CPSC 540: Machine Learning

Fundamentals of Learning

Winter 2018

Admin

- Auditing/registration forms:
  - Submit then at end of class, pick them up at end of next class.
  - I need your prereq form before I'll sign your registration form.
- Website/Piazza:
  - http://www.cs.ubc.ca/~schmidtm/Courses/540-W18
  - https://piazza.com/ubc.ca/winterterm22017/cpsc540
- Tutorials: start Monday after class (no need to formally register).
- Assignment 1 due Friday (pushed back two days).
  - Sign up for CS account so you can hand it in.

Supervised Learning Notation

- We are given training data where we know labels:

X =

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…
0	0.7	0	0.3	0	0	
0.3	0.7	0	0.6	0	0.01	
0	0	0	0.8	0	0	
0.3	0.7	1.2	0	0.10	0.01	
0.3	0	1.2	0.3	0.10	0.01	

y =

Sick?
1
1
0
1
1

- But the goal is to do well on any possible testing data:

$\tilde{X}$=

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…
0.5	0	1	0.6	2	1	
0	0.7	0	1	0	0	
3	1	0	0.5	0	0	

$\tilde{y}$=

Sick?
?
?
?

"Test Set" vs. "Test Error"

- Formally, the "test error" is the expected error of our model:

$$\mathbb{E}[|\hat{y}^i - \tilde{y}^i|]$$

  - Here I'm using absolute error between predictions and true labels.
    - But you could use squared error on other losses.
  - The expectation is taken over distribution of test examples.
    - Think of this as the "error with infinite data".
  - We assume that our training examples are drawn IID from this distribution.
    - Otherwise, "training" might not help to reduce "test error".

- Unfortunately, we cannot compute the test error.
  - We don't have access to the distribution over all test examples.

“Test Set” vs. “Test Error”

- We often approximate “test error” with the error on a “test set”:

$$\frac{1}{t}\sum_{i=1}^{t}|\hat{y}^i - \tilde{y}^i|$$

  - Here, we are using ‘t’ examples drawn IID from the test distribution.
- Note that “test set error” is not the “test error”.
  - The goal is have a low “test error”, not “test set error”.

- The “golden rule” of machine learning:
  - A “test set” cannot influence the “training” in any way.
  - Otherwise, “test set error” is not an unbiased “test error” approximation.
  - We run the risk of “overfitting” to the “test set”.

Typical Supervised Learning Steps (Are Bad?)

- Given data {X,y}, a typical set of supervised learning steps:
  - Data splitting:
    - Split {X,y} into a train set {Xtrain,ytrain} and a validation set {Xvalid,yvalid}.
    - We're going to use the validation set error as approximation of test error.
  - Tune hyper-parameters (number of hidden units, λ, polynomial degree,etc.):
    - For each candidate value "λ" of the hyper-parameters:
      - Fit a model to the train set {Xtrain,ytrain} using the given hyper-parameters "λ".
      - Evaluate the model on the validation set {Xvalid,yvalid}.
  - Choose the model with the best performance on the validation set.
    - And maybe re-train using hyper-parameter "λ" on the full dataset.

- Can this overfit, even though we used a validation set?
  - Yes, we've violated the golden rule. But maybe it's not too bad…

Validation Error, Test Error, and Approximation Error

- 340 discusses the "Fundamental Trade-Off of Machine Learning".
  - Simple identity relating training set error to test error.
- We have a similar identity for the validation error.
  - If E_{test} is the test error and E_{valid} is the error on the validation set, then:

$$E_{test} = \underbrace{(E_{test} - E_{valid})}_{E_{approx}} + E_{valid}$$

- If E_{approx} is small, then E_{valid} is a good approximation of E_{test}.
  - We can't measure E_{test}, so how do we know if E_{approx} is small?

Bounding E_{approx}

- Let's consider a simple case:
  - Labels y^i are binary, and we try 1 hyper-parameter setting.
  - IID assumption on validation set implies E_{valid} is unbiased: $\mathbf{E}[E_{valid}] = E_{test}$.
- We can bound probability E_{approx} is different than E_{test} by ε.
  - Assumptions: data is IID (so E_{valid} is unbiased) and loss is in [0,1].
  - By using Hoeffding's inequality:

$$p(|E_{test} - E_{valid}| > \varepsilon) \leq 2\exp(-2\varepsilon^2 t)$$

E_{approx}

↑ number of examples in validation set

  - Probability that E_{valid} is far from E_{test} goes down exponentially with 't'.
    - This is great: the bigger your validation set, the better approximation you get.

Bounding E_{approx}

- Let's consider a simple case:
  - Labels are binary, and we tried 'k' hyper-parameter values.
  - In this case it's unbiased for each 'k': $E[E_{valid(\lambda)}] = E_{test}$.
  - So for *each* validation error $E_{valid(\lambda)}$ we have:

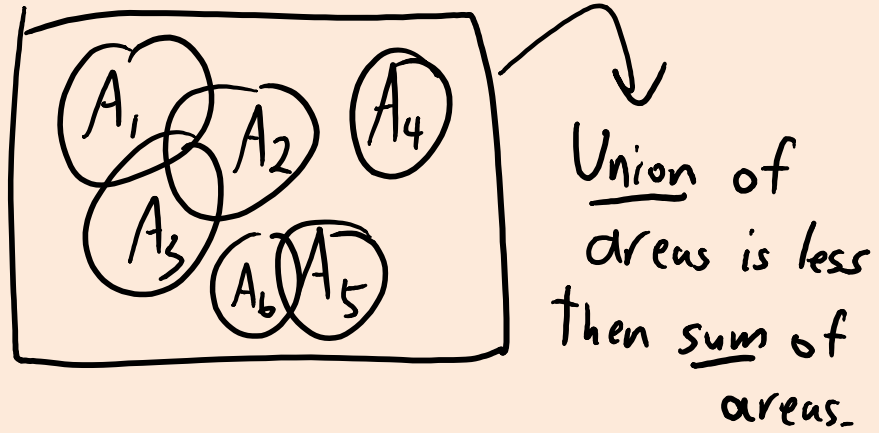
$$p(|E_{test} - E_{valid(\lambda)}| > \varepsilon) \leq 2\exp(-2\varepsilon^2 t)$$

  - But our final $E_{valid} = \min\{E_{valid(\lambda)}\}$, which is biased.
    - We can't apply Hoeffding because we chose best among 'k' values.
- Fix: bound on probability that **all** $|E_{test} - E_{valid(\lambda)}|$ values are $\leq \varepsilon$.
  - Since we showed it holds for all values, it holds for the best value.

Bounding E_{approx}

- The "union bound" for any events $\{A_1, A_2, \ldots, A_k\}$ is that:

$$p(A_1 \cup A_2 \cup \cdots \cup A_K) \leq \sum_{i=1}^{k} p(A_i)$$



- Combining with Hoeffding we can get:

$$p(|E_{test} - \min_{\lambda}\{E_{valid(\lambda)}\}| > \varepsilon) \leq p(|E_{test} - E_{valid(\lambda)}| > \varepsilon \text{ for any } \lambda)$$

$$\leq \sum_{\lambda} p(|E_{test} - E_{valid(\lambda)}| > \varepsilon)$$

$$\leq \sum_{\lambda} 2\exp(-2\varepsilon^2 t)$$

$$= K\, 2\exp(-2\varepsilon^2 t)$$

Bounding E_{approx}

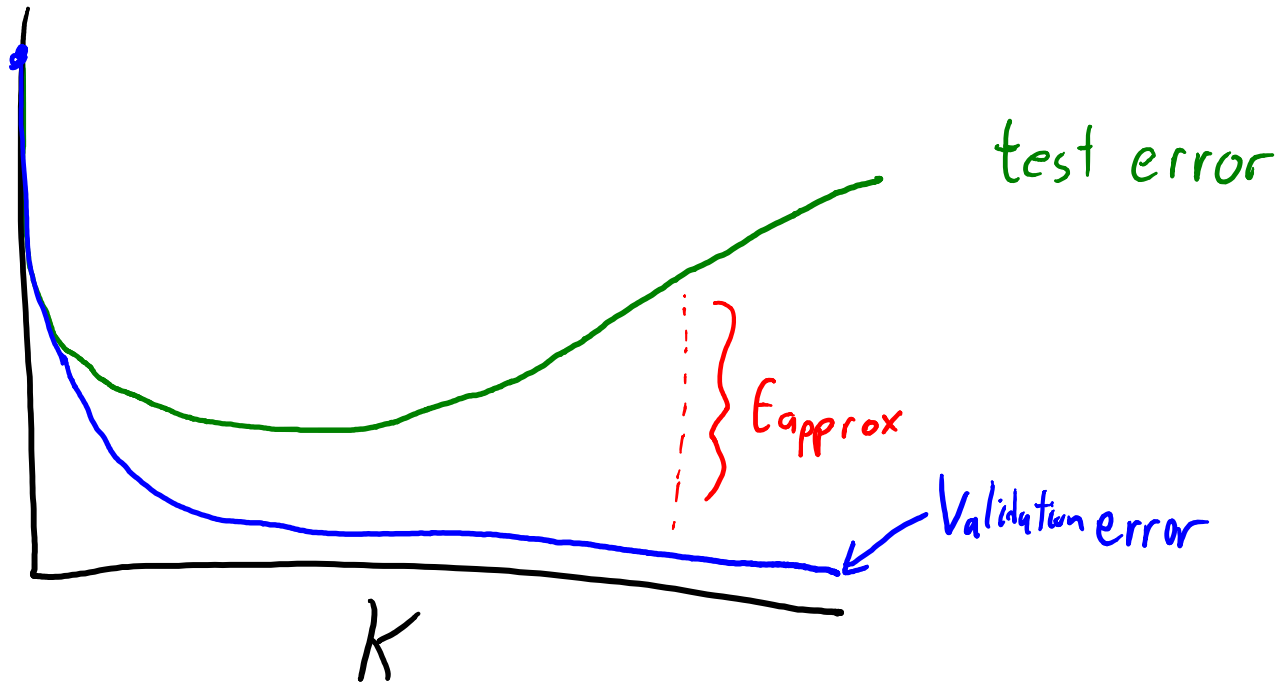
- So if we choose best $E_{valid(\lambda)}$ among 'k' λ values, we have:

$$p(|E_{test} - E_{valid(\lambda)}| > \varepsilon \text{ for } \underline{\text{any}}\ \lambda) \leq k\, 2\exp(-2\varepsilon^2 t)$$

- So optimizing over 'k' models is ok if we have large 't'.
  - But if 'k' is too large or 't' is too small the validation error isn't useful.
- Examples:
  - If k=10 and t=1000, probability that $|E_{approx}| > .05$ is less than 0.14.
  - If k=10 and t=10000, probability that $|E_{approx}| > .05$ is less than 10^{-20}.
  - If k=10 and t=1000, probability that $|E_{approx}| > .01$ is less than 2.7 (useless).
  - If k=100 and t=100000, probability that $|E_{approx}| > .01$ is less than 10^{-6}.

Bounding E_{approx}

- Validation error vs. test error for fixed ‘t’.
  - E_{valid} goes down as we increase ‘k’, but E_{approx} can go up.
    - Overfitting of validation set.



Discussion

- Bound is usually very loose, but data is probably not fully IID.
  - Similar bounds are possible for cross-validation.

- Similar arguments apply for the E_{approx} of the training error.
  - But 'k' is usually huge: you try out k=O(nd) decision stumps.

- What if we train by gradient descent?
  - We're optimizing on continuous space (k=∞), so the bound is useless.
  - In this case, VC-dimension is one way to replace 'k'.
    - "Simpler" models like decision stumps and linear models will have lower VC-dimension.

- Learning theory keywords if you want to go deeper into this topic:
  - Bias-variance (see bonus slides for details and why this is weird), sample complexity, PCA learning, VC dimension, Rademacher complexity.
  - A gentle place to start is the Learning from Data book.

(pause)

Generalization Error

- An alternative measure of performance is the generalization error:
  - Average error over the set of x^i values that are not seen in the training set.

- Test error vs. generalization error when labels are deterministic:

$$E_{test} = \mathbb{E}\left[|\hat{y}^i - \tilde{y}^i| \right]$$

Labels are deterministic, but we still take expectation over data distribution

$$E_{generalize} = \frac{1}{t} \sum_{x^i \notin \{\text{train set}\}} |\hat{y}_i - \tilde{y}_i|$$

number of x^i values not in training set.

average error over unseen x^i values.

“Best” and the “Good” Machine Learning Models

- Question 1: what is the “best” machine learning model?
  - The model that gets lower generalization error than all other models.
- Question 2: which models always do better than random guessing?
  - Models with lower generalization error than random for all problems.

- No free lunch theorem:
  - There is **no** “best” model achieving the best generalization error for every problem.
  - If model A generalizes better to new data than model B on one dataset, there is another dataset where model B works better.

No Free Lunch Theorem

- Let's show the "no free lunch" theorem in a simple setting:
  - The x^i and y^i are binary, and y^i being a deterministic function of x^i.
- With 'd' features, each "learning problem" is a map from each of the 2^d feature combinations to 0 or 1: $\{0,1\}^d$ -> $\{0,1\}$

Feature 1	Feature 2	Feature 3
0	0	0
0	0	1
0	1	0
…	…	…

Map 1	Map 2	Map 3	…
0	1	0	…
0	0	1	…
0	0	0	…
…	…	…	…

- Let's pick one of these maps ("learning problems") and:
  - Generate a set training set of 'n' IID samples.
  - Fit model A (convolutional neural network) and model B (naïve Bayes).

No Free Lunch Theorem

- Define the "unseen" examples as the $(2^d - n)$ not seen in training.
  - Assuming no repetitions of x^i values, and $n < 2^d$.
  - Generalization error is the average error on these "unseen" examples.
- Suppose that model A got 1% error and model B got 60% error.
  - We want to show model B beats model A on another "learning problem".

- Among our set of "learning problems" find the one where:
  - The labels y^i agree on all training examples.
  - The labels y_i disagree on all "unseen" examples.
- On this other "learning problem":
  - Model A gets 99% error and model B gets 40% error.

No Free Lunch Theorem

- Further, across all “learning problems” with these ‘n’ examples:
  - Average generalization error of **every** model is 50% on unseen examples.
  - With ‘k’ classes, the average error is (k-1)/k% (random guessing).

- This is kind of depressing:
  - For general problems, no “machine learning” is better than “predict 0”.

(pause)

Limit of No Free Lunch Theorem

- Fortunately, the world is structured:
  - Some "learning problems" are more likely than others.

- For example, it's usually the case that "similar" x^i have similar y^i.
  - Datasets with properties like this are more likely.
  - Otherwise, you probably have no hope of learning.

- Models with the right "similarity" assumptions can beat "predict 0".

- With assumptions like this, you can consider consistency:
  - As 'n' grows, model A converges to the optimal test error.

Refined Fundamental Trade-Off

- Let E_{best} be the irreducible error (lowest possible error for *any* model).
  - For example, irreducible error for predicting coin flips is 0.5.
- Some learning theory results use E_{best} to further decompose E_{test}:

$$E_{test} = \underbrace{(E_{test} - E_{train})}_{E_{approx}} + \underbrace{(E_{train} - E_{best})}_{E_{model}} + \underbrace{E_{best}}_{\text{"noise"}}$$

- This is similar to the bias-variance trade-off (bonus slide):
  - E_{approx} measures *how sensitive we are to training data* (like "variance").
  - E_{model} measures *if our model is complicated enough to fit data* (like "bias").
  - E_{best} measures how low can **any** model make test error ("irreducible" error).

Refined Fundamental Trade-Off

- Let E_{best} be the irreducible error (lowest possible error for *any* model).
  - For example, irreducible error for predicting coin flips is 0.5.
- Some learning theory results use E_{best} to further decompose E_{test}:

$$E_{test} = \underbrace{(E_{test} - E_{train})}_{E_{approx}} + \underbrace{(E_{train} - E_{best})}_{E_{model}} + \underbrace{E_{best}}_{\text{"noise"}}$$

- This is similar to the bias-variance trade-off (bonus slide):
  - You need to trade between having low E_{approx} and having low E_{model}.
  - Powerful models have low E_{model} but can have high E_{approx}.
  - E_{best} does not depend on what model you choose.

Consistency and Universal Consistency

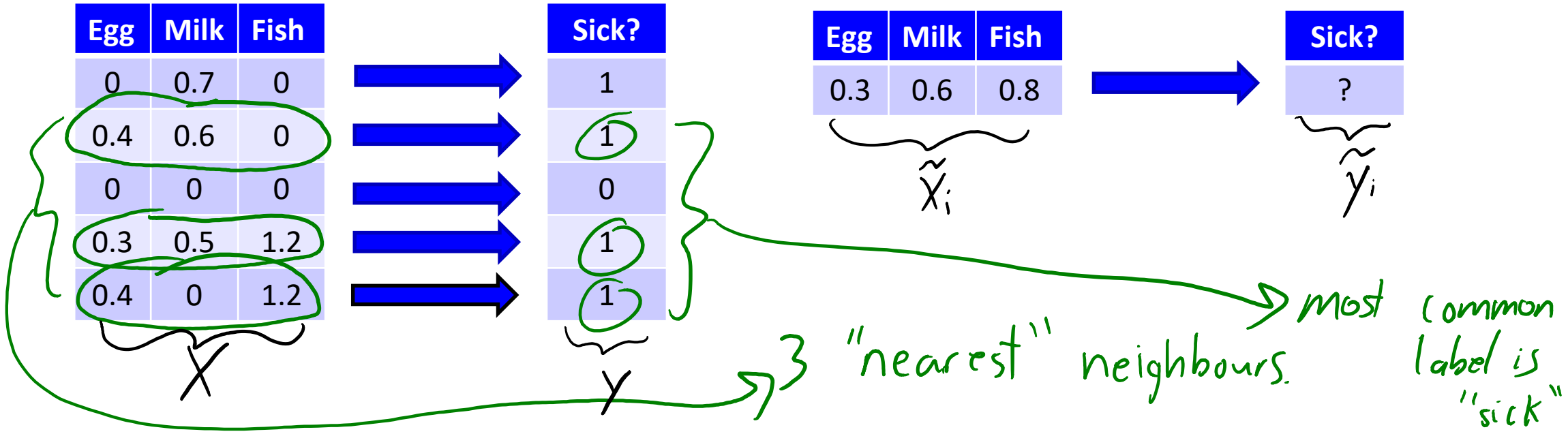
- A model is a consistent for a particular learning problem if:
  - E_{test} converges to E_{best} as 'n' goes to infinity, for that particular problem.

- A model is universally consistent for a class of learning problems if:
  - E_{test} converges to E_{best} as 'n' goes to infinity, for all problems in the class.

- Typically, the class would consist of:
  - A continuity assumption on the labels y^i as a function of x^i.
    - E.g., if x^i is close to x^j then they are likely to receive the same label.
  - A boundedness assumption of the set of x^i.

K-Nearest Neighbours (KNN)

- Classical consistency results focus on k-nearest neighbours (KNN).
- To classify an object $\tilde{x}_i$:
  1. Find the 'k' training examples x_i that are "nearest" to $\tilde{x}_i$.
  2. Classify using the most common label of "nearest" examples.



Consistency of KNN

- KNN consistency properties (under reasonable assumptions):
  - As ‘n’ goes to ∞, $E_{test} <= 2E_{best}$.
    - For fixed ‘k’ and binary labels.

- Stone’s Theorem: KNN is “universally consistent”.
  - If k/n converges to 0 as ‘n’ goes to ∞, E_{test} converges to E_{best}.
    - For example, k = O(log n).
    - First algorithm shown to have this property.

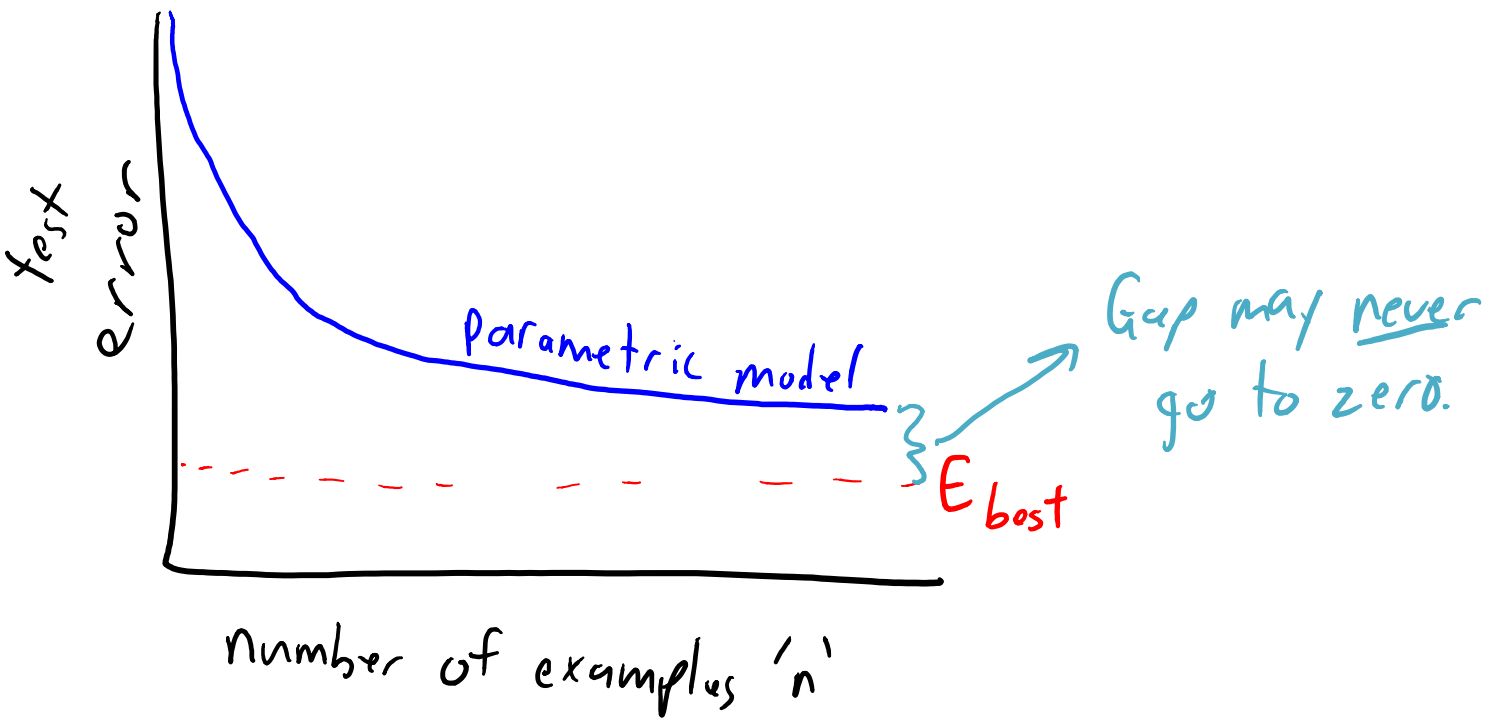
- Consistency says nothing about finite ‘n’.
  - See "Dont Trust Asymptotics”.

Consistency of Non-Parametric Models

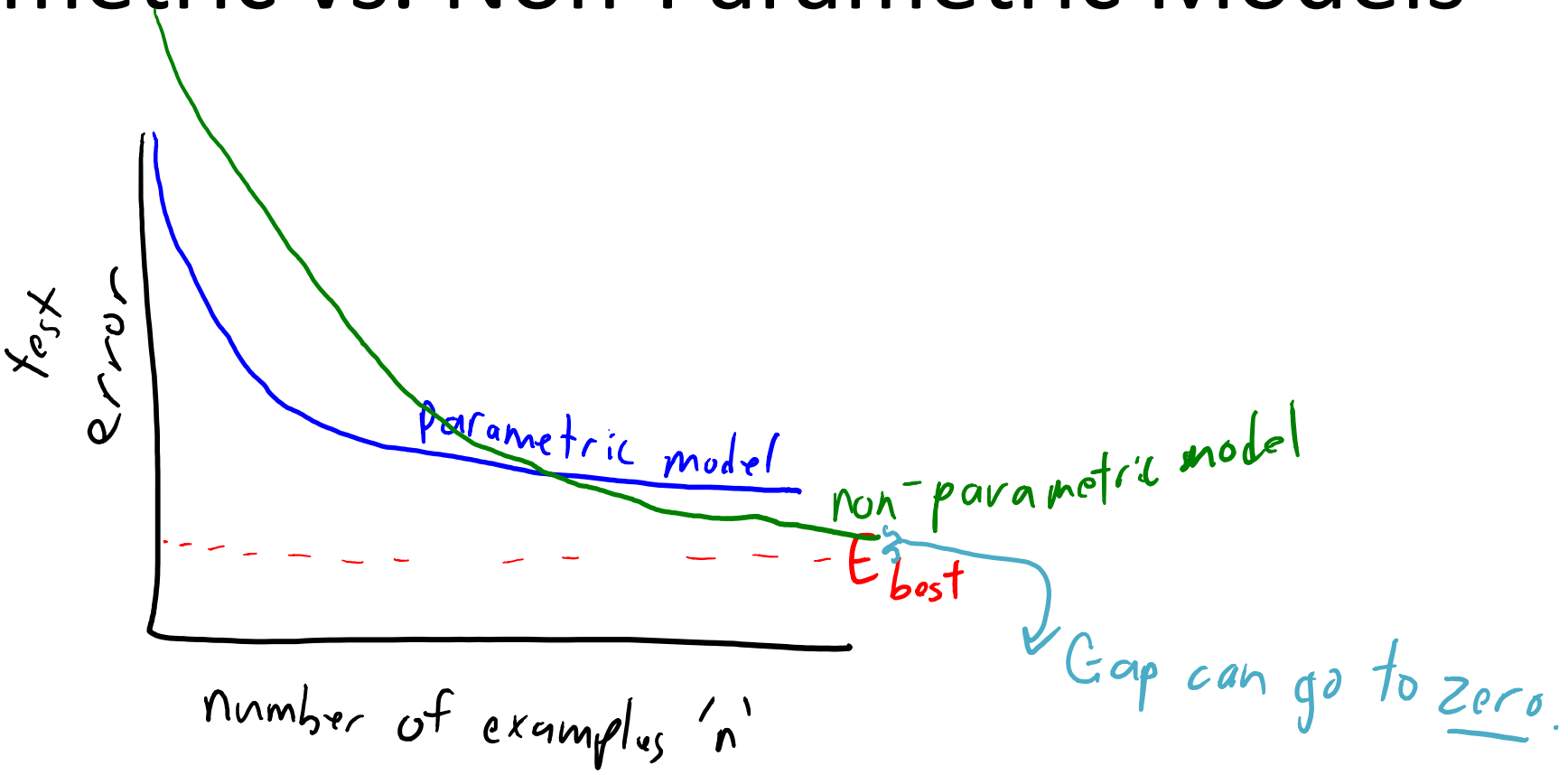
- Universal consistency has been shown for several models:
  - Linear models with polynomial basis.
  - Linear models with Gaussian RBFs.
  - Neural networks with one hidden layer and standard activations.
    - Sigmoid, tanh, ReLU, etc.

- It's non-parametric versions that are consistent:
  - Size of model is a function of 'n'.
  - Examples:
    - KNN needs to store all 'n' training examples.
    - Degree of polynomial must grow with 'n' (not true for fixed polynomial).
    - Number of hidden units must grow with 'n' (not true for fixed neural network).

Parametric vs. Non-Parametric Models



Parametric vs. Non-Parametric Models



Summary

- Test error vs. test set error
  - What we care about is the test error.
- Overfitting hyper-parameters on a validation set:
  - Depends on how many hyper-parameters you try and number of validation examples.
- No free lunch theorem:
  - There is no “best” or even “good” machine learning models across all problems.
- Universal consistency:
  - Some non-parametric models can solve any continuous learning problem.

- Post-lecture bonus slides: bias-variance decomposition.

- Next time:
  - Besides least squares, what other problems can be solved in 1 line of code?

Bias-Variance Decomposition

- Analysis of expected test error of any learning algorithm:

Assume $y_i = f(x_i) + \varepsilon$, for some function 'f' and random error ε with a mean of 0 and a variance of σ^2.

Assume we have a "learner" that can take a training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and use these to make predictions $\hat{f}(x_i)$.

Then for a new example (x_i, y_i) the error averaged over training sets is

$$E[(y_i - \hat{f}(x_i))^2] = \text{Bias}[\hat{f}(x_i)]^2 + \text{Var}[\hat{f}(x_i)] + \sigma^2$$

"Irreducible error": best we can hope for given the noise level.

Expected error due to having wrong model.

Where $\text{Bias}[\hat{f}(x_i)] = E[\hat{f}(x_i)] - f(x_i)$,

How sensitive is the model to the particular training set?

$$\text{Var}[\hat{f}(x_i)] = E[(\hat{f}(x_i) - E[\hat{f}(x_i)])^2]$$

Learning Theory

- Bias-variance decomposition is a bit weird:
  - Considers expectation over *possible training sets*.

- Bias-variance says nothing about your training set.
  - This is different than Hoeffding bounds:
    - Bound the test error based on your actual training set and training/validation error.