Algorithm Independent Aspects of Machine Learning

*Note: this post meant to help clarify the tutorial questions (number 1, 3, 4, 5) for COMP 9417 – Week 11, School of Computer Science and Engineering, UNSW (s1 – 2017)*

Regardless of machine learning algorithm that we choose, we still may want to know about these stuffs (slide – page 6):

* How many training examples a learner should have before it converges to correct hypothesis? (**sample complexity**)
* How large is a hypothesis space? How complex is a learner’s hypothesis? (**hypothesis complexity**)
* How many errors a learner are allowed to misclassify before it finally converges to a successful hypothesis? (**mistake bounds**)

Let’s take a look at first aspect.

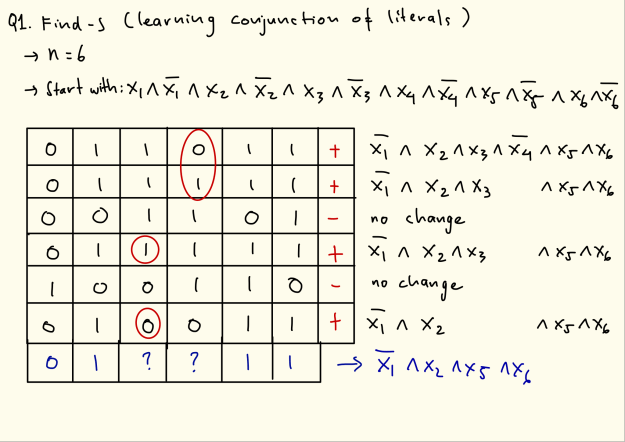
**Sample complexity – a simple example**

Say we have simple **consistent learner** (learner that makes no mistake on the training data) to learn disjunction of *d*features. Target concept *c*is assumed to be expressed as disjunction of literals, where a literal is defined as some features *xi* being true.

First of all, we need to think about how to create such a learner. In lecture slide (page 50), here is **FIND-S** algorithm to learn **conjunction** of literals:

* + Initialise*h* to the most specific hypothesis: l_1 \wedge \lnot l_1 \wedge l_2 \wedge \lnot l_2 \cdots l_n \wedge \lnot l_n 
  + For each **positive** training instance *x*: remove from *h* any literal that is not satisfied by *x*
  + Output hypothesis *h*

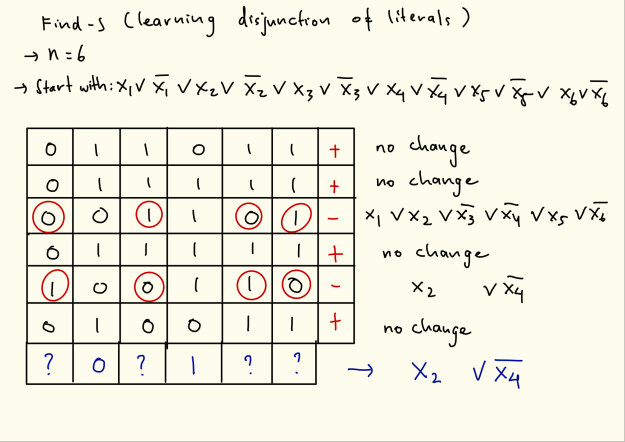
It is easier to understand this simple algorithm with an example.

[](https://i0.wp.com/robotlearning.net/wp-content/uploads/2017/05/Find-s.png)

Then we could just modify above FIND-S algorithm to learn **disjunction** of literals.

* Initialise*h* to the most specific hypothesis: l_1 \wedge \lnot l_1 \wedge l_2 \wedge \lnot l_2 \cdots l_n \wedge \lnot l_n 
* For each **negative** training instance *x*: remove from *h* any literal that is satisfied by *x*
* Output hypothesis *h*

Again, example is the easiest way to understand this concept.

[](https://i1.wp.com/robotlearning.net/wp-content/uploads/2017/05/2.png)

The next step is finding the hypothesis space (\lvert H \rvert ). If we assumed that the target concept is the disjunction of *d*relevant binary features, then the hypothesis space is the amount of possible combination of those literals:

\lvert H \rvert = 2^d 

Once we know \lvert H \rvert , we can analyse the sample complexity of our learner by finding *m*(the amount of training examples needed so the learner can converge to a correct hypothesis):

m \geq \frac{1}{\epsilon}(\ln\lvert H \rvert + \ln(1/\delta)) 

Since we know that : \lvert H \rvert = 2^d  , then

m \geq \frac{1}{\epsilon}(d \ln 2 + \ln(1/\delta)) 

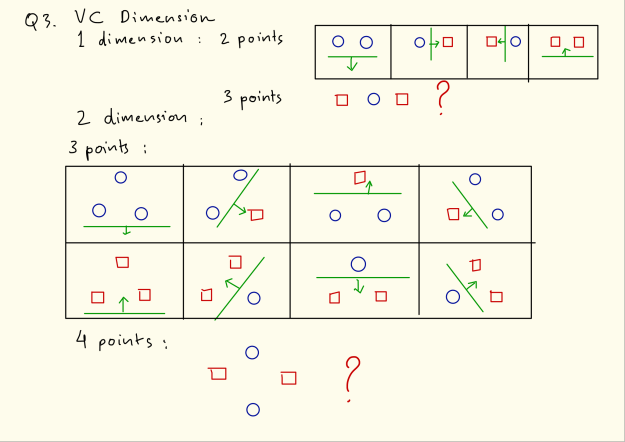
Not just analyse the sample complexity, we can also evaluate our hypothesis complexity, which is the next topic in this post.

**Hypothesis complexity – VC dimension**

So far, we only measure hypothesis complexity from its “size” via a hypothesis space. Although it is useful, a hypothesis space could be very big (even infinite). It is good if we have another measurement that informs us a hypothesis’ expressivity or capacity to classsify the examples. Vapnik–Chervonenkis (VC) dimension provides us with this information. Here is definition of VC dimension:

*size of largest set of instances that can be shattered by a particular hypothesis language or model class*

For example, in 1 dimension world, how many points can be “shattered” into every possible combination? 2 points can be shattered, while 3 points can’t. While for 2 dimension world, the maximum number of points that can be shattered is 3. The learner can’t be shattered 4 points as there will be case of XOR relation between the points. In general, the number of VC dimension is the number of dimension plus 1. See the picture below for the better descriptions.

[](https://i2.wp.com/robotlearning.net/wp-content/uploads/2017/05/Cv.png)

For 2 dimension case, there is possibility that 3 points are colinear. In this case, it is impossible to shatter those points. However, the shattering theory mentions that the shattering does not have to be happened in all possible configurations (or subsets). As long as there is 1 configuration that can be shattered completely, then the amount of points can be taken as VC dimension.

Next, we will discuss about the mistake bounds.

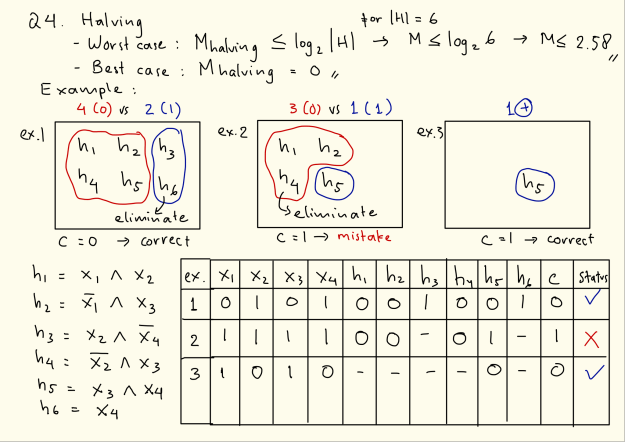
**Mistake bounds**

We want to know how many times our learner are allowed to misclassify before it finally converges to a correct hypothesis.

Here we will analyse the mistake bounds for **Halving Algorithm** for Boolean functions. In general, here is Halving Algorithm mechanism:

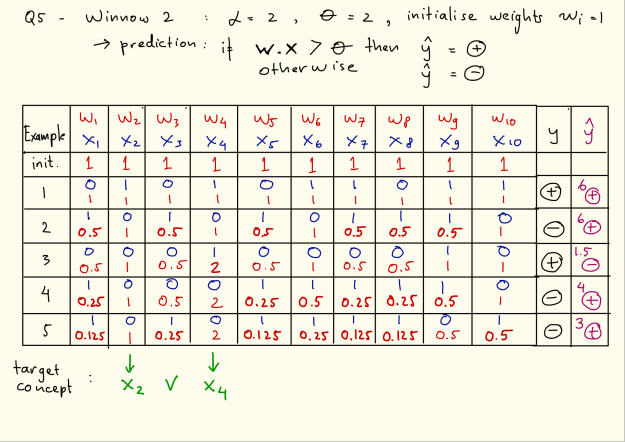
* Initialise the set of consistent hypothesis C = H
* Repeat get new instance x 
  + let \pi_0(C,x)  be subset of C that predict 0
  + let \pi_1(C,x)  be subset of C that predict 1
  + if \lvert \pi_0(C,x) \rvert > \lvert \pi_1(C,x) \rvert  predict 0 else 1
  + if class(x) = 0  then C = \pi_0(C,x)  else C = \pi_1(C,x) 

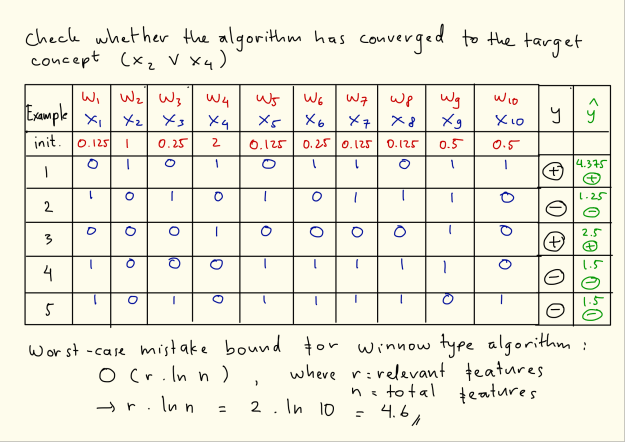
Example of simple Halving algorithm implementation with a learner that has 6 hypotheses. In every iteration, assuming a correct hypothesis is existed, the hypotheses set will be reduced at least a half (if there is a mistake) or up to a half (if the prediction is correct).

[](https://i0.wp.com/robotlearning.net/wp-content/uploads/2017/05/4.png)

From above example, a learner is only making mistake once, which is less than worst-case mistake bound (2.58). The best-case mistake bound will give no mistake at all.

Next, we will evaluate the mistake bounds of a **Winnow 2** algorithm. Winnow 2 is similar with the perceptron algorithm. The main difference is how the weights are updated if misclassifications happen. When correct class is 0 (negative), but the prediction is 1 (positive), the weights are demoted (divided by \alpha  parameter). Otherwise, when correct class is 1 (positive), but the prediction is 0 (negative), the weights are promoted (multipled by \alpha  parameter). Check the explanation below.

[](https://i0.wp.com/robotlearning.net/wp-content/uploads/2017/05/5.png)

Apparently, after 5 examples are executed, there are 2 features that has dominant weights: x_2  and x_4  which actually confirms the target concept (x_2 \vee x_4 ). Are the weights converged after 5 trials? Check the picture below.[](https://i1.wp.com/robotlearning.net/wp-content/uploads/2017/05/6.png)

We can see that our learner makes 4 mistakes before finally converges to a correct hypothesis. This is still below worst-case mistake bound (4.6).