Linear Algebra for Deep Learning

April 11, 2023

Lumi
12112618@mail.sustech.edu.cn

Symmetric, Orthogonal, and Unitary Matrices

If for a square matrix, $\boldsymbol{A}$, we have

$$\boldsymbol{A}^\top = \boldsymbol{A}$$

then $\boldsymbol{A}$ is said to be a **symmetric matrix**.

If the following is true,

$$\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{I}$$

then $\boldsymbol{A}$ is an **orthogonal matrix**, and $\boldsymbol{A}^{-1} = \boldsymbol{A}^\top$.

And as a result, $\det(\boldsymbol{A}) = \pm 1$.

If we count complex matrices, then if

$$\boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{I}$$

we say that $\boldsymbol{U}$ is a **unitary matrix** with $\boldsymbol{U}^*$ being the conjucate transpose of $\boldsymbol{U}$.

Eigenvalues and Eigenvectors

Suppose the eigenvalue and eigenvector of matrix $\boldsymbol{A}$ is $\boldsymbol{v}$ and λ, the property

$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$$

holds.

The code in Python `np.linalg.eig(a)` returns two values. `np.linalg.eig(a)[0]` are the eigen values, and `np.linalg.eig(a)[1]` are the eigen vectors for the corresponding eigenvalues.

```
print(a)
# [[ 0  1]
  [-2 -3]]

print(np.linalg.eig(a)[0])
# [-1. -2.]

print(np.linalg.eig(a)[1])
# [[ 0.70710678 -0.4472136 ]
 [-0.70710678  0.89442719]]
```

Vector Norms and Distance Matrix

For an n-dimensional vector, $\boldsymbol{x}$, we define the p-norm of the vector to be

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

where p is a real number.

The *L2-norm*,

$$||x||_2 = \sqrt{x_0^2 + x_1^2 + \ldots + x_{n-1}^2} = \sqrt{\boldsymbol{x}^\top\boldsymbol{x}}$$

and the *L1-norm*

$$||x||_1 = \sum_i |x_i|$$

are the most frequently used norms in deep learning.

A funny thing is that the L_∞-*norm* finds the maximum absolute value for all the components of $\boldsymbol{x}$.

Switching from norm to distance makes a change to the equations,

$$\mathrm{L}_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_i \left|x_i - y_i\right|^p\right)^{\frac{1}{p}}$$

this is called the *Euclidean distance* between two vectors. The L1-distance is often called the *Manhattan distance* ($\mathrm{L}_1 = \sum_i \left|x_i - y_i\right|$).

Covariance Matrix

The covariance matrix $\boldsymbol{\Sigma}$ discribes how two columns of data vary together, that is to say

$$\Sigma_{ij} = \frac{1}{n-1}\sum_{k=0}^{n-1}(x_{ki} - \overline{x}_i)(x_{kj} - \overline{x}_j)$$

Numpy function for calculating the covariance matrix is `np.cov(X, rowvar = False)`. We also need to set `rowvar = False` because our data for individual group are stored in the columns.

Mahalanobis Distance

With the mean value vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$, with these values, we can define a distance metric called the *Mahalanobis distance*.

$$D_M = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$$

We can use this *Mahalanobis distance* as a simple classifier, called the nearest centroid classifier. D_M is more accurate than the L1-distance in classifiers.

Kullback-Leiber Divergence

The *Kullback-Leiber divergence (KL-divergence)*, or *relative entropy*, is a measure of the similarity between two probability distributions: the lower the value, the more similar the distributions.

If P and Q are discrete probabitlity distributions, the KL-divergence is

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_x P(x) \log_2 \left(\frac{P(x)}{Q(x)} \right)$$

The KL-divergence isn't a distance metrix in mathematical sense because the symmetry property doesn't hold, $D_{\mathrm{KL}}(P \| Q) \neq D_{\mathrm{KL}}(Q \| P)$.

Principle Component Analysis

Principle Component Analysis (PCA) is the technique to learn the directions of the scatter in the dataset, starting with the direction aligned along the greatest scatter. The PCA algorithm generally involves the following steps:

1. Find the mean center of the data.
2. Calculate the covariance matrix, $\boldsymbol{\Sigma}$, of the mean-centered data.
3. Calculate the eigenvalues and the eigenvectors of $\boldsymbol{\Sigma}$.
4. Sort the eigenvalues by decreasing absolute value.
5. Discard the weaker eigenvalues and eigenvectors.
6. Generate the new transformed values from the existing dataset, $\boldsymbol{x}' = \boldsymbol{W}\boldsymbol{x}$.

PCA are often used in machine learning to reduce the model size, thus usually enhancing the results of the deep learning.

Singlar Value Decomposition and Pseudoinverse

Singular value decomposition (SVD) is a power technique to transform any matrix into the product of three matrices. For example, for an input matrix $\boldsymbol{A}$, with real elements and shape $m \times n$, where m might not be equal to n. Then the SVD for $\boldsymbol{A}$ is

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$$

$\boldsymbol{A}$ have been decomposed into three matrices. A $m \times m$ orthogonal matrix $\boldsymbol{U}$, a $m \times n$ "diagonal" matrix $\boldsymbol{\Sigma}$ and a $n \times n$ orthogonal matrix $\boldsymbol{V}$.

The "sigular" comes from the fact that the diagonal elements of the "diagonal" matrix $\boldsymbol{\Sigma}$, are singular values, the square roots of the positive eigenvalues of the matrix $\boldsymbol{A}^\top \boldsymbol{A}$.

In SciPy, the `svd` function returns three values, $\boldsymbol{U}, \boldsymbol{\Sigma}$ and $\boldsymbol{V}^\top$.

We can use the SVD for the PCA, or calculating the pseudoinverse of a matrix.