CPSC 440: Machine Learning

Fundamentals of Learning

Winter 2021

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 or 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 an 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 greater than 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 slightly less-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:

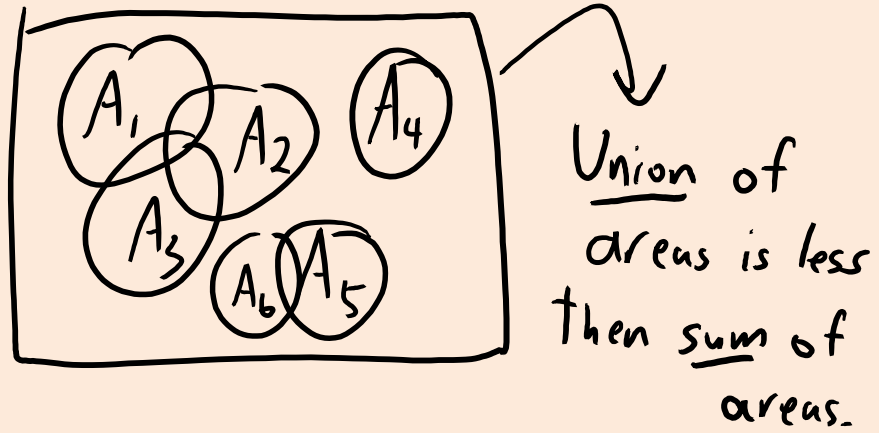
$$p(|E_{test} - E_{valid(\lambda)}| > \varepsilon) \leq 2\exp(-2\varepsilon^2 t)$$

  - But our final validation error is $E_{valid} = \min\{E_{valid(\lambda)}\}$, 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 $\leq \varepsilon$.
  - We show it holds for all values of λ, so it must hold 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:

$$p(|E_{test} - \min_{\lambda}\{E_{valid(\lambda)}\}| > \varepsilon) \leq p(\text{Exists a } \lambda \text{ where } |E_{test} - E_{valid(\lambda)}| > \varepsilon)$$

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

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

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

Bounding E_{approx}

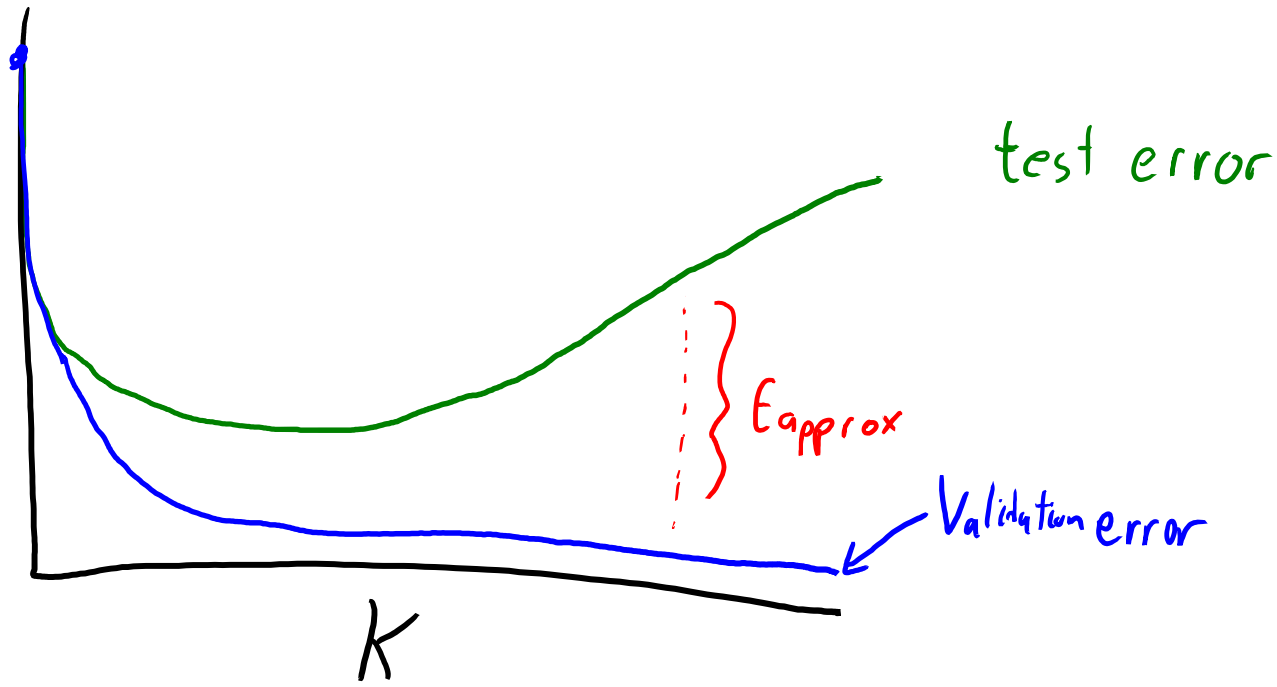
- So if we choose best $E_{valid(\lambda)}$ among 'k' λ values, we have:

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

- So optimizing over 'k' models is ok if we have a 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.
  - Value 'k' is the number of hyper-parameters you are optimizing over (even if don't try them all).
  - So '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, so k=∞ and the bound is useless.
  - In this case, VC-dimension is one way to replace 'k' (doesn't need union bound).
    - "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, PAC learning, VC dimension, Rademacher complexity.
  - A gentle place to start is the Learning from Data book.

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.

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

- Next time:
  - More about convexity than you ever wanted to know.

Bias-Variance Decomposition

- You may have seen "bias-variance decomposition" in other classes:
  - Assumes $\tilde{y}_i = \bar{y}_i + \varepsilon$, where ε has mean 0 and variance σ^2.
  - Assumes we have a "learner" that can take 'n' training examples and use these to make predictions $\hat{y}_i$.
- Expected squared test error in this setting is

$$E[(\tilde{y}_i - \hat{y}_i)^2] = E[(\hat{y}_i - \bar{y}_i)]^2 + (E[\hat{y}_i^2] - E[\hat{y}_i]^2) + \sigma^2$$

"test squared error" "bias" "variance" "noise"

  - Where expectations are taken over possible training sets of 'n' examples.
  - Bias is expected error due to having wrong model.
  - Variance is expected error due to sensitivity to the training set.
  - Noise (irreducible error) is the best can hope for given the noise (E_{best}).

Bias-Variance vs. Fundamental Trade-Off

- Both decompositions serve the same purpose:
  - Trying to evaluate how different factors affect test error.

- They both lead to the same 3 conclusions:
  1. Simple models can have high E_{train}/bias, low E_{approx}/variance.
  2. Complex models can have low E_{train}/bias, high E_{approx}/variance.
  3. As you increase 'n', E_{approx}/variance goes down (for fixed complexity).

Bias-Variance vs. Fundamental Trade-Off

- So why focus on fundamental trade-off and not bias-variance?
  - Simplest viewpoint that gives these 3 conclusions.
  - No assumptions like being restricted to squared error.

  - You can measure E_{train} but not E_{approx} (1 known and 1 unknown).
    - If E_{train} is low and you expect E_{approx} to be low, then you are happy.
      - E.g., you fit a very simple model or you used a huge independent validation set.

  - You can't measure bias, variance, or noise (3 unknowns).
    - If E_{train} is low, bias-variance decomposition doesn't say anything about test error.
      - You only have your training set, not distribution over possible datasets.
      - Doesn't say if high E_{test} is due to bias or variance or noise.

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.