Problem 1

The $\ell_p$-norm of a vector $x \in \mathbb{R}^D$ is defined as

$$
\|x\|_p = \left( \sum_{i=1}^D |x_i|^p \right)^{1/p}.
$$

(a) Complete the MATLAB script `norms.m` that takes in a matrix $X = [x_1 \ x_2 \ \ldots \ x_N] \in \mathbb{R}^{D \times N}$ and computes the $\ell_p$-norm of each column of $X$. **Do not** use any loops. **Turn in** your code and a brief explanation of how it works.

(b) Test your implementation by running `prob1.m`. **Turn in** the resulting plot. This is a way to visualize the fact that in high dimensions, the $\ell_2$-norm of a standard normal random vector concentrates tightly around $\sqrt{D}$.

Problem 2

Define the unit $\ell_p$-norm ball as

$$
B_p = \left\{ x \in \mathbb{R}^D : \|x\|_p \leq 1 \right\}.
$$

It is often useful to visualize the unit ball for different norms, as we saw in the lecture on sparse regression. To do this, we instead visualize the unit sphere for each norm, which is

$$
S_p = \left\{ x \in \mathbb{R}^D : \|x\|_p = 1 \right\}.
$$

One way to visualize this sphere is to generate a large number of random points and normalize these points so that they have $\|x\|_p = 1$.

(a) Complete the MATLAB script `normalize.m` that takes in a matrix $X = [x_1 \ x_2 \ \ldots \ x_N] \in \mathbb{R}^{D \times N}$ and normalizes it so that each column of $X$ has unit $\ell_p$-norm. **Do not** use any loops. **Turn in** your code and a brief explanation of how it works.

(b) Test your implementation by running `prob2.m`. **Turn in** the resulting plot.

(c) Intuitively, a set is convex if the line connecting any two points in a set is contained within that set. Based on this definition and your plot, for which norms (i.e., which values of $p$) is the unit ball convex?
Problem 3

To this point, we’ve been focused on minimizing the least-squares cost function, which can be written as

$$J(w) = \sum_{i=1}^{N} (x_i^T w - y_i)^2 + \lambda \|w\|^2_2. \quad (2)$$

In machine learning, it is common to refer to $$(x_i^T w - y_i)^2$$ as a loss function $$L$$. In this case, (2) can be rewritten as

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, x_i^T w) + \frac{\lambda}{2N} \|w\|^2_2, \quad (3)$$

where $$L(y,t) = (y-t)^2$$ is known as the squared-error loss. Of course, there is no reason we are restricted to this particular choice of loss function. For classification problems, it is much more common to choose the hinge loss

$$L(y,t) = \max(0, 1 - yt). \quad (4)$$

The above loss is convex but not differentiable at 0, so we will make use of subgradients (see notes on sparse regression). Using the above loss and assuming there are two class labels $$y_i \in \{-1, 1\}$$, solving (3) yields the optimal soft margin hyperplane, which is the subspace of dimension $$D-1$$ that best separates data from two classes (where $$D$$ is the dimension of the features/examples/data points). In this problem, you will find this classifier using gradient descent and stochastic gradient descent.

(a) First, rewrite (3) as a summation

$$J(w) = \sum_{i=1}^{N} J_i(w) \quad (5)$$

for an appropriately defined $$J_i(w)$$. Using the loss function defined in (4), differentiate (5) for a single $$i$$ with respect to the vector $$w \in \mathbb{R}^D$$ to obtain a subgradient. Be careful at the point $$yt = 1$$.

(b) Solve (3) with hinge loss using gradient descent by computing the full gradient vector, which is the sum of the above “stochastic” gradients. Test your code on the nuclear.mat data included in the files by running prob4.m. Turn in your code. Plots will be formed in part (c).

Hint: It may be easier to create a separate subg.m file to compute the subgradient, which you can use in both parts (b) and (c).

(c) Solve (3) using stochastic gradient descent by taking 20 full passes through the data in random order. Turn in a plot of the resulting cost as a function of the number of iterations as well as the resulting plot of the two separators (code provided). How do the two methods compare?

Note: For fair comparison, you should compute the cost for SGD after each of the 20 cycles, not after each individual update.