Lecture 7 - Covariance, EVD, PCA & SVD

Summary

Understanding the shape of data in a feature space is important to effectively using it. In addition, by understanding the distribution of really highly dimensional data, it is possible to determine the most important modes of variation of that data, and thus represent the data in a space with many fewer dimensions. This is the goal of Principle Component Analysis (PCA).

Eigendecomposition (EVD) and Singular Value Decomposition (SVD) are important mathematical tools for performing PCA. In addition SVD has numerous applications throughout data-mining, and is perhaps the most important mathematical tool that you'll come across.

Key points

Variance and covariance

- Mathematicians talk about variance and covariance in terms of **random variables** and **expected values**.
  - For our purpose, a random variable can be thought of as the set of values from a *single dimension* of some or all the data in a feature space.
  - The expected value of such a variable is just its mean value.
- Variance ($\sigma^2(x)$) of a set of n data points, $x = [x_1,x_2,...,x_n]$, is the average squared difference from the mean (μ):

$$\sigma^2(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

  - Variance measures how *spread-out* the data is from the mean
- Covariance measures how two variables (x and y) change together:

$$\sigma(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

  - Variance is the covariance when the two variables are the same: $\sigma(x,x) = \sigma^2(x)$
  - A covariance of 0 means that the variables are **uncorrelated**
    - Covariance is in fact related to correlation:

$$\rho(x,y) = \frac{\sigma(x,y)}{\sqrt{\sigma^2(x)}\sqrt{\sigma^2(y)}}$$

  - Also note that $\sigma(x,y) = \sigma(y,x)$
- The covariance matrix, $\mathbf{\Sigma}$, encodes how all possible pairs of dimensions in a n-dimensional dataset (i.e. points in a feature space), $\mathbf{X}$, vary together:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma(X_1,X_1) & \sigma(X_1,X_2) & \dots & \sigma(X_1,X_n) \\ \sigma(X_2,X_1) & \sigma(X_2,X_2) & \dots & \sigma(X_2,X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(X_n,X_1) & \sigma(X_n,X_2) & \dots & \sigma(X_n,X_n) \end{bmatrix}$$

  where the vector X_i is formed from the i-th element of all the vectors in the feature space.

- The covariance matrix is a **symmetric matrix**

Mean centred data

- **Mean centring** a set of vectors is the process of subtracting the mean (computed from all [or a significant sample] of the vectors) from each vector.
- If you have a set of n mean-centred vectors, you can form them into a matrix, $\mathbf{Z}$, where each *row* corresponds to one of your vectors. The covariance matrix is then directly proportional to the transpose of $\mathbf{Z}$ multiplied by $\mathbf{Z}$: $\mathbf{\Sigma} \propto \mathbf{Z}^T\mathbf{Z}$

Principle axes of variation

- A basis is a set of linearly independent vectors that forms a "coordinate system".
  - As the vectors are linearly independent, they are **orthogonal**.
  - For a given dimensionality, there are an infinite number of possible basis.
- In the two-dimensional case, the covariance matrix (or indeed any other 2×2 symmetric matrix) can be seen to define an ellipse with major and minor axes (the actual reason for this is related to a mathematical concept called "Quadratic forms", which is even applicable in higher dimensions).
  - The major axis is along the dimension of which the underlying data is most spread.
  - The minor axis is **perpendicular** to the major axis.
- With more dimensions a similar pattern emerges:
  - The (first) principle axis is along the dimension of which the underlying data is most spread.
    - The second principle axis is in the direction in which the data is most spread orthogonal to the principal axis.
  - The third principle axis is in the direction in which the data is most spread orthogonal to the principal axis and the second principal axis.
  - And so on...
- The set of principal axes is a **basis**.

The Eigendecomposition of the covariance matrix

- An **eigenvector** of a square matrix $\mathbf{A}$ is a non-zero vector v that, when the matrix is multiplied by v, yields a constant multiple of v commonly denoted as λ: $\mathbf{A}v = \lambda v$
  - λ is called the **eigenvalue** of $\mathbf{A}$ corresponding to the vector v.
- If $\mathbf{A}$ is $N \times N$, then there are at most N unique eigenvalue-vector pairs.
- If $\mathbf{A}$ is symmetric, then the set of all eigenvectors of $\mathbf{A}$ is a basis and the eigenvectors are **orthogonal**.
- If the matrix $\mathbf{A}$ is a covariance matrix, then it turns out that **the eigenvectors are the principal components!**
  - The vector with the largest eigenvalue is the principal axis, the vector with the second largest eigenvalue is the second principal axis, and so on.
  - **Eigenvalues turn out to be proportional to the variance along an axis!**
- Formally, the **Eigendecomposition** (EVD) factorises a diagonalisable square matrix $\mathbf{A}$ such that: $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$ where $\mathbf{Q}$ is the square ($N \times N$) matrix whose i-th column is the eigenvector q_i of $\mathbf{A}$ and $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues (i.e., $\mathbf{\Lambda}_{ii} = \lambda_i$).
  - **The Eigendecomposition is thus a way of finding the principal axes**
  - If $\mathbf{A}$ is a real symmetric matrix (such as a covariance matrix) then $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{Q}^{-1} = \mathbf{Q}^T$
  - The Eigendecomposition can be solved analytically for very small matrices (i.e. $N \leq 4$). For larger matrices it is solved using iterative numerical methods (see below).
  - It is common practice to arrange the columns of $\mathbf{Q}$ and corresponding eigenvalues in $\mathbf{\Lambda}$, such

that the eigenvalues decrease (i.e. $\lambda_i > \lambda_{i+1}$).

Dimensionality reduction with Principle Component Analysis

- A linear transform ($\mathbf{W}$) maps vectors z_i (rows of $\mathbf{Z}$) from one space to another: $\mathbf{T} = \mathbf{ZW}$ where $\mathbf{T}$ is the transformed space (vectors t_i from the rows of $\mathbf{T}$ correspond to the original vectors z_i in the transformed space).
  - $\mathbf{T}$ can have fewer dimensions than $\mathbf{Z}$.
- PCA is mathematically defined as an **orthogonal linear transformation** (meaning it rotates and scales) that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.
  - PCA thus projects data in an original space to a new space defined by the basis of principal axes. The transform matrix is just the eigenvector matrix $\mathbf{Q}$: $\mathbf{W} = \mathbf{Q}$
  - Because the new (principle) axes are sorted by variance, we can choose to ignore any axes with small variance, thus providing a way of **reducing the dimensionality** of the data.
    - Keeping only the first L principal components (i.e. columns of $\mathbf{Q}$, assuming the eigenvectors are sorted by decreasing eigenvalue) gives a truncated transformation: $\mathbf{T}_L = \mathbf{ZQ}_L$ where the matrix $\mathbf{T}_L$ now has n rows but only L columns.
  - Given a low-dimensional vector formed from PCA, it is possible to reconstruct the original vector: $t_L = z\mathbf{Q}_L \Rightarrow z = t_L\mathbf{Q}_L^{-1} = t_L\mathbf{Q}_L^{\mathrm{T}}$
    - Then add the mean vector to get back into the original space before mean centring.
  - Summary of the steps for PCA:

1. Mean-centre the data vectors
2. Form the vectors into a matrix $\mathbf{Z}$, such that each row corresponds to a vector
3. Perform the Eigendecomposition of the matrix $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$, to recover the eigenvector matrix $\mathbf{Q}$ and diagonal eigenvalue matrix $\mathbf{\Lambda}$: $\mathbf{Z}^{\mathrm{T}}\mathbf{Z} = \mathbf{Q\Lambda Q}^{-1}$
4. Sort the columns of $\mathbf{Q}$ and corresponding diagonal values of $\mathbf{\Lambda}$ so that the eigenvalues are decreasing.
5. Select the L largest eigenvectors of $\mathbf{Q}$ (the first L columns) to create the transform matrix $\mathbf{Q}_L$.
6. Project the original vectors into a lower dimensional space, $\mathbf{T}_L$: $\mathbf{T}_L = \mathbf{ZQ}_L$

Singular Value Decomposition (SVD)

- Recall the rank, ϱ, of a matrix, $\mathbf{M}$ is the number of linearly independent rows or columns: $\varrho = \mathrm{rank}(\mathbf{M})$
- For a real $m \times n$ matrix, $\mathbf{M}$, the SVD is defined as: $\mathbf{M} = \mathbf{U\Sigma V}^{\mathrm{T}}$
  - Note that to keep with standard notation we're re-using the $\mathbf{\Sigma}$ symbol - other than being square, it's a very different matrix to the covariance matrix in SVD
  - $\mathbf{\Sigma}$ is a real diagonal matrix
    - The diagonal values are the "Singular Values" of $\mathbf{M}$
    - $\mathbf{\Sigma}$ has dimensions $\varrho \times \varrho$
    - The singular values are normally arranged in monotonically decreasing order: $\mathbf{\Sigma}_{i,i} \geq \mathbf{\Sigma}_{i+1,i+1}$
  - The matrices $\mathbf{U}$ and $\mathbf{V}$ are orthogonal: $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{UU}^{\mathrm{T}} = \mathbf{I}$ and $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{VV}^{\mathrm{T}} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix (note that the size of $\mathbf{I}$ is different for the identities involving $\mathbf{U}$ and $\mathbf{V}$ respectively).
    - $\mathbf{U}$ has dimensions $m \times \varrho$
      - The m dimensional vectors that form the columns of $\mathbf{U}$ are called the *left singular vectors*
      - The left singular vectors are the eigenvectors of $\mathbf{MM}^{\mathrm{T}}$

        - $\mathbf{V}^{\perp}$ has dimensions $\varrho \times n$ (equally $\mathbf{V}$ has dimensions $n \times \varrho$)
            - The n dimensional vectors that form the columns of $\mathbf{V}$ (rows of $\mathbf{V}^{\mathrm{T}}$) are called the *right singular vectors*
            - The right singular vectors are the eigenvectors of $\mathbf{M}^{\mathrm{T}}\mathbf{M}$
    - The singular values are the square roots of the corresponding eigenvalues of *both* $\mathbf{M}\mathbf{M}^{\mathrm{T}}$ and $\mathbf{M}^{\mathrm{T}}\mathbf{M}$
- Relationship of SVD to PCA
    - If $\mathbf{Z}$ is a matrix of mean-centred feature vectors, then the left singular values of $\mathbf{Z}$ are the eigenvectors of $\mathbf{Z}\mathbf{Z}^{\mathrm{T}}$ and thus are the principal components of $\mathbf{Z}$
        - This means you can actually perform PCA without explicitly computing the covariance matrix
            - Better numerical stability
            - Potentially much faster
- Truncated SVD
    - Given the relationship between PCA and SVD, it's easy to see that computing a truncated SVD that only considers the top-r singular values and respective left and right singular vectors is useful for dimensionality reduction
    - Low rank matrix approximation is also possible and it can be proved (the Eckart-Young theorem) that keeping only the top-r S.V.s and reconstructing a matrix $\acute{\mathbf{M}} = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^{\mathrm{T}}$ gives the *best* possible rank-r approximation of the original matrix, $\mathbf{M}$, in the sense that the Frobenius norm $||\mathbf{M} - \acute{\mathbf{M}}||_{\mathrm{F}}$ is minimised.
        - Frobenius Norm of $\mathbf{A}$ is just the square root of the sum of all the elements of $\mathbf{A}$ squared; it's basically a generalisation of the Euclidean norm of a vector to a matrix.
- Other uses of SVD
    - Pseudoinverse
        - The Moore-Penrose pseudoinverse is a generalisation of the standard matrix inverse.
            - Can be computed using SVD: $\mathbf{A}^{\dagger} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^{\mathrm{T}}$
        - Key application is in finding least squares ("best-fit") solutions to systems of linear equations
            - Solution of $\mathbf{A}x = b$ for x where $||\mathbf{A}x - b||_2$ is minimised is $x = \mathbf{A}^{\dagger}b$
            - Example: Least squares line fitting using the pseudo-inverse:
                - assume we have a set of (x, y) points $[(x_1, y_1), \ldots, (x_i, y_i)]$ for which we want to fit a straight line of best fit of the form $y = mx + c$. We can write this in the form $\mathbf{A}x = b$ as follows:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_3 \end{bmatrix}$$

                and solve by taking the pseudo-inverse:

$$\begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_i & 1 \end{bmatrix}^{\dagger} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_3 \end{bmatrix}$$

        - Solving homogeneous equations of the form $\mathbf{A}x = 0$
            - Solution for the non-zero x case is any right singular vector with a corresponding singular value of 0.
                - Requires a modification to the SVD, called the "full SVD" to be computed,

where the left and right singular vectors for the null-space of the matrix is computed (i.e. singular vectors that have corresponding singular values equal to zero).

- Data mining
  - Model based CF, Latent factor models, Information retrieval, ... *and many more*

Computing SVD and EVD

- All general EVD algorithms are iterative
- Simplest classical approach to computing eigenvectors is the "Power Iteration"
  - This computes the eigenvector with the biggest eigenvalue
  - To compute more than one eigenvector, compute the biggest, transform (rotate) the input data and truncate it so that the effect of that eigenvector is removed, then repeat...
- More efficient variants like the "Arnoldi Iteration" and the "Lanczos algorithm" exploit the idea of Krylov subspaces to compute approximate eigenvectors and use the "Gram-Schmidt" process to orthonormalise them
- These standard approaches are implemented in packages like LAPACK and ARPACK
  - Allow efficient computation of biggest r eigenvalue-vector pairs (and computation of truncated SVD)
  - Variants also allow the smallest s eigenvector-value pairs to be computed (without the need to compute the biggest)
    - Useful for "spectral clustering" - we'll come back to this when we talk about stream mining and social event detection
  - Can be applied to relatively large sparse input matrices
    - **But not really massive matrices** - more on this in a later lecture when we talk about the Netflix challenge

Further Reading

- Wikipedia has good coverage of all the key ideas:
  - http://en.wikipedia.org/wiki/Variance (http://en.wikipedia.org/wiki/Variance)
  - http://en.wikipedia.org/wiki/Covariance (http://en.wikipedia.org/wiki/Covariance)
  - http://en.wikipedia.org/wiki/Covariance_matrix (http://en.wikipedia.org/wiki/Covariance_matrix)
  - http://en.wikipedia.org/wiki/Eigenvalue,_eigenvector_and_eigenspace (http://en.wikipedia.org/wiki/Eigenvalue,_eigenvector_and_eigenspace)
  - http://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix (http://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix)
  - http://en.wikipedia.org/wiki/Eigenface (http://en.wikipedia.org/wiki/Eigenface)
  - https://en.wikipedia.org/wiki/Singular_value_decomposition (https://en.wikipedia.org/wiki/Singular_value_decomposition)
  - https://en.wikipedia.org/wiki/Power_iteration (https://en.wikipedia.org/wiki/Power_iteration)