Data Mining and Machine Learning

Part 2: Supervised Learning

Melih Kandemir

University of Southern Denmark

Spring 2022

What is machine learning at all?

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured in P, improves with experience E”.

[Mitchell, 1997]

A closer look at supervised learning

We are also given the corresponding outputs for the samples:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Combined, $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ is called a *data set*.

The learning problem is called

- **classification** if $y_n \in \mathbb{Z}$ (set of integers, the magnitudes of which we do not care).
- **regression** if $y_n \in \mathbb{R}$ (set of real numbers).

Supervised learning example: Linear regression

We posit that the predictor function is defined as

$$\hat{y}_n = f(\mathbf{x}_n) = \mathbf{w}^T \mathbf{x}_n,$$

where $\mathbf{w}$ is the vector of model parameters.

For this model, learning means finding such a $\mathbf{w}$ that gives us accurate predictions for y.

Supervised learning example: Linear regression

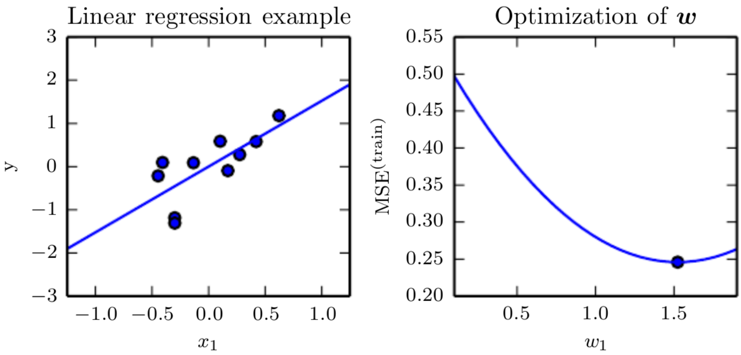


Figure. Goodfellow et al., Deep Learning, MIT Press, 2016

How to quantify the goodness of a predictor?

Mean Squared Error (MSE): Euclidean distance between the observed output y_n and the predicted output $\hat{y}_n$. Formally,

$$MSE = \frac{1}{N} \sum_{n=1}^{N} ||y_n - \hat{y}_n||_2^2.$$

How to quantify the goodness of a predictor?

Mean Squared Error (MSE): Euclidean distance between the observed output y_n and the predicted output $\hat{y}_n$. Formally,

$$MSE = \frac{1}{N}\sum_{n=1}^{N} ||y_n - \hat{y}_n||_2^2.$$

Remember how we predict

$$\hat{y}_n = \mathbf{w}^T \mathbf{x}_n,$$

and plug it into the MSE definition

$$MSE = \frac{1}{N}\sum_{n=1}^{N} ||y_n - \underbrace{\mathbf{w}^T \mathbf{x}_n}_{\hat{y}_n}||_2^2.$$

How to learn?

We need to maximize our performance (remember Mitchell's definition of learning), hence minimize MSE

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} ||y_n - \hat{y}_n||_2^2.$$

How to minimize MSE?

Hint:

$$
\begin{aligned}
||y_n - \hat{y}_n||_2^2 &= (y_n - \hat{y}_n)^2 \\
&= y_n^2 + (\mathbf{w}^T\mathbf{x}_n)^2 - 2y_n\mathbf{w}^T\mathbf{x}_n \\
&= y_n^2 + \mathbf{w}^T\mathbf{x}_n\mathbf{x}_n^T\mathbf{w} - 2y_n\mathbf{w}^T\mathbf{x}_n
\end{aligned}
$$

How to minimize MSE?

Calculate the gradient with respect to the model parameters $\mathbf{w}$ we aim to learn

$$
\begin{aligned}
\nabla_{\mathbf{w}} MSE &= \nabla_{\mathbf{w}} \sum_{n=1}^{N} ||y_n - \mathbf{w}^T \mathbf{x}_n||_2^2 \triangleq 0 \\
&= \nabla_{\mathbf{w}} \sum_{n=1}^{N} (y_n^2 + \mathbf{w}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{w} - 2 y_n \mathbf{w}^T \mathbf{x}_n) \\
&= \sum_{n=1}^{N} \nabla_{\mathbf{w}} (\mathbf{w}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{w}) - 2 \sum_{n=1}^{N} \nabla_{\mathbf{w}} y_n \mathbf{w}^T \mathbf{x}_n \\
&= \sum_{n=1}^{N} 2 \mathbf{x}_n \mathbf{x}_n^T \mathbf{w} - 2 \sum_{n=1}^{N} y_n \mathbf{x}_n
\end{aligned}
$$

How to minimize MSE?

Cont'd...

$$\begin{aligned}\nabla_{\mathbf{w}} MSE &= \sum_{n=1}^{N} 2\mathbf{x}_n \mathbf{x}_n^T \mathbf{w} - 2\sum_{n=1}^{N} y_n \mathbf{x}_n \\ &= 2\left[\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T\right]\mathbf{w} - 2\sum_{n=1}^{N} y_n \mathbf{x}_n \\ &= 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}\end{aligned}$$

Now solve for the gradient at zero:

$$\begin{aligned}\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} &\triangleq 0 \\ \mathbf{X}^T\mathbf{X}\mathbf{w} &\triangleq \mathbf{X}^T\mathbf{y} \\ \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\left[\mathbf{X}^T\mathbf{X}\right]\mathbf{w} &\triangleq \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{y} \\ \mathbf{w} &\triangleq \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

Underfitting and overfitting

We expect a good machine learning model to exhibit two properties

- make the training error small (*underfitting* otherwise),
- make the gap between training and test error small (*overfitting* otherwise).

The continuum between underfitting and overfitting can be traversed by tuning the *model capacity* (e.g. the degree of a polynomial, number of neurons/layers in a deep neural net).

Underfitting and overfitting

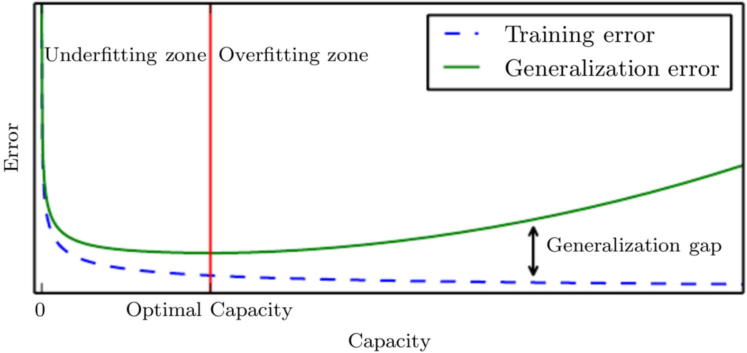


Figure. Goodfellow et al., Deep Learning, MIT Press, 2016

Occam’s razor principle

“Among hypotheses explaining a set of observations with equal success, choose the simplest one.“

(a.k.a. The Law of Parsimony)

Applied to machine learning: *“Among models performing equally well, choose the simplest one.“*

How would you quantify model capacity?

Vapnik-Chervonenskis (VC) dimension

- **Definition.** Maximum number of arbitrarily placed data points that can be perfectly predicted.
- VC dimension of linear regression on D dimensional input and an intercept is $D + 1$ (i.e. number of model parameters).

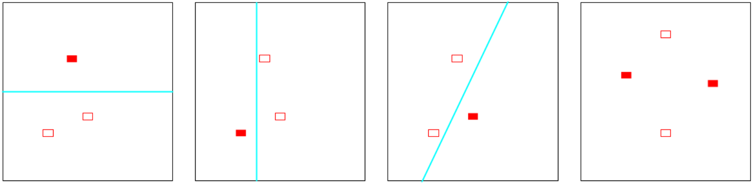


Figure. T. Hastie et al., The Elements of Statistical Learning, Springer, 2001

VC dimension of the sine function

- High capacity does not necessarily imply high parameter count.
- $\mathrm{sine}(\alpha \cdot x)$ has one parameter α, but infinite VC dimension.

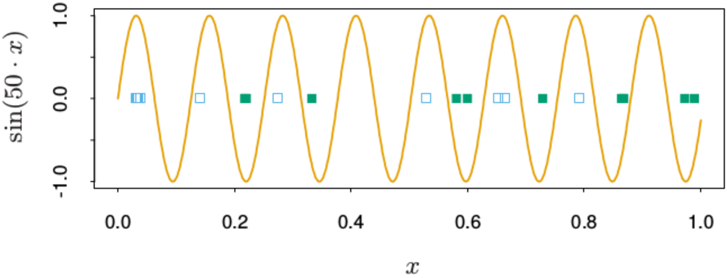


Figure. T. Hastie et al., The Elements of Statistical Learning, Springer, 2001

Quantifying the optimism of training error

(Only for binary classification for simplicity)

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

Here, h is the VC dimension of the predictor, err_{Tr} is the **training error** (or in-sample error) and $\mathrm{Err}_t s$ is the **test error** (or generalization error), and $0 < a_1 \leq 4$ and $0 < a_2 \leq 2$ are arbitrary coefficients satisfying the given inequalities.

Lastly, $opt = \mathrm{Err}_{Ts} - \mathrm{err}_{Tr}$ is the **optimism of the training error**.

For more details, see Cherkassky and Mulier, Learning from Data, Wiley, 2007.

What happens if $h \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

What happens if $h \rightarrow +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

Let's check out the asymptotic behavior: $\lim_{h \rightarrow +\infty} \epsilon = \infty$, hence $opt \rightarrow +\infty$ (overfitting!).

What happens if $N \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

What happens if $N \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

Let's check out the asymptotic behavior: $\lim_{N \to +\infty} \epsilon = 0$, hence $opt \to 0$ (actual fitting!).

Supervised learning as estimator inference

Estimator: A rule for inferring the unknown value of a quantity of interest of a true mechanism from data.

- **True Mechanism:**

$$\begin{array}{ll} x \sim P_{true}(x) & \text{Input distribution} \\ \epsilon \sim P_{true}(\epsilon) & \text{Noise process} \\ y = f(x, \epsilon) & \text{Structural assignment} \end{array}$$

- **Data:** Training set $\mathcal{D} = \{(x_n, y_n) | n = 1, \ldots, N\}$ with $(x_n, y_n) \sim p_{true}(x), \quad \epsilon_n \sim p_{true}(\epsilon), \quad y = f(x_n, \epsilon_n)$.
- **Quantity of Interest:** Expected structural assignment

$$\mathbb{E}[y|x] = \mathbb{E}_{\epsilon|x}[f(x, \epsilon)]$$

- **Estimator:** A hypothesis $h_{\mathcal{D}}(x)$ learned from training data $\mathcal{D}$.

Supervised learning as estimator inference

- **Loss function:** The difference between the hypothesis and the true value of the quantity of interest. Examples:
  - **Zero-one loss:**

    $$L(y, h_{\mathcal{D}}(x)) = I(y \neq h_{\mathcal{D}}(x)) = \begin{cases} 0 & \text{if } y = h_{\mathcal{D}}(x) \\ 1 & \text{if } y \neq h_{\mathcal{D}}(x) \end{cases}$$

  - **Squared loss:** $L\left(y, h_{\mathcal{D}}(x)\right) = \left(y - h_{\mathcal{D}}(x)\right)^2$
- **Expected loss/risk:** Take into account all possible query inputs x weighted proportional to their probability of occurrence with respect to $P_{true}(x)$:

$$\mathbb{E}[L(y, h_{\mathcal{D}}(x))] = \mathbb{E}_x\left[\mathbb{E}_{\epsilon|x}[L(f(x,\epsilon), h_{\mathcal{D}}(x))]\right]$$

Example: Minimizing Expected Zero-One-Loss

- Using the zero-one loss, minimizing the loss corresponds to minimizing the error rate:

$$
\begin{aligned}
\mathbb{E}_{y|x}[L(y, h_{\mathcal{D}}(x))] &= \mathbb{E}_{y|x}\left[I(y \neq h_{\mathcal{D}}(x))\right] \\
&= \sum_{y \in C} I(y \neq h_{\mathcal{D}}(x)) \cdot \Pr(c|x) \\
&= \sum_{y \in C, y \neq h_{\mathcal{D}}(x)} \Pr(y|x) \\
&= 1 - \Pr(h_{\mathcal{D}}(x)|x)
\end{aligned}
$$

  where C is the set of class labels.
- Choose $h_{\mathcal{D}}(x)$ to maximize the posterior probability, i.e., $h_{\mathcal{D}}(x) = \arg\max_{y \in C} \Pr(y|x)$ (i.e. MAP prediction).

Bias of an estimator

A bias of an estimator $\hat{\theta}_N$ is defined as

$$\text{bias}(\hat{\theta}_N) = \mathbb{E}[\hat{\theta}_N] - \theta.$$

An estimator is said to be **unbiased** if

$$\text{bias}(\hat{\theta}_N) = 0,$$

or equivalently,

$$\mathbb{E}[\hat{\theta}_N] = \theta.$$

An estimator is said to be **asymptotically unbiased** if

$$\lim_{N \to \infty} \text{bias}(\hat{\theta}_N) = 0.$$

Example 1: Sample mean is an unbiased estimator

Assume we have N samples $x_1, \cdots, x_N$ coming from a normal distribution $\mathcal{N}(\mu, \sigma^2)$. We are interested in finding out the true mean μ. Choose the **sample mean** as the estimator for μ:

$$\hat{\mu}_N = \frac{1}{N} \sum_{n=1}^{N} x_n.$$

Now check the bias of the estimator:

$$\begin{aligned} \mathrm{bias}(\hat{\mu}_N) &= \mathbb{E}[\hat{\mu}_N] - \mu \\ &= \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^{N} x_n\right] - \mu \\ &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[x_n] - \mu = 0. \end{aligned}$$

It is unbiased!

Example 2: Sample variance is a biased estimator

$$\mathbb{E}[\hat{\sigma}_N^2] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$$

The bias is $-\sigma^2/N$, which is ignorable if N is large [1].

A small fix, called *Bessel's correction*, solves the issue. The sample variance is unbiased if defined as

$$\hat{\sigma}_N^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \hat{\mu})^2.$$

[1] Derivation is lengthy but simple, see: https://www.marcovicentini.it/wp-content/uploads/2014/07/La-correlazione-di-Bessel.pdf

Standard error

The square-root of the variance of an estimator is called its **standard error**:

$$SE(\hat{\theta}) = \sqrt{Var[\hat{\theta}]}.$$

Taking the sample mean as the estimator, we have

$$SE(\hat{\theta}) = \sqrt{Var\left[\frac{1}{M}\sum_{m=1}^{M} x_m\right]} = \frac{\sigma}{\sqrt{M}},$$

which is called the **standard error of the mean.**

Bias, variance, and noise in supervised learning

- **Bias**: Degree of inflexibility of the learning hypothesis (its level of ignorance to observations). Stems from modeling assumptions required to develop generalizeable concepts from individual observations. *"Bias-free learning is futile."* (Mitchell)
- **Variance**: The sensitivity of the learned predictor to individual training samples.
- **Noise**: Incorrectly labeled data for classification or lack of measurement precision for regression.

Where is the trade-off here?

- High bias $\Rightarrow$ Assumptions dominate data $\Rightarrow$ Low variance $\Rightarrow$ Underfitting
- High variance $\Rightarrow$ Data dominate observations $\Rightarrow$ Low bias $\Rightarrow$ Overfitting

The bias-variance trade-off

For a training set $\mathcal{D}$ and a test-time query input x sampled from a true distribution $(x, y) \sim p_{true}(x), \epsilon \sim p_{true}(\epsilon), y = f(x, \epsilon)$, the expected squared loss of hypothesis $h_{\mathcal{D}}$ trained on $\mathcal{D}$ is

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D},\epsilon|x}\left[\left(y - h_{\mathcal{D}}(x)\right)^2\right] &= \mathbb{E}_{\mathcal{D},\epsilon|x}\left[f(x,\epsilon)^2 - 2f(x,\epsilon)h_{\mathcal{D}}(x) + h_{\mathcal{D}}(x)^2\right] \\
&= \mathbb{E}_{\epsilon|x}[f(x,\epsilon)^2] + \mathbb{E}_{\mathcal{D}|x}[h_{\mathcal{D}}(x)^2] - 2\mathbb{E}_{\epsilon|x}[f(x,\epsilon)]\mathbb{E}_{\mathcal{D}|x}[h_{\mathcal{D}}(x)] \\
&= \mathbb{E}_{\epsilon|x}[f(x,\epsilon)]^2 + Var_{\epsilon|x}[f(x,\epsilon)] + \mathbb{E}_{\mathcal{D}|x}[h_{\mathcal{D}}(x)]^2 + Var_{\mathcal{D}|x}[h_{\mathcal{D}}(x)] \\
&\qquad - 2\mathbb{E}_{\epsilon|x}[f(x,\epsilon)]\mathbb{E}_{\mathcal{D}|x}[h_{\mathcal{D}}(x)] \\
&= \Bigg(\underbrace{\mathbb{E}_{\epsilon|x}[f(x,\epsilon)] - \mathbb{E}_{\mathcal{D}|x}[h_{\mathcal{D}}(x)]}_{Estimator\ Bias}\Bigg)^2 + \underbrace{Var_{\mathcal{D}|x}[h_{\mathcal{D}}(x)]}_{Estimator\ Variance} \\
&\qquad + \underbrace{Var_{\epsilon|x}[f(x,\epsilon)]}_{Label\ noise\ variance}
\end{aligned}
$$

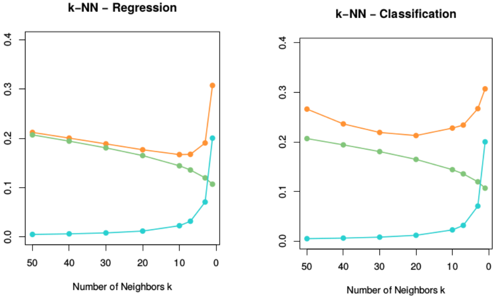
Example: k-nearest neighbor regression

Choose the estimator to be $h_{\mathcal{D}}(x) = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$, where $\{x_1, \cdots, x_k\}$ is the set of k nearest neighbors to x. Then,

$$\text{err} = \Big(f(x) - \frac{1}{k}\sum_{i=1}^{k} f(x_k)\Big)^2 + \underbrace{Var\Big[\frac{1}{k}\sum_{i=1}^{k} f(x_k)\Big]}_{SE(h_{\mathcal{D}}(x)))^2}$$

Increasing k increases bias. If $k = N$, the model outputs the sample mean regardless of the input.

Example: k-nearest neighbor regression



Regression

Classification

Orange: Expected prediction error (MSE), **Green:** squared bias, **Blue:** Variance

Figure. T. Hastie et al., The Elements of Statistical Learning, Springer, 2001

Epistemic and Aleatoric Uncertainty

Statistical Learning Theory is interested in bounding the *expected* risk with respect to all possible query samples

$$\mathbb{E}_x\left[\mathbb{E}_{\epsilon,\mathcal{D}|x}\left[(f(x,\epsilon)-h_{\mathcal{D}}(x))^2\right]\right]=\mathbb{E}_{x,\epsilon,\mathcal{D}}\left[(f(x,\epsilon)-h_{\mathcal{D}}(x))^2\right].$$

Then the bias-variance decomposition will read

$$\mathbb{E}_{x,\epsilon,\mathcal{D}}\left[(f(x,\epsilon)-h_{\mathcal{D}}(x))^2\right]=\underbrace{\mathbb{E}_x\left[\Big(\mathbb{E}_{\epsilon|x}[f(x,\epsilon)]-\mathbb{E}_{\mathcal{D}|x}[h_{\mathcal{D}}(x)]\Big)^2\right]}_{Bias\ \ (Reducible)}$$
$$+\quad\underbrace{\mathbb{E}_x\left[Var_{\mathcal{D}|x}[h_{\mathcal{D}}(x)]\right]}_{Epistemic\ \ Uncertainty\ \ (Reducible)}\quad+\quad\underbrace{\mathbb{E}_x\left[Var_{\epsilon|x}[f(x,\epsilon)]\right]}_{Aleatoric\ \ Uncertainty\ \ (Irreducible)}$$

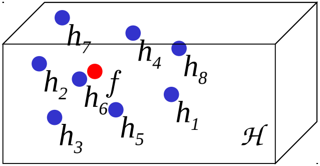
As $|\mathcal{D}|\rightarrow+\infty$, both bias and variance reduces as both are epistemic in origin (i.e. lack of knowledge on p_{true}), but noise variance does not as it is intrinsic noise in the process.

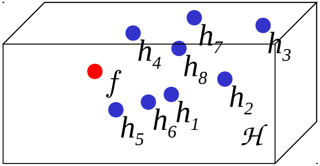
Bias-Complexity Trade-off

- We could optimize (minimize) the bias (systematic deviation from the optimal decision boundary, i.e., approximation error), if we knew which kind of model fits best to the domain.
- If we don't know this — should we choose some learner with a generally weak bias?
- Learners with a weaker bias are typically producing a more complex model and this way approximate better — on the danger of overfitting.
- Overfitting is a manifestation of a large variance component of the error — the estimation error.
- Trade-off between bias and variance:
  - the weaker the bias, the larger the variance
  - the stronger the bias, the smaller the variance

Ensembles

- Any individual classifier would have either a strong bias or a large variance on a non-trivial learning task.
- The combination of classifiers can reduce both, bias and variance:
    - We can combine classifiers with a weak bias, thus a large variance.
    - Averaging reduces the overall variance.





- We can combine classifiers with strong bias (and thus typically small variance), but choose them in a way to diversify the biases.
- Averaging reduces the overall bias.

Information

We would like to *measure* the amount of information received when a binary variable $x \in \{0, 1\}$ is observed.

Information: Degree of surprise after observing x.

Devise a function $h(x)$ to quantify information gained from x.

How should $h(x)$ look like?

When we observe two independent binary variables x and y, the information received should be the sum of the individual events.

Because independence implies $p(x, y) = p(x)p(y)$, it is suitable to measure information by

$$h(x) = -\log_2 p(x).$$

Base 2 is arbitrary, except having historical roots at communication theory. When base 2 is used, the measure is called a *bit*!

Negative sign assures that information with surprise, i.e. occurrence of a low-probability event.

Entropy

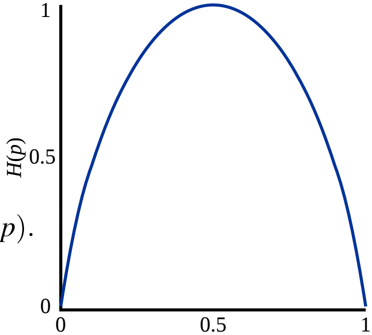
Expected amount of information for random variable x living in a sample space $\mathcal{X}$ and following a distribution $p(x)$:

$$H[x] = -\sum_{x \in \mathcal{X}} \log_2 p(x) p(x).$$

Note that the case for $p(x) = 0$ looks degenerate. Handle this by $\lim_{p \to 0} p \ln p = 0$, hence $H[x] = 0$.

For binary random variables, we have

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$



Example 1

Consider the case where we have four possible states. When they are equally likely, the entropy turns out to be

$$H[x] = 4 \times \left[-\frac{1}{4} \log_2 \frac{1}{4} \right] = 2 \text{ bits}.$$

Example 2

Assume we have again four possible states, this time with probabilities $\left(\frac{5}{8}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16}\right)$. Then the entropy is

$$\begin{aligned} H[x] &= -\frac{5}{8}\log_2\frac{5}{8} - \frac{1}{4}\log_2\frac{1}{4} - 2\frac{1}{16}\log_2\frac{1}{16} \\ &= 0.42 + 0.5 + 0.5 = 1.42 \text{ bits.} \end{aligned}$$

There is more information in the uniform case!

n-ary Entropy (Uniform)

Generalizing, the entropy of a random variable X with n outcomes of equal probability is given by:

$$\begin{aligned}
H(X) &= -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} \\
&= -\sum_{i=1}^{n} \log_2 \left(\frac{1}{n}\right)^{\frac{1}{n}} \\
&= -\log_2 \left(\left(\frac{1}{n}\right)^{\frac{1}{n}}\right)^{n} \\
&= -\log_2 \frac{1}{n} \\
&= \log_2 n
\end{aligned}$$

Intuition

- Imagine a die with 8 sides that show up with equal probability.
- The entropy is 3 bits.
- If the faces of the die were numbered with 0 to 7 in binary code, the outcome of a die roll would give a sequence of 3 bits uniform over the set $\{0, 1\}^3$.
- This shows the equivalence to generating 3 bits independently and uniformly at random.

Note that:

The entropy of a random variable X does not depend on the values that X can take but only on the probability distribution of X over those values.

Interpretations

- One interpretation of entropy is that it measures the amount of randomness (disorder, uncertainty) in a system:
  - For example consider the second law of thermodynamics: the total entropy of an isolated system cannot decrease over time.
- Another interpretation relates entropy to compression and coding theory:
  - Entropy relates to the minimum number of bits per symbol required to encode a message.
  - Encode frequent messages with short bit sequences and rare messages with long bit sequences

Measures of information content

- Measures of information content

$$
\begin{aligned}
-\log_2 p(x) &\rightarrow \text{bits} \\
-\ln p(x) &\rightarrow \text{nats}
\end{aligned}
$$

- Distributions that maximize the entropy
  - Discrete → uniform
  - Continuous (for a given location and spread) → normal!

Entropy in the continuous domain

- No exact counterpart.
- Using mean value theorem, we attain that entropy of a continuous density $p(x)$ differs from the term below by $-\ln\Delta$

$$H[x] = -\int p(x)\log p(x)dx.$$

  This term is called the *differential entropy*.
- Although differential entropy diverges from the exact entropy as $\Delta \to 0$, it is often used in place of the plain entropy for continuous densities. We will adopt the same convention here.

Relative entropy or KL divergence

Suppose for some reason, we need to approximate $p(x)$ by another density $q(x)$, which has some more pleasant properties. The *additional information* (in nats) required to be conveyed as a result of using $q(x)$ in place of $p(x)$ is

$$-\log q(x) - \Big(-\log p(x)\Big) = -\log \frac{q(x)}{p(x)} = \log \frac{p(x)}{q(x)}.$$

Since x follows $p(x)$, the expected additional information is

$$\mathbb{KL}[p||q] = \int \log \frac{p(x)}{q(x)} p(x) dx.$$

This quantity is called *relative entropy* or *Kullback-Leibler divergence* and denoted by $\mathbb{KL}[p||q]$.

Convexity

Consider a parametric line $a\lambda + b(1-\lambda)$ that passes between points a and b and an arbitrary function $f(x)$. If any line passing between $f(a)$ and $f(b)$ is always above $f(x)$, then $f(x)$ is called a *convex function*. More formally, if for any a and b the below inequality satisfies

$$f(a)\lambda + f(b)(1-\lambda) \geq f(a\lambda + b(1-\lambda)),$$

then $f(x)$ is said to be convex.

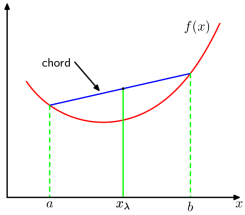


Figure: C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Jensen’s inequality

We can prove by induction that convexity holds also for more than two points:

$$\sum_{i=1}^{M} \lambda_i f(x_i) \geq f\left(\sum_{i=1}^{M} \lambda_i x_i\right),$$

such that $\{x_1, \cdots, x_M\}$ is a set of points on the function domain and $\sum_{i=1}^{M} \lambda_i = 1$ with $\lambda_i \geq 0$. We can think of $\{\lambda_i, \cdots, \lambda_M\}$ as parameters of a categorical distribution with M states. Hence we can equivalently write

$$\mathbb{E}[f(x)] \geq f(\mathbb{E}[x]).$$

This outcome generalizes to continuous variables straightforwardly (use Riemann integration):

$$\int f(x)p(x)dx \geq f\left(\int xp(x)dx\right).$$

KL divergence is a statistical distance

Considering that $-\log x$ is a convex function,

$$\begin{aligned} \mathbb{KL}[p||q] &= -\int p(x) \log \frac{q(x)}{p(x)} dx \\ &\geq -\log \underbrace{\int p(x) \frac{q(x)}{p(x)} dx}_{1} = 0. \end{aligned}$$

Because $-\log x$ is a *strictly* convex function (i.e. equality holds only at intersection points),

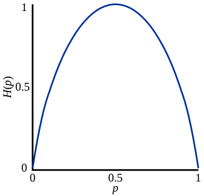
$$p(x) = q(x) \iff \mathbb{KL}[p||q] = 0.$$

Hence, KL divergence is a statistical distance measure between two distributions. Note that $\mathbb{KL}[p||q] \neq \mathbb{KL}[q||p]$.

Entropy of class distributions

$$H(\mathcal{D}) = -\sum_{i=1}^{k} \Pr(c_i|\,\mathcal{D}) \log_2 \Pr(c_i|\,\mathcal{D})$$

- Considering labeled data with $k = 2$, a very pure set (almost all labels belong to class A, only some belong to class B) has very low entropy (i.e., low disorder, low uncertainty):



- If we draw some data object at random, we are very likely to get one that belongs to class A.
- The more equal the proportions of the two classes are, the more uncertainty we have about which class we will likely draw (i.e., higher disorder, higher entropy).

Gini Index

The Gini index as a measure for the (im-)purity of a data set $\mathcal{D}$ w.r.t. k classes $c_1, \ldots, c_k$ is given by:

$$G(\mathcal{D}) = 1 - \sum_{i=1}^{k} \Pr(c_i|\,\mathcal{D})^2$$

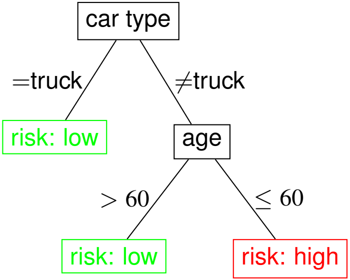
- If a dataset contains only one class, the probability of that class is 1, the dataset has minimal impurity, the Gini index is 0.
- When each class is equally represented, we have $\Pr(c_i|\,\mathcal{D}) = \frac{1}{k}$, the dataset is maximally impure, and the Gini index is $\frac{k-1}{k}$ (i.e., approaching 1 as $k \to \infty$).
- Probabilistic interpretation of the square: if we randomly draw two objects from $\mathcal{D}$, how likely are they belonging to the same class?

Decision Tree – Basic Idea

Data:

ID	age	car type	risk
1	23	family car	high
2	17	sports car	high
3	43	sports car	high
4	68	family car	low
5	32	truck	low

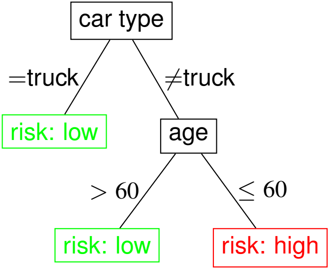
Decision tree:



- A decision tree provides explicit knowledge on the data.
- The classification model is interpretable (a hierarchy of rules).

Decision Tree – Properties

- A decision tree is a tree with the following properties:
  - an inner node represents a test on an attribute
  - an edge represents a test result on the parent node
  - a leaf node represents one of the classes



- Construction: top-down based on the training set
- Application (prediction):
  - traversal according to the tests from the root to some leaf node (deterministic path)
  - class assignment: the class of the leaf node reached in the traversal

Decision Tree – Basic Algorithm

Algorithm 0.1 (Decision Tree)

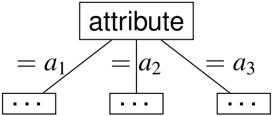
1. *given a dataset, select an attribute and split point (greedy selection, following some split strategy)*
2. *partition the data according to the test on the split attribute*
3. *repeat the procedure recursively for each partition*

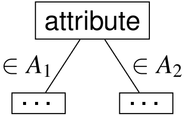
The recursion stops if the partition is "pure" (contains only examples of a single class).

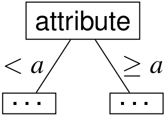
Types of Splits

categorical attributes

- attribute $= a$
- attribute $\in A$







numerical attributes

- attribute $< a, \leq a$
- attribute $\geq a, > a$

Problem: Where to Split Numerical Attributes

Where should we define a split for some numerical attribute?

- We want to maximize the separation of classes.
- Idea: sort attribute values

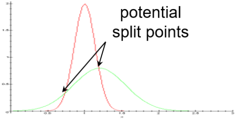
value	0.9	0.8	0.65	0.5	0.45	0.3	0.15	0.01
class	A	A	B	B	B	A	A	A

potential split points

- test combinations with split criterion

alternative:

- fit to each class a Gaussian distribution
- intersections of the Gaussian pdfs are potential split points



Quality Measure for Splits

To select attributes for splitting and to select split points in attributes, we need to assess the quality of partitions induced by a split on some attribute.

given

- a training set TR
- disjunct and complete partitionings $T = T_1, \ldots, T_m$ of TR: $\cup_i T_i = TR, \forall i \neq j : T_i \cap T_j = \emptyset$
- relative frequency of each class c_i in each partition T_j: $p_i = \Pr(c_i | T_j)$

required

- a relative measure for the purity w.r.t. classes of some set of partitions
- a split that optimizes this measure

Measure: Information Gain

- Information gain is a measure based on the entropy.

$$H(T) = -\sum_{i=1}^{k} \Pr(c_i|T) \log_2 \Pr(c_i|T)$$

- Information gain measures the reduction of entropy (i.e., gain of information) by a split of set T into partitions $T_1, \ldots, T_m$:

$$\textit{information gain}(T, T_1, \ldots, T_m) = H(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} H(T_i)$$

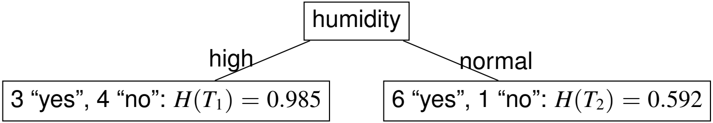
- Higher information gain means larger reduction of entropy.
- We choose the attribute and split point that maximize the information gain.

Example: Should We Play Tennis Today?

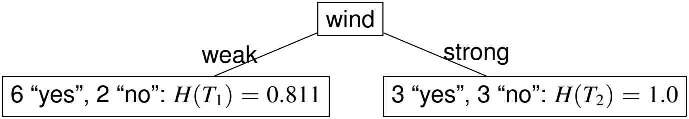
ID	forecast	temperature	humidity	wind	play tennis?
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

Example: Information Gain

9 “yes”, 5 “no”: $H(T) = 0.940$

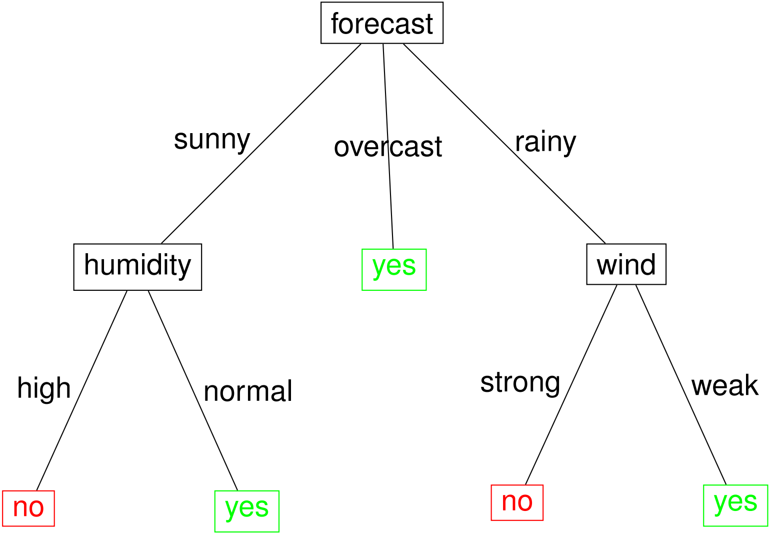

humidity
high
normal
3 “yes”, 4 “no”: $H(T_1) = 0.985$
6 “yes”, 1 “no”: $H(T_2) = 0.592$

$$\textit{inf. gain}(T, T_i(\textit{humidity})) = 0.940 - \frac{7}{14}0.985 - \frac{7}{14}0.592 = 0.151$$


wind
weak
strong
6 “yes”, 2 “no”: $H(T_1) = 0.811$
3 “yes”, 3 “no”: $H(T_2) = 1.0$

$$\textit{inf. gain}(T, T_i(\textit{wind})) = 0.940 - \frac{8}{14}0.811 - \frac{6}{14}1.0 = 0.048$$

Example Decision Tree



Measure: Gini Index

$$G(\mathcal{D}) = 1 - \sum_{i=1}^{k} \Pr(c_i | \mathcal{D})^2$$

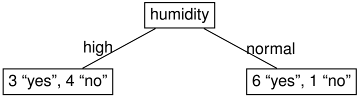
In an analogous way, we can use the weighted Gini index of induced partitions to compare partitionings:

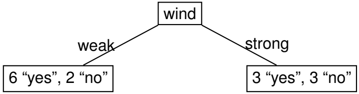
$$G(T_1, \ldots, T_m) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} G(T_i)$$

- Smaller value of the Gini index means lower impurity.
- We choose the attribute and the split that minimizes the Gini index.

Example: Gini Index

9 “yes”, 5 “no”: $|T| = 14$





$$G(T_1)=1-\left(\frac{3^2}{7^2}+\frac{4^2}{7^2}\right)=\frac{24}{49}$$

$$G(T_2)=1-\left(\frac{6^2}{7^2}+\frac{1^2}{7^2}\right)=\frac{12}{49}$$

$$G(T_1)=1-\left(\frac{6^2}{8^2}+\frac{2^2}{8^2}\right)=\frac{24}{64}=\frac{3}{8}$$

$$G(T_2)=1-\left(\frac{3^2}{6^2}+\frac{3^2}{6^2}\right)=\frac{18}{36}=\frac{1}{2}$$

$$G(\textit{split on humidity}) = \frac{7}{14}\cdot\frac{24}{49}+\frac{7}{14}\cdot\frac{12}{49}=\frac{18}{49}$$

$$G(\textit{split on wind}) = \frac{8}{14}\cdot\frac{3}{8}+\frac{6}{14}\cdot\frac{1}{2}=\frac{3}{7}=\frac{21}{49}$$

Axis-parallel Hyperplanes

- A hyperplane $h(x)$ is defined as the set of all points $x \in \mathbb{R}^d$ that satisfy:

  $$h(x) : w \cdot x^{\mathsf{T}} - b = 0,$$

  where w is a normal vector to the hyperplane, and b defines the offset of the hyperplane from the origin.
- For axis-parallel hyperplanes, the normal vector is parallel to one of the axes, i.e., $w \in \{e_1, \ldots, e_d\}$, where $e_i \in \mathbb{R}^d$ has value 1 in dimension X_i and value 0 in every other dimension.

  $$\begin{aligned} h(x) &: e_i \cdot x^{\mathsf{T}} - b = 0 \\ \equiv h(x) &: x_i - b = 0 \end{aligned}$$

- The choice of b yields different hyperplanes parallel to axis X_i.

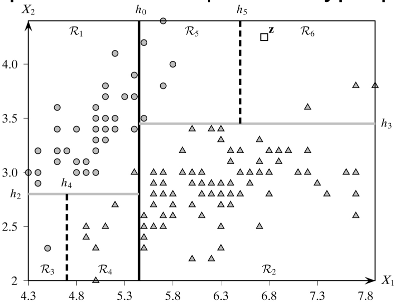
Hyperplanes and Split Points

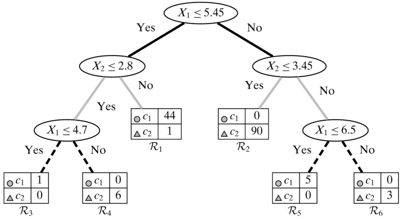
- For decision trees on real-valued attributes, axis-parallel hyperplanes relate to split points.
- A hyperplane splits the data space $\mathbb{R}^d$ into two half-spaces.
- All points with $h(x) < 0$ are on one side of the hyperplane, all points with $h(x) > 0$ are on the other side of the hyperplane.
- We can therefore write the split point as:

$$h(x) \leq 0 \quad \Leftrightarrow \quad x_i - b \leq 0 \quad \Leftrightarrow \quad x_i \leq b$$

Bias: Axis-Parallel Piecewise Linear Separation

The bias of a decision tree is therefore to separate data along piecewise axis-parallel hyperplanes.



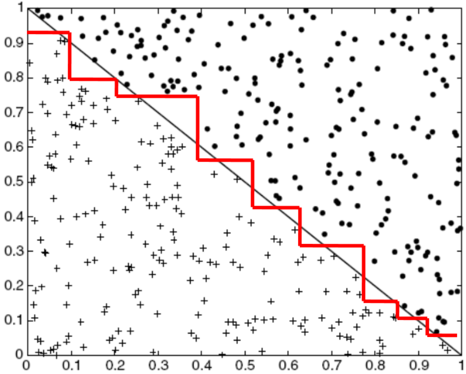


Figures from **?**.

Bias Prevents Detection of Certain Simple Decision Rules

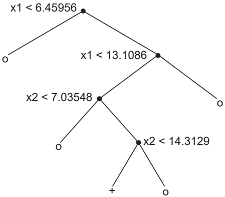
Consider a dataset with a relatively simple decision rule:

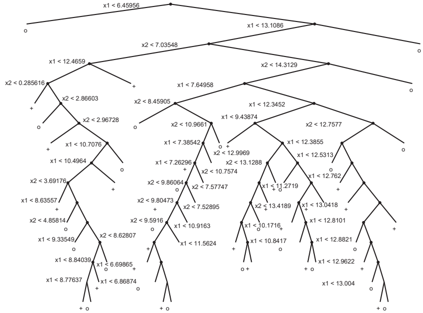
- if $x + y \leq 1 \Rightarrow +$
- else $\Rightarrow \bullet$

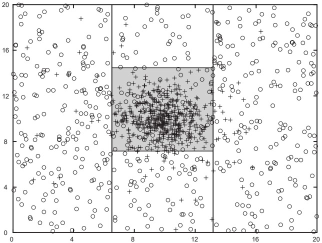


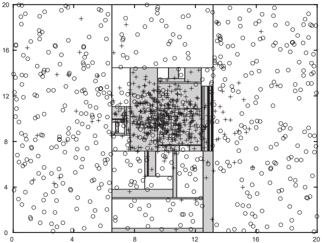
- A decision tree is biased to find a more complex model, as it can only find piecewise axis-parallel hyperplanes as split points.

Overfitting Example



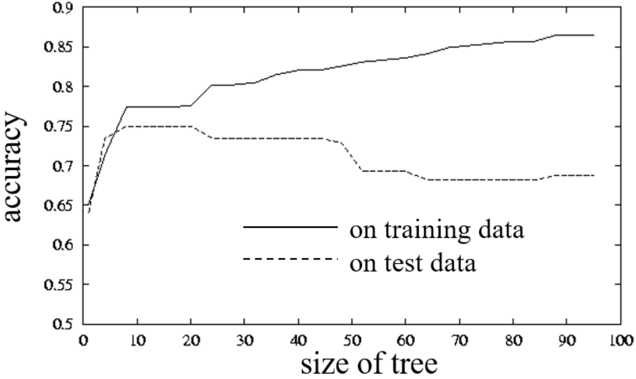






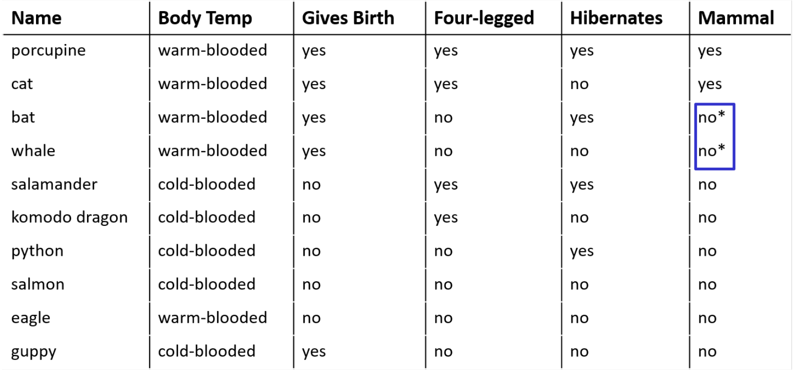
Overfitting in decision trees

- Decision trees are susceptible to overfitting in general.
- With increasing size, decision trees tend to overfit more.



Overfitting scenario

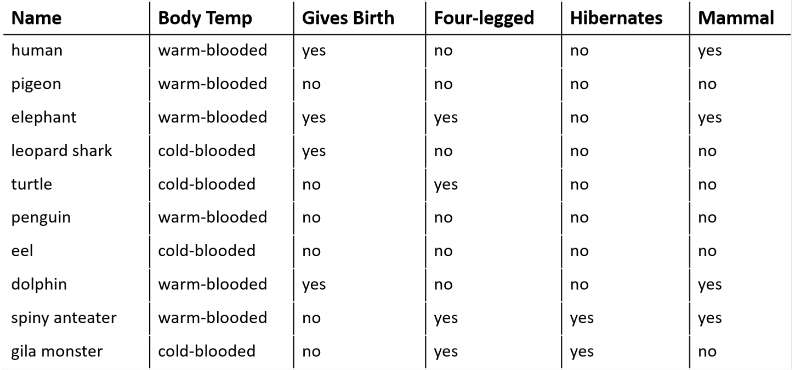
Training data with * mislabeled examples:



Name	Body Temp	Gives Birth	Four-legged	Hibernates	Mammal
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

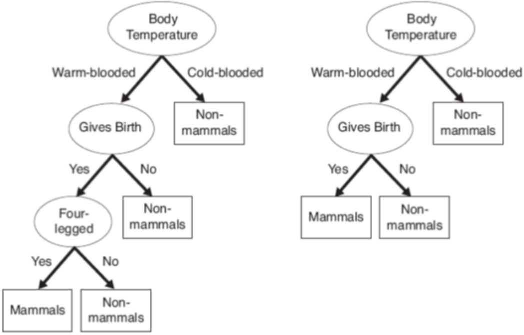
Overfitting Scenario

Test data:



Name	Body Temp	Gives Birth	Four-legged	Hibernates	Mammal
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	warm-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

Two Models



(a) Model M1

(b) Model M2

training error: 0%

test error: 30%

training error: 20%

test error: 10%

M1:

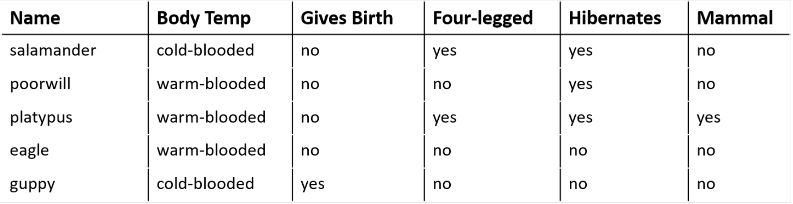
- misclassification of dolphin and human due to overfitting (on mislabeled training data)
- spiny anteater: unusual case

M2:

- misclassification of unusual case cannot be avoided

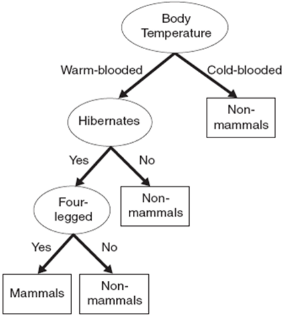
Lack of Training Data

Consider this small training set, containing unusual cases:



Name	Body Temp	Gives Birth	Four-legged	Hibernates	Mammal
salamander	cold-blooded	no	yes	yes	no
poorwill	warm-blooded	no	no	yes	no
platypus	warm-blooded	no	yes	yes	yes
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

The resulting decision tree has a test error of 30%:



Heuristics for Avoiding Overfitting

- remove erroneous training data
  (in particular contradictory training data)
- suitable amount of training data
  (training set should be large enough, but too large can be counterproductive as well, as it might contain too many special cases — it should be *representative*)
- choose a minimum support $\gg 1$ for leaves: number of training examples that need to belong to a leaf node
- choose a minimum confidence $\ll 100\%$ (purity), as fraction that the majority class in a leaf node needs to satisfy
  (leaves can absorb erroneous or unusual data, noise)

Error-Reduction-Pruning

Cut overspecialized branches of the tree (as smaller trees are less susceptible to overfit – cf. Occam's razor):

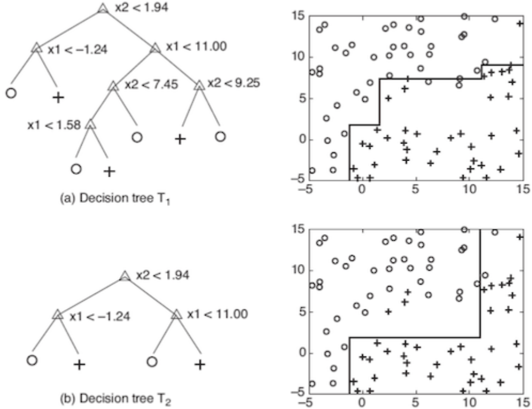
Algorithm 0.2 (Error-Reduction-Pruning)

- *Split training data into model training data MTR and model selection data MS.*
- *Construct decision tree E on MTR.*
- *Prune E on MS:*
  - *Find the subtree E_i such that reduction of the error on MS is maximal for $E \setminus E_i$.*
  - *Remove E_i from E.*
  - *Stop if no such subtree exists.*

General Principle: Model Tuning and Data Snooping

- Assume we want to tune some model, e.g., by optimizing some parameter:
  - Train the classifier with some parameter setting on the training data TR.
  - Test the performance of the learned model with the given parameter on the test set TE.
  - Repeat the procedure for a different parameter setting (possibly using grid search for the optimal setting).
- In such a setting, we made use of the test set TE for optimizing the model: we ask new questions to the training data, based on the answers we got from the test data.
- An additional test set is required, that is independent of both, the training set TR and the test set TE.

Different Approximation Error of Decision Trees of Different Depth



(a) Decision tree T_1

(b) Decision tree T_2

based on a figure by Tan et al.

- T_1 and T_2 are trained on the same data.
- T_2 is a pruned version of T_1.
- T_2 has stronger assumptions on separability of classes, i.e., stronger bias.

Discussion

pro

- The decision tree model is easily interpretable and provides human-readable information on the data.
- The structure of the tree provides an implicit weighting of attributes.
- Decision trees are typically strong classifiers and are often used in practice.
- The classifier facilitates efficient application of the derived classification model on new data.

con

- Finding the optimal tree is exponential.
- Heuristic (greedy!) tree building algorithms can only find local optimum.
- Decision trees are susceptible to overfitting (error reduction pruning should be considered).