Principal Component Analysis
A Review of Linear Algebra

Every point in space can be expressed as a **linear combination** of **standard basis (or natural basis) vectors**

\[
\begin{bmatrix} 3 \\ 2 \end{bmatrix} = 3 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

The **components of the vector** tell you how far along each direction of the basis you must travel to describe your point.

A matrix can be used to transform (rotate and scale) points. This corresponds to a change of basis. The **eigenvectors** describe the new basis of the transformation matrix. For instance, data points transformed by a matrix \( A \)

\[
\begin{bmatrix}
\frac{1}{3} & 2/3 \\
2/3 & 1
\end{bmatrix}
\]

can be described in terms of its eigenvectors

\[
\begin{bmatrix}
\frac{7}{3} \\
4
\end{bmatrix} = 3.28 \cdot \begin{bmatrix} 0.52 \\ 0.85 \end{bmatrix} - 1.5 \cdot \begin{bmatrix} -0.85 \\ 0.52 \end{bmatrix}
\]
A Review of Linear Algebra

What happens when we apply the transformation to the eigenvectors themselves?

The directions of eigenvectors themselves remain unchanged under the transformation! They only get rescaled; the amount of rescaling is captured by the eigenvalue.

\[ A \mathbf{v}_1 = \lambda_1 \mathbf{v}_1 \]
\[ A \mathbf{v}_2 = \lambda_2 \mathbf{v}_2 \]

In matrix form:

\[
A \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}
\]

The eigenvectors are orthonormal, that is, they have magnitude 1 and are perpendicular to each other; which is written as \( V^T V = I \) (and thus, \( V^T = V^{-1} \) for an orthonormal matrix). So we have

\[ A = V^T \Lambda V \]

This is known as the eigen-decomposition of a matrix.

The prefix eigen- is adopted from the German word eigen for "proper" or "characteristic". Eigenvalues and eigenvectors have a wide range of applications, for example in stability analysis, vibration analysis, atomic orbitals, facial recognition, and matrix diagonalization.
A Review of Linear Algebra

If the transformation $A \in \mathbb{R}^{d \times d}$ is symmetric, then it has $d$ linearly independent eigenvectors $v_1, \ldots, v_d$ corresponding to $d$ real eigenvalues; moreover, it has $n$ linearly independent orthonormal eigenvectors

- $v_i^T v_j = 0, \forall i \neq j$
- $v_i^T v_i = 1, \forall i$

- There can be zero, negative or multiple eigenvalues corresponding to a matrix.
- The orthonormal eigenvectors form a basis of $\mathbb{R}^n$ (similar to the standard coordinate axes)
- A symmetric matrix is positive definite if and only if all of its eigenvalues are positive

Examples:

- The $2 \times 2$ identity $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ has all eigenvalues equal to 1 (positive definite) with orthonormal eigenvectors $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

- The matrix $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ has eigenvalues 0 and 2 with orthonormal eigenvectors $\begin{bmatrix} -1 \\ \sqrt{2} \end{bmatrix}$ and $\begin{bmatrix} 1 \\ \sqrt{2} \end{bmatrix}$

- The matrix $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ has eigenvalues 1 and 3 with orthonormal eigenvectors $\begin{bmatrix} -1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \end{bmatrix}$ and $\begin{bmatrix} 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \end{bmatrix}$
Principal Component Analysis: Intuition

Any point \( x \in \mathbb{R}^d \) can be written using the eigenvector basis of a (symmetric) matrix

\[
x = \sum_{i=1}^{d} c_i v_i
\]

- the weight \( c_i \) (also, co-ordinate) is the projection of \( x \) along the line given by the eigenvector \( c_i = v_i^T x \)
- Transformations using a matrix can be written as \( Ax = V^T A V x \)

Intuition: Can we use fewer eigenvectors to obtain a low-dimensional representation that approximates the transformed data point well-enough to be useful?

Original data: 17 features/dimensions

<table>
<thead>
<tr>
<th>Category</th>
<th>England</th>
<th>N Ireland</th>
<th>Scotland</th>
<th>Wales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alcoholic drinks</td>
<td>375</td>
<td>135</td>
<td>458</td>
<td>475</td>
</tr>
<tr>
<td>Beverages</td>
<td>57</td>
<td>47</td>
<td>53</td>
<td>73</td>
</tr>
<tr>
<td>Carcass meat</td>
<td>245</td>
<td>267</td>
<td>242</td>
<td>227</td>
</tr>
<tr>
<td>Cereals</td>
<td>1472</td>
<td>1494</td>
<td>1462</td>
<td>1582</td>
</tr>
<tr>
<td>Cheese</td>
<td>105</td>
<td>66</td>
<td>103</td>
<td>103</td>
</tr>
<tr>
<td>Confectionery</td>
<td>54</td>
<td>41</td>
<td>62</td>
<td>64</td>
</tr>
<tr>
<td>Fats and oils</td>
<td>193</td>
<td>209</td>
<td>184</td>
<td>235</td>
</tr>
<tr>
<td>Fish</td>
<td>147</td>
<td>93</td>
<td>122</td>
<td>160</td>
</tr>
<tr>
<td>Fresh fruit</td>
<td>102</td>
<td>674</td>
<td>957</td>
<td>137</td>
</tr>
<tr>
<td>Fresh potatoes</td>
<td>720</td>
<td>1033</td>
<td>566</td>
<td>874</td>
</tr>
<tr>
<td>Fresh Veg</td>
<td>253</td>
<td>143</td>
<td>171</td>
<td>265</td>
</tr>
<tr>
<td>Other meat</td>
<td>685</td>
<td>586</td>
<td>750</td>
<td>803</td>
</tr>
<tr>
<td>Other Veg</td>
<td>488</td>
<td>355</td>
<td>418</td>
<td>570</td>
</tr>
<tr>
<td>Processed potatoes</td>
<td>198</td>
<td>187</td>
<td>220</td>
<td>203</td>
</tr>
<tr>
<td>Processed Veg</td>
<td>360</td>
<td>334</td>
<td>337</td>
<td>365</td>
</tr>
<tr>
<td>Soft drinks</td>
<td>1374</td>
<td>1504</td>
<td>1572</td>
<td>1586</td>
</tr>
<tr>
<td>Sugars</td>
<td>156</td>
<td>139</td>
<td>147</td>
<td>175</td>
</tr>
</tbody>
</table>

Note that in this example, contrary to common convention, features are rows and training examples are columns.
Example: Face Recognition

Example: Develop a model to quickly and efficiently identify people from photographs, videos etc. in a robust manner (that is, stable and reliable under changing facial expressions, orientations, lighting conditions)

Let's suppose that our data is a collection of images of the faces of individuals
• The goal is, given the "training data" of $n$ images, to correctly match new images to the training data
• Each image is an $s \times s$ array of pixels: $x_i \in \mathbb{R}^d, d = s^2$
• As with digit recognition, construct the matrix $X \in \mathbb{R}^{n \times d}$, whose $i$-th row is the $i$-th vectorized image
  • pre-process to subtract the mean from each image
Principal Component Analysis

- Can be used to **reduce the dimensionality** of the data while still maintaining a good approximation of the sample mean and variance
- Can also be used for **selecting good features** that are combinations of the input features
- **Unsupervised** – just finds a good representation of the data in terms of combinations of the input features

**Principal Component Analysis** identifies the principal components in the **sample covariance matrix** of the data, \( X^T X \)

(∗note that since our data is #examples \( n \) x features \( d \), the covariance matrix will be \( d \times d \))

- PCA finds a set of **orthogonal vectors** that best explain the variance of the sample covariance matrix
- These are exactly the **eigenvectors** of the covariance matrix \( X^T X \)
- We can **discard the eigenvectors** corresponding to small magnitude eigenvalues to yield an approximation

**Simple algorithm** to describe, MATLAB and other programming languages have built in **support** for eigenvector/eigenvalue computations

The covariance matrix of the data \( X^T X \) is 4096 x 4096, as each image has 4096 features! Can we represent each face using significantly fewer features than 4096?

The covariance matrix is symmetric, positive semi-definite; this means all the eigen-values will be positive or zero.
Principal Component Analysis: Training

PCA Training

Given: training data $X \in \mathbb{R}^{n \times d}$
- pre-process and center the training data
- Compute the eigenvalues and eigenvectors of the covariance matrix $[V, \Lambda] = eig(X^T X)$
- Save the top $k$ eigenvectors (columns of $V$) as $V_k \in \mathbb{R}^{d \times k}$

Principal Component Analysis identifies the principal components in the covariance matrix of the face data
- in face recognition, the eigenvectors are called eigenfaces; as there are 4096 features, there are 4096 eigenfaces
- in this example, the first $k = 16$ eigenvectors capture 80.5% of the total variance (sum of all the eigenvalues)
- in practice, we compute the cumulative sum of the eigenvalues and choose $k$ such that we reach a satisfactory approximation threshold (typically, 90% of the variance)
Principal Component Analysis: Prediction

PCA Testing

**Given:** test example \( x_{\text{test}} \in \mathbb{R}^{d \times 1} 

- pre-process and center the test example
- compute the projection of \( x_{\text{test}} \) onto each of the \( k \) eigen-vectors: \( c_{\text{test}} = V_k^T x_{\text{test}} \), where \( c_{\text{test}} \in \mathbb{R}^{k \times 1} \)
- determine if the input image is close to one of the faces in the data set

Each new example can now be represented using \( k \) dimensions, by **projecting it onto the top \( k \) eigen-basis**. This means that instead of \( d = 4096 \) features, PCA now allows us to use \( k = 16 \) features!

Using more eigenvectors improves the accuracy of reconstruction, but also increases the complexity of representation and decreases the efficiency of computation. Here, the choice of \( k = 100 \) is still several orders of magnitude smaller than the original dimension, \( d = 4096 \).
PCA in Practice

Forming the sample covariance matrix $X^T X$ can require a lot of memory (especially if $n \gg d$)

- higher resolution images (256 x 256) say, require that we construct a 65536 x 65536 covariance matrix
- Need a faster way to compute this without forming the covariance matrix explicitly
- Typical approach: use the singular value decomposition

Relationship between the eigenvalue decomposition and the singular value decomposition:

- every matrix $X \in \mathbb{R}^{n \times d}$ admits a decomposition of the form $X = U \Sigma V^T$
  - where $U \in \mathbb{R}^{n \times n}$ is an orthonormal matrix, $\Sigma \in \mathbb{R}^{n \times d}$ is a non-negative diagonal matrix, and $V \in \mathbb{R}^{d \times d}$ is an orthonormal matrix
  - the $\sigma_{ii}$ entries of the diagonal matrix $\Sigma$ are called the singular values
  - $X^T X = V \Sigma^T U^T U \Sigma V^T = V (\Sigma^T \Sigma) V^T$; eigenvalues are squares of singular values; right singular vectors are eigenvectors!
PCA in Practice

While PCA is an un\textit{supervised} method, it is commonly used as a pre-processing/dimensionality reduction step for supervised classification problems. PCA does not take labels into account to determine a low-dimensional projection subspace. This means that if two classes both share a direction of maximum variance, projection into PCA space will make them \textit{inseperable}!

Approaches such as \textit{Linear Discriminant Analysis} handle this drawback by using other criteria to identify a low-dimensional subspace.