DSC 140B - Homework 02

Due: Wednesday, April 19

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope at 11:59 PM.

Problem 1.

Suppose (just like in the last homework) that in a group of 1000 people, 600 currently live in California and 400 currently live in Texas. In any given year, 5% of the people living in California move to Texas, and 3% of the people living in Texas move to California. You may assume that the people do not move to any other states.

We can represent the current number of people living in California and Texas with a *population vector*:

$$\vec{p} = (\#\text{ in California}, \#\text{ in Texas})^T.$$

The initial situation described above is represented by the population vector $(600, 400)^T$.

a) Let $\vec{f}(\vec{p})$ be the linear transformation which takes in a current population vector, $\vec{p} = (c, t)^T$, and returns the population vector after one year has passed. In part (b) of the corresponding problem on the last homework, you should have found the following formula for $\vec{f}$ with respect to the standard basis:

$$\vec{f}(\vec{p}) = (.95c + .03t, \quad .05c + .97t)^T$$

Write the *matrix* A representing $\vec{f}$ with respect to the standard basis.

b) Using a matrix multiplication, find the population vector after one year has passed, given that the initial population vector is $(600, 400)^T$. Your result should not contain decimals.

c) In part (f) of the last homework, we saw that two eigenvectors of A are

$$\vec{u}^{(1)} = \begin{pmatrix} 375 \\ 625 \end{pmatrix} \qquad \vec{u}^{(2)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Verify that these are eigenvectors of the matrix A by performing the matrix multiplication.

d) Write the matrix $A_{\mathcal{U}}$ of the linear transformation $\vec{f}$ with respect to the basis $\mathcal{U} = \{\vec{u}^{(1)}, \vec{u}^{(2)}\}$.

e) In part (f) of the last homework, you found that the initial population vector $\vec{x} = (600, 400)^T$ expressed in the new basis has coordinates $[\vec{x}]_{\mathcal{U}} = (1, 225)^T$.

Compute $A_{\mathcal{U}}[\vec{x}]_{\mathcal{U}}$ and then convert the resulting to a coordinate vector in the standard basis.

Hint: your result should be familiar.

Problem 2.

Let $g(\vec{x}) = g(x_1, x_2) = 4x_1^2 + 3x_2^2 + 10x_1x_2$, where we've defined $\vec{x} = (x_1, x_2)^T$. In this problem, we will consider maximizing g subject to the constraint $x_1^2 + x_2^2 = 1$.

You saw how to solve optimization problems like this in your multivariate calculus class using the method of *Lagrange multipliers*. Informally-speaking, the idea behind Lagrange multipliers is that the gradient vector of g and the gradient of the constraint $x_1^2 + x_2^2 - 1$ should be parallel at a constrained optimum. Since two vectors $\vec{a}$ and $\vec{b}$ are parallel if and only if $\vec{a} = \lambda\vec{b}$ for some λ, and since the gradient of the constraint is simply $(2x_1, 2x_2)^T = 2\vec{x}$, this means that a local optimum should satisfy $\nabla g(\vec{x}) = 2\lambda\vec{x}$. This looks similar to the eigenvector equation $A\vec{x} = \lambda\vec{x}$; in this problem we'll make the connection clearer.

a) The Lagrange multiplier approach says that we should define the *Lagrangian*:

$$\mathcal{L}(x_1, x_2, \lambda) = g(\vec{x}) - \lambda(x_1^2 + x_2^2 - 1)$$

We then solve the system of three equations in three unknowns:

$$\frac{\partial\mathcal{L}}{\partial x_1} = 0$$

$$\frac{\partial\mathcal{L}}{\partial x_2} = 0$$

$$\frac{\partial\mathcal{L}}{\partial \lambda} = 0$$

Write out and solve this system for x_1, x_2, and λ.

Hint: Try to get a formula for x_1^2 in terms of λ only, and same for x_2^2. When you get to this point, you will be able to substitute your formulas for x_1^2 and x_2^2 into $\partial\mathcal{L}/\partial\lambda = 0$ to get a function of the form

$$\frac{a_1}{(b_1\lambda + c_1)^2 + d_1} + \frac{a_2}{(b_2\lambda + c_2)^2 + d_2} - 1 = 0,$$

where the a, b, c, d's are all constants. We want to solve this for λ, which is not easy to do analytically. Instead, solve it numerically using `scipy.optimize.fsolve`, or similar. Once you've solved for λ, you can plug it back in to your equations for x_1^2 and x_2^2.

b) The equation $g(\vec{x}) = 4x_1^2 + 3x_2^2 + 10x_1x_2$ can be written in matrix-vector form as $g(\vec{x}) = \vec{x}^T A\vec{x}$ for an appropriately-defined matrix, A. Find this matrix A, and show that the matrix form is equivalent to the original form.

You can assume that A is symmetric.

c) Using whatever method you choose (e.g., numpy), compute the eigenvectors and eigenvalues of A. Show that the eigenvectors are the same as your solution to part (a).

The columns of the `evecs` array are the two eigenvectors of A. We have two solutions: $\hat{u}^{(1)} = (0.671, -0.741)^T$ and $\hat{u}^{(2)} = (-0.741, -0.671)^T$. These are the same as the solutions to the first part. Note that in the first part, we also have $(-0.671, 0.741)^T$ and $(0.741, 0.671)^T$, but these are just $-\hat{u}^{(1)}$ and $-\hat{u}^{(2)}$, and are therefore not really "different" solutions.

d) We saw in lecture that a matrix can be interpreted as the representation of a linear transformation $\vec{f}(\vec{x})$. It turns out that A represents the *gradient* of $\vec{g}$.

Show that A represents the linear transformation $\vec{f}(\vec{x}) = \frac{1}{2}\nabla g(\vec{x})$, $\nabla g(\vec{x}) = (\partial g/\partial x_1,\, \partial g/\partial x_2)^T$ is the *gradient* of g.

Therefore, for a function g of the form $ax_1^2 + bx_2^2 + cx_1x_2$, the gradient is a linear transformation that can be computed by a matrix multiplication, and the method of Lagrange multipliers is equivalent to finding an eigenvector of the matrix representing the gradient.

Problem 3.

The power method is a simple approach to computing an eigenvector of a matrix A. The method is as follows:

1. Initialize $\vec{x}^{(0)}$ arbitrarily (e.g., randomly).
2. Repeat until convergence:
   (a) Set $\vec{x}^{(i+1)} = (A\vec{x}^{(i)})/\|A\vec{x}^{(i)}\|$.

Eventually, $\vec{x}^{(i)}$ will be the top eigenvector of A.

In this problem, we'll develop an intuition for why this works.

a) The algorithm described above normalizes at each step for numerical precision. This complicates the analysis slightly. Instead, suppose for now that at every step we do not normalize; that is $\vec{x}^{(i+1)} = A\vec{x}^{(i)}$. In other words, after k iterations, we'll have $\vec{x}^{(k)} = A^k\vec{x}^{(0)}$. Only at the very end do we normalize by dividing by $A^k\vec{x}^{(0)}$. That is, after k iterations we'll return $A^k\vec{x}^{(0)}/\|A^k\vec{x}^{(0)}\|$.

Suppose that A is a $d \times d$ symmetric matrix. Let the eigendecomposition of $\vec{x}^{(0)}$ in terms of A be $\vec{x}^{(0)} = a_1\hat{u}^{(1)} + \ldots + a_d\hat{u}^{(d)}$, where $\{\hat{u}^{(1)}, \ldots, \hat{u}^{(d)}\}$ form an orthonormal basis of eigenvectors of A. You may assume that $A\hat{u}^{(i)} = \lambda_i\hat{u}^{(i)}$, where λ_i is the eigenvalue associated with $\hat{u}^{(i)}$, that $|\lambda_1| > |\lambda_2| > |\lambda_3| \cdots$, etc., and that $a_1 \neq 0$. For simplicity, you may assume that all of the eigenvalues are positive, although this is actually not necessary for the proof.

Find a formula for $\|A^k\vec{x}^{(0)}\|$ that involves only $a_1, \ldots, a_d$ and the eigenvalues $\lambda_1, \ldots, \lambda_d$.

b) Show that

$$\lim_{k\to\infty} \frac{A^k\vec{x}^{(0)}}{\|A^k\vec{x}^{(0)}\|} = \hat{u}^{(1)}.$$

That is, the result is the top eigenvector of A (the eigenvector whose eigenvalue is largest in absolute value).

Note: a proof by calculation is needed to earn full credit, but an informal argument with some calculation will earn almost all of the credit.

c) Let A be the matrix

$$\begin{pmatrix} 3 & 7 & 2 \\ 7 & 9 & -3 \\ 2 & -3 & 1 \end{pmatrix}.$$

Implement the power method (the original version, which normalizes at each step) in code, and use it to compute the top eigenvector of A. Attach your code here.

d) Using another method (e.g., `numpy.linalg.eigh`), compute the top eigenvector of A and verify that your implementation of the power method agrees.