MSA220/MVE440 Statistical Learning for Big Data

Lecture 10

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

Data representations

With big data we often need to find efficient data representations of a smaller dimension for both visualization and computation.

- SVD, PCA
- SOM
- MDS, IsoMap, NMF

SVD

SVD (singular value decomposition) is a workhorse that underpins much of the modeling we do!

- Data matrix X of dimension $n \times p$
- Before you do anything, you want to center and scale the columns of X!!!
- Otherwise the scale of individual variables dominate the representation and visualization is weird without centering
- We want to approximate the observations x_i in X by a lower-rank model
- Find the lower-rank model V_q to minimize the L2 error

$$\sum_{i=1}^{N} ||x_i - V_q \lambda_i||^2$$

where V_q is a $p \times q$ has *orthogonal* columns and λ_i is the variable specific coefficient

SVD

- If we knew V_q we can easily solve for λ in the problem below

$$\sum_{i=1}^{N} ||x_i - V_q \lambda_i||^2$$

- $\lambda_i = V_q^T x_i$
- Now given λ we want to find V_q:

$$\sum_{i=1}^{N} ||x_i - V_q V_q^T x_i||^2$$

- $V_q V_q^T = H_q$ is a projection matrix that maps x_i onto the space spanned by columns in V_q (this btw looks a lot like regression, yes?)
- The solution to the problem is the svd of $X = UDV^T$ where V_q is the first q columns of V

SVD

- $X = UDV^T$ where U is a $n \times p$ matrix, D is a diagonal $p \times p$ matrix and V is a $p \times p$ matrix where $U^T U = I$, $V^T V = I$
- We can also write $VX = UD$
- UD are called the *principal components*
- VX is the rotation V applied to the data X to project it onto the principal component space.
- The entries of each column in V are called *loadings* and tell you how much each original variable contribute to the make-up of the new dimension in PC space
- The leading components in V correspond to the largest values of D

SVD

- Another way of looking at SVD is building a structure from orthogonal components. To see this write

$$X = UDV^T = \sum_{j=1}^{P} d_j u_j v_j^T$$

  where u_j is a $n \times 1$ vector and v_j^T is a $1 \times p$ vector.
- Each produce $u_j v_j^T$ construct a $n \times p$ matrix representation of X
- Scaled by d_j they represent approximation of X in orthogonal directions.
- The first component is the best rank 1 approximation of X

- Best rank q approximation

$$X_q = \sum_{j=1}^{q} d_j u_j v_j^T$$

with approximation error

$$||X - X_q||^2 = || \sum_{j=q+1}^{p} d_j u_j v_j^T ||^2 = \sum_{j=q+1}^{p} d_j^2$$

SVD and regression

- Least squares modeling

$$\min_{\beta} ||Y - X\beta||^2$$

- The LS solution $\beta = (X'X)^{-1}X'Y$
- If we plug in $X = UDV^T$ in the above expression we get

$$\beta = (VDU'UDV')^{-1}VDU'Y =$$

$$= (VD^2V')^{-1}VDU'Y = VD^{-2}V'VDU'Y = VD^{-1}U'Y$$

- The expression $VD^{-1}U'$ is called the *pseudo-inverse* of X
- Notice then that the regression coefficients are really obtained through SVD
- Fitted values $\hat{y} = X\hat{\beta} = UDV'VD^{-1}U'Y = U(U'Y)$
- U is the ortogonal basis the spans the columns of X and regression projects onto these components

SVD and classification

- In classification with LDA we sphered the data using U
- We classified using the manahalobis distance

$$c(x) = \arg\min_c (x - \mu_c)' \Sigma^{-1} (x - \mu_c)$$

- We could write $X'X/n = \hat{\Sigma} = (VD^2V')/n$
- And so we can write

$$c(x) = \arg\min_c (V'(x - \mu_c))' D^{-2} (V'(x - \mu_c))$$

- Since the matrix D is diagonal, the sphered data is much easier to work with - just look at one "variable" at a time in this space

SVD and ridge-regression

- $X = UDV' = RV'$ where $R = UD$
- We can write the ridge-regression estimate as

$$\beta_r = (X'X + \lambda I)^{-1} X'y = V(R'R + \lambda I)^{-1} R'y$$

- So β_r is $V\theta$ where θ is the ridge-regression on R instead of X
- Can work in this space instead to select penalty parameters etc.

SVD and ridge-regression

- $X = UDV' = RV'$ where $R = UD$
- We can write the ridge-regression estimate as

$$\beta_r = (X'X + \lambda I)^{-1} X'y = V(R'R + \lambda I)^{-1} R'y$$

- So β_r is $V\theta$ where θ is the ridge-regression on R instead of X
- Can work in this space instead to select penalty parameters etc.

SVD and data representation

- SVD is a component in many methods as you saw above
- We can also use it for data exploration
- We plot the principal components $XV = UD$ for the leading components and since these preserve most of the information in X we get a dense summary of the data
- Excellent for finding groups in the data
- The loadings in V tells you in which variable set the information about X resides
- If PC1 and PC2 separates groups in the data, check which variables contribute to these (loadings in the 1st columns of V)

sparse SVD

- Like in regression, it is not always easy to see which variables contribute to the PCs
- We can look for large factor loadings....
- OR we can adapt SVD to generate sparse V where only few variables do contribute

sparse SVD

- A couple of different variants of sparse SVD have been proposed
- ScotLASS (Joliffe et al), sparse PCA (Witten et al) and sparse SVD (Zou et al) are a few
- They add an L1 penalty to the factor loadings, but how the problem then is solved is different

sparse SVD

- We want to find a sparse SVD
- Let's for now assume we have $X = UDV'$ and call the principal components $Z = UD$ and the loadings are in V
- Let's look at the ith PC $Z_i = U_i D_{ii}$
- Ridge-penalty

$$\min_{\beta} ||Z_i - X\beta||^2 + \lambda||\beta||^2$$

- Solve the ridge-problem

$$\beta_r = (X'X + \lambda I)^{-1} X' U_i D_{ii} = V(D^2 + \lambda I)^{-1} V' VDU' U_i D_{ii} =$$

$$= V(D^2 + \lambda I) DU' U_i D_{ii} = V_i \frac{D_{ii}^2}{D_{ii}^2 + 1}$$

- Which means that $V_i = \beta_r / ||\beta_r||$

sparse SVD

- Recall the elastic net formulation

$$||Y - X\beta||^2 + (1-\alpha)\lambda||\beta||^2 + \alpha\lambda||\beta||$$

- Now we add the L1 penalty to the above to get sparse loadings
- Of course, in this formulation we needed to already have the SVD - iterative method.

- Alternatively, write the whole problem as follows

$$\sum_{i=1}^{N} ||x_i - AB^T x_i||^2$$

  where $A'A = I$ and $B \propto V$ and elastic net penalty on B
- $B = A$ and $\lambda = 0$ this is just the standard PCA problem $\min ||X - AA^T X||^2$
- We will solve for A with B fixed and v.v.
- Given A, we solve for B using elastic net

$$\min_B ||X - XBA^T||^2 + pen(B) = ||XA*||^2 + ||XA - XB||^2 + pen(B)$$

  where $A*$ orthornormal to A.
  This is just a bunch of independent elastic-net problems!!
- A given B

$$\min_A ||X - (XB)A^T||^2$$

sparse SVD

- A given B

$$\min_A ||X - (XB)A^T||^2$$

- That is, find the rotation A to make the data sets X and XB as similar as possible
- This is also solved by an SVD
- Let $SVD(X'(XB)) = UDV^T$ then $A = UV^T$ is the best rotation (see paper for details).
- And not we iterate until convergence

sparse SVD

- `PMA` package and `nsprcomp` package
- Difficult to choose how much to penalize
- but good for visualization and exploration

Self-organizing maps

- We can visualize large data set by looking at the leading principal components
- SOM - self-organizing maps is a very different way of looking at data
- We construct an artificial lower dimensional space where to explore the data

Self-organizing maps

- We construct a rectangular grid of *prototypes* m_j
- The prototypes live in the higher p-dim space but are parameterized by grid-points in a (usually) 2-dim space
- We can initialize with m_j in the two-dimensional space from the leading PC - i.e. draw up a rectangular grid in the PC1-PC2 plot and let m_j be the grid-coordinate points.
- We're now going to update the prototypes to better summarize the data, which corresponds to bending the PC plane to be able to map it to a rectangular grid.
- For each observation x_i we find the closest (euclidean distance) prototype m_j
- For all neighbors (on the grid) m_k of m_j we move them toward x_i (in p-space):

$$m_k = m_k + \alpha(x_i - m_k)$$

- α is the learning rate

- We can be a bit more clever with the updating, taking neighborhood distance into account
- We can also use *supervised* techniques, where some variables (dimensions) matter more in the distance calculation and other distance metrics can be used (more appropriate for categorical data).
- PRO: simple to use and interpret and customize (distance metrics to use)
- CON: need to revisit data points in update so problem with big n. May not be sufficient to visualize data in 2 dimensions when p is large