Preparation for assignment 5

Machine Learning, Summer term 2014, Ulrike von Luxburg

CVX optimization package for MATLAB (unfortunately NOT compatible with Octave)

CVX is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax.
You can download CVX from `http://cvxr.com/cvx/download/`. Installation is relatively simple: unpack the distribution to an empty directory, and then run `cvx_setup` in this directory from the MATLAB command line. For more information, see `http://cvxr.com/cvx/doc/install.html`.
After installation, you can use CVX commands in your matlab files. To use CVX effectively, you need to know at least a bit about convex optimization. To use CVX for solving your optimization problem, you need to

- check if your problem indeed is a convex optimization problem. CVX is not meant to be a tool for checking if your problem is convex
- declare optimization variables, describe the objective function and describe the constraints according to the follwing scheme

```
cvx_begin
    variable x(n)
    minimize ( f(x) )
    subject to
        g(x) <= 0
        h(x) == 0
cvx_end
```

Example 1: Consider the following convex optimization problem

$$\begin{aligned} \text{minimize} \quad & \|Ax - b\|_2 \\ \text{subject to} \quad & Cx = d \\ & \|x\|_\infty \le e \end{aligned}$$

The following code segment generates and solves a random instance of this model:

```
m = 20; n = 10; p = 4; A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end
```

Example 2: You can use CVX to solve soft margin SVM classification problems

$$\min_{\mathbf{w};\ ;b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i$$

$$\begin{aligned} y_i(\mathbf{w}^\top \mathbf{x_i} + b) \ge 1 - \xi_i \quad & i = 1,\dots,m \\ \xi_i \ge 0 \quad & i = 1,\dots,m \end{aligned}$$

To solve this problem in matlab using CVX, first you need to initialize C, n, m, vector y and matrix X. Then

```
cvx_begin
    variables w(n) b xi(m)
    minimize 1/2*sum(w.*w) + C*sum(xi)/m
    y.*(X*w + b) >= 1 - xi;
    xi >= 0;
cvx_end
```

- You can keep CVX quiet during optimization by `cvx_quiet(true);`
- Documentation: `http://cvxr.com/cvx/doc/CVX.pdf`