Machine Learning in Artificial Intelligence

This assignment will explore some traditional philosophical problems surrounding induction, and use them to consider the application of inductive programming techniques in Artificial Intelligence. It will conclude that the implementation on machines have been encouraging, but that the nature of induction itself will prevent a completely succeful... Show the broad problems underpinning induction, from Humean analysis, to more modern considerations of probability and weighting.

`Let us begin with a definition. Induction can be broadly defined as a 'rational' process where, from premises about some things of a certain kind, a conclusion is drawn about some, or all, of the remaining things of that kind. Philosophically it has its history in Hume's exploration of the unreliability of causality.

`Hume (1737) derived his notion of causality from a theory that the mind consisted of nothing but perceptions, containing impressions (sensations, emotions) and ideas (thoughts). He posited that reasoning consists in the discovery of relations - 'relations of ideas' (portraying what is conceivable and what is not, such as logical truths) and relations where objects happen to be contiguous - matters of 'fact'. Matters of fact, such as the fact that John Major is PM, possess a truth or falsehood that is only established via experience. Regrettably, a matter of fact, since the opposite of it is logically conceivable (ie, non-contradictory), cannot be fully demonstrated, so humans must infer facts using probability. Inference depends upon the relation between cause and effect - the only means by which we infer the existence of an object which we have not observed, from the existence of one which we have. Experience, in this sense, teaches us little - it is a priori possible that anything could cause anything.

`From experience we rely upon the 'uniformity of nature' - that causal relations in the future will remain the same as those previously encountered. Hume gave three reasons for this conviction - that a cause will have a certain effect is founded upon:

`(a) The cause immediately preceding the effect

`(b) The cause being contiguous to its effect in space.

`(c) That the effect necessarily follows the cause.

`This last condition is an inference, not a logical connection, and is based upon a sufficient number of 'observed regularities'. The lack of a logical basis, and the unreliability of experience in general, raises important questions about our reliance on this seemingly unimpeachable tenet.

`Induction is a more logical recapitulation of this problem, associated with such figures as Russel, Whitehead and Carnap, who try to examine the problem in the harsh gaze of science.

`One must also bear in mind that there are different types of induction. A complete induction represents a general proposition concerning a class as a whole, to be concluded on the basis of examining all its elements. This gives a true conclusion, but it necessarily limited since it is only applicable to classes where all the members of a class can be easily observed.

`Using simple enumeration in induction is the orthodox manner - here, the presence of a feature in some of the elements of a class, justifies the conclusion that all elements of that particular class possess that feature. This is unlimited in scope, but the conclusions are reliant upon probable propositions which require subsequent proof.

`Scientific induction similarly represents a conclusion concerning a whole class based on a number of the elements of that class, but here the grounds for conclusion are provided by the discovery of essential connections between the elements studied, which demonstrate that the given feature must be possessed by the whole class. Consequently the methods of disclosing the essential connections are vital here.

`An argument, then, is inductive if it claims to draw a conclusion from given premises directly, in a single step. One such argument might be that, because some (or all) x's are y's, therefore further (or all) x's are y's - simple, enumerative deduction.

One might, however, add the following clause, where n has the value of 100:

`"n % of observed x's are y's, so approximately n % of all x's are y's"

Philosophers such as J.S.Mill (1843) argue that in fact, such inference makes the progression, not from particular to general, but from particular to particular - to yet further instances. Or one might retort that a conclusion about particulars can only be derived from the general. Here, one would move from:

`"All (or n % of) observed swans are white" to inferring that:

`"The next swan I observe will be white." by introducing the sub-clause:

`"All (or n % of all) swans are white."

It can be seen that, as the value of n decreases, so does the efficacy of the assertion. If, for example, I embarked upon a holiday to Australia, and encountered four parrots of the same species and colour - birds which I had never come across before - most commentators would deem me rash to pronounce that all parrots are thus of the same colour, arguing that I lack sufficient grounds to say this. One could retort that it is dificult to envisage when the grounds are sufficient - even by observing and classifying all

instances of every parrot on the planet, and somehow finding them all to be of the same colour, does not eliminate their existing a parrot of a different hue, as yet undiscovered. This argument can in principle be extended to other planets or universes where they may exist such a parrot - so we seem to be caught in a loop of infinite uncertainty, where we can never prove our induction to be correct.

`In the 'real world' of course, induction dominates reasoning. Many instances of cogitation rely upon applying general rules to particular cases, from a doctor's diagnosis to a technician's analysis of a computer breakdown. For example, a student might be writing an essay on a word-processor and suddenly encounter a blank screen. A technician might analyse the situation using some of the following rules:

`(a) If the computer fan has ceased to turn, and the l.e.d light indicating power to the monitor is not lit, and another l.e.d light indicating power to the computer itself is similarly dark, then the computer has probably suffered a loss of power.

`(b) If the computer fan is still turning and the l.e.d light indicating power to the monitor is unlit, and the l.e.d light indicating power to the computer itself is lit, then the monitor is probably at fault.

`(c) If the computer fan is whirring at high speed and the monitor is emmitting smoke and the computer itself is extremely hot, then there is a danger the machine may blow up and all parties should vacate the premises immediately...

`Many of these rules are built up by the computer engineer via his experience of computer maintenance - they are not innate rules that he was born with, but inferred from 'observed regularities' of many types of computer system and likely causes of problems.

`Carnap (1945) tried to formalize inductive inferencing The goal was to create an inductive logic which, when given various premises and a suggested explanation, would calculate the degree of belief in probability terms, that is to be placed in this hypothesis. However, induction does not appear to be explicable in such formal terms, and inductive logic suffered. More recent excitement in the field of Artificial Intelligence and 'machine learning' has seen a re-appraisal of traditional induction, and attempts to implement it on vaious computers. Popper (1963) posited a verificationist stance, appealing to a hypo-deductive method in the process. Here, certain propositions are advanced as hypothetical, and subjected to 'verification' by inferring effects based upon available valid knowledge, and comparing these effects to the facts.

`An interesting psychology example was conducted by Bruner et al (1956), which is of computational use. The following figure (1) shows a sequence of cards, each featuring three shapes - a circle, a square and a triangle. Each shape can be coloured either black or white, and there are eight possible variations in the sequence.

`

`

`Now, imagine one is told that some of the cards contain a special property which is known as dax - a nonsense word. Only two of the cards are dax - cards 2 and 4 - and are signified by being positive, with the non-dax cards being indicated by being negative. The subject then had to infer, from these observed instances, a general law which determines the dax of any cards - including the cards which had yet to be shown. Dax, here, is analogous to the fault which the aforementioned computer technician was searching for, and this is a common inductive situation - where a number of positive and negative factors are shown in terms of a set list of observable characteristics. Working from these, we must use inductive inference to predict future occurences of this property.

`Such abstract reasoning is a favourite of Mensa examinations, and it is easy to see from these cards the law which would be required to determine the dax. Yet most valuable inductive problems involve much larger possibilities and features, which is why it would be very useful to implement inference effectively on a computer. (see, for example, expert systems, with their combination of a knowledge base and an 'inference engine'.)

`For the sake of simplicity, however, let us confine the analyis to the card experiment. Note that the cards presented in the experiment were called training examples, and the property which the subjects are supposed to provide a law for was named the target concept (cards 2 and 4, remember, were the only positive examples which possessed dax). Induction, under this conception, then, is the process of inferring a rule to predict instances of a target on the basis of positive and negative training examples.

`It is necessary to formalize things a little for the purpose of writing formulas. Let us represent dax, then, with a D. We can also represent the three shapes on the cards as the letters c, s, and t (circle, square and triangle), and display them in a truth table. Here the truth values correspond to colour - T for white, and F for black. Similarly, the target concept D is given a T for a positive instance of dax, and F for a negative instance.

`c s t D

`card 1 T T T F

`card 2 T T F T

`T F T ?

`T F F ?

`card 3 F T T F

`card 4 F T F T

`card 5 F F T F

`card 6 F F F ?

The missing entries need to be calculated using the rule derived from the already observed training examples - this a formula, Y, in propositional calculus. It can be seen now that the task is indeterminate, in that any way of filling in the missing entries will be perfectly acceptable - there are many rules for doing this, none of which appears to have priority in usefulness or 'truth'. Induction itself, then, governs the creation of rules which are in some ways little more than a guess, and we must appreciate this in trying to implement on a computer.

`Although Y could be any formula of the propositional calculus, it is often desirable to limit it for simplicity - to confine this induced rule to a concept space of the induction problem. One such limitation is a monomial space. If c1...cn are proposition letters, then a monomial over c1...cn is defined as a conjunction of the literals involving c1...cn. For example, over the proposition letters c, s, t are found the monomials c, c & s and Øc &Øs &Øt; whereas ØØc, c v s and (Øc &Øs)(r)Øt are not. Also considered to be monomials are the contradictory formula ^ and the logically true formula _.

`In figure 2 which find a graphical representation of some of these monomials over the letters, where the linking lines indicate relations of implication. It must be said once more that most 'real world' example exceed the limits of predicting simple conjunctions of conditions, although this example nicely illustrates important aspect of inductive inference.

`We can derive a simple monomial induction procedure from the above, called caution-induct since it errs on the side of caution when admitting positive examples of the target concept. Let us suppose that the correct rule for predicting D is given by y. The procedure caution-induct must maintain a current conjecture Æ regarding the correct rule for D, given the training examples it has so far processed. This is a cautious conjecture as it never classifies an example as falling under D unless it is absolutely certain that it is correct:

`Æ_y.

`From diagram 1 it can be seen that initially Æ=^, and that it starts at the bottom of the concept space in diagram 2. Card 1, remember, is assigned the following truth values:

`c=T, s=T, t=T.

`It is a negative example, so is classified correctly by the formula Æ=^, and no change is made to caution-induct. With card 2 we find a positive example. This renders Æ=^ false for all rows of the truth table, which is clearly incorrect. Therefore caution-induct

recognises that its current belief must be weakened slightly - a new rule Æ needs to be made, which classifies this second card whilst still holding true for card 1. It does this by travelling up the lattice of figure 2 until it reaches a monomial which achieves this - in this way, caution-induct will not produce a conjecture that is weaker than it needs to be. This happens to be:

`Æ = c & s & Øt.

Card 3 is again found to be a negative example from the values: c = F, s = T, t = T, so no change is made. However, card 4 has the values: c = F, s = T, t = F which makes it a positive example as in card 2. Once again, caution-induct must modify its conjecture by moving up the lattice to find a suitable monomial, finding: Æ = s & Øt.

`This becomes the new conjecture for D. Card 5 is examined and found to require no change since it bears the negative values of: c = F, s = F, t = T. Having exhausted all its training examples, caution-induce has the rule: Æ = s& Øt as its final result, ie, it has inferred from the five examples that a card has a target concept iff it features a white square and a black triangle. This can be used to make predictions about the as yet unseen cards which follow.

`What is the actual procedure of caution-induct? Envisage a list of m positive and negative training examples of a concept D, x1,...xn. Each example xi is described as a truth-value assignment to the proposition letters c1,...cn , and for each training example, we are told it it contains the propert D. The procedure returns a monomial for predicting such examples of D.

`At its core is ascend, which features as arguments, Æ, a monomial expressing the current conjecture of the rule for D, and x, a new training example known to feature D, expressed as a truth-value assignment to a collection of proposition letters. The output is a monomial which takes the truth value T according to x; which is logically weaker, or equivalent to, Æ, and which is the logically strongest monomial given these two facts. In pseudo-code it becomes:

`begin ascend (Æ, x)

`if Æ is ^ then

`let new- Æ be the conjunction of all the literals

`that are true according to x
`

else if Æ features literals which are false under x, then

`let new- Æ be Æ with these literals deleted

`else

`let new- Æ be Æ

`end if

`return new- Æ

`end ascend

`When applied to new training examples, ascend will maintain Æ _y, and changes this conjecture as it experiences new examples until it reaches: Æ = s& Øt. This is known as the inductive bias of a procedure. When applying the procedure caution-induct, the code is thus:

`begin caution-induct

`let Æ be ^

`for every x in x1,...xm do

`if x is a positive example of D, then

`let Æ be the result of ascend (Æ, x)

`end if

`end until

`end caution-induce

Such caution as this procedure displays is useful to avoid error, but hardly a natural state of affairs in most practical aspects of the world. For 'rational' human beings we place a great deal of reliance upon induction, and take calculated, probabilistic risks that the future will indeed conform to the rules which governed the past.

`A more complex monomial induction procedure is that which is not limited to working, as caution-induct, from the lowest level of the concept space; but which searches simlutaneously from the top and bottom, meeting in the middle. Where caution-induct was conservative in its approach, this new procedure always errs, if at all possible, on the side of liberalism - its conjectures always classify an example as falling under D unless it can be proved otherwise. This intriguingly mirrors the falsificationist stance of accepting given facts in the absence of contradictory data. One can note with interest the debate between verificationists and falsificationists. Although they appear to be an inverse of each other, existence claims such as "There once existed dragons" are not falsifiable, whereas universal one are, eg, "All parrots are pink."

`Hence, instead of a conservative rule of Æ _y, there is a new variable, F, which contains a list of conjectures based on Æ (Æ1 ...Æk). This produces the condition:

`y _ Æi.

`The search method can best be portrayed graphically, where F begins at _, and the variable Æ is set to ^ (figure 3). As in caution-induct, the cards are considered one at a time. Considering a negative example such as card 1, and remembering its truth-value:

`c=T, s=T, t=T.

`It can be seen that the conjecture _ (true for all rows of the truth table) classifies it incorrectly, forcing liberal-induce into strengthening its conjecture with a conjunct.

`In the case of a positive card, such as card 4, whose current Æ is:

`c & s & Øt

`...which is found to be false, so liberal-induce moves up the lattice until a new monomial is found: s & Øt.

`Note that liberal-induce contains several sub-procedures:

`ascend to update Æ as in caution-induct - in response to positive examples.

`descend to update Æ in response to negative examples.

`prune to delete some elements of Æ in response to positive examples.

We can avoid the pseudo-code and make the main point that liberal-induce arrives at a different final rule to caution-induct due to its different search method. As already mentioned, it is less conservative in predicting instances of D, and the two procedures output different results when faced with the same data. This difference in inductive bias is problematic - in order to implement induction on a machine, we must decide which one to choose. We must also have a method of checking the strength of the chosen method. The implication of the above is that different methods of induction are suited to different domains, and selecting the correct one is essential for a competent implementation of induction.

`It would appear that any method of determining the effectiveness of an inductive procedure, will itself rely upon induction; for we assess such procedures on the basis of how well they correspond to our own predictions. In fact, P.F.Strawson accepts that inductive inferences are rendered invalid by deductive methods - the premises "...do not logically entail a conclusion..." - but asserted that it is quite improper to question induction as a whole. To ask "Is induction valid?" is like asking "Is the law legal?" This analysis only seems to hold if we ask the question inductively - if we try to ascertain the

justification of induction using deductive means, Strawson and his tautology does not help us.

`Note also that these algorithms are sensitive to the description of the original task. Where the above cards might be relevant in terms of colour, which might produce the rule that a card is dax iff it has a black triangle (Øt), there are other descriptions, ie, a card is dax iff it (i) does not have a white circle, black square and black triangle and (ii) does not have a black circle and a white triangle and (iii) does not have a white square and a white triangle. (see Pratt (1994) for a fuller explanation of this)

`There are a variety of inductive inference systems, including Meta-DENDRAL, AQ11, and its ancestor INDUCE, THOTH-P, and ID3. This was created by Quinlan (1986), who moved beyond the inducement of monomials and into the realm of boolean functions. This algorithm, rather than search the concept space for evidence of matching conjectures, constructs test trees, with two branches of True and False for each node of the tree. When the branching encounters a Pass of Fail of condition, terminal nodes occur. ID3 uses a measure of efficiency to identify a good test in relation to the target concept. Given a training example, x, specified as a truth-value assignment to c, s and t, ID3 starts at the top of a test tree and examines each branch according to x. Let us suppose that x makes c false, s true and t false, ie, Card 4. The first test, at t, will yield an F result, and following this F branch brings us to the test for s. This is found to be true, and the T branch is traversed correspondingly. At the test for c, the result given is F - and following this gives a pass node, indicating that x does have the concept D.

`It would be very useful to have an efficiency indicator - for the above examples, this is not necessary, as t gives the required information. But if we change the cards which pass the test, and their positive or negative status, we can change the passes and failures of the test c. Examining each indicator in turn, we can ascertain the strength of their indicators - perhaps weak, perhaps strong, or perhaps evenly split. We can, of course, add new tests to the ends of the branches in test trees, until we reach tests which are certain indicators of D. When we have a test tree that classifies training examples correctly, this will form the rule for classifying unseen examples. The procedure here is:

`begin id3 (training-examples, proposition-letters)

`if all examples in training-examples are positive then

`return the terminal test-tree pass

`else if all the examples in training-examples are negative then

`return the terminal test-tree fail

`else

`let first-test be the test corresponding to that proposition letter in proposition-

`letters which is the most efficient indicator for D.

`let new-proposition-letters be proposition-letters with first-test removed

`let pos-examples be those members of training-examples which pass first-test

`let neg-examples be those members of training-examples which fail first-test

`let true-tree be result of

`id3 (pos-examples, new-proposition-letters)

`let test-tree be the tree formed by taking first-test as the first test, and by

`appending true-tree to the T-branch of that test, and false-tree to the F-branch

`return test-tree

`end if

`end id3

ID3 can also employ an 'approximate' inductive process, to cope, for example, when certain relevant features are unknown. This is is a closer simulation of human induction where some features are inadequate. In such a case, one of the recursive calls will be made, where new-proposition-letters is the empty list, and one could expect to terminate the tree with pass if most of the examples are positive, and with fail if mainly negative.

`Recall the aforementioned problem with inferring that all parrots are of the same colour, say, pink. This underlines the uncertainty of inference, and the only solution for rational human agents seems to be to rely, as much as they can, upon probability.

`Carnap has much of interest to say on this. He distinguishes between statements such as "The probability of obtaining heads with a fair coin is 1/2" and "It is highly probable, given the available evidence, that Smith is guilty as charged." The latter statement refers to a logical relation between evidence and the hypothesis based upon it - and this is the relevant one with regards to induction. Regrettably his attempts to logically symbolise this ended before he realised his vision. He did combine simplicity with some intuitive consequences, but again we have the problem of the caution-induct/liberal-induct variety - the problem of how to choose between different inductive biases.

`It can be argued that the entire range of empirical knowledge is based on inductive inference, and unreliable. As a universal law induction fails, for we need every single case of x ever happening in order to be certainly confident of x's occurence in the future - an impossible task. As humans we turn to probability as a way out, and this looks to be the case with most succesful implementations of induction. Remeber that the choice of

algorithm is partly dependent on the context of the problem, and considerations of speed (eg, employing liberal-induct for speed, despite it's zealous nature) Since the traditional problems of induction seem intractable, all we can hope for is, at best, an accurate simulation of our own questionable inference.