# Machine Learning: Chenhao Tan

University of Colorado Boulder

LECTURE 4

Slides adapted from Chris Ketelsen, Noah Smith

## Logistics

- HW1 available on Github, due in 10 days

## Learning objectives

- Understand supervised learning and induction bias
- Understand hyperparameters, underfitting, and overfitting
- Understand bias-variance tradeoff

## Outline

Supervised learning and induction bias

Hyperparameters, underfitting, and overfitting

Bias-variance tradeoff

## Outline

## Supervised Learning

**Hutzler #571 Banana Slicer**

The only banana slicer you will ever need.

Gourmac's easy-to-use Banana Slicer provides a quick solution to slice a banana uniformly each and every time. Simply press the slicer on a peeled banana and the work is done. Safe, fun and easy for children to use. Kids just love eating bananas with this as their favorite kitchen tool. The Banana Slicer may also be used as a quick way to add healthy bananas to breakfast cereal or to make uniform slices for a fruit salad or ice cream dessert.

Data

$X$

Labels

$Y$

- **Supervised methods** find patterns in **fully observed** data and then try to predict something from **partially observed** data.

## Formal Definitions

- Labels $Y$, e.g., binary labels $y \in \{+1, -1\}$
- Instance space $X$, all the possible instances (based on data representation)
- Target function $f$: $X \to Y$ ($f$ is unknown)

## Formal Definitions

- Labels $Y$, e.g., binary labels $y \in \{+1, -1\}$
- Instance space $X$, all the possible instances (based on data representation)
- Target function $f$: $X \to Y$ ($f$ is unknown)
- Example/instance $(\boldsymbol{x}, y)$
- Training data $S_{\text{train}}$: collection of examples observed by the algorithm

## Formal Definitions

- Goal of a learning algorithm:

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

## Supervised learning in a nutshell

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

## Supervised learning in a nutshell

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

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

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

## Supervised learning in a nutshell

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

Given a 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 $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

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

## Supervised learning in a nutshell

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

Given a 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 $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

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

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.

## 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 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.

## Test error

Now, how do we estimate generalization error?

## Test error

Now, how do we estimate generalization error?

- training sample $S_{\text{train}}$
- test sample $S_{\text{test}}$

## 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 $\epsilon$:

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

## 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

## A concrete hypothetical example

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

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

## Outline

## Greedily Building a Decision Tree (Binary Features)

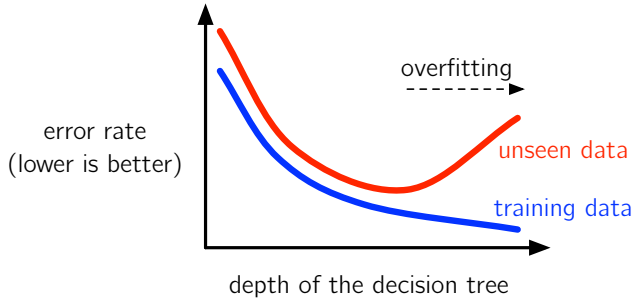
**Algorithm:** DTREETRAIN
**Data:** data $D$, feature set $\Phi$
**Result:** decision tree
**if** *all examples in* $D$ *have the same label* $y$*, or* $\Phi$ *is empty and* $y$ *is the best guess* **then**
  return LEAF($y$);
**else**
  **for** *each feature* $\phi$ *in* $\Phi$ **do**
    partition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

## Greedily Building a Decision Tree (Binary Features)

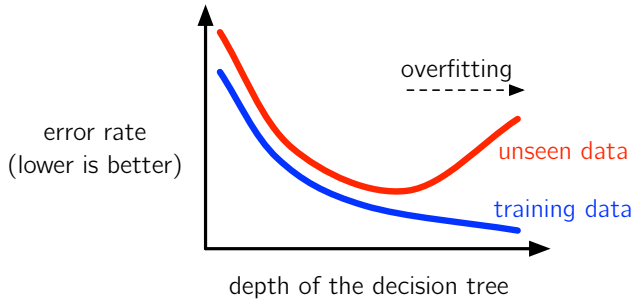
**Algorithm:** DTreeTrain
**Data:** data $D$, feature set $\Phi$
**Result:** decision tree
**if** *all examples in* $D$ *have the same label* $y$*, or* $\Phi$ *is empty and* $y$ *is the best guess* **then**
  return Leaf($y$);
**else**
  **for** *each feature* $\phi$ *in* $\Phi$ **do**
    partition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



## Danger: Overfitting




*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:

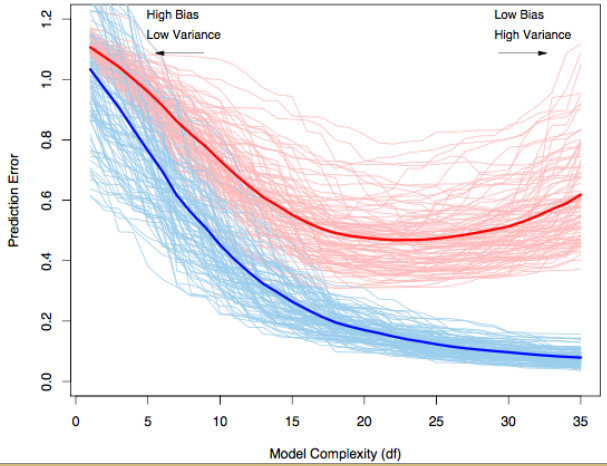
- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

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.

## Outline

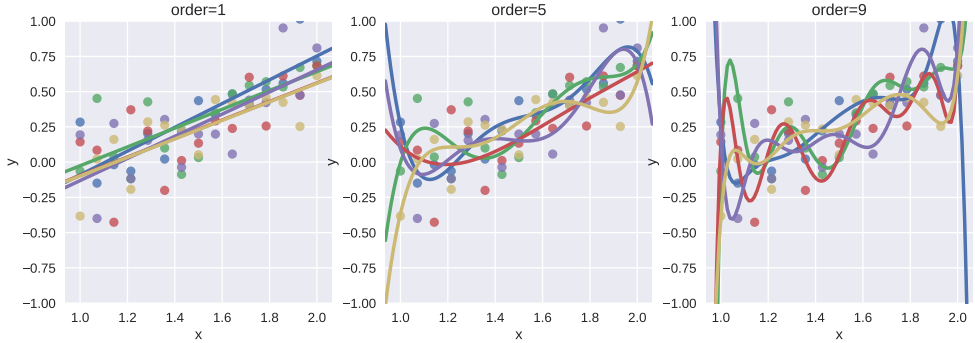
## A more general view of overfitting




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

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

## Bias-Variance Tradeoff

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

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

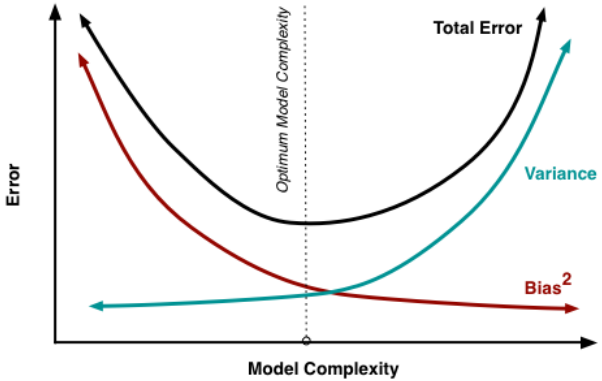
## Bias-Variance Tradeoff

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

- $[\mathrm{E}h(x_0) - f(x_0)]^2$, 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

## References


Hal Daume. *A Course in Machine Learning (v0.9)*. Self-published at `http://ciml.info/`, 2017.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. The parable of Google Flu: traps in big data analysis. *Science*, 343(6176):1203–1205, 2014.

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7): 1341–1390, 1996.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.