Limits and Evaluation I

Dr. Alex Williams

August 28, 2020

THE UNIVERSITY OF TENNESSEE KNOXVILLE

COSC 425: Introduction to Machine Learning
Fall 2020 (CRN: 44874)

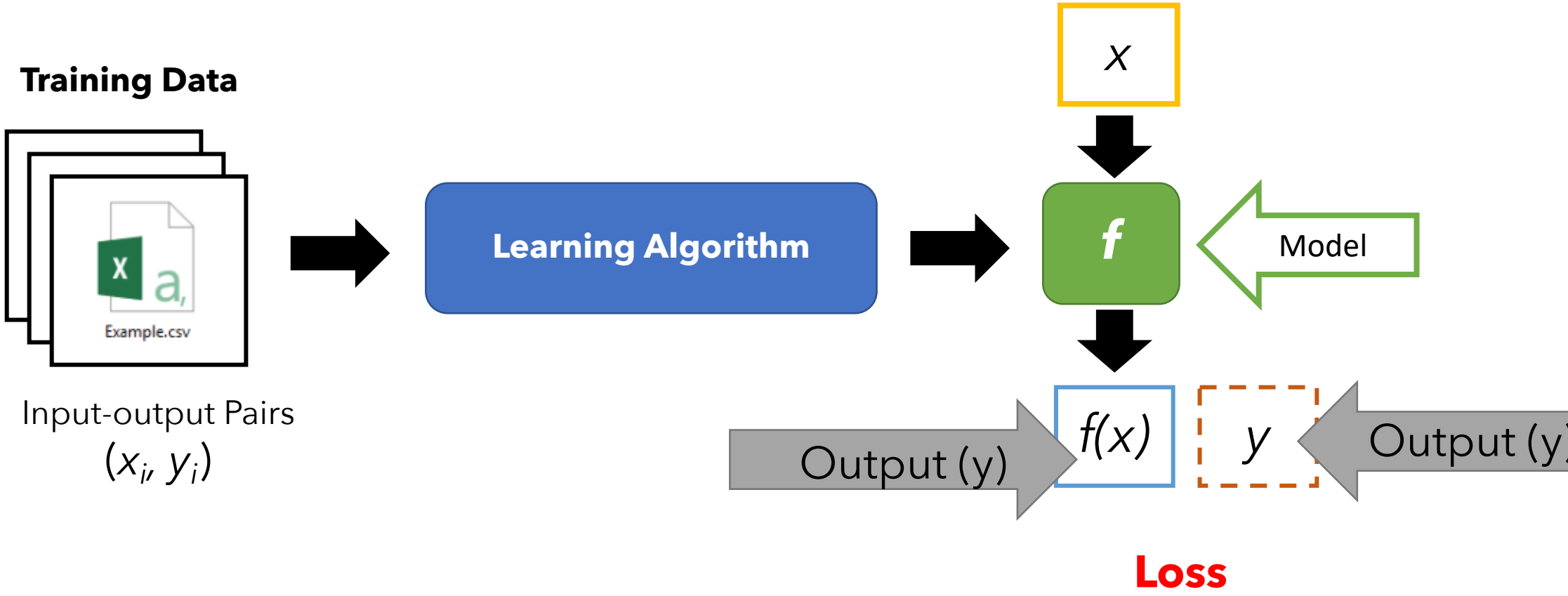
Today's Agenda

We will address:

1. What's the formal definition of performance?
2. How well can we possibly generalize?

1. What's the formal definition of performance?

Supervised Learning: Performance



Supervised Learning: Performance

To measure how well our learning algorithm is performing, we need to setup a loss function $l(y, \hat{y})$ based on our goal of learning, i.e. between *predicted* and *observed*.

squared loss	$l(y, \hat{y}) = (y - \hat{y})^2$
absolute loss	$l(y, \hat{y}) = \lvert y - \hat{y} \rvert$
zero/one loss	$l(y, \hat{y}) = 1_{y \neq \hat{y}}$

Supervised Learning: Assumptions

Supervised learning "models" assume that a probability distribution $\mathbf{D}$ exists over $(\boldsymbol{x}, \boldsymbol{y})$ called the data generating distribution.

Training Data

Input-output Pairs

(x_i, y_i)

Note: The data generating distribution does not take any special form. It simply gives high probability to reasonable $(\boldsymbol{x}, \boldsymbol{y})$ pairs, and low probabilities to unreasonable $(\boldsymbol{x}, \boldsymbol{y})$ pairs.

We assume that the training data is a random sample of $(\boldsymbol{x}, \boldsymbol{y})$ pairs drawn from $\mathbf{D}$.

Bayes Optimal Classifier

Suppose that we have a magic function **computeD** that takes two inputs, ***x*** and ***y***, and returned the probability of ***(x, y)*** pair under **D**.

Then, the Bayes optimal classifier is the classifier that, for any test input x′, returns y′ that maximizes ***computeD(x', y')***:

$$f^{(BO)}(x') = \arg\max_{y' \in Y} D(x', y')$$

The classifier is optimal because, of all possible deterministic classifiers, it achieves the smallest zero/one error.

Intuition: Classify (i.e. by choosing a label) based on the highest probability.

- Minimizes the likelihood that an error will be made in post-training.

Supervised Learning: Performance Formulation

Based on the training data, we need to induce a function ***f*** that maps new inputs ***x'*** to a corresponding prediction ***y'***, such that the function does as well as possible on future examples that are also drawn from D.

$$\epsilon = \mathop{\mathbb{E}}_{(x,y)\sim D}[l(y, f(x))] = \sum_{(x,y)} D(x, y)\ l(y, f(x))$$

All we have are examples from D. We don't know what D really is!

Supervised Learning: Performance Formulation

We can estimate future error by computing an error from training data:

$$\hat{\epsilon} = \frac{1}{N} \sum_{n=1}^{N} l(y_n, f(x_n))$$

The fundamental difficulty in machine learning is that we have access to our training error $\hat{\epsilon}$, but we truly only care about minimizing error on future instances.

In other words:

- We want our function **generalize** beyond the data we have at hand.

Defining Induction

Machine Learning Phrasing

Given a loss function $\mathbb{L}$ and a sample $\mathbf{D}$ from some unknown distribution, you must compute a function $\boldsymbol{f}$ that has low expected error $\hat{\epsilon}$ over $\mathbf{D}$ with respect to $\mathbb{L}$.

Alternative Phrasing:

Induction is the process of reaching a general conclusion from specific examples.

2. How well can we possibly generalize?

Generalization: What's Learnable?

Learning might fail because the inductive bias of the learning algorithm is **too "far away" from the concept of being learned**.

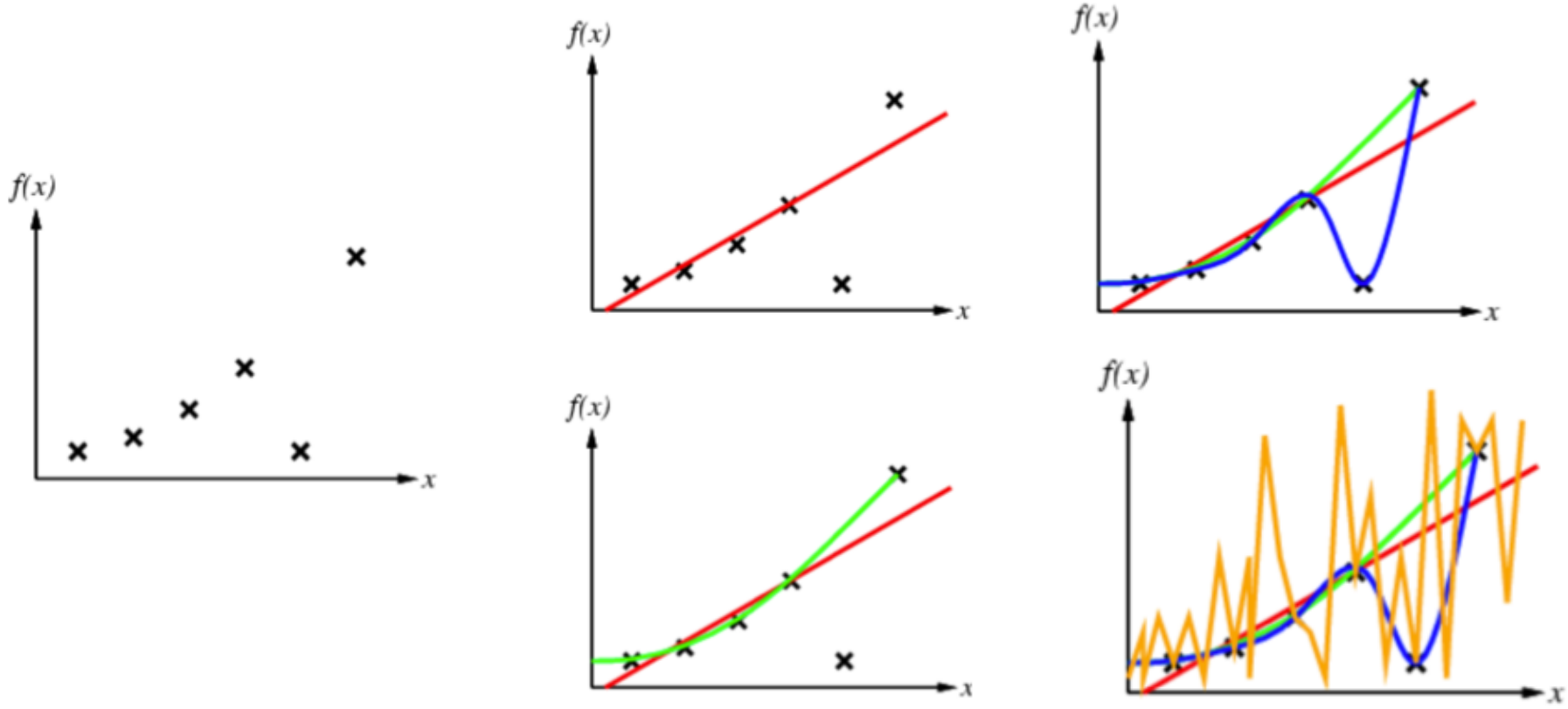
→ Your data might reflect "bird" / "no bird".
→ The algorithm may be learning something else entirely.

Learning can fail because of **bad data**.

- Input features can be noisy, e.g. sensor failures.
- Output labels can be noisy, e.g. annotator failures.
- Feature space may be insufficient, e.g. information missing.
- Output label can be ambiguous, e.g. fake news / not fake news.

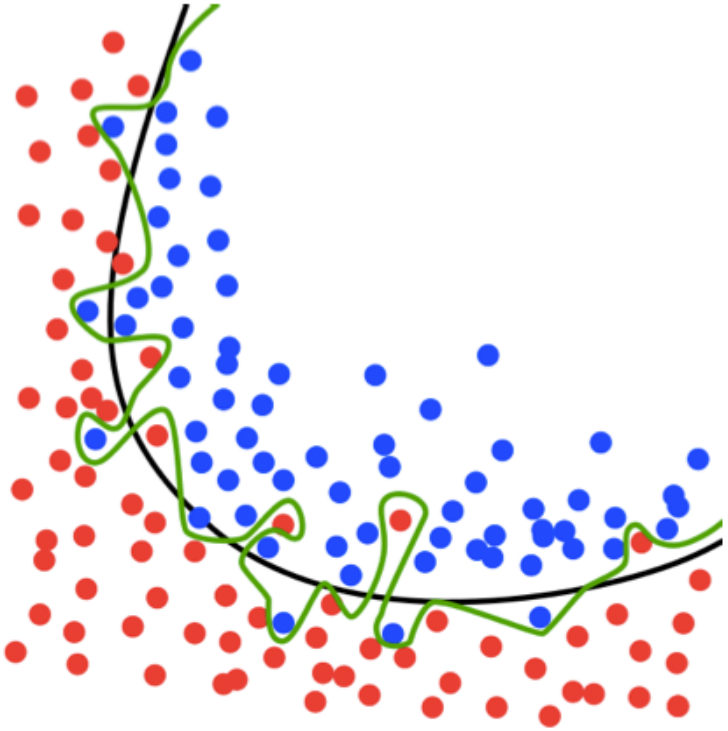
Generalization: Model Selection

Find a function $\mathbf{h}$ that fits $\mathbf{f}$ at instances $\mathbf{x}$.



Generalization: Model Selection

Degrees of freedom (i.e. Features) → Training performance improves.



Generalization: Model Selection

Many learning algorithms are based on the notion of "modelling".

→ i.e. We have some formal "model" of our data.
→ A decision tree is a model of our data.

Each method tells us what we can learn and what assumptions we make.

Models have **parameters** that are a lot like knobs / settings.

- The learning algorithm's job is to find out what parameters work best in a given model, e.g. which questions to ask.

Models also have **hyper-parameters** that are controlled manually.
→ e.g., max depth in a decision tree.

Generalization: Trading Off Bias & Variance

Suppose we're trying to learn a regression function $\mathbf{f(x)}$ to approximate the true function $\boldsymbol{y}$.

We can prove that the expectation generalization error (based on squared loss) is a combination of noise, bias, and variance.

$$E[(y - f(x))^2] = Noise^2 + Bias^2 + Variance$$

https://en.wikipedia.org/wiki/Bias-variance_tradeoff

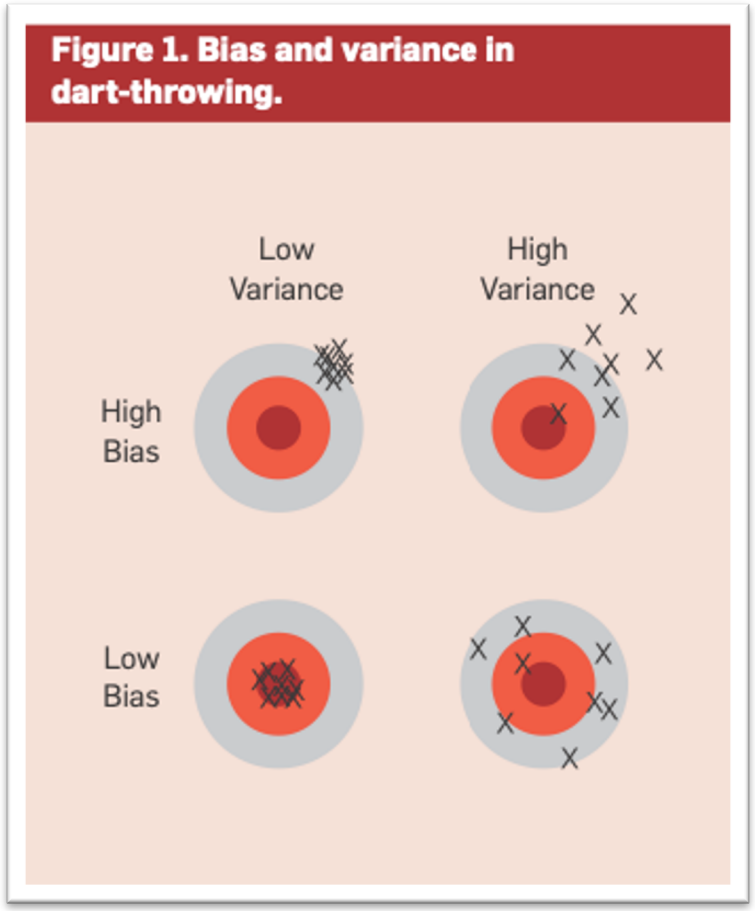
Generalization: Trading Off Bias & Variance

Noise: Irreducible Error
→ Our model will capture some level of irreducible error.

Bias: Tendency to consistency learn the wrong thing.
→ Our model can make erroneous assumptions about our data which may cause our model to be overly simplistic (Underfit!)

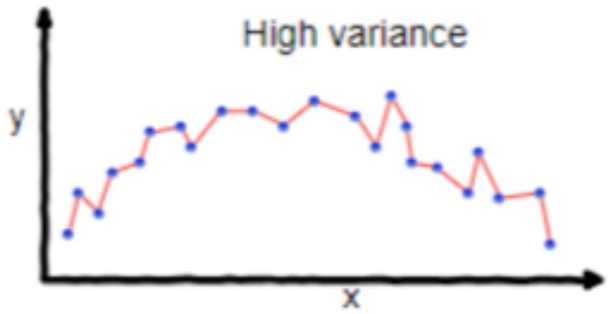
Variance: Tendency to consistently learn random things.
→ Our model can be erroneously sensitive to small changes in our data which may cause our model to map to its patterns too tightly. (Overfit!)

Generalization: Trading Off Bias & Variance

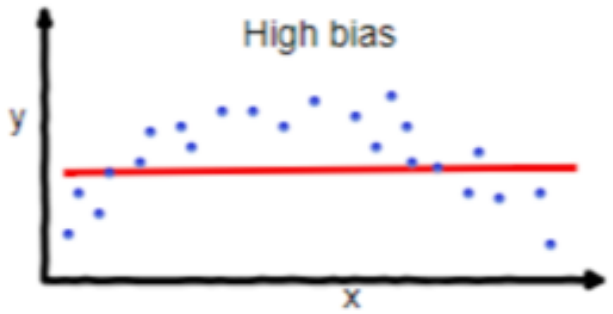


A Few Useful Things to Know about Machine Learning.
Domingos, 2012.

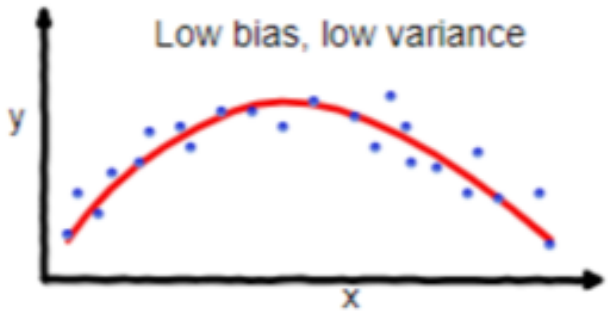
Generalization: Trading Off Bias & Variance



overfitting



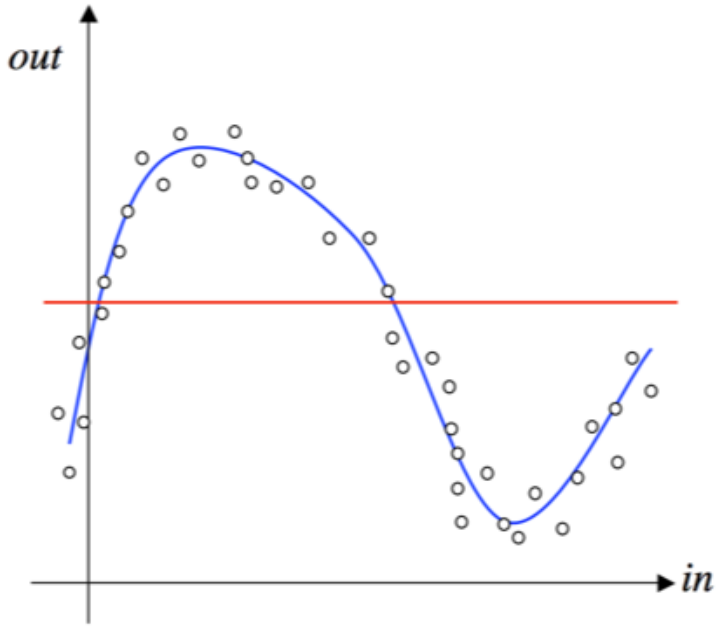
underfitting



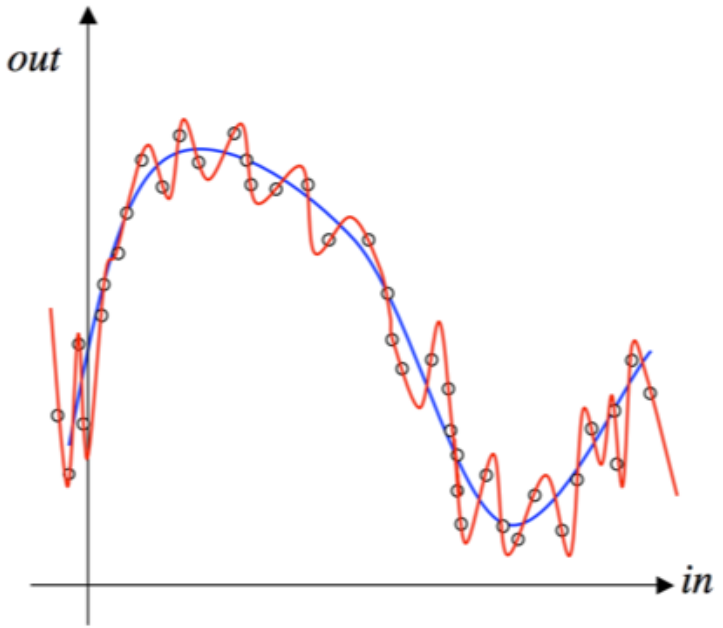
Good balance

https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Generalization: Trading Off Bias & Variance



Ignore the data ⇒
Big approximation error (high bias)
No variation between data sets (no variance)



Fit every data point ⇒
No approximation error (zero bias)
Variation between data sets (high variance)

https://www.cs.bham.ac.uk/~jxb/INC/l9.pdf

Decision Trees: Bias and Variance

Consider an **empty** vs **full** decision tree.

Empty Tree → Same prediction regardless of its input.
(It doesn't ask questions!)

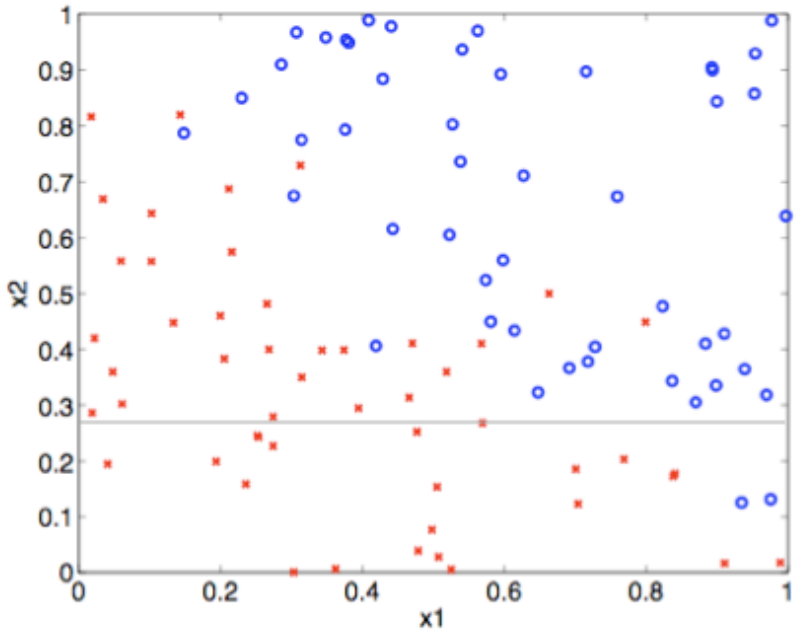
Full Tree → Will always make correct predictions.
(Assuming every leaf has 0 or 1 examples assigned)

How would these trees perform on future data?

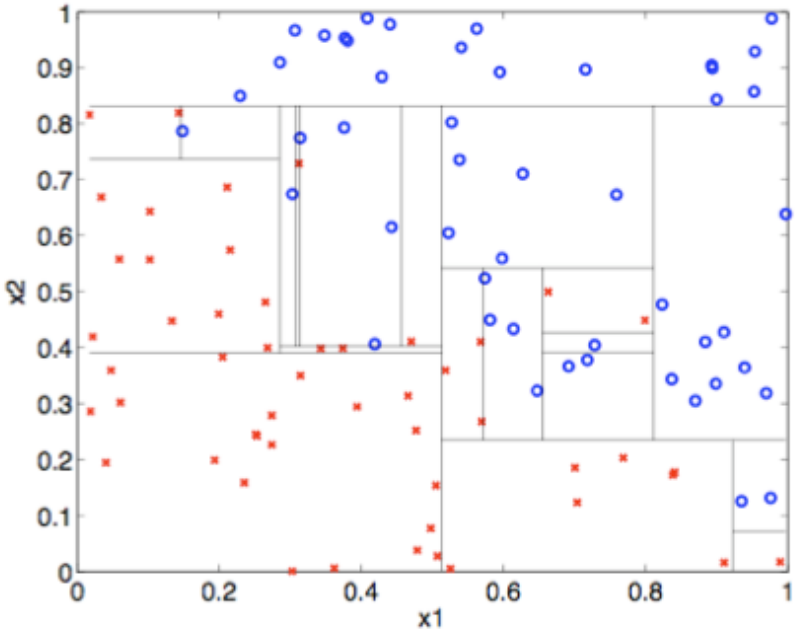
Empty Tree → We expect it to make the same error on all future examples, (i.e. high bias, low variance)

Full Tree→ For examples identical to training, it will do the right thing. Otherwise, significantly large error (i.e. low bias, high variance)

Decision Trees: Bias and Variance



High bias, Low variance



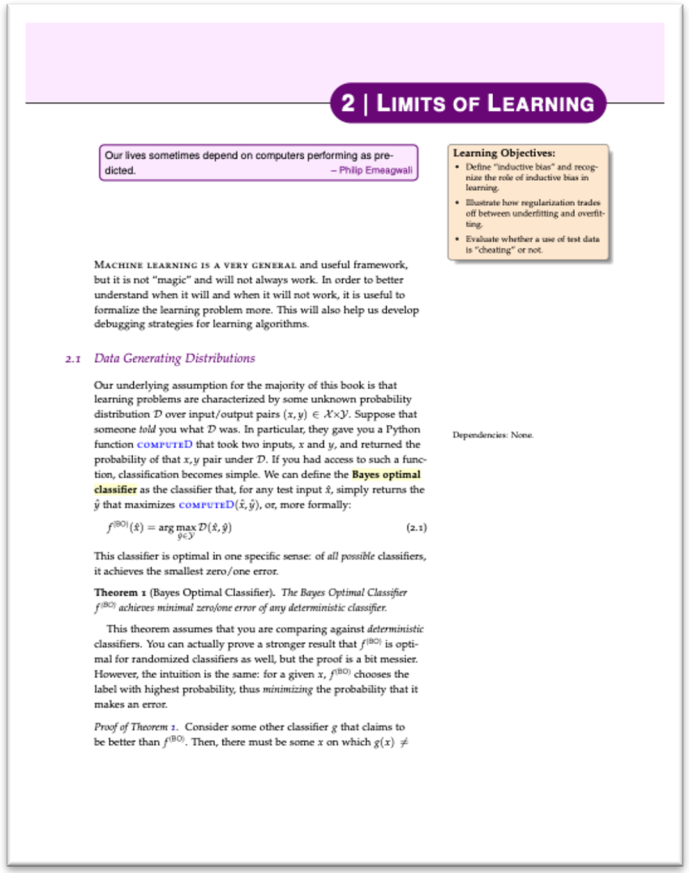
Low bias, high variance

Today's Agenda

We have addressed:

1. What's the formal definition of performance?
2. How well can we possibly generalize?

Reading



2 | LIMITS OF LEARNING

Our lives sometimes depend on computers performing as predicted. – Philip Emeagwali

Learning Objectives:
- Define "inductive bias" and recognize the role of inductive bias in learning.
- Illustrate how regularization trades off between underfitting and overfitting.
- Evaluate whether a use of test data is "cheating" or not.

MACHINE LEARNING IS A VERY GENERAL and useful framework, but it is not "magic" and will not always work. In order to better understand when it will and when it will not work, it is useful to formalize the learning problem more. This will also help us develop debugging strategies for learning algorithms.

2.1 Data Generating Distributions

Our underlying assumption for the majority of this book is that learning problems are characterized by some unknown probability distribution $\mathcal{D}$ over input/output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Suppose that someone *told* you what $\mathcal{D}$ was. In particular, they gave you a Python function COMPUTED that took two inputs, x and y, and returned the probability of that x, y pair under $\mathcal{D}$. If you had access to such a function, classification becomes simple. We can define the **Bayes optimal classifier** as the classifier that, for any test input $\hat{x}$, simply returns the $\hat{y}$ that maximizes COMPUTED$(\hat{x}, \hat{y})$, or, more formally:

$$f^{(\mathrm{BO})}(\hat{x}) = \arg\max_{\hat{y} \in \mathcal{Y}} \mathcal{D}(\hat{x}, \hat{y}) \qquad (2.1)$$

Dependencies: None.

This classifier is optimal in one specific sense: of *all possible* classifiers, it achieves the smallest zero/one error.

Theorem 1 (Bayes Optimal Classifier). *The Bayes Optimal Classifier $f^{(\mathrm{BO})}$ achieves minimal zero/one error of any deterministic classifier.*

This theorem assumes that you are comparing against *deterministic* classifiers. You can actually prove a stronger result that $f^{(\mathrm{BO})}$ is optimal for randomized classifiers as well, but the proof is a bit messier. However, the intuition is the same: for a given x, $f^{(\mathrm{BO})}$ chooses the label with highest probability, thus *minimizing* the probability that it makes an error.

Proof of Theorem 1. Consider some other classifier g that claims to be better than $f^{(\mathrm{BO})}$. Then, there must be some x on which $g(x) \neq$

Daumé, Sec 2.1 – 2.4.

review articles

DOI:10.1145/2347736.2347755

Tapping into the "folk knowledge" needed to advance machine learning applications.

BY PEDRO DOMINGOS

A Few Useful Things to Know About Machine Learning

MACHINE LEARNING SYSTEMS automatically learn programs from data. This is often a very attractive alternative to manually constructing them, and in the last decade the use of machine learning has spread rapidly throughout computer science and beyond. Machine learning is used in Web search, spam filters, recommender systems, ad placement, credit scoring, fraud detection, stock trading, drug design, and many other applications. A recent report from the McKinsey Global Institute asserts that machine learning (a.k.a. data mining or predictive analytics) will be the driver of the next big wave of innovation.[15] Several fine textbooks are available to interested practitioners and researchers (for example, Mitchell[16] and Witten et al.[24]). However, much of the "folk knowledge" that is needed to successfully develop machine learning applications is not readily available in them. As a result, many machine learning projects take much longer than necessary or wind up producing less-than-ideal results. Yet much of this folk knowledge is fairly easy to communicate. This is the purpose of this article.

key insights

- Machine learning algorithms can figure out how to perform important tasks by generalizing from examples. This is often feasible and cost-effective where manual programming is not. As more data becomes available, more ambitious problems can be tackled.
- Machine learning is widely used in computer science and other fields. However, developing successful machine learning applications requires a substantial amount of "black art" that is difficult to find in textbooks.
- This article summarizes 12 key lessons that machine learning researchers and practitioners have learned. These include pitfalls to avoid, important issues to focus on, and answers to common questions.

78 COMMUNICATIONS OF THE ACM | OCTOBER 2012 | VOL. 55 | NO. 10

Optional: Domingos, 2012.

Next Time

We will address:

1. What should we do to improve generalization?
2. What are common performance metrics in machine learning?