Lecture 17 - Notes

Rahul Arya

March 2019

This note is shorter than usual, since Prof. Sahai spent part of the lecture reviewing the content in Module 2 and providing some additional commentary that is available at @683 on Piazza.

1 Overview

When we discussed the stability of a vector system, we showed that discrete-time systems with a diagonalizable state matrix A were stable exactly when all the eigenvalues of A were less than 1 in magnitude. Intuitively, it seems like this result was all there was to the matter - after all, most randomly generated matrices A are diagonalizable, so this result essentially applies in the general case.

However, this intuition breaks down when we begin to consider state feedback. Recall that, for any (potentially unstable) single-input controllable system with state equation

$$\vec{x}[i+1] = A\vec{x}[i] + Bu[i],$$

we can choose a matrix K of feedback gains that can arbitrarily modify the characteristic polynomial of the state matrix of the system with feedback:

$$\vec{x}[i+1] = (A + BK)\vec{x}[i].$$

That is to say, we can choose a K that can set the coefficients d_i of the characteristic polynomial P of $A + BK$ to whatever we want, where P is of the form

$$P(\lambda) = \lambda^n - d_{n-1}\lambda^{n-1} - d_{n-2}\lambda^{n-2} - \ldots - d_1\lambda - d_0.$$

If we want to stabilize our system, and thus minimize its eigenvalues, it would make sense to set all the $d_i = 0$, to obtain the characteristic polynomial $P(\lambda) = \lambda^n$. After all, the only λ satisfying $P(\lambda) = 0$ is $\lambda = 0$, so we have successfully set all the eigenvalues of $A + BK$ to zero!

Unfortunately, observe that the only diagonalizable matrix with all zero eigenvalues is the zero matrix. And since BK is only of rank 1, it is very unlikely that $A + BK = 0$. Instead, we have constructed a non-diagonalizable state matrix (known as a *defective* matrix) from an arbitrary state matrix, simply by applying input feedback intended to make it as stable as possible.

But since this matrix is defective, we have no idea whether it is stable! Should we choose different feedback gains to make the eigenvalues greater, but ensure stability? This is not ideal, since we would be making our system less stable, and anyway how would we be certain that those gains would always produce a diagonalizable state matrix?

Instead, in this lecture and in the next we will develop techniques for analyzing defective matrices. In particular, we will demonstrate that the defective state matrix $A + BK$ that we constructed above is in fact stable, despite not being diagonalizable!

2 Upper-triangular form

We will demonstrate that any square matrix A can be transformed, by a change of basis, into the matrix

$$\widetilde{A} = \begin{bmatrix} \lambda_1 & ? & ? & \cdots & ? \\ 0 & \lambda_2 & ? & \cdots & ? \\ 0 & 0 & \lambda_3 & \cdots & ? \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

where the λ_i are eigenvalues of $\widetilde{A}$ and $\widetilde{A}$ is in upper-triangular form. Notice that, since for defective matrices there are fewer than n distinct eigenvalues, here we will repeat each eigenvalue in the diagonal in accordance with its *multiplicity*. The multiplicity of an eigenvalue λ_i of a matrix A represents the number of times the linear factor $(\lambda - \lambda_i)$ appears in the characteristic polynomial of A. For instance, consider the defective matrix

$$D = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

A simple calculation reveals its characteristic polynomial to be

$$P_D(\lambda) = (\lambda - 1)^2.$$

Thus, we say that the eigenvalue $\lambda = 1$ of D has a multiplicity of 2 (even though the corresponding eigenspace of D is only one-dimensional). Clearly, the sum of the multiplicities of all distinct eigenvalues of an $n \times n$ matrix (such as A) will be n, so we can indeed produce the λ_i that we need for our desired form to make sense.

3 Stability

Before demonstrating how to convert an arbitrary matrix A into $\widetilde{A}$, we will first attempt to determine if the form of $\widetilde{A}$ is in fact useful. In particular, we would

like to demonstrate that if all the λ_i are such that $|\lambda_i| < 1$, that the system $\vec{x}[i+1] = \widetilde{A}\vec{x}[i]$ is stable.

Recall the definition of stability - specifically, bounded-input bounded-output stability. The definition states that the state x of a system experiencing bounded perturbations ω will be bounded by some fixed value even as time goes to infinity. In addition, we demonstrated that, for scalar systems, the state equation

$$x[i+1] = \lambda x[i] + \omega$$

is bounded exactly when $|\lambda| < 1$.

Now, we are considering the vector system

$$\vec{x}[i+1] = D\vec{x}[i] + \vec{\omega},$$

where $\vec{\omega}$ is a bounded perturbation that may vary arbitrarily with time. Expanding this equation out, we obtain the system

$$\begin{bmatrix} x_1[i+1] \\ x_2[i+1] \\ \vdots \\ x_n[i+1] \end{bmatrix} = \begin{bmatrix} \lambda_1 & ? & ? & \cdots & ? \\ 0 & \lambda_2 & ? & \cdots & ? \\ 0 & 0 & \lambda_3 & \cdots & ? \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix} \begin{bmatrix} x_1[i] \\ x_2[i] \\ \vdots \\ x_n[i] \end{bmatrix} + \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix}$$

Clearly, $\vec{x}$ is bounded exactly when all its components $\vec{x}_i$ are themselves bounded. So if we could write scalar systems describing each of the x_i, and use each system to show that its corresponding $\vec{x}_i$ were bounded, then we would be done!

Unfortunately, it is clear that almost all of the $x_i[i+1]$ depend not only on $x_i[i]$, but on some other $x_j[i]$, since $\widetilde{A}$ is not diagonal. However, notice that from the last row of our state, we may write the scalar system

$$\vec{x}_n[i+1] = \lambda_n \vec{x}_n[i] + \omega_n,$$

which, since $|\lambda_n| < 1$, we know to be stable. Thus, we know that there exists some upper bound B_n independent of i such that $|x_n[i]| < B_n$ for all $i \geq 0$. Now, we will look at the second-to-last row of our state, to obtain the state equation

$$\vec{x}_{n-1}[i+1] = \lambda_{n-1}\vec{x}_{n-1}[i] + (?)\vec{x}_n[i] + \omega_{n-1}$$

where the ? represents some unknown constant.

At first glance, it seems impossible to determine whether $x_{n-1}[i]$ will remain bounded, due to the presence of the $x_n[i]$. But now we know $x_n[i]$ is bounded by some constant B_n for all i! Therefore, we can treat the last two terms as some perturbation $w'_{n-1} = (?)x_n[i] + w_{n-1}$ which, critically, is itself bounded by some constant. Rewriting our state equation, we obtain

$$\vec{x}_{n-1}[i+1] = \lambda_{n-1}\vec{x}_{n-1}[i] + \omega'_{n-1},$$

which is a scalar system with a bounded perturbation! Since $|\lambda_{n-1}| < 1$, we know that $\vec{x}_{n-1}[i]$ in turn will be bounded by some B_{n-1} for all $i \geq 0$.

In a similar manner, considering the third-to-last row of $\vec{x}$, we obtain the state equation

$$\vec{x}_{n-2}[i+1] = \lambda_{n-2}\vec{x}_{n-2}[i] + (?_1)\vec{x}_n[i] + (?_2)\vec{x}_n[i] + \omega_{n-2} = \lambda_{n-1}\vec{x}_{n-2}[i] + \omega'_{n-2}$$

for some bounded ω'_{n-2} (since $\vec{x}_n[i]$ and $\vec{x}_{n-1}[i]$ are both bounded). Since $|\lambda_{n-2}| < 1$, $\vec{x}_{n-2}[i]$ is bounded by some B_{n-2} for all $i \geq 0$. Repeating this process, we ultimately see that all the $\vec{x}_j[i]$ are bounded, so $\vec{x}[i]$ itself is bounded!

Thus, our $\widetilde{A}$ exhibits many of the nice properties of diagonal matrices, even though it may be defective.

4 Orthonormalization

We will discuss the construction of $\widetilde{A}$ next lecture. One algorithm that we will need during the construction, that also has applications elsewhere, is known as *Gram-Schmidt orthonormalization*. Essentially, it aims to solve the following problem:

- Start with an empty subspace S.
- We will keep receiving vectors $\vec{v}$, and must return their projection onto S.
- Every so often, we will receive a vector $\vec{u} \notin S$, that must be added into S. That is to say, S must now contain the span of all the previous $\vec{u}$ added to it, as well as this new vector.

When we looked at OMP, we solved this problem in the naive manner. That is to say, we represented S as a matrix whose columns consisted of the $\vec{u}$, and projected $\vec{v}$ onto S using least squares. While this approach certainly worked, it was slow, especially when working in high dimensions.

To quantify its speed (or lack thereof), we will consider the time taken to project a vector $\vec{v}$ onto S, working in n dimensional space with $\Theta(n)$ vectors $\vec{u}$ already entered into S. To calculate this projection, we can use the least squares formula

$$\text{proj}_S(\vec{v}) = S(S^TS)^{-1}S^T\vec{v}.$$

The slowest operation in calculating this quantity is in inverting S^TS, which we know runs in $\Theta(n^3)$ time[1]. All other operations, like multiplying matrices with vectors or taking their transposes, require only $\Theta(n^2)$ time. Thus, each projection takes $\Theta(n^3)$ time overall.

Now, we will conjecture that there exists a way of representing S as a matrix that makes inverting S^TS much faster. If we could do that, then we could bring

[1] Since we must perform Gaussian elimination to calculate this inverse.

the cost of projection down from $\Theta(n^3)$ to $\Theta(n^2)$. Of course, the easiest matrix to invert is the identity matrix, so that's a good place to start. Letting the $\vec{s}_i$ be the columns of S, and setting $S^TS = I$, we obtain

$$\begin{bmatrix} - & \vec{s}_1 & - \\ & \vdots & \\ - & \vec{s}_k & - \end{bmatrix} \begin{bmatrix} | & & | \\ \vec{s}_1 & \cdots & \vec{s}_k \\ | & & | \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\implies \quad \vec{s}_i^T \vec{s}_i = 1$$
$$(\forall i \neq j) \vec{s}_i^T \vec{s}_j = 0,$$

by the definition of matrix multiplication. In other words, all the $\vec{s}_i$ are of unit length, and are mutually orthogonal to one another. Thus, S is an orthogonal matrix.

So if we could represent our subspace as the columnspace of some orthogonal matrix S, then returning the projection of some received vectors $\vec{v}$ would be trivial. But how could we add vectors $\vec{u}$ to our subspace? Clearly, simply concatenating $\vec{u}$ to the right of S would not suffice, since it is unlikely to be orthogonal to all of the pre-existing columns.

Let's establish some notation. Let $S[i]$ be the subspace before we add some vector $\vec{u}[i]$, and let $S[i+1]$ be the subspace afterwards. For convenience, we will use the same notation for the matrix representing the subspace as we do for the subspace itself. Let $\vec{s}_j$ be the columns of S, for $0 \leq j < i$. Thus, we may write our above observation using this notation as the statement that

$$S[i+1] \neq \begin{bmatrix} S[i] & \vec{u}[i] \end{bmatrix}.$$

However, observe that $\text{proj}_{S[i]}(\vec{u}[i]) = S[i]S[i]^T\vec{u}[i]$, since $S[i]^TS[i] = I$. Thus, observe that

$$\begin{aligned} S[i]^T(\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i])) &= S[i]^T\vec{u}[i] - S[i]^T\text{proj}_{S[i]}(\vec{u}[i]) \\ &= S[i]^T\vec{u}[i] - S[i]^TS[i]S[i]^T\vec{u}[i] \\ &= S[i]^T\vec{u}[i] - S[i]^T\vec{u}[i] \\ &= \vec{0}, \end{aligned}$$

so $\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i])$ is orthogonal to all the columns of S. This should not be a surprise, since the projection of a vector onto a subspace produces an error vector that is orthogonal to all vectors within the subspace.

So now, a good question to ask would be: rather than inserting $\vec{u}[i]$ into our subspace, would it be the same thing to insert $\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i])$ into the subspace instead?

From our knowledge of linear combinations, the answer should clearly be yes. To prove this, observe that $\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i])$ differs from $\vec{u}[i]$ only by some vector

that already lies in $S[i]$. Thus, it should be clear that exactly those vectors reachable by a linear combination of the vectors in $S[i]$ and $\vec{u}[i]$ are reachable by a linear combination of the vectors in $S[i]$ and $\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i])$[2].

Similarly, since $\vec{u}[i] \notin S[i]$, it is clear that the normalized vector

$$\vec{u}' = \frac{1}{\left\|\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i])\right\|}(\vec{u}[i] - \text{proj}_{S[i]}(\vec{u}[i]))$$

is defined and can also be added to $S[i]$ instead of $\vec{u}[i]$. Using our least squares formula, we may rewrite $\vec{u}'$ as

$$\vec{u}' = \frac{1}{\|\vec{u}[i] - S[i]S[i]^T\vec{u}[i]\|}(\vec{u}[i] - S[i]S[i]^T\vec{u}[i]).$$

But as $\vec{u}'$ is normalized and orthogonal to all elements of $S[i]$, it is orthogonal to all the columns $\vec{s}_j$ of $S[i]$, so we can simply concatenate it onto the end! Thus, we have

$$S[i+1] = \begin{bmatrix} S[i] & \vec{u}' \end{bmatrix}$$

that satisfies the desired properties of S.

The process we described here can also be viewed as a way of producing an orthonormal basis from a set of vectors spanning a given vector space, by inserting those vectors as our $\vec{u}[i]$ and looking at the columns of the matrix of our final subspace S. This is known as the *Gram-Schmidt* procedure, and will be useful next lecture, when we demonstrate how to produce a change of basis that yields $\widetilde{A}$ from A.

[2] Try to prove this yourself, if it's not clear!