EECS 16B Designing Information Devices and Systems II
Fall 2016 Murat Arcak and Michel Maharbiz

Homework 10

This homework is due October 31, 2016, at Noon.

1. **Homework process and study group**

   (a) Who else did you work with on this homework? List names and student ID's. (In case of homework party, you can also just describe the group.)

   (b) How long did you spend working on this homework? How did you approach it?

2. **Eigenfaces**

   In this problem, we will be be exploring the use of PCA to compress and visualize pictures of human faces. We use the images from the data set Labeled Faces in the Wild. Specifically, we use the set with all images aligned using deep funneling to ensure that the faces are centered in each photo. Each image is a 250x250 image with the face aligned in the center. To turn the image into a vector, we stack each column of pixels in the image on top of each other, and we normalize each pixel value to be between 0 and 1. Thus, a single image of a face is represented by a 625,000 dimensional vector, but a vector this size is a bit challenging to work with directly. We combine the vectors from each image into a single matrix and run PCA on it. For this problem, we will provide you with the first 250 principal components, but you can explore how well the images are compressed with fewer components. Please refer to the IPython notebook to answer the following questions.

   (a) We provide you with a set of faces from the training set and compress them using the first 100 principal components. You can adjust the number of principal components used to do the compression. What changes do you see in the compressed images when you used a small number of components and what changes do you see when you use a large number?

   (b) You can visualize each principal component to see what each dimension "adds" to the high-dimensional image. What visual differences do you see in the first few components compared to the last few components?

   (c) By using PCA on the face images, we obtain orthogonal vectors that point in directions of high variance in the original images. We can use these vectors to transform the data into a lower dimensional space and plot the data points. In the notebook, we provide you with code to plot a subset of 400 images using the first two principal comonents. Try plotting other components of the data, and see how the shape of the points change. What difference do you see in the plot when you use the first two principal components compared with the last two principal components? What do you think is the cause of this difference?

   (d) We can use the principal components to generate new faces randomly. We accomplish this by picking a random point in the low-dimensional space and then multiplying it by the matrix of principal components. In the notebook, we provide you with code to generate faces using the first 250 principal components. You can adjust the number of components used. How does this affect the resulting images?

3. The Moore-Penrose Pseudoinverse for "fat" matrices

Say we have a set of linear equations described as $A\vec{x} = \vec{y}$. If A is invertible, we know that the solution is $\vec{x} = A^{-1}\vec{y}$. However, what if A is not a square matrix? In 16A, you saw how this problem could be approached for tall matrices A where it really wasn't possible to find a solution that exactly matches all the measurements. The Linear Least-Squares solution gives us a reasonable answer that asks for the "best" match in terms of reducing the norm of the error vector.

This problem deals with the other case — when the matrix A is short and fat. In this case, there are generally going to be lots of possible solutions — so which should we choose? Why? We will walk you through the **Moore-Penrose Pseudoinverse** that generalizes the idea of the matrix inverse and is derived from the singular value decomposition.

(a) Say you have the following matrix.

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix}$$

Calculate the SVD decomposition of A. That is to say, calculate U, Σ, V such that,

$$A = U\Sigma V^T$$

What are the dimensions of U, Σ and V?

Note. Do NOT use a computer to calculate the SVD. You may be asked to solve similar questions on your own in the exam.

(b) Let us think about what the SVD does. Let us look at matrix A acting on some vector $\vec{x}$ to give the result $\vec{y}$. We have,

$$A\vec{x} = U\Sigma V^T\vec{x} = \vec{y}$$

Observe that $V^T\vec{x}$ rotates the vector, Σ scales it and U rotates it again. We will try to "reverse" these operations one at a time and then put them together.

If U "rotates" the vector $\left(\Sigma V^T\right)\vec{x}$, what operator can we derive that will undo the rotation?

(c) Derive an matrix that will "unscale", or undo the effect of Σ where it is possible to undo. Recall that Σ has the same dimensions as A. Ignore any division by zeros (that is to say, let it stay zero).

(d) Derive an operator that would "unrotate" by V^T.

(e) Try to use this idea of "unrotating" and "unscaling" to derive an "inverse" (which we will use $A^\dagger$ to denote). That is to say,

$$\vec{x} = A^\dagger\vec{y}$$

The reason why the word inverse is in quotes (or why this is called a pseudo-inverse) is because we're ignoring the "divisions" by zero.

(f) Use $A^\dagger$ to solve for $\vec{x}$ in the following systems of equations.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix}\vec{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

(g) (optional) Now we will see why this matrix is a useful proxy for the matrix inverse in such circumstances. Show that the solution given by the Moore-Penrose Psuedoinverse satisfies the minimality property that if $\vec{\hat{x}}$ is the psuedo-inverse solution to $A\vec{x} = \vec{y}$, then $\|\vec{\hat{x}}\| \leq \|\vec{z}\|$ for all other vectors $\vec{z}$ satisfying $A\vec{z} = \vec{y}$.

(Hint: look at the vectors involved in the V basis. Think about the relevant nullspace and how it is connected to all this.)

This minimality property is useful in both control applications (as you will see in the next problem) and in communications applications.

4. **SVD for minimum energy control**

Given a practical discrete linear system model $\vec{x}(t+1) = A\vec{x}(t)$.

Consider applying open loop control

$$\vec{x}(t+1) = A\vec{x}(t) + Bu(t)$$

to the system to drive it from some initial state $\vec{x}_0$ to $\vec{x}_f$. (for simplicity we considered scaler $u(t)$, but the conclusion of this problem can be readily extended to vector inputs). We know that if A, B are controllable and the dimension is n, then clearly we can get to the desired $\vec{x}_f$ in n steps. However, suppose that we only need to get there by $m > n$ steps. We now have a lot of flexibility.

Among all controls that guarantees "reachability", we could ask for a control that gets us to the desired $\vec{x}_f$ using minimal energy. i.e., having minimal

$$\sum_{t=0}^{m-1} \|u(t)\|^2.$$

(a) Firstly, give an concrete example such that $\sum_{t=0}^{m-1} \|u(t)\|^2$ can be the "energy" of the control inputs.

(b) Consider the system evolution equations from $t = 1$ to $t = m$, obtain an expression of $\vec{x}(m)$ as a function of the initial state $\vec{x}_0$ and control inputs.

(c) Write out the above equation in a matrix form, with $\vec{u} = [u(0), u(1), \cdots, u(m-1)]^T$.

(d) Now you have obtained a linear equation in the form $\vec{y} = C\vec{u}$, where $\vec{y}$ and C contains your results from last question. Recall that in the previous problem, you have shown that the solution obtained by psuedo-inverse (using the SVD) has a nice minimality property. Use this to derive the minimum energy control inputs $\vec{u}$.

(e) How do you extend the above discussion to the case when $u(t)$ is a vector $\vec{u}(t)$?

5. **MIMO wireless signals**

Ever wonder why newer wifi routers and cellular base stations have 4 or sometimes even more antennas on them? New wireless technologies actually use multiple antennas that each send their own signal on the same frequency band. The key here is not only do we encode signals in frequency bands, but also in spatial ones using a technique known as "Spatial Multiplexing".

We call this idea "MIMO" wireless, which stands for "multiple input multiple output". This technique is used in many standards including 802.11n/ac, 4G LTE, and WiMAX.

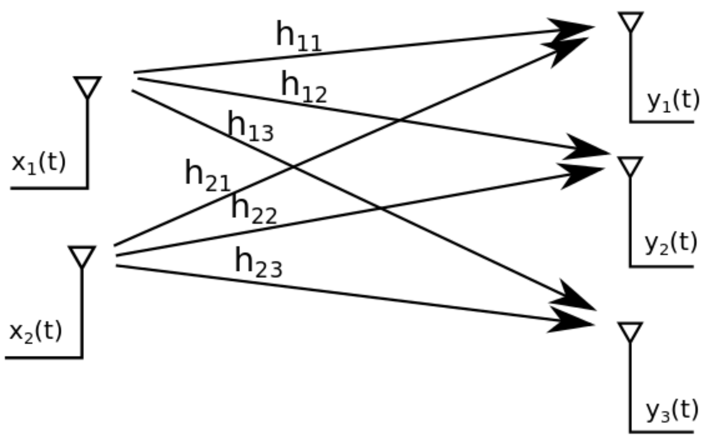
In this problem, we will explore how signals are decoded on the output end.

Consider the following:

We have 2 transmit antennas and 3 receive antennas, each receive antenna gets some signal from each of the receive antennas. We can model the input output relation of the system as follows:

$$\begin{bmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \\ h_{13} & h_{23} \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

or



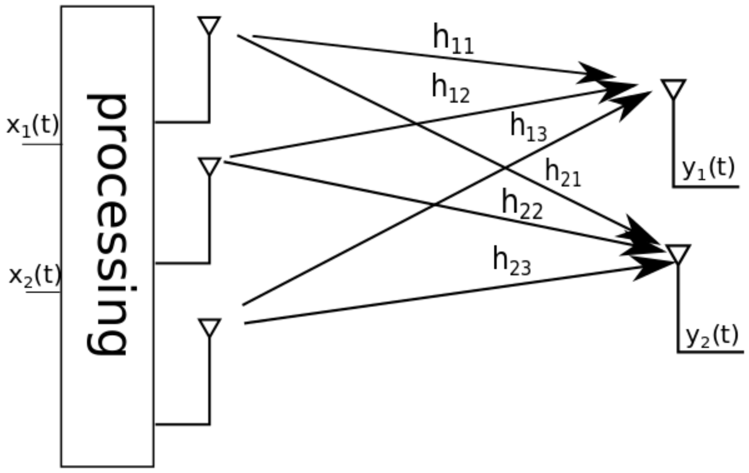
$$H\vec{x}(t) = \vec{y}(t)$$

Here, H is the spatial-response matrix and it acts on the signals instantaneously at each time. For the purpose of this problem, we are going to pretend that there are no echoes across time.

(a) With our new MIMO wireless system, we want to recover the original $\vec{x}(t)$ signal after receiving the $\vec{y}(t)$. In order to do this, we will left multiply $\vec{y}(t)$ by some matrix A; ideally we should then exactly recover $\vec{x}(t)$ ($A\vec{y}(t) = \vec{x}(t)$). Using the SVD to decompose H, analytically write down what this matrix A should be.

(b) How is the solution you found in part (a) related to the least squares solution of this problem?

(c) What we just did is referred to as "post processing" or "post coding", and involves the receive end having more antennas than the send side. Many times this is not the case (eg. a wireless cell tower having many more antennas than a phone). What if we wanted to send 2 streams on 3 antennas and receive precisely those 2 streams back on the other end?



The channel is very similar to the one we had made in part (a). In fact, the original channel modelled with spatial response matrix H is precisely the transpose of this channel! Thus, we can say the spatial response matrix for this channel, lets call it H' is simply the following:

$$H' = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix} = H^T$$

Using the SVD of H and its relation to H', show how you can pre-process $x_1(t)$ and $x_2(t)$ so that you recover them precisely after they have been transmitted across the channel. To be more explicit, after the processing and transmission has been done $y_1(t) = x_1(t), y_2(t) = x_2(t)$.

(d) (Optional) Why do you think that we are using the SVD here? Is there a unique solution of what to transmit that will achieve the desired goal in the previous part? Why choose this approach?

6. Brain-machine interface

An iPython notebook pca_brain_machine_interface.ipynb is available on the course web page. This exercise will guide you through the process of analyzing brain machine interface data using principle component analysis (PCA). This will help you to prepare for the project, where you will need to use PCA as part of a classifier that will allow you to use voice or music inputs to control your car.

Please complete the notebook by following the instructions given.

Contributors:

- Stephen Bailey.
- Siddharth Iyer.
- Ioannis Konstantakopoulos.
- Saavan Patel.