Knowledge Representation Model Selection and Assessment

(c) Marcin Sydow

Topics covered by this lecture:

- knowledge representation
- decision rules
- decision trees and ID3 algorithm
- model complexity
- model selection and assessment
- overfitting and methods of overcoming it
- cross-validation

Variety of ML models

There are many models available in machine learning:

- neural networks
- decision trees
- decision rules
- support vector machines
- and many others ...

Neural Networks as a "black box"

Multi-layer non-linear neural network is a powerful tool used in machine learning and AI.

However, in NN, the learnt "knowledge" is "encoded" with the numerical values of weights and thresholds.
Such encoding is uncomprehensible for humans for analysis.
Due to this, NN are considered as an example of a so-called "black box" model. Providing input, it produces useful output but the internal structure is inpenetrable.

Knowledge Representation

There are models in machine learning, other than NN, that represent the learnt knowledge in much more interpretable way. For example:

- Decision rules
- Decision trees

Example - medicine

Knowledge in the "raw" form of decision table:

age	prescription	astigmatism	tear prod.	DECISION
young	myope	no	reduced	NONE
young	myope	no	normal	SOFT
young	myope	yes	reduced	NONE
young	myope	yes	normal	HARD
young	hypermetrope	no	reduced	NONE
young	hypermetrope	no	normal	SOFT
young	hypermetrope	yes	reduced	NONE
young	hypermetrope	yes	normal	HARD
pre-presbyopic	myope	no	reduced	NONE
pre-presbyopic	myope	no	normal	SOFT
pre-presbyopic	myope	yes	reduced	NONE
pre-presbyopic	myope	yes	normal	HARD
pre-presbyopic	hypermetrope	no	reduced	NONE
pre-presbyopic	hypermetrope	no	normal	SOFT
pre-presbyopic	hypermetrope	yes	reduced	NONE
pre-presbyopic	hypermetrope	yes	normal	NONE
presbyopic	myope	no	reduced	NONE
presbyopic	myope	no	normal	NONE
presbyopic	myope	yes	reduced	NONE
presbyopic	myope	yes	normal	HARD
presbyopic	hypermetrope	no	reduced	NONE
presbyopic	hypermetrope	no	normal	SOFT
presbyopic	hypermetrope	yes	reduced	NONE
presbyopic	hypermetrope	yes	normal	NONE

(the least compact form - each row is a separate case)

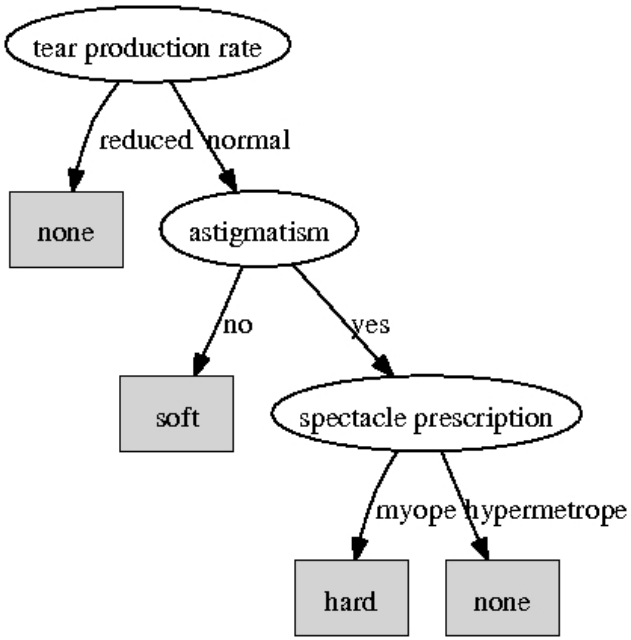
Knowledge in the form of decision rules

Example of the first few decision rules automatically generated by so-called "covering algorithm" for the mentioned problem)

- IF tear production rate = reduced THEN recommendation = NONE
- IF age = young AND astigmatic = no AND tear production rate = normal THEN recommendation = SOFT
- IF age = presbyopic AND astigmatic = no AND tear production rate = normal THEN recommendation = SOFT
- IF age = presbyopic AND spectacle prescription = myope AND astigmatic = no THEN recommendation = NONE

Decision rules are convenient for analysis and are much more compact than decision table.
The covering algorithm in iterations greedily covers maximum possible number of uncovered cases until some stop condition.

Knowledge in the form of decision tree



It is much more compact than decision table. (notice: it represents the whole decision table except 2 cases)

ID3 algorithm for decision tree generation

In short

1. an attribute is selected according to some criteria
2. branches are created for different values of the attribute
3. 1 i 2 are repeated until the leaves are "almost" pure (only 1 category)

Note: the more iterations the higher danger of overfitting

Criteria of selecting the attribute for branching with regard to the following:

- high classification accuracy
- simplicity of the tree

(notice the conflict of interests above)

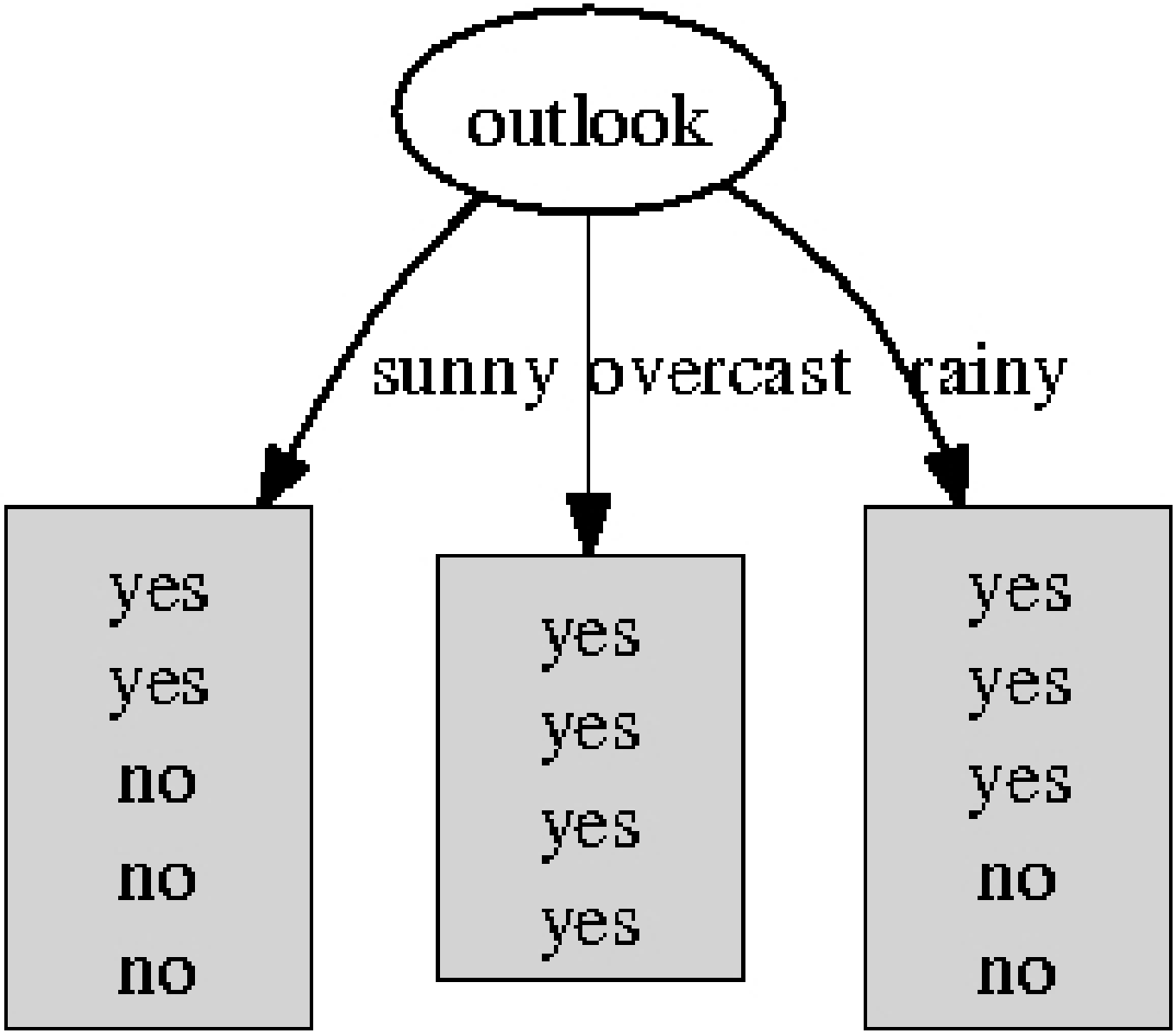
Decision tree generation - example

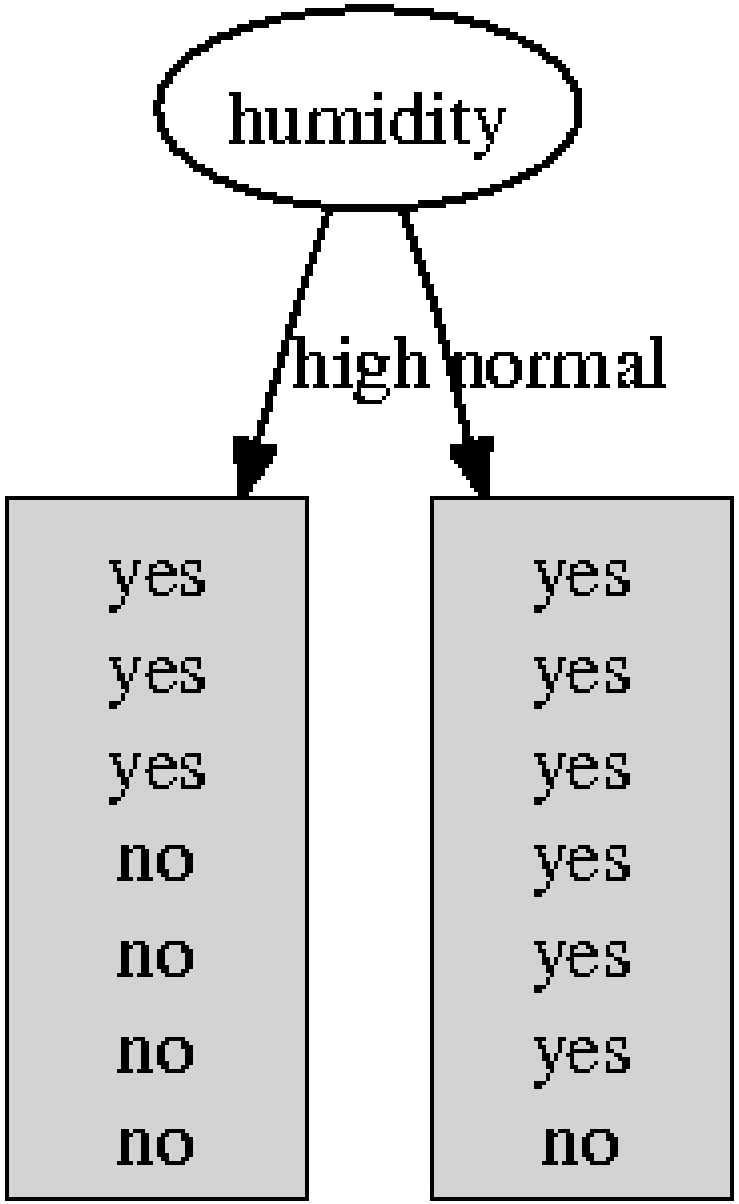
"Outdoor game":

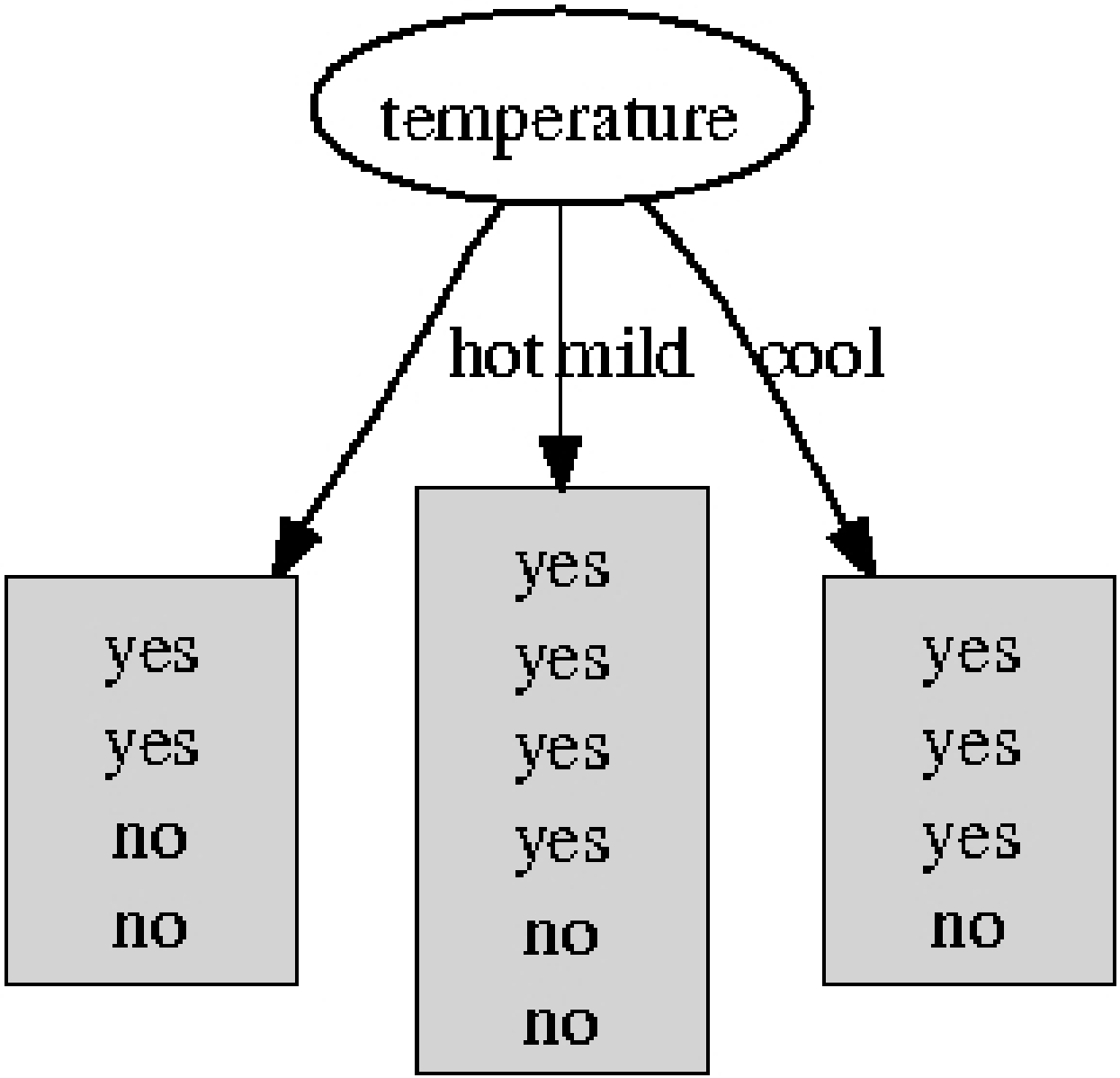
outlook	temperature	humidity	windy	PLAY?
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

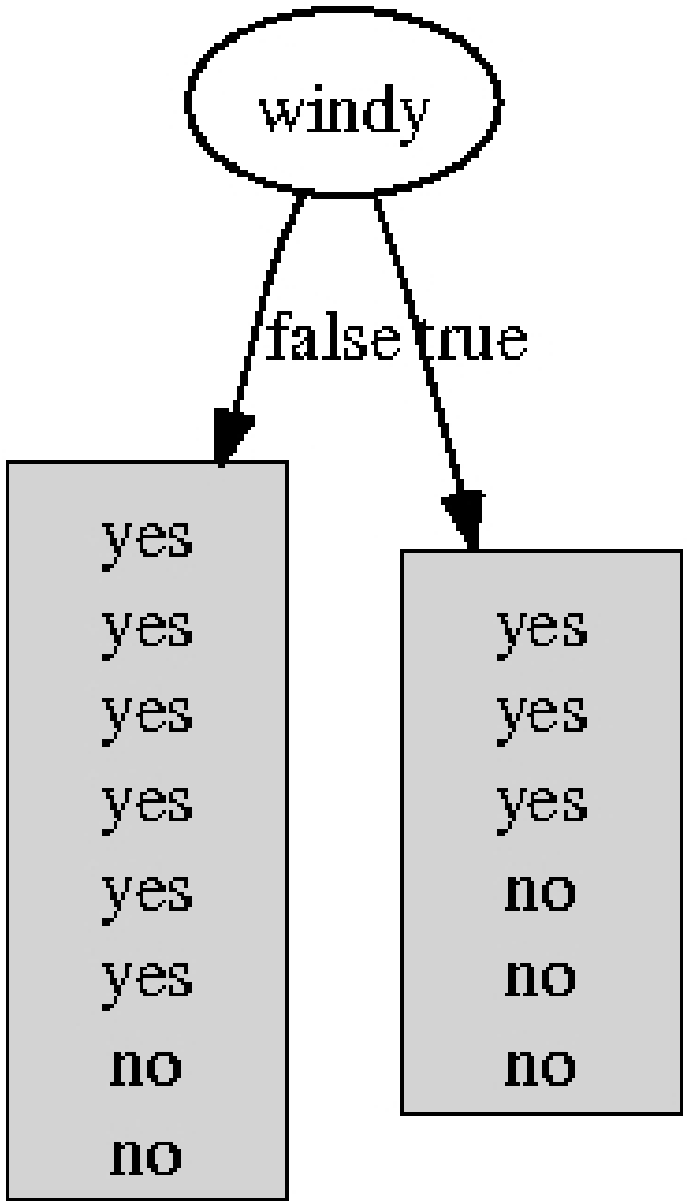
Decision tree generation - example, cont.

We have 4 attributes: outlook, temperature, humidity and wind.
Which is the best?


outlook
sunny
overcast
rainy
yes
yes
no
no
no
yes
yes
yes
yes
yes
yes
yes
no
no


humidity
high
normal
yes
yes
yes
no
no
no
no
yes
yes
yes
yes
yes
yes
no


temperature
hot
mild
cool
yes
yes
no
no
yes
yes
yes
yes
no
no
yes
yes
yes
no


windy
false
true
yes
yes
yes
yes
yes
yes
no
no
yes
yes
yes
no
no
no

How to chose the attribute to split

Algorithm ID3

Intuitively: the attribute is better if it "better distinguishes" the categories (ideally: each leaf contains cases from one category)

More precisely, we can introduce some measure of "quality of split" with each possible attribute and chose that for which this measure is best.

There are many possible ideas:

- fractions of categories in leaves
- information entropy[1]
- information gain (connected with the concept of information entropy)

Which attributes are good in our example?

[1] Introduced by Claude Shannon, the founder of "information theory" in the middle of XX. century

Entropy of information

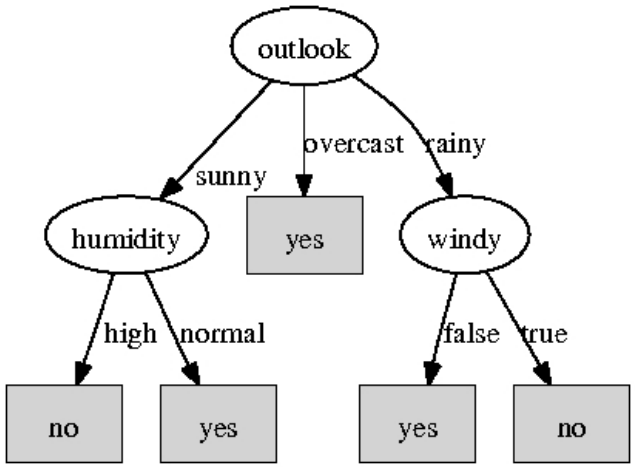
The concept is inspired by the concept of entropy in thermodynamics (where it is a measure of the degree of unorderedness/chaos of the system)
There is given a distribution of a discrete random variable X:
$P(X) = (p_1, ..., p_n)$
$(p_i = P(X = i))$
Entropy is defined as:

$$-\sum_{i=1}^{n} log_2(p_i) \cdot p_i$$

Entropy is maximum if all the probabilities are equal, and minimum (0) if there is no randomness ($p_i = 1$ for one particular i and 0 for the others). Thus, it can be viewed as a measure of the "degree of surprise" (or "chaos") in randomness.
Entropy has many interesting properties.

The resulting tree

After several steps, we obtain the following decision tree:



outlook	temp.	hum.	win.	?
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Improved Decision tree algorithms

One of the most commonly used algorithms is C4.5 that is publicly available.

It is an extension and improvement of the idea of the simple ID3 scheme.

It also includes many additional improvements such as: adaptation to numerical attributes, missing values, noisy data and "tree pruning", that automatically simplifies the resulting tree in order to avoid **overtraining**.
C4.5 algorithm also has its commercial version (C5.2 and higher), that is even more elaborated and has better performance.

Model Complexity and Overfitting

It is a very important concept in machine learning. The more complex the model (i.e. it contains more details, etc.) the more potential capacity it has in modeling the learned problem, but also the more prone it is to the phenomenon of **overfitting**.

Overfitting means too direct adaptation of the model to the training data (similar to "learning by heart" by humans), without the ability to generalise the learned knowledge to new, unknown cases.

Overfitting, cont.

Thus, the complexity of the model should not be to high.

For example:

- in neural networks, the complexity of the model increases with the number of neurons/layers (i.e. the more hidden neurons the more complex model).
- In decision tree, the complexity increases with the number of leaves.
- In decision rules, with the number of rules, etc.

Almost all of the models have some parameters that control the complexity.

Examples: too complex models

For example:

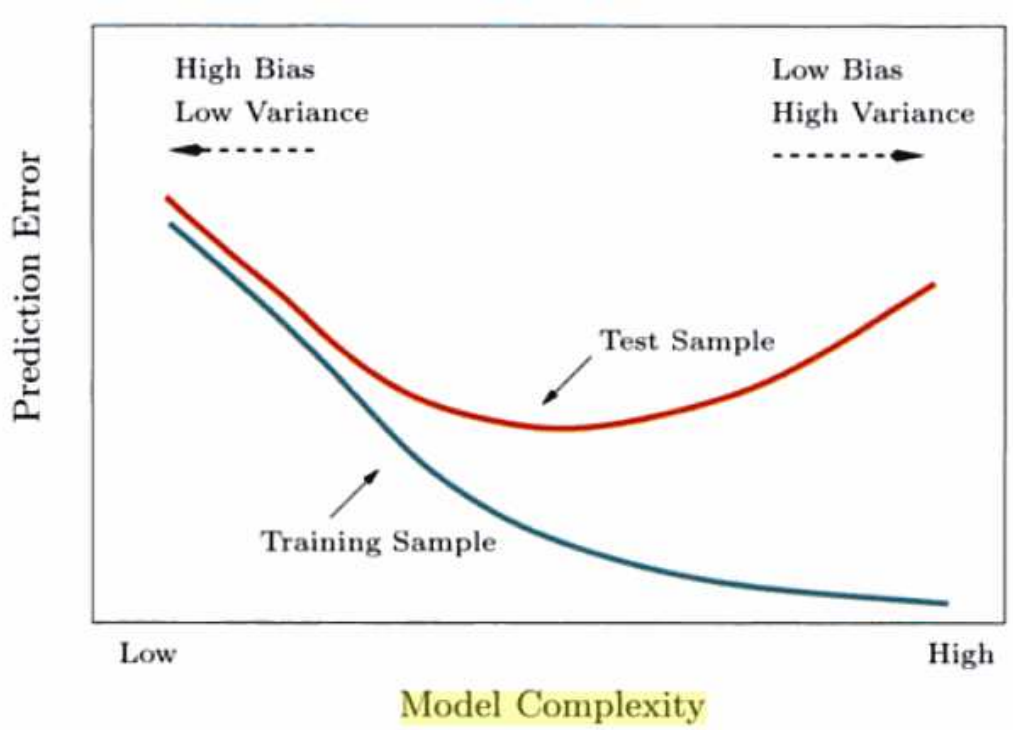
- using a 100-leaf decision tree for the "iris" problem (described before) is unnecessary
- using a multi-layer neural network with 100 neurons for modeling a XoR problem is not a good idea.

Why complexity of the model should be controlled?

Obviously, too simple model cannot learn the concept. E.g. a single neuron by no means can learn the XoR problem.
However, too complex models are also problematic:

- they are more difficult to train
- they can fit too perfectly to the training data (overfitting). Overfitting means that the model fits exactly to the training set without ability to generalise. I.e. it achieves perfect performance on the data on which it was trained, however does poorly on new, unseen examples. This is similar to learning data "by heart" without observing any general rules.

Dependence between model complexity and training/test error: **overfitting**



Overfitting is visible on the rightmost part of the graph (too complex model). As can be seen, the best balance (minimum test error) can be found for "middle" complexity. (statisticians call it "bias vs variance balance") (Hastie, Tibshirani "Elements of Statistical Learning", p. 194)

Model Selection and Assessment

Actually, the selection of the appropriate model complexity is not the only task to be solved. There are two important tasks:

- model selection (choosing appropriate model and its complexity level)
- model assessment (predicting: how well will it perform on new, unseen examples?)

If we measure the performance only on the training data it is **overestimated** (another view on the overfitting problem)

How to avoid overestimation of the model performance? (equivalently: avoid overfitting)

If there is enough labelled data (training data):
The best is to divide it into **three different subsets**:

1. train (for teaching the model on data)
2. validation (model selection and complexity control)
3. test (kept only for final assessment of future generalisation ability of the model)

No single rule for proportions, but can be 50%, 25%, 25%, respectively

if not enough data:

- **cross-validation**
- leave-one-out
- bootstrap

cross-validation is most popular

Cross-Validation

Makes it possible to achieve 2 seemingly conflicting goals:

- use the whole data for training (in some way)
- avoid assessment of the errors on the training examples

Randomly split the data into N non-overlapping parts. Repeat N times (once for each part): take i-th part as the testing set (to compute the error) and the remaining N-1 parts as training parts. Average the error over N iterations.

Quite often N=10 (10-fold cross-validation).

Stratification

The proportions of the cases in the splitted data are kept similar to those in the original (whole) dataset.

Other techniques

"leave-one-out" is a particular case of "cross-validation". N is the number of all cases in the training data.

- validation sets are one-element
- this technique is computationally intensive.
- its result is deterministic (cross validation is not)
- the sets are not stratified (obviously)

Questions/Problems:

- black box model
- knowledge representation
- decision rules & covering algorithm (idea)
- decision trees
- model complexity
- model selection and assessment
- overfitting and overcoming it
- training/testing/validation sets
- cross validation

Thank you for attention