Stochastic Matrices and the Perron–Frobenius Theorem*

René Thiemann

May 4, 2021

Abstract

Stochastic matrices are a convenient way to model discrete-time and finite state Markov chains. The Perron–Frobenius theorem tells us something about the existence and uniqueness of non-negative eigenvectors of a stochastic matrix.

In this entry, we formalize stochastic matrices, link the formalization to the existing AFP-entry on Markov chains, and apply the Perron–Frobenius theorem to prove that stationary distributions always exist, and they are unique if the stochastic matrix is irreducible.

Contents

1 Introduction 1
2 Stochastic Matrices 2
3 Stochastic Vectors and Probability Mass Functions 4
4 Stochastic Matrices and Markov Models 6
5 Eigenspaces 8
6 Stochastic Matrices and the Perron–Frobenius Theorem 11

1 Introduction

In their AFP entry Markov Models [2], Hölzl and Nipkow provide a framework for specifying discrete- and continuous-time Markov chains.

In the following, we instantiate their framework by formalizing right-stochastic matrices and stochastic vectors. These vectors encode probability

*Supported by FWF (Austrian Science Fund) project Y757.
mass functions over a finite set of states, whereas stochastic matrices can be utilized to model discrete-time and finite space Markov chains.

The formulation of Markov chains as matrices has the advantage that certain concepts can easily be expressed via matrices. For instance, a stationary distribution is nothing else than a non-negative real eigenvector of the transition matrix for eigenvalue 1. As a consequence, we can derive certain properties on Markov chains using results on matrices. To be more precise, we utilize the formalization of the Perron–Frobenius theorem [1] to prove that a stationary distribution always exists, and that it is unique if the transition matrix is irreducible.

2 Stochastic Matrices

We define a type for stochastic vectors and right-stochastic matrices, i.e., non-negative real vectors and matrices where the sum of each column is 1. For this type we define a matrix-vector multiplication, i.e., we show that $A \cdot v$ is a stochastic vector, if $A$ is a right-stochastic matrix and $v$ a stochastic vector.

definition non-neg-vec :: 'a :: linordered-idom $ \cdot$ 'n $\Rightarrow$ bool where
  non-neg-vec A $\equiv$ ($\forall$ i. A $\&$ i $\geq$ 0)
definition stoch-vec :: 'a :: comm-ring-1 $\cdot$ 'n $\Rightarrow$ bool where
  stoch-vec v $=$ (sum ($\lambda$ i. v $\&$ i) UNIV $=$ 1)
definition right-stoch-mat :: 'a :: comm-ring-1 $\cdot$ 'n $\cdot$ 'm $\Rightarrow$ bool where
  right-stoch-mat a $=$ ($\forall$ j. stoch-vec (column j a))
typedef 'i st-mat $=$ \{ a :: real $\cdot$ 'i $\cdot$ 'i. non-neg-mat a $\&$ right-stoch-mat a \}
morphisms st-mat Abs-st-mat
  by (rule exI[of - $\chi$ i j. if i $=$ undefined then 1 else 0],
      auto simp: non-neg-mat-def elements-mat-h-def right-stoch-mat-def stoch-vec-def column-def)

setup-lifting type-definition-st-mat

typedef 'i st-vec $=$ \{ v :: real $\cdot$ 'i. non-neg-vec v $\&$ stoch-vec v \}
morphisms st-vec Abs-st-vec
  by (rule exI[of - $\chi$ i. if i $=$ undefined then 1 else 0],
      auto simp: non-neg-vec-def stoch-vec-def)

setup-lifting type-definition-st-vec

lift-definition transition-vec-of-st-mat :: 'i :: finite st-mat $\Rightarrow$ 'i $\Rightarrow$ 'i st-vec
is \( \lambda \ a \ i \). column \( i \) \( a \)

by (auto simp: right-stoch-mat-def non-neg-mat-def stoch-vec-def
    elements-mat-h-def non-neg-vec-def column-def)

lemma non-neg-vec-st-vec: non-neg-vec (st-vec \( v \))
  by (transfer, auto)

lemma non-neg-mat-mult-non-neg-vec: non-neg-mat \( a \) \( \Rightarrow \) non-neg-vec \( v \) \( \Rightarrow \)
  non-neg-vec (\( a \) \( * \) \( v \))
  unfolding non-neg-mat-def non-neg-vec-def elements-mat-h-def
  by (auto simp: matrix-vector-mult-def intro: sum-nonneg)

lemma right-stoch-mat-mult-stoch-vec:
  assumes right-stoch-mat \( a \)
  and stoch-vec \( v \)
  shows stoch-vec (\( a \) \( * \) \( v \))
  proof
    note * = assms
    unfolding right-stoch-mat-def column-def stoch-vec-def
    have \( \text{(?sum = 1)} \)
      unfolding stock-vec-def matrix-vector-mult-def
      by auto
    also have \( ?\sum = (\sum j \in \text{UNIV}. \sum i \in \text{UNIV}. \ a \ i \ j \ v \ j) \)
      by (rule sum.cong[OF refl], insert *, auto simp: sum-distrib-right[symmetric])
    also have \( \ldots = \sum \( j \in \text{UNIV}. \ v \ j \) \)
      by (rule sum.swap)
    also have \( \ldots = 1 \) using * by auto
    finally show \( ?\text{thesis} \) by simp
  qed

lift-definition st-mat-times-st-vec :: \( 'i \rightarrow \text{finite} \rightarrow \text{st-mat} \Rightarrow 'i \rightarrow \text{st-vec} \)
  \((\text{infixl} \ *v) 70) \ is \ (\rightarrow \text{st-vec} \)
  using right-stoch-mat-mult-stoch-vec non-neg-mat-mult-non-neg-vec by auto

lift-definition to-st-vec :: \( \text{real} \rightarrow 'i \rightarrow \text{st-vec} \)
  \( \lambda \ x. \ if \ \text{stoch-vec} \ x \ \	ext{and} \ \text{non-neg-vec} \ x \ \text{then} \ x \ \text{else} \ (\chi \ i. \ if \ i = \text{undefined} \ \text{then} \ 1 \ \text{else} \ 0) \)
  by (auto simp: non-neg-vec-def stoch-vec-def)

lemma right-stoch-mat-st-mat: right-stoch-mat (st-mat \( A \))
  by transfer auto

lemma non-neg-mat-st-mat: non-neg-mat (st-mat \( A \))
  by (transfer, auto simp: non-neg-mat-def elements-mat-h-def)

lemma st-mat-mult-st-vec: st-mat \( A \) \( * \) \( v \) \( \text{st-vec} \ X = \text{st-vec} \ (A \ * st X) \) by (transfer, auto)

lemma st-vec-nonneg[simp]: st-vec \( x \ \$ \ i \geq 0 \)
  using non-neg-vec-st-vec[of \( x \)] by (auto simp: non-neg-vec-def)
3 Stochastic Vectors and Probability Mass Functions

We prove that over a finite type, stochastic vectors and probability mass functions are essentially the same thing: one can convert between both representations.

theory Stochastic-Vector-PMF
begin

lemma sigma-algebra-UNIV-finite[simp]: sigma-algebra (UNIV :: 'a :: finite set)
  UNIV
proof (unfold-locales, goal-cases)
  case (4 a b)
  show ?case by (intro exI[of - {a - b}], auto)
qed auto

definition measure-of-st-vec' :: 'a st-vec ⇒ 'a :: finite set ⇒ ennreal where
  measure-of-st-vec' x I = sum (λi. st-vec x $ i) I

lemma positive-measure-of-st-vec'[simp]: positive A (measure-of-st-vec' x)
  unfolding measure-of-st-vec'-def positive-def by auto

lemma measure-space-measure-of-st-vec': measure-space UNIV UNIV (measure-of-st-vec' x)
  unfolding measure-space-def
proof (simp, simp add: countably-additive-def measure-of-st-vec'-def disjoint-family-on-def, clarify, goal-cases)
  case (1 A)
  let ?x = st-vec x
  define N where N = {i. A i ≠ {}}
  let ?A = ∪(A ∩ N)
  have finite B ⇒ B ⊆ ?A ⇒ ∃ K. finite K ∧ K ⊆ N ∧ B ⊆ ∪(A ∩ K) for B
  proof (induct rule: finite-induct)
    case (insert b B)
    from insert(3-4) obtain K where K: finite K K ⊆ N B ⊆ ∪(A ∩ K) by auto
    from insert(4) obtain a where a: a ∈ N b ∈ A a by auto
    show ?case by (intro exI[of - insert a K], insert a K, auto)
  qed auto
  from this[OF - subset-refl] obtain K where *: finite K K ⊆ N ∪(A ∩ K) = ?A
end
by \(\text{auto}\)

\[
\begin{align*}
&\text{assume } K \subset N \\
&\text{then obtain } n \text{ where } **: n \in N \ and \ n \notin K \text{ by } \text{auto} \\
&\text{from this[unfolded N-def] obtain } a \text{ where } a \in A \ and \ n \text{ by } \text{auto} \\
&\text{with } ** \ast \text{ obtain } k \text{ where } ***: k \in K \ and \ a \in A \ and \ k \text{ by } \text{force} \\
&\text{from } ** \ast \ast \text{ have } n \neq k \text{ by } \text{auto} \\
&\text{from } 1[\text{rule-format, OF this}] \text{ have } A \cap A k = \{\} \text{ by } \text{auto} \\
&\text{with } ** \ast \ast \ast \text{ have False by } \text{auto} \\
\end{align*}
\]

\(\text{with } \ast \text{ have fin: finite } N \text{ by } \text{auto}\)

\(\text{have id: } \bigcup (A \cup \text{UNIV}) = \forall A \text{ unfolding N-def by } \text{auto}\)

\(\text{show } (\sum i. \ ennreal (\sum (\$(i) \ ?x) \ (A i))) = \ennreal (\sum (\$(i) \ ?x) \ (\bigcup (A \cup \text{UNIV}))) \text{ unfolding id}\)

\(\text{apply (subst suminf-finite[OF fin], (auto simp: N-def)[1])}\)

\(\text{apply (subst sum-ennreal, (insert non-neg-vec-st-vec[of x], auto simp: non-neg-vec-def intro!: sum-nonneg)[1])}\)

\(\text{apply (rule arg-cong[of - ennreal])}\)

\(\text{apply (subst sum.\UNION-disjoint[OF fin], insert 1, auto)}\)

\(\text{done}\)

\(\text{qed}\)

\(\text{context begin}\)

\(\text{setup-lifting type-definition-measure}\)

\(\text{lift-definition measure-of-st-vec :: 'a st-vec \Rightarrow 'a :: finite measure is}\)

\(\lambda x. \ (\text{UNIV, UNIV, measure-of-st-vec'} x)\)

\(\text{by (auto simp: measure-space-measure-of-st-vec')}\)

\(\text{lemma sets-measure-of-st-vec[simp]: sets (measure-of-st-vec x) = UNIV}\)

\(\text{unfolding sets-def by (transfer, auto)}\)

\(\text{lemma space-measure-of-st-vec[simp]: space (measure-of-st-vec x) = UNIV}\)

\(\text{unfolding space-def by (transfer, auto)}\)

\(\text{lemma emeasure-measure-of-st-vec[simp]: emeasure (measure-of-st-vec x) I = }\)

\(\text{sum (\lambda i. \ st-vec x $ \ i \ ) \ I}\)

\(\text{unfolding emeasure-def by (transfer', auto simp: measure-of-st-vec'-def)}\)

\(\text{lemma prob-space-measure-of-st-vec: prob-space (measure-of-st-vec x)}\)

\(\text{by (unfold-locales, intro exI[of - UNIV], auto, transfer, auto simp: stoch-vec-def)}\)

\(\text{end}\)

\(\text{lift-definition st-vec-of-pmf :: 'i :: finite pmf \Rightarrow 'i st-vec is}\)

\(\lambda pmF. \ vec-lambda (pmf pmF)\)

\(\text{proof (intro conjI, goal-case)}\)

\(\text{case (2 pmF)}\)

\(\text{show stoch-vec (vec-lambda (pmf pmF))}\)

\(\text{unfolding stoch-vec-def}\)
apply auto
apply (unfold measure-pmf-UNIV[of pmfF, symmetric])
by (simp add: measure-pmf-conv-infsetsum)
qed (auto simp: non-neg-vec-def stoch-vec-def)

context pmf-as-measure
begin
lift-definition pmf-of-st-vec :: 'a :: finite st-vec ⇒ 'a pmf is measure-of-st-vec
proof (goal-cases)
case (1 x)
show ?case by (auto simp: prob-space-measure-of-st-vec measure-def)
(rule AE-I[where N = {i. st-vec x $ i = 0}], auto)
qed

lemma st-vec-st-vec-of-pmf[simp]:
  st-vec (st-vec-of-pmf x) $ i = pmf x i
by (simp add: st-vec-of-pmf.rep-eq)

lemma pmf-pmf-of-st-vec[simp]: pmf (pmf-of-st-vec x) i = st-vec x $ i
by (transfer, auto simp: measure-def)

lemma st-vec-of-pmf-pmf-of-st-vec[simp]: st-vec-of-pmf (pmf-of-st-vec x) = x
proof –
  have st-vec (st-vec-of-pmf (pmf-of-st-vec x)) = st-vec x
    unfolding vec-eq-iff by auto
  thus ?thesis using st-vec-inject by blast
qed

lemma pmf-of-st-vec-inj: (pmf-of-st-vec x = pmf-of-st-vec y) = (x = y)
by (metis st-vec-of-pmf-pmf-of-st-vec)
end
end

4 Stochastic Matrices and Markov Models

We interpret stochastic matrices as Markov chain with discrete time and
finite state and prove that the bind-operation on probability mass functions
is precisely matrix-vector multiplication. As a consequence, the notion of
stationary distribution is equivalent to being an eigenvector with eigenvalue
1.

theory Stochastic-Matrix-Markov-Models
imports
  Markov-Models,Classifying-Markov-Chain-States
  Stochastic-Vector-PMF
begin

definition transition-of-st-mat :: 'i st-mat ⇒ 'i :: finite ⇒ 'i pmf where
transition-of-st-mat a i = pmf-as-measure.pmf-of-st-vec (transition-vec-of-st-mat a i)

lemma st-vec-transition-vec-of-st-mat[simp]:
  st-vec (transition-vec-of-st-mat A a) $ i = st-mat A $ i $ a
  by (transfer, auto simp: column-def)

locale transition-matrix = pmf-as-measure +
  fixes A :: 'i :: finite st-mat
begin
sublocale MC-syntax transition-of-st-mat A.

lemma measure-pmf-of-st-vec[simp]:
  measure-pmf (pmf-of-st-vec x) = measure-of-st-vec x
  by (rule pmf-as-measure.pmf-of-st-vec.rep-eq)

lemma pmf-transition-of-st-mat[simp]:
  pmf (transition-of-st-mat A a) i = st-mat A $ i $ a
  unfolding transition-of-st-mat-def
  by (transfer, auto simp: measure-def)

lemma bind-is-matrix-vector-mult: (bind-pmf x (transition-of-st-mat A)) = pmf-as-measure.pmf-of-st-vec (A *s st-vec-of-pmf x)
proof (rule pmf-eqI, goal-cases)
  case (1 i)
  define X where X = st-vec-of-pmf x
  have pmf (bind-pmf x (transition-of-st-mat A)) i =
    (∑ a∈UNIV. pmf x a *R pmf (transition-of-st-mat A a) i)
    unfolding pmf-bind by (subst integral-measure-pmf[of UNIV], auto)
  also have ... = (∑ a∈UNIV. st-mat A $ i $ a * st-vec X $ a)
    by (rule sum.cong[OF refl], auto simp: X-def)
  also have ... = (st-mat A *v st-vec X) $ i
    unfolding matrix-vector-mult-def by auto
  also have ... = st-vec (A *st X) $ i unfolding st-mat-mult-st-vec by simp
  also have ... = pmf (pmf-of-st-vec (A *st X)) i by simp
  finally show ?case by (simp add: X-def)
qed

lemmas stationary-distribution-alt-def = stationary-distribution-def[unfolded bind-is-matrix-vector-mult]

lemma stationary-distribution-implies-pmf-of-st-vec:
  assumes stationary-distribution N
  shows ∃ x. N = pmf-of-st-vec x
proof -
  from assms[unfolded stationary-distribution-alt-def] show ?thesis by auto
qed

lemma stationary-distribution-pmf-of-st-vec:
stationary-distribution \( (\text{pmf-of-st-vec } x) = (A \ast \text{st } x = x) \)

unfolding \( \text{stationary-distribution-alt-def pmf-of-st-vec-inj} \) by auto

end

end

5 Eigenspaces

Using results on Jordan-Normal forms, we prove that the geometric multiplicity of an eigenvalue (i.e., the dimension of the eigenspace) is bounded by the algebraic multiplicity of an eigenvalue (i.e., the multiplicity as root of the characteristic polynomial.). As a consequence we derive that any two eigenvectors of some eigenvalue with multiplicity 1 must be scalar multiples of each other.

theory Eigenspace

imports

  Jordan-Normal-Form, Jordan-Normal-Form-Uniqueness
  Perron-Frobenius, Perron-Frobenius-Aux

begin

hide-const (open) Coset.order

  The dimension of every generalized eigenspace is bounded by the algebraic multiplicity of an eigenvalue. Hence, in particular the geometric multiplicity is smaller than the algebraic multiplicity.

lemma dim-gen-eigenspace-order-char-poly: assumes jnf: jordan-nf \( A \) n-as
  shows dim-gen-eigenspace \( A \) lam k \leq order lam (char-poly \( A \))
  unfolding jordan-nf-order[OF jnf] dim-gen-eigenspace[OF jnf]
  by (induct n-as, auto)

Every eigenvector is contained in the eigenspace.

lemma eigenvector-mat-kernel-char-matrix: assumes \( A \in \text{carrier-mat } n \ n \) and ev: eigenvector \( A \) v lam
  shows \( v \in \text{mat-kernel } (\text{char-matrix } A \text{ lam}) \)
  using ev[unfolded eigenvector-char-matrix[OF \( A \)]] A
  unfolding mat-kernel-def by (auto simp: char-matrix-def)

If the algebraic multiplicity is one, then every two eigenvectors are scalar multiples of each other.

lemma unique-eigenvector-jnf: assumes jnf: jordan-nf \( A :: \ 'a :: \text{field mat} \) n-as and ord: order lam (char-poly \( A \)) = 1
  and ev: eigenvector \( A \) v lam eigenvector \( A \) w lam
  shows \( \exists \ a. \ v = a \cdot w \)
proof –
  let \( ?cA = \text{char-matrix } A \text{ lam} \)
  from similar-matD jnf[unfolded jordan-nf-def] obtain n where
  \( A : \text{carrier-mat } n \ n \) by auto
  from dim-gen-eigenspace-order-char-poly[OF jnf, of lam 1, unfolded ord]
  have \( \text{dim: kernel-dim } ?cA \leq 1 \)
unfolding dim-gen-eigenspace-def by auto
from eigenvector-mat-kernel-char-matrix{OF A ev(1)}
have vk: v ∈ mat-kernel ?cA .
from eigenvector-mat-kernel-char-matrix{OF A ev(2)}
have wk: w ∈ mat-kernel ?cA .
from ev[unfolded eigenvector-def] A have
v: v ∈ carrier-vec n v ≠ 0_v n and
w: w ∈ carrier-vec n w ≠ 0_v n by auto
have cA: ?cA ∈ carrier-mat n n using A
unfolding char-matrix-def by auto
interpret kernel n n ?cA
by (unfold-locales, rule cA)
from kernel-basis-exists{OF A} obtain B where B: finite B basis B by auto
from this[unfolded Ker.basis-def] have basis: mat-kernel ?cA = span B by auto
{ assume card B = 0
with B basis have bas: mat-kernel ?cA = local.span { } by auto
also have ... = {0_v n} unfolding Ker.span-def by auto
finally have False using v wk by auto
}
with Ker.dim-basis{OF B} dim have card B = Suc 0 by (cases card B, auto)
then obtain b where Bb: B = {b} by blast
from Bb[2][unfolded Bb Ker.basis-def] have bk: b ∈ mat-kernel ?cA by auto
hence b: b ∈ carrier-vec n using cA mat-kernelD(1) by blast
from Bb basis have mat-kernel ?cA = span {b} by auto
also have ... = NC.span {b}
by (rule span-same, insert bk, auto)
also have ... ⊆ { a · v b | a. True}
proof —
{
fix x
assume x ∈ NC.span {b}
from this[unfolded NC.span-def] obtain a A
where x: x = NC.lincomb a A and A: A ⊆ {b} by auto
hence A = {} ∨ A = {b} by auto
hence ∃ a. x = a · v b
proof
assume A = {} thus ?thesis unfolding x using b by (intro exI[of - 0], auto)
next
assume A = {b} thus ?thesis unfolding x using b
by (intro exI[of - a b], auto simp: NC.lincomb-def)
qed
}
thus ?thesis by auto
qed
finally obtain vv ww where wb: v = vv · v b and wb: w = ww · v b using vk wk
by force+
from \( wb \) \( w b \) have \( ww \neq 0 \) by auto
from \( \text{arg-cong}[\OF \ wb \ OF x] \) \( \inverse \ OF \ x \) \( \FOR \ wb \) \( b = \inverse \ OF \ w \)
by \( \text{(auto simp: smult-smult-assoc)} \)
from \( \v b \) \([\text{unfolded this smult-smult-assoc}]\) show \( \ ? \)thesis by auto
qed

Getting rid of the JNF-assumption for complex matrices.

**lemma unique-eigenvector-complex**: assumes \( A \in \carrier-mat \ n \ n \)
and \( \ord: \\order \ lam \ (\char-poly \ A :: \complex \ poly) = 1 \)
and \( \ev: \eigenvector \ A \ v \ lam \ eigenvector \ A \ w \ lam \)
shows \( \exists \ a. \ v = a \cdot w \)
proof –
from \( \text{jordan-nf-exists}[\OF \ A] \) \( \char-poly-factorized[\OF \ A] \)
obtain \( n-as \) where \( \jordan-nf \ A \ n-as \) by auto
from \( \text{unique-eigenvector-jnf}[\OF \ this \ ord \ ev] \) show \( \ ? \)thesis .
qed

Convert the result to real matrices via homomorphisms.

**lemma unique-eigenvector-real**: assumes \( A \in \carrier-mat \ n \ n \)
and \( \ord: \\order \ lam \ (\char-poly \ A :: \real \ poly) = 1 \)
and \( \ev: \eigenvector \ A \ v \ lam \ eigenvector \ A \ w \ lam \)
shows \( \exists \ a. \ v = a \cdot w \)
proof –
let \( ?c = \complex-of-real \)
let \( \tilde{A} = \map-mat \ ?c \ A \)
from \( \tilde{A} \) have \( cA: \tilde{A} \in \carrier-mat \ n \ n \) by auto
have \( \ord: \order \ (?c \ lam \ (\char-poly \ ?A) = 1 \)
unfolding \( \text{of-real-hom.\char-poly-hom}[\OF \ A] \)
by \( \text{(subst map-poly-inj-idom-divide-hom, order-hom, unfold-locales, rule ord)} \)
note \( evc = \text{of-real-hom.\eigenvector-hom}[\OF \ A] \)
from \( \text{unique-eigenvector-complex}[\OF \ A \ \ORD \ evc \ evc, \ OF \ ev] \) obtain \( a :: \complex \)
where \( \text{id}: \map-vec \ ?c \ v = a \cdot \map-vec \ ?c \ w \) by auto
from \( \text{ev}[\text{unfolded eigenvector-def}] \ A \) have \( carr: \ v \in \carrier-vec \ n \ w \in \carrier-vec \ n \)
\( v \neq 0, n \) by auto
then obtain \( i \) where \( i: i < n \ v \ \not\approx \ i \neq 0 \)
unfolding \( \text{Matrix.\vec-eq-iff} \) by auto
from \( \text{arg-cong}[\OF \ id, \ OF x. x \ \not\approx \ i \] \( \text{carr} \ i \)
have \( \ ?c \ (v \ \not\approx \ i) = a \cdot \ ?c \ (w \ \not\approx \ i) \)
by auto
with \( \text{i(2)} \) have \( a \in \text{Reals} \)
by \( \text{(metis Reals-cnj-iff complex-cnj-complex-of-real complex-cnj-mult mult-cancel-right mult-cnj-0-iff of-real-hom.zero-of-real-hom.injectivity)} \)
then obtain \( b \) where \( a = \ ?c \ b \) by \( \text{(rule Reals-cases)} \)
from \( \text{id}[\text{unfolded a}] \) have \( \map-vec \ ?c \ v = \map-vec \ ?c \ (b \cdot w) \) by auto
hence \( v = b \cdot w \) by \( \text{(rule of-real-hom.\vec-hom-inj)} \)
thus \( \ ? \)thesis by auto

10
Finally, the statement converted to HMA-world.

**Lemma** `unique-eigen-vector-real`: **Assumes** `ord: order lam (charpoly A :: real poly) = 1` and `ev: eigen-vector A v lam eigen-vector A w lam` shows `∃ a. v = a * s w using assms`

**Proof** (transfer, goal-cases)
- **Case** `(1 lam A v w)`
  - **Show** `?case`
    - **By** `(rule unique-eigenvector-real[OF 1(1-2,4,6)])`

qed

end

6 Stochastic Matrices and the Perron–Frobenius Theorem

Since a stationary distribution corresponds to a non-negative real eigenvector of the stochastic matrix, we can apply the Perron–Frobenius theorem. In this way we easily derive that every stochastic matrix has a stationary distribution, and moreover that this distribution is unique, if the matrix is irreducible, i.e., if the graph of the matrix is strongly connected.

**Theory** `Stochastic-Matrix-Perron-Frobenius`

**Imports**
- Perron-Frobenius, Perron-Frobenius-Irreducible
- Stochastic-Matrix-Markov-Models
- Eigenspace

begin

hide-const (open) Coset.order

**Lemma** `pf-nonneg-mat-st-mat`: **Assumes** `pf-nonneg-mat (st-mat A)` by `(unfold-locales, auto simp: non-neg-mat-st-mat)`

**Lemma** `stoch-non-neg-vec-norm1`: **Assumes** `stoch-vec (v :: real ^ 'n) non-neg-vec v` shows `norm1 v = 1`

**Unfolding** `assms(1)[unfolded stoch-vec-def, symmetric] norm1-def`

**By** `(rule sum.cong, insert assms(2)[unfolded non-neg-vec-def], auto)`

**Lemma** `stationary-distribution-exists`: `∃ v. A * st v = v`

**Proof**
- **Let** `?A = st-mat A`
- **Let** `?c = complex-of-real`
- **Let** `?B = χ i j. ?c (?A $ i $ j)`
- **Have** `real-non-neg-mat ?B using non-neg-mat-st-mat[of A]`
unfolding real-non-neg-mat-def elements-mat-h-def non-neg-mat-def
by auto
from Perron-Frobenius.perron-frobenius-both[of this] obtain v a where
ev: eigen-vector ?B v (\(?c a\)) and nn: real-non-neg-vec v
and a: a = HMA-Connect.spectral-radius ?B by auto
from spectral-radius-ev[of ?B, folded a] have a0: a \geq 0 by auto
define a where a = (\(\chi i. Re (v \cdot i)\))
from nn have vv: v = (\(\chi i. ?c (v \cdot i)\)) unfolding real-non-neg-vec-def vv
by (auto simp: vec-elements-h-def)
from ev[unfolded eigen-vector-def] have v0: v \neq 0 and ev: ?B *v = ?c a *s v
by auto
from v0 have w0: w \neq 0 unfolding vw by (auto simp: Finite-Cartesian-Product.vec-eq-iff)

let \(?n = \text{norm1 } w\)
from w0 have n0: ?n \neq 0 by auto
hence n-pos: ?n > 0 using norm1-ge-0[of w] by linarith
define a where a = inverse ?n *s w
have nn: non-neg-vec u using nn n-pos unfolding u-def non-neg-vec-def by auto
have nu: norm1 u = 1 unfolding u-def scalar-mult-eq-scaleR norm1-scaleR
using n-pos
by (auto simp: field-simps)
have 1: stoch-vec u unfolding stoch-vec-def nu[symmetric] norm1-def
by (rule sum.cong, insert nn[unfolded non-neg-vec-def], auto)
from arg-cong[OF ev, of \(\lambda x. inverse ?n *s x\)]
have ev: ?A *v u = a *s u unfolding u-def
by (auto simp: ac-simps vector-smult-distrib matrix-vector-mult-def)
from right-stoch-mat-mult-stoch-vec[OF right-stoch-mat-st-mat[of A] 1, unfolded ev]
have st: stoch-vec (a *s u)
from non-neg-mat-mult-non-neg-vec[of non-neg-mat-st-mat[of A] nn, unfolded ev]
have nn': non-neg-vec (a *s u)
from stoch-non-neg-vec-norm1[OF st nn', unfolded scalar-mult-eq-scaleR norm1-scaleR
nn] a0
have a = 1 by auto
with ev st have ev: ?A *v u = u and st: stoch-vec u by auto
show \(?thesis using ev st nn\)
by (intro exI[of - to-st-vec w], transfer, auto)

qed

lemma stationary-distribution-unique:
    assumes fixed-mat.irreducible (st-mat A)
    shows \( \exists! \ v. \ A \ast v = v \)
proof
  from stationary-distribution-exists obtain \( v \) where \( \text{ev}: A \ast v = v \) by auto
  show \( ?thesis \)
  proof (intro exII, rule \text{ev})
    fix \( w \)
    assume \( A \ast w = w \)
    thus \( w = v \) using \text{ev} \ assumptions
  proof (transfer, goal-cases)
    case (1 \( A \ w \ v \))
    interpret perron-frobenius \( A \)
      by (unfold-locales, insert 1, auto)
    from 1 have \( \ast: \text{eigen-vector} \ A \ v \ 1 \text{le-vec} \ 0 \ v \text{eigen-vector} \ A \ w \ 1 \)
    by (auto simp: \text{eigen-vector-def} \text{stoch-vec-def} \text{non-neg-vec-def})
    from nonnegative-eigenvector-has-ev-sr[of \( \ast(1-2) \)] have \( sr1: \text{sr} = 1 \) by auto
    finally have \( a = 1 \) using 1(2)[unfolded \text{stoch-vec-def}]
    by auto
  qed
  qed

Let us now convert the stationary distribution results from matrices to Markov chains.

context transition-matrix
begin

lemma stationary-distribution-exists:
  \( \exists \ x. \text{stationary-distribution} \ (\text{pmf-of-st-vec} \ x) \)
proof
  from stationary-distribution-exists obtain \( x \) where \( \text{ev}: A \ast x = x \) by auto
  show \( ?thesis \)
  by (intro exI[of - \( x \)], unfold stationary-distribution-pmf-of-st-vec, simp add: \text{ev})
qed

lemma stationary-distribution-unique: assumes fixed-mat.irreducible (st-mat A)
shows \( \exists! N. \text{stationary-distribution } N \)

proof –

from \text{stationary-distribution-exists} obtain \( x \) where

\( st: \text{stationary-distribution } (\text{pmf-of-st-vec } x) \) by blast

show \( ?\text{thesis} \)

proof (rule ex1I, rule \( st \))

fix \( N \)

assume \( st': \text{stationary-distribution } N \)

from \( \text{stationary-distribution-implies-pmf-of-st-vec} \) [OF this] obtain \( y \) where

\( N: N = \text{pmf-of-st-vec } y \) by auto

from \( st'[\text{unfolded } N] \) \( st \)

have \( A \ast st x = x A \ast st y = y \) unfolding \text{stationary-distribution-pmf-of-st-vec}

by auto

with \( N \) show \( N = \text{pmf-of-st-vec } x \) by auto

qed

qed

end

Reference
