CS 4104
APPLIED MACHINE LEARNING

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

GRADIENT DESCENT

Gradient Descent

- Gradient descent is an important general paradigm for learning.
- It is a *strategy for searching through a **large or infinite** hypothesis space* that **can be applied whenever**
  1) the hypothesis space contains **continuously parameterized hypotheses** (e.g., the parameter in a linear unit),
  2) the **error can be differentiated** with respect to these hypothesis parameters

Stochastic Gradient Descent

- The idea behind **stochastic gradient descent** is to approximate the gradient descent search by **updating parameters incrementally**, following the calculation of the error for ***each* individual example**.
- One way to view this stochastic gradient descent is to consider a **distinct error function** defined for ***each*** *individual training example* ***d*.**

The Key Difference

- In **stochastic gradient descent**, weights are updated upon examining ***each*** training example.
- Whereas in **standard gradient descent**, the error is summed over ***all*** examples before updating parameters/weights,
  - **Standard gradient descent** requires **more computation** per parameters update step.
  - **Standard gradient descent** is often used with a **larger step size** per parameters update than stochastic gradient descent.

The Key Difference

- When there are **multiple local minima** with respect to $\boldsymbol{E}(\theta)$,
  - The **stochastic gradient descent** can sometimes *avoid falling into these local minima*,
  - It is due to the reason that it uses various $\nabla E_n(\vec{\theta})$ rather than $\nabla E(\vec{\theta})$ to guide its search.

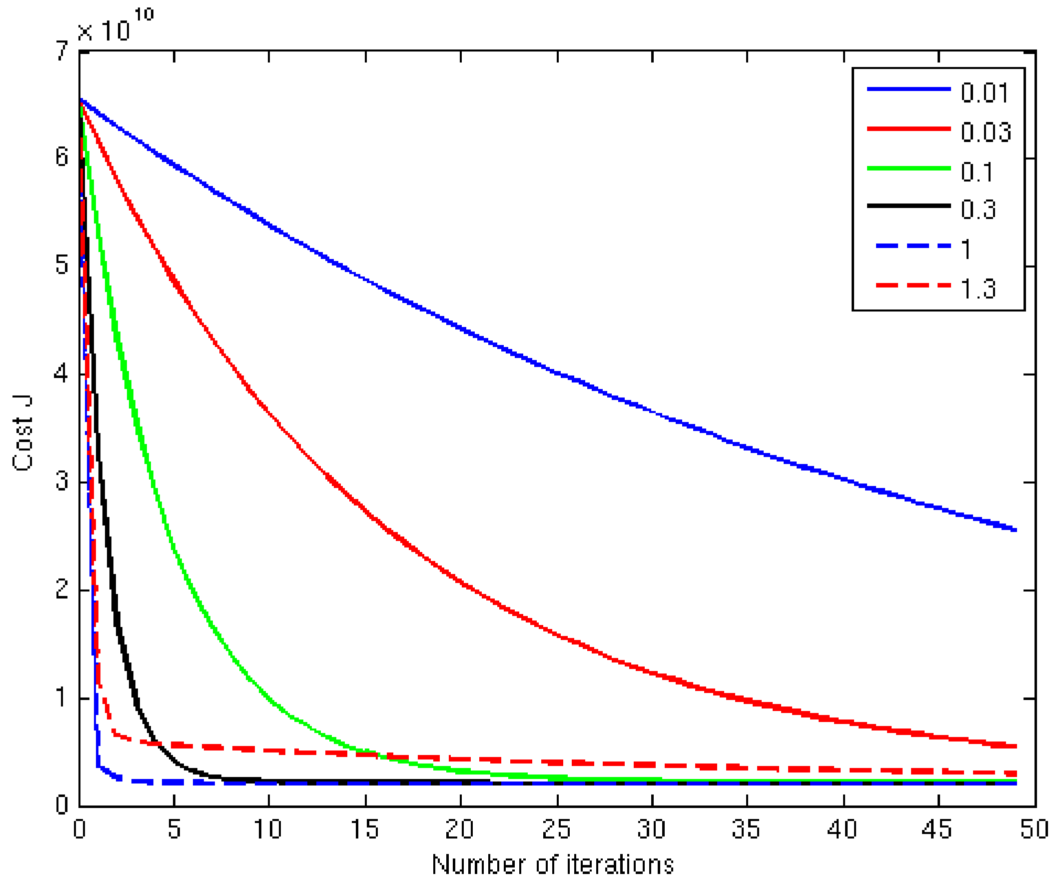
LEARNING RATE

Learning Rate

- When $\alpha = 0.01$, the cost function decreases slowly, which means **slow convergence** during gradient descent.

- While $\alpha = 1.3$ is the largest learning rate, $\boldsymbol{\alpha = 1.0}$ has a faster convergence.

- *After a certain point, increasing the learning rate will no longer increase the speed of convergence.*

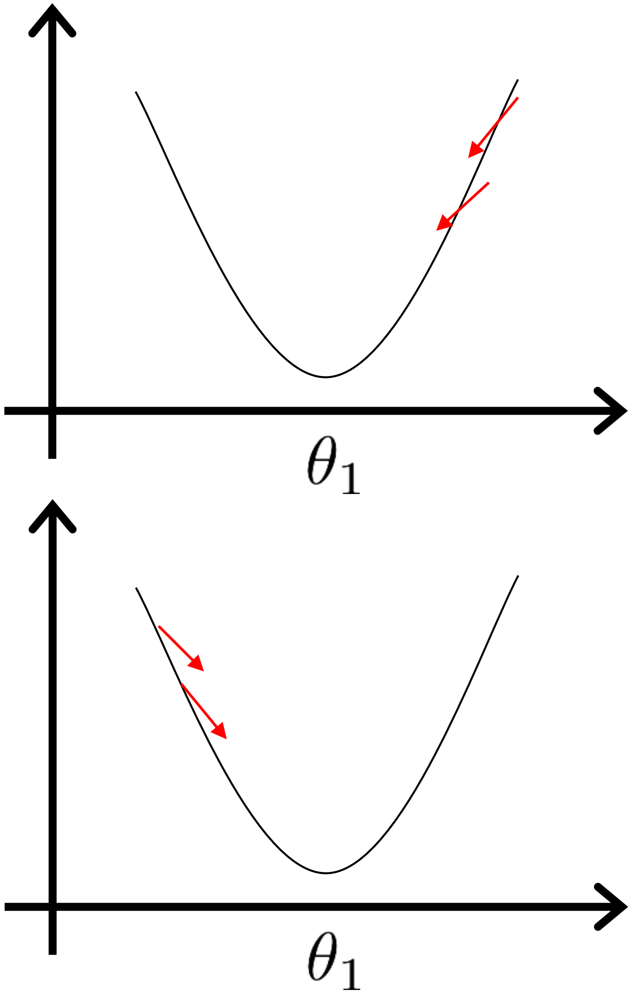


Direction

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If the result is $\theta_1 \geq 0$,
positive slop.

If the result is $\theta_1 \leq 0$,
negative slop.



DATA STANDARDIZATION

Data Standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- Consider the following pair of data points:

$$\boldsymbol{x}_i\text{: } (0.1, 20) \text{ and } \boldsymbol{x}_j\text{: } (0.9, 720)$$

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9 - 0.1)^2 + (720 - 20)^2} = 700.000457,$$

- The distance is almost completely dominated by $(720 - 20) = 700.$
- Standardize attributes: to force the attributes to have a common value range,

Data Standardization

Interval-scaled attributes:

- Their values are *real numbers following a linear scale*.
  - The difference in Age between 10 and 20 is the same as that between 40 and 50.
  - The key idea is that intervals keep the same importance through out the scale
- *Two main approaches* to standardize interval scaled attributes,
  - **Range**
  - **Z-score**

Data Standardization

Range:

- Consider *f* is an attribute

$$range(x_{if}) = \frac{x_{if} - \min(f)}{\max(f) - \min(f)},$$

- also referred as min-max normalization.

Data Standardization

Z-score:

- transforms the attribute values so that they have a mean of zero and a mean absolute deviation of 1. The mean absolute deviation of attribute *f*, denoted by s_f, is computed as follows

$$m_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \ldots + x_{nf}\right),$$

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|\right),$$

$$z(x_{if}) = \frac{x_{if} - m_f}{s_f}.$$

Data Standardization

Z-score:

- transforms the attribute values so that they have a mean of zero and a mean absolute deviation of 1. The deviation of attribute *f*, denoted by s_f, is computed as follows

$$m_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \ldots + x_{nf}\right),$$

$$s_f = \max(x_f) - \min(x_f)$$

$$z(x_{if}) = \frac{x_{if} - m_f}{s_f}.$$

Data Standardization

Ratio-scaled attributes:

- Numeric attributes, but unlike interval-scaled attributes, *their scales are exponential*,
- For example, the total amount of microorganisms that evolve in a time *t* is approximately given by

  $$Ae^{Bt},$$

  - where *A* and *B* are some positive constants.
- Do *log transform*:

  $$\log(x_{if})$$

  - Then treat it as an interval-scaled attribute

OVERFITTING

Overfitting/Underfitting

- Overfitting: h more complex than f ("h too complex")
  - E.g. tree with too many nodes
- Underfitting: h less complex than f ("h too simple")
  - E.g. tree with too few nodes

Overfitting and Underfitting:
Result in selection of a hypothesis that is better on the training data, but worse on test data than best hypothesis

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

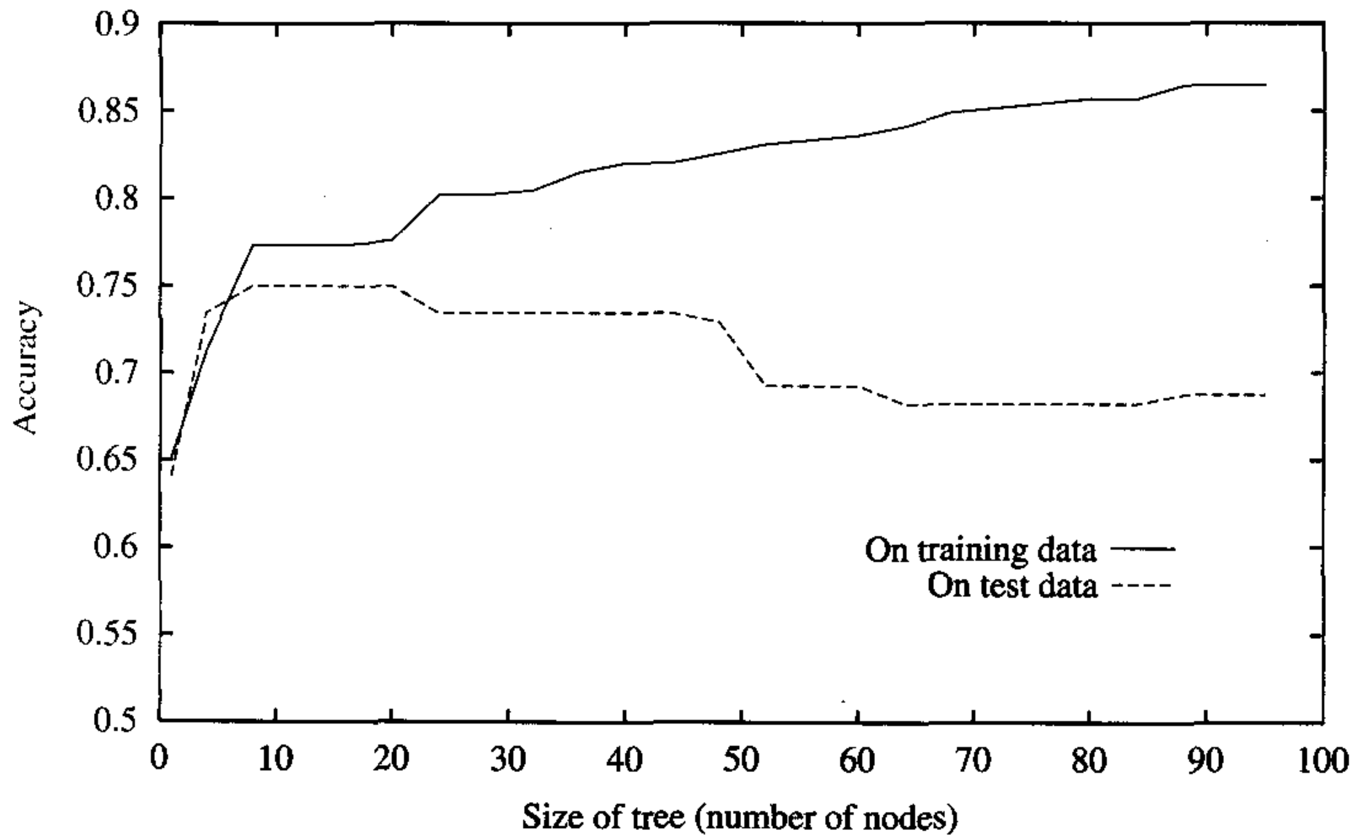
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

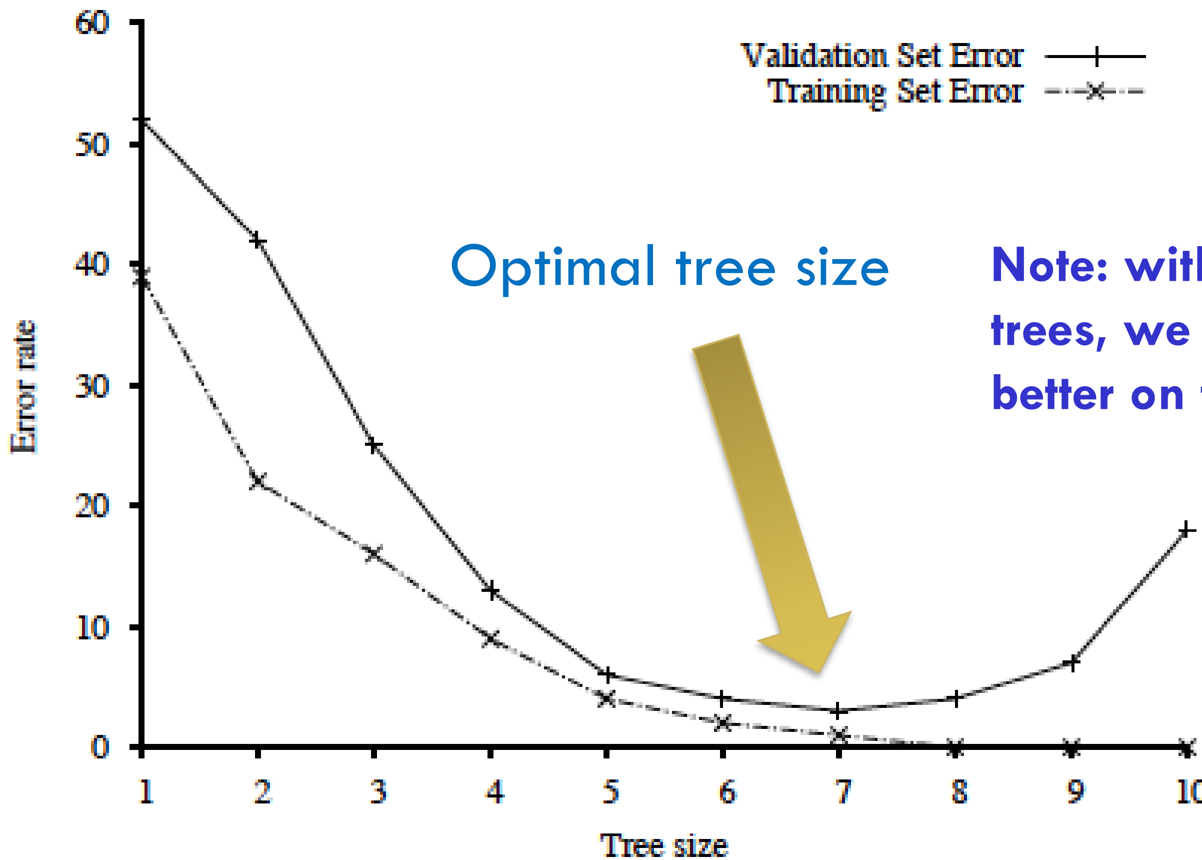
and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting



Overfitting

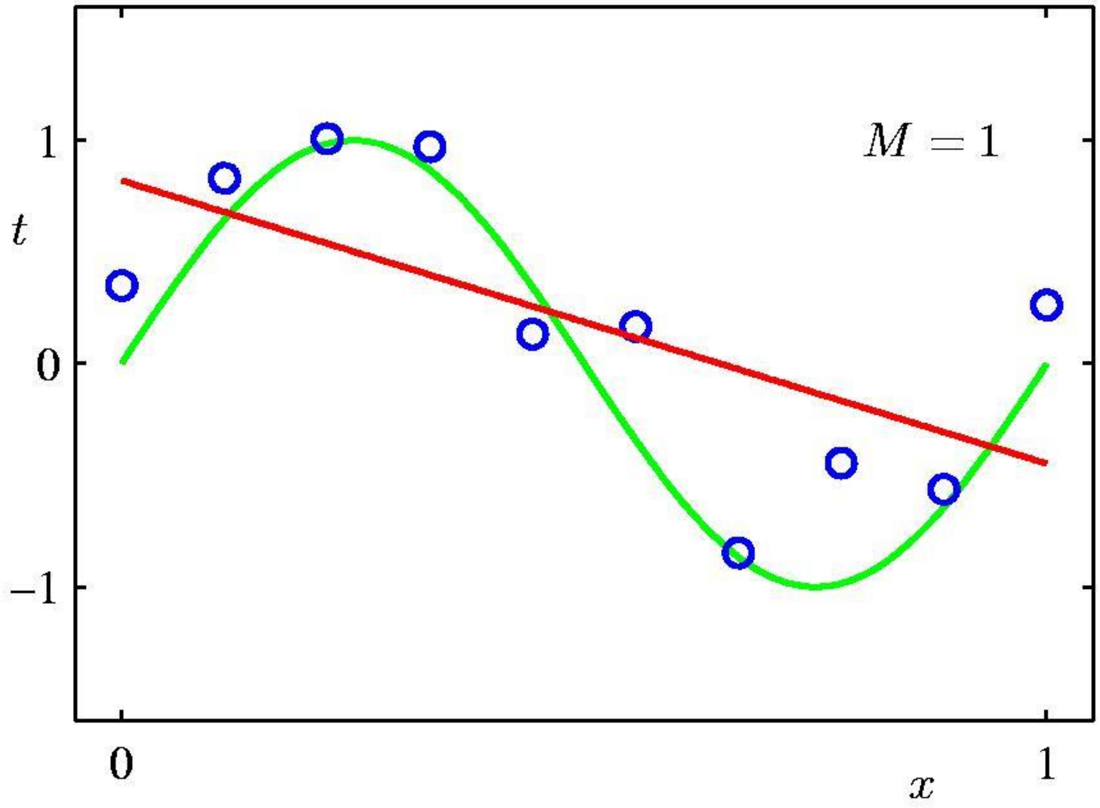


Optimal tree size

Note: with larger and larger trees, we just do better and better on the training set!

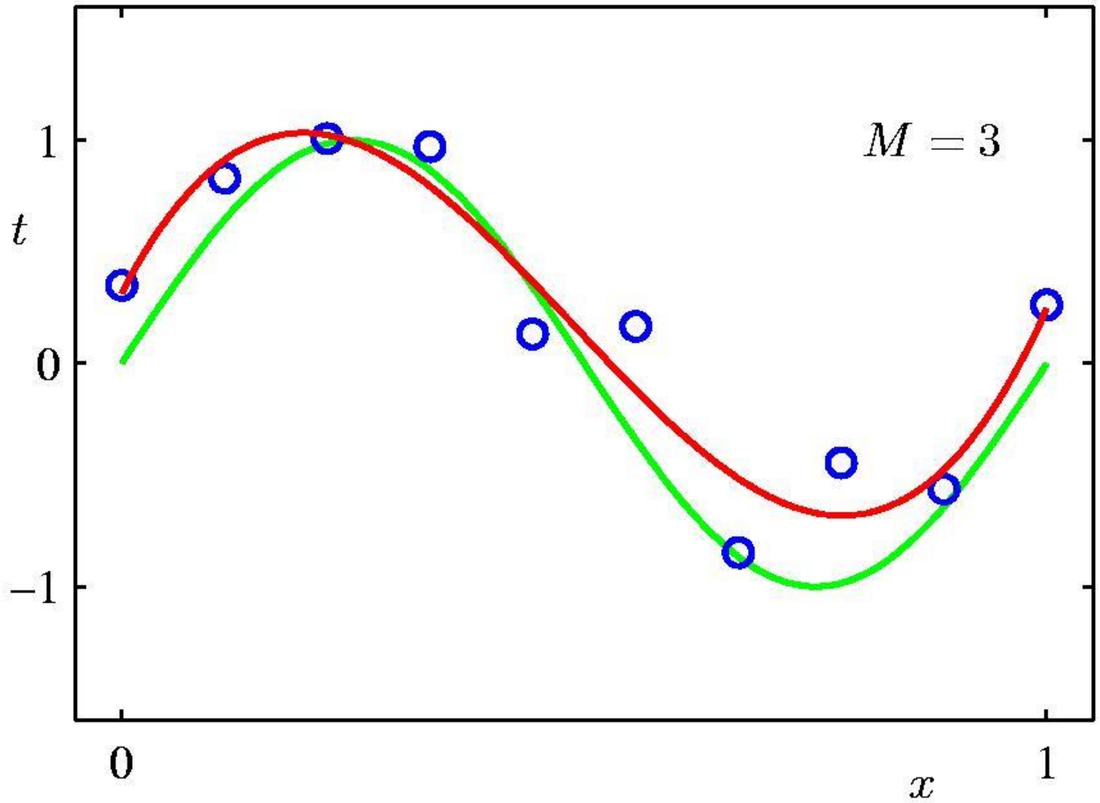
FUNCTION COMPLEXITY & OVERFITTING

1^{st} Order Polynomial



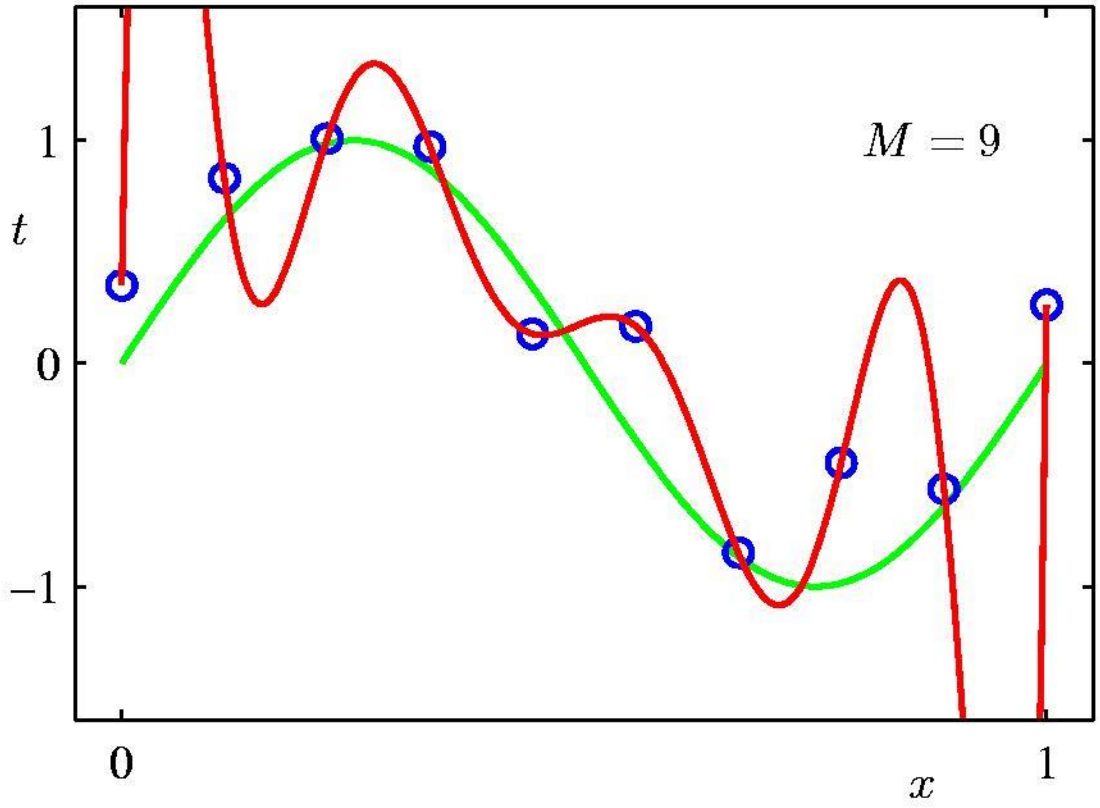
M is the function complexity.

3rd Order Polynomial



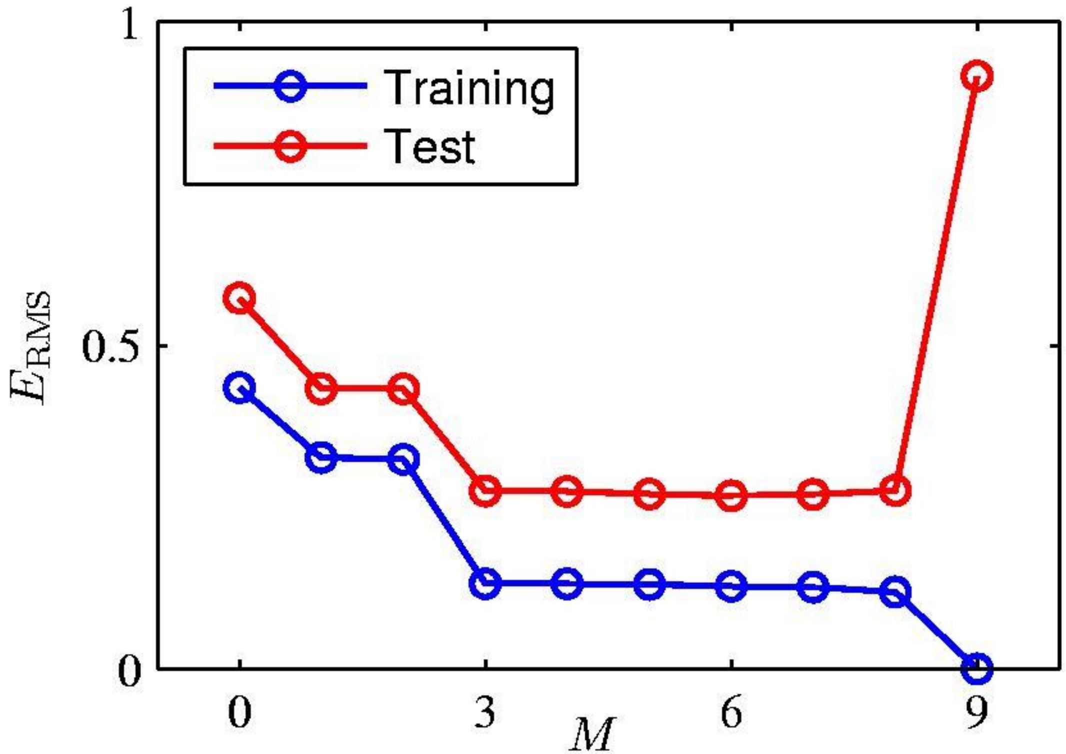
M is the function complexity.

9th Order Polynomial



M is the function complexity.

Overfitting



Ockham's razor: prefer the simplest hypothesis consistent with data

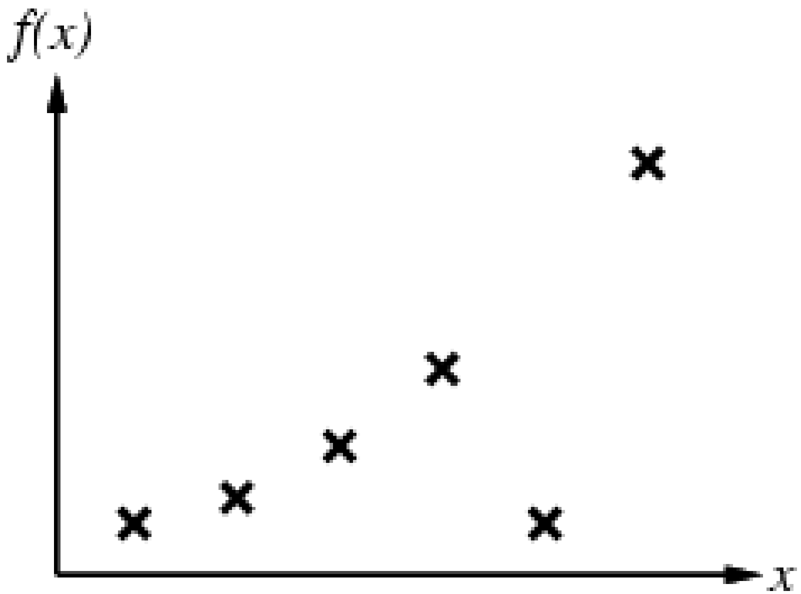
Root-Mean-Square (RMS) Error: E_{RMS}

M is the function complexity.

MORE ABOUT MODEL COMPLEXITY

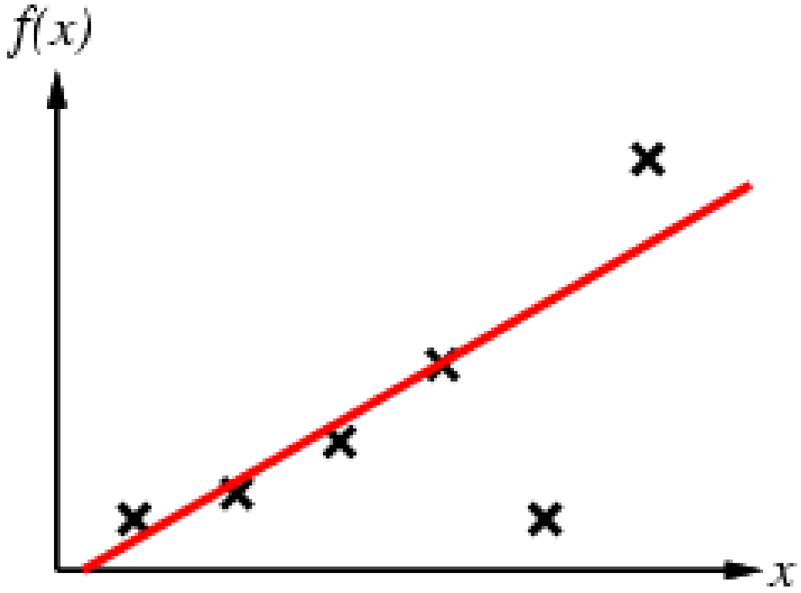
Example 1

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)



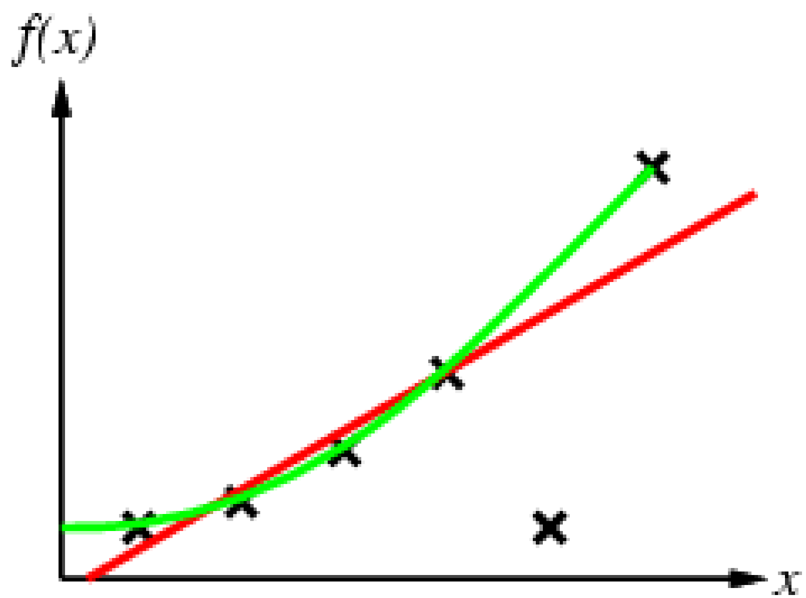
Example 1

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)



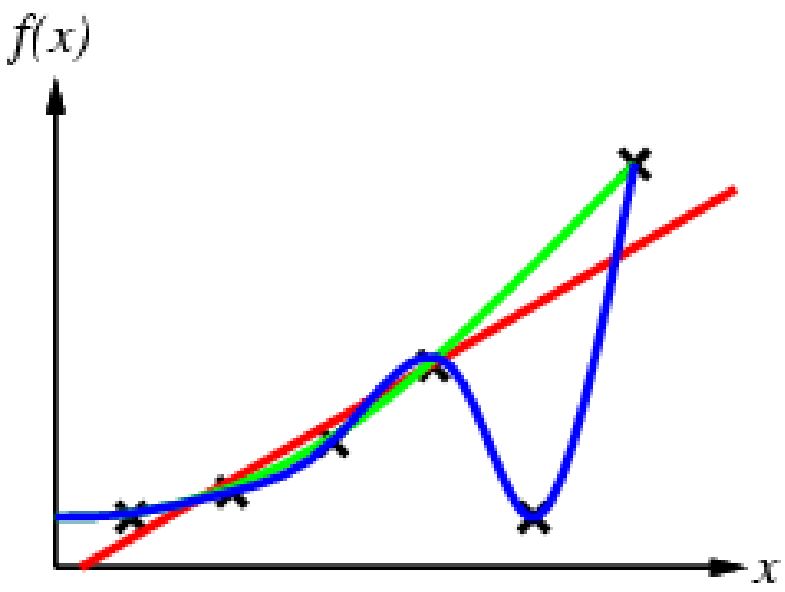
Example 1

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)



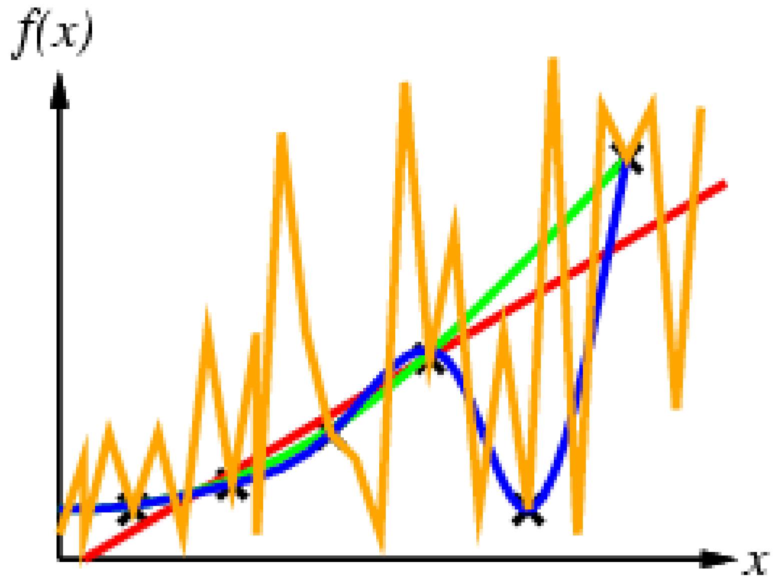
Example 1

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)



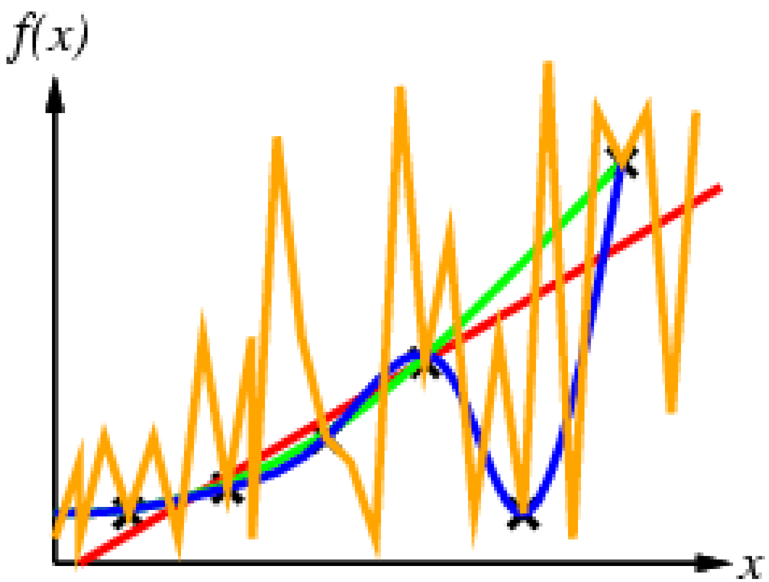
Example 1

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)



Example 1

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)



Ockham's razor: prefer the simplest hypothesis consistent with data

Example 2

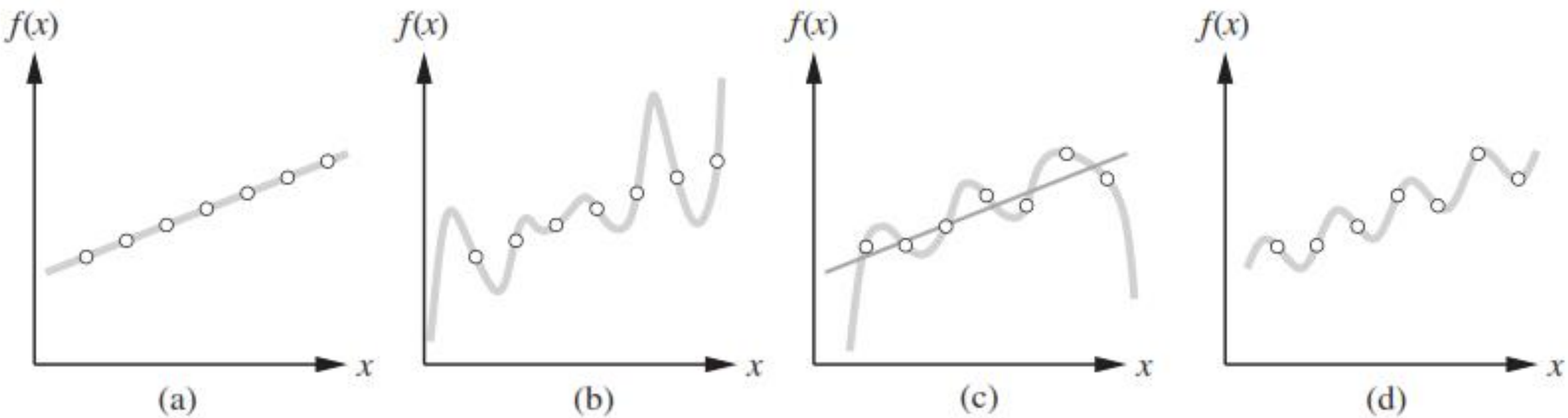


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data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

Ockham's razor: prefer the simplest hypothesis consistent with data

Acknowledgement

Tom Mitchel, Russel & Norvig, Andrew Ng, Alpydin & Ch. Eick.