Introduction to Machine Learning

Concept Learning

Varun Chandola

February 5, 2017

Outline

Contents

1 Concept Learning

The basic form of learning is concept learning. The idea is to learn a general description of a category (or a concept) from specific examples. A concept is essentially a way of describing certain phenomenon; an object such as a table, an idea such as steps that will make me successful in life.

The need to go from specific to general is the core philosophy of machine learning.

In the context of machine learning, concepts are typically learnt from a set of examples belonging to a super-category containing the target category; furniture. Thus the learnt concept needs to be able to distinguish between the target and everything else. The goal of concept learning is:

- Infer a boolean-valued function $c : x \rightarrow \{\texttt{true,false}\}$
- Input: Attributes for input x
- Output: `true` if input belongs to concept, else `false`
- Go from specific to general (Inductive Learning).

1.1 Example – Finding Malignant Tumors

Attributes

1. **Shape** `circular,oval`
2. **Size** `large,small`
3. **Color** `light,dark`
4. **Surface** `smooth,irregular`
5. **Thickness** `thin,thick`

Concept
Malignant tumor.

For simplicity, we assume that the attributes that describe the objects are boolean. One can even think of these attributes as constraints over the actual attributes of the objects.

1.2 Notation

- X - Set of all possible instances.
  - What is $|X|$?
- Example: {`circular,small,dark,smooth,thin`}
- D - Training data set.
  - $D = \{\langle x, c(x)\rangle : x \in X, c(x) \in \{0, 1\}\}$
- Typically, $|D| \ll |X|$

The number of all possible instances will be 2^d, where d is the number of attributes. If attributes can take more than 2 possible values, the number of all possible instances will be $\prod_{i=1}^{d} n_i$, where n_i is the number of possible values taken by the i^{th} attribute.

1.3 Representing a Possible Concept - Hypothesis

As mentioned earlier, a concept can be thought of as a function over the attributes of an object which produces either true or false. For simplicity, we assume that the concept is:

- A conjunction over a subset of attributes
  - A malignant tumor is: circular **and** dark **and** thick
  - {`circular,?,dark,?,thick`}

In practical setting this is too simplistic, and you will rarely find a setting where a simple conjunctive concept will be sufficient.

The target concept is unknown. We have made strong assumptions about the form of the target concept (conjunctive), but it still needs to be learnt from the training examples.

- Target concept c is unknown
  - Value of c over the training examples is known

1.4 Hypothesis Space

- **Hypothesis**: a potential concept
- Example: {`circular,?,?,?,?`}
- **Hypothesis Space** ($\mathcal{H}$): Set of all hypotheses
  - What is $|\mathcal{H}|$?
- Special hypotheses:
  - Accept everything, {`?,?,?,?,?`}
  - Accept nothing, $\{\emptyset, \emptyset, \emptyset, \emptyset, \emptyset\}$

As in most inference type of tasks, concept learning boils down to searching for the best hypothesis that approximates the target concept from the entire hypothesis space.

As shown above, in a hypothesis, each binary attribute can have 3 possibilities, '?' being the third one. There can be total 3^d possible hypotheses. There is one more hypothesis which rejects (gives a negative output) for any example. Thus there can be $3^d + 1$ possibilities. In general, the size of the hypothesis space, $\mathcal{H} = \prod_{i=1}^{d} n_i + 1$.

Note that we represent the accept nothing or reject everything hypothesis with $\{\emptyset, \emptyset, \emptyset, \emptyset, \emptyset\}$ or just $\emptyset$ for simplicity. One can think of the value '$\emptyset$' to signify that the input be rejected no matter what the value for the corresponding attribute.

2 Learning Conjunctive Concepts

2.1 Find-S Algorithm

1. Start with $h = \emptyset$
2. Use next input $\{x, c(x)\}$
3. If $c(x) = 0$, goto step 2
4. $h \leftarrow h \wedge x$ (pairwise-and)
5. If more examples: Goto step 2

6. Stop

Pairwise-and rules:

$$a_h \wedge a_x = \begin{cases} a_x & : \; if \; a_h = \emptyset \\ a_x & : \; if \; a_h = a_x \\ ? & : \; if \; a_h \neq a_x \\ ? & : \; if \; a_h = ? \end{cases}$$

In Mitchell book [1, Ch. 2], this algorithm is called Find-S. The objective of this simple algorithm is to find the maximally specific hypothesis.

We start with the most specific hypothesis, accept nothing. We generalize the hypothesis as we observe training examples. All negative examples are ignored. When a positive example is seen, all attributes which do not agree with the current hypothesis are set to '?'. Note that in the pairwise-and, we follow the philosophy of specific to general. $\emptyset$ is most specific, any actual value for the attribute is less specific, and a '?' is the least specific (or most general).

Note that this algorithm does nothing but take the attributes which take the same value for all positive instances and uses the value taken and replaces all the instances that take different values with a '?'.

This algorithm will never accept a negative example as positive. Why? What about rejecting a positive example? Can that happen? The answer is yes, as shown next using an example.

The reason that a negative example will never be accepted as positive is because of the following reason: The current hypothesis h is consistent with the observed positive examples. Since we are assuming that the true concept c is also in $\mathcal{H}$ and will be consistent with the positive training examples, c must be either more general or equal to h. But if c is more general or equal to h, then an example that is rejected by c must be rejected by h as well. That means that a negative example will never be accepted as positive.

Let us try a simple example.

Target concept
{?,large,?,?,thick}

- How many positive examples can there be?
- What is the minimum number of examples need to be seen to learn the concept?

1. {circular,**large**,light,smooth,**thick**}, malignant
2. {oval,**large**,dark,irregular,**thick**}, malignant

- Maximum?

Target concept
{?,large,?,?,thick}

1. {circular,large,light,smooth,thick}, malignant
2. {circular,large,light,irregular,thick}, malignant
3. {oval,large,dark,smooth,thin}, benign
4. {oval,large,light,irregular,thick}, malignant
5. {circular,small,light,smooth,thick}, benign

- Concept learnt:
  - {?,large,light,?,thick}
  - What mistake can this "concept" make?

This concept cannot accept a malignant tumor of type which has dark as the color attribute. This is the issue with a learnt concept which is more specific than the true concept.

- Objective: Find maximally specific hypothesis
- Admit all positive examples and nothing more
- Hypothesis never becomes any more specific

Questions

- Does it converge to the target concept?
- Is the most specific hypothesis the best?
- Robustness to errors

- **Choosing best among potentially many maximally specific hypotheses**

The Find-S algorithm is easy to understand and reasonable. But it is strongly related to the training examples (only the positive ones) shown to it during training. While it provides a hypothesis consistent with the training data, there is no way of determining how close or far is it from the target concept. Choosing the most specific hypothesis seems to reasonable, but is that the best choice? Especially, when we have not seen all possible positive examples. The Find-S algorithm has no way of accomodating errors in the training data. What if there are some misclassified examples? One could see that even a single bad training example can severely mislead the algorithm.

In many cases, at a given step, there might be several possible paths to explore, and only one or few of them might lead to the target concept. Find-S does not address this issue. Some of these issues will be addressed in the next section.

2.2 Version Spaces

Coming back to our previous example.

1. {circular,large,light,smooth,thick}, malignant
2. {circular,large,light,irregular,thick}, malignant
3. {oval,large,dark,smooth,thin}, benign
4. {oval,large,light,irregular,thick}, malignant
5. {circular,small,light,smooth,thin}, benign

- Hypothesis chosen by **Find-S**:
  - {?,large,light,?,thick}
- Other possibilities that are **consistent** with the training data?
  - {?,large,?,?,thick}
  - {?,large,light,?,?}
  - {?,?,?,?,thick}
- What is **consistency**?

- **Version space**: Set of all consistent hypotheses.

What is the consistent property? A hypothesis is consistent with a set of training examples if it correctly classifies them. More formally:

Definition 1. *A hypothesis h is* **consistent** *with a set of training examples D if and only if $h(x) = c(x)$ for each example $\langle x, c(x)\rangle \in D$.*

The **version space** is simply the set of all hypotheses that are consistent with D. Thus any version-space, denoted as $VS_{\mathcal{H},D}$, is a subset of $\mathcal{H}$.

In the next algorithm we will attempt to learn the version space instead of just one hypothesis as the concept.

2.3 LIST-THEN-ELIMINATE Algorithm

1. $VS \leftarrow \mathcal{H}$
2. For Each $\langle x, c(x)\rangle \in D$:

   Remove every hypothesis h from VS such that $h(x) \neq c(x)$
3. Return VS

- Issues?
- How many hypotheses are removed at every instance?

Obviously, the biggest issue here is the need to enumerate $\mathcal{H}$ which is $O(3^d)$. For each training example, one needs to scan the version space to determine inconsistent hypotheses. Thus, the complexity of the list then eliminate algorithm is $nO(3^d)$. On the positive side, it is guaranteed to produce all the consistent hypothesis. For the tumor example, $|\mathcal{H}| = 244$.

To understand the effect of each training data instance, let us assume that we get a training example $\langle x, 1\rangle$, i.e., $c(x) = 1$. In this case we will remove all hypotheses in the version space that contradict the example. These will be all the hypotheses in which at least one of the d attributes takes a value different (but not '?') from the the value taken by the attribute in the example. The upper bound for this will be 2^d. Obviously, many of the hypotheses might already be eliminated, so the actual number of hypotheses eliminated after examining each example will be lower.

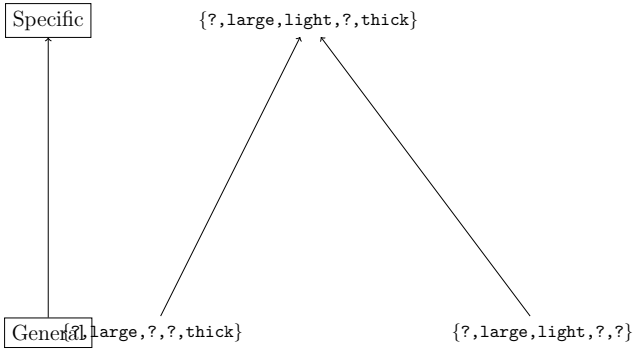
2.4 Compressing Version Space

`More_General_Than` Relationship

$$h_j \geq_g h_k \quad if \quad h_k(x) = 1 \Rightarrow h_j(x) = 1$$
$$h_j >_g h_k \quad if \quad (h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$$

Actually the above definition is for `more_general_than_or_equal_to`. The strict relationship is true if $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$. The entire hypothesis spaces $\mathcal{H}$ can be arranged on a lattice based on this general to specific structure. The Find-S algorithm discussed earlier searches the hypothesis space by starting from the bottom of the lattice (most specific) and then moving upwards until you do not need to generalize any further.

- In a version space, there are:
  1. Maximally general hypotheses
  2. Maximally specific hypotheses
- Boundaries of the version space



The maximally specific and the maximally general hypotheses determine the **boundaries** of the version space. These are defined as:

Definition 2. *The* **general boundary** G*, with respect to hypothesis space* $\mathcal{H}$ *and training data* D*, is the set of maximally general hypotheses consistent with* D*.*

$$G \equiv g \in \mathcal{H} | Consistent(g, D) \wedge (\neg \exists g' \in \mathcal{H})[(g' >_g g) \wedge Consistent(g', D)]$$

Simply put, the general boundary is a set of consistent hypotheses such that there are no other consistent hypotheses which suffice the relation `more_general_than`.

Definition 3. *The* **specific boundary** S*, with respect to hypothesis space* $\mathcal{H}$ *and training data* D*, is the set of maximally specific hypotheses consistent with* D*.*

$$S \equiv s \in \mathcal{H} | Consistent(s, D) \wedge (\neg \exists s' \in \mathcal{H})[(s >_g s') \wedge Consistent(s', D)]$$

Version Space Representation Theorem
Every hypothesis h in the version space is contained within at least one pair of hypothesis, g and s, such that $g \in G$ and $s \in S$, i.e.,:

$$g \geq_g h \geq_g s$$

To prove the theorem we need to prove:

1. Every h that satisfies the above expression belongs to the version space.
2. Every member of the version space satisfies the above expression.

Proof. For the first part of the proof, consider an arbitrary $h \in \mathcal{H}$. Let $s \in S$ and $g \in G$ be two "boundary hypotheses", such that $h \geq_g s$ and $g \geq_g h$. Since $s \in S$, it is satisfied by all positive examples in D. Since h is more general than s, it should also satisfy all positive examples in D. Since $g \in G$, it will not be satisfied by any negative example in D. Since h is more specific than g, it will also not be satisfied by any negative example in D. Since h satisfies all positive examples and no negative examples in D, it should be in the version space.

For the second part, let us assume that there is an arbitrary hypothesis $h \in VS_{\mathcal{H},D}$ which does not satisfy the above expression. If h is maximally general in $VS_{\mathcal{H},D}$, i.e., there is no other hypothesis more general than h which is consistent with D, then there is contradiction, since h should be in G. Same argument can be made for the case when h is maximally specific. Let us assume that h is neither maximally general or maximally specific. But in this case there will a maximally general hypothesis g' such that $g' >_g h$ which is consistent with D. Similarly there will be a maximally specific hypothesis s' such that $h >_g s'$ which will be consistent with D. By definition, $g' \in G$ and $s' \in S$ and $g' >_g h >_g s'$. This is a contradiction. □

This theorem lets us store the version space in a much more efficient representation than earlier, simply by using the boundaries. Next we see how the theorem can help us learn the version space more efficiently.

Another important question is, given S and G, how does one regenerate the $VS_{\mathcal{H},D}$? One possible way is to consider every pair (s, g) such that $s \in S$ and $g \in G$ and generate all hypotheses which are more general than s and more specific than g.

1. Initialize $S_0 = \{\emptyset\}$, $G_0 = \{?, ?, \ldots, ?\}$
2. For every training example, $d = \langle x, c(x)\rangle$

$c(x) = +ve$

1. Remove from G any g for which $g(x) \neq +ve$
2. For every $s \in S$ such that $s(x) \neq +ve$:
   (a) Remove s from S
   (b) For every minimal generalization, s' of s
   - If $s'(x) = +ve$ and there exists $g' \in G$ such that $g' >_g s'$
   - Add s' to S
3. Remove from S all hypotheses that are more general than another hypothesis in S

$c(x) = -ve$

1. Remove from S any s for which $s(x) \neq -ve$
2. For every $g \in G$ such that $g(x) \neq -ve$:
   (a) Remove g from G
   (b) For every minimal specialization, g' of g
   - If $g'(x) = -ve$ and there exists $s' \in S$ such that $g' >_g s'$
   - Add g' to G
3. Remove from G all hypotheses that are more specific than another hypothesis in G

For the candidate elimination algorithm, positive examples force the S boundary to become more general and negative examples force the G boundary to become more specific.

2.5 Analyzing Candidate Elimination Algorithm

- S and G boundaries move towards each other
- Will it converge?
  1. No errors in training examples
  2. Sufficient training data
  3. The target concept is in $\mathcal{H}$
- Why is it better than Find-S?

The Candidate-Elimination algorithm attempts to reach the target concept by incrementally "reducing the gap" between the specific and general boundaries.

What happens if there is an error in the training examples? If the target concept is indeed in $\mathcal{H}$, the error will cause removing every hypothesis inconsistent with the training example. Which means that the target concept will be removed as well.

The reason that Candidate-Elimination is still better than Find-S is because even though it is impacted by errors in training data, it can potentially indicate the "presence" of errors once the version space becomes empty (given sufficient training data). Same will happen if the target concept was never in the version space to begin with.

As mentioned earlier, given sufficient and accurate training examples, (best case $\log_2(VS)$), the algorithm will converge to the target hypothesis, assuming that it lies within the initial version space. But, as seen in example above, if sufficient examples are not provided, the result will be a set of boundaries that "contain" the target concept. Can these boundaries still be used?

- Use boundary sets S and G to make predictions on a new instance x^*
- **Case 1**: x^* is **consistent** with every hypothesis in S
- **Case 2**: x^* is **inconsistent** with every hypothesis in G

For case 1, x^* is a positive example. Since it is consistent with every maximally specific hypothesis in VS, one of which is more specific than the target

concept, it will also be consistent with the target concept. Similar argument can be made for case 2. What happens for other cases?

With partially learnt concepts, one can devise a voting scheme to predict the label for an unseen example. In fact, the number of hypotheses in the $VS_{\mathcal{H},D}$ that support a particular result can act as the confidence score associated with the prediction.

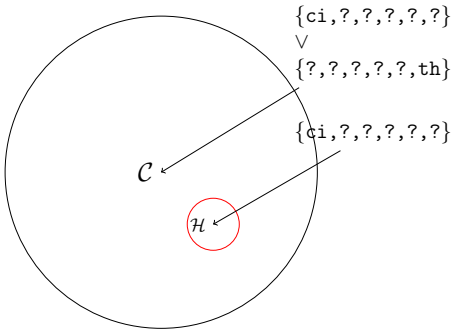
- Predict using the majority of concepts in the version space
- Predict using a randomly selected member of the version space

3 Inductive Bias

- Target concept labels examples in X
- $2^{|X|}$ possibilities ($\mathcal{C}$)
- $|X| = \prod_{i=1}^{d} n_i$
- Conjunctive hypothesis space $\mathcal{H}$ has $\prod_{i=1}^{d} n_i + 1$ possibilities
- Why is this difference?

For our running example with 5 binary attributes, there can be 32 possible instances and 2^{32} possible concepts ($\approx$ 4 Billion possibilities)! On the other hand, the number of possible conjunctive hypotheses are only 244. The reason for this discrepancy is the assumption that was made regarding the hypothesis space.

Hypothesis Assumption
Target concept is conjunctive.



The conjunctive hypothesis space is biased, as it only consists of a specific type of hypotheses and does not include others.

Obviously, the choice of conjunctive concepts is too narrow. One way to address this issue is to make the hypothesis space more expressive. In fact, we can use the entire space of possible concepts ($\mathcal{C}$) as the hypothesis space in the candidate elimination algorithm. Since the target concept is guaranteed to be in this $\mathcal{C}$, the algorithm will eventually find the target concept, given enough training examples. But, as we will see next, the training examples that it will require is actually X, the entire set of possible examples!

- Simple tumor example: 2 attributes - size (sm/lg) and shape (ov/ci)
- Target label - malignant (+ve) or benign (-ve)
- $|X| = 4$
- $|\mathcal{C}| = 16$

Here are the possible instances in X:

x_1: sm,ov

x_2: sm,ci

x_3: lg,ov

x_4: lg,ci

The possible concepts in the entire concept space $\mathcal{C}$:

- $\{\}$ – most specific
- $\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}$
- $\{x_1 \vee x_2\}, \{x_1 \vee x_3\}, \{x_1 \vee x_4\}, \{x_2 \vee x_3\}, \{x_2 \vee x_4\}, \{x_3 \vee x_4\}$
- $\{x_1 \vee x_2 \vee x_3\}, \{x_1 \vee x_2 \vee x_4\}, \{x_1 \vee x_3 \vee x_4\}, \{x_2 \vee x_3 \vee x_4\}$
- $\{x_1 \vee x_2 \vee x_3 \vee x_4\}$ – most general

Start with $G = \{x_1 \vee x_2 \vee x_3 \vee x_4\}$ and $S = \{\}$. Let $\langle x_1, +ve \rangle$ be the first observed example. S is modified to $\{x_1\}$. Let $\langle x_2, -ve \rangle$ be the next example. G is set to $x_1 \vee x_3 \vee x_4$. Let $\langle x_3, +ve \rangle$ be the next example. S becomes $\{x_1 \vee x_3\}$.

In fact, as more examples are observed, S essentially becomes a disjunction of all positive examples and G becomes a disjunction of everything except the negative examples. When all examples in X are observed, both S and G converge to the target concept.

Obviously, expecting all possible examples is not reasonable. Can one use the intermediate or partially learnt version space? The answer is no. When predicting an unseen example, exactly half hypotheses in the partial version space will be consistent.

- **A learner making no assumption about target concept cannot classify any unseen instance**

While the above statement sounds very strong, it is easy to understand why it is true. Without any assumptions, it is not possible to "generalize" any knowledge inferred from the training examples to the unseen example.

Inductive Bias
Set of assumptions made by a learner to generalize from training examples.

For example, the inductive bias of the candidate elimination algorithm is that the target concept c belongs to the conjunctive hypothesis space $\mathcal{H}$. Typically the inductive bias restricts the search space and can have impact on the efficiency of the learning process as well as the chances of reaching the target hypothesis.

Another approach to understand the role of the inductive bias is to first understand the difference between deductive and inductive reasoning. Deductive reasoning allows one to attribute the characteristics of a general class to a specific member of that class. Inductive reasoning allows one to generalize a characteristic of a member of a class to the entire class itself. An example of deductive reasoning:

- All birds can fly. A macaw is a type of bird. Hence, a macaw can fly.

Note that deductive reasoning assumes that the first two statements are true. An example of inductive reasoning is:

- Alligators can swim. Alligator is a type of reptile. Hence, all reptiles can swim.

The inductive reasoning, even when the first two statements are true, is untrue. Machine learning algorithms still attempt to perform inductive reasoning from a set of training examples. The inference on unseen examples of such learners will not be provably correct; since one cannot deduce the inference. The role of the inductive bias is to add assumptions such that the inference can follow deductively.

- Rote Learner – No Bias
- Candidate Elimination – Stronger Bias
- Find-S – Strongest Bias

References

References

[1] T. M. Mitchell. Machine Learning. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.