Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 4

Slides adapted from Chenhao Tan, Chris Ketelsen, Noah Smith

Logistics

- HW1 is available on Github, due on Feb. 5th

Learning objectives

- Revisit supervised learning
- Hyperparameters and overfitting
- Bias-variance tradeoff

Outline

(Review) Supervised learning

Hyperparameters, underfitting, and overfitting

Bias-variance tradeoff

Outline

(Review) Supervised learning

Hyperparameters, underfitting, and overfitting

Bias-variance tradeoff

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.

Notations

- l: Loss function
- Labels Y, e.g., $y \in \{+1, -1\}$, $y \in \mathbb{R}$
- Input Data X
- Target function we wish to learn f: $X \to Y$ (f is unknown)
- Function a machine learning model learns h: $X \to Y$
- A training example $(\boldsymbol{x}, y) \in (X, Y)$
- Training data S_{train}: collection of examples observed during training

Formal Definitions

- Goal of a learning algorithm:

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

Supervised learning

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

Supervised learning

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

Given a loss function l and examples $(\boldsymbol{x}, y) \sim D$, expected loss/generalization error of hypothesis h is defined as:

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

Supervised learning

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

Given a loss function l and examples $(\boldsymbol{x}, y) \sim D$, expected loss/generalization error of hypothesis h is defined as:

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

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

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

Given a loss function l and examples $(\boldsymbol{x}, y) \sim D$, expected loss/generalization error of hypothesis h is defined as:

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

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

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

We minimizing training error in practice, since D is unknown.

Test error

Now, how do we estimate generalization error when evlauating the trained models?

Test error

Now, how do we estimate generalization error when evlauating the trained models?
Answer: Build a separate test set

- training samples S_{train}
- test samples S_{test}

Sample Error vs. Generalization Error

- 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

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

Outline

(Review) Supervised learning

Hyperparameters, underfitting, and overfitting

Bias-variance tradeoff

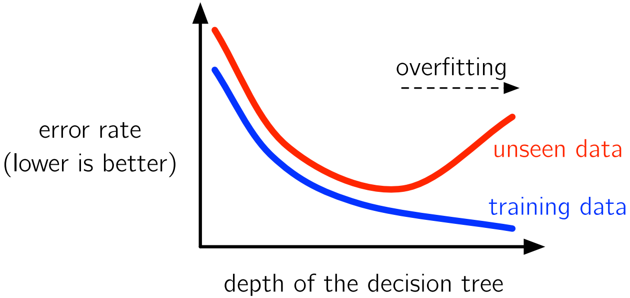
Greedily Building a Decision Tree (Binary Features)

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* **then**
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ;
    calculate $IG(D,\phi)$
  end
  if $IG(D,\phi)=0$ *for all* ϕ **then**
    terminate
  else
    Choose ϕ^* be the feature with the largest $IG(D,\phi)$;
    Create NODE(ϕ^*);
    DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$);
    DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$);
  end
end

What happen if the tree becomes too deep?

Danger: Overfitting



Detecting Overfitting

How do you draw the red curve on the preceding slide if you cannot touch the test data?

Detecting Overfitting

How do you draw the red curve on the preceding slide if you cannot touch the test data?

Solution: hold some out another set of examples as **development data.**

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

Detecting Overfitting

How do you draw the red curve on the preceding slide if you cannot touch the test data?

Solution: hold some out another set of examples as **development data.**

- Decision tree max depth is an example of a **hyperparameter**
- “Used the development data to **tune** the max-depth hyperparameter.”

Development data is used to guide the training process
Test data is used to estimate generalization error.

Detecting Overfitting

How do you draw the red curve on the preceding slide if you cannot touch the test data?

Solution: hold some out another set of examples as **development data.**

- Decision tree max depth is an example of a **hyperparameter**
- “Used the development data to **tune** the max-depth hyperparameter.”

Development data is used to guide the training process
Test data is used to estimate generalization error.
Never *ever* touch your test data!

Detecting Overfitting

How do you draw the red curve on the preceding slide if you cannot touch the test data?

Solution: hold some out another set of examples as **development data**.

- Decision tree max depth is an example of a **hyperparameter**
- “Used the development data to **tune** the max-depth hyperparameter.”

Development data is used to guide the training process
Test data is used to estimate generalization error.
Never *ever* touch your test data!

One common strategy: randomly shuffle examples with an 80%/10%/10% split.

Outline

(Review) Supervised learning

Hyperparameters, underfitting, and overfitting

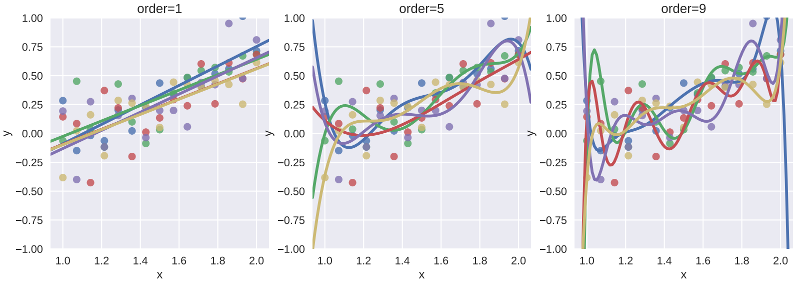
Bias-variance tradeoff

Example: Polynomial Fitting

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

w_i: learnable parameters

- order 1: $y = w_1 x + w_0$
- order 5: $y = w_5 x^5 + \ldots w_1 x + w_0$
- order 9: $y = w_9 x^9 + \ldots w_1 x + w_0$



Bias-Variance Tradeoff

Assume we use

- a model with noise ϵ i.e., $y = f(x) + \epsilon$, $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,
- a squared loss i.e., $l = (y - h(x))^2$
- traing/test data (x, y) is sampled from data generating distribution D i.e., $(x, y) \sim D$

Bias-Variance Tradeoff

Assume we use

- a model with noise ϵ i.e., $y = f(x) + \epsilon$, $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,
- a squared loss i.e., $l = (y - h(x))^2$
- traing/test data (x, y) is sampled from data generating distribution D i.e., $(x, y) \sim D$

$$\epsilon_{\text{general}} \triangleq \mathrm{E}[(y - h(x))^2]$$

Bias-Variance Tradeoff

Assume we use

- a model with noise ϵ i.e., $y = f(x) + \epsilon$, $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,
- a squared loss i.e., $l = (y - h(x))^2$
- traing/test data (x, y) is sampled from data generating distribution D i.e., $(x, y) \sim D$

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

Bias-Variance Tradeoff

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

- $[\mathrm{E}h(x) - f(x)]^2$, high bias means that even with all training data, the error is still high. Model is not flexible enough to model the true function.
- $\mathrm{E}[h(x) - \mathrm{E}h(x)]^2$, 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.

Visualization of Bias and Variance

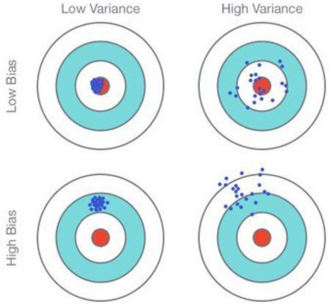
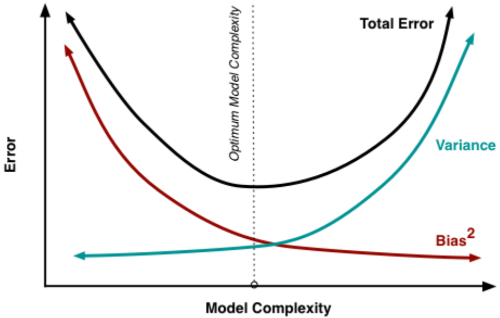


Fig. 1: Graphical Illustration of bias-variance trade-off , Source: Scott Fortmann-Roe., Understanding Bias-Variance Trade-off

https://medium.com/@mp32445/understanding-bias-variance-tradeoff-ca59a22e2a83

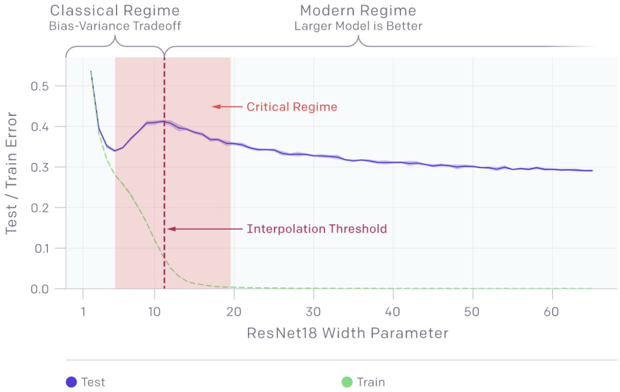
Revisit Overfitting

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



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

(Extended Reading) Beyond the classical bias-variance curve



https://openai.com/blog/deep-double-descent/

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.