Towards Data Science
A Medium publication sharing concepts, ideas, and codes.
Follow

1.3K

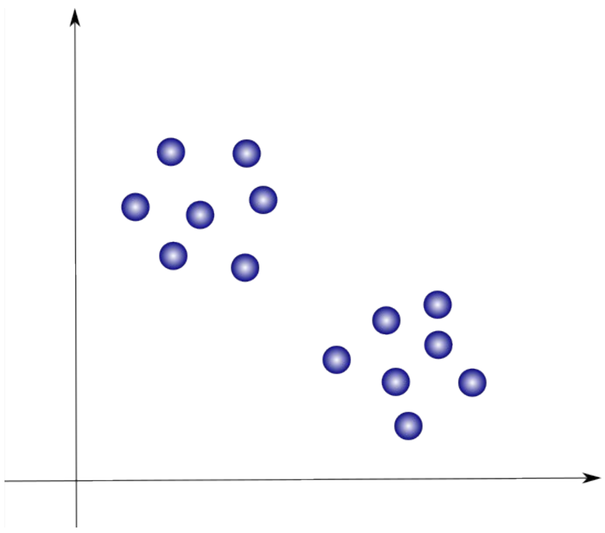
Top highlight

Gaussian Mixture Models Explained

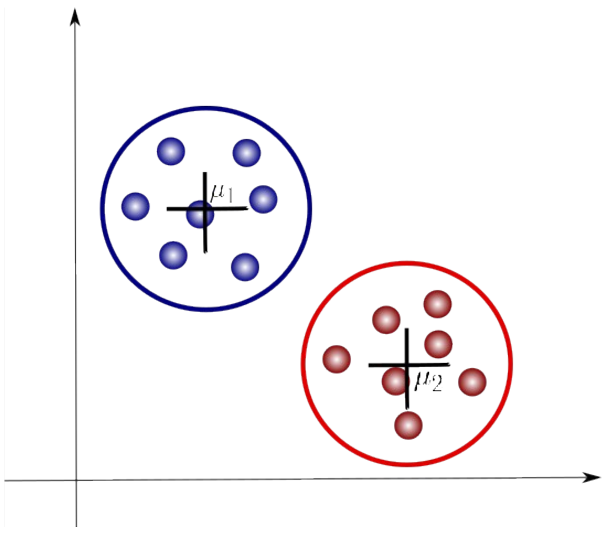
From intuition to implementation

Oscar Contreras Carrasco Follow
Jun 3, 2019 · 12 min read

In the world of Machine Learning, we can distinguish two main areas: Supervised and unsupervised learning. The main difference between both lies in the nature of the data as well as the approaches used to deal with it. Clustering is an unsupervised learning problem where we intend to find clusters of points in our dataset that share some common characteristics. Let's suppose we have a dataset that looks like this:



Our job is to find sets of points that appear close together. In this case, we can clearly identify two clusters of points which we will colour blue and red, respectively:



Please note that we are now introducing some additional notation. Here, μ1 and μ2 are the centroids of each cluster and are parameters that identify each of these. A popular clustering algorithm is known as K-means, which will follow an iterative approach to update the parameters of each clusters. More specifically, what it will do is to compute the means (or centroids) of each cluster, and then calculate their distance to each of the data points. The latter are then labeled as part of the cluster that is identified by their closest centroid. This process is repeated until some convergence criterion is met, for example when we see no further changes in the cluster assignments.

One important characteristic of K-means is that it is a *hard clustering method*, which means that it will associate each point to one and only one cluster. A limitation to this approach is that there is no uncertainty measure or *probability* that tells us how much a data point is associated with a specific cluster. So what about using a soft clustering instead of a hard one? This is exactly what Gaussian Mixture Models, or simply GMMs, attempt to do. Let's now discuss this method further.

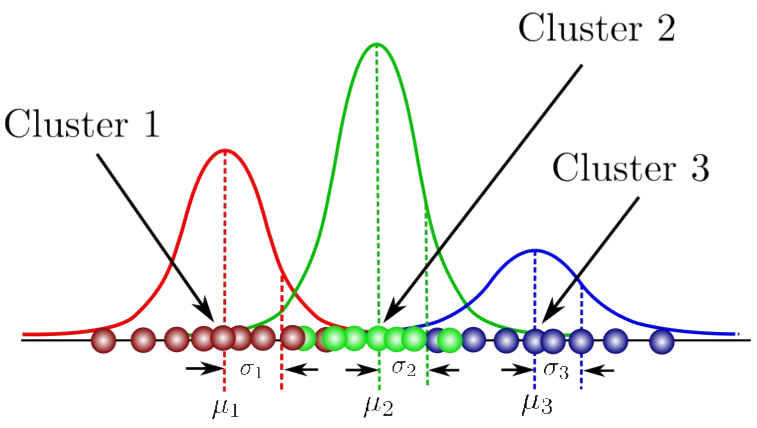
. . .

Definitions

A *Gaussian Mixture* is a function that is comprised of several Gaussians, each identified by $k \in \{1,\dots, K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.

Let us now illustrate these parameters graphically:



Here, we can see that there are three Gaussian functions, hence $K = 3$. Each Gaussian explains the data contained in each of the three clusters available. The mixing coefficients are themselves probabilities and must meet this condition:

$$\sum_{k=1}^{K} \pi_k = 1 \qquad (1)$$

Now how do we determine the optimal values for these parameters? To achieve this we must ensure that each Gaussian fits the data points belonging to each cluster. This is exactly what maximum likelihood does.

In general, the Gaussian density function is given by:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Where **x** represents our data points, D is the number of dimensions of each data point. μ and Σ are the mean and covariance, respectively. If we have a dataset comprised of N = 1000 three-dimensional points (D = 3), then **x** will be a 1000 × 3 matrix. μ will be a 1 × 3 vector, and Σ will be a 3 × 3 matrix. For later purposes, we will also find it useful to take the log of this equation, which is given by:

$$\ln \mathcal{N}(\mathbf{x}|\mu,\Sigma) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln \Sigma - \frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) \qquad (2)$$

If we differentiate this equation with respect to the mean and covariance and then equate it to zero, then we will be able to find the optimal values for these parameters, and the solutions will correspond to the Maximum Likelihood Estimates (MLE) for this setting. However, because we are dealing with not just one, but many Gaussians, things will get a bit complicated when time comes for us to find the parameters for the whole mixture. In this regard, we will need to introduce some additional aspects that we discuss in the next section.

. . .

Initial derivations

We are now going to introduce some additional notation. Just a word of warning. Math is coming on! Don't worry. I'll try to keep the notation as clean as possible for better understanding of the derivations. First, let's suppose we want to know what is the probability that a data point **x***n* comes from Gaussian *k*. We can express this as:

$$p(z_{nk} = 1|\mathbf{x}_n)$$

Which reads "*given a data point* **x**, *what is the probability it came from Gaussian k?"* In this case, *z* is a *latent variable* that takes only two possible values. It is one when **x** came from Gaussian *k*, and zero otherwise. Actually, we don't get to see this *z* variable in reality, but knowing its probability of occurrence will be useful in helping us determine the Gaussian mixture parameters, as we discuss later.

Likewise, we can state the following:

$$\pi_k = p(z_k = 1)$$

Which means that the overall probability of observing a point that comes from Gaussian *k* is actually equivalent to the mixing coefficient for that Gaussian. This makes sense, because the bigger the Gaussian is, the higher we would expect this probability to be. Now let **z** be the set of all possible latent variables *z*, hence:

$$\mathbf{z} = \{z_1, \ldots, z_K\}$$

We know beforehand that each *z* occurs independently of others and that they can only take the value of one when *k* is equal to the cluster the point comes from. Therefore:

$$p(\mathbf{z}) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} \ldots p(z_K = 1)^{z_K} = \prod_{k=1}^{K} \pi_k^{z_k}$$

Now, what about finding the probability of observing our data given that it came from Gaussian *k*? Turns out to be that it is actually the Gaussian function itself! Following the same logic we used to define *p*(**z**), we can state:

$$p(\mathbf{x}_n|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)^{z_k}$$

Ok, now you may be asking, why are we doing all this? Remember our initial aim was to determine what the probability of *z* given our observation **x**? Well, it turns out to be that the equations we have just derived, along with the Bayes rule, will help us determine this probability. From the product rule of probabilities, we know that

$$p(\mathbf{x}_n, \mathbf{z}) = p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z})$$

Hmm, it seems to be that now we are getting somewhere. The operands on the right are what we have just found. Perhaps some of you may be anticipating that we are going to use the Bayes rule to get the probability we eventually need. However, first we will need ***p*(x*n*)**, not ***p*(x*n*, z)**. So how do we get rid of **z** here? Yes, you guessed it right. Marginalization! We just need to sum up the terms on **z**, hence

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)$$

This is the equation that defines a Gaussian Mixture, and you can clearly see that it depends on all parameters that we mentioned previously! To determine the optimal values for these we need to determine the maximum likelihood of the model. We can find the likelihood as the joint probability of all observations **x***n*, defined by:

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) = \prod_{n=1}^{N}\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)$$

Like we did for the original Gaussian density function, let's apply the log to each side of the equation:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \qquad (3)$$

Great! Now in order to find the optimal parameters for the Gaussian mixture, all we have to do is to differentiate this equation with respect to the parameters and we are done, right? Wait! Not so fast. We have an issue here. We can see that there is a logarithm that is affecting the second summation. Calculating the derivative of this expression and then solving for the parameters is going to be very hard!

What can we do? Well, we need to use an iterative method to estimate the parameters. But first, remember we were supposed to find the probability of *z* given **x**? Well, let's do that since at this point we already have everything in place to define what this probability will look like.

From Bayes rule, we know that

$$p(z_k = 1|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(\mathbf{x}_n|z_j = 1)p(z_j = 1)}$$

From our earlier derivations we learned that:

$$p(z_k = 1) = \pi_k, \qquad p(\mathbf{x}_n|z_k = 1) = \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)$$

So let's now replace these in the previous equation:

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}) \qquad (4)$$

And this is what we've been looking for! Moving forward we are going to see this expression a lot. Next we will continue our discussion with a method that will help us easily determine the parameters for the Gaussian mixture.

. . .

Expectation — Maximization algorithm

Well, at this point we have derived some expressions for the probabilities that we will find useful in determining the parameters of our model. However, in the past section we could see that simply evaluating (3) to find such parameters would prove to be very hard. Fortunately, there is an iterative method we can use to achieve this purpose. It is called the *Expectation — Maximization*, or simply *EM algorithm*. It is widely used for optimization problems where the objective function has complexities such as the one we've just encountered for the GMM case.

Let the parameters of our model be

$$\theta = \{\pi, \mu, \Sigma\}$$

Let us now define the steps that the general EM algorithm will follow¹.

Step 1: Initialise θ accordingly. For instance, we can use the results obtained by a previous K-Means run as a good starting point for our algorithm.

Step 2 (Expectation step): Evaluate

$$\mathcal{Q}(\theta^*, \theta) = \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z}|\theta^*)] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta) \ln p(\mathbf{X}, \mathbf{Z}|\theta^*) \qquad (5)$$

Well, actually we have already found $p(\mathbf{Z}|\mathbf{X}, \theta)$. Remember the γ expression we ended up with in the previous section? For better visibility, let's bring our earlier equation (4) here:

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}) \qquad (4)$$

For Gaussian Mixture Models, the expectation step boils down to calculating the value of γ in (4) by using the old parameter values. Now if we replace (4) in (5), we will have:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{\mathbf{Z}} \gamma(z_{nk}) \ln p(\mathbf{X}, \mathbf{Z}|\theta^*) \qquad (6)$$

Sounds good, but we are still missing $p(\mathbf{X}, \mathbf{Z}|\theta^*)$. How can we find it? Well, actually it's not that difficult. It is just the complete likelihood of the model, including both **X** and **Z**, and we can find it by using the following expression:

$$p(\mathbf{X}, \mathbf{Z}|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi^{z_{nk}} \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_{nk}}$$

Which is the result of calculating the joint probability of all observations and latent variables and is an extension of our initial derivations for $p(\mathbf{x})$. The log of this expression is given by

$$\ln p(\mathbf{X}, \mathbf{Z}|\theta^*) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}[\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)] \qquad (7)$$

Nice! And we have finally gotten rid of this troublesome logarithm that affected the summation in (3). With all of this in place, it will be much easier for us to estimate the parameters by just maximizing Q with respect to the parameters, but we will deal with this in the *maximization step*. Besides, remember that the latent variable z will **only** be 1 once everytime the summation is evaluated. With that knowledge, we can easily get rid of it as needed for our derivations.

Finally, we can replace (7) in (6) to get:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)] \qquad (8)$$

In the maximization step, we will find the revised parameters of the mixture. For this purpose, we will need to make Q a restricted maximization problem and thus we will add a Lagrange multiplier to (8). Let's now review the maximization step.

Step 3 (Maximization step): Find the revised parameters θ^* using:

$$\theta^* = \arg\max_\theta \mathcal{Q}(\theta^*, \theta)$$

Where

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)] \qquad (8)$$

Which is what we ended up with in the previous step. However, Q should also take into account the restriction that all π values should sum up to one. To do so, we will need to add a suitable Lagrange multiplier. Therefore, we should rewrite (8) in this way:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k)] - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \qquad (8)$$

And now we can easily determine the parameters by using maximum likelihood. Let's now take the derivative of *Q* with respect to π and set it equal to zero:

$$\frac{\partial Q(\theta^*, \theta)}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(z_{nk})}{\pi_k} - \lambda = 0$$

Then, by rearranging the terms and applying a summation over *k* to both sides of the equation, we obtain:

$$\sum_{n=1}^{N} \gamma(z_{nk}) = \pi_k \lambda \implies \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma(z_{nk}) = \sum_{k=1}^{K} \pi_k \lambda$$

From (1), we know that the summation of all mixing coefficients π equals one. In addition, we know that summing up the probabilities γ over *k* will also give us 1. Thus we get $\lambda = N$. Using this result, we can solve for π:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$$

Similarly, if we differentiate *Q* with respect to μ and Σ, equate the derivative to zero and then solve for the parameters by making use of the log-likelihood equation (2) we defined, we obtain:

$$\mu_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \qquad \Sigma_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

And that's it! Then we will use these revised values to determine γ in the next EM iteration and so on and so forth until we see some convergence in the likelihood value. We can use equation (3) to monitor the log-likelihood in each step and we are always guaranteed to reach a local maximum.

It would be nice to see how we can implement this algorithm using a programming language, wouldn't it? Next, we will see parts of the Jupyter notebook I have provided so you can see a working implementation of GMMs in Python.

. . .

Implementation in Python

Just as a side note, the full implementation is available as a Jupyter notebook at https://bit.ly/2MpiZp4

I have used the Iris dataset for this exercise, mainly for simplicity and fast training. From our previous derivations, we stated that the EM algorithm follows an iterative approach to find the parameters of a Gaussian Mixture Model. Our first step was to initialise our parameters. In this case, we can use the values of K-means to suit this purpose. The Python code for this would look like:

Next, we execute the expectation step. Here we calculate

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}$$

And the corresponding Python code would look like:

Note that in order to calculate the summation we just make use of the terms in the numerator and divide accordingly.

We then have the maximization step, where we calculate

$$\pi_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$$

$$\mu_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \qquad \Sigma_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

The corresponding Python code for this would be the following:

Note that in order to simplify the calculations a bit, we have made use of:

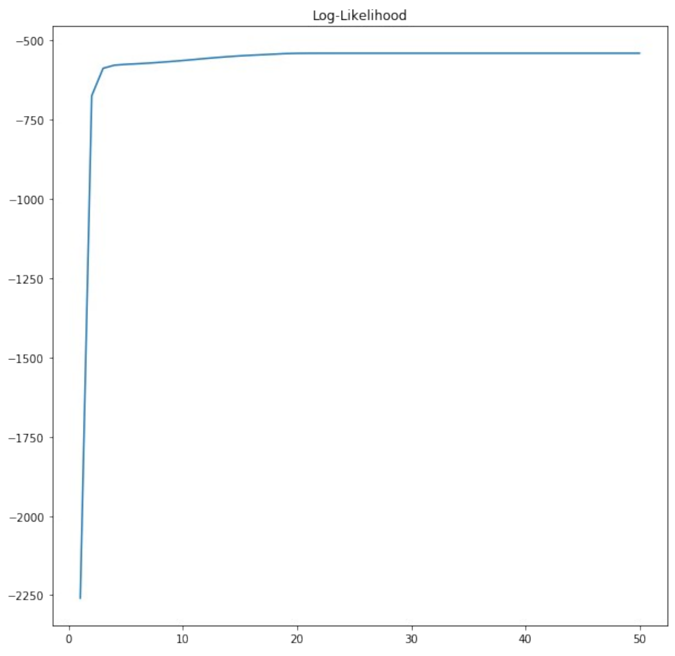
$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Finally, we also have the log-likelihood calculation, which is given by

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)$$

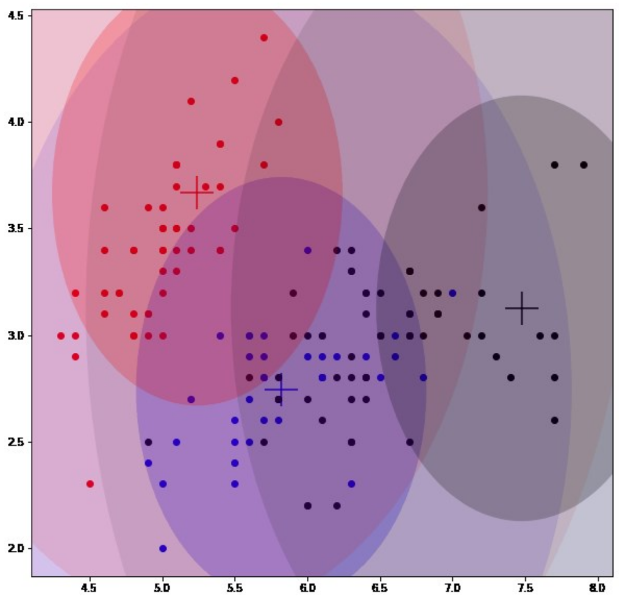
The Python code for this would be

We have pre-computed the value of the second summation in the expectation step, so we just make use of that here. In addition, it is always useful to create graphs to see how the likelihood is making progress.



We can clearly see that the algorithm converges after about 20 epochs. EM guarantees that a local maximum will be reached after a given number of iterations of the procedure.

Finally, as part of the implementation we also generate an animation that shows us how the cluster settings improve after each iteration.



Note how the GMM improves the centroids estimated by K-means. As we converge, the values for the parameters for each cluster do not change any further.

. . .

Final remarks

Gaussian Mixture Models are a very powerful tool and are widely used in diverse tasks that involve data clustering. I hope you found this post useful! Feel free to approach with questions or comments. I would also highly encourage you to try the derivations yourself as well as look further into the code. I look forward to creating more material like this soon.

Enjoy!

. . .

[1] Bishop, Christopher M. *Pattern Recognition and Machine Learning* (2006) Springer-Verlag Berlin, Heidelberg.

[2] Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective* (2012) MIT Press, Cambridge, Mass,

Machine Learning | Gaussian Mixture Model | Gmm | Clustering | Towards Data Science

1.3K claps

WRITTEN BY

Oscar Contreras Carrasco

Follow

Data Engineer, AI & ML Enthusiast. Machine learning educator, volunteer and entrepreneur. LinkedIn: https://www.linkedin.com/in/oscar-contreras/

Towards Data Science

Follow

A Medium publication sharing concepts, ideas, and codes.

See responses (8)

More From Medium

More from Towards Data Science

Bye-bye Python. Hello Julia!

Rhea Moutafis in Towards Data Science
May 2 · 8 min read ★ 9.94K

More from Towards Data Science

Don’t Become a Data Scientist

Chris in Towards Data Science
May 4 · 6 min read ★ 6.6K

More from Towards Data Science

Do Not Use “+” to Join Strings in Python

Christopher Tao in Towards Data Science
May 10 · 5 min read ★ 1.3K