University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chenhao Tan

Lecture 3: Limits of Learning/Bias-Variance Trade-off

Slides adapted from Noah Smith

Administrivia

- HW 1 available on github, due in 10 days
- Piazza etiquette
- numpy vs Python native lists

Learning Objectives

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off

Learning Objectives

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off

https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Interpretability of machine learning

Example definitions:

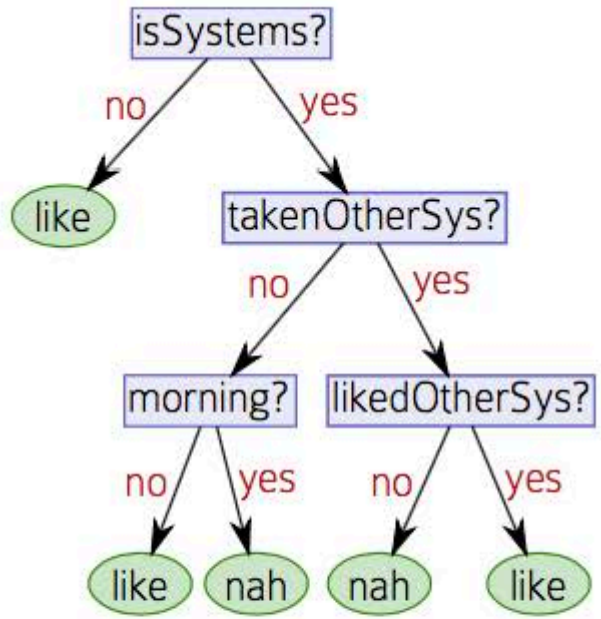
- Can you understand how a prediction is made?
- Can you explain how the model works as a whole?
- Can you make better decisions with assistance of the model?

Interpretability of decision trees

Decision trees are generally considered interpretable.

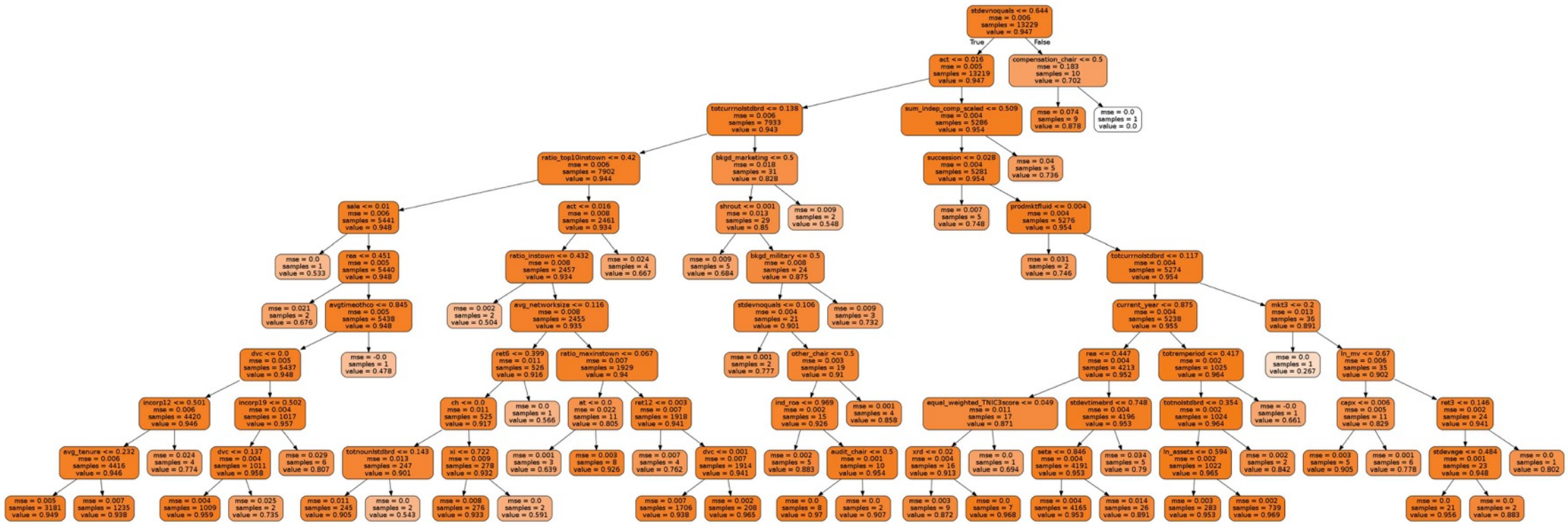
Interpretability of decision trees

Decision trees are generally considered interpretable.



Interpretability of decision trees

Decision trees are generally considered interpretable.
But what about this?



Learning Objectives

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off

Formal definition of supervised learning

- Goal of a learning algorithm:

  Find a function $h : X \rightarrow Y$ from training data S_{train} so that h approximates f

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

but we only have access to training error for a training sample $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

$$\hat{\epsilon} \triangleq \frac{1}{N} \sum_{1}^{N} l(h(\boldsymbol{x}_i), y_i).$$

Minimizing training error is not ideal.

No Free Lunch Theorems

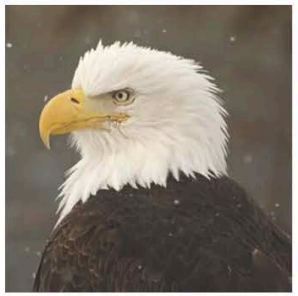
- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.
  Corollary I: there is no single ML algorithm that works for everything.
  Corollary II: every successful ML algorithm makes assumptions.
- No free lunch for search/optimization [Wolpert and Macready, 1997]: All algorithms that search for an extremum of a cost function perform exactly the same when averaged over all possible cost functions.

So how does machine learning work at all?

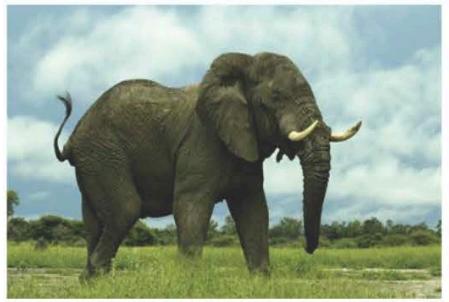
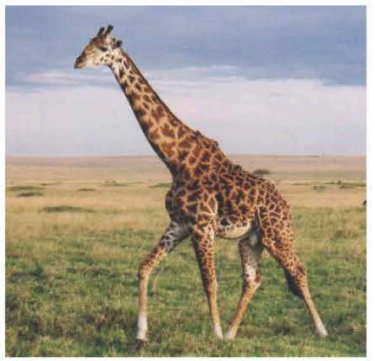
An Exercise, Daume [2017], chapter 2.

Class A



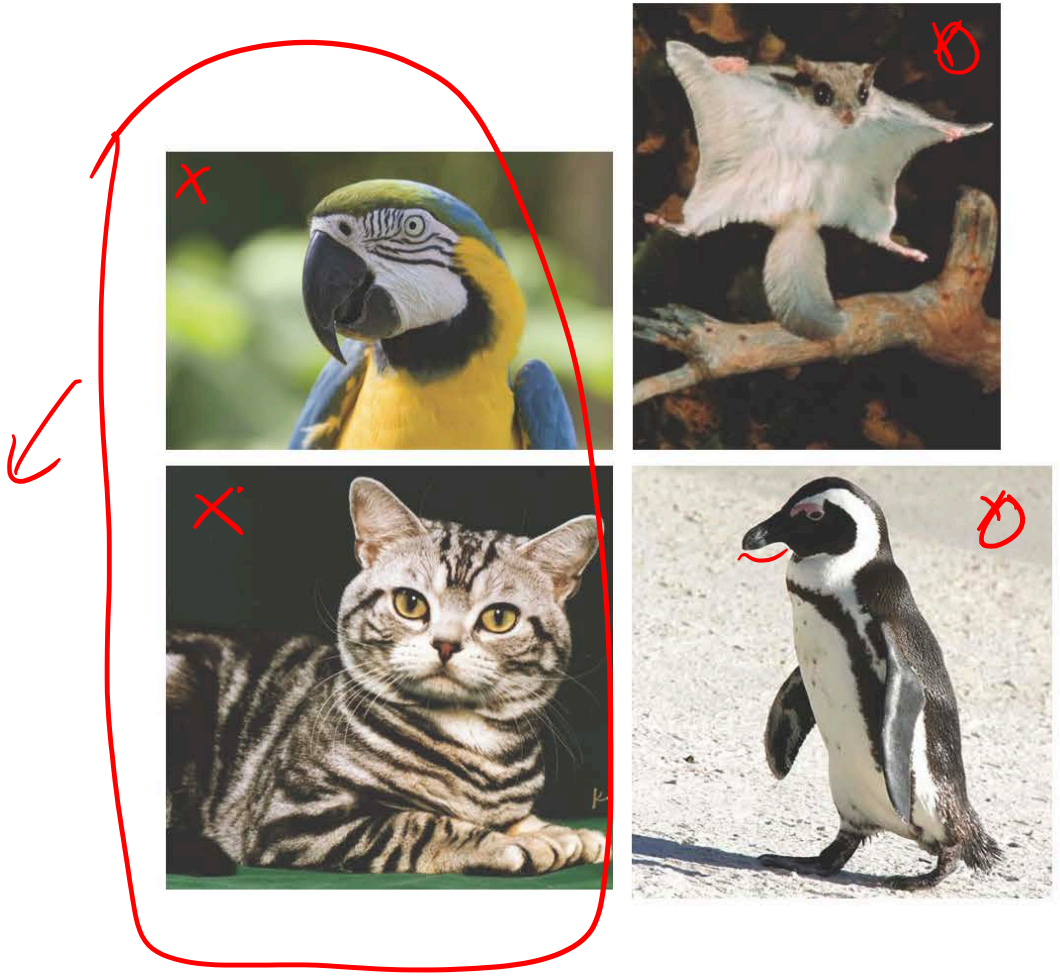



Class B

An Exercise, Daume [2017], chapter 2.

Test



So how does machine learning work at all?

Just as you had a tendency to focus on a certain type of function f, machine learning algorithms correspond to classes of functions (F) and preferences within the class.

So how does machine learning work at all?

Just as you had a tendency to focus on a certain type of function f, machine learning algorithms correspond to classes of functions (F) and preferences within the class.

Inductive bias

So how does machine learning work at all?

Just as you had a tendency to focus on a certain type of function f, machine learning algorithms correspond to classes of functions (F) and preferences within the class.

Inductive bias

We will see an example in decision tree and change our algorithm slightly.

Learning Objectives

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off

Test error

Now, how do we estimate generalization error?

Test error

Now, how do we estimate generalization error?

- training sample S_{train}
- test sample S_{test}

Sample Error vs. Generalization Error

- Generalization error of a hypothesis h for a learning task:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}),y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X=\boldsymbol{x}, Y=y)l(h(\boldsymbol{x}),y),$$

Sample Error vs. Generalization Error

- Generalization error of a hypothesis h for a learning task:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}),y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X=\boldsymbol{x}, Y=y)\, l(h(\boldsymbol{x}),y),$$

- Use test sample to estimate ϵ:

$$\hat{\epsilon}_{\text{test}} \triangleq \frac{1}{|S_{\text{test}}|} \sum_{(\boldsymbol{x}_i,y_i)\in S_{\text{test}}} l(h(\boldsymbol{x}_i),y_i).$$

Training error vs. Test error

- $S_{\text{train}} \rightarrow h$

Training error vs. Test error

- $S_{\text{train}} \rightarrow h$
- Train error $= \hat{\epsilon}_{\text{train}}(h)$
- test error $= \hat{\epsilon}_{\text{test}}(h)$

Training error vs. Test error

- $S_{\text{train}} \rightarrow h$
- Train error $= \hat{\epsilon}_{\text{train}}(h)$
- test error $= \hat{\epsilon}_{\text{test}}(h)$

Never *ever* touch your test data!

A concrete hypothetical example

- Predict flu trends using search data
- X: search data, Y: fraction of population with flu

S train S test

Random: 80% 20%

time: 2016-2017 2018

For all day in (2016, 2018)

$x_d \rightarrow$ {"flu": 2000, "cold": 5000, ...}

$y_d \rightarrow 0.2$

$x_d \rightarrow y_{d+1}$

A concrete hypothetical example

- Predict flu trends using search data
- X: search data, Y: fraction of population with flu
- S_{train} = all data before 2012
- S_{test} = all data in 2012
- What is the problem of generalization error estimation?

[Lazer et al., 2014]

A concrete hypothetical example

- Predict flu trends using search data
- X: search data, Y: fraction of population with flu
- S_{train} = all data before 2012
- S_{test} = all data in 2012
- What is the problem of generalization error estimation?

[Lazer et al., 2014]
Predicting future is hard!

Learning Objectives

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off

Greedily Building a Decision Tree (Binary Features)

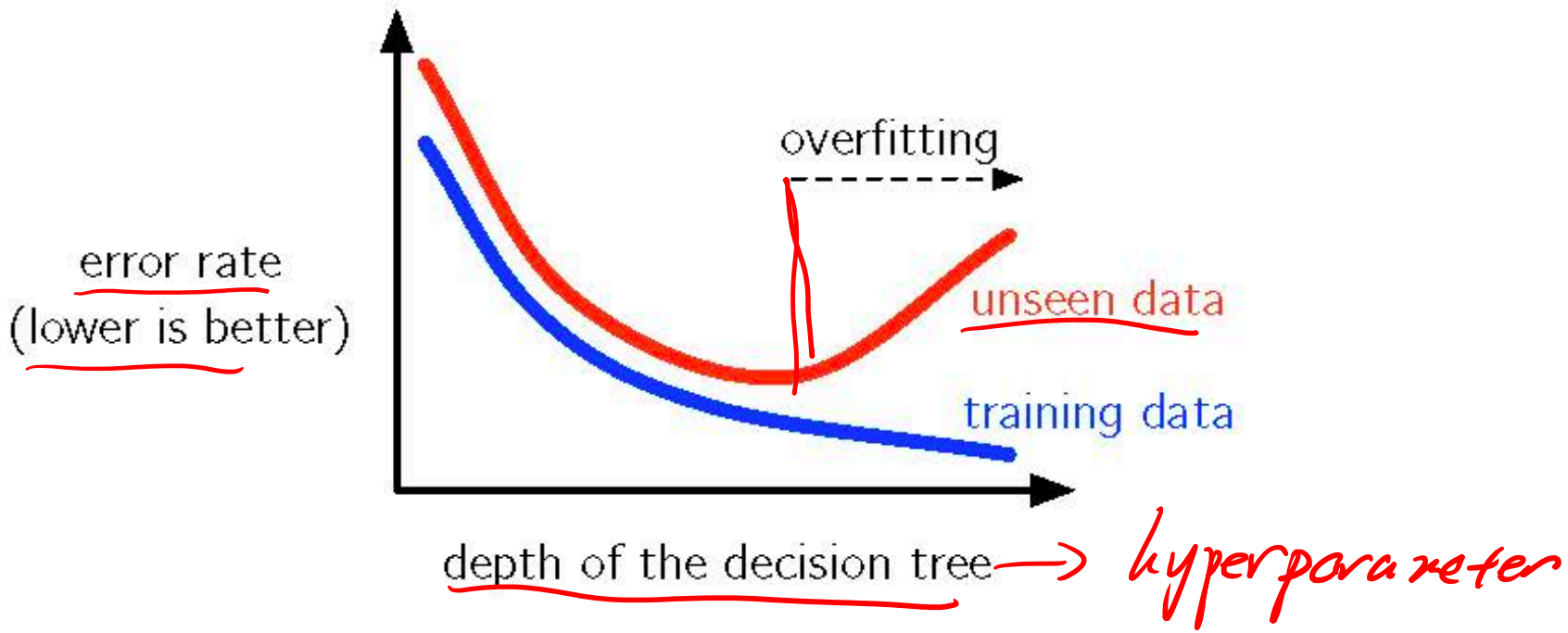
Algorithm: DTREETRAIN
Data: data D, feature set Φ
Result: decision tree
if *all examples in* D *have the same label* y*, or* Φ *is empty and* y *is the best guess*
then
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN(D_0, $\Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN(D_1, $\Phi \setminus \{\phi^*\}$)});
end

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN
Data: data D, feature set Φ
Result: decision tree
if *all examples in* D *have the same label* y*, or* Φ *is empty and* y *is the best guess*
then
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN(D_0, $\Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN(D_1, $\Phi \setminus \{\phi^*\}$)});
end

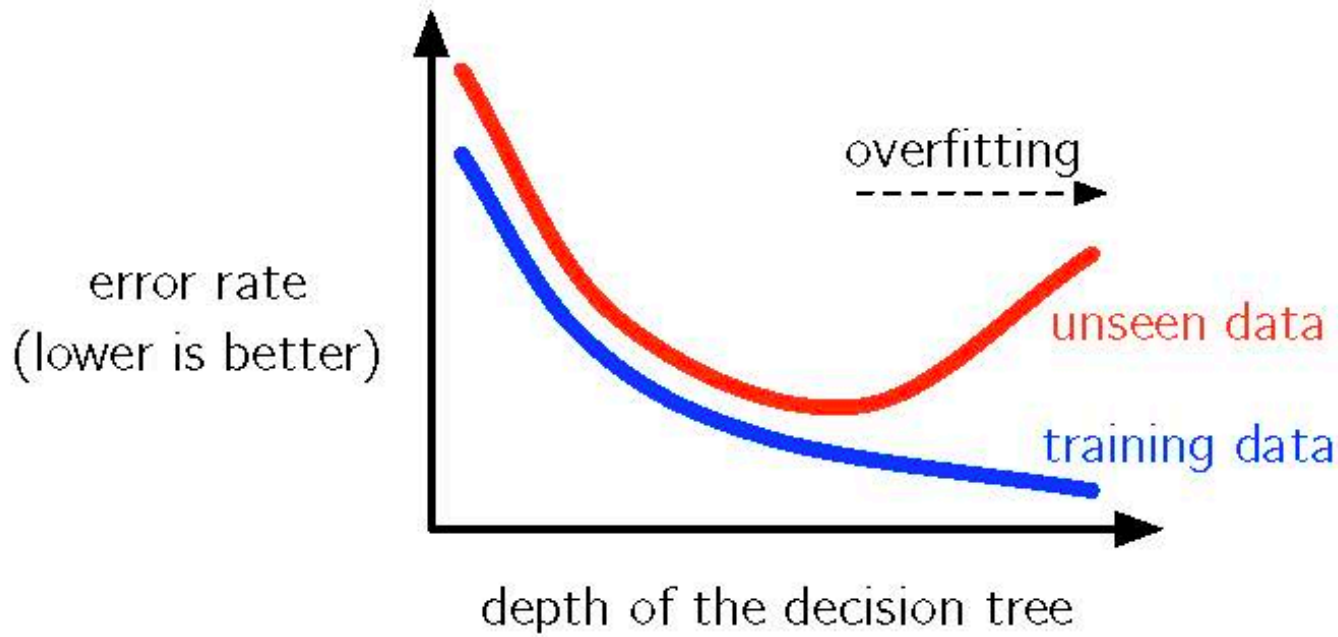
Is this algorithm ideal for optimizing generalization error?

Danger: Overfitting

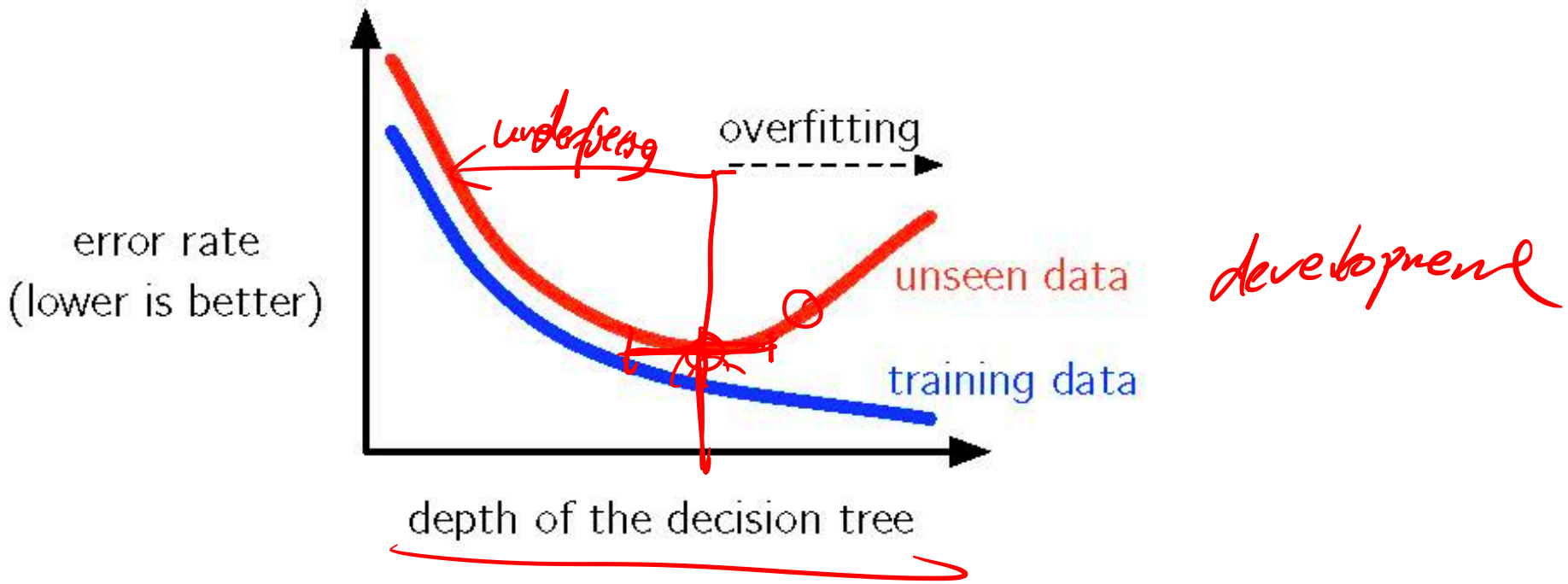


→ hyperparameter

Danger: Overfitting



Danger: Underfitting



Underfitting refers to when your model is not fitting the training data well enough.

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Note that development data is used to guide the training process, while test data is used to estimate generalization error.

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Note that development data is used to guide the training process, while test data is used to estimate generalization error.
Never *ever* touch your test data!

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Note that development data is used to guide the training process, while test data is used to estimate generalization error.
Never *ever* touch your test data!

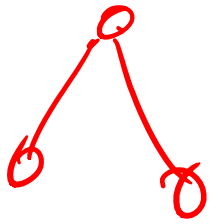
Splitting your data into training/development/test requires careful thinking.
One common starting point is to randomly shuffle examples with an 80%/10%/10% split.

70% (3) 20%

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN
Data: data D, feature set Φ, max depth d_{max}
Result: decision tree
if *all examples in D have the same label y, or Φ is empty and y is the best guess*
then
  return LEAF(y);
else if $d_{max} == 0$: return Leaf(majority(D));
else
  for *each feature ϕ in Φ* **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN(D_0, $\Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN(D_1, $\Phi \setminus \{\phi^*\}$)});
end

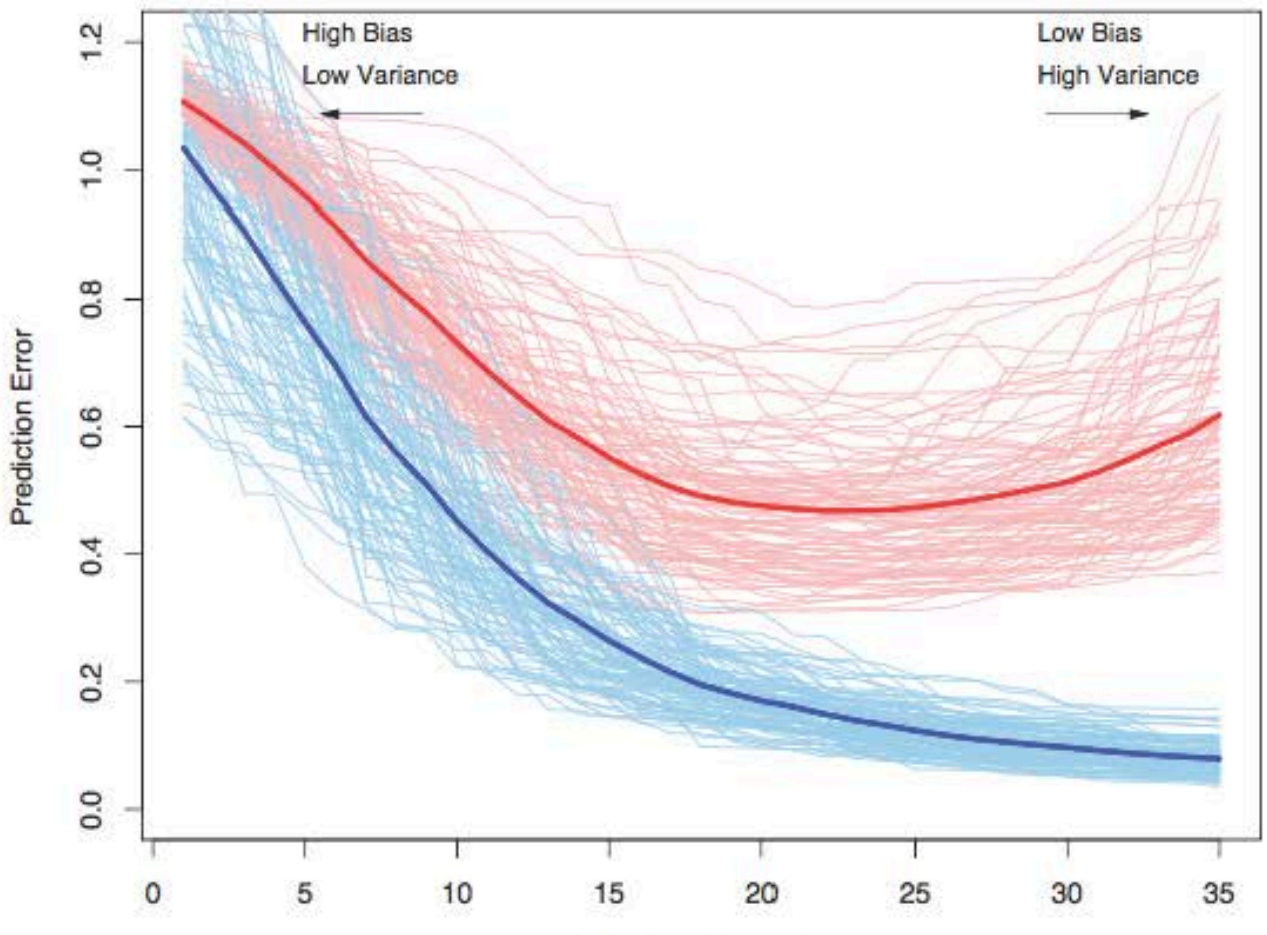
$d_{max}-1$ $d_{max}-1$



Learning Objectives

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off

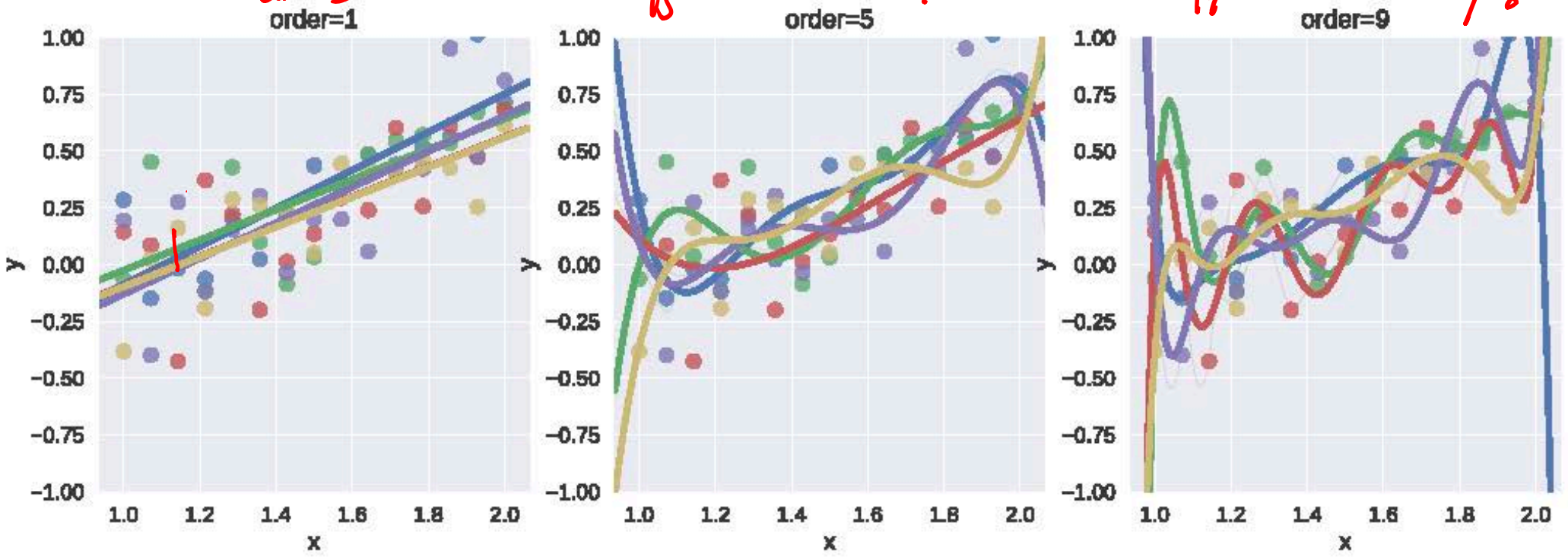
A more general view of overfitting



[Friedman et al., 2001]

Example

$$f(x) = 0.5x^2 - 0.8x + 0.3 + \epsilon, \epsilon \sim \mathcal{N}(0, 1)$$



Bias-Variance Tradeoff

Assume a simple model $y = f(x) + \epsilon$, $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,

Bias-Variance Tradeoff

Assume a simple model $y = f(x) + \epsilon$, $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,

$$\mathrm{Err}(x_0) = \mathrm{E}[(y - h(x_0))^2]$$

$$= E\left((f(x_0)+\varepsilon-h(x_0))^2\right)$$

$$= E\left(\varepsilon^2+(f(x_0)-h(x_0))^2+2\varepsilon(f(x_0)-h(x_0))\right)$$

$$= \sigma_\varepsilon^2 + E\left(f(x_0)-h(x_0)\right)^2$$

$$E\left(f(x_0)-h(x_0)\right)^2$$

$$= E\left(f(x_0)-Eh(x_0)+Eh(x_0)-h(x_0)\right)^2$$

$$= E\left(f(x_0)-Eh(x_0)\right)^2 + E\left(Eh(x_0)-h(x_0)\right)^2 + 2E\left[Eh(x_0)-h(x_0)\right]\left(f(x_0)-Eh(x_0)\right)$$

$$E\left[Eh(x_0)-h(x_0)\right]\left(f(x_0)-Eh(x_0)\right)$$

$$= Eh(x_0)\left(f(x_0)-Eh(x_0)\right) - h(x_0)\left(f(x_0)-Eh(x_0)\right)$$

$$= 0$$

Bias-Variance Tradeoff

Assume a simple model $y = f(x) + \epsilon$, $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,

$$
\begin{aligned}
\mathrm{Err}(x_0) &= \mathrm{E}[(y - h(x_0))^2] \\
&= \sigma_\epsilon^2 + [\mathrm{E}h(x_0) - f(x_0)]^2 + \mathrm{E}[h(x_0) - \mathrm{E}h(x_0)]^2 \\
&= \sigma_\epsilon^2 + \mathrm{Bias}^2(h(x_0)) + \mathrm{Var}(h(x_0)) \\
&= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

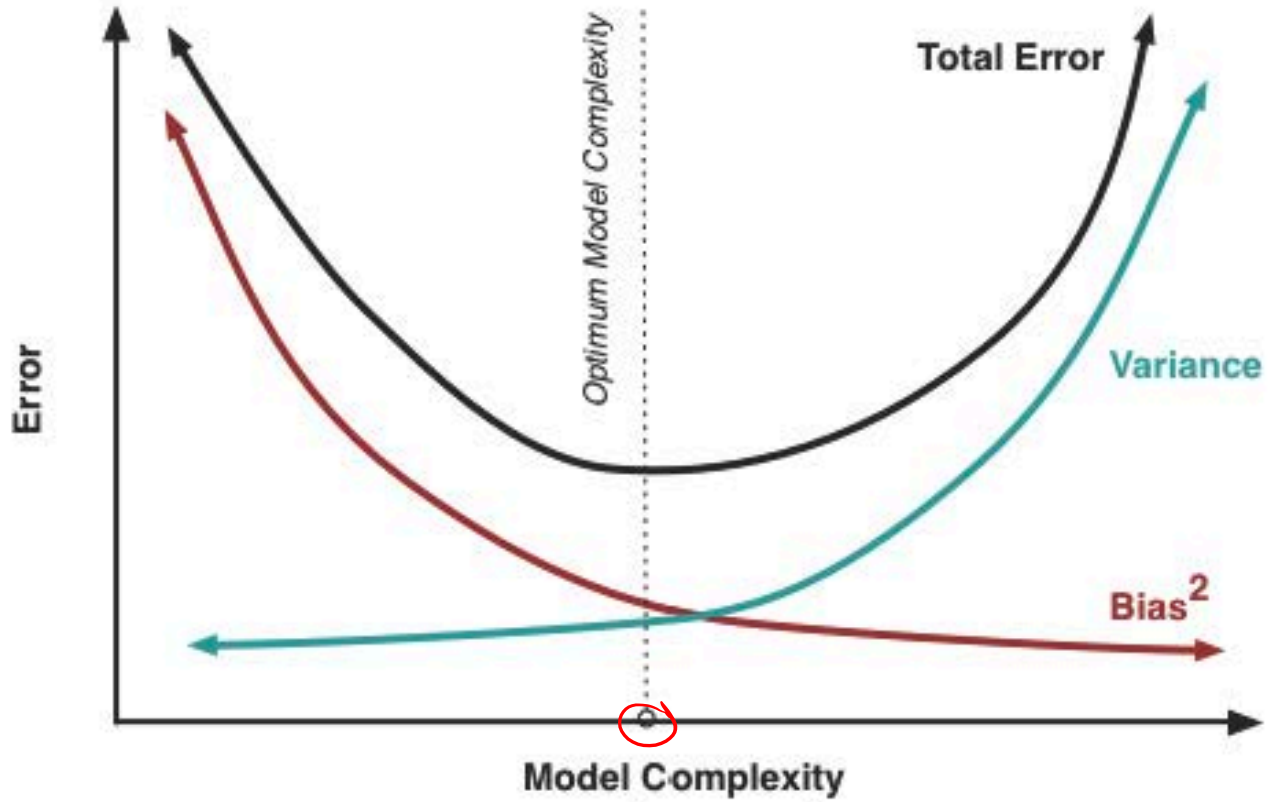
Bias-Variance Tradeoff

$$\text{Generailization error} \quad = \quad \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$$

- $[\mathrm{E}h(x_0) - f(x_0)]^2$, high bias means that even with all training data, the error is still high. Model is not flexible enough to model the true function.
- $\mathrm{E}[h(x_0) - \mathrm{E}h(x_0)]^2$, high variance means that a small variation of training data leads to a great change in the learned model. Model is very sensitive to training data.

Revisit Overfitting

$$\text{Generailization error} = \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$$



http://scott.fortmann-roe.com/docs/BiasVariance.html

Recap

- Understand interpretability of decision trees
- Understand induction bias
- Understand generalization error
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance trade-off