CSE 5523: Lecture Notes 11
Dimensionality Reduction

Imagine we have the following data for nouns and verbs that precede them:

$$\mathbf{X} = \begin{array}{r} \\ \text{orzo} \\ \text{penne} \\ \text{ziti} \\ \text{pici} \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \text{buy} & \text{cook} & \text{eat} \end{array} \\ \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 3 & 6 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \end{pmatrix} \end{array}$$

This is pretty sparse, e.g. no instances of *orzo* modified by *buy*.

We can generalize over limited data if we blur or 'smooth' it by removing dimensions of variance.

11.1 Center and scale

First we center and scale our data $\mathbf{X} \in \mathbb{R}^{N \times V}$:

$$\mathbf{X}' \stackrel{\text{def}}{=} (\mathbf{X} - \overbrace{\frac{\mathbf{1}\mathbf{1}^\top \mathbf{X}}{N}}^{\text{broadcasted means}}) \qquad \text{center}$$

$$\mathbf{X}^{(0)} \stackrel{\text{def}}{=} \mathbf{X}' \underbrace{(\mathbf{X}'^\top \mathbf{X}' \odot \text{diag}(\mathbf{1}))^{-\frac{1}{2}}}_{\text{diagonal of inverse standard deviations}} \qquad \text{scale by standard deviation}$$

11.2 Best-fit line

Then we find a line $\mathbf{r}_{\mathbf{X}}^{(I)} \in \mathbb{R}^V$ capturing the most variance in centered data $\mathbf{X}$.

Start with random initial line $\mathbf{r}_{\mathbf{X}}^{(0)}$, then iteratively project it through variance $\mathbf{X}^\top\mathbf{X}$ and renormalize:

$$\mathbf{r}_{\mathbf{X}}^{(i)} = \frac{\mathbf{X}^\top \mathbf{X}\, \mathbf{r}_{\mathbf{X}}^{(i-1)}}{||\mathbf{X}^\top \mathbf{X}\, \mathbf{r}_{\mathbf{X}}^{(i-1)}||_2} \tag{1}$$

(Weight all data points by similarity to $\mathbf{r}^{(i-1)}$, then average coordinates, then move to unit circle.)

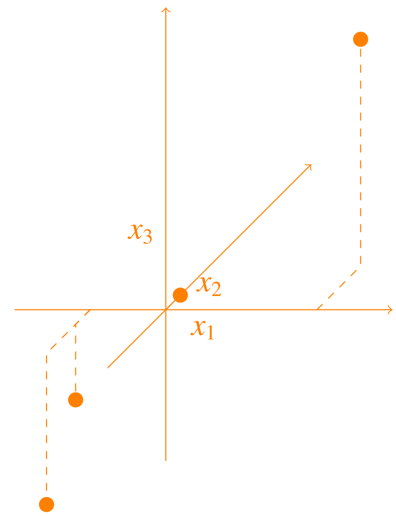
This proceeds until i converges ($i = I$).

For example:

$$\mathbf{X} = \mathbf{X} - \frac{\overbrace{\mathbf{1}^{N\times N}\,\mathbf{X}}^{\text{column means}}}{N} = \begin{bmatrix} -1 & -.5 & -1 \\ 0 & .5 & 0 \\ 2 & 1.5 & 3 \\ -1 & -1.5 & -2 \end{bmatrix} \qquad \text{(centered)}$$

$$\mathbf{r}_{\mathbf{X}}^{(0)} = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix}$$

$$\mathbf{X}^{\top}\mathbf{X}\,\mathbf{r}_{\mathbf{X}}^{(0)} = \begin{bmatrix} -1 & 0 & 2 & -1 \\ -.5 & .5 & 1.5 & -1.5 \\ -1 & 0 & 3 & -2 \end{bmatrix} \begin{bmatrix} -1 & -.5 & -1 \\ 0 & .5 & 0 \\ 2 & 1.5 & 3 \\ -1 & -1.5 & -2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} = \begin{bmatrix} \frac{20}{\sqrt{3}} \\ \frac{18}{\sqrt{3}} \\ \frac{31}{\sqrt{3}} \end{bmatrix}$$

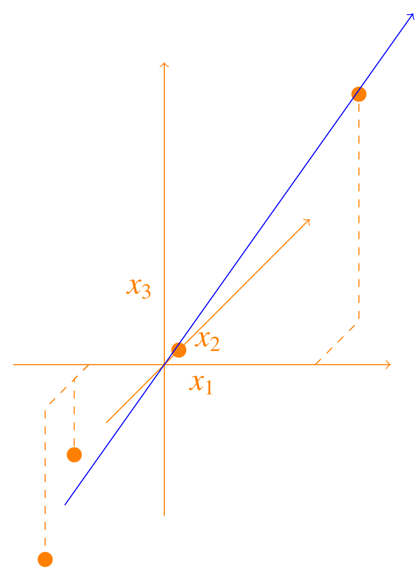


$$\mathbf{r}_{\mathbf{X}}^{(1)} = \begin{bmatrix} 0.48722554 \\ 0.43850298 \\ 0.75519958 \end{bmatrix}$$

$$\mathbf{r}_{\mathbf{X}}^{(2)} = \begin{bmatrix} 0.48765374 \\ 0.43679415 \\ 0.75591316 \end{bmatrix}$$

$$\mathbf{r}_{\mathbf{X}}^{(3)} = \begin{bmatrix} 0.48767114 \\ 0.4367649 \\ 0.75591884 \end{bmatrix}$$

$$\mathbf{r}_{\mathbf{X}}^{(4)} = \begin{bmatrix} 0.48767151 \\ 0.43676433 \\ 0.75591892 \end{bmatrix}$$



11.3 Principal Components Analysis

Next we collapse the space of the data along this line $\mathbf{r}$ of greatest variance.

Done by projecting remaining variance $\mathbf{X}^{(\ell-1)}$ onto $\mathbf{r}$, then back using $\mathbf{r}^{\top}$, and subtracting from $\mathbf{X}^{(\ell-1)}$.

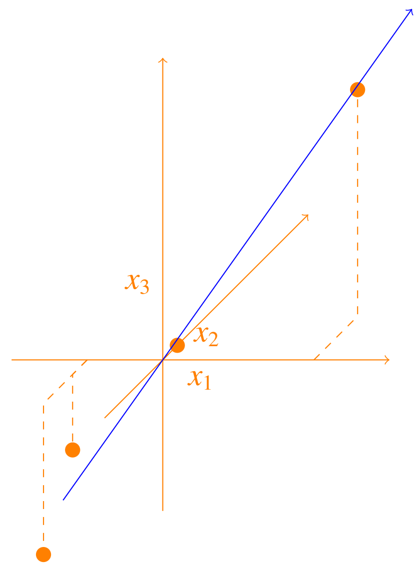
Each time we do this makes a simpler, lower-dimensional space $\mathbf{X}^{(\ell)}$ of the remaining variance:

$$\mathbf{X}^{(\ell)} = \mathbf{X}^{(\ell-1)} - \mathbf{X}^{(\ell-1)}\mathbf{r}^{(l)}_{\mathbf{X}^{(\ell-1)}}\mathbf{r}^{(l)\top}_{\mathbf{X}^{(\ell-1)}} \tag{2}$$

We keep doing this until we have a set of L lines (principal components) that approximate the data.

For example:

$$\mathbf{X} = \begin{array}{r} \\ \text{orzo} \\ \text{penne} \\ \text{ziti} \\ \text{pici} \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \text{buy} & \text{cook} & \text{eat} \end{array} \\ \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 3 & 6 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \end{pmatrix} \end{array}$$

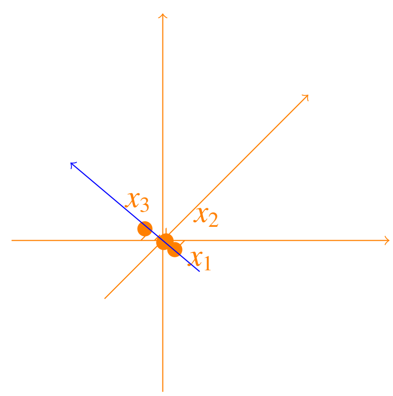


$$\mathbf{X}^{(0)} = \mathbf{X} - \frac{\mathbf{1}^{N\times N}\,\mathbf{X}}{N} = \begin{bmatrix} -1 & -.5 & -1 \\ 0 & .5 & 0 \\ 2 & 1.5 & 3 \\ -1 & -1.5 & -2 \end{bmatrix} \qquad \text{(centered)}$$

$$\mathbf{r}^{(I)}_{\mathbf{X}^{(0)}} = \begin{bmatrix} .49 \\ .44 \\ .76 \end{bmatrix}$$

Now let's add another component:

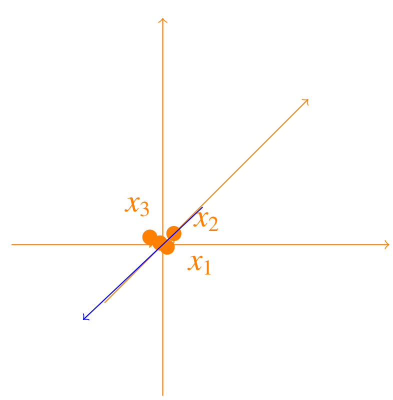
$$\mathbf{X}^{(1)} = \begin{bmatrix} -0.2870376 & 0.13853747 & 0.10513275 \\ -0.10649876 & 0.40461846 & -0.16507921 \\ 0.0989363 & -0.20261489 & 0.05324187 \\ 0.29460005 & -0.34054104 & 0.00670458 \end{bmatrix}$$



$$\mathbf{r}^{(I)}_{\mathbf{X}^{(1)}} = \begin{bmatrix} -.56 \\ .82 \\ -.11 \end{bmatrix}$$

Now we could add another component (but this wouldn't be *reduced* anymore):

$$\mathbf{X}^{(2)} = \begin{bmatrix} -0.14021386 & -0.07691476 & 0.13489797 \\ 0.12318608 & 0.06757412 & -0.11851576 \\ -0.0284893 & -0.01562789 & 0.02740919 \\ 0.04551707 & 0.02496854 & -0.0437914 \end{bmatrix}$$



$$\mathbf{r}^{(I)}_{\mathbf{X}^{(2)}} = \begin{bmatrix} .67 \\ .37 \\ -.64 \end{bmatrix}$$

Now define a 'smoothed' matrix $\hat{\mathbf{X}}^{(0)} \in \mathbb{R}^{N \times V}$ by projecting $\mathbf{X}^{(0)}$ into this reduced space, then back:

$$\hat{\mathbf{X}}^{(0)} = \underbrace{\mathbf{X}^{(0)} \left[\mathbf{r}^{(1)} \cdots \mathbf{r}^{(L)} \right]}_{\text{data points in } L\text{-space}} \begin{bmatrix} \mathbf{r}^{(1)\top} \\ \vdots \\ \mathbf{r}^{(L)\top} \end{bmatrix} \tag{3}$$

Then un-center it to get $\hat{\mathbf{X}}$ – a 'smoothed' version of $\mathbf{X}$:

$$\hat{\mathbf{X}} = \hat{\mathbf{X}}^{(0)} + \frac{\mathbf{1}^{N \times N} \mathbf{X}}{N} \tag{4}$$

Here's what the reconstruction looks like using the first two principal components:

$$\hat{\mathbf{X}} = \begin{bmatrix} -1 & -.5 & -1 \\ 0 & .5 & 0 \\ 2 & 1.5 & 3 \\ -1 & -1.5 & -2 \end{bmatrix} \begin{bmatrix} .49 & -.56 \\ .44 & .82 \\ .76 & -.11 \end{bmatrix} \begin{bmatrix} .49 & .44 & .76 \\ -.56 & .82 & -.11 \end{bmatrix} + \begin{bmatrix} 1 & 1.5 & 3 \\ 1 & 1.5 & 3 \\ 1 & 1.5 & 3 \\ 1 & 1.5 & 3 \end{bmatrix}$$

$$= \begin{array}{r} \\ \text{orzo} \\ \text{penne} \\ \text{ziti} \\ \text{pici} \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \text{buy} & \text{cook} & \text{eat} \end{array} \\ \left(\begin{array}{ccc} 0.13 & 1.07 & 1.85 \\ 0.87 & 1.93 & 3.12 \\ 3.04 & 3.04 & 6.00 \\ -0.05 & -0.04 & 1.02 \\ \vdots & \vdots & \vdots \end{array} \right) \end{array}$$

That solved our zero-count problem for *orzo*!

Not so much for *pici* though (it has negative counts!)... That's a problem with linear regression.

We might fix this by *not* centering first, or by using other techniques, like neural nets (later)!

Reduced dimensionality vectors are also associated with words ('word embeddings').

- Data dimensionality V is very large, e.g. set of co-occurring words at various offset distances.
- Reduced dimensionality L is usually about 100 to 1000.
- Dimensionality reduction uses recurrent neural networks.

11.4 Sample PCA code

Sample PCA code in pandas:

```
import sys
import numpy as np
import pandas as pd
```

```
X = pd.read_csv( sys.argv[1], index_col=0 )                      ## read data
N = len(X)
V = len(X.columns)
L = 2
                                                                 ## center and z-scale
Xc = X - ( pd.DataFrame( np.ones((N,N)), X.index, X.index ) @ X / N )
Xr = Xz = Xc @ pd.DataFrame( np.linalg.inv( Xc.T @ Xc * np.eye(V) ), X.columns, X.columns )

R = pd.DataFrame( np.random.rand(V,L), X.columns, range(L) )   ## random initial vectors
for l in range(L):                                               ## each principal component
  for i in range(10):                                            ## each epoch of best-fit
    R[l] = Xr.T @ Xr @ R[[l]] / np.linalg.norm( Xr.T @ Xr @ R[[l]] )   ## fit to variance
  Xr = Xr - Xr @ R[[l]] @ R[[l]].T                                      ## remove dimension

Xze = Xz @ R @ R.T                                               ## project to reduced space

Xce = Xze @ ( Xc.T @ Xc * np.eye(V) )                            ## un-z-scale and un-center
Xe = Xce + pd.DataFrame( np.ones((N,N)), X.index, X.index ) @ X / N

print( Xe )
```

Sample input data file ‘X.csv’:

```
,buy,cook,eat
orzo,0,1,2
penne,1,2,3
ziti,3,3,6
pici,0,0,1
```

Output smoothed counts:

```
            buy      cook       eat
orzo   0.047038  1.018739  1.748499
penne  0.955731  1.982364  3.236695
ziti   3.013222  3.005267  5.929305
pici  -0.015991 -0.006371  1.085501
```