Machine Learning

Evaluation (cont')
Online Learning

Dan Goldwasser

dgoldwas@purdue.edu

Goal for Today's class

- Model Evaluation and Online learning intro

- Evaluation Metrics
- Hypothesis tests intro
- Online learning and mistake bounds

Model Selection

- All the algorithms that we saw (and that we will see) can be parameterized, to help control their behavior
  - *I.e., control the properties of the type of models they will produce.*
- These can capture preferences that we have about the classifiers
  - *Smaller trees, preference towards one type of error, etc.*
- In some cases, we just want the settings that get us the **best** classifier
  - *But, well.. what is **best**? and how do we know that we found it? (what can we trust?)*

Model Selection

- ***What are the algorithm hyper-parameters?***
  - ***Decision trees:***
    - Depth of the tree
    - Pruning strategy
    - Pruning decision
    - Attribute selection heuristic
    - *Other choices?*
  - ***KNN***
    - Value of K
    - Similarity metric
- ***Every learning algorithm we will cover has a set of hyper-parameters***

Model Selection

- *We can think about selecting the best model as a* <u>secondary learning problem</u>
  - Split the data into: (1) **train** set (2) **test** set
  - Split the **train** data into: (1) **train** set (2) **validation** set
  - **<u>Training</u>**: train m models, with different parameters
    - E.g., Different ways to control the size of the tree
  - **<u>Validation</u>:** estimate the prediction error for each model
  - **<u>Testing</u>**: use the model with the least validation error
- ***The secondary learning problem***:
  - New hypothesis space: m different hypothesis to chose from
  - Pick the one that minimizes validation error

K-Fold Cross validation

- You could get really unlucky..
  - *..but that's not likely to happen too frequently!*

- K-Fold cross validation: repeat the process K times, and average the results.

- Randomly partition the data into K equal-size subsets $S_1..S_k$
  - For i=1… K
    - Train a hypothesis on $S_1..S_{i-1}\ S_{i+1}..S_k$
    - Evaluate on S_i ($Err(S_i)$)
  - Return ($\Sigma_i\ Err(S_i)/K$)

Formalizing the learning process

- We assume the data is sampled from an *unknown* distribution

$$D = P(x,y)$$

  - D assigns high probability to “reasonable” (x,y) pairs
  - Unreasonable (x,y) pairs:
    - *x is an unusual input (*Purdue + Hot days + January*)*
    - *Y an unlikely label for x (*Purdue + January ➔ Swimming*)*

- Performance is defined with respect to:
  - **Loss function**: What “*matters”* in the learning task
  - **Data generating distribution** (D)

Learning Definitions

$$P(S) = P((x_1, y_1), ..., (x_n, y_n))$$

- Given S, *a dataset*, examples in S are assumed to be *Independent and identically distributed* (iid):

$$P(S) = \prod_i P(x_i, y_i)$$

  - *Independently drawn from the same distribution*
    - When are examples not independent?
    - When are examples not identically distributed?
  - The key assumption behind machine learning algorithms and their theoretic analysis.

Formalizing the learning process

- Learning goal: Minimize expected loss over D w.r.t *l*

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

We do not know D in advance!

- *But, we have access to training data sampled from D*
- *Instead, compute Empirical loss*
  - ***What is the difference?***

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

- **Learning**: *find a function with low* ***expected*** *loss over D w.r.t l.*
- ***This is where inductive bias comes in!***

Evaluating the Learned Hypothesis

- What is a “good” error for a learned hypothesis?
    - ***How do I know my classifier is good enough?***
        - For example, $\text{Err}_p(h) < 0.1$
        - Is the training error a good estimate of the error?
        - Testing error? *Is it a good approximation for the* ***true error****?*
            - On average, that’s the true performance.
            - We have to account to variability!
            - i.e., different choices of testing sets could lead to different results!

    - **How can we account for the test error variability ?**
        - Run multiple times, and average results
        - Closed form solution

Quick Detour: *Binomial Distribution*

$$P(X = x|p, n) = \frac{n!}{x!(n-x)!}\ p^x(1-p)^{n-x}$$

x number of errors we observe
p is the probability of errors
n is the number of test examples

Evaluating the Learned Hypothesis

- Our algorithm produced a model (h)
  - 10 errors out of 500 test examples
  - Is that significant evidence for the $\mathrm{Err}_P(h)<$**0.1?**
- Significance test:
  - **Null hypothesis**: p ≥ **0.1**
  - *what is the probability that we see at most 10 errors out of 500?*

$$P(x \leq 10 | p = 0.1, n = 500) \approx 10^{-12} \leq 0.05$$

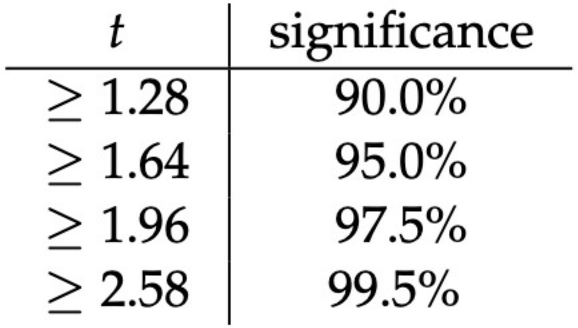
  - *If the null hypothesis was true, the observed performance is unlikely*
  - ***We can reject the null hypothesis!***

Comparing learning algorithms

- Null hypothesis: Alg 1 is no better than Alg 2.
- We would like to find the probability that the null hypothesis is false.
  - Common values: 90%, 95%, 99.5% (depending on community)
- T-test: assume that difference between the two error distributions has mean zero, and that the data is normally distributed
  - Binomial distribution can be approximated using a Gaussian for large N
- Assume N examples, `a1,…,an` and `b1,…,bn` error on the examples by algorithm 1 and algorithm 2 respectively

Comparing learning algorithms

- The mean of the error: μ1 (alg 1) μ2 (alg 2)
- Center the data: $\hat{a} = \boldsymbol{a} - \mu_a$ and $\hat{b} = \boldsymbol{b} - \mu_b$.
- Compute the t-statistic $$t = (\mu_a - \mu_b)\sqrt{\frac{N(N-1)}{\sum_n (\hat{a}_n - \hat{b}_n)^2}}$$



t	significance
≥ 1.28	90.0%
≥ 1.64	95.0%
≥ 1.96	97.5%
≥ 2.58	99.5%

Precision and Recall

- Given a dataset, we train a classifier that gets 99% accuracy
- **Did we do a good job?**
- Build a classifier for brain tumor:
  - 99.9% of brain scans do not show signs of tumor
  - ***Did we do a good job?***
- By simply saying "NO" to all examples we reduce the error by a factor of 10!
  - ***Clearly Accuracy is not the best way to evaluate the learning system when the data is heavily skewed!***
- **Intuition**: we need a measure that captures the class we care about! (rare)

Precision and Recall

- The learner can make two kinds of mistakes:
  - False Positive
  - False Negative

	True Label: 1	True Label: 0
Predicted: 1	True Positive	False Positive
Predicted: 0	False Negative	True Negative

- **Precision:**
- *“when we predicted the rare class, how often are we right?”*

$$\frac{\text{True Pos}}{\text{Predicted Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos}}$$

- **Recall**
- “Out of all the instances of the rare class, how many did we catch?”

$$\frac{\text{True Pos}}{\text{Actual Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Neg}}$$

F-Score

- **Precision and Recall give us two reference points to compare learning performance**

	Precision	**Recall**
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1

- *Which algorithm is better?*

We need a single score

- Option 1: Average $$\frac{P+R}{2}$$
- Option 2: F-Score $$2\frac{PR}{P+R}$$

Properties of f-score:

- *Ranges between 0-1*
- *Prefers precision and recall with similar values*

Confusion matrix

Predicted True	**0**	**1**	**2**
0	200	2	3
1	50	32	2
2	30	10	21

Ablation Study

- Making predictions often relies on many different attributes of the input object
- Let’s consider email phishing detection:
  - Baseline system: lexical features
    - *“The excellent prince of Mars wants to give you a g1ft”*
    - *Accuracy : 70%*
  - *Complex system:*
    - *Lexical features, sender, email headers, servers, images, dictionaries of suspicious terms, spelling mistakes*
    - *Accuracy: 85%*
- *What aspects are responsible for the improvement?*
  - ***Run an ablation study***

Ablation Study

- Remove one feature and train+test the model
- **Other things to check:**
  - *What is the influence of features choices on **precision/recall**?*
  - Similar mistakes or different mistakes?

Features	ACC
Lexical features	66
Sender	79
Email headers	83
Servers	80
Images	85
Suspicious terms	82
Baseline (lexical features)	70

Error Analysis:

You can also look at the **type** of mistakes your model is making:

Some Phishing scams could be easier to detect than others.

- *Check the influence of different features by mistake types*

Error Analysis

- Identify the root cause of the mistakes your algorithm makes
  - **Aspects not captured by your features**

“We wish you a happy new year”

  - Noisy feature extraction
    - E.g., “Suspicious terms” detector is not comprehensive enough
  - Many other reasons: noisy labels,..

Online Learning and Linear models

We will introduce a new way to quantify performance,
by ***measuring and bounding the number of mistake an algorithm makes***.

We'll show that depending on the hypothesis class we choose, we can use algorithms that have better mistake bounds

Specifically, we'll compare learning disjunctions using Boolean functions and linear functions, and analyze their behavior.

Quantifying Performance

- We want to be able to say something rigorous about the performance of our **learning algorithm**.
    - *Several ways of doing it*

- We will concentrate on discussing the number of examples one needs to **see** before we can say that our learned hypothesis is good.

Learning Conjunctions

- There is a hidden (monotone) conjunction the learner (you) is to learn

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

- High dimensional space, but concept only uses a few
- *How many examples are needed to learn it ?*
    - **Protocol I**: The learner proposes instances as queries to the teacher ("active learning")
    - **Protocol II**: The teacher (*who knows f*) provides training examples ("hands-on teacher")
    - **Protocol III:** *Some random source (e.g., Nature) provides training examples; the Teacher (Nature) provides the labels (f(x))*
        - *Recall the definition of a data generating distribution P(X,Y)*

Learning Conjunctions

- **Protocol I:** ***Learner proposes instances as queries to the teacher***
  - **Queries**: Students picks an instance, teacher labels it (<(1,0,1),1>)
  - *What are the question we ask?*
- Since we know we are after a **monotone** conjunction:
  - Is x_{100} in? <(1,1,1…,1,0), ?> f(x)=0 (conclusion: Yes)
  - Is x_{99} in? <(1,1,…1,0,1), ?> f(x)=1 (conclusion: No)
  - Is x_1 in ? <(0,1,…1,1,1), ?> f(x)=1 (conclusion: No)
- *How many queries are needed for learning perfectly?*
  - A straight forward algorithm requires n=100 queries
  - It will produce the hidden conjunction (exactly).

$$h = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

- **Protocol II**: *The teacher (who knows f) provides training examples*
  - *Tell student it's a monotone conjunction*
  - ***How do you start?***
    - <(0,1,1,1,1,0,…,0,1), 1> *(We learned a superset of the good variables)*
  - To show you that all these variables are required
    - <(0,0,1,1,1,0,…,0,1), 0> need x_2
    - <(0,1,0,1,1,0,…,0,1), 0> need x_3
    - …..
    - <(0,1,1,1,1,0,…,0,0), 0> need x_{100}

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

- A straight forward algorithm requires k = 6 examples to produce the hidden conjunction (exactly).

Learning Conjunctions

- **Protocol III**: Some random source (e.g., Nature) provides training examples
- Teacher (Nature) provides the labels (f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

Learning Conjunctions

- Protocol III: Some random source (e.g., Nature) provides training examples
  - **Algorithm: Elimination**
  - Start with the set of all literals as candidates
  - ***Eliminate a literal that is not active (0) in a positive example***

$$f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge \ldots \wedge x_{100}$$

Learning Conjunctions

- **Protocol III**: Some random source (e.g., Nature) provides training examples
  - **Algorithm: Elimination**
  - Start with the set of all literals as candidates
  - ***Eliminate a literal that is not active (0) in a positive example***

$$f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge \ldots \wedge x_{100}$$

<(1,1,1,1,1,1,…,1,1), 1>

<(1,1,1,0,0,0,…,0,0), 0> ← learned nothing

<(1,1,1,1,1,0,...0,1,1), 1> ← **Eliminate literals**

<(1,0,1,1,1,0,...0,1,1), 0> ← learned nothing

<(1,1,1,1,1,0,...0,0,1), 1> ← **Eliminate literals**

<(1,0,1,0,0,0,...0,1,1), 0>

<(1,1,1,1,1,1,…,0,1), 1>

<(0,1,0,1,0,0,...0,1,1), 0>

Final Hypothesis:
(not the target!)

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Is that good ?
We only learned an approximation to the true concept!

Is it GOOD? *Two Directions*

- Follow a probabilistic intuition (PAC)
  - Considers the *Data generating distribution*
  - Never saw x_1=0 in positive examples, maybe we never will?
  - **if** we will, *it will be with small probability*, so the concepts we learn may be pretty good
  - Good: in terms of performance on **future** data
- Mistake Driven Learning Algorithms
  - Learn from a stream of examples, update only when you make a mistake
  - Good: in terms of **how many mistakes** you make before you stop, happy with your hypothesis**.**

Online Learning

- **Online learning**
    - Learn from one example at a time (unlike batch)
    - Update current hypothesis based on that example
- **Mistake (*error*) driven learning**
    - **Update only on mistakes**
    - *Not all online learning algorithms are mistake driven*
- **Discuss two learning algorithms for linear functions**
    - Perceptron and Winnow

Online Learning: *Motivation*

- Consider a learning problem in a very high dimensional space

$$\{x_1, x_2, x_3, \ldots\ldots, x_{1000000}\}$$

- Assume the function space is very *sparse*
  - every function of interest depends on a small number of attributes.)

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge .x_{100}$$

*"I don't know {**whether**, weather} to laugh or cry"*

- Can we develop an algorithm that:
  - Depends only **weakly on the space dimensionality**
  - Mostly on the **number of relevant attributes** ?
- How should we represent the hypothesis?
  - Target function is a conjunction, but we can learn ***others***

Online Learning: *Motivation*

- **Simple + Intuitive + Broadly applicable model**
  - Robot in an assembly line, language learning,…

- **Realistic settings**: very large data sets, when the data cannot fit memory – Streaming data

- **Evaluation**: We will try to make the smallest number of mistakes in the long run (**mistake bounds**)

Online Learning: *Evaluation*

- Model:
  - Instance space: X (*dimensionality – n*)
  - Target: f: X →{0,1}, f ∊ C, concept class (***parameterized by n***)

- Protocol:
  - learner is given x ∊ X
  - learner predicts h(x), and is then given f(x) (*feedback*)

- Performance*: learner makes a mistake when h(x) ≠ f(x)*
  - # mistakes algorithm A makes on sequence S of examples, for target function f

$$M_A(C) = \max_{f \in C, S} M_A(f, S)$$

- A is a mistake bound algorithm for the concept class C, **if** M_A(c) is polynomial in n, the complexity parameter of the target concept.
  - Worse case model – No notion of distribution

Question:

Is it a realistic analysis?

Can we bound the number of mistakes?

Generic Mistake Bound Algorithms

- Let C be a concept class. Learn f ∈ C
- **CON**:
  - In the i-th stage of the algorithm:
    - C_i all concepts in C consistent with all (*i-1*) previously seen examples
    - Choose randomly f∈ C_i and use to predict the next example
- Clearly, $C_{i+1} \subseteq C_i$ and, if a mistake is made on the i^{th} example, then $|C_{i+1}| < |C_i|$ so progress is made.
- *The CON algorithm makes at most |C|-1 mistakes*
- **Can we do better ?**

The **Halving** Algorithm

- Let C be a concept class. Learn f ∈ C
- **Halving**:
- In the i-th stage of the algorithm:
  - C_i all concepts in C consistent with all (i-1) previously seen examples
- Given an example e_i consider the value $f_j(e_i)$ for all $f_j \in C_i$ and **predict by majority**.
- Predict 1 if $|\{f_j \in C_i; f_j(e_i)=0\}| < |\{f_j \in C_i; f_j(e_i)=1\}|$
- Clearly $C_{i+1} \subseteq C_i$ and if a mistake is made in the i-th example,

then $|C_{i+1}| < \frac{1}{2}|C_i|$

- The Halving algorithm makes at most log(|C|) mistakes

The Halving Algorithm

- **Hard to compute** (why?)
- In some cases Halving is optimal (C - class of all Boolean functions)

- We discuss these algorithms since they give us an idea of the ***theoretical bound for mistake driven learning***
  - *Can we find efficient algorithms that are close to the bounds?*

Learning Disjunctions

- There is a hidden disjunction the learner is to learn

$$f = x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_{100}$$

- The number of disjunctions: 3^n
    - log(|C|) = n
- *Can you find a mistake bound algorithm for disjunctions?*
    - The elimination algorithm makes n mistakes
        - *How would you adapt it to the disjunctive case?*
    - The Halving Algorithm makes n mistakes as well

- Great news!
    - We have a mistake bound algorithm for disjunctions!

The Importance of Representation

- **Assume that you want to learn disjunctions. Should your hypothesis space be the class of disjunctions?**

Theorem [Haussler 1988]: *Given a sample on n attributes consistent with a disjunctive concept, it is NP-hard to find a pure disjunctive hypothesis that is both consistent with the sample and has the minimum number of attributes*

- Intuition: Reduction to minimum set cover problem.
- ➔ Cannot learn the concept efficiently **as a disjunction.**
- But, we will see that we can do that, if we are willing to learn the concept as a **Linear Threshold function**.
  - In a more expressive class, the search for a good hypothesis sometimes becomes combinatorially easier.

Summary

- **We want to discuss learning in a formal way**
  - *Did our algorithm learn?*
- Difficult question! *Two approaches*
  - **Probabilistic Intuition**: Data generating distribution
    - Provides a probabilistic measure of “what’s important”
  - ***Mistake driven learning***: simpler (and **strict**) way to discuss learning
    - “when have we made enough mistakes”