Optimization and Incomplete Markets with Applications

Programming Project

Sergio Maffra

King's College London

4 de abril de 2019

Outline

1. Programming project
2. Introduction to CVXPY
3. Basic setup
4. Creating a package
5. Creating test functions
6. Introduction to git

Programming project

Objective:

- We will create a python package for the pricing of index options

The plan:

- Tutorial #1: basic setup
- Tutorial #2: solve the pricing problem

Project based on the following paper:

- “Pricing index options by static hedging under finite liquidity”, by Armstrong et al. [2018]

Introduction to CVXPY

"CVXPY is a Python-embedded modeling language for convex optimization problems. It allows you to express your problem in a natural way that follows the math, rather than in the restrictive standard form required by solvers."

from cvxpy.org

Implements

- Disciplined convex programming (DCP) (See Grant et al. [2006])

DCP is a strategy to make convex programming more user-friendly.

Disciplined convex programming

Strategy

- Construct models using building blocks
  - Covexity on algebraic operations (Section 1.4 of the lecture notes)
- Automate most of the work needed to analyze and solve models

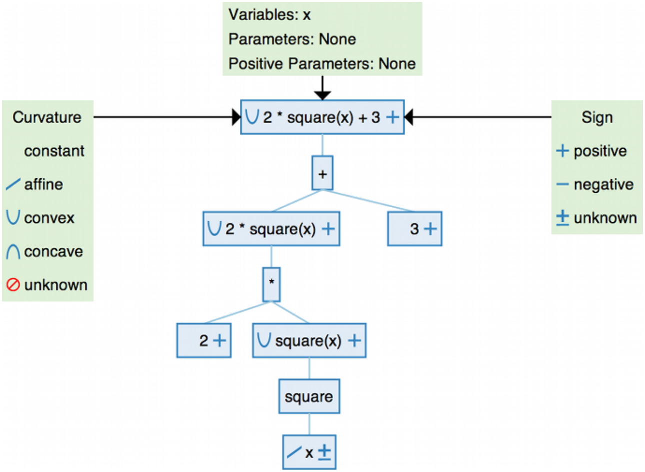
Building blocks

- Basic operators: + - * /
- Powers: x^p or (x**p in Python)
- Transpose
- Scalar functions: *square*, *geo_mean*, *norm*, ...
- Elementwise functions: *abs*, *exp*, *log*, ...
- Vector/matrix functions: *diag*, *diff*, *cumsum*, ...

For a complete list of building blocks, check:
https://www.cvxpy.org/tutorial/functions/index.html

Disciplined convex programming

Example 1 (from cvxpy.org):



Disciplined convex programming

Example 1 in Python:

```python
import cvxpy as cp

# Defining a variable
x = cp.Variable()

# Defining the object function f(x) = 2 * x^2 + 3
objective = cp.Minimize(2 * cp.square(x) + 3)

# Build a problem
problem = cp.Problem(objective)

# solving the problem
result = problem.solve()

print("f(x) = {}, x = {}".format(result, x.value))
```

Disciplined convex programming

Adding a constraint to Example 1:

```
import cvxpy as cp

# Defining a variable
x = cp.Variable()

# Defining the object function f(x) = 2 * x^2 + 3
objective = cp.Minimize(2 * cp.square(x) + 3)

# Defining a constraint
constraints = [ x >= 4 ]

# Build a problem
problem = cp.Problem(objective, constraints)

# solving the problem
result = problem.solve()

print("f(x) = {}, x = {}".format(result, x.value))
```

Disciplined convex programming

Example 2

```
import cvxpy as cp
import numpy as np

# Problem data.
m = 20; n = 10;
np.random.seed(2)
A = np.exp(np.random.randn(m, n))
b = np.exp(np.random.randn(m))
# Defining a vector variable of size n
x = cp.Variable(n)
# Defining a constraint
constraints = [0 <= x, x <= 1]
# Defining the objective function
objective = cp.Minimize(cp.log_sum_exp(A * x - b))
# Building the problem
prob = cp.Problem(objective, constraints)
# Solving the problem
result = prob.solve()
print("f(x) = {}, x = {}".format(result, x.value))
```

Basic setup

First install Anaconda, VSCode, and CVXPY. Then, run Example 1.

1. Anaconda (Python 3.7 distribution)
   - https://www.anaconda.com/distribution
2. Visual studio code
   - https://code.visualstudio.com/Download
3. CVXPY
   - https://www.cvxpy.org/install/index.html
4. MOSEK
   - https://www.mosek.com/downloads/9.0.81/
5. git
   - https://git-scm.com/downloads

Creating a package

To create a package, let's create the following directory structure:

```
tutorial/
└── oim2019/
    ├── oim2019/
    │   ├── tests/
    │   ├── data/
    │   │   └── 20171115T160000.csv
    │   ├── example_1_constrained.py
    │   ├── example_1.py
    │   ├── utils.py
    │   └── __init__.py
    ├── setup.py
    └── README.str
```

Creating a package

setup.py

```
from setuptools import setup, find_packages

def readme():
    with open('README.rst') as f:
        return f.read()

setup(name='oim2019',
        version='0.1',
        description='Optimization and Incomplete Markets with Applications - oim2019',
        long_description='',
        classifiers=['Programming Language :: Python :: 3.6'],
        keywords='incomplete markets option pricing',
        url='', author='', author_email='',
        license='MIT',
        packages=['oim2019'],
        package_data={'oim2019': ['data/*']},
        install_requires=['cvxpy', 'numpy', 'pandas'],
        test_suite='nose.collector',
        tests_require=['nose'],
        zip_safe=False)
```

Creating a package

README.rst

```
Programming project for the course OIM 2019
```

__init__.py

```
from .example_1 import *
from .example_1_constrained import *
from .utils import *
```

Creating a package

example_1.py

```
import cvxpy as cp

def run_example_1():
    # Defining a variable
    x = cp.Variable()

    # Defining the object function f(x) = 2 * x^2 + 3
    objective = cp.Minimize(2 * cp.square(x) + 3)

    # Build a problem
    problem = cp.Problem(objective)

    # solving the problem
    result = problem.solve()
    print("f(x) = {}, x = {}".format(result, x.value))
    return x.value
```

Creating a package

example_1_constrained.py

```
import cvxpy as cp

def run_example_1_constrained():
    # Defining a variable
    x = cp.Variable()

    # Defining the object function f(x) = 2 * x^2 + 3
    objective = cp.Minimize(2 * cp.square(x) + 3)

    # Defining a constraint
    constraints = [ x >= 4 ]

    # Build a problem
    problem = cp.Problem(objective, constraints)

    # solving the problem
    result = problem.solve()
    print("f(x) = {}, x = {}".format(result, x.value))
    return x.value
```

Creating a package

utils.py

```
import pandas as pd
import pkg_resources

def options_data():
    filename = pkg_resources.resource_filename('oim2019', 'data/20171115T160000.csv')
    return pd.read_csv(filename)
```

Creating a package

Finally, we install the package in development mode:

```
> python setup.py develop
```

You can now import package oim2019 in python scripts!

As a test, run the following commands in a Jupyter notebook.

```
from oim2019 import *

run_example_1()

run_example_1_constrained()

print(options_data())
```

Creating test functions

Unit tests allow the systematic testing of your code.

As an example, we create two test functions for Example 1:

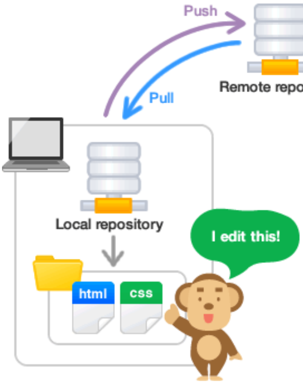
```
import oim2019

def test_example_1():
    x = oim2019.run_example_1()
    assert abs(x - 0.0) < 1e-5

def test_example_1_constrained():
    x = oim2019.run_example_1_constrained()
    assert abs(x - 4.0) < 1e-5
```

To run your tests, use the nosetests command:

```
tutorial/oim2019> nosetests
```

git

git is the most popular distributed version control system.



from https://kevintshoemaker.github.io/StatsChats/GIT_tutorial.html

git – Creating a repository

Inside your oim2019 directory:

1. Initialize a repository

```
tutorial> git init
```

2. Staging files for commit

```
tutorial> git add .
```

3. Committing files

```
tutorial> git commit -m "description of your changes!"
```

4. Link to a remote repository (created on bitbucket.com)

```
tutorial> git remote add origin https://xxx@bitbucket.org/xxx/oi
```

5. Pushing files to remote repository

```
tutorial> git push -u origin master
```

git – How to use it?

How should we use git?

1. Update local repository

```
tutorial> git pull
```

2. Do a bit of programming...
3. Stage changes for commit

```
tutorial> git add .
```

4. Committing changes to local repository

```
tutorial> git commit -m "description of your changes"
```

5. Upload changes to remote repository

```
tutorial> git push
```

References

John Armstrong, Teemu Pennanen, and Udomsak Rakwongwan. Pricing index options by static hedging under finite liquidity. *International Journal of Theoretical and Applied Finance*, 21(06), 2018.

Michael Grant, Stephen Boyd, and Yinyu Ye. Disciplined convex programming. In *Global optimization*, pages 155–210. Springer, 2006.