UNIT 1 U1/0/2024

Machine Learning

COS4852

Year module

Department of Computer Science

School of Computing

CONTENTS

This document contains the material for UNIT 1 for COS4852 for 2024.

Define tomorrow.

UNISA university of south africa

1 OUTCOMES

In this unit you will learn to describe and solve a learning problem as a concept-learning task with specific reference to the concept of the hypothesis space.

More specifically, you will:

1. Learn to design a *Learning System*.
2. Learn about *Concept Learning*, which introduces the concept of a *hypothesis*.
3. Understand the *General-to-Specific ordering of hypotheses*, which form the basis of what most Machine Learning algorithms do with data.
4. Learn how the *Find-S* and *Find-G* algorithms work.
5. Understand the concept of *Version Spaces*.
6. Learn how to the *Candidate-Elimination* algorithm works.
7. Understand the problem of *Inductive Bias*, which is built into all Machine Learning algorithms.

After completion of this Unit you will be able to:

1. Discuss *Concept Learning* and what role *hypotheses* play in machine learning.
2. Formulate a given problem as a *Concept Learning* task.
3. Discuss the *General-to-Specific ordering of hypotheses* and why this ordering forms the basis of how Machine Learning algorithms work.
4. Apply the FIND-S and FIND-G algorithms.
5. Discuss the concept of *Version Spaces*.
6. Apply the CANDIDATE-ELIMINATION algorithm.
7. Understand and describe the implications of partial learning concepts.
8. Discuss the problem of *Inductive Bias* and techniques to address this problem.

2 INTRODUCTION

In this Unit you will investigate Tom Mitchell's theoretical background behind learning theory. This will be done using his idea of Concept Learning. This is covered quite well in Tom Mitchell's textbook, though you we will be using other sources and need not buy the book (though the book gives you a thorough working of this)

2.1 How do machines learn?

To understand how computers can learn from data, you need to first learn a bit more about how human learning works, and what you can use there to create machine learning algorithms. Human learning can be divided into 5 aspects:

1. Rote learning - simple memorising
2. Passive learning - getting taught by a teacher in class
3. Analogy - learning by experience, usually through activity
4. Inductive learning - learning by experience over time by creating a generalised concept
5. Deductive learning - deriving new facts from past facts

Traditional Artificial Intelligence focuses mostly on deductive processes - using logic to derive new facts. Expert Systems is a good example here. Most Machine Learning algorithms are variants on the Concept Learning theme, where concepts are learned from specific examples of what needs to be learned. Classification, regression, clustering, etc. can all be seen as specific variants of the process of finding a specific hypothesis (solution) or a set of specific hypotheses that fit best, from a (usually) pre-defined set of hypotheses.

Tom Mitchell defines this as the ”Problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training example.”

The following are the range of topics that will be covered:

1. Learning Systems
2. Concept learning
3. The general-to-specific ordering of hypotheses
4. The Find-S algorithm
5. Version spaces
6. The Candidate-Elimination algorithm
7. Inductive bias

Many of these concepts were developed by Tom Mitchell in his PhD, and subsequently in papers and books. They form a very useful theoretical framework to describe and assess machine learning algorithms.

3 PREPARATION

3.1 Online textbooks

Here are references to two textbooks that are available online, that will give you a good overview (and lots of detail) on Machine Learning.

Nils Nilsson wrote what he calls ‘notes’, which is the draft of a textbook he intended to publish. Go to `http://robotics.stanford.edu/people/nilsson/mlbook.html` and download the textbook “Introduction to Machine Learning” (`http://robotics.stanford.edu/people/nilsson/MLBOOK.pdf`).

You could also go online and try to find Max Welling’s textbook, titled “A first encounter with Machine Learning”.

Also check on the *myUnisa* site under *Additional Resources* for copies of these textbooks.

3.2 Online courses

Andrew Ng created one of the best, free online courses on Supervised Machine Learning. YOu can find this course at: `https://www.coursera.org/learn/machine-learning`.

As part of your activities later you will be asked to find other online courses online.

3.3 Online material

There are many excellent resources available online on the topic of Machine Learning. In this module we will be using some of these resources. We will provide you with links to some of these, but these are not exclusive and not necessarily the only or best. We encourage you to do your own searches (using Google or DuckDuckGo) to find more. The more different angles and approaches to learning this field you can find, the better.

Computer Science and Mathematical essentials

To learn Machine Learning you have to have a solid background in Computer Science and Mathematics, and ideally some basic Statistics as well. The following two documents give you an overview of the mathematics that you need to master to do well in Machine Learning:

- `http://courses.washington.edu/css490/2012.Winter/lecture_slides/02_math_essentials.pdf`

- http://courses.washington.edu/css490/2012.Winter/lecture_slides/06a_math_essentials_2.pdf

💡 The 'mother' site http://courses.washington.edu/css490/2012.Winter/ here contains another Machine Learning course, with some useful resources.

The following article gives a brief overview of the fundamentals in Machine Learning:

- https://towardsdatascience.com/machine-learning-basics-part-1-a36d38c7916

Designing a learning system

- Gives three examples of defining a Learning System in terms of TPE: https://www.studytonight.com/post/designing-a-learning-system-the-first-step-to-machine-learning

Concept learning

- https://www.asquero.com/article/concept-and-concept-learning/
- Uses a redacted EnjoySport example to explain *Concept Learning*, and the *Find-S* and *List-then-Eliminate* algorithms to find a list of valid *hypotheses*: https://www.studytonight.com/post/what-is-concept-learning-in-ml

The general-to-specific ordering of hypotheses

Here we see that it is possible to define a hierarchy of specificity that allows us to order the hypotheses from most general to most specific. This creates a space wherein we can search for an optimal hyypothesis (or solution to the training problem).

- Here is a summary of Mitchell's chapter 2 that discuss the concept of hypothesis ordering, and gives a short example: https://www.i2tutorials.com/machine-learning-tutorial/machine-learning-general-to-specific-ordering-of-hypothesis/
- The link here gives a godd overview of the ordering principle, but also summarises most of the other concepts you are learning in this unit: https://bi.snu.ac.kr/Courses/g-ai04_2/ML02.pdf

Find-S and Find-G algorithms

- A basic layman's decription of *Find-S* https://www.asquero.com/article/working-of-the-find-s-algorithm

Version spaces

- https://www.studytonight.com/post/what-is-concept-learning-in-ml
- https://machinelearningmastery.com/basic-concepts-in-machine-learning/
- PDF document covering *Specific to General search*, *General to Specific search*, and the *Candidate Elimination Algorithm* https://cs.ccsu.edu/~markov/ccsu_courses/lnml-ch4.pdf
- Gives a good definition of *Version Space* http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/vspace/3_vspace.html

The Candidate-Elimination algorithm

The Candidate-Elimination algorithm effectively extends Find-S and Find-G to approach a set of valid hypotheses from both ends, by considering each instance one at a time, whether they are positive or negative.

- This YouTube video shows a step-by-step worked example of the Candidate-Elimination algorithm using the example in Tom Mitchell's textbook: https://www.youtube.com/watch?v=O2wYwFOMQ24
- Here are two links that use the same example in a text-based format:
  - https://www.getwayssolution.com/2019/12/candidate-elimination-algorithm-concept.html
  - https://www.geeksforgeeks.org/ml-candidate-elimination-algorithm/
- For those of you who like to see the implementation in code, here is some Python code that does the same: https://www.vtupulse.com/machine-learning/candidate-elimination-algorithm-in-python/

Inductive bias

In any learning system there is a trade-off between bias and being able to generalise beyond the data that was learned. An un-biased learner cannot generalise, while a learning system that needs to be able to classify unseen instances will always have some form of bias built into it.

- The link we've seen earlier also gives a good summary of the problem of bias in a learning system: https://bi.snu.ac.kr/Courses/g-ai04_2/ML02.pdf
- Here you will find a set of examples of inductive learning in a range of learning problems. It also gives examples of different types of inductive bias we find in various learning parameters, assumptions, and algorithms: https://www.i2tutorials.com/machine-learning-tutorial/machine-learning-inductive-bias-in-machine-learning/

4 DISCUSSION

4.1 Learning systems

A computer can be said to learn from *experience E* with respect to some class of *tasks T* and *performance measure P*, if its performance *P* at tasks in *T* improves with experience *E*. This is a formal definition of learning as defined by Tom Mitchell. Such a formal definition allows us to build and compare machine learning systems.

If we can find suitable *T*, *P*, and *E* parameters for a given learning problem, we can define the parameters and type of target function, choose/define the target function to be learned, and decide on an appropriate learning algorithm.

Here are three examples of learning problems and their *T*, *P* and *E* values:

Problem 1: Handwritten Character Recognition

T: Classify handwritten characters given images of characters.

P: Percentage of characters correctly classified.

E: A set of labeled images of handwritten characters, where the label is the associated alphabet character.

Problem 2: Mail SPAM detector

T: Classify a given e-mail message as either *SPAM* or *NOT SPAM*.

P: Percentage of e-mails correctly classified as either *SPAM* or *NOT SPAM*.

E: A set of labeled e-mail messages, where the label is either *SPAM* or *NOT SPAM*.

Problem 3: Play the game of Checkers

T: Make the moves required to play a game of Checkers.

P: Percentage of Checkers games won.

E: A set of supervised games playing against itself or a teacher.

Let's take the example of a checkers-playing program to investigate this further:

Choose a target function

Design an evaluation function that will assign a numerical value to any given board state, in such a way that higher values are associated with better board states (from the machine learner's perspective). Let:

- $V : B \to \mathbb{R}$ – the value function V maps any given board state B to a real number.
- 1. If b is a final board state that has won, then $V(b) = 100$
  2. If b is a final board state that has drawn, then $V(b) = 0$
  3. If b is a final board state that has lost, then $V(b) = -100$
  4. If b is a not final state, then $V(b) = V(b')$, where b' is the best final board state that can be achieved from board position b, assuming both players are playing optimally.

We now need to find a working definition of the ideal target function V that can be implemented in a learning algorithm. In other words, we need to write the target function in terms of variables and coefficients. The learning algorithm will only acquire an approximation of the target function, and for this reason the process of learning the target function is called *function approximation*. This means that we need to find a balance between a representation of the target function that is more expressive (contains more detail), but suffers from requiring much more training data, to a less expressive function that does not need as much data, and may not learn as well.

We could define the target function as a linear combination of what we see in the current board position. The variables could be:

- x_1 is the number of black pieces on the board
- x_2 is the number of red pieces on the board
- x_3 is the number of black kings on the board
- x_4 is the number of red kings on the board
- x_5 is the number of threatened black pieces on the board
- x_6 is the number of threatened red pieces on the board

Threatened pieces are those that can be captured on the next move. Each variable can be assigned a parameter that could be learned by the system:

$$\hat{V}(b) = w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 + x_5 w_5 + x_6 w_6$$

becomes our target function. The task of the learning system is to find values for the weights w_1 that will maximise $\hat{V}$. The weight w_0 is an additive constant.

Training set

To learn the target function V we require a set of training examples. Each training example is an ordered pair of the form:

$$\langle b, V_{train}(b)\rangle$$

where b is a specific board state and $V_{train}(b)$ is the training value for b.

We can estimate training values for looking at possible subsequent board positions of b, and using the V-value for that board position:

$$V_{train}(b) \leftarrow \hat{V}(Successor(b)$$

Adjusting the weights

There are many possible ways to define the best hypothesis (the set of weights in this case). A good first approach is to minimise the squared error E between the training values and those predicted by the hypothesis $\hat{V}$:

$$E = \sum_{\langle b, V_{train}(b)\rangle} (V_{train}(b) - \hat{V}(b))^2$$

over all the training samples.

This means that we are trying to find the weights, or $\hat{V}$ that minimises E for the observed training examples.

The training algorithm

An appropriate training algorithm will search through the possible legal board states and try to find values for p_i that maximises V. With a game such as Checkers the possible legal board positions are huge, and an exhaustive search would be impossible.

The Least Means Square (LMS) training rule is one of many algorithms that could be used to incrementally adjust the weights. The LMS update rule is a simple algorithm that works as follows:

For each training example $\langle b, V_{\mathbf{train}}(b)\rangle$:

- Use the current weights to calculate $\hat{V}(b)$
- For each weight w_i update it as: $w_i \leftarrow w_i + \eta(V_{train}(b) - \hat{V}(b)x_i)$

where η is a training parameter that determines the scale of weight updates.

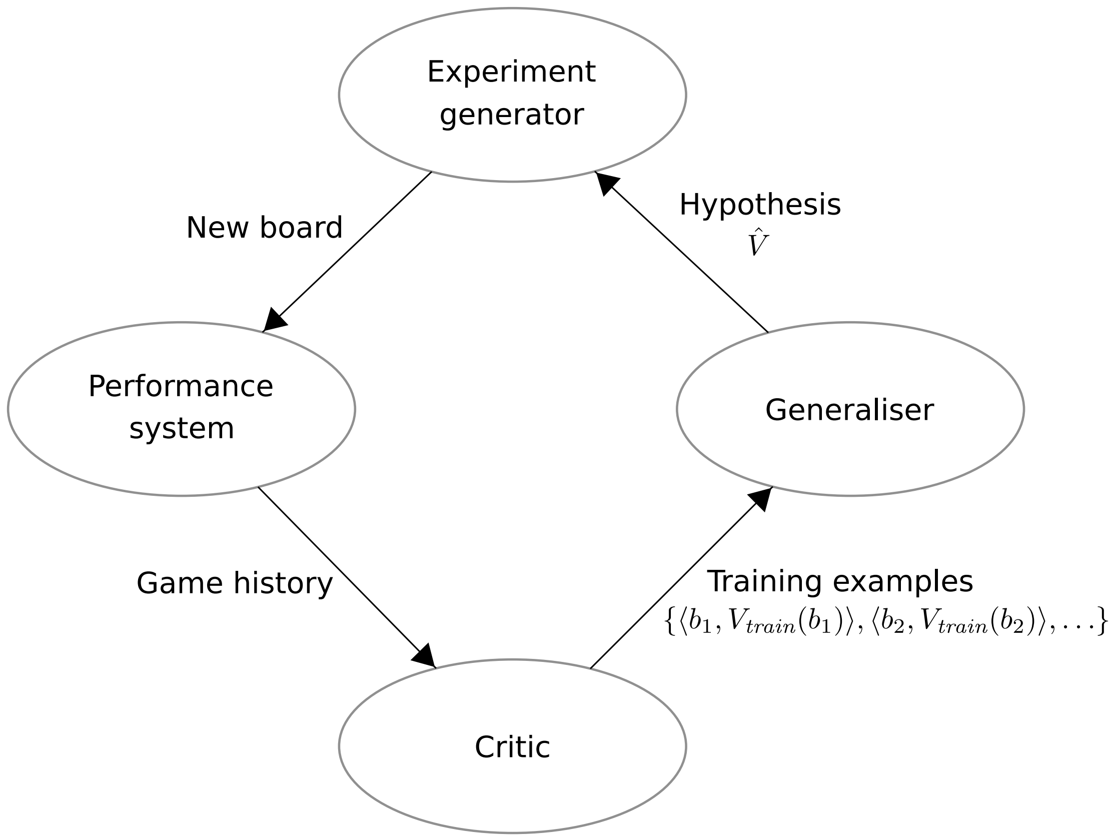
Experiment generator
New board
Hypothesis $\hat{V}$
Performance system
Generaliser
Game history
Training examples $\{\langle b_1, V_{train}(b_1)\rangle, \langle b_2, V_{train}(b_2)\rangle, \ldots\}$
Critic

Figure 1: The training system

Final design of the learning system

The learning system can now be defined as in Figure 1.

4.2 Version spaces

Let X be an instance space consisting of points in the Euclidian plane with integer coordinates (x, y), with positive and negative instances as shown in Figure 2. Positive intances are indicated as green circles, and negative instances are indicated as red triangles.

Let **H** be the set of hypotheses consisting of the region between two vertical lines. Formally, this hypothesis has the form $h \leftarrow \langle a < x < b\rangle$, where $a < b$ and $a, b \in \mathbb{Z}$ ($\mathbb{R}$ is the set of integers, $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$). This can be shortened to $h \leftarrow \langle a, b\rangle$. One such hypothesis, $h \leftarrow \langle -5, 1\rangle$ is shown in green in Figure 3.

If the instance space is not limited there are an infinite number of hypotheses. Assume (for purposes of explanation) that the instance space is limited to $-11 \leq x \leq 7$. The hypothesis space will then be all the green areas that can be drawn with $-11 \leq a \leq 6$ and $-10 \leq b \leq 7$, with $a < b$ (remember that $a, b \in \mathbb{R}$). A quick calculation will show that there are $18 \times 18 = 324$ possible hypotheses, given this limited instance space. One of these hypotheses is shown in Figure 3. Most real-world problems

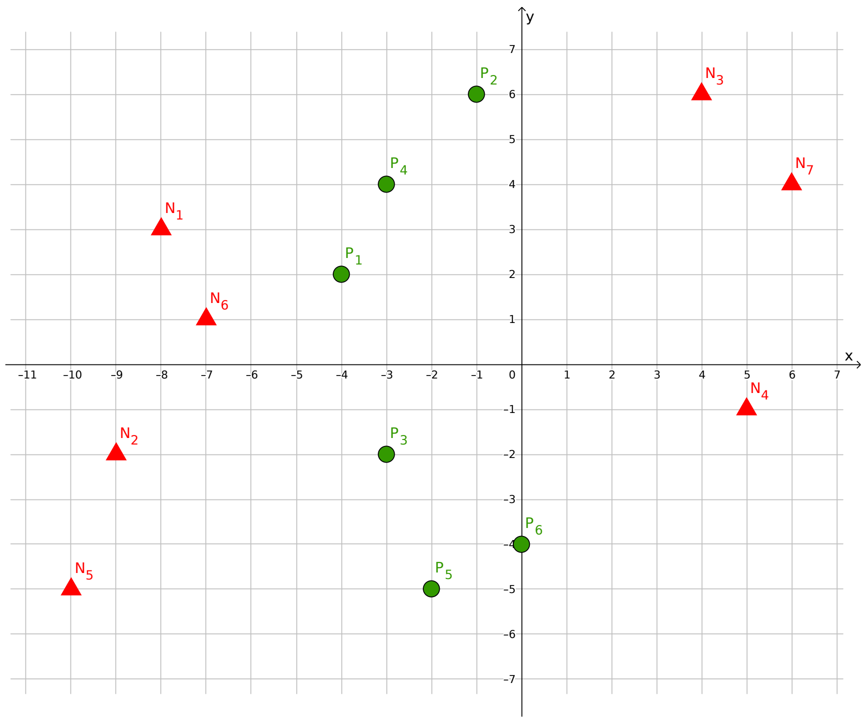

Figure 2: Instance space X.

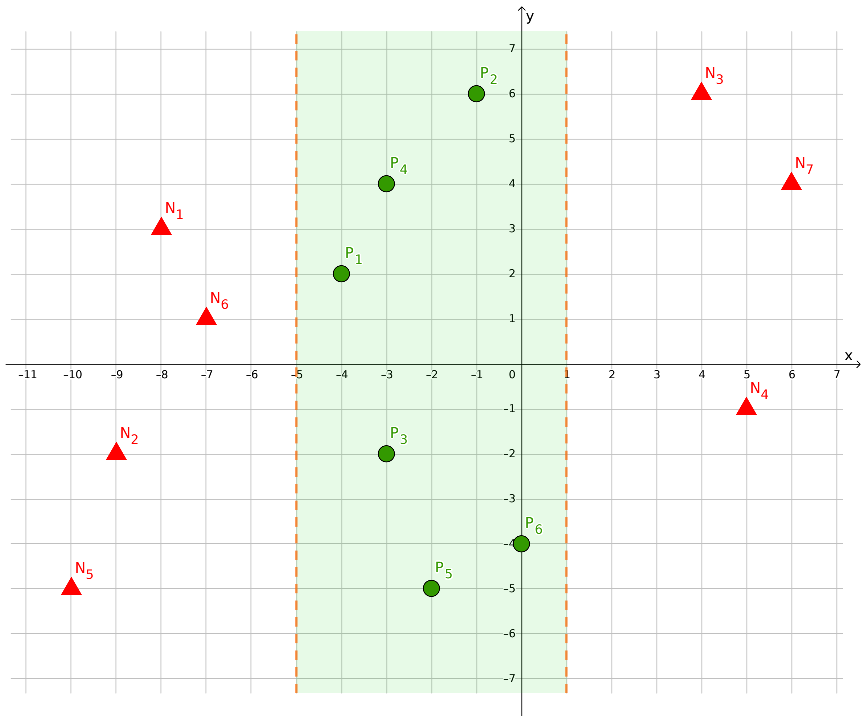

Figure 3: Instance space X with hypothesis $h_{\langle -5,1\rangle}$

have infinite instance- and search spaces. Care needs to be taken when stating the assumptions (such as instance space boundaries) so that the models remain valid.

$$
\begin{aligned}
H_{324} = \{ & \langle -11,-10\rangle, \langle -11,-9\rangle, \langle -11,-8\rangle, \langle -11,-7\rangle, \langle -11,-6\rangle, \langle -11,-5\rangle, \langle -11,-4\rangle, \langle -11,-3\rangle, \langle -11,-2\rangle, \langle -11,-1\rangle, \langle -11,0\rangle, \langle -11,1\rangle, \langle -11,2\rangle, \\
& \langle -11,3\rangle, \langle -11,4\rangle, \langle -11,5\rangle, \langle -11,6\rangle, \langle -11,7\rangle, \\
& \langle -10,-9\rangle, \langle -10,-8\rangle, \langle -10,-7\rangle, \langle -10,-6\rangle, \langle -10,-5\rangle, \langle -10,-4\rangle, \langle -10,-3\rangle, \langle -10,-2\rangle, \langle -10,-1\rangle, \langle -10,0\rangle, \langle -10,1\rangle, \langle -10,2\rangle, \langle -10,3\rangle, \\
& \langle -10,4\rangle, \langle -10,5\rangle, \langle -10,6\rangle, \langle -10,7\rangle, \\
& \langle -9,-8\rangle, \langle -9,-7\rangle, \langle -9,-6\rangle, \langle -9,-5\rangle, \langle -9,-4\rangle, \langle -9,-3\rangle, \langle -9,-2\rangle, \langle -9,-1\rangle, \langle -9,0\rangle, \langle -9,1\rangle, \langle -9,2\rangle, \langle -9,3\rangle, \langle -9,4\rangle, \langle -9,5\rangle, \langle -9,6\rangle, \langle -9,7\rangle, \\
& \langle -8,-7\rangle, \langle -8,-6\rangle, \langle -8,-5\rangle, \langle -8,-4\rangle, \langle -8,-3\rangle, \langle -8,-2\rangle, \langle -8,-1\rangle, \langle -8,0\rangle, \langle -8,1\rangle, \langle -8,2\rangle, \langle -8,3\rangle, \langle -8,4\rangle, \langle -8,5\rangle, \langle -8,6\rangle, \langle -8,7\rangle, \\
& \langle -7,-6\rangle, \langle -7,-5\rangle, \langle -7,-4\rangle, \langle -7,-3\rangle, \langle -7,-2\rangle, \langle -7,-1\rangle, \langle -7,0\rangle, \langle -7,1\rangle, \langle -7,2\rangle, \langle -7,3\rangle, \langle -7,4\rangle, \langle -7,5\rangle, \langle -7,6\rangle, \langle -7,7\rangle, \\
& \langle -6,-5\rangle, \langle -6,-4\rangle, \langle -6,-3\rangle, \langle -6,-2\rangle, \langle -6,-1\rangle, \langle -6,0\rangle, \langle -6,1\rangle, \langle -6,2\rangle, \langle -6,3\rangle, \langle -6,4\rangle, \langle -6,5\rangle, \langle -6,6\rangle, \langle -6,7\rangle, \\
& \langle -5,-4\rangle, \langle -5,-3\rangle, \langle -5,-2\rangle, \langle -5,-1\rangle, \langle -5,0\rangle, \langle -5,1\rangle, \langle -5,2\rangle, \langle -5,3\rangle, \langle -5,4\rangle, \langle -5,5\rangle, \langle -5,6\rangle, \langle -5,7\rangle, \\
& \langle -4,-3\rangle, \langle -4,-2\rangle, \langle -4,-1\rangle, \langle -4,0\rangle, \langle -4,1\rangle, \langle -4,2\rangle, \langle -4,3\rangle, \langle -4,4\rangle, \langle -4,5\rangle, \langle -4,6\rangle, \langle -4,7\rangle, \\
& \langle -3,-2\rangle, \langle -3,-1\rangle, \langle -3,0\rangle, \langle -3,1\rangle, \langle -3,2\rangle, \langle -3,3\rangle, \langle -3,4\rangle, \langle -3,5\rangle, \langle -3,6\rangle, \langle -3,7\rangle, \\
& \langle -2,-1\rangle, \langle -2,0\rangle, \langle -2,1\rangle, \langle -2,2\rangle, \langle -2,3\rangle, \langle -2,4\rangle, \langle -2,5\rangle, \langle -2,6\rangle, \langle -2,7\rangle, \\
& \langle -1,0\rangle, \langle -1,1\rangle, \langle -1,2\rangle, \langle -1,3\rangle, \langle -1,4\rangle, \langle -1,5\rangle, \langle -1,6\rangle, \langle -1,7\rangle, \\
& \langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 0,4\rangle, \langle 0,5\rangle, \langle 0,6\rangle, \langle 0,7\rangle, \\
& \langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \langle 1,6\rangle, \langle 1,7\rangle, \\
& \langle 2,3\rangle, \langle 2,4\rangle, \langle 2,5\rangle, \langle 2,6\rangle, \langle 2,7\rangle, \\
& \langle 3,4\rangle, \langle 3,5\rangle, \langle 3,6\rangle, \langle 3,7\rangle, \\
& \langle 4,5\rangle, \langle 4,6\rangle, \langle 4,7\rangle, \\
& \langle 5,6\rangle, \langle 5,7\rangle, \\
& \langle 6,7\rangle \}
\end{aligned}
$$

General-to-specific ordering of hypotheses In order to sequence the hypotheses from 'most specific' to 'most general' we need to define what is meant by 'more specific' and 'more general' (and 'less general' and 'less specific'). The most specific hypothesis would be the 'smallest' one that include none of the negative instances and all the positive instances. The most general hypothesis would be the 'largest' one with the same criteria. In the case of our instance space X, the most specific hypotheses would be those with the largest value for a and the smallest value for b that include all the positive instances and none of the negative instances. For the most general hypotheses would be those that have the smallest a value and the largest b value that include all the positive instances and none of the negative instances.

In between these two sets of hypotheses all the hypotheses can ordered from the most specific to the most general. For example, use $len(h) = |a - b|$ as the criterium to order our hypotheses. We then find that $h_{\langle -5,1\rangle}$ is more specific than $h_{\langle -5,2\rangle}$, because $len(h_{\langle -5,1\rangle}) = 6$ and $len(h_{\langle -5,2\rangle}) = 7$.

We can reduce the search space by only considering valid hypotheses. For example, the hypothesis $h_{\langle -11,7\rangle}$ is not valid because it contains both positive and negative instances. The only valid hypotheses will be those between $h_{\langle -6,1\rangle}$ and $h_{\langle -5,3\rangle}$. This reduces the search space to:

$$H_6 = \{\langle -6,1\rangle, \langle -6,2\rangle, \langle -6,3\rangle, \langle -5,1\rangle, \langle -5,2\rangle, \langle -5,3\rangle\}$$

We can order the hypotheses from most specific to most general, as shown in Figure 4.

FIND-S and FIND-G We will now use the *FIND-S* algorithm to find the most specific set of hypotheses, called the S-boundary. The process only looks at the positive instances, so only the 6 positive instances need to be considered.

The sequence of considering the instances does not effect the outcome of the algorithm if there are no errors in the data.

The set most specific hypotheses is initially empty:

$$\begin{tikzcd}
 & & & len(h) \\
 & h_{\langle 6,3\rangle} & & 9 \\
h_{\langle -5,3\rangle} \arrow[ur] & & h_{\langle -6,2\rangle} \arrow[ul] & 8 \\
h_{\langle -5,2\rangle} \arrow[u] \arrow[urr] & & h_{\langle -6,1\rangle} \arrow[u] \arrow[ull] & 7 \\
 & h_{\langle -5,1\rangle} \arrow[ul] \arrow[ur] & & 6
\end{tikzcd}$$

Figure 4: Ordering of the valid hypotheses

$$S_0 \leftarrow \{\emptyset\}$$

After observing the first positive instance $P_1 = (-4, 2)$, choose *a* and *b* to get the smallest hypothesis that still contain all the instances considered up to this point. This gives us:

$$S_1 \leftarrow \{\langle \text{-5, -3} \rangle\}$$

After considering the second positive instance $P_2 = (-1, 6)$, *S* grows to:

$$S_2 \leftarrow \{\langle \text{-5, -0} \rangle\}$$

The next three positive instances does not expand *S*. P_6 expands S to its final state:

$$S = S_6 \leftarrow \{\langle \text{-5, 1} \rangle\}$$

In a similar way we can find the set of most general hypotheses, by starting with *G* containing unknown values:

$$G_0 \leftarrow \{\langle ?, ? \rangle\}$$

We consider the negative instances one after the other. After N_1 we get:

$$G_1 \leftarrow \{\langle \text{-7}, ? \rangle\}$$

N_2 has no effect:

$$G_2 \leftarrow \{\langle \text{-7}, ? \rangle\}$$

N_3 gives us:

$$G_3 \leftarrow \{\langle -7, 3\rangle\}$$

N_4 and N_5 as no effect:

$$G_5 \leftarrow \{\langle -7, 3\rangle\}$$

N_6 reduces G to:

$$G_6 \leftarrow \{\langle -6, 3\rangle\}$$

N_7 has not further effect, and the final G-boundary becomes:

$$G = G_8 \leftarrow \{\langle -6, 3\rangle\}$$

The hypothesis shown in Figure 3 is also our S-boundary, while the G-boundary is shown in Figure 5.

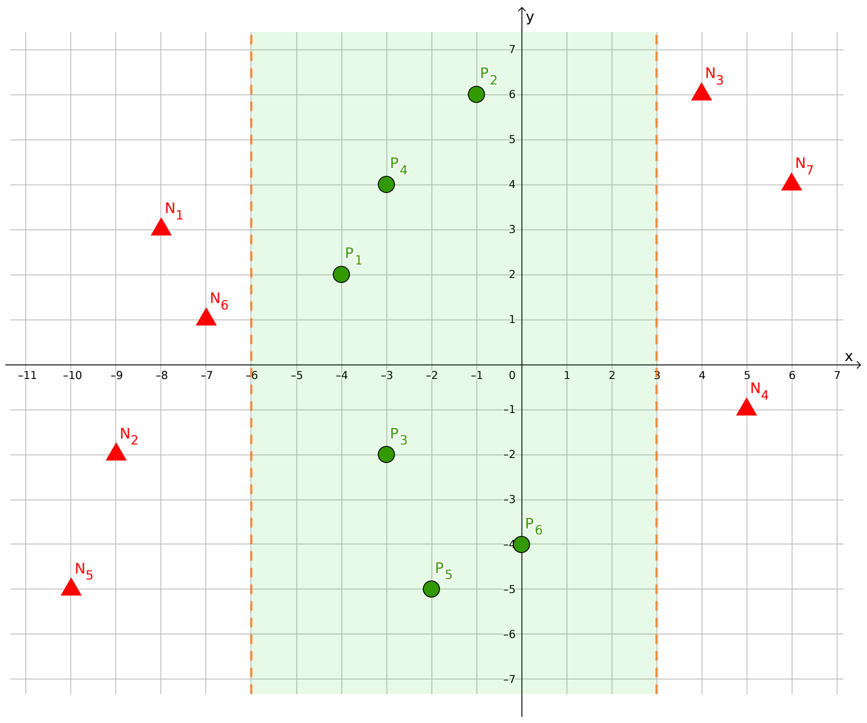

Figure 5: Instance space X with the G-boundary.

Other hypotheses It is of course possible to define any other hypothesis. Two different hypotheses are shown in Figure 6. With our current hypothesis we need two parameters (a and b) to describe it. The ellipse hypothesis will need three parameters, the center and the two axes. The blue hypotheses would be much more complex te define.

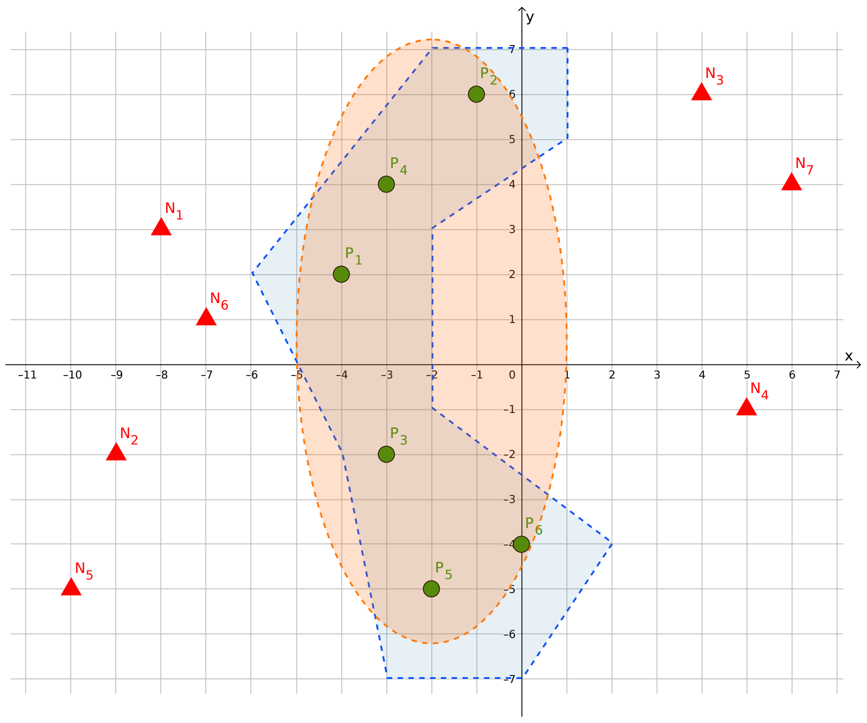

Figure 6: Alternative hypotheses for instance space X.

5 ACTIVITIES

5.1 TASK 1 - BACKGROUND

Write a report, with the following subtasks.

Online Machine Learning courses

Andrew Ng's Coursera course can be found at:

`https://www.coursera.org/learn/machine-learning`

Find at least five (5) more online courses in Machine Learning. Discuss and compare their content, and the pros and cons of each, in detail.

List the URL's so that we can find them as well. You can use them for later reference in the module.

Universities

Find at least ten (10) University-based courses (not online courses, and not the same 5 as in Subtask 1), and discuss and compare the syllabus covered by each of theses. List the URL's.

Here is one from the University of Washington, which you may include as an eleventh one:

`http://courses.washington.edu/css490/2012.Winter/CSS%20490-590%20-%20Introduction%20to%20Machine%20Learning.html`

Summarise what you've learned

Summarise what you have learnt about the structure and scope of Machine Learning from the 15+1 websites that you have found in the previous tasks. Discuss what background knowledge you would require to master Machine Learning. Find resources on the web that contain material needed to learn these background knowledge skills (HINT: see subtask 4 for one such source). List the URL's.

Mathematics background

Go to Additional Resources: Mathematics on the COS4852 site and download the files:

- ML math essentials 1
- ML math essentials 2

If you are not already well versed in these mathematical skills, use these documents as pointers, and start learning the skills.

These documents come from the University of Washington course, where you could find other useful material as well:

`http://courses.washington.edu/css490/2012.Winter/lecture_slides/02_math_essentials.pdf` `http://courses.washington.edu/css490/2012.Winter/lecture_slides/06a_math_essentials_2.pdf`

6 SUMMARY

- A potential solution to a learning problem is called a *hypothesis*.
- Concept learning is the process of searching through a (potentially infinite) predefined space of potential hypotheses.
- Hypotheses can be ordered from more general to more specific, which gives a structure in which to search for valid hypotheses.
- The Find-S algorithm performs specific-to-general search to find the most specific hypothesis.

- Find-G does a similar task to find the most general hypothesis.
- The Candidate-Elimination algorithm creates the version space by incrementally computing the sets of maximally specific (S) and maximally general (G) hypotheses - the S- and G-boundaries.
- The S- and G-boundaries delimits the entire set of valid hypotheses - those consistent with the data.
- Version spaces and the Candidate-elimination algorithm is a useful framework to understand how system learn concepts.
- Noisy and incomplete data breaks the CE algorithm, as these cannot be expressed as hypotheses.
- The indicative bias in CE is that the target concept exist in the hypothesis space.
- If the hypothesis space is enlarged to include all possible hypotheses are included, it will remove the inductive bias, but it also means that the learner cannot generalise beyong the given data.

6.1 TASK 1

Find other onlines sources on the following topics (as discussed above). Study these in detail and summarise. Use detailed worked examples to illustrate the concepts:

- Concept learning
- The general-to-specific ordering of hypotheses
- The FIND-S algorithm
- Version spaces
- The CANDIDATE-ELIMINATION algorithm
- Inductive bias

These are all concepts that were developed by Tom Mitchell as part of his PhD, and subsequently in papers and books. They form a very useful theoretical framework to describe and assess machine learning algorithms.
