Mathematics for Machine Learning

Samuel Ireson
samuel-a.ireson@db.com

Klejdi Sevdari
klejdi-a.sevdari@db.com

Finlay Pillar
finlay-pillar@db.com

Wing Lam Wong
wing-lam-a-wong@db.com

November 7, 2024

Contents

1. Introduction

We discuss motivations for studying the mathematics behind machine learning, give an outline of the content of the course, and mention the references which were used in putting it together.

Machine learning is a term which is commonly heard and spoken, and one less commonly understood. This is in part due to the rapid growth in experimental machine learning techniques, and the lagging in mathematical explanations of these techniques. Another prominent reason is that the mathematical understanding of machine learning can, in enterprise settings, be considered as 'less useful' or 'less relevant'. Some people may also perceive the inner workings of machine learning to be a 'black box', and a subject which they couldn't understand.

We will focus on tackling the final obstruction. The perception of the value of understanding finds itself in the hands of others, but the understanding in question is firmly in our own. There are only a few mathematical concepts needed to gain a working understanding of machine learning, and we will cover these as they are needed.

We hope to present a self-contained course, covering the key topics in detail, gradually working towards popular models and algorithms which are seen in real-world machine learning applications. We will place a heavy focus on examples, ones which are hopefully relevant and familiar to the reader, rooting the work we are completing in the context of its application.

For those interested in reading more than we have time to cover, we recommend the work of Schwartz et al.[3], Bishop[1], and Faisal et al.[2], all of which were to a lesser or greater extent consulted in the writing of this course.

1.1 What is learning?

Sometimes, it is those concepts of which we have an intuitive idea which are hardest to rigorously define. What does it mean to learn? What does it mean to have learned? What does it mean to understand? These are all relevant questions in the theory of machine learning, especially the first, and we will first consider them in the context of human learning.

In the loosest sense, *learning* is the process of converting past experiences into applicable knowledge. For humans, the input to this learning can come in any number of formats. For example, we could consider the knowledge obtained from reading a book. What exactly is the process occurring in the mind of the reader which allows them to retain, and importantly, re-apply this information. We note that the process is decidedly not memorisation. As a child who has memorised their times tables cannot claim the ability of multiplication, the reader who cannot call upon knowledge in unseen circumstances cannot claim to fully understand what they have read. This gives a good hint as to an important facet of learning systems; *generalisation*.

For machines, the process is not dissimilar. The input for a learning algorithm is data, and the output (the expertise) are some tuned parameters which allow the model produce outputs from unseen data in the future. There is more nuance to this, but as we move towards mathematical rigour, it will be useful to keep this in mind.

1.2 Structure of the course

As previously mentioned, there are only few mathematical concepts which are absolutely necessary for understanding machine learning. An overview of the structure of this course is as follows,

- Linear Algebra

- Multivariable calculus
- Probability
- Optimisation
- Linear regression
- Logistic regression

Those familiar with the above topics may note that this course is structured in a 'bottom-up' fashion – we build on the theory as we move forwards. This is somewhat in contrast to other machine learning courses which take a 'top-down' approach in which the theory behind the application is uncovered as it is encountered. Both of these strategies have their strengths, and courses which are mathematically inclined typically take the same approach as us. It is however necessary to have patience with this approach. If you have previously only encountered machine learning in a practical, hands-on setting, you may struggle to understand how the early chapters of this course fit into the wider picture, but eventually all will be clear.

2. Linear Algebra

3. Multivariable Calculus

3.1 Introduction

4. Probability

Test

5. Learning Problems

Author — Samuel Ireson

We now aim to combine the concepts which we have thus far encountered, and construct a formal definition for a learning problem. We will begin by defining the necessary inputs, and the desired outputs, before working towards the fundamental definition of PAC learnability, which will allow us to have some notion of the possibility of learning a certain problem.

5.1 Introduction

We begin the chapter by laying out an example which we will reference and build around throughout.

Suppose that you are a loaner, aiming to determine if an arbitrary loanee will or will not default on the loan you give them. We phrase the problem deliberately in this manner, to emphasize the simplistic scenario in which we are working – the value of the loan is predetermined, and the goal is to provide a binary classification of loanees.

What information will be useful in this task? How will we determine if our solution is adequate? How will be provide a useful notation to describe the problem?

5.2 A formal learning model (Basic)

5.2.1 Learning input

The learner of a statistical learning problem has access to the following,

- Domain set: an arbitrary set which we typically label by $\mathcal{X}$. This set is nothing more than the objects to which we aim to assign labels. In the example outlined in the previous section, this set $\mathcal{X}$ is the set of potential loanees.
- Label set: A set which we typically label by $\mathcal{Y}$. This is the set of possible labels, e.g., $\{\text{default}, \text{no default}\}$. Of course, it is more common to denote the possible labels simply by integers, and in our binary classification example, we will take $\mathcal{Y} = \{\pm 1\}$.
- Training data: a finite sequence of pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, i.e., a sequence of labelled objects. In our case, this will be a sequence of past loaners together with information relating to their repayment. We will tend to denote the training data by $\mathcal{S} = ((x_1, y_1), ..., (x_m, y_m))$, where m is the size of the training data.

> Remark. It is important to note that the training data is described as a sequence rather than a set. This is a rather nuanced point, and not one which we will dedicate much thought to. The reason for such pedanticity is that there exist learning algorithms which are dependent on the order of the training points, and there is no necessity for $\mathcal{S}$ to be duplicate free.
>
> It is however still common to refer to $\mathcal{S}$ as the training set.

5.2.2 Learning output

- Prediction rule: the only output from a statistical learning problem is a function $h : \mathcal{X} \to \mathcal{Y}$. We also refer to the function as a hypothesis, and is a rule which the learner can use to label new elements of the domain space.

If we are considering an algorithm $\mathcal{A}$, then we will denote by $\mathcal{A}(\mathcal{S})$ or $h_{\mathcal{S}}$ the output hypothesis of the algorithm.

5.2.3 A data generating model

It is important to consider how exactly the training data is generated, as this is integral to how we consider the learning to occur. We assume that there is some probability distribution $\mathcal{D}$ over $\mathcal{X}$ which, importantly, the learner does not know about. For the time being, we also make the assumption that there is some 'correct' labelling function $f : \mathcal{X} \to \mathcal{Y} : x_i \mapsto y_i$, which maps the object x_i to it's correct label in every case. Explicitly,

$$f(x_i) = y_i \quad \forall i.$$

In every learning scenario, it is exactly this function f which the learner aims to obtain. In our example, we can suppose that we have past data, $\mathcal{S}$ which provides a source of truth in the way that each of the training examples is an individual of whom we *know* the loaning outcome.

5.2.4 Measure of success

The final element of our formal learning model is the means to quantify the success of our learning, or more strictly our hypothesis, h. In our classification problem, the error of the hypothesis h is defined to be the probability that for a given data point $x \in \mathcal{X}$, the predicted classification is not the same as the 'correct' classification. Notatively,

$$L_{\mathcal{D},f}(h) := \mathbb{P}_{x\sim\mathcal{D}}\left[h(x) \neq f(x)\right]$$

> Remark. The notation of $L_{\mathcal{D},f}$ is well chosen since it makes clear the dependence of the error of the classifier on the underlying distribution $\mathcal{D}$ and correct classification function f, although if the context is clear we will omit this for brevity.
>
> It is also common to refer to the probability $L_{\mathcal{D},f}(h)$ as the 'risk', and we will often refer to it as such.

5.3 Empirical Risk Minimisation (Intermediate)

We now have a good idea about the inputs, outputs and measures of success of a formal learning problem. The natural next step is to explore the strategy by which we can solve such problems.

It maybe initially seems obvious that we would like to minimise the risk, explicitly reducing the probability of incorrect classification, implicitly bringing our hypothesis output h closer to the true classifier f. Although this is the right idea, we are the learner is limited by their knowledge of the problem. In particular, the learning knows nothing of the distribution $\mathcal{D}$, nothing of the true classifier f, and hence nothing of the risk $L_{\mathcal{D},f}$. As a result, we must construct a different quantity which we would like to minimise.

> Definition 5.1 (Empirical risk). Given a training sequence $\mathcal{S} = ((x_1, y_1), ..., (x_m, y_m))$, the *empirical risk* with respect to $\mathcal{S}$ is defined and denoted by,
>
> $$L_{\mathcal{S}}(h) := \frac{|\{i \in \{1, ..., m\} : h(x_i) \neq f(x_i)\}|}{m}$$

This definition is basic in spite of it's verbosity. The numerator is expressing the process of counting misclassifications on the training set, and we are dividing by the size of the training set to ensure that $L_{\mathcal{S}}$ is bounded between 0 and 1.

The empirical risk is a good quantity to consider in learning problems since it is related to the training data available to us, i.e., "the snapshot of the world available to the learner"[3]. So, our strategy of learning comes down to minimising the empirical risk – finding a hypothesis h, which agrees well with f on the training set. Learning by this strategy is called *empirical risk minimisation*, referred to by ERM.

5.3.1 Overfitting

We now explore a pitfall into which many learning algorithms fall, and one which we will illustrate using the framework so far developed in this chapter, working again with the example of loaning.

> Example 5.2. In order to illustrate overfitting in the ERM paradigm, we suppose that the underlying distribution $\mathcal{D}$ on loan applicants $\mathcal{X}$ is uniform, that is, the correct classification of the feature space is half-half between default (1) and non-default (-1).
>
> It may seem initially that the process of finding an ERM hypothesis is challenging, but this is not actually the case. We can find a scenario agnostic ERM hypothesis,
>
> $$h_{\mathcal{S}}(x) = \begin{cases} y_i & \text{if } \exists i \in \{1, ..., m\} \text{ s.t. } x_i = x \\ 1 & \text{otherwise} \end{cases}$$
>
> What exactly is this hypothesis doing? If the selected data point $x \in \mathcal{X}$ can be found in the training set $\mathcal{S}$, then the hypothesis returns the label which is assigned to x in the training set. Otherwise, the hypothesis returns 1, i.e., predicts that the loanee will be good for their loan and won't default. This hypothesis $h_{\mathcal{S}}$ is ERM since is agrees exactly with the labels provided in the dataset, and hence has $L_{\mathcal{S}}(h_{\mathcal{S}}) = 0$ – there is zero probability of the hypothesis incorrectly classifying an instance of the training set. However, if the hypothesis returns -1 on only a finite number of instances, then $L_{\mathcal{D},f}(h_{\mathcal{S}}) = 1/2$.
>
> We have found a hypothesis which according to our strategy provides perfect learning, but in reality, performs very badly. That is, the hypothesis cannot generalise. It is 'overfitted' to the training set.

The previous example outlines a clear failure of the ERM paradigm, but we shouldn't let this provide obstruction to it's use in total; instead we will find ways to reprimand it's issues.

5.4 Limiting the hypothesis space

One way to address the problem of overfitting in ERM is to limit the possible hypotheses which we will output from learning. In particular, the learner should explicitly define a set of possible predictors. We call this set the *hypothesis class*, and typically label it by $\mathcal{H}$, which contains hypotheses of the form we have seen previously; functions $h : \mathcal{X} \to \mathcal{Y}$.

In order to distinguish the ERM learning rule from the hypothesis restricted analogue, we use $\text{ERM}_{\mathcal{H}}$. Furthermore, we can understand the $\text{ERM}_{\mathcal{H}}$ learning rule as minimisation of the empirical risk over the set of hypotheses. That is,

$$\text{ERM}_{\mathcal{H}}(\mathcal{S}) \in \arg\min_{h \in \mathcal{H}} \{L_{\mathcal{S}}(h)\}.$$

> Notation. For those who haven't seen it before, the notation $\arg\min_{x \in \mathbb{R}}\{f(x)\}$ denotes the value (or set of values) of x in the set $\mathbb{R}$ which minimises the function f. So, in our case, we are looking for the hypothesis h in the hypothesis set $\mathcal{H}$ which minimises the empirical risk function $L_{\mathcal{S}}$.

It is natural to ask how exactly a learner should go about limiting/defining a hypothesis set. The restriction to a certain set of hypotheses should be based on some prior knowledge of the problem, the process of choosing this hypothesis set is referred to as *inductive bias*. It should also be fully considered whether or not the restricted hypothesis set is a good one; will the restriction actually prevent overfitting of the learner? We will explore some examples in this direction later on.

EXAMPLE 5.3. A natural way to limit the potential hypotheses is to limit the size of the hypothesis set. That is, we may impose a finiteness condition on $\mathcal{H}$. We could let $\mathcal{H}$ be the set of all predictors which can be implemented via python programs of at most 10^{32} characters. There are a number of other common examples.

5.4.1 Assumptions on the learning problem

We are limiting the hypothesis space in order to prevent overfitting, but how can we be sure that this doesn't limit our ability to learn the problem. That is, how can we be sure that there is an ERM hypothesis h contained in the limited space $\mathcal{H}$. The short answer is that we can't, but we actually only want to consider situations where this is the case.

DEFINITION 5.4 (Realisability). The *realisability assumption* is the assumption that there exists $h^* \in \mathcal{H}$ such that $L_{\mathcal{D},f}(h^*) = 0$. In particular, our hypothesis set $\mathcal{H}$ contains a hypothesis which minimises the true risk.

Under the realisability assumption, we know that $L_{\mathcal{S}}(h^*) = 0$, and furthermore know that every ERM hypothesis is also such that $L_{\mathcal{S}}(h_{\mathcal{S}}) = 0$.

It is also important to mention an assumption we make on the training set $\mathcal{S}$. Since the learner only has access to $\mathcal{S}$, and we are interested in the true risk of the hypothesis h, we must consider the relationship between $\mathcal{S}$ and $\mathcal{D}$.

DEFINITION 5.5 (Independent and identically distributed). The *i.i.d. assumption* is the assumption that the training set $\mathcal{S}$ is constructed as a sequence of independent, and identically distributed examples according to the true distribution $\mathcal{D}$. We denote this relationship by $\mathcal{S} \sim \mathcal{D}^m$.

REMARK. It is a useful exercise to track the randomness of our process.

- The training set $\mathcal{S} \sim \mathcal{D}^m$ is an m-collection of randomly chosen i.i.d. examples labelled by f.
- The hypothesis $h_{\mathcal{S}}$ inherits this randomness since it is selected according to it's minimisation of $L_{\mathcal{S}}$.
- The true risk of this hypothesis, $L_{\mathcal{D},f}(h_{\mathcal{S}})$ inherits this randomness from $h_{\mathcal{S}}$.

It is not reasonable to expect with certainty that the training sample $\mathcal{S}$ will be representative of $\mathcal{D}$, and as such we introduce a parameter δ, to quantify this uncertainty. Explicitly, δ denotes the probability of attaining a non-representative sample $\mathcal{S}$ from $\mathcal{D}$.

5.5 PAC Learnability (Challenging)

DEFINITION 5.6 (PAC learnability). A hypothesis class $\mathcal{H}$ is said to be *Probably Approximately Correct (PAC) learnable* if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm $\mathcal{A}$ that satisfy the following condition.

For every $\epsilon, \delta \in (0,1)$, for every probability distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labelling function

$f : \mathcal{X} \to \mathcal{Y}$, under the assumption of realisability, and with $m \geq m_{\mathcal{H}}(\epsilon, \delta)$,

$$\mathbb{P}_{\mathcal{S} \sim \mathcal{D}^m} \left[L_{\mathcal{D},f}(h_{\mathcal{S}}) \leq \epsilon \right] \geq 1 - \delta$$

Remark. If you haven't taken any higher level mathematics courses, it can be said, almost with certainty, that this is the most intimidating definition you have seen. The majority of the intimidation is down to (necessarily) verbose notation. The idea hidden in the definition is however fairly intuitive.

If we think only of the name of the definition, we can gain some insight as to what we are trying to describe.

- **Correct**: the algorithm returns a hypothesis which is correct. That is, the risk of $h_{\mathcal{S}}$ is zero.
- **Approximately Correct**: the algorithm returns a hypothesis which is correct within some bound. That is, the risk of $h_{\mathcal{S}}$ is less than some approximation parameter ϵ.
- **Probably Approximately Correct**: the algorithm returns a hypothesis which is approximately correct with probability $1 - \delta$ where δ is a parameter quantifiying the probability.

So, as we make the restrictions on the quality of the returned hypothesis stricter, i.e., we decrease ϵ to ensure that the hypothesis performs better on the problem, and we decrease δ to ensure that we obtain a suitably learned hypothesis with greater probability, we have a function $m_{\mathcal{H}}$ which will tell us how *large* our training data set $\mathcal{S}$ must be, in order to learn the problem according to our restrictions.

Proposition 5.7. Every finite hypothesis class $\mathcal{H}$ is PAC learnable.

5.6 Exercises

5.6.1 Exercises (Basic)

Problem 5.1. What are the elements of a statistical learning problem which the learner has access to?

Solution 5.1. There are three elements of the problem which the learner has access to,

- Domain set
- Label set
- Training data

You could make an argument that the training data which is available to the learner is representative of the access which the learner has to the domain set $\mathcal{X}$. If the learner had full access to $\mathcal{X}$, learning by memorisation would be a successful method.

5.6.2 Exercises (Intermediate)

5.6.3 Exercises (Challenging)

6. Convex Learning Problems

Another common method of limiting the hypothesis space is by asserting it to be convex. We will explore learning problems in this area in the following chapter, encountering some challenging associated concepts.

6.1 Convexity

Definition 6.1 (Convex set). A set C is said to be *convex* if for every pair of points $a, b \in C$, the following inclusion holds,

$$\{\lambda a + (1 - \lambda)b : \lambda \in [0, 1]\} \subseteq C$$

There is a good geometric intuition for this definition. In particular, the set which must be included in C describes the straight line from a to b. So, a set C is convex if and only if it contains every straight line between any two points. See the figure below.

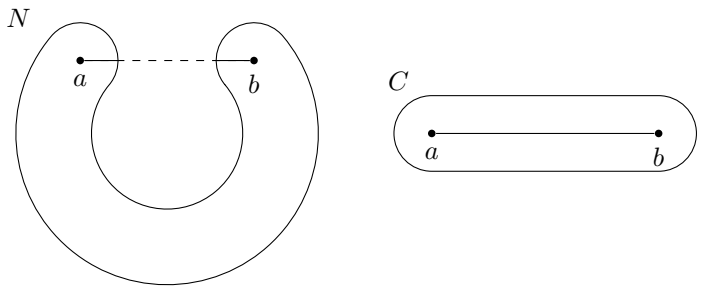

Figure 6.1: The set C is convex, but since we can find a line which is not contained in the set N, this set is not convex.

7. Optimisation Theory

Author — Samuel Ireson

8. Linear Regression

9. Logistic Regression

Bibliography

[1] Christopher Bishop. *Pattern Recognition and Machine Learning*. 2006.

[2] Marc Peter Faisal Aldo; Deisenroth and Cheng Soon Ong. *Mathematics for Machine Learning*. 2020.

[3] Shalev-Schwartz and Ben-David. *Understanding Machine Learning - from theory to algorithms*. 2014.