Learning from Examples
Part 1

Margrit Betke

CS 440/640

Fall 2019

Based on Russell and Norvig, 3rd edition, Sections 1, 2, and 4.

Forms of Learning

AI system "learns" if it improves its performance *based on observations & feedback from its environment*

Unsupervised learning (=clustering):

Input: vector of attributes. No explicit feedback.

Supervised learning:

Input: vector of attributes. Feedback = output of continuous or discrete value(s) = labels of input examples.

Reinforcement Learning:

Actions are rewarded or punished.

In 440/640: Supervised Learning

Training set = N example input-output pairs

(x_1,y_1), (x_2,y_2), ..., (x_N,y_N)

where each y_j was generated by an unknown function f, such that $f(x) = y$. Function f needs to be learned.

- The AI system finds a function h that approximates f. For example, the AI system trains a neural net that computes $h(x_j)=y_j$ for all examples in the training set.
- There are no guarantees that new inputs $h(x_{new}) \approx f(x_{new})$.
- To measure accuracy (Is $h \approx f$?), we use a test set of labeled examples = input-output pairs (≠ training set!):

A neural net is trained well if $h(x_{test}) \approx y_{test}$ for all test example pairs (x_{test}, y_{test}).

Classification versus Regression

Depending on the type of output, the learning problem is a

- Classification problem:

  Output values: number of classes (discrete, finite)

- Regression problem:

  Output values are numbers, e.g., tomorrow's temperature

Occham's Razor

= Law of succinctness

Which hypothesis among h_1, h_2, h_3 ... should the AI system choose?

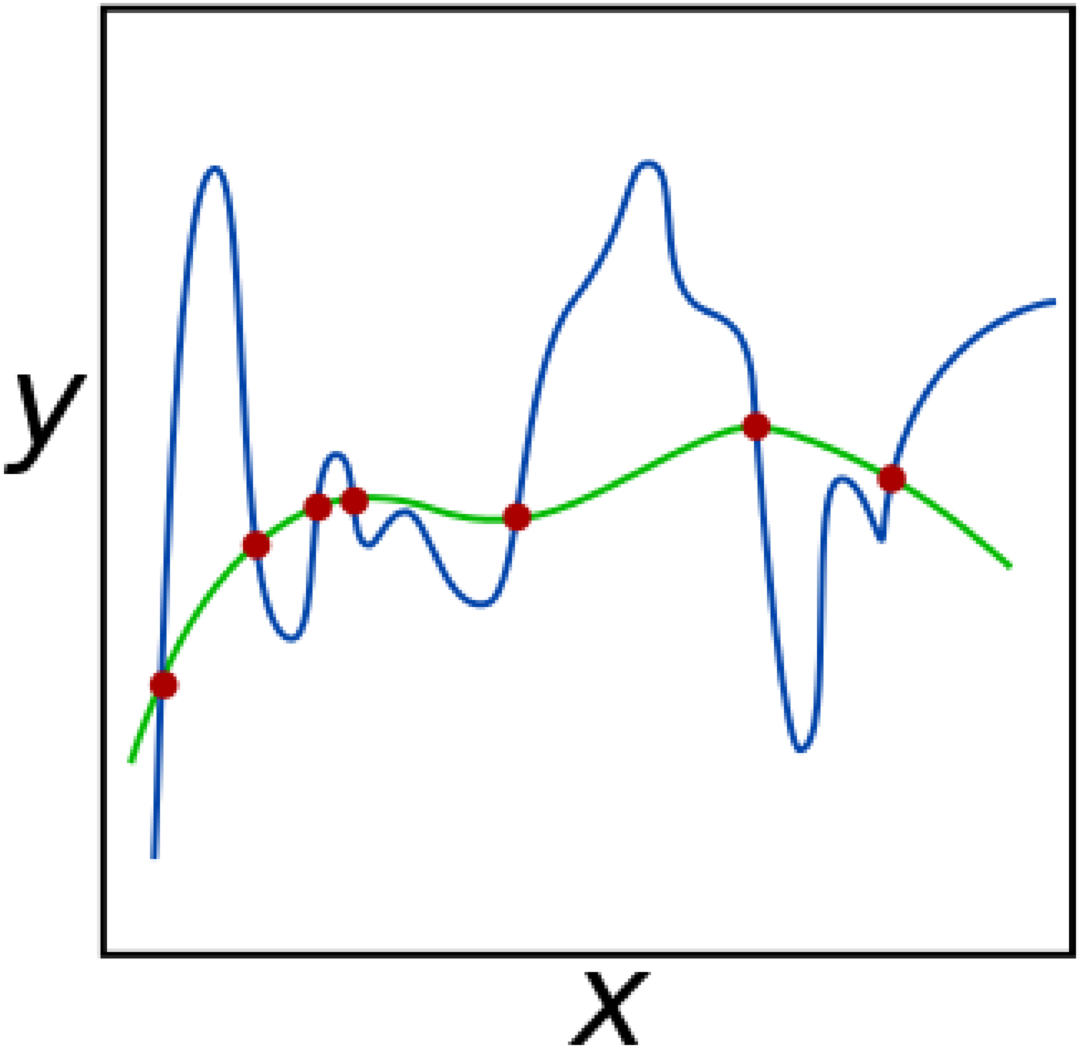
Choose the simplest hypothesis consistent with the data.

The simplest explanation will be the most plausible until evidence is presented to prove it false.

Example: Prefer a degree-1 polynomial (line) over a degree-7 polynomial

Trade-off between complex hypothesis that fit training data well and simpler hypotheses that may generalize better (and can typically be computed faster)

Occam’s Razor: Choose green over blue model for h



Source: Wikipedia

Overfitting

- Avoid choosing an excessively complex learning system= model= hypothesis=neural net h.
- h is too complex if it has too many parameters relative to the number of observations.
- A model which has been overfitted will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
- Higher-degree polynomials or complicated neural nets with many hidden layers and nodes fit the data better but may lead to overfitting.

Overfitting

- Avoid choosing an excessively complex learning system= model= hypothesis=neural net h.
- h is too complex if it has too many parameters relative to the number of observations.
- A model which has been overfit will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
- Higher-degree polynomials or complicated neural nets with many hidden layers and nodes fit the data better but may lead to overfitting.

Solutions:

- Use “wrapper” to enumerate models h according to model size (e.g., number of nodes in neural net h). Select model with smallest error.
- Feature selection: Simplify model by discarding irrelevant attributes (dimensionality reduction).
- Minimum description length: Select model with smallest number of bits required to encode program and data.

Loss Functions: SPAM Example

Loss value $L(y_{true}, y)$

= cost of misclassifying email:

A "false positive," e.g. hypothesize "non-spam" but it is truly "spam" L(spam,non-spam) = 1

Annoying but simply delete email.

A "false negative," e.g. hypothesize "spam" but it is truly "non-spam" L(non-spam, spam) = 10

Much worse, you may miss an important email.

Loss Functions

- Absolute value loss: $L_1(y_{true}, y) = |y_{true} - y|$
- Squared error loss = Euclidean loss:

$$L_2(y_{true}, y) = (y_{true} - y)^2$$

- 0/1 loss: $L_{0/1}(y_{true}, y) = 0$ if $y_{true} = y$, else 1
- Find h that minimizes the empirical loss

$$\text{EmpLoss}(h) = 1/N \sum L(y_{true,\, i}, h(x_i))$$

= mean error over a set of N examples $(x_i, y_{true,\, i})$

Cross-Validation

Holdout cross-validation =
Randomly split available (input,output) pairs into a training set to learn h and a test set to test the learned h.

k-fold cross-validation =

- Split data into k equal subsets.
- Perform k rounds of learning. Each round leaves 1/k examples out of the training set that can then be used as the test set.
- The average test set score should be a better estimate than a single score (need to keep k h's around for prediction). Typically, k=5 or 10.

Leave-one-out cross validation: k=N.