Overview

- Experimental methodology
- Probably approximately correct learning
- Mistake bounds
- Vapnik-Chervonenkis dimension

Experimental methodology

- How do we measure performance of a learning algorithm?
- We need to measure on test data
- Need to perform experiment multiple times

Best possible method

- Divide data into many partitions
- Each partition has training and test sets
- Train on training data, test on test set
- Compute average performance on partitions
Problem

- Requires lots of data!
- Only train on small fraction of data

Cross validation

- Divide data into M folds
- Run M experiments
- Each experiment: use M-1 folds for training, remaining fold for testing
- Use different fold as test set each time
- Compute average performance over experiments

5-fold cross-validation

<table>
<thead>
<tr>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Test</td>
</tr>
</tbody>
</table>

Cross-validation, validation set

- Different use of “validation”
- Suppose you want to measure the performance of decision trees with validation set pruning
- You can use cross-validation
- Validation set needs to be separate from both training and test sets!
### 5-fold cross-validation

<table>
<thead>
<tr>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
<th>Training</th>
</tr>
</thead>
</table>

### Overview
- Experimental methodology
- Probably approximately correct learning
- Mistake bounds
- Vapnik-Chervonenkis dimension

### Why study learning theory
- Provide performance guarantees
- Determine how many samples we need
- Develop new algorithms

### Inductive bias
- With inductive bias, possibility of learning
- How much learning is possible with a particular inductive bias?
- What does “how much learning” mean?

### PAC learning
- PAC: probably approximately correct
- With high probability, learner will generate good hypothesis

### Approximately correct
- Unreasonable to expect to learn perfectly correct hypothesis
  - more than one consistent hypothesis
  - can’t guarantee we will choose the right one
- **Approximately** correct = learns good hypothesis
  - “good” = error < $\varepsilon$
Probably approximately correct

- Unreasonable to expect good hypothesis for every training set
  - training set may be unlucky
- **Probably** approximately correct = good on most training sets
  - "most" = probability > 1-δ

Assumptions

- Distribution P over instances \( \langle X, C \rangle \)
- Distribution is stationary
  - doesn’t change over time
  - same for future instances as training instances
- Instances are independent samples from P
- i.i.d: “independently identically distributed”

Definitions of error

- \( \text{error}_D(h) \) =
  - error of h on training set
  - fraction of training set misclassified by h
- \( \text{error}_P(h) \) =
  - error of h on future instances
  - probability h will misclassify an instance \( \langle x, c \rangle \) \~ P
  - \( P(h(x) \neq c) \)

Probably approximately correct

- Learner L produces hypothesis based on data \( D \)
- \( P(\text{error}_P(L(D)) < \epsilon) > 1 - \delta \)
- \( P(\text{error}_P(L(D)) < \epsilon) \) is taken under probability distribution over training sets

Definition: PAC-learnability

- \( n \) = number of attributes, \( H \) = hypothesis space
- \( H \) is PAC-learnable if
  - there exists a learning algorithm L such that
    - for every deterministic domain whose true model \( f \in H \)
    - for every distribution P over instances
      - for \( 0 < \epsilon < 0.5 \) and \( 0 < \delta < 0.5 \)
        - L will, with probability > 1-δ, output a hypothesis \( h \in H \) such that \( \text{error}(h) < \epsilon \)
      - in time polynomial in \( 1/\epsilon, 1/\delta, n \)

Polynomial time

- This requires
  - Polynomial number of training instances
  - Polynomial run time given a training set
- Neither implies the other
- Required number of training instances is usually the limiting factor
- How many instances are required?
Definition: sample complexity

- Number of training instances \( N \) required to guarantee that
  - for every distribution \( P \) over instances
  - for every true model \( f \in H \)
    - with probability \( \geq 1-\delta \), every hypothesis \( h \in H \) consistent with training set of size \( N \) has \( \text{error}_P(h) \leq \varepsilon \)

Sample complexity

- What is a bad hypothesis?
  - \( \text{error}_P(h) > \varepsilon \)
- Given one training instance \( x \), what is probability that bad hypothesis is consistent with it?
  - \( P(\text{bad hyp. consistent with one instance}) \leq 1-\varepsilon \)
- Given \( N \) training instances, what is probability that bad hypothesis is consistent with all of them?
  - \( P(\text{bad hyp. consistent with } N \text{ instances}) \leq (1-\varepsilon)^N \)
  - (assuming instances are independent)

Sample complexity

- \( P(\text{bad hyp. consistent with one instance}) \leq 1-\varepsilon \)
- \( P(\text{bad hyp. consistent with } N \text{ instances}) \leq (1-\varepsilon)^N \)

- What is probability that there is any bad hypothesis consistent with \( N \) training instances?

Union bound

\[
P(A_1 \lor A_2 \lor \ldots \lor A_n) \leq P(A_1) + P(A_2) + \ldots + P(A_n)
\]

\[
P(\text{any bad hyp. consistent with } N \text{ instances}) \leq P(\text{bad hyp. 1 consistent with instances}) + P(\text{bad hyp. 2 consistent with instances}) + \ldots + P(\text{bad hyp. } h_n \text{ consistent with instances})
\]

\[
\leq (\# \text{ bad hyp.})^* (1-\varepsilon)^N
\]

Number of bad hypotheses

- Upper bound on number of bad hypotheses?
  - \( |H| \)
- \( P(\text{any bad hyp. consistent with } N \text{ instances}) \leq |H|(1-\varepsilon)^N \)

- Upper bound
  - may be quite weak

Back to sample complexity

- \( P(\text{any bad hyp. consistent with } N \text{ instances}) \leq |H|(1-\varepsilon)^N \)
- How many instances are required to guarantee that this probability is \( < \delta \)?
  - \( |H|(1-\varepsilon)^N < \delta \)
  - ...
  - \( N > \frac{1}{\varepsilon} \left( \ln |H| + \ln \frac{1}{\delta} \right) \)
Definition: PAC-learnability

• $n =$ number of attributes, $H =$ hypothesis space
• $H$ is PAC-learnable if
- there exists a learning algorithm $L$ such that
  • for every deterministic domain whose true model $f \in H$
  • for every distribution $P$ over instances
  • for $0 < \varepsilon < 0.5$ and $0 < \delta < 0.5$
- $L$ will, with probability $> 1 - \delta$, output a hypothesis $h \in H$ such that $\text{error}_P(h) < \varepsilon$
  - in time polynomial in $1/\varepsilon$, $1/\delta$, $n$

Conclusion

• $N > \frac{1}{\varepsilon} \left( \ln |H| + \ln \frac{1}{\delta} \right)$
• Upper bound on sample complexity
  - smaller $N$ might suffice
• Always polynomial in $1/\varepsilon$ and $1/\delta$
• Polynomial in $n$?
  - depends on $H$

Example: conjunctive concepts

• Literal: attribute or its negation
  - e.g. $X_1, \neg X_2$
• Conjunctive formula: conjunctions of literals
  - e.g. $\neg X_3 \land X_2$
• Let $H$ be set of conjunctive formulas

Sample complexity

• How big is $|H|$ for $n$ attributes?
  - each attribute appears positively, negatively, not at all
  - $3^n$
• Sample complexity: $N > \frac{1}{\varepsilon} \left( n \ln 3 + \ln \frac{1}{\delta} \right)$
  - polynomial sample complexity
  - conjunctive formulae are PAC-learnable

Example: Boolean formulae

• Any formula from $n$ attributes to a Boolean class
• How big is $|H|$?
  - a truth table has $2^n$ rows
  - each row can have one of two classes (T/F)
  - So $2^{2^n}$ possible hypotheses
• Sample complexity: $N > \frac{1}{\varepsilon} \left( 2^n \ln 2 + \ln \frac{1}{\delta} \right)$

PAC-learnability

• $N > \frac{1}{\varepsilon} \left( 2^n \ln 2 + \ln \frac{1}{\delta} \right)$
• Looks like Boolean formulae have exponential sample complexity
• Not a proof: upper bound
• In fact, they do have exponential sample complexity
  - not surprising: no inductive bias
**Example: k-term DNF**

- Disjunction of k conjunctive formulae
  \[ c_1 \lor c_2 \lor \ldots \lor c_k \]
- where \( c_i \) is conjunctive formula of \( n \) attributes
- Size of \( |H| \)?
  - each conjunction has \( 3^n \)
  - k conjunctions: \( (3^n)^k = 3^{nk} \)

**Sample complexity: k-term DNF**

- \( |H| = 3^{nk} \)
- Sample complexity: \( N > \frac{1}{\epsilon} \left( nk \ln 3 + \frac{1}{\delta} \right) \)
- Polynomial sample complexity
- But no polynomial algorithm (unless RP=NP)
- Not PAC-learnable

**Conjecture**

If a hypothesis space is PAC-learnable, so is any subset of that space

**Example: k-CNF**

- Arbitrary length conjunction of disjunctive terms
  \[ d_1 \land d_2 \land \ldots \land d_j \]
- where \( d_i \) is disjunction of \( k \) attributes
- Fact: k-term DNF \( \subset \) k-CNF
  - k-term DNF formula can be rewritten as k-CNF
- But k-CNF is PAC-learnable
  - polynomial computation complexity per instance

**A true statement**

If a hypothesis space is PAC-learnable, so is any subset of that space

If a hypothesis space has polynomial sample complexity, so does any subset of that hypothesis space.

**Assumption**

- So far, we assumed that the true concept was in out hypothesis space
- What if it is not?
- Two possibilities:
  - there exists a hypothesis consistent with training set
  - all hypotheses have some error on training set
First case: consistent learner

Consistent learner either
- produces hypothesis consistent with training set if one exists
- reports failure

First case: example

- $H = \text{conjunctive formulae}$
- True concept: $X_2 \lor X_3$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>P</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>P</td>
</tr>
</tbody>
</table>

Hypothesis $X_2$ is consistent with training set

Consistent learner: guarantee

- $L$ is consistent learner with hypothesis space $H$
- If $N$ training examples are sufficient that with high probability, no bad hypotheses in $H$ will be consistent with the training data...
- Then a hypothesis returned by $L$ is probably approximately correct
  - even if not consistent with true concept

Second case: agnostic learner

- Returns hypothesis with minimum error on training set
- Agnostic because no assumption that true concept contained in hypothesis space

Reminder: definitions of error

- $\text{error}_D(h) =$ error of $h$ on training set
- $\text{error}_P(h) =$ error of $h$ on future instances

If we know $\text{error}_D(h)$, can we say something about $\text{error}_P(h)$?
Hoeffding bound

\[ P(\text{error}_D(h) > \text{error}_p(h) - \varepsilon) \leq e^{-2N\varepsilon^2} \]

Union bound

\[ P(\text{error}_D(h) > \text{error}_p(h) - \varepsilon) \leq e^{-2N\varepsilon^2} \]
\[ P(\exists h \in H : \text{error}_D(h) > \text{error}_p(h) - \varepsilon) \leq |H|e^{-2N\varepsilon^2} \]

Minimum-error hypothesis

\[ P(\text{error}_D(h) > \text{error}_p(h) - \varepsilon) \leq e^{-2N\varepsilon^2} \]
\[ P(\exists h \in H : \text{error}_D(h) > \text{error}_p(h) - \varepsilon) \leq |H|e^{-2N\varepsilon^2} \]
\[ P(\text{error}_D(h^*) > \text{error}_p(h^*) - \varepsilon) \leq |H|e^{-2N\varepsilon^2} \]
where h* is the hypothesis returned by L

Agnostic learner: guarantee

- Bound probability by \( \delta \), do some algebra...
- If \( N > \frac{1}{2\varepsilon^2} \left( \ln|H| + \ln\frac{1}{\delta} \right) \)
- Then with probability 1-\( \delta \), error on future instances is less than \( \varepsilon \) greater than error on training set
- Assumes same distribution for training and test sets

What’s wrong with this statement?

I have tried several learning algorithms on my data, and algorithm A has the best performance on the test set. The hypothesis space for algorithm A has sample complexity N, and I have more than N training instances. Therefore, I expect that its performance on real-world data will, with high probability, be close to its performance on the test set.

The design process

- Try multiple algorithms, say A and B
- Choose one with best performance on test set
- Any hypothesis in \( H_A \) or \( H_B \) could be returned
- Effective H is \( H_A \cup H_B \)
Meta learning algorithm

- Learning algorithm design process is a more powerful learning algorithm
- Test set is functioning like validation set
- Hypothesis class is larger than individual hypothesis classes
- Sample complexity is larger

Overview

- Experimental methodology
- Probably approximately correct learning
- Mistake bounds
- Vapnik-Chervonenkis dimension

Mistake bound

- Online learner
  - receives instance \( x \)
  - predicts class for \( x \)
  - provided with correct class, updates hypothesis
- How many mistakes will learner make before converging to correct concept?

Example: conjunctive formulae

FindS(D,X) =
  Initialize h to \( X_1 \land \neg X_1 \land X_2 \land \neg X_2 \land \ldots \land X_n \land \neg X_n \)
  For each positive training instances \( x \in D \)
    Remove from h any literal not satisfied by \( x \)
  Output h

Example: conjunctive formulae

\[ h = X_1 \land \neg X_1 \land X_2 \land \neg X_2 \land X_3 \land \neg X_3 \land X_4 \land \neg X_4 \]

Training data:

<table>
<thead>
<tr>
<th>Example</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>P</td>
</tr>
</tbody>
</table>

After example 1:

\[ h = X_1 \land X_2 \land \neg X_3 \land X_4 \]
Example: conjunctive formulae

\[ h = X_1 \land X_2 \land \neg X_3 \land X_4 \]

Training data:

<table>
<thead>
<tr>
<th>Example</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>P</td>
</tr>
</tbody>
</table>

After example 2:
\[ h = X_1 \land X_2 \land X_4 \]

Example: conjunctive formulae

\[ h = X_1 \land X_2 \land \neg X_3 \land X_4 \]

Training data:

<table>
<thead>
<tr>
<th>Example</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>P</td>
</tr>
</tbody>
</table>

Skip example 3 (negative example)
\[ h = X_1 \land X_2 \land X_4 \]

Example: conjunctive formulae

\[ h = X_1 \land X_2 \land \neg X_3 \land X_4 \]

Training data:

<table>
<thead>
<tr>
<th>Example</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>P</td>
</tr>
</tbody>
</table>

After example 4 (final hypothesis):
\[ h = X_1 \land X_2 \]

Mistake bound

- Assume consistent domain
- Assume true concept is in hypothesis class
- How many mistakes will FindS make before it converges on true concept?

Example: halving algorithms

- Start with set of hypotheses \( H \)
- For each instance, remove inconsistent hypotheses from \( H \)
- To classify instance, majority vote
  - majority say positive, classify positive, else negative
- What is mistake bound of halving algorithms?

FindS mistake bound

- Will algorithm ever misclassify negative instances as positive?
  - no: hypothesis is most specific
- Initial hypothesis has \( 2n \) terms (T/F for each attribute)
- First instance (always misclassified): how many terms are removed from hypothesis?
  - \( n \)
- For subsequent mistakes, how many terms are removed?
  - at least 1
- After first mistake, only \( n \) terms, so only \( n \) more mistakes before hypothesis is empty: mistake bound = \( n+1 \)
### Halving algorithm mistake bound

- If instance $x$ is misclassified, how many hypotheses in $H$ are inconsistent with $x$?
  - at least $|H|/2+1$
- So at most $|H|/2$ hypotheses remain
- After $\log_2(|H|)$ mistakes, only one hypothesis can remain

### Overview

- Experimental methodology
- Probably approximately correct learning
- Mistake bounds
- Vapnik-Chervonenkis dimension

### Terminology

- A **dichotomy** on $S$ is a partition of $S$ into positive and negative instances
  - For $|S|=n$, $2^n$ dichotomies
- Every hypothesis represents a dichotomy
- Hypothesis space $H$ **captures** a dichotomy $d$ if there exists $h \in H$ such that $h$ represents $d$
- Hypothesis space $H$ **shatters** set $S$ if it captures every dichotomy in $S$.

### VC dimension

The **VC dimension** of a hypothesis space $H$ is the size of the largest set shattered by $H$, if finite, otherwise it is $\infty$.

### Example: number line

- $X = $ real numbers on the line
- $H = $ closed intervals

What is the VC dimension of $H$?

### Can $H$ shatter two points?

**Yes**

![Diagram of number line with closed intervals and points]
Can H shatter three points?

No

+ - +

Therefore, VC(H) = 2

Example: plane

- X = the x,y-plane
- H = linearly separable hypotheses in the plane
- What is the VC dimension of H?

Can H shatter two points?

Yes: same argument as before

Can H shatter three points?

Yes

Sometimes yes, sometimes no

What does this say about VC(H)?

Sometimes yes, sometimes no

Definition

The VC dimension of a hypothesis space H is the size of the largest set shattered by H, if finite, otherwise it is $\infty$.

Therefore, VC(H) is at least 3...

Can H shatter four points?

No

+ - +

This cover all cases

Therefore VC(H) = 3
In general

• \(X = \text{point in } n \text{ dimensions}\)
• \(H = \text{linear separators}\)
• \(\text{VC}(H) = n+1\)

Example: Boolean conjunctions

• \(X = \text{conjunctions of three literals}\)
• \(H = \text{conjunctions of up to three literals}\)
• What is the VC dimension of \(H\)?

Construct dichotomy

• Consider three instances

<table>
<thead>
<tr>
<th>#</th>
<th>(X_1)</th>
<th>(X_2)</th>
<th>(X_3)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

• For any set of class labels, construct dichotomy:
  - to exclude instance \(i\), add \(\neg X_i\) to hypothesis
• Argument can be applied to \(n\) literals

Boolean conjunctions

• Construction argument says that \(\text{VC}(H)\) is at least \(n\) (for \(n\) literals)
• Harder to show exactly \(n\): most prove that \(n+1\) instances can't be shattered

In general

• Can we bound \(\text{VC}(H)\) based on \(|H|\)?
• Suppose \(\text{VC}(H) = d\)
• How many hypotheses are required to shatter \(d\) instances?
  - \(2^d\)
• So \(\text{VC}(H) \leq \log_2|H|\)

VC dimension and sample complexity

• Before we bounded sample complexity using the size of the entire hypothesis space:
  \[
  N > \frac{1}{\varepsilon} \left( \ln |H| + \ln \frac{1}{\delta} \right)
  \]
• We can use VC dimension as a measure of the complexity of \(H\) instead
  \[
  N > \frac{1}{\varepsilon} \left( 4 \log_2 \frac{2}{\delta} + 8 \text{VC}(H) \log_2 \frac{13}{\varepsilon} \right)
  \]
Sample complexity lower bound

- Upper bound: how many samples are *sufficient* for successful learning
- With VC dimension, we can bound the number of samples *necessary* for successful learning:

\[ N < \max \left[ \frac{1}{\varepsilon} \log \frac{1}{\delta}, \frac{\text{VC}(F) - 1}{32\varepsilon} \right] \]

- VC(F) is VC dimension of concept class