STAT 214 Spring 2025 Week 4

Austin Zane

Based on Anthony Ozerov’s STAT 215A Materials.

Required prep - setting up computing resources

We will use computing resources provided by NSF ACCESS, a program which gives access to several high-performance computing platforms.

In the next discussion session, I will help you access Bridges-2.

Before that happens, we need to set up your accounts.

Required prep - setting up computing resources

1. Make account with NSF ACCESS here:

   `operations.access-ci.org/identity/new-user`

2. Click “**Register with existing identity**”

3. Under “**Select an Identity Provider**”, select “**University of California, Berkeley**”

4. Click “**LOG ON**” and follow the steps to make an account.

5. Once you have an account, send email to GSIs with your ACCESS username.

   austin.zane@berkeley.edu, aa3797@berkeley.edu, zachrewolinski@berkeley.edu

6. After we add you, the Pittsburgh Supercomputing Center will send you an email with account info.

Required prep - setting up computing resources

If you somehow didn’t complete these steps, check the “Making an account” section of:

`stat-214-gsi/computing/psc-instructions.md`

Dimension-Reduction

Why do we do dimension-reduction?

Why do we do dimension-reduction?

- Data visualization
  - How do you visualize a 50-dimensional dataset??
- Alleviating the *curse of dimensionality* when plugging data into downstream models—both computational and statistical issues.
  - e.g. we can't run a linear regression with 500 observations and 1000 features
  - High-dimensional data are sparse, which leads to poor performance in traditional stats/ML methods
  - Reduce size of model.
- Want data to lie in low-dimensional subspace.

Dimension-reduction methods

Linear

- PCA
- Random projections
- NMF

Nonlinear

- Kernel PCA
- Autoencoders
- Isomap
- t-SNE
- UMAP
- Laplacian eigenmap
- etc…

PCA

https://vdsbook.com/06-pca

PCA: what is it?

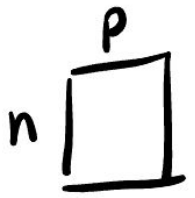
PCA: preprocessing

- **Mean-centering:** shifting a data variable to make its mean 0.
  - PCA identifies directions with maximum variance *with reference to the origin*.
  - If 0 is not meaningful, it may make sense to mean-center.
  - If absolute scale matters, it may not make sense to mean-center.
- **SD-scaling:** scaling a data variable to make its standard deviation 1.
  - If I change the units of variable A from kilometers to meters, the first principal component will probably be pretty much the “A” axis. This doesn’t make sense, and it is why we might want to scale the variables to put them all on a level playing field.
  - If all of the variables are on a common scale (e.g. binary), it may not make sense to rescale them, as the scaling will be for a variable which is almost all-zeros than for one which is evenly split between 0s and 1s.

If unsure, do something that makes sense, then do a stability check! Are the results sensitive to these preprocessing choices?

Some PCA math

$\underset{n\times p}{X}$: data matrix



SVD: $\underset{n\times p}{X} = \underset{n\times n}{U}\ \underset{n\times p}{D}\ \underset{p\times p}{V^T}$

U & V are orthogonal

- $U^TU = I_{n\times n}$
- $V^TV = I_{p\times p}$

(assuming $n \geq p$)

D is diagonal

$$\underbrace{\begin{bmatrix} d_1 & 0 & 0 \\ & d_2 \ \ddots & 0 \\ 0 & 0 & d_p \\ 0 & 0 & 0 \end{bmatrix}}_{n\times p}$$

Some PCA math

Sample covariance: $\frac{1}{n} X^T X$

$$\underset{p\times p}{X^T X} = V D^T \underbrace{U^T U}_{I_{n\times n}} D V^T = \overbrace{\underset{p\times p}{V} \underset{p\times p}{D^2} \underset{p\times p}{V^T}}^{\text{eigendecomposition of } X^T X}$$

V: matrix of eigenvectors

$\underset{p\times p}{D^2}$: diagonal of eigenvalues

$\underset{n\times p}{X} \underset{p\times p}{V} \leftarrow X$ rotated such that all variables are orthogonal

$$(XV)^T XV = V^T X^T X V = \ldots = V^T V D^2 V^T V = \underbrace{D^2}_{\text{diagonal}}$$

Some PCA math

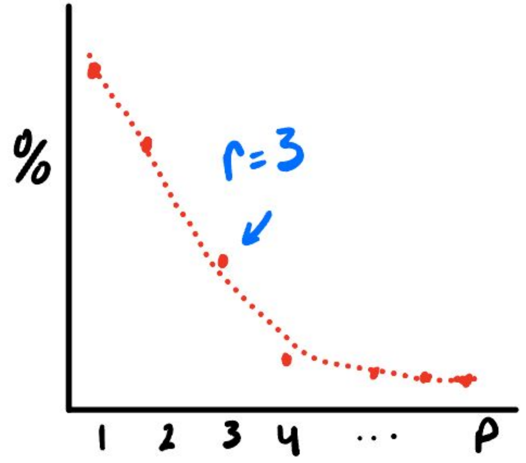
$$V = \underset{p\times p}{\begin{bmatrix} | & & | \\ v_1 & \cdots & v_p \\ | & & | \end{bmatrix}}, \quad D^2 = \underset{p\times p}{\begin{bmatrix} d_1^2 & & 0 \\ & \ddots & \\ 0 & & d_p^2 \end{bmatrix}}$$

v_k = kth principal component (PC)
d_k^2 = variance explained by kth PC

$$\frac{d_k^2}{\sum_{i=1}^{p} d_i^2} = \text{prop. of variance explained by } k\text{th PC} \qquad d_k^2 = v_k X^T X v_k$$

$$\text{Choose } r < p, \quad \tilde{V} = \underset{p\times r}{\begin{bmatrix} | & & | \\ v_1 & \cdots & v_r \\ | & & | \end{bmatrix}} \longrightarrow \underset{n\times r}{Z} = \underset{n\times p}{X}\,\underset{p\times r}{\tilde{V}}$$

(Optional) Rotate back: $\underset{n\times p}{\tilde{X}} = \underset{n\times r}{Z}\,\underset{r\times p}{\tilde{V}^T} = \underset{n\times p}{X}\,\underset{p\times r}{\tilde{V}}\,\underset{r\times p}{\tilde{V}^T}$



When might PCA not work well?

- Data has a structure (“lies on a manifold”) which can’t be linearly transformed to a low-dimensional space
  - e.g. if the data is on the shell of a 3D ball, how can we represent that in 2D??
- Data is non-gaussian
  - The good theoretical properties of PCA for data from a multivariate gaussian don’t apply here
  - If distribution is heavy-tailed, outliers can have a big effect
  - Still can work well in practice (lack of good theoretical properties does not imply poor performance)

See:

`stat-214-gsi/discussion/week4/pca.ipynb`

for some PCA examples and practice, if you would like.

NMF

$$\text{NMF}: \min_{W \geq 0, H \geq 0} L\left(\underset{n \times p}{X}, \underset{n \times k}{W} \underset{k \times p}{H}\right)$$

The transformed data matrix

Difference between different kinds of NMF lies in:

- The optimization algorithm. This is a difficult non-convex optimization problem and different algorithms (or different random seeds) can give different results
- The loss function L (e.g. it can be the norm of X minus WH)

Nonnegative Matrix Factorization (NMF)

Advantages

- “Soft” clustering in W (each observation is decomposed into positive components of different features)
- Inherently gives sparse feature matrix H
- Great for positive data!

Disadvantages

- Extremely nontrivial optimization problem—results depend on initialization and algorithm details
- Components not ordered and not nested—changing number of components can significantly change results
- Cant find the “strength” of patterns as we can in PCA

Choosing the number of dimensions to reduce to

- If you are doing visualization, consider reducing to 2 or 3, we often can't really visualize more super effectively!
    - Though NMF can be visualized pretty nicely even with large dictionary size, see Bin's slides
- Many methods have a method-specific way which makes sense
    - PCA: large enough *k* to explain a lot of variance
    - NMF: cross-validation / missing data imputation
- Stability: does some value of *k* yield more stable embeddings under bootstrap, subsampling, a different random seed, or some other perturbation?
- Downstream task: some values of *k* may yield better performance on a downstream task (e.g. regression), or greater stability.

An idea to choose *k* in NMF

- Randomly leave out entries from the data matrix X when performing optimization (e.g. leave out the (i, j) value from the loss function)

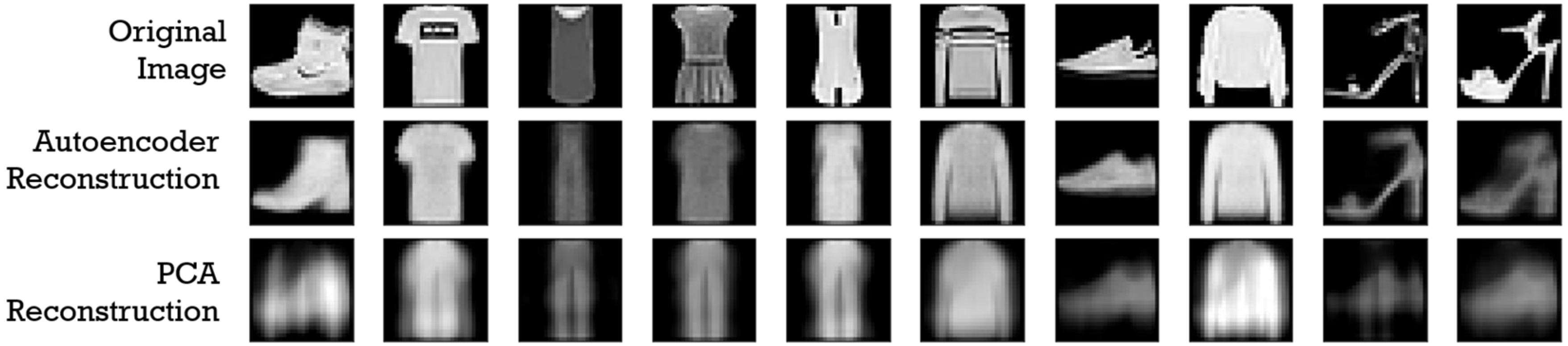
$$\underset{\mathbf{W}\geq 0, \mathbf{H}\geq 0}{\operatorname{argmin}} \sum_{\substack{(i,j) \\ \text{not missing}}} (\mathbf{X}_{ij} - \mathbf{W}_i^\top \mathbf{H}_j)^2$$

- For each potential choice of K:
  - Apply NMF to to get W_M and H_M
  - Impute the missing values of X using the corresponding entries of $W_M H_M$
  - Compute the imputation error (MSE of difference between imputed and observed values)
  - Repeat a bunch of times and compute the mean and SE for this K
- Choose K by taking the minimum or using some other rule

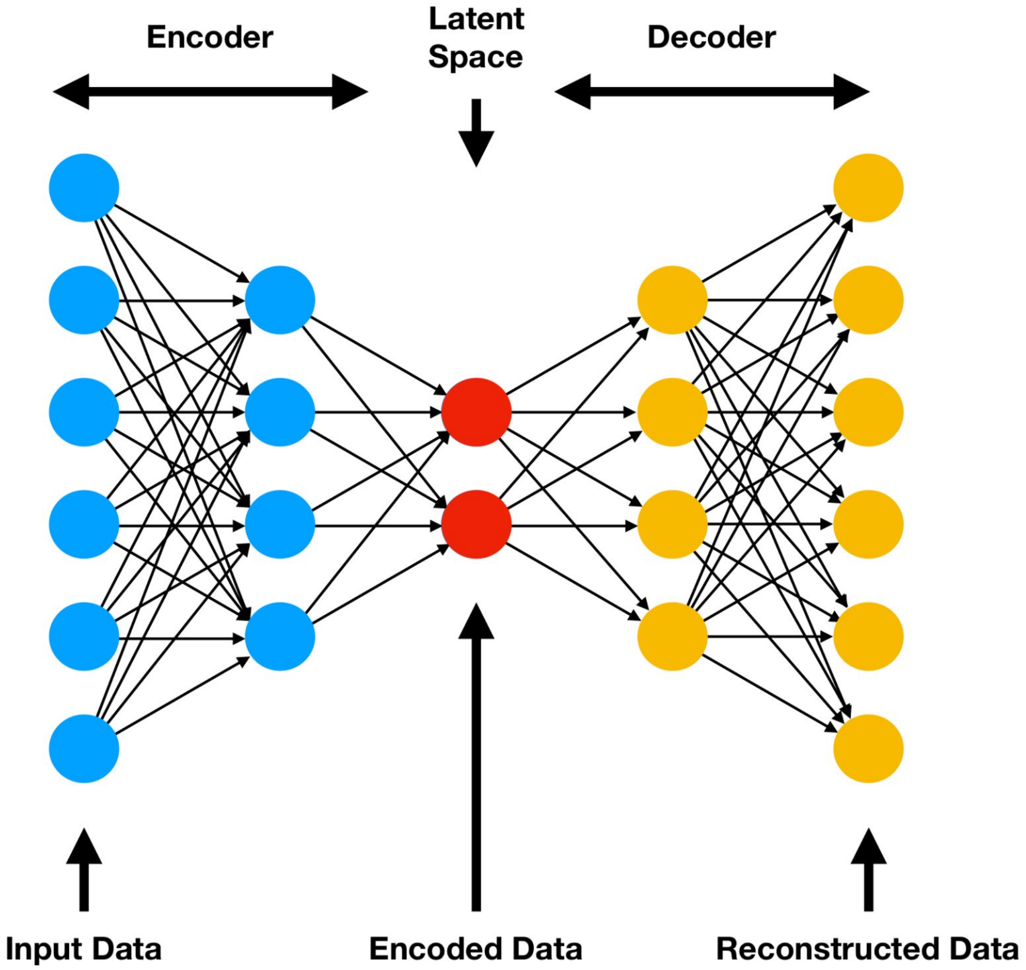
Additional resources

- cdr-cookbook: a document Anthony made which tries to synthesize common clustering and dimension reduction methods using a common notation, pointing out similarities.
- some t-SNE visualizations
- The why and how of nonnegative matrix factorization (note the different definitions of X, W, and H from those used by e.g. sklearn)

Autoencoders



https://commons.wikimedia.org/wiki/File:Reconstruction_autoencoders_vs_PCA.png



https://www.compthree.com/blog/autoencoder/

Dimension reduction in Python

Scikit-Learn (`sklearn`)

scikit-learn

- `scikit-learn` is the main (non-DL) ML package used in Python
- Implementations of all of the most common ML tools
- Be **very careful** with it: it is designed for practitioners, not statisticians/researchers
  - e.g. `sklearn.linear_model.LogisticRegression` applies regularization by default (with arbitrary choice of λ=1)???
- Workflow is:

```
from sklearn.something import some_model
X = some n×p matrix
y =  n-vector or n×k matrix
model = some_model(some_params)
result = model.fit(X, y, other_params) # for something like PCA, don’t need to give y
# Now you can look at the attributes of result to get what you want
```

🚨 Announcements 🚨

- Lab 1 is due next Friday, 02/21.
- Lab 2 will be assigned on the same day.