Chapter 18 (AIAMA)
Learning From Examples

Learning Agent

- An agent is **learning** if it improves its performance on future tasks after making observations about the world.

- Learning can be **online or offline**

  - Online: You learn as you observe each data; learn continuously over time

  - Offline: You are given with a set of pre-observed data and you learn from the data; learn once

Learning Agent

- **Input data:**
  - Traffic info of several days in a month
  - Data may have labels:
    - Rows are marked with “Good” and “Bad” traffic days

- **Learning algorithm**:
  - Allow agent learns from the input data to predict new days as “Good” or “Bad”

Learning types

Unsupervised: Learn patterns in the input

- Example: “Good traffic days” and “bad traffic days” learned by a taxi driver without any label to a day
- Also known as clustering
- Data do not have labels

Learning types

Supervised:

- Learn from input-output pairs.
- Data have labels (i.e., class)
- Can predict “Good” or “Bad” from traffic info of a day

Learning types

Reinforcement:

- Learn from reinforcements, rewards and punishments
- Example: lack of a tip at the end of a journey an indication that taxi driver did something wrong

Supervised Learning

- Given a **training set** of N example input–output pairs (x_1, y_1), $(x_2, y_2), \ldots (x_N, y_N)$ here each y_j was generated by an unknown function $y = f(x)$, discover a function h that approximates the true function f.

- **Hypothesis**: The function h is a hypothesis.

- **State space search**: Learning is a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set.

Supervised Learning

- **Accuracy**: To measure the accuracy of a hypothesis we give it a test set of examples that are distinct from the training set.

- **Generalization**: We say a hypothesis generalizes well if it correctly predicts the value of y for novel examples. Sometimes the function f is stochastic—it is not strictly a function of x, and what we have to learn is a conditional probability distribution, $P(Y \mid x)$.

Supervised Learning

- **Classification**: When the output y is one of a finite set of values (such as sunny, cloudy or rainy), the learning problem is called classification, and is called Boolean or binary classification if there are only two values.

- **Regression**: When y is a number (such as tomorrow's temperature), the learning problem is called regression.

  -Technically, solving a regression problem is finding a conditional expectation or average value of y, because the probability that we have found exactly the right real-valued number for y is 0.

Supervised Learning

What is the best hypothesis to choose from the hypotheses space? All are consistent (agrees with all data points)

(a) Linear

(b) 6-deg polynomial

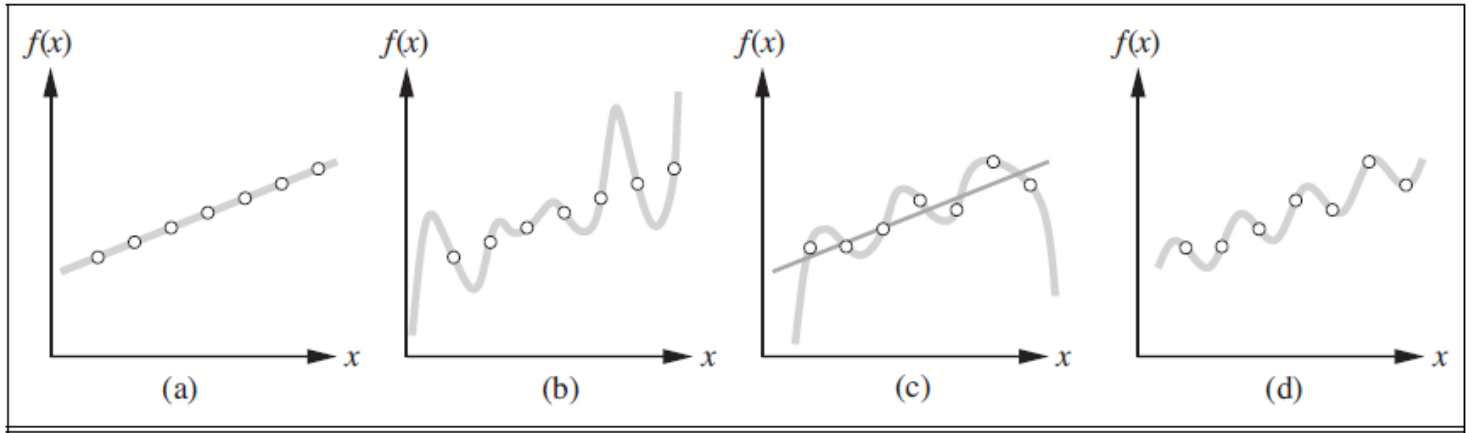
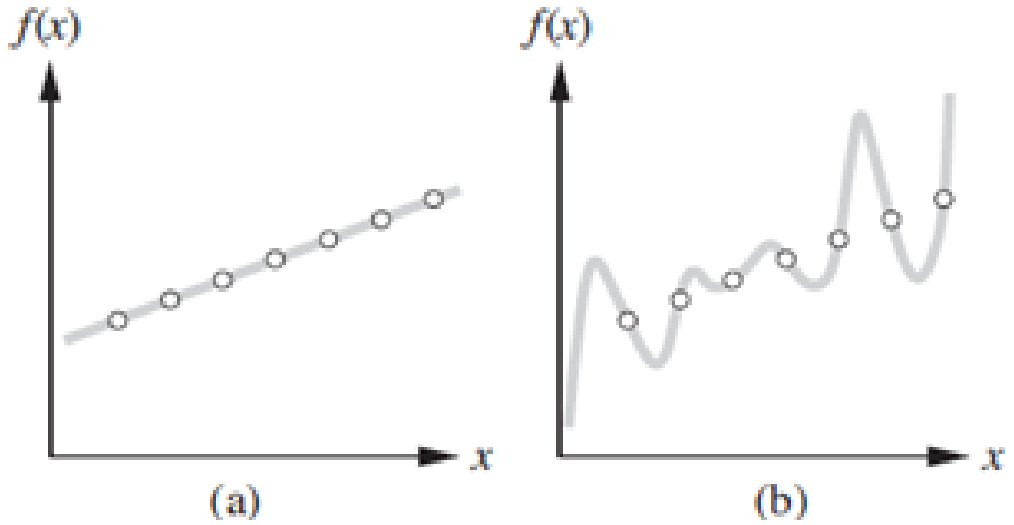


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

Supervised Learning

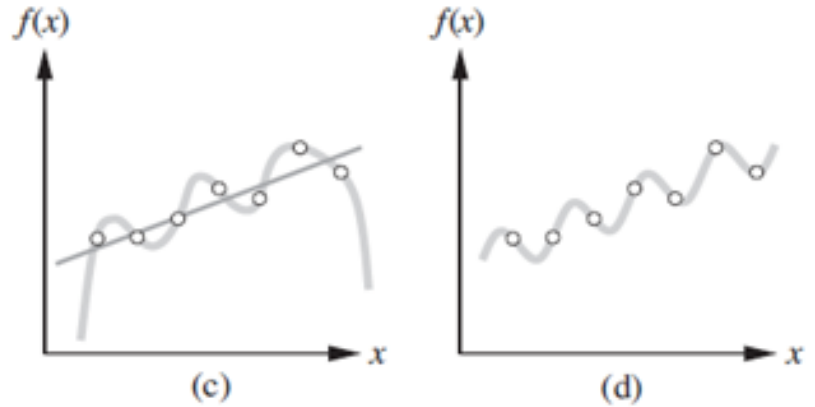
Ockham's razor: Select the simplest hypothesis consistent with the data.

- After 14th-century English philosopher William of Ockham
- Defining simplicity is not easy, but it seems clear that a degree-1 polynomial is simpler than a degree-7 polynomial, and thus (a) should be preferred to (b).



Supervised Learning

Consistency vs Generalization



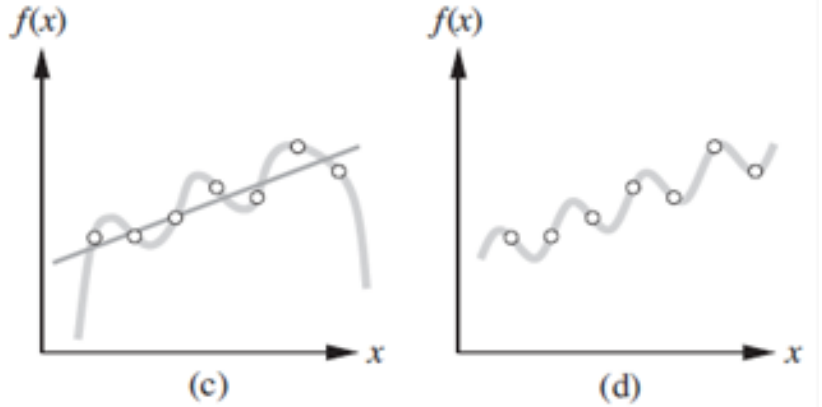
- Figure (c) shows a second data set.
- There is no consistent straight line for this data set;
- A degree-6 polynomial is an exact fit: may not find any pattern in the data and we do not expect it to generalize well.
- A straight line that is not consistent with any of the data points, but might generalize fairly well for unseen values of x, is also shown in
- *In general, there is a tradeoff between complex hypotheses that fit the training data well and simpler hypotheses that may generalize better.*

Supervised Learning

Choice of hypothesis space

- In (d) we expand the hypothesis space H to allow polynomials over both x and $\sin(x)$, and find that the data in (c) can be fitted exactly by a simple function of the form $ax + b + c\sin(x)$.

- This shows the importance of the choice of hypothesis space.



- **Realizable:** We say that a learning problem is realizable if the hypothesis space contains the true function. Unfortunately, we cannot always tell whether a given learning problem is realizable, because the true function is not known.

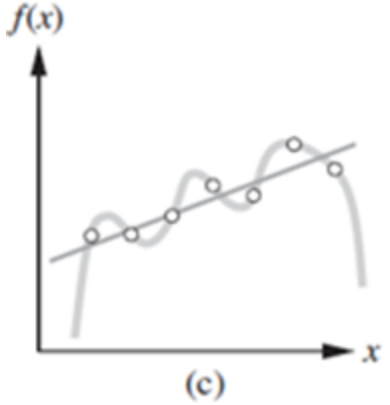
Supervised Learning

Selecting most probable hypothesis: Supervised learning can be done by choosing the hypothesis $h*$ that is most probable given the data:

$$h^* = \operatorname*{argmax}_{h \in \mathcal{H}} P(h|data) \ .$$

By Bayes' rule this is equivalent to

$$h^* = \operatorname*{argmax}_{h \in \mathcal{H}} P(data|h) \, P(h) \ .$$



- **Prior probability** $P(h)$ is high for a degree-1 or -2 polynomial, lower for a degree-7 polynomial, and especially low for degree-7 polynomials with large spikes as in Figure (c).

Supervised Learning

Expressiveness vs computational complexity: Why not let *H* be the class of all Java programs, or Turing machines? After all, every computable function can be represented by some Turing machine, and that is the best we can do. Two issues:

1. **Computation complexity of learning**: There is a tradeoff between the expressiveness of a hypothesis space and the complexity of finding a good hypothesis within that space. For example, fitting a straight line to data is an easy computation; fitting high-degree polynomials is somewhat harder; and fitting Turing machines is in general undecidable.

Supervised Learning

Expressiveness vs computational complexity: Why not let H be the class of all Java programs, or Turing machines? After all, every computable function can be represented by some Turing machine, and that is the best we can do. Two issues:

2. Simpler hypothesis easier to use: A second reason to prefer simple hypothesis spaces is that presumably we will want to use h after we have learned it, and computing $h(x)$ when h is a linear function is guaranteed to be fast, while computing an arbitrary Turing machine program is not even guaranteed to terminate. For these reasons, most work on learning has focused on simple representations.