Modeling Convex Optimization Problems

CVX and CVXOPT

Vishal Gupta

Jan 31, 2013

Outline

- CVX Basics
  - What is CVX?
  - Convexity and DCP Convexity
- Advanced CVX
  - Dual variables
  - SDPs, GPs and MICPs
  - Solver settings
- CVXPY and CVX_OPT
  - CVXPY (brief)
  - Modeling language vs. solver
  - CVXOPT Basic Usage and documentation
  - Specializing Linear Algebra (time permitting)
- Course Wrap-up

Full Disclosure

- I ***strongly dislike*** Matlab
  - Hangs/becomes unstable
  - Many Coding conventions are frustrating (namespaces?)
  - Third-party libraries (when they exist) can be buggy
  - Expensive, commercial software

- *Why am I teaching CVX (a Matlab based software)?*

In my experience, CVX is the fastest way to prototype a convex optimization problem.

Outline

- CVX Basics
  - What is CVX?
  - Convexity and DCP Convexity
- Advanced CVX
  - Dual variables
  - SDPs, GPs and MICPs
  - Solver settings
- CVXPY and CVX_OPT
  - CVXPY (brief)
  - Modeling language vs. solver
  - CVXOPT Basic Usage and documentation
  - Specializing Linear Algebra (time permitting)
- Course Wrap-up

What is CVX?

- CVX is a Matlab based language for modeling convex optimization problems
  - Originally academic, now commercial (http://cvxr.com/)
  - Very intuitive, well-documented
  - Variety of back-end solvers (SeDumi, SDPT3, MOSEK, GUROBI)

  - Well suited for:
    - Exploring small / medium problems
    - Comparing formulations
    - Prototyping optimizations in research
    - Coding small MILPs for Gurobi in MATLAB

What is CVX ***NOT***?

- CVX is NOT
  - A solver
  - Tool to test convexity
  - Particularly clear with error messages (as of writing)

- (In my opinion), CVX is not well-suited for:
  - Limited support for mixed-integer convex problems… (Not yet mature)
  - Solving very large instances
  - Solving many, many small instances (as in simulation)

Learn by doing…

- The best way to learn CVX by examples
  - Second best: Consult documentation
  - http://cvxr.com/cvx/doc/
- Please open “testCVX.m” and run it.
- Questions:
  - What is this optimization doing?
  - What is the dimension of x? The value of x_1?
  - How many non-zero values of x are there?

CVX Basics

- Problem surrounded by cvx_begin … cvx_end
  - cvx_begin quiet … cvx_end will suppress output
- Variables are declared first
  - Variable x(n) ~ indicates a vector
  - Variable y(m, n) ~ an n by m matrix
  - Variables x(n) y(m, n) z ~ a vector, a matrix and a scalar!
  - We'll see other possibilities later
- After cvx_end, automatically solves…
  - cvx_optval contains optimal value
  - Variables are replaced with their optimal values

Your turn…

The previous least-squares problem was badly conditioned.

- Modify the optimization to solve a regularized problem

$$\min \|\mathbf{r}\|_2 + \lambda \|\mathbf{x}\|_1$$

for $\lambda = 1$

- Embed this optimization into a script to solve the above for $\lambda = 2^{-3}, 2^{-2.5}, \ldots, 2^{-3}$. Plot the trade-off curve. (Hint: Use a "for" loop. Suppress output in the loop.)

- Challenge Questions:
  - Is CVX white-space sensitive?
  - How would you incorporate the constraints

$$-10 \leq x_i \leq 10 \quad i = 1, \ldots, n$$

Summary of the basics

- Wrap optimization in cvx_begin, cvx_end
- Write functions and constraints as you would write them in Matlab
- Automatically attempts to solve, and replaces variables with their optimal values
- Can embed within Matlab scripts

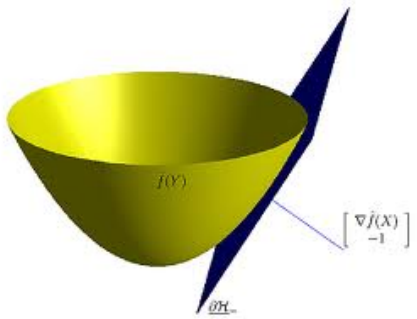
Review: Convexity

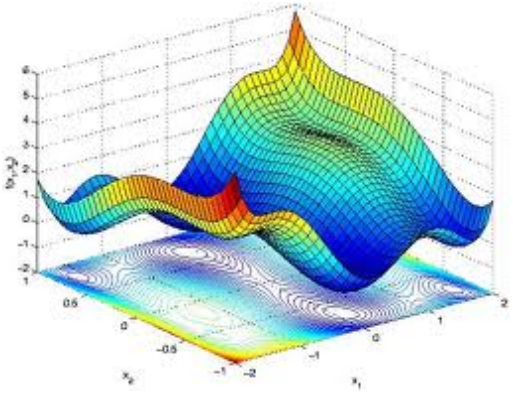
- Recall the definition of convexity:

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \quad \forall \lambda \in [0,1]$$

- Intuitively, convex functions are like bowls (not hats)

- Convex functions
  - $\mathbf{a}^T\mathbf{x} + b$
  - e^x, e^{-x}
  - $\|\mathbf{x}\|$
  - $\mathbf{x}^T\mathbf{Q}\mathbf{x}, \quad \mathbf{Q} \succeq \mathbf{0}$
- Non-convex functions
  - $\mathbf{x}^T\mathbf{Q}\mathbf{x}, \quad \mathbf{Q}$ indefinite
  - $\log x, \ \sqrt{x}$





Review: Convexity II

- Combining convex functions in certain ways preserves their convexity:
  - f(x) convex implies g(x) = f(Ax + b) is convex.
  - f(x) concave implies g(x) = -f(x) is convex
  - $f_1(x)$, $f_2(x)$ convex implies $g(x) = c_1 f_1(x) + c_2 f_2(x)$ is convex if c_1, $c_2 > 0$
  - See *Convex Optimization* by Boyd and Vandenberghe for advanced examples

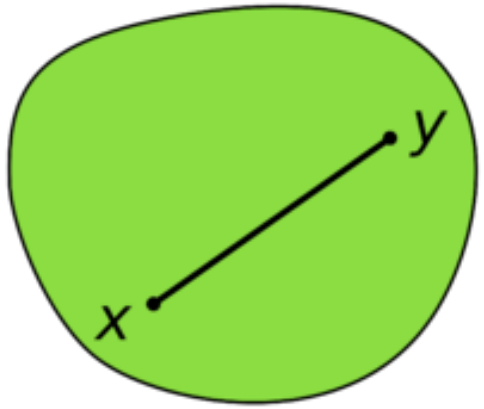
Convex Sets

- Recall the definition of a convex set

$$\mathbf{x} \in S, \mathbf{y} \in S \Rightarrow \lambda\mathbf{x} + (1-\lambda)\mathbf{y} \in S$$

- Intuitively, convex sets are “round”

- Convex sets:
  - $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$
  - $\mathbb{R}^n_+$
  - $\{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}\}$
  - $\{(\mathbf{x}, t) : \|\mathbf{x}\| \leq t\}$



Images taken from Wikipedia: http://en.wikipedia.org/wiki/Convex_set

Convex Optimization Problems

- A minimization problem with a convex objective function and convex feasible region is called a ***convex optimization***

- Otherwise, we say it is ***non-convex***

Disciplined Convex Programming (DCP)

- ***CVX does NOT model convex optimization***
  - Popular misconception
  - CVX models a subset of convex optimizations

- Specifically, CVX models problems obeying Disciplined Convex Programming (Grant, Boyd and Ye)

> Disciplined convex programming imposes a set of convention or rules… Problems which adhere to the ruleset can [automatically] converted to solvable forms. Problem which violate the ruleset [even if they are convex] are rejected.
>
> Excerpted from http://cvxr.com/cvx/doc/intro.html

DCP… intuitively

- There are certain *atomic* functions that CVX knows are convex
  - $e^x, \mathbf{a}^T\mathbf{x} + b, x^4, \|\mathbf{x}\|, \mathbf{x}^T\mathbf{Q}\mathbf{x}\ (\mathbf{Q} \succeq \mathbf{0})$
  - See documentation for full list

- CVX also knows a set of rules for combining convex functions to build new convex functions (DCP Rule set)
  - $\|\mathbf{A}\mathbf{x} + \mathbf{b}\|$ (composition of convex and affine function)
  - See documentation for full list

DCP (continued)

- We will say that a function is “DCP-convex” if it is either atomic and convex, or can be constructed from the atomic functions by the DCP rule set

- Similar definitions apply to “DCP-concave”, “DCP-affine”, “DCP-non-decreasing”, “DCP-non-increasing”, etc.

Valid optimization problems

- A Valid DCP objective is either
  - $\min f(\mathbf{x})$ where f is DCP-convex
  - $\max f(\mathbf{x})$ where f is DCP concave

- Valid DCP constraints are of the form
  - $f(\mathbf{x}) \leq g(\mathbf{x})$ where f is DCP convex, g is DCP concave
  - $f(\mathbf{x}) = g(\mathbf{x})$ where f, g are DCP-affine
  - $x \in S$ where S is an atomic, convex set (more on this later)

- Not all convex optimizations are a valid DCP
- Almost all can be transformed to be a valid DCP

Examples:

- Constraint: $\|\mathbf{x}\|_2 \leq 1$
  - $\|\mathbf{x}\|_2$ is convex and DCP convex, 1 is DCP concave
  - This is a valid constraint

- Constraint: $\|\mathbf{x}\|_2 \geq 1$
  - $\|\mathbf{x}\|_2$ is not DCP concave. In fact, it's DCP convex (and convex).
  - This is an invalid constraint.

- Constraint: $\sqrt{x_1^2 + x_2^2} \leq 1$
  - $\sqrt{x_1^2 + x_2^2}$ is convex, but NOT DCP convex
  - Hence, this is an invalid constraint
  - Can be rewritten in DCP way

Error using cvx/sqrt (line 61)
Disciplined convex programming error:
Illegal operation: sqrt({convex})

Some practical advice….

- It might seem that you have to memorize all the DCP rules in order to know what you can and cannot model.

- Personally, I have not memorized the rules.
  - In my experience, most convex functions you come across in practice are DCP convex.
  - When they aren't, scanning the documentation for CVX for two minutes usually suggests the correct reformulation.
  - "Easier to ask forgiveness than to get permission."

Let's try some examples!

Your turn…Maximal entropy problems

Let X be random variable distributed on $\{1, 2, \ldots, 10\}$ such that

$$\mathbb{P}(X = i) = p_i \quad i = 1, \ldots, 10$$

The entropy of X, denoted H(X) is defined by:
(Entropy is a concave function of the p's).

$$H(X) = -\sum_{i=1}^{10} p_i \log p_i$$

Suppose we know that $\mathbb{E}[X] = 4$

Formulate an optimization problem to find the maximal entropy distribution with these properties.

Max Entropy II

$$\max_{\mathbf{p}} \quad -\sum_{i=1}^{10} p_i \log(p_i)$$

$$\text{s.t.} \quad \sum_{i=1}^{10} p_i = 1, \quad \mathbf{p} \geq \mathbf{0}$$

$$\sum_{i=1}^{10} i p_i = 4$$

1. Solve this problem in CVX.
2. Add the constraint that the standard deviation of X is at most 4. Resolve.
3. Read the output of the solver. Notice anything different?

Summary on DCP Convexity

- CVX only models DCP-convex problems
- Remember my advice
  - “Easier to ask forgiveness than permission”
- When in doubt, the documentation is very good:
  - http://cvxr.com/cvx/doc/index.html

Outline

- CVX Basics
  - What is CVX?
  - Convexity and DCP Convexity
- Advanced CVX
  - Dual variables
  - SDPs, GPs and MICP
  - Solver settings
- CVXPY and CVX_OPT
  - CVXPY (brief)
  - Modeling language vs. solver
  - CVXOPT Basic Usage and documentation
  - Specializing Linear Algebra (time permitting)
- Course Wrap-up

Dual Variables

- Possible to extract the dual variables for convex models.
  - Frequently no extra cost computational cost
  - Always non-negative for inequality constraints
- CVX automatically determines the dimensions of the dual
  - Error if you try to specify
  - Useful for matrix inequalities like x <= 1
- After solving, dual variables (like decision variables) populated with optimal values

Example 1 Constraint:

- Constraint our Max-Entropy Problem (maxEntropy.m)

```
%Constrain mean
supp * p == 4;
```

- With dual variable (See SimpleDual.m)

```
%Incorporate Dual variable
dual variable lambda
lambda : supp * p == 4;
```

Size of lambda implicitly determined to be 1

- Specifying dimension yields error

```
dual variable lambda(1)
```

Invalid dual variable specification: lambda(1)

Example: Vector Constraint

- Constraint from our Regularized Least Squares

```
r == y - A * x
```

- With dual variable:

```
dual variable lambda
lambda: r == y - A * x
```

Size of lambda implicitly determined to be m = 100

Constraints in a Loop I

- Consider following optimization (LoopedLeastSquares.m):

```
cvx_begin
  variables x(n) r(m)
    for i = 1:m
        r(i) == y(i) - A(i, :)*x
    end
    minimize norm(r, 2) + norm(x, 1)
cvx_end
```

- How do we incorporate dual variables now?

Constraints in a Loop II

- Use Matlab's cell array feature

```
cvx_begin
  variables x(n) r(m)
  dual variable p{m}
  for i = 1:m
    p{i} : r(i) == y(i) - A(i, :) * x
  end

  minimize norm(r, 2) + norm(x, 1)
cvx_end
```

- Simultaneously creates (m) dual variables
  - Notice the {} instead of the usual () notation
  - Only really necessary for looping constructs

Your turn … Max Entropy revisited

- Open the file “MomentProblems.m “
  - What is this code doing?
  - Is it possible to rewrite this optimization without the loop?

- Add dual variables to this optimization for:
  - The normalization constraint
  - Each of the moment constraints
  - Plot the dual variables for each moment

Max Entropy Solution

- See “MomentProblemsWithDual.m”

SDP mode

- You should have read a little bit about SDPs as part of your homework.

$$\begin{aligned} \min \quad & 3x_{11} - 2x_{23} \\ \text{s.t.} \quad & 5x_{11} + 2x_{23} - x_{33} = 1 \\ & 2x_{22} + 3x_{33} \leq 2 \\ & \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{12} & x_{22} & x_{23} \\ x_{13} & x_{23} & x_{33} \end{pmatrix} \succeq \mathbf{0} \end{aligned}$$

- Recall:
  - Optimization over matrices
  - Objective and constraints are *linear* functions of the *entries*
  - Matrix is constrained to be *symmetric* and *positive semidefinite*

Take my word for it…

- SDPs are extremely common and very useful…
  - Applications to probability, control theory, combinatorial optimization
  - Multiple courses at MIT that cover their usage

- For today, just take my word that if you work long enough in optimization, eventually you will want to solve an SDP

Your first SDP in CVX

- (See SimpleSDP.m)

$$
\begin{aligned}
\min \quad & 3x_{11} - 2x_{23} \\
\text{s.t.} \quad & 5x_{11} + 2x_{23} - x_{33} = 1 \\
& 2x_{22} + 3x_{33} \leq 2 \\
& \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{12} & x_{22} & x_{23} \\ x_{13} & x_{23} & x_{33} \end{pmatrix} \succeq \mathbf{0}
\end{aligned}
$$

```
cvx_begin
    variable X(3, 3)

    5 * X(1,1) + 2 * X(2, 3) - X(3, 3) == 1
    2 * X(2, 2) + 3 * X(3, 3) <= 2
     X == semidefinite(3)

   minimize 3 * X(1, 1) - 2 * X(2, 3)
cvx_end
```

Other matrix operations

- $5x_{11} + 2x_{23} - x_{33} = 1 \;\Leftrightarrow\; \mathbf{C} \circ \mathbf{X} == 1, \quad \mathbf{C} = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}$
  - `trace(C * X) == 1`
  - `vec(C)' * vec(X) == 1` (my preference… why?)

- $x_{11} + x_{22} + x_{33} = 1 \;\Leftrightarrow\; \mathrm{tr}(\mathbf{X}) = 2$
  - `trace(X) == 2`

Other Matrix Operations Ct'd

- $\begin{pmatrix} 1 & \mathbf{y}^T \\ \mathbf{y}^T & \mathbf{X} \end{pmatrix} \succeq \mathbf{0}$ (X and y are variables)

```
variables y(3, 1) X(3, 3)
variable Z(4, 4)
Z == semidefinite(4)
Z(1, 1) == 1
Z(2:end, 1) == y
Z(2:end, 2:end) == X
```

```
variables y(3, 1) X(3, 3)
[ 1, y';
  y, X ] == semidefinite(4)
```

Preferred method… why?

Your turn… Asset Correlation

- Consider three types of commodities futures: Oil, Natural Gas, and Electricity. (Index them 1, 2, 3)

- Suppose we know from the markets that

$$.6 \leq \rho_{12} \leq .8$$
$$.95 \leq \rho_{23} \leq .98$$

- And assume we know that

$$\sigma_1 = .6, \quad \sigma_2 = .4, \quad \sigma_3 = 1.2$$

- What is the value of ρ_{13} ?

Formulate as an SDP

- $$\begin{aligned} \min \quad & \rho_{13} \\ \text{s.t.} \quad & .6 \leq \rho_{12} \leq .8 \\ & .95 \leq \rho_{23} \leq .98 \\ & \begin{pmatrix} .6^2 & \rho_{12} * .6 * .4 & \rho_{13} * .6 * 1.2 \\ \rho_{12} * .6 * .4 & .4^2 & \rho_{23} * .4 * 1.2 \\ \rho_{13} * .6 * 1.2 & \rho_{23} * .4 * 1.2 & 1.2^2 \end{pmatrix} \succeq \mathbf{0} \end{aligned}$$

  (Other constraints here…)

- Code this up in CVX. What is optimal solution?
- Change your code to instead minimize

$$\min e^{\rho_{13}} + \|\rho_{12} - \rho_{23}\|$$

SDP Review

- Constraining a matrix to be PSD is a DCP-Convex constraint
  - There are other structured classes which are also DCP Convex (e.g. symmetric, hermitian, topelitz)
- We can combine SDP constraints with other DCP-convex constraints and objectives
- There is a separate “SDP mode” for CVX
  - Useful if you are coding up a large set of LMIs
  - Consult the documentation. Very straightforward.

Geometric programming

- GPs are a more advanced class of optimization problems
  - By a suitable transformation, they can be made DCP convex

- Check the documentation for details (GP mode)
  - GP tutorial
  - CVX documentation on GPs

Mixed Integer Programming

- It is possible to solve mixed integer convex problems
  - This is still a developing field
  - With non-Gurobi solvers, uses a proprietary branch and bound
  - Need to be careful in terms of computational power etc.

- For MILP, MIQP possible to use Gurobi as back-end
  - Requires a “professional” license (free for academics)
  - Set up is relatively easy. See the documentation.

- To use: Simply define variables with integer type

```
variable p(10) integer
variable z(10) binary
```

Fine-tuning Solver Settings

- Change the precision
  - old_precision = cvx_precision('high')
  - Possible choices: low, medium, default, high, best
- Change the solver
  - cvx_solver sedumi
  - Possible choices: sedumi, sdpt3, gurobi, mosek
  - Last two require separate installation and a (free) academic license
- Pass specific settings to solver
  - Badly documented
  - cvx_solver_settings(“method”, 1) // Dual simplex alg for Gurobi
  - Key-value pairs are solver dependent. Will throw if not supported.

Summary of Advanced Features

- Dual Variables
  - Size is implicitly determined
  - Use cell array {} for loops
- Semidefinite Programming
  - Membership to the semidefinite cone is DCP-convex
  - Many ways to write same constraints.
  - SDP mode may be useful to you
- Fine tuning solver settings
  - Generally shouldn’t need to alter these
  - Documentation is a little spotty. Guess and check is best bet.

Where do I learn more?

- CVX Documentation is your friend…
  - http://cvxr.com/cvx/doc/index.html
- There are a ton of examples that ship with the distribution
  - <path to CVX>/examples
  - Many of them paralell the textbook *Convex Optimization* by Boyd and Vanderberghe
- Experiment on your own!
  - Most things in CVX are intuitive.

Outline

- CVX Basics
  - What is CVX?
  - Convexity and DCP Convexity
- Advanced CVX
  - Dual variables
  - MIPs, SDPs, GPs
  - Solver settings
- CVXPY and CVX_OPT
  - CVXPY (brief)
  - Modeling language vs. solver
  - CVXOPT Basic Usage and documentation
  - Specializing Linear Algebra (time permitting)
- Course Wrap-up

Performance Bottlenecks… When not to use CVX?

- Recall our friend Lasso Regression

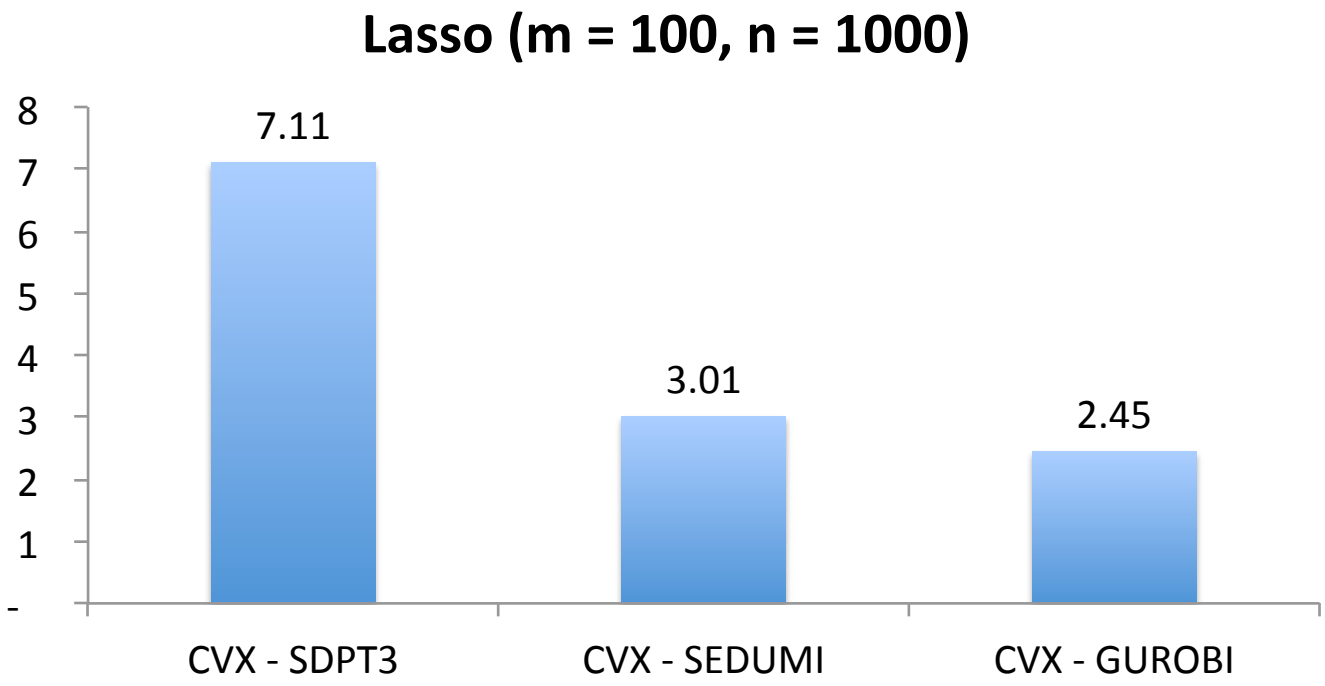
$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 + \lambda \|\mathbf{x}\|_2$$

  - Dim(A) = 100 by 1000, dense
  - $\lambda \in \{2^{-5}, 2^{-4.75}, \ldots, 2^{5}\}$
  - Sparse regression contexts

- Solving this problem in CVX takes about 7s per instance

- Is this slow? Why?

Could vary the solver…



Helps a little… Can we do better?

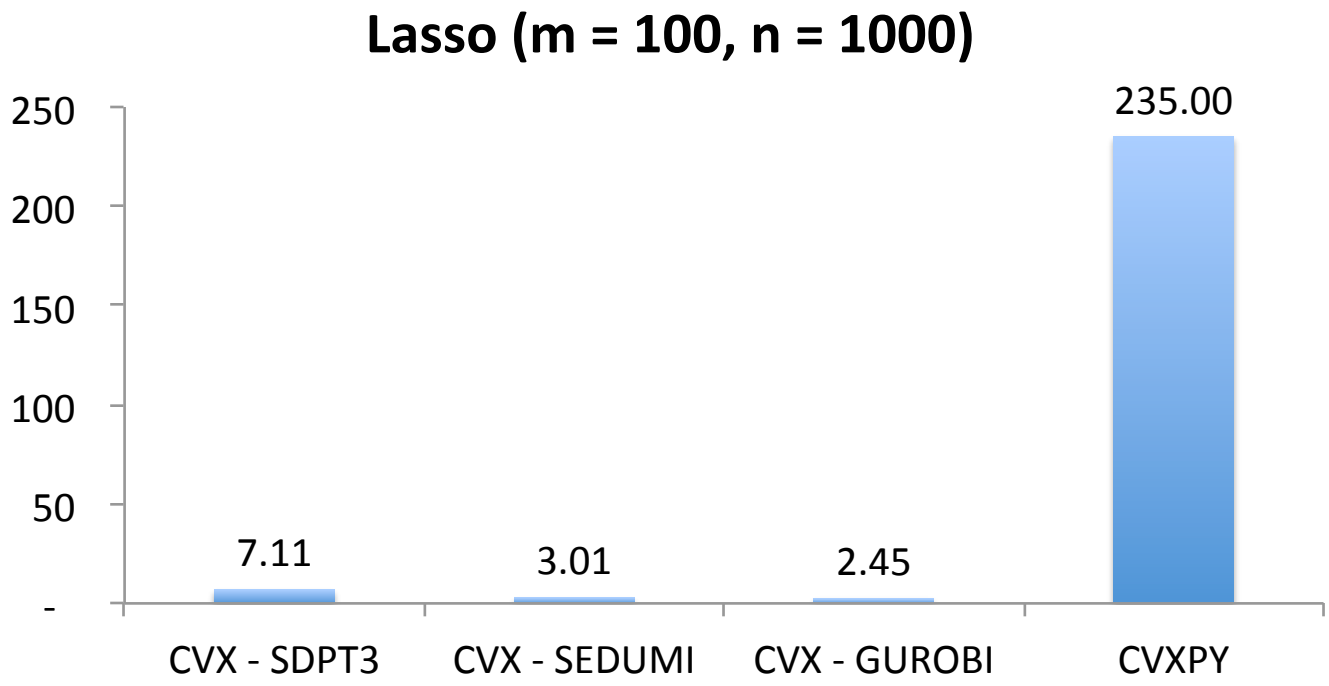
CVXPY

- Similar to CVX but implemented in Python
- Once you understand CVX, simple to learn CVXPY from documentation

- Some sample code (“cvxpy_timing.py”)

```
x = variable(n, 1, name="x")
obj = minimize( norm2( y - A * x) + norm1(x))
cnsts = []
p = program(obj, cnsts)
p.solve()
```

And the results…



What happened?

An important warning

- Slow code is slow no matter what language it is in…
- CVXPY was developed as academic software and then seemingly abandoned.
- CVX on the other hand is now sold commercially.
  - Has undergone multiple releases
  - At this point is highly optimized as a modeling language

From modeling language to solver

- A smarter approach: bypass the modeling language and go straight to the solver

- CVXOPT is a collection of solvers for conic programs,
  - LP, QP, GP and generic cone solvers
  - Sparse Matrix functionality, and high-performance linear algebra

- Challenge is that solvers expect you to specify your problem in ***standard form***
  - Converting your problem (on paper) can be tricky
  - Coding up the reformulation can be tedious

Example QP Solver

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

- This is the standard form cvxopt expects for QPs
- Solve by calling cvxopt.qp(P, c, G, h, A, b)
- Can frame our Lasso problem in this form?

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 + \lambda\|\mathbf{x}\|_2$$

Board Work


IT'S A PLEASURE TO MEEET YOU
I'M A PANDA
THE LOLBRARY.com/post/14062/

Solution

$$\text{“}P\text{”} = \begin{pmatrix} 2A^TA & 0 \\ 0 & 0 \end{pmatrix} \qquad \text{“}h\text{”} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\text{“}c\text{”} = \begin{pmatrix} -2A^Tb \\ \lambda e \end{pmatrix}$$

$$\text{“}G\text{”} = \begin{pmatrix} I & -I \\ -I & -I \end{pmatrix} \qquad \text{“}A\text{”} = \text{“}b\text{”} = 0$$

Coding this up in CVXOpt

- CVXOpt has *many* convenience for building up (sparse) matrices to pass to the optimizers

$$\text{matrix}(1, (2,3)) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Repeat a value in given shape

$$\text{matrix}([\ [1,2,3],[4,5,6]\ ]) = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

List of Lists Populated by Column

$$\text{matrix}([\ [A,1],[0,B]\ ]) = \begin{pmatrix} A & 0 \\ 1 & B \end{pmatrix}$$

Works for Block Matrices, too

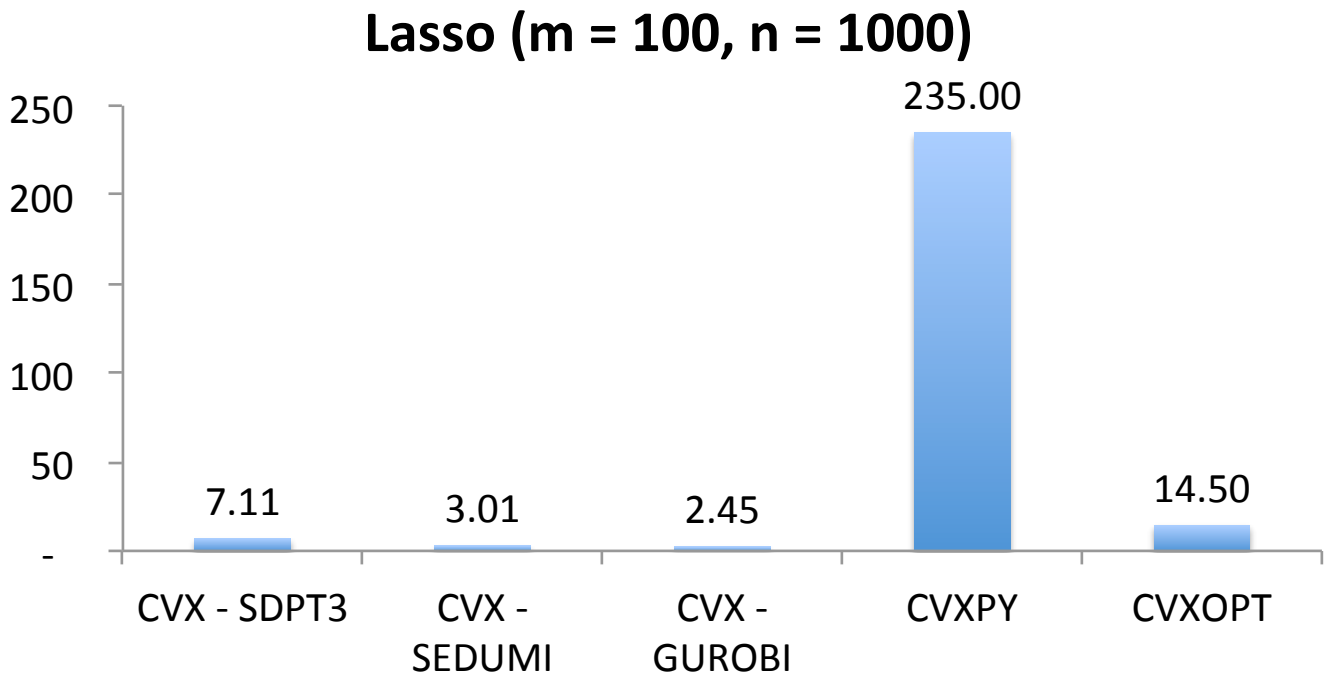
$$\text{spdiag}([A,B,1]) = \begin{pmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Can also create sparse block-diagonal matrices

Let's (together) code up our Lasso problem

- "cvxopt_timing.py"

And the results….



- Certainly better than CVXPy
- But the CVXOPT cone solvers are not quite as good as others

Customizing Linear Algebra

- How does Newton's method work? Where is the computational burden?
  - Interior point solvers have a similar problem. Need to take solve a large linear system from KKT conditions

- Often, if your problem has special structure, you can customize the linear algebra to solve this system
  - General consensus is that this is tedious, but the benefits are enormous
  - The paper by Andersen, et.al on website some examples and explains the background mathematics

Example…

- For the case of lasso regression, the KKT equations look like:

$$\begin{pmatrix} A^T A + W_1^{-2} + W_2^{-2} & W_2^{-2} - W_1^{-2} \\ W_2^{-2} - W_1^{-2} & W_2^{-2} + W_1^{-2} \end{pmatrix}$$

where W_1, and W_2 are diagonal.

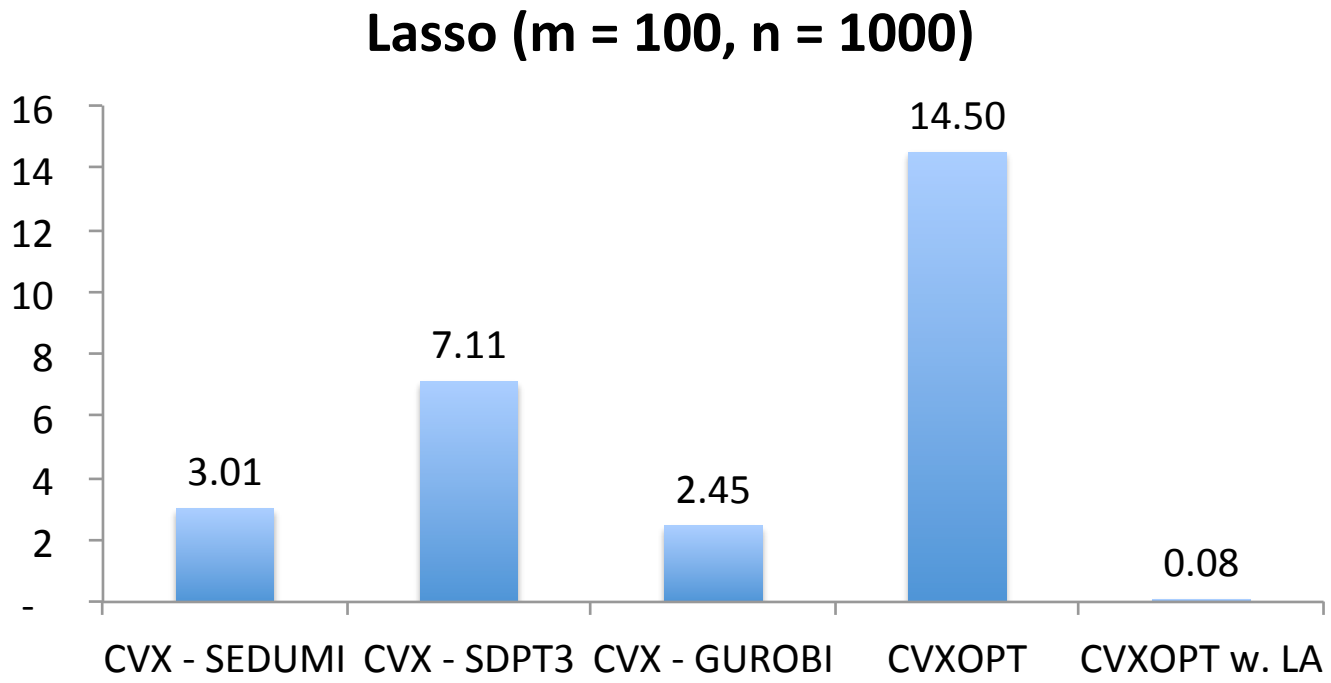
- How is this matrix structured?

- Turns out we can reduce this system to a much smaller system…

$$(A^T A + D)x = rhs$$
$$D = 4(W_1^2 + W_2^2)^{-1}$$

And the results…

- Take a look at “l1regls.py” to see how this is implemented…



Morale of the story…

- CVX is my choice for *rapid* prototyping
  - Especially good when you want to change the formulation repeatedly
- If you need to code in python
  - CVXPY is an acceptable choice for small problems
  - CVXOPT is a much better choice.
- If you have the need, patience and skill, customizing linear algebra is a good way to speed things up *significantly.*

Outline

- CVX Basics
  - What is CVX?
  - Convexity and DCP Convexity
- Advanced CVX
  - Dual variables
  - SDPs, GPs and MICPs
  - Solver settings
- CVXPY and CVX_OPT
  - CVXPY (brief)
  - Modeling language vs. solver
  - CVXOPT Basic Usage and documentation
  - Specializing Linear Algebra (time permitting)
- Course Wrap-up

Course Contents

- We've covered many tools in a short time…
  - [R] for Data Analysis
  - D3.js for Interactive Visualization
  - Python (PuLP) for Linear Optimization Modeling
  - Java and CPLEX for customizing linear optimization solvers
  - Distributed Computing on the EC2 Cloud
  - CVX and CVXOpt for Convex Optimization Modeling

- Why?
  - This suite of tools spans the various phases of research, development and implementation.
  - It's always important to think about what is the right tool for the job.

Situation 1

- You have a dataset of transaction data for a large retailer organized by customer. You're looking to build a new model to describe customer preferences.

- What would you use to do the initial data analysis?

- How would you implement and back-test various models to test their predictive power?

Situation 2

- You do research in linear integer programming. You're interested in developing heuristics and algorithms for a specific class of problems (say tripartite graph matching).
- To gain some intuition, you'd like to solve some sample instances, and study the structure of the solutions.
- What tools would you use to do this?

Situation 3

- After studying the solutions, you come up with (paper and pencil) a number of possible linear programming relaxations and would like to compare their relative strength.

- What tools would you use?

Situation 4

- Your genius and hardwork has finally paid off! You have a great new linear programming based heuristic.

- You would like to run the heuristic on many simulated instances to collect computational evidence for your forthcoming journal paper.

- How would you do this?
- How would you make a high-quality graphic for the paper?

Situation 5

- Even better than a paper, a company wants to use your heuristic as part of their operations.

- How would you demonstrate your work to non-technical audiences?

Thank you