Some of the question below have instructions in **bold** that require you to make plots or answer questions. Create a short pdf document called "hmwk6.pdf" with your answers for each question. You don't need to make an elaborate writeup, just put your plots into the pdf with labels indicating what they are, and answer questions when prompted. **Turn in your code along with the pdf on the gradeserver**.

1. Write an "unwrapped ADMM" solver for the support-vector machine problem

$$\text{minimize} \quad \frac{1}{2}\|x\|^2 + Ch(Ax) \tag{1}$$

where $h(z) = \sum_i \max\{1 - z_i, 0\}$ is the hinge loss function, and $A = LD$ is the product of the (diagonal) label matrix with the data matrix. If the ith training vector is d_i and has binary (± 1) label l_i, then the ith row of A is $l_i d_i$.

   (a) **Using the change of variables $y \leftarrow Ax$, write problem (1) as an "unwrapped" constrained problem.**

   (b) **Write the augmented Lagrangian for this constrained problem. Use λ to denote the dual variable.**

   (c) **Write the system of equations that needs to be solved to update x.** Your implementation must compute and cache the Cholesky factorization of this system before iterations begin, and re-use the factorization on each iteration.

   (d) **Write the y update in terms of the proximal operator of h (you can just write $\text{prox}_h(\cdot,\cdot)$ for the proximal operator).**

   (e) **Write the λ update.**

   (f) Use your method to solve a classification problem with 20 features, and N vectors (you can pick N). **Create a convergence curve that plots the residuals vs iteration count. The primal and dual residuals for this problem are given by**

$$p^k = \|Ax^k - y^k\|$$

$$d^k = \|\tau A^T(y^k - y^{k-1})\|.$$

   Automatically stop your iteration when residuals get "small." One possible stopping rule is to stop when $p^k \le 10^{-3}\max_{i<k} p^i$, and $d^k \le 10^{-3}\max_{i<k} d^i$. However you may also just stop when $p^k, d^k < \epsilon$ for some small ϵ.

   Choose a reasonable value for τ that gives good performance. **How big can you make N and still have the solver terminate within 30 seconds?**

   Hint: The prox operator of h is given by

$$\text{prox}_h(z,\delta)_k = z_k + \max\{\min\{1 - z_k, \delta\}, 0\}$$

   where $\text{prox}_h(z,\delta)_k$ denotes the kth entry in the output, and z_k denote the kth entry in the input.

2. (a) (Matlab only) Solve the SVM problem using the built in Matlab command `svmtrain(D,l)`. Use the largest value for N that worked in 30 seconds for your ADMM code. **Which code is more efficient? Which code do you prefer?**

(b) (Python only) Solve the SVM problem using the Python class `sklearn.svm`. You will need to run a sequence of commands that looks like this:

```
from sklearn import svm
data = ... # define data
labels = ... # define labels
clf = svm.SVC()
clf.fit(data, labels)
```

Which code is more efficient? Which code do you prefer?