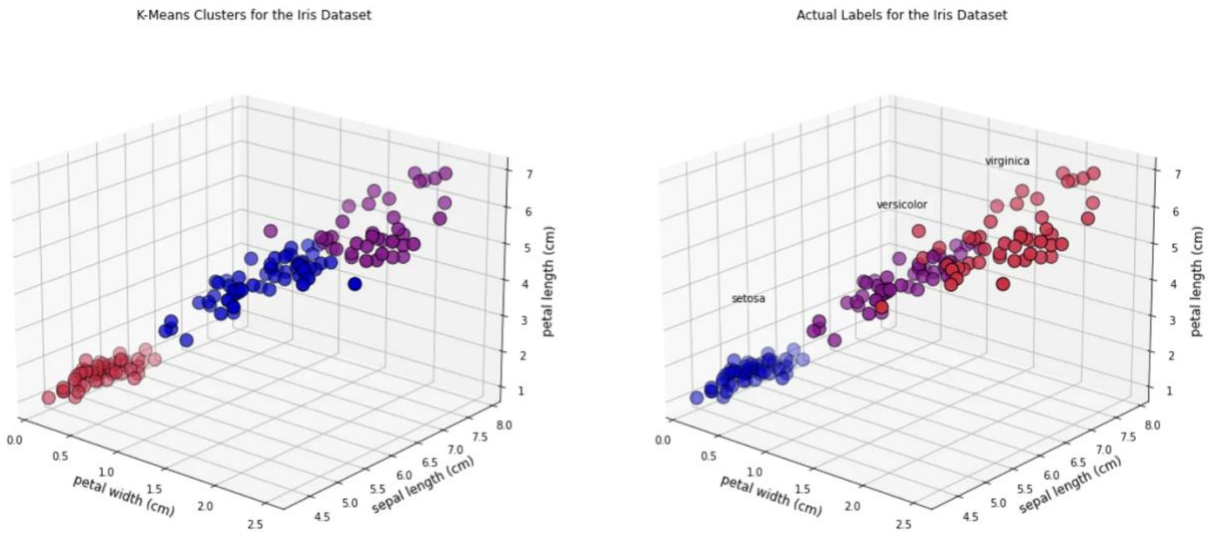
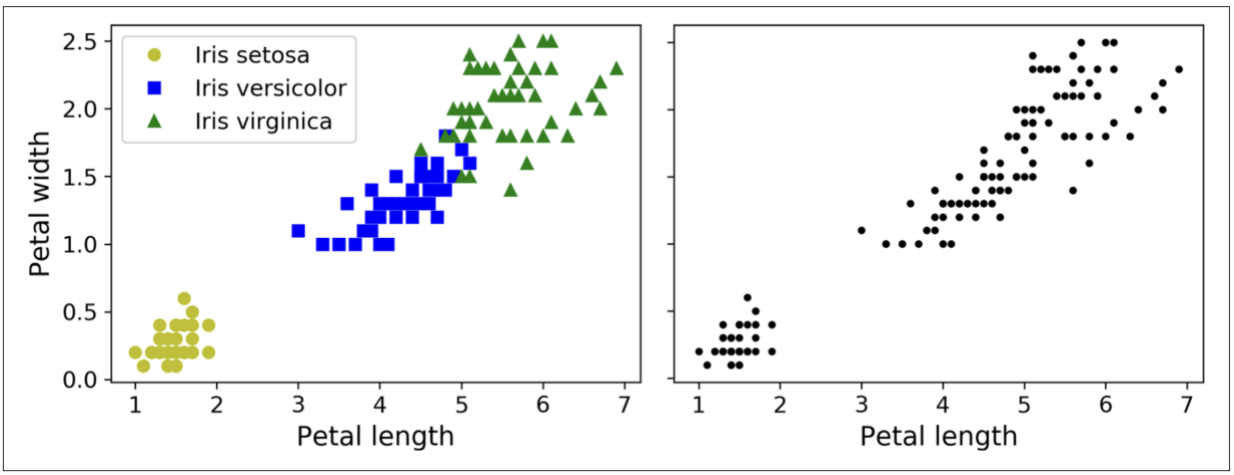
9. Unsupervised Learning Techniques



The computer scientist Yann LeCun famously said that “if intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake.

Clustering

It is the task of identifying similar instances and assigning them to clusters, or groups of similar instances.



Applications of clustering: For customer segmentation , For data analysis , As a dimensionality reduction technique , For anomaly detection (also called outlier detection) , For semi-supervised learning , For search engines

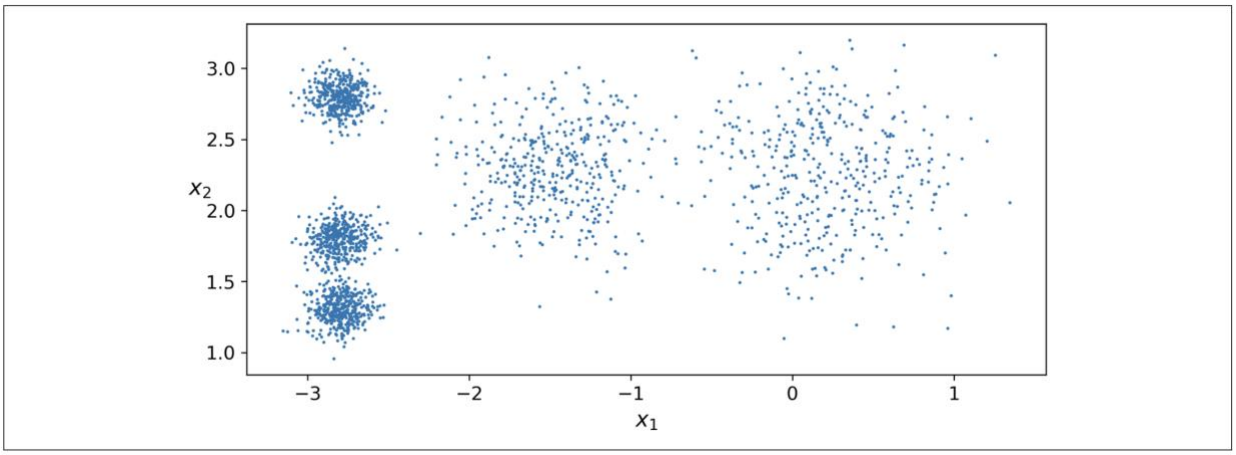
There is no universal definition of what a cluster is: it really depends on the context, and different algorithms will capture different kinds of clusters. Some algorithms look for instances centered around a particular point, called a centroid. Others look for continuous regions of densely packed instances: these clusters can take on any shape. Some algorithms are hierarchical, looking for clusters of clusters.

Here are 2 of the most used clusters:

K-Means

K-means is like a smart organizer, grouping similar items (data points) together into clusters. It's a fundamental tool in data analysis, helping to uncover patterns and insights in complex datasets.

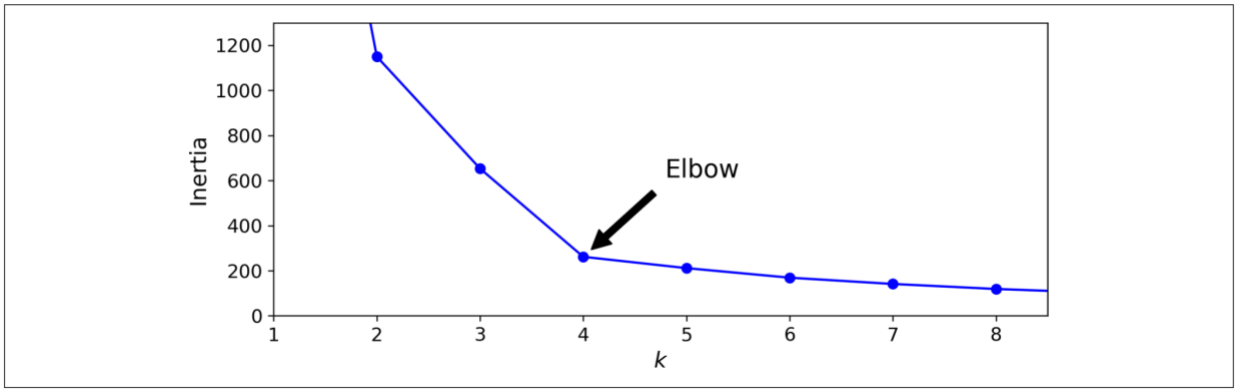
The "K" in K-means represents the number of clusters you want to divide your data into. If you choose K=3, the algorithm will create 3 clusters.



How K-means Works

1. **Initialization**: Randomly pick K points from your data. These points are the initial "centroids" of your clusters.
2. **Assignment**: Assign each data point to the nearest centroid. This is done by measuring the distance (usually Euclidean distance) between data points and centroids. A data point is assigned to the cluster of the centroid it is closest to.

3. **Update**: Calculate new centroids. After all data points are assigned to clusters, the centroid of each cluster is recalculated. This is done by taking the average of all the points in the cluster.
4. **Repeat**: Repeat the assignment and update steps until the centroids no longer change much (the distance of the center of the points to the center of the data is minimum) This means your clusters are stable, and the algorithm has done its job.



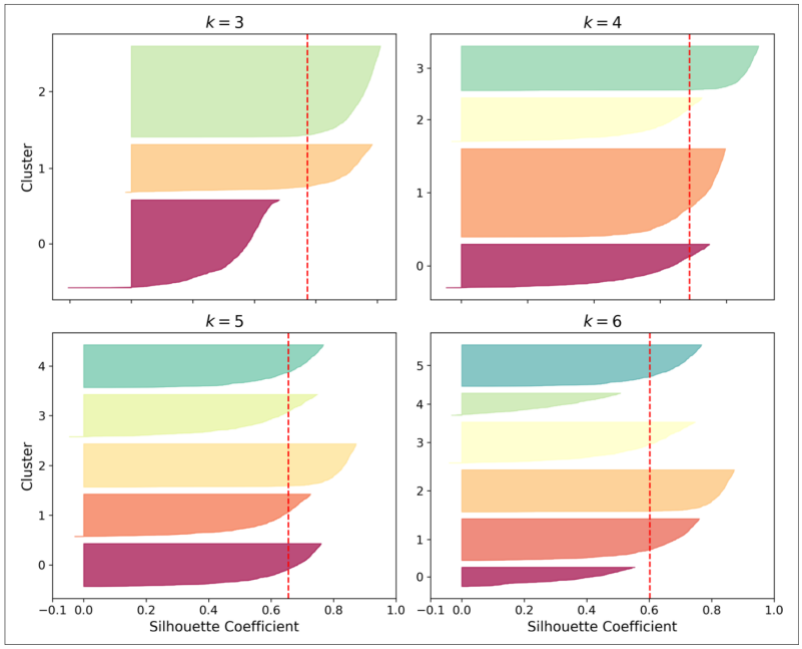
Importance of Inertia

- **Evaluating Cluster Quality**: It helps in assessing how well the K-means algorithm has performed. Lower inertia means that the clusters are more compact and better separated.
- **Choosing the Best K**: Inertia can be used in methods like the Elbow Method, where you plot the inertia values for different numbers of clusters (K) and look for a 'knee' in the graph. This 'knee' is often considered a good choice for K.

Limitations

- **Not a Perfect Metric**: A lower inertia doesn't always mean better clustering, especially if clusters have different densities or non-spherical shapes.
- **Sensitive to Scale**: Inertia is sensitive to the scale of the data. Features with larger scales will dominate the inertia value. Hence, data normalization is often required before applying K-means.

How to find the right number of clusters



The vertical dashed lines represent the silhouette score for each number of clusters. When most of the instances in a cluster have a lower coefficient than this score (i.e., if many of the instances stop short of the dashed line, ending to the left of it), then the cluster is rather bad since this means its instances are much too close to other clusters. We can see that when k = 3 and when k = 6, we get bad clusters. But when k = 4 or k = 5, the clusters look pretty good: most instances extend beyond the dashed line, to the right and closer to 1.0.

Other useful applications of kmeans

Image Segmentation : The pages that compress images they can do this slightly, reduce the number of encoded rgb to compress the image channels, by clustering them

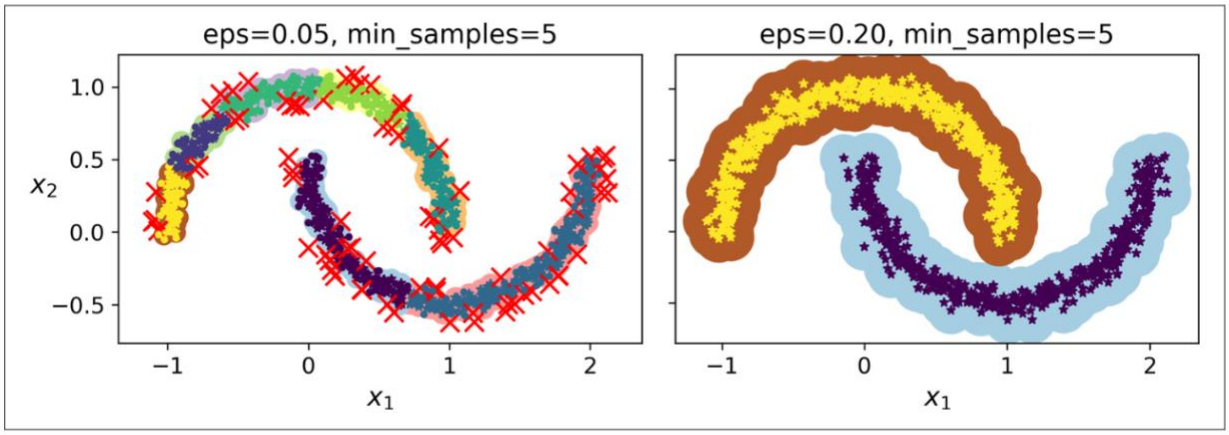
Dimensionality Reduction: For example, preprocessing the iris dataset

Semi-Supervised Learning: On the instances that are labeled properly, we can propagate the label to the other unknown data, based on the distance of the cluster.

To enhance the model on the training set, we could do some few rounds of Active Learning, one of its forms its called uncertainty sampling: The worst predictions done on our model are corrected to an expert to corrected, in an iterative way.

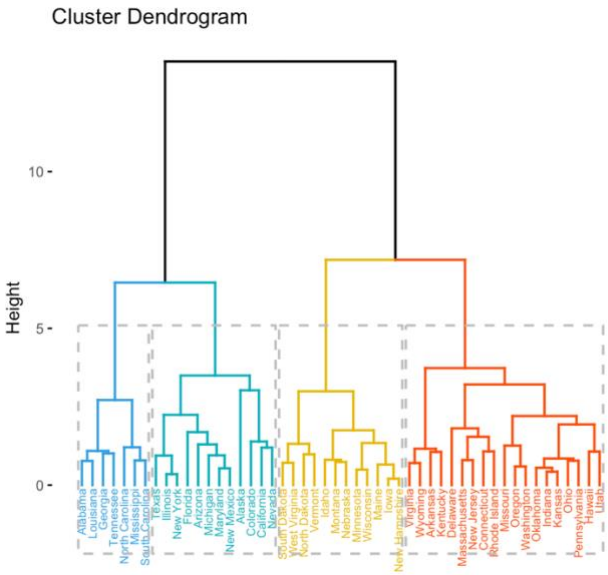
DBSCAN

- For each instance, the algorithm counts how many instances are located within a small distance ε (epsilon) from it. This region is called the instance's ε- neighborhood.
- If an instance has at least min_samples instances in its ε-neighborhood (including itself), then it is considered a core instance. In other words, core instances are those that are located in dense regions.
- All instances in the neighborhood of a core instance belong to the same cluster. This neighborhood may include other core instances; therefore, a long sequence of neighboring core instances forms a single cluster.
- Any instance that is not a core instance and does not have one in its neighbor- hood is considered an anomaly.



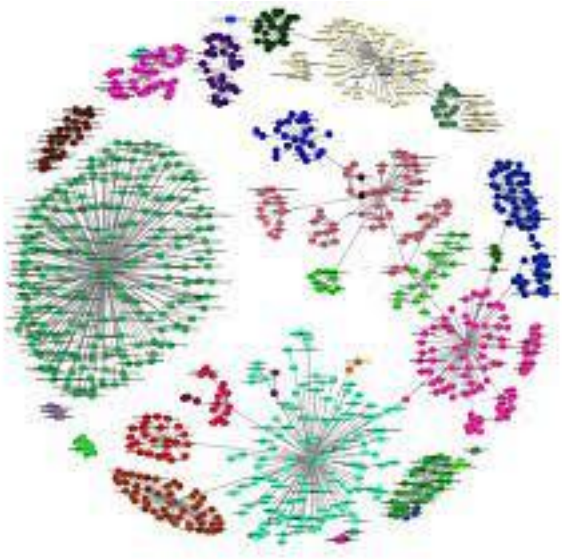
DBSCAN is a very simple yet powerful algorithm capable of identifying any number of clusters of any shape. It is robust to outliers, and it has just two hyperparameters (eps and min_samples). If the density varies significantly across the clusters, however, it can be impossible for it to capture all the clusters properly. Its computational complexity is roughly O(m log m), making it pretty close to linear with regard to the number of instances

Other Clustering Algorithms



Agglomerative clustering

A hierarchy of clusters is built from the bottom up. Think of many tiny bubbles floating on water and gradually attaching to each other until there's one big group of bubbles. Similarly, at each iteration, agglomerative clustering connects the nearest pair of clusters (starting with individual instances). Without a connectivity matrix, the algorithm does not scale well to large datasets.

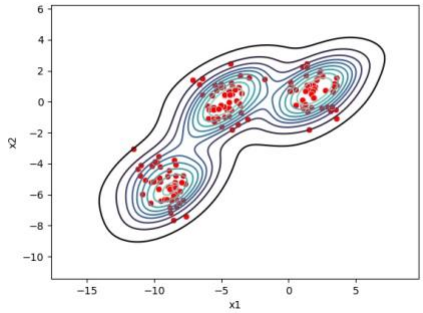


BIRCH

The BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm was designed specifically for very large datasets, and it can be faster than batch K-Means, with similar results, as long as the number of features is not too large (<20). During training, it builds a tree structure containing just enough information to quickly assign each new instance to a cluster, without having to store all the instances in the tree: this approach allows it to use limited memory, while handling huge datasets.

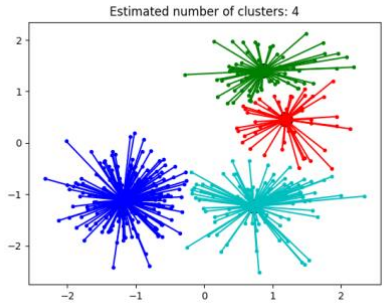
Mean-Shift

This algorithm starts by placing a circle centered on each instance; then for each circle it computes the mean of all the instances located within it, and it shifts the circle so that it is centered on the mean. Next, it iterates this mean-shifting step until all the circles stop moving (i.e., until each of them is centered on the mean of the instances it contains). Mean-Shift shifts the circles in the direction of higher density, until each of them has found a local density maximum. Finally, all the instances whose circles have settled in the same place (or close enough) are assigned to the same cluster.



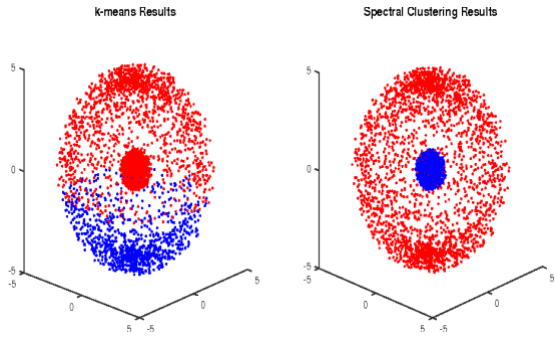
Affinity propagation

This algorithm uses a voting system, where instances vote for similar instances to be their representatives, and once the algorithm converges, each representative and its voters form a cluster. Affinity propagation can detect any number of clusters of different sizes. Unfortunately, this algorithm has a computational complexity of $O(m^2)$, so it too is not suited for large datasets.



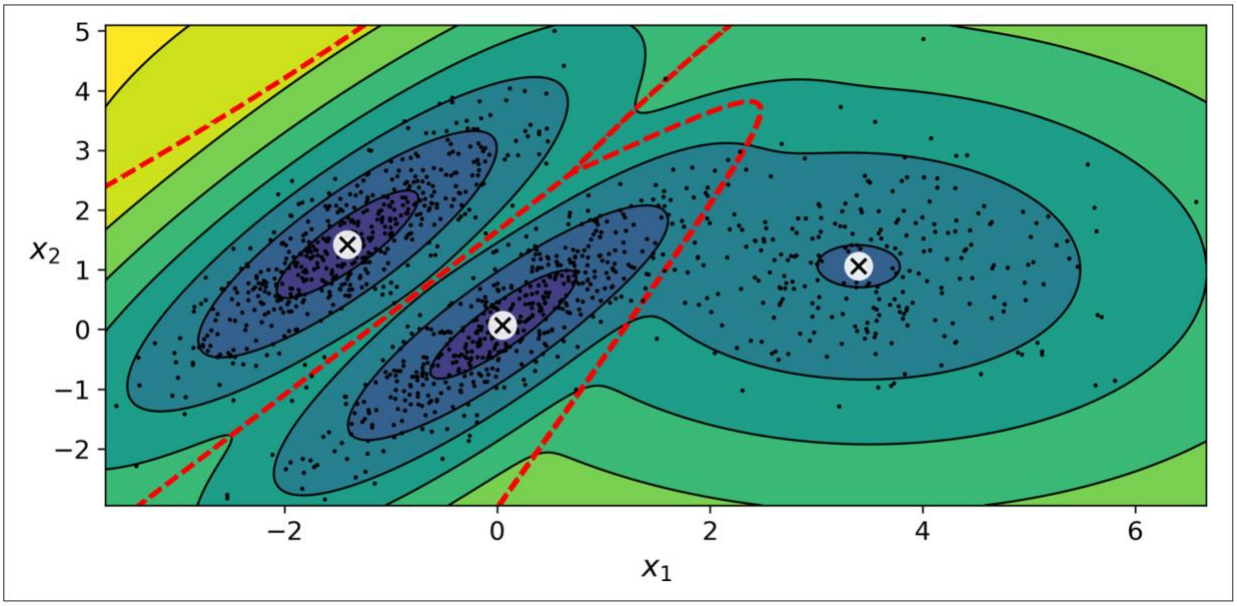
Spectral clustering

This algorithm takes a similarity matrix between the instances and creates a low- dimensional embedding from it (i.e., it reduces its dimensionality), then it uses another clustering algorithm in this low-dimensional space (Scikit-Learn's implementation uses K-Means.) Spectral clustering can capture complex cluster structures, and it can also be used to cut graphs (e.g., to identify clusters of friends on a social network). It does not scale well to large numbers of instances, and it does not behave well when the clusters have very different sizes.



Gaussian Mixtures

A Gaussian mixture model (GMM) is a probabilistic model that assumes that the instances were generated from a mixture of several Gaussian distributions whose parameters are unknown. All the instances generated from a single Gaussian distribution form a cluster that typically looks like an ellipsoid. Each cluster can have a different ellipsoidal shape, size, density, and orientation. When you observe an instance, you know it was generated from one of the Gaussian distributions, but you are not told which one, and you do not know what the parameters of these distributions are.



The computational complexity of training a GaussianMixture model depends on the number of instances m, the number of dimensions n, the number of clusters k, and the constraints on the covariance matrices. If covariance_type is "spherical or "diag", it is O(kmn), assuming the data has a clustering structure. If cova riance_type is "tied" or "full", it is $O(kmn^2 + kn^3)$, so it will not scale to large numbers of features.

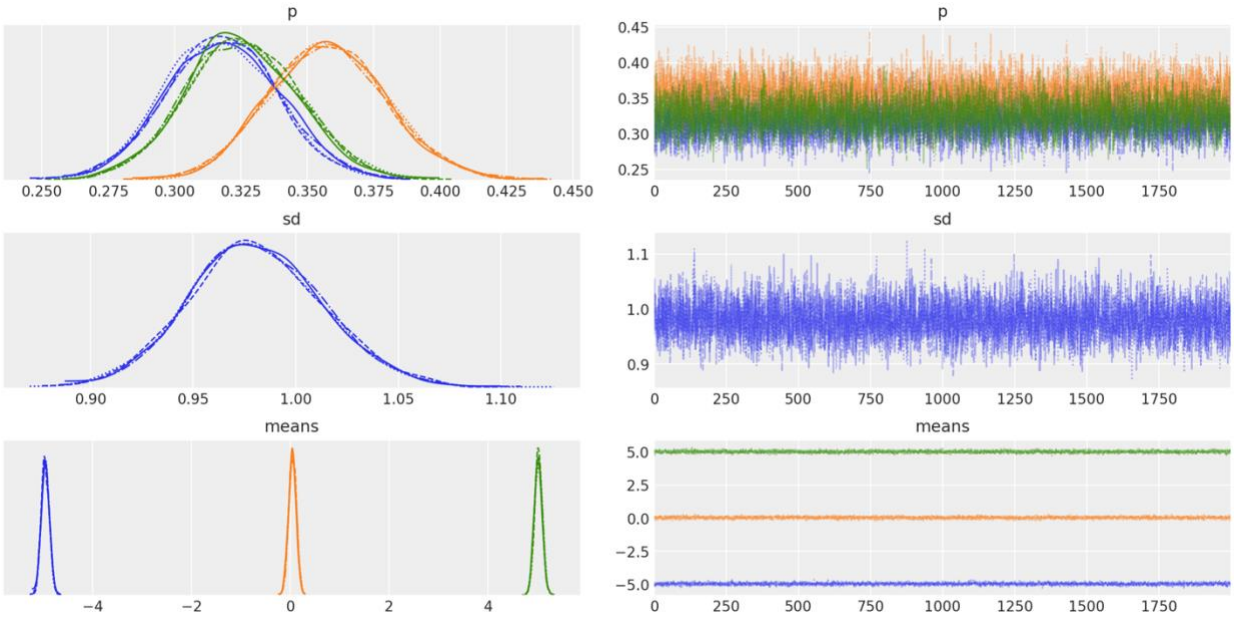
Anomaly Detection Using Gaussian Mixtures

Using a Gaussian mixture model for anomaly detection is quite simple: any instance located in a low-density region can be considered an anomaly. You must define what density threshold you want to use.

Understanding Gaussian Mixture Models (GMMs)

First, let's start with Gaussian Mixture Models. A GMM is a probabilistic model for representing normally distributed subpopulations within an overall population. Imagine you have data points and you suspect these points come from different groups (or 'clusters'), but you don't know which points belong to which group.

- **Gaussians**: Each group is modeled as a Gaussian distribution. A Gaussian (or normal) distribution is defined by its mean (center point) and variance (spread or width).
- **Mixture**: You assume that your entire dataset is a mixture of these Gaussian distributions. Each point in your dataset is considered to have been drawn from one of these distributions.



The Bayesian Approach

Now, let's add the Bayesian aspect. Bayesian methods involve using prior knowledge (priors) to influence your model. It's like saying, "Based on what I already know, what can I infer about my data?"

- **Priors**: In the context of GMMs, Bayesian methods allow you to incorporate prior beliefs about the parameters of the Gaussians (like means and variances).
- **Posterior Inference**: You use your data to update these beliefs. The result is a 'posterior' distribution which combines your prior knowledge with the evidence from the data.

Why Bayesian GMMs?

1. **Handling Uncertainty**: The Bayesian approach naturally handles uncertainty. Instead of just finding a single 'best' estimate for each parameter, it gives you a distribution representing all plausible values given the data and your priors.
2. **Regularization**: It can provide a form of regularization (preventing overfitting), especially useful if you don't have a lot of data. The priors can stop the model from becoming too complex.
3. **Model Complexity**: Bayesian GMMs can also determine the number of clusters (components) in your data in a more principled way. Traditional GMMs require you to specify the number of components, but Bayesian GMMs can infer this from the data.

Working Mechanism

1. **Initialization**: Start with some initial guess about the parameters (means, variances) of the Gaussians.
2. **Expectation-Maximization (EM)**: The model uses an iterative process (EM algorithm) to update these parameters. In each iteration:
    - **Expectation Step**: Estimate the probability that each data point belongs to each cluster.
    - **Maximization Step**: Update the parameters of the Gaussians based on these probabilities.
3. **Bayesian Update**: After each iteration, the Bayesian aspect comes into play. The model updates the parameters not just based on the data (like traditional GMMs) but also considering the prior distributions.
4. **Convergence**: This process is repeated until the parameters converge to stable values.

The Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) are both methods used for model selection, particularly in the context of statistical models like Gaussian Mixture Models (GMMs). They help to choose the best model among a set of candidates by balancing model fit and complexity.

Akaike Information Criterion (AIC)

- **Purpose**: AIC is used to quantify the information loss when using a particular model to represent the process that generated the data. It favors models that give a good fit with fewer parameters.
- **Formula**: AIC=2k−2ln⁡(L)AIC=2k−2ln(L)
    - kk is the number of parameters in the model.
    - LL is the likelihood of the model, i.e., how well the model explains the data.
- **Usage**: When comparing models, the one with the lower AIC value is generally preferred.
- **Trade-off**: AIC penalizes the complexity of the model (number of parameters), thus discouraging overfitting.

Bayesian Information Criterion (BIC)

- **Purpose**: Similar to AIC, BIC is also a criterion for model selection, but it introduces a stronger penalty for the number of parameters in the model.
- **Formula**: BIC=ln⁡(n)k−2ln⁡(L)BIC=ln(n)k−2ln(L)
    - nn is the number of data points.
    - kk and LL are the same as in AIC.
- **Usage**: Like AIC, a lower BIC score indicates a better model.
- **Trade-off**: BIC's penalty term is larger than that of AIC, especially for larger datasets, leading to simpler models. It's based on Bayesian probability and often considered more rigorous for model selection.

AIC vs BIC in Context

- **Model Complexity**: AIC tends to favor more complex models than BIC.
- **Sample Size Sensitivity**: BIC penalizes model complexity more heavily, especially as the sample size increases. This can lead to the selection of simpler models in cases of large datasets.
- **Application**: AIC is generally used when the primary goal is to find the model that best explains the data, regardless of its complexity. BIC is used when the goal is to find the simplest model that adequately explains the data.

In Gaussian Mixture Models

In the context of Gaussian Mixture Models, both AIC and BIC can be used to determine the optimal number of Gaussian components (clusters) in the model. By computing AIC and BIC for different numbers of components, you can select the model that strikes the best balance between goodness of fit and simplicity. This approach is particularly valuable when there's no clear prior knowledge about the number of clusters in the data.