A Mathematicians Guide to Deep Learning (A Series of Articles)

Table of Contents

Introduction

This will be the first in a series of articles that I have written for two reasons. Firstly to solidify my knowledge of the mathematical foundations of the subject. Secondly, I am posting the series online to help those, like myself, who are interested in the mathematics of deep learning but find it hard to know where to start or where to look.

I have a friend with a similar mathematical background to myself, studying for a PhD in Mathematical Physics. I would say his research interests lie in geometrically understanding physical theories (some might say he is interested in a version of Hilbert's sixth problem). He is also interested in the mathematical aspects of deep learning. It is to him that I aim this work and thus the level of mathematics needed is high. One does not need to have a PhD to approach the topic (I do not), but an undergraduate mathematical education will suffice.

The purpose is to provide an insight into the overlap of physics/geometry with machine learning theory. Specifically how the paper on Gauge-CNN's uses gauge theory to solve some problems in modern deep learning (DL) theory.

List of mathematical topics that it is assumed the reader is at least familiar with the notation of the following topics (to be updated as th series expands):

- Measure theory
- Probability theory
- Differential Geometry
- Basic Gauge Theory
- ...

Mathematically Stating Machine Learning Problems

A formal model of the task of learning from data is given by the Statistical Learning framework.

This consists of six main ingredients: 1) ***Domain space***: The set of all possible inputs 2) ***Output space***: The set of all possible outputs 3) ***Training data/ dataset***: A set of examples on input-output pairs that the model will learn from. 4) ***Predictor***: A function from the domain space to the output space which represents the patterns learned from the training data. 5) ***Data model***: A probability distribution that we assume is 'responsible' for generating the data. 6) ***Success measure/ Loss function***: A measure telling us how close our predictor was to the correct output.

Let's define these more rigorously:

> A ***dataset*** $S = \{(x_1, y_1), ..., (x_n, y_n)\}_{i \in I}$ is a finite collection of pairs of points in the cartesian product of vector spaces $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is called the ***domain space*** & $\mathcal{Y}$ is called the ***output space***. The dataset consists of n data points indexed by the indexing set I.

The goal of the machine learning algorithm is then to learn the patterns that give rise to this input-output correspondence represented by the dataset. It is for this reason that the algorithm is often called the learner.

We postulate that this correspondence is described by an underlying probability distribution. That is, the output space is related to the input space by a **fixed unknown probability distribution** $\mathcal{D}$ over the **data space** $\mathcal{X} \times \mathcal{Y}$ which is *identically and independently distributed.* Meaning that the samples are all generated from the same distribution and have no knowledge of other samples.

Assuming the existence of a probability space $(\mathcal{X} \times \mathcal{Y}, \mathcal{F}, P)$ where $\mathcal{F}$ is a $\sigma-$algebra over $\mathcal{X} \times \mathcal{Y}$ and P is a probability measure. Then our data is assumed to be random variables sampled from some unknown distribution $\mu : \mathcal{B} \to [0,1]$, $\mathcal{B}$ the set of Borel subsets of $\mathbb{R}$.

Alternatively one may assume that the sample space is simply $\mathcal{X}$ and then also assume the existence of a **target function** $f : \mathcal{X} \to \mathcal{Y}$ which 'labels' the input space points with some value $y \in \mathcal{Y}$. Herein we shall use $f(x)$ and $y \in \mathcal{Y}$ interchangeably.

To learn the patterns that are assumed to arise from this distribution, we define two functions: > The **predictor/hypothesis** is a function $h : \mathcal{X} \to \mathcal{Y}$. We suppose that the function belongs to some function space $-$ which the leaner then 'searches'.

That is, the hypothesis $h \in A(S) \subset -$, where $A(S)$ is the subset of the function space formed through the choice of algorithm A and training of the learner over the dataset S. This training is usually some method of minimisation of a **loss function**

The loss function is the probability that the hypothesis does not predict the correct label $(y \in \mathcal{Y})$ for a given random data point $(x \in \mathcal{X})$ sampled from the underlying distribution

$$\mathcal{L}_{\mathcal{D},f}(h) = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq f(x)].$$

f is considered to be the target function, which always takes a randomly sampled $x \in \mathcal{X}$ to the correct $y \in \mathcal{Y}$.

Essentially, this is the error of incorrect prediction after random sampling from the underlying distribution, which it is important to note that the learner has no knowledge of.

So how does the learner determine this error if it has no knowledge of either the distribution or the target function? This is where the concept of training comes in.

During the training phase of a ML algorithm we provide the learner with a snapshot of the dataset along with the correct labels $y \in \mathcal{Y}$ of the data points $x \in \mathcal{X}$. The learner then iteratively tweaks the hypothesis function to learn the patterns which minimise a imitiation of the true loss function, called the training loss:

The training loss of a hypothesis function $h_U : \mathcal{X} \to \mathcal{Y}$ on a given proper subset of the total dataset $U \subset S$ is

$$\mathcal{L}_U(h) := \frac{\Big|\{(u,v) \in U \mid h(u) \neq v\}\Big|}{|U|}$$

There are many problems with this approach, such as: 1) There is a tendency to overfit the snapshot, meaning the learner only learns the patterns which have occured in the snapshot which of course may not be representative of the whole distribution. 2) It is often the case that biases can be introduced through the choice of snapshot. Such as, favouring a particular label class that appeared much more frequently in the snapshot or even having an example of a label class which never appeared in the snapshot. Clearly a learner trained on a snapshot with missing label classes will not generalise well to the actual full dataset.

The list continues but this gives a decent overview to appreciate the problems that we will undoubtedly come across in future articles

---

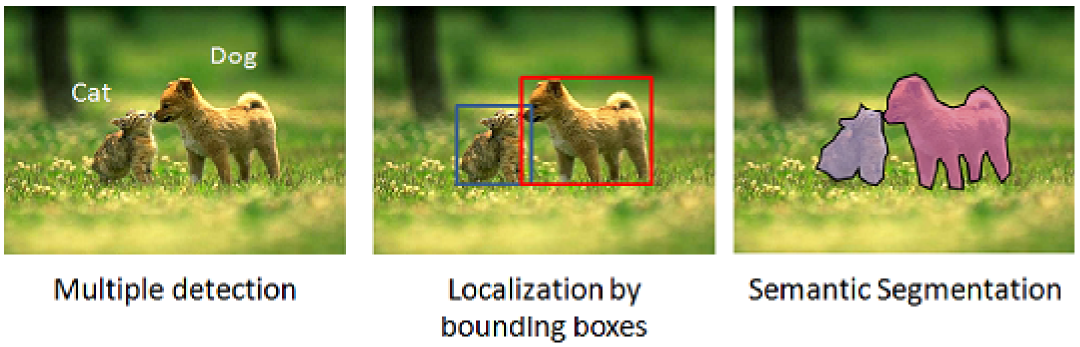
Concluding Remarks

Mathematically machine learning algorithms are procedures which iteratively establish a function $h : \mathcal{X} \to \mathcal{Y}$ called the **hypothesis**. The domain of the function is some **input space** $\mathcal{X}$ and maps into the **output space** $\mathcal{Y}$. These are generally considered to be vector spaces but often more structure is attributed to them. For example, one may assume they are Banach spaces, graphs, manifolds & Lie group structures. This is much more common these days with the rise of geometric deep learning.

It is assumed that **there exists an underlying probability distribution** $\mu : \mathcal{B} \to [0,1]$, $\mathcal{B}$ the set of Borel subsets of $\mathbb{R}$ from which random variables are sampled as input data. This is defined on the probability space $(\mathcal{X} \times \mathcal{Y}, \mathcal{F}, P)$ where $\mathcal{F}$ is a $\sigma-$algebra over $\mathcal{X} \times \mathcal{Y}$ and P is a probability measure.

Alternatively one may assume that the distribution is over another probability space over just the input space $\mathcal{X}$, along with some way to assign labels to these points, usually done using a **target function** $f : \mathcal{X} \to \mathcal{Y}$. That is, the function associates some value to each point in the input space $\mathcal{X}$.

In practice, these labels/output points can be categories for the task of prediction. Say in a image recognition context where we want to distinguish cats from dogs in images. But more complicated labels exist, for example in segmentation of those same animals within an image. I.e. where the entity which is a dog is not only labeled as such but its outline is gathered by the algorithm.



Detection, localization and segmentation example from the Coco dataset.

The iterative process by which one associates the hypothesis functions is called **training**. In essence it is some minimisation process, generally encapsulated by the term empirical risk minimisation. The function that we want to minimise is called a **loss function**, it is simply the probability that the hypothesised label is incorrect. However, due to the fact that the true underlying distribution/target function are not known during training, we use a substitute loss function called the **training loss**. This is a measure of how accurate our predictions were against some subset of the data which we know the labels of, this subset is called the **training dataset**.

References

This article is my personal take on some sections from a combination of the following books/papers:

- Shalev-Shwartz, S. and Ben-David, S., 2016. Understanding Machine Learning. New York: Cambridge University Press.
- Jeffrey S Rosenthal.A First Look at Rigorous Probability Theory. WORLDSCIENTIFIC, 2nd edition, 2006.
- Guy Lebanon and John Lafferty. Riemannian Geometry and Statistical Machine Learning. PhD thesis, USA, 2005. AAI3159986.
- Dog/cat segmentation image source