Feature Projection

Week	THURS. Week 2
Assignment Due	
Assignment Done	☐
Due Date	
Notes Done	☑

Presentation

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/64cd0819-62b2-4762-a9ed-5ed5c294e222/L2-FeatureProjection.pdf

Class Notes

Feature Construction

Feature selection was one of two approaches to reducing dimensionality of our data, which included selecting features by a(n):

- Exhaustive approach
- Greedy approach
- Selective feature approach

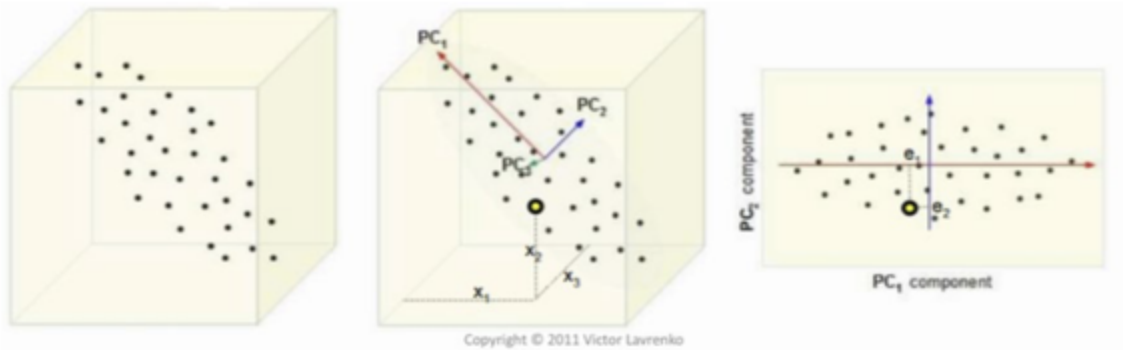
Another approach commonly done is **feature construction**, where we create a whole new set of features from existing features where most of the relevant information is encoded in a few of the new features.

- From this, we can use the most important features that encodes most of the important info, discarding extra unnecessary info.
- We'll look at the idea of **feature projection.**

Feature Projection by PCA

We'll look at feature projection by **principal component analysis (PCA)**, which defines a set of principal components (basis):

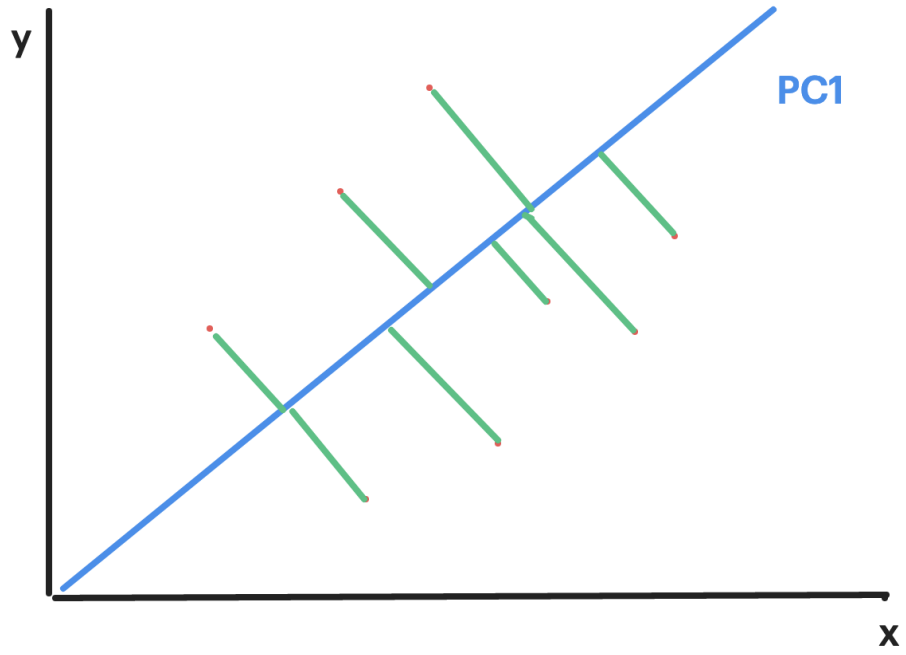
- Principal component 1 is the direction of the greatest variability in the data..
- Principal component 2 is perpendicular to the 1st, greatest variability of what's left
  - Continues until D, the original dimensionality.
- PCA looks are structure of our data, trying to find a coordinate system that better aligns with the data.



  - In the 3D case, the data looks elongated.
    - Instead of having original 3D coordinate system, we can consider how we can capture essence of data if we realign the coordinate system by projecting the data onto a different coordinate system.
      - Essentially, we're trying to go from 3D space into 2D space.
    - We want to look at **covariance (variation in data)** and look at the principal component that goes along the data the best in that when we project our data onto the axis, we end up maximizing the spread of the data.
      - If we chose the PC_2 above an axis as a principal component and projected the data onto it, the data would be condensed, which is NOT

what we want.

        - We may also get a wider spread of the data on another axis.
    - PCA effectively tries to find an axis that we can project on is are independent of one another, so the axes need to be perpendicular.



PCA Derivation

We want to be able to take our data matrix and take an axis $\mathbf{w}$ that is a $D \times 1$ dimensional axis.

- Projection of that point on our axis gives us new points: $Z = X\mathbf{w}$
    - Say we have a bunch of data in space and have an axis.
        - This calculation is trying to figure out where the data is on the space.
- For our purposes, we want to **maximize** the variance of Z.
    - It's better for the data to be spread out.
        - We want to create an **objective function**, defining something we're trying to minimize or maximize.

        - In this case, we're interested in the projected data, Z, to have a large variance, so our objective function may look as follows:
            - $J = Var(Z) = Var(X\mathbf{w})$
        - Objective: Find $\mathbf{w}$, principal component, s.t. when we project our data on it, we get ***maximum variance***
- If our X data is 0-meaned, then we calculate the covariance of the data as follows:

$$J = \text{Var}(X\mathbf{w}) = \frac{(X\mathbf{w})^T(X\mathbf{w})}{N-1} = \frac{\mathbf{w}^T X^T X \mathbf{w}}{N-1} = \mathbf{w}^T \Sigma \mathbf{w}$$

Where Σ is the covariance matrix of X

Covariance of the original data

    - Note: We know from linear algebra that if the data's columns are zero-meaned that we can compute the covariance matrix easily through the following:
        - $\Sigma = \frac{X^T X}{N-1}$

PCA via Singular Value Decomposition

A trivial solution to this maximization problem would be to set $\mathbf{w}$ to be a vector of infinities!

- However, this isn't useful for us in our specific problem.
- Not to mention, conceptually $\mathbf{w}$ is supposed to describe an axis of a coordinate system:
    - So, its length, $|\mathbf{w}| = \sqrt{\mathbf{w}^T \mathbf{w}}$ should be one!!!!! IMPORTANT!!!!
- Fortunately, there is a special problem in linear algebra where we're trying to maximize an equation in the form $\mathbf{w}^T \mathbf{w}$, where $\mathbf{w}$ is an unknown vector and Σ is a known vector, whiel requiring that $\mathbf{w}$ is unit length.
    - This is called **singular value decomposition**.

Singular value decomposition attempts to *decompose* Σ into sets of *unit-length vectors* called ***eigenvectors*** and associated ***eigenvalues*** s.t.:

- $\Sigma = USV^T$
  - Where:
    - U is a matrix of the *left eigenvectors* (assembled as columns)
    - V are the *right eigenvectors*
    - S is a matrix with the *eigenvalues* on its diagonal
  - NOTE: If Σ is symmetric, then $U = V$.
- Since we're trying to *MAXIMIZE* $\mathbf{w}^T\Sigma\mathbf{w}$, the *best eigenvector* is one associated with the **largest** eigenvalue.

PCA via Eigen-Decomposition

Another way to formulate the problem is to explicitly add a penalty for the (squared) magnitude of $\mathbf{w}$ becoming larger than 1:

$$J = \boldsymbol{w}^T\Sigma\boldsymbol{w} - \lambda(\boldsymbol{w}^T\boldsymbol{w} - 1)$$

Tells us how much we care about its length of being 1

- The λ term tells us about how much we care about its length being 1.

Math to take the derivate of this objective function:

Taking the derivate of this we get:

$$\frac{dJ}{d\boldsymbol{w}} = (\Sigma^T + \Sigma)\boldsymbol{w} - 2\lambda\boldsymbol{w}$$

Since Σ in this application is the covariance matrix of our data, Σ is symmetric, and therefore $\Sigma^T = \Sigma$.

Thus:

$$\frac{dJ}{d\boldsymbol{w}} = 2\Sigma\boldsymbol{w} - 2\lambda\boldsymbol{w}$$

To find the extrema, we can set this derivative term equal to 0 and try solving for $\mathbf{w}$:

- $(\Sigma - \lambda)\mathbf{w} = 0$

    - To solve this equation, a trivial solution would be $\mathbf{w} = [0, 0, .., 0]^T$ → again, NOT useful

- Fortunately, we can rewrite this equation as $\Sigma\mathbf{w} = \lambda\mathbf{w}$, we have a special linear algebra problem called **eigen-decomposition**!

Eigendecomposition is a problem where if we have an equation where we have a ***known matrix*** (Σ) multiplied by an ***unknown vector*** ($\mathbf{w}$) being EQUAL to an ***unknown scalar (***λ) multiplied by the same unknown vector ($\mathbf{w}$), solutions to the equation that require the combination of a scalar (λ) and a vector ($\mathbf{w}$), are known as ***eigenvalue-eigenvector*** combinations.

- The *decomposition* of the known matrix, Σ, gives use plausible eigenvalue, eigenvector combinations to make this equation true.

PCA in Our Course

In this course, we'll just use a linear algebra package to decompose a matrix into eigenvectors and eigenvalues.

- In MATLAB (`numpy` is similar), decomposing our variance matrix via **eigen-decomposition** is done as:
    - $[W, \lambda] = eig(\Sigma)$
        - Where:
            - W is a matrix s.t. its column are the eigenvectors.
            - λ is a **diagonal** matrix (all zeroes except on the diagonal), s.t. the corresponding eigenvalues are on the diagonal.
- Doing this via **singlar value decomposition** in MATLAB (`numpy` is similar) is:
    - $[W, \lambda, V] = svd(\Sigma)$

Choosing Eigenvectors

In our formulation above, in considering **eigendecomposition**, the closer that $|\mathbf{w}| = 1$, the larger λ can be without much (if any) penalty being incurred.

- Therefore, for this objective function, a larger λ indicates a "better" solution.

In thinking about how many *eigenvectors* to use, we must consider:

- Consider: Do we want only one eigenvector?
  - Using *one eigenvector* will give us only ONE axis/dimension.
- If we want more axis/dimensions, then we grab the k most useful eigenvectors.
  - Consider: How do we choose k?
    - This can be determined by the user/problem at hand.
    - Or, we can make sure to include up to some percent of the total eigenvalues:
      - Find k s.t.:

$$\frac{\sum_{i=1}^{k} |\lambda_i|}{\sum_{i=1}^{D} |\lambda_i|} \geq \alpha$$

        - The typical values for the threshold α are 0.9 or 0.95.

Using PCA for Dimensionality Reduction

Now that we have k principal components, we can grab the associated eigenvectors that are in a $D \times 1$ vector.

- Concatenated, they form a $D \times k$ **projection matrix** $W = [e_1, .., e_k]$
- Now, we can project our D-dimensional data into k-dimensions:
  - $Z = XW$

Example with PCA

$$X = \begin{bmatrix} 7 & 1 & 2 \\ 2 & 4 & 0 \\ 2 & 3 & -8 \\ 3 & 6 & 0 \\ 4 & 4 & 0 \\ 9 & 4 & 1 \\ 6 & 8 & -2 \\ 9 & 5 & 1 \\ 8 & 7 & 11 \\ 10 & 8 & -5 \end{bmatrix} \qquad X' = \begin{bmatrix} 1 & -4 & 2 \\ -4 & -1 & 0 \\ -4 & -2 & -8 \\ -3 & 1 & 0 \\ -2 & -1 & 0 \\ 3 & -1 & 1 \\ 0 & 3 & -2 \\ 3 & 0 & 1 \\ 2 & 2 & 11 \\ 4 & 3 & -5 \end{bmatrix}$$

1. Zero mean data first so we can calculate the **covariance matrix**, $\frac{X^T X}{N-1}$
   a. N = number of observations
2. In computing our covariance matrix, we can:
   a. Expect a 3×3 matrix and should be symmetric due to the comparisons made between features that are symmetrical (e.g.: comparison between feature 1 and feature 2 is the same as the comparison made between feature 2 and feature 1)

$$\Sigma = \frac{X'^T X'}{N-1} = \begin{bmatrix} 9.33 & 2.22 & 4.67 \\ 2.22 & 5.11 & 0.89 \\ 4.67 & 0.89 & 24.4 \end{bmatrix}$$

   1. The number 4.67 on the bottom left (row 3, column 1) tells us what is the correlation between feature 1 and feature 3.
   2. The values that compare the same row and column, the value in the covariance matrix is simply the variance in that data.
3. Get the eigenvalues/vectors of the covariance matrix:

Eigenvalues = [4.09, 8.92, 25.87]

Eigenvectors

$$\begin{bmatrix} 0.44 \\ -0.90 \\ -0.06 \end{bmatrix}, \begin{bmatrix} 0.86 \\ 0.43 \\ -0.28 \end{bmatrix}, \begin{bmatrix} 0.28 \\ 0.07 \\ 0.96 \end{bmatrix}$$

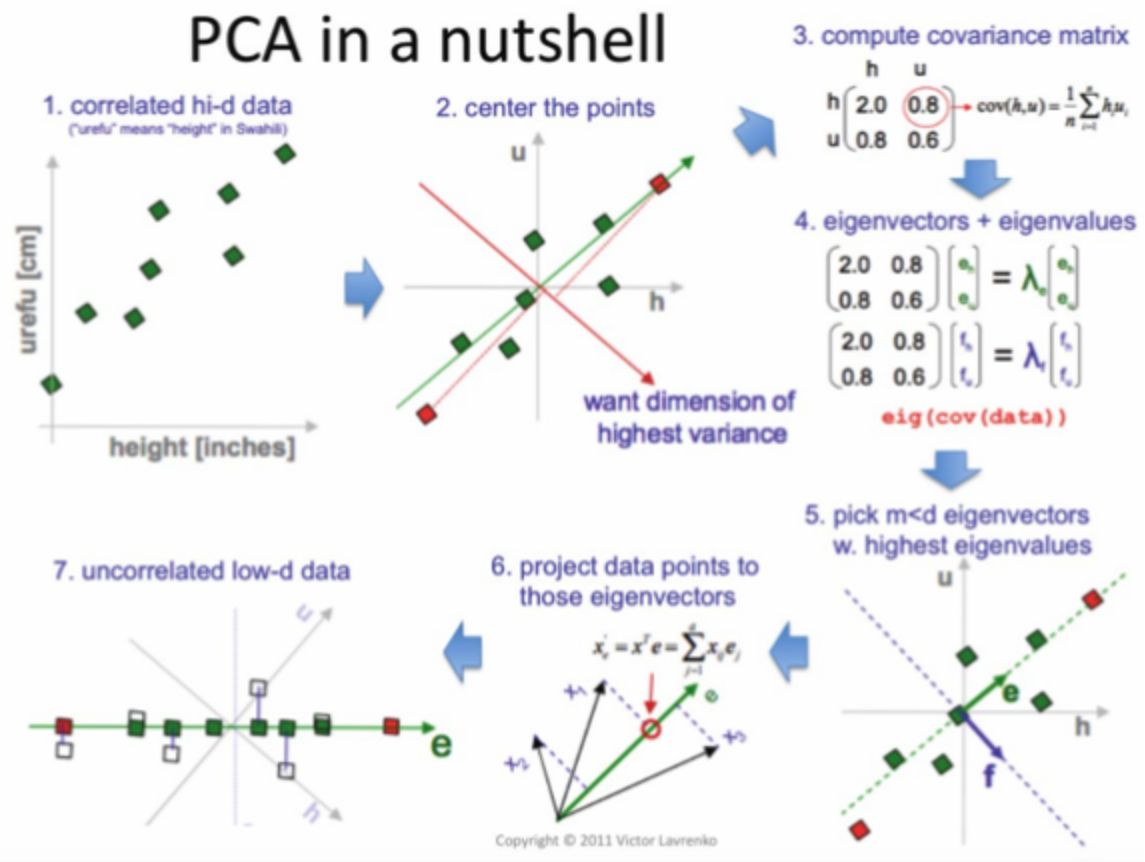
1. Since we care about the most relevant eigenvector, we look for the most relevant eigenvalue of the covariance matrix.

4. Upon having these relevant eigenvalues, we project our points onto the single best vector (i.e., the one with the highest eigenvalue). This will give us a single value.
   a. NOTE: We'll do this projection on the zero-centered data all at once. We'll multiply the $\mathbf{w}$ matrix by one row for demonstration purposes.

$$Z_{1,1} = X_1' W = [1,-4, 2] \begin{bmatrix} 0.28 \\ 0.07 \\ 0.96 \end{bmatrix} = 1.91$$

1. X_1' is the first row of the normalized X' matrix.

PCA in a Nutshell



Reconstruction

PCA for feature reduction can be thought of as a *compression* or *encoding* process.

- We can think of PCA as a mechanism for encoding/lossy compression.
- We can actually *decode* our reduced features to return to our ORIGINAL number of features.
    - The reconstruction, however, will NOT be exact, but close enough for what we may need.

We can start off with the lossless case:

- It helps to think of eigenvectors as axes of coordinate systems:
    - If we took the first (zero-meaned) observation, $\mathbf{x} = [1, 4, 2]$ and projected it using all three eigenvectors, then its location in the new coordinate system

(defined by the eigenvectors) would be $\mathbf{x} = [1, -4, 2] \rightarrow \mathbf{z} = [1.91, -1.44, 3.91]$

- How can we get back to the original coordinate system?
  - We can move 1.91 units down the first eigenvector, down -1.44 the second one, and 3.91 down the third:

$$\hat{\boldsymbol{x}}^T = 1.91 \begin{bmatrix} 0.28 \\ 0.07 \\ 0.96 \end{bmatrix} - 1.44 \begin{bmatrix} 0.86 \\ 0.43 \\ -0.28 \end{bmatrix} + 3.91 \begin{bmatrix} 0.44 \\ -0.90 \\ -0.06 \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \\ 2 \end{bmatrix}$$

First eigenvector Second eigenvector Third eigenvector

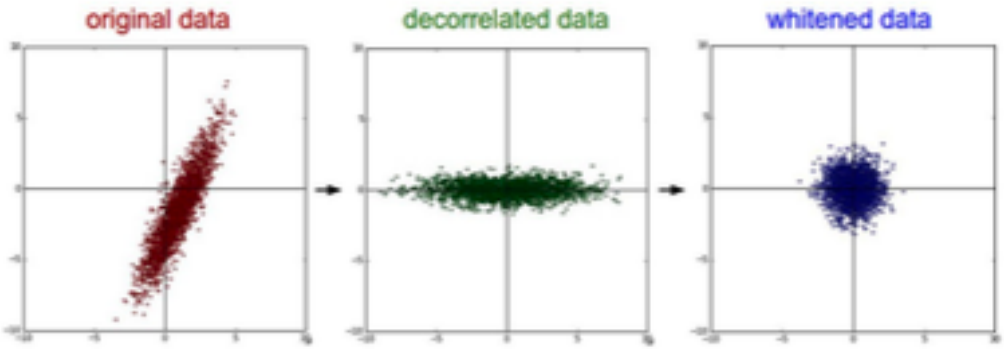
- In linear algebra, we can write this as $\hat{x} = \mathbf{z}W^T$ or for ALL observations $\hat{X} = ZW^T$.
- If we opt to use less eigenvectors in our projection, then we *will* have lossy reconstruction.
  - Good news is that most of the "movement" is captured in the first few most relevant eigenvectors!

PCA Whitening

Recall that the goal of PCA is to project data onto the direction of maximum variance:

- Then onto the next direction of maximum variance, perpendicular to the first, etc..
  - This can result in larger variance in the new first feature, and decreasing variance thereafter.
- As a result of this, PCA can end up resulting in an elongated visualization.
  - Often, instead of wanting the data to be *spread out* over the axis, we want the data to be less spread out.
    - The process to transform the data to be less spread out is called **whitening**.
    - To reduce this effect, we can divide each eigenvector by the square root of its eigenvalue:

- $e_i = \frac{e_i}{\sqrt{|\lambda_i|}}$



Rationale for PCA in Machine Learning

There is a higher chance to overfit out models if we have higher dimensions in our data, which is the rationale for learning how to reduce dimensionality intelligently.

Resources

Principal Component Analysis (PCA) | Note of Thi

👉 Note: UMAP & t-SNE. What? Sometimes we need to "compress" our data to speed up algorithms or to visualize data. One way is...

🔥 https://dinhanhthi.com/principal-component-analysis/