Introduction to Machine Learning

CSE474/574: Lecture 4

Varun Chandola <chandola@buffalo.edu>

2 Feb 2015

Outline

Contents

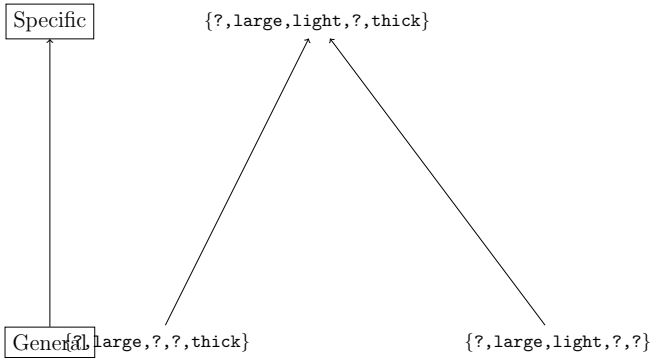
1 Compressing Version Spaces

`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 1. *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 2. *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 right hand side of the above expression belongs to the version space.
2. Every member of the version space satisfies the right hand side of 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 right hand side of 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.

1.1 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.

- **Halving Algorithm**
  - Predict using the majority of concepts in the version space
- **Randomized Halving Algorithm** [1]
  - Predict using a randomly selected member of the version space

References

References

[1] W. Maass. On-line learning with an oblivious environment and the power of randomization. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, pages 167–178, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.