Just Enough Spectral Theory for Machine Learning

Ng Yen Kaow

Notations (Important)

- A vector is by default a column
  - For vectors x and y, their inner (or dot) product, $\langle x, y \rangle = x^{\mathrm{T}} y$
    - $\langle x + z, y \rangle = \langle x, y \rangle + \langle z, y \rangle = x^T y + z^T y$
  - Beware: some texts use row vectors and $\langle x, y \rangle = x y^{\mathrm{T}}$

- For a matrix an example is a row
  - An example (or datapoint) is a row x_i while each feature is a columns
    - Features are like fixed columns in a spreadsheet
  - For matrices X and Y, $\langle X, Y \rangle = X Y^{\mathrm{T}}$ or $\sum_i \left(x_i y_i^{\mathrm{T}} \right)$
  - Beware: some texts use column for examples and let $\langle X, Y \rangle = X^{\mathrm{T}} Y$

- So it's $x^{\mathrm{T}} x$, $x^{\mathrm{T}} M x$, but $X X^{\mathrm{T}}$ and $Q \Lambda Q^{\mathrm{T}}$

What about outer product?

- The outer product of two vectors is a matrix

$$\begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} c & d \end{pmatrix} = \begin{pmatrix} ac & ad \\ bc & bd \end{pmatrix}$$

- The outer product (or Kronecker product) of two matrices is a **tensor**

- We don’t deal with outer products yet

Python call for inner product

- Inner products are performed with `np.dot()`
  - When called on two arrays, the arrays are **automatically** oriented to perform inner product
    - Note that `[[1],[1]]` is a 1×2 matrix
  - When called on an array `x` and a matrix `X`, the array is **automatically** read as a row for `np.dot(x,X)`, and column for `np.dot(X,x)` to perform inner product
  - When called on two matrices, make sure that the matrices are oriented correctly, or you will get $X^{\mathrm{T}}X$ when you want XX^{T}
  - Impossible to get outer product with `np.dot()`
- If you write `x*y` or `X*Y`, what you get is an element-wise multiplication

Eigenvectors and eigenvalues

- Only concerned with square matrices
  - Most matrices we consider are furthermore **symmetric** (and of only **real** values)
- A eigenvector for a square matrix M is vector u where $Mu = \lambda u$
  - u is invariant under transformation M
  - The scaling factor λ is called a eigenvalue

Eigendecomposition

- A eigendecomposition of matrix M is
$$M = Q\Lambda Q^{-1}$$
where Λ is diagonal, and Q contains (not necessarily orthogonal) eigenvectors of M
  - Any normal M can be eigendecomposed
  - **The set of eigenvalues for M is unique**
  - **There can be different eigenvectors of the same eigenvalue** (hence not unique)
    - **For real symmetric M, eigenvectors that correspond to distinct eigenvalues are orthogonal**
- For an orthogonal matrix Q, $Q^{-1} = Q^{\mathrm{T}}$
- **Only consider real symmetric** $M \Rightarrow M = Q\Lambda Q^{\mathrm{T}}$

Eigenspace

- The eigenspace of a matrix M is the set of all the vectors u that fulfills $Mu = \lambda u$
  - The rank of M is its number of non-zero λ
  - Any feature vector v_k in M can be written as a linear combination of the eigenspace, i.e. $v_k = \sum_j \langle v_k, u_j \rangle u_j$
  - Any eigenvector u_k of M can be written as a linear combination of the feature vectors v_i in M, by solving the system of equations $v_i = \sum_j \langle v_i, u_j \rangle u_j$ to obtain u_k entirely in terms of v_i
- A eigenbasis of a $n \times n$ matrix M is a set of n orthogonal eigenvectors of M (including those with zero eigenvalues)

Rayleigh Quotient

- Consider an $n \times n$ real symmetric M
- $M = Q\Lambda Q^T$, where Λ is diagonal, and Q is the eigenbasis of M

- Denote the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.
  - Then, for all unit vector u

Min-max Theorem

$$\max_{\|u\|=1} \frac{u^{\mathrm{T}} M u}{u^{\mathrm{T}} u} = \lambda_1$$

Similarly, λ_n is the minimum of the Rayleigh Quotient

  - And for all orthogonal matrix P and $k \leq n$

Minimax Principle

$$\max_{P \in \mathbb{R}^{k \times n}, P^{\mathrm{T}} P = I} \mathrm{tr}\left(P^{\mathrm{T}} M P\right) = \lambda_1 + \cdots + \lambda_k$$

Similarly, $\min_{P \in \mathbb{R}^{k \times n}, P^{\mathrm{T}} P = I} \mathrm{tr}\left(P^{\mathrm{T}} M P\right) = \lambda_{n-k+1} + \cdots + \lambda_n$

Eigendecomposition applications

- Matrix inverse
- Matrix approximation
- Matrix factorization
  - Multidimensional Scaling
- Minimization or maximization through the Rayleigh Quotient
  - PCA
    - Max of covariance matrix
  - Spectral clustering
    - Min of graph Laplacian

Singular Value Decomposition

- Any matrix can be singular value decomposed
- $M = U\Sigma \mathrm{V}^*$
  - M is $m \times n$ matrix
  - U is an $m \times m$ unitary (orthogonal) matrix
  - Σ is an $m \times n$ diagonal matrix
  - V is an $n \times n$ unitary matrix

- For a real M, $V^* = V^{\mathrm{T}}$ (and $U = U^{\mathrm{T}}$) hence

$$M = U\Sigma V^{\mathrm{T}}$$

SVD applications

- Solving linear equations
- Linear regression
- Pseudoinverse
- Kabsch algorithm
- Matrix approximation
- As a eigendecomposition (see next slide)

SVD and eigendecomposition

- SVD is a eigendecomposition but not of M
  - Given an SVD of $M = U\Sigma V^*$
  - Then, clearly
    - $M^*M = V\Sigma^*U^*U\Sigma V^* = V(\Sigma^*\Sigma)V^*$
    - $MM^* = U\Sigma V^*V\Sigma^*U^* = U(\Sigma^*\Sigma)U^*$
  - Hence V is the eigenbasis of M^*M and U is the eigenbasis of MM^* respectively
  - That is, U and V are **eigenbases of the squared matrices of M**
    - However the eigenbasis of M^*M and MM^* are in general not the eigenbasis of M