Introduction to Machine Learning

CSE474/574: Lecture 3

Varun Chandola <chandola@buffalo.edu>

30 Jan 2015

Outline

Contents

1 Learning Conjunctive Concepts **1**
1.1 Find-S Algorithm 1
1.2 Version Spaces 3
1.3 LIST-THEN-ELIMINATE Algorithm 3
1.4 Compressing Version Space 4

1 Learning Conjunctive Concepts

1.1 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 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,

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?

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 lecture.

1.2 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.

1.3 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.

1.4 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

References

References

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