CPSC 340: Machine Learning and Data Mining

Fundamentals of Learning

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Admin

- Assignment 0 is due tonight: you should be almost done.
- Assignment 1 coming very soon: you should have declared partners
- Waitlist at 80 students, with one week to go.
  – Please do not ask me to make an exception.
- Prerequisites:
  – If there is a problem, you will be emailed. Follow instructions there.
- Important webpages:
  – https://www.cs.ubc.ca/getacct/
  – https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/home
  – https://piazza.com/class/j9uk5ecmb7e4ks
  – https://www.cs.ubc.ca/students/undergrad/courses-deadlines/prerequisites

Last Time: Supervised Learning Notation

$X =$

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

'n' 'd'

$y =$

Sick?
1
1
0
1
1

'n'

- Feature matrix 'X' has rows as objects, columns as features.
  - x_{ij} is feature 'j' for object 'i' (quantity of food 'j' on day 'i').
  - x_i is the list of all features for object 'i' (all the quantities on day 'i').
  - x^j is column 'j' of the matrix (the value of feature 'j' across all objects).
- Label vector 'y' contains the labels of the objects.
  - y_i is the label of object 'i' (1 for "sick", 0 for "not sick").

Supervised Learning Application

- We motivated supervised learning by the "food allergy" example.

- But we can use supervised learning for any input:output mapping.
  - E-mail spam filtering.
  - Optical character recognition on scanners.
  - Recognizing faces in pictures.
  - Recognizing tumours in medical images.
  - Speech recognition on phones.
  - Your problem in industry/research?

(Switch to Jupyter demo)

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 there is also testing data we want to label:

$\tilde{X}=$

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?
?
?
?

Supervised Learning Notation

- Typical supervised learning steps:
  1. Build model based on training data X and y.
  2. Model makes predictions $\hat{y}$ on test data $\tilde{X}$.

- Instead of training error, consider test error:
  - Are predictions $\hat{y}$ similar to true unseen labels $\tilde{y}$?

Goal of Machine Learning

- In machine learning:
  - What we care about is the test error!
- Midterm analogy:
  - The training error is the practice midterm.
  - The test error is the actual midterm.
  - Goal: do well on actual midterm, not the practice one.
- Memorization vs learning:
  - Can do well on training data by memorizing it.
  - You've only learned if you can do well in new situations.

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.
- We’re measuring test error to see how well we do on new data:
  - If used during training, doesn’t measure this.
  - You can start to overfit if you use it during training.
  - Midterm analogy: you are cheating on the test.

Is Learning Possible?

- Does training error say anything about test error?
  - In general, NO: Test data might have nothing to do with training data.
  - E.g., test labels are random numbers
- In order to learn, we need assumptions:
  - The training and test data need to be related in some way.
  - Most common assumption: independent and identically distributed (IID).

IID Assumption

- Training/test data is independent and identically distributed (IID) if:
  - All objects come from the same distribution (identically distributed).
  - The object are sampled independently (order doesn't matter).

Row 1 comes from same distribution as rows 2-3.

Age	Job?	City	Rating	Income
23	Yes	Van	A	22,000.00
23	Yes	Bur	BBB	21,000.00
22	No	Van	CC	0.00
25	Yes	Sur	AAA	57,000.00

Row 4 does not depend on values in rows 1-3.

- Examples in terms of cards:
  - Pick a card, put it back in the deck, re-shuffle, repeat. → IID
  - Pick a card, put it back in the deck, repeat.
  - Pick a card, don't put it back, re-shuffle, repeat.

Not IID

Testing IID?

IID Assumption and Food Allergy Example

- Is the food allergy data IID?
  - Do all the objects come from the same distribution?
  - Does the order of the objects matter?
- No!
  - Being sick might depend on what you ate yesterday (not independent).
  - Your eating habits might changed over time (not identically distributed).
- What can we do about this?
  - Just ignore that data isn't IID and hope for the best?
  - For each day, maybe add the features from the previous day?
  - Maybe add time as an extra feature?

Learning Theory

- Why does the IID assumption make learning possible?
  - Patterns in training examples are likely to be the same in test examples.
- The IID assumption is rarely true:
  - But it is often a good approximation.

- Learning theory explores how training error is related to test error.
- We'll look at a simple example, using this notation:
  - E_{train} is the error on training data.
  - E_{test} is the error on testing data.

Fundamental Trade-Off

- Start with $E_{test} = E_{test}$, then add and subtract E_{train} on the right:

$$E_{test} = \underbrace{(E_{test} - E_{train})}_{\text{"approximation error"}\ E_{approx}} + \underbrace{E_{train}}_{\text{"training error"}}$$

"test error"

- How does this help?
  - If E_{approx} is small, then E_{train} is a good approximation to E_{test}.
- What does E_{approx} depend on?
  - It tends to gets smaller as 'n' gets larger.
  - It tends to grow as model get more "complicated".

Fundamental Trade-Off

- This leads to a fundamental trade-off:
  1. E_{train}: how small you can make the training error.

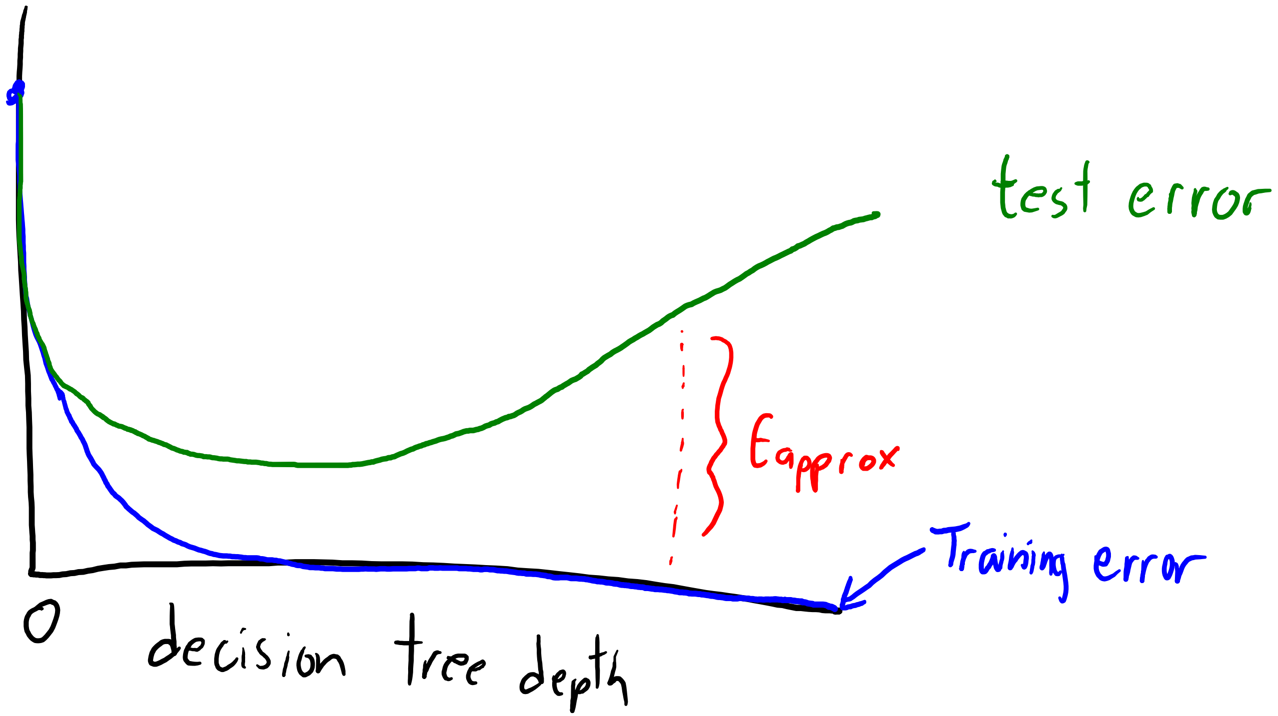
     vs.

  2. E_{approx}: how well training error approximates the test error.

- Simple models (like decision stumps):
  - E_{approx} is low (not very sensitive to training set).
  - But E_{train} might be high.
- Complex models (like deep decision trees):
  - E_{train} can be low.
  - But E_{approx} might be high (very sensitive to training set).

Fundamental Trade-Off

- Training error vs. test error for choosing depth:
  - Training error gets better with depth.
  - Test error initially goes down, but eventually increases (overfitting).



Validation Error

- How do we decide decision tree depth?
- We care about test error.
- But we can’t look at test data.
- So what do we do?????

- One answer: Use part of your dataset to approximate test error.
- Split training objects into training set and validation set:
  - Train model based on the training data.
  - Test model based on the validation data.

Validation Error

$$X = \begin{bmatrix} \quad \\ \hdashline \quad \end{bmatrix} \qquad y = \begin{bmatrix} \quad \\ \hdashline \quad \end{bmatrix} \begin{matrix} \left.\vphantom{\Big|}\right\} \text{"train"} \\ \left.\vphantom{\Big|}\right\} \text{"validation"} \end{matrix}$$

Step 1 is <u>training</u>: model = train(X_{train}, y_{train})

Step 2 is <u>predicting</u>: $\hat{y}$ = predict(model, $X_{validate}$)

Step 3 is <u>validating</u>: error = sum($\hat{y} \neq y_{validate}$)

Note: if examples are ordered, split should be <u>random</u>.

Validation Error

- Validation error gives an unbiased approximation of test error.

- Midterm analogy:
  - You have 2 practice midterms.
  - You hide one midterm, and spend a lot of time working through the other.
  - You then do the other practice term, to see how well you'll do on the test.

- We typically use validation error to choose "hyper-parameters"…

Notation: Parameters and Hyper-Parameters

- The decision tree rule values are called "parameters".
  - Parameters control how well we fit a dataset.
  - We "train" a model by trying to find the best parameters on training data.

- The decision tree depth is a called a "hyper-parameter".
  - Hyper-parameters control how complex our model is.
  - We can't "train" a hyper-parameter.
    - You can always fit training data better by making the model more complicated.
  - We "validate" a hyper-parameter using a validation score.

Choosing Hyper-Parameters with Validation Set

- So to choose a good value of depth (“hyper-parameter”), we could:
  - Try a depth-1 decision tree, compute validation error.
  - Try a depth-2 decision tree, compute validation error.
  - Try a depth-3 decision tree, compute validation error.
  - …
  - Try a depth-20 decision tree, compute validation error.
  - Return the depth with the lowest validation error.

- After you choose the hyper-parameter, we usually re-train on the full training set with the chosen hyper-parameter.

Choosing Hyper-Parameters with Validation Set

- This leads to much less overfitting than using the training error.
  - We optimize the validation error over 20 values of “depth”.
  - Unlike training error, where we optimize over tons of decision trees.

- But it can still overfit (very common in practice):
  - Validation error is only an unbiased approximation if you use it once.
  - If you minimize it to choose a model, introduces optimization bias:
    - If you try lots of models, one might get a low validation error by chance.

- Remember, our goal is still to do well on the test set (new data), not the validation set (where we already know the labels).

Summary

- Training error vs. testing error:
  - What we care about in machine learning is the testing error.
- Golden rule of machine learning:
  - The test data cannot influence training the model in any way.
- Independent and identically distributed (IID):
  - One assumption that makes learning possible.
- Fundamental trade-off:
  - Trade-off between getting low training error and having training error approximate test error.
- Validation set:
  - We can save part of our training data to approximate test error.
- Hyper-parameters:
  - Parameters that control model complexity, typically set with a validation set.

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.

http://www.technologyreview.com/view/538111/why-and-how-baidu-cheated-an-artificial-intelligence-test/

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.

- You also shouldn't change the test set to get the result you want.

DECEPTION AT DUKE: FRAUD IN CANCER CARE?

Were some cancer patients at Duke University given experimental treatments based on fabricated data? Scott Pelley reports.

  - http://blogs.sciencemag.org/pipeline/archives/2015/01/14/the_dukepotti_scandal_from_the_inside

https://www.cbsnews.com/news/deception-at-duke-fraud-in-cancer-care/

Bounding E_{approx}

- Let's assume we have a fixed model 'h' (like a decision tree), and then we collect a training set of 'n' examples.
- What is the probability that the error on this training set (E_{train}), is within some small number ε of the test error (E_{test})?

- From "Hoeffding's inequality" we have:

$$P[|E_{train}(h) - E_{test}(h)| > \epsilon] \leq 2\exp(-2\epsilon^2 n)$$

- This is great! In this setting the probability that our training error is far from our test error goes down exponentially in terms of the number of samples 'n'.

Bounding E_{approx}

- Unfortunately, the last slide gets it backwards:
  - We usually don't pick a model and then collect a dataset.
  - We usually collect a dataset and then pick the model 'w' based on the data.
- We now picked the model that did best on the data, and Hoeffding's inequality doesn't account for the optimization bias of this procedure.

- One way to get around this is to bound ($E_{test} – E_{train}$) for *all* models in the space of models we are optimizing over.
  - If bound it for all models, then we bound it for the best model.
  - This gives looser but correct bounds.

Bounding E_{approx}

- If we only optimize over a finite number of events 'k', we can use the "union bound" that for events $\{A_1, A_2, \ldots, A_k\}$ we have:

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

- Combining Hoeffding's inequality and the union bound gives:

$$p(|E_{train}(h_1) - E_{test}(h_1)| > \varepsilon \cup |E_{train}(h_2) - E_{test}(h_2)| > \varepsilon \cup \cdots \cup |E_{train}(h_k) - E_{test}(h_k)| > \varepsilon)$$

$$\leq \sum_{i=1}^{k} p(|E_{train}(h_i) - E_{test}[h_i]| > \varepsilon)$$

$$\leq \sum_{i=1}^{k} 2\exp(-2\varepsilon^2 n)$$

$$\leq 2k\exp(-2\varepsilon^2 n)$$

Bounding E_{approx}

- So, with the optimization bias of setting "h*" to the best 'h' among 'k' models, probability that (Etest – Etrain) is bigger than ε satisfies:

$$p(|E_{train}(h^*) - E_{test}(h^*)| > \epsilon) \leq 2k\exp(-2\epsilon^2 n)$$

- So optimizing over a few models is ok if we have lots of examples.
- If we try lots of models then (E_{test} – E_{train}) could be very large.
- Later in the course we'll be searching over continuous models where k = infinity, so this bound is useless.
- To handle continuous models, one way is via the VC-dimension.
  - Simpler models will have lower VC-dimension.

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 futher decompose E_{test}:

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

- This is similar to the bias-variance decomposition:
  - Term 1: measure of variance (how sensitive we are to training data).
  - Term 2: measure of bias (how low can we make the training error).
  - Term 3: measure of noise (how low can any model make test error).

Refined Fundamental Trade-Off

- Decision tree with high depth:
  - Very likely to fit data well, so bias is low.
  - But model changes a lot if you change the data, so variance is high.
- Decision tree with low depth:
  - Less likely to fit data well, so bias is high.
  - But model doesn’t change much you change data, so variance is low.

- And degree does not affect irreducible error.
  - Irreducible error comes from the best possible model.

Bias-Variance Decomposition

- Analysis of expected test error of any learning algorithm:

Assume $y_i = f(x_i) + \epsilon$, 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 $\text{Bias}[\hat{f}(x_i)] = E[\hat{f}(x_i)] - f(x_i)$,

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

How sensitive is the model to the particular training set?

Learning Theory

- Bias-variance decomposition is a bit weird compared to our previous decompositions of E_{test}:
  - Bias-variance decomposition considers expectation over *possible training sets*.
  - But doesn't say anything about test error with *your* training set.

- Some keywords if you want to learn about learning theory:
  - Bias-variance decomposition, sample complexity, probably approximately correct (PAC) learning, Vapnik-Chernovenkis (VC) dimension, Rademacher complexity.

- A gentle place to start is the "Learning from Data" book:
  - https://work.caltech.edu/telecourse.html

A Theoretical Answer to “How Much Data?”

- Assume we have a source of IID examples and a fixed class of parametric models.
  - Like “all depth-5 decision trees”.
- Under some nasty assumptions, with ‘n’ training examples it holds that:

  E[test error of best model on training set] – (best test error in class) = O(1/n).

- You rarely know the constant factor, but this gives some guidelines:
  - Adding more data helps more on small datasets than on large datasets.
    - Going from 10 training examples to 20, difference with best possible error gets cut in half.
      - If the best possible error is 15% you might go from 20% to 17.5% (this does **not** mean 20% to 10%).
    - Going from 110 training examples to 120, error only goes down by ~10%.
    - Going from 1M training examples to 1M+10, you won’t notice a change.
  - Doubling the data size cuts the error in half:
    - Going from 1M training to 2M training examples, error gets cut in half.
    - If you double the data size and your test error doesn’t improve, more data might not help.

Can you test the IID assumption?