COMP30030: Introduction to Artificial Intelligence

Neil Hurley

School of Computer Science
University College Dublin
neil.hurley@ucd.ie

October 25, 2018

Machine Learning I

- The term Machine Learning refers to "a scientific discipline that explores the construction and study of algorithms that can learn from data" (Wikipedia).
- In general, there are two broad ways that we can learn from data:
  1. Supervised methods learn by generalising from training data in the from of sets of inputs with desired outputs.
  2. Unsupervised methods learn from input data, which has no desired outputs associated with it. Instead these method search for structure in the input data.
- Another important branch of machine learning is reinforcement learning, in which a program must interact with a dynamic environment to perform a certain goal or task.

1. Problem Solving by Search
   - Uninformed Search
   - Informed Search
   - Adversarial Search
   - Game Playing with Reinforcement Learning
2. Optimisation
   - Optimisation Overview
   - Combinatorial Optimisation Problems
   - Simulated Annealing
   - Optimisation Problem Examples
   - Convergence of Simulated Annealing
   - Genetic Algorithms
   - Optimisation in Continuous Spaces
3. Machine Learning
   - Supervised Machine Learning

Supervised Machine Learning

- Suppose we are have a problem that we wish to learn from.
- Let's call the set of possible problem *instances*, $\mathcal{X}$. So an example problem is an element $x \in \mathcal{X}$.
- We can think of the solution to this problem as a map $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ is the set of solutions to problems in $\mathcal{X}$.
- So, given x, the solution to x is $f(x)$.
- We don't know this function $f(.)$ — we would like to learn an approximation to it.

Supervised Machine Learning

- A learning algorithm is supervised, if, along with a set of instances $x \in \mathcal{X}$, we have their solutions $f(x)$.
- So we have a training set $D_{\text{train}} = \{(x, f(x)) | x \in \mathcal{X}\}$.
- If we are lucky, we may have many training examples and supervised Machine Learning algorithms generally work better, the more data that is available.
- The learning algorithm is guided, so *supervised* by the solutions to the training examples that are available.
- The goal is to *generalise* from the training examples, so that solutions to new, unseen instances can be found.

Types of Supervised Learning Task

- While we can imagine many problem types, in fact there are two broad types of machine learning tasks, which apply to many real-world scenarios, which may be distinguished by the type of solution that we seek.

1. The solution space is a set of real-valued numbers $\mathcal{Y} = \mathbb{R}^n$ for some dimension n. In this case the problem is referred to as a regression problem.
2. The solution space is discrete, consisting of a finite set of $k > 0$ *labels*. $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$. In this case, the problem is referred to as a classification problem.
   - Many approaches have been developed for the simplest case of *binary classification*, where there are just two possible labels.
   - In binary classification, the goal is to determine which, of two classes, each problem instance belongs to.

Types of Supervised Learning Task

- These two problem categories may seem to simple to cover all the many learning tasks we can imagine in the real-world, but actually they are quite general.
- Note that we've (so far), said very little about the representation of the problem instances, nor the representation of their solutions.
- All of the above notation simply says that,
  - a problem is a regression problem if we can find a useful representation of its solution as a set of real-valued numbers.
  - a problem is a classification problem if wish to determine class which among a finite set of possibilities a problem instance belongs to.

Real-world Examples

Determining if an email is Spam

- x is a text – a sequence of alphabetic characters, corresponding to an email message.
- $f(x)$ is the binary label indicating, which of the two cases, Spam or Not Spam, applies to the given text.

Real-world Examples

Medical Diagnosis

- x is a set of properties of the patient – symptoms, results of lab tests, previous medical history etc. etc.
- $f(x)$ is a disease. A set of possible diseases has been preselected, e.g. $\{$`NO_DISEASE, CANCER, HEART_DISEASE`$\}$.
- The problem then is, given x, label x with the correct disease — a classification problem.

Real-world Examples

Making a Computer Read

- x is a text – a sequence of alphabetic characters.
- $f(x)$ is the sound signal corresponding to the utterance of x. $f(y)$ will be represented by an appropriate set of real-valued numbers, corresponding to a representation of the sound signal, or *features* of the sound signal.
- Typically, a database of examples is gathered from human reading of set texts.
- The goal is to train the computer to automatically read texts which are not in the training set.
- In the heart of this problem there is a regression – mapping each x to a real-valued representation of the sound. However, to successfully achieve such a learning task, we need to apply detailed knowledge of speech and linguistics.

Hypothesis Space I

- How is it possible to learn from training examples, $D_{\mathrm{train}} = \{x, f(x)\}$?
  - One way might be to try to create a (close to) exact model for $f(.)$
  - We can write down, for example, models of physical processes, such as fluid dynamics and use these to predict the weather tomorrow.
  - Take our reading example – can we write down the detailed steps by which a text becomes spoken language?
- Instead, we give up on creating such a detailed model of $f(x)$ and instead we choose a set of *candidate* functions, from which a good approximation of $f(x)$ can be selected.

Hypothesis Space II

- We call such a set a Hypothesis Space, $\mathcal{H}$
- The Hypothesis Space should be simple enough that it is tractable to select a good function from this space.
- The Hypothesis Space should be complex enough that a good candidate function exists in that space that well approximates $f(x)$.

Inductive Learning Hypothesis

- But how will we know that the function we select from $\mathcal{H}$ is really a good approximation to $f(x)$?
  - In practise, we cannot test it on all instances in the problem space.
- The Inductive Learning Hypothesis
  - Any $h(.) \in \mathcal{H}$ that approximates $f(x)$ well on the training examples, will also approximate $f(x)$ well on unseen examples.
- Whether or not this holds depends strongly on the training examples (and the learning algorithm applied to them).
- In statistical terms, the training examples need to be a representative sample of the full set of problems that occur in $\mathcal{X}$ and this may not always be the case.
- Nevertheless, supervised machine learning algorithms proceed by, as a primary objective, looking for functions in the Hypothesis space that approximate the training examples well.

Some Terminology

- $h(.) \in \mathcal{H}$ is called a consistent hypothesis if it agrees with $f(.)$ on all training examples. Given the training data, only some hypothesis in $\mathcal{H}$ are consistent. The set of consistent hypotheses is called the Version Space: $\{h(.) \in \mathcal{H} : h(x) = f(x) \forall x \in D_{\text{train}}\}$
- A consistent learner always outputs a consistent hypothesis
- The empirical error is the fraction of the training examples for which $h(x) \neq f(x)$.
- So, for a consistent learning, the empirical error is 0%.

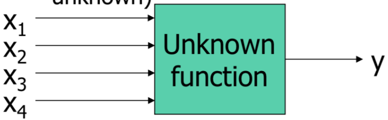
Example – An ML approach to learning a Boolean Function I

- The following example shows how a machine learning algorithm might proceed to learn a function from examples.
- We assume that there is some unknown Boolean function, over four Boolean variables:
  - $x = (x_1, x_2, x_3, x_4)$ and each variable $x_i \in \{0, 1\}$
  - $y = f(x) \in \{0, \}$
- How hard is the problem? Well, how many possible Boolean functions over 4 variables exist?
  - There are $2^4 = 16$ possible inputs to this unknown function. The function is fully defined, once it is specified what the output is for each of these 16 inputs.
  - There are 2 choices for the output $f(x)$. So there are 2 ways to set the output for each possible input. Hence, there are 2^{16} different possible functions.

Example – An ML approach to learning a Boolean Function II

- One of these functions is generating output, but we do not known which one it is – all we have available to us is a set of training instances x and their associated output $f(x)$.
- If there are very few training instances, then we will not be able to learn much.
- However, tractable learning depends not just on the instances we have, but on good selection of a Hypothesis space, that
  1. Is likely to contain a good approximation of $f(x)$
  2. But which can be searched efficiently for such an approximation.

$f(x_1,x_2,x_3,x_4) = \neg x_2 \wedge x_4$ (but it is unknown)



Training examples

x_1	x_2	x_3	x_4	y
0	0	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Hypothesis Space of *all* Boolean functions

- We do not restrict the Hypothesis space, andx consider all possible Boolean functions among our candidates
- We have $2^{16} = 64k$ possible functions.
- 5 outputs are specified in the training set, so there are $2^{16-5} = 2048$ consistent hypothesis.
- We can do no better than choose randomly between them.
- How likely is it that our unknown function agrees with the chosen one on unseen examples?

x_1	x_2	x_3	x_4	Y
0	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	?
1	0	0	0	?
1	0	0	1	?
1	0	1	0	?
1	0	1	1	?
1	1	0	0	?
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

x_1	x_2	x_3	x_4	Y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	?
0	0	1	1	1
0	1	0	0	?
0	1	0	1	?
0	1	1	0	?
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	1
1	1	0	0	0
1	1	0	1	?
1	1	1	0	0
1	1	1	1	?

Hypothesis Space : Conjunction of Literals

- A literal is a variable x_i or its negation $\neg x_i$
- A term is a *conjunction* of literals.
- We define the hypothesis space $\mathcal{H} = \{\text{terms over } x_1, x_2, x_3, x_4\}$
- Now the hypothesis space only contains $3^4 = 18$ possible functions. (Each term can be affirmed or negated or isn't present).
- Learning algorithm
  1. Initially $h(.) =$ conjunction of all possible literals
  2. Remove literals associated with inconsisten *positive* examples.

Learning Conjunctions

1. $h = \neg x_1 x_1 \neg x_2 x_2 \neg x_3 x_3 \neg x_4 x_4$
2. Observe 1st training example, remove literals x_1, x_2, $\neg x_3$, and $\neg x_4$
   $h = \neg x_1 \neg x_2 \, x_3 \, x_4$
3. Observe 2nd training example, remove literals $\neg x_1$ and x_3
   $h = \neg x_2 \, x_4$
4. Observe 3rd training example: nothing to do ($h = \neg x_2 \, x_4$)
5. No more positive training examples
   Output $h = \neg x_2 \, x_4$

x_1	x_2	x_3	x_4	y
0	0	1	1	1
1	0	0	1	1
1	0	1	1	1

Learning as Refinement

- Start with a small hypothesis class, such as Boolean conjunctions – we need domain knowledge to choose such as suitable class.
- Use examples to infer the particular function within this class.