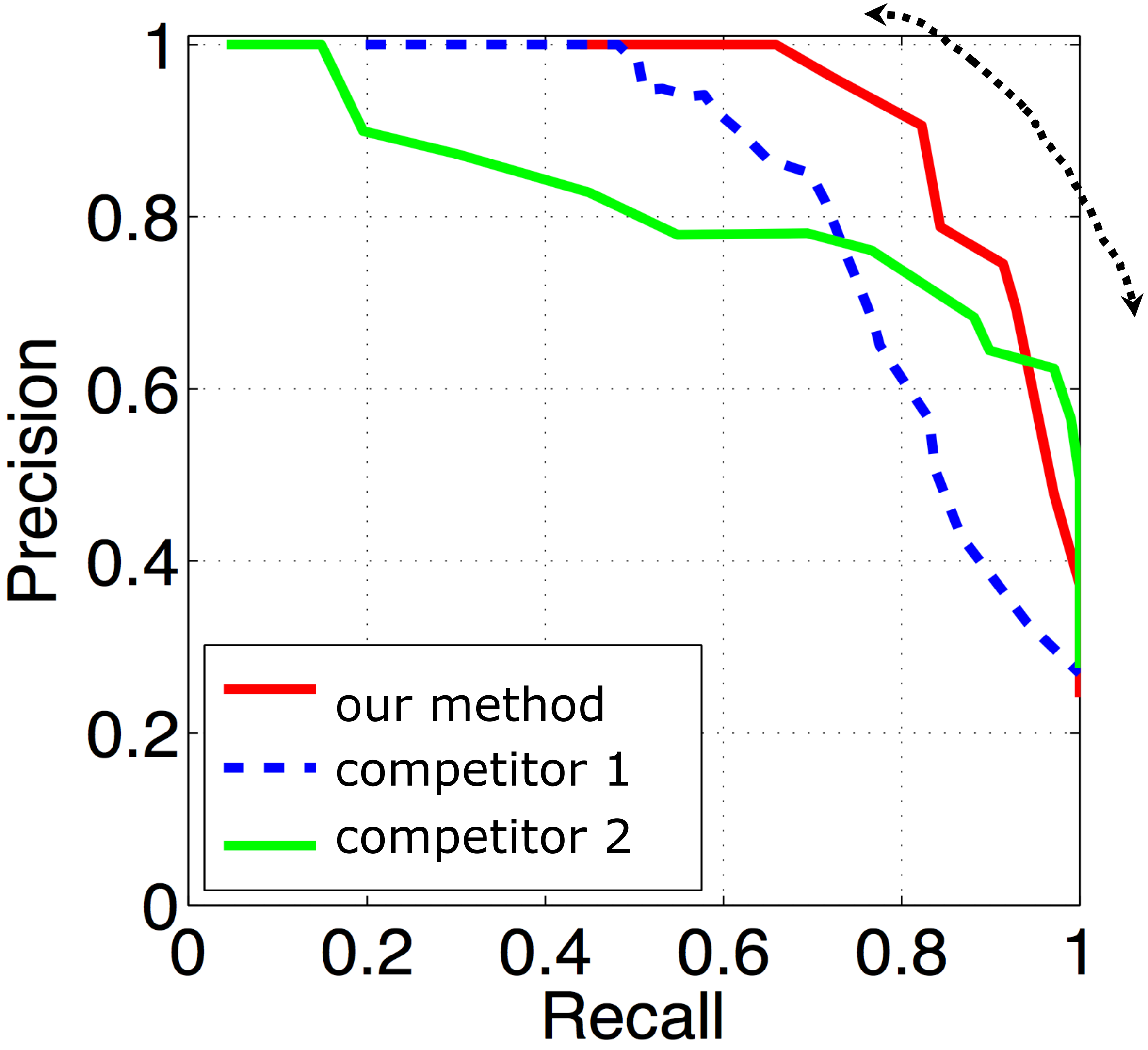
- **Precision** is the probability that a randomly selected, positively classified example is positive in reality

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall** (=sensitivity, true positive rate) is the probability that a randomly selected, in reality positive example is classified as positive

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision Recall Plots



precision recall
values obtained
for different
parameters

F-score / F_1 score / F-measure

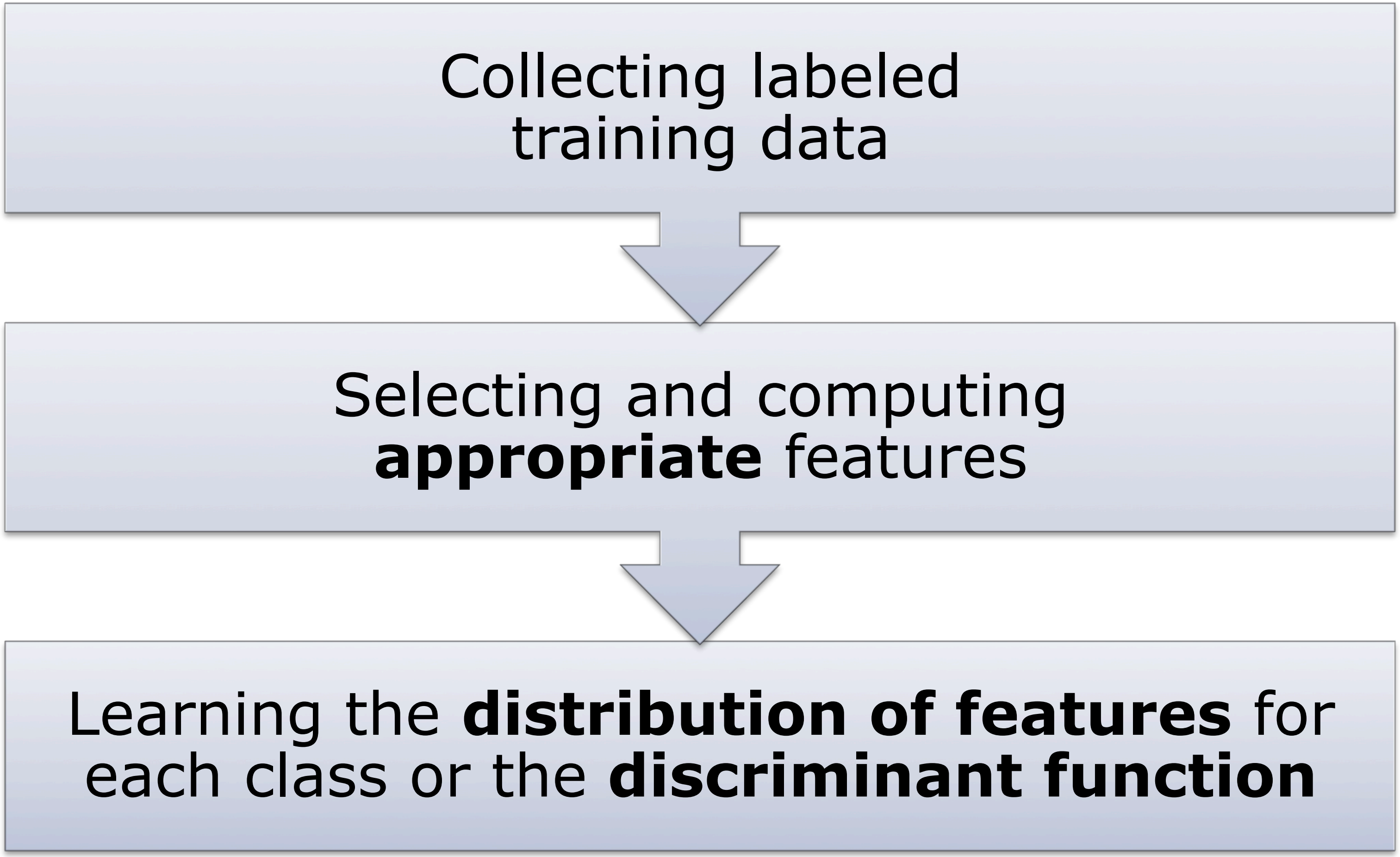
- Combines precision and recall into one value (harmonic mean)
- F-score reaches its best value at 1 and its worst score at 0

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- Note: There is a large number of different measures…

Designing a Traditional Classifier

Traditional Classification: Training

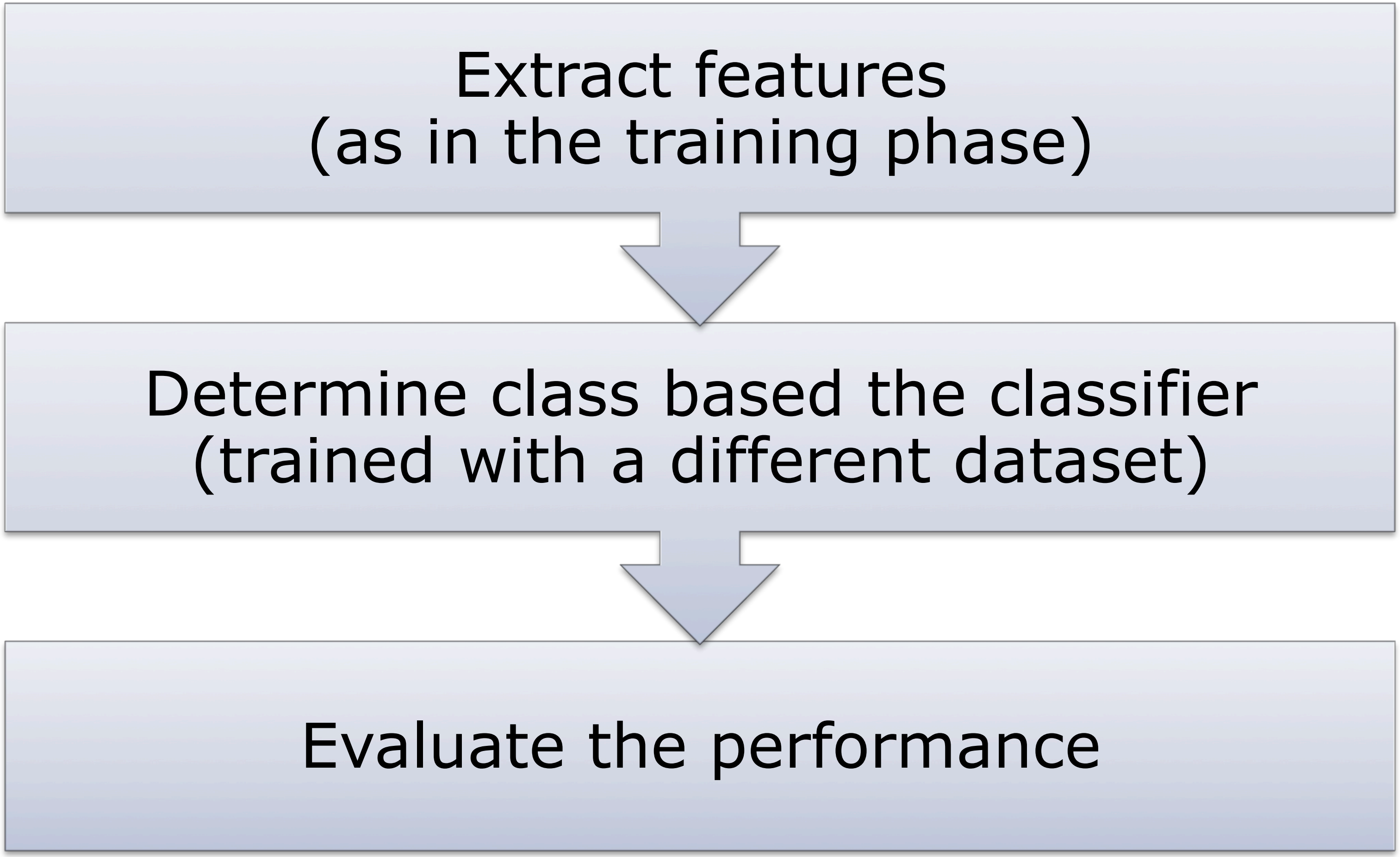


Collecting labeled training data

Selecting and computing **appropriate** features

Learning the **distribution of features** for each class or the **discriminant function**

Traditional Classification: Testing (on different datasets)

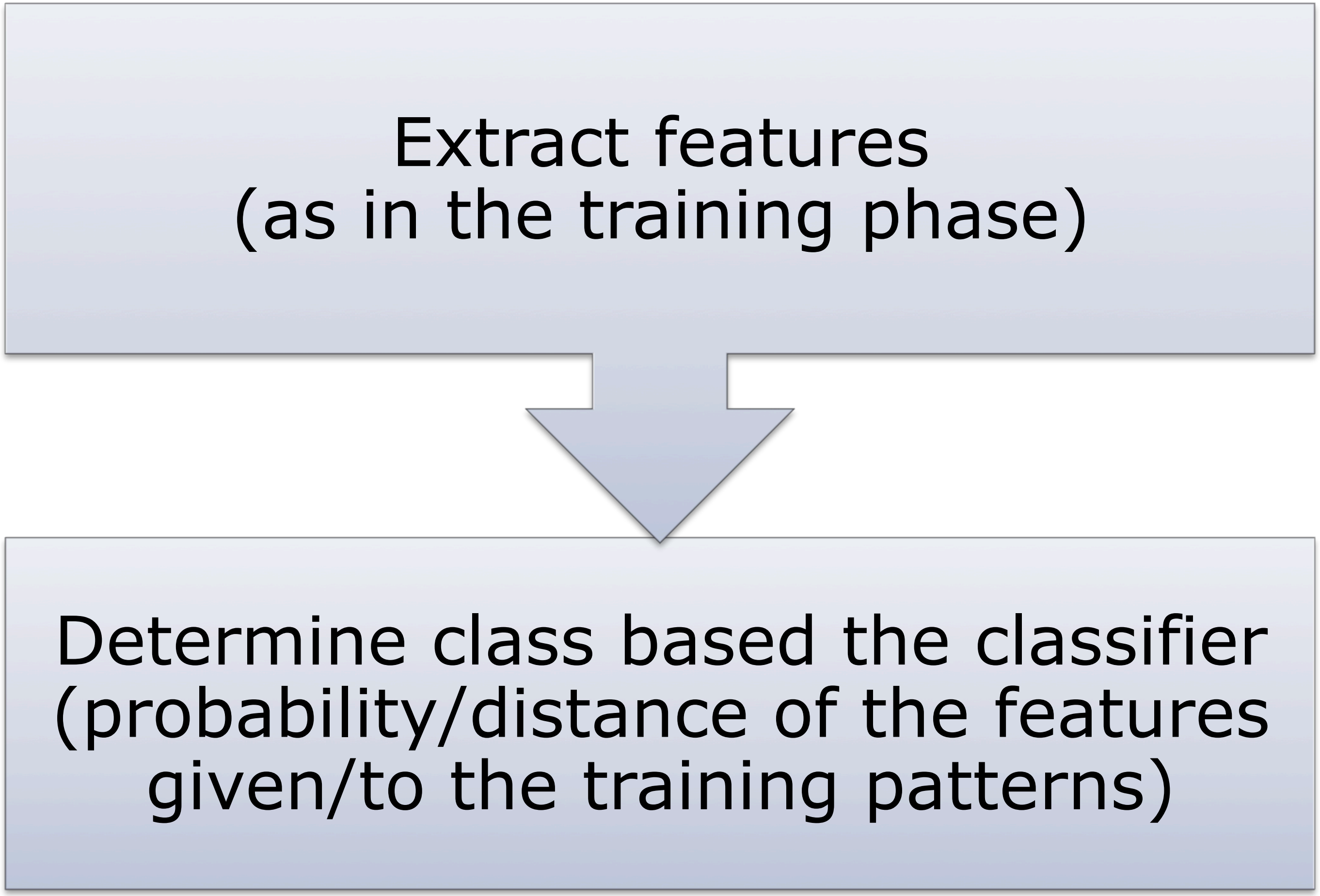


Extract features
(as in the training phase)

Determine class based the classifier
(trained with a different dataset)

Evaluate the performance

Traditional Classification: Operation

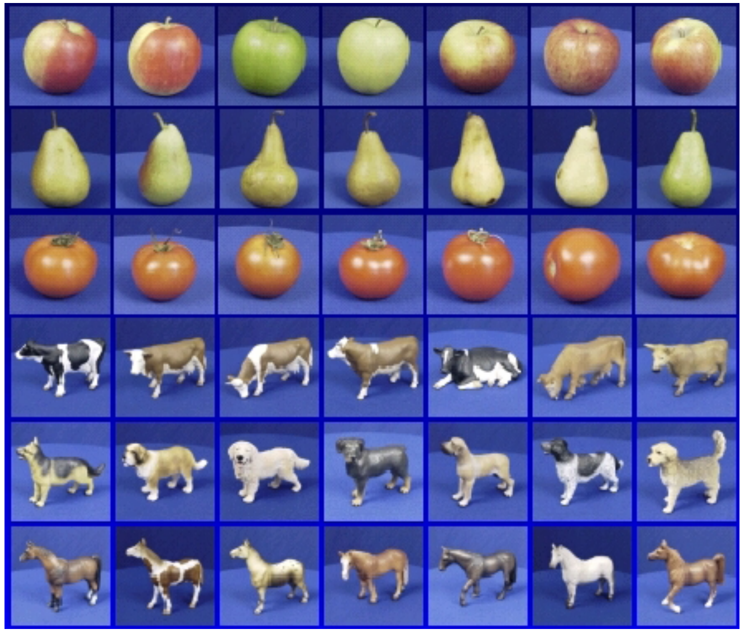


Extract features
(as in the training phase)

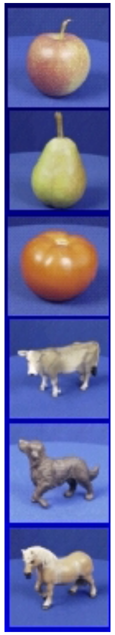
Determine class based the classifier
(probability/distance of the features
given/to the training patterns)

Generalization

How well does a model generalize from the data it was trained on to a new test set?



Training set (labels known)



Test set
(labels unknown)

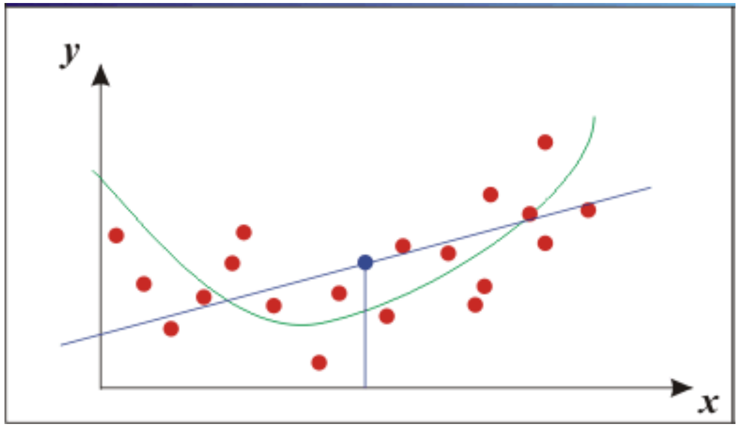
Slide courtesy: Lazebnik

Components of the Generalization Error

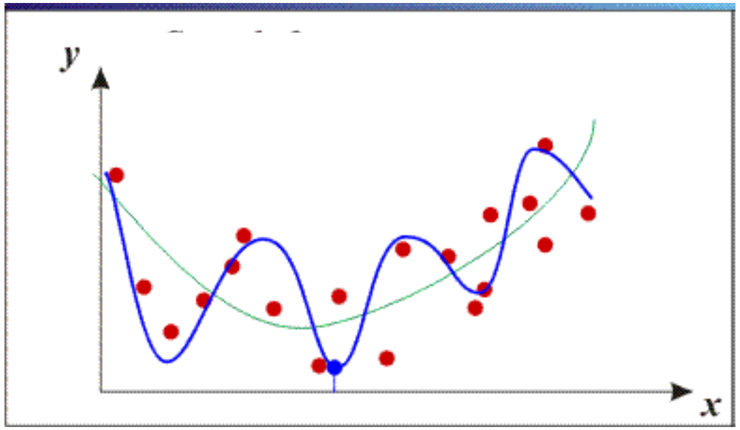
- **Bias** describes how much the average model over all training sets differ from the true model. These are errors due to inaccurate assumptions/ simplifications made in the model
- **Variance** describes how much models estimated from different training sets differ from each other

Slide courtesy: Lazebnik

Bias-Variance Trade-Off



Not enough flexibility: models with too few parameters are inaccurate because of a **large bias**



Fitting to the noise in the training data: models with too many parameters are inaccurate because of a **large variance**

Slide courtesy: Hoiem

Bias-Variance Trade-Off

$$E(MSE) = \text{noise}^2 \quad + \quad \text{bias}^2 \quad + \quad \text{variance}$$

unavoidable error

error due to incorrect assumptions

error due to variance of training samples

More explanations on the bias-variance trade-off

http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Slide courtesy: Hoiem

Underfitting & Overfitting

- **Underfitting:** model is too “simple” to represent the relevant characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide courtesy: Lazebnik

Rules of Thumb

- Try **simple classifiers first**
- Use increasingly more **powerful classifiers** with **more training data**
- **Find good features:** Better to have smart features and simple classifiers than simple features and smart classifiers

Slide courtesy: Hoiem

Remember…

No classifier is inherently better than any other: we need to **make assumptions to generalize**

Three **types of errors**

- Inherent **noise:** unavoidable
- **Bias:** due to over-simplifications
- **Variance:** due to inability to perfectly estimate parameters from limited data

Slide courtesy: Hoiem

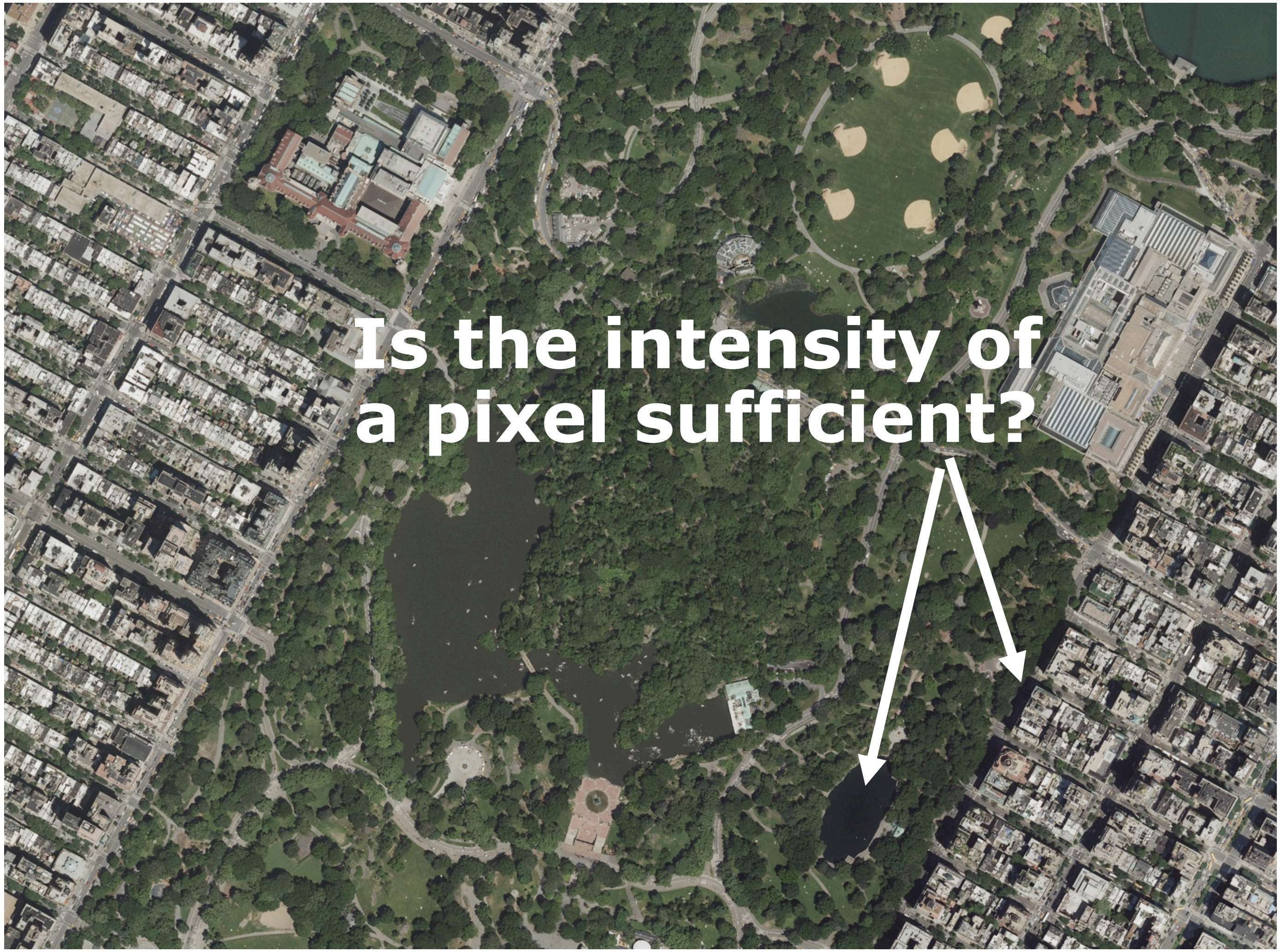
5x2 Cross Validation

- Randomly split up labeled dataset into 2 parts of equal size
- Use one for training, the second one for testing (validation)
- Swap both sets
- Repeat 5 times (called folds)
- Analyze the classification errors

➡ Results in different 10 classifiers

Feature Example for Remote Sensing Data

Features

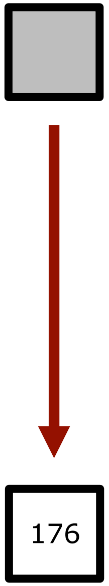


Is the intensity of a pixel sufficient?

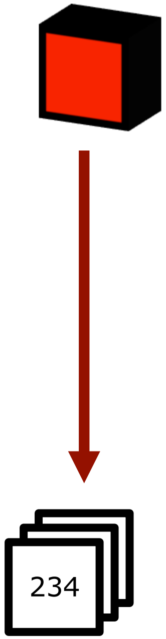
Slide courtesy: Roscher; Image Courtesy: USGS EarthExplorer

Features

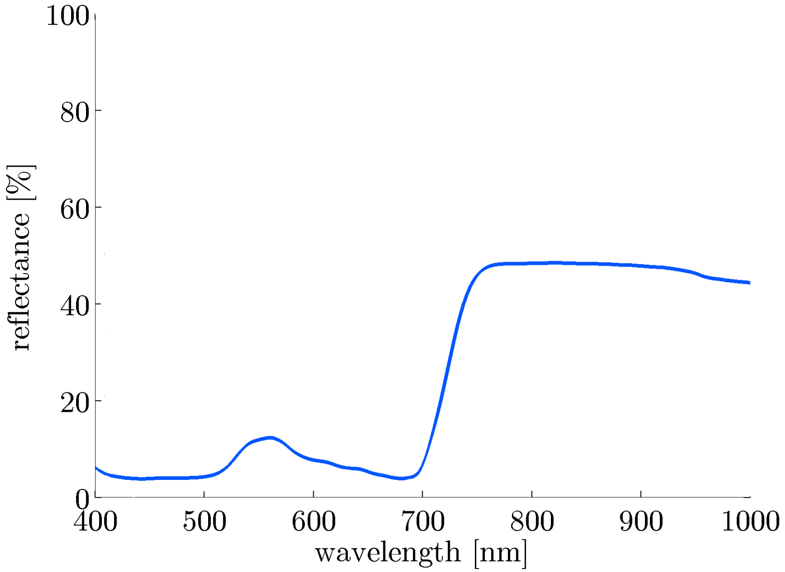
Intensity

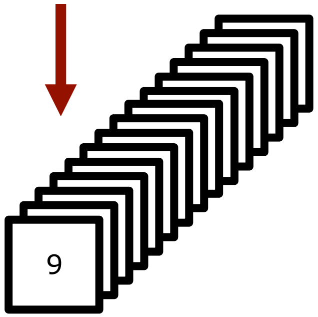


RGB-value



Hyperspectral signature



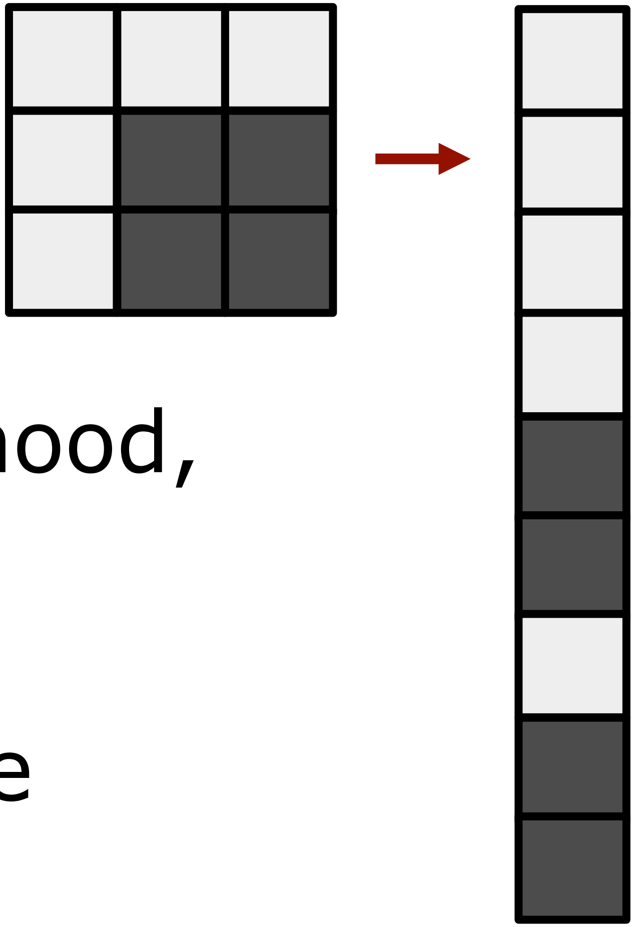


One pixel usually contains a lot of information.

Slide courtesy: Roscher

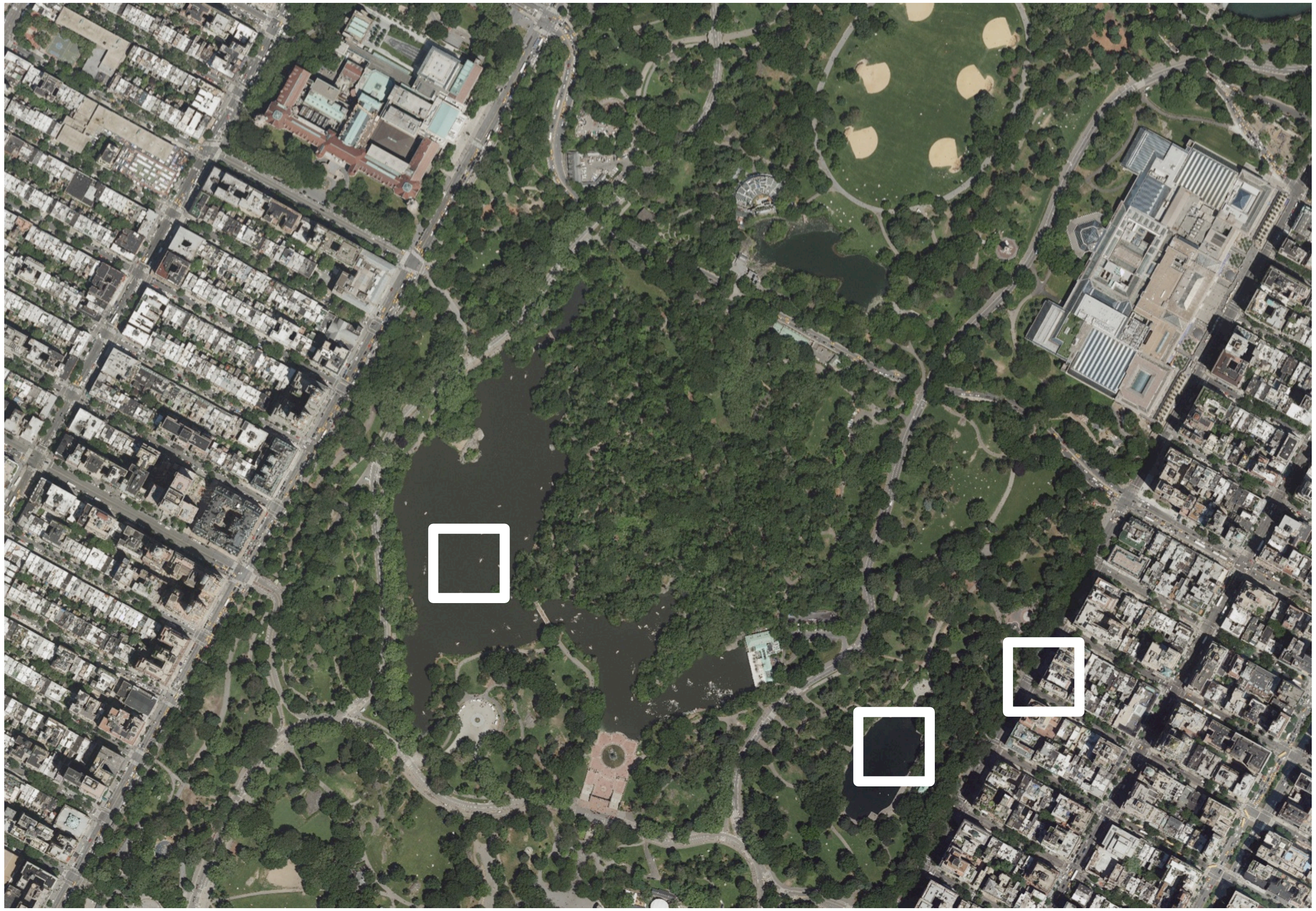
Neighborhood Information

- A feature can also cover multiple pixels, i.e., information about **neighborhood**
- The larger the neighborhood, the higher the feature dimension
- The neighborhood can be patch-based or region/ segment-based



Slide courtesy: Roscher

Higher-Dimensional Features for Neighborhood Information



Slide courtesy: Roscher; Image Courtesy: USGS EarthExplorer

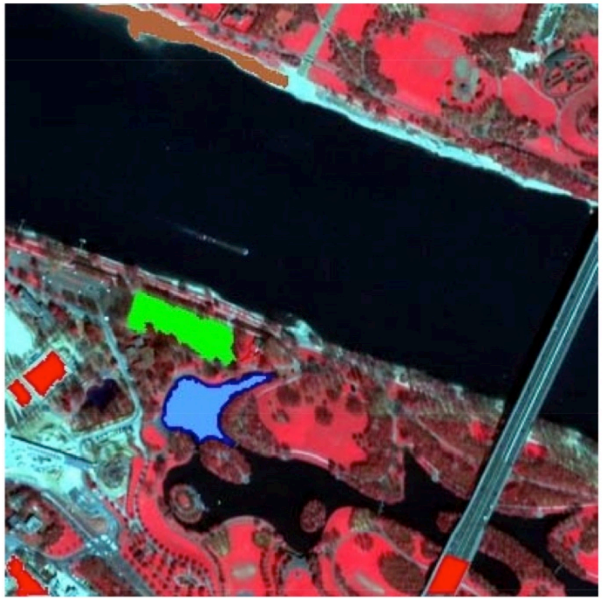
Nearest Neighbor Classification

Nearest Neighbor Approach to Classification

- The feature distribution is modeled by the training data (or a subset)
- The class is **assigned** based on the **closest feature** in the training data

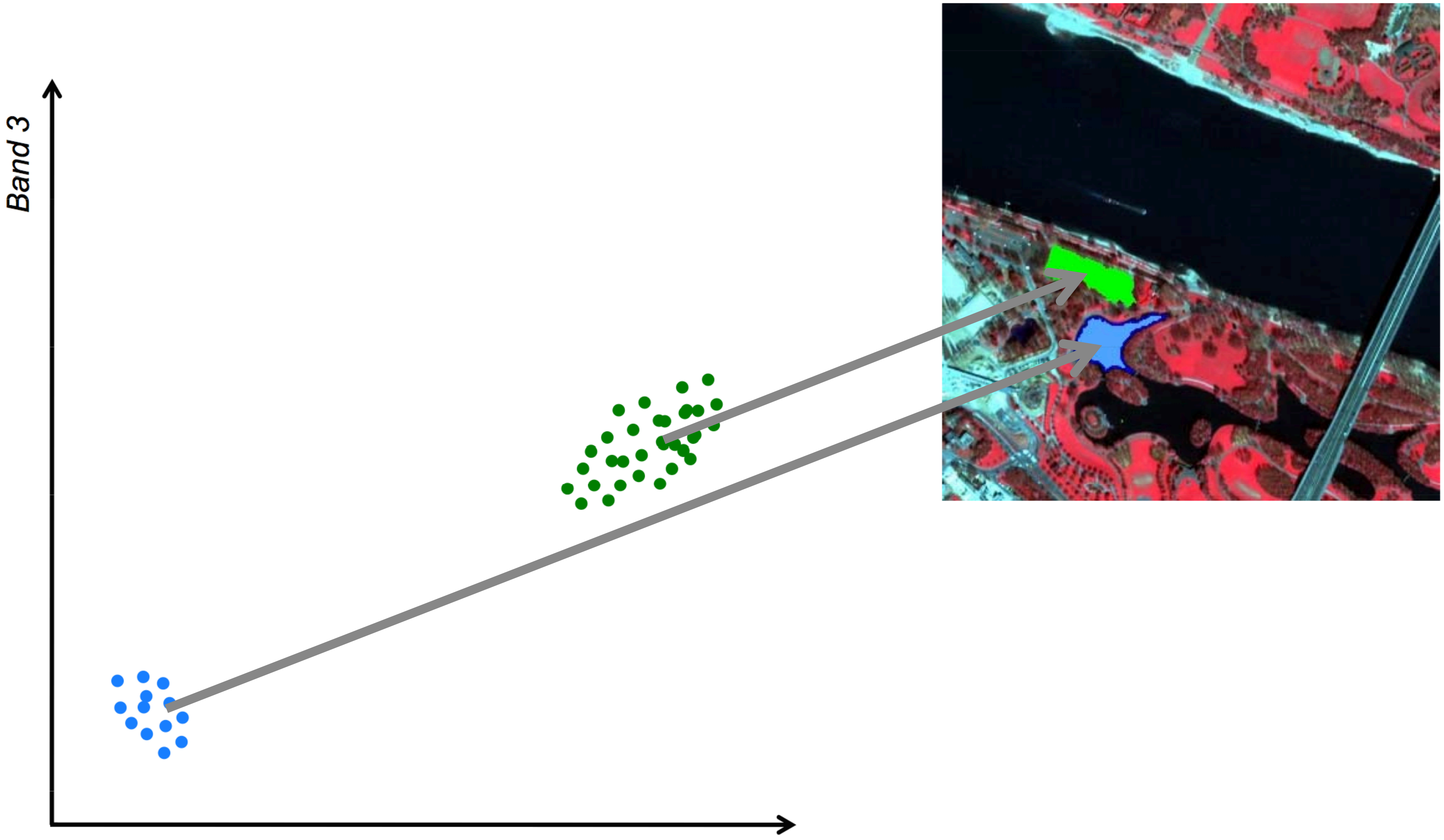
Example: Nearest Neighbor

- Classification in aerial (NIR) images
- Features: intensity values at two different channels
- 4 classes
- NN classifier based on the Euclidian distance



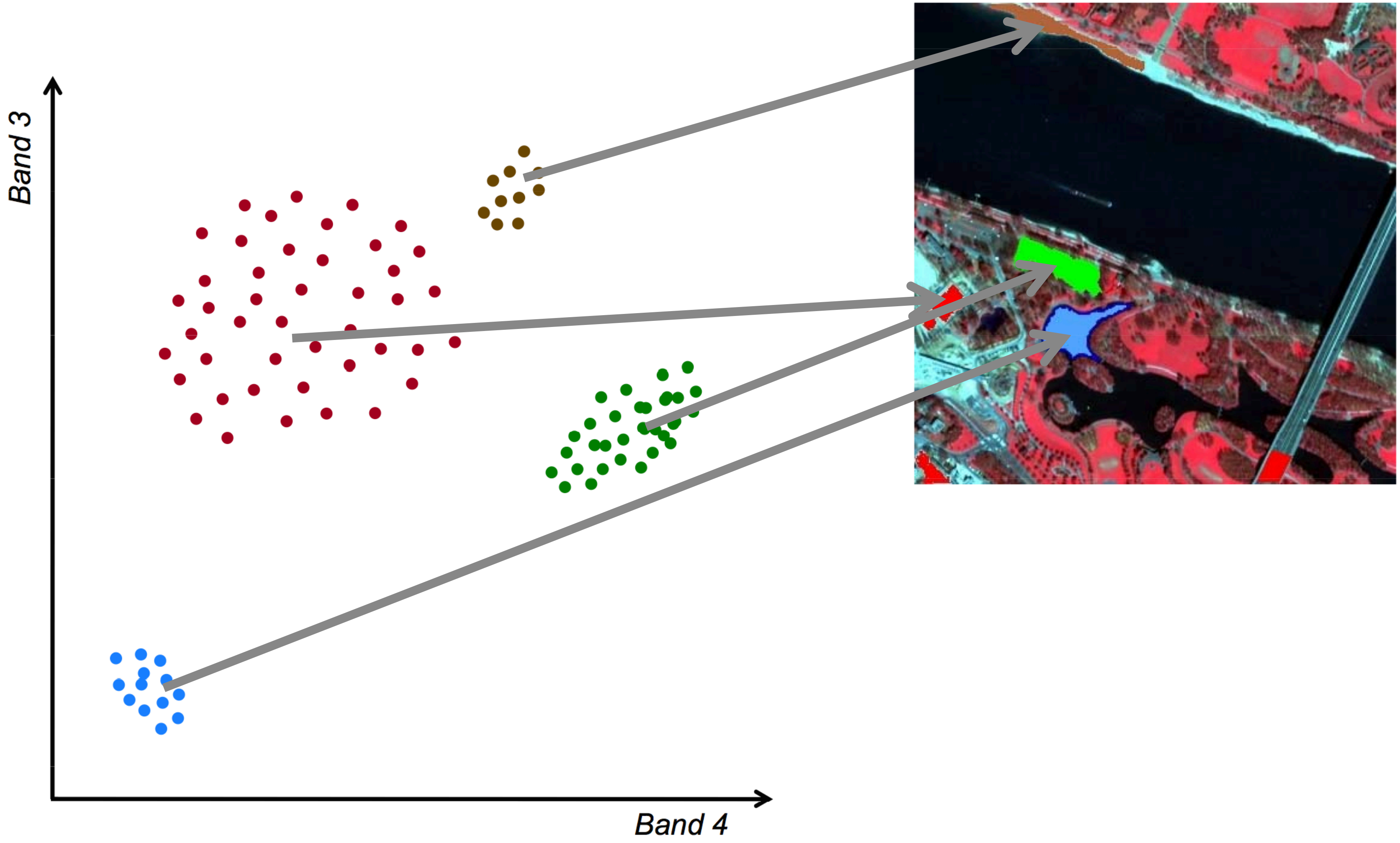
Slide courtesy: Waske/Stefanski

Example: Nearest Neighbor



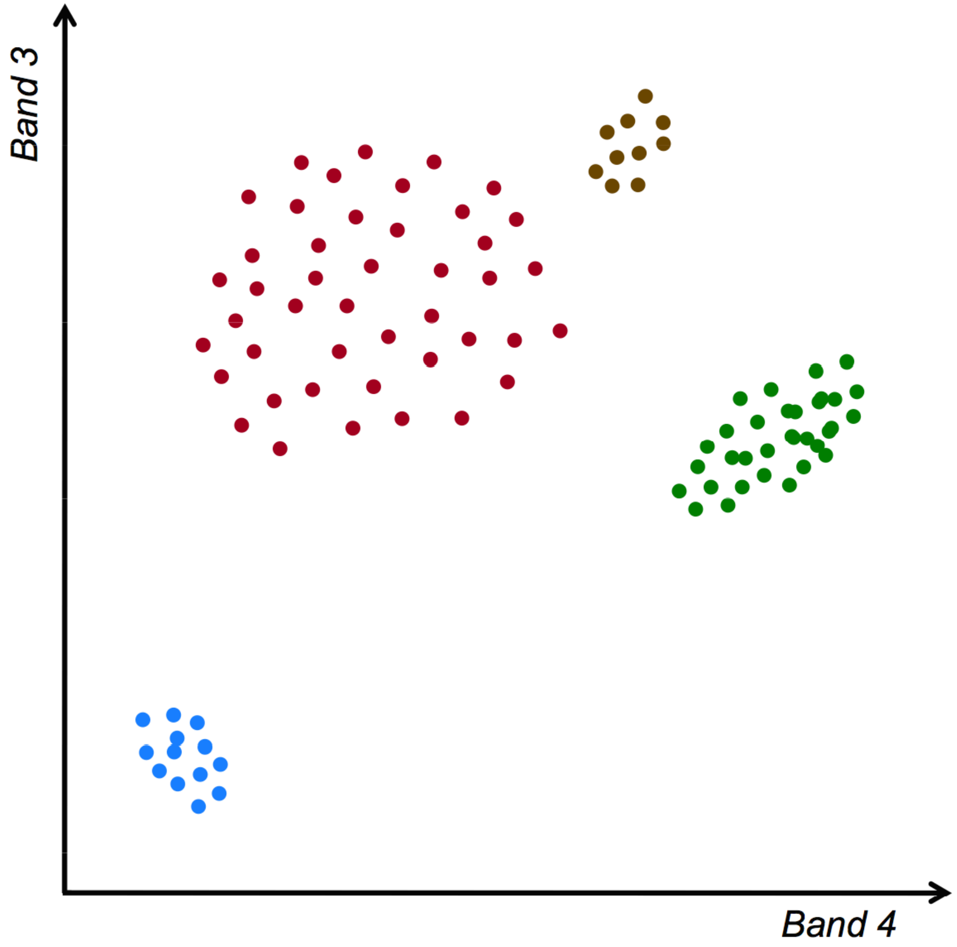
Slide courtesy: Waske/Stefanski

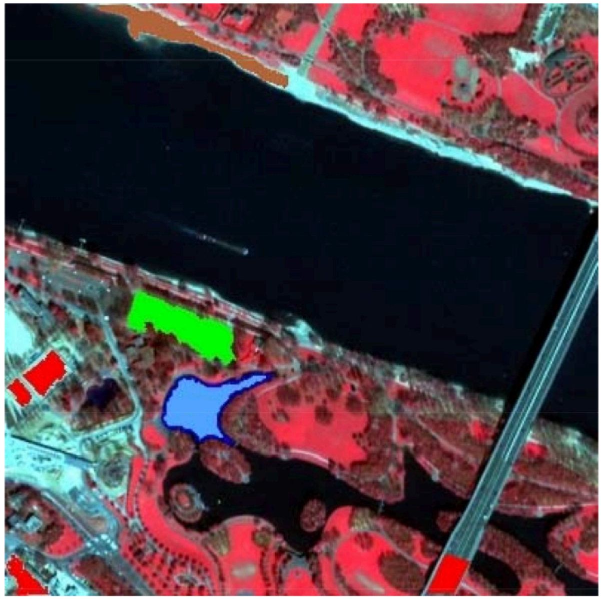
Example: Nearest Neighbor



Slide courtesy: Waske/Stefanski

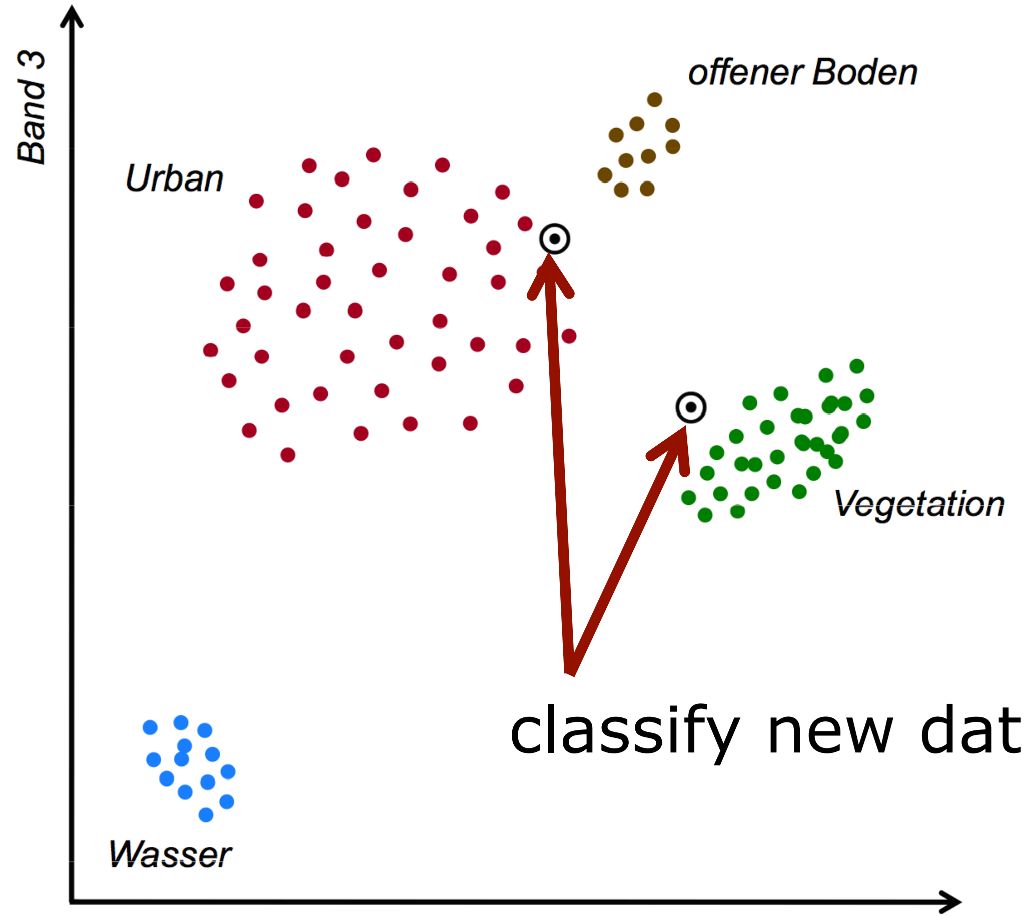
Learning is Completed!





Slide courtesy: Waske/Stefanski

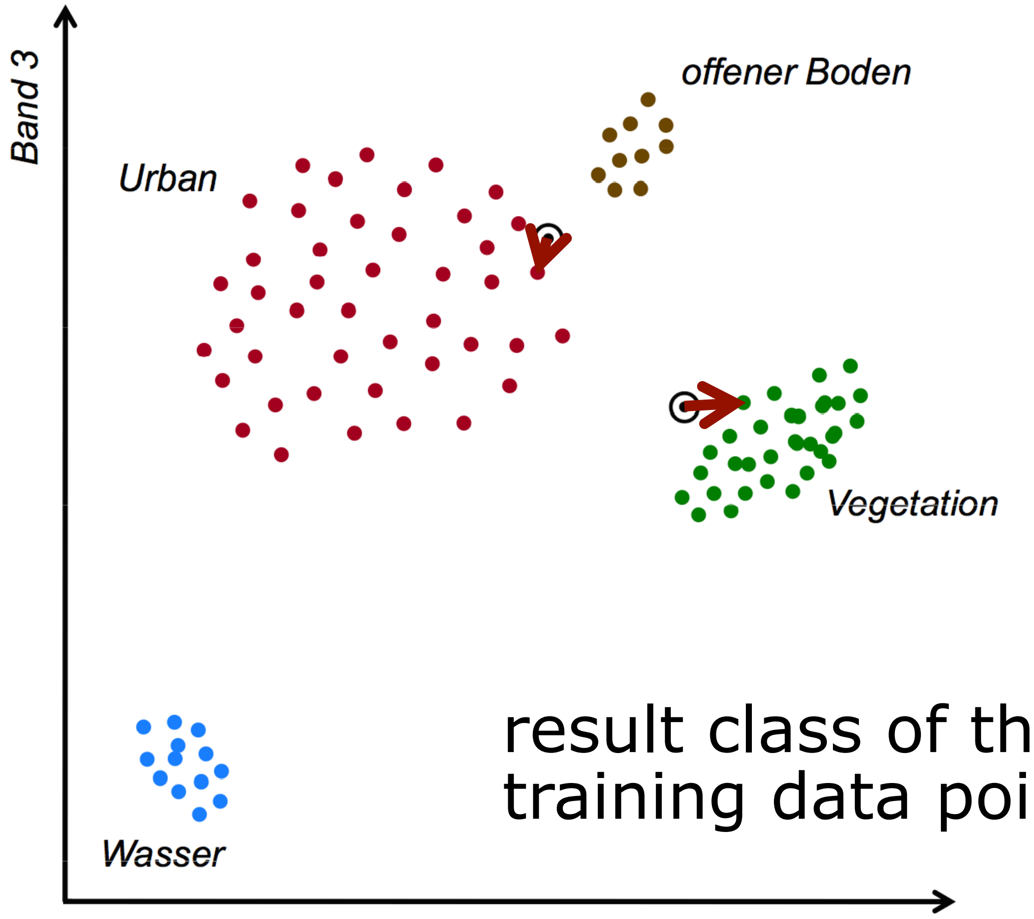
Example: Nearest Neighbor



classify new data points

Slide courtesy: Waske/Stefanski

Example: Nearest Neighbor



result class of the nearest neighbor training data point in feature space

Slide courtesy: Waske/Stefanski

Nearest Neighbor Approach

- Training data represents the distribution of features directly
- Problematic in the presence of noise
- Problematic for classes/features with different variances
- Often requires a densely sampled space
- Discriminant function is a Voronoi diagram

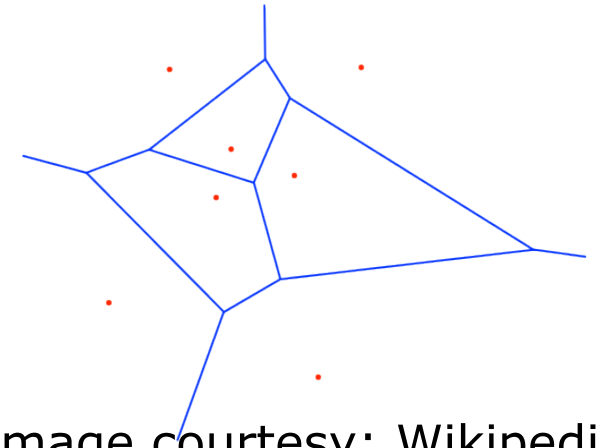


Image courtesy: Wikipedia

NN and k-NN Approach

NN:

- The feature distribution is modeled by the training data (or a subset)
- The class is **assigned** based on the **closest feature** in the training data

k-NN:

- The **k nearest neighbors are considered** for the classification decision (majority vote or weighted)

k-Nearest Neighbor Approach

- Robustified variant of NN
- Choice of k is often done heuristically
- Small k: less robust to noise
- Large k: may considers far away neighbors

Decision Trees

Decision Trees for Classification

- **Idea: sequences of splits** of the input space define **regions that correspond to classes**
- Hierarchical data structure realizing a divide-and-conquer strategy
- Setup of the tree through training data
- Efficient nonparametric method for classification (and regression)

Decision Tree Example

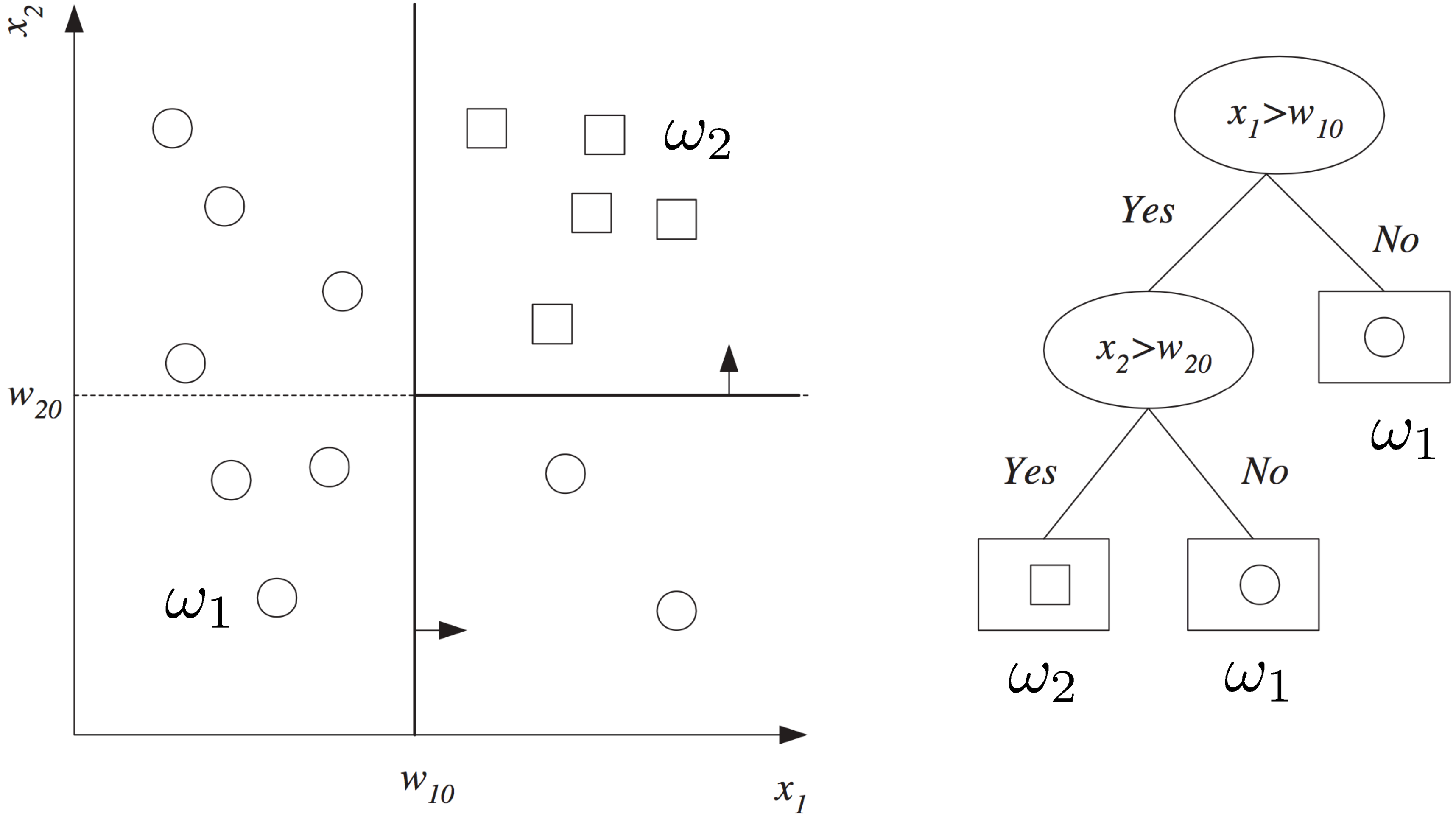
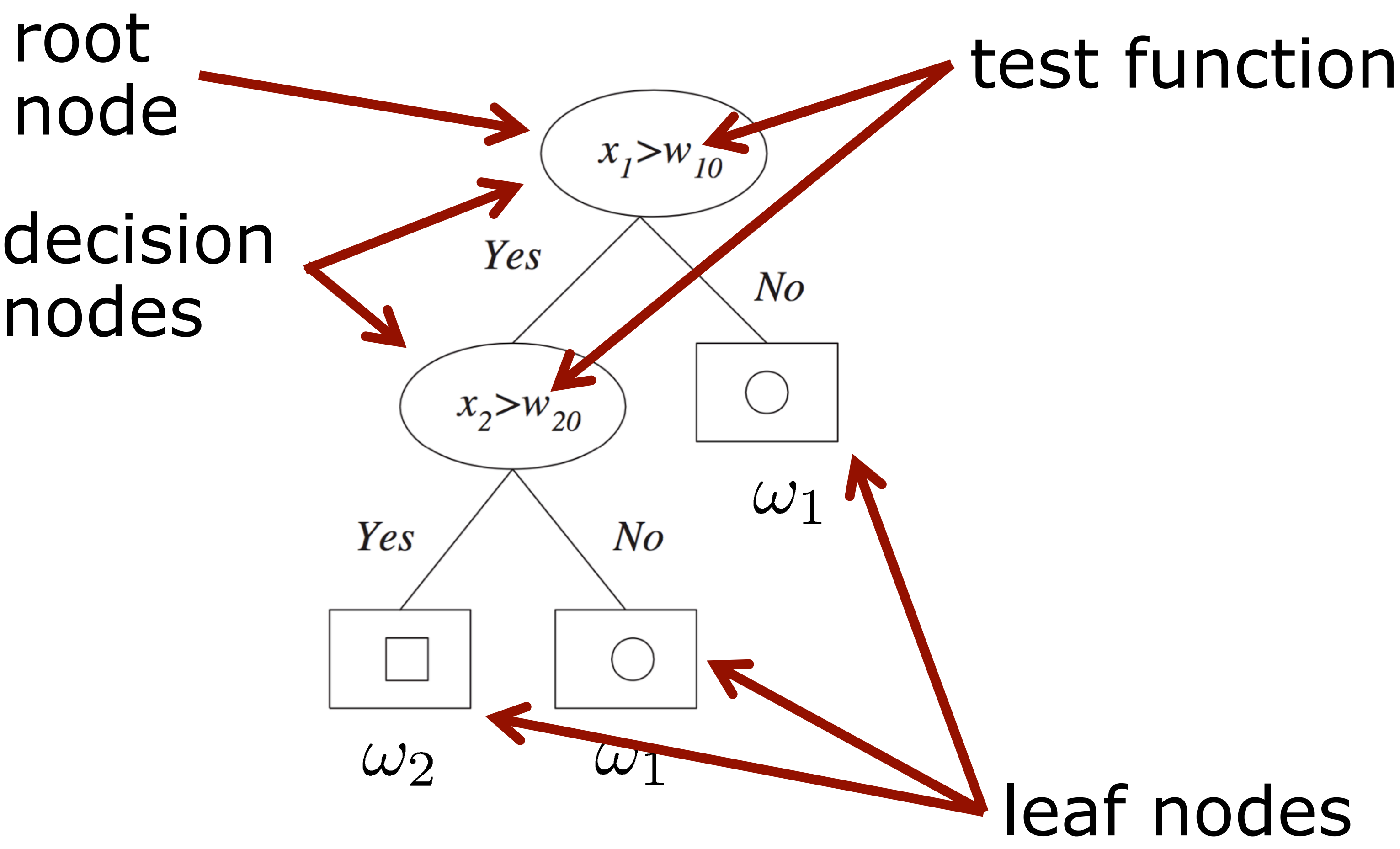


Image courtesy: Aplaydin

Elements of a Decision Tree



root node

decision nodes

test function

leaf nodes

Image courtesy: Aplaydin

Decision Nodes

- Each **decision node** implements a **test function** with discrete outcomes
- The test function of each decision node splits the input space into regions
- Also called split node

Leaf Nodes

- A **leaf** node symbolizes the **end of a sequence of decisions**
- A single (output) class is associated to leach leaf node
- A leaf node defines a localized region in the input space where instances falling in this region have the same label

Classification for a Given Decision Tree

1. Start at the root node
2. If current node is a leaf node, return its class label
3. Perform the test of the current decision node and follow the corresponding branch
4. Goto 2

Learning a Decision Tree

- The order in which split decisions are made influences the complexity and performance of the tree
- Finding the **optimal arrangement** of tests is **NP hard**, thus heuristics are needed

Question: What would be a good strategy to arrange the split nodes?

Which Decision to Make Next?

- Select the **test that best separates** the class labels in the data
- The “purer” the children, the better the split
- For any “pure” child branch, we can create a leaf node (no further splits)

Impurity

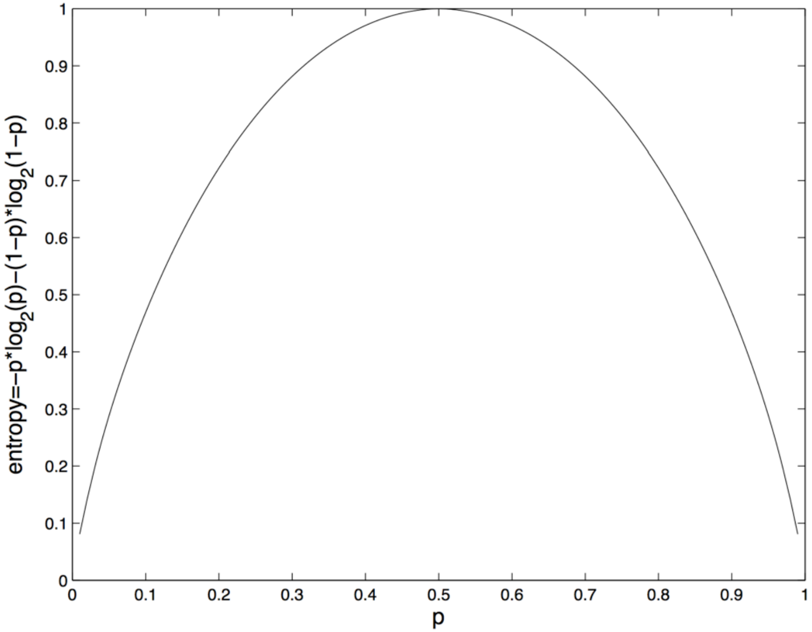
- Purity (or impurity) can be defined through the **uncertainty** in the distribution **over the class label** in the current vs. the split-up region
- **Goal:** Always select the split that **minimizes impurity**
- Different impurity measures:
  - Entropy
  - Gini index
  - …

Entropy as Impurity

- Entropy over the classes

$$H(\Omega) \quad = \quad - \sum_{\omega_i \in \Omega} p(\omega_i) \, \ln p(\omega_i)$$

- For two classes



Entropy as Impurity

- Entropy over the classes

$$H(\Omega) \quad = \quad -\sum_{\omega_i \in \Omega} p(\omega_i)\, \ln p(\omega_i)$$

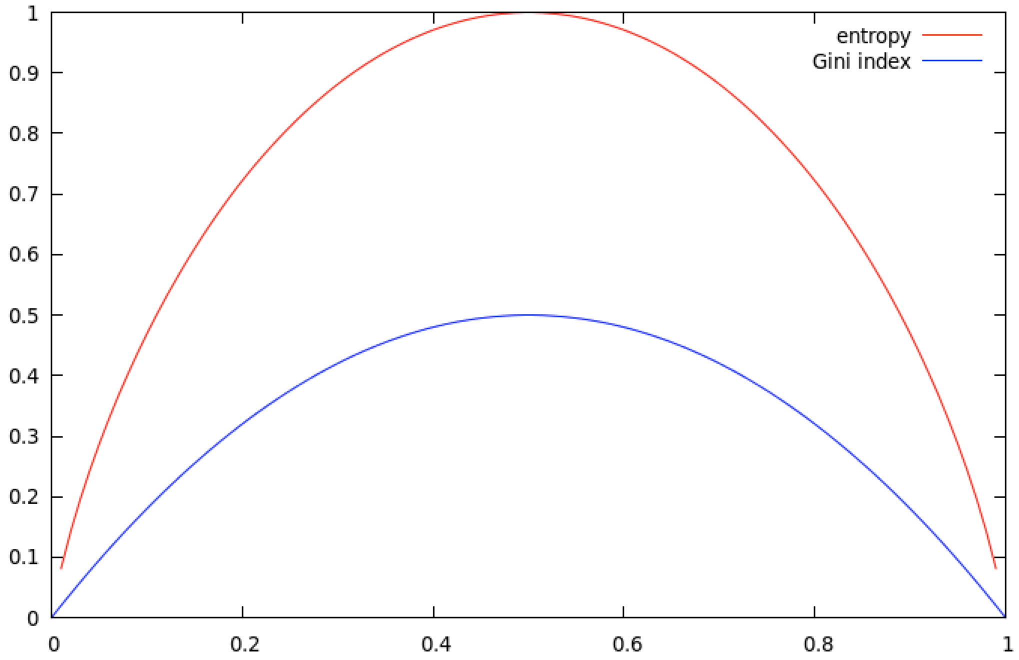
- Select the split that reduces the entropy at most

$$\underbrace{\Delta H}_{\substack{\text{change in}\\ \text{entropy}}} \quad = \quad \underbrace{H(\Omega)}_{\substack{\text{entropy}\\ \textbf{before}\\ \text{the split}}} - \underbrace{(p_{\text{left}} H_{\text{left}}(\Omega) + (1 - p_{\text{left}}) H_{\text{right}}(\Omega))}_{\substack{\text{entropy}\\ \textbf{after}\\ \text{the split}}}$$

Gini Index

- Alternative criterion to entropy
- Gini index

$$G(\Omega) = 1 - \sum_{\omega_i \in \Omega} p(\omega_i)^2 = 1 - p(\omega_l)^2 - p(\omega_r)^2$$



very similar!

When to Stop?

- **Intuitive idea: add a leaf node after a split leads to a pure node**

What could be problematic about this strategy?

When to Stop?

- Intuitive idea: add a leaf note after a split leads to a pure node
- **Overfitting problem**: the tree perfectly splits the classes **on the training dataset but does not generalize** well to other datasets
- Standard approach: stop after a certain level of purity is reached
- A leaf stores the posterior probabilities of classes, instead of best label

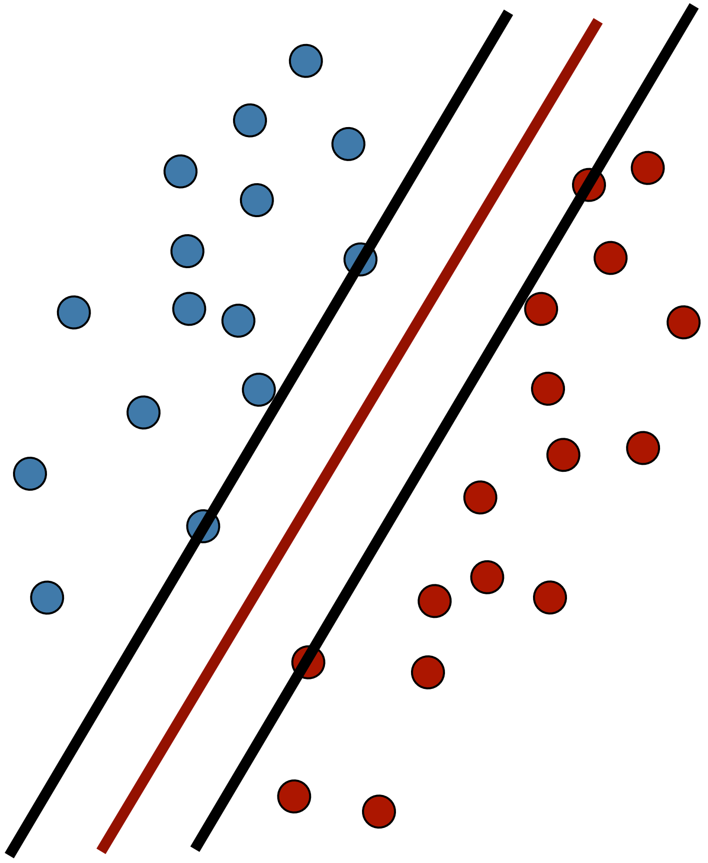
Decision Trees for Classification

- Comparably **easy to understand** and implement
- Works well to **high-dimensional** data
- Allow to handle **numerical** and **categorical variables** easily
- Finding the optimal split is NP hard
- Heuristics are used (e.g., entropy)

Support Vector Machines

Support Vector Machines

- How to select the classifier with the best generalization performance?



Key idea: select the hyperplane that maximizes the margin between both classes

Support Vector Machines

- SVMs seek to maximize the margin between both classes
- Search for the separating hyperplane is formulated as a convex optimization problem
- Optimal solution for computing the hyperplane

Support Vector Machines

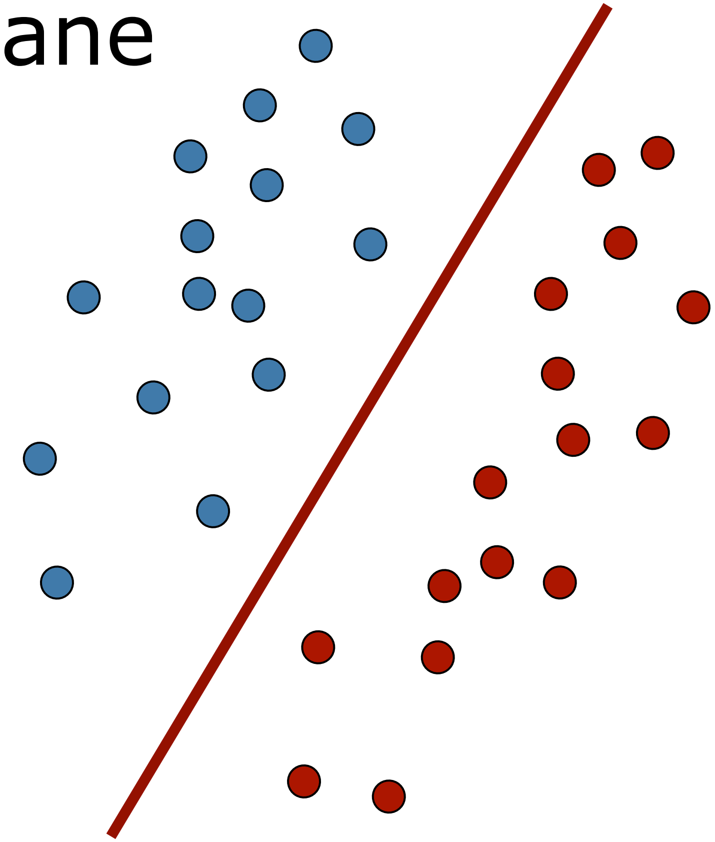
- We assume linearly separable data
- Linearly separating plane can be written as

$$\boldsymbol{a}^{\top}\boldsymbol{e} + b = 0$$

weight vector

data

threshold ("bias")



Support Vector Machines

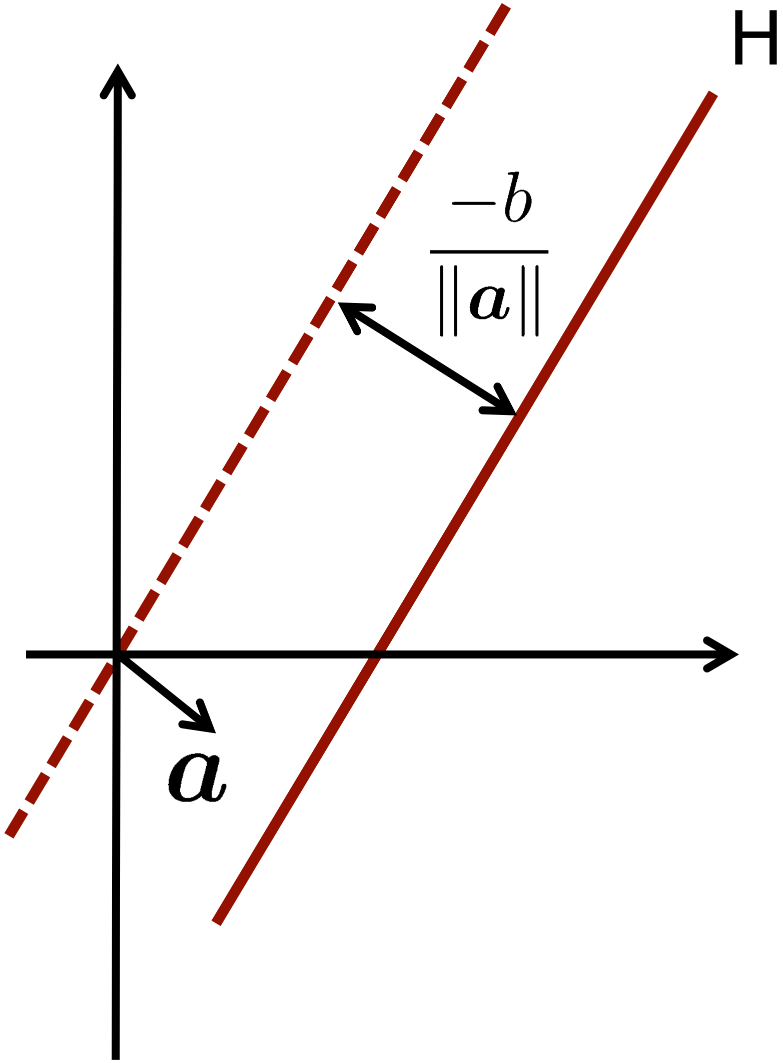
- Linearly separating plane

$$\boldsymbol{a}^{\mathsf{T}}\boldsymbol{e} + b = 0$$

weight vector

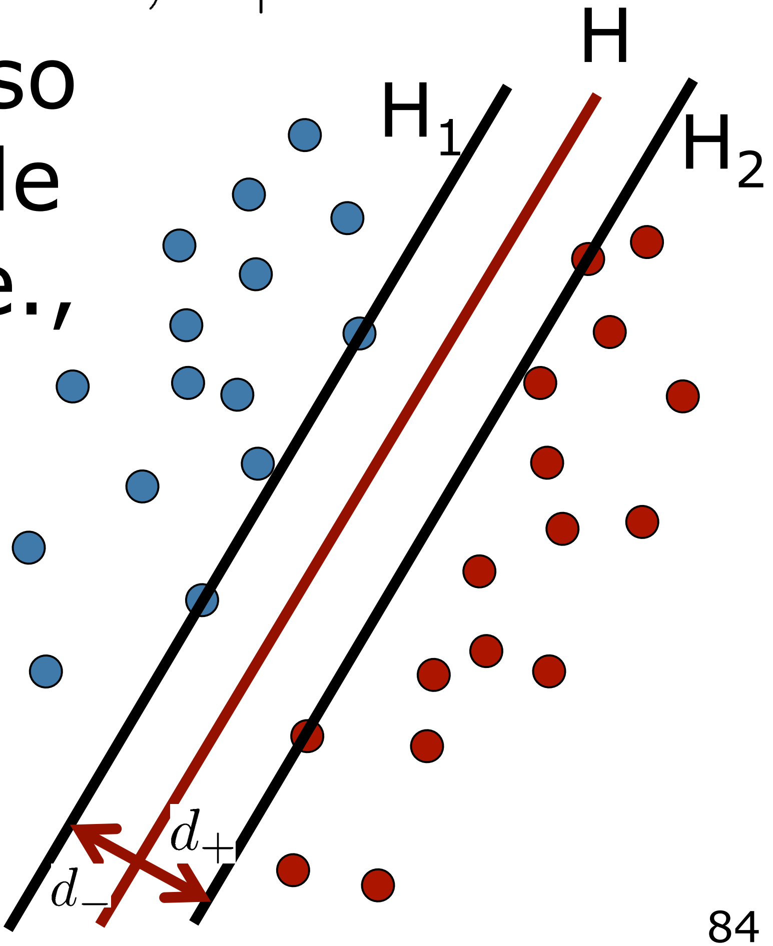
data

threshold ("bias")



Support Vector Machines

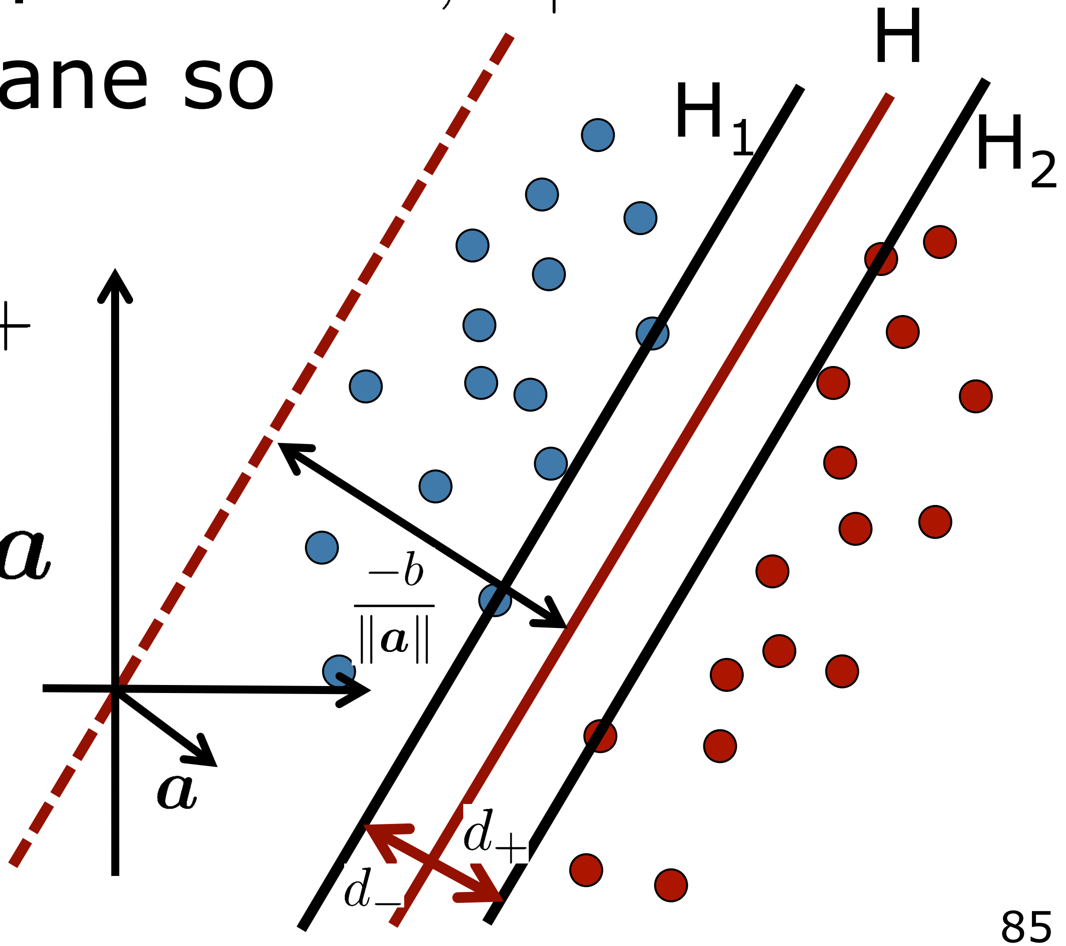
- Distance of the closest neg./pos. example to the plane: d_-, d_+
- Choose hyperplane H so that it lies in the middle between H_1 and H_2, i.e., $d_- = d_+$
- Margin: $d_- + d_+$



Margin

- Distance of the closest neg./pos. example to the plane: d_-, d_+
- Choose hyperplane so that $d_- = d_+$
- Margin: $d_- + d_+$
- **We can scale $\boldsymbol{a}$ so that:**

$$d_- = d_+ = 1/\|\boldsymbol{a}\|$$



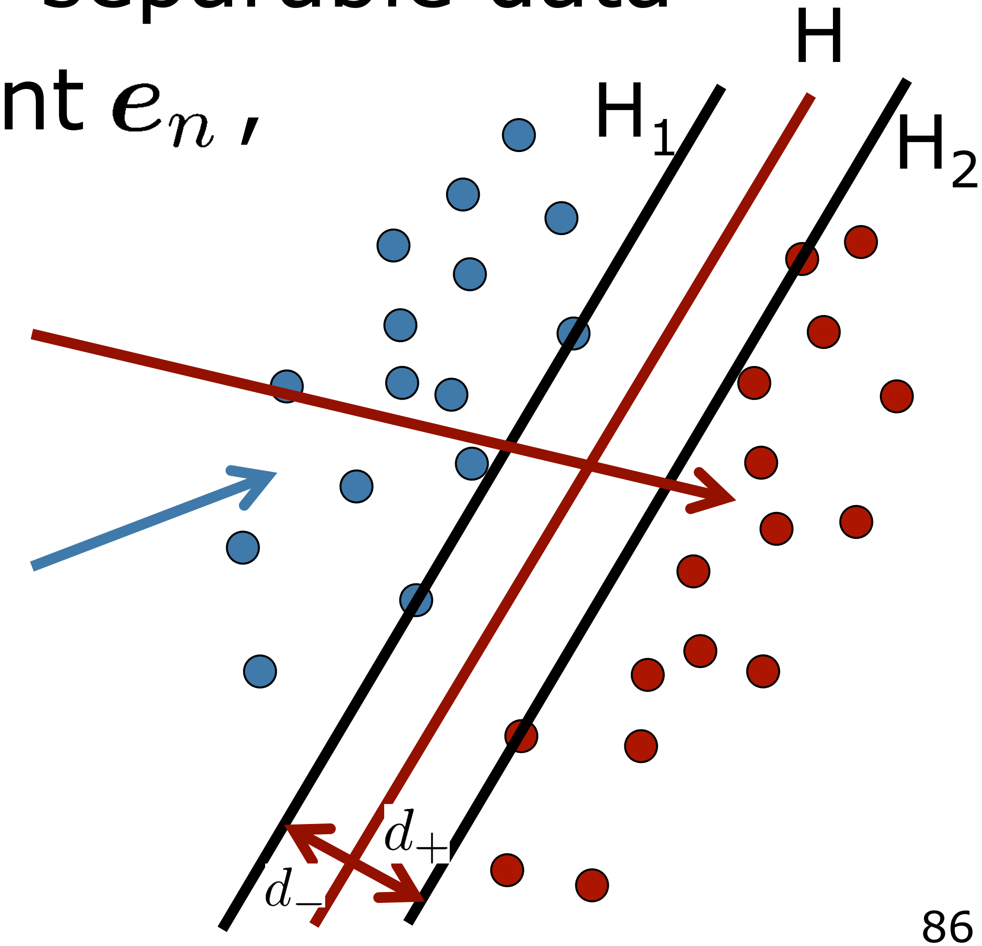
Constrains

- Using the scaling $d_- = d_+ = 1/\|\boldsymbol{a}\|$
- Assuming linearly separable data
- For each data point $\boldsymbol{e}_n$, we can write

  $$\boldsymbol{a}^\mathsf{T}\boldsymbol{e}_n + b \geq +1$$

  or

  $$\boldsymbol{a}^\mathsf{T}\boldsymbol{e}_n + b \leq -1$$



Support Vectors

- The separating hyperplane is define by the **support vectors**
- For each support vector, we can write

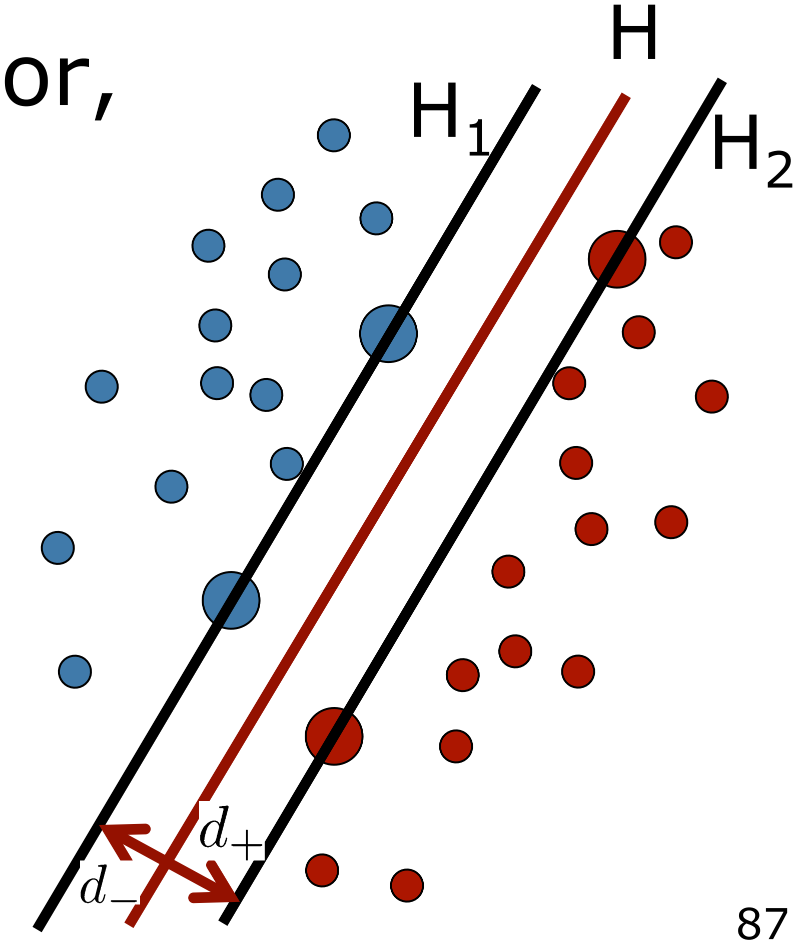
$$\omega_n(\boldsymbol{a}^\mathsf{T}\boldsymbol{e}_n + b) = 1$$

with the class label

$$\omega_n \in \{-1, +1\}$$

- Margin

$$d_- + d_+ = 2/\|\boldsymbol{a}\|$$



SVM Optimization Problem

- Find hyperplane that maximizes the margin

$$\arg\min_{\boldsymbol{a},b} \|\boldsymbol{a}\|^2$$

- with the constraints

$$\omega_n(\boldsymbol{a}^\top \boldsymbol{e}_n + b) \geq 1 \quad \forall n$$

- Can be solved through quadratic programming with linear constraints.

SVM Testing

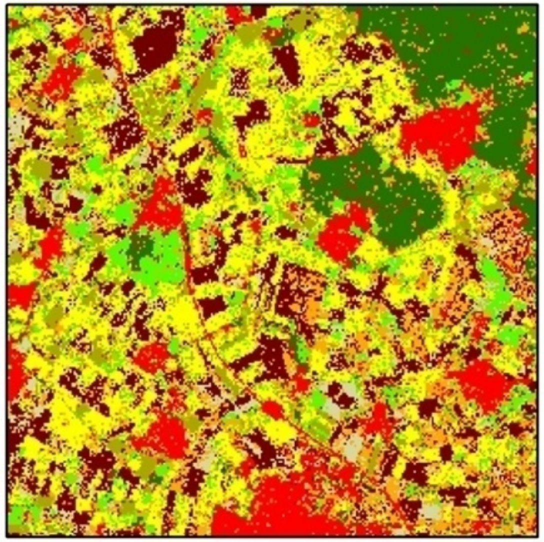
- Classifying an new data point just requires to test on which side of the hyperplane the points is located

$$\omega_{new} = \mathrm{sign}\left(\boldsymbol{a}^\top \boldsymbol{e}_{new} + b\right)$$

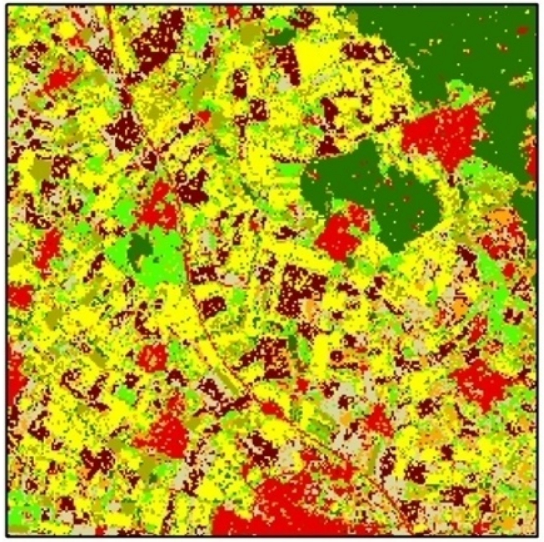
Linear Separable?

- Introduce “some tolerance” for data points are not perfectly separable
(done via slack variables)
- Kernel-trick: move to a different, high-dimensional space
(done via kernel functions)

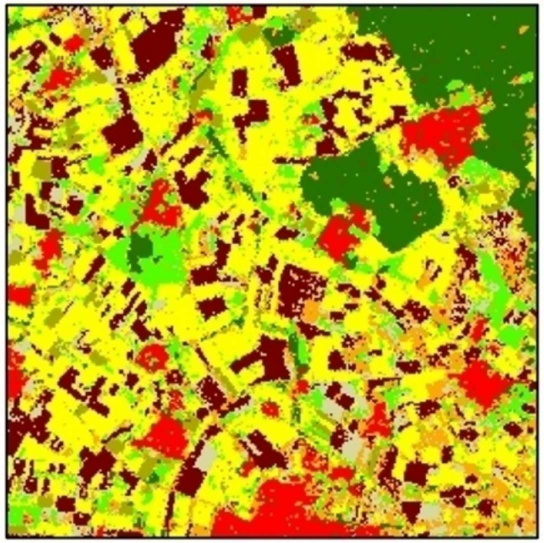
Classification Examples



ML: 64.8% correct



DT: 61.2% correct



SVM: 72.6% correct

- 🟧 Baumschulen / Obstbau
- 🟨 Getreide
- 🟩 Grünland
- 🟫 Hackfrüchte
- 🟫 Raps
- 🟥 Siedlung
- ⬜ Sonderkulturen
- 🟩 Wald

- SVMs often outperform ML and DT approaches for multisensor RS data
- SVMs often yield more homogenous classification results

[Waske & Benediktsson'07]

Maximum-A-Posteriori Classification

Maximum-A-Posteriori (MAP) Classification

- Classification is decision making
- Probability theory as the framework for making decisions under uncertainty
- Based on **Bayes' rule**

Maximum-A-Posteriori (MAP) Classification

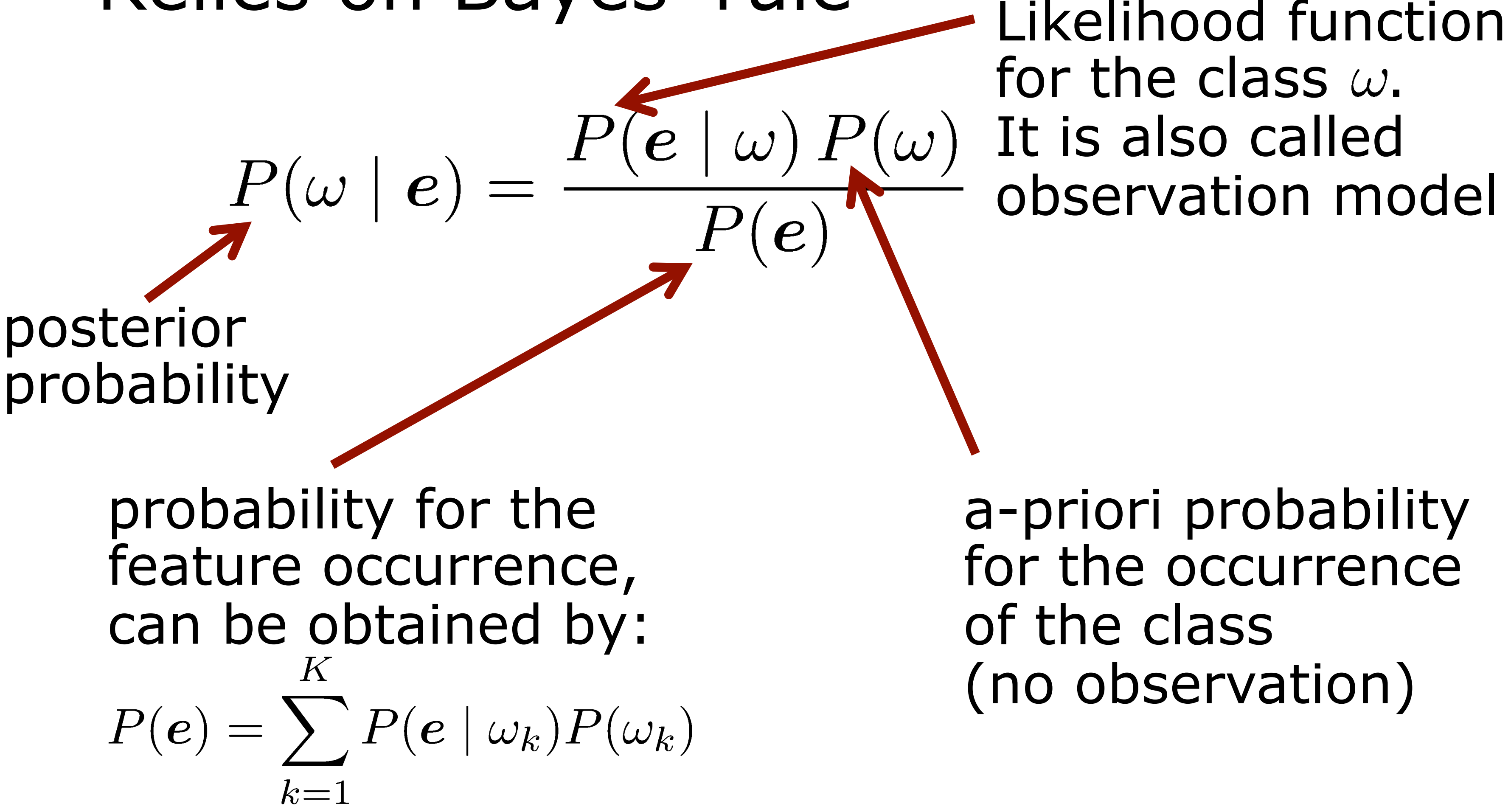
- MAP relies on Bayes' rule:

$$P(\underset{\substack{\uparrow\\ \text{class}}}{\omega} \mid \underset{\substack{\uparrow\\ \text{feature}}}{\boldsymbol{e}}) = \frac{P(\boldsymbol{e} \mid \omega)\, P(\omega)}{P(\boldsymbol{e})}$$

- Answers: "What is the probability of a class given an observed feature?"

Maximum-A-Posteriori (MAP) Classification

- Relies on Bayes' rule



$$P(\omega \mid \boldsymbol{e}) = \frac{P(\boldsymbol{e} \mid \omega)\, P(\omega)}{P(\boldsymbol{e})}$$

Likelihood function for the class ω. It is also called observation model

posterior probability

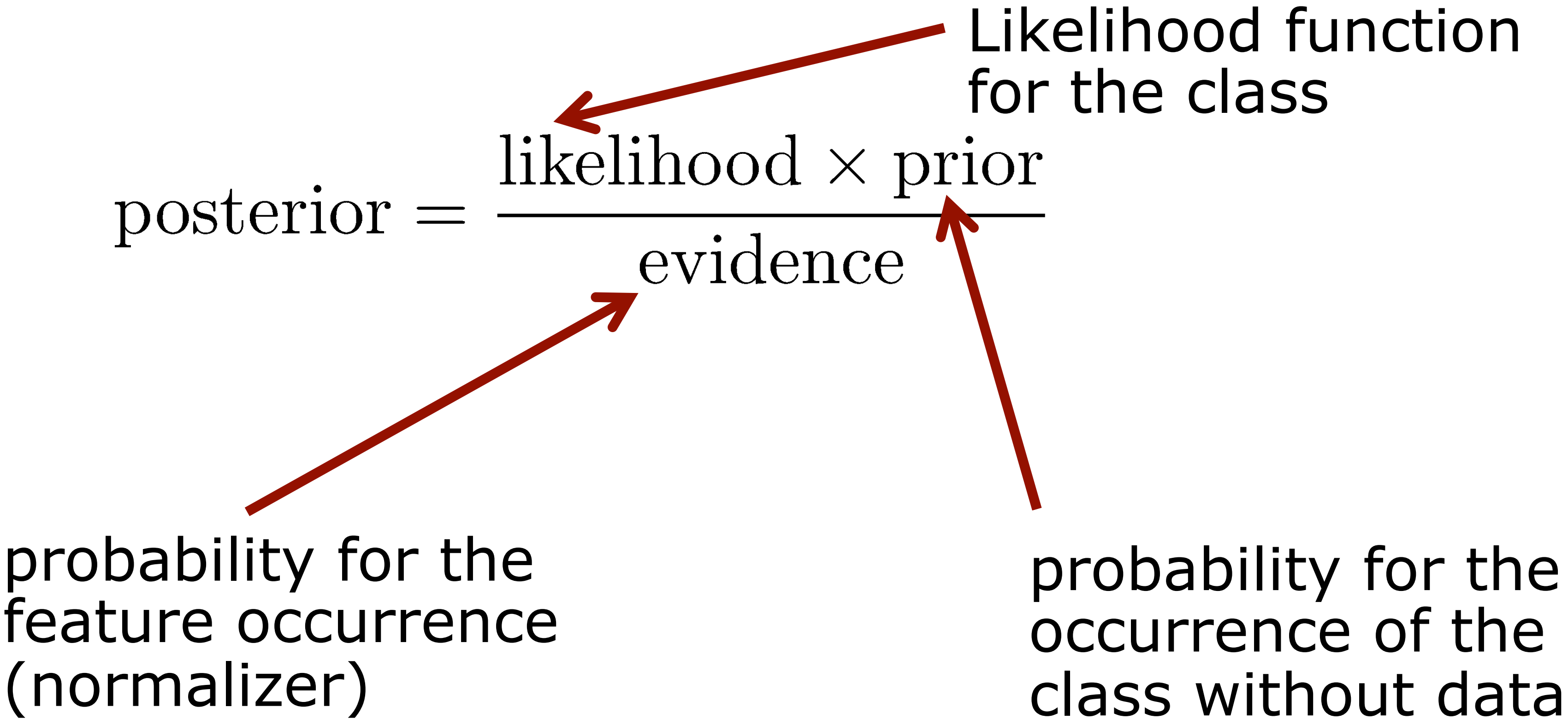
probability for the feature occurrence, can be obtained by:

$$P(\boldsymbol{e}) = \sum_{k=1}^{K} P(\boldsymbol{e} \mid \omega_k) P(\omega_k)$$

a-priori probability for the occurrence of the class (no observation)

Maximum-A-Posteriori (MAP) Classification

- Relies on Bayes' rule



$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

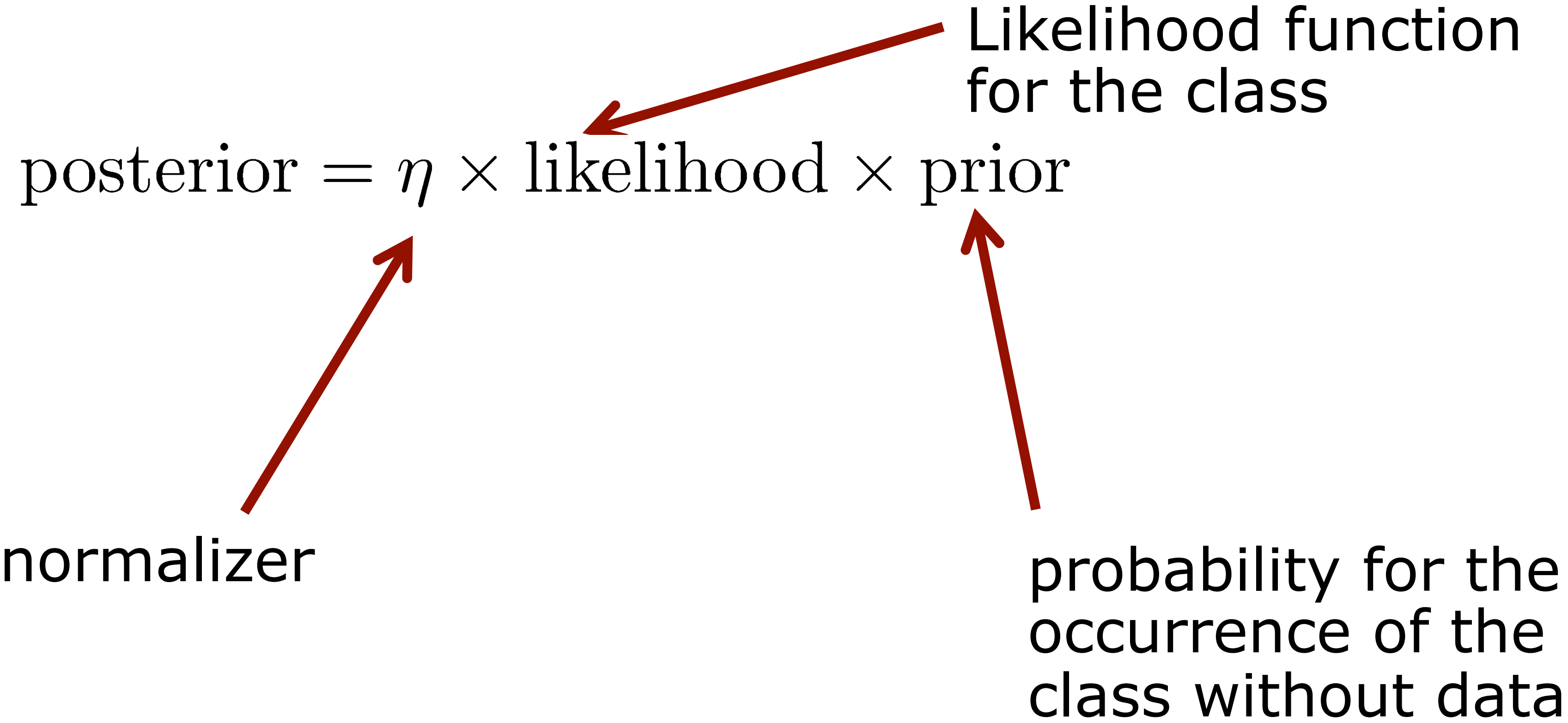
Likelihood function for the class

probability for the feature occurrence (normalizer)

probability for the occurrence of the class without data

Maximum-A-Posteriori (MAP) Classification

- Relies on Bayes’ rule



Likelihood function for the class

$$\text{posterior} = \eta \times \text{likelihood} \times \text{prior}$$

normalizer

probability for the occurrence of the class without data

Probability Distributions

- $P(\omega|e)$: Class probability given the observed feature
- $P(e|\omega)$: Sensor model: the probability of observing a feature given the class
- $P(\omega)$: A-priori probability for the occurrence of the class
- $P(e)$: Normalizer

Distributions $P(e|\omega)$ and $P(\omega)$ must be learned from training data!

MAP Classification

1. Compute for each class

$$P(\omega_i \mid \boldsymbol{e}) = \frac{P(\boldsymbol{e} \mid \omega_i)\, P(\omega_i)}{\sum_{k=1}^{K} P(\boldsymbol{e} \mid \omega_k) P(\omega_k)}$$

← identical for all classes

(thus it can be ignored)

2. Select the MAP class

$$\omega_{i^*} \quad \text{with} \quad i^* = \arg\max_i P(\omega_i \mid \boldsymbol{e})$$

Losses and Risks

- **What if decisions are not equally good?**

Losses and Risks

- What if decisions are not equally good?
- Definition of the risk of an action

$$R(a_i \mid \boldsymbol{e}) = \sum_{k=1}^{K} \lambda_{ik} P(\omega_k \mid \boldsymbol{e})$$

action of classifying the class ω_i

loss when classifying as ω_i if the class is ω_k

- Select the action a_{i^*} that minimizes the risk: a_{i^*} with $i^* = \arg\min_i R(a_i|\boldsymbol{e})$

0/1 Loss

- Under a 0/1 loss, i.e. $\lambda_{ik} = \begin{cases} 0 & \text{if } i = k \\ 1 & \text{if } i \neq k \end{cases}$
- We minimize the risk

$$\begin{aligned} R(a_i \mid \boldsymbol{e}) &= \sum_{k=1}^{K} \lambda_{ik} P(\omega_k \mid \boldsymbol{e}) \\ &= \sum_{k \neq i} P(\omega_k \mid \boldsymbol{e}) \\ &= 1 - P(\omega_i \mid \boldsymbol{e}) \end{aligned}$$

- by selecting the MAP class (expected result)

Rejecting All Classes

- For most applications, it is useful to reject an action (a_0) in case of doubt
- Loss:

$$\lambda_{ik} = \begin{cases} 0 & \text{if } i = k \\ \lambda & \text{if } i = 0 \\ 1 & \text{otherwise} \end{cases} \qquad 0 < \lambda < 1$$

- Risk: $R(a_i \mid \boldsymbol{e}) = \begin{cases} \lambda & \text{if } i = 0 \\ 1 - P(\omega_i \mid \boldsymbol{e}) & \text{otherwise} \end{cases}$
- Choose: a_{i^*} with $i^* = \arg\min_i R(a_i \mid \boldsymbol{e})$

Example

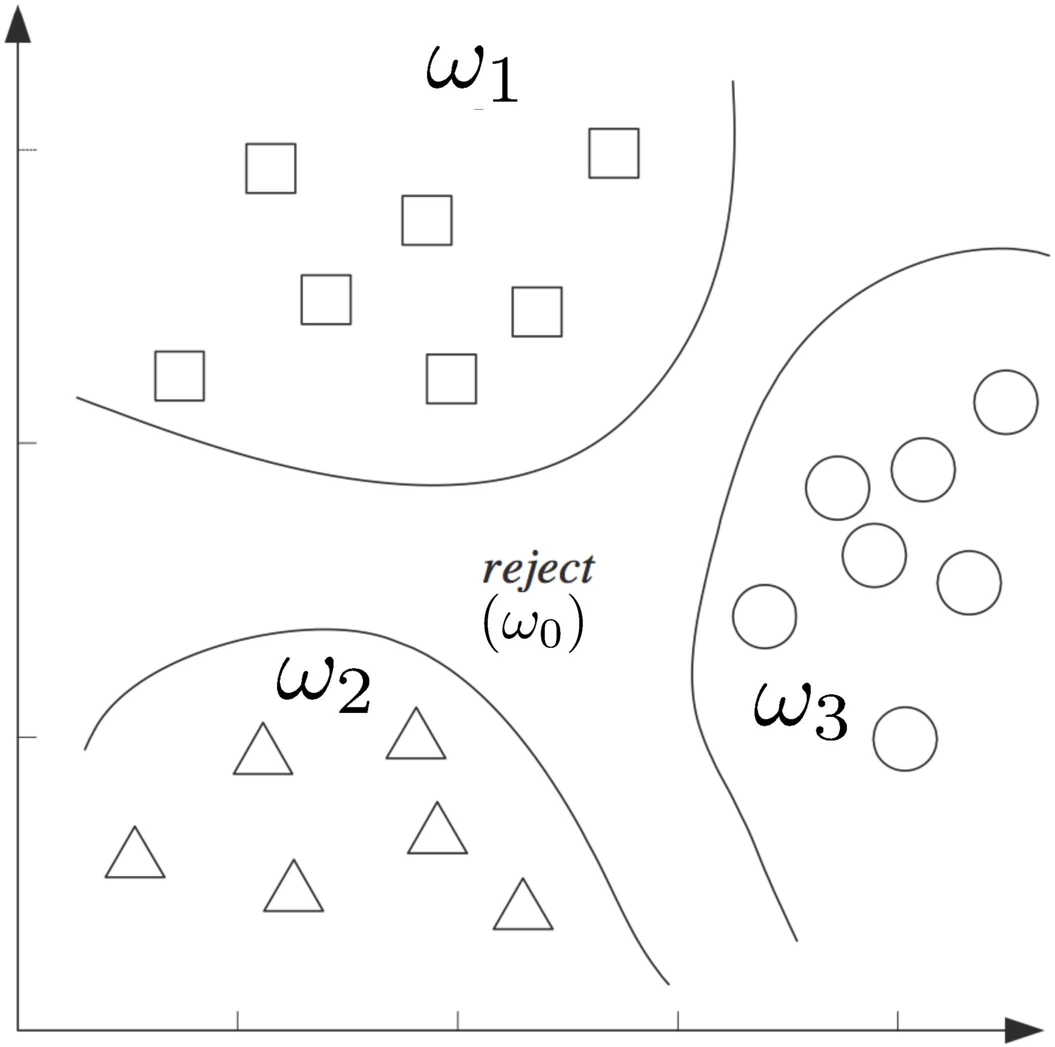
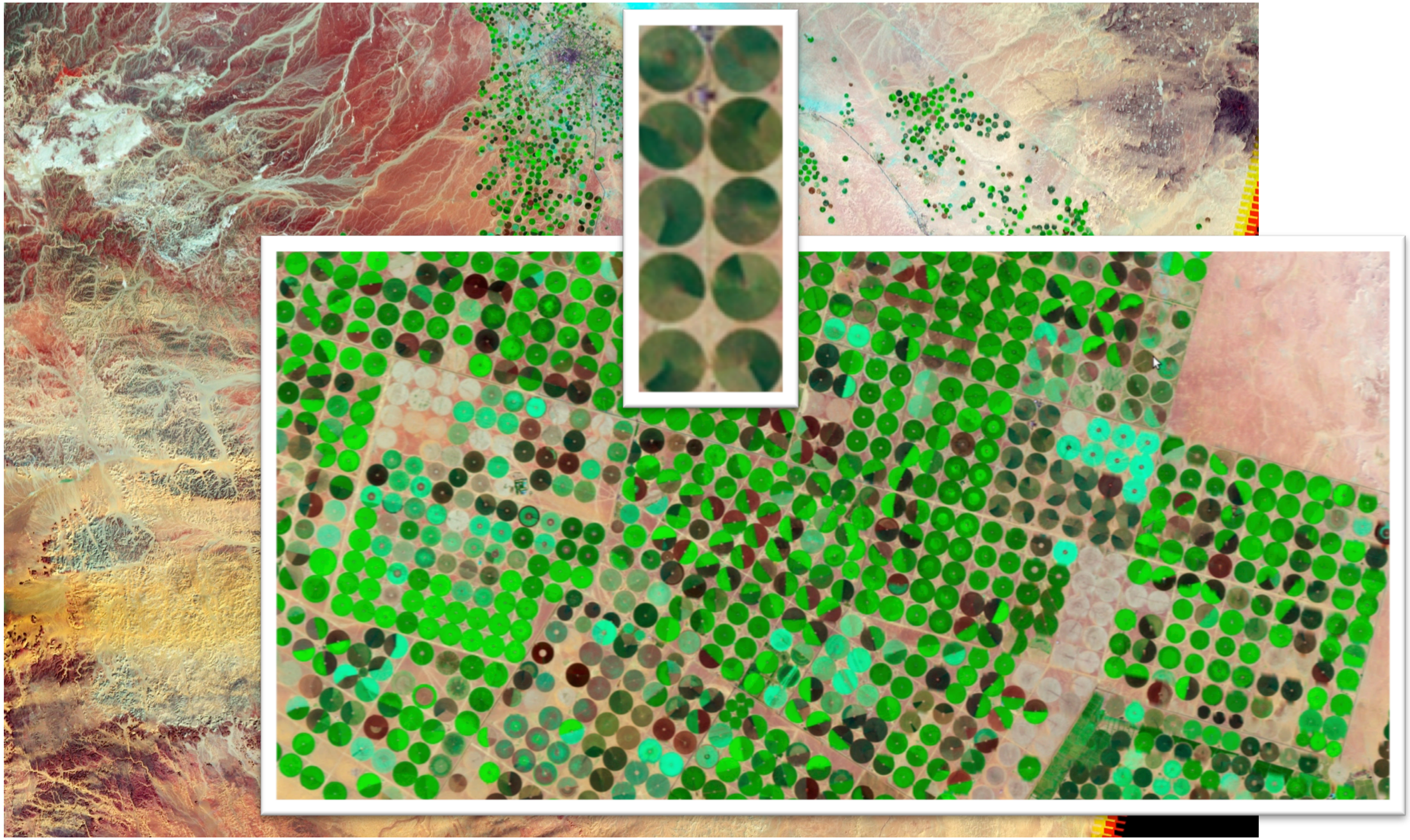


Image courtesy: Aplaydin

Remote Sensing Example: Land Cover Classification



Slide courtesy: Roscher

Remote Sensing Example: Input and Annotations

Features: RGB

red = arable land;
blue = desert

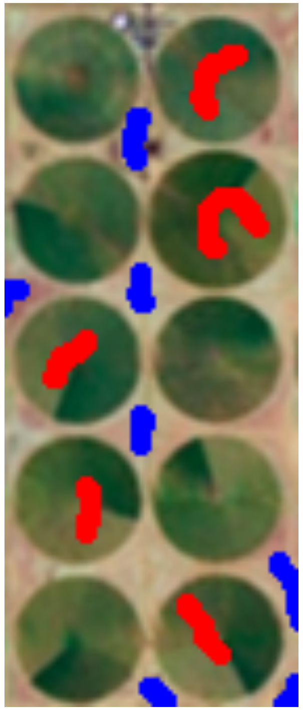


Image courtesy: Roscher

Remote Sensing Example: Feature Space

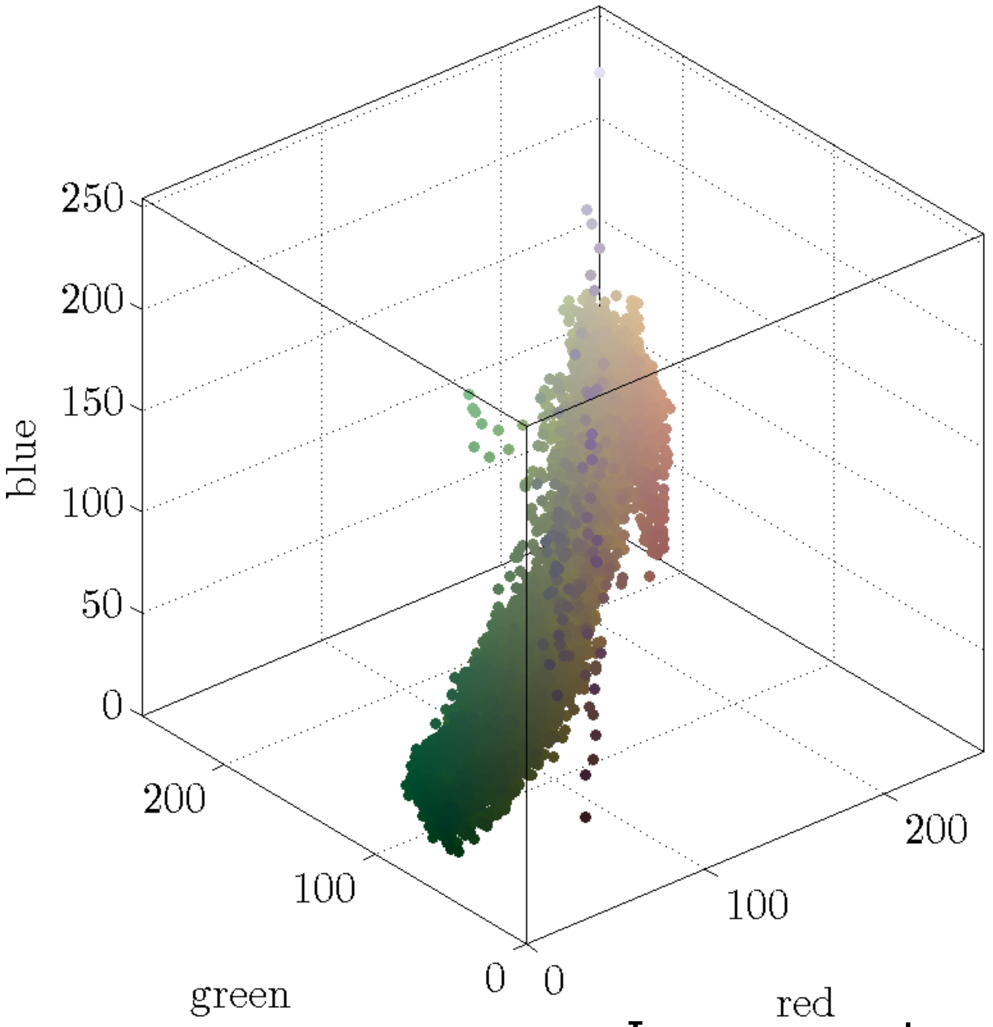


Image courtesy: Roscher

Remote Sensing Example: Training Data Points

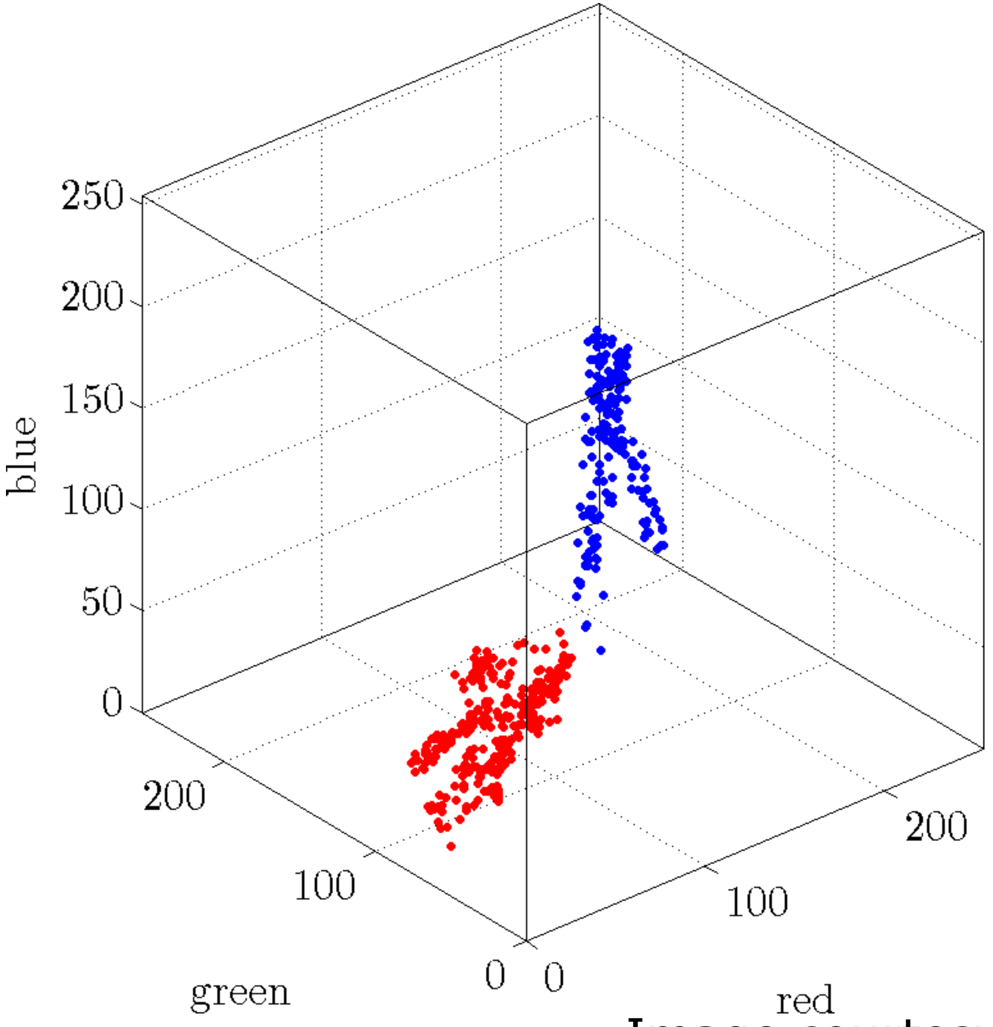


Image courtesy: Roscher

Remote Sensing Example: Likelihood Function (3D Gauss.)

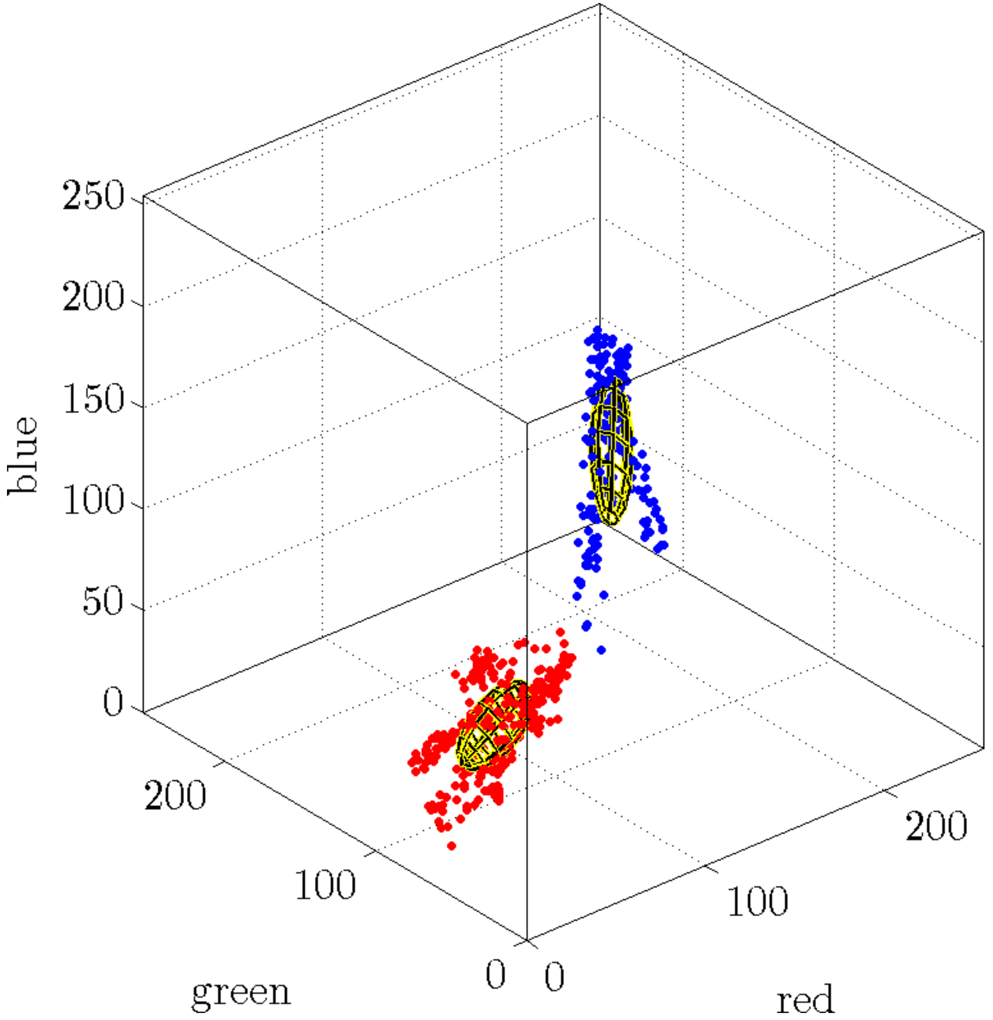


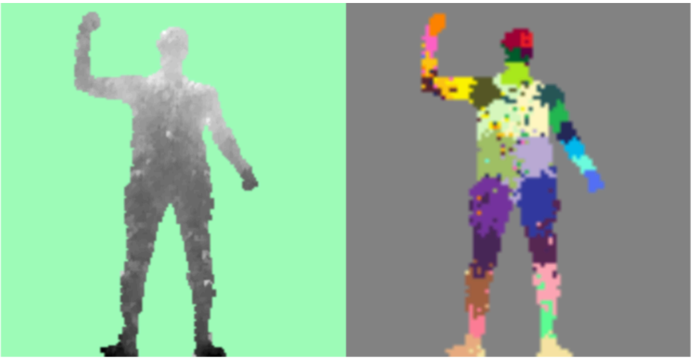
Image courtesy: Roscher

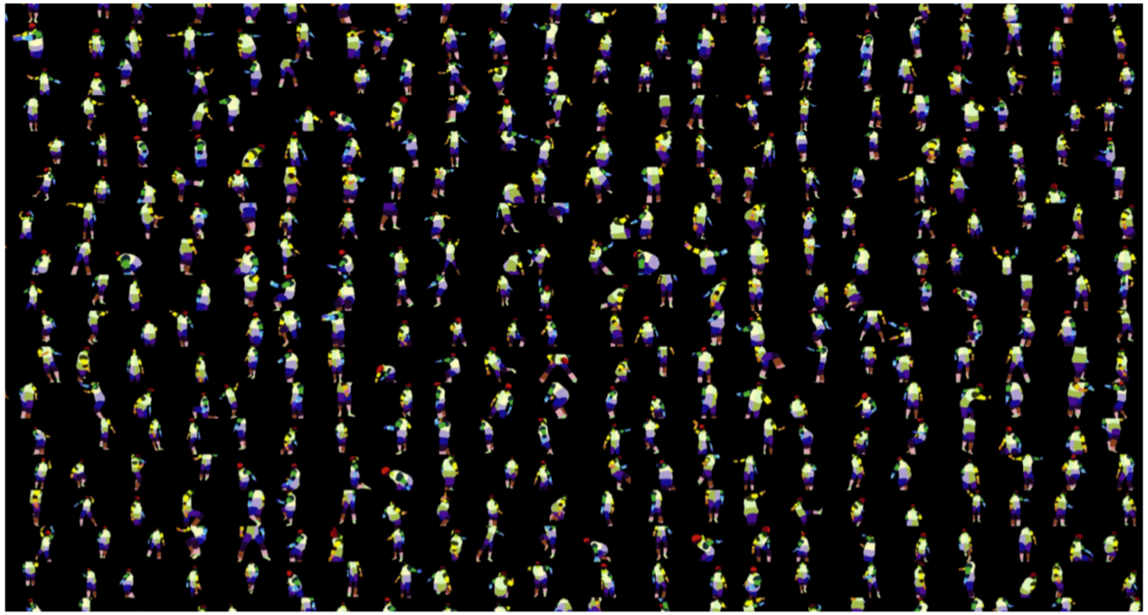
Parametric vs. Non-Parametric Classification Approaches

- The Gaussian assumption for $P(\boldsymbol{e} \mid \omega)$ leads to a fixed number of parameters (examples: MAP with Gaussians)
- Non-parametric models do not have a fixed number of parameters (examples: NN, kNN, decision trees)
- Non-parametric models grow in size to accommodate the data

Classification Showcases

Body part classification in the Kinect

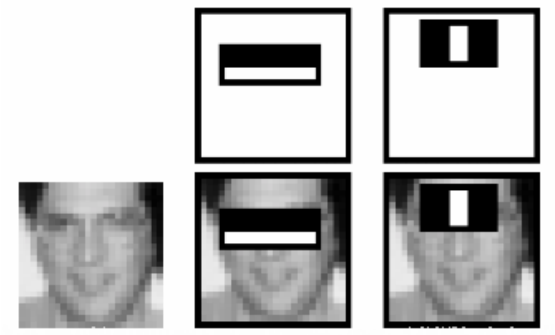




1 million test images, 1 day using a 1000 core cluster

Classification Showcases

Face detection by Viola & Jones



+ AdaBoost

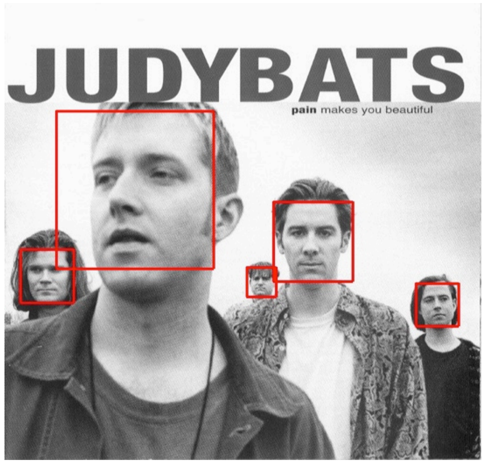
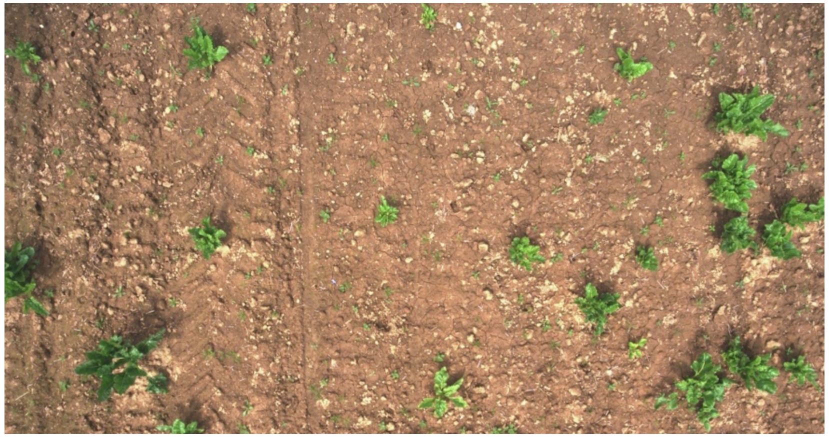
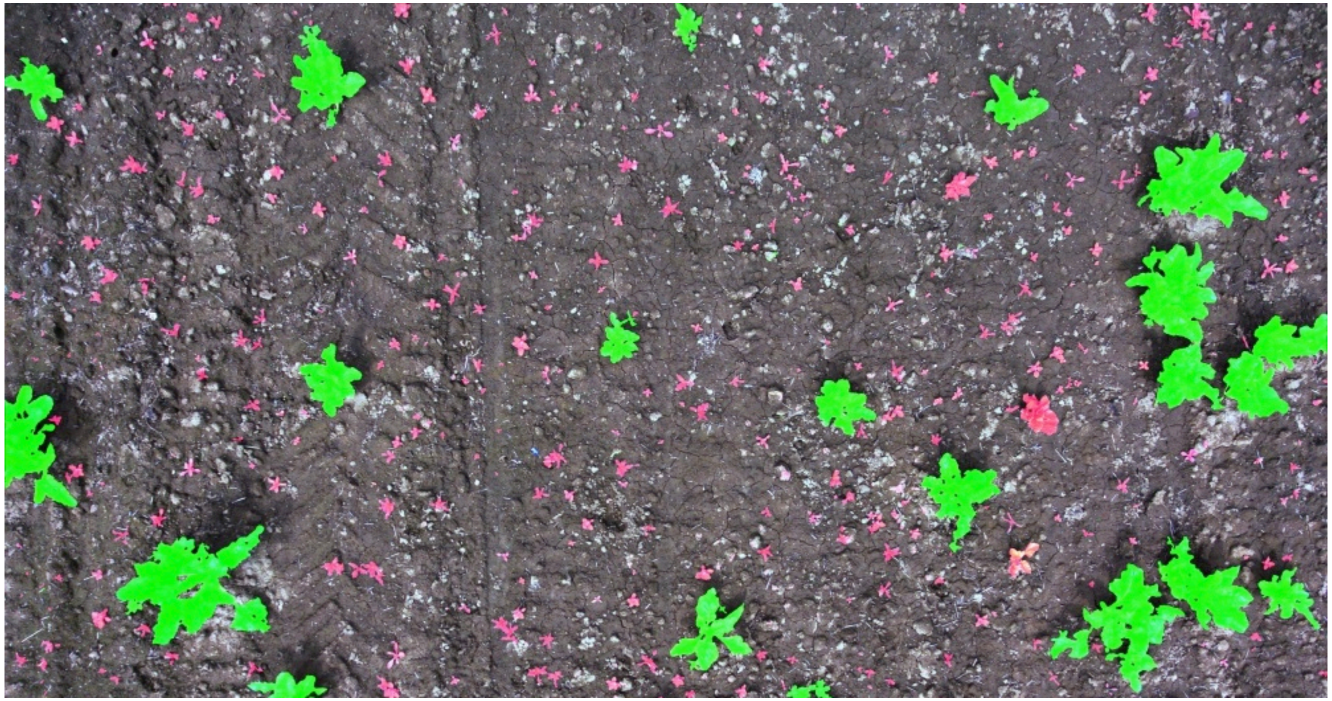


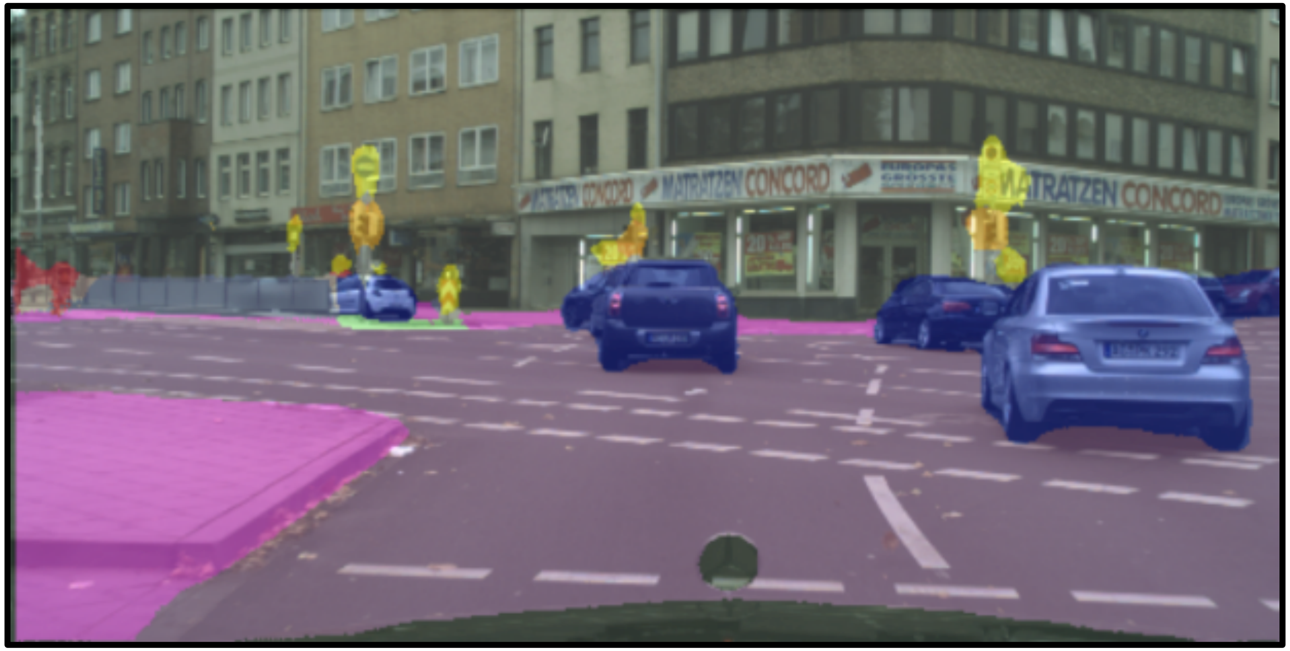
Image courtesy: Viola/Jones

Crop Weed Classification





Scene Understanding



Summary

- Introduction to classification
- Building a simple classifier
- Different types of errors
- Classifier Evaluation
- Nearest neighbor classifier
- Decision trees for classification
- MAP approach to classification
- Highly relevant for real world applications

Literature

- Alpaydin, Introduction to Machine Learning, Chapter 2, 3, 4.5, 5.5, 9.2

Slide Information

- The slides have been created by Cyrill Stachniss as part of the photogrammetry and robotics courses.
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**
- The photogrammetry material heavily relies on the very well written lecture notes by Wolfgang Förstner and the Photogrammetric Computer Vision book by Förstner & Wrobel.
- Parts of the robotics material stems from the great Probabilistic Robotics book by Thrun, Burgard and Fox.
- If you are a university lecturer, feel free to use the course material. If you adapt the course material, please make sure that you keep the acknowledgements to others and please acknowledge me as well. To satisfy my own curiosity, please send me email notice if you use my slides.

Cyrill Stachniss, cyrill.stachniss@igg.uni-bonn.de

(additional material)

Generative vs. Discriminative Approaches

- Generative approaches use data to calculate the posterior densities and then get the discriminant function
- The densities can be used to draw possible features (“to generate”)
- Discriminative approaches bypasses the estimation of densities and directly estimate the discriminants

MAP Classification with Gaussian Distributed Features

- Let us look into features that follow a Gaussian given a class

$$\boldsymbol{e} = \boldsymbol{x} \qquad \boldsymbol{x} \mid \omega_i \sim g(\boldsymbol{\mu}_i, \Sigma_i) \qquad i = 1, 2$$

- Thus, we can write

$$P(\omega_i \mid \boldsymbol{x}) = \eta \, g(\boldsymbol{x}, \boldsymbol{\mu}_i, \Sigma_i) \, P(\omega_i)$$

- and the negative log likelihood

$$-\ln P(\omega_i \mid \boldsymbol{x}) = -\ln \eta - \ln g(\boldsymbol{x}, \boldsymbol{\mu}_i, \Sigma_i) - \ln P(\omega_i)$$

MAP Approach

- The MAP classifier directly yields

$$\begin{aligned}
&\arg\max_{\omega_i} P(\omega_i \mid \boldsymbol{x}) \\
&\quad = \quad \arg\min_{\omega_i} \left(-\ln P(\omega_i \mid \boldsymbol{x})\right) \\
&\quad = \quad \arg\min_{\omega_i} \left(-\ln g(\boldsymbol{x}, \mu_i, \Sigma_i) - \ln P(\omega_i)\right)
\end{aligned}$$

- and the classification boundary are points in which the function

$$D(\boldsymbol{x}) \quad = \quad -\ln P(\omega_1 \mid \boldsymbol{x}) + \ln P(\omega_2 \mid \boldsymbol{x})$$

changes its sign

Classification Boundaries

- The classification boundaries are points in which $D(\boldsymbol{x})$ changes its sign

$$D(\boldsymbol{x}) \quad = \quad -\ln P(\omega_1 \mid \boldsymbol{x}) + \ln P(\omega_2 \mid \boldsymbol{x})$$

Classification Boundaries

- The classification boundaries are points in which $D(\boldsymbol{x})$ changes its sign

$$
\begin{aligned}
D(\boldsymbol{x}) &= -\ln P(\omega_1 \mid \boldsymbol{x}) + \ln P(\omega_2 \mid \boldsymbol{x}) \\
&= -\ln g(\mu_1, \Sigma_1) - \ln P(\omega_1) \\
&\quad + \ln g(\mu_2, \Sigma_2) + \ln P(\omega_2)
\end{aligned}
$$

Classification Boundaries

- The classification boundaries are points in which $D(\boldsymbol{x})$ changes its sign

$$
\begin{aligned}
D(\boldsymbol{x}) &= -\ln P(\omega_1 \mid \boldsymbol{x}) + \ln P(\omega_2 \mid \boldsymbol{x}) \\
&= -\ln g(\mu_1, \Sigma_1) - \ln P(\omega_1) \\
&\quad + \ln g(\mu_2, \Sigma_2) + \ln P(\omega_2) \\
&= \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_1)^T \Sigma_1^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_1) \\
&\quad - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_2)^T \Sigma_2^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_2) \\
&\quad - (\ln P(\omega_1) - \ln P(\omega_2)) + \text{const.}
\end{aligned}
$$

Classification Boundaries

- The function $D(\boldsymbol{x})$ is a quadratic function as:

$$
\begin{aligned}
&(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)-(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) \\
&= \quad \boldsymbol{x}^{\mathsf{T}}(\Sigma_1^{-1}-\Sigma_2^{-1})\boldsymbol{x} && \textbf{quadratic} \\
&\qquad -2\boldsymbol{x}^{\mathsf{T}}(\Sigma_1^{-1}\boldsymbol{\mu}_1-\Sigma_2^{-1}\boldsymbol{\mu}_2)+ && \textbf{linear} \\
&\qquad \boldsymbol{\mu}_1^{\mathsf{T}}\Sigma_1^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^{\mathsf{T}}\Sigma_2^{-1}\boldsymbol{\mu}_2 && \textbf{constant}
\end{aligned}
$$

- The shape of $D(\boldsymbol{x})=0$ depends on the Eigenvalues of $\Sigma_1^{-1},\Sigma_2^{-1},$

Example

- Elliptic and hyperbolic boundaries

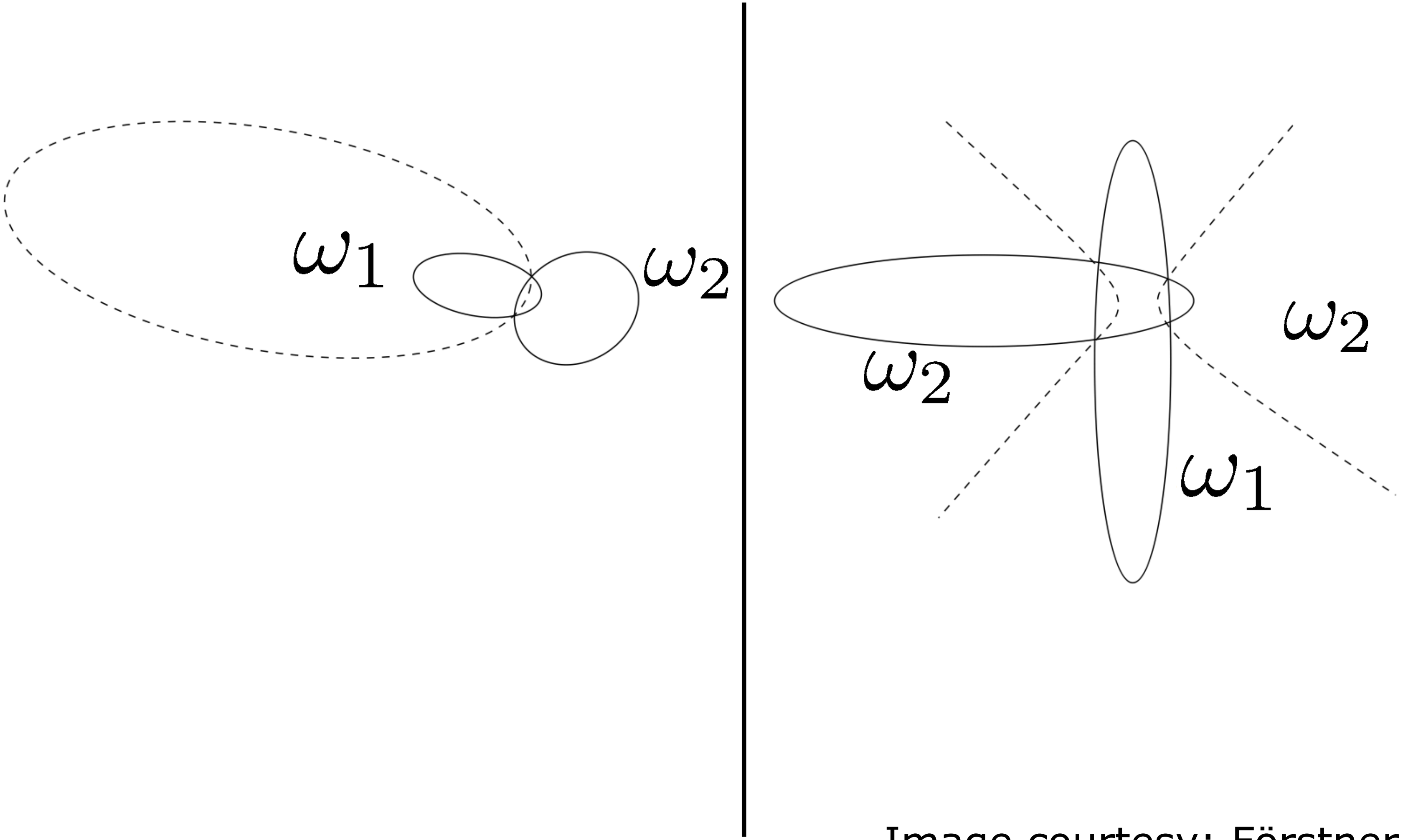


Image courtesy: Förstner

(additional material)

Extension to Discrete and Gaussian Distributed Features

- The features consist of discrete $\boldsymbol{b}$ and Gaussian distributed $\boldsymbol{x}$ features

$$\boldsymbol{e} = (\boldsymbol{b}, \boldsymbol{x})^\top \qquad \boldsymbol{x} \mid \omega_i \sim g(\boldsymbol{\mu}_i, \Sigma_i) \qquad i = 1, 2$$

- Thus, we can write

$$P(\omega_i \mid \boldsymbol{b}, \boldsymbol{x}) \propto P(\boldsymbol{b} \mid \omega_i)\, g(\boldsymbol{x}, \boldsymbol{\mu}_i, \Sigma_i)\, P(\omega_i)$$

- assuming independence in $\boldsymbol{b}$

$$P(\omega_i \mid \boldsymbol{b}, \boldsymbol{x}) \propto \prod_k P(b_k \mid \omega_i)\, g(\boldsymbol{x}, \boldsymbol{\mu}_i, \Sigma_i)\, P(\omega_i)$$

(additional material)

Classification Boundaries

- This leads to a similar result:

$$\begin{aligned} D(\boldsymbol{x}) &= -\ln P(\omega_1 \mid \boldsymbol{b}, \boldsymbol{x}) + \ln P(\omega_2 \mid \boldsymbol{b}, \boldsymbol{x}) \\ &= \boxed{-\sum_k \ln P(b_k \mid \omega_1) + \sum_k \ln P(b_k \mid \omega_2)} \quad \text{different} \\ & \boxed{\begin{array}{l} +\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T \Sigma_1^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_1) \\ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T \Sigma_2^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_2) \\ -(\ln P(\omega_1) - \ln P(\omega_2)) + \text{const.} \end{array}} \quad \text{same} \end{aligned}$$

Effect

This leads to a selection of the class for which

- the probability of the discrete features b_k is large,
- the deviation in x from the mean with respect to the variance is small, and
- the class ω_i has a large a-priori probability.

(additional material)

The Likelihood Function $P(\boldsymbol{e} \mid \omega)$

... for discrete features for class i:

$$\boldsymbol{b}_i = (b_{i1}, \dots, b_{iM})^\top$$

$$p(b_{i1}, \dots, b_{iM} \mid \omega_i) = \frac{\#\boldsymbol{b} = (b_{i1}, \dots, b_{ik}, \dots, b_{iM})}{N_i}$$

occurrences of
class i (not N)

The Likelihood Function $P(\boldsymbol{e} \mid \omega)$

... for Gaussian distributed features for class i:

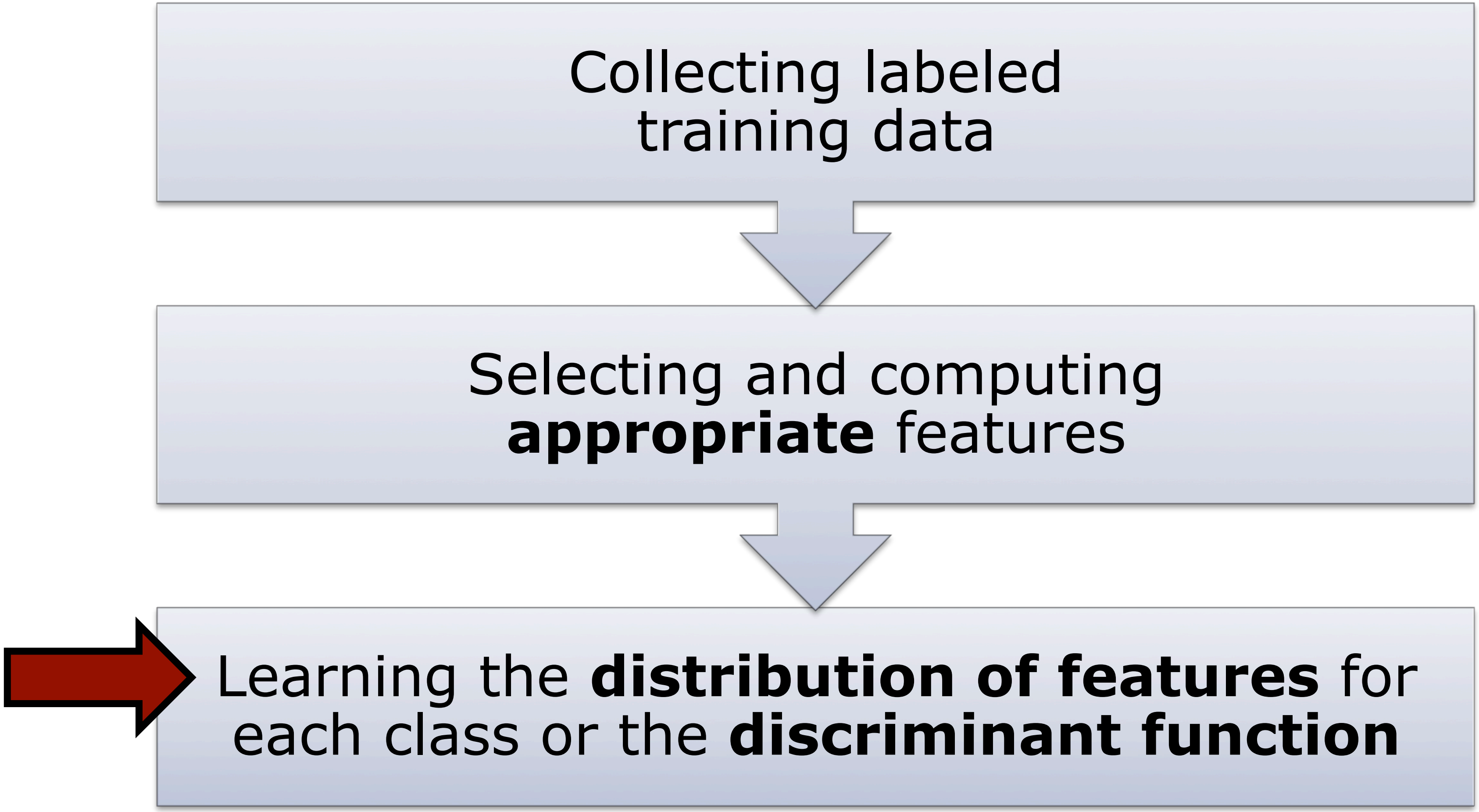
$$p(\boldsymbol{x}_i \mid \omega_i) = g(\boldsymbol{x}_i, \boldsymbol{\mu}_i, \Sigma_i)$$

$$\boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{x}_{ij}$$

all feature vectors corresponding to data of class i

$$\Sigma_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^{\mathsf{T}}$$

Traditional Classification: Training



Collecting labeled training data

Selecting and computing **appropriate** features

Learning the **distribution of features** for each class or the **discriminant function**

Learning a Classifier

- In practice the distributions $P(\boldsymbol{e} \mid \omega)$ and $P(\omega)$ are **not known**
- Both must be **learned from data**

Training data

- Sample set of size N
- N_i samples correspond to class ω_i

Prior Distribution $P(\omega)$

- The prior distribution $P(\omega)$, which models the probability that a random sample corresponds to class ω_i is

$$P(\omega_i) = \frac{N_i}{N}$$