COMP4680/COMP8650: Advanced Topics in SML

Assignment #5: Programming Assignment

Due: 11:55pm on Sunday 2 October, 2016.
Submit as a single ZIP file containing code and output via Wattle.

This assignment gives you an opportunity to explore applications of convex optimization and actually solve some convex programs. You can use either:

- the Matlab package `CVX` developed by Michael Grant and Stephen Boyd, which can be downloaded from `http://www.cvxr.com/cvx/download`; or
- the Python package `CVXPY` developed by Steven Diamond, Eric Chu and Stephen Boyd, which can be doenloaded from `http://www.cvxpy.org/en/latest/`.

Follow the installation instructions and browse through some of the user documentation (we will step you through most of what you need to know for solving the problems in this assignment).

Data for this assignment can be downloaded from Wattle.

- **Linear Programming.** Write a Matlab script using `cvx` to solve the following linear over $x \in \mathbb{R}^4$:

$$\begin{array}{rl} \text{minimize} & x_1 + 2x_2 + 3x_3 + 4x_4 \\ \text{subject to} & x_1 + x_2 + x_3 + x_4 = 1 \\ & x_1 - x_2 + x_3 - x_4 = 0 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

  Include the optimal value ($x^\star$ and $p^\star$) and a printout of your script in your solutions.

- **Regularized Maximum Likelihood Estimation.** In this problem we will develop a convex optimization problem for learning the parameters of a Gaussian distribution using samples generated from the distribution.

  (a) Let $\mathcal{D} = \{x_1, \ldots, x_m\}$ be a set of samples drawn from a Gaussian distribution with mean μ and covariance Σ. Set μ to the empirical mean (i.e., $\mu = \frac{1}{m}\sum_{i=1}^m x_i$). Show that the average log-likelihood of the data can be written as

$$\ell(\mathcal{D}) = \log\det\Sigma^{-1} - \mathrm{tr}(\hat{\Sigma}\Sigma^{-1}) + \text{const.}$$

  where $\hat{\Sigma} = \frac{1}{m}\sum_{i=1}^m (x_i - \mu)(x_i - \mu)^T$.

  (b) Suppose we wish to learn a sparse representation of the Gaussian. (There are a number of reasons why we may wish to do this). We can induce sparsity by placing an ℓ_1-penalty on the off-diagonal terms in the inverse covariance matrix. The resulting regularized maximum likelihood optimization problem can be expressed as

$$\begin{array}{rl} \text{minimize} & -\log\det K + \mathrm{tr}(\hat{\Sigma}K) + \lambda\sum_{i\neq j}|K_{ij}| \\ \text{subject to} & K \succ 0 \end{array}$$

  where $K = \Sigma^{-1}$. Show that this is a convex optimization problem.

(c) Using the data provided in `asgn5q2.mat` write `cvx` code to solve the above regularized optimization problem. You can load the data and compute the empirical covariance matrix $\hat{\Sigma}$ using

```
% Matlab
clear all;
load 'asgn5q2.mat';
[m, n] = size(X);
sigmaHat = cov(X);
```

```
# Python
import cvxpy as cvx
import numpy as np
import scipy.io

data = scipy.io.loadmat('asgn5q2.mat')
m, n = data['X'].shape
sigmaHat = np.cov(data['X'], rowvar=0)
```

Plot the optimal objective value, log-likelihood (without regularization term), and number of non-zeros in the inverse covariance matrix as a function of λ. Include a printout of your code with your solutions.

Hints. 1. Use `A == semidefinite(n)` as a constraint in Matlab or `A = semidefinite(n)` as a variable declaration in Python to indicate that A is a positive semidefinite matrix. 2. Use the function `log_det` for $\log\det(A)$. 3. When checking for sparsity truncate all entries in the inverse covariance with magnitude less than 10^{-6} to zero. In Python, you may need to reduce the accuracy of the solver: call the solve method with something like `solve(solver=cvx.CVXOPT, kktsolver='robust', feastol=1.0e-4)`.

- **Total Variation Denoising.** In this question we investigate the problem of signal denoising. Consider a signal $x \in \mathbb{R}^n$ corrupted by noise. We measure the corrupted signal x_{corr} and wish to recover a good estimate $\hat{x}$ of the original signal. To do this we solve the total variation denoising problem

$$\text{minimize} \quad \|\hat{x} - x_{\text{corr}}\|_2^2 + \lambda \|D\hat{x}\|_1$$

where D is the discrete derivative operator. Using the data supplied in `asgn5q3.mat` write a `cvx` program to solve the above optimization problem. You can load and plot the corrupted signal using the following code:

```
clear all;
load 'asgn5q3.mat';
n = length(x_corr);
plot(1:n, x_corr, 'LineWidth', 2);
grid on;
```

```
# Python
import cvxpy as cvx
import numpy as np
import scipy.io
import matplotlib.pyplot as plt

data = scipy.io.loadmat('asgn5q3.mat')
n = len(data['x_corr'])
plt.plot(data['x_corr'], linewidth=2)
plt.show()
```

You should experiment with your code to find a "good" value for λ. Include a printout of your code and a plot of the recovered signal.