Mathematical Tools for Neuroscience (NEU 314)
Princeton University, Spring 2016
Jonathan Pillow

Homework 3: Linearity, Orthogonal Matrices and SVD

Due: Friday, Mar 11, 11:59pm

Linear Systems

1. **Vectors as linear operators.** We have discussed vectors as oriented line segments that have a particular length and point in a particular direction in an n-dimensional space. But we can also view them as linear operators, that is, as specifying linear functions that map an input vector to a scalar via the dot product.

   For each of the following, specify a 10×1 column vector $\vec{x}$ such that $\vec{x} \cdot \vec{y}$ performs the corresponding linear operation on the 10×1 vector $\vec{y}$:

   a. **averager**. What is the vector $\vec{x}$ such that $\vec{x} \cdot \vec{y}$ produces the mean (average) value of the components in $\vec{y}$?

   b. **windowed averager**. The average of just elements 3 through 6 of y.

   c. **local differencer**. The difference $y(6) - y(5)$.

   d. **component selector**. The 9'th element of y.

2. **Rotation matrices.** One simple example of an orthogonal matrix is the 2×2 rotation matrix, which takes any 2-component vector and rotates it counterclockwise by an angle θ. We can identify such a matrix by the effect it has on the identity basis. That is, it should map the "standard" basis vector $[1; 0]$ (corresponding to the unit-vector pointing rightwards on the x axis) to the column vector $[\cos(\theta); \sin(\theta)]$, and the basis vector $[0; 1]$ (which points up along the y axis) to the column vector $[-\sin(\theta); \cos(\theta)]$.

   a. Using the information above, form the $2x2$ matrix M that maps any 2-vector counterclockwise by an angle $0.01 * \pi$.

   b. Verify that the euclidean norm $||\vec{x}||$ and $||M * \vec{x}||$ for some example vector $\vec{x}$.

   c. Create an arbitrary unit vector and make a movie showing it completing 2 full revolutions (i.e., 4π radians or 720deg). Use a for loop, and in each iteration rotate the vector by multiplying it by the rotation matrix M defined above, then plot it. Use the command "`draw now;`" at the end of each iteration to force matlab to display the updated plot for each iteration of the for loop. You can make the movie play more slowly by inserting a `pause(.1)` after `draw now`, which will cause matlab to wait for 0.1s before continuing. (3 points).

Singular Value Decomposition (SVD)

3. **Pseudoinverse**.

   a. Write your own function to compute the *pseudoinverse* of a matrix, using no "advanced" commands besides the function `svd` (which computes the singular value decomposition of a matrix). (2 points).

   b. Generate a 3×3 matrix A that has rank 2. What happens when you try to invert A, using the matlab function `inv`? Verify that your pseudoinverse function returns the same pseudoinverse as matlab's built-in `pinv(A)`.

   c. Generate a vector $\vec{x}$ that lies fully within the row space of A. (A simple way to do this is just to make a column vector out of one of the rows of A). Is $A^\dagger * A * \vec{x}$ equal to $\vec{x}$?

   d. Generate a vector $\vec{y}$ that lies in the null space of A. (Feel free to use the built-in function `null` if you like). Is $A^\dagger * A * \vec{y}$ equal to $\vec{y}$?

   e. What do we get if we multiply $A^\dagger * A * (\vec{x} + \vec{y})$? Give a one-sentence explanation of what's happening here.

4. **Eigenvectors of a psd (positive semi-definite) matrix**.

   a. Generate a positive semi-definite 5×5 matrix A by generating a random 5×5 matrix and multiplying it by its transpose. (The product of a matrix and its transpose is always positive semi-definite!). Now compute the eigenvector $\vec{x}$ with the largest eigenvalue. Verify that $A\vec{x}$ is equal to $\vec{x}$ stretched by its eigenvalue.

   b. Repeat for the eigenvector with the second-smallest eigenvalue.

5. **Determinant of a psd matrix**.

   Write your own function called `mydet.m` to compute the determinant of a square, positive definite matrix using no advanced matlab commands besides `svd`. (Commands like `diag` and `prod` are fine). Test it out by computing the determinant of the matrix A you defined above and comparing it to `det(A)`.