DATA11002

Introduction to Machine Learning

Lecturer: Teemu Roos
TAs: Ville Hyvönen and Janne Leppä-aho

Department of Computer Science
University of Helsinki

(based in part on material by Patrik Hoyer and Jyrki Kivinen)

November 2nd–December 15th 2017

Ingredients of machine learning

Where we are, and what's happening next

- Today, we'll take another look into some basic components of a machine learning scenario, such as task, data, model, algorithm.
- Then we look at the single most important topic of this course.
- **For next time, read the textbook until p. 33.**

Summary of the setting

- *Task* is what an end user actually wants to do.
- *Model* is a (hopefully good) solution to the task.
- *Data* consists of objects in the domain the user is interested in, with perhaps some additional information attached.
- *Machine learning algorithm* produces a model based on data.
- *Features* are how we represent the objects in the domain.

Task

- Task is an actual data processing problem some end user needs to solve.
- Examples were given in lecture 1 (image recognition, fraud detection, ranking web pages, collaborative filtering, . . .)
- Typically, a task involves getting some input and then producing the appropriate output.
- For example, in hand-written digit recognition, the input is a pixel matrix representing an image of a digit, and the output is one of the labels '0', . . . , '9'.
- Machine learning is a way to find a solution (basically, an algorithm) for this data processing problem when it's too complicated or poorly understood for a programmer (or an application specialist) to figure it out.

Supervised learning

Many common tasks belong to the area of *supervised learning* where we neeed to produce some target value (often called *label*):

- binary classification: divide inputs into two categories
  - *Example:* classify e-mail messages into spam and non-spam
- multiclass classification: more than two categories
  - *Example:* classify an image of a digit into one of the classes '0', . . . , '9'
- multilabel classification: multiple classes, of which more than one may match simultaneously
  - *Example:* classify news stories based on their topics
- regression: output is a real-valued
  - *Example:* predict the value of a house based on its size, location etc.

Unsupervised learning

In unsupervised learning, there is no specific target value of interest.
Examples of unsupervised learning tasks:

- *clustering:* partition the given set of data into *clusters* so that elements that belong to same cluster are similar (in terms of some given similarity measure)

- *association rules:* for example, given shopping cart contents of different customers, find product combinations that are often bought together

- *dimensionality reduction:* if the data is high-dimensional (each data point is described by a large number of variables), find an alternative lower-dimensional representation that retains as much of the structure as possible

Semisupervised learning

Unsupervised learning can be used as pre-processing to help supervised learning

- for example, classifying images
- Internet has as many non-classified images as we could ever want
- less easy to *label* the data (assign the correct class to each image)
- solution: use the unlabelled images to find a low-dimensional representation, then use a smaller set of labelled images to solve the hopefully easier lower-dimensional classification problem

Predictive vs. descriptive model

- **Predictive model:** our goal is generalization, i.e., predict outcomes in future data
- **Descriptive model:** no guarantees about generalizaton to future data, we want to understand the data (a.k.a. exploratory analysis, data mining)
- Distinction between supervised vs. unsupervised learning is whether the target values are available to the learning algorithm: both can be either predictive or descriptive
- Keep in mind: *Look at the data!* Many a f**k up have been caused by blindly fitting models to data without slowing down and taking a closer look.

NOW COMES THE SINGLE MOST IMPORTANT TOPIC OF THIS COURSE

Evaluating performance on a task (Sec. 2.2)

- When we apply machine learning techniques, we have available some *training data* which we use to come up with a good model
- However, often we care more about *generalisation*: performance on future *unseen* data, *not* the training set
- Performance on training data can be much better than performance on future *test* data:
  - choosing a model that performs best on the training data can favor a model that was good by chance (it got "lucky")
  - the more models there are to choose from, and the less the training data, the worse this gets
  - performance on unseen test data can be much worse
- To get an unbiased estimate of performance on unseen data, one can withhold part of the training data and use it as *test set* – but only after choosing a model; why?

Multiple testing (multiple comparison)

Example:

- A number of investment advisors who are predicting whether the market will rise or fall in the next day.
- No advisor is actually any better than a coin-flip
- We have the record of 50 advisors for 10 consecutive days.
- Probability that a specific advisor will be correct at least 8 days out of 10:

$$\frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} \approx 0.0547 \tag{1}$$

- Probability that at least one of them gets at least 8 correct guesses:

$$1 - (1 - 0.0547)^{50} \approx 0.9399 \tag{2}$$

- The moral of the story: If you are comparing a number of random predictors, it is likely that *some* will have very good empirical performance *even if they are all quite random.*

- While the training set performance is related to generalization, one should not expect similar test set performance unless one tests the model on a fresh dataset *after selection*

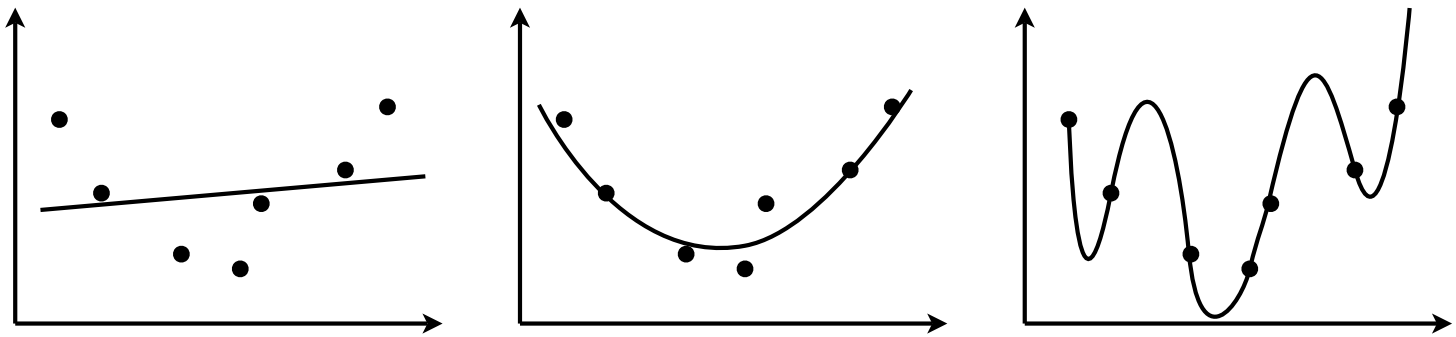
- The bigger the set of models to choose from, the worse it gets.

- This issue is fundamental in machine learning:

    1. A tempting approach is to evaluate the performance of each model on the training set and to choose the best (a.k.a. empiricial risk minimization).
    2. However, this is usually not the best approach. Can you think of a better one? (It's *very* tough.)

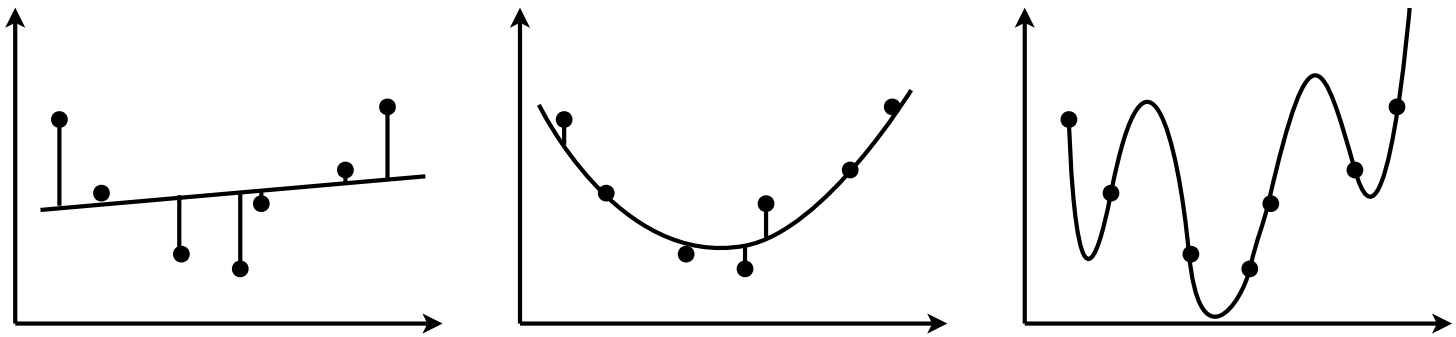
Analogous problem: Curve fitting

- Which of the following curves 'fits best' to the data?



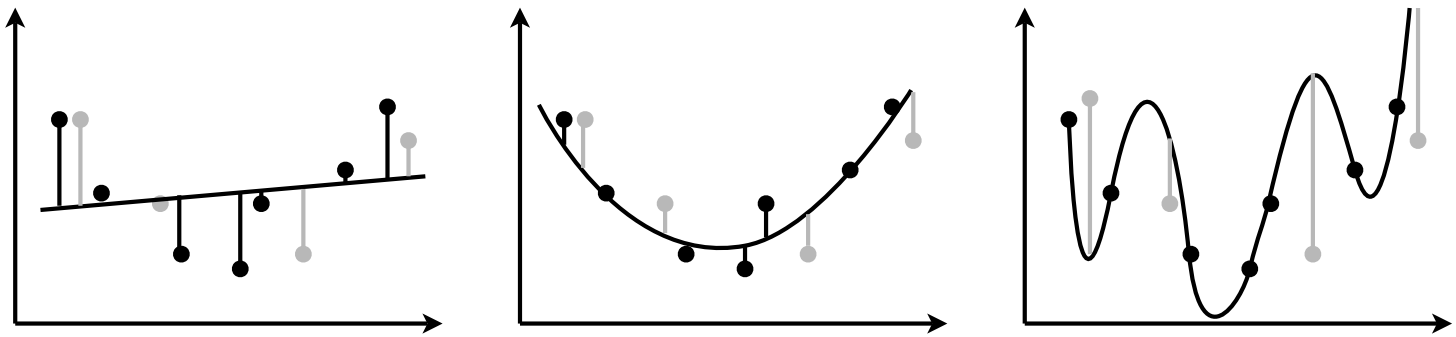
Analogous problem: Curve fitting

- Which of the following curves 'fits best' to the data?



Analogous problem: Curve fitting

- Which of the following curves 'fits best' to the data?



'underfit' 'overfit'

- The more flexible the curve...
  - ...the better you can make it fit your data...
  - ...but the more likely it is to overfit

$\Rightarrow$ ...so you need to be careful to strive for both model simplicity and for good fit to data!

Estimating generalization performance

- To get an unbiased estimate of the generalization performance, test it on data that *has not been used in any way to guide the search for the best model*. (The test data should be 'locked in a vault'. No peeking!)
- But again, if you are testing several models, beware that the best/worst results may not be good estimates of the generalization error of those models. (Don't fall into that trap, again!)

Model complexity in regression

- E.g. What degree polynomial to fit to the data?
  - Note that low degree polynomials are *special cases* of higher-degree polynomials, with some coefficients set to zero (i.e. the model classes are 'nested'), e.g. second-degree polynomial as special case of fourth-degree:

$$y = c_0 + c_1 \cdot x + c_2 \cdot x^2 + 0 \cdot x^3 + 0 \cdot x^4 \tag{3}$$

  $\Rightarrow$ a high-degree polynomial is guaranteed to fit the data *at least* as well as a low-degree one!

  $\Rightarrow$ a high-degree polynomial will *never* look worse (on the training data) but as we saw, it can yield much worse predictions

- Main learning objective of this course: Understand why overfitting is a problem and how it can be avoided