Computational Learning Theory

What general laws constrain inductive learning?

We seek theory to relate:

- Probability of successful learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples presented

Slide CS478–1

PAC Learning Setting

Given:

- set of instances X
- set of hypotheses H
- set of possible target concepts C
- training instances generated by a fixed, unknown probability distribution $\mathcal{D}$ over X

Learner observes a seqence D of training examples of form $\langle x, c(x) \rangle$, for some target concept $c \in C$

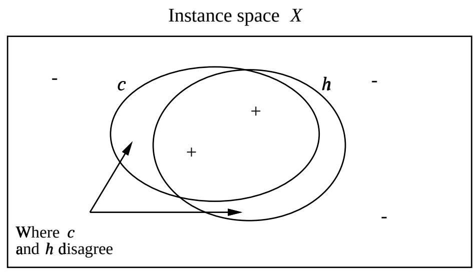
- instances x are drawn from distribution $\mathcal{D}$
- teacher provides target value $c(x)$ for each

Learner must output a hypothesis h estimating c

- h is evaluated by its performance on subsequent instances drawn according to $\mathcal{D}$

Slide CS478–2

True Error of a Hypothesis



Definition: The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Slide CS478–3

PAC Learning

Consider a class C of possible target concepts defined over a set of instances X of length n, and a learner L using hypothesis space H.

> *Definition:* C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$,
>
> learner L will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n and $size(c)$.

Slide CS478–4

Mistake Bounds

So far: how many examples needed to learn?

What about: how many mistakes before convergence?

Let's consider similar setting to PAC learning:

- Instances drawn at random from X according to distribution $\mathcal{D}$
- Learner must classify each instance before receiving correct classification from teacher
- Can we bound the number of mistakes learner makes before converging?

Mistake Bounds: Find-S

Consider Find-S when H = conjuntion of boolean literals

Find-S:

- Initialize h to the most specific hypothesis $l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \ldots l_n \wedge \neg l_n$
- For each positive training instance x
  - Remove from h any literal that is not satisfied by x
- Output hypothesis h.

How many mistakes before converging to correct h?

Mistake Bounds: Halving Algorithm

Consider the Halving Algorithm:

- Learn concept using version space Candidate-Elimination algorithm
- Classify new instances by majority vote of version space members

How many mistakes before converging to correct h?

- ... in worst case?
- ... in best case?

Slide CS478–7

Optimal Mistake Bounds

Let $M_A(C)$ be the max number of mistakes made by algorithm A to learn concepts in C. (maximum over all possible $c \in C$, and all possible training sequences)

$$M_A(C) \equiv \max_{c \in C} M_A(c)$$

Definition: Let C be an arbitrary non-empty concept class. The **optimal mistake bound** for C, denoted $Opt(C)$, is the minimum over all possible learning algorithms A of $M_A(C)$.

$$Opt(C) \equiv \min_{A \in learning\ algorithms} M_A(C)$$

Slide CS478–8

Weighted Majority Algorithm

- generalization of the HALVING algorithm
- makes predictions by taking a weighted vote among a pool of prediction algorithms
- learns by altering the weight associated with each prediction algorithm
- accomodates inconsistent training data
- can bound the number of mistakes made

Tom's slide goes here. Table 7.1.

Relative Mistake Bound for Weighted Majority

Let D be any sequence of training examples.

Let A be any set of n prediction algorithms.

Let k be the minimum number of mistakes made by any algorithm in A for the training sequence D.

Then the number of mistakes of D made by the WEIGHTED-MAJORITY algorithm using $\beta = 1/2$ is at most

$$2.4(k + log_2\ n)$$

Slide CS478–11

Empirical Support for Multiplicative Update Algorithms

Calendar scheduling

Given: Description of an event to be scheduled
Predict: Event's location, duration, start time, day of week.

Features:

- type of event
- name of the seminar
- position of attendees
- are attendees in the user's group
- names of the attendees in alphabetical order

Slide CS478–12

Example

(req-event-type meeting)
(req-seminar-type nil)
(sponsor-attendees no-value)
(department-attendees cs)
(position-attendees faculty)
(group-attendees? no)
(req-course-name nil)
(department-speakers no-value)
(group-name no-value)
(lunchtime? no)
(single-person? yes)
(number-of-person 1)
(req-location dh4301c)

1685 examples

Slide CS478–13

Features of the Learning Task

- *"Target concept"* changes with time.
- Set of possible values for each feature may not be known.

Baseline system: Calendar ApPrentice System

- decision-tree based learning method
- acquires rules sorted by observed performance
- system is run each night using the most recent 180 examples
- merges the new rules into the existing rule set

Slide CS478–14

Weighted Majority Implementation

Assumption: Some small set of features will be enough to construct a good predictor.

1. For each pair of features, create one "expert" (prediction algorithm) that examines only those two features and makes predictions based on their values.
2. Weight update has $\beta = 1/2$
3. Each expert performs a simple table lookup.
   - Given a pair of values for its two features, look at the last k times that the pair of values occurred and predict the outcome that occurred most often out of those k. ($k = 5$)
   - If the pair of values has never occurred before, predict the most common class value seen so far.

Weighted Majority Extension

Speedup strategy:

- Discard experts if their weights drop too low.
- Allows algorithm to speed up as it learns more.
- Danger if too aggressive in discarding experts.
- Found that for a wide range of thresholds, one can achieve both a significant speedup and negligible loss in performance.

Winnow

Combines opinions of "specialists" that can **abstain** on any example.

- Create one specialist for each pair of feature=value conditions seen so far.
- Specialist wakes up to make a prediction if both conditions are true.
- Predicts the most popular outcome out of the last $k = 5$ times it had a chance to predict.
- Global prediction is based on a wighted majority vote over all predicting specialists.

- When specialist i first appears, $w_i \leftarrow 1$ and abstains for this example.
- Weight update strategy:
  - If global prediction incorrect,
    - $w_i = 1/2\ w_i$ for a_i that predict incorrectly
    - $w_i = 3/2\ w_i$ for a_i that predict correctly
  - If global prediction correct,
    - $w_i = 1/2\ w_i$ for a_i that predict incorrectly

Experimental Results

Task	CAP	Winnow	Winnow-big	WM	WM-big
location	0.64	0.75	0.76	0.70	0.74
duration	0.63	0.71	0.74	0.64	0.73
start-time	0.34	0.51	0.53	0.39	0.50
day-of-week	0.50	0.57	0.57	0.56	0.56
AVERAGE	0.53	0.63	0.65	0.57	0.63

Slide CS478–19

Comments

For the Weighted Majority algorithm, the weights answer the question: "if you were only allowed to look at two features, which two do you choose?"

When predicting location,

- best feature: number of people
- best pair: number of people + seminar type

Slide CS478–20

Winnow assigns weights to each possible rule of length 2, indicating the extent to which that rule should be trusted:

- If there is a single attendee and he/she is from the ECE department, then 30 minutes.
- If there is more than one attendee and they are research programmers, the 60 minutes.
- If the attendees are faculty members and not from CMU, then 60 minutes.

Bagging Classifiers

Bagging = **B**ootstrap **agg**regat**ing**

- A learning(data) set L consists of data $\{(y_n, \mathbf{x}_n), n = 1, \ldots, N\}$. Each $\mathbf{x}_n$ is a feature vector; y_n is a class.
- Assume that we have some learning algorithm that can use L to form a classifier $\varphi(\mathbf{x}, L)$ that predicts y given $\mathbf{x}$.
- Given a sequence of data sets $\{L_k\}$ each with N independent observations drawn from the same distribution as L, we can form a sequence of predictors $\{\varphi(\mathbf{x}, L_k)\}$.

Goal in **bagging** is to use the $\{L_k\}$ to get a better predictor than the single data set predictor $\varphi(\mathbf{x}, L)$.

One obvious procedure is to replace $\varphi(\mathbf{x}, L)$ by:

discrete y: the majority vote of the k φ's

numeric y: the average prediction of the k φ's

Slide CS478–23

Bagging Approximates Multiple Data Sets

Take repeated bootstrap samples $\{L^{(B)}\}$ from L and form $\{\varphi(\mathbf{x}, L^{(B)})\}$.

Bootstrap sampling: Given set L containing N training examples, create L^i by drawing N examples at random **with replacement** from L.

Hypothesis: aggregating over bootstrap samples yields higher accuracy than a single classifier.

Bagging:

- Create k bootstrap samples $L^1 \ldots L^k$.
- Train distinct classifier on each L^i.
- Classify new instance by majority vote / average.

Slide CS478–24

Experimental Method

Given sample S of labeled data, do 100 times and report average

1. Split S randomly into test set T (10%) and training set D (90%).
2. Learn decision tree from D
   - $e_S \leftarrow$ error of tree on T
3. Repeat 50 times: Create bootstrap set D^i, construct decision tree using D.
   - $e_B \leftarrow$ error of majority vote using trees to classify T

Slide CS478–25

Results

Data Set	e_S	e_B	Decrease
Waveform	29.1	19.3	34%
Heart	4.9	2.8	43%
Breast Cancer	5.9	3.7	37%
Ionosphere	11.2	7.9	29%
Diabetes	25.3	23.9	6%
Glass	30.4	23.6	22%
Soybean	8.6	6.8	21%

Slide CS478–26

How many bootstrap samples are enough

Number bootstrap samples	Misclassification Rate
1	29.1
10	21.8
25	19.4
50	19.3
100	19.3

Slide CS478–27

When Will Bagging Improve Accuracy?

Depends on the stability of the base-level classifiers.

A learner is *unstable* if a small change to the training set causes a large change in the output hypothesis.

- If small changes in L cause small changes in φ then $\varphi \approx \varphi_B$.
- If small changes in L cause large changes in φ then there will be an improvement in performance.

Slide CS478–28

Conclusion of Experiments

- Bagging helps unstable procedures.
- Bagging hurts the performance of stable procedures.
- Neural nets, decision/regression trees, linear regression are unstable.
- k-nn is stable.

Bagging Nearest Neighbor Classifiers

No difference between e_S and e_B

Reason:

Probability than a particular instance will be in any one Bootstrap replicate is .632

An instance x will have a different label predicted for it by the aggregate method only if x's nearest neighbor is missing from at least half of bootstrap learning sets

The probability of this happening is P(number of heads in N tosses is less than N/2) when the probability of a head is .632

Clearly as N grows this gets small.