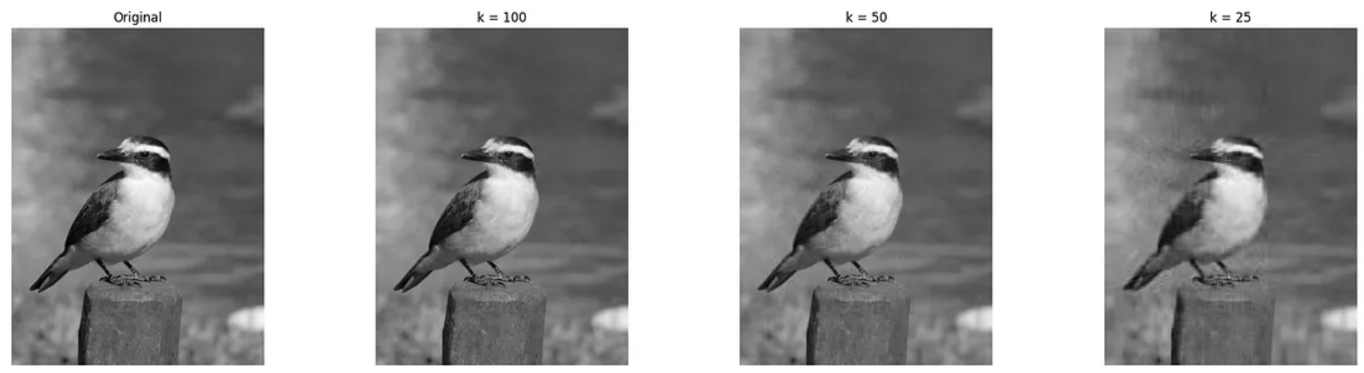
Member-only story

Singular Value Decomposition (SVD) in Data Science

Linear Algebra for Data Science Series (8): Singular Value Decomposition

Sunghyun Ahn Follow 10 min read · Dec 6, 2024

178 2



SVD-based compression of an image of a bird. k indicates the number of retained singular values

> If you are not a paid member on Medium, I make my stories available for free: Friends link

Singular Value Decomposition (SVD) is a powerful mathematical technique used in linear algebra to factorize a matrix into three simpler matrices. It's widely used in data science, machine learning, and signal processing for dimensionality reduction, noise reduction, and feature extraction.

Remember how the Eigenvalue Decomposition had a severe restriction: it only works for a square matrix. It's unlucky because in a lot of real-life applications, square matrices are pretty rare to come-by.

Eigenvectors and Eigenvalues for Data Science

Linear Algebra for Data Science Interview Series (7): Eigenvectors and Eigenvalues

medium.com

To circumvent this, there is a more general decomposition that works for any `m x n` matrix. I'm sure you guys may have heard of it, but it's called the **Singular Value Decomposition (SVD).** Here's how it works.

Table of Contents

1. Introduction to SVD

Start with an `m x n` ***matrix A*** with rank ***r***. For a singular value decomposition, our goal is still to diagonalize A as we do in Eigenvalue Decomposition:

$$A = S\Lambda S^{-1}$$

Eigenvalue Decomposition, Λ is the diagonal matrix of all eigenvalues

However, because `m x n` isn't square matrix, we need two sets of "eigenvectors" to handle the two different dimensions ***(m and n)***. We will call these more general "singular vectors" called :

- U (Left Singular Vectors, represents the dimension "***m***")
- V (Right Singular Vectors, represents the dimension "***n***")

Also, instead of eigenvalues (remember that these were scalar values), we will have "singular values" called:

- Σ (Singular Values)

With these singular vectors and singular values, we can turn the Eigenvalue Decomposition equation into a Singular Value Decomposition equation:

$$A = U\Sigma V^T$$

Singular Value Decomposition

But why can we do this you may wonder… That's exactly what I'm here to explain! Let's look why we can write it in this form by looking at the how the matrices U, V, and Σ are formed.

1.1. Matrices in SVD

Remember in the Eigenvalue Decomposition, we constructed a matrix called "S", whose columns where the "n" independent eigenvectors of A.

$$S = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix}$$

Where $\mathbf{x}_n$ are the **eigenvectors of A**

Just like how we did for the Eigenvalue Decomposition, these singular vectors can be compiled in to a matrix form.

$$V = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_n \end{bmatrix}$$

Represents the dimension **"n"**

$$U = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_m \end{bmatrix}$$

Represents the dimension **"m"**

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Dimension **“m x n”**

Okay, so it’s obvious that we can write each vectors as a matrix form. For those that still have a hard time understanding, the “n” independent vectors of U (or V) are becoming the columns of the matrices. The values inside these independent vectors are in the rows of hte matrices.

Alright, that’s cleared up… but… why can we put this into the SVD equation of A=UΣV^T? How does it relate to eigenvectors?

For this, we have to go back to the Eigenvector Equation.

2. Rewriting the Eigenvector Equation Ax = λx

In our previous article, remember how we used the Eigenvector equation to create the Eigenvalue Decomposition equation:

$$A\vec{x} = \lambda\vec{x}$$

Eigenvector equation

Since the Eigenvalue Decomposition only worked for a square matrix, we didn’t really need to care about dimensions. (A was an `n x n` matrix, S was an `n x n` matrix, Λ was a `n x n` matrix)

$$AS = S\Lambda$$

S: A matrix of eigenvectors — Λ: A diagonal matrix of eigenvalues.

However, for our current case, we are dealing with a `m x n` matrix. The dimensions are completely different! Regardless, we can still write our singular value decomposition equation into the same form using our U, V, and Σ.

$$AV = U\Sigma$$

U, V are matrices of singular vectors, Σ is a matrix of singular values

From only looking at it, you might say… Huh..? How can we do that? Are the dimensions okay?

Yes! The left and right side of the equation will have equal dimension because

- Matrix A: `m x n` matrix
- Matrix V: `n x n` matrix (input matrix)
- Matrix U: `m x m` matrix (output matrix)
- Matrix Σ: `m x n` matrix

Thus, the dot product between A·V and U·Σ both result in a `m x n` matrix! This is why we can write the SVD equation like the Eigenvalue Decomposition equation!

$$A = U\Sigma V^T$$

Singular Value Decomposition (SVD)

$$A = S\Lambda S^{-1}$$

Eigenvalue Decomposition

3. Look into SVD

Before we move on, it might be a good idea to look at how what the matrices in SVD represents and also why V is tranposed.

This is a good place to introduce a key concept in linear algebra (as we talk about these vectors and values): **In linear algebra, the *rows* of a matrix A**

correspond to observations or data points (outputs), while the *columns* **correspond to features (inputs).**

Let's look at a matrix A that has dimension of `3 x 2` like so:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

3 × 2 Matrix

3.1. U

Remember that U is an `m x m` orthogonal matrix, meaning our matrix U is created using *m*=3 (number of rows of A). The columns of U are the **left singular vectors (u_1, u_2, u_3).**

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix}$$

These are derived from eigenvectors of AA^T (a 3×3 matrix)

The left singular vectors (U) span the space of the rows of A, often referred to as the **output space (feature space).**

What we are essentially doing by creating a matrix U that are full of the left singular vectors is.... We are ***mapping these singular vectors into an output space*** that we can obtain!

3.2. V^T

Remember that V is an `n x n` orthogonal matrix, meaning our matrix V is created using *n*=2 (number of columns of A). The columns of V **are the right singular vectors (v_1, v_2).**

$$V = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$$

These are derived from eigenvectors of A^TA (a 2×2 matrix)

The left singular vectors (V) span the space of the columns of A, often referred to as the **input space.**

Same as matrix U, when we create a matrix V that are full of the right singular vectors, we are ***mapping these singular vectors into an input space*** that we can obtain.

> Okay take a pause to reflect that because…

The above statement is exactly the reason why we need the matrix V to be transposed when solving for SVD. It's because SVD works on these steps:

1. **V^T:** Maps input space to singular value space. ***(Input** -> **Singular)***
2. **Σ:** Scales components along the principal directions.
3. **U:** Maps from singular value space to output space. ***(Singular** -> **Output)***

If V were not transposed, the multiplication order and alignment between input/output spaces would break.

It's easy to think if you go through it logically. We would want to give our inputs, scale them along some direction, then create an output from the scaled values!

3.3. Σ

Remember that Σ is a diagonal matrix of size `m x n` (3×2) that contains the singular values (σ_1, σ_2) of A, with all other entries zero.

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix}$$

Here, $\sigma_1 > \sigma_2$, and they are the square roots of the eigenvalues.

Represents the "strength" of the transformation in each direction. Larger singular values indicate stronger contributions in those directions.

I think the most important thing to remember about this diagonal matric is that… The singular values (σ_i) in Σ are ordered from largest to smallest. These values represent the importance (or weight) of the corresponding singular vectors in capturing the structure of the matrix A.

Still a bit confused on these singular values? It's okay! I'll first explain how they are calculated and talk about why these are so important.

3.4 How Singular Values Are Computed

Singular values are derived mathematically from the matrix A. They are related to the eigenvalues of A^T·A (or A·A^T, depending on the dimensions of A).

- For a given matrix A, compute A^T·A. This produces a symmetric $n \times n$ matrix.
- A^T·A is guaranteed to have non-negative eigenvalues.

We can find the eigenvalues of A^T·A (λ_1, λ_2, ... λ_n) by using the equation that we know already:

$$\det(A^T A - \lambda I) = 0$$

Remember that this is derived from the Eigenvector Equation

Also, it's important to know that the singular values are the square roots of the eigenvalues.

$$\sigma_i = \sqrt{\lambda_i}$$

How Singular Values relate to Eigenvalues

Now once we get the singular values, we can order them from largest to smallest to get our Σ!

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$$

Sort largest to smallest

You may wonder what does the singular values and the sorting these values exactly do? Well... If we think of the matrix A as informations of image, the first layer (corresponding to σ_1) contains the most dominant structure (e.g., the overall shape). The subsequent layers (σ_n) add finer details.

You'll most likely be using a programming language to calculate U, V, and Σ so I won't go to much more in depth. Just know that these algorithms automatically order them in descending order, making it straightforward to select the largest ones!

3.4 Important Summary

Since we know now how these singular vectors and values play a role, I'll go over again on how SVD works.

$$A = U\Sigma V^T$$

Singular Value Decomposition

1. **V^T:** Maps input space to singular value space. ***(Input** -> **Singular)***
2. **Σ:** Scales components along the principal directions.
3. **U:** Maps from singular value space to output space. ***(Singular** -> **Output)***

One more thing. If we think about it, we can expand this form into ***a summation form.*** This looks at each individual singular vectors that matches up as well as the singular values.

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T$$

Summation form of SVD

This is super important for us to talk about why SVD is so important.

> If you made it this far into my article, I would appreciate it if you could leave a few claps! 😊 It really motivates me to write more educational content!

4. Why Is This Important?

SVD is extremely useful for understanding the structure of the matrix A, as it allows you to approximate A by retaining only the top k singular values and their corresponding vectors. For instance:

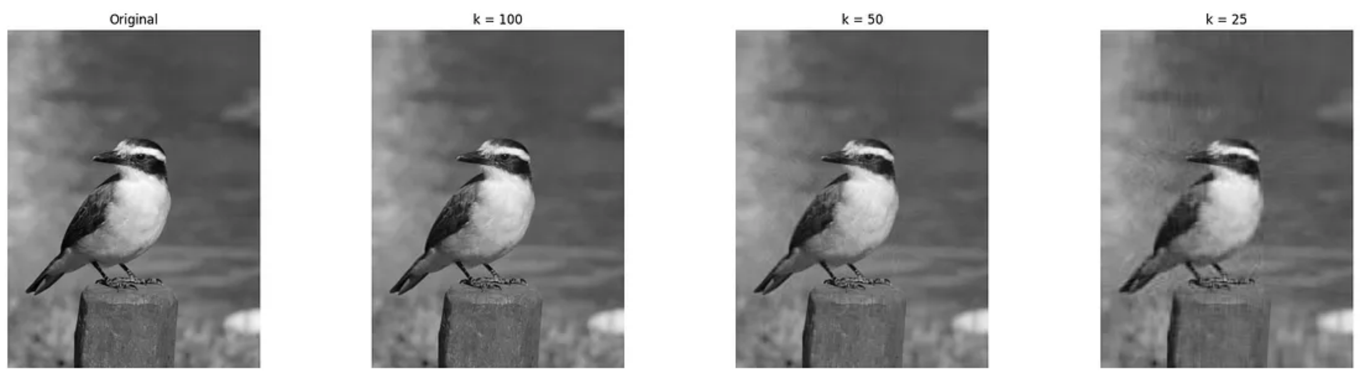
$$A_k = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T$$

In SVD, these terms (σ_i, u_i, v_i) are sorted in **largest to smallest importance** (as determined by the singular values σ_i). The largest singular value σ_1 corresponds to the "most significant" component of A, and so on.

This means that the first few terms (σ_i, u_i, v_i) are capturing the most important features in Matrix A while discarding noise. So, when we calculate for important features, we really only need to care about the first few terms in an SVD.

4.1. Let's see why this is so useful.

Say you are working on a computer vision problem involving very high resolution images of birds.



SVD-based compression of an image of a bird. k indicates the number of retained singular values

These images are stored as matrices of grayscale pixel values, with size their original size. This full-size image has rank based on the largest column or row pixel value, but it can be approximated as a matrix with a lower rank *k*.

This `k` represents the number of retained singular values when recreating the output image.

Like I mentioned before, the first layer (corresponding to σ_1) contains the most dominant structure (e.g., the overall shape). The subsequent layers (σ_n) add finer details.

As you can see, the fewer singular values (lower the rank), the more blurry and pixelated the image becomes (you are losing the finer details). However, even with the significant memory savings of k = 100 or k = 50, the image is still easy to see.

```python
from PIL import Image
import numpy as np
from scipy.linalg import svd
import matplotlib.pyplot as plt

# Load and convert the image to grayscale
full_image = Image.open("bird.jpg").convert("L")
full_image_array = np.array(full_image)

# Perform SVD
U, S, Vt = svd(full_image_array, full_matrices=False)
S = np.diag(S)

# Function to reconstruct the image with k singular values
def reconstruct_image(U, S, Vt, k):
    return np.dot(U[:, :k], np.dot(S[:k, :k], Vt[:k, :]))

# Reconstruct the images with different values of k (100, 50, 25)
k_values = [100, 50, 25]
reconstructed_images = [reconstruct_image(U, S, Vt, k) for k in k_values]

fig, axes = plt.subplots(1, 4, figsize=(20, 5))
axes[0].imshow(full_image_array, cmap="gray")
axes[0].set_title("Original")
axes[0].axis("off")

for i, (k, image) in enumerate(zip(k_values, reconstructed_images)):
    axes[i + 1].imshow(image, cmap="gray")
    axes[i + 1].set_title(f"k = {k}")
    axes[i + 1].axis("off")

plt.tight_layout()
plt.show()
```

This is why SVD is so important!

Conclusion

With the final example, you may be able to see why Singular Value Decomposition (SVD) plays such a crucial role in modern data science and machine learning.

By breaking a matrix into its singular vectors and singular values, SVD allows us to simplify the complex relationships while preserving the most important features.

From dimensionality reduction to noise suppression and image compression, SVD lets us to work efficiently with large datasets while maintaining meaningful insights. The ability to approximate matrices using only a subset of singular values highlights its practicality in real-world scenarios like recommendation systems, natural language processing, and computer vision.

As demonstrated with the bird image example, SVD is not only theoretically significant but also highly practical. By retaining only the top k singular values, we achieve significant memory savings while preserving most of the image's visual details — a powerful demonstration of SVD's ability to compress information without losing its essence.

Most importantly though… I hope you were able to learn something from this article in the end!

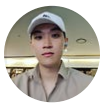
Connect with me!

If you made it this far, I assume you are an aspiring data scientist, a teacher in the data science field, a professional looking to hone your craft, or just an avid learner in a different field!

Linkedin, Instagram

I would love to have a chat with you on anything!

Data Data Science AI Machine Learning Mathematics



Written by Sunghyun Ahn

812 followers · 7 following

Follow

ex-Analyst @ Chemtopia | Graduate Research Assistant @ Seattle University |

Responses (2)

Jagannath Rao

What are your thoughts?

Syed M Aqil Burney
May 11

Nice writing and no one can take SVD lightly as most of the search algorithms dependent on linear algebra of this type are used for internet search like google search algorithm and in multivariate statistics as well. while it is feasible to work... more

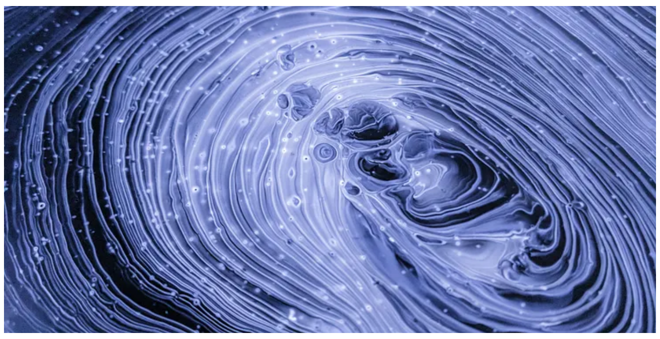
Reply

Kbdub
Jan 19

very nice

Reply

More from Sunghyun Ahn



In Data Science Collective by Sunghyun Ahn

Transfer Learning: Why We Freeze and Unfreeze Model Layers

Choosing the Right Layers to Freeze When Fine-tuning | Data Science for Machine...

Apr 29 138 3

In Data Science Collective by Sunghyun Ahn

Why Backpropagation Is So Important For Models In Machine...

How gradients drive learning in neural networks | Data Science for Machine Learnin...

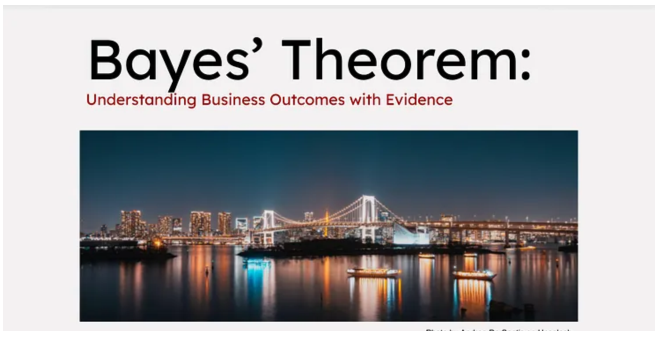
May 3 108 3

DSC In Data Science Collective by Sunghyun Ahn

How Optimizers and Loss Functions Work in Machine...

What those two lines in your model.compile() are really doing behind the scenes | Data...

May 3 · 76 · 3



In TDS Archive by Sunghyun Ahn

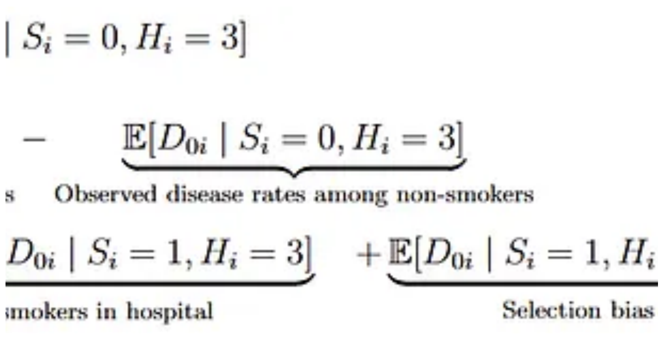
Bayes' Theorem: Understanding Business Outcomes with Evidence

A practical introduction to Bayes' Theorem: Probability for Data Science Series (2)

Dec 12, 2024 · 1.4K · 20

See all from Sunghyun Ahn

Recommended from Medium



$| S_i = 0, H_i = 3]$

$- \underbrace{\mathbb{E}[D_{0i} \mid S_i = 0, H_i = 3]}_{\text{Observed disease rates among non-smokers}}$

$\underbrace{D_{0i} \mid S_i = 1, H_i = 3]}_{\text{mokers in hospital}} + \underbrace{\mathbb{E}[D_{0i} \mid S_i = 1, H_i}_{\text{Selection bias}}$

Xwang

Why More Control Can Hurt: Understanding Bad Controls...

Whether evaluating a new product or a regulatory change, getting the causal...

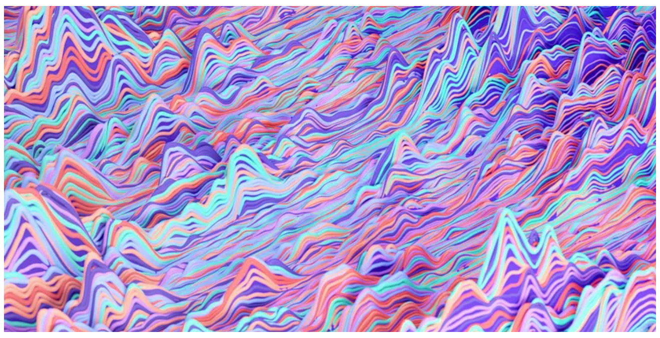
May 3 · 6

DSC In Data Science Collec... by Herman Jangsett Mos...

Loss Functions and Their Origins

A look at how we derive loss functions in machine learning and how they relate to...

Oct 1 · 303 · 4

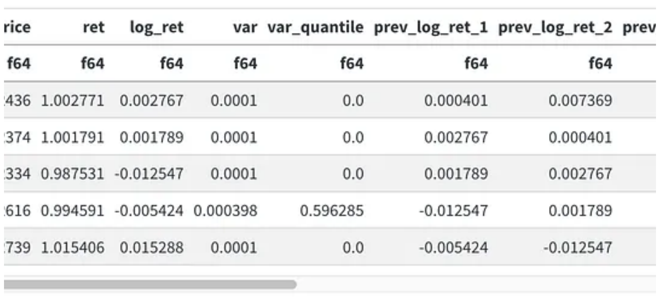


Data Overload

The Kolmogorov-Smirnov Test

In statistics, evaluating whether a dataset follows a specific distribution is a...

May 24 13



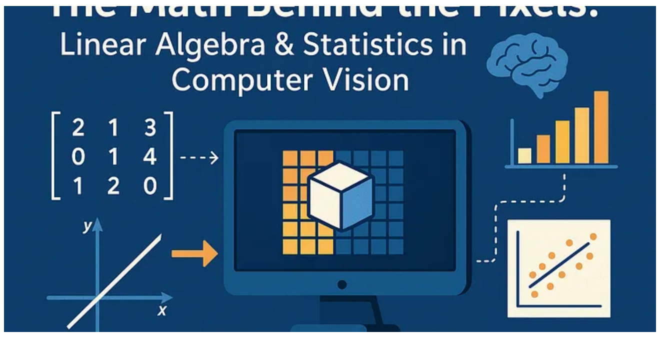
rice	ret	log_ret	var	var_quantile	prev_log_ret_1	prev_log_ret_2	prev
f64	f64	f64	f64	f64	f64	f64	
436	1.002771	0.002767	0.0001	0.0	0.000401	0.007369	
374	1.001791	0.001789	0.0001	0.0	0.002767	0.000401	
334	0.987531	-0.012547	0.0001	0.0	0.001789	0.002767	
616	0.994591	-0.005424	0.000398	0.596285	-0.012547	0.001789	
739	1.015406	0.015288	0.0001	0.0	-0.005424	-0.012547	

Konstantin Sofeikov

Modeling Heating Oil Prices With Bayesian Inference and Generativ...

Read the full version and checkout the code.

May 13

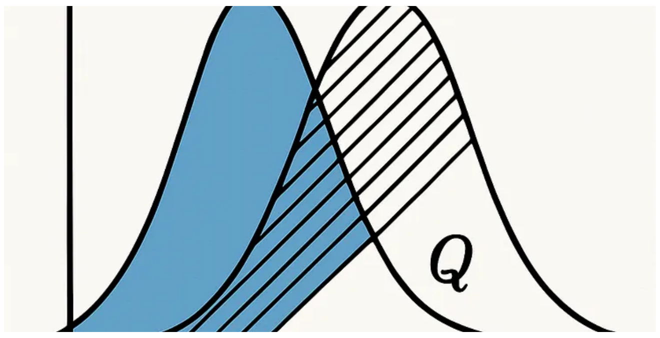

The Math Behind the Pixels:
Linear Algebra & Statistics in Computer Vision
2 1 3
0 1 4
1 2 0
y
x

In Python in Plain En... by Muhammad Syaoki Far...

The Math Behind the Pixels: Linear Algebra & Statistics in Computer...

At its core, computer vision is an applied field of AI, and its language is mathematics. To...

Aug 12 1



In T3CH by Francesco Franco

Kullback-Leibler Divergence with Keras

When you train a supervised machine learning model, you feed forward data and...

May 2 231 2

See more recommendations