Kolmogorov's 0-1 law

James Gibson

April 2022

1 Introduction

This project will regard proving in Lean Kolmogorov's 0-1 Law - a standard result in probability theory regarding the probabilities of sets in the Tail σ-algebra. For a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ and independent random variables $\{f_n\}_{n=0}^{\infty}$ the Tail σ-algebra is constructed as follows:

1. First, for n and k in the naturals, define the σ-algebra $\mathcal{F}_n^{n+k} := \sigma\left(f_n, ..., f_{n+k}\right)$.
2. Second, for n in the naturals, define a second σ-algebra by $\mathcal{F}_i := \sigma\left(\bigcup_{i=0}^{\infty} \mathcal{F}_n^{n+i}\right)$
3. Finally, define the Tail σ-algebra by $\tau := \bigcap_{n=0}^{\infty} \mathcal{F}_n$

The Tail σ-algebra can be thought of as the σ-algebra consisting of events pertaining to the limiting behaviour of $\{f_n\}_{n=0}^{\infty}$.
For example, if $\forall n \in \mathbb{N}, f_n \to \mathbb{R}$, for $B \subset \mathbb{B}\left(\mathbb{R}\right)$, the set

$$\{\omega \in \Omega : f_n(\omega) \in B \text{ infinitely often}\} = \bigcap_{n=0}^{\infty} \bigcup_{n}^{\infty} \{\omega \in \Omega : f_n(\omega) \in B\}$$

relates to the limiting behaviour of the f_ns and so is in the Tail σ-algebra. On the other hand, the set

$$\{\omega \in \Omega : f_{37} \in B\}$$

does not depend on the limiting behaviour of the f_ns so is not in the Tail σ-algebra.

The statement of Kolmogorov's 0-1 law is that for any set in the Tail σ-algebra, this set either has probability 0 or probability 1. This has very general assumptions on the random variables $\{f_n\}_{n=0}^{\infty}$, namely that they are mutually independent, and no assumptions on the underlying probability space $(\Omega, \mathcal{F}, \mathcal{P})$, meaning it is very widely applicable and arises in many contexts.

2 Proof Outline

We construct our proof as follows:

1. Show that for all n, the σ-algebra $\mathcal{F}_0^n$ is independent with the σ-algebra $\mathcal{F}_{n+1}$ w.r.t. P. Independence in this context means all sets in the first σ-algebra are independent to all sets in the second σ-algebra.
2. Show that the Tail σ-algebra, τ, is a subset of σ-algebra $\mathcal{F}_{n+1}$. Use this and step 1 to show the σ-algebra $\mathcal{F}_0^n$ is independent to τ w.r.t. P.
3. Show that the Tail σ-algebra is independent to the collection of sets $\bigcup_{n=0}^{\infty} \mathcal{F}_0^n$. Independence here means each set in τ is independent to each set in the collection.
4. Prove if a σ-algebra, m, is independent to a collection of measurable sets, col, then m is also independent to the σ-algebra generated by col
5. Show the Tail σ-algebra is independent to itself w.r.t. P using step 4, and show this implies each set in the Tail has probability either 1 or 0.

We won't follow this outline strictly as I've realised its easier to build up the preliminary components of a proof initially and then put this all together at the end. Having these preliminary components also allows them to be used individually in other contexts so in general its better practice.

Since none of these σ-algebras are defined as of yet in the Mathlib library, we will have to define them in this project and show that they obey all the σ-algebra conditions.

3 Definitions

To begin with, we'll define each of our σ-algebras. For each of the definitions, fn is the sequence of functions we are considering, β maps from $j \in \mathbb{N}$ to the codomain of the function fn j, and m maps from j to a σ-algebra on the type β j.

In Lean, when creating a term of type `measurable_space α`, we need to provide additional terms which show our definition satisfies the structure conditions, in this case these conditions are:

1. Contains the empty set
2. Closed under taking complements of measurable sets
3. Closed under taking countable unions of measurable sets

We start by defining starting with $\mathcal{F}_n^k$:

```
(Fnk_algebra β fn m n k).measurable_set' = λ (s : set α), ∀ (i ∈ finset.range (k + 1)),
  ∃(s' : set (β (n + i))), (m (n + i)).measurable_set' s' ∧ fn (n + i) ⁻¹' s' = s,
```

We use the name `Fnk_algebra` instead of F_n^k as we'll be writing this out a lot so its easier not bother with the sub/superscripting.

`Fnk_algebra.measurable_set'` maps from sets of α to Prop so can be considered as a collection of sets of α. Sets are in this collection if for all $n \leq i \leq n+k$, there exists a β i measurable set s' satisfying that the image (fn j) s = s'. Clearly if s is in the collection, its complement will be in the collection too as for each i, (fn i) $s^c = s'^c \in \beta$ i. As such the collection is closed under taking complements and by a similar logic its closed under taking countable union and contains the empty set. Therefore `Fnk_algebra` satisfies the σ-algebra conditions and so we can conclude our definition.

We now define $\mathcal{F}_n$ by:

```
def Fn_σ_algebra (β : ℕ → Type) (m : Π (n : ℕ), measurable_space (β n))
  (fn : Π (n: ℕ), α → β n) (n : ℕ) : measurable_space α :=
  measurable_space.generate_from
    {s : set α | ∃ (i : ℕ),  (Fnk_algebra β fn m n i).measurable_set' s}
```

In this case, measurable_space.generate_from constructs a σ-algebra for use inductively so we don't need to check the σ-algebra conditions.

For the Tail σ-algebra, the method measurable_set' which determines whether or not a set is in the algebra is defined as follows:

```
(tail_σ_algebra β fn m).measurable_set' =
    λ s, ∀ (n : ℕ), (Fn_σ_algebra β m fn n).measurable_set' s
```

Since it is an intersection of σ-algebras, it will contain the empty set and will be closed under countable unions and complements so will also be a σ-algebra.

4 Independence of $\mathcal{F}_0^n$ with $\mathcal{F}_n$

To prove this part, we well first show $\mathcal{F}_0^n$ is independent to $\bigcup_{i=0}^{\infty} \mathcal{F}_{n+1}^{n+i}$, and then use that the independence is preserved under generating a σ-algebra from a collection of sets.

We introduce arbitrary sets s_1, s_2 in $\mathcal{F}_0^n$ and $\bigcup_{i=0}^{\infty} \mathcal{F}_{n+1}^{n+1+i}$ respectively, a step which will be ubiquitous in all our proofs of the independence of 2 objects. We then use Lean's `choose` tactic to find a specific j satisfying $s_2 \in \mathcal{F}_n^{n+j}$ and use this to find a β $(n+j)$ measurable set s_2' with fn $(n+1+j)^{-1} s_2' = s_2$. Next we do the same for s_1, this time finding a β 0 measurable set s_1' for which (fn $0)^{-1} = s_1$. After showing the sets s_1, s_2 are in the comap σ-algebras generated by fn 0 and fn $(n+1+j)$, using that the functions fn 0 and fn $n+1+j$ are independent, we can conclude that the measure of the intersection of s_1 and s_2 is the products of the measures.

This proof was straightforward but was a good introduction to proving lemmas about independent objects. Initially, I relied on unfolding definitions to make progress but after developing a good mental model of what each term means, this became unnecessary as I was familiar enough to make progress directly.

5 Independence preserved by taking the generated σ-algebra

This section was the hardest part of the proof and I have so far been unable to finish it. We start by introducing arbitrary sets s_1, s_2 in the starting σ-algebra and the generated σ-algebra respectively. Since the generated σ-algebra is defined inductively, we can now employ the `cases` tactic to break down the term hs_1 into multiple different conditions.

5.1 s_2 belongs to the initial collection

This first case is trivial since by assumption we have that sets in the collection are independent to the σ-algebra. As such this case is solved in one line.

5.2 $s_2 = \emptyset$

For this case, we can ignore the fact that s_2 is generated by the collection and instead rely on the fact that the empty set is independent to all sets in the power set over α. Here we also use the `congr'` tactic to recursively break down and show the equivalence between the following

```
have : (λ (a : α), false) = (∅ : set α), by congr',
```

5.3 s_2 is the complement of a set which is independent to the algebra

This is the first stage where I got stuck. I needed a term which states that the hs_2_s was independent to the set s_1, but due to my unfamiliarity of using inductive constructions I was unable to do this. I think this reflects how since all of my projects for this course have not regarded any inductive types. I think I would readily be able to solve this given more time as the problem stems more from my limited knowledge than the problem being exceptionally hard.

As an interesting sidenote, this is the only stage in the proof where its necessary for the sets in the collection to be measurable, as the equality

```
μ (s₁ ∩ hs₂_sᶜ) = μ s₁ * μ hs₂_sᶜ
```

requires `hs₂_s` to be measurable. If this condition is dropped, then the sum of the measures of hs_2_2 and its complement can be greater than one as the measure of a non-measurable set is taken to be outer measure of the set so we lose sigma additivity but preserve sigma sub-additivity.

5.4 s_2 is the countable union of sets independent to s_1

Similar to the above, this was also out of reach given my current knowledge of inductive types.

6 Measurable functions implies measurable $\bigcup_{i=0}^{\infty} \mathcal{F}_{n+1}^{n+1+i}$

At a later point, we will need that the collection of sets $\bigcup_{i=0}^{\infty} \mathcal{F}_{n+1}^{n+1+i}$ is a subset of the starting σ-algebra $\mathcal{F}$, I.e. that each set in the collection is measurable. This lemma will provide this given the assumption that the functions $\{f_n\}_{n=0}^{\infty}$ are measurable.

As usual, we'll introduce an arbitrary set, s, in $\bigcup_{i=0}^{\infty} \mathcal{F}_{n+1}^{n+1+i}$ and choose a k s.t $s \in \mathcal{F}_{n+1}^{n+1+k}$. We will then specialize this statement to find a β $(n+1)$ measurable set s' with a proof that $(\text{fn}\ (n+1))(s) = s'$. We can now use the measurability of $(\text{fn}\ n+1)$ using s and s' to conclude that s must be measurable w.r.t. the σ-algebra $\mathcal{F}$.

This was another case where the choice of k didnt really matter as we will always use the $n+1^{th}$ function as it will always be in $s \in \mathcal{F}_{n+1}^{n+1+k}$. For this reason, there's probably a way to golf away a portion of this proof. However, it works in its current form and having the minimal length proofs isn't an aim of this project.

One difficulty I encountered was resolving $k + 0 = k$ in one of my hypotheses. Despite trying the `rw`, `simp_rw` and `simp` tactics, neither could rewrite as I wanted. I was particularly surprised `simp_rw` wouldn't work as this is able to unpack binders and then execute the desired rewrite, indicating the problem was more complex than binders. I thought the problem was the term `l 0 this` had type `set (β (k + 0))` and the tactics were unable to realise this is the same type as `set (β k)`. However, after investigating I came to learn the types `set (β (k + 0))` and `set (β k)` are definitionally equivalent and their equality can be proved by `refl`, leaving me even more confused. In the end, rather than converting the $k+0$ to k using multiple have commands, I decided to instead leave the $k+0$ as they were and specialize my measurable of the fns to fn $(k+0)$. This was shorter, more direct and easier.

7 Self Independent σ-algebras result

In the proof, we'll rely on the fact that if a set belongs to a σ-algebra which is independent with itself, then the set must have probability either 1 or 0. The proof of this is straightforward:

$$\mathcal{P}(s) = \mathcal{P}(s \cap s) = \mathcal{P}(s) * \mathcal{P}(s)$$

$$\implies \mathcal{P}(s) = 0 \vee \mathcal{P}(s) = 1$$

In Lean, measures are defined to map to the type `ℝ≥0∞` which is the standard non-negative reals plus $+\infty$. This type is not a ring so we won't be able to rely on the ring tactic which makes things much more difficult. For example, in the step:

$$\mathcal{P}(s) = \mathcal{P}(s) * \mathcal{P}(s)$$

$$\implies \mathcal{P}(s) - \mathcal{P}(s) * \mathcal{P}(s) = 0$$

We first have to show that $\mathcal{P}(s) - \mathcal{P}(s) = 0$, for which we need to show that $\mathcal{P}(s) \neq T$, which in turn we show by using that $\mathcal{P}(s) \leq 1 < T$ and the lemma `lt_top_iff_ne_top`. This example just goes to show how awkward using the ennreal type can be.

A second difficult step I faced was proving

$$\mathcal{P}(s) * (1 - \mathcal{P}(s)) = \mathcal{P}(s) - \mathcal{P}(s) * \mathcal{P}(s)$$

The result

```
lemma mul_sub (h : 0 < c → c < b → a ≠ ∞) : a * (b - c) = a * b - a * c
```

provides a term of the desired type, however the assumptions of this lemma are too strict, with c being in the open interval $(0, b)$. Looking back, I now realise I could have done cases on $\mathcal{P}(s) = 0$, $\mathcal{P}(s) \in (0, 1)$, and finally $\mathcal{P}(s) = 1$. Clearly on the first and third case, the full result would follow trivially, and we could then apply the `mul_sub` lemma on the second case.

8 Final result

This final theorem follows directly from the proof outline of earlier and acts to piece together all the work we've done up until now. Normally in Lean its good practice to work backwards from the desired result as this tends to reduce the complexity of the proof. However, since we have all the component lemmas we need, I decided it would be just as easy to prepare each part using `have` statements and work forward. Apart from this, this section presented no notable difficulties because we have already compartmentalised the individual difficult parts of the proof.

9 Remarks

The most difficult part of this third project was choosing a suitable result to formalise which was approximately the correct length and difficulty. I began working on 3 separate projects before realising they were either too difficult or there wasn't enough preliminary material in Mathlib to be able to complete them. I think this is partially caused by the varying levels of development of Mathlib across different subjects. For example, convergence in probability was only recently formalised into the library which is only a 2nd year notion, and convergence in distribution still has yet to be defined. Meanwhile, other subjects have been formalised to a degree where its possible to do research level mathematics in lean. This seems to be reflective of different subjects being less or more partial to being formalised in lean, and also perhaps indicates formalising is more attractive to mathematicians working in some areas than others. Regardless of the reason, it took around 10 hours of working on researching other projects before I finally came to this one.

One thing which I would change in this project is the number of explicit arguments I used in my definitions. This made using each definition unwieldy and they should be replaced by implicit arguments using curly brackets instead. For example I would change `Fnk_algebra` to

```
def Fnk_algebra {β : ℕ → Type} (fn : Π (n: ℕ), α → β n)
  {m : Π (n : ℕ), measurable_space (β n)} (n k : ℕ) : measurable_space α
```

This would ensure we only have to pass the relevant information to our definitions and would make the project more coherent and easier to navigate.

Overall, I'm happy with the final state of my project. Given another 5 hours to work on it, I would be able to enact the changes I've pointed out in this document at which point the project would be complete without sorrys. Despite this, I'm happy and surprised how much I was able to do without needing to assume many lemmas to be true. Additionally, I have found that through formalising this result, I now understand the proof more so than when I learned it as many of the lemmas the proof relied on were not proved.