Eigenvalue Decomposition

Predictive Modeling & Statistical Learning

Gaston Sanchez

Introduction

Two Special Decompositions

Last time we talked about the Singular Value Decomposition (SVD).

In these slides, we'll talk about a closely related decomposition of SVD: the so-called Eigenvalue Decomposition (EVD) or Spectral Decomposition

Recap

Matrix decompositions, also known as matrix factorizations

$$\mathbf{M} = \mathbf{AB} \qquad \text{or} \qquad \mathbf{M} = \mathbf{ABC}$$

are a means of expressing a matrix as a product of usually two or three simpler matrices.

Types of matrices

Two types of matrices

We said that in data analysis we typically concentrate on two types of matrices:

- general **rectangular** matrices used to represent data tables.
- **positive semi-definite** matrices used to represent covariance matrices, correlation matrices, and any matrix that results from a crossproduct.

Eigenvalue Decomposition

- In contrast with SVD, the eigenvalue decomposition does NOT apply to any matrix.
- EVD applies to square matrices in general.
- A special type of square matrices are **symmetric** matrices.
- In data analysis methods, these matrices usually appear in the form of cross-product association matrices $\mathbf{X}^\mathsf{T}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\mathsf{T}$

EVD

Eigenvalue Decomposition

This decomposition applies to **symmetric** matrices such as the cross-product association matrices $\mathbf{X}^\mathsf{T}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\mathsf{T}$

Symmetric Matrices

The attractive thing about EVD is that when applied to symmetric matrices the results have a "simple" nice structure.

Eigen-Value Decomposition

EVD

An $n \times n$ symmetric matrix $\mathbf{M}$ can be decomposed as:

$$\mathbf{M} = \mathbf{A}\mathbf{B}\mathbf{A}^{\mathsf{T}}$$

where

- $\mathbf{A}$ is a $n \times p$ column **orthonormal** matrix containing the eigen-vectors of $\mathbf{M}$
- $\mathbf{B}$ is a $p \times p$ **diagonal** matrix containing the eigen-values of $\mathbf{M}$

Eigen-Value Decomposition

EVD

A more common notation for EVD is:

$$\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}}$$

where

- $\mathbf{U}$ is a $n \times p$ column **orthonormal** matrix containing the eigen-vectors of $\mathbf{M}$
- $\mathbf{\Lambda}$ is a $p \times p$ **diagonal** matrix containing the eigen-values of $\mathbf{M}$

EVD

$$\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$$

$$\mathbf{M} = \begin{bmatrix} u_{11} & \cdots & u_{1p} \\ u_{21} & \cdots & u_{2p} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{np} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} u_{11} & \cdots & u_{n1} \\ u_{12} & \cdots & u_{n2} \\ \vdots & \ddots & \vdots \\ u_{1p} & \cdots & u_{np} \end{bmatrix}$$

Eigenvectors

Vectors, which under a given transformation $\mathbf{M}$ map into themselves or multiples of themselves, are called invariant vectors under that transformation. It follows that such vectors satisfy the relation:

$$\mathbf{Mx} = \lambda\mathbf{x}$$

where λ is a scalar.

Eigenvectors

The matrix equation:

$$\mathbf{M}\mathbf{x} = \lambda\mathbf{x}$$

can be rearranged as follows:

$$\mathbf{M}\mathbf{x} - \lambda\mathbf{x} = \mathbf{0}$$

Eigenvectors

Given

$$\mathbf{M}\mathbf{x} - \lambda\mathbf{x} = \mathbf{0}$$

We can factor out $\mathbf{x}$

$$(\mathbf{M}\mathbf{x} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$$

Eigenvectors

Obtaining the eigenstructure of a (square) matrix involves solving the **characteristic equation**

$$det(\mathbf{Mx} - \lambda_i \mathbf{I}) = 0$$

If $\mathbf{M}$ is of order $n \times n$, then we can obtain n roots of the equation. These roots are called the **eigenvalues**.

EVD in R

`eigen()` in R

`eigen()` function

R provides the function `eigen()` to perform an eigenvalue decomposition of a square matrix.

`eigen()` output

A list with the following components

- `values` a vector containing the eigenvalues
- `vectors` a matrix whose columns contain the eigenvectors

EVD example in R

```r
# X'X matrix
set.seed(22)
X <- as.matrix(USArrests)
XtX <- t(X) %*% X

# eigenvalue decomposition
EVD = eigen(XtX)

# elements returned by eigen()
names(EVD)
```

```
## [1] "values"  "vectors"
```

```r
# vector of eigenvalues
(lambdas = EVD$values)
```

```
## [1] 2013735.2431   37957.1103    2084.9578     326.5089
```

EVD example in R (con't)

```
# matrix of eigenvectors
(V <- EVD$vectors)
```

```
##             [,1]        [,2]        [,3]        [,4]
## [1,] -0.04239181  0.01616262  0.06588426  0.99679535
## [2,] -0.94395706  0.32068580 -0.06655170 -0.04094568
## [3,] -0.30842767 -0.93845891 -0.15496743  0.01234261
## [4,] -0.10963744 -0.12725666  0.98347101 -0.06760284
```

Power Method

About the Power Method

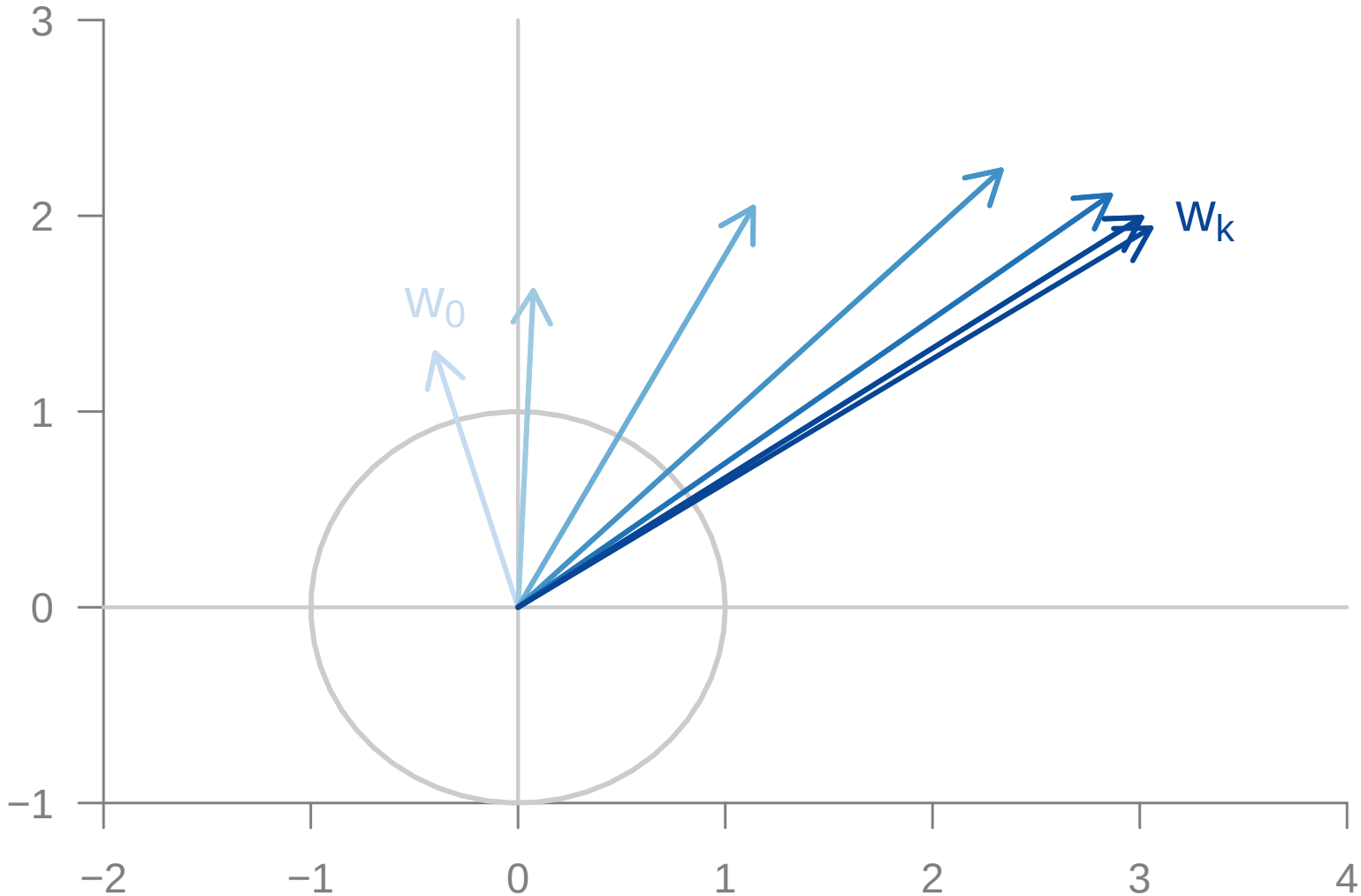
One of the basic procedures following a successive approximation approach is precisely the Power Method.

In its simplest form, the Power Method (PM) allows us to find **the largest** eigenvector and its corresponding eigenvalue.

About the Power Method

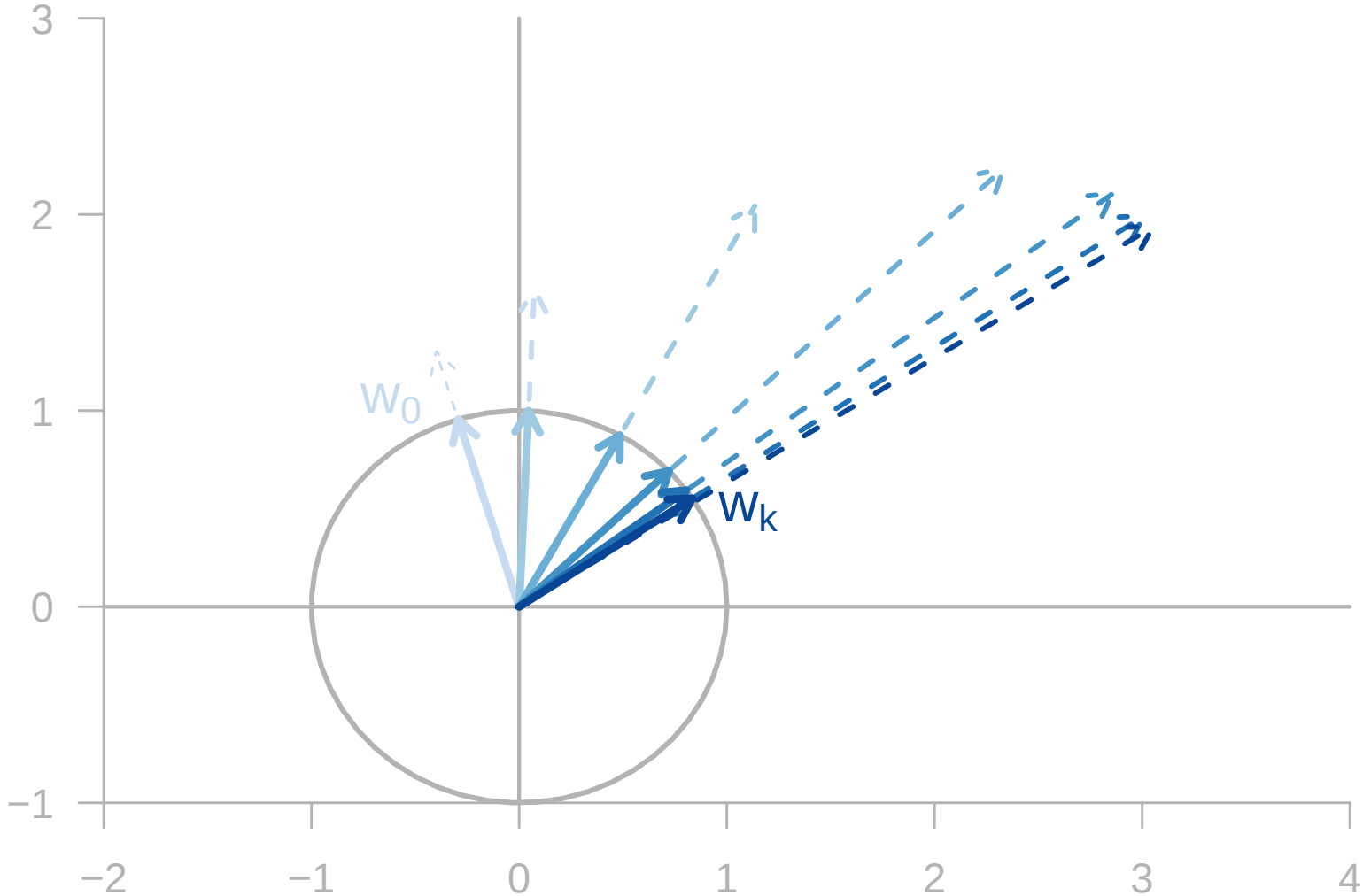
Choose an arbitrary vector $\mathbf{w_0}$ to which we will apply the symmetric matrix $\mathbf{S}$ repeatedly to form the following sequence:

$$\begin{aligned}
\mathbf{w_1} &= \mathbf{S}\mathbf{w_0} \\
\mathbf{w_2} &= \mathbf{S}\mathbf{w_1} = \mathbf{S^2}\mathbf{w_0} \\
\mathbf{w_3} &= \mathbf{S}\mathbf{w_2} = \mathbf{S^3}\mathbf{w_0} \\
&\vdots \\
\mathbf{w_k} &= \mathbf{S}\mathbf{w_{k-1}} = \mathbf{S^k}\mathbf{w_0}
\end{aligned}$$


3
2
1
0
−1
−2
−1
0
1
2
3
4
w_0
w_k

About the Power Method

- In practice, we must rescale the obtained vector $\mathbf{w_k}$ at each step.
- The rescaling will allows us to judge whether the sequence is converging.
- After some iterations, the vector $\mathbf{w_{k-1}}$ and $\mathbf{w_k}$ will be very similar
- Assuming a reasonable scaling strategy, the sequence will usually converge to the dominant eigenvector of $\mathbf{S}$.


3
2
1
0
−1
−2
−1
0
1
2
3
4
w_0
w_k

Dominant Eigenvalue

The obtained vector is the dominant eigenvector. To get the corresponding eigenvalue we calculate the so-called **Rayleigh quotient** given by:

$$\lambda = \frac{\mathbf{w_k'} \mathbf{S'} \mathbf{w_k}}{\left\| \mathbf{w_k} \right\|^2}$$

Remarks

Conditions for the power method to be succesfully used:

- The matrix must have a *dominant* eigenvalue.
- The starting vector $\mathbf{w_0}$ must be nonzero.
- We need to scale each of the vectors $\mathbf{w_k}$
  otherwise the algorithm will "explode"

PM Pseudocode

Let's consider a more detailed version of the PM algorithm:

1. Start with an arbitraty initial vector $\mathbf{w}$
2. Obtain product $\tilde{\mathbf{w}} = \mathbf{S}\mathbf{w}$
3. Normalize $\tilde{\mathbf{w}}$

$$\text{e.g.} \quad \mathbf{w} = \frac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|}$$

4. Compare $\mathbf{w}$ with its previous version
5. Repeat steps 2 till 4 until convergence

Why does the PM work?

Assume that the matrix $\mathbf{S}$ has p eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$, and that they are ordered in decreasing way $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_p|$.

Note that the first eigenvalue is strictly greater than the second one. This is a very important assumption.

In the same way, we'll assume that the matrix $\mathbf{S}$ has p linearly independent vectors $\mathbf{u_1}, \ldots, \mathbf{u_p}$ ordered in such a way that $\mathbf{u_j}$ corresponds to λ_j.

Why does the PM work?

The initial vector $\mathbf{w_0}$ may be expressed as a linear combination of $\mathbf{u_1}, \ldots, \mathbf{u_p}$

$$\mathbf{w_0} = a_1\mathbf{u_1} + \cdots + a_p\mathbf{u_p}$$

At every step of the iterative process the vector $\mathbf{w_k}$ is given by:

$$\mathbf{S}^k = a_1\lambda_1^k\mathbf{u_1} + \cdots + a_p\lambda_p^k\mathbf{u_p}$$

Why does the PM work?

Since λ_1 is the dominant eigenvalue, the component in the direction of $\mathbf{u_1}$ becomes relatively greater than the other components as k increases. If we knew λ_1 in advance, we could rescale at each step by dividing by it to get:

$$\left(\frac{1}{\lambda_1^k}\right)\mathbf{S}^k = a_1\mathbf{u_1} + \cdots + a_p\left(\frac{\lambda_p^k}{\lambda_1^k}\right)\mathbf{u_p}$$

which converges to the eigenvector $a_1\mathbf{u_1}$, provided that a_1 is nonzero.

Why does the PM work?

Of course, in real life this scaling strategy is not possible—we don't know λ_1. Consequenlty, the eigenvector is determined only up to a constant multiple, which is not a concern since the really important thing is the *direction* not the length of the vector.

The speed of the convergence depends on how bigger λ_1 is respect with to λ_2, and on the choice of the initial vector $\mathbf{w_0}$. If λ_1 is not much larger than λ_2, then the convergence will be slow.

More Remarks

- The power method is that it is a sequential method.
- We can obtain $\mathbf{w_1}, \mathbf{w_2}$, and so on.
- If we only need the first k vectors, we can stop the procedure at the desired stage.

Obtaining more eigenvectors?

Once we've obtained the first eigenvector $\mathbf{w_1}$, we can compute the second vector by reducing the matrix $\mathbf{S}$ by the amount explained by the first eigenvector.

This operation of reduction is called **deflation** and the residual matrix is obtained as:

$$\mathbf{S_1} = \mathbf{S} - \mathbf{w_1}^\top \mathbf{w_1}$$

To get the second eigenvalue and its corresponding eigenvector, we operate on $\mathbf{S_1}$ in the same way as the operations on $\mathbf{S}$.