Lecture 7: Algorithm Independent Principles

Model Class Complexity, Bias-Variance Analysis, Validation Protocols

Machine Learning, Summer Term 2019

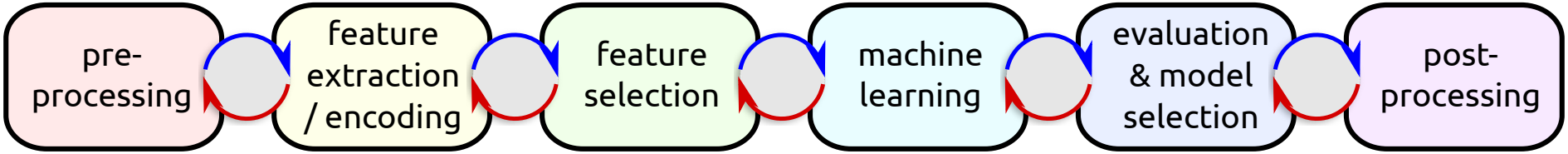
Michael Tangermann Frank Hutter Marius Lindauer

University of Freiburg

Lecture Overview

Lecture Overview

Reminder: ML Design Cycle



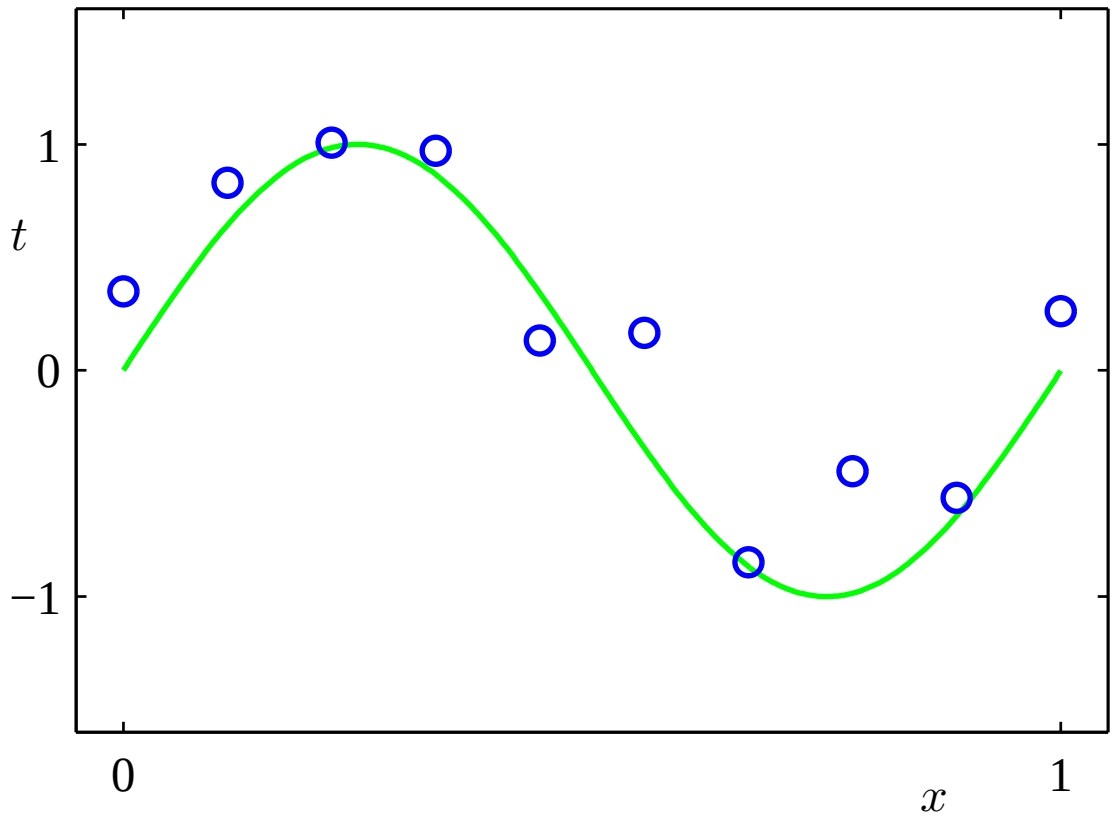
Today, we'll talk about algorithm-independent principles that concern several of the steps in the design cycle.

Notation

- $L(\cdot, \cdot)$ - loss function
- $h(\mathbf{x}; \mathbf{w})$ - our hypothesis with parameters $\mathbf{w}$ for input $\mathbf{x}$ (our model, was called f in previous lectures)
- $\hat{h}$ - estimated optimal hypothesis
- h^* - optimal (generating) hypothesis
- $\mathcal{H}$ - hypothesis set under consideration

Example: Polynomial Curve Fitting

Consider a curve fitting example where we are given a labeled data set of N examples $\langle (x_i, y_i) \rangle_{i=1}^N$. Labels are generated from the target function $\sin(2\pi x)$ plus a bit of Gaussian noise.



Example: Polynomial Curve Fitting

We will fit a polynomial regression model of the form:

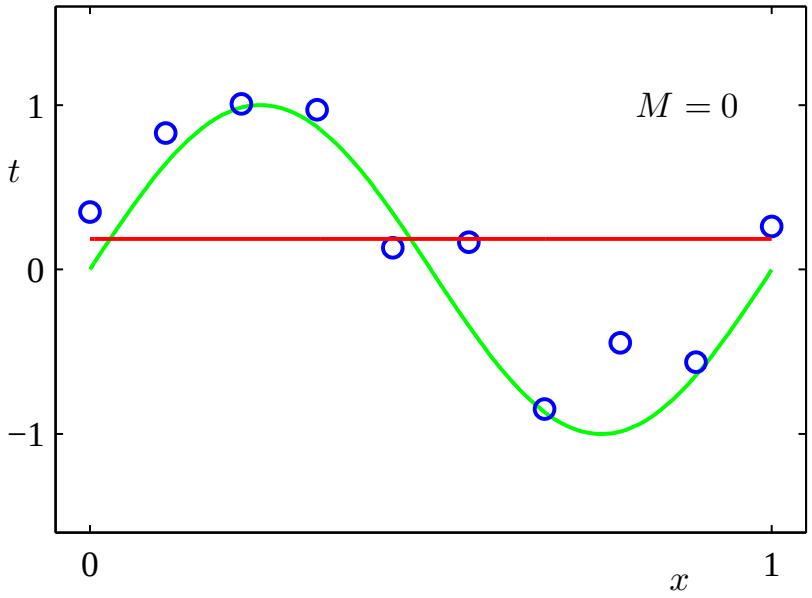
$$h(x, \mathbf{w}) = w_0 + w_1 x^1 + w_2 x^2 + \ldots + w_M x^M$$

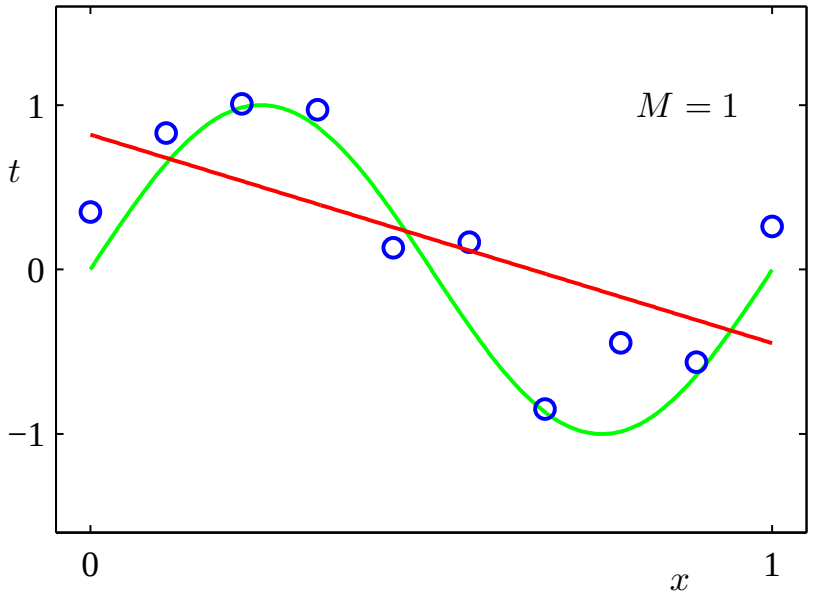
Or, using features $\phi_j(x) = x^j$, in basis function notation:

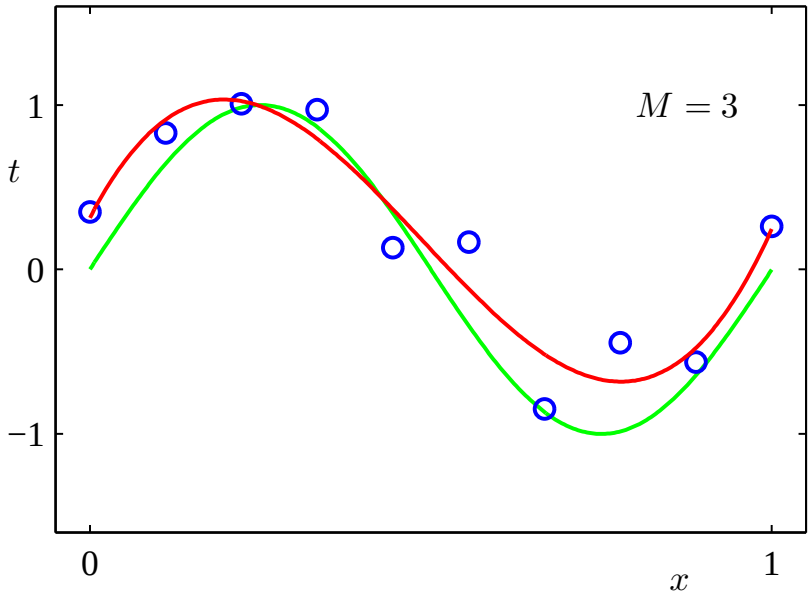
$$h(x, \mathbf{w}) = \sum_{j=0}^{M} w_j \phi_j = \mathbf{w}^T \phi(x)$$

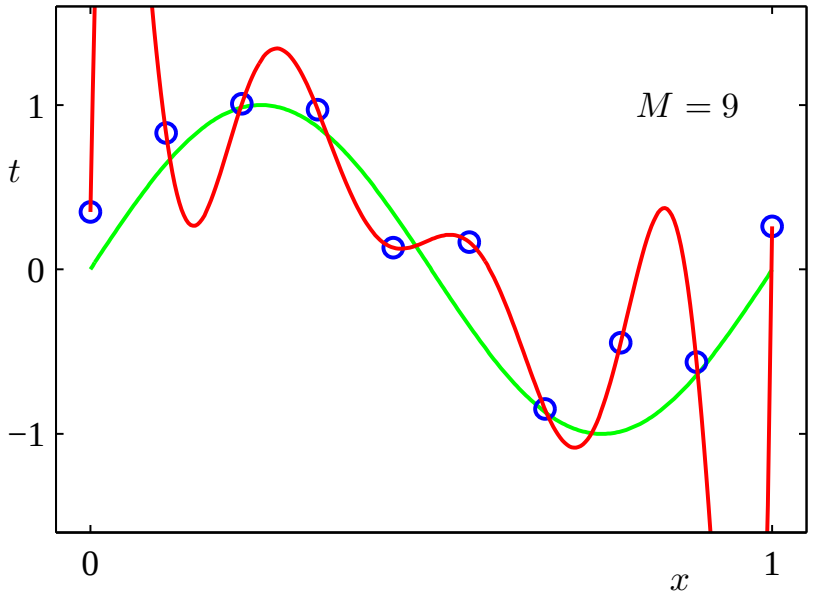
How should we choose M? 

How to choose M?





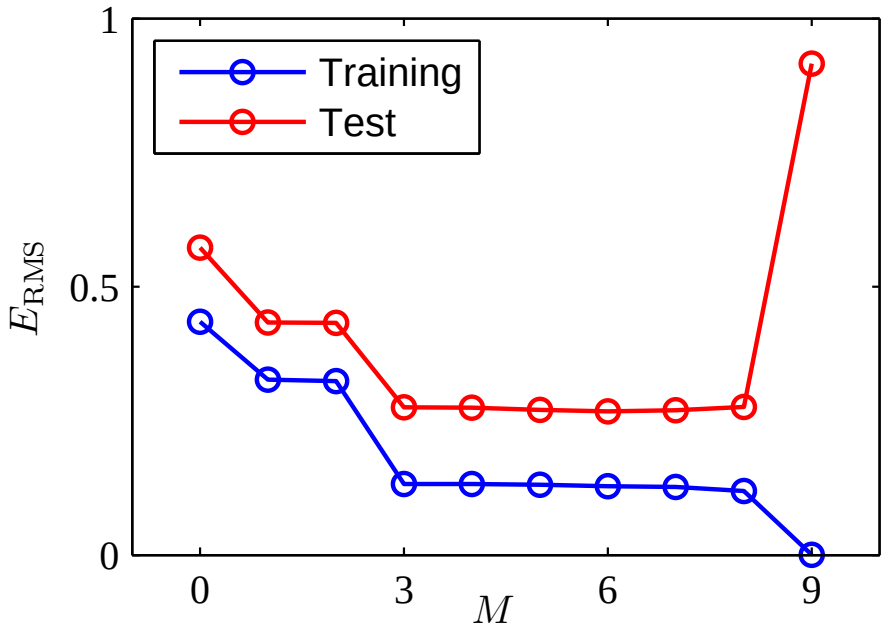




Generalization Performance

We care about good generalization, i.e. accurate predictions for new data.

- Quantify dependence of generalization performance on M on separate test set



- Overfitting: *Fitting the data more than is warranted*
- Problem: Flexible hypothesis class starts to fit the noise to reduce training error

Approximation-Generalization Tradeoff

- More complex hypothesis class $\rightsquigarrow$
  more flexibility to approximate target function

- Less complex hypothesis class $\rightsquigarrow$
  more likely to generalize well on new data

Questions:

- How to quantify this tradeoff?
- What is the correct hypothesis class $\mathcal{H}$ to choose from?

Loss Function and Risk Minimization

In general, when taking a decision based on our learned model, we strive to minimize wrong decisions as measured by a user-defined loss function (e.g. squared error loss for regression).
The *expected* loss is also called risk, defined as:

$$R(h) = \mathbb{E}_{p(x,y)}[L(h(x), y)] = \int L(h(x), y)p(x, y)dxdy$$

This is the quantity we would ultimately like to minimize. The optimal hypothesis would then be:

$$h^* \in \arg\min_{h \in \mathcal{H}} R(h)$$

However, $p(x, y)$ is generally unknown.

Loss Function and Risk Minimization - contd

We approximate the risk with sampled data (empirical risk):

$$R_{\text{emp}}(h) = \frac{1}{N} \sum_{i=1}^{N} L(h(x_i), y_i)$$

The estimated optimal hypothesis is then:

$$\hat{h} \in \arg\min_{h \in \mathcal{H}} R_{\text{emp}}(h)$$

It depends, of course, on the available dataset:

$$\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} = \langle (\mathbf{x_i}, y_i) \rangle_{i=1}^{N} \sim p(x, y)$$

Question:

- Can we estimate the error we make by this approximation?

Lecture Overview

Risk Decomposition

Decomposing the risk for regression with the squared error function:

$$L(h(x), y) = \left(\hat{h}(x) - y\right)^2$$

The risk can be decomposed into (somewhat lengthy derivation, see [Bishop, Section 1.5.5 and 3.2]):

$$\begin{aligned} R(h) &= \mathbb{E}_{p(x,y)}[L] \\ &= \int \left(\hat{h}(x) - h^*(x)\right)^2 p(x)dx + \int (h^*(x) - y)^2 \, p(x,y)dxdy \end{aligned}$$

- First term can be minimized by choosing the right hypothesis.
- Second term is the noise in the data, independent of our hypothesis.

Average Performance and Average Hypothesis

We focus on the first integrand and see what happens when we fit the hypothesis depending on datasets $\hat{h}(x) = \hat{h}(x, \mathcal{D})$ trying to minimize the empirical risk R_{emp}.

- For different data sets, we obtain different prediction functions $\hat{h}(x, \mathcal{D})$.
- Results in different values of the loss; e.g. for data set $\mathcal{D}$, consider $(\hat{h}(x, \mathcal{D}) - h^*(x))^2$.

Suppose now, we had a large number of data sets $\mathcal{D}$ of size N, each drawn independently from $p(x, y)$.

$\rightarrow$ get a better performance estimate of the learning algorithm by considering its average performance over data sets, i.e. evaluating $\mathbb{E}_{\mathcal{D}}[\left(\hat{h}(x, \mathcal{D}) - h^*(x)\right)^2]$.

Average Performance and Average Hypothesis - contd

Define average hypothesis: $\bar{h}(x) = \mathbb{E}_{\mathcal{D}}[\hat{h}(x, \mathcal{D})]$. Now use this in the average performance:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - h^*(\mathbf{x})\}^2] &= \mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - \bar{h}(\mathbf{x}) + \bar{h}(\mathbf{x}) - h^*(\mathbf{x})\}^2] \\
&= \mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - \bar{h}(\mathbf{x})\}^2 + \{\bar{h}(\mathbf{x}) - h^*(\mathbf{x})\}^2 \\
&\quad +2\{\hat{h}(\mathbf{x};\mathcal{D}) - \bar{h}(\mathbf{x})\}\{\bar{h}(\mathbf{x}) - h^*(\mathbf{x})\}] \\
&= \underbrace{\{\bar{h}(\mathbf{x}) - h^*(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - \bar{h}(\mathbf{x})\}^2]}_{\text{variance}}
\end{aligned}$$

Bias and Variance

Overall, we get:

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

where

$$(\text{bias})^2 = \int \{\bar{h}(x) - h^*(x)\}^2 p(x) \mathrm{d}x$$

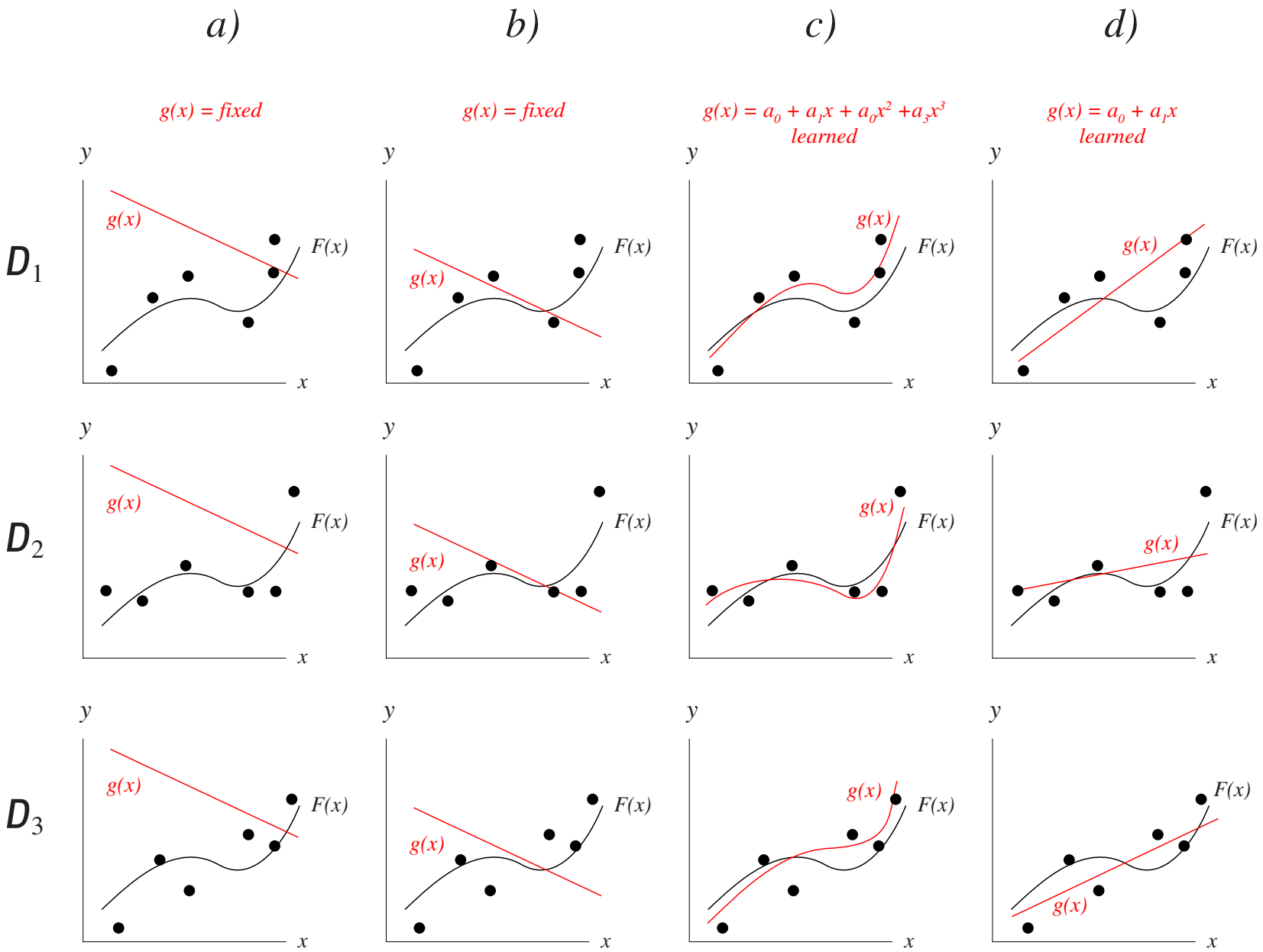
$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x}; \mathcal{D}) - \bar{h}(\mathbf{x})\}^2] p(x) \mathrm{d}x$$

$$\text{noise} = \int \{h^*(\mathbf{x}) - y)\}^2 p(x, y) \mathrm{d}x \mathrm{d}y$$

where $\bar{h}(x) = \mathbb{E}_{\mathcal{D}}[\hat{h}(x, \mathcal{D})]$ is the average hypothesis in expectation over many datasets drawn from $\mathcal{D}$

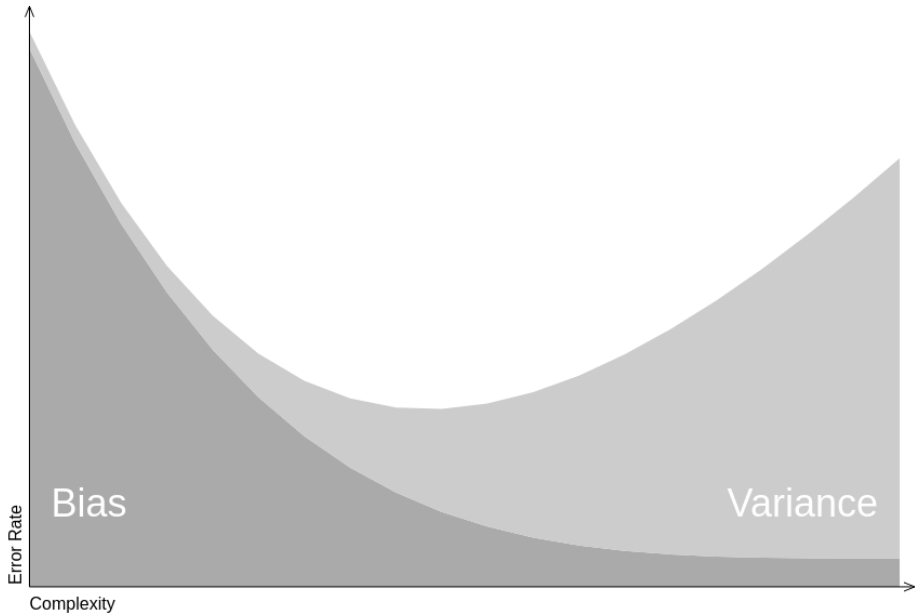
$\rightarrow$ Bias-Variance decomposition lets us judge model class complexity by the data resources available for training

Bias - Variance: Examples



Approximation-Generalization Tradeoff – Revisited

- More complex models $\rightarrow$ more flexibility to approximate target function (low bias, high variance)
- Less complex models $\rightarrow$ more likely to generalize well on new data (high bias, low variance)



Balancing bias and variance gives optimal predictive capability.

Remarks on Bias-Variance-Analysis

Bias-Variance-Analysis is not very useful in practice, because ...

1. requires the ground truth for the bias and noise terms
2. splitting the data into many datasets to estimate the variance

In practice, we need something feasible!

Lecture Overview

Validation: Hold-out

- Validation is the process to check the performance of a learned function on independent validation data.
- Hold-out validation:
  - split the available data from the underlying distribution $p(x, y)$ into two disjoint subsets: training set $\mathcal{D}_{\text{train}}$ and validation set $\mathcal{D}_{\text{validation}}$
  - train your model only on the training set
  - evaluate empirical risk of the learned hypothesis on the validation set
- Remark: Since $\mathcal{D}_{\text{validation}}$ is often used to optimize hyperparameters, it's crucial to also keep yet another separate test set $\mathcal{D}_{\text{test}}$ to test final performance.

Remarks: Hold-out

- There is no clear answer on how many percent of the data should go into $\mathcal{D}_{\text{train}}$ and how many into $\mathcal{D}_{\text{validation}}$
- Two problematic cases:
  - $\mathcal{D}_{\text{validation}}$ is too small $\rightsquigarrow$ your loss estimate will be poor
  - $\mathcal{D}_{\text{train}}$ is too small $\rightsquigarrow$ your estimated loss will be too pessimistic because you used less data for training than you actually would use in practice
- if your $\mathcal{D}$ is quite small, different hold-out splits might give your different results
- Recommendation: Only use hold-out validation if you have a large dataset where you can afford to put some data to the side and if a different split of your data will likely lead to similar results

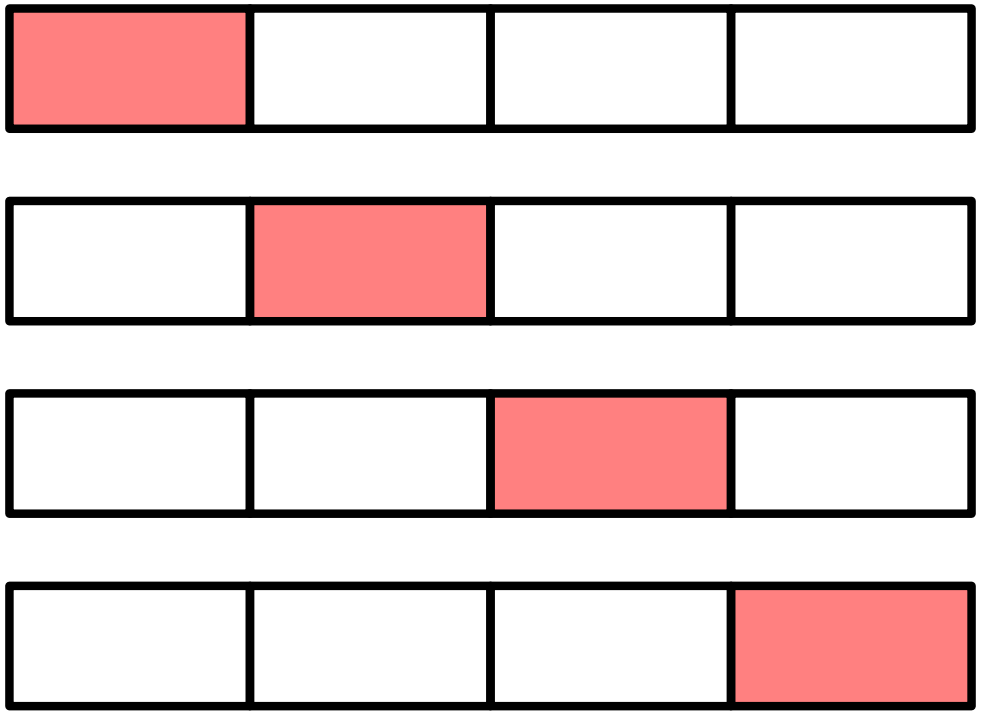
k-fold Cross-Validation

Require: $\mathcal{D}$ with N data points, $2 \leq k \leq N$

1: split $\mathcal{D}$ into k disjoint subsets of (roughly) equal size: $\mathcal{D}_1, \ldots, \mathcal{D}_k$
2: **for** $i = 1$ to k **do**
3: (re-) initialize hypothesis parameters
4: train hypothesis on set $\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_{i-1} \cup \mathcal{D}_{i+1} \cup \cdots \cup \mathcal{D}_k$
5: calculate empirical risk R^i_{emp} on $\mathcal{D}_i$
6: **end for**
7: **return** $\frac{1}{k} \sum_{i=1}^{k} R^i_{\text{emp}}$

- Advantage: each hypothesis is learned on most data points ($\frac{k-1}{k} N$) and every data point is used at some point for validation
- Disadvantage: hypothesis has to be learned k times
- Recommendation: quite common to set k to 10

Cross-Validation - Illustration



Cross-Validation - Example

- Example: $\mathcal{D} = \{z_1, \ldots, z_{17}\}$, 3-fold crossvalidation
- Split training set randomly:

$$\mathcal{D}_1 = \{z_1, z_2, z_8, z_{13}, z_{14}, z_{15}\}$$
$$\mathcal{D}_2 = \{z_4, z_5, z_{10}, z_{11}, z_{12}, z_{17}\}$$
$$\mathcal{D}_3 = \{z_3, z_6, z_7, z_9, z_{16}\}$$

- Train the model three times:
  1. use $\mathcal{D}_2 \cup \mathcal{D}_3$ for training, $\mathcal{D}_1$ for validation: empirical risk R^1_{emp}
  2. use $\mathcal{D}_1 \cup \mathcal{D}_3$ for training, $\mathcal{D}_2$ for validation: empirical risk R^2_{emp}
  3. use $\mathcal{D}_1 \cup \mathcal{D}_2$ for training, $\mathcal{D}_3$ for validation: empirical risk R^3_{emp}
- Calculate cross-validation loss (average empirical risk over folds):

$$R_{\text{cv}} = \frac{R^1_{\text{emp}} + R^2_{\text{emp}} + R^3_{\text{emp}}}{3}$$

Leave-One-Out (LOO)

- The extreme case of cross-validation is to set k to $|\mathcal{D}|$

$\rightsquigarrow$ in each iteration, only a single point is used for validation

- Super expensive!
  - you can only use it on very small datasets
- Nevertheless, it is a good estimate of your loss since you use $|\mathcal{D}| - 1$ points for training

- (the method is also known as jackknifing)

Out-of-Bootstrap Validation

- cross-validation has the problem that the splits are not independent, and thus the loss estimates are not independent

$\rightsquigarrow$ you cannot apply some tests on these statistics
(e.g., statistical hypothesis tests)

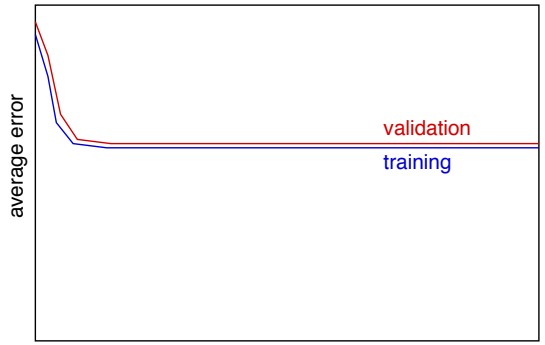
- Out-of-Bootstrap Validation:
  1. Bootstrap sample (with replacement!) from $\mathcal{D}$
  2. All samples not being in the training set are put into the test set

- by repeating the above procedure, we get independent estimates of the loss

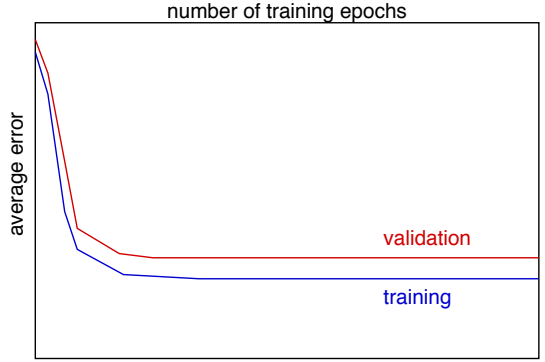
- Problem:
  since we sample with replacement, we change the training distribution

Overfitting / Underfitting: Training and Validation error behavior

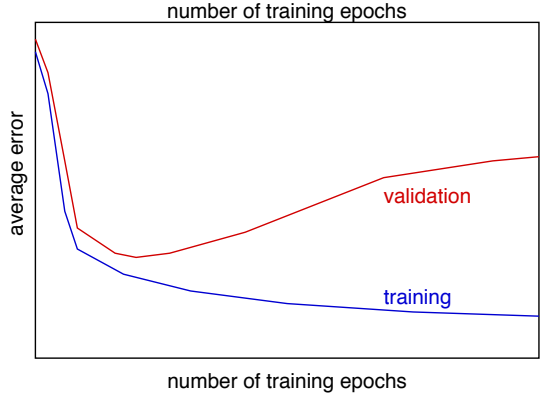
Example of underfitting:
validation and training error remain large
(often too large bias in hypothesis)



Example of successful learning:
validation error and training error monotonically decrease



Example of overfitting:
validation error increases while training error decreases
(often too large variance in hypothesis)



Lecture Overview

Summary by learning goals

Having heard this lecture, you can now . . .

- describe what overfitting means and why it happens
- explain the approximation-generalization tradeoff
- quantify the approximation-generalization tradeoff using the bias-variance analysis
- explain different validation policies for estimating your empirical loss