Install Notes - CVXPY 1.0.8

Windows 10, Anaconda

In general, follow the steps of the official install guide.

<https://www.cvxpy.org/install/index.html>

I found that even when I installed following the recommendations of the guide, it failed in multiple ways. I recorded the steps to fix it here. Everything is done from within the Anaconda Prompt (including pip installs).

Helpful conda commands:

<https://conda.io/docs/_downloads/conda-cheatsheet.pdf>

Detailed steps on next page:

Here are the steps I had to take to get a working install of CVXPY on Windows 10 using Python 3.6 in Anaconda. I pretty much gave up all hope of having any kind of automatic package management of CVXPY.

1. Make and activate a new environment
   1. Example: Run the command: conda create --name py36 python=3.6
   2. You can also use your existing environment but it is probably safer to try installing to a throwaway environment first in case it breaks
2. Install some CVXPY dependencies (if not already installed)
   1. Run the command: conda install numpy
3. (optional) Upgrade pip
   1. Run the command: python -m pip install --upgrade pip
4. Install ecos
   1. Run the command: pip install ecos==2.0.7rc2
      1. For me, just trying “pip install ecos” did not work
   2. I borrowed this step from the instructions here:
      1. <https://github.com/embotech/ecos-python/issues/10>
   3. I found that simply updating conda was not enough to fix this issue as was suggested by ‘nzhou’ in this thread:
      1. <https://github.com/cvxgrp/cvxpy/issues/492>
5. Install SCS
   1. <https://www.lfd.uci.edu/~gohlke/pythonlibs/#scs>
   2. Download the file: scs‑1.2.7‑cp36‑cp36m‑win\_amd64.whl
   3. Place the file somewhere in your filesystem
   4. Navigate (cd) your Anaconda prompt to the location where the file is
   5. Run the command: pip install scs‑1.2.7‑cp36‑cp36m‑win\_amd64.whl
   6. I borrowed this step from the instructions here:
      1. <http://i-systems.github.io/HSE545/machine%20learning%20all/cvxpy_install/CVXPY%2BInstallation%2BGuide%2Bfor%2BWindows.html>
6. Install CVXYPY
   1. Run the command: pip install cvxpy
7. Install nose
   1. Run the command: conda install nose
8. Run nose tests
   1. Run command: nosetests cvxpy
   2. If everything went well it should say “OK” at the end
   3. Possible errors:
      1. “Failed building wheel for ecos”
         1. Should be resolved if you followed step 4
      2. “Solver ‘SCS’ failed”
         1. Should be resolved if you followed step 5
      3. “Module (or package?) cvxpy not found”
         1. Should be resolved if you followed step 7
9. Beware that the “official” examples have apparently not been updated to CVXPY 1.0 which can cause some issues.
   1. <https://www.cvxpy.org/examples/index.html>
   2. For example, the control example
      1. <http://nbviewer.jupyter.org/github/cvxgrp/cvx_short_course/blob/master/intro/control.ipynb>
      2. <https://groups.google.com/forum/#!topic/cvxpy/5hBSB9KVbuI>
   3. The developers made a script to automatically convert CVXPY 0.4.11 code to CVXPY 1.0.8 code, but it apparently is may not be complete/fool-proof based on the above example
      1. <http://www.cvxpy.org/updates/index.html>

\*Python 2.7 Note:

For Python 2.7 I had to install SCS manually as well.

Try running the code on the following pages to see if you have basic functionality working within your Python console/editor.

# Try running these simple examples to check if your install is working

# Note that many of the examples on <http://www.cvxpy.org/examples/index.html> did not work as-pasted for me

#

# This is the example code from

# https://cvxpy.readthedocs.io/en/latest/tutorial/intro/index.html

import cvxpy as cvx

import numpy

import matplotlib.pyplot as plt

# Create two scalar optimization variables.

x = cvx.Variable()

y = cvx.Variable()

# Create two constraints.

constraints = [x + y == 1,

x - y >= 1]

# Form objective.

obj = cvx.Minimize((x - y)\*\*2)

# Form and solve problem.

prob = cvx.Problem(obj, constraints)

prob.solve() # Returns the optimal value.

print("status:", prob.status)

print("optimal value", prob.value)

print("optimal var", x.value, y.value)

# Replace the objective.

prob2 = cvx.Problem(cvx.Maximize(x + y), prob.constraints)

print("optimal value", prob2.solve())

# Replace the constraint (x + y == 1).

constraints = [x + y <= 3] + prob.constraints[1:]

prob2 = cvx.Problem(prob.objective, constraints)

print("optimal value", prob2.solve())

x = cvx.Variable()

# An infeasible problem.

prob = cvx.Problem(cvx.Minimize(x), [x >= 1, x <= 0])

prob.solve()

print("status:", prob.status)

print("optimal value", prob.value)

# An unbounded problem.

prob = cvx.Problem(cvx.Minimize(x))

prob.solve()

print("status:", prob.status)

print("optimal value", prob.value)

# Problem data.

m = 10

n = 5

numpy.random.seed(1)

A = numpy.random.randn(m, n)

b = numpy.random.randn(m)

# Construct the problem.

x = cvx.Variable(n)

objective = cvx.Minimize(cvx.sum\_squares(A\*x - b))

constraints = [0 <= x, x <= 1]

prob = cvx.Problem(objective, constraints)

print("Optimal value", prob.solve())

print("Optimal var")

print(x.value) # A numpy ndarray.

# Problem data.

n = 15

m = 10

numpy.random.seed(1)

A = numpy.random.randn(n, m)

b = numpy.random.randn(n)

# gamma must be nonnegative due to DCP rules.

gamma = cvx.Parameter(nonneg=True)

# Construct the problem.

x = cvx.Variable(m)

error = cvx.sum\_squares(A\*x - b)

obj = cvx.Minimize(error + gamma\*cvx.norm(x, 1))

prob = cvx.Problem(obj)

# Construct a trade-off curve of ||Ax-b||^2 vs. ||x||\_1

sq\_penalty = []

l1\_penalty = []

x\_values = []

gamma\_vals = numpy.logspace(-4, 6)

for val in gamma\_vals:

gamma.value = val

prob.solve()

# Use expr.value to get the numerical value of

# an expression in the problem.

sq\_penalty.append(error.value)

l1\_penalty.append(cvx.norm(x, 1).value)

x\_values.append(x.value)

plt.rc('text', usetex=False) # Change to usetex=True if you have working LaTeX

plt.rc('font', family='serif')

plt.figure(figsize=(6,10))

# Plot trade-off curve.

plt.subplot(211)

plt.plot(l1\_penalty, sq\_penalty)

plt.xlabel(r'\|x\|\_1', fontsize=16)

plt.ylabel(r'\|Ax-b\|^2', fontsize=16)

plt.title('Trade-Off Curve for LASSO', fontsize=16)

# Plot entries of x vs. gamma.

plt.subplot(212)

for i in range(m):

plt.plot(gamma\_vals, [xi[i] for xi in x\_values])

plt.xlabel(r'\gamma', fontsize=16)

plt.ylabel(r'x\_{i}', fontsize=16)

plt.xscale('log')

plt.title(r'\text{Entries of x vs. }\gamma', fontsize=16)

plt.tight\_layout()

plt.show()