Lecture 12: Exploring Data Through Preprocessing and Unsupervised ML
Part 2

Feb 22, 2023
CIS 4190/5190
Spring 2023
Recap: Clustering

What natural groupings exist in this data?
Recap: K-Means Clustering

K-Means \( (K, X) \)
- Randomly choose \( K \) cluster center locations (centroids)
- Loop until convergence, do:
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster
Recap: The Choice of Distance Function

- Clustering techniques all usually accept a matrix of pairwise distances between data points as input.

- The choice of distance function affects the clustering outcomes. This boils down to: different distance functions might consider different point pairs more similar.

\[
L_\infty(a, b) = 5 \quad L_\infty(c, d) = 4 \\
L_2(a, b) = (5^2 + \varepsilon^2)^{\frac{1}{2}} = 5 + \varepsilon \quad L_2(c, d) = (4^2 + 4^2)^{\frac{1}{2}} = 4\sqrt{2} = 5.66
\]
• One common choice is to tie the distance measure itself to the structure of the data.

**Mahalanobis Distance:**

\[ d(x, y) = \sqrt{(x - y)^T \Sigma^{-1}(x - y)} \]

- \( \mu = \frac{1}{m} \sum_{i=1}^{m} x_i \) is the mean vector, which represents the average of the data.
- \( \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x - \mu)(x - \mu)^T \) is the covariance matrix of the data.

• When \( \Sigma \) is identity, this is the same as Euclidean distance.

• In 1D, this measures how many standard deviations away two points are.

• The Mahalanobis distance generalizes this to higher dimensions ...
Covariance Matrix Of Data

For zero-centered data,

\[
\text{Covariance} = \Sigma = \mathbb{E}[xx^T] = \mathbb{E}\begin{bmatrix}
    x_{i1}x_{i1} & \cdots & x_{i1}x_{iD} \\
    \vdots & \ddots & \vdots \\
    x_{iD}x_{i1} & \cdots & x_{iD}x_{iD}
\end{bmatrix}
\]

\[
\sigma(x, y) = \mathbb{E}[(x - \mathbb{E}(x))(y - \mathbb{E}(y))]
\]

\[
\Sigma = \begin{bmatrix}
    \sigma(x, x) & \sigma(x, y) \\
    \sigma(y, x) & \sigma(y, y)
\end{bmatrix}
\]

Covariance Matrix in Terms of Data Matrix $X$

\[
\text{Covariance} = \Sigma = \mathbb{E}[x_i x_i^T] = \mathbb{E} \begin{bmatrix} x_{i1} x_{i1} & \cdots & x_{i1} x_{iD} \\ \vdots & \ddots & \vdots \\ x_{iD} x_{i1} & \cdots & x_{iD} x_{iD} \end{bmatrix} = \frac{1}{N} \sum_{i} x_i x_i^T
\]

\[
X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \quad X^T = [x_1 \ x_2 \ \cdots \ x_n]
\]

\[
\frac{1}{N} X^T X = \frac{1}{N} (x_1 x_1^T + x_2 x_2^T + \cdots + x_N x_N^T)
\]

Thus, the data covariance matrix is typically computed as $\frac{1}{N} X^T X$
Covariance Matrix Is Related to Dataset “Shape”

Mahalanobis Distance: \( d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)} \)

“Distances matter more when they are along directions in which the data varies less.”

Covariance Matrix Of Data

“Distances matter more when they are along directions in which the data varies less.”

Mahalanobis Distance: \( d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)} \)

pink and green distance are equal in the Euclidean distance sense.
pink distance > green distance in the Mahalanobis distance sense.

Summary of Clustering

• Critical to understanding the structure of our data

• Often useful for creating high-level features useful for supervised learning

• We saw one approach in detail: K-Means
Optional readings: Clustering

- Hastie and Tibshirani, Elements of Statistical Learning, Ch 14.5.1 and 14.5.2. https://hastie.su.domains/ElemStatLearn/
Dimensionality Reduction
Dimensionality Reduction

Map samples $x_i \in \mathbb{R}^D$ to $f(x_i) \in \mathbb{R}^{D' \ll D}$

Can think of this as generalizing clustering, $f(x_i) \in \mathbb{N}^1 \rightarrow f(x_i) \in \mathbb{R}^{D' \ll D}$

• Rather than groupings, we want to recover “low-dimensional structure”

Also a generalization of “feature selection”.

• Dimensionality-reduced $f(x_i)$ need not just have a subset of the elements of the original vector $x_i$. 
What Is The “Structure” Of A Dataset?
The Uses of Dimensionality Reduction

- **Feature Learning**: For preprocessing inputs to an ML algorithm, since lower-dimensional features permit smaller models and fewer data samples.

- **Compression (for storage)**: e.g. JPEG standard for images is now adopting unsupervised ML approaches [https://jpeg.org/items/20190327_press.html](https://jpeg.org/items/20190327_press.html)

- **Visualization**: Exploring a dataset, or an ML model’s outputs
Consider: Visualizing High-Dimensional Data


<table>
<thead>
<tr>
<th>LotFrontage</th>
<th>LotArea</th>
<th>Street</th>
<th>LotShape</th>
<th>Utilities</th>
<th>LandSlope</th>
<th>OverallQual</th>
<th>OverallCond</th>
<th>YearBuilt</th>
<th>YearRemodAdd</th>
<th>MasVnrArea</th>
<th>ExtarQual</th>
<th>ExterCond</th>
<th>BsmtQual</th>
<th>BsmtExposure</th>
<th>BsmtFinType1</th>
<th>BsmtFinSF1</th>
<th>BsmtFinType2</th>
<th>BldgType</th>
<th>BsmtFullBath</th>
<th>FullBath</th>
<th>HalfBath</th>
<th>Bedroom</th>
<th>Kitchen</th>
<th>Bath</th>
<th>functional</th>
<th>Fireplaces</th>
<th>GarageArea</th>
<th>GarageType</th>
<th>SaleCondition</th>
<th>SalePrice</th>
<th>SaleCondition</th>
<th>Abnormal</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.0</td>
<td>8450</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2003</td>
<td>2003</td>
<td>196.0</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>706</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>80.0</td>
<td>9600</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>1976</td>
<td>1976</td>
<td>0.0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>978</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>68.0</td>
<td>11250</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2001</td>
<td>2002</td>
<td>162.0</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>486</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>60.0</td>
<td>9550</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>1915</td>
<td>1970</td>
<td>0.0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>216</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>84.0</td>
<td>14260</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>2000</td>
<td>2000</td>
<td>350.0</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>655</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>85.0</td>
<td>14115</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1993</td>
<td>1995</td>
<td>0.0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>732</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>75.0</td>
<td>10084</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>2004</td>
<td>2005</td>
<td>186.0</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>1369</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.0</td>
<td>10382</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>1973</td>
<td>1973</td>
<td>240.0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>859</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

"To deal with hyper-planes in a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it."

- Geoff Hinton

---

227 features

Data Visualization

Is there a representation better than the raw features?

Maybe it isn’t necessary to visualize all 227 dimensions

Idea: find a lower-dimensional subspace that retains most of the information about the original data

There are many methods; our focus will be on Principal Components Analysis
Principal Components Analysis
Dimensionality Reduction Through Orthogonal Projections?

- We often view 3D objects in 2D by “projecting them” onto a plane. Drop perpendicular lines from every point on the object to the plane.

- “Good projections” are views that preserve information about the shape of the data.

- PCA does something similar to every instance in a dataset. Finds good “views” of the dataset.

Fig: [http://www.grad.hr/geomteh3d/Monge/06projekcije/projekcije_eng.html](http://www.grad.hr/geomteh3d/Monge/06projekcije/projekcije_eng.html)
Orthogonal Projection Example: from 2D to 1D

- Let’s project \( x \in \mathbb{R}^2 \) down to a new vector \( v \in \mathbb{R}^1 \) (i.e., a scalar), by orthogonally projecting onto the direction represented by the unit vector \( v \)

\[
y = (x^T v)v
\]
Orthogonal Projection Of An Entire Dataset?

• Every point in the set is projected
• E.g., projecting a 3D dataset in XYZ (see figure, left) onto:
  ▪ the XY plane (top), or
  ▪ the YZ plane (bottom)
• Which of these “views” is better in terms of preserving info about the structure of the data?
• In general, projections need not be axis-aligned. How to find good structure-preserving views?
  ▪ Solution: PCA!

Fig: https://www.geeksforgeeks.org/dimensionality-reduction/
Orthogonal Projection Of An Entire Dataset?

Thus, each choice of view can be parameterized by the basis vectors. So, finding good views = finding good basis vectors.
We are looking for a new coordinate system $\mathbf{v}_1, \ldots, \mathbf{v}_D$, to approximate all $\mathbf{x}_i$:

$$x_i \approx (x_i \cdot \mathbf{v}_1) \mathbf{v}_1 + (x_i \cdot \mathbf{v}_2) \mathbf{v}_2 + \cdots + (x_i \cdot \mathbf{v}_D) \mathbf{v}_D$$

where the new axes $\mathbf{v}_d$’s are all $D$-dimensional unit norm, and $D' \ll D$
Terminology

We are looking for a new coordinate system $\mathbf{v}_1, \ldots, \mathbf{v}_{D'}$ to approximate all $x_i$:

$$x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{iD} \end{bmatrix} \approx (x_i \cdot \mathbf{v}_1)\mathbf{v}_1 + (x_i \cdot \mathbf{v}_2)\mathbf{v}_2 + \cdots + (x_i \cdot \mathbf{v}_{D'})\mathbf{v}_{D'}$$

where the new axes $\mathbf{v}_d$’s are all $D$-dimensional unit norm, and $D' \ll D$

- The axis unit vectors $\mathbf{v}_d$ of the projection are also called “basis” vectors

- The final $D'$- dimensional vector representation is simply the vector of projections

$$\begin{bmatrix} (x_i \cdot \mathbf{v}_1) \\ \vdots \\ (x_i \cdot \mathbf{v}_{D'}) \end{bmatrix}$$
Simplest Case: Reduce to $D' = 1$ dimension

We are looking for a new coordinate system $v_1, ..., v_{D'}$ to approximate all $x_i$:

$$x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{iD} \end{bmatrix} \approx (x_i \cdot v_1)v_1 + (x_i \cdot v_2)v_2 + \cdots + (x_i \cdot v_{D'})v_{D'}$$

where the new axes $v_d$’s are all $D$-dimensional unit norm, and $D' \ll D$

Simplest case: $D' = 1$?

We want to find unit $v_1$ such that:

$$(x_i \cdot v_1)v_1$$ best approximates $x_i$
The Meaning Of “Approximating” The Data

PCA looks for the projection that:
- minimizes mean squared distance between data point and projections (sum of squared blue lines)
- maximizes variance of projected data (roughly, length of purple line)

Here, $D = 2$  
$D' = 1$

Based on slide by Barnabás Póczos, UAlberta
Objective Function: Maximizing Variance

Find unit vector $\mathbf{v}_1$ (with $\|\mathbf{v}_1\|_2 = 1$), to optimize:

$$\min \frac{1}{N} \sum_{i} \| (\mathbf{x}_i \cdot \mathbf{v}_1) \mathbf{v}_1 - \mathbf{x}_i \|^2$$

Can show, exactly equal to:

$$\max_{\|\mathbf{v}_1\|_2 = 1} \text{variance}(\mathbf{x}_i \cdot \mathbf{v}_1)$$

Intuitively, if the variance of the projection on $\mathbf{v}_1$ was low, then $\mathbf{v}_1$ would not be very informative about samples $\mathbf{x}_i$.

Conversely, directions with high variance projections preserve the most information.

So, how to find this direction of maximum variance?

(Fig: stats.stackexchange)
Covariance Matrix To The Rescue Again

• Recall:

\[
\text{Covariance Matrix Of Data}
\]

For zero-centered data, the covariance matrix is given by:

\[
\Sigma = \text{Covariance} = \mathbb{E} [x_i x_j^T] = \mathbb{E} \left[ \begin{array}{cccc} x_{11} x_{11} & \cdots & x_{11} x_{ID} \\ \vdots & \ddots & \vdots \\ x_{ID} x_{11} & \cdots & x_{ID} x_{ID} \end{array} \right]
\]

\[
\sigma(x, y) = \mathbb{E}[(x - \mathbb{E}(x))(y - \mathbb{E}(y))]
\]

\[
\Sigma = \begin{bmatrix} \sigma(x, x) & \sigma(x, y) \\ \sigma(x, y) & \sigma(y, y) \end{bmatrix}
\]

\[
\text{Covariance Matrix Is Related to Dataset “Shape”}
\]

“Distances matter more when they are along directions in which the data varies less.”

Mahalanobis Distance: \(d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}\)
Covariance Matrix Represents a Linear Transformation

\[ \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

Linear transformation

Data \( D \) “white data”

\[ M \]

Data \( D' \) “transformed data”

\[ \Sigma' = \begin{bmatrix} 4.25 & 3.10 \\ 3.10 & 4.29 \end{bmatrix} = MM^T \]

Arrows are eigenvectors of \( \Sigma \), size represents eigenvalues

Refresher on Eigenvectors & Singular vectors
Eigendecomposition

A square matrix $A_{D \times D}$ can be decomposed as:

$$A = U \Lambda U^{-1}$$

$\Lambda$ is a DxD diagonal matrix of “eigenvalues” $\text{diag}(\lambda_1, ..., \lambda_D)$ usually sorted in descending order. Hence, “first eigenvalue” means “largest eigenvalue”

$U$ is a DxD matrix $[u_1, u_2, ..., u_D]$, whose columns are called “eigenvectors”. We usually assume these are normalized to be unit length, i.e., unit eigenvectors.

“First eigenvector” = “largest eigenvector” = “eigenvector with largest eigenvalue”
Eigenvectors: geometric intuition

The eigenvectors \( \mathbf{u}_i \) of a matrix \( A \) are vectors that remain invariant under the linear transformation represented by \( A \) i.e. \( \mathbf{x} \to A\mathbf{x} \)

\[
A\mathbf{u}_i = \lambda_i \mathbf{u}_i
\]

\( \lambda_i \) is the eigenvalue corresponding to \( \mathbf{u}_i \).
Singular vectors: geometric intuition

**Eigenvectors of M**

$$M = \begin{bmatrix} 1 & 1/3 \\ 4/3 & 1 \end{bmatrix}$$

Vectors that remain unchanged after the transformation

https://mathformachines.com/posts/eigenvalues-and-singular-values/
Singular value decomposition (SVD)

Any matrix $A$ can be decomposed as:

$$ A = U \hat{\Lambda} V^T $$

$\hat{\Lambda}$ is a DxD diagonal matrix of “singular values” $\text{diag}(\hat{\lambda}_1, ..., \hat{\lambda}_D)$ usually sorted in descending order. Hence, “first singular value” means “largest” etc.

$U, V$ are DxD orthogonal matrices $[u_1, u_2, ..., u_D]$ and $[v_1, v_2, ..., v_D]$, whose columns are called “left singular vectors” and “right singular vectors”.

Orthogonal $\Rightarrow U^T U = V^T V = I$
Singular vectors: geometric intuition

\[ M = \begin{bmatrix} 1 & 1/3 \\ 4/3 & 1 \end{bmatrix} \]

**Eigenvectors of M**

Vectors that remain unchanged after the transformation

**Singular vectors of M**

Orthogonal set of vectors that remain orthogonal after the transformation

https://mathformachines.com/posts/eigenvalues-and-singular-values/
Note: Left Singular Vectors of $M = \text{Eigenvectors of } MM^T$

- Suppose the SVD of $M = U\hat{\Lambda}V^T$
- Then $MM^T = U\hat{\Lambda}V^T V\hat{\Lambda}U^T = U\hat{\Lambda}^2 U^T$ = eigendecomposition of $MM^T$
- In other words,
  - Eigenvectors $U$ of $\Sigma = MM^T$ are the same as left singular vectors of $M$
    - Also implies that they are orthogonal!
  - Eigenvalues $\hat{\Lambda}^2$ of $\Sigma = MM^T$ are the squares of the singular values of $M$

So, remember: eigenvectors of covariance matrix = left singular vectors of the corresponding linear transformation
Back to PCA
Objective Function: Maximizing Variance

Find unit vector $v_1$ (with $\|v_1\|_2 = 1$), to optimize:

$$\min_{\|v_1\|_2=1} \frac{1}{N} \sum_i \| (x_i \cdot v_1) v_1 - x_i \|_2^2$$

Project error

Can show, exactly equal to:

$$\max_{\|v_1\|_2=1} \text{variance}(x_i \cdot v_1)$$

Intuitively, if the variance of the projection on $v_1$ was low, then $v_1$ would not be very informative about samples $x_i$.

Conversely, directions with high variance projections preserve the most information.

So, how to find this direction of maximum variance?

Covariance Matrix Represents a Linear Transformation

"white data"  

Linear transformation $M$

"transformed data"

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma' = \begin{bmatrix} 4.25 & 3.10 \\ 3.10 & 4.29 \end{bmatrix} = MM^T$$

Arrows are eigenvectors of $\Sigma$, size represents eigenvalues
The Largest Eigenvector of the Covariance Matrix

We can show:

To maximize variance \((x_i \cdot \nu_1)\), we can set \(\nu_1 = e_1(\Sigma)\), the first unit eigenvector of \(\Sigma\)

(proof sketch on the next slide)
Proof Sketch (For Your Curiosity)

First, easy to show: variance($x_i \cdot v_1$) = $v_1^T \Sigma v_1$

Claim: To maximize $v_1^T \Sigma v_1$, we can set $v_1 = e_1(\Sigma)$, the first unit eigenvector of $\Sigma$

Unit eigenvectors $e_d(\Sigma)$ for symmetric matrices form an orthonormal basis, so any $v$ can be written:

\[
v = \sum_{d=1}^{D} (v \cdot e_d(\Sigma))e_d(\Sigma)
\]

\[
\Sigma v = \Sigma \sum_{d=1}^{D} (v \cdot e_d(\Sigma))e_d(\Sigma) = \sum_{d=1}^{D} (v \cdot e_d(\Sigma))\Sigma e_d(\Sigma)
\]

\[
\Sigma v = \sum_{d=1}^{D} (v \cdot e_d(\Sigma))\lambda_d e_d(\Sigma)
\]

\[
v^T \Sigma v = v \cdot (\Sigma v) = \sum_{d=1}^{D} (v \cdot e_d(\Sigma))\lambda_d (v \cdot e_d(\Sigma)) = \sum_{d=1}^{D} (v \cdot e_d(\Sigma))^2 \lambda_d
\]

To maximize the weighted average, assign all your weight to the highest number!

So, must set $v \cdot e_1(\Sigma) = 1$ \implies \quad v = e_1(\Sigma)
Recap for $D' = 1$ case

- Subtract means, then compute covariance matrix as $\Sigma_1 = X^T X$
- Compute eigendecomposition of $\Sigma_1$ (e.g., using singular value decomposition)
- Set $v_1 = e_1(\Sigma_1)$
Note: Right Singular Vectors \((X) = \text{Eigenvectors} (Σ)\)

• Let data matrix \(X = U\hat{Λ}V^T\) (SVD)

• Then \(Σ = \frac{1}{N}X^TX = \frac{1}{N}V\hat{Λ}U^TU\hat{Λ}V^T = \frac{1}{N}V\hat{Λ}^2V^T\)

• So eigenvectors of covariance matrix are also the right singular vectors of the data matrix!
More than 1 dimension?

Repeat for \(d = 1, \ldots, D'\)

- Subtract means of all dimensions of \(X\)
- Compute \(\Sigma_d = X^T X\)
- Set \(\mathbf{v}_d = e_1(\Sigma_d)\)
- Set \(x_i = x_i - (x_i \cdot \mathbf{v}_d)\mathbf{v}_d\) (i.e., subtract current reconstructions to compute residuals... a little bit like gradient boosting!)

Equivalent to simply:

Repeat for \(d = 1, \ldots, D'\)

- Set \(\mathbf{v}_d = e_d(\Sigma_1)\)

We are looking for a new coordinate system \(\mathbf{v}_1, \ldots, \mathbf{v}_{D'}\) to approximate \(x_i\):

\[
x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{iD} \end{bmatrix} \approx (x_i \cdot \mathbf{v}_1)\mathbf{v}_1 + (x_i \cdot \mathbf{v}_2)\mathbf{v}_2 + \cdots + (x_i \cdot \mathbf{v}_{D'})\mathbf{v}_{D'}
\]

where the new axes \(\mathbf{v}_d\)'s are all \(D\)-dimensional unit norm, and \(D' \ll D\)

So, the new low-dimensional representation is:

\[
f(x_i) = [x_i \cdot \mathbf{v}_1, x_i \cdot \mathbf{v}_2, \ldots, x_i \cdot \mathbf{v}_{D'}]
\]
PCA on a 2D Gaussian Dataset

Each subsequent principal component:
- is orthogonal to all previous components
- indicates the direction of largest variance of the residuals

Basis vectors originate from the mean

1st principal component indicates the direction of largest variance

2nd principal component

By Nicoguaro - Own work, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=46871195
PCA Algorithm Summary So Far

Given data \( \{x_1, \ldots, x_n\} \), compute covariance matrix \( \Sigma \)

- \( X \) is the \( N \times D \) data matrix
- Compute data mean (average over all rows of \( X \))
- Subtract mean from each row of \( X \) (centering the data)
- Compute covariance matrix \( \Sigma = \frac{1}{N} X^T X \) (\( \Sigma \) is \( D \times D \))

PCA basis vectors (new coordinate axes) are given by the eigenvectors of \( \Sigma \)

- \( U, \Lambda = \text{numpy.linalg.eig}(\Sigma) \)
- \( \{u_d, \lambda_d\}_{d=1,\ldots,D} \) are the eigenvectors/eigenvalues of \( \Sigma \)
  \[ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \]

But there are \( D \) eigenvectors, so where is the dimensionality reduction?

A: Larger eigenvalue \( \Rightarrow \) “more important” eigenvectors
Dimensionality Reduction

• Can *ignore* the components of lesser significance

• You do lose some information, but if the eigenvalues are small, you don’t lose much
  – choose only the first $D'$ eigenvectors, based on their eigenvalues
  – final data set has only $D'$ dimensions
Recap

• Want to reconstruct data approximately in a new coordinate space
• Must find axes of this coordinate space, because the weights on those axes are just projections
• Objective: axes with lowest reconstruction error
  ▪ Same as axes with high variance projections
• Solution straight from linear algebra. Axes are eigenvectors of covariance matrix
PCA Example

$X = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & \cdots \\
1 & 1 & 0 & 1 & 1 & \cdots \\
0 & 0 & 1 & 1 & 1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
1 & 0 & 1 & 0 & 1 & \cdots 
\end{bmatrix}$

- $X$ has $D$ columns
- (just happens to be binary)

$U$ is the eigenvectors of $\Sigma = X^TX$; columns are ordered by importance (highest eigenvalues first)

$U = \begin{bmatrix}
0.34 & 0.23 & -0.30 & -0.23 & \cdots \\
0.04 & 0.13 & -0.40 & 0.21 & \cdots \\
-0.64 & 0.93 & 0.61 & 0.28 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
-0.20 & -0.83 & 0.78 & -0.93 & \cdots 
\end{bmatrix}$

- Each row of $U$ corresponds to a feature; keep only first $D'$ columns of $U$
- $U$ is $D \times D$
PCA

• Each column of $U$ gives weights for a linear combination of the original features

\[
U = \begin{bmatrix}
0.34 & 0.23 & -0.30 & -0.23 & \cdots \\
0.04 & 0.13 & -0.40 & 0.21 & \cdots \\
-0.64 & 0.93 & 0.61 & 0.28 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
-0.20 & -0.83 & 0.78 & -0.93 & \cdots
\end{bmatrix}
\]

\[= 0.34 \times \text{feature}_1 + 0.04 \times \text{feature}_2 - 0.64 \times \text{feature}_3 + \cdots\]
PCA

Compute $x \cdot e_d$ to get the new representation for each instance $x$

\[
X = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & \cdots \\
1 & 1 & 0 & 1 & 1 & \cdots \\
0 & 0 & 1 & 1 & 1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
1 & 0 & 1 & 0 & 1 & \cdots \\
\end{bmatrix} \quad x_3 \quad \hat{U} = \begin{bmatrix}
0.34 & 0.23 \\
0.04 & 0.13 \\
-0.64 & 0.93 \\
\vdots & \vdots \\
-0.20 & \vdots \\
-0.83 & \vdots \\
\end{bmatrix}
\]

The new 2D representation for $x_3$ is given by $[\hat{x}_{31} = x_3 \cdot u_1, \hat{x}_{32} = x_3 \cdot u_2]$:  
\[
\hat{x}_{31} = 0.34(0) + 0.04(0) - 0.64(1) + \cdots \\
\hat{x}_{32} = 0.23(0) + 0.13(0) + 0.93(1) + \cdots 
\]

The re-projected data matrix can be conveniently computed as $\hat{X} = X\hat{U}$
Eigenfaces

What happens when you compute the principal components of face images?

(1000 64×64 images)

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184
Eigenfaces

What happens when you compute the principal components of face images?

“Eigenfaces”: main directions of deviation from the mean face

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184
Let’s try reconstructing these faces with the eigenfaces now!
Eigenfaces

... with 1000 eigenvectors
Eigenfaces

... with 250 eigenvectors

https://towardsdatascience.com/eigenfacesrecovering-humans-from-ghosts-17606c328184
Eigenfaces

... with 100 eigenvectors
Eigenfaces

... with 50 eigenvectors
PCA Visualization of Digits

Fig: Laurens van der Maaten
Utility of PCA

• PCA is often used as a preprocessing step for supervised learning
  ▪ reduces dimensionality
  ▪ eliminates redundant features (i.e. linearly dependent features)

• Can also be used to aid in visualization
PCA Doesn’t Always Work Well

• Here, principal components in red don’t capture the main directions in the data.

• In general, PCA is not guaranteed to recover semantically aligned features from the data.

• The true data “shape” might not be captured by a simple linear projection of the original data.
Beyond PCA: Non-linear dimensionality reduction

Manifold Learning with 1000 points, 10 neighbors

- LLE (0.11 sec)
- LTSA (0.19 sec)
- Hessian LLE (0.37 sec)
- Modified LLE (0.22 sec)

- Isomap (0.34 sec)
- MDS (2.5 sec)
- SpectralEmbedding (0.16 sec)
- t-SNE (5.8 sec)
Beyond PCA: Non-linear dimensionality reduction

T-SNE and ISOMAP are popular and powerful nonlinear approaches, but:
• Require careful hyperparameter tuning
• Harder to optimize
• Not as easy to interpret, no easy projection back to original data

Fig: Laurens van der Maaten
Recap: Unsupervised Learning

Basic idea: reduce feature space to a much lower set of dimensions

- Clustering: find structural similarity, return one k-valued higher-level feature
- PCA: find orthonormal dimensions in order of most to least variance

- Can be useful for human inspection (visualization) as well as supervised ML