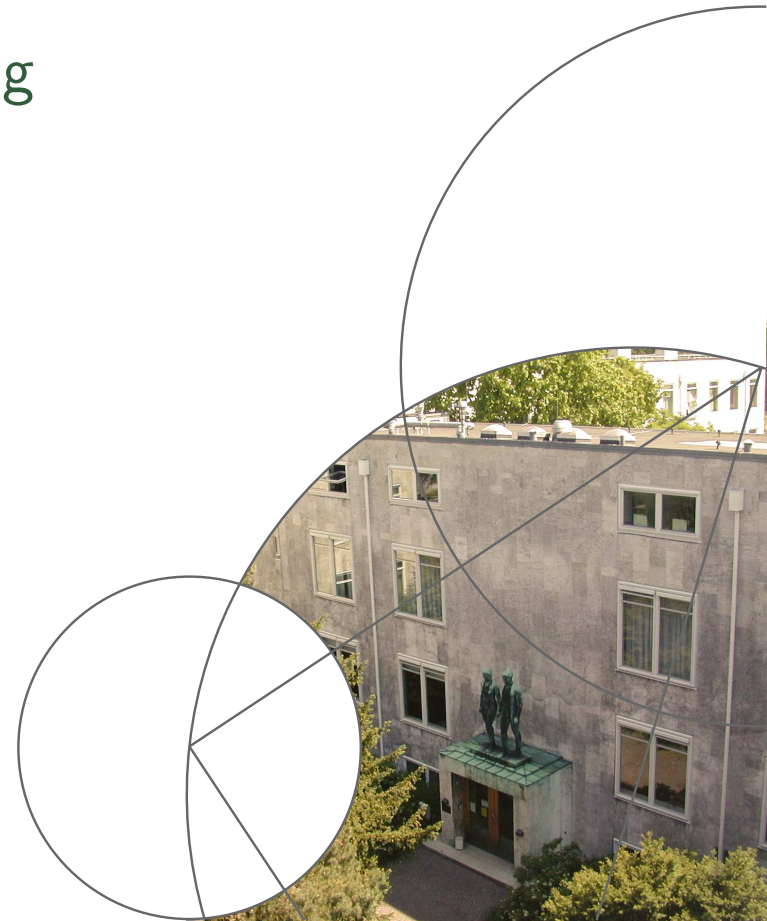
The Learning Problem

Statistical Methods for Machine Learning

Christian Igel
Department of Computer Science



Machine learning

Machine learning is a branch of computer science and applied statistics covering software that improves its performance at a given task based on sample data or experience.

Why machine learning?

- Computer systems are required for tasks for which solutions cannot be specified in the traditional way, e.g., because
  - the designer's knowledge is limited, and/or
  - the sheer complexity and variability precludes an accurate description.
- However, large amounts of data describing the task are often available or can be automatically obtained.
- To take proper advantage of this information, we need systems that self-adapt and automatically improve based on sample data – systems that learn.
- Machine learning (ML) is a vital ingredient of data mining, data analytics, knowledge discovery, etc.

Machine learning turns data into knowledge

Outline

Outline

Inductive inference

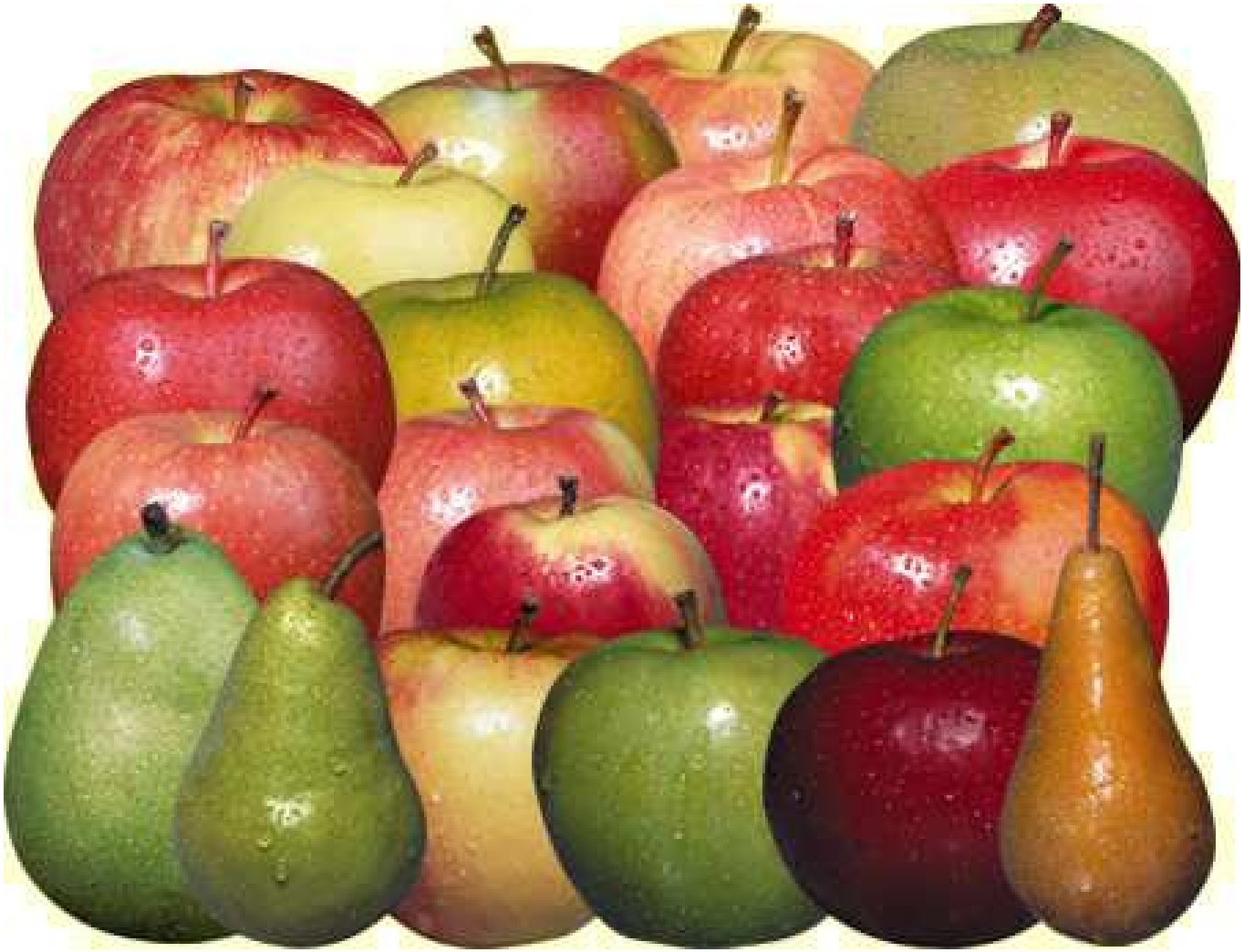
Learning is goal-directed changing of behaviour based on experience.

Supervised machine learning formalizes and automates the process of *inductive inference*:

1. Observe a phenomenon
2. Construct a model of that phenomenon
3. Make predictions using this model

Binary classification

two classes: $\{\text{apples}, \text{pears}\}$, $\{-1, 1\}$

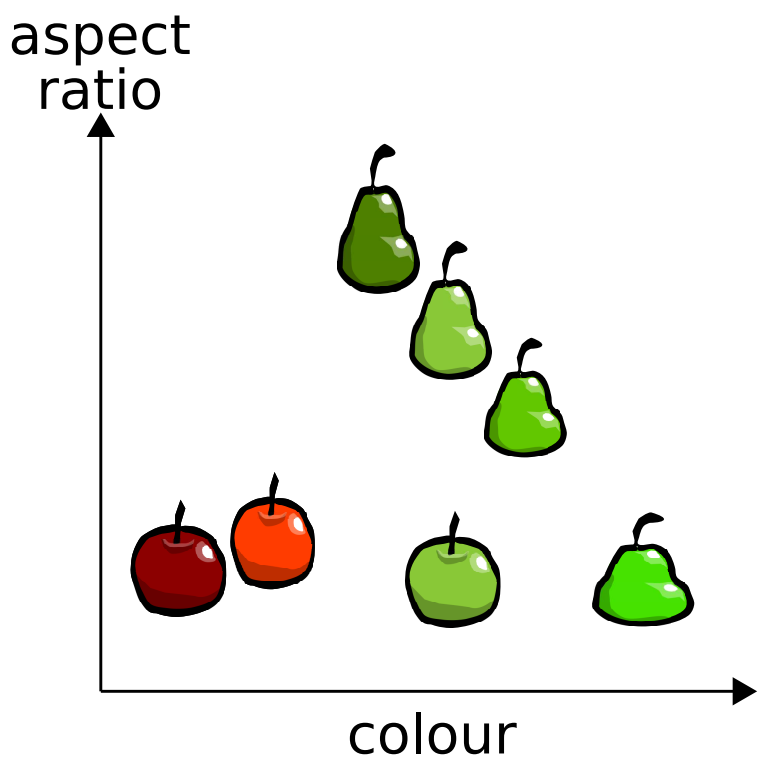


Supervised learning

The goal of supervised learning is to uncover an unknown relation between

- input space $\mathcal{X}$,
  e.g., real-valued vector describing color and shape of fruit, and
- output space $\mathcal{Y}$,
  e.g., {apples, pears},

based on sample input-output data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$.



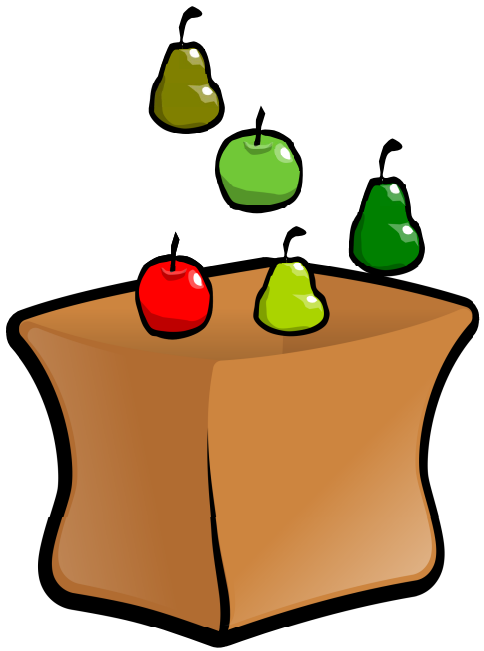
I.i.d. assumption

- *I.i.d.* (identically, independently distributed) assumption: Each pattern is drawn independently from an (unknown) stable distribution p over $(\mathcal{X}, \mathcal{Y})$:

$$p((x, y)) = p(x) \cdot p(y \mid x)$$

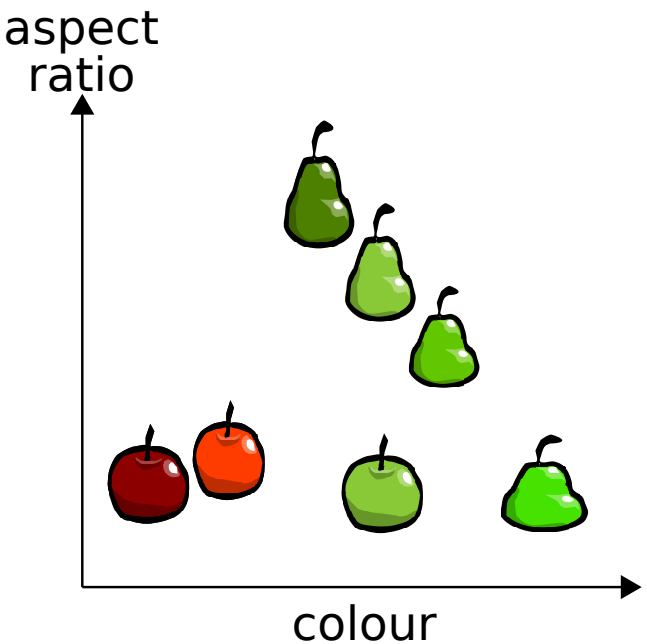
- Example: Probability of $(\text{red}, 8\,\text{cm}, \text{round})$ is always the same for each training or test pattern

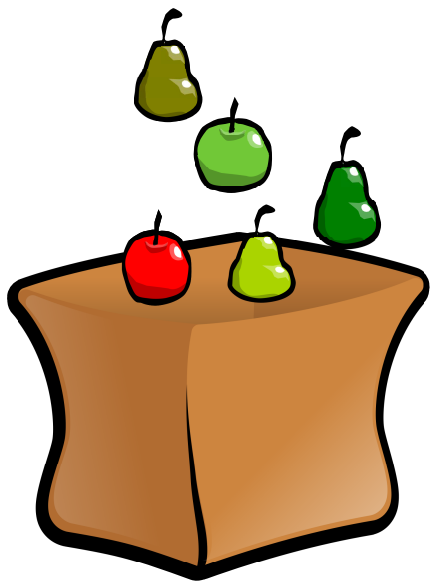
  Probability of being an apple is the same for each fruit that is $(\text{red}, 8\,\text{cm}, \text{round})$



The goal of learning

Given sample data, we want to find a hypothesis predicting output from input showing good average performance on patterns $(x_i, y_i) \sim p$.





Broad division of supervised learning

1. Classification: Assigning an input to one class of a finite set of classes $\mathcal{Y} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$, $2 \leq m < \infty$.

2. Regression: Given an input, predict corresponding output from $\mathcal{Y} \subseteq \mathbb{R}^m$, $1 \leq m < \infty$.

3. Density estimation: Given an input x, predict the probability distribution $p(y|x)$ on $\mathcal{Y}$.

We focus on classification and regression in the following.

Learning machines

- A *hypothesis* $h : \mathcal{X} \to \mathcal{Y}$ maps inputs to outputs, a hypothesis class $\mathcal{H}$ is a set of such functions

  E.g., a hypothesis makes a prediction whether a fruit is an apple or a pear given shape, color, and size

- Applying a learning algorithm means coming up with a hypothesis given sample data (formally, it maps from $\{((x_1, y_1), \ldots, (x_\ell, y_\ell)) \,|\, 1 \leq i \leq \ell < \infty, x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ to $\mathcal{H}$).

Loss/risk functions

The quality of the prediction of a hypothesis is quantified by a *task-dependent* loss function.

A function $L : \mathcal{Y} \times \mathcal{Y} \to [0, \infty[$ with the property $L(y, y) = 0$ for $y \in \mathcal{Y}$ is called a *loss function*.

Typical loss for classification is the 0-1 loss:

$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$

Goal of learning

Goal: Given sample data, we want to find a hypothesis h showing good average performance on patterns $(x, y) \sim p$.

That is, we want to find h minimizing the *risk*:

$$\mathcal{R}_p(h) = \mathbb{E}_{(x,y)\sim p}\{L(y, h(x))\}$$

The *empirical risk* of h on sample data S ($|S| = \ell$) is defined as:

$$\mathcal{R}_S(h) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i, h(x_i))$$

Typical loss functions

- Classification: Under the 0-1 loss $L(y, h(x)) = \mathbb{I}\{h(x) \neq y\}$ the risk of hypothesis h is the probability of error

$$\mathcal{R}_p(h) = \Pr(h(X) \neq Y) = \mathbb{E}_p\{\mathbb{I}\{h(X) \neq Y\}\}$$

and the empirical risk on sample data S ($|S| = \ell$) counts errors:

$$\mathcal{R}_S(h) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}\{h(x_i) \neq y_i\}$$

(indicator function $\mathbb{I}\{\cdot\}$ is 1 if its argument is true and 0 otherwise)

- Regression: Under the squared error

$$L(y, \hat{y}) = (y - \hat{y})^2$$

the empirical risk corresponds to the sum-of-squares error

Bayes risk and consistency

- Minimum risk over *all possible measurable functions* h is the Bayes risk

$$\mathcal{R}_p^{\text{Bayes}} = \mathcal{R}_p(h^{\text{Bayes}}) = \inf_h \mathcal{R}_p(h)$$

usually differs from $\mathcal{R}_p^* = \mathcal{R}_p(h^*) = \inf_{h \in \mathcal{H}} \mathcal{R}_p(h)$

for $\mathcal{Y} = \{-1, 1\}$ we have

$$h^{\text{Bayes}}(x) = \operatorname{sgn}\left[\mathbb{E}_p(Y \mid X = x)\right]$$

(here $\operatorname{sgn}(x) := 2\mathbb{I}\{x > 0\} - 1$)

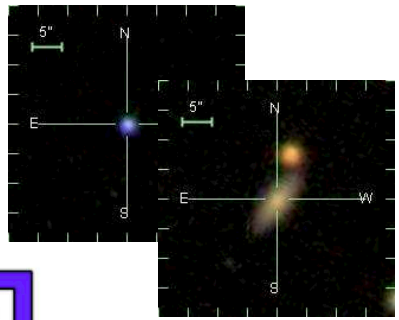
- Algorithm a is *consistent* if $\lim_{\ell \to \infty} \mathcal{R}_S(a(S)) = \mathcal{R}_p^{\text{Bayes}}$ almost surely

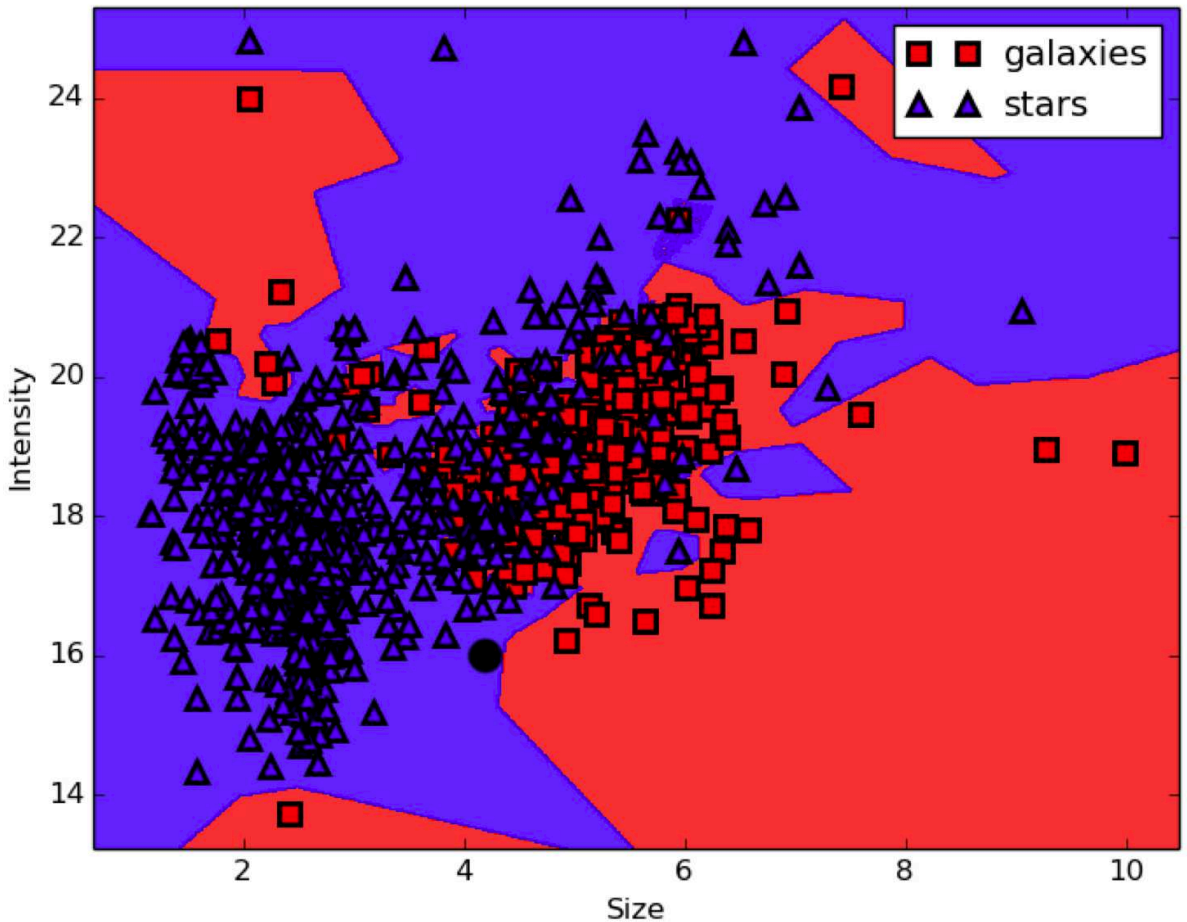
Outline

Distinguishing stars and galaxies

A typical classification task

L. K. Polsterer, P.-C. Zinn, F. Gieseke. Finding New High-Redshift Quasars by Asking the Neighbours. *Monthly Notices of the Royal Astronomical Society* (MNRAS) 428(1):226–235, 2013





Nearest neighbor classification

- Idea:
  - Given training data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and new data point x
  - Look in S for x_i most similar to x
  - Assign new record to class of x_i
- This requires a notion of similarity

Recall: Metric

Definition: A *metric* or *distance function* on $\mathcal{X}$ is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with:

1. $d(x, y) \geq 0$ with $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$ (symmetry),
3. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

for all $x, y \in \mathcal{X}$.

Example: Euclidean metric

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

for $\boldsymbol{p} = (p_1, \ldots, p^n)^\mathsf{T}, \boldsymbol{q} = (q_1, \ldots, q^n)^\mathsf{T} \in \mathbb{R}^n$

1-Nearest neighbor (1-NN) classification

Algorithm 1: 1-nearest neighbor

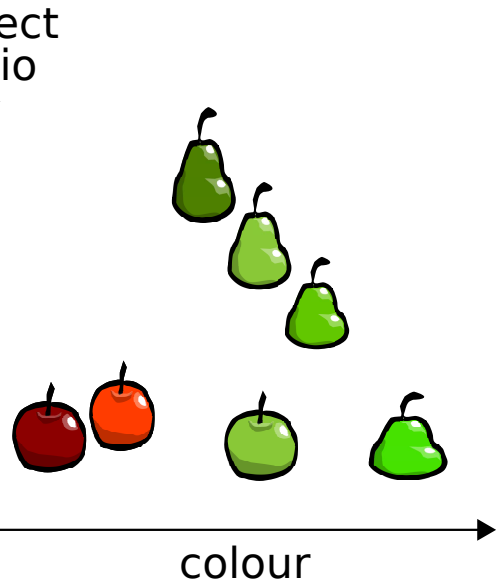
Input: metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, training data $S = \{(x_1, y_1), \dots\} \in (\mathcal{X} \times \{-1, 1\})^\ell$, new input x to be classified

Output: predicted label y of x

`/* ties are broken at random */`

1 $(x_{\text{min}}, y_{\text{min}}) = \operatorname{argmin}_{(x_i,y_i)\in S} d(x_i, x)$

Result: $y = y_{\text{min}}$



K-Nearest neighbor (K-NN) classification

Algorithm 2: K-nearest neighbor

Input: metric d, training data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$, input x to be classified, number of neighbors K

Output: predicted label y of x

1 **while** $|S^\star| < K$ **do**

    `/* ties broken at random */`

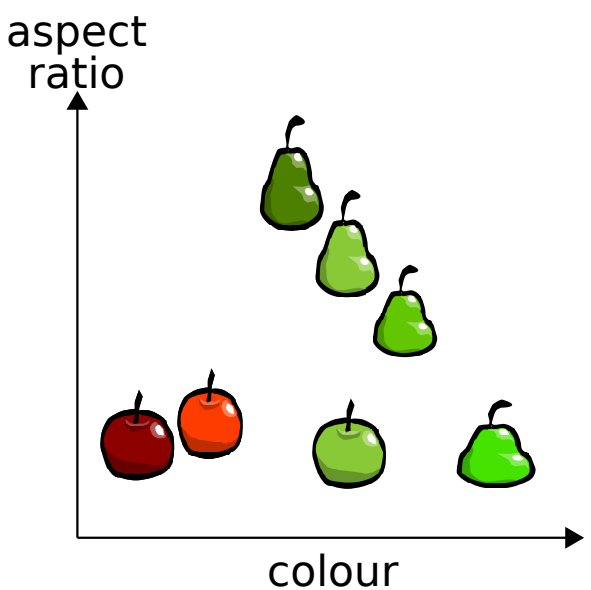
2 $(x_{\text{min}}, y_{\text{min}}) = \operatorname{argmin}_{(x_i,y_i)\in S} d(x_i, x)$

3 $S \leftarrow S \setminus \{(x_{\text{min}}, y_{\text{min}})\}$

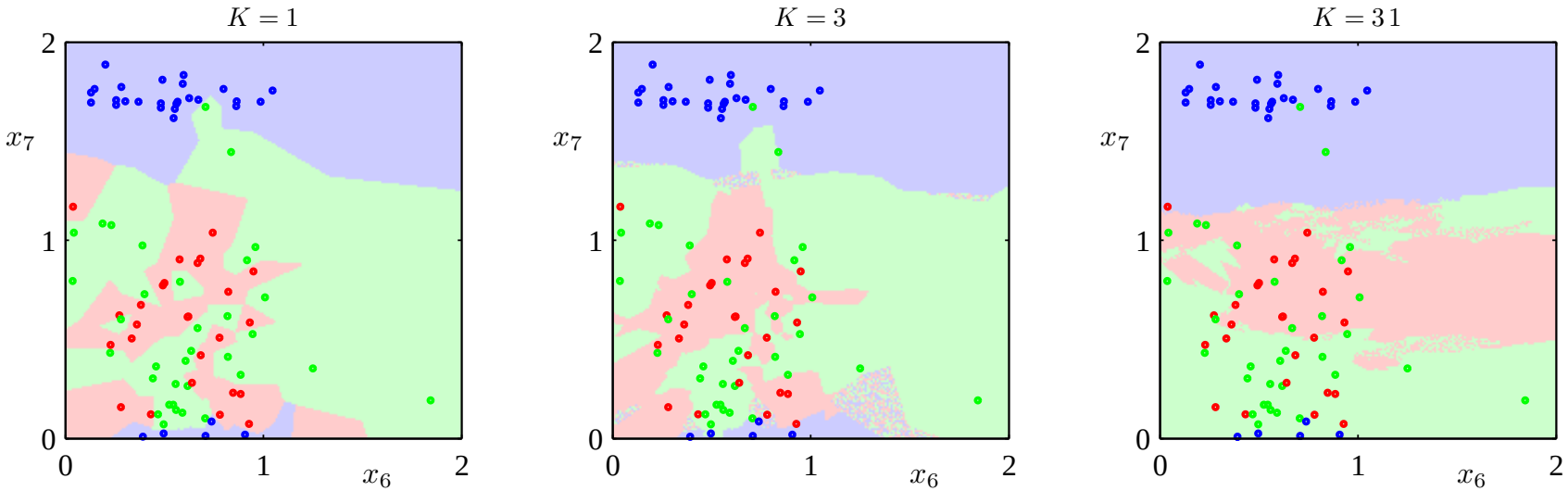
4 $S^\star \leftarrow S^\star \cup \{(x_{\text{min}}, y_{\text{min}})\}$

`/* ties broken at random */`

Result: $y = \operatorname{argmax}_c |\{(x, y) \,|\, (x, y) \in S^\star \wedge y = c\}|$



Influence of K



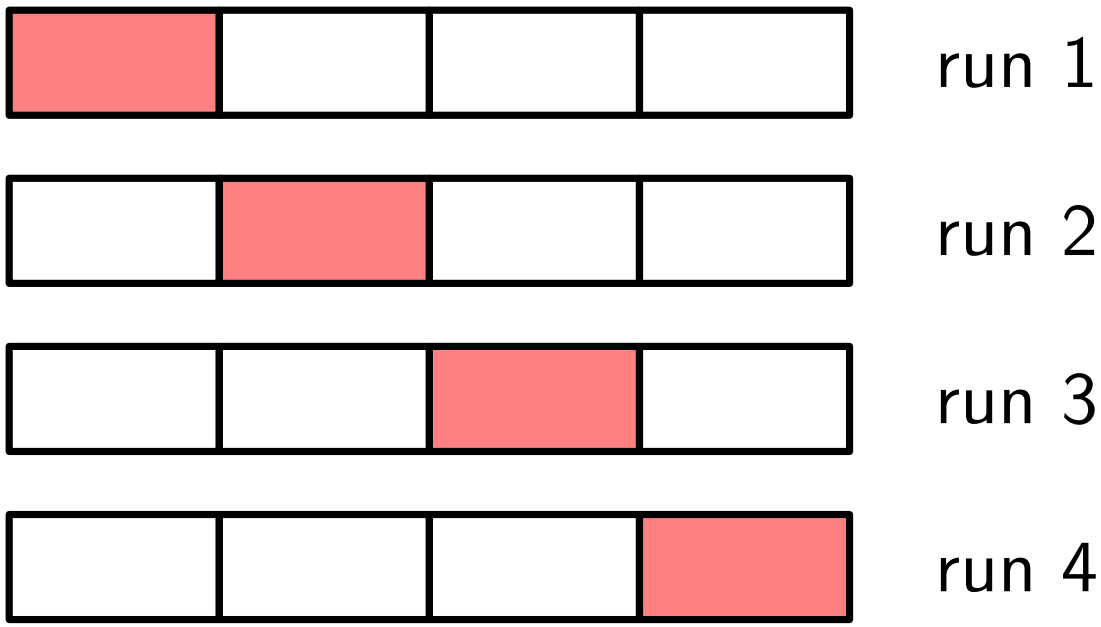
C. M. Bishop. *Pattern Recognition and Machine Learning*, Springer, 2006

Outline

Hyperparameter selection

- The results of K-NN depends on the number of neighbors – now, how choose K?
- Algorithm parameters such as K are called *hyperparameters*.
- Finding hyperparameters is often called *model selection*.
- A standard method for selection of a few hyperparameters is *κ-fold cross-validation*.

Cross-validation



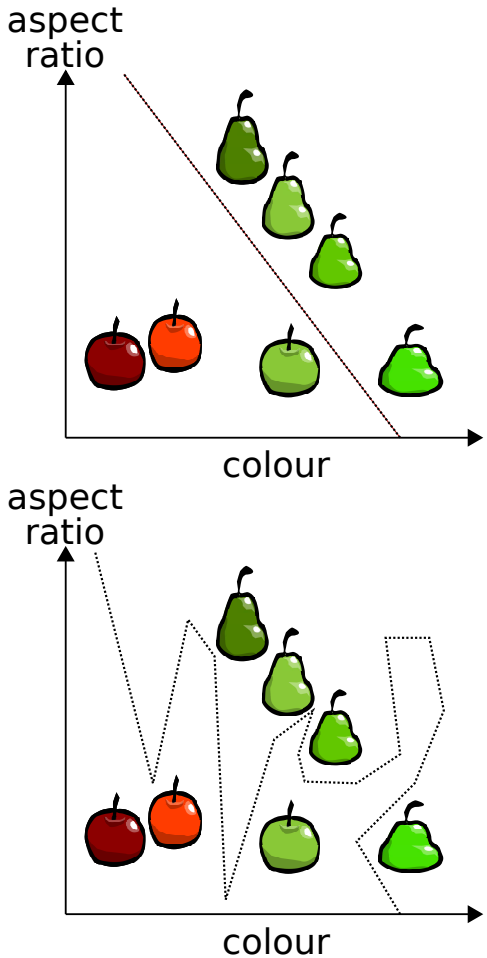
C. M. Bishop. *Pattern Recognition and Machine Learning*, Springer, 2006

- Split S into κ (almost) even parts
- Train κ times on $\kappa - 1$ parts and test on the held-out part
- Take hyperparameters maximizing mean test performance for building the model using all data

Outline

Learning by heart is not enough

- 1-NN has always zero error on the training set. And on a test set?
- Minimizing empirical risk does not imply good *generalization* (i.e., good performance on patterns not seen during training).
- *Overfitting:* The hypothesis faithfully reflects idiosyncrasies of the training data rather than the underlying distribution.



Learning strategies

- Empirical risk minimization given $\mathcal{H}$
  ("learning by heart restricted to $\mathcal{H}$")

$$h_S = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{R}_S(h)$$

- Regularized risk minimization

$$h_S = \operatorname*{argmin}_{h \in \mathcal{H}} \left[\mathcal{R}_S(h) + \text{penalty}(h)\right]$$

Examples of regularization

- Using the norm for regularization

$$h_S = \operatorname*{argmin}_{h \in \mathcal{H}} \left[\mathcal{R}_S(h) + \gamma \|h\| \right] \qquad (\gamma > 0)$$

- Bayesian normalization

$$p(h \mid S) = \frac{p(S \mid h) p(h)}{p(S)} \propto p(S \mid h) p(h)$$

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

instead of maximizing the a posteriori probability (MAP) it is convenient to minimize its negative logarithm

likelihood $\propto \exp(-\gamma \mathcal{R}_S(h))$ yields for MAP

$$h_S = \operatorname*{argmin}_{h \in \mathcal{H}} \left[\gamma \mathcal{R}_S(h) - \log \text{prior}(h) \right]$$

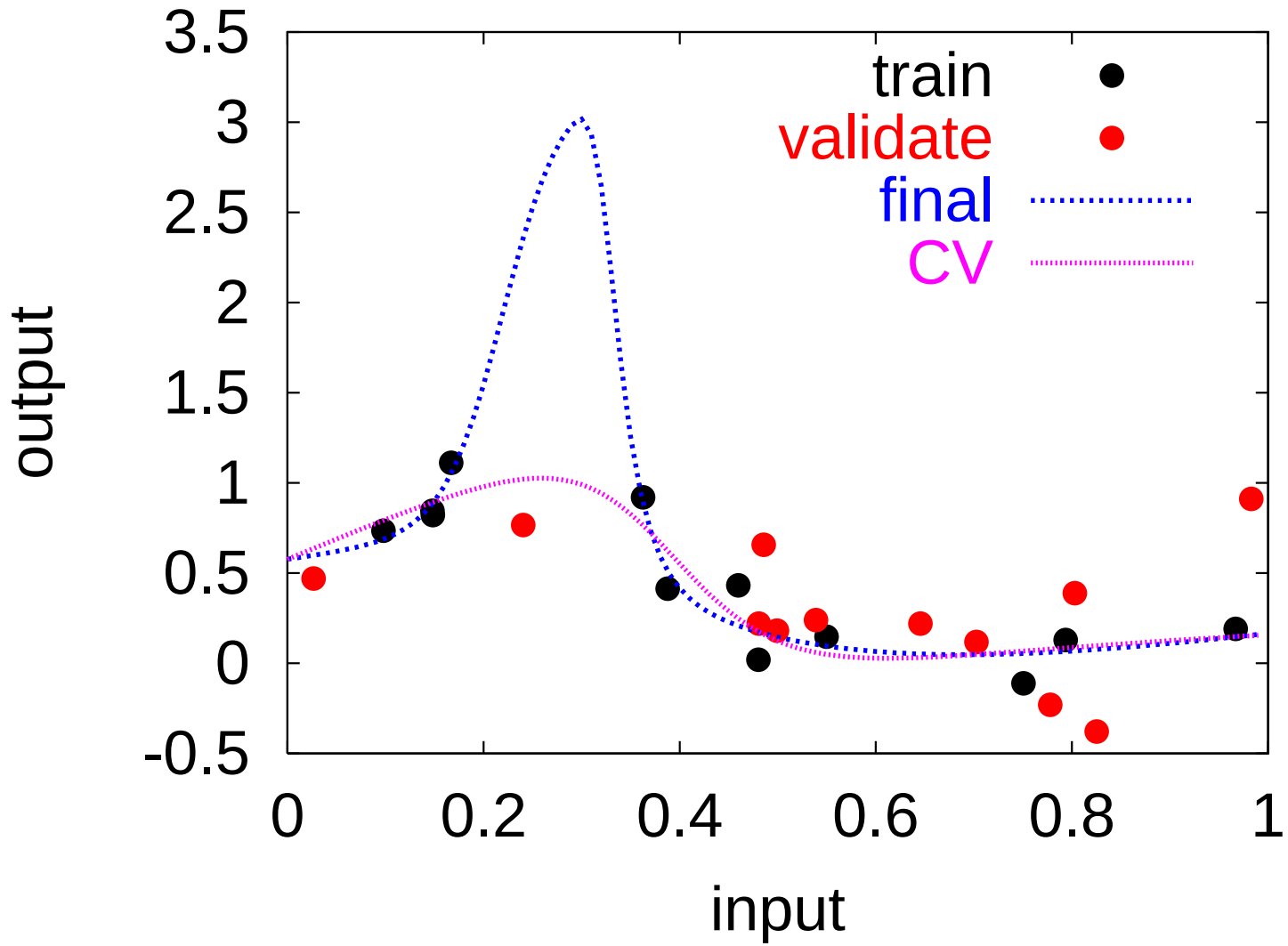
Early stopping

- There are ways of avoiding overfitting not falling in the general regularization framework outlined before.
- *Early-stopping*: The learning algorithm
  - partitions sample S into training S_{train} and validation S_{val} data,
  - produces iteratively a sequence of hypotheses

    $$h_1, h_2, h_3, \ldots$$

    based on S_{train}, ideally corresponding to a nested sequence of hypothesis spaces $\mathcal{H}_1 \subseteq \mathcal{H}_2 \ldots$ with $h_i \in \mathcal{H}_i$ and
    - non-decreasing complexity and
    - decreasing empirical risk $\mathcal{R}_{S_{\text{train}}}(h_i) > \mathcal{R}_{S_{\text{train}}}(h_{i+1})$ on S_{train},
  - monitors empirical risk $\mathcal{R}_{S_{\text{val}}}(h_i)$ on the validation data, and
  - outputs the hypothesis h_i minimizing $\mathcal{R}_{S_{\text{val}}}(h_i)$.

Early stopping example



Errors & bounds

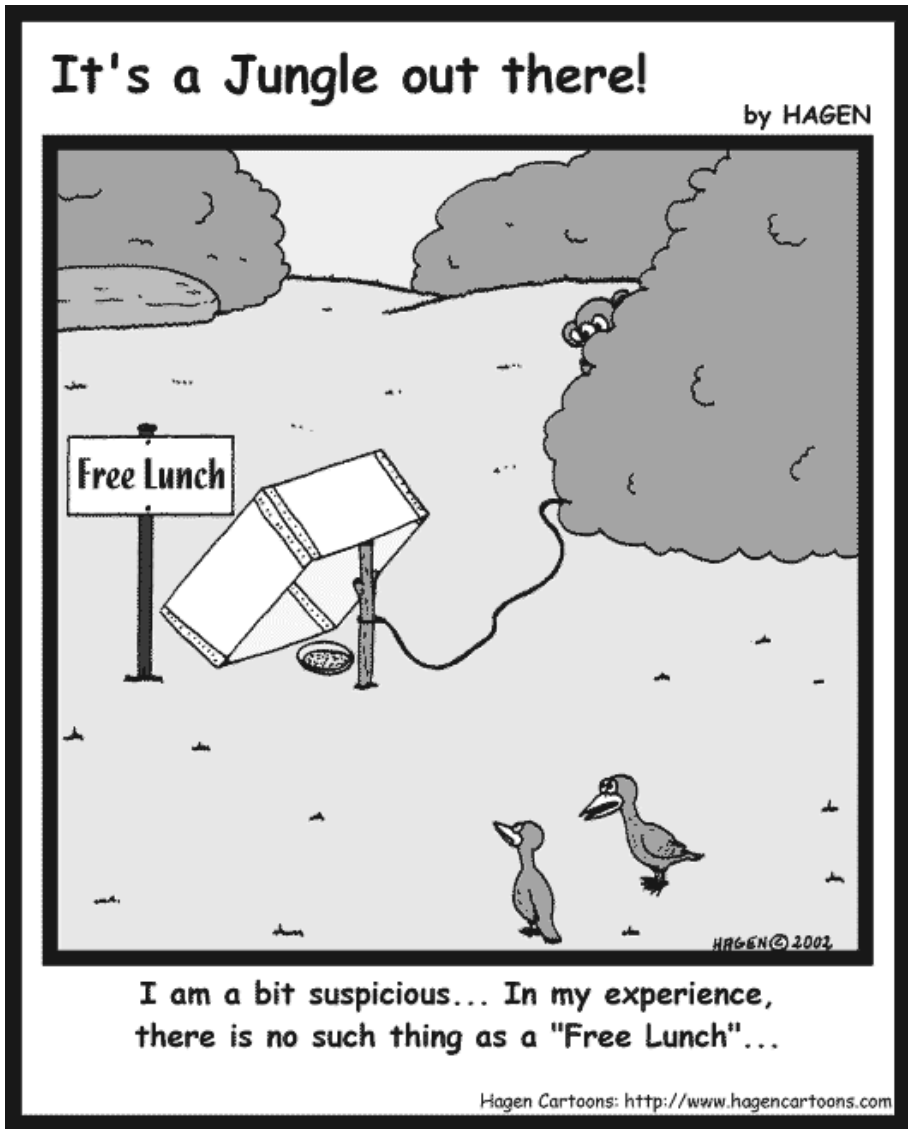
Typical bound:

With probability of at least $1 - \delta$

$$\mathcal{R}_p(h_S) \leq \mathcal{R}_S(h_S) + B(\ell, \delta, \mathcal{H})$$

Carefully check assumptions and to which entities expectations refer!

Free lunch...


It's a Jungle out there!
by HAGEN
Free Lunch
HAGEN © 2002
I am a bit suspicious... In my experience,
there is no such thing as a "Free Lunch"...
Hagen Cartoons: http://www.hagencartoons.com

No free lunch

1. If there is no assumption how the past (training data) is related to the future (test data), prediction is impossible.
2. If there is no a priori restriction on the possible phenomena that are expected, it is impossible to generalize and thus no algorithm is superior to another.
3. Any consistent algorithm can have arbitrarily bad behavior when given a finite, incomplete training set.

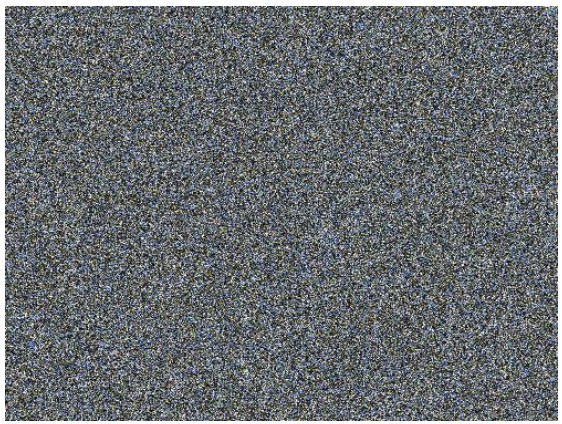
$$\text{Generalization} = \text{Data} + \text{Knowledge}$$

Olivier Bousquet

O. Bousquet, S. Boucheron, G. Lugosi. Introduction to Statistical Learning Theory. In *Advanced Lectures in Machine Learning*, LNAI 3176, 2004

Prior knowledge

But in real-world applications we can make assumptions about the expected phenomena!



Prior knowledge can be incorporated by

- a proper restriction of the hypothesis space,
- a proper definition of neighbourhood/similarity (e.g., by defining a proper scalar product inducing a metric), and/or
- penalising the "complexity" of classifiers (i.e., preferring "simple" solutions).

Outline

Summary

- The goal of supervised learning is to uncover an unknown relation between an input and an output space given examples.
- Memorizing training examples is not enough to achieve generalization (i.e., to classify correctly unseen examples).
- Incorporation of prior knowledge is required for generalization, for example by restricting the hypotheses space, by carefully designed distance measures, and/or by penalties on solution complexity.
- Nearest neighbor classification is a simple algorithm giving good (baseline) results in practice.
- Cross-validation can be used to find hyperparameters.