EEE4114F: Introduction to Machine Learning

Jarryd Son
Dept. Electrical Engineering
University of Cape Town

May 2020

Contents

The Weird and Wonderful World of Machine Learning

1

In this chapter we will look at an overview of the field of machine learning. You will be introduced to the core concepts and terminology, as well as some examples of its applications.

1.1 What is it all about?

It's difficult to come across a piece of technology without seeing the words "Machine Learning" or perhaps more frequently "Artificial Intelligence" being thrown about by marketing campaigns. Is it really all that special and what exactly is it about? Artificial Intelligence (AI) is a subfield of computer science whose definition is particularly difficult to narrow down, but many attempts have been made to define it. For now we will say that the field of AI attempts to understand and build intelligent entities [1]. Of course the potential roadblock here is finding a concrete definition for *intelligence*, however, it is a concept that humans have a good intuition for so we will leave it at that. For those interested there is more information regarding the ideas, origins and history of Artificial Intelligence see [1, 2].

[1]: Russell et al. (2010), *Artificial intelligence: a modern approach*

[2]: Bringsjord et al. (2020), 'Artificial Intelligence'

Machine learning (ML) on the other hand is perhaps easier to define. Arthur L. Samuel [3] provides a succinct definition:

> Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed

[1]: Russell et al. (2010), *Artificial intelligence: a modern approach*

[3]: Samuel (1959), 'Some studies in machine learning using the game of checkers'

It is a field that certainly plays a key role in accomplishing the goals of AI, however it does not encapsulate the entirety of AI. To me the relationship would look as shown in Figure 1.1

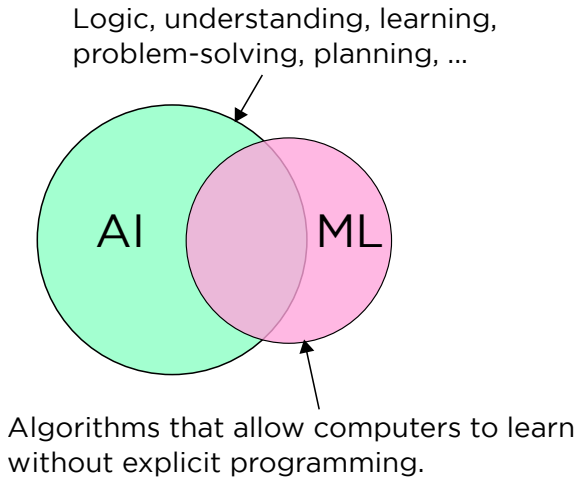

Figure 1.1: The relationship between AI and ML

AI is concerned with many aspects associated intelligence such as logic, understanding, problem-solving etc., some of which could be accomplished without self-learning algorithms. The intersection would repre-

sent the algorithms that can self-learn capabilities needed by intelligent entities. The parts of ML that lie outside of AI would be algorithms that self-learn, but aren't necessarily forming an intelligent entity.

1.2 Why would we want to use it in engineering?

A large portion of engineering revolves around understanding phenomena of interest, using our understanding to develope appropriate models of these phenomena to design and build devices or algorithms that fulfill some engineering requirement.

Much of the time the phenomena we study are highly complex and non-linear, which require large amounts of time, effort and expertise to accomplish our goals. Sometimes we resort to using greatly simplified models, but that can limit our possible solutions. Instead of doing all of this by ourselves we could try find a way to leverage machines to provide some assistance. Machines are perfectly suited to churning through raw data to find useful information from it. They don't tire, don't lose focus, they are precise, they have instant access to perfect memories etc.

Not only has machine learning proven useful in speeding up the time taken to solve complex problems, they have completely outperformed "hand-engineered" solutions across a wide variety of tasks in many different fields.

1.3 Applications of Machine Learning

As mentioned before machine learning has found a place in a variety of tasks. Some of the most prominent application areas are highlighted below:

- Computer vision: Probably one of the most researched areas in ML. This covers tasks such as object recognition, object detection
- Image processing: Image upscaling, style transfer
- Natural Language Processing (NLP): Translation, automatic captioning, identification
- Computation biology: Gene analysis, protein function prediction
- Playing games: AlphaGo, AlphaStar
- Mechatronic systems: control and navigation of mobile robots and self-driving vehicles
- Signal processing: acoustic modelling, telecommunications, beamforming, noise cancellation
- Power systems: Optimal power flow, load forecasting, fault detection
- Electronics: Antenna design [4], High-performance ADCs [5], Environmental sensor analysis

[5]: Xu et al. (2019), 'Deep-learning-powered photonic analog-to-digital conversion'

Machine Learning Algorithm Overview

2

This chapter provides an overview of the different types of machine learning tasks and scenarios.

2.1 Learning Tasks

There are a few types of common tasks that are solved machine learning. The ones we will study include: classification, regression and clustering and reinforcment learning problems.

Classification

A classification task attempts to categorize input examples (instances) into discrete classes. For example, given an image of an animal can you build an algorithm that can learn to distinguish what species of animal it is? The output would be a discrete class i.e. cat, dog, horse etc.

Perhaps a more interesting task would be to predict the authenticity of a power supply. Say, you know of some original power supplies denoted by 'O' that have good quality components, and some fake power supplies denoted by 'F'. We could find some pieces of information to characterise them, such as their average current and voltage and plot the instances on a set of axes shown in Figure 2.1. It might be possible to determine a decision boundary based on these pieces of information that allows us to make a prediction whether an unknown power supply, denoted by 'X', is an original or a fake.

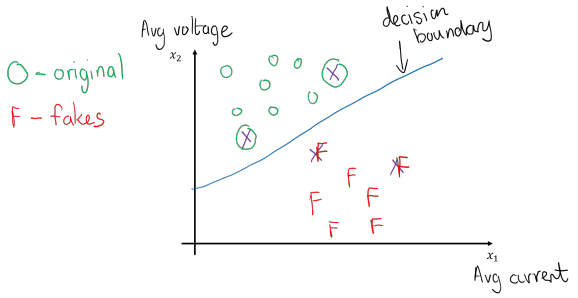

Figure 2.1: An example of a classification task to determine fake power supplies from original ones

Regression

A regression task attempts to predict a real value for a given item. For example forecasting electrical power load would take some kind of input data (historical load values, environmental data such as temperature) and could produce a real-valued output of the predicted load for the next day.

Another example could be that you are trying to develop an algorithm to set better screen brightness values for your smartphone based on ambient

light. Here the output to be predicted is the screen brightness, and the piece of information is the ambient light which is measured by your device. You could collect data of your own screen brightness habits and plot them on a set of axes, as shown in Figure 2.2

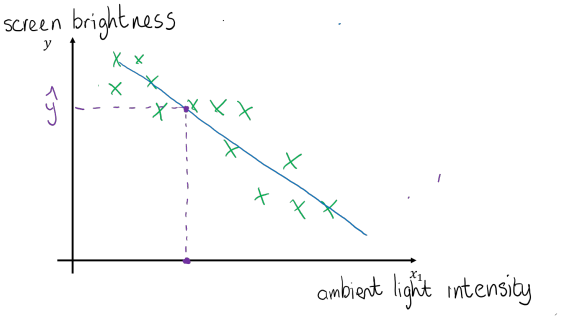

Figure 2.2: An example of a regression task that tries to predict a value for screen brightness

Clustering

A clustering task attempts to partition (group) samples together to form homogenous subsets [1]. Say you are on the hunt for extraterrestrial life and you pick up signals on your radio telescope. You have no idea what the signals mean but you would like to put similar signals together. You could frame this as a clustering task where you try and put similar signals into a given subset, as shown in Figure **??**.

1: Similar to classification except that we don't know the classes. More on this when we cover the various paradigms

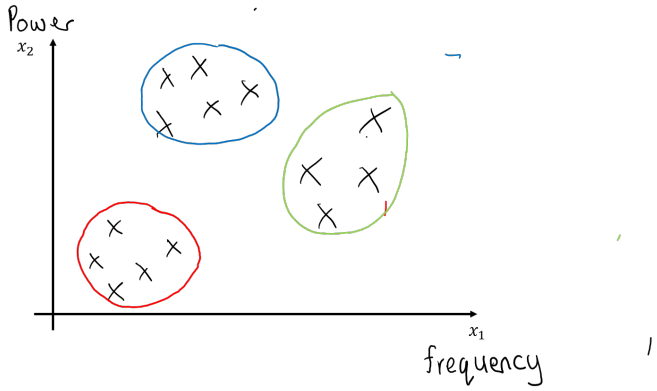

Figure 2.3: An example of a clustering task that tries to group signals from a radio telescope

Reinforcement Learning Problems

There isn't really a specific word used to describe the type of tasks tackled by reinforcement learning algorithms. Sutton and Barto [6] describe reinforcement learning as "... simultaneously a problem, a class of solution methods that work well on the problem, and the field that studies this problem and its solution". They go on to formalize the task as the optimal control of incompletely-known Markov decision processes [2]. For now you can think of it as learning how to act in a given situation to maximize some reward. For example, a mobile robot designed to sweep a minefield should learn what actions to take, in a given scenario, in

[6]: Sutton et al. (2018), *Reinforcement learning: An introduction*

2: Sounds very complicated, but more on this later

order to maximize the number of mines destroyed. An RL task could be illustrated as shown in Figure 2.4

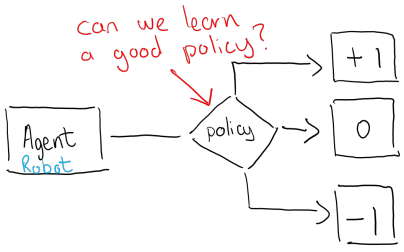

Figure 2.4: A RL task tries to learn a policy (decision make function) that allows an agent to maximize some reward

2.2 Learning scenarios

Given that we have framed a few typical machine learning tasks, there are a few scenarios [3] (paradigms) that could provide the solutions to those tasks.

3: I like how Mohri et. al [7] shown in Figure 2.5 describes them as machine learning "scenarios" rather than "categories"

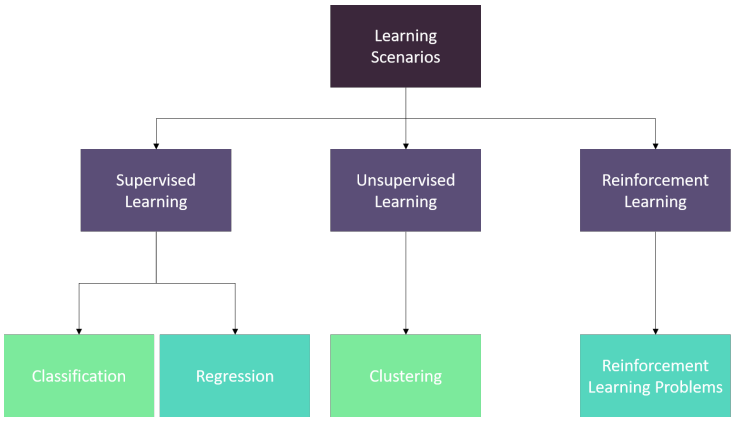

Figure 2.5: A breakdown of learning scenarios and the tasks they might provide solutions to

Supervised Learning

In a supervised learning scenario the learner is provided with **labelled** data. What this means is that we have access to both the input data and the ground truth response for that given input data. This ground truth could be created automatically by the data generating process.

For example when you setup your phone for voice recognition it prompts you to say a specific word a few times (“Hey Google!"). By doing this the dataset generated for the speech recognition algorithm contains both the input data (the audio signal) and the label (“Hey, Google”) without needing an expert to label the recorded sound after the fact.

Often this is not possible and a human expert has to manually label the data. For example, with self-driving vehicles you might want to detect and track certain entities. You would need bounding boxes around pedestrians, other vehicles or road signs. A human “expert" would be required to do the labelling process [4] in this case, which becomes very costly especially for large datasets.

4: You could use a crowd sourcing service such as Amazon’s Sagemaker Ground Truth [8]

Classification and regression tasks can usually be solved in a supervised learning scenario.

Unsupervised Learning

In an unsupervised learning scenario the learning is provided with **unlabelled** data. Due to the costs and complexities of creating labelled data we might have to provide the learning algorithm with only the input data and no ground truth. This is especially important in the era of "Big Data" where there is an over abundance of data, but most of it is unlabelled.

Clustering and dimensionality reduction are tasks that can be accomplished in the unsupervised learning scenario.

Reinforcement Learning

As mentioned before, we can tie the reinforcement learning problem to a reinforcement learning scenario. In this scenario a learning agent [5] can observe and interact with the environment. The actions the agent may take could result in a reward/penalty that evaluates how well it is performing the desired task.

5: agents can perceive and act on the world

The RL scenario can be described by the block diagram in Figure 2.6

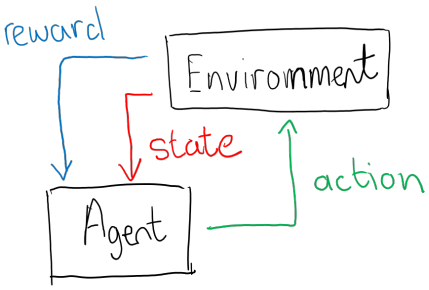

Figure 2.6: A RL scenario. An agent can sense the environment and act upon it

2.3 Learning Components

Thus far we have looked at the types of machine learning tasks we can hope to accomplish, as well as the machine learning scenarios that maybe provide solutions to these tasks. There are a few other components we need to consider before we dive deeper.

Optimization

An optimization metric is used to guide the learning process. By optimizing the model parameters based on this metric, the learning algorithm should produce desirable outputs. For example, a common metric would be the the Mean Squared Error (MSE).

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(output_\theta - y_i)^2$$

Once you have chosen a suitable optimization metric you would need to choose or design an algorithm to perform the optimization. For most of this course we will concern ourselves with gradient descent based optimization algorithms.

Evaluation

Once the learning algorithm has been trained through the optimization process it needs to be evaluated on the desired task. The optimization metric and evaluation metric might not be the same. For example, a classification algorithm might use a mean squared error (MSE) loss internally to optimize its parameters, however, the final algorithm would be evaluated on classification accuracy - what percentage of all predictions made on a set of data were correct.

Generalization

A trained model might perform exceptionally well on the exact data it learned from, however, extending it should be able to perform well on unseen data as well. This is referred to as *generalization*.

I like to think of this in terms of how students learn for a course. When you are studying you might use tutorials and past papers to learn from. Then once you are finished learning, you are presented with tests and exams to evaluate how you perform. If these tests and exams used to evaluate you are exactly the same as the tutorials, it becomes very difficult to judge how you would perform on unseen [6] data (questions).

6: Hopefully

You will often see this described by way of a decision boundary as shown in Figure 2.7

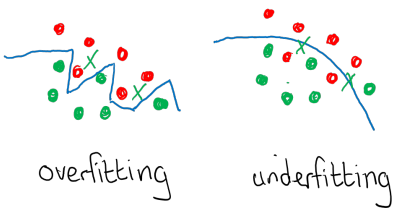

Figure 2.7: If the decision boundary is too strict it may be possible that the model has only captured the information contained directly in the training data. It is not clear that it would be able to generalise to previously unseen data.

If the model has learned a strict boundary we often say that it has *overfit* the data. If the model hasn't learned a good enough boundary we would say that it has underfit (not really the case in the figure). This also applies to other machine learning tasks, not only classification.

2.4 Terminology and notation

Here are some concepts that will be useful to know that applies to most learning tasks and scenarios. More specialised concepts will be covered in future lectures.

- *Features*: These are the individual attributes that can be measured or recorded about an event or entity being studied. This could be categorical [7] data or numerical data.
  One aspect of machine learning that needs careful attention is which features to use for your learning algorithm. Some features might have more of a significant impact than others so it might be wise to not include insignificant data [8].
  For example, let's take a look at the COVID-19 statistics in South Africa, as reported by the NICD.

7: Quite often converted to numerical values through a process called embedding

8: In an ideal situation one would hope to include as much data as possible, so that all possibilities are covered, but this is not always practical and can negatively impact how an algorithm learns

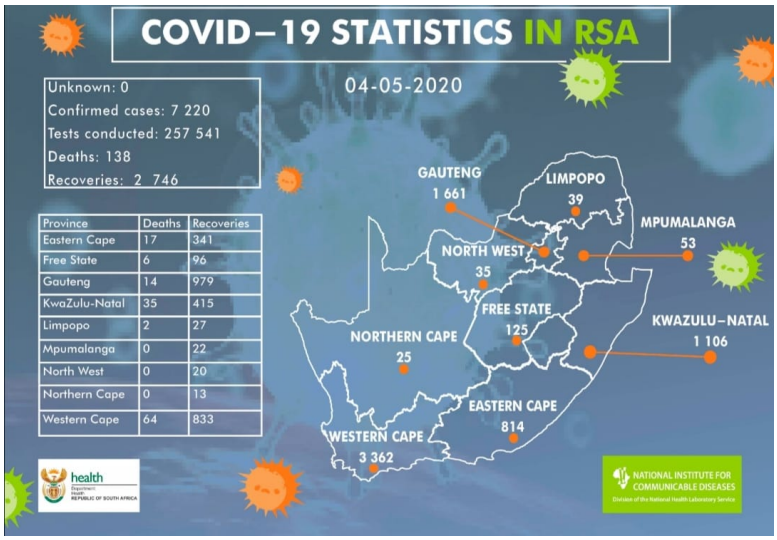

Figure 2.8: COVID-19 Statistics from 4th May 2020

There are many ways you could choose features. For this example you could decide to only look at the total statistics for the country in which case this would provide four different features:

- Confirmed Cases
- Test Conducted
- Deaths
- Recoveries

Maybe you think each individual provinces data is valuable which means you would have nine sets of the previously mentioned features for a total of 36 features. Maybe you need other features such as population density for each province, weather conditions, living standards etc.
In a computer vision task each individual pixel could be an input feature for your algorithm, or perhaps you choose to use other

features such as colour histograms etc. In future lectures we will have a brief look at some techniques that could be used to determine the most useful features from a given dataset.

- *Example*: An "example" [9] is a single instance of data used for learning or evaluation. Using the COVID-19 data you could say that the data provided for one day represents one example. A given example would have an associated set of features which are usually denoted as a vector. To formalise this we can call an example $\mathbf{x}$ which has a feature vector given by:

9: sometimes this will referred to as a sample, although this is traditionally reserved for a collection of examples

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Where $x_m^{[i]}$ represents the m^{th} feature of an example. If we use the features from the COVID-19 example we would have

$$\mathbf{x} = \begin{bmatrix} confirmed\_cases \\ tests\_conducted \\ deaths \\ recoveries \end{bmatrix} = \begin{bmatrix} 7220 \\ 257541 \\ 138 \\ 2746 \end{bmatrix}$$

The order of the features may or may not matter depending on how you approach the problem. For example, the order of the COVID-19 dataset features does not matter, but if you rearranged pixels in an image you would have a completely different instance.

- *Labels*: In the supervised learning scenario the learner has access to the ground truth labels [10] for every example. Once again these are usually denoted by a vector [11], in this case:

10: Often called target values

11: or a tensor (you can think of it as a multi-dimensional array - although not strictly correct)

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}$$

For example, the label could indicate the class [12] that an example belongs to, or maybe real values for regression.

12: usually this will be one-hot encoded e.g. If there are four possible classes you could represent this as $[0, 1, 0, 0]$, where the target vector has a value of 1 to indicate it belongs to the second class

- *Predicted labels*: A learning algorithm produces a response to a given input example. This output response is often termed a "prediction". To distinguish the ground truth labels from these predictions the *hat* symbol, $\hat{}$, is often used.

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_k} \end{bmatrix}$$

- *Hyperparameters*: Parameters not determined through learning. We will come across parameters such as learning rates, batch sizes etc. These are not learned, instead they are selected by a human or

perhaps some other process outside of the learning algorithm.

- *Training sample (set)*: The collection of examples used during a training phase. During the training phase the algorithm uses the given data to learn the necessary adjustments to its parameters
  The collection of examples can be denoted in a matrix format

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_n}^T \end{bmatrix} = \begin{bmatrix} x_1^{[1]} & x_2^{[1]} & \dots & x_m^{[1]} \\ x_1^{[2]} & x_2^{[2]} & \dots & x_m^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{[n]} & x_2^{[n]} & \dots & x_m^{[n]} \end{bmatrix}$$

  Where m is the number of input features nad n is the number of examples.
  In supervised learning you would have a corresponding sample of the labels.

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y_1}^T \\ \mathbf{y_2}^T \\ \vdots \\ \mathbf{y_n}^T \end{bmatrix} = \begin{bmatrix} y_1^{[1]} & y_2^{[1]} & \dots & y_k^{[1]} \\ y_1^{[2]} & y_2^{[2]} & \dots & y_k^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ y_1^{[n]} & y_2^{[n]} & \dots & y_k^{[n]} \end{bmatrix}$$

  Where k is the number of output variables nad n is number of samples.
- *Validation sample (set)* The collection of examples, separate from the training set, used during the validation phase. During validation the algorithm does not update its parameters in order to get an idea of how well it is generalising to unseen data with its existing parameters. One could use the performance on the validation set to make adjustments to hyperparameters to improve performance on the next training phase.
- *Test sample (set)* Once the training/validation phases have been completed, we need to know how well the final learned model performs on completely unseen data. The test set should be separate from the training and validation sets.

Introduction to Supervised Learning

3

We will begin our descent into machine learning with a more thorough exploration of supervised learning scenarios.

3.1 The Supervised Learning Process

Here we will formalize the whole supervised learning process. In a supervised learning scenario we have access to some labelled dataset consisting of N examples denoted:

$$\mathcal{D} = \{(\mathbf{x}^{[\mathbf{i}]}, \mathbf{y}^{[\mathbf{i}]}), \ldots, (\mathbf{x}^{[\mathbf{n}]}, \mathbf{y}^{[\mathbf{n}]})\}$$

Where $\mathbf{x}^{[\mathbf{i}]}$ is the i^{th} example with a corresponding ground truth label $\mathbf{y}^{[\mathbf{i}]}$.

The goal is to learn some function, which we will call a "hypothesis" function, that predicts taget values using examples (represented by vectors/tensors of features) from the dataset as inputs. This gives us:

$$h_\theta(\mathbf{x}) = \mathbf{y}$$

Where subscript θ is used to indicate that the hypothesis function is parameterised by some set of values, which we will call θ We will need to learn these parameters in order to maximize/minimize some objective function.

The process can be visualised with an informal a block diagram as shown in Figure 3.1.

The learning phase is often repeated a number of times [1]. The model can be tested on the holdout test set to determine whether or not it is appropriate for use on external data. If the model does not test well it is likely you will have to make modifications to the data, learning algorithm etc. in order to improve this. Once you are satisfied with the model you can save its parameters and deploy it on the real world.

1: depends on the algorithm, but many of the ones we will look at have an iterative approach

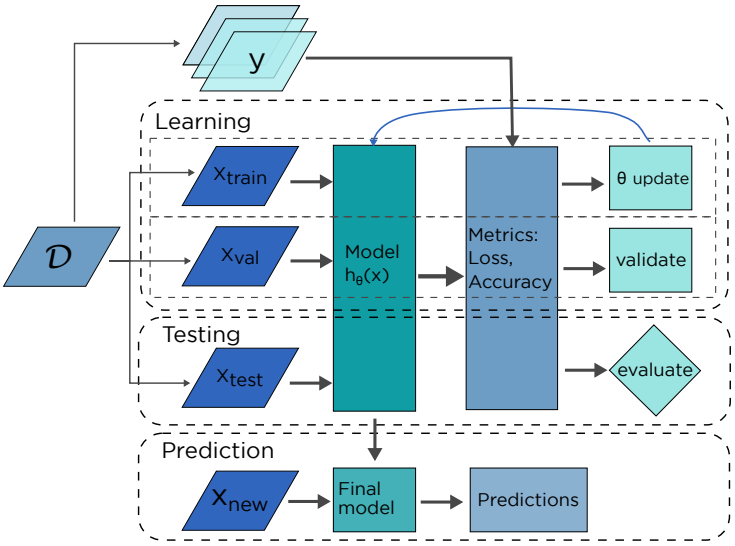

Figure 3.1: A block diagram of the supervised learning process

3.2 Simple Supervised Learning Algorithms

Now that we have an idea of what supervised learning entails we can explore some basic examples.

Linear Regression

Linear regression is often used as a starting base for understanding supervised learning. It is easy to understand and implement, but makes some assumptions that may not hold true for many real datasets. For simple linear regression [9] we will attempt to predict a single response, y, from a single predictor variable (feature), x. As the name implies a linear regression model assumes that there is an approximately linear relationship between the reponse and predictor variable, such that the hypothesis function that we will attempt to learn has the form:

[9]: James et al. (2017), *An introduction to statistical learning*

$$h_\theta(x) = y \approx \theta_0 + \theta_1 x$$

The goal is to learn the estimates of the parameters $\hat{\theta}_0$ and $\hat{\theta}_1$ that optimize some objective function that relates the gound truth value, y, with the predicted value from our algorithm, $\hat{y}$.

Suppose we have collected data regarding the battery life of a smartphone compared to it's screen size [2]. Battery life would be the response we are trying to predict, and screen size would be the feature used to predict it. There are of course other things that likely impact battery life, such as the kinds of apps the user runs and the length of time they use it for, but who knows, maybe just screen size alone gives us a good estimate. You could see the linear regression problem as:

2: Being electrical engineers we could of course look through datasheets to figure out current draw estimates for the components and use that to estimate battery life, but that could take a fair bit of effort and wouldn't be much fun

$$battery\_life \approx \theta_0 + \theta_1 \times screen\_size$$

From Section 2.3 we know that we need to choose an objective function and a method to optimize that objective function with respect to our

parameters. Let's choose our objective function to be the Mean Squared Error function given by:

$$\begin{aligned} J(\theta_0, \theta_1) &= \frac{1}{N}\sum_{i=1}^{N}(h_\theta(x_i) - y_i)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}((\theta_0 + \theta_1 x_i) - y_i)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \end{aligned}$$

Where $\hat{y}_i$ is the predicted value from our hypothesis function parameterised by θ.

It turns out that it's fairly simple to find an analytical solution to this specific problem, however, we will use a gradient descent approach, as it will be used extensively in many of the algorithms we will cover in this course.

Gradient Descent

Performing optimization by gradient descent is quite elegant in its simplicity and is quite intuitive. We can try find the gradient of a given loss function with respect to each of its parameters. If you can calculate the gradient you could change the parameter to move in the opposite direction to the gradient. Doing so multiple times should [3] move the position on the loss curve closer and closer towards a minimum. See the illustration in Figure 3.2.

3: many caveats that we will discuss at a later stage

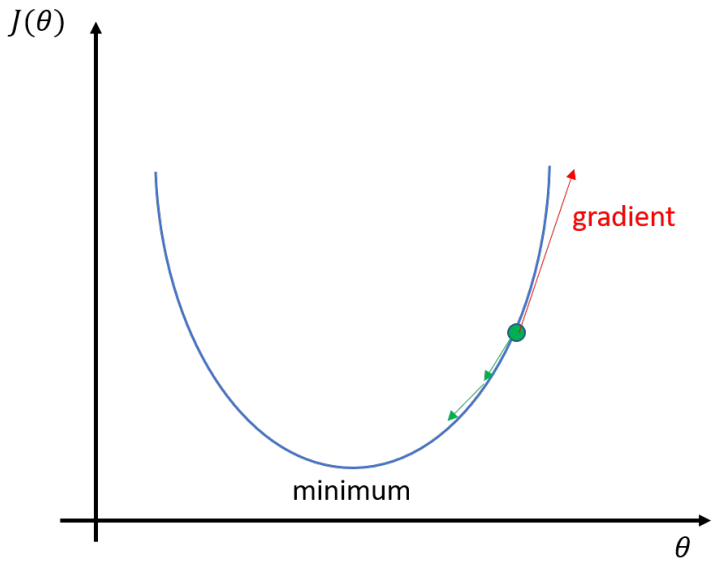

Figure 3.2: Successive movements in the opposite direction to the gradient would lead to a minimum

Here we will introduce a hyperparameter known as the learning rate, η which scales how much we move the parameter in the opposite direction to the gradient. This gives us our first learning rule that can be formalised as:

$$\theta_j := \theta_j - \eta\frac{\partial J(\theta)}{\partial \theta_j} \tag{3.1}$$

Where ":=" is an assignment operation.

So let's find the expressions for the gradients (partial derivatives) for each parameter with respect to the objective function.

For θ_0

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_0} &= \frac{\partial}{\partial \theta_0}\left[\frac{1}{N}\sum_{i=1}^{N}((\theta_0 + \theta_1 x_i) - y_i)^2\right] \\ &= \frac{2}{N}\sum_{i=1}^{N}\left[((\theta_0 + \theta_1 x_i) - y_i) \cdot \frac{\partial}{\partial \theta_0}\left[(\theta_0 + \theta_1 x_i) - y_i\right]\right] \\ &= \frac{2}{N}\sum_{i=1}^{N}\left[((\theta_0 + \theta_1 x_i) - y_i) \cdot 1\right]\end{aligned}$$

For θ_1

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_1} &= \frac{\partial}{\partial \theta_1}\left[\frac{1}{N}\sum_{i=1}^{N}((\theta_0 + \theta_1 x_i) - y_i)^2\right] \\ &= \frac{2}{N}\sum_{i=1}^{N}\left[((\theta_0 + \theta_1 x_i) - y_i) \cdot \frac{\partial}{\partial \theta_1}\left[(\theta_0 + \theta_1 x_i) - y_i\right]\right] \\ &= \frac{2}{N}\sum_{i=1}^{N}\left[((\theta_0 + \theta_1 x_i) - y_i) \cdot x_i\right]\end{aligned}$$

Once the gradients have been determined it is possible to perform the parameter update from 3.1 We can repeat this updating process until we have reach satisfactory values for the objective function, or we can choose to stop after a certain number of passes over the dataset [4]

4: The term *epoch* describes one pass over all examples

Bibliography

[1] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. 3rd ed. 2010 (cited on page 1).

[2] Selmer Bringsjord and Naveen Sundar Govindarajulu. 'Artificial Intelligence'. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2020. Metaphysics Research Lab, Stanford University, 2020 (cited on page 1).

[3] Arthur L Samuel. 'Some studies in machine learning using the game of checkers'. In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229 (cited on page 1).

[4] Gregory Hornby et al. 'Automated antenna design with evolutionary algorithms'. In: *Space 2006*. 2006, p. 7242 (cited on page 2).

[5] Shaofu Xu et al. 'Deep-learning-powered photonic analog-to-digital conversion'. In: *Light: Science & Applications* 8.1 (2019), pp. 1–11 (cited on page 2).

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cited on page 4).

[7] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018 (cited on page 5).

[8] Amazon. *Amazon Sagemaker Ground Truth*. 2020. URL: https://aws.amazon.com/sagemaker/groundtruth/ (cited on page 5).

[9] Gareth James et al. *An introduction to statistical learning*. Springer, 2017 (cited on page 12).