Dimensionality Reduction/Feature Extraction

Sunday 28 April 2024 5:47 AM

Curse Of Dimensionality

Sunday 28 April 2024 6:05 AM

→ When developing ML models, the aim is to maximise the amount of info provided to the algorithm in order to produce accurate & informative predictions.

→ The temptation is to blindly include as many predictors as possible into the model. However this does not always improve performance & in some circumstances can reduce accuracy.

→ COD refers to the situation where providing additional predictors to ML models can lead to reduced performance. (optimal # of features)

→ When allocating new variables to a ML model, if the number of training data points remain constant then issues can develop. It is typically necessary to collect more training samples which is necessary to cover enough of the problem space that a model needs to properly learn the dataset.

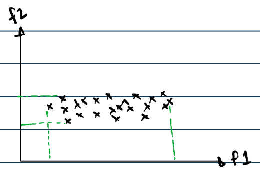
→ Data becomes sparse in higher dimensions.

→ Unsupervised machine problem

→ PCA is a technique which can transform a higher dimension data into lower dimension data while keeping the essence of the data.

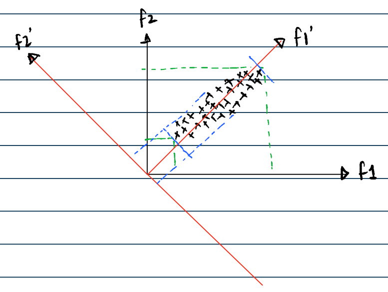
→ Benefits
- faster algorithm execution
- Visualization

→ choosing feature/axis on which the dataset displays more variance i.e. losing less information



$$X = \begin{array}{c} f1 \quad f2 \\ \begin{bmatrix} & \\ & \end{bmatrix}_{n\times 2} \end{array} \longrightarrow X' = \begin{array}{c} f1 \\ \begin{bmatrix} \\ \end{bmatrix}_{n\times 1} \end{array}$$

→ What happens if the variances are similar?

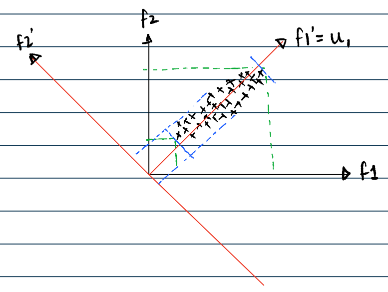


① $f1' \perp f2'$ ← find a direction f1' such that the variance of xi's projected onto f1' maximal.

spread on f2' <<< spread f1'

$$X = \begin{array}{c} f1 \quad f2 \\ \begin{bmatrix} & \\ & \end{bmatrix} \end{array} \longrightarrow X' = \begin{array}{c} f1' \\ \begin{bmatrix} \\ \end{bmatrix} \end{array}$$

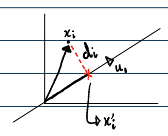
→ **Mathematical formulation**



find u_1 s.t $\text{var}\{\text{proj}_{u_1} x_i\}_{i=1}^{n}$ is maximum

u_1 = unit-vector

$\|u_1\| = 1$



x_i' = projection of x_i onto u_1 $\longrightarrow \dfrac{u_1 \cdot x_i}{\|u_1\|^2 = 1} = u_1^T x_i$

$D = \{x_i\}_{i=1}^{n} \longrightarrow D' = \{x_i'\}_{i=1}^{n}$

$x_i' = u_1^T x_i$

$\bar{x}_i' = u_1^T \bar{x}_i$ → $\bar{x}_i$: mean $\{x_i\}_{i=1}^{n}$

$$\boxed{\max\left(\text{Var}\{u_1^T x_i\}_{i=1}^{n}\right) = \max\left(\frac{1}{n}\sum_{i=1}^{n}\left(u_1^T x_i - \underbrace{u_1^T \bar{x}_i}_{0 \text{ (as we mean centered)}}\right)^2\right)} \longrightarrow \underbrace{\left|\max_{u_1} \frac{1}{n}\sum_{i=1}^{n}(u_1^T x_i)^2\right|}_{\text{variance maximization}} \text{ such that } u_1^T u_1 = 1 = \|u\|^2 \quad \text{OR} \quad \underbrace{\left|\min_{u_1} \sum_{i=1}^{n} \overbrace{x_i^T x_i - (u_1^T x_i)^2}^{d_i^2}\right|}_{\text{Distance Minimization}} \text{ such that } u_1^T u_1 = 1$$

→ Covariance matrix [square & symmetrical]

$$\begin{array}{c|ccc} & x & y & z \\ \hline x & V_x & C_{x,y} & C_{x,z} \\ y & C_{y,x} & V_y & C_{y,z} \\ z & C_{z,x} & C_{z,y} & V_z \end{array}$$

defines both the spread & orientation of our data

$\boxed{\text{cov}(X) = \dfrac{X^T X}{n-1}}$ → * make sure features are mean centered / column standardized

→ Eigenvectors are vectors which maintain their direction after Linear transformations

→ Eigenvalues are the scaling factor of the eigenvectors after Linear transformations

$\boxed{A\vec{v} = \lambda\vec{v}}$

→ The largest eigenvector of the covariance matrix always points into the direction of the largest variance of the data & the magnitude of this vector equals the corresponding eigenvalue. The second largest eigenvector is always ⊥ to the largest eigenvector, & points into the direction of the second largest spread of the data.

1) mean centering the data

2) find covariance matrix

3) find the eigen values/eigen vectors of cov matrix

4) Transform data $U^T \cdot x$

from sklearn.decomposition import PCA

(eigen values) .explained_variance_ ← → n_components

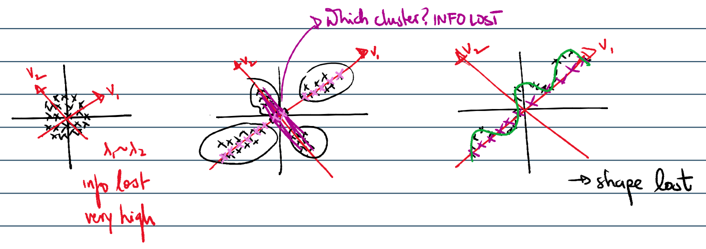
(eigen vectors) .components_ ← → fit_transform

.explained_variance_ratio ← → transform

$$\left(\frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i} \right) \times 100$$, optimal principal components = total explained variance is 90%

→ % of variance explained

→ When does PCA fail/limitations? →



→ Kernel PCA [Research]

→ Incremental & Randamized PCA [Research]

→ PCA
- eigen decomposition
- SVD → $X_c = U \Sigma V^T$

$X_c^T X_c = V \gamma V^T$

→ cov

→ Explainability is lost/Difficult esp. Pharma & Finance

→ effects Decision Making

747

+200

Let the real values data matrix $\mathbf{X}$ be of $n \times p$ size, where n is the number of samples and p is the number of variables. Let us assume that it is *centered*, i.e. column means have been subtracted and are now equal to zero.

Then the $p \times p$ covariance matrix $\mathbf{C}$ is given by $\mathbf{C} = \mathbf{X}^\top \mathbf{X}/(n-1)$. It is a symmetric matrix and so it can be diagonalized:

$$\mathbf{C} = \mathbf{V}\mathbf{L}\mathbf{V}^\top,$$

where $\mathbf{V}$ is a matrix of eigenvectors (each column is an eigenvector) and $\mathbf{L}$ is a diagonal matrix with eigenvalues λ_i in the decreasing order on the diagonal. The eigenvectors are called *principal axes* or *principal directions* of the data. Projections of the data on the principal axes are called *principal components*, also known as *PC scores*; these can be seen as new, transformed, variables. The j-th principal component is given by j-th column of $\mathbf{XV}$. The coordinates of the i-th data point in the new PC space are given by the i-th row of $\mathbf{XV}$.

If we now perform singular value decomposition of $\mathbf{X}$, we obtain a decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top,$$

where $\mathbf{U}$ is a unitary matrix (with columns called left singular vectors), $\mathbf{S}$ is the diagonal matrix of singular values s_i and $\mathbf{V}$ columns are called right singular vectors. From here one can easily see that

$$\mathbf{C} = \mathbf{V}\mathbf{S}\mathbf{U}^\top\mathbf{U}\mathbf{S}\mathbf{V}^\top/(n-1) = \mathbf{V}\frac{\mathbf{S}^2}{n-1}\mathbf{V}^\top,$$

meaning that right singular vectors $\mathbf{V}$ are principal directions (eigenvectors) and that singular values are related to the eigenvalues of covariance matrix via $\lambda_i = s_i^2/(n-1)$. Principal components are given by $\mathbf{XV} = \mathbf{U}\mathbf{S}\mathbf{V}^\top\mathbf{V} = \mathbf{US}$.

https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues/140579#140579

To summarize:

1. If $\mathbf{X} = \mathbf{USV}^\top$, then the columns of $\mathbf{V}$ are principal directions/axes (eigenvectors).
2. Columns of $\mathbf{US}$ are principal components ("scores").
3. Singular values are related to the eigenvalues of covariance matrix via $\lambda_i = s_i^2/(n-1)$. Eigenvalues λ_i show variances of the respective PCs.
4. Standardized scores are given by columns of $\sqrt{n-1}\mathbf{U}$ and loadings are given by columns of $\mathbf{VS}/\sqrt{n-1}$. See e.g. here and here for why "loadings" should not be confused with principal directions.
5. **The above is correct only if $\mathbf{X}$ is centered.** Only then is covariance matrix equal to $\mathbf{X}^\top\mathbf{X}/(n-1)$.
6. The above is correct only for $\mathbf{X}$ having samples in rows and variables in columns. If variables are in rows and samples in columns, then $\mathbf{U}$ and $\mathbf{V}$ exchange interpretations.
7. If one wants to perform PCA on a correlation matrix (instead of a covariance matrix), then columns of $\mathbf{X}$ should not only be centered, but standardized as well, i.e. divided by their standard deviations.
8. To reduce the dimensionality of the data from p to $k < p$, select k first columns of $\mathbf{U}$, and $k \times k$ upper-left part of $\mathbf{S}$. Their product $\mathbf{U}_k\mathbf{S}_k$ is the required $n \times k$ matrix containing first k PCs.
9. Further multiplying the first k PCs by the corresponding principal axes $\mathbf{V}_k^\top$ yields $\mathbf{X}_k = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^\top$ matrix that has the original $n \times p$ size but is *of lower rank* (of rank k). This matrix $\mathbf{X}_k$ provides a *reconstruction* of the original data from the first k PCs. It has the lowest possible reconstruction error, see my answer here.
10. Strictly speaking, $\mathbf{U}$ is of $n \times n$ size and $\mathbf{V}$ is of $p \times p$ size. However, if $n > p$ then the last $n - p$ columns of $\mathbf{U}$ are arbitrary (and corresponding rows of $\mathbf{S}$ are constant zero); one should therefore use an *economy size* (or *thin*) SVD that returns $\mathbf{U}$ of $n \times p$ size, dropping the useless columns. For large $n \gg p$ the matrix $\mathbf{U}$ would otherwise be unnecessarily huge. The same applies for an opposite situation of $n \ll p$.

t-sne

Saturday 11 May 2024 11:36 PM

t-SNE (t-distributed Stochastic Neighbor Embedding) is an unsupervised non-linear dimensionality reduction technique for data exploration and visualizing high-dimensional data.

Neighborhood preserving embedding

→ t-SNE aims to preserve local structure while PCA aims to preserve the global structure of the data.
→ neighborhood relationships gaurenteed, relationship of points outside neighbourhood not gaurenteed.

→ We can make t-sne preserve the global structure by tuning the 'perplexity' parameter

PCA → vizualization
PCA → prediction/feature extraction

t-sne → Visualization NOT FOR prediction/ feature extraction

→ Complicated Math formulation [LATER]

→ Crowding Problem: Sometimes its impossible to preserve distances in all the neighbourhoods.

→ t-distribution used to resolve this problem. [Try it's best]

→ Role of t-distribution → Complicated

→ t-sne is an iterative algorithm.

↓ perplexity — ↑ perplexity
local — global
balanced

** → Always run t-sne algo with multiple perplexity values **

→ # of neighbors whose distances you wish to perserve

→ Run atleast for 1000 iterations to reach stable state

→ On the same dataset with the same parameter settings, if run multiple times output may differ.

t-sne is not a deterministic algorithm [stochastic/Probabilistic]

→ t-sne tries to expand dense cluster & shrink sparse clusters s.t. the densities of groups of points are roughly similar.

WE CANNOT READ CLUSTER SIZES FROM t-sne

→ Distances b/w clusters might not mean anything [inter-cluster distances]

→ Don't fall in the trap of trying to make sense of random/noisy/junk data.

→ t-sne tries to spread points evenly

→ Always re-run t-sne with p & steps-size & see if shape is stable

* standardize features

→ Neighbourhoods of all points should be preserved if t-sne is working perfectly.

[Learn Later]

Saturday 11 May 2024 11:37 PM

- → Linear Discriminant Analysis *
- → UMAP *
- → MDS
- → Sammon mapping
- → Graph-based tech