Further Topics in Computational Learning Theory

Lecture Outline

- Brief Review of the PAC-Learning Model
- Sample Complexity for Infinite Hypothesis Spaces
  - "Shattering" Sets of Instances
  - Vapnik-Chervonenkis (VC) Dimension
  - Sample Complexity and VC Dimension
- The Mistake Bound Model of Learning
  - Mistake Bound for FIND-S
  - Mistake Bound for the Halving Algorithm
  - Optimal Mistake Bounds
  - Weighted-Majority Algorithm

Reading

Mitchell, Chapter 7.4-7.5

M. Anthony and N. Briggs *Computational Learning Theory*, Cambridge University Press, Cambridge, 1992.

Sample Complexity for Infinite Hypothesis Spaces

- Our initial central result on sample complexity for consistent learners over finite hypothesis spaces was:

$$m \geq \frac{1}{\varepsilon}(\ln |H| + \ln(1/\delta))$$

  I.e. m training examples are required to ensure with probability $1 - \delta$ that any consistent hypothesis will be correct with true error no more than ε.

- Note that sample complexity here grows as the log of the size of the hypothesis space H.

Sample Complexity for Infinite Hypothesis Spaces

- Our initial central result on sample complexity for consistent learners over finite hypothesis spaces was:

$$m \geq \frac{1}{\varepsilon}(\ln |H| + \ln(1/\delta))$$

  I.e. m training examples are required to ensure with probability $1 - \delta$ that any consistent hypothesis will be correct with true error no more than ε.

- Note that sample complexity here grows as the log of the size of the hypothesis space H.

- This result has two limitations:
  1. It leads to weak bounds (overestimates of sample complexity)
  2. It only applies to finite hypothesis spaces

Sample Complexity for Infinite Hypothesis Spaces

- Our initial central result on sample complexity for consistent learners over finite hypothesis spaces was:

$$m \geq \frac{1}{\varepsilon}(\ln |H| + \ln(1/\delta))$$

  I.e. m training examples are required to ensure with probability $1 - \delta$ that any consistent hypothesis will be correct with true error no more than ε.

- Note that sample complexity here grows as the log of the size of the hypothesis space H.
- This result has two limitations:
  1. It leads to weak bounds (overestimates of sample complexity)
  2. It only applies to finite hypothesis spaces
- An alternative is to consider other measures for the complexity of H, aside from simply its size.
- One such measure is the **Vapnik-Chervonenkis (VC) dimension**. VC dimension:
  1. Allows tighter bounds on sample complexity
  2. Can be applied to infinite hypothesis spaces

“Shattering” Sets of Instances

- *Definition:* a **dichotomy** of a set S is a partition of S into two disjoint subsets.

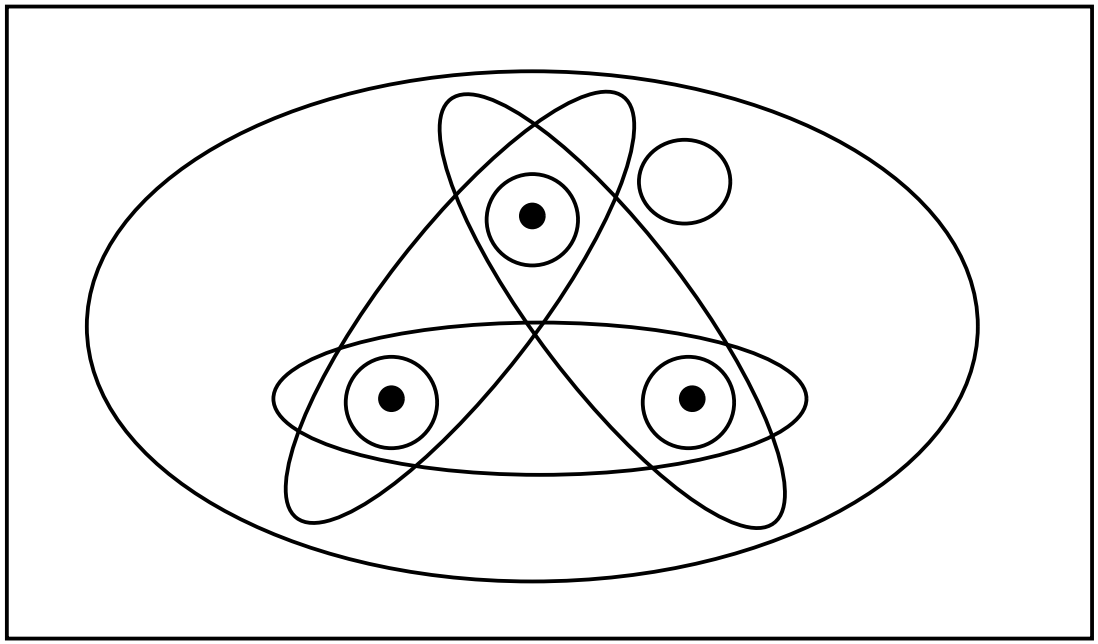
"Shattering" Sets of Instances

- *Definition:* a **dichotomy** of a set S is a partition of S into two disjoint subsets.
- *Definition:* a set of instances S is **shattered** by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.

"Shattering" Sets of Instances

- *Definition:* a **dichotomy** of a set S is a partition of S into two disjoint subsets.
- *Definition:* a set of instances S is **shattered** by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.
- Example: A set of 3 instances shattered by 8 hypotheses.

Instance space X



Vapnik-Chervonenkis (VC) Dimension

- *Definition:* The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

Vapnik-Chervonenkis (VC) Dimension

- *Definition:* The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

- Recall earlier notion of *unbiased hypothesis space* – one capable of representing every possible concept (dichotomy) definable over the instance space X.

  So, an unbiased hypothesis space is one that shatters X.

Vapnik-Chervonenkis (VC) Dimension

- *Definition:* The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

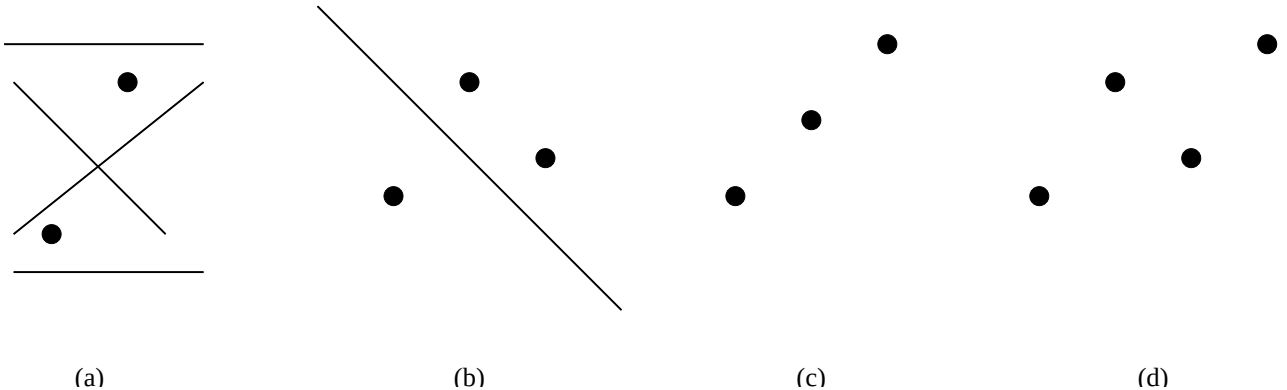
- Recall earlier notion of *unbiased hypothesis space* – one capable of representing every possible concept (dichotomy) definable over the instance space X.

  So, an unbiased hypothesis space is one that shatters X.

- The intuition behind VC dimension is that the larger the subset of X that can be shattered by H, the more expressive is H.

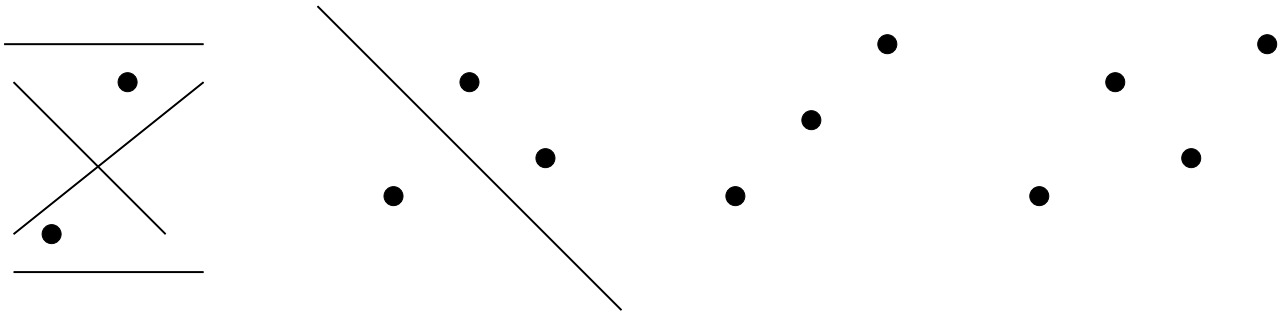
  Hence, VC dimension serves as an alternative measure of the complexity of H – alternative to simple size.

Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane

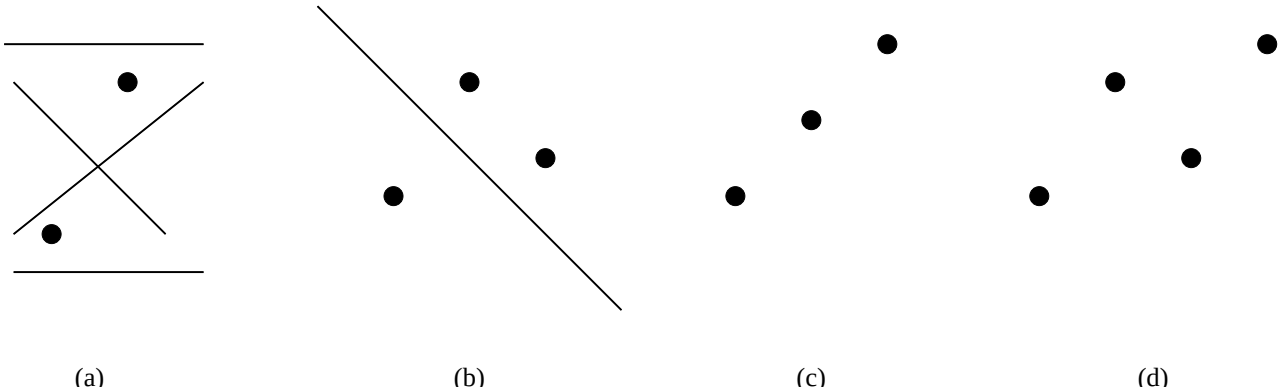
Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



(a) (b) (c) (d)

- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane
- Note
  1. can shatter two points with $2^2 = 4$ lines (a)

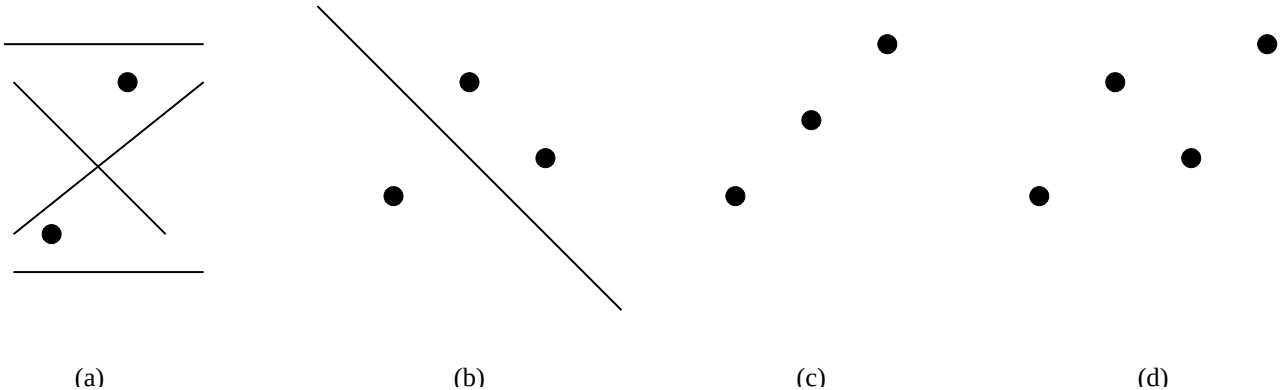
Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



(a) (b) (c) (d)

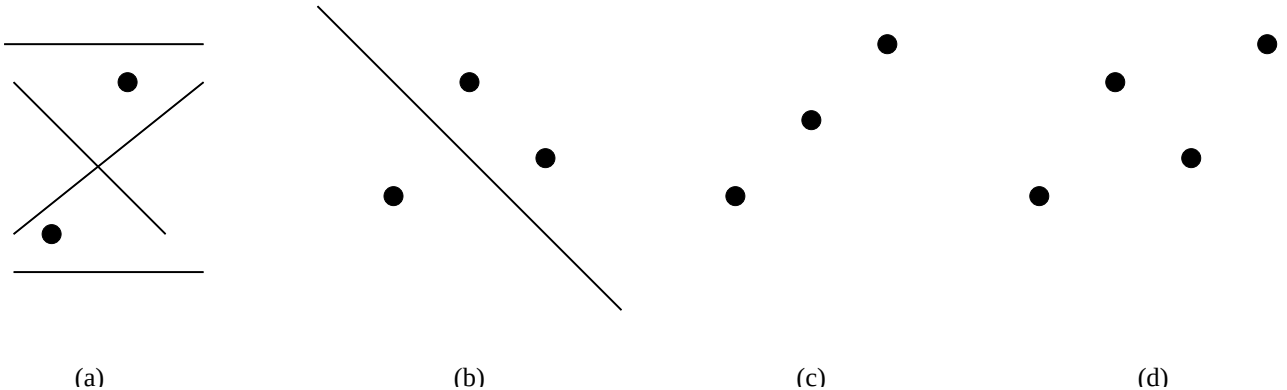
- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane
- Note
  1. can shatter two points with $2^2 = 4$ lines (a)
  2. can shatter three non-colinear points with $2^3 = 8$ lines (b)

Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



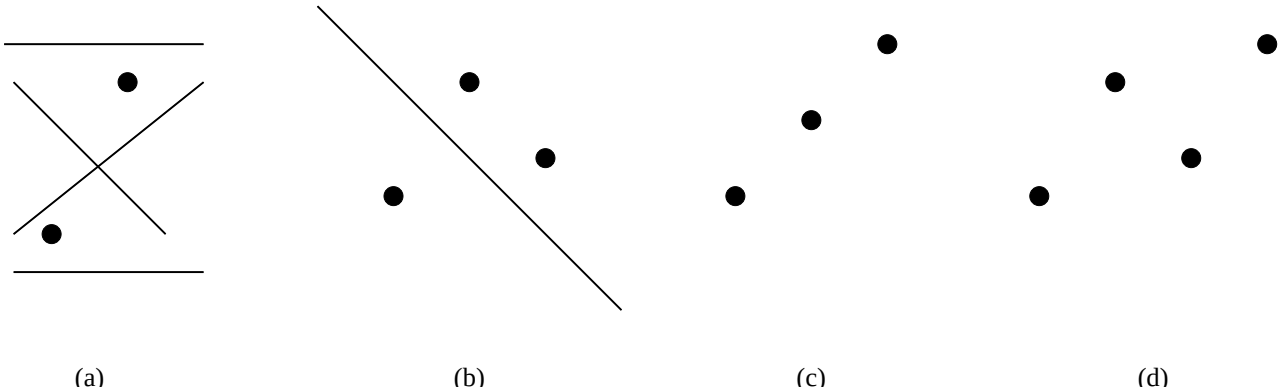
- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane
- Note
  1. can shatter two points with $2^2 = 4$ lines (a)
  2. can shatter three non-colinear points with $2^3 = 8$ lines (b)
  3. cannot shatter three colinear points (c)

Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



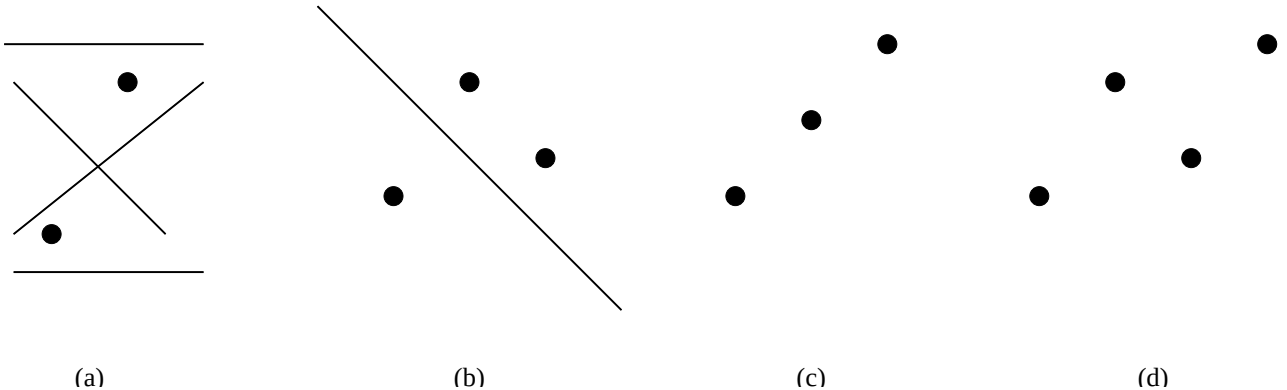
- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane
- Note
  1. can shatter two points with $2^2 = 4$ lines (a)
  2. can shatter three non-colinear points with $2^3 = 8$ lines (b)
  3. cannot shatter three colinear points (c)
  4. cannot shatter any set of four points (d)

Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane
- Note
  1. can shatter two points with $2^2 = 4$ lines (a)
  2. can shatter three non-colinear points with $2^3 = 8$ lines (b)
  3. cannot shatter three colinear points (c)
  4. cannot shatter any set of four points (d)
- Therefore VC dimension of H ($VC(H)$) defined over X is 3.

Vapnik-Chervonenkis (VC) Dimension: Example – Linear Decision Surfaces



- Suppose
  - X is the set of points in the x, y plane
  - H is the set of linear decision surfaces in the plane
- Note
  1. can shatter two points with $2^2 = 4$ lines (a)
  2. can shatter three non-colinear points with $2^3 = 8$ lines (b)
  3. cannot shatter three colinear points (c)
  4. cannot shatter any set of four points (d)
- Therefore VC dimension of H ($VC(H)$) defined over X is 3.
- More generally, VC dimension of linear decision surfaces in an r dimensional space is $r+1$

Sample Complexity and VC Dimension

- Using VC dimension as a measure of hypothesis space complexity it is now possible to calculate a new **upper bound** on sample complexity.

  m randomly drawn examples suffice to ε-exhaust $VS_{H,D}$ with probability at least $(1-\delta)$, where

$$m \geq \frac{1}{\varepsilon}(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon))$$

Sample Complexity and VC Dimension

- Using VC dimension as a measure of hypothesis space complexity it is now possible to calculate a new **upper bound** on sample complexity.

  m randomly drawn examples suffice to ε-exhaust $VS_{H,D}$ with probability at least $(1-\delta)$, where

$$m \geq \frac{1}{\varepsilon}(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon))$$

- A **lower bound** on the number of training examples to ensure PAC-learnability of every concept in C has also been established.

  Theorem: Consider any concept class C such that $VC(C) \geq 2$, any learner L, and any $0 < \varepsilon < \frac{1}{8}$, and $0 < \delta < \frac{1}{100}$. Then there exists a distribution $\mathcal{D}$ and target concept C such that if L observes fewer examples than

$$max\left[\frac{1}{\varepsilon}log(1/\delta), \frac{VC(C)-1}{32\varepsilon}\right]$$

  then with probability at least δ, L outputs a hypothesis h having error $error_{\mathcal{D}} > \varepsilon$.

The Mistake Bound Model of Learning

- So far we have addressed the question:

  How many examples are needed to learn?

The Mistake Bound Model of Learning

- So far we have addressed the question:

  How many examples are needed to learn?

- An alternative question that leads to a different framework for characterising learning algorithms is:

  How many mistakes does the learner make before converging to a correct hypothesis?

The Mistake Bound Model of Learning

- So far we have addressed the question:

  How many examples are needed to learn?

- An alternative question that leads to a different framework for characterising learning algorithms is:

  How many mistakes does the learner make before converging to a correct hypothesis?

- Consider a 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

  Ask: Can we bound the number of mistakes learner makes before converging?

The Mistake Bound Model of Learning

- So far we have addressed the question:

  How many examples are needed to learn?

- An alternative question that leads to a different framework for characterising learning algorithms is:

  How many mistakes does the learner make before converging to a correct hypothesis?

- Consider a 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

  Ask: Can we bound the number of mistakes learner makes before converging?

- Mistake bound model can be of practical interest in settings where learning must take place during use of system, rather than in off-line training stage.

  Clearly want to minimise errors during this process.

Mistake Bound for FIND-S

- Consider FIND-S when H = conjunction of boolean literals (e.g. *Studies* $\wedge$ *Attends_Lectures* $\wedge \neg$ *Drinks*)

  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 Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):

Mistake Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):
  1. FIND-S never mistakenly classifies a negative example (current hypothesis always at least as specific as c).

Mistake Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):
  1. FIND-S never mistakenly classifies a negative example (current hypothesis always at least as specific as c).
  2. So, total mistakes made by FIND-S will be those made in classifying positive examples as negative.

Mistake Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):
  1. FIND-S never mistakenly classifies a negative example (current hypothesis always at least as specific as c).
  2. So, total mistakes made by FIND-S will be those made in classifying positive examples as negative.
  3. Since initial hypothesis predicts every instance is negative, first positive example will be misclassified.

Mistake Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):
  1. FIND-S never mistakenly classifies a negative example (current hypothesis always at least as specific as c).
  2. So, total mistakes made by FIND-S will be those made in classifying positive examples as negative.
  3. Since initial hypothesis predicts every instance is negative, first positive example will be misclassified.
  4. But, first positive example eliminates $1/2$ of the $2n$ literal terms in h, leaving n terms.

Mistake Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):
  1. FIND-S never mistakenly classifies a negative example (current hypothesis always at least as specific as c).
  2. So, total mistakes made by FIND-S will be those made in classifying positive examples as negative.
  3. Since initial hypothesis predicts every instance is negative, first positive example will be misclassified.
  4. But, first positive example eliminates $1/2$ of the $2n$ literal terms in h, leaving n terms.
  5. Each subsequent misclassified positive example will cause at least one further term to be removed.

Mistake Bound for FIND-S

- Reason as follows (supposing target concept $c \in H$):
  1. FIND-S never mistakenly classifies a negative example (current hypothesis always at least as specific as c).
  2. So, total mistakes made by FIND-S will be those made in classifying positive examples as negative.
  3. Since initial hypothesis predicts every instance is negative, first positive example will be misclassified.
  4. But, first positive example eliminates $1/2$ of the $2n$ literal terms in h, leaving n terms.
  5. Each subsequent misclassified positive example will cause at least one further term to be removed.
  6. Therefore, in the worst case, the total number of mistakes will be $n+1$.

Mistake Bound for the 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?

Mistake Bound for the 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?
- Reason as follows (again, suppose target concept $c \in H$):
  1. New instance misclassified only when majority of hypotheses in current version space misclassify it.

Mistake Bound for the 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?
- Reason as follows (again, suppose target concept $c \in H$):
  1. New instance misclassified only when majority of hypotheses in current version space misclassify it.
  2. Once correct classification is revealed, only correct hypotheses are retained in version space.

Mistake Bound for the 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?
- Reason as follows (again, suppose target concept $c \in H$):
  1. New instance misclassified only when majority of hypotheses in current version space misclassify it.
  2. Once correct classification is revealed, only correct hypotheses are retained in version space.
  3. Therefore each mistake causes version space to be reduced to at most half its current size.

Mistake Bound for the 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?
- Reason as follows (again, suppose target concept $c \in H$):
  1. New instance misclassified only when majority of hypotheses in current version space misclassify it.
  2. Once correct classification is revealed, only correct hypotheses are retained in version space.
  3. Therefore each mistake causes version space to be reduced to at most half its current size.
  4. Since initial version space contains $|H|$ members, maximum number of mistakes before version space contains one member is $\lfloor log_2(|H|) \rfloor$ (worst case result).

Mistake Bound for the 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?
- Reason as follows (again, suppose target concept $c \in H$):
  1. New instance misclassified only when majority of hypotheses in current version space misclassify it.
  2. Once correct classification is revealed, only correct hypotheses are retained in version space.
  3. Therefore each mistake causes version space to be reduced to at most half its current size.
  4. Since initial version space contains $|H|$ members, maximum number of mistakes before version space contains one member is $\lfloor log_2(|H|) \rfloor$ (worst case result).
  5. In best case, majority is always right, minority hypotheses are removed from version space at each step, and HALVING algorithm converges to a single member version space making 0 errors.

Optimal Mistake Bounds

- Have considered mistake bounds for two specific algorithms.

  Can also ask: what is optimal mistake bound for arbitrary concept class C?

  I.e., what is the lowest worst-case mistake bound over all possible learning algorithms?

Optimal Mistake Bounds

- Have considered mistake bounds for two specific algorithms.

  Can also ask: what is optimal mistake bound for arbitrary concept class C?

  I.e., what is the lowest worst-case mistake bound over all possible learning algorithms?

- 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)

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

Optimal Mistake Bounds

- Have considered mistake bounds for two specific algorithms.

  Can also ask: what is optimal mistake bound for arbitrary concept class C?

  I.e., what is the lowest worst-case mistake bound over all possible learning algorithms?

- 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 $M_A(C)$.

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

Optimal Mistake Bounds

- Have considered mistake bounds for two specific algorithms.

  Can also ask: what is optimal mistake bound for arbitrary concept class C?

  I.e., what is the lowest worst-case mistake bound over all possible learning algorithms?

- 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 $M_A(C)$.

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

  I.e. $Opt(C)$ is the number of mistakes made

  - for the hardest concept in C
  - using the hardest training sequence
  - by the best algorithm

Optimal Mistake Bounds (cont)

- The following relationship has been proved to hold:

$$VC(C) \leq Opt(C) \leq M_{Halving}(C) \leq log_2(|C|).$$

Weighted-Majority Algorithm

- The WEIGHTED-MAJORITY algorithm is a generalisation of the HALVING algorithm.
  - Like the HALVING algorithm it takes a vote amongst alternative hypotheses (or even alternative learning algorithms).
  - Unlike the HALVING algorithm the votes are weighted, and instead of eliminating incorrect hypotheses, their weights are adjusted downwards after each mistake.

- The WEIGHTED-MAJORITY algorithm begins by assigning a weight of 1 to each predictor and then considers the training example sequence.

  Whenever a predictor misclassifies a training example its weight is decreased by multiplying it by a factor β (so, when $\beta = 0$, WEIGHTED-MAJORITY algorithm = HALVING algorithm).

Weighted-Majority Algorithm (cont)

- Two interesting features of WEIGHTED-MAJORITY algorithm
  1. Handles inconsistent/noisy data (hypotheses never eliminated, just down-weighted)
  2. Can establish mistake bounds in terms of the best of the predictors:

     Let D be any sequence of training examples, A any set of n prediction algorithms, k the minimum number of mistakes made by any algorithm in A for D.

     Then the number of mistakes made over D by the WEIGHTED-MAJORITY algorithm is at most

$$\frac{k log_2 \frac{1}{\beta} + log_2 n}{log_2 \frac{2}{1+\beta}}$$

     where $0 \leq \beta < 1$ is the weight adjustment factor applied after each error is encountered.

Summary

- Computational learning theory examines questions such as:
  1. How many examples are needed for a learner to successfully learn the target function?
  2. How many mistakes will the learner make before succeeding?

Summary

- Computational learning theory examines questions such as:
  1. How many examples are needed for a learner to successfully learn the target function?
  2. How many mistakes will the learner make before succeeding?

- The *probably approximately correct (PAC)* learning model characterises concept classes, learners, and hypothesis spaces in terms of whether the learner can *probably* learn a hypothesis that is *approximately* correct.

Summary

- Computational learning theory examines questions such as:
  1. How many examples are needed for a learner to successfully learn the target function?
  2. How many mistakes will the learner make before succeeding?

- The *probably approximately correct (PAC)* learning model characterises concept classes, learners, and hypothesis spaces in terms of whether the learner can *probably* learn a hypothesis that is *approximately* correct.

- *Sample complexity*, the number of training examples required by a learner, can be quantitatively expressed as
  - a function of the size of the hypotheses space, for finite hypothesis spaces
  - a function of the size of the VC dimension of the hypotheses space (intuitively, the expressivity of the hypothesis space) for non-finite hypothesis spaces.

Summary (cont)

- An alternative model for assessing learning algorithms, is the *mistake bound* model which analyses the number of mistakes a learning algorithm makes before learning a target concept.

Summary (cont)

- An alternative model for assessing learning algorithms, is the *mistake bound* model which analyses the number of mistakes a learning algorithm makes before learning a target concept.

  – General results here show that for an arbitrary concept class C, the best worst-case algorithm makes $Opt(C)$ mistakes where

$$VC(C) \leq Opt(C) \leq log_2(|C|)$$

Summary (cont)

- An alternative model for assessing learning algorithms, is the *mistake bound* model which analyses the number of mistakes a learning algorithm makes before learning a target concept.

  – General results here show that for an arbitrary concept class C, the best worst-case algorithm makes $Opt(C)$ mistakes where

  $$VC(C) \leq Opt(C) \leq log_2(|C|)$$

  – The WEIGHTED-MAJORITY algorithm, which combines the votes of multiple predictors, can be shown to have a mistake bound limited by that of the best of its predictors.