

Title: PSI 2019/2020 - Computational Physics - Lecture 15

Speakers: Erik Schnetter

Collection: PSI 2019/2020 - Computational Physics

Date: March 11, 2020 - 12:30 PM

URL: <http://pirsa.org/20030032>

Activities Document Viewer Mar 11 12:36 • en 172.8%

Opinionated survey of (non)convex optimization tools(and related topics)
Notes.pdf

1 of 7

1 Reference ... 1
2 Definitions 1
3 Approaches 3
Bibliography 7

Opinionated survey of (non)convex optimization tools (and related topics)

BY DENIS ROSSET
Perimeter Institute

A generic optimization problem is written

$$\begin{aligned} & \text{minimize} && f(\vec{x}) \\ & \text{over} && \vec{x} \in \mathbb{R}^n \\ & \text{such that} && x_i \in \mathbb{Z} \quad i \in \mathcal{I} \subset \{1, \dots, n\} \\ & && g_j(\vec{x}) \leq 0 \quad j \in \mathcal{J} \\ & && h_k(\vec{x}) = 0 \quad k \in \mathcal{K} \end{aligned} \tag{1}$$

where $\mathcal{J} = \{1, \dots, |\mathcal{J}|\}$ and $\mathcal{K} = \{1, \dots, |\mathcal{K}|\}$.

1 Reference material

The material below is a synthesis of our first-hand experiences with various solvers and problems. Our reference for optimization is the book [9] by Nocedal and Wright (copy here <http://fourier.eng.hmc.edu/e176/lectures/NumericalOptimization.pdf>), and an excellent reference for *convex* optimization is the book [2] by Boyd, available on GitHub. For the practical aspects, the MOSEK modeling cookbook [1] is excellent; we cite the Release 3.2.1 available on GitHub.

Those books have been written at the turn of the century, but the theory has not changed much; machine

Activities Document Viewer

Mar 11 12:36

en

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

1 of 7

172.8%

1 Reference ... 1

2 Definitions 1

3 Approaches 3

Bibliography 7

1 Reference material

The material below is a synthesis of our first-hand experiences with various solvers and problems. Our reference for optimization is the book [9] by Nocedal and Wright (copy here <http://fourier.eng.hmc.edu/e176/lectures/NumericalOptimization.pdf>), and an excellent reference for *convex* optimization is the book [2] by Boyd, available on GitHub. For the practical aspects, the MOSEK modeling cookbook [1] is excellent; we cite the Release 3.2.1 available on GitHub.

Those books have been written at the turn of the century, but the theory has not changed much; machine learning has simply spurred interest in first order methods due to the size of the models.

2 Definitions

2.1 Number of objectives

An optimization problem without an objective is a *satisfiability problem* or *feasibility problem*.

When numerical errors are present, we can reformulate the feasibility problem

$$\begin{array}{ll} \text{Find} & \vec{x} \in \mathbb{R}^n \\ \text{such that} & x_i \in \mathbb{Z} \quad i \in \mathcal{I} \subset \{1, \dots, n\} \\ & g_j(\vec{x}) \leq 0 \quad j = 1, \dots, J \\ & h_k(\vec{x}) = 0 \quad k = 1, \dots, K \end{array} \quad (2)$$

as the minimization problem

$$\begin{array}{ll} \text{minimize} & s \\ \text{over} & \vec{x} \in \mathbb{R}^n, s \in \mathbb{R} \\ \text{such that} & x_i \in \mathbb{Z} \quad i \in \mathcal{I} \subset \{1, \dots, n\} \\ & g_j(\vec{x}) \leq s \quad j = 1, \dots, J \\ & -s \leq h_k(\vec{x}) \leq s = 0 \quad k = 1, \dots, K \end{array} \quad (3)$$

so that there is always a feasible solution with s big enough. This is especially helpful for convex problems (more on that later). Then “solver returns infeasible” means that the problem formulation is not numerically stable; otherwise a solution with $s \leq 0$ is feasible and $s > 0$ describes infeasibility.

Activities

Document Viewer

2 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale o... 3

2.5 Proble... 3

3 Approaches 3

Bibliography 7

When several objectives are present, we are speaking of multiobjective optimization.

The simplest approach is to minimize a linear combination of objectives

$$f(\vec{x}) = \alpha_1 f_1(\vec{x}) + \alpha_2 f_2(\vec{x}) + \dots$$

for various values of $\alpha_1, \alpha_2, \dots$; this is part of the *scalarization* approaches. To organize the results, plot the *Pareto front*. We also had success with genetic algorithms which maintain good diversity in the population of solutions (for example NSGA-II [4]).

2.2 Type of objective

The simplest type of objective function is *linear*:

$$f(\vec{x}) = \vec{c}^\top \cdot \vec{x} + d. \tag{4}$$

(Strictly speaking, it is *affine* for $d \neq 0$ although this subtle point is often lost. We use *linear* for both.)

Otherwise, the problem is said to have *nonlinear* constraints.

An objective function is *convex* when [2, Chap. 2 and 3]

$$f(\alpha \vec{x}_1 + (1 - \alpha)\vec{x}_2) \leq \alpha f(\vec{x}_1) + (1 - \alpha)f(\vec{x}_2), \quad \alpha \in [0, 1]. \tag{5}$$

This implies that any local minimum of f is also a global minimum [2, 4.2.2].

Polynomial objective functions have the form

$$f(\vec{x}) = \alpha + \sum_{i_1} \beta_{i_1} x_{i_1} + \sum_{i_1 \leq i_2} \gamma_{i_1 i_2} x_{i_1} x_{i_2} + \dots \tag{6}$$

Sometimes, objective functions are not known in closed form: it can be obtained from an integration, solving a differential equation, solving an inner optimization problem, even running an experiment. The computation of $f(\vec{x})$ may even be *noisy*. In those cases, the derivatives $\partial f / \partial x_i$, $\partial^2 f / \partial x_i \partial x_j$ may not be available, and cannot be computed by finite difference schemes: the solving algorithm needs to be *derivative-free* [9, Chap. 9].

Pirsa: 20030032

Page 4/54

Activities

Document Viewer

3 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale o... 3

2.5 Proble... 3

3 Approaches 3

Bibliography 7

A problem has *convex constraints* when all inequality constraints g_j are convex (5), all equality constraints h_k are linear (4).

Constraints can also be polynomial as in (6).

A *semidefinite programming* (SDP) constraint is constructed as follows. Given fixed symmetric matrices C and $\{A_i\}_i$, we write the affine combination $\chi = C - \sum_{i=1}^n A_i x_i$ using the optimization variables (the signs are somewhat arbitrary for now). We then impose that χ is a *semidefinite positive* matrix, i.e. that its eigenvalues $\text{eig}(\chi)$ are all nonnegative [2, 4.6.2]. Semidefinite constraints are quite expressive (two examples: [10] [5]) and relatively efficient to compute.

2.4 Scale of problems

This relates to the number of variables and constraints, and will guide algorithm selection.

Small problems include a handful of variables and constraints (say 1 – 10). They can sometimes be solved by exact/algebraic methods.

Medium-scale problems usually involve less than a thousand variables and constraints: for those, second-order derivative information (the Hessian) can be stored in memory using a dense matrix.

Large-scale problems are such that second-order derivative information is too expensive to be stored in memory (in our experience, algorithms are more memory bound than CPU bound). Algorithms use the structure of the problem to reduce computational requirements: for example sparsity, approximating second-order information or even not using second-order information at all (at the price of convergence speed).

2.5 Problem types

Objective type	Constraint type	Integrality	Problem type
Linear	Linear	$\mathcal{I} = \emptyset$	Linear program (LP)
Linear	Linear	$\mathcal{I} \neq \emptyset$	Mixed integer linear program (MILP)
Linear	Semidefinite	$\mathcal{I} = \emptyset$	Semidefinite program (SDP)
Convex	Convex	$\mathcal{I} = \emptyset$	Convex program
Polynomial	Polynomial	$\mathcal{I} = \emptyset$	Polynomial optimization problem (POP)
Nonlinear	None	$\mathcal{I} = \emptyset$	Unconstrained optimization problem
Nonlinear	Bounds	$\mathcal{I} = \emptyset$	Bounded optimization problem
Nonlinear	Nonlinear	$\mathcal{T} \neq \emptyset$	Mixed integer nonlinear program (MINLP)

Pirsa: 20030032

Page 5/54

Activities

Document Viewer

4 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

Q

≡

⌵

⌶

⌵

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale O... 3

2.5 Proble... 3

3 Approaches 3

3.1 Proble... 3

3.1.1 De... 4

3.1.2 De... 4

3.2 Proble... 4

3.3 Using a... 5

3.4 Special... 6

Bibliography 7

3.1 Problems specified by user-defined functions

In this approach, algorithms start with a user-given point \vec{x}_0 , gather information about the objective and constraints around \vec{x}_0 , compute a first iterate \vec{x}_1 that respects the constraints and such that $f(\vec{x}_1) < f(\vec{x}_0)$, repeat the procedure to obtain \vec{x}_2 , etc..., until one of the stopping criteria of the algorithm is achieved (for example, the improvement $f(\vec{x}_{i+1}) - f(\vec{x}_i)$ is small, the norm of the estimated gradient is below some tolerance, or the step size $\|\vec{x}_{i+1} - \vec{x}_i\|$ is too small).

The main difference here is between convex and nonconvex programs. If the problem is not convex, there is no guarantee that the returned solution is globally optimal, or even feasible, while convex programs provide such certainty.

3

(Several heuristics are available for nonconvex programs, but they are outside the scope of this survey).

3.1.1 Derivative-free algorithms

Objective type	Constraint type	Integrality	Problem type
• Nonlinear	None	$\mathcal{I} = \emptyset$	Unconstrained optimization problem
Nonlinear	Bounds	$\mathcal{I} = \emptyset$	Bounded optimization problem

The algorithms below do not require gradient/Hessian information, see [9, Chap. 9].

The Nelder-Mead or amoeba method is common for unconstrained problems: `fminsearch` in Matlab standard toolbox; available in the <https://github.com/JulianLSolvers/Optim.jl> Julia toolbox. It works on functions that are not differentiable, but can converge to non-optimal points. It is best used on problems

Pirsa: 20030032

Page 6/54

Activities

Document Viewer

3 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale o... 3

2.5 Proble... 3

3 Approaches 3

3.1 Proble... 3

3.1.1 De... 4

3.1.2 De... 4

3.2 Proble... 4

3.3 Using a... 5

3.4 Special... 6

Bibliography 7

Nonlinear

Nonlinear

$\mathcal{L} \neq \emptyset$

Mixed integer nonlinear program (MINLP)

Note that LP, SDP are convex programs. Integrality constraints appear mostly in MILP, generic MINLP are an active area of research.

3 Approaches

3.1 Problems specified by user-defined functions

In this approach, algorithms start with a user-given point \vec{x}_0 , gather information about the objective and constraints around \vec{x}_0 , compute a first iterate \vec{x}_1 that respects the constraints and such that $f(\vec{x}_1) < f(\vec{x}_0)$, repeat the procedure to obtain \vec{x}_2 , etc..., until one of the stopping criteria of the algorithm is achieved (for example, the improvement $f(\vec{x}_{i+1}) - f(\vec{x}_i)$ is small, the norm of the estimated gradient is below some tolerance, or the step size $\|\vec{x}_{i+1} - \vec{x}_i\|$ is too small).

The main difference here is between convex and nonconvex programs. If the problem is not convex, there is no guarantee that the returned solution is globally optimal, or even feasible, while convex programs provide such certainty.

3

(Several heuristics are available for nonconvex programs, but they are outside the scope of this survey).

3.1.1 Derivative-free algorithms

Objective type	Constraint type	Integrality	Problem type
NT	NT	NT	NT

Activities Document Viewer Mar 11 12:37 • en 172.8%

Opinionated survey of (non)convex optimization tools(and related topics)
Notes.pdf

4 of 7

1 Reference ... 1
2 Definitions 1
2.1 Numbe... 1
2.2 Type of... 2
2.3 Type of... 2
2.4 Scale o... 3
2.5 Proble... 3
3 Approaches 3
3.1 Proble... 3
3.1.1 De... 4
3.1.2 De... 4
3.2 Proble... 4
3.3 Using a... 5
3.4 Special... 6
Bibliography 7

approximate the second-order derivative information (Hessian). It is provided by the Optimization Toolbox in MATLAB through the `quasi-newton` algorithm of the function `fminunc`; in Julia, see the <https://github.com/JuliaNLSolvers/Optim.jl> toolbox. In Julia, a limited memory version (LBFGS) is provided for large scale problems, where the Hessian matrix is approximated using the last m iterates, reducing memory use.

For constrained problems, interior-point methods try to move in a central path at a certain distance of the constraints. In the MATLAB Optimization Toolbox, `fmincon` has an implementation of the interior-point method; the `Ipopt` solver has Julia bindings: <https://github.com/JuliaOpt/Ipopt.jl>. MATLAB additionally has an sequential quadratic programming algorithm that can works more precisely for medium scale problems (instead of following a central path, it actively searches for inequality constraints that are be saturated).

The Knitro nonlinear solver is state-of-the-art but academic licenses are not free.

Final note: if the Julia code is written with generic types in mind, the gradients can be automatically computed by forward autodifferentiation.

3.2 Problems given in a canonical form

Some problem types have canonical forms. For example, linear programs (LP) have the (primal) form:

$$\begin{aligned} &\text{minimize} && \vec{c}^\top \vec{x} \\ &\text{over} && \vec{x} \in \mathbb{R}^n \\ &\text{such that} && A\vec{x} = \vec{b} \\ &&& \vec{x} \geq 0 \end{aligned} \tag{8}$$

4

Activities

Document Viewer

5 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale o... 3

2.5 Proble... 3

3 Approaches 3

3.1 Proble... 3

3.1.1 De... 4

3.1.2 De... 4

3.2 Proble... 4

3.3 Using a... 5

3.3.1 YA... 5

3.3.2 CV... 5

3.3.3 Ju... 6

3.3.4 Co... 6

3.4 Special... 6

Bibliography 7

The solver Gurobi <https://www.gurobi.com/> is excellent for linear and mixed-integer linear programming, though MOSEK is close (and MOSEK supports SDPs).

For convex problems given in canonical forms, the MOSEK <https://www.mosek.com/> solver represents the state of the art. It has a free academic license. For extended precision arithmetic, use the SDPA family <http://sdpa.sourceforge.net/family.html>. We did not have great experiences with the SCS solver (<https://github.com/cvxgrp/scs>), sometimes recommended in Julia examples, but it seems to have applications. Its code is particularly simple to follow though.

All these solvers interface with Matlab and Julia; for Julia, see in particular <https://github.com/JuliaOpt>.

3.3 Using a modeling framework

Writing a convex program in a canonical form (8)-(9) can be tedious. Luckily, both MATLAB and Julia have modeling frameworks that enable the use of standard syntax. Under the hood, the problems are transformed in the canonical form and solved. We describe below the different frameworks available. Some of them work even for nonconvex programs whose objective and constraints are given in a closed form.

Still, modeling frameworks are not magical: they often result in canonical formulations that are more complex than necessary, with obvious redundancies (some frameworks, for example, do not substitute simple equality constraints such as $x_1 = x_2$).

3.3.1 YALMIP (Matlab)

YALMIP is the one of the oldest framework in existence. It's based on MATLAB, though it runs on the open source clone Octave (<https://www.gnu.org/software/octave/>).

The core of YALMIP is convex optimization over real-valued linear or semidefinite programs, and that core is stable, and interfaces with myriads of solvers.

Beyond convex programming, YALMIP has support for nonconvex problems, complex variables, dualization, global optimization, polynomial optimization, ... but often it is not possible to use two extensions at the same time (for example, getting the dual variable corresponding to a complex SDP constraint). We observed that YALMIP performs the translation of the problem to its canonical form quite mechanically, without applying obvious eliminations.

YALMIP has amazing documentation.

3.3.2 CVX (Matlab)

Activities Document Viewer Mar 11 12:38 • en 172.8%

Opinionated survey of (non)convex optimization tools(and related topics)
Notes.pdf

6 of 7

1 Reference ... 1
2 Definitions 1
2.1 Numbe... 1
2.2 Type of... 2
2.3 Type of... 2
2.4 Scale o... 3
2.5 Proble... 3
3 Approaches 3
3.1 Proble... 3
3.1.1 De... 4
3.1.2 De... 4
3.2 Proble... 4
3.3 Using a... 5
3.3.1 YA... 5
3.3.2 CV... 5
3.3.3 Ju... 6
3.3.4 Co... 6
3.4 Special... 6
Bibliography 7

obvious to the toolbox (in practice, this is not a huge restriction). CVX encourages a modular style of convex programming: convex sets of importance can be defined in user functions and reused.

5

The version 3 has critical bugs, but the version 2.2 is rock solid. It supports complex variables, including getting the dual variables of complex constraints. The documentation is great. CVX supports a limited number of solvers, compared to YALMIP, but it supports MOSEK, which is often all that is needed.

CVX performs model translation better than YALMIP, and applies simple variable substitutions.

3.3.3 JuMP (Julia)

JuMP (<https://github.com/JuliaOpt/JuMP.jl>) looks like the Julia equivalent of YALMIP. It interfaces myriads of solvers, offers nonconvex modeling and a quantity of extensions. It is actively used in research. It does not seem to be as versatile as YALMIP yet, and the documentation is sometimes lacking, in particular for its extensions. Of course YALMIP has nearly two decades of existence.

3.3.4 Convex.jl (Julia)

On the other hand, Convex.jl (<https://github.com/JuliaOpt/Convex.jl>) is a Julia implementation of Disciplined Convex Programming, and has a limited scope compared to JuMP. In our limited experience, it seems more robust and ready to tackle medium or large scale optimization problems; it supports a large variety of solvers. Its documentation is pretty great.

3.4 Special cases

Activities

Document Viewer

6 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale o... 3

2.5 Proble... 3

3 Approaches 3

3.1 Proble... 3

3.1.1 De... 4

3.1.2 De... 4

3.2 Proble... 4

3.3 Using a... 5

3.3.1 YA... 5

3.3.2 CV... 5

3.3.3 Ju... 6

3.3.4 Co... 6

3.4 Special... 6

Bibliography 7

PANDA	http://comopt.ifl.uni-heidelberg.de/software/PANDA/	Binary
-------	---	--------

Julia has bindings to many libraries: <https://juliapolyhedra.github.io/> under a common interface, which we haven't tried.

We also used rational LP solvers, which can be very efficient for medium scale problems:

Software	Link	Platform
QSopt_ex	https://www.math.uwaterloo.ca/~bico/qsopt/ex/index.html	Binary
SoPlex	https://soplex.zib.de/index.php	Binary
GLPK (exact mode)	https://www.gnu.org/software/glpk/	Julia, Matlab

For polynomial problems, the elimination of quantifiers can be done using the Cylindrical Algebraic Decomposition [3], implemented in Mathematica.

3.4.2 Verified bounds

Semidefinite programs are often too expensive to be solved exactly, apart from small examples [7]. Nevertheless certified bounds can be obtained using the VSDP package in Matlab [8].

3.4.3 Discrete/combinatorial optimization

When all variables are discrete and bounded, ideally all $x_i \in \{0, 1\}$, consider using a SAT solver. We recommend MiniSat, <http://minisat.se/>. Beyond that, the Z3 solver <https://github.com/Z3Prover/z3>.

Activities

Document Viewer

6 of 7

Opinionated survey of (non)convex optimization tools(and related topics)

Notes.pdf

172.8%

1 Reference ... 1

2 Definitions 1

2.1 Numbe... 1

2.2 Type of... 2

2.3 Type of... 2

2.4 Scale O... 3

2.5 Proble... 3

3 Approaches 3

3.1 Proble... 3

3.1.1 De... 4

3.1.2 De... 4

3.2 Proble... 4

3.3 Using a... 5

3.3.1 YA... 5

3.3.2 CV... 5

3.3.3 Ju... 6

3.3.4 Co... 6

3.4 Special... 6

Bibliography 7

variety of solvers. Its documentation is pretty great.

3.4 Special cases

3.4.1 Exact arithmetic

To solve linear programs, including those with multiple objectives, the feasible set can be manipulated using Fourier-Motzkin elimination [12]. We employed successfully, for combinatorial problems.

Software	Link	Platform
PORTA	http://porta.zib.de/	Binary
PANDA	http://comopt.ifl.uni-heidelberg.de/software/PANDA/	Binary

Julia has bindings to many libraries: <https://juliapolyhedra.github.io/> under a common interface, which we haven't tried.

We also used rational LP solvers, which can be very efficient for medium scale problems:

Software	Link	Platform
QSopt_ex	https://www.math.uwaterloo.ca/~bico/qsopt/ex/index.html	Binary
SoPlex	https://soplex.zib.de/index.php	Binary
GLPK (exact mode)	https://www.gnu.org/software/glpk/	Julia, Matlab

For polynomial problems, the elimination of quantifiers can be done using the Cylindrical Algebraic Decomposition [3], implemented in Mathematica.

3.4.2 Verified bounds

Semidefinite programs are often too expensive to be solved exactly, apart from small examples [7]. Nevertheless certified bounds can be obtained using the VSDP package in Matlab [8].

3.4.3 Discrete/combinatorial optimization

When all variables are discrete and bounded, ideally all $x_i \in \{0, 1\}$, consider using a SAT solver. We recommend MiniSat, <http://minisat.se/>. Beyond that, the Z3 solver <https://github.com/Z3Prover/z3>.

Pirsa: 20030032

Page 12/54

The screenshot displays a web browser window with multiple tabs open, including "Home Page - Select or ch...", "class-2020-03-11/", "Step1_ConvexProgram...", and "Step2_CausalStructures". The active tab is "Step1_ConvexProgram...". The address bar shows the URL "localhost:8888/notebooks/class-2020-03-11/Step1_ConvexProgramming.ipynb".

The Jupyter Notebook interface shows the title "jupyter Step1_ConvexProgramming (autosaved)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The right side of the interface indicates "Trusted" status and "Julia 1.3.1".

The notebook content begins with a code cell:

```
In [1]: using Convex  
using GLPK  
using Printf
```

This is followed by a section titled "Introduction to convex optimization". The mathematical formulation for minimizing a function $f(\vec{x})$ over $\vec{x} \in R^n$ is shown, along with constraints $g_j(\vec{x}) \leq 0$ for $j \in J$ and $h_k(\vec{x}) = 0$ for $k \in K$.

A definition of convexity is provided: where all f , g_j and h_k are convex; for $f(\vec{x})$:

$$f(a\vec{x}_1 + (1-a)\vec{x}_2) \leq af(\vec{x}_1) + (1-a)f(\vec{x}_2), \quad a \in [0, 1]$$

The text explains that on the left, an example of a convex function is given, plotting the line segment $af(\vec{x}_1) + (1-a)f(\vec{x}_2)$ in dark blue. On the right, an example of a nonconvex function is shown.

Two graphs illustrate these concepts. The left graph, labeled "Convex", shows a U-shaped curve with points \vec{x}_1 and \vec{x}_2 on the x-axis. Dashed vertical lines connect these points to the curve at $f(\vec{x}_1)$ and $f(\vec{x}_2)$. A solid blue line segment connects these two points on the curve, representing the chord. The area under the curve is shaded light blue. The right graph, labeled "Nonconvex", shows a W-shaped curve with a local minimum between two other minima. A red line segment connects two points on the curve, showing it lies above the curve, indicating nonconvexity.

Below the graphs, the text states: "In the same spirit, convex sets include the line segment between any two of their points:". This is illustrated by a green circle containing a line segment connecting two points labeled x and y .

The final sentence reads: "while convex sets do not:", suggesting a contrast with non-convex sets.

Activities

Document Viewer

Mar 11 12:39

en

Step1_ConvexProgramming - Jupyter Notebook - Mozilla Firefox

Home

w

Computati... Winter-2020

class-2020-03-11

Step1_ConvexProgramming.pdf

82.4%

1 of 5

Introduction ... 1

Linear opti... 2

Duality 3

Step1_ConvexProgramming

March 11, 2020

```
[1]: using Convex
      using GLPK
      using Printf
```

1 Introduction to convex optimization

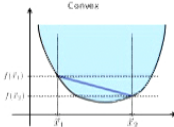
$$\begin{aligned} & \text{minimize} && f(\bar{x}) \\ & \text{over} && \bar{x} \in K^n \\ & && g_j(\bar{x}) \leq 0 \quad j \in \mathcal{J} \\ & && h_k(\bar{x}) = 0 \quad k \in \mathcal{K} \end{aligned} \quad (1)$$

where all f , g_j and h_k are convex, for $f(\bar{x})$:

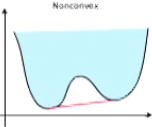
$$f(a\bar{x}_1 + (1-a)\bar{x}_2) \leq af(\bar{x}_1) + (1-a)f(\bar{x}_2), \quad a \in [0,1] \quad (2)$$

On left, an example of a convex function is given below, where we plot the line segment $af(\bar{x}_1) + (1-a)f(\bar{x}_2)$ in dark blue. On the right, an example of a nonconvex function.

Convex

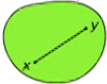


Nonconvex



ConvexFun.svg

In the same spirit, convex sets include the line segment between any two of their points:



1

Size

Modified

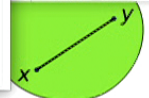
5.0 kB	12:30
7.4 kB	12:30
6.8 kB	12:30
12.9 kB	12:30
5.3 kB	12:30
11.1 kB	12:30
8.2 kB	12:30
14.8 kB	12:30
1.3 kB	12:30
12.8 kB	12:35
191.3 kB	12:30
38.3 kB	12:30
9.1 kB	12:30
123.7 kB	12:30
28.2 kB	12:30
8.2 kB	12:30
15.6 kB	12:30
21 kB	12:30

Logout

Julia 1.3.1

right, an example of

Step1_ConvexProgramming.pdf" selected (191.3 kB)



while convex sets do not:

Activities Document Viewer Mar 11 12:39 en 172.8%

1 of 5 Step1_ConvexProgramming.pdf

Introduction ... 1
Linear opti... 2
Duality 3

Step1_ConvexProgramming

March 11, 2020

```
[ ]: using Convex  
      using GLPK  
      using Printf
```

1 Introduction to convex optimization

$$\begin{aligned} & \text{minimize} && f(\vec{x}) \\ & \text{over} && \vec{x} \in R^n \\ & && g_j(\vec{x}) \leq 0 \quad j \in \mathcal{J} \\ & && h_k(\vec{x}) = 0 \quad k \in \mathcal{K} \end{aligned} \tag{1}$$

where all f , g_j and h_k are convex; for $f(\vec{x})$:

$$f(\alpha \vec{x}_1 + (1 - \alpha) \vec{x}_2) \leq \alpha f(\vec{x}_1) + (1 - \alpha) f(\vec{x}_2), \quad \alpha \in [0, 1] \tag{2}$$

On left, an example of a convex function is given below, where we plot the line segment

Activities

Document Viewer

1 of 5

Step1_ConvexProgramming.pdf

172.8%

Introduction ... 1

Linear opti... 2

Duality 3

March 11, 2020

```

[]: using Convex
    using GLPK
    using Printf

```

1 Introduction to convex optimization

$$\begin{aligned}
 &\underset{\vec{x} \in R^n}{\text{minimize}} && f(\vec{x}) \\
 &&& g_j(\vec{x}) \leq 0 \quad j \in \mathcal{J} \\
 &&& h_k(\vec{x}) = 0 \quad k \in \mathcal{K}
 \end{aligned}
 \tag{1}$$

where all f , g_j and h_k are convex; for $f(\vec{x})$:

$$f(\alpha \vec{x}_1 + (1 - \alpha) \vec{x}_2) \leq \alpha f(\vec{x}_1) + (1 - \alpha) f(\vec{x}_2), \quad \alpha \in [0, 1] \tag{2}$$

On left, an example of a convex function is given below, where we plot the line segment $\alpha f(\vec{x}_1) + (1 - \alpha) f(\vec{x}_2)$ in dark blue. On the right, an example of a nonconvex function.

Convex

Nonconvex

Pirsa: 20030032

Page 16/54

Activities

Document Viewer

2 of 5

Step1_ConvexProgramming.pdf

172.8%

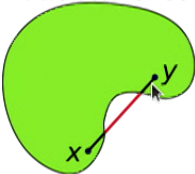
Introduction ... 1

Linear opti... 2

Primal c... 2

Duality 3

while convex sets do not:



Convex programs (1) have one crucial property: any local minimum is also a global minimum. In particular, this guarantees that (non pathological) optimization algorithms always find the global minimum up to numerical precision.

1.1 Linear optimization

The simplest kind of convex programs is a *linear program*, where the objective and all constraints are represented by linear functions such as (technically, these are *affine functions*):

$$f(\vec{x}) = \vec{c}^T \vec{x} + d \tag{3}$$

$$\begin{aligned} p^* = \text{minimize} \quad & x_1 + 2x_2 - x_3 \\ \text{over} \quad & \vec{x} \in R^3 \\ & x_1, x_2, x_3 \geq 0 \\ & x_1 + x_2 + x_3 = 1 \end{aligned} \tag{4}$$

We now solve that problem using the [Convex.jl](#) optimization toolbox.

```
[ ]: x = Variable(3)
constraints = [x >= 0, sum(x) == 1]
objective = x[1] + 2*x[2] - x[3]
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL ))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
```

Pirsa: 20030032

Page 19/54

Activities

Document Viewer

2 of 5

Step1_ConvexProgramming.pdf

172.8%

Introduction ... 1

Linear opti... 2

Primal c... 2

Duality 3

are represented by linear functions such as (technically, these are *affine functions*):

$$f(\vec{x}) = \vec{c}^\top \vec{x} + d \quad (3)$$

$$\begin{aligned} p^* = \text{minimize} \quad & x_1 + 2x_2 - x_3 \\ \text{over} \quad & \vec{x} \in \mathbb{R}^3 \\ & x_1, x_2, x_3 \geq 0 \\ & x_1 + x_2 + x_3 = 1 \end{aligned} \quad (4)$$

We now solve that problem using the [Convex.jl](#) optimization toolbox.

```
[ ]: x = Variable(3)
constraints = [x >= 0, sum(x) == 1]
objective = x[1] + 2*x[2] - x[3]
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL ))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
@printf("Optimal point      x = %s\n", x.value)
```

1.1.1 Primal canonical form of a linear program

The standard form of a linear program is as follows, with $\vec{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$.

$$\begin{aligned} \text{minimize} \quad & \vec{c}^\top \vec{x} \\ \text{over} \quad & \vec{x} \in \mathbb{R}^n \\ & A\vec{x} = \vec{b} \\ & \vec{x} \geq 0 \end{aligned} \quad (5)$$

The problem is fully specified by the matrix A and the vectors \vec{b} and \vec{c} .

Lecture exercise Complete the program data below to solve (4).

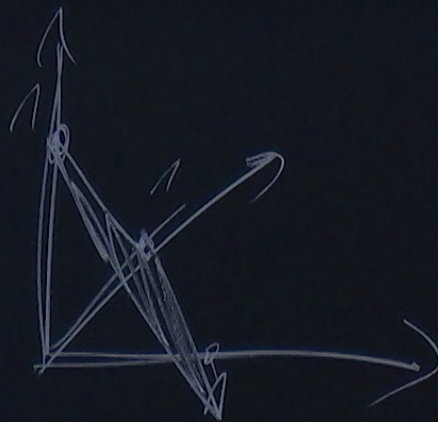
```
[ ]: using Convex
      using GLPK
```

Pirsa: 20030032

Page 20/54

$$p_{\ell} \epsilon_{lmq} + \epsilon_{pm\ell} \epsilon_{lkq} = 0$$

$$\begin{aligned} & \left(\epsilon^{lmn} \right) \Lambda^q \left(c \right)^2 x \\ & \Lambda^i, \Lambda^j = \underbrace{\epsilon_{jk} \left(\Lambda^i \right)^j \Lambda^k}_{= \Lambda^i, T^j} T_i \end{aligned}$$



Activities

Document Viewer

2 of 5

Step1_ConvexProgramming.pdf

172.8%

Introduction ... 1

Linear opti... 2

Primal c... 2

Duality 3

are represented by linear functions such as (technically, these are *affine functions*):

$$f(\vec{x}) = \vec{c}^\top \vec{x} + d \quad (3)$$

$$\begin{aligned} p^* = \text{minimize} \quad & x_1 + 2x_2 - x_3 \\ \text{over} \quad & \vec{x} \in \mathbb{R}^3 \\ & x_1, x_2, x_3 \geq 0 \\ & x_1 + x_2 + x_3 = 1 \end{aligned} \quad (4)$$

We now solve that problem using the [Convex.jl](#) optimization toolbox.

```
[ ]: x = Variable(3)
constraints = [x >= 0, sum(x) == 1]
objective = x[1] + 2*x[2] - x[3]
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL ))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
@printf("Optimal point      x = %s\n", x.value)
```

1.1.1 Primal canonical form of a linear program

The standard form of a linear program is as follows, with $\vec{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$.

$$\begin{aligned} \text{minimize} \quad & \vec{c}^\top \vec{x} \\ \text{over} \quad & \vec{x} \in \mathbb{R}^n \\ & A\vec{x} = \vec{b} \\ & \vec{x} \geq 0 \end{aligned} \quad (5)$$

The problem is fully specified by the matrix A and the vectors \vec{b} and \vec{c} .

Lecture exercise Complete the program data below to solve (4).

```
[ ]: using Convex
      using GLPK
```

Pirsa: 20030032

Page 22/54

Activities Firefox Web Browser Mar 11 12:45

Step1_ConvexProgramming - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgramming Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step1_ConvexProgramming.ipynb

jupyter Step1_ConvexProgramming (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

Linear optimization

The simplest kind of convex programs is a *linear program*, where the objective and all constraints are represented by linear functions such as (technically, these are *affine functions*):

$$f(\vec{x}) = \vec{c}^T \vec{x} + d$$

$$p^* = \underset{\vec{x} \in R^3}{\text{minimize}} \quad x_1 + 2x_2 - x_3$$

$$x_1, x_2, x_3 \geq 0$$

$$x_1 + x_2 + x_3 = 1$$

We now solve that problem using the [Convex.jl](#) optimization toolbox.

```
In [2]: x = Variable{3}
constraints = [x >= 0, sum(x) == 1]
objective = x[1] + 2*x[2] - x[3]
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
@printf("Optimal point x = %s\n", x.value)
```

```
GLPK Simplex Optimizer, v4.64
5 rows, 4 columns, 10 non-zeros
0: obj = 0.000000000e+00 inf = 1.000e+00 (1)
2: obj = 0.000000000e+00 inf = 0.000e+00 (0)
* 4: obj = -1.000000000e+00 inf = 0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND

Optimal objective p = -1.000000
Optimal point x = [0.0; 0.0; 1.0]
```

Primal canonical form of a linear program

The standard form of a linear program is as follows, with $\vec{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$.

$$\underset{\vec{x} \in R^n}{\text{minimize}} \quad \vec{c}^T \vec{x}$$

$$A\vec{x} = \vec{b}$$

$$\vec{x} \geq 0$$

Activities Firefox Web Browser Mar 11 12:47

Step1_ConvexProgramming - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgramming Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step1_ConvexProgramming.ipynb

jupyter Step1_ConvexProgramming (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

$x_1 + x_2 + x_3 = 1$

We now solve that problem using the [Convex.jl](#) optimization toolbox.

```
In [2]: x = Variable{3}
constraints = [x >= 0, sum(x) == 1]
objective = x[1] + 2*x[2] - x[3]
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
@printf("Optimal point x = %s\n", x.value)
```

```
GLPK Simplex Optimizer, v4.64
5 rows, 4 columns, 10 non-zeros
0: obj = 0.000000000e+00 inf = 1.000e+00 (1)
2: obj = 0.000000000e+00 inf = 0.000e+00 (0)
* 4: obj = -1.000000000e+00 inf = 0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND

Optimal objective p = -1.000000
Optimal point x = [0.0; 0.0; 1.0]
```

Primal canonical form of a linear program

The standard form of a linear program is as follows, with $\vec{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$.

$$\begin{aligned} & \text{minimize} && \vec{c}^T \vec{x} \\ & \text{over} && \vec{x} \in \mathbb{R}^n \\ & && A\vec{x} = \vec{b} \\ & && \vec{x} \geq 0 \end{aligned}$$

The problem is fully specified by the matrix A and the vectors \vec{b} and \vec{c} .

Lecture exercise

Complete the program data below to solve (??).

```
In [ ]: using Convex
using GLPK
# remember [1, 1, 1] is a 1D vector, [1; 1; 1] is a column matrix, [1 1 1] is a row matrix
x = Variable{3}
c = []
A = []
```


Activities Firefox Web Browser Mar 11 12:56

Step1_ConvexProgramming - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgramming Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step1_ConvexProgramming.ipynb 133%

Jupyter Step1_ConvexProgramming (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run Code

Duality

We remind the primal canonical form (???):

$$p^* = \underset{\vec{x} \in \mathbb{R}^n}{\text{minimize}} \quad \vec{c}^\top \vec{x}, \quad A\vec{x} = \vec{b}, \quad \vec{x} \geq 0,$$

and introduce the notation

- a variable with a \star superscript is an optimal solution (as in p^\star is the minimum of the LP),
- a variable with a $*$ superscript is a feasible solution (as in \vec{x}^* satisfies the constraints),

with the conventions

- $p^\star = +\infty$ if the problem is infeasible,
- p^\star is finite if the problem has an optimal (thus feasible) solution,
- $p^\star = -\infty$ if the problem is unbounded, as there are feasible solutions with $\vec{c}^\top \vec{x} \rightarrow -\infty$.

We introduce the Lagrange multipliers $\vec{y} \in \mathbb{R}^m$ corresponding to the constraint $A\vec{x} = \vec{b}$, and the Lagrange multipliers $\vec{s} \in \mathbb{R}^n$ corresponding to the constraint $\vec{x} \geq 0$, with $\vec{s} \geq 0$ as they correspond to an inequality constraint. The Lagrangian function is:

$$L(\vec{x}, \vec{y}, \vec{s}) = \vec{c}^\top \vec{x} + \vec{y}^\top (\vec{b} - A\vec{x}) - \vec{s}^\top \vec{x}.$$

For any feasible \vec{x}^* and any (\vec{y}^*, \vec{s}^*) with $\vec{s} \geq 0$, we have

$$L(\vec{x}^*, \vec{y}^*, \vec{s}^*) = \vec{c}^\top \vec{x}^* + (\vec{y}^*)^\top \underbrace{(\vec{b} - A\vec{x}^*)}_{=0} - (\vec{s}^*)^\top \vec{x}^* \leq \vec{c}^\top \vec{x}^*$$

Let us now minimize $L(\vec{x}, \vec{y}, \vec{s})$ over \vec{x} :

Activities Document Viewer Mar 11 13:02 Step1_ConvexProgramming.pdf 172.8%

3 of 5

- Introduction ... 1
- Linear opti... 2
- Primal c... 2
- Duality 3
- Dual ca... 4
- Recover... 4

with the conventions

- $p^* = +\infty$ if the problem is infeasible,
- p^* is finite if the problem has an optimal (thus feasible) solution,
- $p^* = -\infty$ if the problem is unbounded, as there are feasible solutions with $\vec{c}^\top \vec{x} \rightarrow -\infty$.

We introduce the Lagrange multipliers $\vec{y} \in \mathbb{R}^m$ corresponding to the constraint $A\vec{x} = \vec{b}$, and the Lagrange multipliers $\vec{s} \in \mathbb{R}^n$ corresponding to the constraint $\vec{x} \geq 0$, with $\vec{s} \geq 0$ as they correspond to an inequality constraint. The Lagrangian function is:

$$L(\vec{x}, \vec{y}, \vec{s}) = \vec{c}^\top \vec{x} + \vec{y}^\top (\vec{b} - A\vec{x}) - \vec{s}^\top \vec{x}. \quad (6)$$

For any feasible \vec{x}^* and any (\vec{y}^*, \vec{s}^*) with $\vec{s} \geq 0$, we have

$$L(\vec{x}^*, \vec{y}^*, \vec{s}^*) = \vec{c}^\top \vec{x}^* + (\vec{y}^*)^\top \underbrace{(\vec{b} - A\vec{x}^*)}_{=0} - (\vec{s}^*)^\top \vec{x}^* \leq \vec{c}^\top \vec{x}^* \quad (7)$$

Let us now minimize $L(\vec{x}, \vec{y}, \vec{s})$ over \vec{x} :

$$g(\vec{y}, \vec{s}) = \min_{\vec{x}} \vec{x}^\top (\vec{c} - A^\top \vec{y} - \vec{s}) + \vec{b}^\top \vec{y} = \begin{cases} \vec{b}^\top \vec{y}, & \vec{c} - A^\top \vec{y} - \vec{s} = 0, \\ -\infty, & \text{otherwise.} \end{cases} \quad (8)$$

Note that for every (\vec{y}, \vec{s}) (with $\vec{s} \geq 0$ by definition), the value $g(\vec{y}, \vec{s})$ is a lower bound for p^* by (7).

In (8), only the first case ($\vec{c} - A^\top \vec{y} - \vec{s} = 0$) leads to a nontrivial upper bound. We now maximize over such (\vec{y}, \vec{s}) to get the best lower bound.

3

Activities

Document Viewer

4 of 5

Step1_ConvexProgramming.pdf

172.8%

Introduction ... 1

Linear opti... 2

Primal c... 2

Duality 3

Dual ca... 4

Recover... 4

3

1.2.1 Dual canonical form of a linear program

We obtain:

$$\begin{aligned}
 d^* = \text{maximize} \quad & \vec{b}^\top \vec{y} \\
 \text{over} \quad & \vec{y} \in \mathbb{R}^m, \vec{s} \in S^n \\
 & \vec{c} - A^\top \vec{y} = \vec{s} \\
 & \vec{s} \geq 0
 \end{aligned}
 \tag{9}$$

which is the dual problem. Through the Lagrangian mechanism, we have $d^* \leq p^*$ which is *weak duality*.

We have the cases, for the dual problem: - $d^* = +\infty$, dual unbounded, thus $+\infty \leq p^*$ and the primal problem is infeasible, - d^* is finite, the dual is feasible, the primal problem can either be feasible or infeasible (but not unbounded), - $d^* = -\infty$, the dual problem is infeasible.

For linear programs only, when d^* is finite, then $p^* = d^*$; when p^* is finite, also $d^* = p^*$ (this is called *strong duality*).

Lecture exercise

Fill the problem data below to solve the dual of (4).

```

[ ]: using Convex
      using GLPK
      y = Variable(1)
      s = Variable(3)

```

Pirsa: 20030032

Page 28/54

Activities

Document Viewer

4 of 5

Step1_ConvexProgramming.pdf

172.8%

Introduction ... 1

Linear opti... 2

Primal c... 2

Duality 3

Dual ca... 4

Recover... 4

over $y \in \mathbb{R}^n, s \in \mathbb{S}$

$\vec{c} - A^T \vec{y} = \vec{s}$

$\vec{s} \geq 0$

(9)

which is the dual problem. Through the Lagrangian mechanism, we have $d^* \leq p^*$ which is *weak duality*.

We have the cases, for the dual problem: $-d^* = +\infty$, dual unbounded, thus $+\infty \leq p^*$ and the primal problem is infeasible, $-d^*$ is finite, the dual is feasible, the primal problem can either be feasible or infeasible (but not unbounded), $-d^* = -\infty$, the dual problem is infeasible.

For linear programs only, when d^* is finite, then $p^* = d^*$; when p^* is finite, also $d^* = p^*$ (this is called *strong duality*).

Lecture exercise Fill the problem data below to solve the dual of (4).

```

[: using Convex
using GLPK
y = Variable(1)
s = Variable(3)
c = []
A = []
b = []
constraints = [s >= 0, s == c - transpose(A)*y]
objective = dot(b, y)
problem = Convex.maximize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL))
@printf("\n")
@printf("Optimal objective d = %f\n", problem.optval)
@printf("Optimal point    y = %s\n", y.value)
@printf("Optimal point    s = %s\n", s.value)l

```

1.2.2 Recovering dual variables using Convex.jl

Most convex solvers solve the primal and the dual problem at the same time, as doing so lead to better robustness (the primal-dual gap $p^* - d^*$ is then a measure of convergence).

Activities Firefox Web Browser Mar 11 13:09

Step1_ConvexProgramming - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgramming Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step1_ConvexProgramming.ipynb 133%

Jupyter Step1_ConvexProgramming (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help

Run

Optimal objective value = -1.000000
 Optimal point y = -1.0
 Optimal point s = [2.0; 3.0; 0.0]

Recovering dual variables using Convex.jl

Most convex solvers solve the primal and the dual problem at the same time, as doing so lead to better robustness (the primal-dual gap $p^* - d^*$ is then a measure of convergence).

Remember the primal problem:

$$p^* = \underset{\vec{x} \in R^n}{\text{minimize}} \quad \vec{c}^T \vec{x}, \quad A\vec{x} = \vec{b}, \quad \vec{x} \geq 0,$$

and realize that any feasible \vec{x}^* provides an objective p^* that is an upper bound on the true minimum by definition: $p^* \leq p^*$.

The same happens for the dual problem: any feasible pair (\vec{y}^*, \vec{s}^*) provides an objective d^* that is a lower bound on the true dual maximum.

Thus if we have a primal feasible solution \vec{x}^* and a dual feasible solution (\vec{y}^*, \vec{s}^*) , we sandwich the true solution $d^* = p^*$:

$$d^* \leq d^* = p^* \leq p^*.$$

Linear programming solvers try to return a feasible solution pair for both the primal and dual, with primal objective \tilde{p}^* and dual objective \tilde{d}^* . It would be tempting to imagine that the true solution lies in the internal $[\tilde{d}^*, \tilde{p}^*]$. However the primal and dual solutions are usually *slightly* infeasible due to numerical errors, and the relation $(???)$ only holds approximately. Worse, it's difficult to assess the impact of such numerical errors just by looking at the problem data.

Below, we provide the primal problem to Convex.jl, and recover the dual variables from the solution. Compare and contrast with the dual formulation above.

```
In [ ]: using Convex
using GLPK
x = Variable{3}
c = []
A = []
```

$$A\vec{x} = \vec{b}$$

$$\begin{cases} 1 \cdot x_1 + 1 \cdot x_2 = 1 \\ (1+\varepsilon)x_1 + 1 \cdot x_2 = 1 \\ 1 \cdot x_1 + (1+\varepsilon)x_2 = 1 \end{cases}$$

$\varepsilon > 0 \Rightarrow$ system is inconsistent

$$A\vec{x} = \vec{b}$$

$$\begin{cases} 1 \cdot x_1 + 1 \cdot x_2 = 1 \\ (1+\varepsilon)x_1 + 1 \cdot x_2 = 1 \\ 1 \cdot x_1 + (1+\varepsilon)x_2 = 1 \end{cases}$$

$\varepsilon > 0 \Rightarrow$ system is inconsistent

$$A\vec{x} = \vec{b}$$

$$\begin{cases} 1 \cdot x_1 + 1 \cdot x_2 = 1 \\ (1+\varepsilon)x_1 + 1 \cdot x_2 = 1 \\ 1 \cdot x_1 + (1+\varepsilon)x_2 = 1 \end{cases}$$

$\varepsilon > 0 \Rightarrow$ system is inconsistent

$$\textcircled{A} \bar{x} = \bar{b}$$

$$\begin{cases} 1 \cdot x_1 + 1 x_2 = 1 \\ (1+\varepsilon) x_1 + 1 x_2 = 1 \\ 1 x_1 + (1+\varepsilon) x_2 = 1 \end{cases}$$

$\varepsilon > 0 \Rightarrow$ system is inconsistent

$$\textcircled{A} \bar{x} = \bar{b}$$

$$\left\{ \begin{array}{l} 1 \cdot x_1 + 1 x_2 = 1 \\ (1+\varepsilon) x_1 + 1 x_2 = 1 \\ 1 x_1 + (1+\varepsilon) x_2 = 1 \end{array} \right.$$

$\varepsilon > 0 \Rightarrow$ system is inconsistent

Activities Firefox Web Browser Mar 11 13:16 en

Step1_ConvexProgramming - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgramming Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step1_ConvexProgramming.ipynb 133%

Jupyter Step1_ConvexProgramming (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

$$a^* \leq a^{**} = p^* \leq p^*.$$

Linear programming solvers try to return a feasible solution pair for both the primal and dual, with primal objective \bar{p}^* and dual objective \bar{d}^* . It would be tempting to imagine that the true solution lies in the interval $[\bar{d}^*, \bar{p}^*]$. However the primal and dual solutions are usually *slightly* infeasible due to numerical errors, and the relation (???) only holds approximately. Worse, it's difficult to assess the impact of such numerical errors just by looking at the problem data.

Below, we provide the primal problem to Convex.jl, and recover the dual variables from the solution. Compare and contrast with the dual formulation above.

```
In [ ]: using Convex
using GLPK
x = Variable{3}
c = []
A = []
b = []
constraints = [A*x == b, x >= 0]
objective = dot(c, x)
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL))
@printf("\n")
@printf("Optimal objective d = %f\n", problem.optval)
@printf("Optimal point x = %s\n", x.value)
@printf("Dual variable 1 %s\n", problem.constraints[1].dual)
@printf("Dual variable 2 %s\n", problem.constraints[2].dual)
```

In []:

$$A\vec{x} = \vec{b}$$

$$x_1 + 1x_2 = 1$$

$$x_1 + 1x_2 = 1$$

$$x_1 + (1+\varepsilon)x_2 = 1$$

\Rightarrow System is inconsistent

Convex

$$\begin{cases} \min f(x) \xrightarrow{\text{convex (Today)}} \\ \text{s.t. } g_i(x) \leq 0 \xrightarrow{\text{convex (Today)}} \\ \quad h_j(x) = 0 \end{cases}$$

linear

$$\begin{cases} \min \vec{c}^T \vec{x} \\ \text{s.t. } \vec{x} \geq 0 \\ A\vec{x} = \vec{b} \end{cases}$$

$$\vec{a}^T \vec{x} \leq \beta$$

$$\beta - \vec{a}^T \vec{x} = t$$

$$t \geq 0$$

LP



Convex

$$\begin{cases} \min f(x) \leftarrow \text{convex (Today)} \\ \text{s.t. } g_i(x) \leq 0 \leftarrow \text{convex (Today)} \\ h_j(x) = 0 \end{cases}$$

linear

$$\begin{cases} \min & \vec{c}^T \vec{x} \\ \text{s.t.} & \vec{x} \geq 0 \\ & A\vec{x} = \vec{b} \end{cases}$$

$$\vec{\alpha}^T \vec{x} \leq \beta$$

$$\beta - \vec{\alpha}^T \vec{x} = t \quad t \geq 0$$

Mosek

$$LP \subseteq SOCP \subseteq SDP \subseteq \text{SDP} + \text{exp} + \text{power}$$

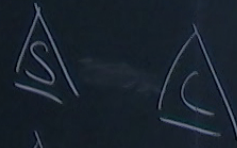
$$\vec{x} \leq t \quad \text{eig} \begin{pmatrix} x_1 & x_2 \\ x_2 & x_3 \end{pmatrix} \geq 0$$

$$\vec{x} \in \left\{ \vec{x} \mid \begin{pmatrix} x_1 & x_2 \\ x_2 & x_3 \end{pmatrix} \right\}$$

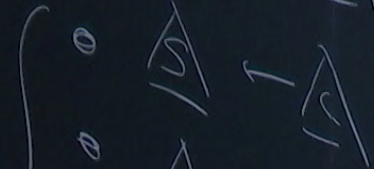
Conv

S Smoking

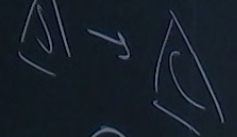
C Cancer



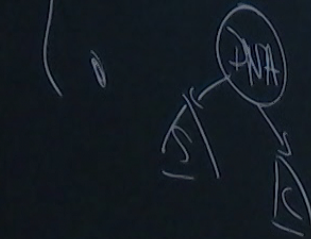
$$P_{sc}(s,c) = P_S(s) P_C(c)$$



$$P_{sc}(sc) = P_{S|C}(s|c) P_C(c)$$



$$P_{sc}(sc) = P_{C|S}(c|s) P_S(s)$$



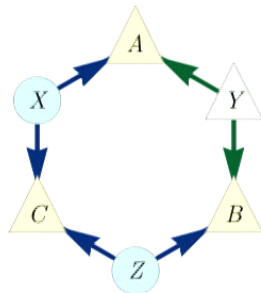
$$P_{sc}(sc) = \sum_d P_D(d) P_{C|D}(c|d) P_{S|D}(s|d)$$

[illegible]

we need.

causal structure

g causal structure:



scen15DAG.svg

the variables A , B and C that take the values a, b, c :

$$P_{ABC}(a, b, c) = \begin{cases} 1/2, & \text{if } a = b = c, \\ 0, & \text{otherwise.} \end{cases}$$

and whether this perfectly correlated distribution can be explained by a causal structure where A , B and C only depend on information that is shared between them.

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

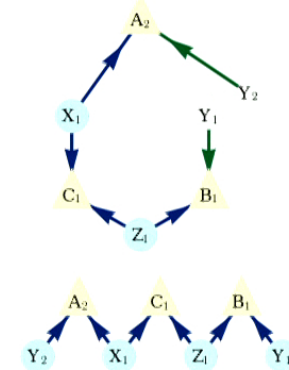
jupyter Step2_CausalStructures

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run C Markdown

Test using the "inflation technique" which maps to LP

We will use another method, amenable to linear programming. We will make a (numerical) proof by contradiction. Assume that P_{ABC} has a model of the form $(???)$. Then, we can imagine a variation on that model, where we duplicate the variable Y , and wire the relations between the variables a bit differently.



This is called an *inflated* scenario. There, we obtain the slightly different correlations:

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z_1) P_{A_2 | X_1 Y_2}(a_2 | x_1, y_2)$$

Activities Firefox Web Browser Mar 11 13:33

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

class-2020-03-11/ Step1_ConvexProgramm Step2_CausalStructures

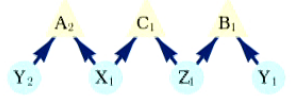
localhost:8888/notebooks/class-2020-03-11/Step2_CausalStructures.ipynb 133%

Jupyter Step2_CausalStructures (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run

Markdown



This is called an *inflated* scenario. There, we obtain the slightly different correlations:

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) \stackrel{!}{=} \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z_1) P_{A_2|X_1 Y_2}(a_2|x_1, y_2) P_{B_1|Y_1 Z_1}(b_1|y_1, z_1) P_{C_1|X_1 Z_1}(c_1|x_1, z_1).$$

Note that, however, the marginal distribution of the inflated correlations

$$P_{A_2 C_1} = \sum_{b_1} P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_2 z} P_{X_1}(x_1) P_{Y_2}(y_2) P_{Z_1}(z_1) P_{A_2|X_1 Y_2}(a_2|x_1, y_2) P_{C_1|X_1 Z_1}(c_1|x_1, z_1)$$

has the same form as the marginal distribution of the original scenario

$$P_{AC}(a, c) = \sum_b P_{ABC}(a, b, c) = \sum_{xy z} P_X(x) P_Y(y) P_Z(z) P_{A|XY}(a|x, y) P_{C|XZ}(c|x, z).$$


We thus have

$$P_{AC}(l, k) = P_{A_2 C_1}(l, k), \quad \forall l, k.$$

The same argument holds for

$$P_{BC}(j, k) = P_{B_1 C_1}(j, k), \quad \forall j, k.$$

Now, let us examine $P_{A_2 B_1}(a_2, b_1) = \sum_{c_1} P_{A_2 B_1 C_1}(a_2, b_1, c_1)$. It corresponds, after removal of C_1 , to the graph:



Activities Document Viewer

Step2_CausalStructures.pdf 231.5%

Application t... 1
 An exampl... 1
 A causal... 1
 Test usi... 2
 The linear ... 3

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z_1) P_{A_2}(a_2) P_{B_1}(b_1) P_{C_1}(c_1)$$

Note that, however, the marginal

$$P_{A_2 C_1} = \sum_{b_1} P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z_1) P_{A_2}(a_2) P_{C_1}(c_1)$$

has the same form as the marginal

$$P_{AC}(a, c) = \sum_b P_{ABC}(a, b, c)$$

We thus have

The same argument holds for

$$P_{BC}(b, c) = \sum_a P_{ABC}(a, b, c)$$

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

Home Page - Select or c... class-2020-03-11/ Step1_ConvexProgram Step2_CausalStructures

localhost:8888/notebooks/class-2 133%

jupyter Step2_CausalStructures Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

Now, let us examine $P_{A_2 B_1}(a_2, b_1) = \sum_{c_1} P_{A_2 B_1 C_1}(a_2, b_1, c_1)$. It corresponds, after removal of C_1 , to the graph:

where the variables A_2 and B_1 are independent. We thus have:

$$P_{A_2 B_1}(a_2, b_1) = P_{A_2}(a_2) P_{B_1}(b_1)$$

which we can now match with the original problem:

$$P_{A_2 B_1}(i, j) = P_A(i) P_B(j), \quad \forall i, j.$$

The linear program

Writing all these constraints together, we have ($\forall i, j, k$ is implicit):

$$\begin{aligned} \sum_j P_{A_2 B_1 C_1}(i, j, k) &= \sum_j P_{ABC}(i, j, k), \\ \sum_i P_{A_2 B_1 C_1}(i, j, k) &= \sum_i P_{ABC}(i, j, k), \\ \sum_k P_{A_2 B_1 C_1}(i, j, k) &= P_A(i) P_B(j), \\ P_{A_2 B_1 C_1}(i, j, k) &\geq 0 \end{aligned}$$

on that model, where we duplicate it a bit differently.

This is called an *inflated* scenario.

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z) P_{A_2}(a_2 | x_1, z) P_{B_1}(b_1 | y_1, z) P_{C_1}(c_1 | x_1, y_1, z)$$

Note that, however, the marginal

$$P_{A_2 C_1} = \sum_{b_1} P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_2 z} P_{X_1}(x_1) P_{Y_2}(y_2) P_{Z_1}(z) P_{A_2}(a_2 | x_1, z) P_{C_1}(c_1 | x_1, y_2, z)$$

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new notebook

class-2020-03-11/ Step1_ConvexProgram Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step2_CausalStructures

jupyter Step2_CausalStructures

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1 Logout

Now, let us examine $P_{A_2 B_1}(a_2, b_1) = \sum_{c_1} P_{A_2 B_1 C_1}(a_2, b_1, c_1)$. It corresponds, after removal of C_1 , to the graph:

where the variables A_2 and B_1 are independent. We thus have:

$$P_{A_2 B_1}(a_2, b_1) = P_{A_2}(a_2) P_{B_1}(b_1)$$

which we can now match with the original problem:

$$P_{A_2 B_1}(i, j) = P_A(i) P_B(j), \quad \forall i, j.$$

The linear program

Writing all these constraints together, we have ($\forall i, j, k$ is implicit):

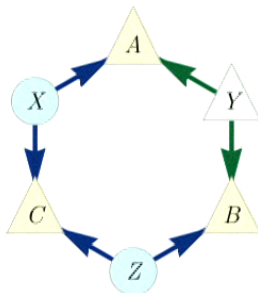
$$\begin{aligned} \sum_j P_{A_2 B_1 C_1}(i, j, k) &= \sum_j P_{ABC}(i, j, k), \\ \sum_i P_{A_2 B_1 C_1}(i, j, k) &= \sum_i P_{ABC}(i, j, k), \\ \sum_k P_{A_2 B_1 C_1}(i, j, k) &= P_A(i) P_B(j), \\ P_{A_2 B_1 C_1}(i, j, k) &\geq 0 \end{aligned}$$

1 structure discovery

Read in particular Section I, section II can be skimmed. Understood by reading III.B; we'll use Example 1 in our proof in Example 1, we'll formulate it as a linear program. Through the technical definitions, pay attention to as detailed in Section IV, particularly IV.B. Skip the rest.

structure

structure:



scen15DAG.svg

The linear program

Writing all these constraints together, we have ($\forall i, j, k$ is implicit):

$$\begin{aligned} \sum_j P_{A_2 B_1 C_1}(i, j, k) &= \sum_j P_{ABC}(i, j, k), \\ \sum_i P_{A_2 B_1 C_1}(i, j, k) &= \sum_i P_{ABC}(i, j, k), \\ \sum_k P_{A_2 B_1 C_1}(i, j, k) &= P_A(i)P_B(j), \\ P_{A_2 B_1 C_1}(i, j, k) &\geq 0 \end{aligned}$$

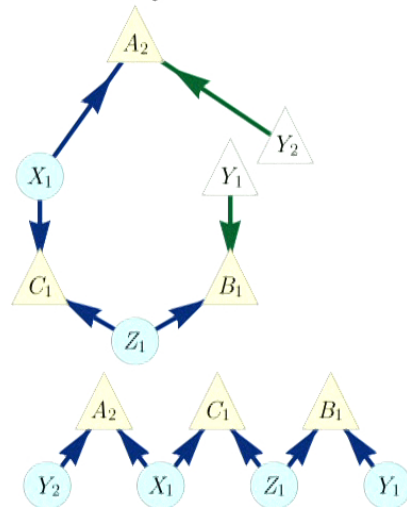
Now, the inflated correlations $P_{A_2 B_1 C_1}$ may obey additional constraints, but we remark that the constraints listed above correspond to a linear program in the primal form: indeed, the right-hand side of the equations are constant values that depend only on the coefficients $P_{ABC}(i, j, k)$ which are known.

$$\begin{aligned} &\text{minimize} && 0 \\ &\text{over} && \vec{v} \in \mathbb{R}^n \\ &&& M\vec{v} = \vec{b} \\ &&& \vec{v} \geq 0 \end{aligned}$$

where the objective is trivial, the constraint right-hand side \vec{b} is the only part of the problem that depends on P_{ABC} , and the matrix M only depends on the problem structure (matching the marginals).

Now, if this linear program is infeasible, it proves by contradiction that no model exists for the original problem (because original problem has model \Rightarrow inflation).

We will use another method, amenable to by contradiction. Assume that P_{ABC} has a on that model, where we duplicate the va bit differently.



This is called an *inflated* scenario. There

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z)$$

Note that, however, the marginal distr

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

Home Page - Select or class-2020-03-11/ Step1_ConvexProgram Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/ Step2_CausalStructures 133%

Jupyter Step2_CausalStructures Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

$$M\vec{v} = \vec{b}$$

$$\vec{v} \geq 0$$

where the objective is trivial, the constraint right-hand side \vec{b} is the only part of the problem that depends on P_{ABC} , and the matrix M only depends on the problem structure (matching the marginals).

Now, if this linear program is infeasible, it proves by contradiction that no model exists for the original problem (because original problem has a model \Rightarrow inflation has a model).

Homework 1

Write the linear program using Convex.jl, and verify if the distribution P_{ABC} is compatible with the inflation (hint: it should not).

You can use the numerically better behaved variant that has the slack variable z :

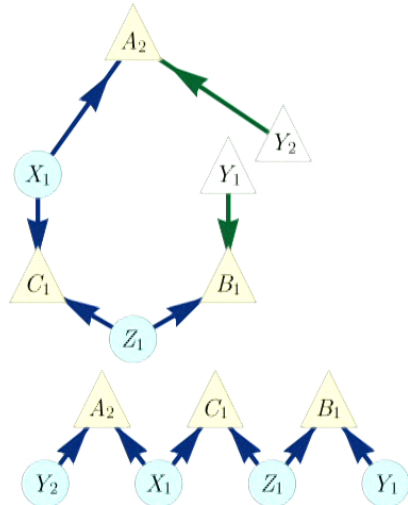
$$\begin{aligned} &\text{maximize} && z \\ &\text{over} && z \in \mathbb{R}, \vec{v} \in \mathbb{R}^n \\ &&& M\vec{v} = \vec{b} \\ &&& \vec{v} \geq z \end{aligned}$$

Homework 2

For which values of t the following distribution is compatible with the inflated model?

$$f(a,b,c) = \frac{1}{2}t \quad \text{if } a = b = c$$

We will use another method, amenable to by contradiction. Assume that P_{ABC} has a on that model, where we duplicate the va bit differently.



This is called an *inflated* scenario. There

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z)$$

Note that, however, the marginal distr

Homework 3

Test the distribution:

$$P_{ABC}(a, b, c) = \begin{cases} 1/3, & \text{if } a + b + c = 1, \\ 0, & \text{otherwise.} \end{cases}$$

This distribution should be compatible with the inflation above; nevertheless it is not compatible with the causal structure we test (see Example 2 of the [paper](#)). Why is the linear program feasible then?

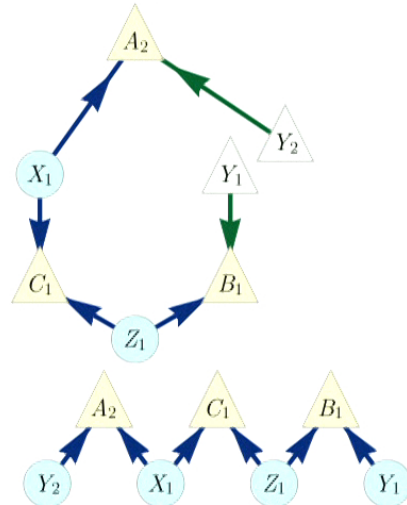
Homework 4

Solve one of the following questions:

- Consider the dual problem of (???). How to interpret the dual variables and the dual objective? Read the part about infeasibility certificates in the Mosek Cookbook, section 2.3. Can you derive a causal compatibility inequality as in Example 4 of the paper?
- Implement the Spiral inflation given in FIG. 3 of the paper, and verify that the distribution (???) is incompatible.

In []:

We will use another method, amenable to by contradiction. Assume that P_{ABC} has a on that model, where we duplicate the va bit differently.



This is called an *inflated* scenario. There

$$P_{A_2 B_1 C_1}(a_2, b_1, c_1) = \sum_{x_1 y_1 y_2 z} P_{X_1}(x_1) P_{Y_1}(y_1) P_{Y_2}(y_2) P_{Z_1}(z)$$

Note that, however, the marginal distr

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

Home Page - Select or class-2020-03-11/ Step1_ConvexProgram Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/ Step2_CausalStructures 133%

jupyter Step2_CausalStructures Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

Homework 3

Test the distribution:

$$P_{ABC}(a, b, c) = \begin{cases} 1/3, & \text{if } a + b + c = 1, \\ 0, & \text{otherwise.} \end{cases}$$

This distribution should be compatible with the inflation above; nevertheless it is not compatible with the causal structure we test (see Example 2 of the [paper](#)). Why is the linear program feasible then?

Homework 4

Solve one of the following questions:

- Consider the dual problem of (???). How to interpret the dual variables and the dual objective? Read the part about infeasibility certificates in the [Mosek Cookbook, section 2.3](#). Can you derive a causal compatibility inequality as in Example 4 of the [paper](#)?
- Implement the Spiral inflation given in FIG. 3 of the [paper](#), and verify that the distribution (???) is incompatible.

In []:

Activities Firefox Web Browser Mar 11 13:40

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgram Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step2_CausalStructures.ipynb 133%

Jupyter Step2_CausalStructures (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

Run

over $z \in \mathbb{R}, \vec{v} \in \mathbb{R}^n$

$$M\vec{v} = \vec{b}$$

$$\vec{v} \geq z$$

Homework 2

For which values of t the following distribution is compatible with the inflated model?

$$P_{ABC}(a, b, c) = \begin{cases} 1/2t, & \text{if } a = b = c, \\ (1-t)/6, & \text{otherwise.} \end{cases}$$

Homework 3

Test the distribution:

$$P_{ABC}(a, b, c) = \begin{cases} 1/3, & \text{if } a + b + c = 1, \\ 0, & \text{otherwise.} \end{cases}$$

This distribution should be compatible with the inflation above; nevertheless it is not compatible with the causal structure we test (see Example 2 of the [paper](#)). Why is the linear program feasible then?

Homework 4

Solve one of the following questions:

- Consider the dual problem of (???). How to interpret the dual variables and the dual objective? Read the part about infeasibility certificates in the [Mosek Cookbook, section 2.3](#). Can you derive a causal compatibility inequality as in Example 4 of the [paper](#)?
- Implement the Spiral inflation given in FIG. 3 of the [paper](#), and verify that the distribution (???) is incompatible.

$$\vec{v} = \vec{P}_{inf} = \begin{pmatrix} P_{A_2 B_1 C_1}(000) \\ P_{A_2 B_1 C_1}(100) \\ \vdots \\ P_{A_2 B_1 C_1}(001) \\ P_{A_2 B_1 C_1}(101) \\ \vdots \\ P_{A_2 B_1 C_1}(111) \end{pmatrix} \in \mathbb{R}^8$$

$$\vec{P}_{inf} \geq 0 \quad (11111111) \vec{P}_{inf} = 1$$

$$V_1 + V_3 = \begin{cases} P_{A_2 C_1}(ij) = P_{A_1 C_1}(ij) \end{cases}$$

$$P_{A_2 B_1 C_1}(000) + P_{A_2 B_1 C_1}(001) = P_{A_2 C_1}(00) = P_{A_1 C_1}(00)$$

$$(10100000) \cdot \vec{P}_{inf} = \begin{cases} P_{A_2 C_1}(10) \\ P_{A_2 C_1}(01) \\ P_{A_2 C_1}(11) \end{cases}$$

Activities Firefox Web Browser Mar 11 13:55

Step2_CausalStructures - Jupyter Notebook - Mozilla Firefox

Home Page - Select or create a new tab class-2020-03-11/ Step1_ConvexProgram Step2_CausalStructures

localhost:8888/notebooks/class-2020-03-11/Step2_CausalStructures.ipynb 133%

Jupyter Step2_CausalStructures (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 1.3.1

problem structure (matching the marginals).

Now, if this linear program is infeasible, it proves by contradiction that no model exists for the original problem (because original problem has model => inflation has a model).

Homework 1

Write the linear program using Convex.jl, and verify if the distribution P_{ABC} is compatible with the inflation (hint: it should not).

You can use the numerically better behaved variant that has the slack variable z :

$$\begin{aligned} &\text{maximize} && z \\ &\text{over} && z \in \mathbb{R}, \vec{v} \in \mathbb{R}^n \\ &&& M\vec{v} = \vec{b} \\ &&& \vec{v} \geq z \end{aligned}$$

Homework 2

For which values of t the following distribution is compatible with the inflated model?

$$P_{ABC}(a, b, c) = \begin{cases} 1/2t, & \text{if } a = b = c, \\ (1 - t)/6, & \text{otherwise.} \end{cases}$$

Homework 3

Test the distribution:

$$P_{ABC}(a, b, c) = \begin{cases} 1/3, & \text{if } a + b + c = 1, \\ 0, & \text{otherwise.} \end{cases}$$