Data Mining and Machine Learning

Part 3: Classification

Melih Kandemir

University of Southern Denmark

DM566, Spring 2023

SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

- Introduction
- Bias-free Learning?
- Evaluation of Classifiers
- k-Nearest Neighbor Classification
- Summary

SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

Introduction

Bias-free Learning?

Evaluation of Classifiers

k-Nearest Neighbor Classification

Summary

Recommended Reading

Recommended Reading:

- *Mitchell [1997], Chapter 1*

Classification – Supervised Learning

- For some domain $\mathcal{D}$ and a set of classes $C = \{c_1, \ldots, c_k\}$, $k \geq 2$, each $o \in \mathcal{D}$ belongs uniquely to some $c \in C$, i.e., there is a function $f : \mathcal{D} \rightarrow C$ (see Slide **??**).
- Given a set of objects $O = \{o_1, o_2, \ldots, o_n\} \subseteq \mathcal{D}$ and a mapping $(O \rightarrow C) \subset f$ (examples):
  We want to also map any object $o_m \in \mathcal{D} \setminus O$ to C.
- Supervised vs. unsupervised:
  - In clustering, we don't have any information on C.
  - In classification, we have examples (a training set) to guide (supervise) the learning process.

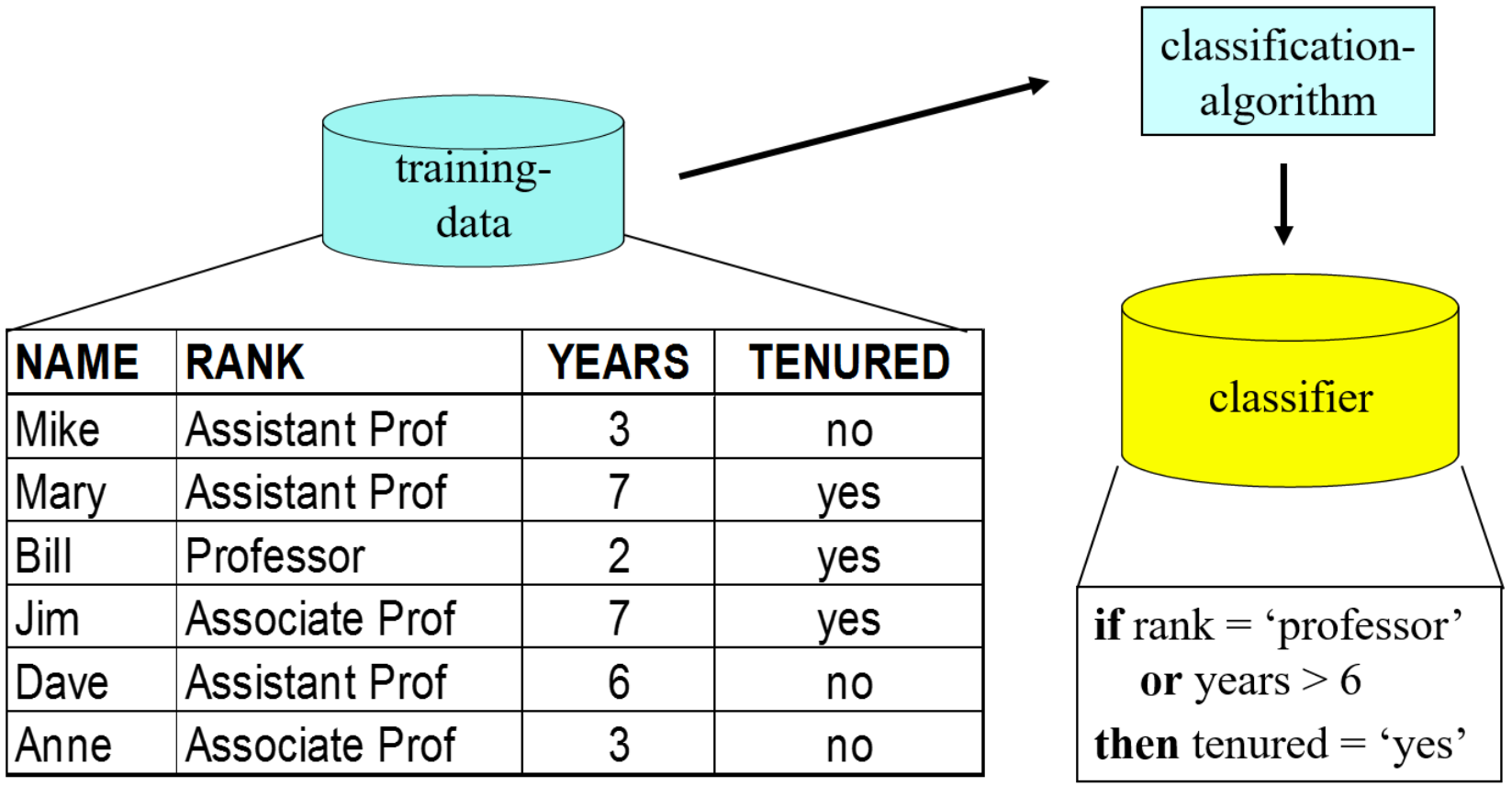
Classification – Supervised Learning

- A classifier is trained on some training set $TR \subseteq O$ to learn the mapping function (a model or hypothesis) $h : \mathcal{D} \rightarrow C$.
- Ideally we have $\forall o \in TR : h(o) = f(o)$ (if not for all, we should have this at least for most examples o).
- After training, the classifier should also work on $\mathcal{D} \setminus TR$ and predict the correct class, i.e., $h \approx f$.
- f is called "the target function".

Assumption 1.1 (The inductive learning assumption)

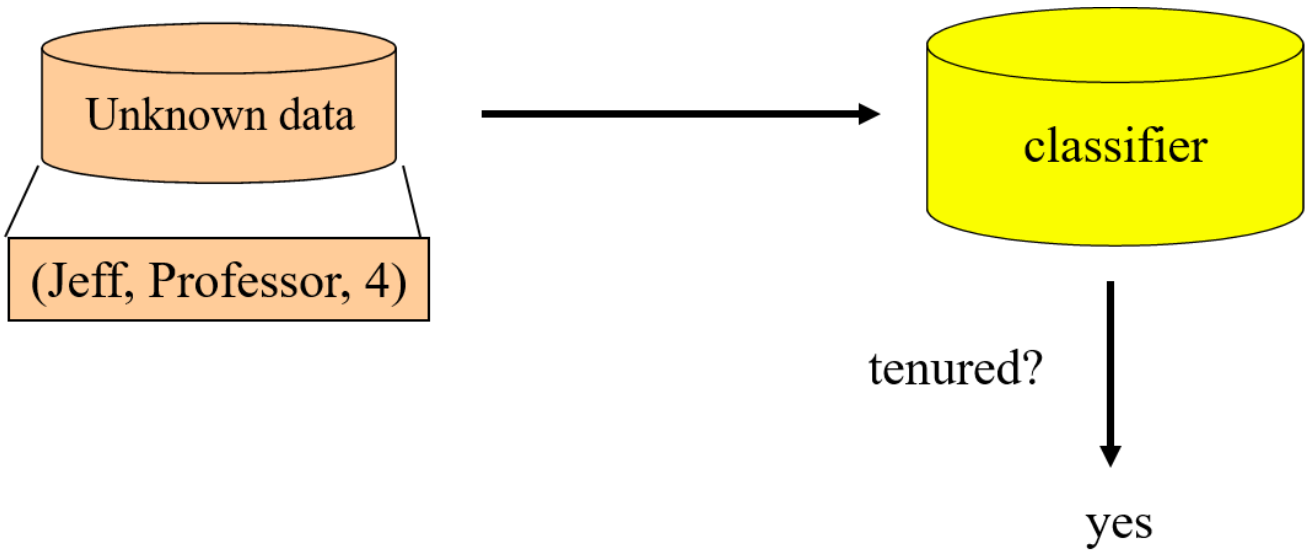
Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other examples.

Training of a Classifier



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

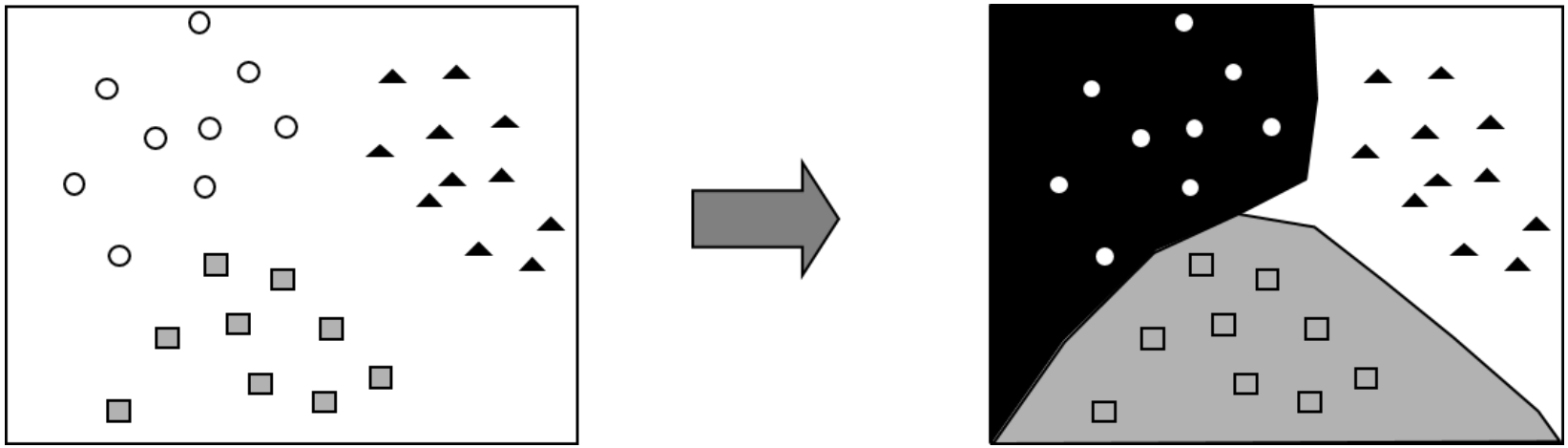
Application of a Classifier



Intuition

- Data $\subset \mathcal{D} \times C$ are generated by some (natural / technical / social / ...) statistical process.
- The observed data are examples for the effect of the process.
- The challenge for the learning algorithm is to generate a model to explain the process.
- h is an approximation of f, a hypothesis to explain the data.
- In the ideal case, the hypothesis h is interpretable and helps to understand the data-generating process.
- Pragmatically, h might be useful for predictions although it might not be interpretable.

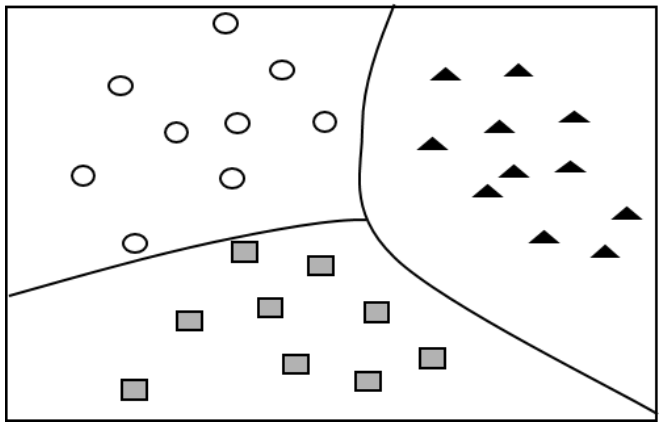
Classification Algorithms and their Hypothesis-Space

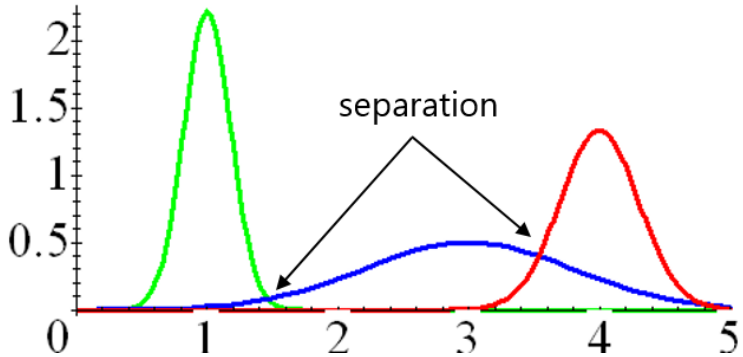


- Based on training data, a classifier typically provides a hypothesis that separates the examples belonging to different classes from each other.
- Each classification algorithm comes with (more or less strict) assumptions on how separation can be achieved or defined.

Classification Algorithms and their Hypothesis-Space

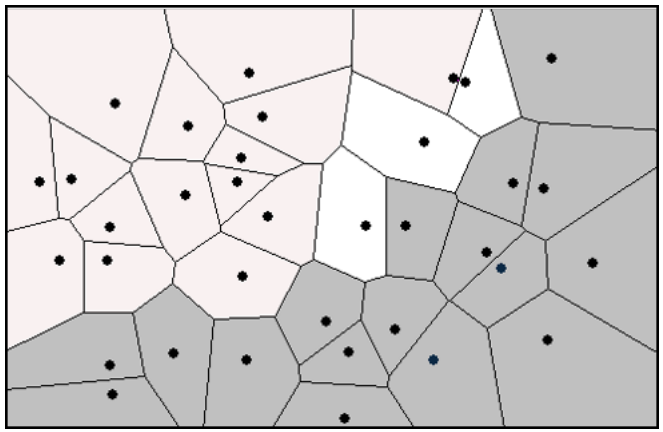
- These assumptions (implicitly) define a hypothesis space $\mathcal{H}$, the space of hypotheses that could possibly be learned by the given algorithm.





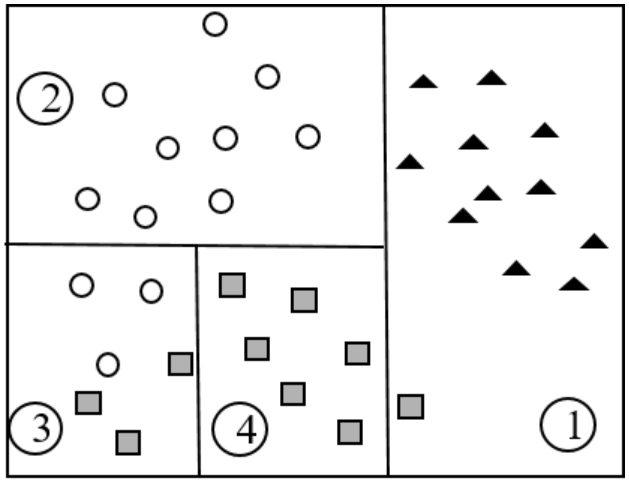
Classification Algorithms and their Hypothesis-Space

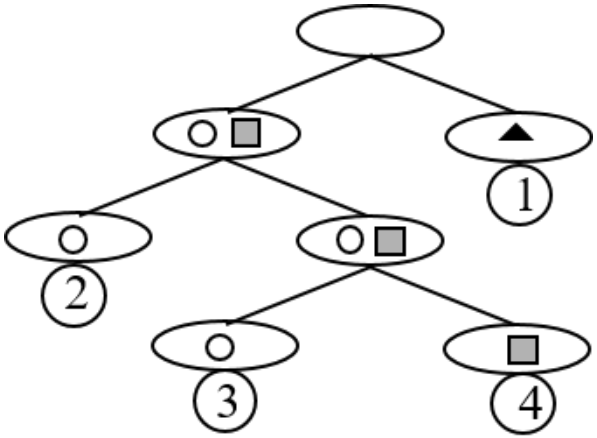
- These assumptions (implicitly) define a hypothesis space $\mathcal{H}$, the space of hypotheses that could possibly be learned by the given algorithm.



Classification Algorithms and their Hypothesis-Space

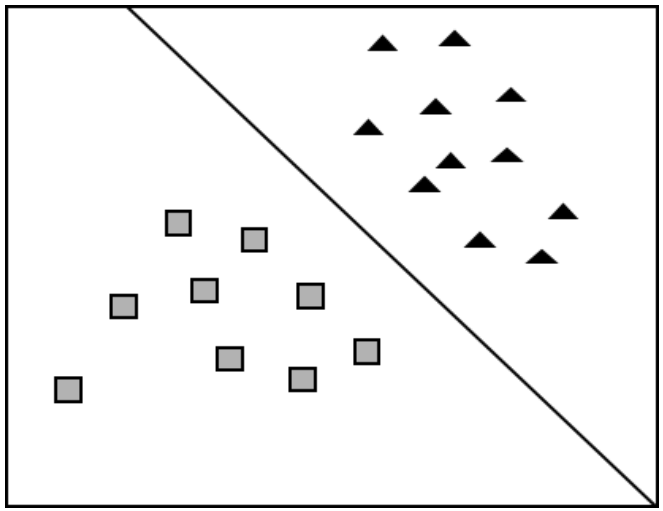
- These assumptions (implicitly) define a hypothesis space $\mathcal{H}$, the space of hypotheses that could possibly be learned by the given algorithm.





Classification Algorithms and their Hypothesis-Space

- These assumptions (implicitly) define a hypothesis space $\mathcal{H}$, the space of hypotheses that could possibly be learned by the given algorithm.



SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

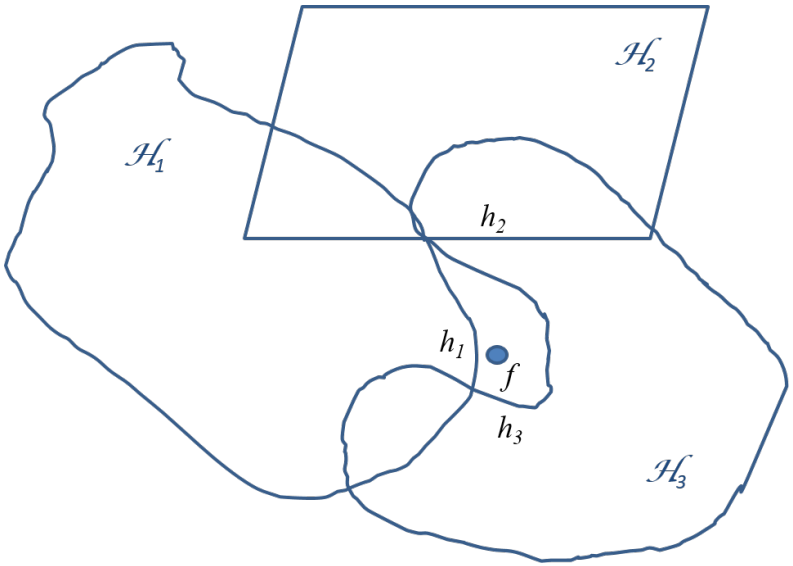
Recommended Reading

Recommended Reading:

- *Mitchell [1997], Chapter 2*

Hypothesis Space and Bias

- The hypothesis space is a manifestation of the so-called "bias".
- Some algorithm, due to the restrictions of its hypothesis space, will prefer certain hypotheses, i.e., the algorithm has a bias.



- In general, we do not know if $f \in \mathcal{H}$, i.e., whether or not we can actually approximate f (well enough) with some (or any) specific classification algorithm.
- Note that a hypothesis space can be infinite and yet restricted.

Example: Some Concept of Enjoying Sport

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

f: Sky $\times$ AirTemp $\times$ Humidity $\times$ Wind $\times$ Water $\times$ Forecast $\rightarrow$ {Yes, No}

- Is there a general concept when to enjoy sport?
- To learn a concept from the example data, we need some assumption on this concept.
- Here we can assume, for example, that the concept is a conjunction of some attribute values.
- We thus have a finite hypothesis space, containing all possible conjunctions of concrete attribute values.

Coverage of Data Instances by Hypotheses

For this toy example, we define:

Example An example or instance x is a possible day, described as $x \in \text{Sky} \times \text{AirTemp} \times \text{Humidity} \times \text{Wind} \times \text{Water} \times \text{Forecast} = \mathcal{D}$

Positive/negative example An example x is positive, if $f(x) =$ Yes, negative otherwise.
(Note that we know f on the training data only.)

Hypothesis A hypothesis defines a subset of $\mathcal{D}$.
We write a hypothesis as vector containing

- specific values for attributes
- wildcards ('?') indicating that for some attribute any attribute value is acceptable
- $\emptyset$ indicating that no attribute value is acceptable.

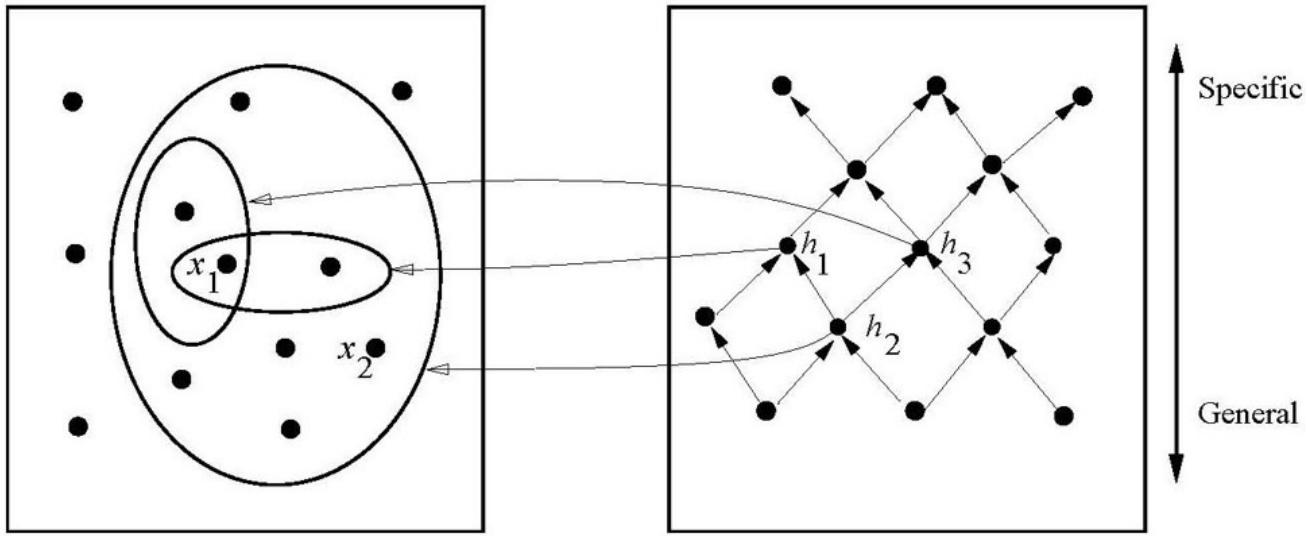
Coverage of Data Instances by Hypotheses

- For example the hypothesis $\langle \text{Sunny}, ?, ?, ?, \text{Cool}, ? \rangle$ defines all examples where Sky=Sunny and Water=Cool.
- If we are interested in hypotheses about positive examples, we can interpret $\langle \text{Sunny}, ?, ?, ?, \text{Cool}, ? \rangle$ as a rule:

```
if Sky=Sunny and Water=Cool
then EnjoySport=Yes
```

- An example x *satisfies* a hypothesis h, if and only if $f(x) =$ Yes.

The Assumptions of a Learning Algorithm Define the Hypothesis Space



$x_1 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Cool}, \text{Same} \rangle$
$x_2 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Light}, \text{Warm}, \text{Same} \rangle$

$h_1 = \langle \text{Sunny}, ?, ?, \text{Strong}, ?, ? \rangle$
$h_2 = \langle \text{Sunny}, ?, ?, ?, ?, ? \rangle$
$h_3 = \langle \text{Sunny}, ?, ?, ?, \text{Cool}, ? \rangle$

Definition 1.1

For any two hypotheses, h_j and h_k, over X, h_j is *more general than or equal to* h_k if and only if:

$$\forall x \in X : h_k(x) = \text{Yes} \Rightarrow h_j(x) = \text{Yes}$$

Basic Algorithm

Algorithm 1.1 (Find-S [Mitchell, 1997])

1. *Initialize h to the most specific hypothesis in $\mathcal{H}$*
2. *For each positive training instance x*
   For each attribute constraint a_i in h
   If the constraint a_i is satisfied by x
   Then do nothing
   Else replace a_i in h by the next more general constraint that is satisfied by x
3. *output hypothesis h*

Discussion:

- Finds the most specific hypothesis consistent with the positive examples in the training data – is it the only consistent hypothesis?
- Why should we prefer more specific hypotheses over more general ones?

Possible Hypotheses Under the Assumption of a Conjunctive Concept

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

The following conjunctive hypotheses describe the concept "Yes" correctly for the training data:

$\langle \text{Sunny}, ?, ?, ?, ?, ? \rangle$

$\langle ?, \text{Warm}, ?, ?, ?, ? \rangle$

$\langle \text{Sunny}, \text{Warm}, ?, ?, ?, ? \rangle$

$\langle \text{Sunny}, ?, ?, \text{Strong}, ?, ? \rangle$

$\langle ?, \text{Warm}, ?, \text{Strong}, ?, ? \rangle$

$\langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ? \rangle$

Reduce the Bias?

For some other training data, our model assumptions seem too strict:

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
Sunny	Warm	Normal	Strong	Cool	Change	Yes
Cloudy	Warm	Normal	Strong	Cool	Change	Yes
Rainy	Warm	Normal	Strong	Cool	Change	No

- No consistent hypothesis is possible under our assumptions:
- the most specific hypothesis for positive examples is $\langle ?, \text{Warm}, \text{Normal}, \text{Strong}, \text{Cool}, \text{Change} \rangle$
- this hypothesis covers also the negative example

Bias-free Learning

- Should we allow for more complex/generous assumptions to reduce the bias (i.e., to get rid of restrictions of the hypothesis space)?
- Allow disjunctions? Negations?
- A disjunctive hypothesis
  "if Sky=Sunny *or* Sky=Cloudy, then Yes"
  can list all positive examples.
- We could actually cover *any* concept with an arbitrarily complex hypothesis.
- Is this what we want?

Bias-free Learning is Futile

- Being able to cover *any* concept means, we don't rely on any assumption regarding possible concepts.
- The most complex hypothesis would describe the training data perfectly well: we can learn by heart all the examples.
- Learning by heart means we can perfectly predict all classes correctly for the *training examples* but we don't have any concept to *generalize to unseen data*.
- In general, a bias is necessary to avoid learning by heart.

SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

Recommended Reading

Recommended Reading:

Best coverage related to our course:

- *Witten et al. [2011], Chapter 5*

The corresponding parts in other textbooks are either rather short or contain more advanced concepts as well:

- *Zaki and Meira Jr. [2020], Chapter 22*
- *Tan et al. [2006], Chapter 4.5 (4.6)*
- *Tan et al. [2020], Chapter 3.6*
- *Mitchell [1997], Chapter 5*

The Problem of Evaluation

- A database $\mathcal{D}$ is representing a domain by the sample of available data, there is a function $f : \mathcal{D} \rightarrow C$, mapping each object to a class $c_i \in C$.
- $O \subseteq \mathcal{D}$ is the set of objects where we know about the class (i.e., we know $f(o)$ for all $o \in O$, but not for any $o \in \mathcal{D} \setminus O$).
- Let h be a classifier (model, hypothesis), trained on a training set $TR \subseteq O$.

Problem:

- We want a good quality (performance, approximation of the target function f) of h over $\mathcal{D}$, yet we cannot know anything about the quality of h over $\mathcal{D} \setminus O$.

The Problem of Evaluation

- h has been optimized for TR.
- If we test h on TR, we will typically get an optimistic estimate of the performance over $\mathcal{D}$.
- The phenomenon that h performs better on TR than on $\mathcal{D}$ overall is called *overfitting*:
  - Often, the amount of training data is not sufficient to generalize reliably (“small sample size bias”).
  - If the data sample is too small to truly represent the domain, there is no sufficient ground to reject overspecialized hypotheses.
  - In general, the weaker the bias of a learning algorithm, the more susceptible is it to overfitting.

The Separation of Training and Testing Data

To get a more realistic estimate of the performance of some classifier over $\mathcal{D}$, we separate training data $TR \subset O$ and test data $TE \subset O$, $TR \cap TE = \emptyset$.

- TR is used to train the classifier (fit the model, select the hypothesis).
- TE is used to evaluate the classifier.
- The purpose is to estimate the performance (success-/error-rate) of the classifier. Therefore:
  - TR and TE need to be independent ($TR \cap TE = \emptyset$).
  - Both, TR and TE should represent the classification problem as good as possible.
  - For some learning problems (benchmark datasets), a separation into TR and TE is available.
- Problem: If it is not clear whether O is already too small to allow for a good generalization, we do not really want TR to be even smaller.

m-fold Cross-Validation

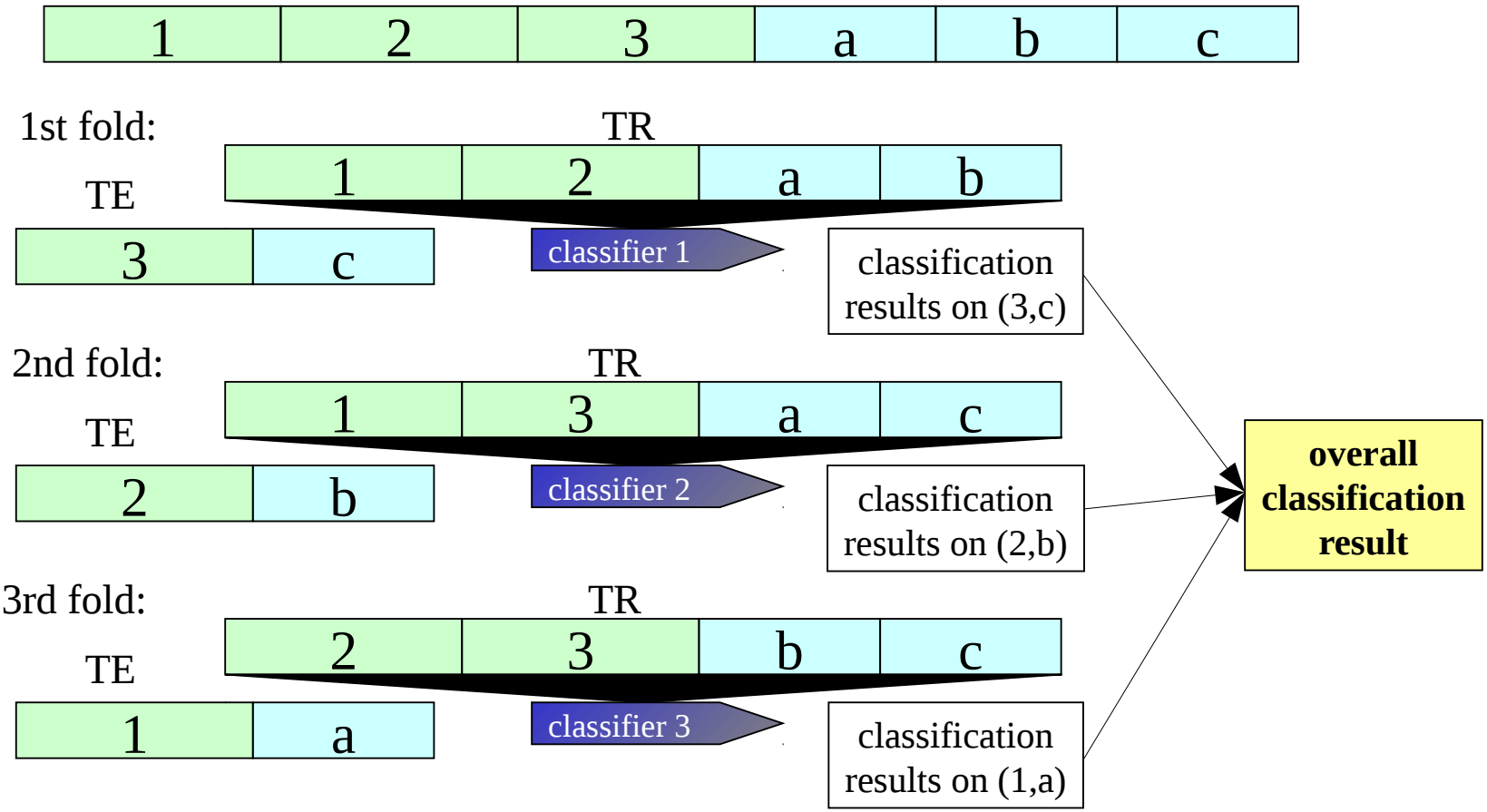
Motivation: We want to use as much data as possible for training, and as much as possible for testing.

Algorithm 1.2 (m-fold cross-validation)

1. *Separate the set O in m equal-size, mutually disjoint subsets.*
2. *Get m different pairs of TR and TE by using each of the m subsets as TE once and the remaining $m-1$ subsets for training.*
3. *On these m pairs of TR and TE, train and test m independent classifiers.*
4. *Average the m observed performances.*
5. *Repeat 1-4 several times.*

m-fold Cross-Validation

For one iteration with $m = 3$, we have T_1 with class information a, T_2 with class information b, and T_3 with class information c:



Stratification

Stratification aims at representing the class proportions in each fold.

- minimum requirement: each class should be present in the training set.
- stratified cross-validation: the distribution of classes within each training and test set should reflect the distribution of the classes in $\mathcal{O}$

Standard approach (rule of thumb):
10-fold, stratified cross-validation, repeated 10 times

Note that:

The evaluation procedure has the purpose of estimating the performance on $\mathcal{D} \setminus \mathcal{O}$. In order to get the best possible classifier, we would use all available labeled data ($\mathcal{O}$) for training.

The Bootstrap

In the bootstrap procedure, a training set is created from O by drawing with replacement:

- We take $|O|$ objects from O, where the same object could be drawn several times.
- A sample TR contains on average 63% of the objects in O. Some are present several times in TR, some ($\approx 37\%$) are not present at all:
  - For $|O| = n$, an individual object in O has a chance of being drawn of $\frac{1}{n}$ each turn, that is, it is *not* drawn with probability $1 - \frac{1}{n}$.
  - After n draws, a specific object has not been drawn with probability $\left(1 - \frac{1}{n}\right)^n$
  - For large n: $\left(1 - \frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$, hence this procedure is also called “the 0.632 bootstrap”

Leave-one-out or Jackknife Test

Algorithm 1.3 (Leave-one-out)

- $\forall o_j \in O$: *take* o_j *as test object for a classifier trained on* $O \setminus \{o_j\}$.
- *average the performance estimate over all test objects*

Discussion:

- For $|O| = n$, this is an n-fold cross-validation.
- Pro: no random effect
- Con: stratification is not possible
- In general, the Jackknife test leads to a relatively pessimistic performance estimate.

Confusion Matrix

The confusion matrix represents the number of correctly and incorrectly predicted classes per actual and per predicted class:

predicted class

actual class	class 1	class 2	class 3	class 4	class 5
class 1	35	1	1	1	4
class 2	0	31	1	1	5
class 3	3	1	50	1	2
class 4	1	0	1	10	2
class 5	3	1	9	15	13

correctly predicted objects

Quality Measures for Classifiers

Given a classifier h, a training set $TR \subseteq O$, and a test set $TE \subseteq O$. $f(o)$ is the actual class of o, $h(o)$ is the class predicted by the classifier h. Then we have:

accuracy of h on TE:

$$\mathrm{acc}_{TE}(h) = \frac{|\{o \in TE | h(o) = f(o)\}|}{|TE|}$$

true classification error:

$$\mathrm{err}_{TE}(h) = \frac{|\{o \in TE | h(o) \neq f(o)\}|}{|TE|}$$

apparent classification error:

$$\mathrm{err}_{TR}(h) = \frac{|\{o \in TR | h(o) \neq f(o)\}|}{|TR|}$$

Focus on Individual Classes

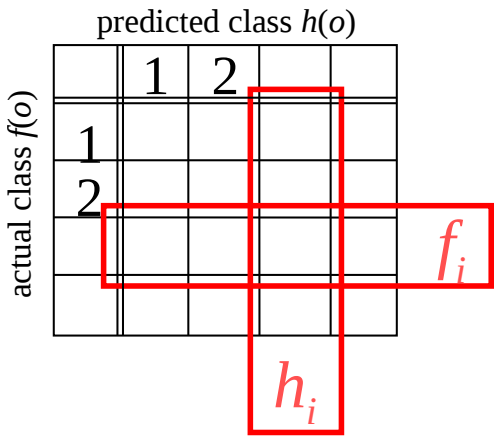
If we focus on a single class (the “positive” class) vs. all other classes (the “negative” class), the confusion matrix can be read as follows:

	predicted positive	predicted negative
given positive	TP (true positive)	FN (false negative)
given negative	FP (false positive)	TN (true negative)

This notation is also often used in two-class problems, where we have a particular interest to detect cases of the “positive” class, e.g., medical tests on specific diseases.

Quality Measures for Individual Classes

recall: proportion of test objects of some class c_i that have been predicted correctly



$$f_i = \{o \in TE | f(o) = c_i\} :$$

$$\text{recall}_{TE}(h, i) = \frac{|\{o \in f_i | h(o) = f(o)\}|}{|f_i|}$$

precision: proportion of test objects predicted as class c_i that actually belong to class c_i

$$h_i = \{o \in TE | h(o) = c_i\} :$$

$$\text{precision}_{TE}(h, i) = \frac{|\{o \in h_i | h(o) = f(o)\}|}{|h_i|}$$

SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

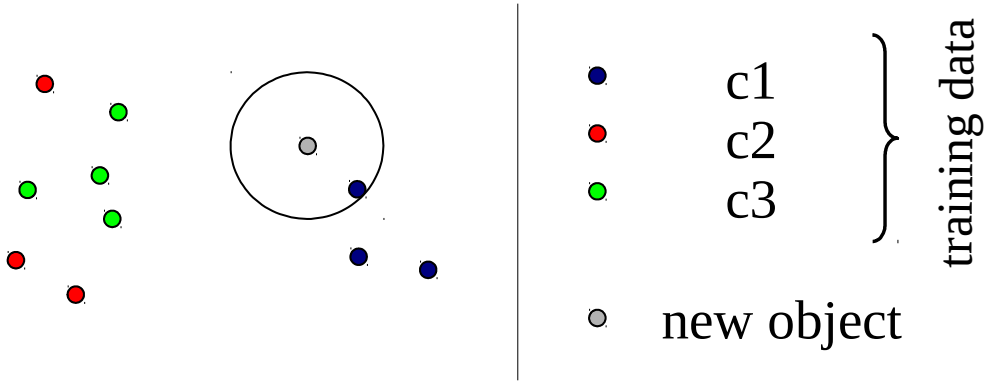
Recommended Reading

Recommended Reading:

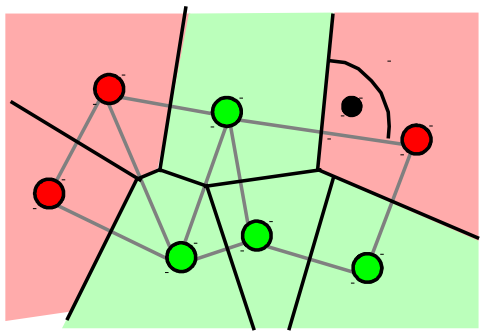
- *Tan et al. [2006], Chapter 5.2*
- *Tan et al. [2020], Chapter 6.3*
- *Witten et al. [2011], Chapter 4.7*
- *Hastie et al. [2001], Chapter 2.3*
- *Mitchell [1997], Chapter 8*

Instance-based Learning

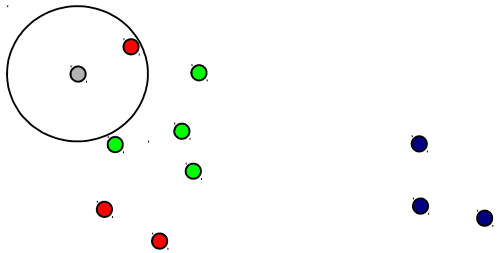
- a simple classifier: assign to a new object the class of the nearest neighbor in the training data



- we can visualize class regions by Voronoi cells



k-Nearest Neighbor Classification



- potential problem: nearest neighbor might be an outlier, somehow unusual, misleading
- take more than one neighbor into consideration: k nearest neighbor classifier

decision set: the set of (k) nearest neighbors considered for the classification decision

decision rule: how to decide the class, given the potentially different classes of the k nearest neighbors

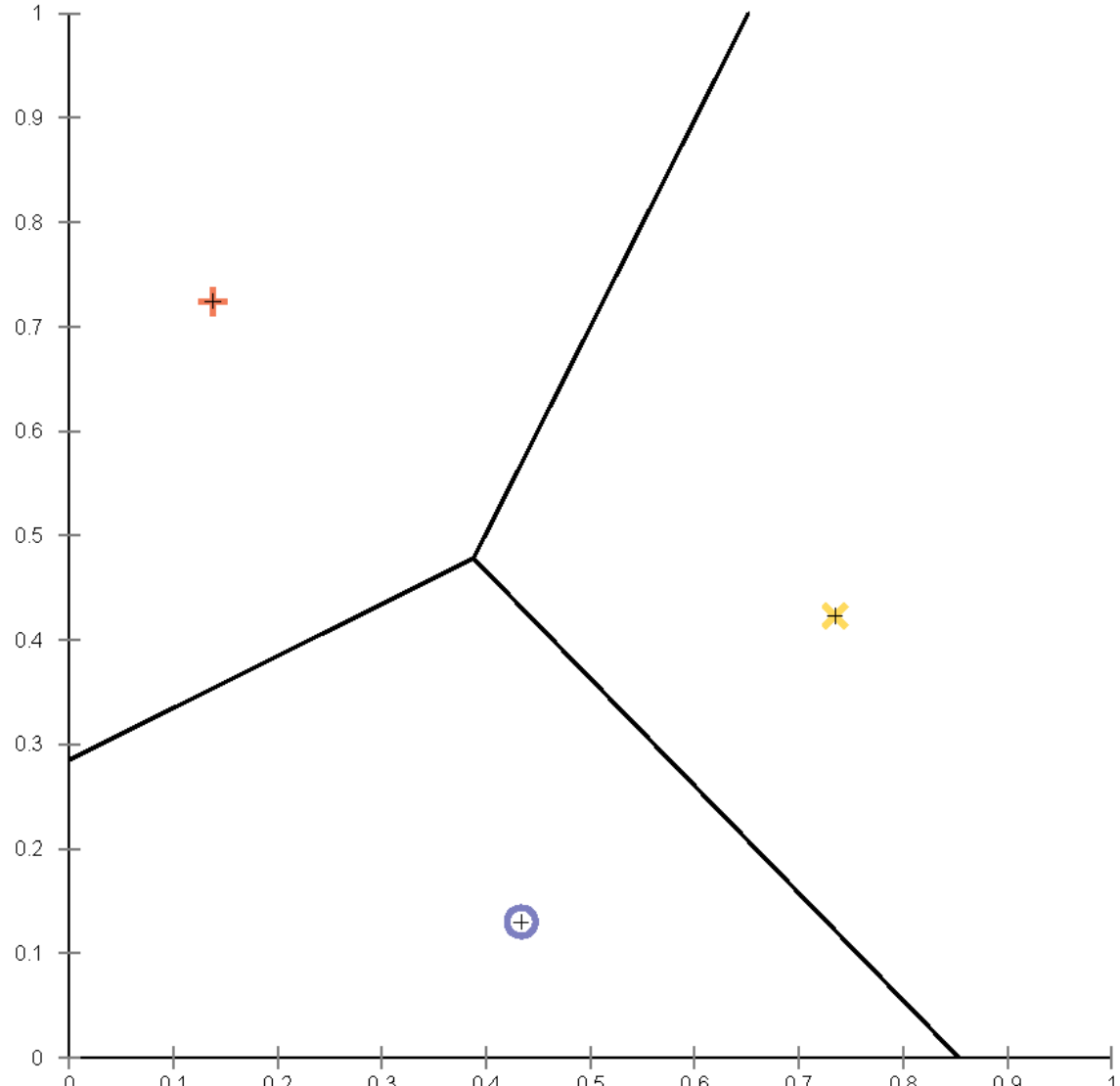
- take the majority vote
- potentially weighted votes

Let $x_1, \ldots, x_k$ be the k nearest neighbors of instance x_q:

$$h(x_q) = \arg\max_{c \in C} \sum_{i=1}^{k} w_i \delta(c, f(x_i)) \text{ where } \delta(a,b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ otherwise} \end{cases}$$

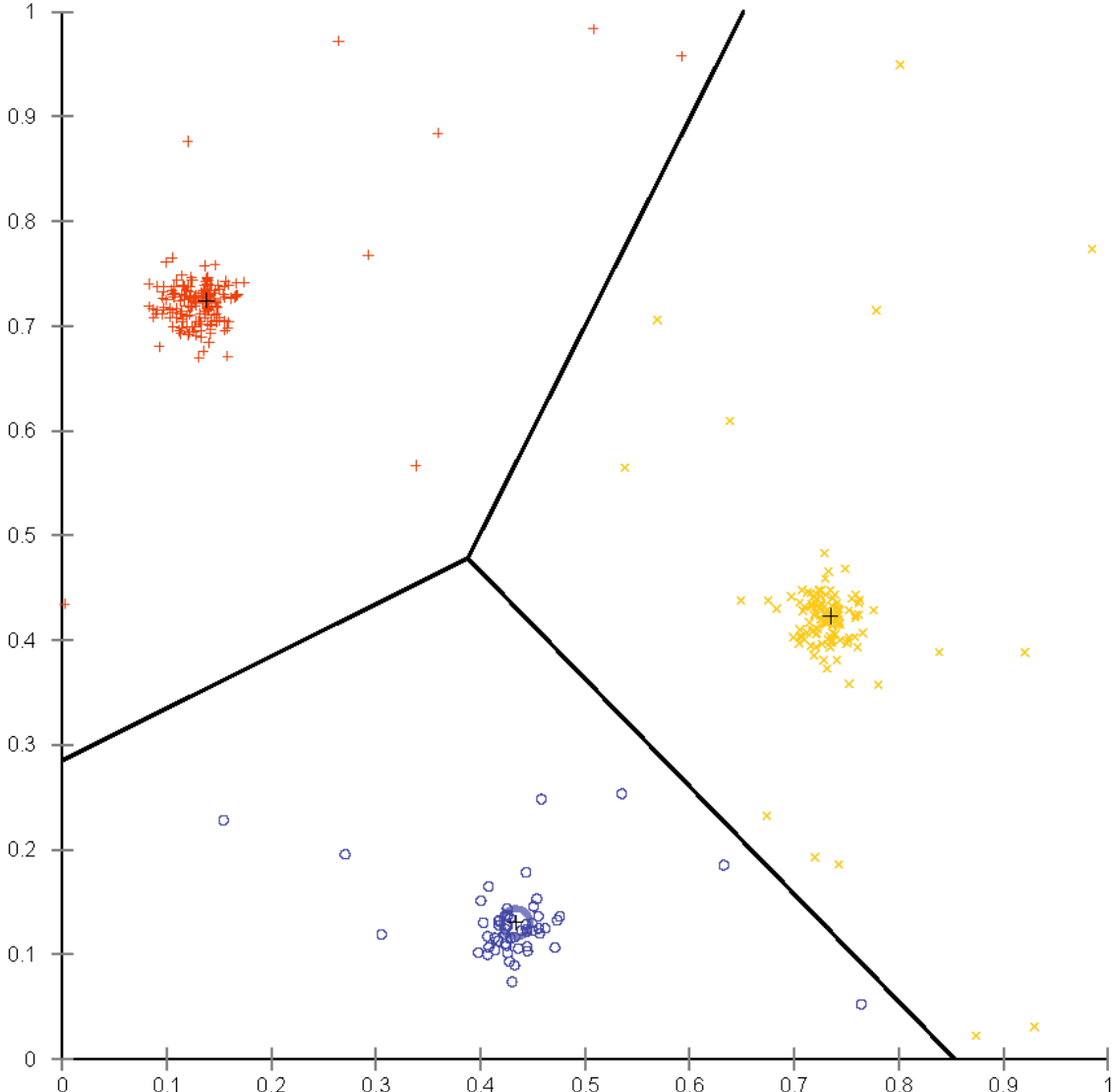
"Training" and Prediction for kNN Classifier

small training set: simple decision boundaries



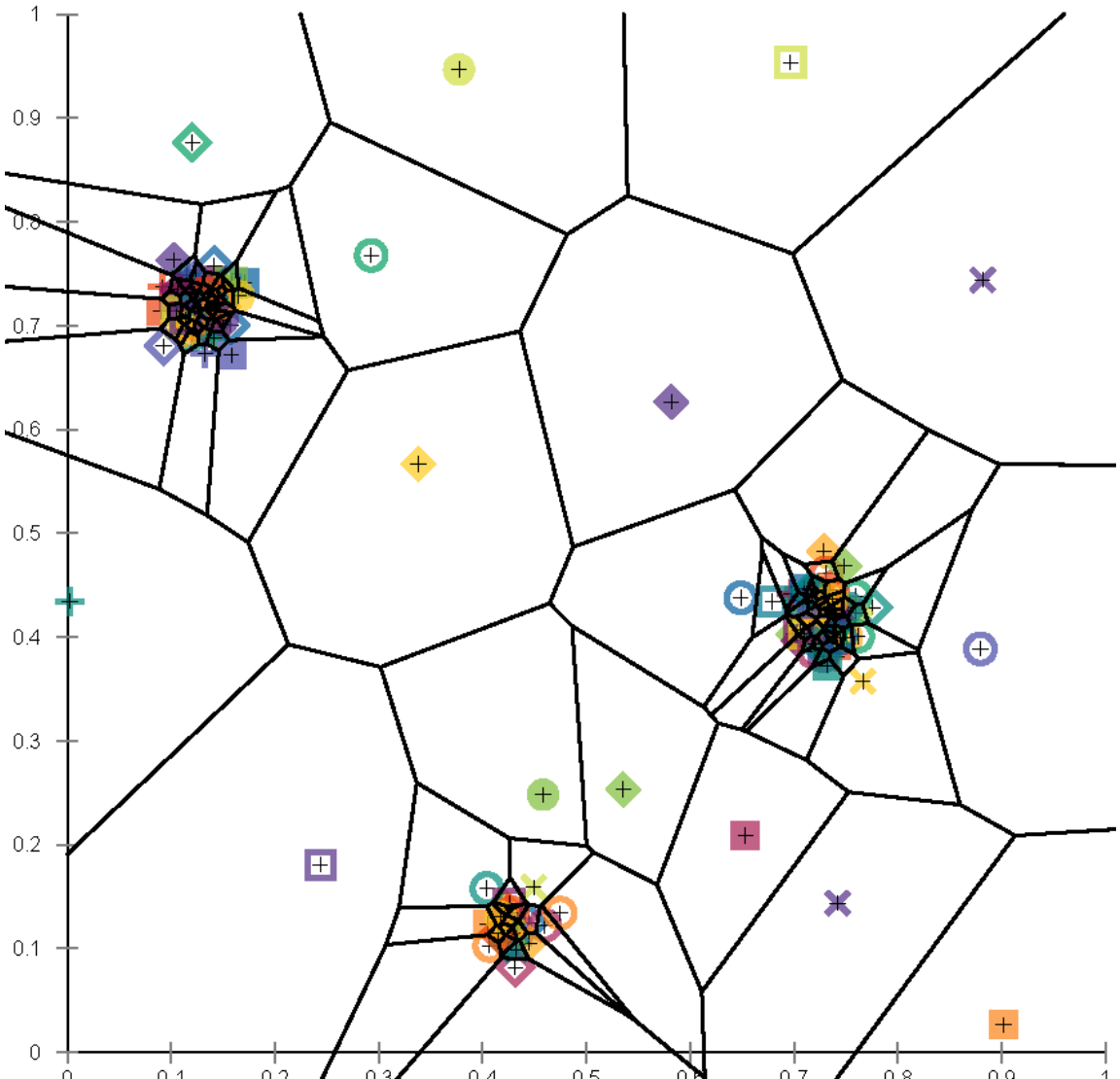
"Training" and Prediction for kNN Classifier

small training set: simple decision boundaries



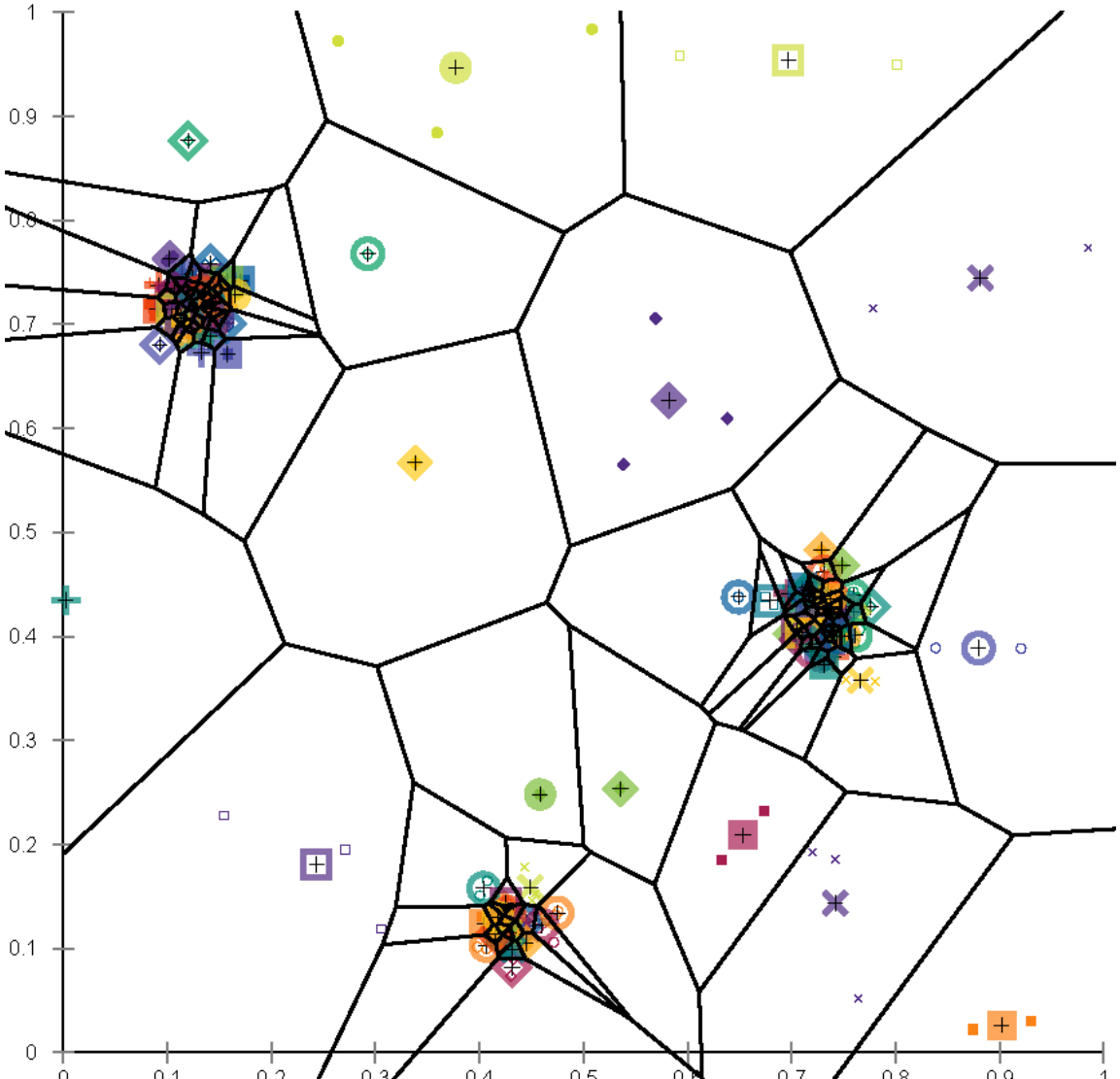
"Training" and Prediction for kNN Classifier

large training set: potentially complex decision boundaries



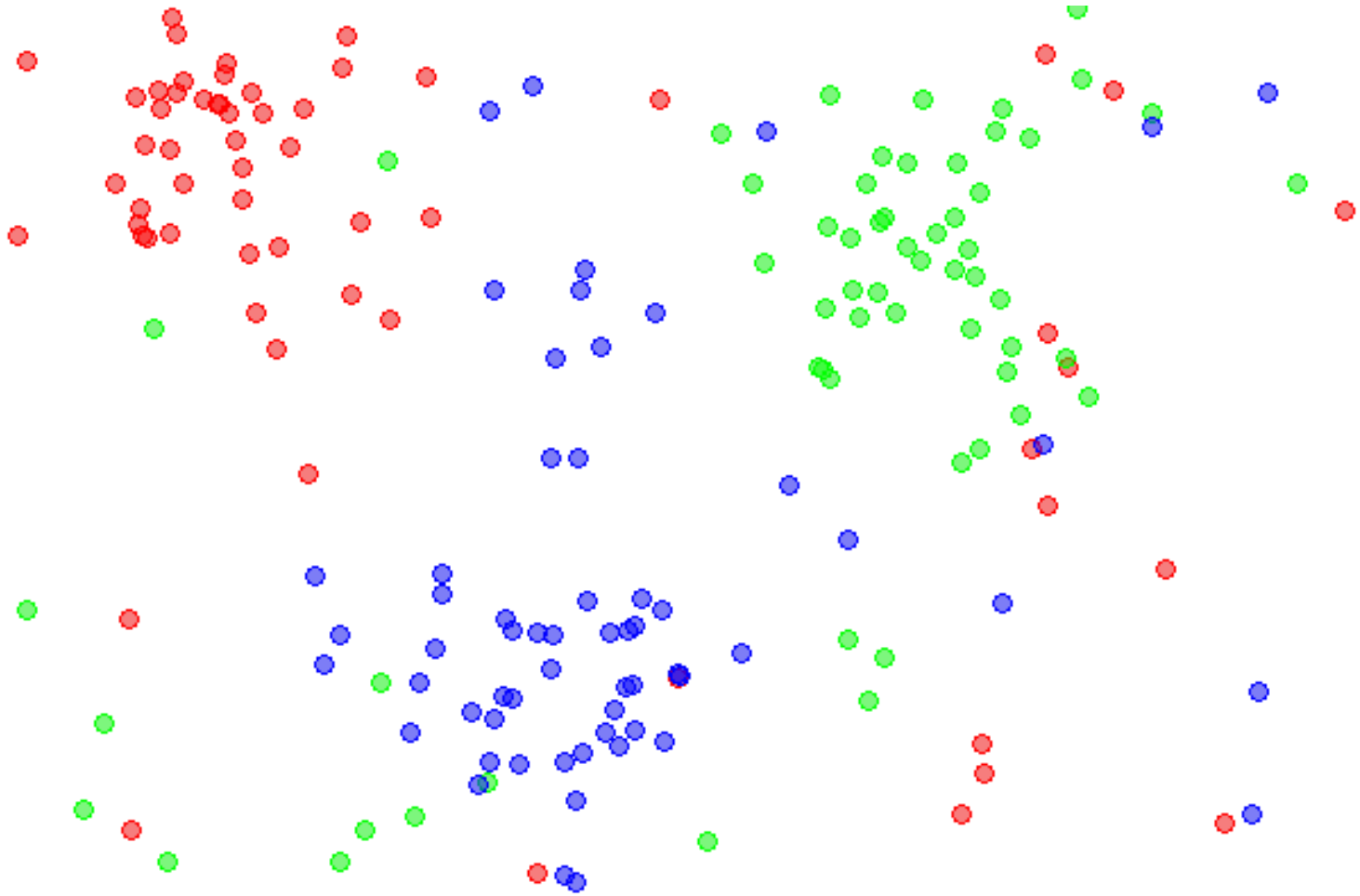
"Training" and Prediction for kNN Classifier

large training set: potentially complex decision boundaries



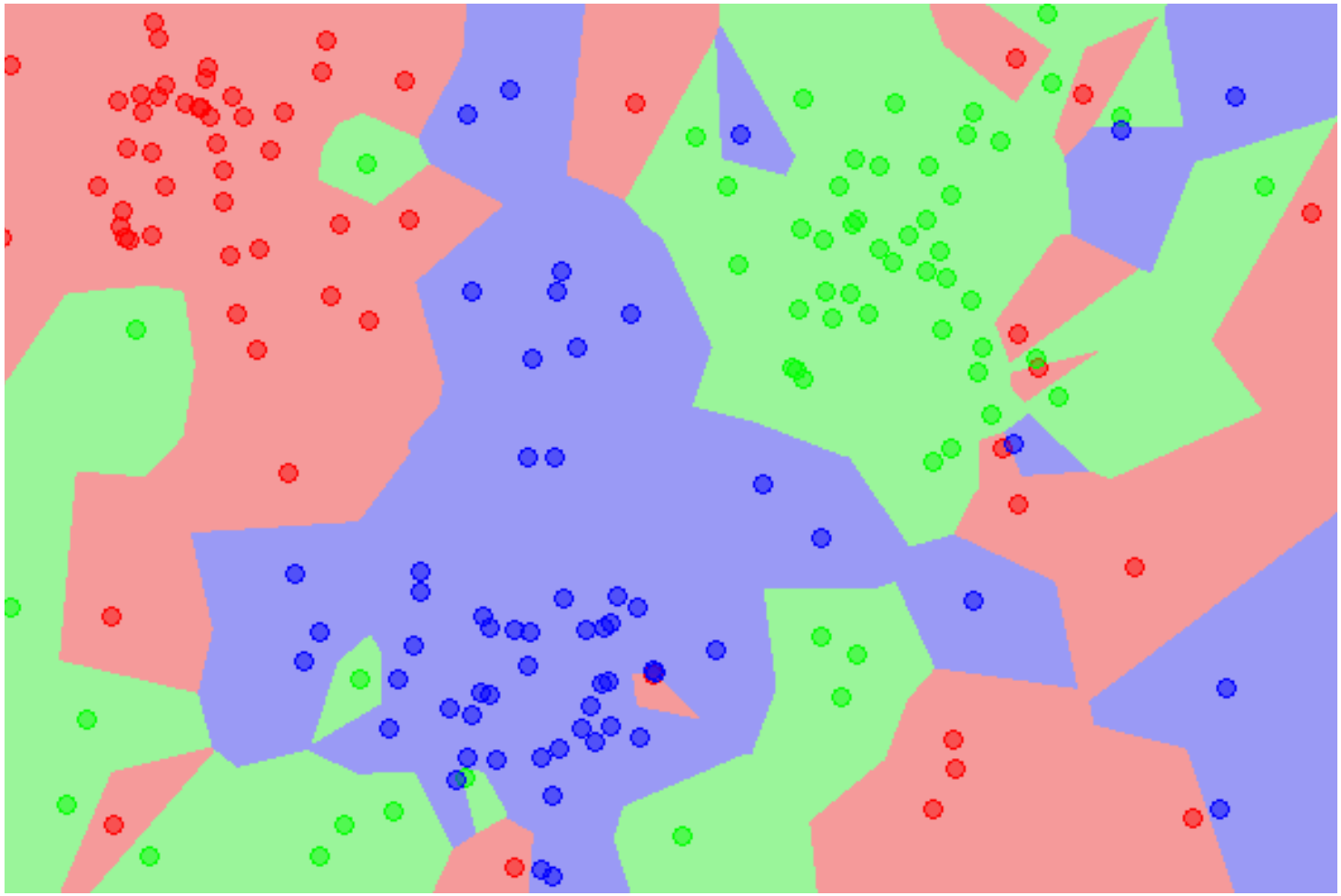
Complex Class Boundaries with Larger k

dataset



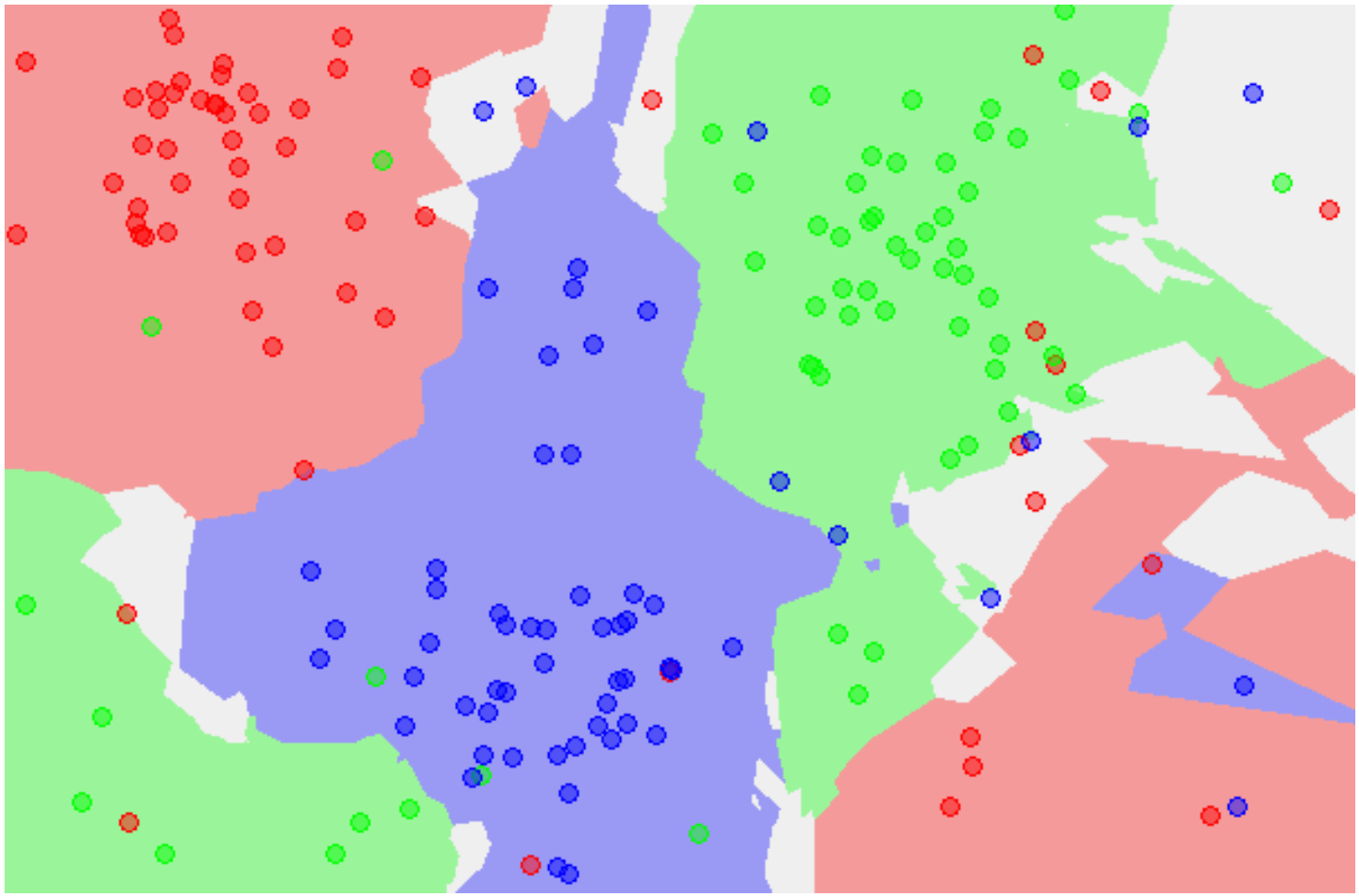
Complex Class Boundaries with Larger k

$k = 1$

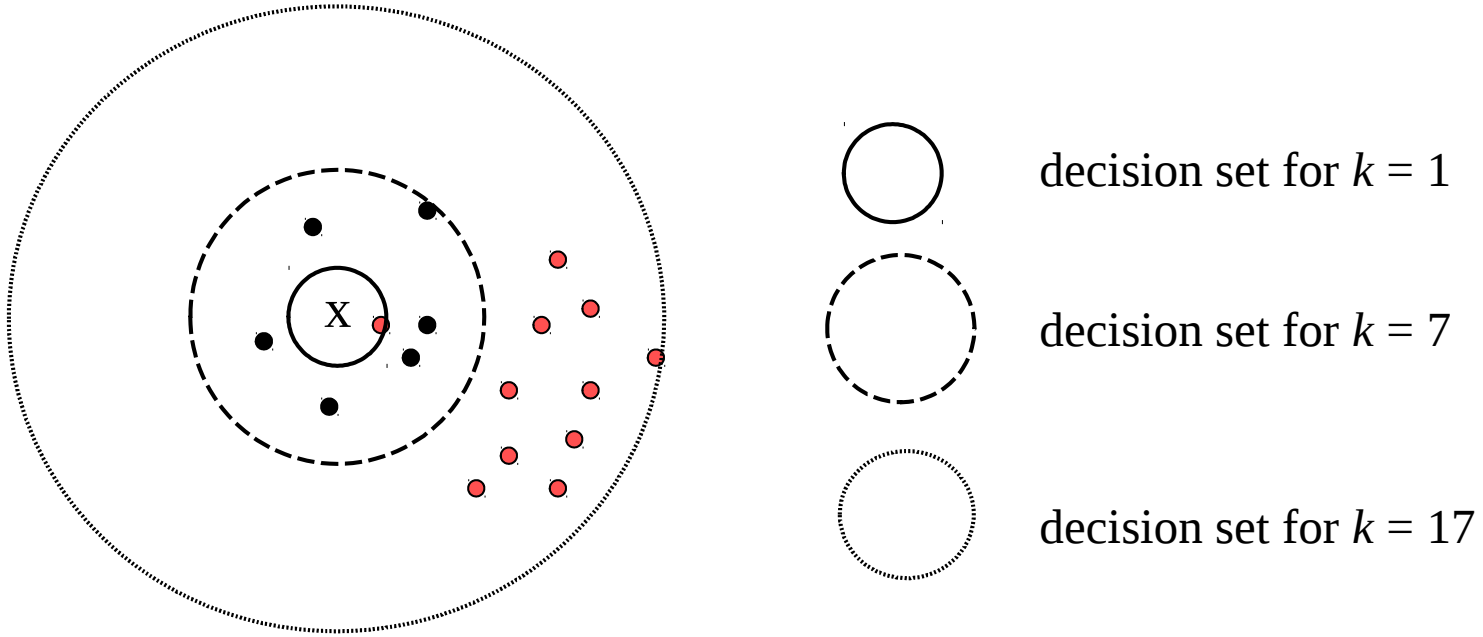


Complex Class Boundaries with Larger k

$k = 5$



Decision Set



- k too small: classifier is sensitive to outliers
- k too large: potentially takes objects belonging to other classes into the decision set
- medium k: best quality
- rule of thumb: $1 \ll k \leq 10$, but consider, e.g., size of classes in training set

Decision Rule

- standard: choose majority class in decision set
- advanced: put weights on the class votes
  - by distance, typically squared inverted distance:

$$\text{weight}(\text{dist}) = \frac{1}{\text{dist}^2}$$

  - by class proportions:
    If a class is small, it has a smaller chance of being the majority in some decision set.

Example: 2 classes, 95% A, 5% B, the labels of some decision set (e.g., labels of the 7 nearest neighbors of x) are $\{A, A, A, A, B, B, B\}$

- standard decision rule: $h(x) = A$
- votes weighted based on class size: $h(x) = B$

Instance-based Learning

- Instance-based learning does not provide an explicit description of the target function.
- Training examples are simply stored.
- Generalization beyond the training examples is postponed until a new instance must be classified (“lazy learner”).
- High flexibility (low bias) because the target function is actually estimated locally and differently for each new instance instead of once for the entire instance space.

Discussion

pros

- easy to apply: requires only training data and distance function
- often good classification accuracy
- incremental: easy adaptation to new training data
- no training required (“lazy learner”)

cons

- inefficient prediction: each decision requires k nearest neighbor query
- does not deliver explicit knowledge about classes
- difficult to choose k, esp. if classes are of very different size

SDU
SYDDANSK UNIVERSITET

Outline

Classification – Basics and a Basic Classifier

Summary

You learned in this section:

- *What is Classification?*
- *hypothesis-space and bias*
- *Why is a bias unavoidable for learning and generalization?*
- *evaluation procedures (cross-validation, bootstrap, leave-one-out)*
- *quality measures for classifiers:*
  - *confusion matrix*
  - *accuracy & error (apparent vs. true)*
  - *precision & recall*
- *a simple classifier: k-nearest neighbors*
  - *properties, challenges, variants*
  - *lazy learning*

References I

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, 2001.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2nd edition, 2020.

I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 3rd edition, 2011.

M. J. Zaki and W. Meira Jr. *Data Mining and Analysis. Fundamental Concepts and Algorithms*. Cambridge University Press, 2nd edition, 2020.