Regression and generalization

CE-717: Machine Learning
Sharif University of Technology

M. Soleymani
Fall 2018
Topics

- Beyond linear regression models
- Evaluation & model selection
- Regularization
- Bias-Variance
Recall: Linear regression (squared loss)

- Linear regression functions

\[ f : \mathbb{R} \to \mathbb{R} \quad f(x; \mathbf{w}) = w_0 + w_1 x \]

\[ f : \mathbb{R}^d \to \mathbb{R} \quad f(x; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d \]

\[ \mathbf{w} = [w_0, w_1, \ldots, w_d]^T \] are the parameters we need to set.

- Minimizing the squared loss for linear regression

\[ J(\mathbf{w}) = \| \mathbf{y} - X\mathbf{w} \|^2_2 \]

- We obtain \( \hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{y} \)
Beyond linear regression

- How to extend the linear regression to non-linear functions?
  - Transform the data using basis functions
  - Learn a linear regression on the new feature vectors (obtained by basis functions)
Beyond linear regression

- $m^{th}$ order polynomial regression (univariate $f : \mathbb{R} \to \mathbb{R}$)

$$f(x; \mathbf{w}) = w_0 + w_1 x + \ldots + w_{m-1} x^{m-1} + w_m x^m$$

- Solution: $\widehat{\mathbf{w}} = (X'^T X')^{-1} X'^T \mathbf{y}$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad X' = \begin{bmatrix} 1 & x^{(1)1} & x^{(1)2} & \ldots & x^{(1)m} \\ 1 & x^{(2)1} & x^{(2)2} & \ldots & x^{(2)m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x^{(n)1} & x^{(n)2} & \ldots & x^{(n)1} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} \widehat{w}_0 \\ \widehat{w}_1 \\ \vdots \\ \widehat{w}_m \end{bmatrix}$$
Polynomial regression: example

\[ m = 1 \]

\[ m = 3 \]

\[ m = 5 \]

\[ m = 7 \]
Generalized linear

- Linear combination of fixed non-linear function of the input vector

\[ f(x; w) = w_0 + w_1 \phi_1(x) + \ldots + w_m \phi_m(x) \]

\{\phi_1(x), \ldots, \phi_m(x)\}: set of basis functions (or features)

\[ \phi_i(x): \mathbb{R}^d \rightarrow \mathbb{R} \]
Basis functions: examples

- **Linear**
  \[
  \text{If } m = d, \phi_i(x) = x_i, \ i = 1, \ldots, d, \text{ then}
  \]
  \[
  f(x; \mathbf{w}) = w_0 + w_1x_1 + \cdots + w_dx_d
  \]

- **Polynomial (univariate)**
  \[
  \text{If } \phi_i(x) = x^i, \ i = 1, \ldots, m, \text{ then}
  \]
  \[
  f(x; \mathbf{w}) = w_0 + w_1x + \cdots + w_{m-1}x^{m-1} + w_mx^m
  \]
Gaussian: $\phi_j(x) = \exp\left\{-\frac{(x-c_j)^2}{2\sigma_j^2}\right\}$

Sigmoid: $\phi_j(x) = \sigma\left(\frac{\|x-c_j\|}{\sigma_j}\right)$

$\sigma(a) = \frac{1}{1+\exp(-a)}$
Radial Basis Functions: prototypes

- Predictions based on similarity to “prototypes”:
  \[ \phi_j(x) = \exp \left\{ -\frac{1}{2\sigma_j^2} \|x - c_j\|^2 \right\} \]

- Measuring the similarity to the prototypes \( c_1, \ldots, c_m \)
  - \( \sigma_j^2 \) controls how quickly it vanishes as a function of the distance to the prototype.
  - Training examples themselves could serve as prototypes
Generalized linear: optimization

\[ J(w) = \sum_{i=1}^{n} \left( y(i) - f(x(i); w) \right)^2 \]
\[ = \sum_{i=1}^{n} \left( y(i) - w^T \phi(x(i)) \right)^2 \]

\[ y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \quad \Phi = \begin{bmatrix} 1 & \phi_1(x^{(1)}) & \cdots & \phi_m(x^{(1)}) \\ 1 & \phi_1(x^{(2)}) & \cdots & \phi_m(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x^{(n)}) & \cdots & \phi_m(x^{(n)}) \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \]

\[ \hat{w} = (\Phi^T \Phi)^{-1} \Phi^T y \]
Model complexity and overfitting

- With limited training data, models may achieve zero training error but a large test error.

Training (empirical) loss
\[ \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - f(x^{(i)}; \theta) \right)^2 \approx 0 \]

Expected (true) loss
\[ E_{x,y} \{ (y - f(x; \theta))^2 \} \gg 0 \]

- **Over-fitting**: when the training loss no longer bears any relation to the test (generalization) loss.
  - Fails to generalize to unseen examples.
Polynomial regression

\[ m = 0 \]

\[ m = 1 \]

\[ m = 3 \]

\[ m = 9 \]

[Bishop]
Polynomial regression: training and test error

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left( y^{(i)} - f(x^{(i)}; \theta) \right)^2}{n}}
\]

[Graph showing training and test error over different values of m.

[Bishop]
Over-fitting causes

- Model complexity
  - E.g., Model with a large number of parameters (degrees of freedom)

- Low number of training data
  - Small data size compared to the complexity of the model
Model complexity

- Example:
  - Polynomials with larger $m$ are becoming increasingly tuned to the random noise on the target values.

\[
y = \begin{cases} 
  m = 0 \\
  m = 1 \\
  m = 3 \\
  m = 9 
\end{cases}
\]
Number of training data & overfitting

- Over-fitting problem becomes less severe as the size of training data increases.

\[ m = 9 \]
\[ n = 15 \]

\[ m = 9 \]
\[ n = 100 \]

[Bishop]
How to evaluate the learner’s performance?

- Generalization error: true (or expected) error that we would like to optimize

- Two ways to assess the generalization error is:
  - Practical: Use a separate data set to test the model
  - Theoretical: Law of Large numbers
    - statistical bounds on the difference between training and expected errors
Avoiding over-fitting

- Determine a suitable value for model complexity (Model Selection)
  - Simple hold-out method
  - Cross-validation

- Regularization (Occam’s Razor)
  - Explicit preference towards simple models
  - Penalize for the model complexity in the objective function

- Bayesian approach
Evaluation and model selection

- **Evaluation:**
  - We need to measure how well the learned function can predicts the target for unseen examples

- **Model selection:**
  - Most of the time we need to select among a set of models
    - Example: polynomials with different degree $m$
  - and thus we need to evaluate these models first
Model Selection

- **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)

- **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select
Model Selection

- Model selection is the process by which we choose the “best” model from among a set of candidates
  - assume access to a function capable of measuring the quality of a model
  - typically done “outside” the main training algorithm

- Model selection / hyperparameter optimization is just another form of learning
Simple hold-out: model selection

Steps:
- Divide training data into training and validation set $v_{set}$
- Use only the training set to train a set of models
- Evaluate each learned model on the validation set
  \[ J_v(w) = \frac{1}{|v_{set}|} \sum_{i \in v_{set}} \left( y^{(i)} - f(x^{(i)}; w) \right)^2 \]
- Choose the best model based on the validation set error

Usually, too wasteful of valuable training data
- Training data may be limited.
- On the other hand, small validation set give a relatively noisy estimate of performance.
Simple hold out: training, validation, and test sets

- Simple hold-out chooses the model that minimizes error on validation set.

- $J_v(\hat{w})$ is likely to be an optimistic estimate of generalization error.
  - extra parameter (e.g., degree of polynomial) is fit to this set.

- Estimate generalization error for the test set
  - performance of the selected model is finally evaluated on the test set
Cross-Validation (CV): Evaluation

- **k-fold cross-validation steps:**
  - Shuffle the dataset and randomly partition training data into $k$ groups of approximately equal size
  - for $i = 1$ to $k$
    - Choose the $i$-th group as the held-out validation group
    - Train the model on all but the $i$-th group of data
    - Evaluate the model on the held-out group
  - Performance scores of the model from $k$ runs are averaged.
    - The average error rate can be considered as an estimation of the true performance.

<table>
<thead>
<tr>
<th></th>
<th>First run</th>
<th>Second run</th>
<th>...</th>
<th>(k-1)th run</th>
<th>k-th run</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cross-Validation (CV): Model Selection

- For each model we first find the average error find by CV.

- The model with the best average performance is selected.
Cross-validation: polynomial regression example

- 5-fold CV
- 100 runs
  - average

\[ m = 1 \]
CV: \( MSE = 0.30 \)

\[ m = 3 \]
CV: \( MSE = 1.45 \)

\[ m = 5 \]
CV: \( MSE = 45.44 \)

\[ m = 7 \]
CV: \( MSE = 31759 \)
Leave-One-Out Cross Validation (LOOCV)

- When data is particularly scarce, cross-validation with $k = N$
  - Leave-one-out treats each training sample in turn as a test example and all other samples as the training set.

- Use for small datasets
  - When training data is valuable
  - LOOCV can be time expensive as $N$ training steps are required.
Regularization

- Adding a penalty term in the cost function to discourage the coefficients from reaching large values.

- Ridge regression (weight decay):

\[
J(w) = \sum_{i=1}^{n} \left( y^{(i)} - w^T \phi(x^{(i)}) \right)^2 + \lambda w^T w
\]

\[
\hat{w} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y
\]
Polynomial order

- Polynomials with larger $m$ are becoming increasingly tuned to the random noise on the target values.
  - magnitude of the coefficients typically gets larger by increasing $m$.

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 6$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>-1.27</td>
<td>7.99</td>
<td></td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$w_3^*$</td>
<td></td>
<td>17.37</td>
<td></td>
<td>48568.31</td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td></td>
<td>-231639.30</td>
<td></td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td>640042.26</td>
<td></td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td>-1061800.52</td>
<td></td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td>1042400.18</td>
<td></td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td>-557682.99</td>
<td></td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td>125201.43</td>
<td></td>
</tr>
</tbody>
</table>

[ Bishop ]
### Regularization parameter

For $m = 9$:

<table>
<thead>
<tr>
<th></th>
<th>$\ln \lambda = -\infty$</th>
<th>$\ln \lambda = -18$</th>
<th>$\ln \lambda = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{w}_0$</td>
<td>0.35</td>
<td>0.35</td>
<td>0.13</td>
</tr>
<tr>
<td>$\hat{w}_1$</td>
<td>232.37</td>
<td>4.74</td>
<td>-0.05</td>
</tr>
<tr>
<td>$\hat{w}_2$</td>
<td>-5321.83</td>
<td>-0.77</td>
<td>-0.06</td>
</tr>
<tr>
<td>$\hat{w}_3$</td>
<td>48568.31</td>
<td>-31.97</td>
<td>-0.05</td>
</tr>
<tr>
<td>$\hat{w}_4$</td>
<td>-231639.30</td>
<td>-3.89</td>
<td>-0.03</td>
</tr>
<tr>
<td>$\hat{w}_5$</td>
<td>640042.26</td>
<td>55.28</td>
<td>-0.02</td>
</tr>
<tr>
<td>$\hat{w}_6$</td>
<td>-1061800.52</td>
<td>41.32</td>
<td>-0.01</td>
</tr>
<tr>
<td>$\hat{w}_7$</td>
<td>1042400.18</td>
<td>-45.95</td>
<td>-0.00</td>
</tr>
<tr>
<td>$\hat{w}_8$</td>
<td>-557682.99</td>
<td>-91.53</td>
<td>0.00</td>
</tr>
<tr>
<td>$\hat{w}_9$</td>
<td>125201.43</td>
<td>72.68</td>
<td>0.01</td>
</tr>
</tbody>
</table>

---

[2] Bishop
Regularization parameter

- **Generalization**
  - $\lambda$ now controls the effective complexity of the model and hence determines the degree of over-fitting

![Graph showing \( E_{\text{RMS}} \) versus \( \ln \lambda \).]
Choosing the regularization parameter

- A set of models with different values of $\lambda$.

- Find $\hat{\mathbf{w}}$ for each model based on training data

- Find $J_v(\hat{\mathbf{w}})$ (or $J_{cv}(\hat{\mathbf{w}})$) for each model
  
  \[ J_v(\mathbf{w}) = \frac{1}{n_v} \sum_{i \in v \text{-} set} \left( y^{(i)} - f(x^{(i)}; \mathbf{w}) \right)^2 \]

- Select the model with the best $J_v(\hat{\mathbf{w}})$ (or $J_{cv}(\hat{\mathbf{w}})$)
The approximation-generalization trade-off

- Small true error shows good approximation of $f$ out of sample

- More complex $\mathcal{H}$ $\Rightarrow$ better chance of approximating $f$

- Less complex $\mathcal{H}$ $\Rightarrow$ better chance of generalization out of $f$
Complexity of Hypothesis Space: Example

This example has been adapted from: Prof. Andrew Ng’s slides
Complexity of Hypothesis Space: Example

This example has been adapted from: Prof. Andrew Ng’s slides
Complexity of Hypothesis Space: Example

\[ J_v(w) = \frac{1}{n_v} \sum_{i \in \text{eval set}} (y^{(i)} - f(x^{(i)}; w))^2 \]

\[ J_{\text{train}}(w) = \frac{1}{n_{\text{train}}} \sum_{i \in \text{train set}} (y^{(i)} - f(x^{(i)}; w))^2 \]
Complexity of Hypothesis Space

- Less complex $\mathcal{H}$:
  - $J_{\text{train}}(\hat{w}) \approx J_v(\hat{w})$ and $J_{\text{train}}(\hat{w})$ is very high

- More complex $\mathcal{H}$:
  - $J_{\text{train}}(\hat{w}) \ll J_v(\hat{w})$ and $J_{\text{train}}(\hat{w})$ is low

![Diagram showing the relationship between $J_v(\hat{w})$, $J_{\text{train}}(\hat{w})$, and the degree of polynomial $m$.](attachment:diagram.png)
Size of training set

\[
J_v(w) = \frac{1}{n_v} \sum_{i \in \text{val_set}} (y^{(i)} - f(x^{(i)}; w))^2
\]

\[
J_{\text{train}}(w) = \frac{1}{n_{\text{train}}} \sum_{i \in \text{train_set}} (y^{(i)} - f(x^{(i)}; w))^2
\]

\[f(x; w) = w_0 + w_1 x + w_2 x^2\]

This slide has been adapted from: Prof. Andrew Ng's slides
Less complex $\mathcal{H}$

If model is very simple, getting more training data will not (by itself) help much.
More complex $\mathcal{H}$

For more complex models, getting more training data is usually helps.

\[ f(x; \mathbf{w}) = w_0 + w_1x + \cdots w_{10}x^{10} \]
Regularization: Example

\[ f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \]

\[ J(w) = \frac{1}{n} \left( \sum_{i=1}^{n} \left( y^{(i)} - f(x^{(i)}; \mathbf{w}) \right)^2 + \lambda \mathbf{w}^T \mathbf{w} \right) \]

\[ w_1 = w_2 \approx 0 \]

This example has been adapted from: Prof. Andrew Ng’s slides
Theoretical Part
Model complexity: Bias-variance trade-off

- Least squares, can lead to severe over-fitting if complex models are trained using data sets of limited size.

- A frequentist viewpoint of the model complexity issue, known as the *bias-variance trade-off*. 
Formal discussion on bias, variance, and noise

- Best unrestricted regression function
- Noise
- Bias and variance
The learning diagram: deterministic target

**UNKNOWN TARGET FUNCTION**

\[ h: x \rightarrow y \]

**TRAINING EXAMPLES**

\((x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\)

**LEARNING ALGORITHM**

\[ \mathcal{A} \]

**HYPOTHESIS SET**

\[ \mathcal{H} \]

**FINAL HYPOTHESIS**

\[ f: x \rightarrow y \]

**PROBABILITY DISTRIBUTION**

\[ P \text{ on } \mathcal{X} \]

\[ x^{(1)}, \ldots, x^{(N)} \]

[Y.S. Abou Mostafa, 2012]
The learning diagram including noisy target

- **Type**

  - **Unknown Target Distribution**: $P(y | x)$
    - Target function $h: X \rightarrow Y$ plus noise
  - **Training Examples**: $(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})$
  - **Learning Algorithm**: $\mathcal{A}$
  - **Final Hypothesis**: $f: X \rightarrow Y$
  - **Hypothesis Set**: $\mathcal{H}$
  - **Probability Distribution on Features**: $P$ on $X$
  - **Target Distribution**: $P(x, y) = P(x)P(y|x)$

- [Y.S. Abou Mostafa, 2012]
Best unrestricted regression function

- If we know the joint distribution $P(x, y)$ and no constraints on the regression function?
- cost function: mean squared error

$$ h^* = \arg\min_{h: \mathbb{R}^d \to \mathbb{R}} \mathbb{E}_{x,y} \left[ (y - h(x))^2 \right] $$

$$ h^*(x) = \mathbb{E}_{y|x}[y] $$
Best unrestricted regression function: Proof

\[ \mathbb{E}_{x,y} \left[ (y - h(x))^2 \right] = \int \int (y - h(x))^2 p(x,y) dx dy \]

- For each \( x \) separately minimize loss since \( h(x) \) can be chosen independently for each different \( x \):

\[ \frac{\delta \mathbb{E}_{x,y} \left[ (y - h(x))^2 \right]}{\delta h(x)} = - \int 2(y - h(x))p(x,y)dy = 0 \]

\[ \Rightarrow h(x) = \frac{\int yp(x,y)dy}{\int p(x,y)dy} = \frac{\int yp(x,y)dy}{p(x)} = \int yp(y|x)dy = \mathbb{E}_{y|x} [y] \]

\[ \Rightarrow h^*(x) = \mathbb{E}_{y|x} [y] \]
Error decomposition

\( E_{true}(f_D(x)) = \mathbb{E}_{x,y}[(f_D(x) - y)^2] \)

\( = \mathbb{E}_{x,y}[(f_D(x) - h(x) + h(x) - y)^2] \)

\( = \mathbb{E}_x[(f_D(x) - h(x))^2] + \mathbb{E}_{x,y}[(h(x) - y)^2] + 2\mathbb{E}_{x,y}[(f_D(x) - h(x))(h(x) - y)] + 2\mathbb{E}_x[(f_D(x) - h(x))\mathbb{E}_{y|x}[(h(x) - y)]] \)

\( = 0 \)

\((x, y) \sim P\)

\( h(x) \) : minimizes the expected loss
Error decomposition

\[ E_{\text{true}}(f_D(x)) = \mathbb{E}_{x,y}[(f_D(x) - y)^2] \]

\[ = \mathbb{E}_{x,y} [(f_D(x) - h(x) + h(x) - y)^2] \]

\[ = \mathbb{E}_x [(f_D(x) - h(x))^2] + \mathbb{E}_{x,y}[(h(x) - y)^2] + 0 \]

- Noise shows the irreducible minimum value of the loss function

\[(x, y) \sim P \]

\[ h(x) : \text{minimizes the expected loss} \]
Expectation of true error

\[
E_{true}(f_D(x)) = \mathbb{E}_{x,y}[(f_D(x) - y)^2] \\
= \mathbb{E}_x[(f_D(x) - h(x))^2] + \text{noise}
\]

\[
\mathbb{E}_D \left[ \mathbb{E}_x[(f_D(x) - h(x))^2] \right] \\
= \mathbb{E}_x \left[ \mathbb{E}_D[(f_D(x) - h(x))^2] \right]
\]

We now want to focus on \( \mathbb{E}_D[(f_D(x) - h(x))^2] \).
The average hypothesis

\[ \bar{f}(x) \equiv E_D[f_D(x)] \]

\[ \bar{f}(x) \approx \frac{1}{K} \sum_{k=1}^{K} f_{D(k)}(x) \]

*K* training sets (of size *N*) sampled from *P(x, y)*:

\[ D^{(1)}, D^{(2)}, ..., D^{(K)} \]
Using the average hypothesis

\[
\mathbb{E}_D \left[ (f_D(x) - h(x))^2 \right] \\
= \mathbb{E}_D \left[ (f_D(x) - \bar{f}(x) + \bar{f}(x) - h(x))^2 \right] \\
= \mathbb{E}_D \left[ (f_D(x) - \bar{f}(x))^2 + (\bar{f}(x) - h(x))^2 \right. \\
+ 2 (f_D(x) - \bar{f}(x)) (\bar{f}(x) - h(x)) \left. \right]\ \\
= \mathbb{E}_D \left[ (f_D(x) - \bar{f}(x))^2 \right] + (\bar{f}(x) - h(x))^2
\]
Bias and variance

\[ \mathbb{E}_D \left[ (f_D(x) - h(x))^2 \right] = \mathbb{E}_D \left[ (f_D(x) - \bar{f}(x))^2 \right] + (\bar{f}(x) - h(x))^2 \]

\[ \mathbb{E}_x \left[ \mathbb{E}_D \left[ (f_D(x) - h(x))^2 \right] \right] = \mathbb{E}_x [\text{var}(x) + \text{bias}(x)] \]

\[ = \text{var} + \text{bias} \]
Bias-variance trade-off

\[ \text{var} = \mathbb{E}_x \left[ \mathbb{E}_\mathcal{D} \left[ (f_\mathcal{D}(x) - \bar{f}(x))^2 \right] \right] \]

\[ \text{bias} = \mathbb{E}_x [\bar{f}(x) - h(x)] \]

More complex \( \mathcal{H} \Rightarrow \) lower bias but higher variance

[Y.S. Abou Mostafa, 2012]
Example: sin target

- Only two training example \( N = 2 \)

- Two models used for learning:
  - \( \mathcal{H}_0 : f(x) = b \)
  - \( \mathcal{H}_1 : f(x) = ax + b \)

- Which is better \( \mathcal{H}_0 \) or \( \mathcal{H}_1 \)?
Learning from a training set

$\mathcal{H}_0$

$\mathcal{H}_1$

[Y.S. Abou Mostafa, 2012]
Variance $\mathcal{H}_0$

$\mathcal{H}_0$
Variance $\mathcal{H}_1$

\[ \mathbb{E}(f(\mathbf{x})) \]
Which is better?

\[ \mathcal{H}_0 \]

\[ \mathcal{H}_1 \]

\[ \sin(x) \]

\[ \tilde{f}(x) \]

\[ f(x) \]

\[ y \]

\[ x \]

bias = 0.50  \hspace{1cm} var = 0.25

bias = 0.21  \hspace{1cm} var = 1.69

[Y.S. Abou Mostafa, 2012]
Lesson

Match the **model complexity**
to the **data sources**
not to the complexity of the **target function**.
Expected training and true error curves

- Errors vary with the number of training samples

**Expected True Error**

\[ E_{\text{true}} \]

**Expected Training Error**

\[ E_{\text{train}} \]

- **Simple Model**
- **Complex Model**

**Expected True Error**

\[ \mathbb{E}_D[E_{\text{true}}(f_D(x))] \]

**Expected Training Error**

\[ \mathbb{E}_D[E_{\text{train}}(f_D(x))] \]

[Y.S. Abou Mostafa, 2012]
Regularization

[Image of graph showing a comparison between a curve without regularization and one with regularization, labeled as [Y.S. Abou Mostafa, 2012]]
Regularization: bias and variance

\[ \bar{f}(x) = \sin(\pi x) \]

**Without regularization**
- Bias: 0.21
- Variance: 1.69

**With regularization**
- Bias: 0.23
- Variance: 0.33

[Y.S. Abou Mostafa, 2012]
Winner of $\mathcal{H}_0$, $\mathcal{H}_1$, and $\mathcal{H}_1$ with regularization

\[ f(\bar{x}) \]

\[ \mathcal{H}_0 \]
\[ \mathcal{H}_1 \]
\[ \mathcal{H}_1 \text{ with regularization} \]

bias = 0.50 \hspace{1cm} \text{var} = 0.25

bias = 0.21 \hspace{1cm} \text{var} = 1.69

bias = 0.23 \hspace{1cm} \text{var} = 0.33

[Y.S. Abou Mostafa, 2012]
Regularization and bias/variance

$\lambda$ is large

$\lambda$ is intermediate

$\lambda$ is small

$L = 100$ data sets
$n = 25$
$m = 25$

[Bishop]
Learning curves of bias, variance, and noise

[Bishop]
Bias-variance decomposition: summary

- The noise term is unavoidable.
- The terms we are interested in are bias and variance.
- The approximation-generalization trade-off is seen in the bias-variance decomposition.
Resources

- C. Bishop, “Pattern Recognition and Machine Learning”, Chapter 1.1, 1.3, 3.1, 3.2.