1. Review: The Randomized "Power Method"

This section is a review from class on 02/12/2016. Let $A$ be an $m \times n$ matrix. Further, define $k$ to be our target rank and $p$ the oversampling parameter. For notational convenience, let $l = k + p$. We are seeking an approximate SVD of $A$: $A \approx UDV^*$. Recall the familiar process:

- Draw random matrix $G = \text{randn}(n,l)$
- Create sampling matrix $Y = AG$
- Form $Q = \text{orth}(Y)$
- Let $B = Q^*A$
- Calculate an SVD of $B$, $B = \hat{U}DV^*$
- Finally, $U = Q\hat{U}$

One can prove, for $q = 0$ (the number of power iterations) and $C$ a constant:

$$\mathbb{E}||A - UDV^*|| = \mathbb{E}||A - QQ^*A|| \leq C(\sum_{j > k} \sigma_j^2)^{\frac{1}{2}} \leq C(\sqrt{n} - k)\sigma_{k+1}$$

With the worst case occurring when no decay is present in the singular values past $\sigma_{k+1}$. We will now look at how these bounds change when we increment $q$. For $q > 0$ we have:

$$\mathbb{E}||A - QQ^*A|| \leq C(\sqrt{n} - k)^{\frac{1}{2q+1}}\sigma_{k+1}$$

From these bounds, we infer that the usage of power iterations can be advantageous in the reduction of expected error. Let’s take a closer look at this method.

2. Power Method

For simplicity, assume that $A$ is Hermitian ($A = A^*$, In the case where $A$ is not Hermitian, we can adapt the process to accommodate.) Consider the eigendecomposition of $A$: $A = VDV^*$ where $V$ contains the eigenvectors of $A$ and $D$ is diagonal whose elements are the ordered eigenvalues of $A$ ($|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$). With this, we can compute different integer powers of our matrix $A$:

2.1. Powers of $A$.

$$A^2 = AA = VDV^*VDV^* = VDIDV^* = VD^2V^*$$
$$A^3 = (A^2)A = (VD^2V^*)VDV^* = VD^2IV^* = VD^3V^*$$
$$\vdots$$
$$A^q = VD^qV^*$$

And so, if $\{\lambda, v\}$ is an eigenpair of $A$ then $\{\lambda^q, v\}$ is an eigenpair of $A^q$. Suppose we seek to approximate the dominant eigenvector of $A$, say $v_1$.

2.2. Classical Power Iterations.

- Draw starting vector $g \in \mathbb{R}^n$. A common choice is to choose $g$ from a Gaussian distribution, but this is not a requirement.
Let:
\[ y_1 = A g \]
\[ y_2 = A y_1 = A^2 g \]
\[ y_3 = A y_2 = A^3 g \]
\[ y_4 = A y_3 = A^4 g \]
\[ y_5 = A y_4 = A^5 g \]
\[ \vdots \]

This says that \( y_n \) will get closer to alignment with \( v_1 \) as \( n \) is incremented. To see why it works, write \( g = g_1 v_1 + g_2 v_2 + \ldots + g_n v_n \) (works since \( \{v_i\}_{i=1}^n \) forms an orthonormal basis). Then \( y_q = A^q g = g_1 \lambda_1^q v_1 + g_2 \lambda_2^q v_2 + \ldots + g_n \lambda_n^q v_n \). If \( |\lambda_1| > |\lambda_2| \), the first term, \( g_1 \lambda_1^q v_1 \), will dominate as \( q \) increases (which of course can go wrong if \( g_1 = 0 \)).

**Theorem 1.** Suppose \( \lambda_1 > 0 \) and \( |\lambda_1| > |\lambda_2| \), then \( \frac{y_q}{||y_q||} \rightarrow \pm v_1 \) as \( q \rightarrow \infty \).

The proof of this is left as an exercise for the reader. Upon closer inspection of this process, it is clear there are some drawbacks. Used as a numerical method, it can be rather primitive.

### 2.3. Drawbacks and Remedies.

- If \( |\lambda_1| \approx |\lambda_2| \) the rate of convergence can be quite slow
- A needs to be accessed many different times
- An unlucky draw of \( g \) can yield a small \( g_1 v_1 \) which will result in a large number of iterations required.
- Quite inefficient if you desire more than one eigenvector

These concerns can be ameliorated by choosing multiple starting vectors.

- Draw \( l \) starting vectors \( g_{i=1}^l \in \mathbb{R}^n \). Let \( G = [g_1, g_2, \ldots, g_l] \).
- Let:
  \[ Y_1 = AG \]
  \[ Y_2 = AY_1 = A^2 G \]
  \[ Y_3 = AY_2 = A^3 G \]
  \[ Y_4 = AY_3 = A^4 G \]
  \[ Y_5 = AY_4 = A^5 G = [A^3 q_1, A^3 q_2, \ldots, A^3 q_l] \]
  \[ \vdots \]

When performing this, one needs to be quite careful, round-off errors can hurt you!
2.4. Example 1: Let $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \beta \end{bmatrix}$ where $1 > \alpha > \beta \geq 0$

The eigenpairs of $A$ are easily calculated as:

$\{\lambda_1, v_1\} = \{1, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\}$, $\{\lambda_2, v_2\} = \{\alpha, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\}$, $\{\lambda_3, v_3\} = \{\beta, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\}$

Let us try to calculate $v_1$ and $v_2$ via the proposed remedy to our drawbacks. We run the scheme and find:

$Y_q = A^q G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha^q & 0 \\ 0 & 0 & \beta^q \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \\ g_{31} & g_{32} \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ \alpha^q g_{21} & \alpha^q g_{22} \\ \beta^q g_{31} & \beta^q g_{32} \end{bmatrix}$

In precise arithmetic, there are no issues, we are successful! However, in floating point arithmetic, we are far from successful. Recall, $|\alpha|, |\beta|$ are both smaller than 1, suppose $q$ is large enough to force $\alpha^q < \epsilon_{machine} \approx 10^{-16}$ (say $\alpha = 0.1$, $q = 20$). In this case, since $\beta < \alpha$, we have:

$Y_q = \begin{bmatrix} g_{11} & g_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$

This successfully captures $v_1$ but yields no information for $v_2$. Once, again, this can be fixed! To do so, we must orthonormalize between each iteration.

- Draw $l$ starting vectors $g_{i=1}^l \in \mathbb{R}^n$. Let $G = [g_1, g_2, \ldots, g_l]$.
- Let:

  $Y_1 = AG$
  $Q_1 = \text{orth}(Y_1)$

  $Y_2 = AQ_1$
  $Q_2 = \text{orth}(Y_2)$

  $Y_3 = AQ_2$
  $Q_3 = \text{orth}(Y_3)$

  $\vdots$

We end this lecture with a theorem:

**Theorem 2.** $\text{Col}(Y_q) = \text{Col}(A^q G)$ in exact arithmetic

The proof of which is too small to be contained within the margin...(possibly next lecture?)