Algorithm-Independent Aspects of Machine Learning

COMP9417 Machine Learning and Data Mining



May 9, 2017

Updated: May 15, 2017

Acknowledgements

Material derived from slides for the book "Elements of Statistical Learning (2nd Ed.)" by T. Hastie. R. Tibshirani & J. Friedman. Springer (2009) http://statweb.stanford.edu/~tibs/ElemStatLearn/ Material derived from slides for the book "Machine Learning: A Probabilistic Perspective" by P. Murphy MIT Press (2012) http://www.cs.ubc.ca/~murphyk/MLbook Material derived from slides for the book "Machine Learning" by P. Flach Cambridge University Press (2012) http://cs.bris.ac.uk/~flach/mlbook Material derived from slides for the book "Bayesian Reasoning and Machine Learning" by D. Barber Cambridge University Press (2012) http://www.cs.ucl.ac.uk/staff/d.barber/brml Material derived from figures for the book "Python Data Science Handbook" by J. VanderPlas O'Reilly Media (2017) http://shop.oreillv.com/product/0636920034919.do Material derived from slides for the course "Machine Learning" by A. Srinivasan

BITS Pilani, Goa, India (2016)

Contents

Aims

This lecture will introduce you to some foundational results that apply in machine learning irrespective of any particular algorithm, and will enable you to define and reproduce some of the fundamental approaches and results from the computational and statistical theory. Following it you should be able to:

 describe a basic theoretical framework for sample complexity of learning

- describe a basic theoretical framework for sample complexity of learning
- describe the Probably Approximately Correct (PAC) learning framework

- describe a basic theoretical framework for sample complexity of learning
- describe the Probably Approximately Correct (PAC) learning framework
- describe the Vapnik-Chervonenkis (VC) dimension framework

- describe a basic theoretical framework for sample complexity of learning
- describe the Probably Approximately Correct (PAC) learning framework
- describe the Vapnik-Chervonenkis (VC) dimension framework
- ullet describe the Mistake Bounds framework and apply the Winnow algorithm



- describe a basic theoretical framework for sample complexity of learning
- describe the Probably Approximately Correct (PAC) learning framework
- describe the Vapnik-Chervonenkis (VC) dimension framework
- describe the Mistake Bounds framework and apply the WINNOW algorithm
- outline the "No Free Lunch" and "Ugly Duckling" Theorems

Contents

2 Introduction

Are there general laws that apply to inductive learning?

Are there general laws that apply to inductive learning?

Are there general laws that apply to inductive learning?

From computational learning theory and statistical learning theory:

 in both cases, theoretical results have been obtained that apply in very general settings

Are there general laws that apply to inductive learning?

- in both cases, theoretical results have been obtained that apply in very general settings
- provide elegant frameworks leading to results on what can or cannot be learned algorithmically

Are there general laws that apply to inductive learning?

- in both cases, theoretical results have been obtained that apply in very general settings
- provide elegant frameworks leading to results on what can or cannot be learned algorithmically
- however, theoretical results are not always easy to obtain for practically useful machine learning methods and applications

Are there general laws that apply to inductive learning?

- in both cases, theoretical results have been obtained that apply in very general settings
- provide elegant frameworks leading to results on what can or cannot be learned algorithmically
- however, theoretical results are not always easy to obtain for practically useful machine learning methods and applications
 - need to make (sometimes unrealistic) assumptions

Are there general laws that apply to inductive learning?

- in both cases, theoretical results have been obtained that apply in very general settings
- provide elegant frameworks leading to results on what can or cannot be learned algorithmically
- however, theoretical results are not always easy to obtain for practically useful machine learning methods and applications
 - need to make (sometimes unrealistic) assumptions
 - results may be overly pessimistic, e.g., worst-case bounds

Are there general laws that apply to inductive learning?

- in both cases, theoretical results have been obtained that apply in very general settings
- provide elegant frameworks leading to results on what can or cannot be learned algorithmically
- however, theoretical results are not always easy to obtain for practically useful machine learning methods and applications
 - need to make (sometimes unrealistic) assumptions
 - results may be overly pessimistic, e.g., worst-case bounds
- nonetheless, both areas have contributed results which are important for understanding some key issues in machine learning

Are there general laws that apply to inductive learning?

- in both cases, theoretical results have been obtained that apply in very general settings
- provide elegant frameworks leading to results on what can or cannot be learned algorithmically
- however, theoretical results are not always easy to obtain for practically useful machine learning methods and applications
 - need to make (sometimes unrealistic) assumptions
 - results may be overly pessimistic, e.g., worst-case bounds
- nonetheless, both areas have contributed results which are important for understanding some key issues in machine learning
- have also led to important advances in practical algorithms, such as boosting (next lecture) and support vector machines (earlier lecture)

Contents

3 Computational Learning Theory

We seek theory to relate:

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



Characterise *classes* of algorithms using questions such as:

- Sample complexity
 - How many training examples required for learner to converge (with high probability) to a *successful* hypothesis?
- Computational complexity
 - How much computational effort required for learner to converge (with high probability) to a successful hypothesis ?
- Hypothesis complexity
 - How do we measure the complexity of a hypothesis?
 - How large is a hypothesis space?
- Mistake bounds
 - How many training examples will the learner misclassify before converging to a successful hypothesis?

What do we consider to be a *successful* hypothesis:

- identical to target concept ?
- mostly agrees with target concept . . . ?
- ... does this most of the time?

Some questions to ask, without focusing on any particular algorithm:

Sample complexity

- Sample complexity
 - How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?

- Sample complexity
 - How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
- Computational complexity

- Sample complexity
 - How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
- Computational complexity
 - How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?

- Sample complexity
 - How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
- Computational complexity
 - How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?
- Mistake bounds

Some questions to ask, without focusing on any particular algorithm:

- Sample complexity
 - How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
- Computational complexity
 - How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?
- Mistake bounds
 - How many training examples will the learner misclassify before converging to a successful hypothesis?

The framework of Probably Approximately Correct (PAC) learning can be used to answer the first two questions.

We will start by introducing the approach of Concept Learning.

Contents

4 PAC Learning

Prototypical Concept Learning Task

Given:

```
Instances X: Possible days, each described by the attributes Sky, AirTemp, Humidity, Wind, Water, Forecast
```

Target function
$$c: EnjoySport: X \rightarrow \{0,1\}$$

Hypotheses
$$H$$
: Conjunctions of literals. E.g.

$$\langle ?, Cold, High, ?, ?, ? \rangle$$

Training examples D: Positive and negative examples of the target function $\langle x_1, c(x_1) \rangle, \ldots \langle x_m, c(x_m) \rangle$

Determine:

A hypothesis h in H such that h(x) = c(x) for all x in D?

A hypothesis h in H such that h(x) = c(x) for all x in X?

Sample Complexity

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 sequence 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

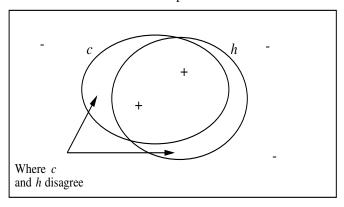
Learner must output a hypothesis h estimating c

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

Note: randomly drawn instances, noise-free classifications

e Error of a Hypothesis

Instance space X



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} .

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

Two Notions of Error

Training error of hypothesis h with respect to target concept c

• How often $h(x) \neq c(x)$ over training instances

True error of hypothesis h with respect to c

• How often $h(x) \neq c(x)$ over future random instances

Our concern:

- Can we bound the true error of h given the training error of h?
- First consider when training error of h is zero (i.e., $h \in VS_{H,D}$)

Concept Learning as Search



Question: What can be learned?

Answer: (only) what is in the hypothesis space

How big is the hypothesis space for EnjoySport?

Instance space

Concept Learning as Search

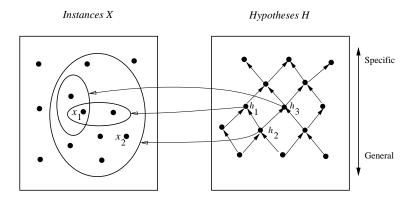
Hypothesis space

$$\begin{array}{rcl} \textit{Sky} \times \textit{AirTemp} \times \ldots \times \textit{Forecast} &=& 5 \times 4 \times 4 \times 4 \times 4 \times 4 \\ &=& 5120 \\ &\text{(semantically distinct* only)} &=& 1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3) \\ &=& 973 \end{array}$$

* any hypothesis with an \emptyset constraint covers no instances, hence all are semantically equivalent.

The learning problem \equiv searching a hypothesis space. How ?

Instances, Hypotheses, and More-General-Than



 $x_1 = \langle Sunny, Warm, High, Strong, Cool, Same \rangle$ $x_2 = \langle Sunny, Warm, High, Light, Warm, Same \rangle$

A generality order on hypotheses



Definition: Let h_j and h_k be Boolean-valued functions defined over instances X. Then h_j is more_general_than_or_equal_to h_k (written $h_j \geq_q h_k$) if and only if

$$(\forall x \in X)[(h_k(x) = 1) \to (h_j(x) = 1)]$$

Intuitively, h_j is **more_general_than_or_equal_to** h_k if any instance satisfying h_k also satisfies h_j .

 h_j is (strictly) more_general_than h_k (written $h_j >_g h_k$) if and only if $(h_j \ge_g h_k) \wedge (h_k \not\ge_g h_j)$.

 h_j is more_specific_than h_k when h_k is more_general_than h_j .

Version Spaces



A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if h(x) = c(x) for each training example $\langle x, c(x) \rangle$ in D.

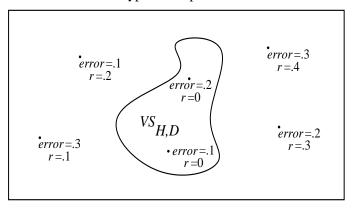
$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) \ h(x) = c(x)$$

The version space, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

Exhausting the Version Space

Hypothesis space H



Exhausting the Version Space

Note: in the diagram

$$(r = training error, error = true error)$$

Definition: The version space $VS_{H,D}$ is said to be ϵ -exhausted with respect to c and \mathcal{D} , if every hypothesis h in $VS_{H,D}$ has error less than ϵ with respect to c and \mathcal{D} .

$$(\forall h \in VS_{H,D}) \ error_{\mathcal{D}}(h) < \epsilon$$

So $VS_{H,D}$ is not ϵ -exhausted if it contains at least one h with $error_{\mathcal{D}}(h) \geq \epsilon$.

How many examples will ϵ -exhaust the VS?

Theorem: [Haussler, 1988].

If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent random examples of some target concept c, then for any $0 \leq \epsilon \leq 1$, the probability that the version space with respect to H and D is not ϵ -exhausted (with respect to e) is less than

$$|H|e^{-\epsilon m}$$

Interesting! This bounds the probability that any consistent learner will output a hypothesis h with $error(h) \geq \epsilon$



many examples will ϵ -exhaust the VS?

If we want this probability to be below δ

$$|H|e^{-\epsilon m} \le \delta$$

then

$$m \ge \frac{1}{\epsilon} (\ln|H| + \ln(1/\delta))$$

Learning Conjunctions of Boolean Literals

every h in $VS_{H,D}$ satisfies $error_{\mathcal{D}}(h) \leq \epsilon$

How many examples are sufficient to assure with probability at least $(1-\delta)$ that

Use our theorem:

$$m \ge \frac{1}{\epsilon} (\ln|H| + \ln(1/\delta))$$

Learning Conjunctions of Boolean Literals



Suppose H contains conjunctions of constraints on up to n Boolean attributes (i.e., n Boolean literals – any h can contain a literal, or its negation, or neither). Then $|H|=3^n$, and

$$m \ge \frac{1}{\epsilon} (\ln 3^n + \ln(1/\delta))$$

or

$$m \ge \frac{1}{\epsilon} (n \ln 3 + \ln(1/\delta))$$

How About EnjoySport?

$$m \ge \frac{1}{\epsilon} (\ln|H| + \ln(1/\delta))$$

If H is as given in EnjoySport then |H|=973, and

$$m \ge \frac{1}{\epsilon} (\ln 973 + \ln(1/\delta))$$

How About EnjoySport?

... if want to assure that with probability 95%, VS contains only hypotheses with $error_{\mathcal{D}}(h) \leq .1$, then it is sufficient to have m examples, where

$$m \ge \frac{1}{.1}(\ln 973 + \ln(1/.05))$$
$$m \ge 10(\ln 973 + \ln 20)$$
$$m \ge 10(6.88 + 3.00)$$
$$m \ge 98.8$$



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 $error_{\mathcal D}(h) \le \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n and size(c).

Probably Approximately Correct Learning (Valiant, 1984).

nostic Learning

So far, assumed $c \in H$ — consistent learners

Agnostic learning setting: don't assume $c \in H$

- What do we want then?
 - ullet The hypothesis h that makes fewest errors on training data
- What is sample complexity in this case?

$$m \ge \frac{1}{2\epsilon^2} (\ln|H| + \ln(1/\delta))$$

derived from Hoeffding bounds:

$$Pr[error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h) + \epsilon] \le e^{-2m\epsilon^2}$$

piased Learners

Unbiased concept class ${\cal C}$ contains all target concepts definable on instance space ${\cal X}.$

$$|C| = 2^{|X|}$$

Say X is defined using n Boolean features, then $|X| = 2^n$.

$$|C| = 2^{2^n}$$

An unbiased learner has a hypothesis space able to represent *all* possible target concepts, i.e., H=C.

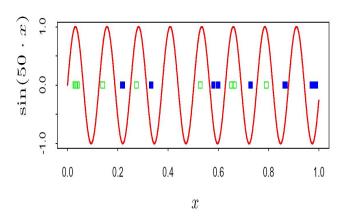
$$m \ge \frac{1}{\epsilon} (2^n \ln 2 + \ln(1/\delta))$$

i.e., exponential (in the number of features) sample complexity

Contents

5 VC Dimension

How *Complex* is a Hypothesis?



How Complex is a Hypothesis?

The solid curve is the function $\sin(50x)$ for $x \in [0, 1]$.

The blue (solid) and green (hollow) points illustrate how the associated indicator function $I(\sin(\alpha x) > 0)$ can shatter (separate) an arbitrarily large number of points by choosing an appropriately high frequency α .

Classes separated based on $\sin(\alpha x)$, for frequency α , a *single* parameter.

Shattering a Set of Instances

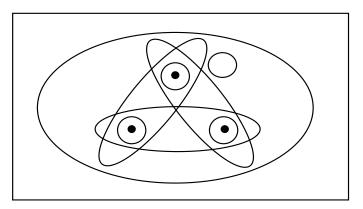


Definition: a **dichotomy** of a set S is a partition of S into two disjoint subsets.

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.

Three Instances Shattered

Instance space X



Chattering a set of instances

Consider the following instances:

```
m = \mathsf{ManyTeeth} \land \neg \mathsf{Gills} \land \neg \mathsf{Short} \land \neg \mathsf{Beak}
```

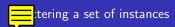
$$g = \neg \mathsf{ManyTeeth} \ \land \ \mathsf{Gills} \ \land \ \neg \mathsf{Short} \ \land \ \neg \mathsf{Beak}$$

$$s = \neg \mathsf{ManyTeeth} \ \land \ \neg \mathsf{Gills} \ \land \ \mathsf{Short} \ \land \ \neg \mathsf{Beak}$$

$$b = \neg \mathsf{ManyTeeth} \land \neg \mathsf{Gills} \land \neg \mathsf{Short} \land \mathsf{Beak}$$

There are 16 different subsets of the set $\{m,g,s,b\}$. Can each of them be represented by its own conjunctive concept? The answer is yes: for every instance we want to exclude, we add the corresponding negated literal to the conjunction. Thus, $\{m,s\}$ is represented by $\neg \mathsf{Gills} \land \neg \mathsf{Beak}, \{g,s,b\}$ is represented by $\neg \mathsf{ManyTeeth}, \{s\}$ is represented by

 \neg ManyTeeth $\land \neg$ Gills $\land \neg$ Beak, and so on. We say that this set of four instances is *shattered* by the hypothesis language of conjunctive concepts.



Suppose we have a dataset described by d Boolean features, and a hypothesis space of conjunctions of up to d Boolean literals. Then the largest subset of instances that can be shattered is at least d.

To see this argument more clearly, suppose d=3 and that the i-th instance is represented by having only the i-th Boolean literal set to true, simplified here to the corresponding bitstring:

instance₁: 100 instance₂: 010 instance₃: 001

Now any dichotomy can be constructed by a conjunctive hypothesis that excludes any instance simply by adding the appropriate literal. For example, the hypothesis including only instance₂ is $\neg l_1 \land \neg l_3$. In fact, it can be shown that the largest subset of instances that can be shattered in this setting has size exactly d.

The Vapnik-Chervonenkis 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 $VC(H) \equiv \infty$.

Note: the VC dimension can be defined for an infinite hypothesis space H since it depends only on the size of finite subsets of X.

VC Dimension of Conjunctive Concepts

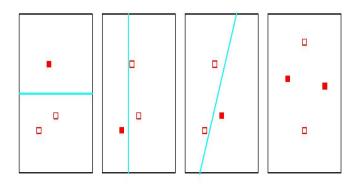
If we have an instance space X where each instance is described by d Boolean features, and a hypothesis space H of conjunctions of up to d Boolean literals, then the VC Dimension VC(H)=d.

What about some other type of hypothesis space ?

Let's look at linear classifiers.

VC Dimension of Linear Decision Surfaces





From the left, shown are three dichotomies of the same three instances. Can a linear classifier be found for the other five dichotomies? On the right, this set of four instances clearly cannot be shattered by a hypothesis space of linear classifiers.

VC Dimension of Linear Decision Surfaces

Consider linear classifiers in two dimensions. What is the VC dimension of this class of hypotheses H?

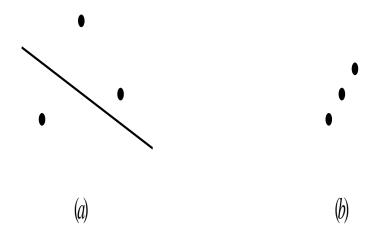
Clearly, for a subset of 2 instances we can find a linear classifier for all possible dichotomies. What about a subset of 3 instances?

The same argument as for 2 instances applies (see first three examples on previous slide, and case (a) on next slide), as long as the instances are not collinear (case (b) on next slide). So the VC dimension is at least 3.

However, in this setting, there is no set of 4 points that can be shattered.

In general, for linear classifiers in d dimensions the VC dimension is d+1.

VC Dimension of Linear Decision Surfaces



pple Complexity from VC Dimension

We can now generalise the PAC-learning result obtained earlier to answer the question: how many randomly drawn examples suffice to ϵ -exhaust $VS_{H,D}$ with probability at least $(1-\delta)$?

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

So we see that the concept of the VC dimension of a hypothesis class gives us a general framework for characterising the complexity or *capacity* of hypotheses in terms of their ability to express all possible target concepts in a particular learning setting.

For example, the argument developed for the VC dimension of linear classifiers can be extended to multi-layer perceptrons to obtain sample complexity bounds in the PAC-learning framework.

Contents

6 Mistake Bounds



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?

Again, we start by analysing concept learning.

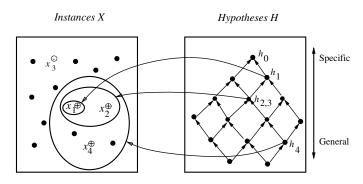
The FIND-S Algorithm



An online, specific-to-general, concept learning algorithm:

- Initialize h to the most specific hypothesis in H
- For each positive training instance x
 - For each attribute constraint a_i in h
 - If the constraint a_i in h is satisfied by x
 - Then do nothing
 - ullet Else replace a_i in h by the next more general constraint satisfied by x

Hypothesis Space Search by FIND-S



$$\begin{split} x_1 &= \langle Sunny\ Warm\ Normal\ Strong\ Warm\ Same \rangle, + \\ x_2 &= \langle Sunny\ Warm\ High\ Strong\ Warm\ Change \rangle, + \\ x_3 &= \langle Rainy\ Cold\ High\ Strong\ Warm\ Change \rangle, + \\ x_4 &= \langle Sunny\ Warm\ High\ Strong\ Cool\ Change \rangle, + \end{split}$$

$$\begin{split} &h_0 = <\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing > \\ &h_1 = <Sunny \ Warm \ Normal \ Strong \ Warm \ Same > \\ &h_2 = <Sunny \ Warm \ ? \ Strong \ Warm \ Same > \\ &h_3 = <Sunny \ Warm \ ? \ Strong \ Warm \ Same > \\ &h_4 = <Sunny \ Warm \ ? \ Strong \ ? \ ? > \end{split}$$

FIND-S - does it work?

Assume: a hypothesis $h_c \in H$ describes target function c, and training data is error-free.

By definition, h_c is consistent with all positive training examples so can never cover a negative example.

For each h generated by FIND-S h_c is $more_general_than_or_equal_to$ h.

So h can never cover a negative example.

Complaints about FIND-S

- Can't tell whether it has learned concept
 learned hypothesis may not be the only consistent hypothesis
- Can't tell when training data inconsistent cannot handle noisy data
- Picks a maximally specific h (why?)
 might require maximally general h



Mistake bounds are based on the work of Nick Littlestone¹ who developed the WINNOW family of algorithms.

- online, mistake-driven algorithm
 - similar to perceptron
 - · learns linear threshold function
- designed to learn in the presence of many irrelevant features
 - number of mistakes grows only logarithmically with number of irrelevant features
- Next slide shows the algorithm known as WINNOW2
 - in WINNOW1 attribute elimination not demotion

COMP9417 ML & DM Learning Theory / 73

¹N. Littlestone. "Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm". Machine Learning 2(4): 285-318 (1987)

Winnow2

```
While some instances are misclassified
  For each instance x
    classify x using current weights w
    If predicted class is incorrect
       If x has class 1
         For each x_i = 1, w_i \leftarrow \alpha w_i
                                                  # Promotion
          (if x_i = 0, leave w_i unchanged)
       Otherwise
         For each x_i = 1, w_i \leftarrow \frac{w_i}{\alpha}
                                                   # Demotion
          (if x_i = 0, leave w_i unchanged)
```

Here x and w are vectors of features and weights, respectively.

Winnow2

- user-supplied threshold heta
 - class is 1 if $\sum w_i a_i > \theta$
- note similarity to perceptron training rule
 - WINNOW2 uses multiplicative weight updates
 - $\alpha > 1$
- will do *much better* than perceptron with many irrelevant attributes
 - an attribute-efficient learner
- what are irrelevant attributes?
 - ullet assume that the target concept can be expressed using a *finite* subset r of attributes or features
 - if there are n attributes in total, n-r are irrelevant
 - mistake-bounds for WINNOW-type algorithms are logarithmic in the number of irrelevant attributes
 - typically, the worst-case mistake-bound is something like $\mathcal{O}(r \log n)$

Consider FIND-S when H = conjunction of Boolean literals FIND-S:

- Initialize h to the most specific hypothesis $l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \dots l_n \wedge \neg l_n$
- ullet 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?

FIND-S will converge to error-free hypothesis in the limit if

- $C \subseteq H$
- data are noise-free

How many mistakes before converging to correct h?

- FIND-S initially classifies all instances as negative
- will generalize for positive examples by dropping unsatisfied literals
- since $c \in H$, will never classify a negative instance as positive (if it did:
 - would exist a hypothesis h' such that $c \ge_q h'$
 - would exist an instance x such that h'(x) = 1 and $c(x) \neq 1$
 - which contradicts definition of generality ordering \geq_g)
- mistake bound from number of positives classified as negatives

How many mistakes before converging to correct h?

- 2n terms in initial hypothesis
- first mistake, remove half of these terms, leaving n
- each further mistake, remove at least 1 term
- ullet in worst case, will have to remove all n remaining terms
 - would be most general concept everything is positive
- worst case number of mistakes would be n+1
- worst case sequence of learning steps, removing only one literal per step

vivistake Bounds: HALVING ALGORITHM

 $\operatorname{FIND-S}$ returns the single most-specific consistent hypothesis.

An extension of the FIND-S concept learning algorithm is the CANDIDATE-ELIMINATION algorithm which returns *all* consistent hypotheses, i.e., it finds the Version Space.

Now consider the Halving Algorithm:

- Learns concept using CANDIDATE-ELIMINATION algorithm
- Classifies new instances by majority vote of Version Space hypotheses

How many mistakes will the Halving Algorithm make before converging to correct h?

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



Mistake Bounds: HALVING ALGORITHM

HALVING ALGORITHM learns exactly when the version space contains only one consistent hypothesis which corresponds to the target concept

- at each step, remove all hypotheses whose vote was incorrect
- mistake when majority vote classification is incorrect
- mistake bound in converging to correct h ?

Mistake Bounds: HALVING ALGORITHM

- how many mistakes worst case ?
 - on every step, mistake because majority vote is incorrect
 - · each mistake, number of hypotheses reduced by at least half
 - hypothesis space size |H|, worst-case mistake bound $\lfloor \log_2 |H| \rfloor$
- how many mistakes best case ?
 - on every step, no mistake because majority vote is correct
 - · still remove all incorrect hypotheses, up to half
 - best case, no mistakes in converging to correct h

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 $M_A(C)$.

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

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

$_{ m W_E}$ ighted Majority

Based on²:

- a generalisation of HALVING ALGORITHM
- predicts by weighted vote of set of prediction algorithms
- learns by altering weights for prediction algorithms
- any prediction algorithm simply predicts value of target concept given an instance; they can be
 - elements of hypothesis space H, or even
 - different learning algorithms
- if there is inconsistency between prediction algorithm and training example
 - then reduce weight of prediction algorithm
- bound number of mistakes of ensemble by number of mistakes made by best prediction algorithm

²N. Littlestone, M. Warmuth. "The Weighted Majority Algorithm". Inf. Comput. 108(2): 212-261 (1994)

WEIGHTED MAJORITY

```
a_i is the i-th prediction algorithm
w_i is the weight associated with a_i
For all i, initialize w_i \leftarrow 1
For each training example \langle x, c(x) \rangle
  Initialize q_0 and q_1 to 0
  For each prediction algorithm a_i
     If a_i(x) = 0 then q_0 \leftarrow q_0 + w_i
     If a_i(x) = 1 then q_1 \leftarrow q_1 + w_i
  If q_1 > q_0 then predict c(x) = 1
  If q_0 > q_1 then predict c(x) = 0
  If q_0 = q_1 then predict 0 or 1 at random for c(x)
  For each prediction algorithm a_i
     If a_i(x) \neq c(x) then w_i \leftarrow \beta w_i
```

WEIGHTED MAJORITY

 ${
m Weighted}\ {
m Majorithm}\ {
m begins}\ {
m by}\ {
m assigning}\ {
m weight}\ {
m of}\ 1$ to each prediction algorithm.

On misclassification by a prediction algorithm its weight is reduced by multiplication by a constant β , $0 \le \beta \le 1$.

Equivalent to Halving Algorithm when $\beta=0$. Otherwise, this just down-weights contribution of algorithms which make errors.

Key result: number of mistakes made by WEIGHTED MAJORITY algorithm will never be greater than a constant factor times the number of mistakes made by the best member of the pool, plus a term that grows only logarithmically in the number of prediction algorithms in the pool.

Contents

7 Limits on Machine Learning

1 Are there reasons to prefer one learning algorithm over another?

- ① Are there reasons to prefer one learning algorithm over another?
- 2 Can we expect any method to be superior overall?

- 1 Are there reasons to prefer one learning algorithm over another ?
- 2 Can we expect any method to be superior overall?
- 3 Can we even find an algorithm that is overall superior to random guessing?

• Perhaps surprisingly, the answer to each of these questions is no!

- Perhaps surprisingly, the answer to each of these questions is no!
- Unless one has some prior knowledge on, or can make some assumptions about, the distribution of target functions.

- Perhaps surprisingly, the answer to each of these questions is no!
- Unless one has some prior knowledge on, or can make some assumptions about, the distribution of target functions.
- This is a consequence of a number of results collectively known as the "No Free Lunch" Theorem

- Perhaps surprisingly, the answer to each of these questions is no!
- Unless one has some prior knowledge on, or can make some assumptions about, the distribution of target functions.
- This is a consequence of a number of results collectively known as the "No Free Lunch" Theorem
- Since these results specifically address the issue of generalising from a training-set to minimize off training-set error they are also referred to as "Conservation Laws of Generalization".

Two main results are:

Two main results are:

• Uniformly averaged over all target functions, the expected off-training-set error for all learning algorithms is the same.

Two main results are:

- Uniformly averaged over all target functions, the expected off-training-set error for all learning algorithms is the same.
- Assuming that the training set \mathcal{D} can be learned correctly by all algorithms, averaged over all target functions no learning algorithm gives an off-training set error superior to any other:

$$\Sigma_F[\mathbb{E}_1(E|F,\mathcal{D}) - \mathbb{E}_2(E|F,\mathcal{D})] = 0$$

where F is the set of possible target functions, E is the off-training set error, and \mathbb{E}_1 , \mathbb{E}_2 are expectations for two learning algorithms.



• There are also related results for cases where the distribution on target functions is not uniform.

- There are also related results for cases where the distribution on target functions is not uniform.
- Therefore, all statements of the form "learning algorithm 1 is better than algorithm 2" are ultimately statements about the relevant target functions.

 \bullet Consider a data set with three Boolean features, labelled by a target function F

- ullet Consider a data set with three Boolean features, labelled by a target function F
- Suppose we have two different deterministic algorithms that generate two different hypotheses, both of which fit the training data $\mathcal D$ exactly

- \bullet Consider a data set with three Boolean features, labelled by a target function F
- Suppose we have two different deterministic algorithms that generate two different hypotheses, both of which fit the training data $\mathcal D$ exactly
- ullet For this particular target function F one algorithm is clearly superior in terms of off training-set error

- ullet Consider a data set with three Boolean features, labelled by a target function F
- Suppose we have two different deterministic algorithms that generate two different hypotheses, both of which fit the training data $\mathcal D$ exactly
- ullet For this particular target function F one algorithm is clearly superior in terms of off training-set error
- But this cannot be determined from the performance on training data ${\cal D}$ alone !



	x	F	h_1	h_2
	000	1	1	1
\mathcal{D}	001	-1	-1	-1
	010	1	1	1
	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

$$\mathbb{E}_1(E|F,\mathcal{D}) = 0.4$$

$$\mathbb{E}_2(E|F,\mathcal{D}) = 0.6$$

BUT

• If we have *no prior knowledge* about which F we are trying to learn, neither algorithm is superior to the other

- If we have *no prior knowledge* about which F we are trying to learn, neither algorithm is superior to the other
- Both fit the training data correctly, but there are 2^5 target functions consistent with ${\cal D}$

- If we have no prior knowledge about which F we are trying to learn, neither algorithm is superior to the other
- Both fit the training data correctly, but there are 2^5 target functions consistent with ${\cal D}$
- For each there is exactly one other function whose output is inverted with respect to each of the off-training set patterns

- If we have no prior knowledge about which F we are trying to learn, neither algorithm is superior to the other
- Both fit the training data correctly, but there are 2^5 target functions consistent with ${\cal D}$
- For each there is exactly one other function whose output is inverted with respect to each of the off-training set patterns
- So the performance of algorithms 1 and 2 will be inverted

- If we have no prior knowledge about which F we are trying to learn, neither algorithm is superior to the other
- Both fit the training data correctly, but there are 2^5 target functions consistent with ${\cal D}$
- For each there is exactly one other function whose output is inverted with respect to each of the off-training set patterns
- So the performance of algorithms 1 and 2 will be inverted
- Thus ensuring expected (average) error difference of zero

For every possible learning algorithm for binary classification the sum of performance over all possible target functions is exactly zero!

- on some problems we get positive performance
- so there *must* be other problems for which we get an *equal and* opposite amount of negative performance

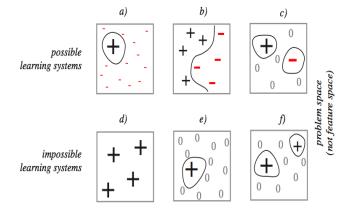


FIGURE 9.1. The No Free Lunch Theorem shows the generalization performance on the off-training set data that can be achieved (top row) and also shows the performance that *cannot* be achieved (bottom row). Each square represents all possible classification problems consistent with the training data—this is not the familiar feature space. A + indicates that the classification algorithm has generalization higher than average, a indicates lower than average, and a 0 indicates average performance. The size of a symbol indicates the amount by which the performance differs from the average. For instance, part a shows that it is possible for an algorithm to have high accuracy on a small set of problems so long as it has mildly poor performance on all other problems. Likewise, part b shows that it is possible to have excellent performance throughout a large range of problem, but this will be balanced by very poor performance on a large range of other problems. It is impossible, however, to have good performance throughout the full range of problems, shown in part d. It is also impossible to have higher-than-average performance on some problems while having average performance everywhere else, shown in part e. From: Richard O. Duda, Peter E. Hart, and David G. Stork, Pattern Classification. Copyright © 2001 by John Wiley & Sons, Inc.

Takeaways for machine learning practitioners:

Takeaways for machine learning practitioners:

• Even popular, theoretically well-founded algorithms will perform poorly on some domains, i.e., those where the learning algorithm is not a "good match" to the class of target functions.

Takeaways for machine learning practitioners:

- Even popular, theoretically well-founded algorithms will perform poorly on some domains, i.e., those where the learning algorithm is not a "good match" to the class of target functions.
- It is the *assumptions* about the learning domains, i.e., hidden target functions, that are relevant.

Takeaways for machine learning practitioners:

- Even popular, theoretically well-founded algorithms will perform poorly on some domains, i.e., those where the learning algorithm is not a "good match" to the class of target functions.
- It is the *assumptions* about the learning domains, i.e., hidden target functions, that are relevant.
- Experience with a broad range of techniques is the best insurance for solving arbitrary new classification problems.

See: Duda, Hart & Stork (2001)

Ugly Duckling Theorem



In the absence of assumptions there is no privileged or "best" feature representation.

For n objects, the set of concepts is 2^n . The number of concepts of which any pair of objects is a member (has the same concept definition, or pattern) is 2^{n-1}

using a finite number of predicates to distinguish any two patterns
denoting sets of objects, the number of predicates shared by any two
such patterns is constant and independent of those patterns.

Therefore, even the notion of similarity between patterns depends on assumptions, i.e., *inductive bias*.

Contents

8 Summary

• Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results over-conservative from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting
- VC dimension

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting
- VC dimension
 - SVM

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting
- VC dimension
 - SVM
- Mistake Bounds

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting
- VC dimension
 - SVM
- Mistake Bounds
 - Winnow

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results over-conservative from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting
- VC dimension
 - SVM
- Mistake Bounds
 - Winnow
- Bias-variance decomposition (next lecture)

- Algorithm independent analyses using techniques from computational complexity, pattern recognition, statistics, etc.
- Some results *over-conservative* from practical viewpoint, e.g., worst-case rather than average-case analyses
- Nonetheless, has led to many practically useful algorithms, e.g.,
- PAC
 - Boosting
- VC dimension
 - SVM
- Mistake Bounds
 - Winnow
- Bias-variance decomposition (next lecture)
 - Ensemble learning methods