

Finally! A Clear Derivation of the VAE KL Loss



Jae H. Park · [Subscribe](#)

9 min read · Mar 27, 2025



107



1



While analyzing the loss function of the variational autoencoder (VAE) model, I noticed that the **Kullback-Leibler (KL) loss term has a nice closed form solution**, but I could not wrap my head around how to derive it from the VAE loss setup. So, I decided to take this challenge to understand the loss function better.

First, let me summarize the VAE training and loss setup before delving into the derivation of the closed form solution. If you are already familiar with it, you can skip this section and continue to the derivation.

Unlike most autoencoders, which basically compress (or encode) data into latent variables like a fixed discrete value to be reconstructed to the original data, **variational** autoencoders (VAEs) encode latent variables like a continuous range of possibilities in the form of a probability distribution. In this context, you can think of autoencoders as a compression method whereas variational autoencoders as a new data generator.

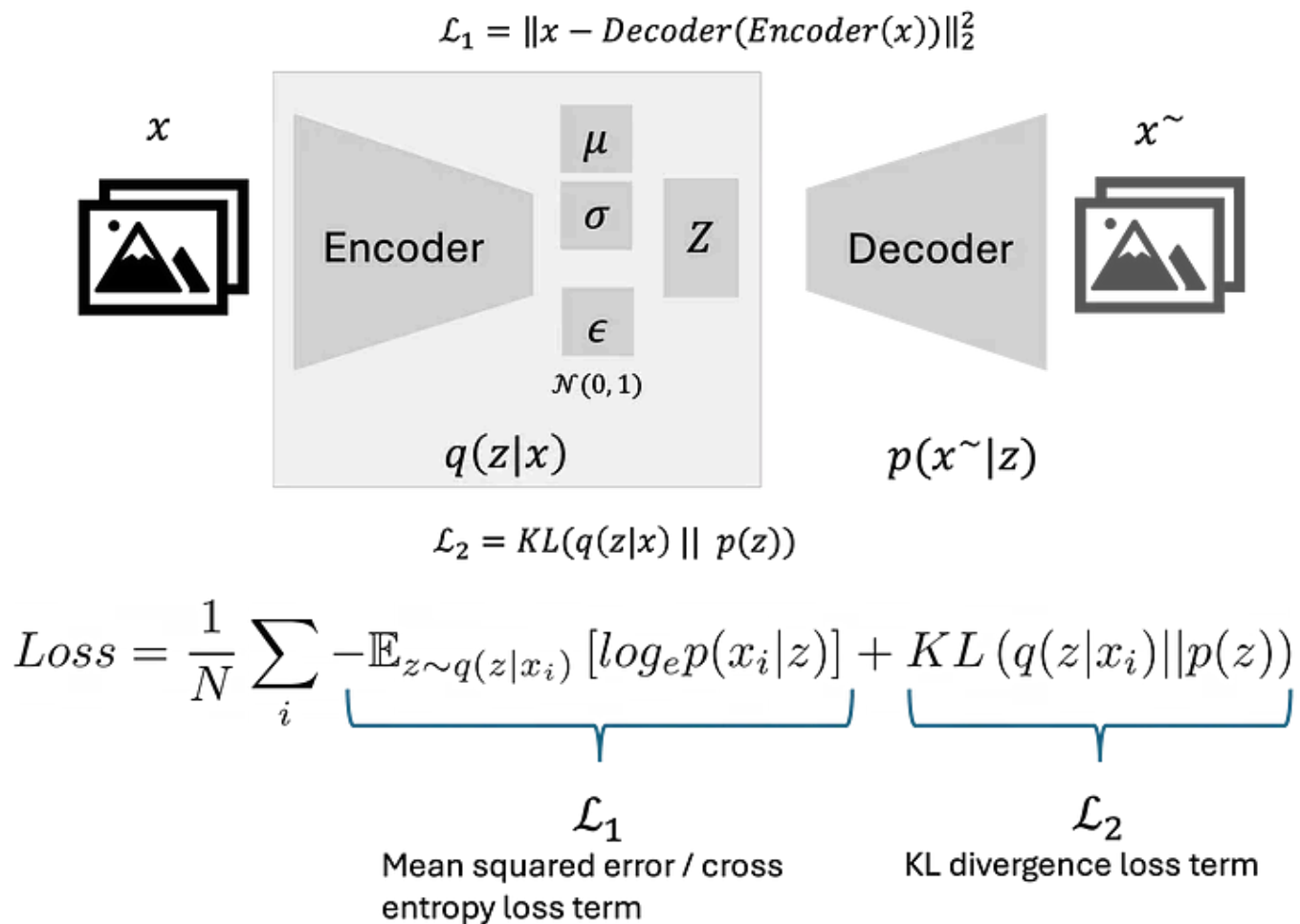


Figure 1: VAE Architecture and Loss function

In Figure 1, the encoder, $q(z|x)$ represents a multi-layer neural network (e.g., linear and convolutional layers) that progressively compresses training data and outputs two vectors, μ and σ , which represent the range of possibilities in the form of the probability distribution, in this case, a normal distribution (Gaussian). To generate new data, the decoder, $p(x^{\sim}|z)$ samples from anywhere in the latent space (Z) between original data points and through multiple layers of the decoder neural network, it synthesizes new data that are unique and original but somewhat resemble the original training data.

During training, VAE needs to minimize the difference (loss) between the original data and the generated version of that data by the decoder, which is

represented as a L_1 loss. This loss can be either a cross entropy or a mean squared error (MSE), but in Figure 1, the L_1 loss is represented as a MSE. However, the goal of VAE is not to reconstruct the original data but to generate new sample that resemble the original data. The reconstruction error alone is not sufficient in VAE because it can overfit the training data and does not generalize well to new samples. Therefore, the KL divergence regularization or KL loss term (L_2) comes into play. It minimizes the KL divergence between the learned distribution of latent variables, q and a simple normal distribution, p to force the learned latent variables to follow a normal distribution.

That was quite an involved summary of the VAE model. **To reiterate, my goal is to derive the closed form solution for the KL divergence term, L_2 as shown in Figure 2.**

$$L_2 = KL(q(z|x_i)||p(z)) = KL(N(\mu, \Sigma)||N(0, 1))$$

$$= -\frac{1}{2} \sum_j (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

Figure 2: A closed form solution of KL divergence loss term

Derivation

First, let me derive the loss function. I am going to start with the log likelihood of data points:

$$\frac{1}{N} \sum_i \log_e p(x_i)$$

which is equal to:

$$= \frac{1}{N} \sum_i \log_e \int_z p(x_i|z)p(z)dz$$

because I am basically summing up the joint probability of X and Z. This is called marginalization. For details, you can check my previous article explaining [the Bayes's theorem and basic probability concepts](#). Keep in mind that the integral symbol is equivalent to summation within a continuous domain.

Then, I am going to introduce a distribution, q_i as shown below. I can do this since I am multiplying and dividing the expression by q_i . It does not change anything.

$$= \frac{1}{N} \sum_i \log_e \int_z p(x_i|z)p(z) \frac{q_i(z|x_i)}{q_i(z|x_i)} dz$$

Given the expectation formula,

$$\mathbb{E}_{x \sim p(x)}[x] = \int p(x)x dx$$

I can reformulate it as an expectation, placing the expectation outside the log because of Jensen's inequality, $\log \mathbb{E}[x] \geq \mathbb{E}[\log(x)]$. Since log is a concave function, Jensen's inequality ensures that

$$= \frac{1}{N} \sum_i \log_e \mathbb{E}_{z \sim q_i(z|x_i)} \left[\frac{p(x_i|z)p(z)}{q_i(z|x_i)} \right]$$

$$\geq \frac{1}{N} \sum_i \mathbb{E}_{z \sim q_i(z|x_i)} \left[\log_e \frac{p(x_i|z)p(z)}{q_i(z|x_i)} \right]$$

This approach, commonly known as the evidence lower bound (**ELBO**), is valid because maximizing this objective inherently leads to the maximization of the original objective which is the log likelihood.

Let me perform algebraic manipulation to arrive three expectation terms.

$$\begin{aligned} &= \frac{1}{N} \sum_i \mathbb{E}_{z \sim q_i(z|x_i)} [\log_e p(x_i|z) + \log_e p(z) - \log_e q_i(z|x_i)] \\ &= \frac{1}{N} \sum_i \mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(x_i|z) + \mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(z) - \mathbb{E}_{z \sim q_i(z|x_i)} \log_e q_i(z|x_i) \end{aligned}$$

If you look carefully, you can see both entropy and cross entropy terms:

$$-\mathbb{E}_{z \sim q_i(z|x_i)} \log_e q_i(z|x_i)$$

Entropy

$$-\mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(z)$$

Cross Entropy

The KL divergence (aka, relative entropy) is, in fact, the difference between cross entropy and entropy.

$$= \frac{1}{N} \sum_i \mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(x_i|z) - [(-\mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(z)) - (-\mathbb{E}_{z \sim q_i(z|x_i)} \log_e q_i(z|x_i))]$$

$$\begin{aligned}
&= \frac{1}{N} \sum_i \mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(x_i|z) - [H(q_i(z|x_i), p(z)) - H(q_i(z|x_i))] \\
&= \frac{1}{N} \sum_i \mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(x_i|z) - KL(q_i(z|x_i) || p(z))
\end{aligned}$$

Starting with the log likelihood formulation, I derived the expression above. Given that **the negative log likelihood** is commonly used as a **loss function**, **negating** this expression leads to the **final loss form**.

$$Loss = \frac{1}{N} \sum_i -\mathbb{E}_{z \sim q_i(z|x_i)} \log_e p(x_i|z) + KL(q_i(z|x_i) || p(z))$$

However, this formulation has a problem: the computational burden of fitting q_i separately for each data point makes it impractical. To overcome this, people use *amortized* variational inference which employs a neural network (encoder) rather than doing the inference in every data point, individually. The encoder network is represented as q , and the final loss function is:

$$Loss = \frac{1}{N} \sum_i -\mathbb{E}_{z \sim q(z|x_i)} \log_e p(x_i|z) + KL(q(z|x_i) || p(z))$$

Next, I am going to derive the closed form solution for the KL divergence term. Remember that both q and p are normal distributions where q is a learned distribution of latent variables and p is a simple normal distribution,

$$q(z) \sim \mathcal{N}(\mu_1, \Sigma_1)$$

$$p(z) \sim \mathcal{N}(\mu_2, \Sigma_2) = \mathcal{N}(0, \mathcal{I})$$

and the functional form of the multivariate normal distribution is presented below.

$$f(x) = \frac{1}{\sqrt{(2\pi)^k}} \frac{1}{\sqrt{|\Sigma|}} e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)}$$

The KL divergence can also be expressed as below.

$$KL(q(z|x_i)||p(z)) = \mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \frac{q(z|x_i)}{p(z)} \right]$$

Let's replace q and p with their respective functional forms,

$$KL(q(z|x_i)||p(z)) = \mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \frac{\frac{1}{\sqrt{(2\pi)^k}} \frac{1}{\sqrt{|\Sigma_1|}} e^{\left(-\frac{1}{2}(z-\mu_1)^T \Sigma_1^{-1} (z-\mu_1)\right)}}{\frac{1}{\sqrt{(2\pi)^k}} \frac{1}{\sqrt{|\Sigma_2|}} e^{\left(-\frac{1}{2}(z-\mu_2)^T \Sigma_2^{-1} (z-\mu_2)\right)}} \right]$$

and perform algebraic manipulation to get a simplified expression.

$$= \mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right)^{\frac{1}{2}} + \log_e e^{\left(-\frac{1}{2}(z-\mu_1)^T \Sigma_1^{-1} (z-\mu_1)\right)} - \log_e e^{\left(-\frac{1}{2}(z-\mu_2)^T \Sigma_2^{-1} (z-\mu_2)\right)} \right]$$

$$= \mathbb{E}_{z \sim q(z|x_i)} \left[\frac{1}{2} \log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) + \left(-\frac{1}{2} (z-\mu_1)^T \Sigma_1^{-1} (z-\mu_1) \right) - \left(-\frac{1}{2} (z-\mu_2)^T \Sigma_2^{-1} (z-\mu_2) \right) \right]$$

$$= \frac{1}{2} \mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) - (z - \mu_1)^T \Sigma_1^{-1} (z - \mu_1) + (z - \mu_2)^T \Sigma_2^{-1} (z - \mu_2) \right]$$

Let's review the dimension of vectors and covariance matrices.

$\mu = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} \quad z = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix}$ <p style="text-align: center;">$k \times 1 \quad \quad k \times 1$</p>	$\Sigma = \begin{bmatrix} a_{1,1} & \dots & a_{1,k} \\ \vdots & a_{i,j} & \vdots \\ a_{k,1} & \dots & a_{k,k} \end{bmatrix}$ <p style="text-align: center;">$k \times k$</p>
$\mu^T = \begin{bmatrix} a_1 & \dots & a_k \end{bmatrix}$ <p style="text-align: center;">$1 \times k$</p> $z^T = \begin{bmatrix} a_1 & \dots & a_k \end{bmatrix}$ <p style="text-align: center;">$1 \times k$</p>	$\Sigma^{-1} = \begin{bmatrix} b_{1,1} & \dots & b_{1,k} \\ \vdots & b_{i,j} & \vdots \\ b_{k,1} & \dots & b_{k,k} \end{bmatrix}$ <p style="text-align: center;">$k \times k$</p>

From the dimensional analysis, I can determine that the expression below yields a scalar value.

$$\begin{matrix} (z - \mu_1)^T & \Sigma_1^{-1} & (z - \mu_1) & = & \alpha \\ (1 \times k) & (k \times k) & (k \times 1) & & (1 \times 1) \end{matrix}$$

$$\begin{bmatrix} a_1 & \dots & a_k \end{bmatrix} \begin{bmatrix} b_{1,1} & \dots & b_{1,k} \\ \vdots & b_{i,j} & \vdots \\ b_{k,1} & \dots & b_{k,k} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} = |\alpha|$$

Given that the trace of a scalar, which can be represented in a 1x1 matrix, is the scalar value itself,

$$\text{tr}(|\alpha|) = \alpha$$

I can express it like this.

$$= \frac{1}{2} \mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) - \text{tr} \left((z - \mu_1)^T \Sigma_1^{-1} (z - \mu_1) \right) + \text{tr} \left((z - \mu_2)^T \Sigma_2^{-1} (z - \mu_2) \right) \right]$$

In addition, I am going to use the cyclic property of the trace, which states that $\text{Tr}(ABC) = \text{Tr}(BCA)$. Let me apply this property to the formula above and obtain the expression below.

$$= \frac{1}{2} \mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) - \text{tr} \left(\Sigma_1^{-1} (z - \mu_1) (z - \mu_1)^T \right) + \text{tr} \left(\Sigma_2^{-1} (z - \mu_2) (z - \mu_2)^T \right) \right]$$

At this point, you may notice that the dimensionality of the expression has changed after applying the cyclic property of the trace, which moves the inverse covariance matrix to the front. Previously, the expression within the trace was a scalar (1x1), but after applying the cyclic property of the trace, it becomes a $k \times k$ matrix. The reason I can do this is that the trace of the resultant matrix is the same as the scalar from before the cyclic property was applied. Below is not a formal proof, but it may offers a visual understanding why this property works.

$$\text{tr}((z - \mu)^T \Sigma^{-1} (z - \mu))$$

$$\text{tr} \left(\begin{bmatrix} g & h \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} \right) = \text{tr} \left(\begin{bmatrix} ga + hc & gb + hd \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} \right) = \text{tr}(|\textcolor{red}{aeg} + \textcolor{red}{ceh} + \textcolor{red}{bfg} + \textcolor{red}{dfh}|)$$

$$\text{tr}(\Sigma^{-1} (z - \mu) (z - \mu)^T)$$

$$\text{tr} \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} \begin{bmatrix} g & h \end{bmatrix} \right) = \text{tr} \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} eg & eh \\ fg & fh \end{bmatrix} \right)$$

$$= \text{tr} \left(\begin{bmatrix} \textcolor{red}{aeg} + \textcolor{red}{ceh} & beg + deh \\ afg + cfh & \textcolor{red}{bfg} + \textcolor{red}{dfh} \end{bmatrix} \right) = |\textcolor{red}{aeg} + \textcolor{red}{ceh} + \textcolor{red}{bfg} + \textcolor{red}{dfh}|$$

I am going to perform algebraic manipulation to move the expectation inside the trace since both trace and expectation are linear operators.

$$= \frac{1}{2} \left(\mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) \right] - \text{tr} \left(\Sigma_1^{-1} \mathbb{E}_{z \sim q(z|x_i)} \left[(z - \mu_1)(z - \mu_1)^T \right] \right) + \text{tr} \left(\Sigma_2^{-1} \mathbb{E}_{z \sim q(z|x_i)} \left[(z - \mu_2)(z - \mu_2)^T \right] \right) \right)$$

Before proceeding with further algebraic manipulation, recall that q is a normal distribution where the mean and Sigma are shown below.

$$q(z) \sim \mathcal{N}(\mu_1, \Sigma_1)$$

Therefore, the expression below describes the variance, which is represented by a covariance matrix within a multivariate normal distribution.

$$\mathbb{E}_{z \sim q(z|x_i)} \left[(z - \mu_1)(z - \mu_1)^T \right] = \Sigma_1$$

And I can further simplify the expression as shown below.

$$= \frac{1}{2} \left(\mathbb{E}_{z \sim q(z|x_i)} \left[\log_e \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) \right] - \text{tr} (\Sigma_1^{-1} \Sigma_1) + \text{tr} (\Sigma_2^{-1} \mathbb{E}_{z \sim q(z|x_i)} [(zz^T - z\mu_2^T - \mu_2 z^T + \mu_2 \mu_2^T)]) \right)$$

Also, I am going to distribute the expectation to multiple terms within a parenthesis since they are random variables. Keep in mind that the expectation of a constant is the constant itself.

$$= \frac{1}{2} \left(\log_e \frac{|\Sigma_2|}{|\Sigma_1|} - \text{tr}(\Sigma_1^{-1} \Sigma_1) + \text{tr}(\Sigma_2^{-1} (\mathbb{E}_{z \sim q(z|x_i)} [zz^T] - \mathbb{E}_{z \sim q(z|x_i)} [z] \mu_2^T - \mu_2 \mathbb{E}_{z \sim q(z|x_i)} [z^T] + \mathbb{E}_{z \sim q(z|x_i)} [\mu_2 \mu_2^T])) \right)$$

If I multiply a covariance matrix (square matrix) by its inverse, I get the identity matrix, and the trace of a $k \times k$ identity matrix is k .

$$= \frac{1}{2} \left(\log_e \frac{|\Sigma_2|}{|\Sigma_1|} - k + \text{tr}(\Sigma_2^{-1} (\mathbb{E}_{z \sim q(z|x_i)} [zz^T] - \mathbb{E}_{z \sim q(z|x_i)} [z] \mu_2^T - \mu_2 \mathbb{E}_{z \sim q(z|x_i)} [z^T] + \mathbb{E}_{z \sim q(z|x_i)} [\mu_2 \mu_2^T])) \right)$$

Since the expectation of a random variable, z is its mean, μ ,

$$\mathbb{E}_{z \sim q(z|x_i)} [z] = \mu$$

and

$$\begin{aligned} \Sigma &= \mathbb{E}_{z \sim q(z|x)} [(z - \mu)(z - \mu)^T] \\ \Sigma &= \mathbb{E}_{z \sim q(z|x)} [zz^T - z\mu^T - \mu z^T + \mu\mu^T] \\ \Sigma &= \mathbb{E}_{z \sim q(z|x)} [zz^T] - \mathbb{E}_{z \sim q(z|x)} [z] \mu^T - \mu \mathbb{E}_{z \sim q(z|x)} [z^T] + \mu\mu^T \\ \Sigma &= \mathbb{E}_{z \sim q(z|x)} [zz^T] - \mu\mu^T - \mu\mu^T + \mu\mu^T \\ \Sigma &= \mathbb{E}_{z \sim q(z|x)} [zz^T] - \mu\mu^T \\ \mathbb{E}_{z \sim q(z|x)} [zz^T] &= \Sigma + \mu\mu^T \end{aligned}$$

I can further rearrange the expression below.

Open in app ↗



$$\begin{aligned}
&= \frac{1}{2} \left(\log_e \frac{|\Sigma_2|}{|\Sigma_1|} - k + \text{tr} \left(\Sigma_2^{-1} \left(\Sigma_1 + (\mu_2 - \mu_1) (\mu_2 - \mu_1)^T \right) \right) \right) \\
&= \frac{1}{2} \left(\log_e \frac{|\Sigma_2|}{|\Sigma_1|} - k + \text{tr} \left(\Sigma_2^{-1} \Sigma_1 \right) + \text{tr} \left(\Sigma_2^{-1} (\mu_2 - \mu_1) (\mu_2 - \mu_1)^T \right) \right)
\end{aligned}$$

Now I am going to apply the cyclic property of the trace.

$$= \frac{1}{2} \left(\log_e \frac{|\Sigma_2|}{|\Sigma_1|} - k + \text{tr} \left(\Sigma_2^{-1} \Sigma_1 \right) + \text{tr} \left((\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right) \right)$$

Because the trace yields a 1x1 dimension after applying the cyclic property, this leads to this **final expression**.

$$= \frac{1}{2} \left(\log_e \frac{|\Sigma_2|}{|\Sigma_1|} - k + \text{tr} \left(\Sigma_2^{-1} \Sigma_1 \right) + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right)$$

It has been quite a journey to get to this final expression. Now I simply need to plug in the values for the mean and Sigma and perform algebraic manipulation to get to the final form shown earlier.

Keep in mind that both q and p are normal distributions where q is a learned distribution of latent variables and p is a simple normal distribution,

$$q(z) \sim \mathcal{N}(\mu_1, \Sigma_1)$$

$$p(z) \sim \mathcal{N}(\mu_2, \Sigma_2) = \mathcal{N}(0, \mathcal{I})$$

$$\begin{aligned} &= \frac{1}{2} \left(\log_e \frac{|I|}{|\Sigma_1|} - k + \text{tr}(I^{-1} \Sigma_1) + \left(\vec{0} - \mu_1 \right)^T I^{-1} \left(\vec{0} - \mu_1 \right) \right) \\ &= \frac{1}{2} \left(-\log_e |\Sigma_1| - k + \text{tr}(\Sigma_1) + \mu_1^T \mu_1 \right) \end{aligned}$$

Before I proceed with the derivation, I want to emphasize that for computational efficiency, **the covariance matrix of the learned normal distribution is often modeled as a diagonal matrix in VAE**. This means that the learned variables are assumed to be independent.

Let's finish with the remaining derivation.

$$\begin{aligned} &= \frac{1}{2} \left(-\log_e \prod_j \sigma_j^2 - k + \sum_j \sigma_j^2 + \sum_j \mu_j^2 \right) \\ &= \frac{1}{2} \left(-\sum_j \log_e \sigma_j^2 - k + \sum_j \sigma_j^2 + \sum_j \mu_j^2 \right) \\ &= \frac{1}{2} \left(-\sum_j (\log_e \sigma_j^2 + 1) + \sum_j \sigma_j^2 + \sum_j \mu_j^2 \right) \end{aligned}$$

The expression can be rearranged to yield the **final form**.

$$= -\frac{1}{2} \sum_j (1 + \log_e \sigma_j^2 - \mu_j^2 - \sigma_j^2)$$

During the derivation of the KL term, I quickly realized my gaps in linear algebra, especially with vector and matrix operations. Fortunately, the “The Matrix Cookbook”, which helped me immensely in graduate school, was again valuable in completing this derivation. I hope this derivation is helpful.

Variational Autoencoder

KL Divergence

Loss Function

Formula Derivation

**Written by Jae H. Park**

8 Followers · 1 Following

Subscribe



I'm a machine learning/AI enthusiast who believes in its ability to solve humanity's biggest challenges. Find me on LinkedIn:

www.linkedin.com/in/jae-h-park

Responses (1)



Alex Mylnikov

What are your thoughts?



Graphodio
3 hours ago

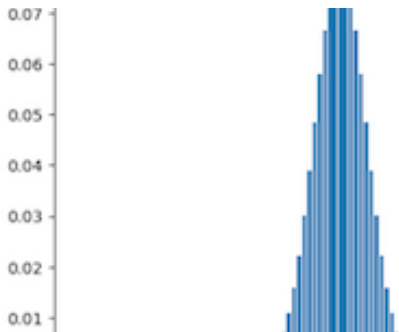


Nailed it.



[Reply](#)

More from Jae H. Park

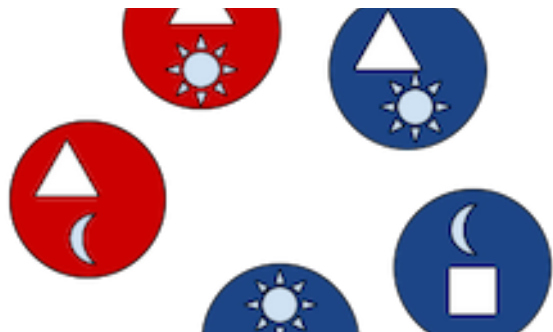


Jae H. Park

What is Bayesian belief? (Part2)

Likelihood, Prior, Evidence and Posterior

Feb 26, 2024



Jae H. Park

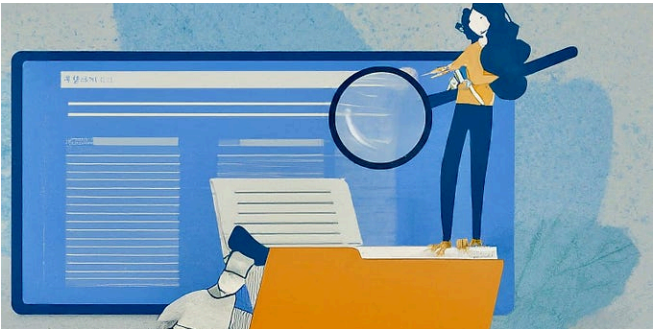
What is Bayesian belief? (Part 1)


Basic Probability Concepts

Feb 25, 2024



Probability of event	# of possible events ($\frac{1}{p}$)		Log scale
12.5% (1/8)	$\frac{1}{p} = \frac{1}{(1/8)} = 8$	$2^3 = 8$	$\log_2(8) = 3$
~16.66% (1/6)	$\frac{1}{p} = \frac{1}{(1/6)} = 6$	$2^{(\sim 2.58)} = 6$	$\log_2(6) = \sim 2.58$
25%	$\frac{1}{p} = \frac{1}{(1/4)} = 4$	$2^2 = 4$	$\log_2(4) = 2$
50%	$\frac{1}{p} = \frac{1}{(1/2)} = 2$	$2^1 = 2$	$\log_2(2) = 1$
75%	$\frac{1}{p} = \frac{1}{(3/4)} = \sim 1.33 (4/3)$	$2^{(\sim 0.42)} = \sim 1.33 (4/3)$	$\log_2(4/3) = \sim 0.42$
100%	$\frac{1}{p} = \frac{1}{(1/1)} = 1$	$2^0 = 1$	$\log_2(1) = 0$



 Jae H. Park

How can you measure “Uncertainty” ?

How can you measure “uncertainty”? This was a question I have posed to my kids when the...


Aug 19, 2023

 2

 1






 Jae H. Park

Forget Keywords, Think Meaning: The New Era of Searching

In 2020, during a seminar discussion about using AI to improve corporate search, I...

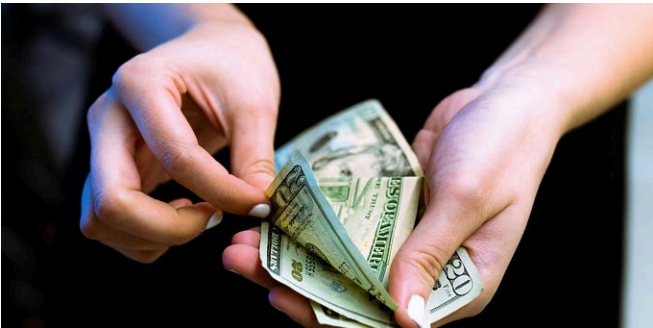
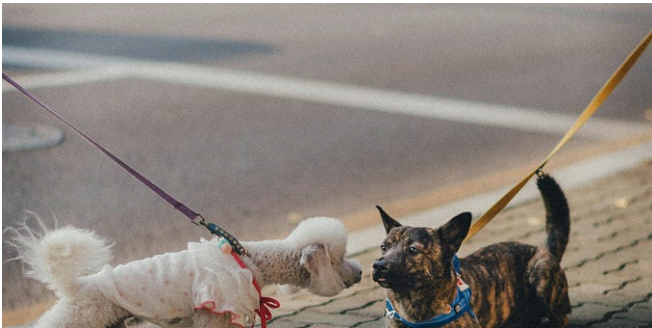
May 21, 2024





See all from Jae H. Park

Recommended from Medium



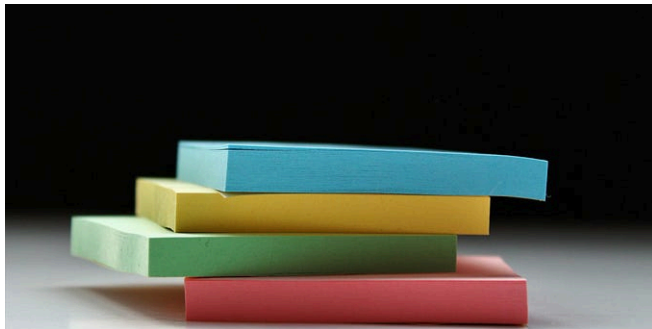


In Psyc Digest by Alessia Fransisca

The 1-Minute Introduction That Makes People Remember You...

A Behavioral Scientist's Trick to Hack the "Halo Effect"

Mar 19 12K 254

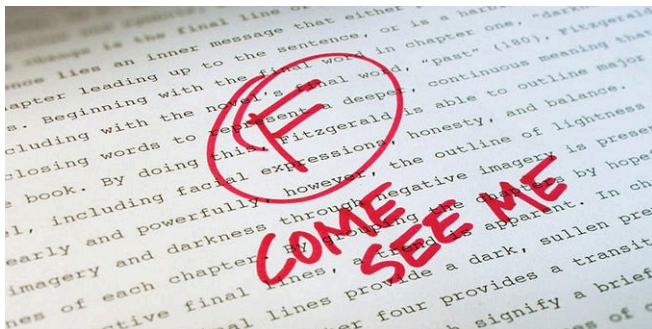


In Data Science Collecti... by Bradley Stephen Sh...

Teach Your GBM to Extrapolate with Model Stacking

Background

4d ago 146 4



Jesse Owen

Trump's Tariff Plan Stinks of AI

As a Professor, I've learned to pick out AI plagiarism—and I'd send this plan back with...

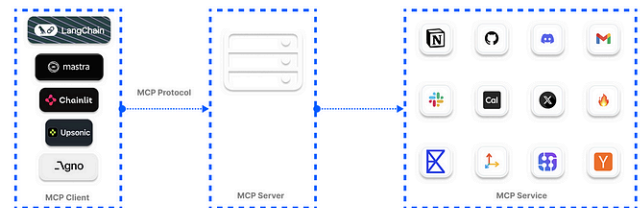


In Learn AI for Profit by Nipuna Maduranga

You Can Make Money With AI Without Quitting Your Job

I'm doing it, 2 hours a day

Mar 24 2.9K 135

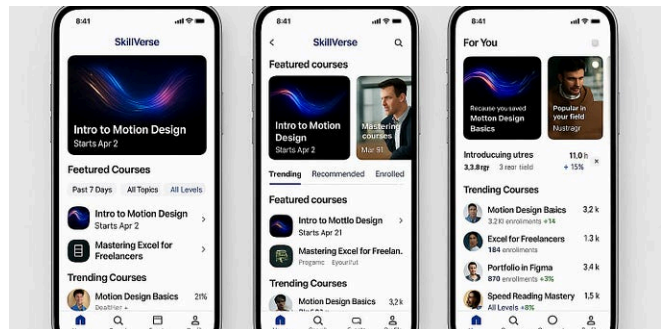


Amos Gyamfi

The Top 7 MCP-Supported AI Frameworks

Create AI apps with Python and Typescript frameworks that leverage MCP servers to...

Apr 1 728 17



In Bootcamp by Xinran Ma

How I created UI with ChatGPT's new image generator (4o)

Prompts, walkthroughs, and surprises

4d ago  1.97K  42   Mar 31  1.3K  27  

See more recommendations