The Laws of Large Numbers

Manuel Eberl

Abstract

The Law of Large Numbers states that, informally, if one performs a random experiment X many times and takes the average of the results, that average will be very close to the expected value $E[X]$.

More formally, let $(X_i)_{i\in\mathbb{N}}$ be a sequence of independently identically distributed random variables whose expected value $E[X_1]$ exists. Denote the running average of $X_1, \ldots, X_n$ for $\overline{X}_n$. Then:

- The Weak Law of Large Numbers states that $\overline{X}_n \longrightarrow E[X_1]$ in probability for $n \to \infty$, i.e. $\mathcal{P}(|\overline{X}_n - E[X_1]| > \varepsilon) \longrightarrow 0$ for $n \to \infty$ for any $\varepsilon > 0$.
- The Strong Law of Large Numbers states that $\overline{X}_n \longrightarrow E[X_1]$ almost surely for $n \to \infty$, i.e. $\mathcal{P}(\overline{X}_n \longrightarrow E[X_1]) = 1$.

In this entry, I formally prove the strong law and from it the weak law. The approach used for the proof of the strong law is a particularly quick and slick one based on ergodic theory, which was formalised by Gouëzel in another AFP entry.

Contents

1 The Laws of Large Numbers

theory *Laws-of-Large-Numbers*
imports *Ergodic-Theory.Shift-Operator*
begin

We prove the strong law of large numbers in the following form: Let $(X_i)_{i\in\mathbb{N}}$ be a sequence of i.i.d. random variables over a probability space M. Further assume that the expected value $E[X_0]$ of X_0 exists. Then the sequence of random variables

$$\overline{X}_n = \frac{1}{n}\sum_{i=0}^{n} X_i$$

of running averages almost surely converges to $E[X_0]$. This means that

$$\mathcal{P}[\overline{X}_n \longrightarrow E[X_0]] = 1 \ .$$

We start with the strong law.

1.1 The strong law

The proof uses Birkhoff's Theorem from Gouëzel's formalisation of ergodic theory [1] and the fact that the shift operator $T(x_1, x_2, x_3, \ldots) = (x_2, x_3, \ldots)$ is ergodic. This proof can be found in various textbooks on probability theory/ergodic theory, e.g. the ones by Krengel [2, p. 24] and Simmonet [3, Chapter 15, pp. 311–325].

theorem (**in** *prob-space*) *strong-law-of-large-numbers-iid*:
  fixes X :: *nat* ⇒ *'a* ⇒ *real*
  assumes *indep*: *indep-vars* (λ-. *borel*) *X UNIV*
  assumes *distr*: ⋀*i*. *distr M borel* (*X i*) = *distr M borel* (*X 0*)
  assumes *L1*: *integrable M* (*X 0*)
  shows *AE x in M*. (λ*n*. (∑ *i*<*n*. *X i x*) / *n*) ⟶ *expectation* (*X 0*)
proof −

We adopt a more explicit view of M as a countably infinite product of i.i.d. random variables, indexed by the natural numbers:

  define *M'* :: (*nat* ⇒ *real*) *measure* **where** *M'* = Pi_M *UNIV* (λ*i*. *distr M borel* (*X i*))
  have [*measurable*]: *random-variable borel* (*X i*) **for** *i*
    using *indep* **by** (*auto simp*: *indep-vars-def*)
  have *M'-eq*: *M'* = *distr M* (Pi_M *UNIV* (λ*i*. *borel*)) (λ*x*. λ*i*∈*UNIV*. *X i x*)
    using *indep* **unfolding** *M'-def* **by** (*subst* (*asm*) *indep-vars-iff-distr-eq-PiM*) *auto*
  have *space-M'*: *space M'* = *UNIV*
    by (*simp add*: *M'-def space-PiM*)
  have *sets-M'* [*measurable-cong*]: *sets M'* = *sets* (Pi_M *UNIV* (λ*i*. *borel*))
    by (*simp add*: *M'-eq*)
  interpret *M'*: *prob-space M'*
    unfolding *M'-eq* **by** (*intro prob-space-distr*) *auto*

We introduce a shift operator that forgets the first variable in the sequence.

  define *T* :: (*nat* ⇒ *real*) ⇒ (*nat* ⇒ *real*) **where**
    T = (λ*f*. *f* ∘ *Suc*)
  have *funpow-T*: (*T* ^^ *i*) = (λ*f*. *f* ∘ (λ*n*. *n* + *i*)) **for** *i*
    by (*induction i*) (*auto simp*: *T-def*)

  interpret *T*: *shift-operator-ergodic distr M borel* (*X 0*) *T M'*
  proof −
    interpret *X0*: *prob-space distr M borel* (*X 0*)
      by (*rule prob-space-distr*) *auto*

```
    show shift-operator-ergodic (distr M borel (X 0))
      by unfold-locales
    show M′ ≡ Pi_M UNIV (λ-. distr M borel (X 0))
      unfolding M′-def by (subst distr)
  qed (simp-all add: T-def)

  have [intro]: integrable M′ (λf. f 0)
    unfolding M′-eq by (subst integrable-distr-eq) (use L1 in auto)
  have AE f in M′. (λn. T.birkhoff-sum (λf. f 0) n f / real n)
          ⟶ real-cond-exp M′ T.Invariants (λf. f 0) f
    by (rule T.birkhoff-theorem-AE-nonergodic) auto
  moreover have AE x in M′. real-cond-exp M′ T.Invariants (λf. f 0) x =
                  M′.expectation (λf. f 0) / M′.prob (space M′)
    by (intro T.Invariants-cond-exp-is-integral-fmpt) auto
  ultimately have AE f in M′. (λn. T.birkhoff-sum (λf. f 0) n f / real n)
                  ⟶ M′.expectation (λf. f 0)
    by eventually-elim (simp-all add: M′.prob-space)
  also have M′.expectation (λf. f 0) = expectation (X 0)
    unfolding M′-eq by (subst integral-distr) simp-all
  also have T.birkhoff-sum (λf. f 0) = (λn f. sum f {..<n})
    by (intro ext) (simp-all add:T.birkhoff-sum-def funpow-T)
  finally show ?thesis
    unfolding M′-eq by (subst (asm) AE-distr-iff) simp-all
qed
```

1.2 The weak law

To go from the strong law to the weak one, we need the fact that almost sure convergence implies convergence in probability. We prove this for sequences of random variables here.

```
lemma (in prob-space) AE-convergence-imp-convergence-in-prob:
  assumes [measurable]: ⋀i. random-variable borel (X i) random-variable borel Y
  assumes AE: AE x in M. (λi. X i x) ⟶ Y x
  assumes ε > (0 :: real)
  shows   (λi. prob {x∈space M. |X i x − Y x| > ε}) ⟶ 0
proof −
  define A where A = (λi. {x∈space M. |X i x − Y x| > ε})
  define B where B = (λn. (⋃i∈{n..}. A i))
  have [measurable]: A i ∈ sets M  B i ∈ sets M for i
    unfolding A-def B-def by measurable

  have AE x in M. x ∉ (⋂i. B i)
    using AE unfolding B-def A-def
    by eventually-elim
     (use ‹ε > 0› in ‹fastforce simp: tendsto-iff dist-norm eventually-at-top-linorder›)
  hence (⋂i. B i) ∈ null-sets M
    by (subst AE-iff-null-sets) auto

  show (λi. prob (A i)) ⟶ 0
```

```
  proof (rule Lim-null-comparison)
    have (λi. prob (B i)) ⟶ prob (⋂ i. B i)
    proof (rule finite-Lim-measure-decseq)
      show decseq B
        by (rule decseq-SucI) (force simp: B-def)
    qed auto
    also have prob (⋂ i. B i) = 0
      using ‹(⋂ i. B i) ∈ null-sets M› by (simp add: measure-eq-0-null-sets)
    finally show (λi. prob (B i)) ⟶ 0 .
  next
    have prob (A n) ≤ prob (B n) for n
      unfolding B-def by (intro finite-measure-mono) auto
    thus ∀F n in at-top. norm (prob (A n)) ≤ prob (B n)
      by (intro always-eventually) auto
  qed
qed
```

The weak law is now a simple corollary: we again have the same setting as before. The weak law now states that $\overline{X}_n$ converges to $E[X_0]$ in probability. This means that for any $\varepsilon > 0$, the probability that $|\overline{X}_n - X_0| > \varepsilon$ vanishes as $n \to \infty$.

```
corollary (in prob-space) weak-law-of-large-numbers-iid:
  fixes X :: nat ⇒ 'a ⇒ real and ε :: real
  assumes indep: indep-vars (λ-. borel) X UNIV
  assumes distr: ⋀i. distr M borel (X i) = distr M borel (X 0)
  assumes L1:    integrable M (X 0)
  assumes ε > 0
  shows   (λn. prob {x∈space M. |(∑ i<n. X i x) / n − expectation (X 0)| > ε})
⟶ 0
proof (rule AE-convergence-imp-convergence-in-prob)
  show AE x in M. (λn. (∑ i<n. X i x) / n) ⟶ expectation (X 0)
    by (rule strong-law-of-large-numbers-iid) fact+
next
  have [measurable]: random-variable borel (X i) for i
    using indep by (auto simp: indep-vars-def)
  show random-variable borel (λx. (∑ i<n. X i x) / real n) for n
    by measurable
qed (use ‹ε > 0› in simp-all)

end
```

1.3 Example

```
theory Laws-of-Large-Numbers-Example
  imports Laws-of-Large-Numbers
begin
```

As an example, we apply the strong law to the proportion of successes in an independent sequence of coin flips with success probability p. We will show that proportion of successful coin flips among the first n attempts almost surely converges to p as $n \to \infty$.

lemma (**in** *prob-space*) *indep-vars-iff-distr-eq-PiM'*:
 fixes I :: *'i set* **and** X :: $'i \Rightarrow 'a \Rightarrow 'b$
 assumes $I \neq \{\}$
 assumes *rv*: $\bigwedge i.\ i \in I \Longrightarrow$ *random-variable* $(M'\ i)\ (X\ i)$
 shows *indep-vars* $M'\ X\ I \longleftrightarrow$
  distr $M\ (\Pi_M\ i{\in}I.\ M'\ i)\ (\lambda x.\ \lambda i{\in}I.\ X\ i\ x) = (\Pi_M\ i{\in}I.$ *distr* $M\ (M'\ i)$ $(X\ i))$
proof −
 from *assms* **obtain** j **where** j: $j \in I$
  by *auto*
 define N' **where** $N' = (\lambda i.$ *if* $i \in I$ *then* $M'\ i$ *else* $M'\ j)$
 define Y **where** $Y = (\lambda i.$ *if* $i \in I$ *then* $X\ i$ *else* $X\ j)$
 have *rv*: *random-variable* $(N'\ i)\ (Y\ i)$ **for** i
  using j **by** (*auto simp*: *N'-def Y-def intro*: *assms*)

 have *indep-vars* $M'\ X\ I =$ *indep-vars* $N'\ Y\ I$
  by (*intro indep-vars-cong*) (*auto simp*: *N'-def Y-def*)
 also have ... $\longleftrightarrow$ *distr* $M\ (\Pi_M\ i{\in}I.\ N'\ i)\ (\lambda x.\ \lambda i{\in}I.\ Y\ i\ x) = (\Pi_M\ i{\in}I.$ *distr* $M\ (N'\ i)\ (Y\ i))$
  by (*intro indep-vars-iff-distr-eq-PiM rv assms*)
 also have $(\Pi_M\ i{\in}I.\ N'\ i) = (\Pi_M\ i{\in}I.\ M'\ i)$
  by (*intro PiM-cong*) (*simp-all add*: *N'-def*)
 also have $(\lambda x.\ \lambda i{\in}I.\ Y\ i\ x) = (\lambda x.\ \lambda i{\in}I.\ X\ i\ x)$
  by (*simp-all add*: *Y-def fun-eq-iff*)
 also have $(\Pi_M\ i{\in}I.$ *distr* $M\ (N'\ i)\ (Y\ i)) = (\Pi_M\ i{\in}I.$ *distr* $M\ (M'\ i)\ (X\ i))$
  by (*intro PiM-cong distr-cong*) (*simp-all add*: *N'-def Y-def*)
 finally show *?thesis* **.**
qed

lemma *indep-vars-PiM-components*:
 assumes $\bigwedge i.\ i \in A \Longrightarrow$ *prob-space* $(M\ i)$
 shows *prob-space.indep-vars* (*PiM* $A\ M$) $M\ (\lambda i\ f.\ f\ i)\ A$
proof (*cases* $A = \{\}$)
 case *False*
 have *distr* $(Pi_M\ A\ M)\ (Pi_M\ A\ M)\ (\lambda x.$ *restrict* $x\ A) =$ *distr* $(Pi_M\ A\ M)\ (Pi_M\ A\ M)\ (\lambda x.\ x)$
  by (*intro distr-cong*) (*auto simp*: *restrict-def space-PiM PiE-def extensional-def Pi-def*)
 also have ... $= Pi_M\ A\ M$
  by *simp*
 also have ... $= Pi_M\ A\ (\lambda i.$ *distr* $(Pi_M\ A\ M)\ (M\ i)\ (\lambda f.\ f\ i))$
  by (*intro PiM-cong refl*, *subst distr-PiM-component*) (*auto simp*: *assms*)
 finally show *?thesis*
  by (*subst prob-space.indep-vars-iff-distr-eq-PiM'*) (*simp-all add*: *prob-space-PiM assms False*)
next

case *True*
interpret *prob-space PiM A M*
by (*intro prob-space-PiM assms*)
show *?thesis*
unfolding *indep-vars-def indep-sets-def* **by** (*auto simp*: *True*)
qed

lemma *indep-vars-PiM-components'*:
assumes $\bigwedge i.\ i \in A \Longrightarrow$ *prob-space* (*M i*)
assumes $\bigwedge i.\ i \in A \Longrightarrow g\ i \in M\ i \rightarrow_M N\ i$
shows *prob-space.indep-vars* (*PiM A M*) *N* ($\lambda i\ f.\ g\ i\ (f\ i)$) *A*
by (*rule prob-space.indep-vars-compose2*[*OF prob-space-PiM indep-vars-PiM-components*])
(*use assms* **in** *simp-all*)

lemma *integrable-bernoulli-pmf* [*intro*]:
fixes *f* :: *bool* ⇒ *'a* :: {*banach*, *second-countable-topology*}
shows *integrable* (*bernoulli-pmf p*) *f*
by (*rule integrable-measure-pmf-finite*) *auto*

lemma *expectation-bernoulli-pmf*:
fixes *f* :: *bool* ⇒ *'a* :: {*banach*, *second-countable-topology*}
assumes *p*: $p \in \{0..1\}$
shows *measure-pmf.expectation* (*bernoulli-pmf p*) $f = p *_R f$ *True* + $(1 - p)$
$*_R f$ *False*
using *p* **by** (*subst integral-measure-pmf*[*of UNIV*]) (*auto simp*: *UNIV-bool*)

experiment
fixes *p* :: *real*
assumes *p*: $p \in \{0..1\}$
begin

definition *M* :: (*nat* ⇒ *bool*) *measure*
where $M = (\Pi_M\ i{\in}(UNIV :: nat\ set).$ *measure-pmf* (*bernoulli-pmf p*))

definition *X* :: *nat* ⇒ (*nat* ⇒ *bool*) ⇒ *real*
where *X* = ($\lambda i\ f.$ *if f i then 1 else 0*)

interpretation *prob-space M*
unfolding *M-def* **by** (*intro prob-space-PiM measure-pmf.prob-space-axioms*)

lemma *random-variable-component*: *random-variable* (*count-space UNIV*) ($\lambda f.\ f$
i)
unfolding *X-def M-def* **by** *measurable*

lemma *random-variable-X* [*measurable*]: *random-variable borel* (*X i*)
unfolding *X-def M-def* **by** *measurable*

lemma *distr-M-component*: *distr M* (*count-space UNIV*) ($\lambda f.\ f\ i$) = *measure-pmf* (*bernoulli-pmf p*)
proof –
have *distr M* (*count-space UNIV*) ($\lambda f.\ f\ i$) = *distr M* (*measure-pmf* (*bernoulli-pmf p*)) ($\lambda f.\ f\ i$)
by (*rule distr-cong*) *auto*
also have ... = *measure-pmf* (*bernoulli-pmf p*)
unfolding *M-def* **by** (*subst distr-PiM-component*) (*simp-all add*: *measure-pmf.prob-space-axioms*)
finally show *?thesis* **.**
qed

lemma *distr-M-X*:
distr M borel (*X i*) = *distr* (*measure-pmf* (*bernoulli-pmf p*)) *borel* ($\lambda b.$ *if b then 1 else 0*)
proof –
have *distr M borel* (*X i*) = *distr* (*distr M* (*count-space UNIV*) ($\lambda f.\ f\ i$))
borel ($\lambda b.$ *if b then 1 else 0* :: *real*)
by (*subst distr-distr*) (*auto simp*: *M-def X-def o-def*)
also note *distr-M-component*[*of i*]
finally show *?thesis*
by *simp*
qed

lemma *X-has-expectation*: *integrable M* (*X 0*)
proof –
have *integrable* (*bernoulli-pmf p*) ($\lambda b.$ *if b then 1 else 0* :: *real*)
by *auto*
also have *measure-pmf* (*bernoulli-pmf p*) = *distr M* (*count-space UNIV*) ($\lambda f.\ f$ *0*)
by (*simp add*: *distr-M-component*)
also have *integrable* ... ($\lambda b.$ *if b then 1 else 0* :: *real*) = *integrable M* (*X 0*)
unfolding *X-def* **using** *random-variable-component* **by** (*subst integrable-distr-eq*) *auto*
finally show *?thesis* **.**
qed

lemma *indep*: *indep-vars* (λ-. *borel*) *X UNIV*
unfolding *M-def X-def*
by (*rule indep-vars-PiM-components'*) (*simp-all add*: *measure-pmf.prob-space-axioms*)

lemma *expectation-X*: *expectation* (*X i*) = *p*
proof –
have *expectation* (*X i*) =
lebesgue-integral (*distr M* (*count-space UNIV*) ($\lambda f.\ f\ i$)) ($\lambda b.$ *if b then 1 else 0* :: *real*)
by (*subst integral-distr*) (*simp-all add*: *random-variable-component X-def*)
also have *distr M* (*count-space UNIV*) ($\lambda x.\ x\ i$) = *measure-pmf* (*bernoulli-pmf p*)

by (*rule distr-M-component*)
also have *measure-pmf.expectation* (*bernoulli-pmf p*) (λb. *if b then 1 else 0* :: *real*) = *p*
using *p* **by** (*subst integral-bernoulli-pmf*) *auto*
finally show *?thesis* **.**
qed

theorem *AE f in M*. (λn. *card* {*i*. $i < n \wedge f\ i$} / *n*) $\longrightarrow$ *p*
proof −
have *AE f in M*. (λn. ($\sum i{<}n$. *X i f*) / *real n*) $\longrightarrow$ *expectation* (*X 0*)
by (*rule strong-law-of-large-numbers-iid*)
(*use indep X-has-expectation* **in** ‹*simp-all add*: *distr-M-X*›)
also have *expectation* (*X 0*) = *p*
by (*simp add*: *expectation-X*)
also have ($\lambda x\ n$. $\sum i{<}n$. *X i x*) = ($\lambda x\ n$. $\sum i{\in}\{i{\in}\{..{<}n\}$. *x i*}. *1*)
by (*intro ext sum.mono-neutral-cong-right*) (*auto simp*: *X-def*)
also have . . . = ($\lambda x\ n$. *real* (*card* {*i*. $i < n \wedge x\ i$}))
by *simp*
finally show *?thesis* **.**
qed

end

end

Acknowledgements. I thank Sébastien Gouëzel for providing advice and context about the law of large numbers and ergodic theory. I do not actually know any ergodic theory and without him, I would probably have shied away from formalising this.

References

[1] Sébastien Gouëzel. Ergodic theory. *Archive of Formal Proofs*, December 2015. ISSN 2150-914x. https://isa-afp.org/entries/Ergodic_Theory.html, Formal proof development.

[2] Ulrich Krengel. *Ergodic Theorems*. De Gruyter, January 1985. doi:10.1515/9783110844641.

[3] Michel Simonnet. *Measures and Probabilities*. Springer New York, New York, NY, 1996. ISBN 978-1-4612-4012-9. doi:10.1007/978-1-4612-4012-9_15.