

CS205 – Class 6

Reading: Heath 3.6 (p137-143), 4.7 (p202)

Singular Value Decomposition (SVD) contd.

1. SVD is a transformation into a diagonal axis aligned space.
 - a. Transform b into the space spanned by U^T , $U^T U \Sigma V^T x = \Sigma V^T x = U^T b = \hat{b}$. No information is lost going from b to \hat{b} because U^T is square and orthogonal.
 - b. Replace $V^T x$ by \hat{x} to get a diagonal system, $\Sigma V^T x = \Sigma \hat{x} = \hat{b}$.
 - c. Now solve the system $\Sigma \hat{x} = \hat{b}$ simply by scaling elements of \hat{b} by the singular values.
 - d. The original x is then recovered as $x = V \hat{x}$.
 - e. Essentially the SVD solves the matrix by transforming the vectors in a space with eigenvectors along the unit axis.

$$2. A = U \Sigma V^T = \sum_i \sigma_i u_i v_i^T$$

proof: define $l = \min(m, n)$, \hat{U} the first l columns of U , $\hat{\Sigma}$ the square $l \times l$ submatrix from the upper left corner of Σ , \hat{V} the first l columns of V . Then

$$A = U \Sigma V^T = \hat{U} \hat{\Sigma} \hat{V}^T = \begin{pmatrix} u_1 & \cdots & u_l \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_l \end{pmatrix} \begin{pmatrix} v_1^T \\ \vdots \\ v_l^T \end{pmatrix} = \begin{pmatrix} u_1 & \cdots & u_l \end{pmatrix} \begin{pmatrix} \sigma_1 v_1^T \\ \vdots \\ \sigma_l v_l^T \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^l \sigma_i u_i^1 v_i^1 & \cdots & \sum_{i=1}^l \sigma_i u_i^1 v_i^n \\ \vdots & & \vdots \\ \sum_{i=1}^l \sigma_i u_i^m v_i^1 & \cdots & \sum_{i=1}^l \sigma_i u_i^m v_i^n \end{pmatrix} = \sum_{i=1}^l \begin{pmatrix} \sigma_i u_i^1 v_i^1 & \cdots & \sigma_i u_i^1 v_i^n \\ \vdots & & \vdots \\ \sigma_i u_i^m v_i^1 & \cdots & \sigma_i u_i^m v_i^n \end{pmatrix} = \sum_{i=1}^l \sigma_i u_i v_i^T$$

- a. Note that “zero” or “small” σ_i produce terms that contribute little to the sum, and that large σ_i produce terms that contribute significantly to the sum.
 - b. If the “zero” or “small” σ_i are omitted from the summation, one obtains a matrix with lower rank. For example, if only the first k terms are summed, the result has rank k .
 - i. Moreover, it can be shown that this new rank k matrix is the closest rank k matrix to A in both the L_2 and the Frobenius norm.
 - ii. This is the key idea in PCA, clustering/data mining algorithms, etc.
3. The “pseudo-inverse” of a matrix A is defined by $A^+ = V \Sigma^+ U^T$ where Σ^+ is obtained from Σ by replacing all “nonzero” σ_i with $1/\sigma_i$, and leaving all the zero entries *identically zero*.
 - a. If A is square and nonsingular ($\sigma_i \neq 0$), $A^+ = A^{-1}$.
 - b. The least squares solution to $Ax=b$ is $x = A^+ b = V \Sigma^+ U^T b = \sum_{\sigma_i \neq 0} (u_i^T b / \sigma_i) v_i$. (Note Σ^+ contains a transpose)

- i. Moreover, small σ_i can be dropped from the summation stabilizing the solution, and effectively improving the condition number. This amounts to “dropping columns” from the original matrix A.

Numeric Linear Algebra Summary

When your matrix A is:

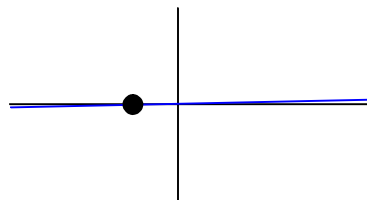
1. Non-singular use LU decomposition.
2. Over determined use QR with Householder.
3. Under determined use SVD. This means some of your variables don't have any meaning on your solution, i.e. where you parked your car.
4. Principal Component Analysis
 - a. PCA let's you throw away 10,000 terms and keep 6.
 - b. But **don't** use the SVD (too slow and gives you everything)! Instead, find your singular values (i.e. σ_i) using the power method. Then can get eigenvectors of $A^T A$ and AA^T using division.

Covered in class: 1, 4, 5, 7, 8

Reading: Heath 5.1-5.5

Nonlinear Equations

1. Nonlinear equations are much more difficult to solve than simple linear equations, thus we will move from systems of equations to scalar equations to get started.
 - a. Before we move from linear to nonlinear equations, let's first move from systems of equations to scalar equations in the linear case to get warmed up.
 - i. consider $ax=b$ where a and b are merely numbers
 1. of course the solution to this is simply $x=b/a$
 2. the only worry here would be if a was small or “zero” in which case the $\det(A)=\det(a)$ is zero and the matrix [a] is essentially singular or near singular
 - ii. let's take a different, functional look at this
 1. assume that we had a straight line $y=ax+b$ and that we wanted to find the roots where $y=0$, then we need to solve $0=ax+b$ or $ax=-b$
 2. This is our linear equation with solution $x=-b/m$ and now we see that a small slope is equivalent to ill-conditioning
 - a. Consider $y=(1e-16)x-1e-16$ where the root is $x=1$ but the division $x=(1e-16)/(1e-16)$ looks like $0/0$ and is indeterminate and full of errors.



- b. Now for nonlinear equations we are solving $A(x)=b$ where A is a vector valued function of x.
 - i. Moreover we can write $A(x)=0$ since the b term can be incorporated into the vector valued function A of x.

- ii. As a scalar equation, we have $a(x)=0$ which looks odd, so we switch notation to $f(x)=0$.
 - iii. Note how this looks like a root finding problem. In general, any $g(x)=b$ can be rewritten as $f(x)=0$ by moving b to the left hand side.
 - 1. There may be any number of roots from 0,1,2, etc. to infinity.
 - 2. Remember “roots”=“solutions” from here on out.
 - iv. Since $f'(x^*)$ is the slope of the function f at and “near” a root x^* , $f'(x^*)$ gives us an indication of the condition number. That is, it takes the place of “a” in the linear case.
 - 1. If $f'(x^*)$ is small or “zero” then we could be in trouble. That is, the condition number is large. Moreover the function is locally *flat*.
 - 2. A simple root has $f'(x^*) \neq 0$.
 - 3. A multiple root has $f'(x^*) = 0$ which could be problematic.
2. Previously when looking at systems of linear equations, we introduced direct methods like Gaussian Elimination and the Cholesky factorization. The other kind of method is called an iterative method where one starts with an initial guess x_1 and iterates through a sequence $x_1, x_2, x_3 \dots$ ending up with a final guess of x_m .
- a. Issues here include both finding an initial guess x_1 and deciding on a stopping criterion that says that x_m , for some m , is good enough.
3. We usually solve nonlinear equations with iterative methods: $x_1, x_2, x_3 \dots x_n$ stopping when the error, $e_k = x_k - x_{exact}$, is small, i.e. when $\|e_k\| < \varepsilon$.
- a. The **convergence rate** is determined by looking at the equation $\|e_{k+1}\| = C\|e_k\|^r$ with $C \geq 0$ in the limit as $k \rightarrow \infty$.
 - i. If $r = 1$, then we need $C < 1$ to guarantee convergence. In this case we say that the convergence rate is *linear*.
 - ii. If $r > 1$ we say that the convergence rate is *superlinear*.
 - iii. If $r = 2$ we say that the convergence rate is *quadratic*.
 - iv. These terms state how fast we converge once convergence is occurring. However, there is no guarantee of converging to the root we want, or to any root in general.
 - 1. Moreover, recall that our nonlinear function might be an approximation to what we really want, so how many digits of accuracy we need is important in deciding how fast things need to converge.
4. **fixed point iteration** - iterate $x_{k+1} = g(x_k)$ to find $x = g(x)$
- a. *locally convergent* if $|g'(x^*)| < 1$, i.e. if the initial guess x_o is *close enough* to x^* the method will converge
 - i. $e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*) = g'(\theta_k)(x_k - x^*) = g'(\theta_k)e_k$ for θ_k between x_k and x^* as determined by the Mean Value Theorem. *Show that if $g'(x^*) = 0$ and $g''(x)$ is bounded, then convergence is in fact quadratic*
 - ii. if all the $|g'(\theta_k)| \leq C < 1$, then $|e_k| \leq C|e_{k-1}| \leq C^2|e_{k-2}| \leq \dots \leq C^k|e_o|$ so that as $k \rightarrow \infty$, $C^k \rightarrow 0$ and $|e_k| \rightarrow 0$
 - b. only converges if the initial guess is *close enough* to the solution
 - c. functions may be *very expensive* to evaluate - try to evaluate them as few times as possible and save the old values for future use

5. **Newton's method** $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
- careful when $f'(x^*) = 0$, i.e. poorly conditioned
 - stop when $|f(x_k)| < \varepsilon$ which is equivalent to $|x_{k+1} - x_k| < \frac{\varepsilon}{|f'(x_k)|}$
 - uses the tangent line to estimate the zero - *draw a picture of this*
 - quadratic convergence in general, but only linear convergence for multiple roots.
 - both $f(x_k)$ and $f'(x_k)$ need to be evaluated each iteration and $f'(x_k)$ may be expensive. Think of $f(x)$ possibly being a computer program that is difficult to get derivatives from.
6. **secant method** $x_{k+1} = x_k - f(x_k) \left(\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right)$
- Newton method with the slope estimated by a secant line $f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$
 - don't need to evaluate $f'(x_k)$
 - moreover only one $f(x_k)$ evaluation for each iteration
 - other secant methods that do not use two previous iterates to compute the secant line are twice as costly for function evaluations
 - superlinear convergence with $r = 1.618$
 - almost always beats Newton since only a couple more iterations are needed at half the cost
7. **bisection method** – guaranteed to converge to a root in the interval, as opposed to the fixed point iteration methods, e.g. Newton and secant
- find an interval $[a, b]$ with $f(a)f(b) < 0$ that contains the solution
 - note that $f(a)f(b) = 0$ implies $f(a) = 0$ or $f(b) = 0$ and we're done
 - find the midpoint $m = (a + b)/2$, and then if $f(a)f(m) < 0$ set $b = m$, otherwise set $a = m$ since $f(b)f(m) < 0$,
 - note that $f(m) = 0$ implies we are done
 - given an error tolerance ε , continue until $b - a < \varepsilon$
 - the interval size decreases by $1/2$ each iteration, so it is linearly convergent with $C = 1/2$
 - only $f(m)$ needs to be evaluated each iteration
 - mixed methods** – e.g. bisection for safety and secant for speed
 - start with a bracketing interval $[a, b]$ and set $x_0 = a$ and $x_1 = b$
 - apply the secant method to find x_{k+1} from x_k and x_{k-1}
 - if $x_{k+1} \in [a, b]$ set $m = x_{k+1}$ and update the bracketing interval $[a, b]$ using the bisection method, that is if $f(a)f(m) < 0$ set $b = m$, otherwise $f(b)f(m) < 0$ where you set $a = m$
 - if $x_{k+1} \notin [a, b]$ or $f(b) - f(a)$ is too small to apply the secant method, apply the bisection method with the usual $m = (a + b)/2$ and then set $x_k = a$ and $x_{k+1} = b$
 - note that $x_{k-1} = a$ and $x_k = b$ implies that x_{k+1} comes from linear interpolation
 - needs only one $f(x_k)$ function evaluation per iteration