12/4/19

8803 Learning from Data: Lecture 26

- Last day of class! Reminder: Your SEI comments will be valuable for potential future editions of this class.
- I'm out of town next Monday through Wednesday, but otherwise available for consultation on project notebooks or other assignments.

- Instant feedback question: The Graduate Studies Committee is discussing the core courses and a current issue is integrating more computational aspects in these courses. A faculty survey indicates the large majority of faculty favor such integration but was unclear of how to do it.
  - Based on your experience, what would you think of Jupyter notebooks as a vehicle?
  - Eg., a repository of notebooks available for instructors who may be inexpert in Python and computational aspects, but the notebooks would be self-contained.
  - Bootcamp for incoming students to get everyone up to speed on Python and Jupyter notebooks.
  - Experienced students available as consultants.

- Today: An introduction to singular value decomposition via a Jupyter notebook: regression_parameter_estimation/linear_algebra_games_including_SVD.ipynb
  - We won't be able to look at everything (or prove everything) in details, so we will start with the highlights and fill in what we can.

note: $(\theta)_j (A^T)_{jl} (\Sigma^{-1})_{ll'} (A)_{l'j'} (\theta)_{j'} = \theta_j A_{ij} \Sigma^{-1}_{ll'} A_{i'j'} \theta_{j'}$

12/4/19

Preliminary exercise: linear algebra using the index form. Recall $(AB)^T = B^T A^T$, $(A^T)_{ji} = A_{ij}$.

$$\chi^2 = (Y - A\theta)^T \Sigma^{-1} (Y - A\theta) = (Y_i - A_{ij}\theta_j) \Sigma^{-1}_{ii'} (Y_{i'} - A_{i'j'}\theta_{j'})$$

i, j, i', j'

- summed over i, j, i', j' because they each appear twice.
- $\Sigma^{-1}_{ii'}$ means $(\Sigma^{-1})_{ii'} \neq (\Sigma_{ii'})$
- χ^2 is a scalar because no free indices for a vector, matrix, tensor

$$\frac{\partial \chi^2}{\partial \theta_k} = \left[\frac{\partial}{\partial \theta_k}(Y_i - A_{ij}\theta_j)\right] \Sigma^{-1}_{ii'} (Y_{i'} - A_{i'j'}\theta_{j'}) + (Y_i - A_{ij}\theta_j) \Sigma^{-1}_{ii'} \frac{\partial}{\partial \theta_k}(Y_{i'} - A_{i'j'}\theta_{j'})$$

assuming only θ_k dependence is explicit [Y, A, Σ don't depend on θ]

$$= (-A_{ij}\delta_{jk}) \Sigma^{-1}_{ii'} (Y_{i'} - A_{i'j'}\theta_{j'}) + (Y_i - A_{ij}\theta_j) \Sigma^{-1}_{ii'} (-A_{i'j'}\delta_{j'k}) = 0$$

$-A_{ik}$ $\qquad$ $-A_{i'k}$

Claim: this is same as $2(A^T \Sigma^{-1} Y) = 2(A^T \Sigma^{-1} A)\theta$

Do the left side only: $A_{ik} \Sigma^{-1}_{ii'} Y_{i'} + Y_i \Sigma^{-1}_{ii'} A_{i'k}$

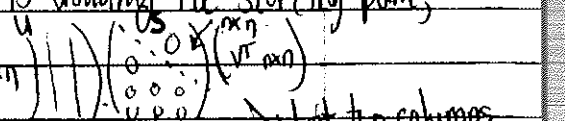
(rest as an exercise or see Lecture 11)

$(A^T)_{ki} (\Sigma^{-1})_{ii'} Y_{i'}$ $\qquad$ $(A^T)_{ki'} (\Sigma^{-1})^T_{i'i} Y_i$

$= (A^T \Sigma^{-1} Y)_k$ $\qquad$ since $(\Sigma^{-1})^T = \Sigma^{-1}$ (can you prove this?)

Main point of SVD is the decomposition of a matrix into three other matrices.

- We can do full SVD or reduced SVD. Former is usually the starting point, but latter is used in practice. Full $m > n$



last two columns in U always multiply zeros in S, so take $U(m \times m) \to U(m \times n)$, $S(m \times n) \to S(n \times n)$, $V^T(n \times n) \to V^T(n \times n)$

- Reduced form is

$$A = U S V^T \quad \text{with } S \text{ diagonal}$$

$m \times n$ $\quad$ $(m \times n)(n \times n)(n \times n)$ $\quad$ reduced is unchanged content

- Elements of $S \Rightarrow S_{kk'} \equiv S_k \delta_{kk'}$ are singular values.
- key feature: $A_{ij} \approx \sum_{k=1}^{\rho} U_{ik} S_k V_{jk}$ with $\rho < n \Rightarrow$ truncated representation with most of the content if "dominant" ρ vectors kept. (ρ ← explicit sum now) $\quad$ similar for $n > m$

12/9/19

- Solving matrix equations with ill-conditioned matrices, which means that the smallest eigenvalue is zero or close to zero. (If zero, then singular.)
  - The condition number of a matrix is the ratio of largest to smallest eigenvalue.
  - If solving $AX = b \Rightarrow x = A^{-1}b$ then error in b is magnified by condition number to give error in X. So finite precision in b leads to nonsense for X if condition number is larger than machine or calculational precision. $X(A) = 10^k$ means up to k digits of accuracy lost (roughly).
    - condition number

- Data reduction application: identify the important basis elements → eg what linear combinations of parameters are most important.
  - Python ⇒ note matrix multiplication with @, transpose with T.
  - Step through the Python example for image compression. You are identifying the most important eigenvalues for the image.
  - If we keep 1 in 10 of singular values, then numbers to be stored are significantly reduced.
  - Note the spectrum of eigenvalues.

Covariance, PCA and SVD

- Consider the covariance matrix and find eigenvalues and eigenvectors. "Center" the data first (subtract means)
- PCA means to rank these. Typically done via SVD. ← better numerically
- Then use fewer singular values (which are the eigenvalues for a square matrix) ⇒ we have reduced the number of parameters in a model (for example).