Learning Theory

COMP9417 Machine Learning and Data Mining

Term 2, 2021

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.oreilly.com/product/0636920034919.do`

Material derived from slides for the course
“Machine Learning” by A. Srinivasan
BITS Pilani, Goa, India (2016)

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:

- outline the "No Free Lunch" Theorem
- describe a basic theoretical framework for sample complexity of learning
- describe the Probably Approximately Correct (PAC) learning framework
- describe the Vapnik-Chervonenkis (VC) dimension
- describe the Mistake Bounds framework
- outline the "Winnow" and "Weighted Majority" Algorithms

Some questions about Machine Learning

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 ?

Some questions about Machine Learning

- 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".

No Free Lunch Theorem

One of the main results:

- 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.

No Free Lunch example

- 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
- The first algorithm assumes all instances x are in the target function F, unless labelled otherwise in training set $\mathcal{D}$, and the second algorithm assumes the opposite
- For *this particular target function* F the first algorithm is clearly superior in terms of off-training-set error
- But this cannot be determined from the performance on training data $\mathcal{D}$ alone !

No Free Lunch example

	x	F	h_1	h_2
$\mathcal{D}$	000	1	1	1
	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

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

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

No Free Lunch example

- But note that the algorithm designer does not *know* F here
- Furthermore, 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 a total of 2^5 target functions consistent with $\mathcal{D}$
- For each of these target functions there is exactly one other function whose output is inverted with respect to each of the off-training set instances
- So the performance of algorithms 1 and 2 will be inverted, and their off-training-set error differences cancel
- Thus ensuring expected (average) error difference of zero

A Conservation Theorem of Generalization Performance

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

A Conservation Theorem of Generalization Performance

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 et al. (2001).

Can we have a Theory of Machine Learning ?

Are there general laws that apply to inductive learning?

Key Idea Leaning theory aims at a body of theory that captures all important aspects of the fundamentals of the learning process and any algorithm or class of algorithms designed to do learning — i.e., we desire theory to capture the *algorithm-independent aspects of machine learning*.

BUT: we're not there yet . . .

Learning Theory

From **computational learning theory** and **statistical learning theory**:

- 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 and support vector machines

Computational Learning Theory

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

- 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 ?

Computational Learning Theory

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

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

Computational Learning Theory

Instead of focusing on particular algorithms, learning theory aims to characterise *classes* of algorithms.

The framework of Probably Approximately Correct (PAC) learning can be used to address questions such as:

- How many training examples ?
- How much computational effort required ?
- How complex a hypothesis class needed ?
- How to quantify hypothesis complexity ?
- How many mistakes will be made ?

We start to look at PAC learning using Concept Learning.

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

Version Space

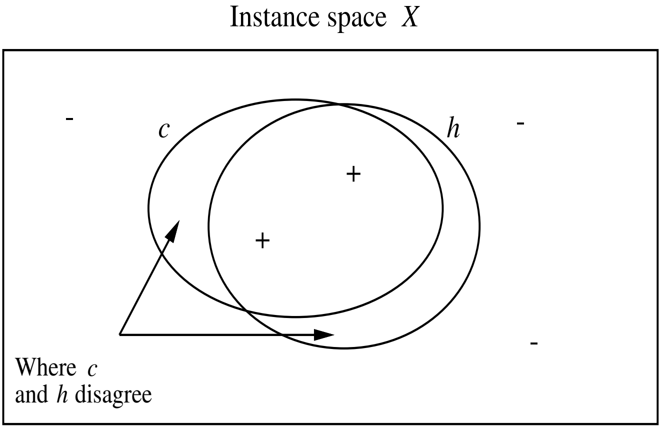
A hypothesis h is **consistent** *with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D.*

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

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.*

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

True Error of a Hypothesis



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

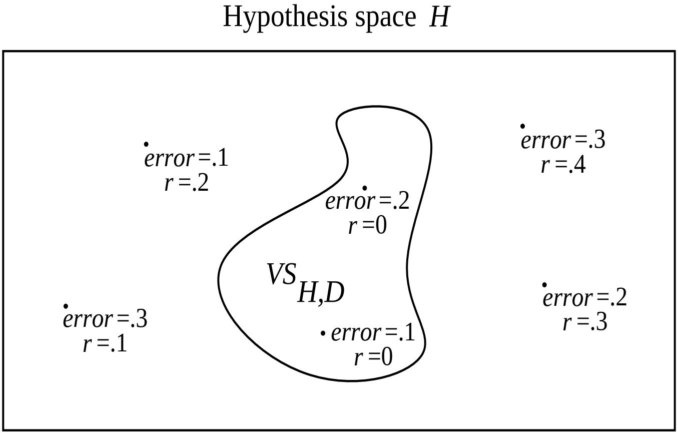
True error of hypothesis h with respect to c

- How often $h(x) \neq c(x)$ over future randomly-drawn 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., $h \in VS_{H,D}$)

Exhausting the Version Space



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 c) is less than*
>
> $$|H|e^{-\epsilon m}$$

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

How many examples will ϵ-exhaust the VS?

If we want this probability to be below δ

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

then

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

How many examples will ϵ-exhaust the VS?

How many examples are sufficient to assure with probability at least $(1-\delta)$ that every h in $VS_{H,D}$ satisfies $error_{\mathcal{D}}(h) \leq \epsilon$?

Use our theorem:

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

Example: Learning Conjunctions of Boolean Literals

Suppose H contains conjunctions of up to n Boolean literals (i.e., constraints on Boolean features, such as $x_3 = true$) — any $h \in H$ can contain a literal, or its negation, or neither (i.e., a feature variable or *don't care*).

Then $|H| = 3^n$, and

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

or

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

PAC Learning[1]

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* $size(c)$.

[1] See: Valiant (1984).

A PAC Analysis Shows The Need For Inductive Bias

Unbiased concept class C contains all target concepts definable on instance space X.

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

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

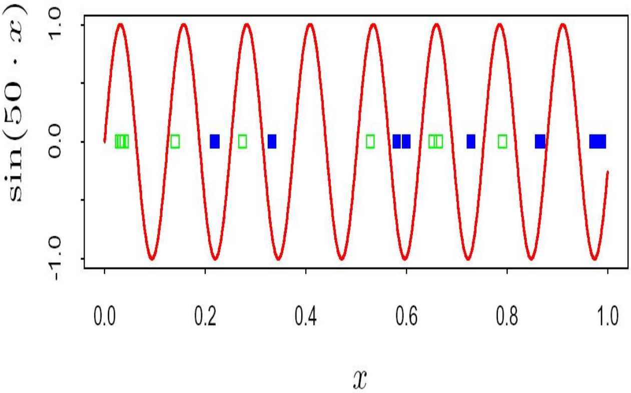
$$|C| = 2^{2^n}$$

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

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

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

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.

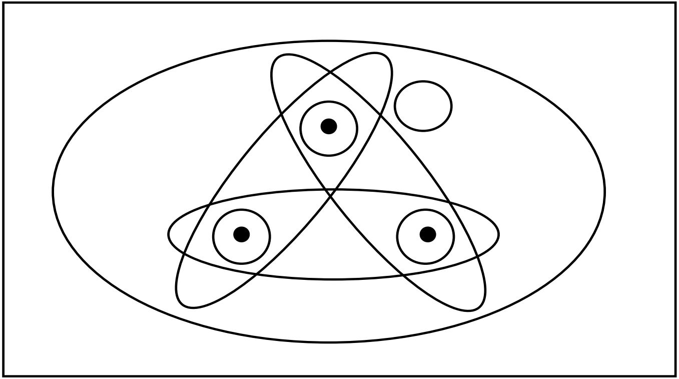
Shattering a Set of Instances

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



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.

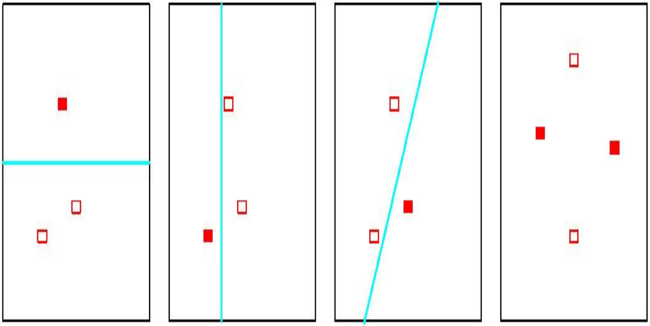
VC Dimension of Conjunctive Concepts

From the earlier slide on shattering a set of instances by a conjunctive hypothesis, 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?

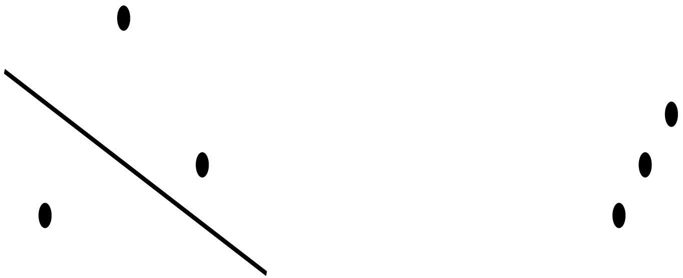
Clearly, for a subset of 2 instances we can find a linear classifier for all possible dichotomies. What about a subset of 3 instances ?

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



(a)

(b)

Sample 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 $(1-\delta)$?

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

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.

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?

Winnow

Mistake bounds are based on the work of Nick Littlestone[2] who developed the Winnow family of algorithms.

- online, mistake-driven algorithm
    - similar to perceptron, but converges faster[3]
    - 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

[2] See: Littlestone (1988)

[3] See: Haussler (1990).

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 θ
  - 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 ?
  - 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 $\mathcal{O}(r \log n)$

The WEIGHTED MAJORITY Algorithm

- learn to predict by *weighted vote* of set of prediction algorithms[4]
- learns by updating weights for prediction algorithms
- any *prediction algorithm* simply predicts value of target concept given an instance; a prediction algorithm 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

[4] See: Littlestone and Warmuth (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

Weighted Majority algorithm begins by assigning weight of 1 to each prediction algorithm.

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

Works by down-weighting contribution of prediction 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.

Summary

- Learning theory seeks fundamental analyses of machine learning
    - combines 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, Weighted Majority

Duda, R., Hart., P., and Stork, D. (2001). *Pattern Classfication*. Wiley.

Haussler, D. (1990). Probably Approximately Correct Learning. In *Proceedings of AAAI*, pages 1101–1108.

Littlestone, N. (1988). Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, 2:285–318.

Littlestone, N. and Warmuth, M. (1994). The Weighted Majority Algorithm. *Information and Computation*, 108(2):212–261.

Valiant, L. (1984). A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142.