Computational Learning Theory

Tim Kovacs
Department of Computer Science
University of Bristol
Kovacs@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~kovacs

The following is an approximate timetable for the course:

Week	Subject (Chapter)
1	Concepts, Hypotheses, and Learning Algorithms (Ch. 1)
2	Boolean Formulae and Representations (Ch. 2)
3	Inductive Bias
4	Probabilistic Learning (Ch. 3)
5	Consistent Algorithms and Learnability (Ch. 4)
6	Efficient Learning I (Ch. 5)
7	Efficient Learning II – A Hardness Result (Ch. 5)
8	Efficient Learning III – EPAC Learning (Ch. 6)
9	The VC Dimension (Ch. 7)
10	The Mistake-Bound Model for On-line Learning

The course text is:

- M. Anthony and N. Biggs. Computational Learning Theory. Cambridge University Press, 1997. ISBN 0-521-59922-9.

Chapters 2 and 7 of this text will also be useful:

- Tom M. Mitchell. Machine Learning. McGraw-Hill, 1997.

I Concepts, Hypotheses, and Learning Algorithms.

1 Introduction

Computational Learning Theory is concerned with mathematical models of learning. These models are analysed to develop our understanding of their properties, that is, to develop a body of theoretical results. Theory can tell us how learning tasks and learning algorithms interact.

- Are some kinds of problems inherently difficult?
- How much training will an algorithm need to solve a particular problem?
- How many mistakes will an algorithm make before solving a problem?

See [Mitchell 1997] section 7.1 for further discussion.

The Limits of Theory I

"It's a dangerous thing to try to formalize an enterprise as complex and varied as machine learning so that it can be subjected to rigorous mathematical analysis. To be tractable, a formal model must be simple. Thus, inevitably, most people will feel that important aspects of the activity have been left out of the theory. Of course, they will be right. Therefore, it is not advisable to present a theory of machine learning as having reduced the entire field to its bare essentials. All that can be hoped for is that some aspects of the phenomenon are brought more clearly into focus using the tools of mathematical analysis, and that perhaps a few new insights are gained." [Haussler 1990].

The Limits of Theory II

"Fundamental research in machine learning is inherently empirical, because the performance of machine learning algorithms [on real-world tasks] is determined by how well their underlying assumptions match the structure of the world. Hence, no amount of mathematical analysis can determine whether a machine learning algorithm will work well. Experimental studies are required." [Dietterich 1997].

How to Make Progress in Machine Learning

"Our understanding of the fundamental questions in machine learning has been advancing rapidly (along with the quality of the resulting learning algorithms). Much of this progress can be traced to the merging of experimental research . . . and theoretical research . . . along with the advances in computer hardware that permit large scale experimentation." [Dietterich 1997].

Major Subjects of the Course

1. The Probably Approximately Correct (PAC) Model

- We identify hypothesis classes which can or cannot be learned efficiently.
- We study the VC dimension, a measure of the complexity of hypothesis classes which allows us to bound the number of training examples needed.
- We see how to reduce one problem to another to show that no efficient algorithm exists for it.

2. The Mistake-Bound Model

- We study the number of errors made during on-line learning.

Major Subjects (Continued)

3. Inductive Bias and No Free Lunch

- We see the importance of bias and the impossibility of finding one best algorithm and bias for all problems.

4. Complexity Theory

- We define what it means for an algorithm to be efficient or inefficient, and for a problem to be hard or easy.

How the Course Proceeds

We must begin by introducing some ideas and notation.

It will take several lectures to do this, so we will not see many results at first.

2 A Model of Supervised Learning

We model supervised learning as follows. A teacher has a *target concept* t in mind, and presents the learner with a series of *examples* of the concept labelled as either positive or negative. *Positive examples* are instances of the target concept. *Negative examples* are not. The learner's goal is to find a *hypothesis* h which is as similar as possible to the target concept.

3 Examples

We denote examples using an alphabet Σ which is either $\{0,1\}$ or the set of real numbers $\mathbf{R}$.

Σ^* is the set of all non-empty finite strings of elements of Σ; the set of all syntactically possible examples.

Σ^n is the set of n-tuples (n-character strings) of elements of Σ.

X is the *example space*; the set of examples which may be used by the teacher. Sometimes $X = \Sigma^*$, sometimes $X = \Sigma^n$ for a given n, and occasionally $X \subset \Sigma^n$.

Example: A Familiar Alphabet

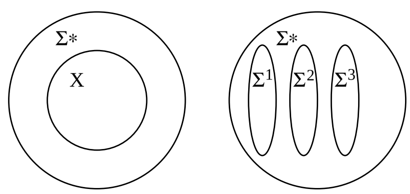
Let $\Sigma = \{$a,b, …, z$\}$. Then:

Σ^1 = the set of all 1 letter strings, e.g., a, b, c, d, e.

Σ^4 = the set of all 4 letter strings, e.g., abcd, aaaa, bbbb, loud, door.

Σ^* = the set of all finite, non-empty strings, e.g., a, aa, aaa, abcdef, house, car.

Visualising the Sets



Note:

- $X \subseteq \Sigma^*$.
- For a given n, $\Sigma^n \subset \Sigma^*$.

Example: the Binary Alphabet

Let $\Sigma = \{0,1\}$. Then:

Σ^1 = the set of all 1 letter strings, i.e., all 1 digit binary numbers: $\{0,1\}$.

Σ^4 = the set of all 4 digit binary numbers: $\{0000, 0001, 0010, 0011, \ldots 1111\}$.

Σ^* = the set of all finite, non-empty strings, e.g., 0, 00, 000, 0001, 1010001001.

4 Concepts

A *concept* over Σ is a function $c : X \to \{0,1\}$. That is, it maps each example $y \in X$ to either 0 or 1.

Example: The logical AND function.

Let $\Sigma = \{0,1\}$ and define $c : \Sigma^2 \to \{0,1\}$ as follows: if $y = y_1y_2$ then:

$$c(y) = \begin{cases} 1 & \text{if } y_1 \text{ and } y_2 \text{ are 1;} \\ 0 & \text{otherwise.} \end{cases}$$

Truth Tables

Another way to define a function is to exhaustively list each input and the corresponding output. This is called the *truth table* for the function. The truth table for the logical AND function is:

y_1	y_2	$c(y)$
0	0	0
0	1	0
1	0	0
1	1	1

Positive and Negative Examples

A *positive example* is one for which $c(y) = 1$ and a *negative example* is one for which $c(y) = 0$.

The *on-set* is the set of positive examples and the *off-set* is the set of negative examples.

y_1	y_2	$c(y)$
0	0	0
0	1	0
1	0	0

y_1	y_2	$c(y)$
1	1	1

Note:

$$\text{domain}(c) \quad = \quad \text{on-set}(c) \bigcup \text{off-set}(c) \quad \supseteq \quad X$$

5 Learning as Approximating a Function

The *concept space* C is the set of possible concepts the teacher may have.

We refer to the learner's concepts as *hypotheses*. The *hypothesis space* H is the set of possible hypotheses the learner may have.

Learning can be seen as search in H for a hypothesis which is as similar as possible to the target concept $t \in C$. Generally, the learner has insufficient information to find a hypothesis which matches t exactly, and must be content with finding a hypothesis which *probably* represents t *approximately*.

We'll learn more about Probably Approximately Correct (PAC) learning when we cover chapter 3.

6 Training

A *sample* $\mathbf{x}$ of length m is an m-tuple of examples.

$$\mathbf{x} = (x_1, x_2, \ldots, x_m)$$

A *training sample* $\mathbf{s}$ is a sequence of pairs, each of which consists of an example x_i and a bit b_i which indicates whether it is positive or negative.

$$\mathbf{s} = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m))$$

$\mathbf{s}$ is a training sample for the target concept t if:

$$b_i = t(x_i)$$

for $1 \le i \le m$, i.e., if $\mathbf{s}$ labels all its examples in the way that t does.

Example: The Odd-Parity Function

Let $\Sigma = \{0,1\}$ and define $p : \Sigma^* \to \{0,1\}$ as follows. If $y = y_1y_2 \ldots y_n$ then:

$$p(y) = \begin{cases} 1 & \text{if an odd number of } y_i\text{'s are 1;} \\ 0 & \text{otherwise.} \end{cases}$$

Here is a training sample of length $m = 3$ for the odd-parity function:

$$(\underbrace{(0010, 1)}_{(x_1,b_1)}, \underbrace{(01100, 0)}_{(x_2,b_2)}, \underbrace{(000, 0)}_{(x_3,b_3)})$$

Note that if we represent p with a truth table, each pair in the training sample is a row from that table.

7 Learning Algorithms

A (C,H)-*learning algorithm* is a function L which assigns to any training sample $\mathbf{s}$ for a target concept $t \in C$ a hypothesis $h \in H$.

A hypothesis $h \in H$ is *consistent* with $\mathbf{s}$ (or *agrees* with $\mathbf{s}$) if for each example in $\mathbf{s}$ it outputs the label (positive or negative) of that example.

A learning algorithm is consistent if, given any training sample, it would output a hypothesis consistent with that sample.

To simplify matters we normally assume we are dealing with consistent algorithms. However, this can be unrealistic. E.g., a neural net can forget.

8 Learning by Construction

When X is known, a concept is determined by its on-set, since any examples not in the on-set must be in the off-set. I.e., any examples which are not positive must be negative.

Consequently, we can learn a target concept by collecting all its positive examples:

```
1. Initialise h so it is negative for all inputs
2. For each example i
3.     If i is positive then make h positive for that input
```

(Alternatively, we can construct the off-set instead of the on-set, since it too determines a concept when X is known.)

Example: Learning the 3-bit Parity Function by Construction

Let $\Sigma = \{0,1\}$, $X = \Sigma^3$ and define $p : X \to \{0,1\}$ as before.

Let the training sample be:

$$((000,0),(001,1),(010,1),(011,0),(100,1),(101,0),(110,0),(111,1))$$

The first step of the algorithm is:

1. Initialise h so it is negative for all inputs

Let's represent the hypothesis h as a truth table. According to step 1, we initialise all the values in the $c(y)$ column to 0.

y_1	y_2	y_3	$c(y)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Learning the 3-Parity (Continued)

2. For each example i
3. If i is positive then make h positive for that input

Steps 2 and 3 instruct us to go through each pair in the training sample and set the rows of the truth table for h accordingly. This yields:

y_1	y_2	y_3	$c(y)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Notes on Learning by Construction

The learning by construction algorithm is consistent: the hypothesis it produces classifies all the examples in the training sample correctly.

It is also a *memoryless on-line* algorithm: for any given training sample, it produces a sequence of hypotheses $h_0, h_1 \ldots, h_m$, such that h_{i+1} depends only on h_i and the current labelled example (x_i, b_i).

9 Learning by Enumeration

If we can enumerate the hypothesis space $H = \{h^{(1)}, h^{(2)}, h^{(3)}, \ldots\}$, we can check each hypothesis in turn against the elements of the training sample.

If the current hypothesis $h^{(r)}$ is found to be inconsistent with the training sample we reject it and start over with the next hypothesis $h^{(r+1)}$.

1. For each hypothesis h
2. For each example i
3. If h is not consistent with i then
4. Reset i to the start of the training sample
5. Reject h and try the next hypothesis
6. Return h

Example: Learning the 2-bit Odd Parity Function by Enumeration

Hypothesis space:

y_1	y_2	$h^{(1)}$	$h^{(2)}$	$h^{(3)}$	$h^{(4)}$	$h^{(5)}$	$h^{(6)}$	$h^{(7)}$	$h^{(8)}$	$\cdots$	$h^{(16)}$
0	0	0	0	0	0	0	0	0	0	$\cdots$	1
0	1	0	0	0	0	1	1	1	1	$\cdots$	1
1	0	0	0	1	1	0	0	1	1	$\cdots$	1
1	1	0	1	0	1	0	1	0	1	$\cdots$	1

Training sample:

$$((00,0),(01,1),(10,1),(11,0))$$

Notes on Learning by Enumeration

Learning by enumeration is *not* memoryless.

There are 2^{2^n} functions for binary strings of n bits, i.e., C and H can each be as large as 2^{2^n}. To test each possible hypothesis could take a very long time.

- If we do enumerate all hypotheses, computational resources limit us to small n.
- To deal with larger n we can enumerate only a subset of H. However, this has implications; we are biasing the learner so that it considers only some hypotheses.

10 Summary of Lecture

- Computational learning theory is about mathematical models of learning.
- Supervised learning is modelled as a teacher presenting a training sample of labelled examples of a target concept to a learner who seeks a hypothesis which approximates it.
- Concepts and hypotheses are functions.
- An algorithm is consistent if, given any training sample, it would output a consistent hypothesis.
- Learning by construction involves collecting positive (or, equivalently, negative) examples to build an approximation of the target concept.
- Learning by enumeration involves checking each possible hypothesis against the training sample.

11 References

- Dietterich, T. G. (1997). Fundamental Experimental Research in Machine Learning. A section of *Basic Topics in Experimental Computer Science* edited by John McCarthy. http://www.cs.orst.edu/~tgd/
- Haussler, D. (1990). Probably Approximately Correct Learning. In the *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 1101–1108. AAAI / MIT Press.
- Mitchell, T. (1997). *Machine Learning*. McGraw–Hill.