Solving linear games with cone programs

Michael Orlitzky

Part 1, Section 1

Matrix games: Introduction

Matrix games: Introduction

A *two-person game* involves two players:

1. Alice
2. Bob

Both players make a move, and then some rule is applied to determine the winner.

The winner gets a prize.

Matrix games: Introduction

Example (rock-paper-scissors).

The players choose either *rock*, *paper*, or *scissors*.

- Rock beats scissors
- Scissors beats paper
- Paper beats rock

The loser pays the winner one dollar.

Matrix games: Introduction

The legal moves and payoffs for this game can be described by a table (Bob pays Alice):

$$
\begin{array}{cr|ccc}
 & & & \textbf{Alice} & \\
 & & \textit{rock} & \textit{paper} & \textit{scissors} \\
\hline
\textbf{Bob} & \textit{rock} & 0 & 1 & -1 \\
 & \textit{paper} & -1 & 0 & 1 \\
 & \textit{scissors} & 1 & -1 & 0
\end{array}
$$

Matrix games: Introduction

When Alice loses a dollar, we think of it instead as winning -1 dollars. The winnings of the two players therefore always sum to zero.

Any game where the winnings of all the players sum to zero is called a *zero-sum game*.

Matrix games: Introduction

These games were introduced and studied by

1. von Neumann and Morgenstern (1944)
2. Kaplansky (1945)
3. Karlin (1959)
4. Dantzig (1963)

(among others)

Matrix games: Introduction

Problem.

There is no “best” choice in rock-paper-scissors.

If you play rock every time (a *pure* strategy), I can beat you.

Matrix games: Introduction

Solution.

Allow *mixed* strategies.

A *mixed* strategy assigns probabilities to the available moves.

Matrix games: Introduction

Example.

Alice plays rock 50% of the time, and paper 50% of the time.

Bob always plays paper.

Alice's strategy is mixed. Bob's strategy is pure.

Matrix games: Introduction

The *expected* payoff (to Alice) in this case is

$$\overbrace{\frac{1}{2}\cdot 1\cdot(-1)}^{\text{rock/paper}} + \overbrace{\frac{1}{2}\cdot 1\cdot 0}^{\text{paper/paper}} = -\frac{1}{2}.$$

This is nothing but the probability of each outcome times the payoff for that outcome.

Matrix games: Introduction

Probability interpretation of mixed strategies:

1. Players assign probabilities to the moves
2. Each player chooses a move randomly
3. Afterwards, we compute the expected payoff

Matrix games: Introduction

von Neumann proved that there are always optimal mixed strategies.

“Optimal” means you would be no better off doing something else (on average).

Matrix games: Introduction

von Neumann's argument is elementary [3], based on the existence of a min/max on compact sets.

More generally, optimal strategies are guaranteed by the famous *Nash existence theorem*.

Matrix games: Introduction

Problem.

Probabilities and expectations are gross.

Solution.

Use linear algebra instead.

Matrix games: Introduction

Recall our table for rock-paper-scissors:

		Alice		
		rock	*paper*	*scissors*
Bob	*rock*	0	1	-1
	paper	-1	0	1
	scissors	1	-1	0

Matrix games: Introduction

Clearly we can represent the payoffs by a matrix,

$$L := \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}.$$

Problem.

How to find the payoff from the players' moves?

Matrix games: Introduction

Solution.

Identify the moves with standard basis vectors,

$$\begin{aligned} \text{rock} &\cong e_1 := (1,0,0)^T \\ \text{paper} &\cong e_2 := (0,1,0)^T \\ \text{scissors} &\cong e_3 := (0,0,1)^T \ldots \end{aligned}$$

Matrix games: Introduction

and use a payoff function

$$p(x, y) = y^T L x$$

that picks out rows/columns of L:

$$p(e_j, e_i) = e_i^T L e_j = L_{ij}.$$

Matrix games: Introduction

Example.

If Alice plays paper $\cong e_2$ and Bob plays rock $\cong e_1$, then

$$p\left(e_2, e_1\right) = e_1^T L e_2 = L_{12} = 1$$

is the amount that Alice wins.

Matrix games: Introduction

So that works for pure strategies.

What about mixed strategies?

Probabilities are simply nonnegative weights summing to one. Can we assign similar weights to the basis vectors in our geometric model?

Matrix games: Introduction

Suppose

$$x = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3$$

where

$$\alpha_i \geq 0 \text{ and } \sum \alpha_i = 1.$$

Then

$$x = (\alpha_1, \alpha_2, \alpha_3)^T$$

and x belongs to the *convex hull* of $\{e_1, e_2, e_3\}$.

Matrix games: Introduction

The scalars α_i are the probabilities that Alice assigns to rock, paper, and scissors, respectively.

Do the same thing for Bob:

$$y = \beta_1 e_1 + \beta_2 e_2 + \beta_3 e_3 = (\beta_1, \beta_2, \beta_3)^T \,.$$

Matrix games: Introduction

Using the bilinearity of our payoff function,

$$\begin{aligned} p(x, y) &= y^T L x \\ &= \sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_j \beta_i e_i^T L e_j \\ &= \sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_j \beta_i \cdot p(e_j, e_i) \end{aligned}$$

is precisely the expected payoff.

Matrix games: Introduction

Why? If Alice plays e_j with probability α_j and Bob plays e_i with probability β_i, then the term

$$\alpha_j \beta_i \cdot p\left(e_j, e_i\right)$$

is the payoff $p\left(e_j, e_i\right)$ for that outcome times the probability $\alpha_j \beta_i$ that it occurs.

Matrix games: Introduction

Moral: our geometric model works for mixed strategies, too.

Example.

Alice plays rock 50% of the time, and paper 50% of the time.

Bob always plays paper.

Matrix games: Introduction

We represent Alice's strategy as

$$x = \frac{1}{2}e_1 + \frac{1}{2}e_2 = \left(\frac{1}{2}, \frac{1}{2}, 0\right)^T,$$

or "halfway between rock and paper."

Matrix games: Introduction

Similarly, Bob's strategy is $y = (0, 1, 0)^T$. The associated payoff is

$$\begin{aligned} p(x, y) &= y^T L x \\ &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \end{bmatrix} = -\frac{1}{2}. \end{aligned}$$

Matrix games: Introduction

This is the same expected payoff that we computed previously.

It says that “Bob wins half the time.”

Matrix games: Introduction

Geometric interpretation of mixed strategies:

1. Both players choose a vector from the convex hull of the standard basis vectors.
2. We compute the payoff $(x, y) \mapsto y^T L x$.

It is equivalent to the probability interpretation.

Part 1, Section 2

Matrix games: Formality

Matrix games: Formality

Definition.

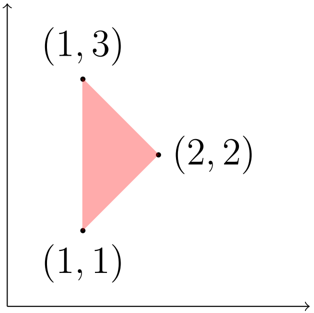
The *convex hull* of a nonempty subset X of V is

$$\operatorname{conv}(X) := \left\{ \sum_{i=1}^{m} \alpha_i x_i \;\middle|\; \begin{array}{ll} x_i \in X, & \alpha_i \geq 0 \\ m \in \mathbb{N}, & \sum_{i=1}^{m} \alpha_i = 1 \end{array} \right\}.$$

The convex hull of X is also the set of all convex combinations of elements of X.

Matrix games: Formality

Example. If $X = \left\{(1,1)^T, (1,3)^T, (2,2)^T\right\}$, then the convex hull of X is



Matrix games: Formality

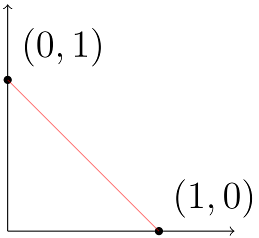
Definition.

The *unit simplex* in $\mathbb{R}^n$ is the convex hull of the standard basis,

$$\Delta := \operatorname{conv}\left(\{e_1, e_2, \ldots, e_n\}\right).$$

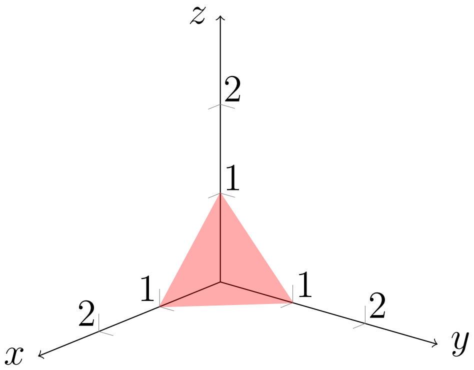
Matrix games: Formality

The unit simplex in $\mathbb{R}^2$:



Matrix games: Formality

The unit simplex in $\mathbb{R}^3$:



Matrix games: Formality

Another way to think of the unit simplex is

$$\Delta = \{x \in \mathbb{R}^n_+ \mid \langle x, e \rangle = 1\}$$

where $\mathbb{R}^n_+$ is the nonnegative orthant and

$$e := (1, 1, \ldots, 1)^T,$$
$$\langle x, e \rangle = x_1 + x_2 + \cdots + x_n.$$

Matrix games: Formality

A *two-person zero-sum matrix game* consists of,

- A matrix $L \in \mathbb{R}^{n \times n}$,
- The unit simplex $\Delta \subseteq \mathbb{R}^n$,
- The payoff function $(x, y) \mapsto y^T L x$.

Matrix games: Formality

A matrix game is played as follows:

1. Alice chooses an $x \in \Delta$,
2. Bob simultaneously chooses a $y \in \Delta$,
3. Then, the payoff is made from Bob to Alice.

Matrix games: Formality

Each player wants to maximize his or her payoff.

The game is zero-sum, so that goal is equivalent to minimizing the payoff to his or her opponent.

Matrix games: Formality

Traditionally one assumes that each player wants to maximize his *worst-case* payoff.

That is, he wants to *guarantee* himself the largest payoff possible.

Matrix games: Formality

Thus Alice's goal is to find

$$\operatorname*{argmax}_{x \in \Delta} \left(\min_{y \in \Delta} \left(y^T L x \right) \right),$$

and Bob's goal is to find

$$\operatorname*{argmin}_{y \in \Delta} \left(\max_{x \in \Delta} \left(y^T L x \right) \right).$$

Matrix games: Formality

Theorem (von Neumann, 1944).

There exists a pair $(\bar{x}, \bar{y})$ of strategies that simultaneously solves both of these problems.

Definition. The associated payoff

$$v(L) \coloneqq \bar{y}^T L \bar{x}$$

is the *value of the game.*

Matrix games: Formality

Theorem (Dantzig, 1951).

Every two-person zero-sum matrix game is solved by a linear program.

Matrix games: Formality

Alice wants to:

- Maximize (over x) some number ν,
- Subject to the fact that ν is a lower bound on her payout,
- And subject to $x \in \Delta$.

Matrix games: Formality

If we translate those goals to a constrained convex optimization problem, we obtain:

$$\begin{aligned} \text{maximize} \quad & \nu \\ \text{subject to} \quad & Lx \geq \nu e \\ & e^T x = 1 \\ & x \geq 0. \end{aligned}$$

This is a linear program in nonstandard form. Its dual turns out to be Bob's problem.

Part 2, Section 3

Linear games: Symmetric

Linear games: Symmetric

Definition. A *symmetric* cone is,

- A cone
- Closed
- Convex
- Solid
- Pointed
- Self-dual
- Homogeneous

Linear games: Symmetric

Definition.

Suppose that K is a cone and $B \subseteq K$ does not contain the origin. If any nonzero $x \in K$ can be uniquely represented $x = \lambda b$ where $\lambda > 0$ and $b \in B$, then B is a base of K.

In English: a base is a cross-section of the cone.

Linear games: Symmetric

Observations:

- The strategy set Δ in the classical case forms a base for the cone $\mathbb{R}^n_+$.
- The cone $\mathbb{R}^n_+$ is *symmetric*.

Linear games: Symmetric

Idea (Gowda and Ravindran[1], 2015).

Why don't we replace $\mathbb{R}^n_+$ with some other *symmetric* cone K, and Δ with a base of K?

Instead of a matrix, we'll have a linear operator L, and the payoff function $(x, y) \mapsto \langle L(x), y \rangle$.

A lot of stuff still works.

Linear games: Symmetric

If K is a symmetric cone and $e \in \operatorname{int}(K)$, then

$$\Delta \coloneqq \{x \in K \mid \langle x, e \rangle = 1\}$$

still forms a compact base for K.

Linear games: Symmetric

As a result, von Neumann's proof still works.

Theorem.

There exists a pair $(\bar{x}, \bar{y}) \in \Delta \times \Delta$ such that

$$\langle L(x), \bar{y} \rangle \leq \langle L(\bar{x}), \bar{y} \rangle \leq \langle L(\bar{x}), y \rangle$$

for all $(x, y) \in \Delta \times \Delta$.

Linear games: Symmetric

Ok, but why bother?

In the classical case, Raghavan [2] derived results for games whose payoff matrix is an **M**-matrix—a family related to the **Z**-matrices and nonnegative matrices.

Linear games: Symmetric

Those matrix families have been generalized to linear operators on closed convex cones, and we are interested in their properties.

Existing game theory results may provide insight.

In fact, many results survive the generalization.

Part 2, Section 4

Linear games: Asymmetric

Linear games: Asymmetric

We can go further!

- The cone doesn't have to be symmetric.
- The players can have different strategy sets.

Question. But why?

Linear games: Asymmetric

Answer. Because why not?

The operator families that we are interested in are defined for any closed convex cone.

And most things done by Gowda and Ravindran generalize with little additional work.

Linear games: Asymmetric

In these games we will consider a proper cone K with dual K^*. Take two points $e_1 \in \operatorname{int}(K)$ and $e_2 \in \operatorname{int}(K^*)$, and define the strategy sets,

$$\begin{aligned}\Delta_1 &\coloneqq \{x \in K \mid \langle x, e_2 \rangle = 1\}\\ \Delta_2 &\coloneqq \{y \in K^* \mid \langle y, e_1 \rangle = 1\}.\end{aligned}$$

These are compact bases for their cones.

Linear games: Asymmetric

This game is played in a familiar manner:

1. Alice chooses an $x \in \Delta_1$.
2. At the same time, Bob chooses a $y \in \Delta_2$.
3. Bob pays Alice $\langle L(x), y \rangle$.

Once more, we are promised optimal strategies.

Linear games: Asymmetric

So we have the data:

1. A linear operator L.
2. A proper cone K.
3. A point $e_1 \in \operatorname{int}(K)$.
4. A point $e_2 \in \operatorname{int}(K^*)$.

Linear games: Asymmetric

And Alice wants to find

$$\operatorname*{argmax}_{x \in \Delta_1} \left(\min_{y \in \Delta_2} \left(\langle L(x), y \rangle \right) \right)$$

and Bob wants to find

$$\operatorname*{argmin}_{y \in \Delta_2} \left(\max_{x \in \Delta_1} \left(\langle L(x), y \rangle \right) \right).$$

Linear games: Asymmetric

That's all good, but how do we solve them?

Theorem.

In our game,

$$\nu e_2 - L^*(\bar{y}) \in K^* \text{ and } L(\bar{x}) - \nu e_1 \in K$$

for $\nu \in \mathbb{R}$ if and only if ν is the value of the game and $(\bar{x}, \bar{y})$ is optimal for it.

Linear games: Asymmetric

From that theorem, Alice's goal is to

$$\begin{array}{lr} \text{maximize} & \nu \\ \text{subject to} & x \in K \\ & \langle x, e_2 \rangle = 1 \\ & \nu \in \mathbb{R} \\ & L\left(x\right) - \nu e_1 \in K \end{array}$$

Linear games: Asymmetric

Definition.

The primal cone program in standard form is,

$$\begin{array}{ll} \text{minimize} & \langle b, \xi \rangle \\ \text{subject to} & \Lambda\left(\xi\right) - c \in K_2 \\ & \xi \in K_1 \end{array}$$

where K_1 and K_2 are closed convex cones.

Linear games: Asymmetric

Theorem.

Alice is trying to solve the primal cone program, and Bob is trying to solve its dual.

Proof.

Make clever substitutions and then check that everything works. $\square$

Linear games: Asymmetric

Problem.

We don't know how to solve cone programs, either.

We do however know how to solve a few *symmetric* cone programs.

Linear games: Asymmetric

Corollary.

If K is a symmetric cone, then the associated game is solved by a symmetric cone program.

This brings us back to the setting of Gowda and Ravindran, albeit with two strategy sets Δ_1 and Δ_2 instead of just Δ.

Part 3, Section 5

Dunshire: CVXOPT

Dunshire: CVXOPT

CVXOPT is a free software package for convex optimization in the Python language.

It uses interior point methods to solve cone programs.

The solvers are described in *The CVXOPT linear and quadratic cone program solvers*, by Lieven Vandenberghe (2010).

Dunshire: CVXOPT

The CVXOPT *conelp* solver can solve a cone program over a cartesian product of,

- The nonnegative orthant.
- The Lorentz ice-cream cone.
- The symmetric positive semidefinite cone.

Dunshire: CVXOPT

Problem.

CVXOPT expects this primal problem:

$$\begin{aligned} \text{minimize } & c^T x \\ \text{subject to } & Gx + s = h \\ & Ax = b \\ & s \in C. \end{aligned}$$

Dunshire: CVXOPT

Solution. Divine inspiration: let,

$$
\begin{aligned}
C &= K \times K \\
x &= (\nu, p)^T \\
b &= 1,\ h = 0 \\
c &= (-1, 0)^T \\
A &= \begin{bmatrix} 0 & e_2^T \end{bmatrix} \\
G &= \begin{bmatrix} 0 & -I \\ e_1 & -L \end{bmatrix}.
\end{aligned}
$$

Dunshire: CVXOPT

Then the CVXOPT problem becomes,

$$\begin{aligned} \text{minimize} \quad & -\nu \\ \text{subject to} \quad & p = s_1 \\ & Lp - \nu e_1 = s_2 \\ & \langle e_2, p \rangle = 1 \\ & \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \begin{bmatrix} K \\ K \end{bmatrix}. \end{aligned}$$

This is Alice's problem in the variable p.

Dunshire: CVXOPT

Since $s_1, s_2 \in K$ are arbitrary, the constraint

$$p = s_1$$

simply means that $p \in K$. The same goes for

$$Lp - \nu e_1 = s_2 \in K.$$

Dunshire: CVXOPT

Thus we have presented Alice's goal to CVXOPT:

$$\begin{aligned} \text{maximize} \quad & \nu \\ \text{subject to} \quad & p \in K \\ & Lp - \nu e_1 \in K \\ & \langle e_2, p \rangle = 1. \end{aligned}$$

Its dual is Bob's goal as we would hope.

Part 3, Section 6

Dunshire: The library

Dunshire: The library

Dunshire is a Python library for solving linear games over symmetric cones.

It uses CVXOPT, but performs the tedious pre- and post-processing for you.

Dunshire: The library

Dunshire gives you access to the symmetric cones that CVXOPT can handle.

```
>>> from dunshire import *
>>> K1 = NonnegativeOrthant(5)
>>> print(K1)
Nonnegative orthant in the real 5-space
>>> K2 = IceCream(2)
>>> print(K2)
Lorentz "ice cream" cone in the real 2-space
```

Dunshire: The library

```
>>> K3 = SymmetricPSD(7)
>>> print(K3)
Cone of symmetric positive-semidefinite
matrices on the real 7-space
```

Cartesian products can be created, too:

```
>>> K = CartesianProduct(K1, K2, K3)
```

Dunshire: The library

Let's solve *rock-paper-scissors*!

Dunshire requires four data,

- A linear operator L.
- A symmetric cone K.
- A point $e_1 \in \operatorname{int}(K)$.
- A point $e_2 \in \operatorname{int}(K) = \operatorname{int}(K^*)$.

Dunshire: The library

First we set up the problem. . .

```
>>> L = [ [ 0, 1,-1],
...       [-1, 0, 1],
...       [ 1,-1, 0] ]
>>> K = NonnegativeOrthant(3)
>>> e2 = e1 = [1,1,1]
>>> G = SymmetricLinearGame(L,K,e1,e2)
```

Then, we solve it.

Dunshire: The library

```
>>> print(G.solution())
Game value: 0.0000000
Player 1 optimal:
  [0.3333333]
  [0.3333333]
  [0.3333333]
Player 2 optimal:
  [0.3333333]
  [0.3333333]
  [0.3333333]
```

Dunshire: The library

As we all know, nobody wins *rock-paper-scissors* on average—the expected payoff is zero.

The optimal strategy is to choose randomly between the options, each with probability $\frac{1}{3}$.

Dunshire: The library

Of course, solving linear programs is nothing new. So let's switch to the Lorentz ice-cream cone:

```
>>> from dunshire import *
>>> L = [[1,-1,12],[0,1,22],[-17,1,0]]
>>> K = IceCream(3)
>>> e1 = [1, 1/2, 1/4]
>>> e2 = [1, 1/4, 1/2]
>>> G = SymmetricLinearGame(L,K,e1,e2)
```

Dunshire: The library

```
>>> print(G.solution())
Game value: -11.4912789
Player 1 optimal:
  [0.7403323]
  [0.7233499]
  [0.1576604]
Player 2 optimal:
  [ 1.2547399]
  [-0.9304062]
  [ 0.8418529]
```

Dunshire: The library

Dunshire can catch common mistakes:

```
>>> from dunshire import *
>>> L = [[1,0],[0,1]]
>>> K = NonnegativeOrthant(2)
>>> e2 = e1 = [-1,1]
>>> G = SymmetricLinearGame(L,K,e1,e2)
Traceback (most recent call last):
...
ValueError: the point e1 must lie in the
interior of K
```

Dunshire: The library

Caveat.

The concept of “interior” is meaningless when working with floating point numbers.

If the distance between $x \in K$ and $\operatorname{bdy}(K)$ is 10^{-12}, does x lie in the interior of K?

Both answers lead to contradictions.

Part 3, Section 7

Dunshire: Starting points

Dunshire: Starting points

Most optimization algorithms need a starting point, where the search begins.

In our case, we have access to a decent starting point given to us in the problem data.

Dunshire: Starting points

Recall Alice's problem:

$$\begin{aligned} \text{maximize} \quad & \nu \\ \text{subject to} \quad & x \in K \\ & \langle x, e_2 \rangle = 1 \\ & \nu \in \mathbb{R} \\ & L(x) - \nu e_1 \in K \end{aligned}$$

We need a feasible ν and x for this problem.

Dunshire: Starting points

All we need is for ν, x to satisfy

$$\begin{aligned} x &\in K \\ \langle x, e_2 \rangle &= 1 \\ L\left(x\right) - \nu e_1 &\in K. \end{aligned}$$

But we already know that $e_2 \in K$, so a multiple of e_2 is a good candidate.

Dunshire: Starting points

If we let $x = \alpha e_2$, then

$$\langle x, e_2 \rangle = \alpha \left\| e_2 \right\|^2 = 1 \implies \alpha = \frac{1}{\left\| e_2 \right\|^2},$$

and thus

$$x = \alpha e_2 = \frac{e_2}{\left\| e_2 \right\|^2} \in K.$$

Dunshire: Starting points

Now the other constraint was,

$$L\left(x\right) - \nu e_1 \in K.$$

Assume that $\nu > 0$ and divide,

$$L\left(x\right) - \nu e_1 \in K$$

$$\Updownarrow$$

$$\frac{L\left(e_2\right)}{\nu \left\|e_2\right\|^2} - e_1 \in K.$$

Dunshire: Starting points

Let $d\left(e_1, K^c\right)$ be the distance from e_1 to the outside of K. Since $e_1 \in \operatorname{int}\left(K\right)$,

$$\frac{L\left(e_2\right)}{\nu\left\|e_2\right\|^2} - e_1 \in K$$

can be satisfied if we make

$$\left\|\frac{L\left(e_2\right)}{\nu\left\|e_2\right\|^2}\right\| \le d\left(e_1, K^c\right).$$

Dunshire: Starting points

If we arrange the scalar factors just right, then

$$\left\| L \left(\frac{e_2}{\|e_2\|} \right) \right\| \leq \|L\|_2 \, .$$

Rearranging, we find,

$$\nu \geq \frac{\|L\|_2}{d\left(e_1, K^c\right) \|e_2\|}.$$

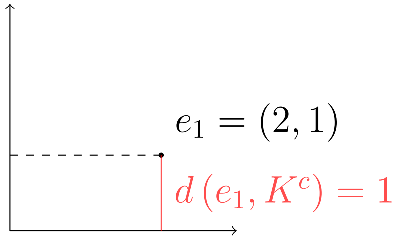
Dunshire: Starting points

But isn't computing $d\left(e_1, K^c\right)$ difficult?

In general, yes, but not for us!

Dunshire: Starting points

Example. When $K = \mathbb{R}^n_+$, we just need the smallest coordinate of e_1:



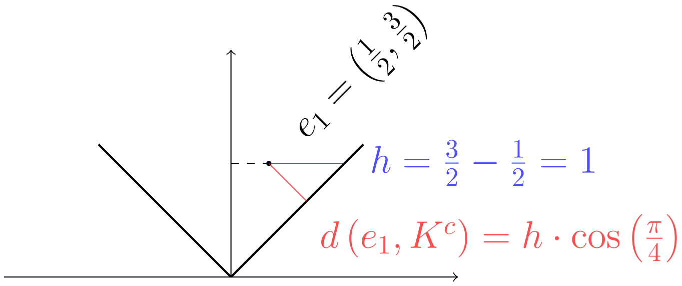
Dunshire: Starting points

For safety, we throw in another factor of two:

```
>>> from dunshire import *
>>> from cvxopt import matrix
>>> e1 = matrix([2,1])
>>> K = NonnegativeOrthant(2)
>>> K.ball_radius(e1)
0.5
```

Dunshire: Starting points

Example. The ice-cream cone falls to basic trigonometry:



$e_1 = \left(1/2, 3/2\right)$
$h = \frac{3}{2} - \frac{1}{2} = 1$
$d\left(e_1, K^c\right) = h \cdot \cos\left(\frac{\pi}{4}\right)$

Dunshire: Starting points

Rather than involve $\cos\left(\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2}$, we multiply h by $\frac{1}{2} < \frac{\sqrt{2}}{2}$ instead to get a safe distance:

```
>>> from dunshire import *
>>> from cvxopt import matrix
>>> e1 = matrix([3/2, 1/2])
>>> K = IceCream(2)
>>> K.ball_radius(e1)
0.5
```

Dunshire: Starting points

Proposition (Sossa, 2014).

Let V be an EJA with scaling factor $\theta := \frac{\langle e, e \rangle}{\operatorname{trace}(e)}$, and let $\Omega = \lambda^{-1}(Q)$ be a spectral set. Then for all $x \in V$,

$$\frac{1}{\sqrt{\theta}} d(x, \Omega) = \inf_{u \in Q} \left\| \lambda(x) - u \right\|_2 .$$

Applied to $x \in K$ and $\Omega = \operatorname{bdy}(K)$, this means we just need to find the smallest eigenvalue of x.

Part 3, Section 8

Dunshire: The future

Dunshire: The future

Missing: the Symmetric PSD cone.

Not hard, but requires a different representation than the other two cones.

Dunshire: The future

Missing: Cartesian products in the default order.

CVXOPT supports cartesian products, but only in a particular order. Once $\mathcal{S}^n_+$ is implemented, this should be easy.

Dunshire: The future

Missing: Cartesian products in arbitrary order.

We can reorder any Cartesian product with isomorphisms, but then we need to pre- and post-process the data/solution.

Part 4, Section 9

Applications: Confirm theory

Applications: Confirm theory

Theorem. Every game has a solution.

```
>>> from test.randomgen import *
>>> G = random_orthant_game()
>>> G.solution().game_value()
-0.03335527701955072
>>> G = random_icecream_game()
>>> G.solution().game_value()
3.5685813042518917
```

Applications: Confirm theory

Proposition. The value of the game for αL is $\alpha \geq 0$ times the value of the game for L.

```
>>> from test.randomgen import *
>>> G = random_game()
>>> (alpha, H) = random_nn_scaling(G)
>>> val1 = alpha*G.solution().game_value()
>>> val2 = H.solution().game_value()
>>> val1 - val2
6.155475018587708e-10
```

Applications: Confirm theory

Proposition. The value for $L + \alpha\,(e_1 \otimes e_2)$ is α plus the value for L.

```
>>> from test.randomgen import *
>>> G = random_game()
>>> (alpha, H) = random_translation(G)
>>> val1 = alpha + G.solution().game_value()
>>> val2 = H.solution().game_value()
>>> val1 - val2
7.506487015307428e-09
```

Applications: Confirm theory

Proposition. The value of (L, K, e_1, e_2) is negative the value of $(-L^*, K, e_2, e_1)$.

```
>>> from test.randomgen import *
>>> G = random_game()
>>> H = SymmetricLinearGame(-G.L(), G.K(),
...                          G.e2(), G.e1())
>>> val1 = G.solution().game_value()
>>> val2 = H.solution().game_value()
>>> val1 + val2
1.7701395904623496e-12
```

Applications: Confirm theory

Proposition. The value of a Lyapunov game (L is Lyapunov-like) is the same as its dual.

```
>>> from test.randomgen import *
>>> G = random_ll_game()
>>> H = G.dual()
>>> val1 = G.solution().game_value()
>>> val2 = H.solution().game_value()
>>> val1 - val2
-2.942365862068641e-08
```

Part 4, Section 10

Applications: Find $\mathbf{S}$*-operators*

Applications: Find **S**-operators

Definition.

L is an ***S**-operator* on K if there is a $d \in \operatorname{int}(K)$ such that $L(d) \in \operatorname{int}(K)$.

S-operators are important for complementarity problems, but we don't know how to find them.

Applications: Find **S**-operators

Proposition.

L is an **S**-operator on K if and only if the value of the game (L, K, e_1, e_2) is positive.

Applications: Find **S**-operators

Example.

The matrix

$$L := \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}$$

is an **S**-operator on the ice-cream cone in $\mathbb{R}^2$.

Applications: Find S-operators

Proof.

```
>>> from dunshire import *
>>> L = [[1, 2], [-3, 4]]
>>> K = IceCream(2)
>>> e1 = [1, 1/2]
>>> e2 = e1
>>> G = SymmetricLinearGame(L, K, e1, e2)
>>> G.solution().game_value()
1.777777803973679
```

References

[1] M. S. Gowda and G. Ravindran.
On the game-theoretic value of a linear transformation relative to a self-dual cone.
Linear Algebra and its Applications, 469:440–463, 2015.

[2] T. Raghavan.
Completely mixed games and M-matrices.
Linear Algebra and its Applications, 21:35–45, 1978.

[3] J. von Neumann and O. Morgenstern.
Theory of Games and Economic Behavior.
Princeton University Press, 1944.