Photogrammetry & Robotics Lab

Introduction to Classification

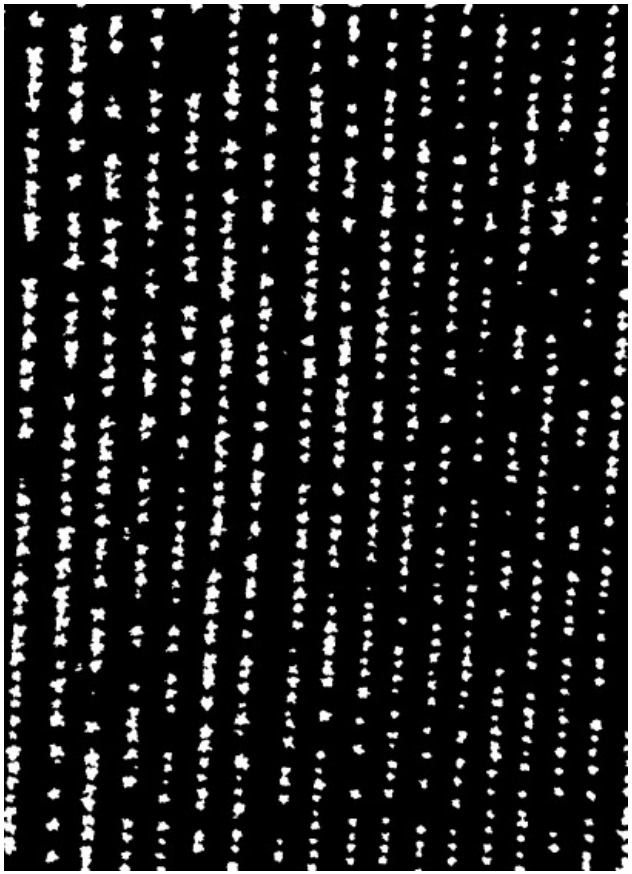
Cyrill Stachniss

The slides have been created by Cyrill Stachniss.

Classification Example

Detecting vegetation

vegetation?
yes/no



Classification Problem

- Given a set of K classes

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

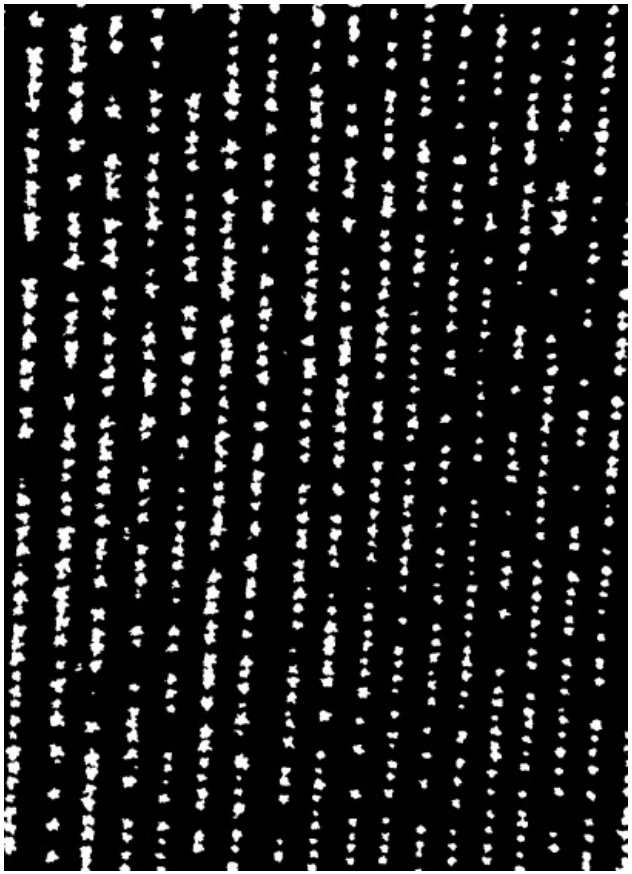
- and features $\boldsymbol{e}$
- learn a function f that assigns a class given feature

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

Example

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

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



State of the Art Examples

Example

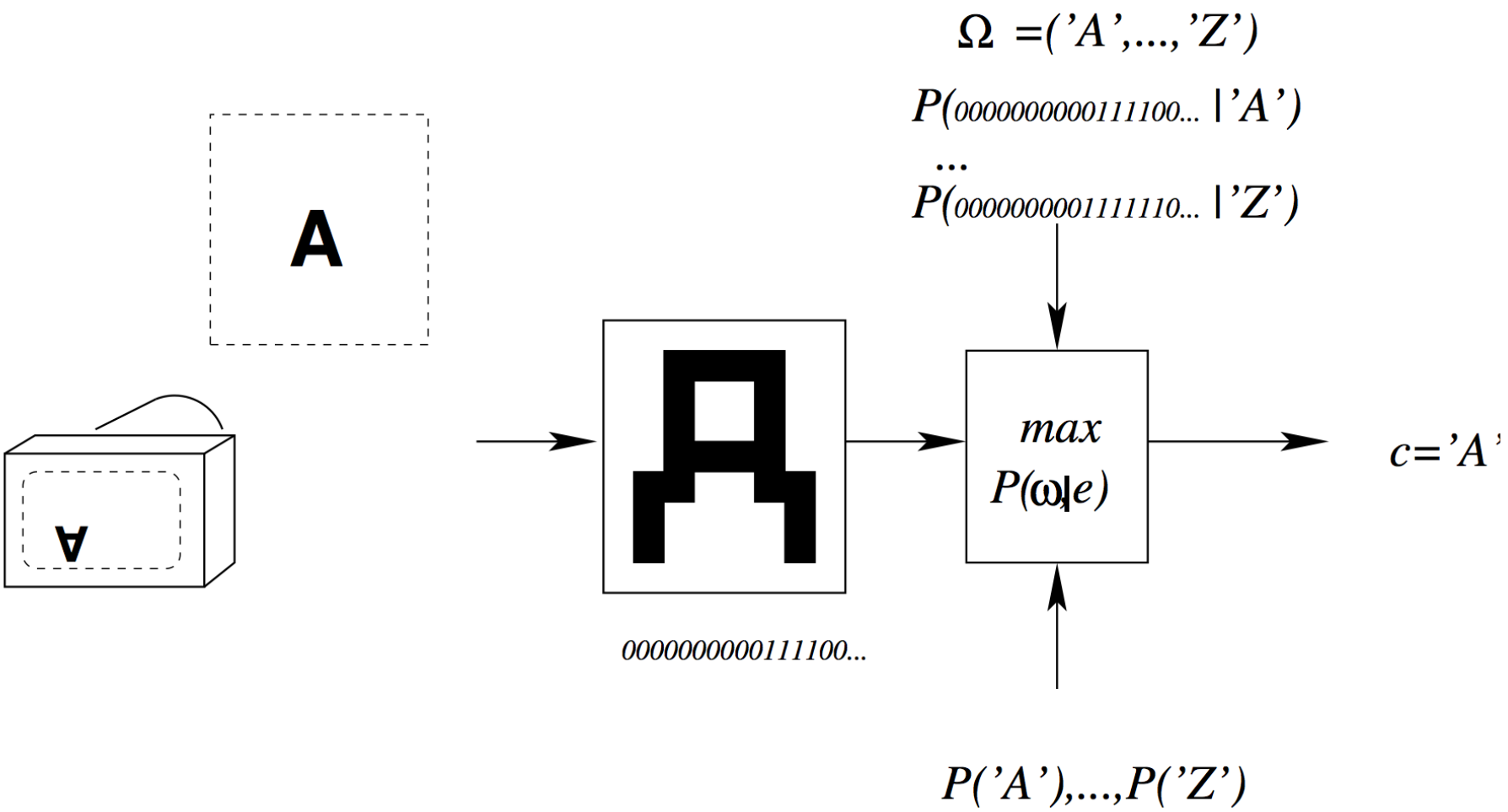


Image courtesy: Förstner

Learning Needs Information

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

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

$$P(00000\ldots \mid' A')$$

$$\ldots$$

$$P(11111\ldots \mid' Z')$$

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

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

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?

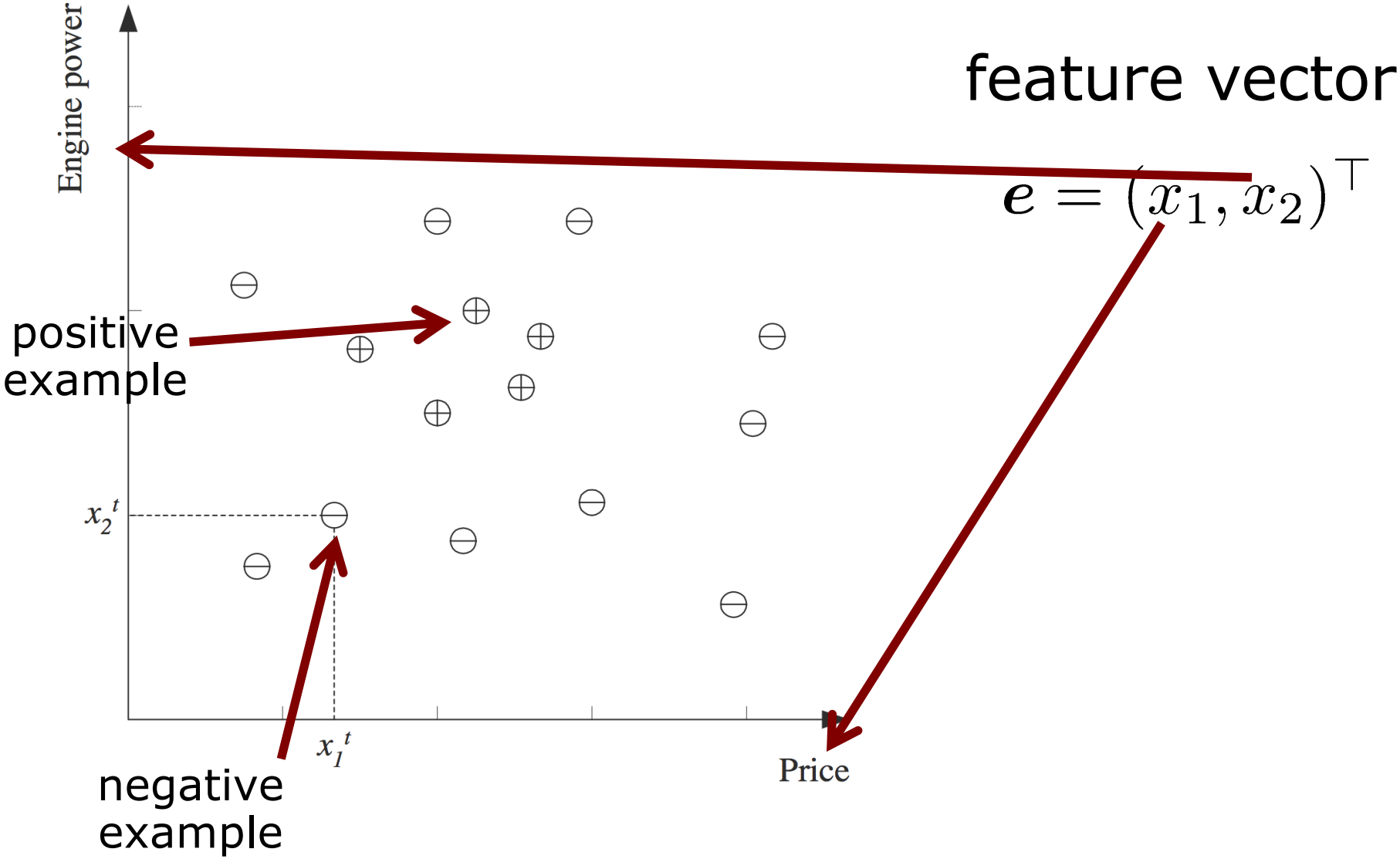
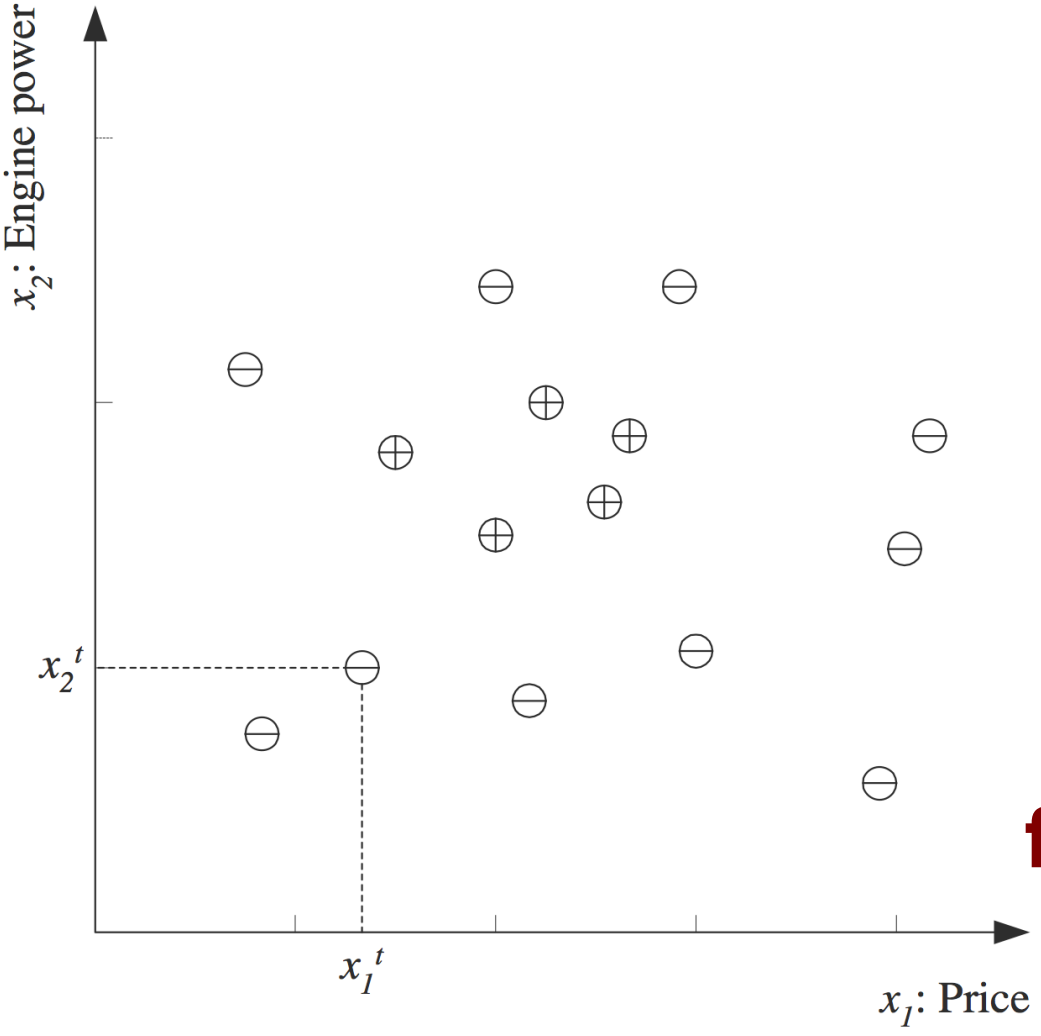


Image courtesy: Aplaydin

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?

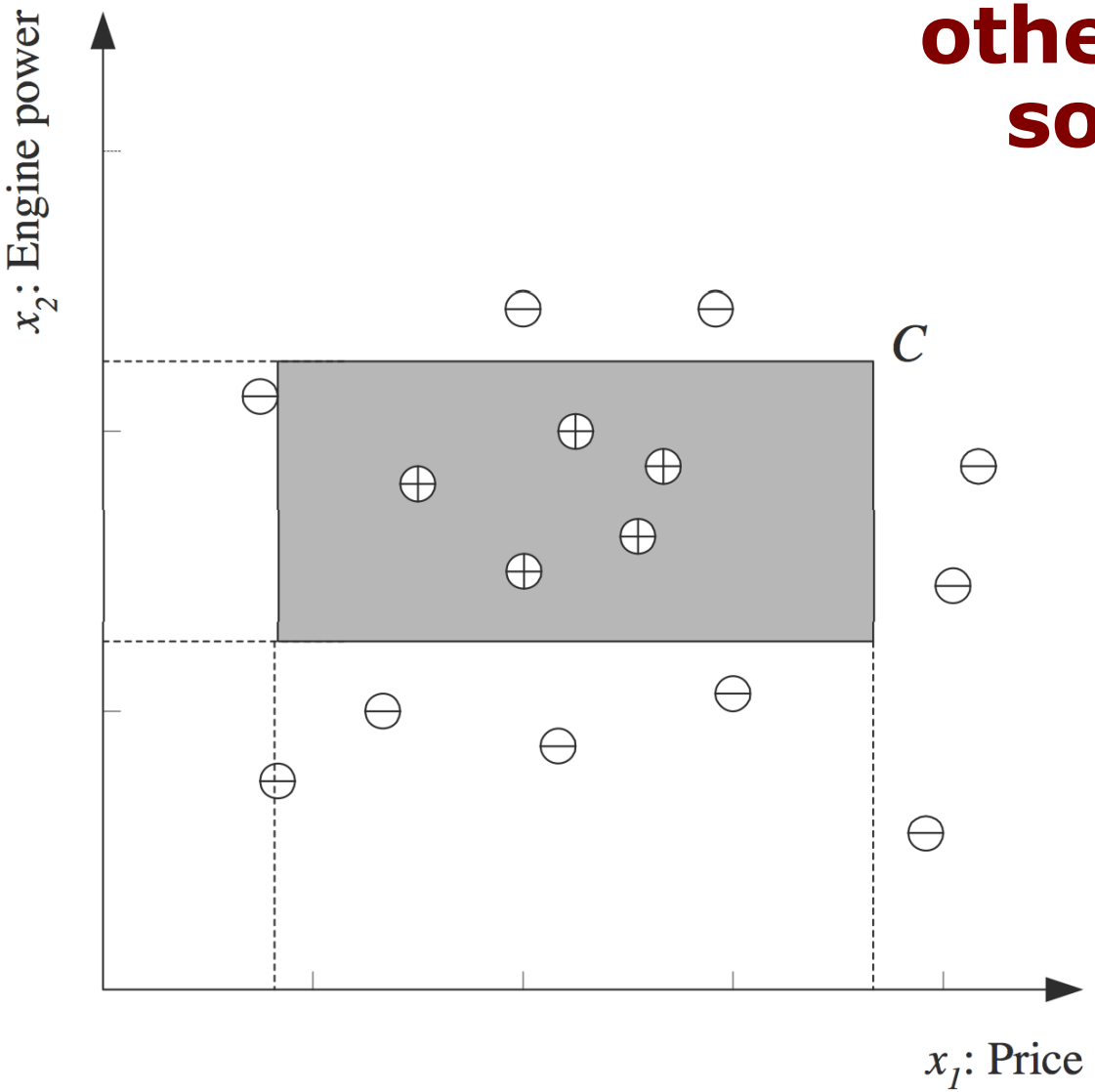


Image courtesy: Aplaydin

Multiple Consistent Hypotheses with the Training Data

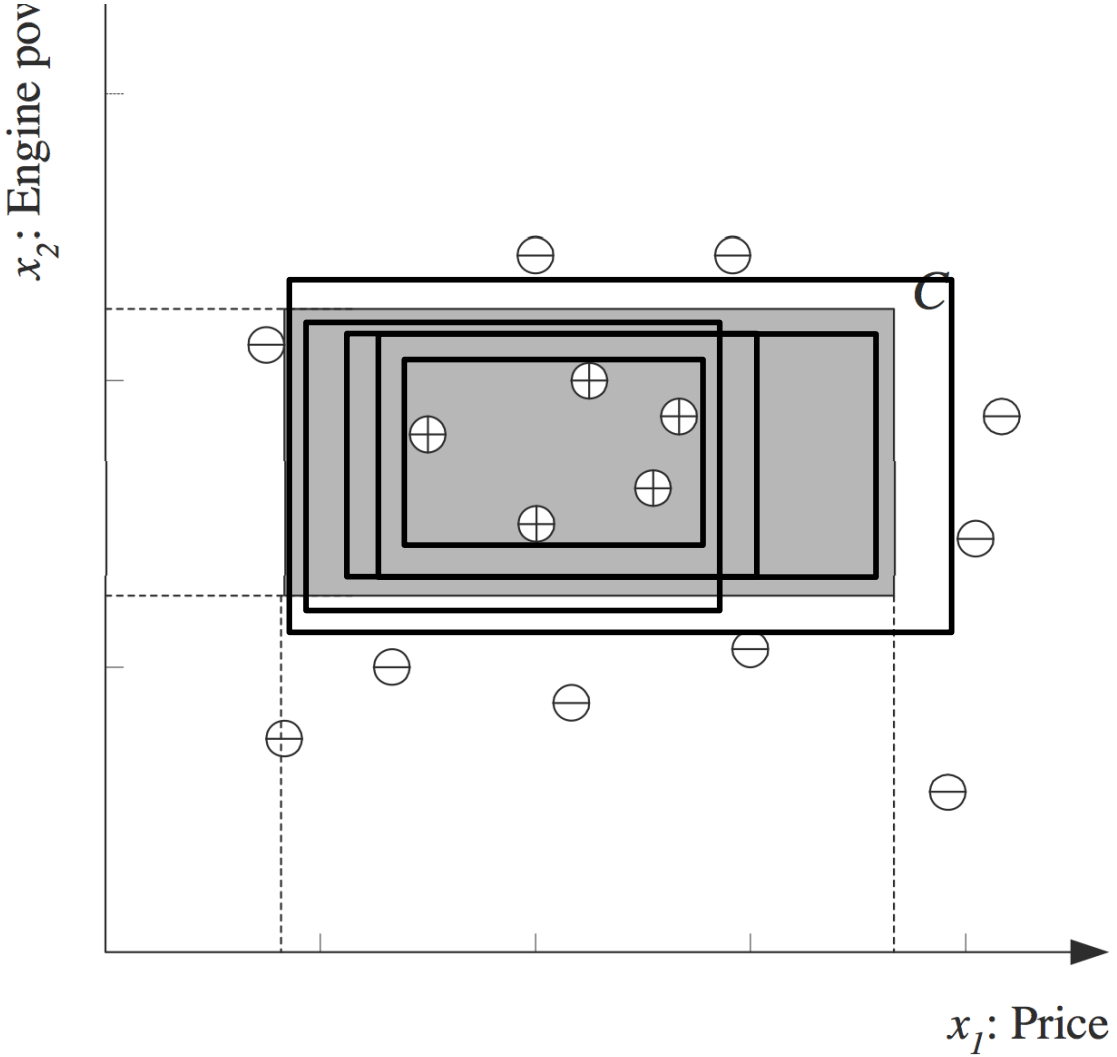


Image courtesy: Aplaydin

Multiple Consistent Hypotheses with the Training Data

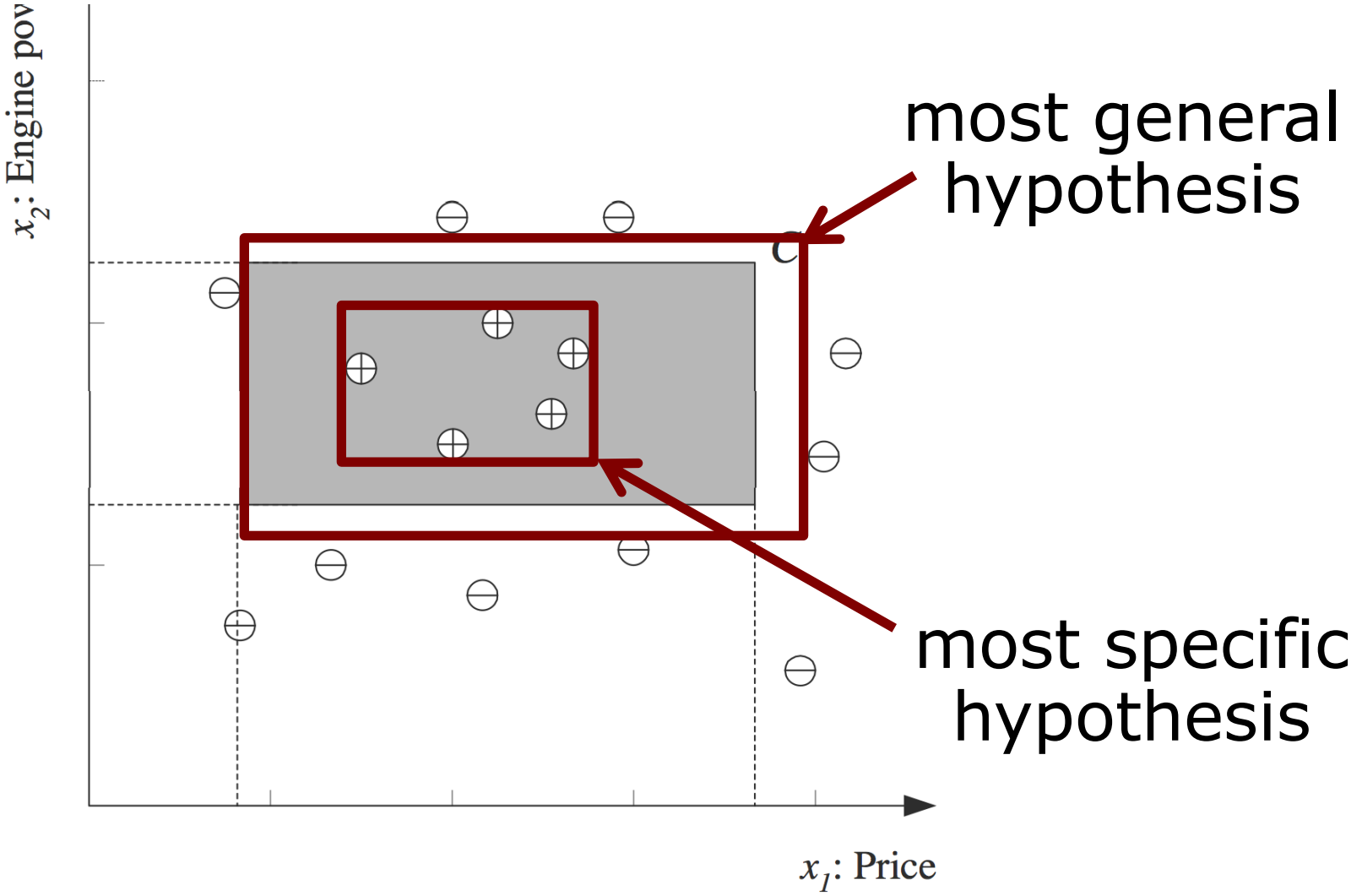


Image courtesy: Aplaydin

Most Specific and Most General

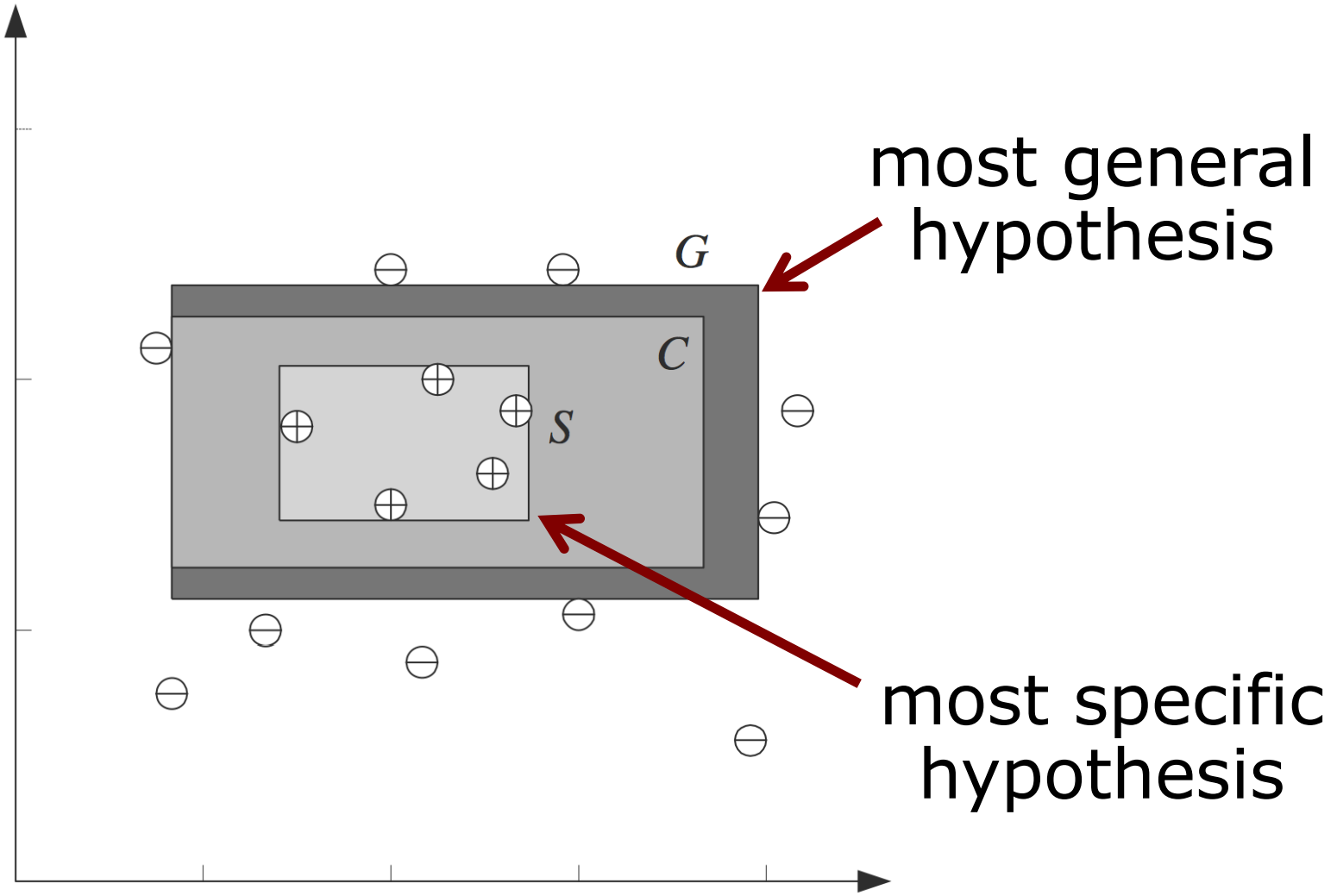


Image courtesy: Aplaydin

Version Space is Between the Most Specific and Most General

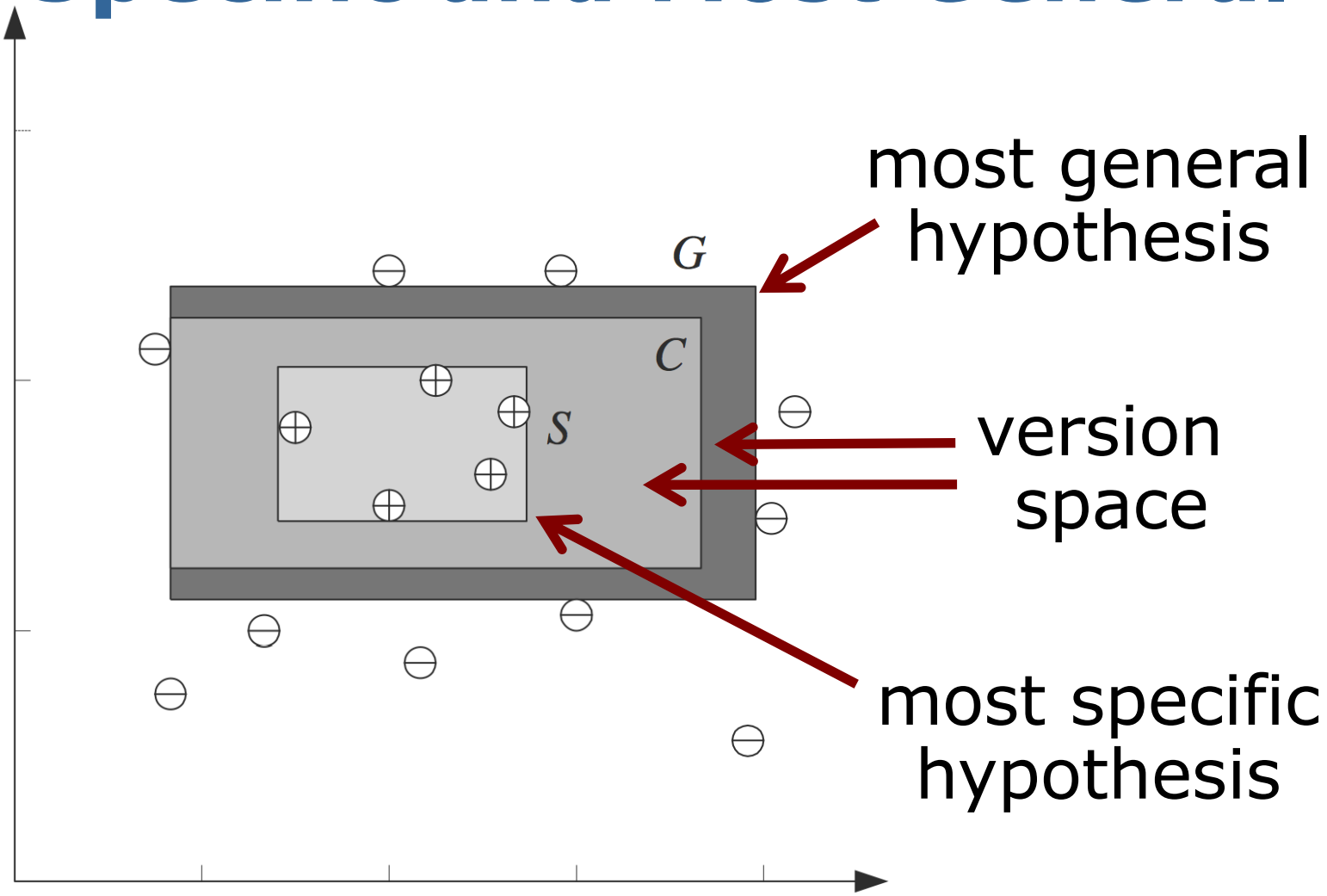


Image courtesy: Aplaydin

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

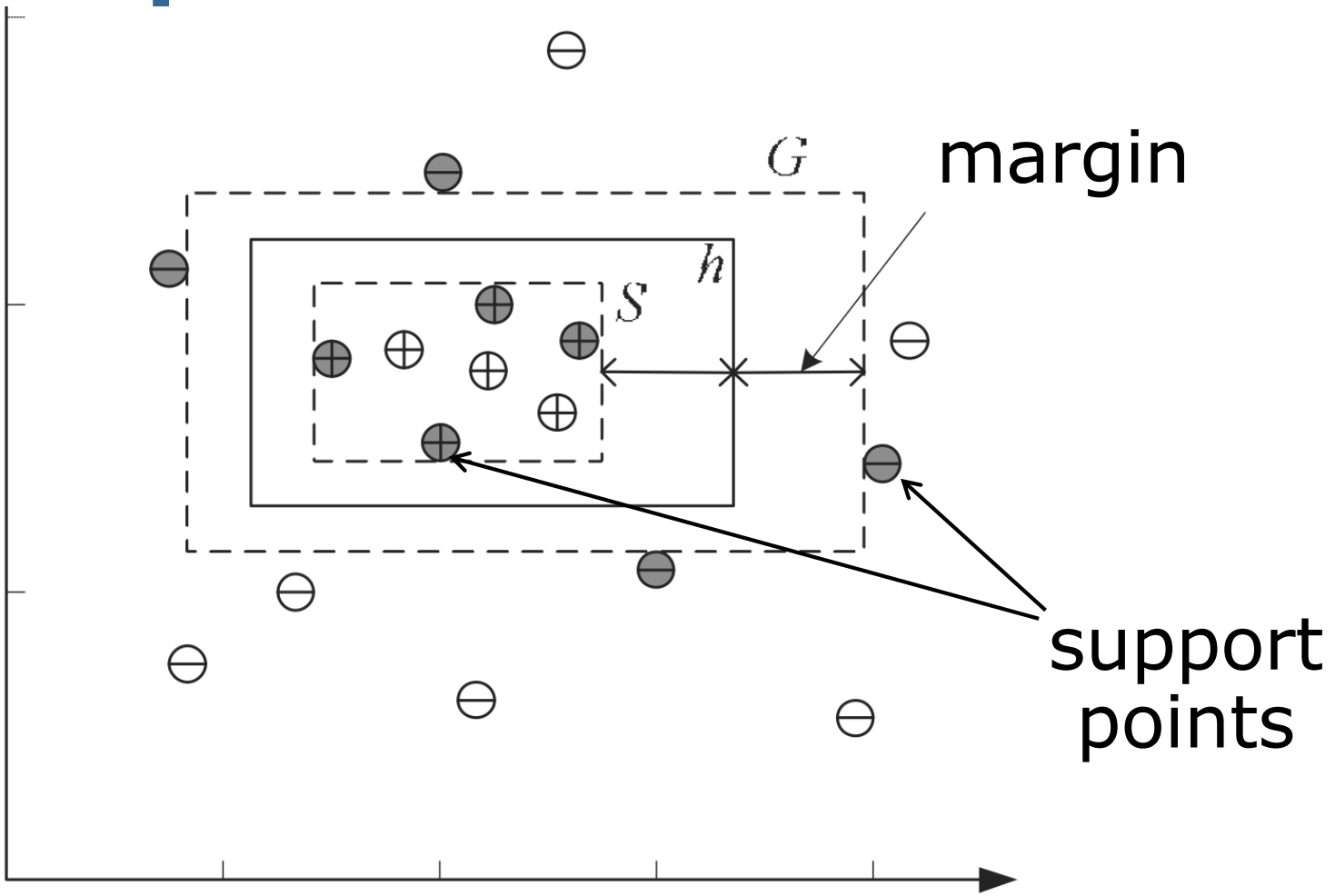


Image courtesy: Aplaydin

Classification Errors

Classification Errors

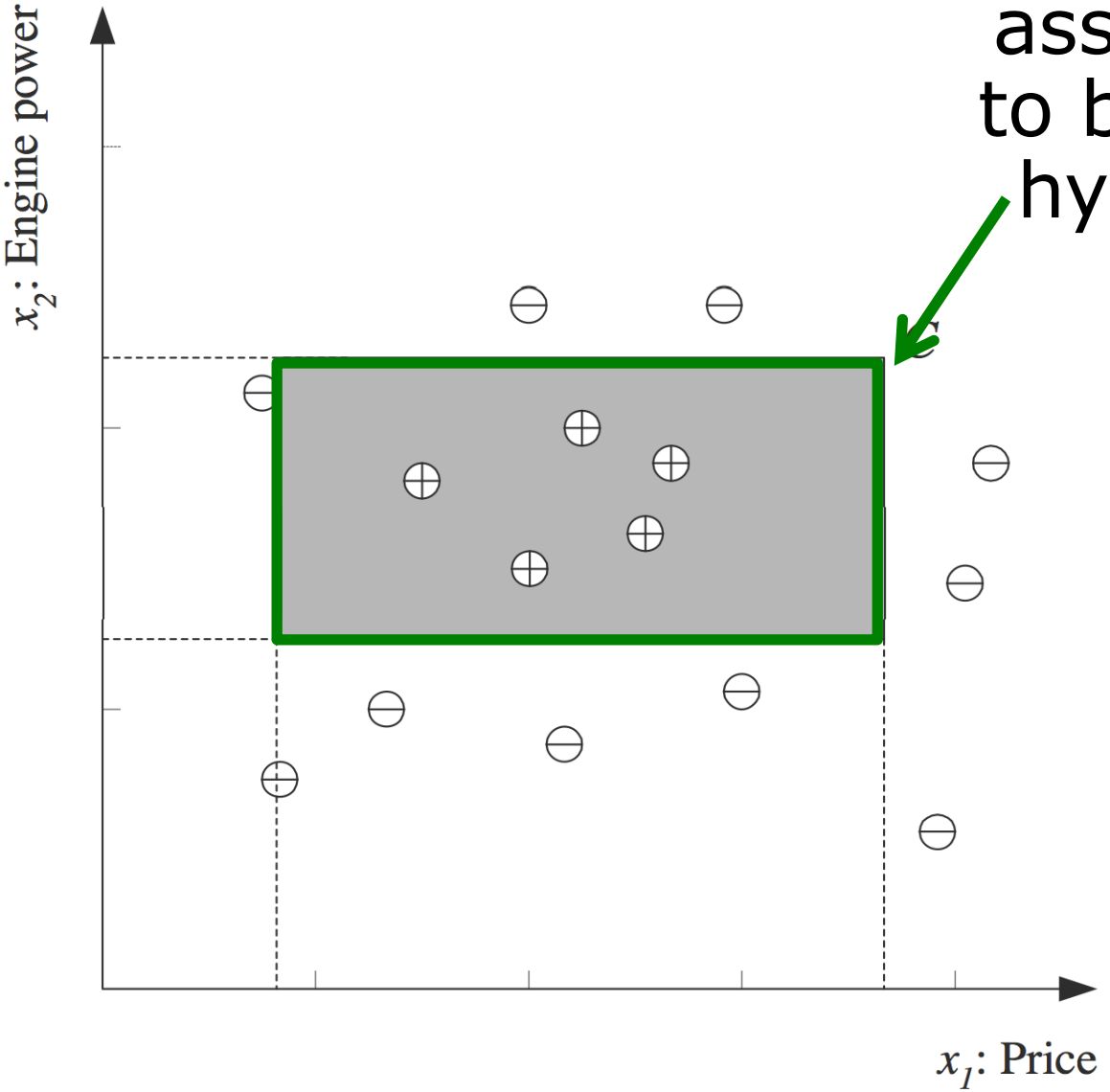


Image courtesy: Aplaydin

Classification Errors

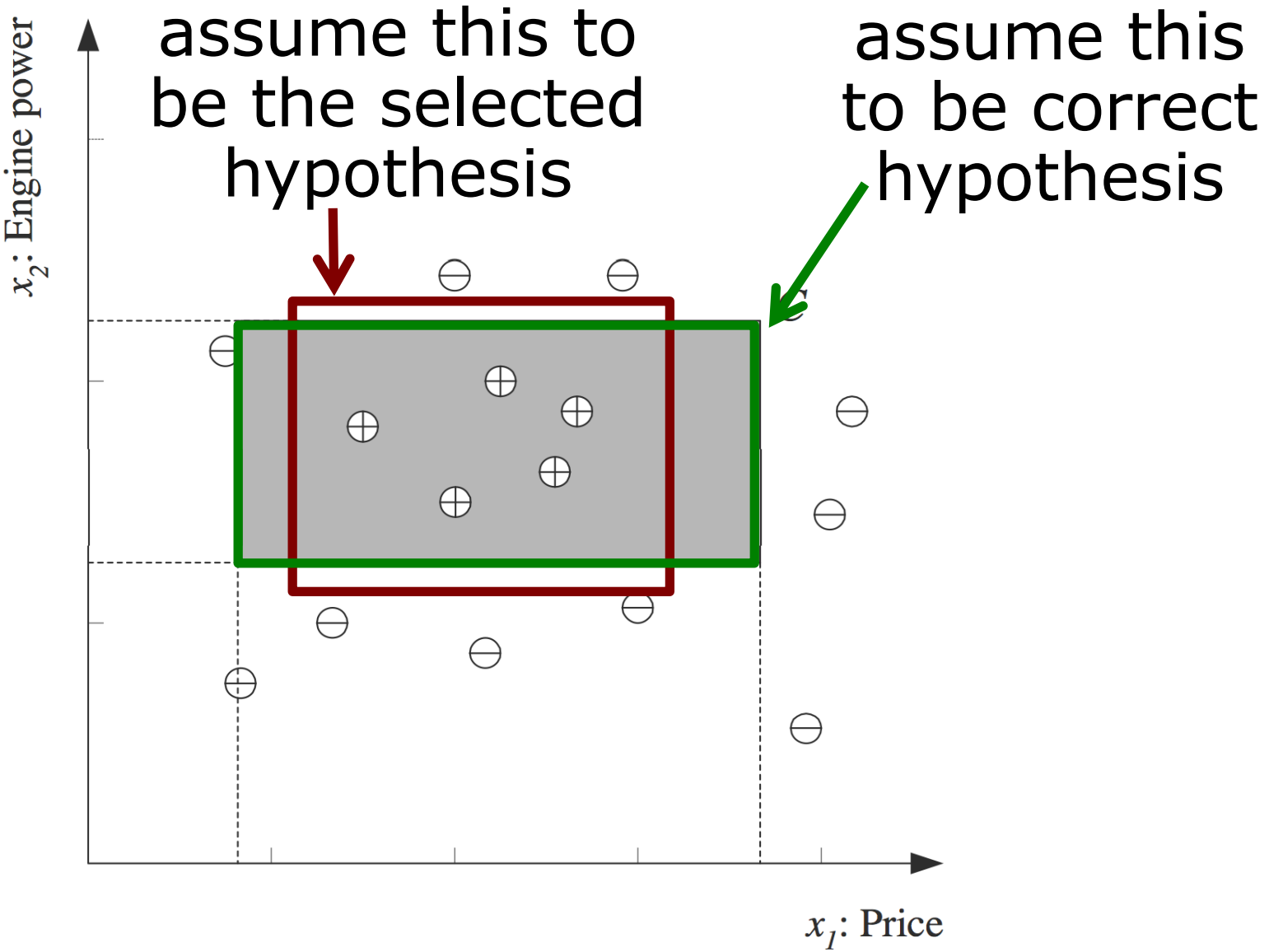


Image courtesy: Aplaydin

Classification Errors

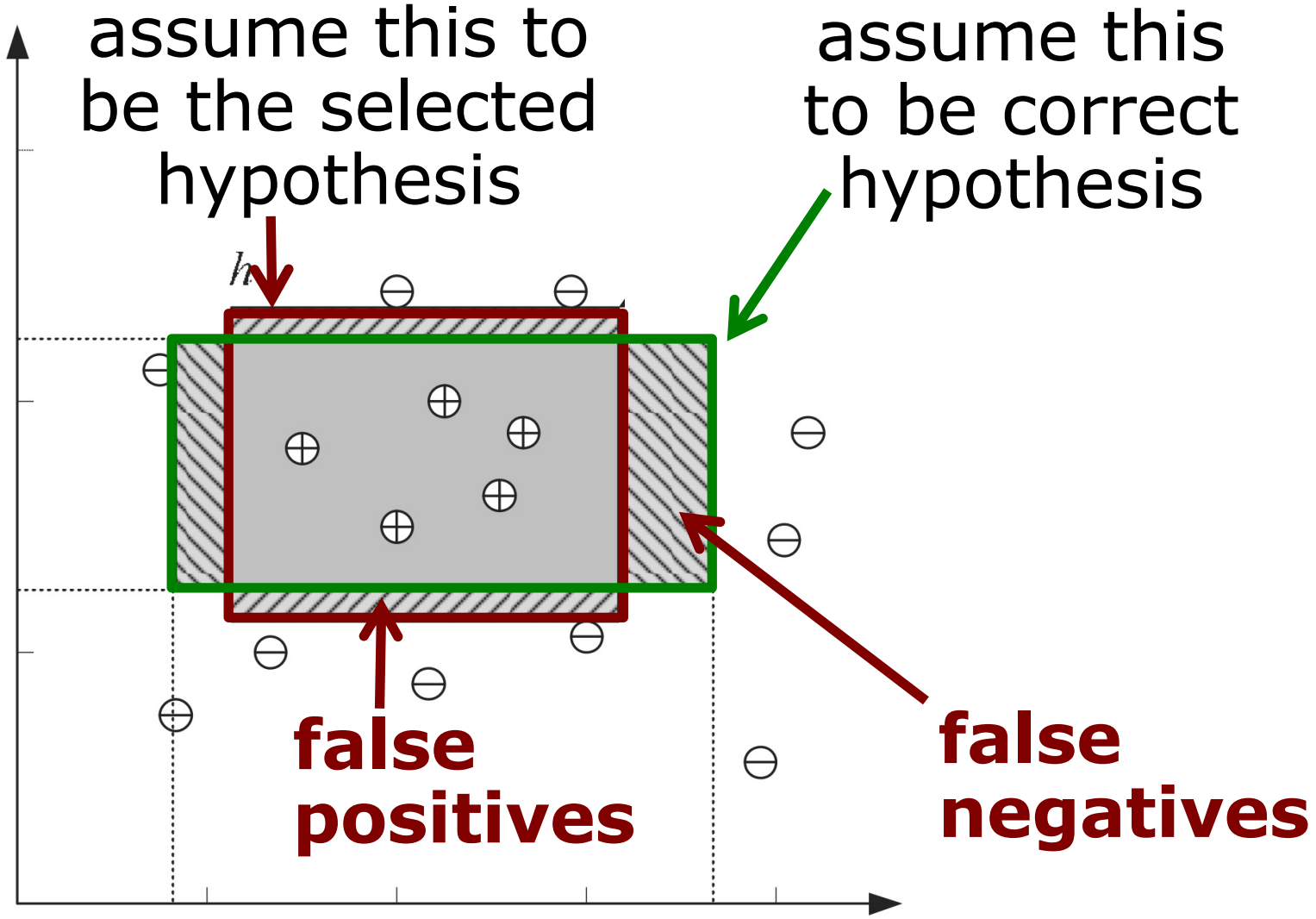


Image courtesy: Aplaydin

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	
		positive	negative
classified as	positive	**TP**	**FP**
	negative	**FN**	**TN**

confusion matrix

		in reality	
		class 1	class 2
classified as	class 1	**1 as 1**	**2 as 1**
	class 2	**1 as 2**	**2 as 2**

Evaluating a Classifier

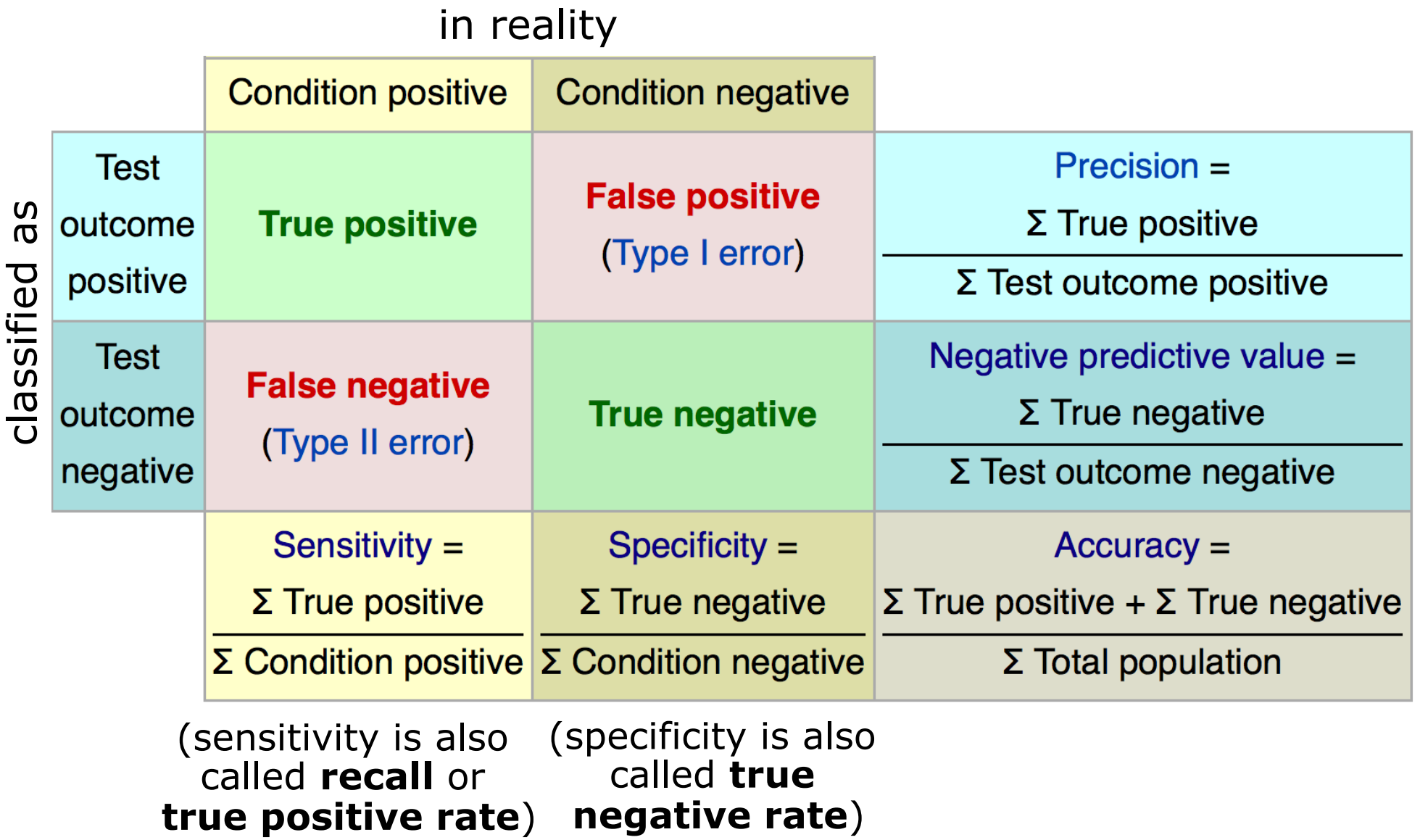


Image courtesy: Wikipedia.org

False and True Positive Rate

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

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

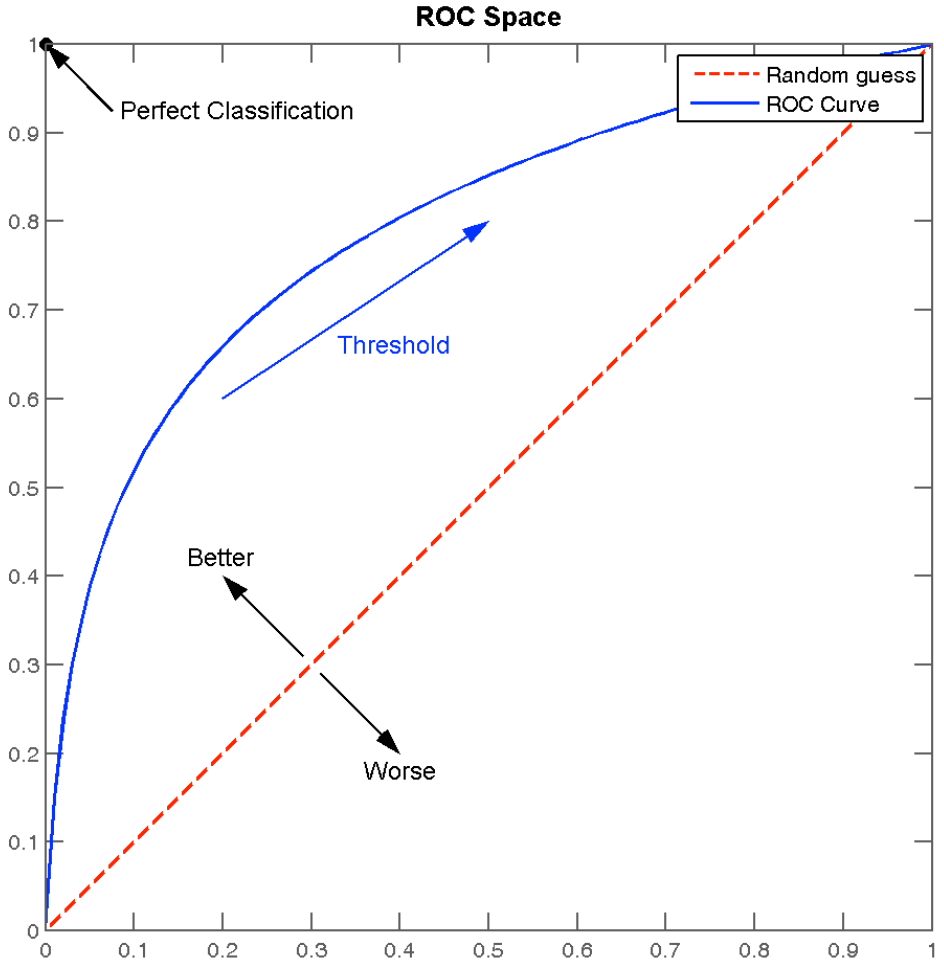
- **True positive rate** (=sensitivity, recall) is the probability that a randomly selected, in reality positive example is classified as positive

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

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

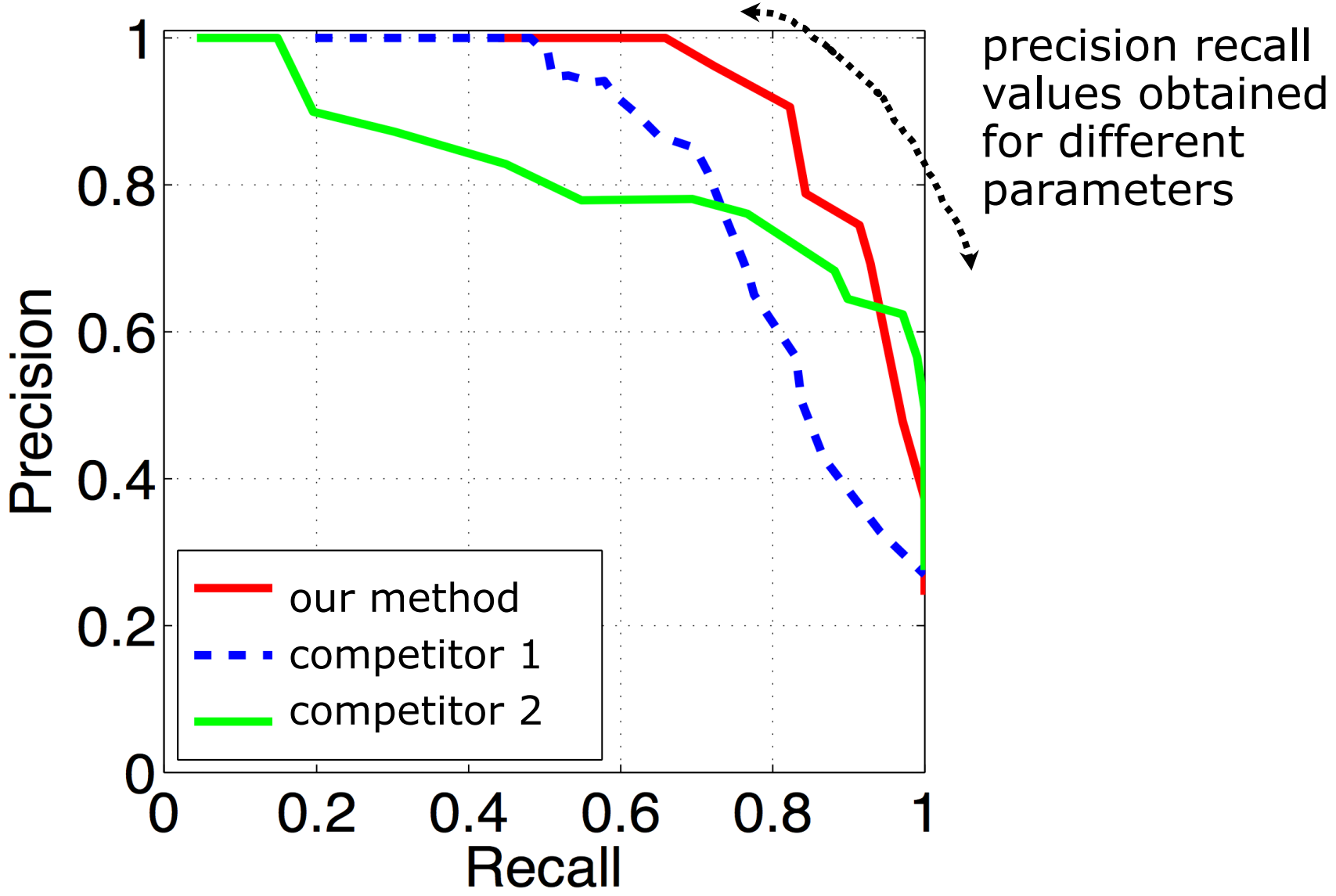
- **Precision** is the probability that a randomly selected, positively classified example is positive in reality

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

- **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



F-score / F_1 score / F-measure

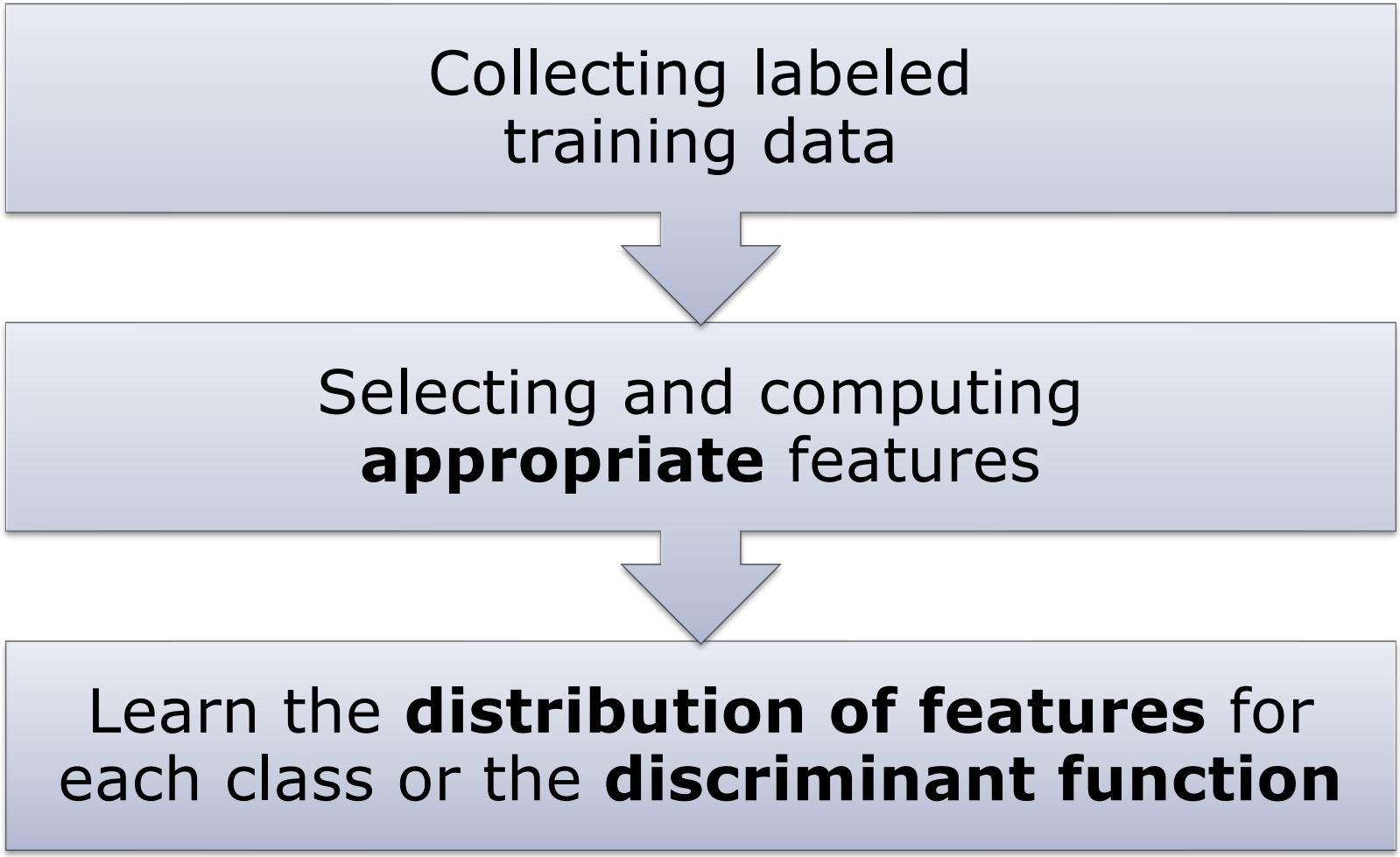
- Combines precision and recall into one value (harmonic mean)
- F-score reaches its best value at 1 and its worst score at 0

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

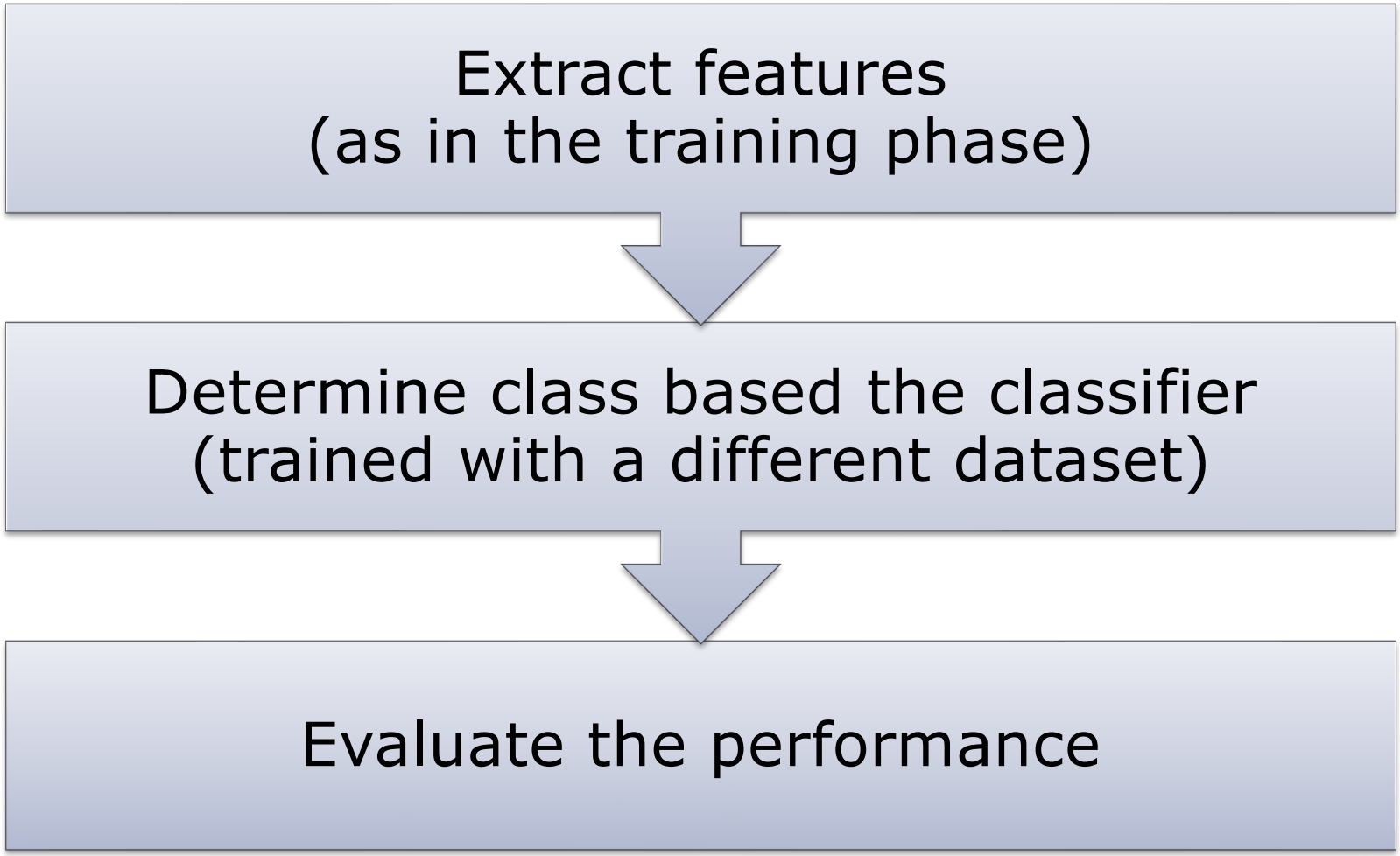
- Note: There is a large number of different measures…

Designing a Traditional Classifier

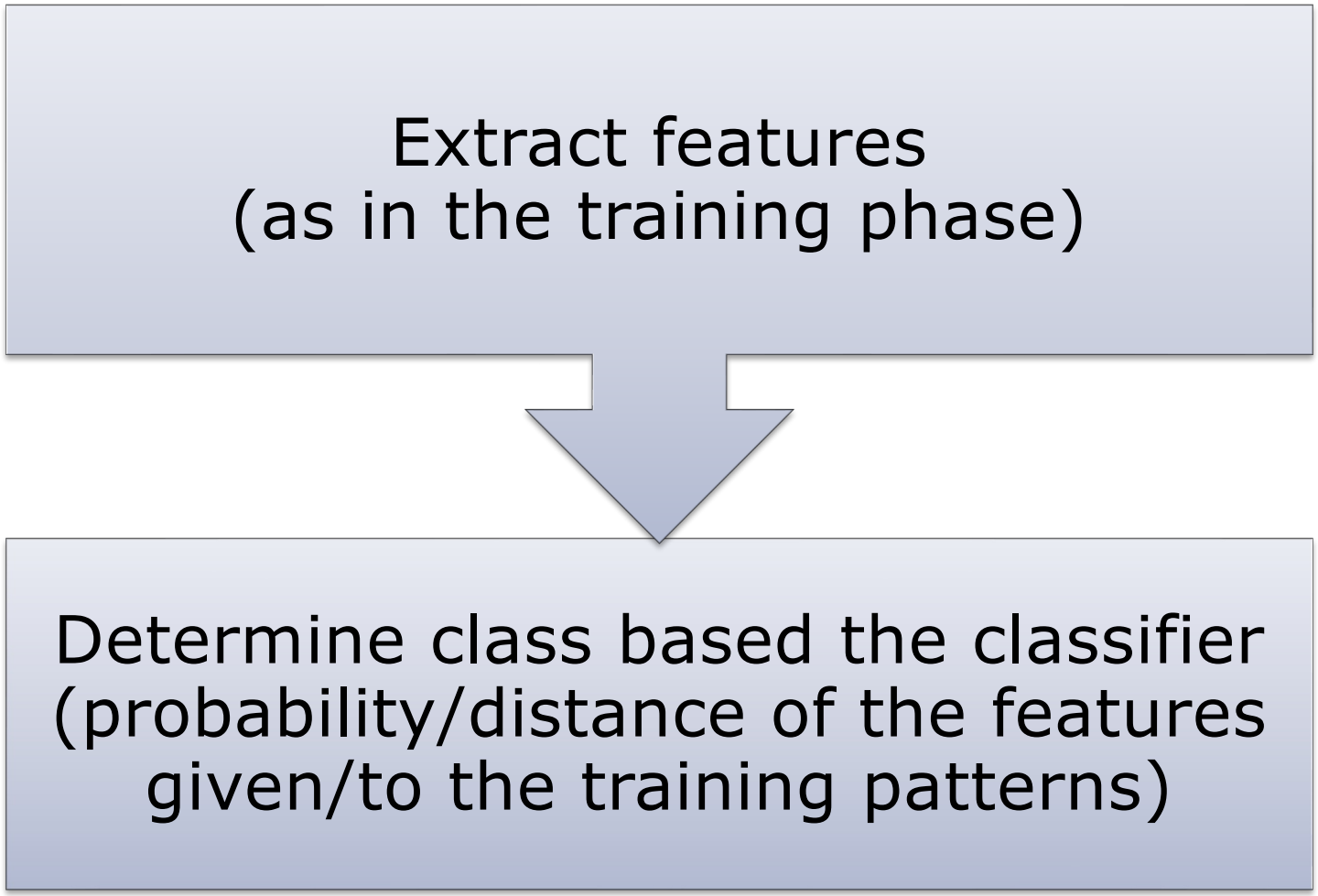
Traditional Classification: Training



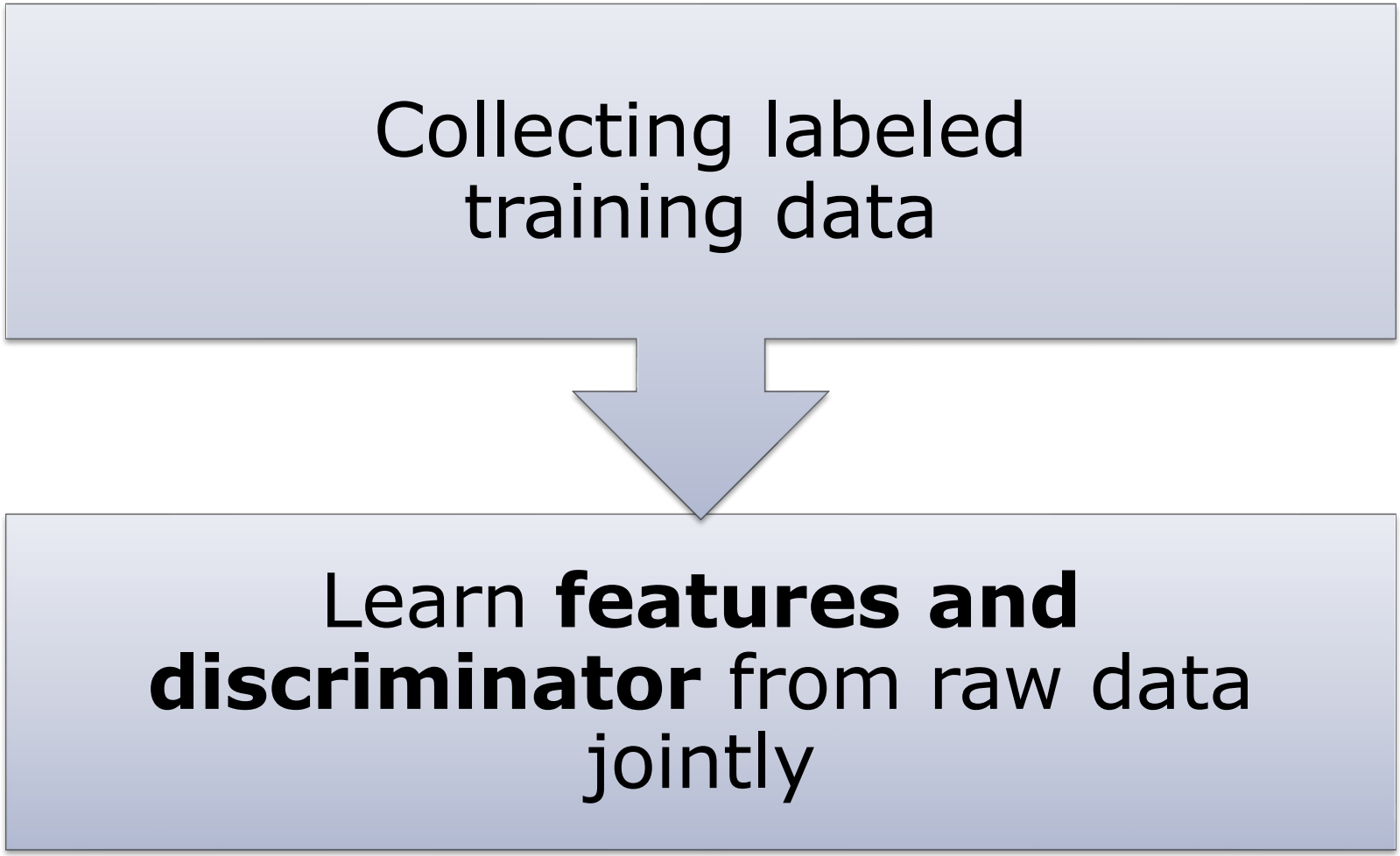
Traditional Classification: Testing (on different datasets)



Traditional Classification: Operation



End-to- End Classification (Neural Networks): Training



Generalization

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

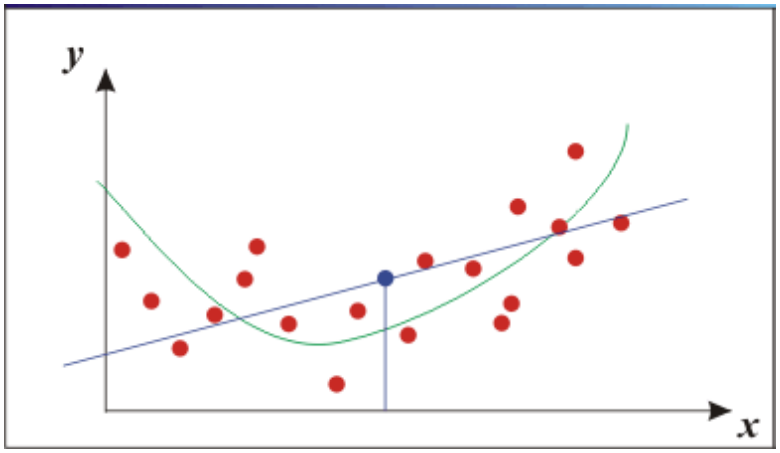
Training set (labels known)

Test set
(labels unknown)

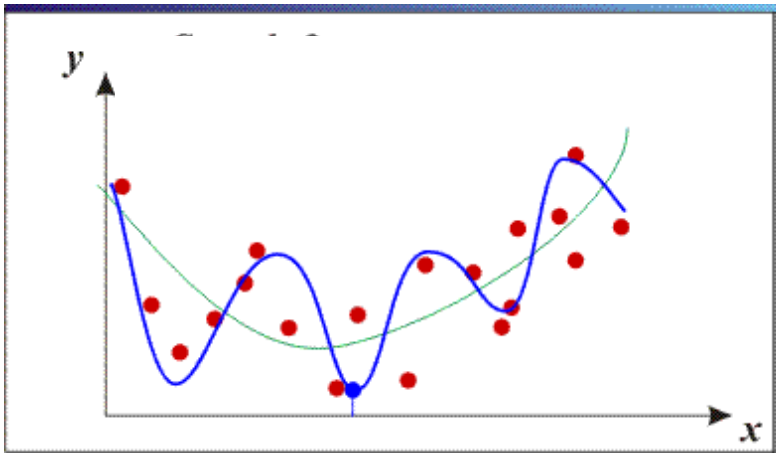
Components of the Generalization Error

- **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

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

$$E(MSE) = noise^2 + bias^2 + variance$$

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

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

Remember…

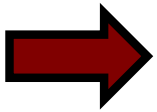
No classifier is inherently better than any other: we need to **make assumptions to generalize**

Three **types of errors**

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

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

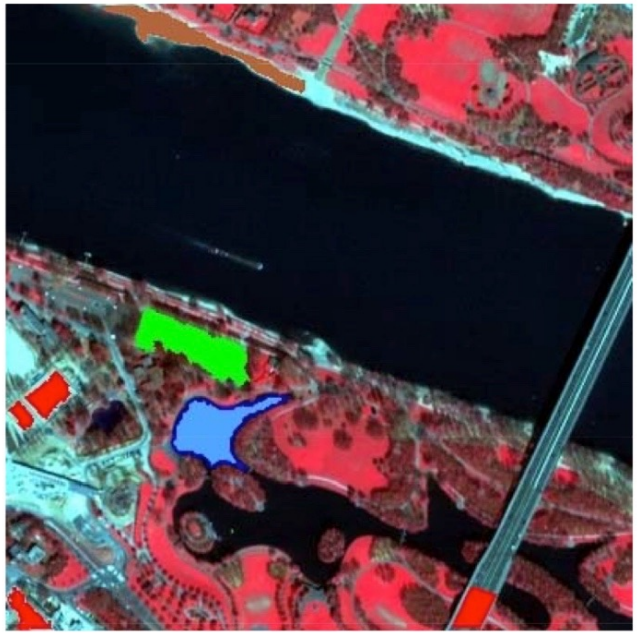
Nearest Neighbor Classification

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

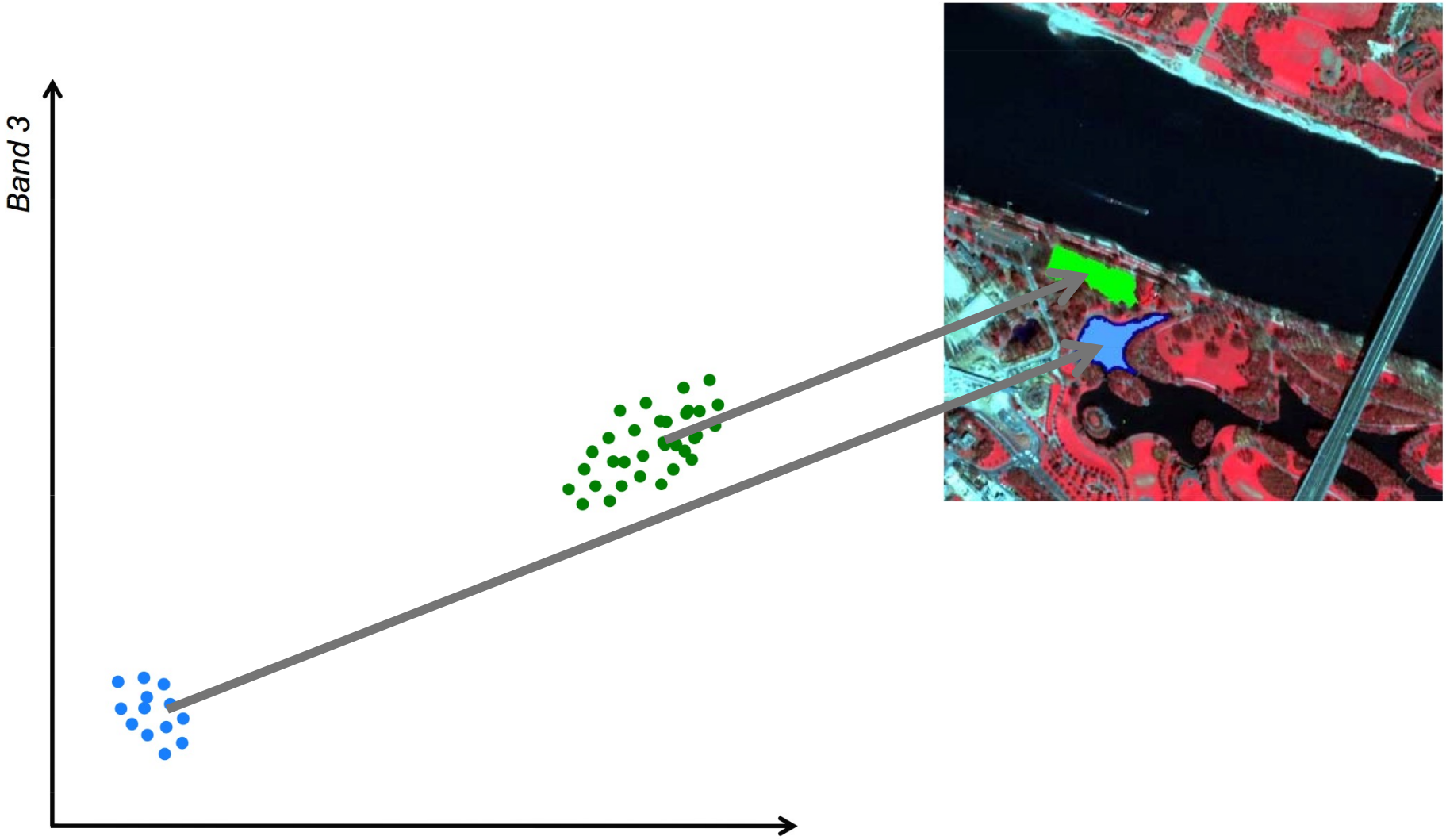
Example: Nearest Neighbor

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



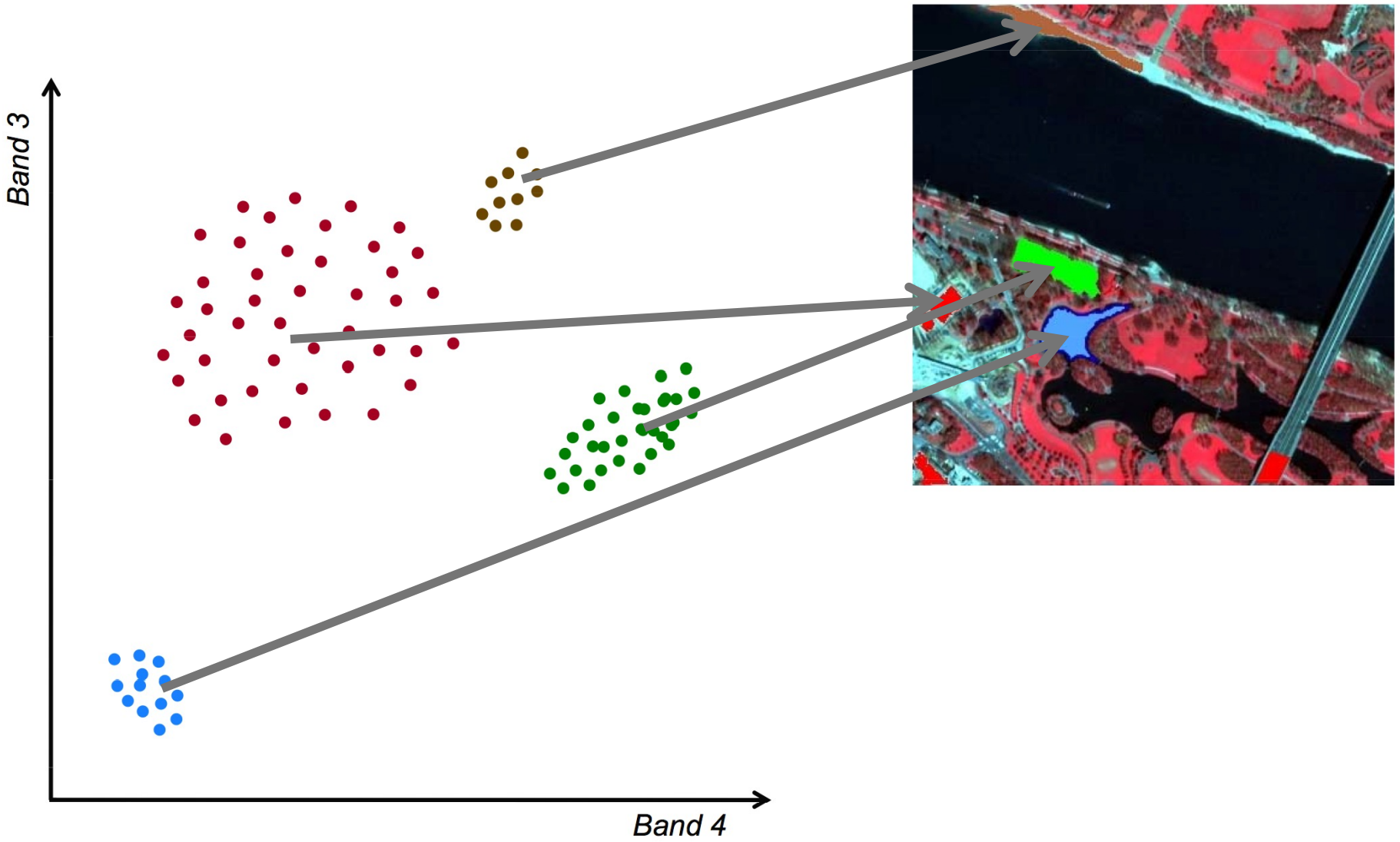
Slide courtesy: Waske/Stefanski

Example: Nearest Neighbor



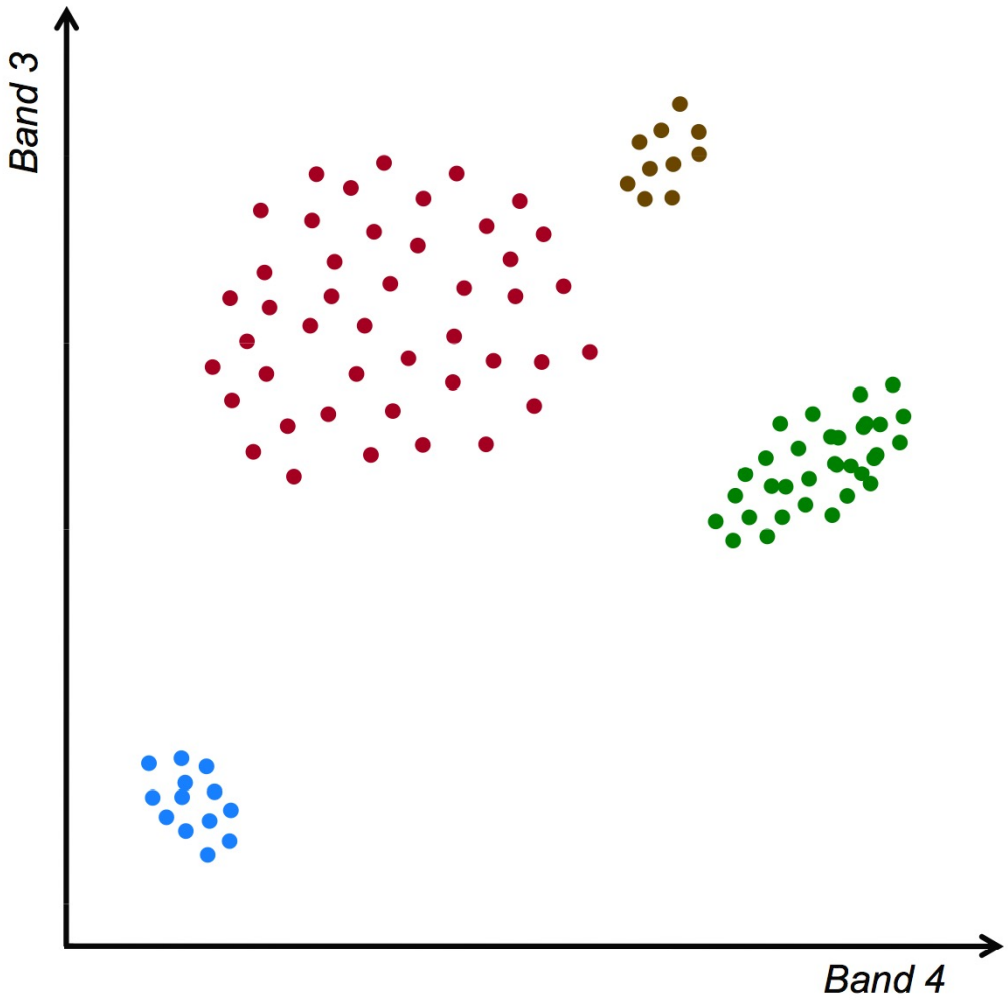
Slide courtesy: Waske/Stefanski

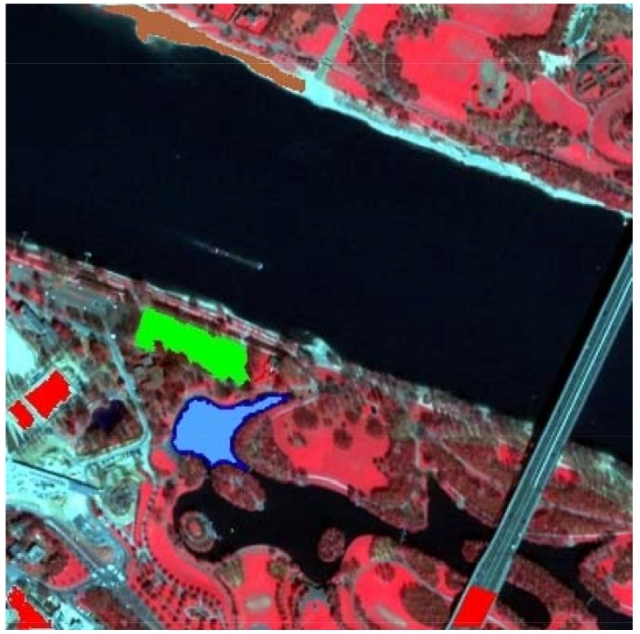
Example: Nearest Neighbor



Slide courtesy: Waske/Stefanski

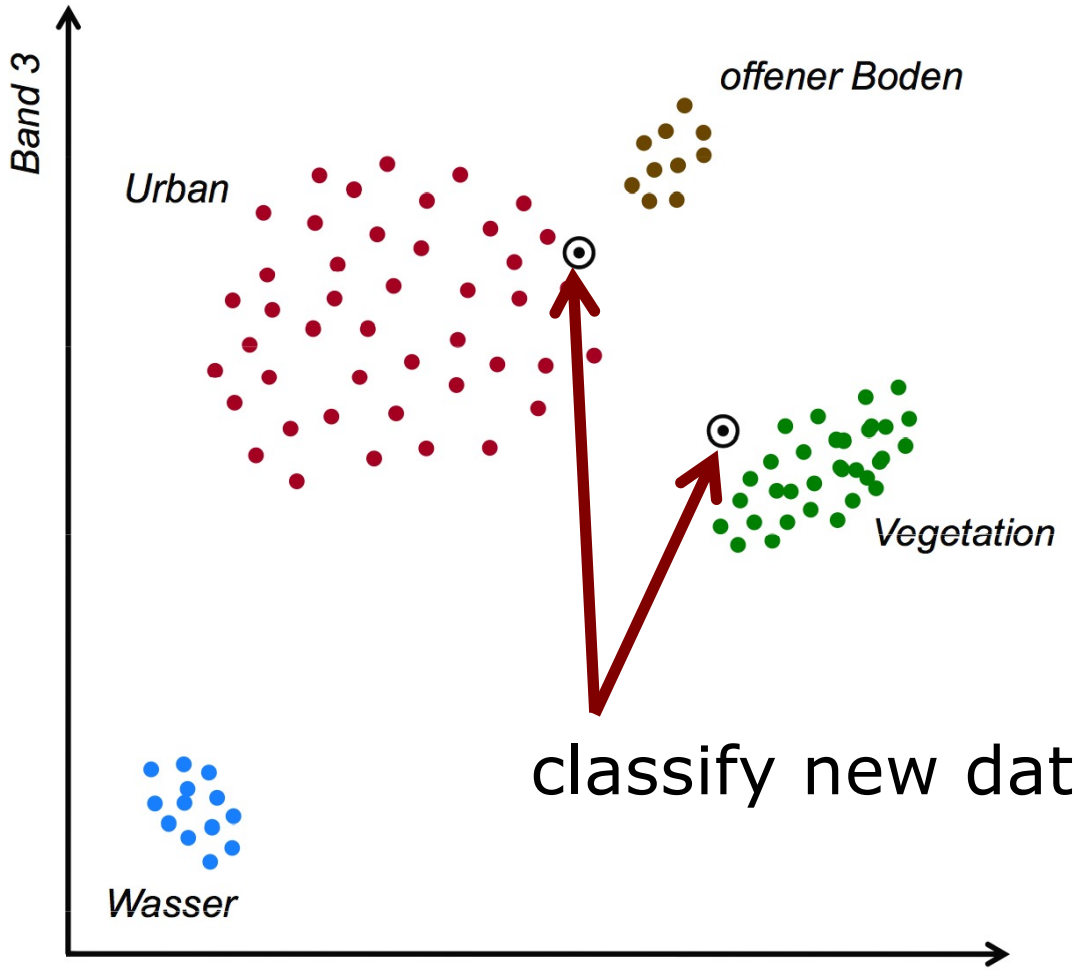
Learning is Completed!





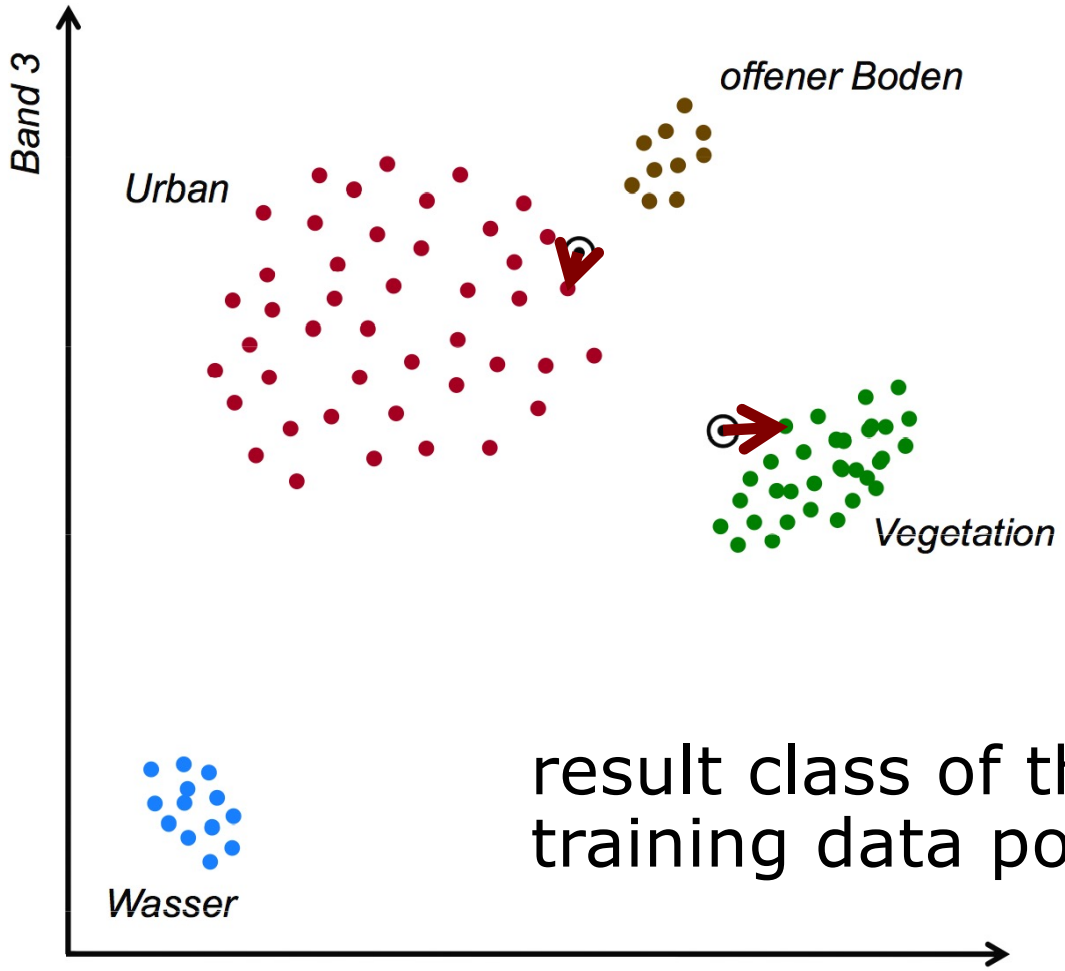
Slide courtesy: Waske/Stefanski

Example: Nearest Neighbor



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

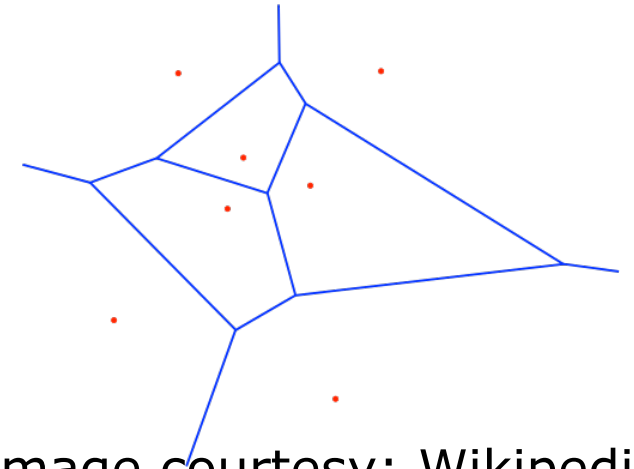


Image courtesy: Wikipedia

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)

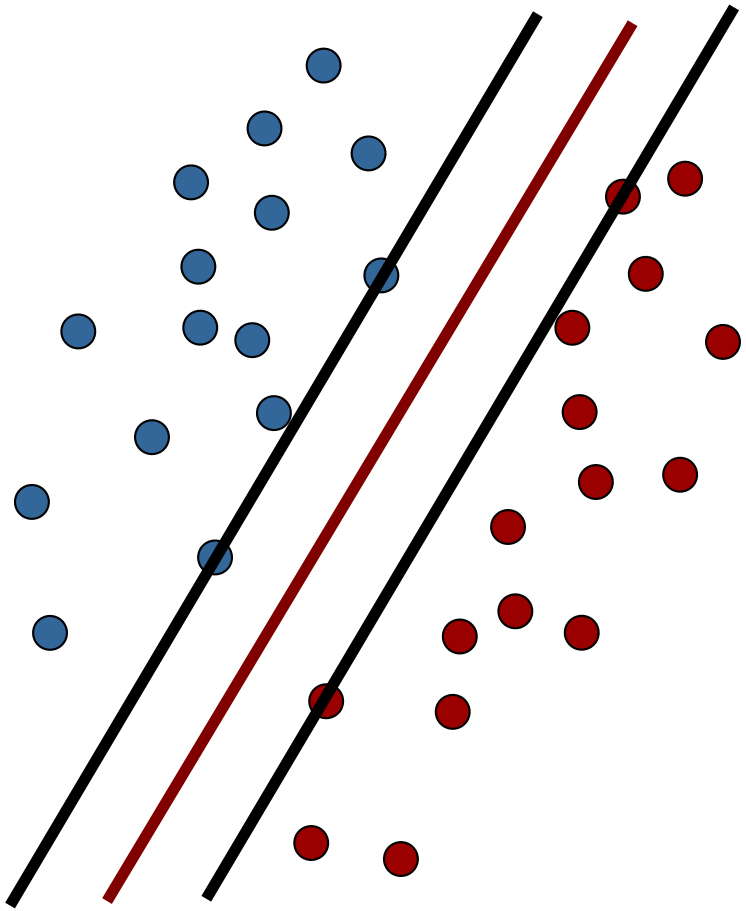
k-Nearest Neighbor Approach

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

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

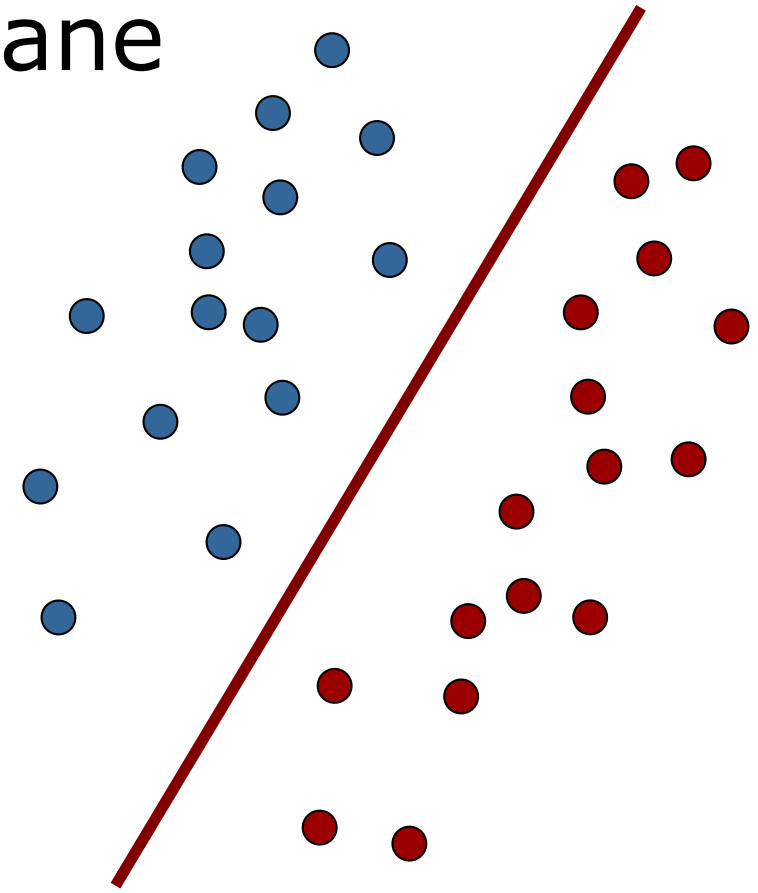
- SVMs seek to maximize the margin between both classes
- Search for the separating hyperplane is formulated as a convex optimization problem
- Optimal solution for computing the hyperplane

Support Vector Machines

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

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

weight vector — data — threshold ("bias")



Support Vector Machines

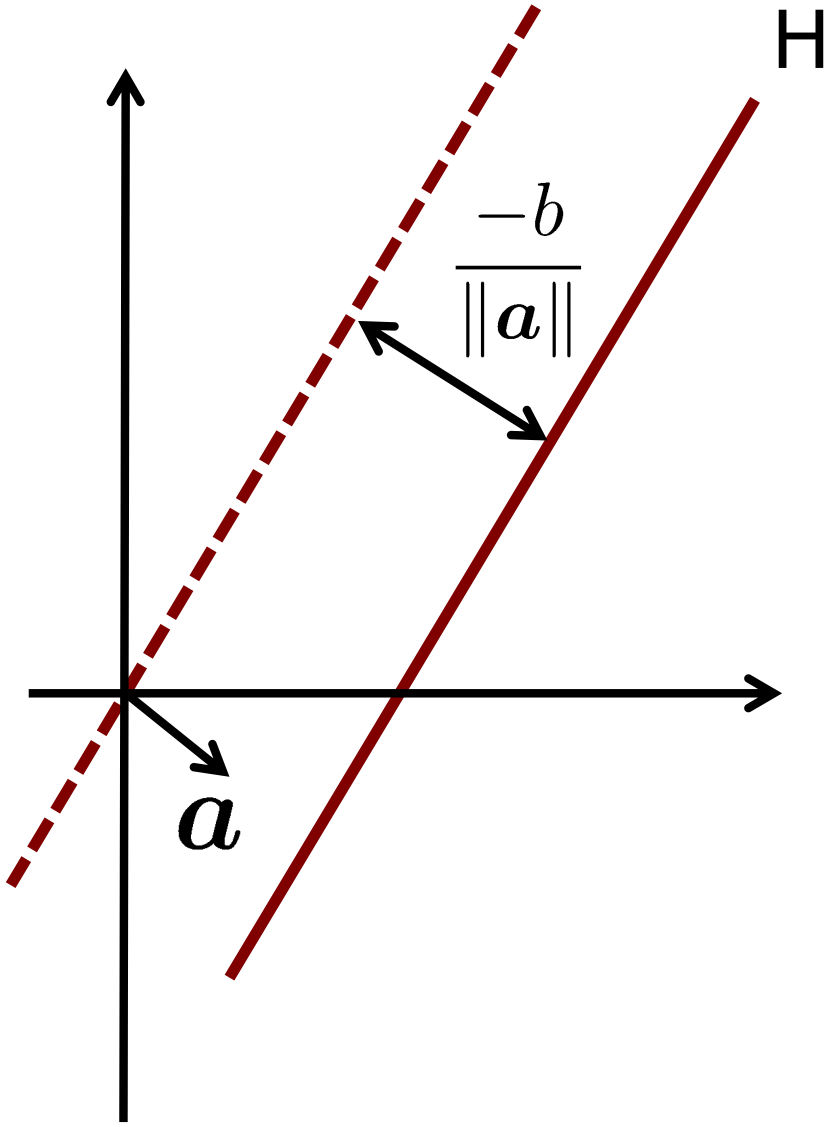
- Linearly separating plane

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

weight vector

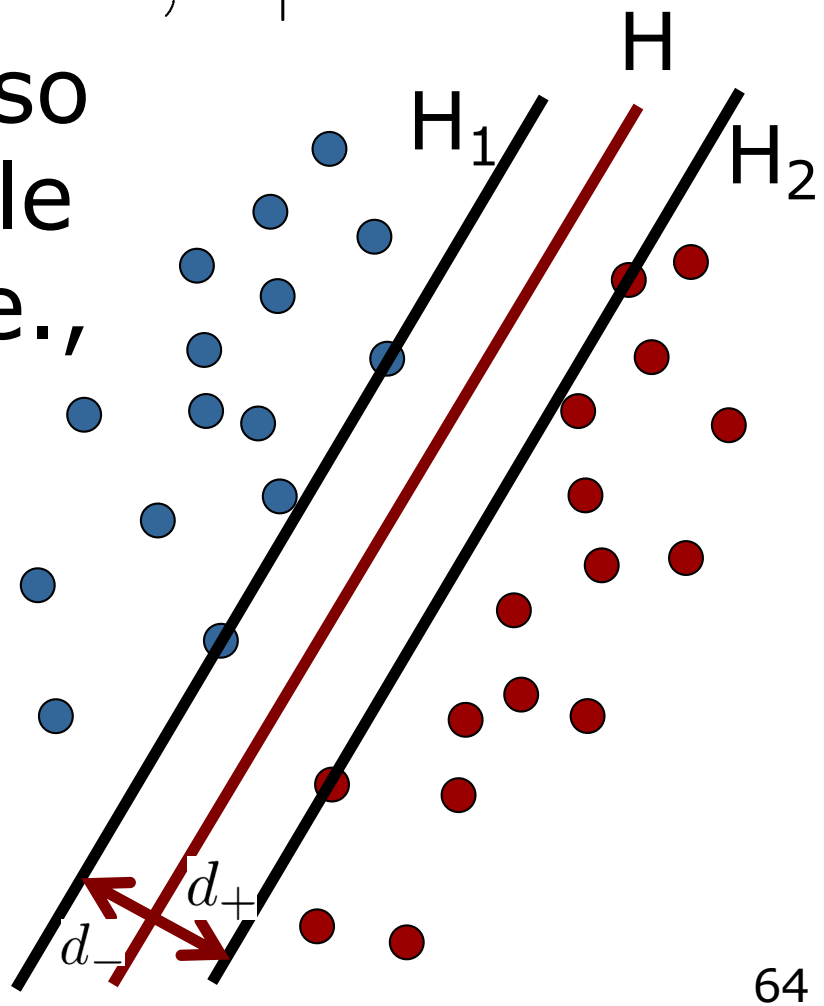
data

threshold ("bias")



Support Vector Machines

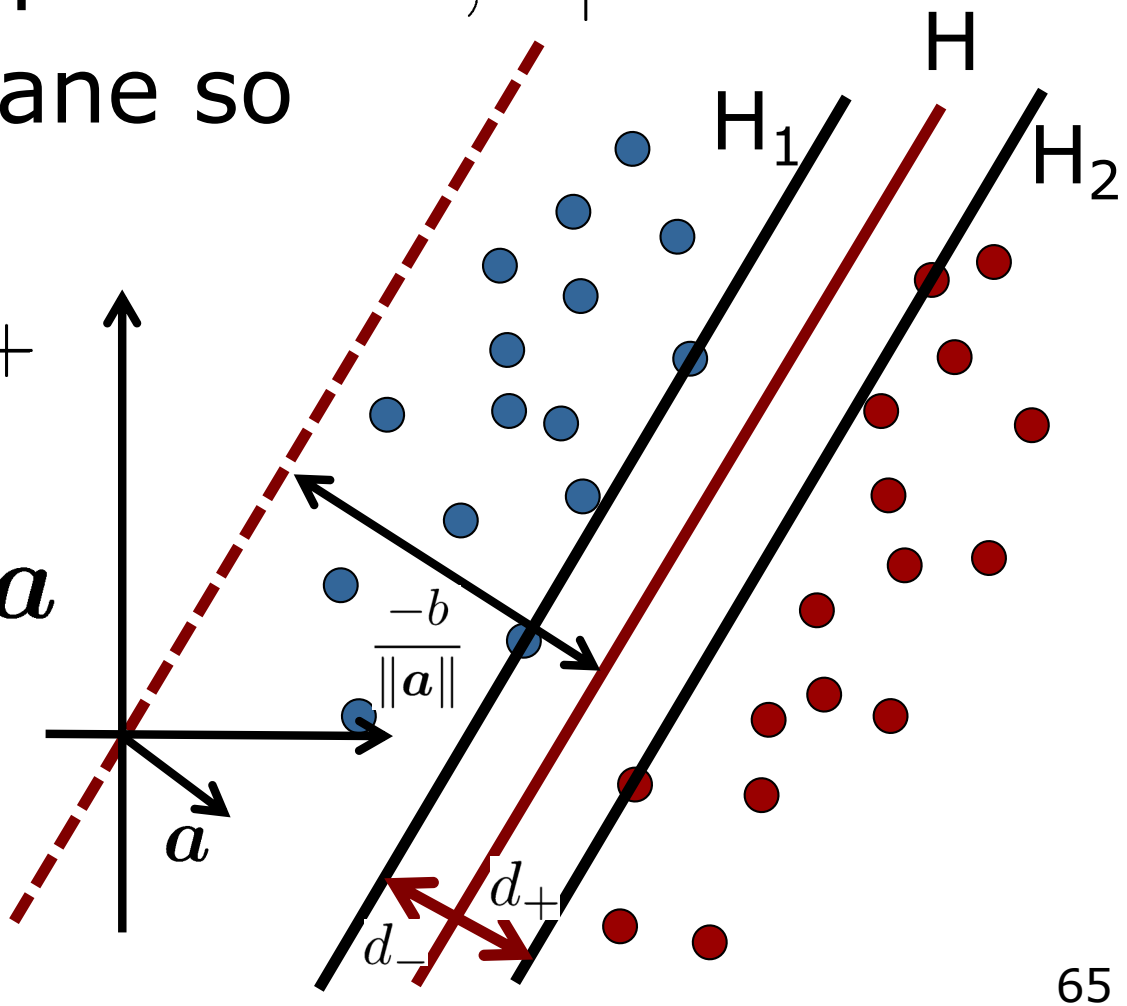
- Distance of the closest neg./pos. example to the plane: d_-, d_+
- Choose hyperplane H so that it lies in the middle between H_1 and H_2, i.e., $d_- = d_+$
- Margin: $d_- + d_+$



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:**

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



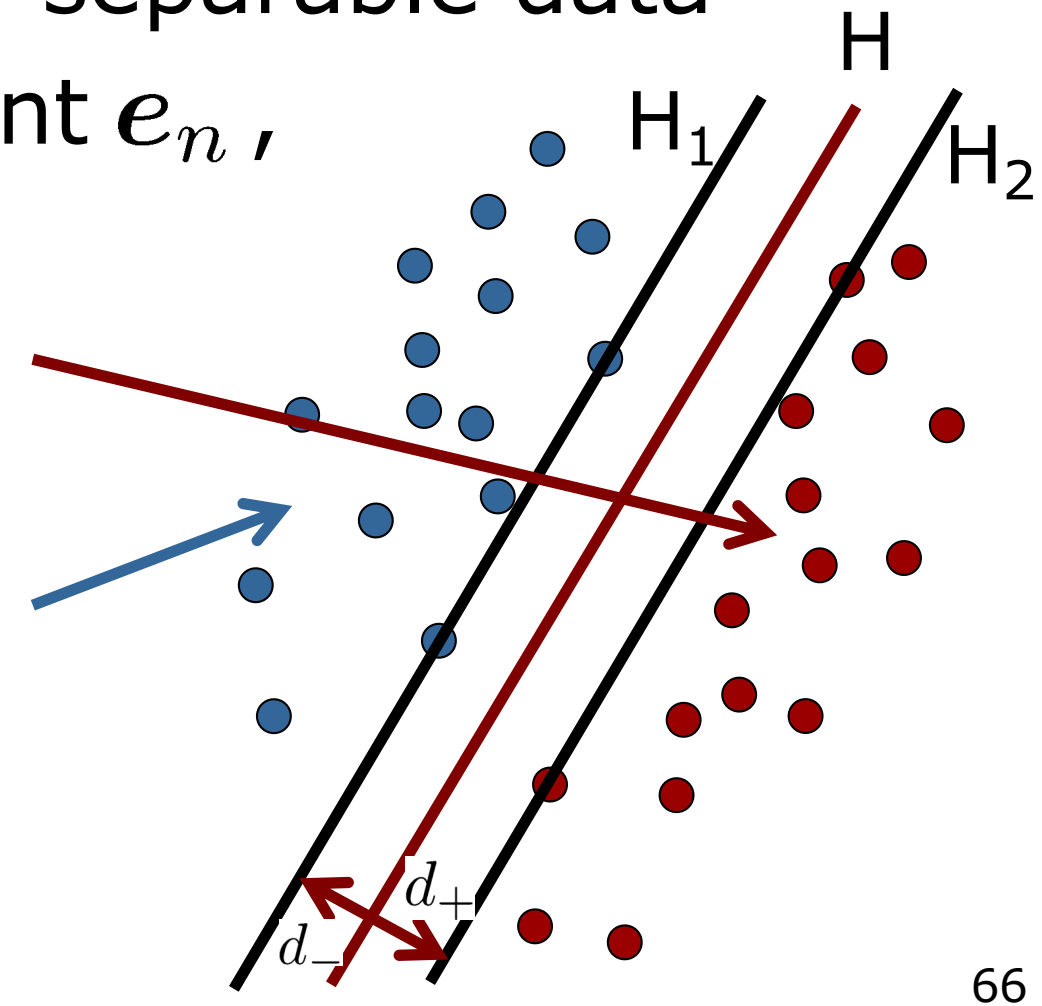
Constrains

- 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

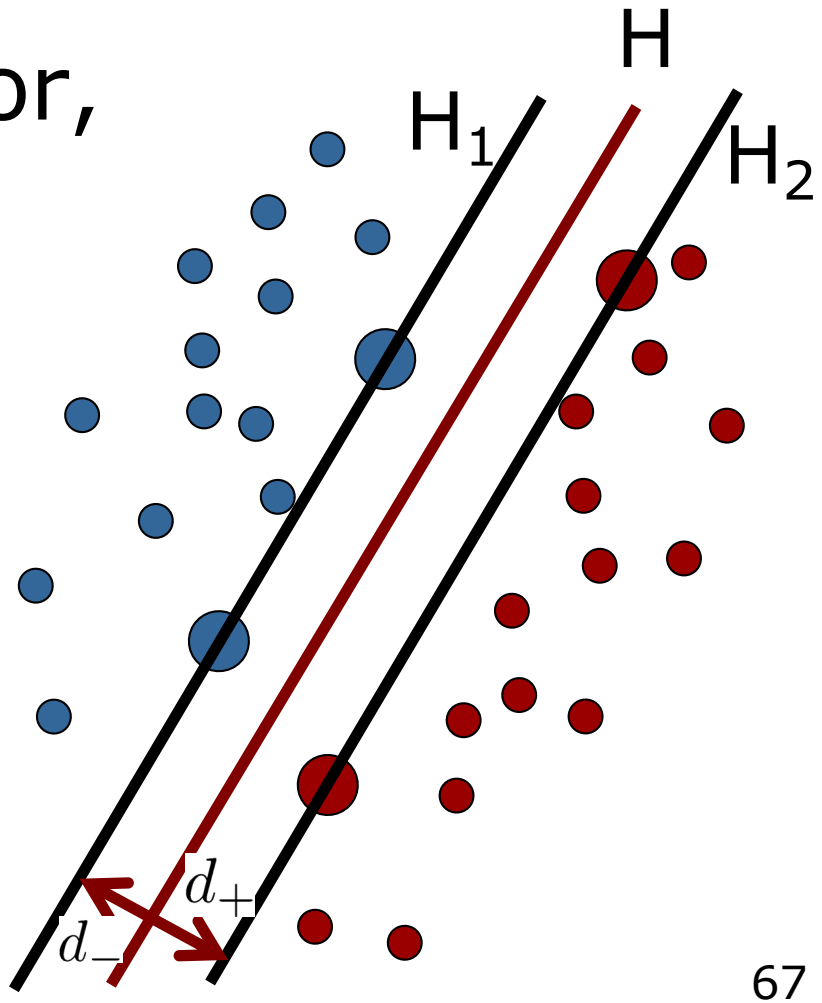
$$\omega_n(\boldsymbol{a}^{\mathsf{T}}\boldsymbol{e}_n + b) = 1$$

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

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

- Can be solved through quadratic programming with linear constraints.

SVM Testing

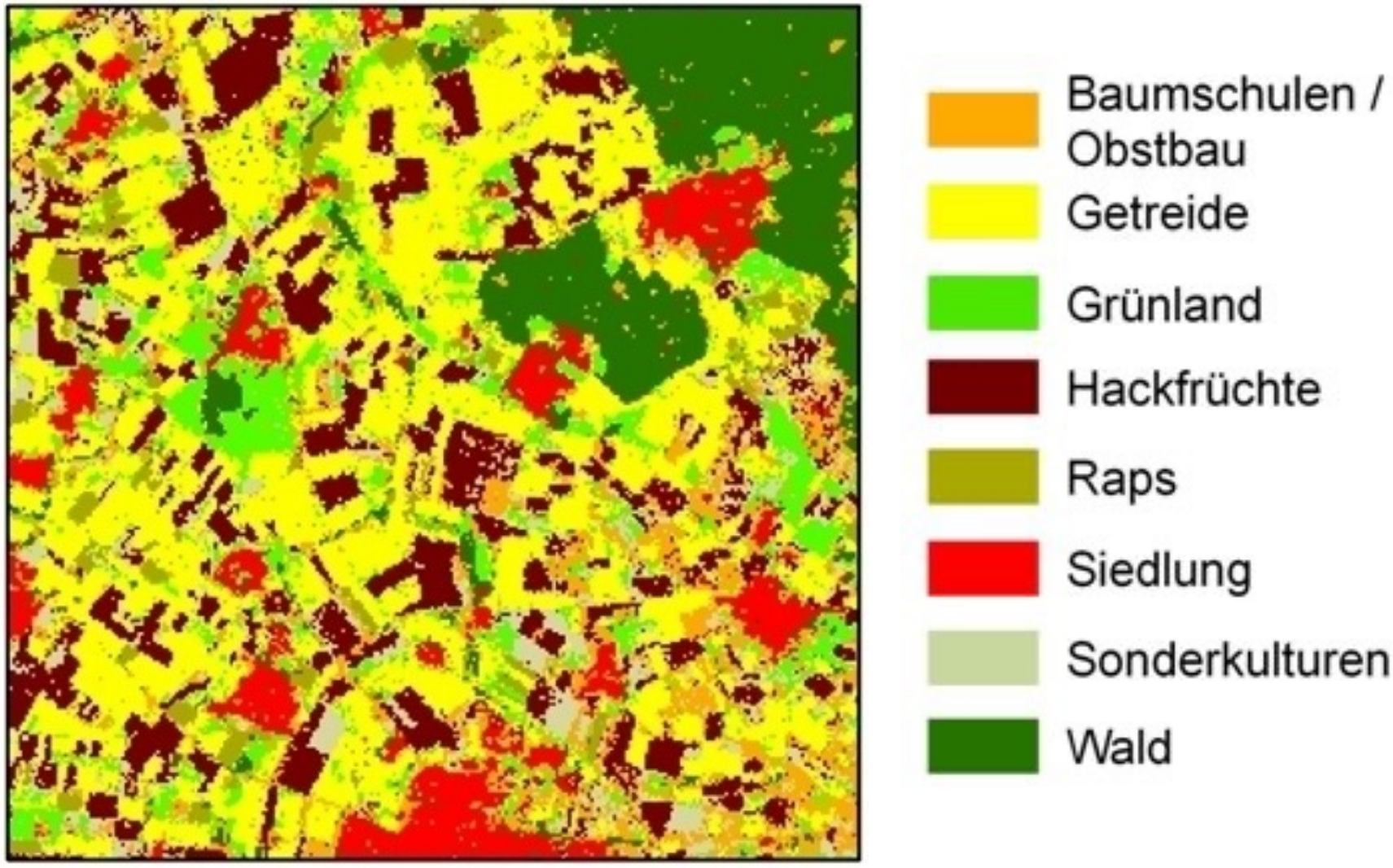
- Classifying an new data point just requires to test on which side of the hyperplane the points is located

$$\omega_{new} = \text{sign}\left(\boldsymbol{a}^{\mathsf{T}}\boldsymbol{e}_{new} + b\right)$$

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)

SVM Classification Example



[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:

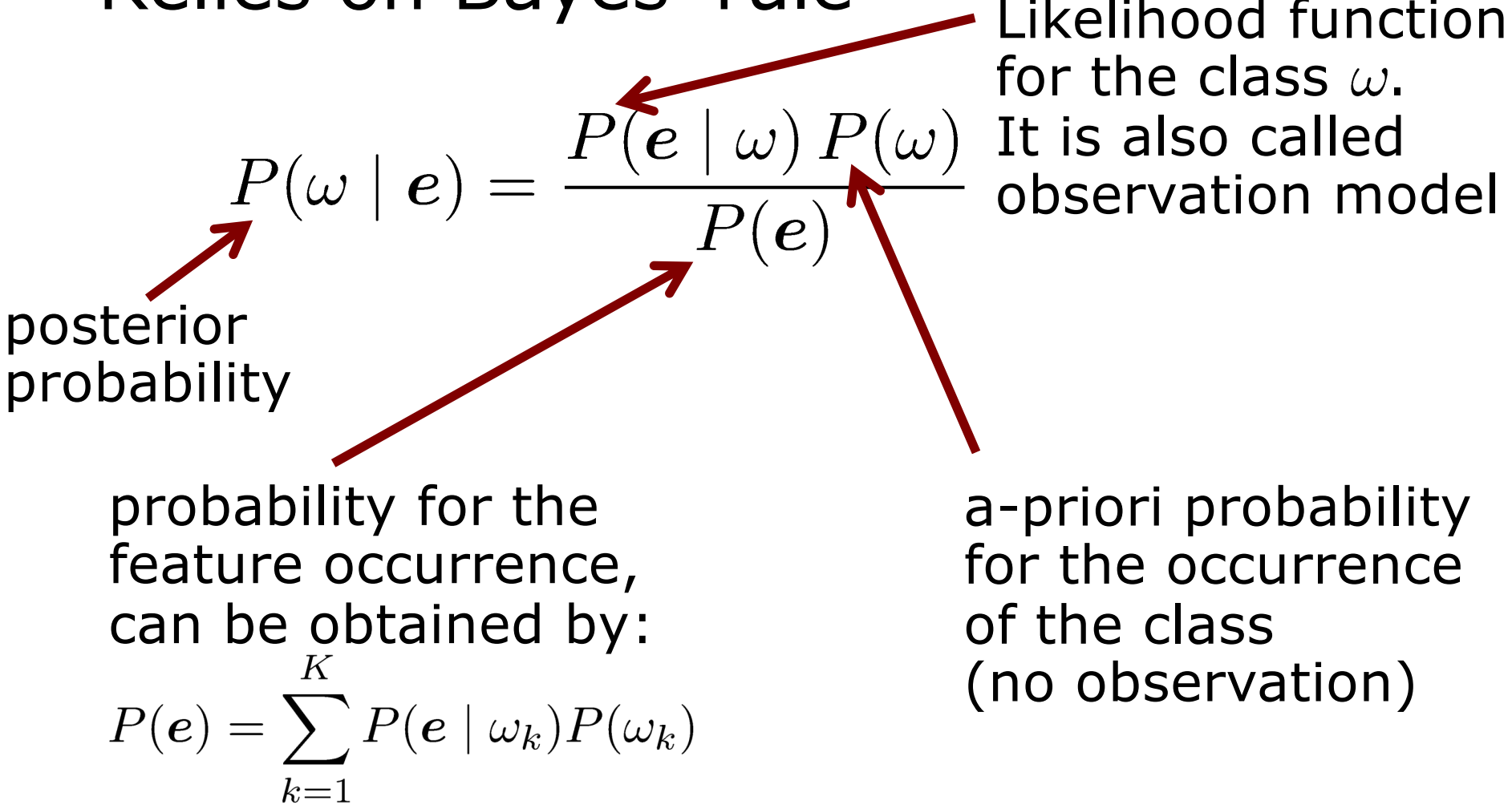
$$P(\omega \mid \boldsymbol{e}) = \frac{P(\boldsymbol{e} \mid \omega)\, P(\omega)}{P(\boldsymbol{e})}$$

class feature

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

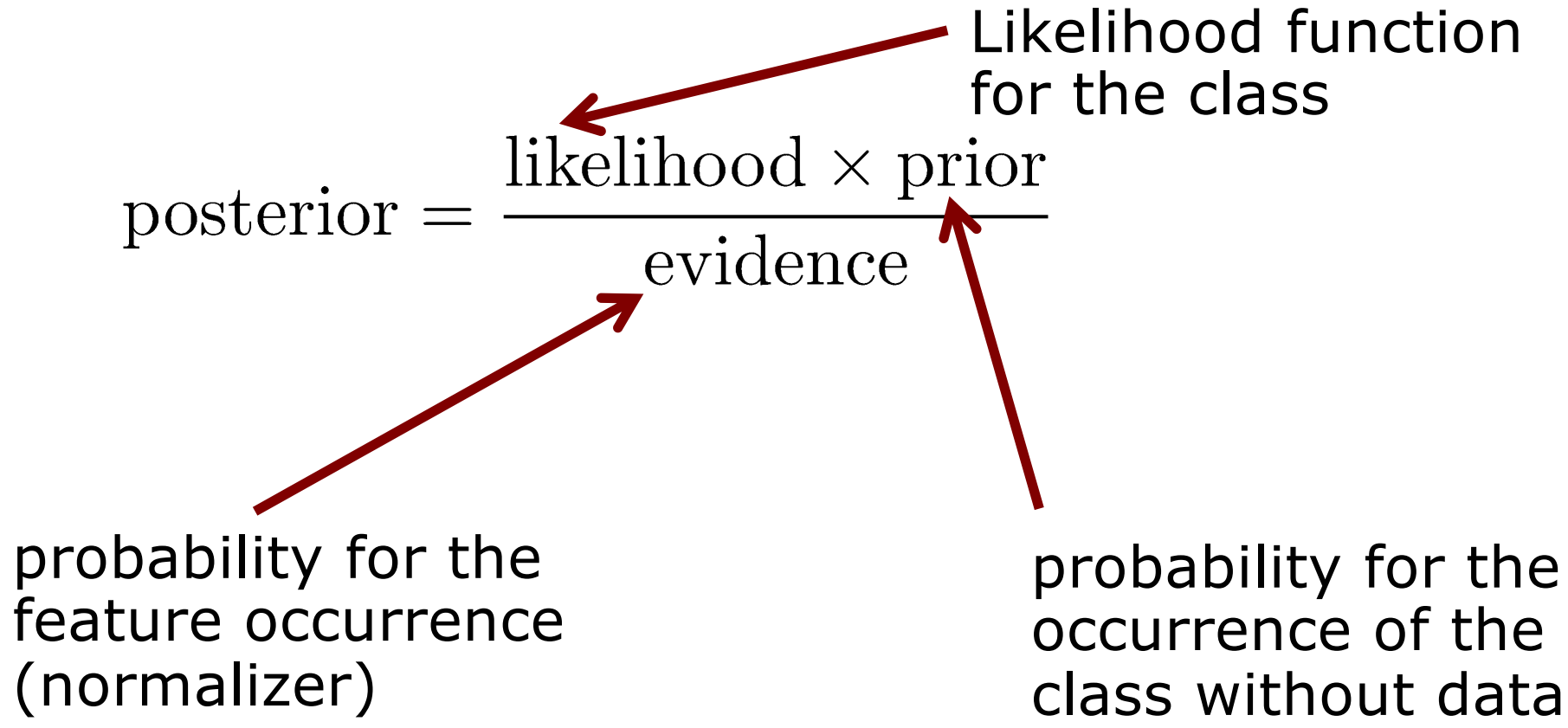
Maximum-A-Posteriori (MAP) Classification

- Relies on Bayes' rule



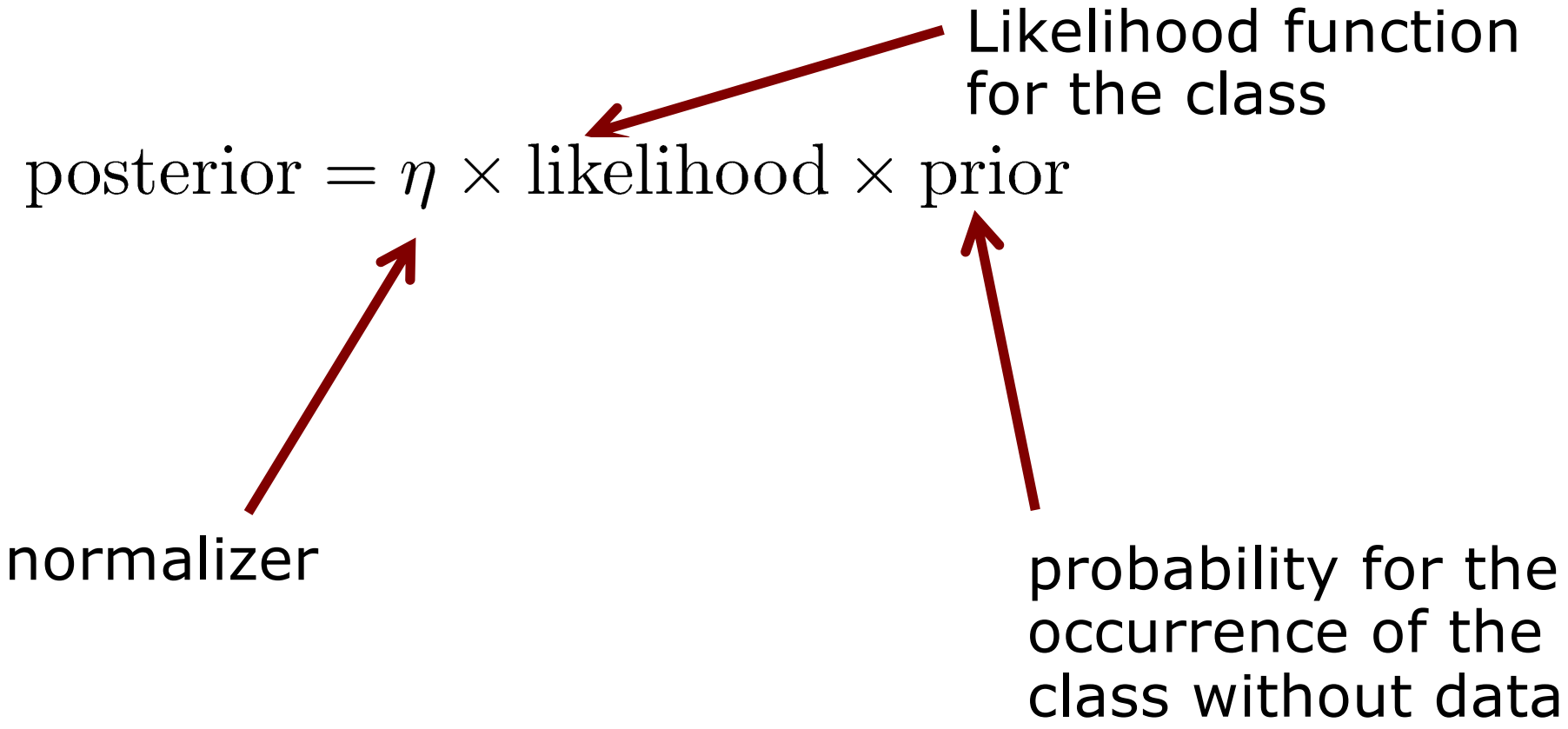
Maximum-A-Posteriori (MAP) Classification

- Relies on Bayes' rule



Maximum-A-Posteriori (MAP) Classification

- Relies on Bayes' rule



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

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

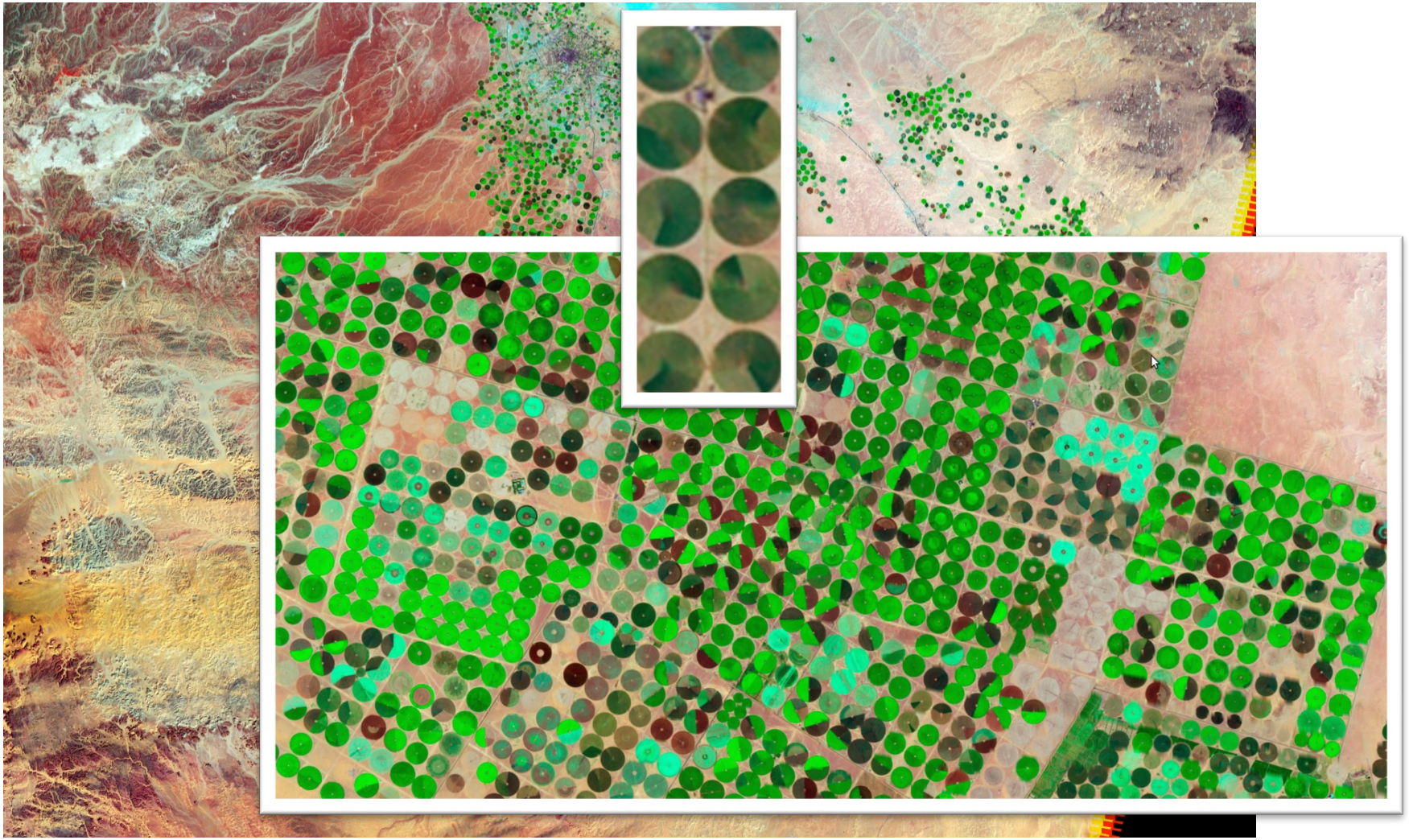
identical for all classes

(thus it can be ignored)

2. Select the MAP class

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

Remote Sensing Example: Land Cover Classification



Slide courtesy: Roscher

Remote Sensing Example: Input and Annotations

Features: RGB

red = arable land;
blue = desert

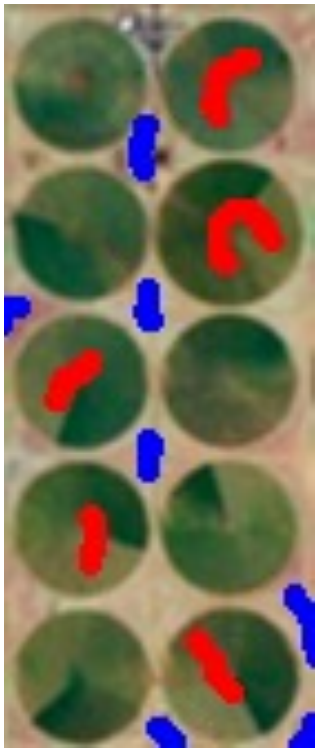


Image courtesy: Roscher

Remote Sensing Example: Feature Space

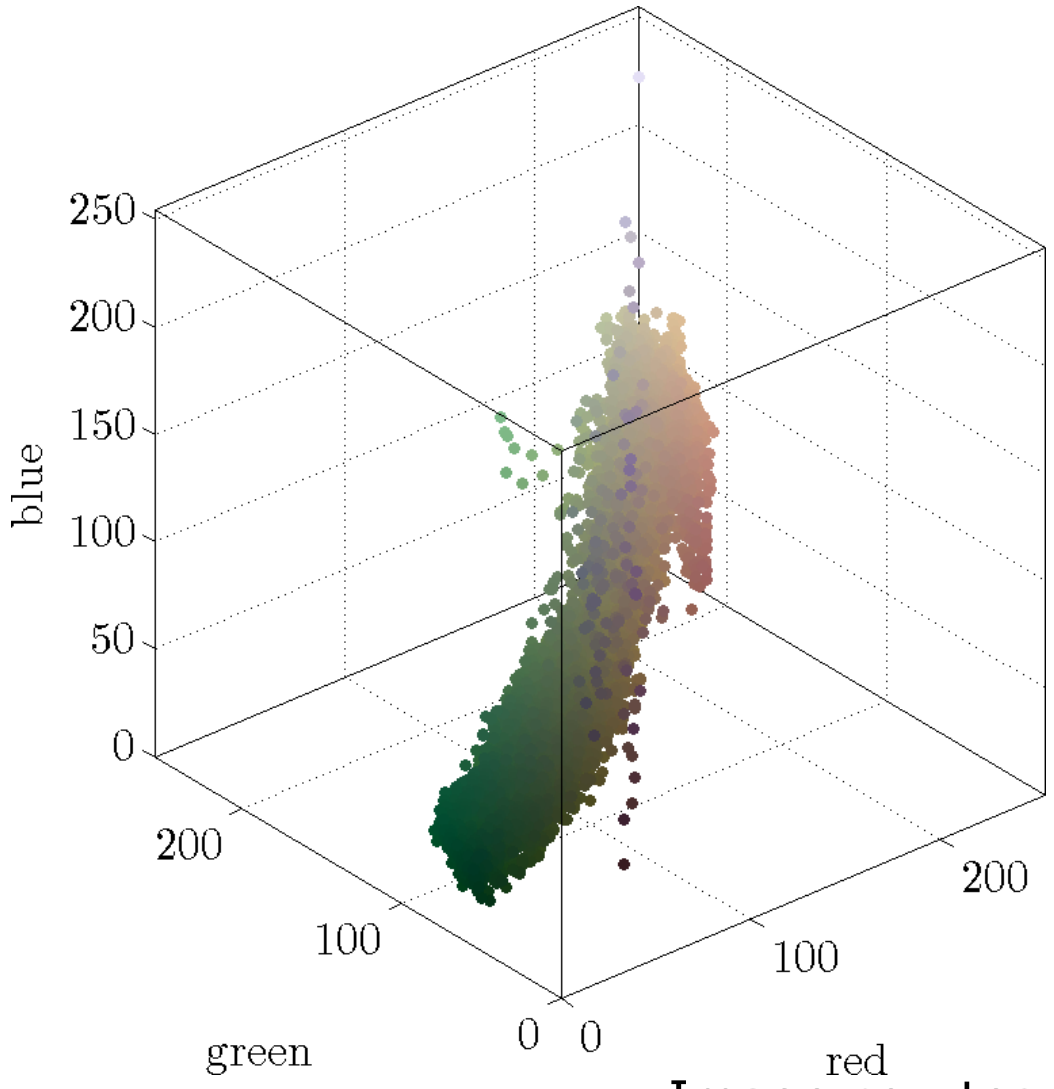


Image courtesy: Roscher

Remote Sensing Example: Training Data Points

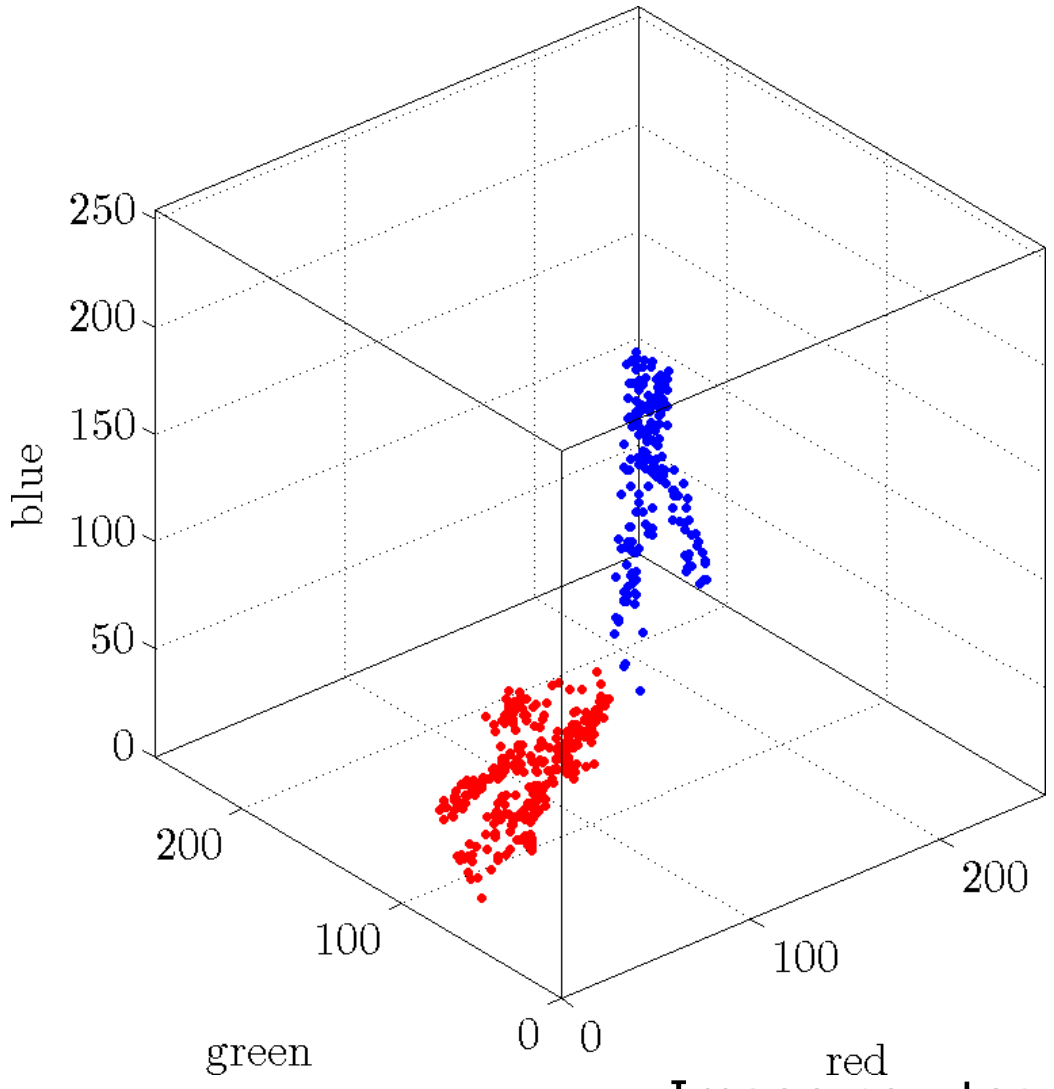


Image courtesy: Roscher

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

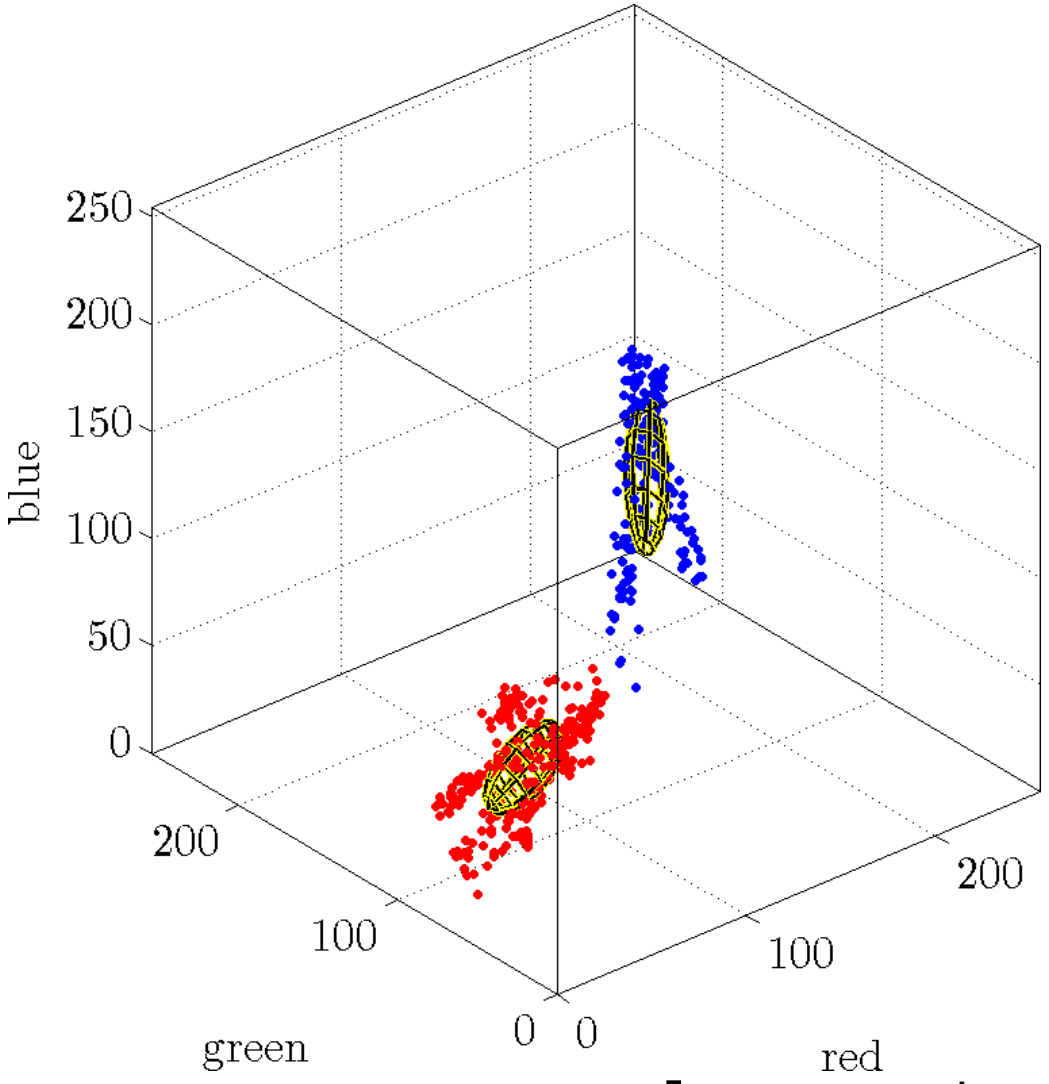


Image courtesy: Roscher

Losses and Risks

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

Scene Understanding

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

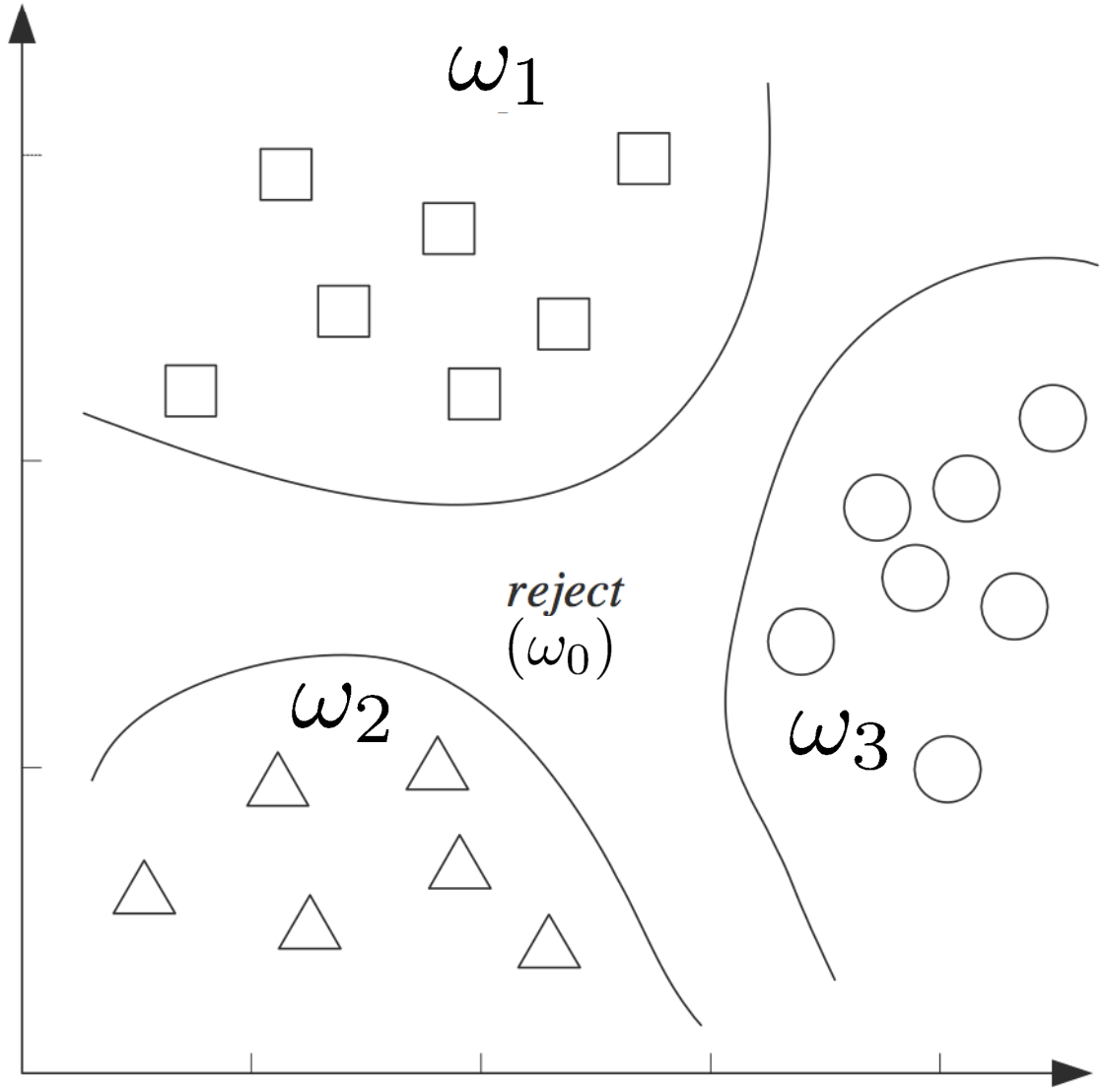
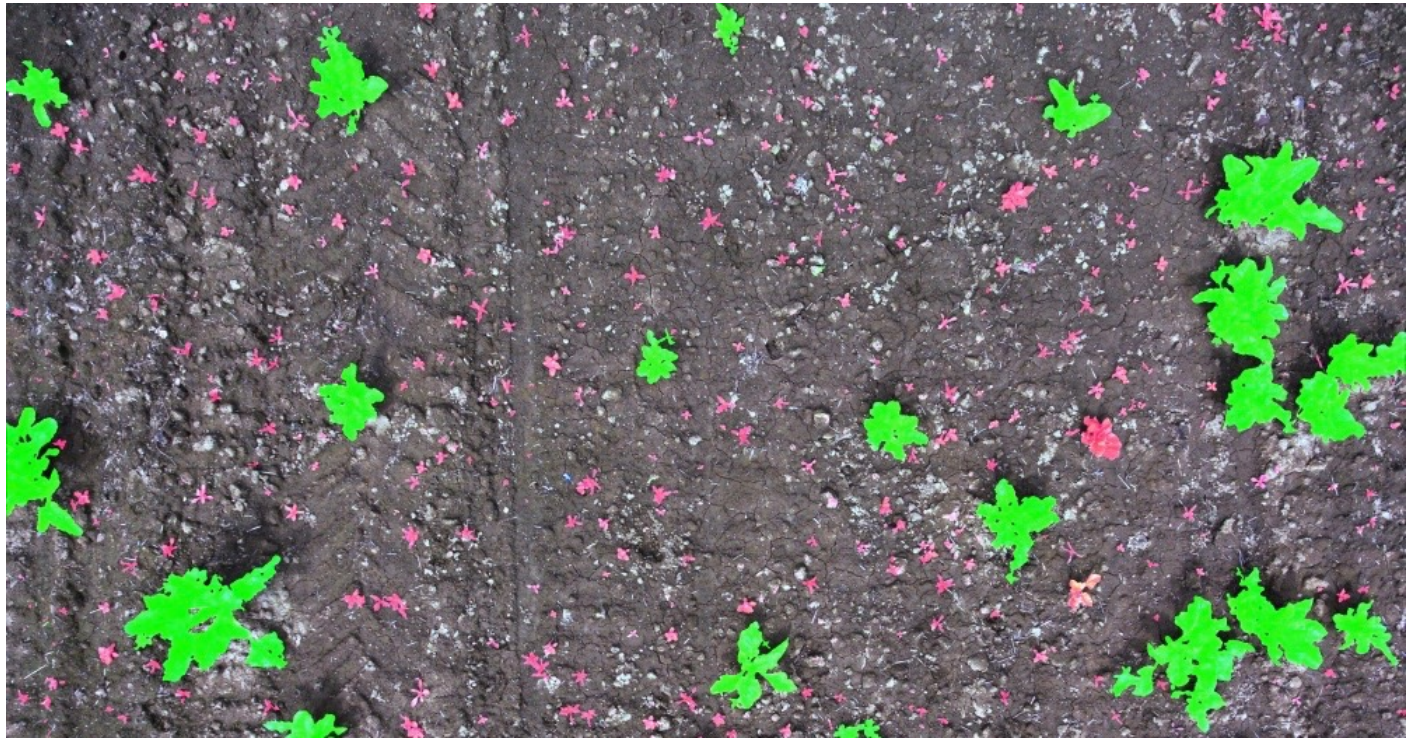


Image courtesy: Aplaydin

Crop Weed Classification



Summary

- Introduction to classification
- Building a simple classifier
- Different types of errors
- Classifier Evaluation
- Nearest neighbor classifier
- Support vector machines
- 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