UNIT IV

ENSEMBLE TECHNIQUES AND UNSUPERVISED LEARNING

Combining multiple learners: Model combination schemes, Voting, Ensemble Learning - bagging, boosting, stacking, Unsupervised learning: K-means, Instance Based Learning: KNN, Gaussian mixture models and Expectation maximization

PART A

1. **What is unsupervised learning?**
   Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets.
   These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

2. **What is semi supervised learning?**
   Semi-supervised machine learning is **a combination of supervised and unsupervised learning**.
   It uses a small amount of labeled data and a large amount of unlabeled data, which provides the benefits of both unsupervised and supervised learning while avoiding the challenges of finding a large amount of labeled data.

3. **What is Ensemble method?**
   Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model.
   The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine learning

4. **Explain Clustering.**
   Clustering is **the act of organizing similar objects into groups within a machine learning algorithm**. Assigning related objects into clusters is beneficial for AI models. Clustering has many uses in data science, like

image processing, knowledge discovery in data, unsupervised learning, and various other applications.

5. What is Cluster?

Cluster is a group of objects that belongs to the same class. In other words the similar objects are grouped in one cluster and dissimilar are grouped in other cluster.

6. What is bagging?

Bagging is also known as Bootstrap aggregation, ensemble method works by training multiple models independently and combining later to result in strong model.

7. Define Boosting.

Boosting refers to a group of algorithms that utilize weighted averages to make weak learning algorithms to stronger learning algorithms.

8. What is K-Nearest neighbor Methods?

K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

9. Which are the performance factors that influence KNN algorithm?

1. The distance function or distance metric used to determine the nearest neighbors
2. The Decision rule used to derive a classification from the K-Nearest neighbors.
3. The number of neighbors used to classify the new example.

10. What is K Means Clustering?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

11. List the properties of K-Means algorithm.

1. There are always K clusters
2. There is always at least one item in each cluster.
3. The clusters are non-hierarchical and they do not overlap

12. What is stacking?

Stacking, sometimes called stacked generalization, is an ensemble machine learning method that combines heterogeneous base or component models via a meta model.

13. How do GMMs differentiate from K-means clustering?

GMMs and K-means, both are clustering algorithms used for unsupervised learning tasks. However, the basic difference between them is that K-means is a distance-based clustering method while GMMs is a distribution based clustering method.

14. What is 'Over fitting' in Machine learning?

In machine learning, when a statistical model describes random error or noise instead of underlying relationship 'over fitting' occurs. When a model is excessively complex, over fitting is normally observed, because of having too many parameters with respect to the number of training data types. The model exhibits poor performance which has been over fit.

15. What is ensemble learning?

To solve a particular computational program, multiple models such as classifiers or experts are strategically generated and combined. This process is known as ensemble learning.

16. What are the two paradigms of ensemble methods?

The two paradigms of ensemble methods are

- Sequential ensemble methods
- Parallel ensemble methods

17. What is voting?

A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output.

18. What is Error-Correcting Output Codes?

The main classification task is defined in terms of a number of subtasks that are implemented by the base learners.

The idea is that the original task of separating oneclass from all other classes may be a difficult problem.

We want to define a set of simpler classification problems, each specializing in one aspect of the task, and combining these simpler classifiers, we get the final classifier.

$$y_i = \sum_{j=1}^{L} w_{ij} d_j$$

19. What is Gaussian Mixture models?

This model is a soft probabilistic clustering model that allows us to describe the membership of points to a set of clusters using a mixture of Gaussian densities.

20. Differentiate between Bagging and Boosting.

Sl no	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.

PART - B

1. Give Short notes on combining multiple learners.

Combining multiple learners is a models composed of multiple learners that complement each other so that by combining them, we attain higher accuracy.

Rationale

- ✓ In any application, we can use one of several learning algorithms, and with certain algorithms, there are hyper parameters that affect the final learner.
- ✓ For example, in a classification setting, we can use a parametric classifier or a multilayer perceptron, and, for example, with a multilayer perceptron, we should also decide on the number of hidden units.
- ✓ The **No Free Lunch Theorem** states that there is no single learning algorithm that in any domain always induces the most accurate learner. The usual approach is to try many and choose the one that performs the best on a separate validation set.
- ✓ Each learning algorithm dictates a certain model that comes with a set of assumptions. This inductive bias leads to error if the assumptions do not hold for the data.
- ✓ The performance of a learner may be fine-tuned to get the highest possible accuracy on a validation set, but this fine tuning is a complex task and still there are instances on which even the best learner is not accurate enough.
- ✓ Data fusion is the process fusing multiple records representing the same real world object into a single , consistent, and clean Representation
- ✓ Fusion of data for improving prediction accuracy and reliability is an important problem in machine learning
- ✓ Combining different models is done to improve the performance of deep learning models
- ✓ Building a new model by combining requires less time, data and computational resources
- ✓ The most common method to combine models is by averaging multiple models, where taking a weighted average improves the accuracy.

Generating Diverse Learners

Different Algorithms

We can use different learning algorithms to train different base-learners. Different algorithms make different assumptions about the data and lead to different classifiers.

Different Hyper parameters

We can use the same learning algorithm but use it with different hyper parameters.

Different Input Representations

Separate base-learners may be using different *representations* of the same input object or event, making it possible to integrate different types of Sensors/measurements/modalities.

Different representations make different characteristics explicit allowing better identification.

Different Training Sets

Another possibility is to train different base-learners by different subsets of the training set. This can be done randomly by drawing random training sets from the given sample this is called *bagging*.

Diversity vs. Accuracy

This implies that the required accuracy and diversity of the learners also depend on how their decisions are to be combined.

In a voting scheme, a learner is consulted for all inputs, it should be accurate everywhere and diversity should be enforced everywhere.

Model Combination Schemes

There are also different ways the multiple base-learners are combined to generate the final output:

Multiexpert combination

Multiexpert combination methods have base-learners that work in parallel. These methods can in turn be divided into two:

A) The global approach, also called learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking.

B) The local approach, or learner selection, for example, in mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.

Multistage combination methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough.

- ✓ Let us say that we have L base-learners. We denote by $dj(x)$ the prediction of base-learner Mj given the arbitrary dimensional input x

In the case of multiple representations,

- ✓ Each Mj uses a different input representation xj. The final prediction is calculated from the predictions of the baselearners:

$$Y= f(d1,d2,....,d_L|\Phi) \quad \textbf{Eq no 1}$$

- ✓ where $f(\cdot)$ is the combining function with Φ denoting its parameters.

When there are K outputs, for each learner there are $dji(x)$, $i = 1, \ldots, K$, $j = 1, \ldots, L$, and, combining them, we also generate K values, yi, $i = 1, \ldots, K$ and then for example in classification,we choose the class with the maximum yi value:

$$\text{Choose } C_i \text{ if } y_i = \max_{K=1}^{k} y_k$$

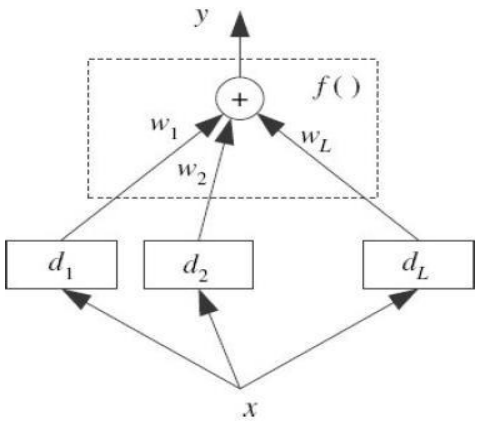

Figure 4.1 Base-learners are dj and their outputs are combined using $f(\cdot)$.

Voting

The simplest way to combine multiple classifiers is by *voting*, which corresponds to taking a linear combination of the learners (see figure 4.1)

$$y_i = \sum_j w_j d_{ji} \text{ where } w_j \geq 0, \sum_j w_j = 1$$

Eq no 02

This is also known as *ensembles* and *linear opinion pools*. In the simplest case, all learners are given equal weight and we have *simple voting* that corresponds to taking an average.

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j\, d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

Table 4.1 Classifier combination rules

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

Table 4.2 Example of combination rules on three learners and three classes

An example of the use of these rules is shown in table 4.2,

- ✓ which demonstrates the effects of different rules. Sum rule is the most intuitive and is the most widely used in practice.
- ✓ Median rule is more robust to outliers; minimum and maximum rules are pessimistic and optimistic, respectively.

In weighted sum, *dji* is the vote of learner *j* for class *Ci* and *wj* is the weight of its vote.

Simple voting is a special case where all voters have equal weight, namely, *wj* = 1/*L*. In classification, this is called *plurality voting* where the class having the maximum number of votes is the winner.

Voting schemes can be seen as approximations under a Bayesian framework with weights approximating prior model probabilities, and model decisions approximating model conditional likelihoods. This is *Bayesian model combination*

$$P(C_i|x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i|x, \mathcal{M}_j)P(\mathcal{M}_j)$$

Eq no 3

Let us assume that *dj* are iid with expected value *E*[*dj*] and variance Var(*dj*), then when we take a simple average with *wj* = 1/*L*, the expected value and variance of the output are

$$E[y] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} LE[d_j] = E[d_j]$$

$$\text{Var}(y) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2}\text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L\text{Var}(d_j) = \frac{1}{L}\text{Var}(d_j)$$

Eq no 4

- We see that the expected value does not change, so the bias does not change.
- But variance and therefore mean square error, decreases as the number of independent voters, *L*, increases.

$$\text{Var}(y) = \frac{1}{L^2}\text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \text{Var}(d_j) + 2\sum_j \sum_{i<j} \text{Cov}(d_j, d_i)\right]$$

Eq no 5

- which implies that if learners are positively correlated, variance (and error) increase.
- We can thus view using different algorithms and input features as efforts to decrease, if not completely eliminate, the positive correlation.

Error-Correcting Output Codes

- The main classification task is defined in terms of a number of subtasks that are implemented by the base learners.
- The idea is that the original task of separating one class from all other classes may be a difficult problem.
- We want to define a set of simpler classification problems, each specializing in one aspect of the task, and combining these simpler classifiers, we get the final classifier.
- Base-learners are binary classifiers having output −1/ + 1, and there is a *code matrix* W of *K* × *L* whose *K* rows are the binary codes of classes in terms of the *L* base-learners *dj*.

- Code Matrix W codes classes in terms of Learners One per class L=k

$$W = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

- The problem here is that if there is an error with one of the base-learners, there may be a misclassification because the class code words are so similar. So the approach in error correcting codes is to have $L > K$ and increase the Hamming distance between the code words.

- One possibility is *pairwise separation* of classes where there is a separate base-learner to separate *Ci* from *Cj*, for $i < j$
  In this case, $L = K(K-1)/2$ and with $K = 4$, the code matrix is

$$W = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

where a 0 entry denotes "don't care."

- With reasonable L, find W such that the hamming distance between rows and columns are maximized.
- ECOC can be written as a voting scheme where the entries of W, *wij*, are considered as vote weights:

$$y_i = \sum_{j=1}^{L} w_{ij} d_j$$

  and then we choose the class with the highest *yi*.
- One problem with ECOC is that because the code matrix Wis set a priori, there is no guarantee that the subtasks as defined by the columns of W will be simple.

2. Explain Ensemble learning Technique in detail.

- One of the most powerful machine learning techniques is ensemble learning. Ensemble learning is the use of multiple machine learning models to improve the reliability and accuracy of predictions.
- Put simply, ensemble learning is the process of training multiple machine learning models and combining their outputs together. The different models are used as a base to create one optimal predictive model.

- Simple ensemble learning techniques include things like averaging the outputs of different models, while there are also more complex methods and algorithms developed especially to combine the predictions of many base learners/models together.

Why Use Ensemble Training Methods?

- Machine learning models can be different from each other for a variety of reasons.
- Different machine learning models may operate on different samples of the population data, different modeling techniques may be used, and a different hypothesis might be used.

Simple Ensemble Training Methods

- Simple ensemble training methods typically just involve the application of statistical summary techniques, such as determining the mode, mean, or weighted average of a set of predictions.

Advanced Ensemble Training Methods

- There are three primary advanced ensemble training techniques, each of which is designed to deal with a specific type of machine learning problem.
- "Bagging" techniques are used to decrease the variance of a model's predictions, with variance referring to how much the outcome of predictions differs when based on the same observation.
- "Boosting" techniques are used to combat the bias of models.
- Finally, "stacking" is used to improve predictions in general.

Ensemble learning methods can be divided into one of two different groups:

1. sequential methods

- Sequential ensemble methods get the name "sequential" because the base learners/models are generated sequentially.
- In the case of sequential methods, the essential idea is that the dependence between the base learners is exploited in order to get more accurate predictions.
- Examples of sequential ensemble methods include AdaBoost, XGBoost, and Gradient tree boosting.

2. parallel ensemble methods

- parallel ensemble methods generate the base learners in parallel.
- When carrying out parallel ensemble learning, the idea is to exploit the fact that the base learners have independence, as the general error rate can be reduced by averaging the predictions of the individual learners.

3. Explain in Detail about Bagging Technique in Ensemble Learning.

- **Bagging**: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.
- *Bagging* is a voting method whereby base-learners are made different by training them over slightly different training sets.
- Ensemble learning helps improve machine learning results by combining several models.
- This approach allows the production of better predictive performance compared to a single model.
- Basic idea is to learn a set of classifiers (experts) and to allow them to vote. **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models.
- So the result may be a model with higher stability.
- Bootstrap Aggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.
- It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.

Pseudocode

1. Given training data(x1,y1),….,(xm,ym)
2. For t=1,…..,T:
   a. From bootstrap replicate dataset St by selecting m random examples from the training set with replacement.
   b. Let ht be the result of training base learning algorithm on si
3. Output combined classifier:

H(x)=majority(h1(x),….,hT(x)).

Implementation Steps of Bagging

Step 1: Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.

Step 2: A base model is created on each of these subsets.

Step 3: Each model is learned in parallel with each training set and independent of each other.

Step 4: The final predictions are determined by combining the predictions from all the models

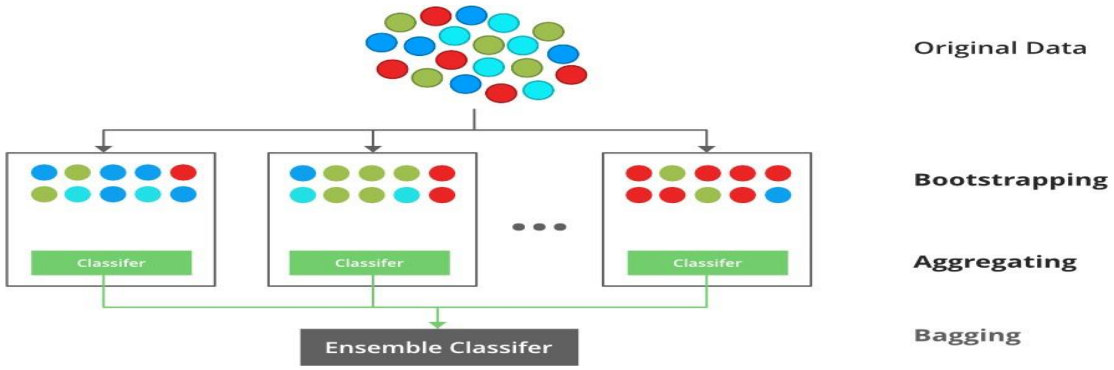

Figure 4.2 Bagging Technique.

Advantages:

1. Reduce overfitting of the model.
2. Handles higher dimensionality data very well.
3. Maintains accuracy for missing data

Disadvantage:

Since final prediction is based on mean prediction from the subset trees, it won't give precise values for the classification and regression model

4. Explain boosting Technique in Ensemble learning.

- In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method.
- In boosting, we actively try to generate complementary base learners

  by training the next learner on the mistakes of the previous learners.
- The original *boosting* algorithm combines three weak learners to generate a strong learner. A *weak learner* has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a *strong learner* has arbitrarily small error probability.
- Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers.
- It is done by building a model by using weak models in series. Firstly, a model is built from the training data.
- Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

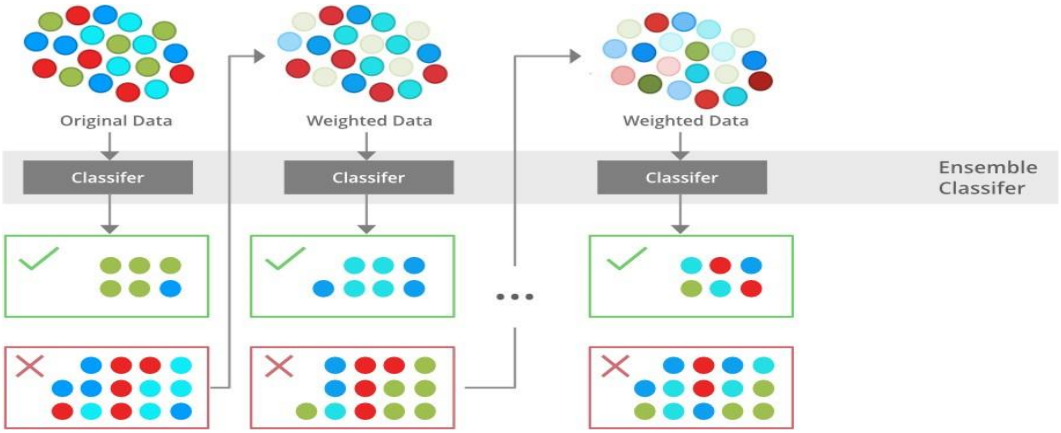

Figure 4.3 An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted dataset.

- AdaBoost can also combine an arbitrary number of base learners, not three.
- Many variants of AdaBoost have been proposed; here, we discuss the original algorithm AdaBoost.
- In AdaBoost, although different base-learners have slightly different training sets, this difference is not left to chance as in bagging, but is a function of the error of the previous baselearner.
- The actual performance of boosting on a particular problem is clearly dependent on the data and the base-learner.
- There should be enough training data and the base-learner

should be weak but not too weak, and boosting is especially susceptible to noise and outliers.

Training:
For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$
For all base-learners $j = 1, \ldots, L$
Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t
Train d_j using $\mathcal{X}_j$
For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$
Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$
If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop
$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$
For each (x^t, r^t), decrease probabilities if correct:
If $y_j^t = r^t$, then $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$
Normalize probabilities:
$Z_j \leftarrow \sum_t p_{j+1}^t$; $p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$
Testing:
Given x, calculate $d_j(x), j = 1, \ldots, L$
Calculate class outputs, $i = 1, \ldots, K$:
$y_i = \sum_{j=1}^L \left(\log \frac{1}{\beta_j}\right) d_{ji}(x)$

Similarities Between Bagging and Boosting

Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

Differences between Bagging and Boosting

Sl no	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built	New models are influenced

Sl no	Bagging	Boosting
	independently.	by the performance of previously built models.

5. Explain in Detail about Stacking.

- Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance.
- Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance.
- Stacking is a way of ensembling classification or regression models it consists of two-layer estimators.
- The first layer consists of all the baseline models that are used to predict the outputs on the test datasets.
- The second layer consists of Meta-Classifier or Regressor which takes all the predictions of baseline models as an input and generate new predictions.

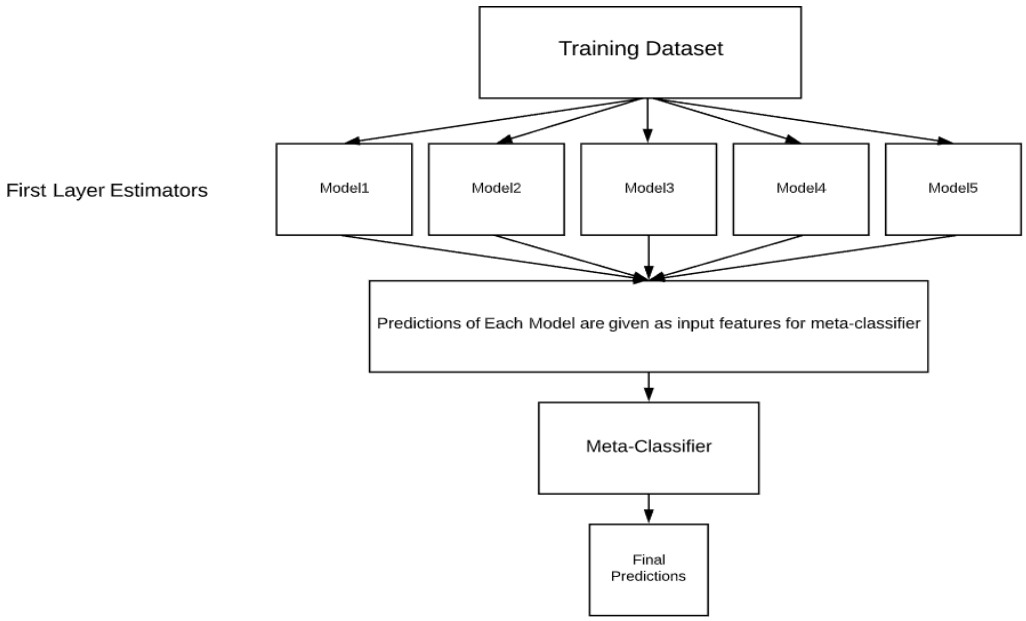

Figure 4.3 Stacking Architecture

Steps to implement Stacking models:

- Split training data sets into n-folds using the **Repeated Stratified KFold** as this is the most common approach to preparing training datasets for meta-models.

- Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
- The prediction made in the above step is added to the x1_train list.
- Repeat steps 2 & 3 for remaining n-1folds, so it will give x1_train array of size n,
- Now, the model is trained on all the n parts, which will make predictions for the sample data.
- Add this prediction to the y1_test list.
- In the same way, we can find x2_train, y2_test, x3_train, and y3_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
- Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model (refer Figure 4.3).
- Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

6. Explain in detail about Unsupervised Learning

- In supervised learning, the aim is to learn a mapping from the input to an output whose correct values are provided by a supervisor. In unsupervised learning, there is no such supervisor and we have only input data.
- The aim is to find the regularities in the input. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not. In statistics, this is called ***density estimation***
- One method for density estimation is ***clustering*****,** where the aim is to find clusters or groupings of input.
- Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.

Clustering

- Given a set of objects, place them in a group such that the objects in a group are similar to one another and different from the objects in other groups
- Cluster analysis can be a powerful data-mining tool for any organization.
- Cluster is a group of objects that belongs to the same class
- Clustering is a process of partitioning a set of data in a meaningful subclass.

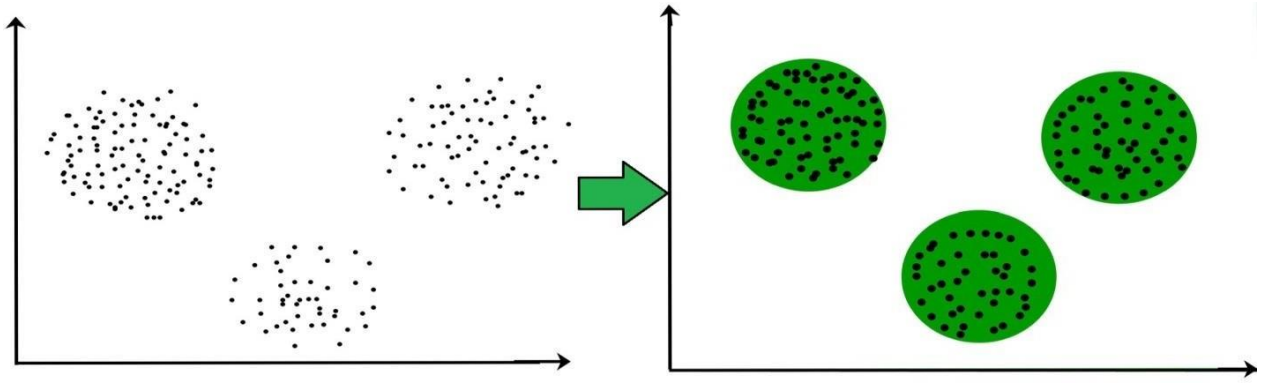

Figure 4.4 Clustering

Clustering Methods :

- **Density-Based Methods:** These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters. Example *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)*, *OPTICS (Ordering Points to Identify Clustering Structure)*, etc.
- **Hierarchical Based Methods:** The clusters formed in this method form a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category
  - **Agglomerative** (bottom-up *approach*)
  - **Divisive** (top-down *approach*

Unsupervised Learning : K means

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.
- It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training as in Figure 4.4.
- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

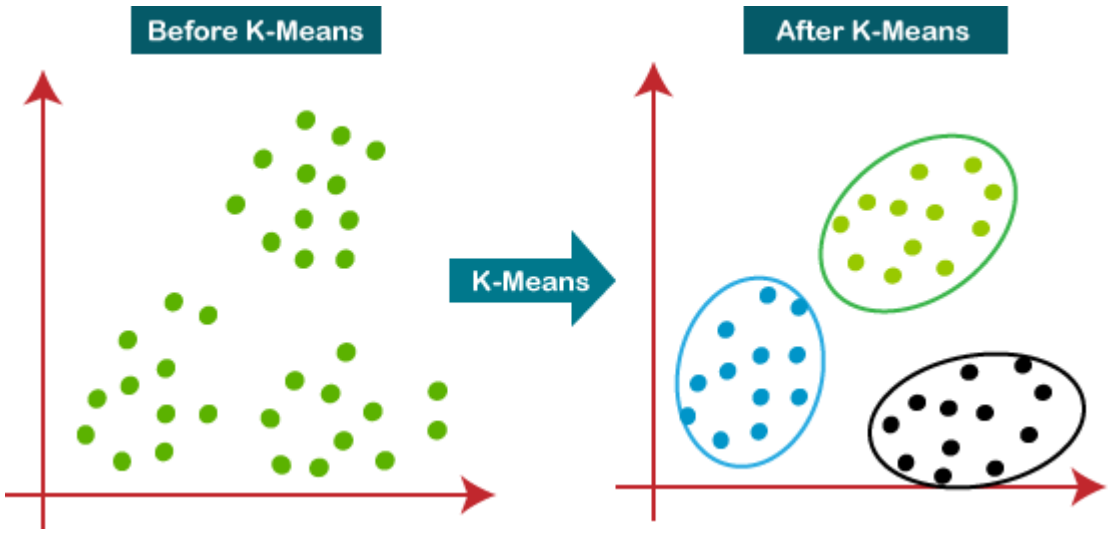

Figure 4.5 Explains the working of the K-means Clustering Algorithm

How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Instance based learning:KNN

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

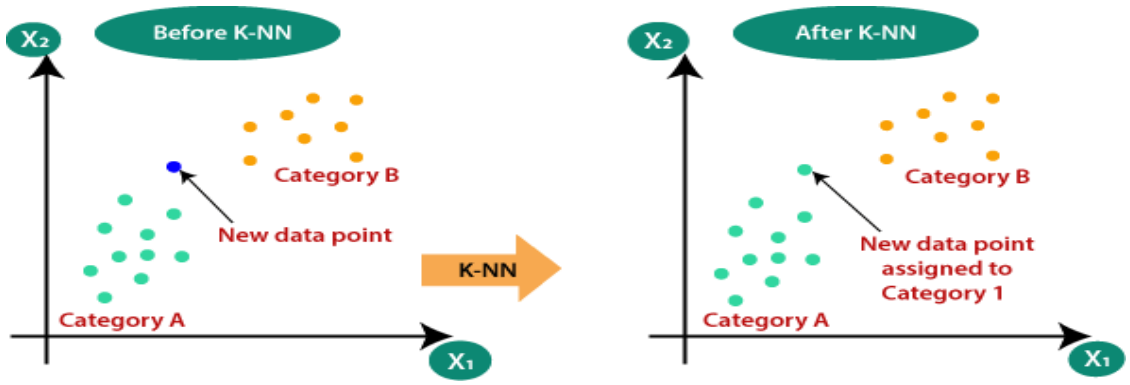

Figure 4.6 Explains the working of the K-NN Algorithm

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

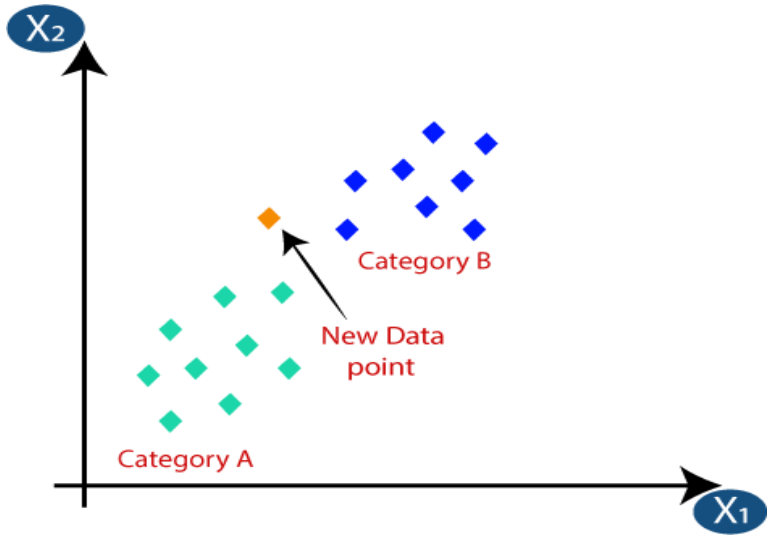

Figure 4.7 Suppose we have a new data point and we need to put it in the required category.

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

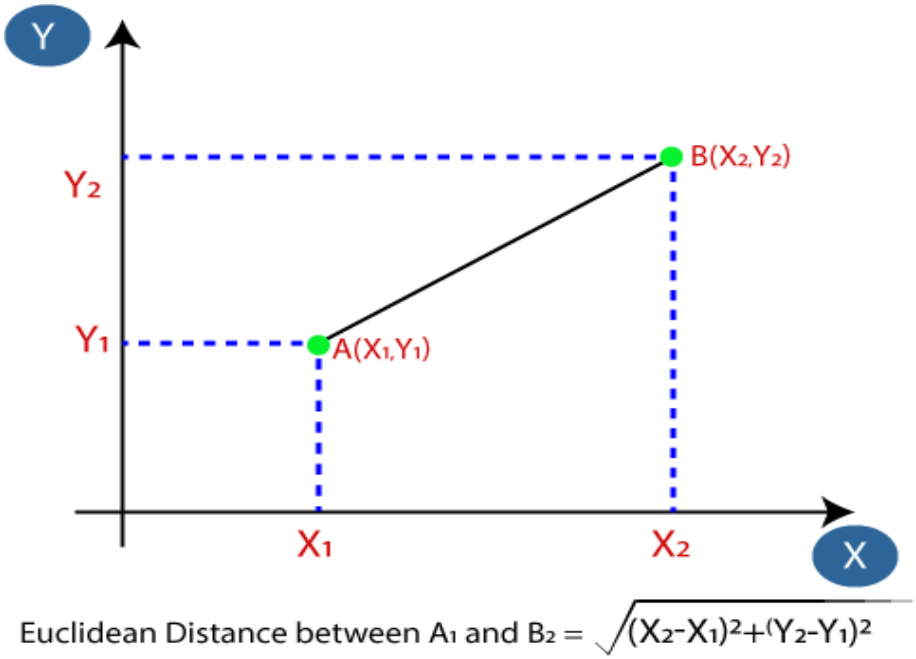

Figure 4.8 Euclidean distance

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

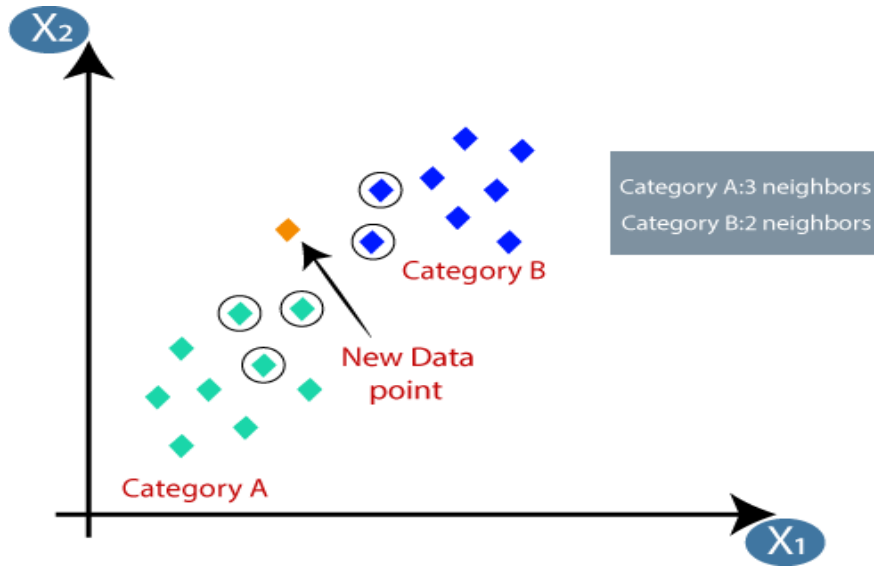

Figure 4.9 nearest neighbors

- As we can see the 3 nearest neighbors are from category A in Figure 4.9 , hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

7. Explain in detail about Gaussian Mixture models and Expectation Maximization.

Gaussian Mixture Model

- This model is a soft probabilistic clustering model that allows us to describe the membership of points to a set of clusters using a mixture of Gaussian densities.
- It is a soft classification (in contrast to a hard one) because it assigns probabilities of belonging to a specific class instead of a definitive choice. In essence, each observation will belong to every class but with different probabilities.
- While Gaussian mixture models are more flexible, they can be more difficult to train than K-means. K-means is typically faster to converge and so may be preferred in cases where the runtime is an important consideration.
- In general, K-means will be faster and more accurate when the data set is large and the clusters are well-separated. Gaussian mixture models will be more accurate when the data set is small or the clusters are not well-separated.
- Gaussian mixture models take into account the variance of the data, whereas K-means does not.
- Gaussian mixture models are more flexible in terms of the shape of the clusters, whereas K-means is limited to spherical clusters.
- Gaussian mixture models can handle missing data, whereas K-means cannot. This difference can make Gaussian mixture models more effective in certain applications, such as data with a lot of noise or data that is not well-defined.
- The mixture model where we write the density as a weighted sum of component densities.

$$p(\boldsymbol{x}) = \sum_{i-1}^{k} P(G_i)\, p(\boldsymbol{x} | G_i)$$

- Where $P(Gi)$ are the mixture proportions and $p(x|Gi)$ are the component densities.

- For example, in Gaussian mixtures, we have $p(x|Gi) \sim N(\mu i, \Sigma i)$, and defining $\pi i \equiv P(Gi)$, we have the parameter vector as

$$\Phi = \{\pi_i, \mu_i, \Sigma_i\}_{i-1}^{k}$$

that we need to learn from data.

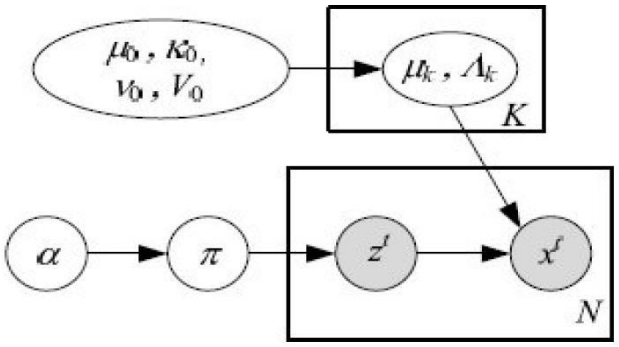

Figure 4.10 The generative graphical representation of a Gaussian mixture model.

The EM algorithm that is a maximum likelihood procedure:

$$\Phi_{MLE} = \arg\max_{\Phi} \log p(\mathcal{X}|\Phi)$$

If we have a prior distribution $p(\Phi)$, we can devise a Bayesian approach. For example, the MAP estimator is

$$\Phi_{MAP} = \arg\max_{\Phi} \log p(\Phi|\mathcal{X}) = \arg\max_{\Phi} \log p(\mathcal{X}|\Phi) + \log p(\Phi)$$

The mean and precision (inverse covariance) matrix, we can use a normal-Wishart prior

$$\begin{aligned} p(\Phi) &= p(\boldsymbol{\pi}) \prod_i p(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) \\ &= \text{Dirichlet}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_i \text{normal-Wishart}(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \mathbf{V}_0) \end{aligned}$$

Expectation-Maximization Algorithm

- In *k*-mean, clustering is the problem of finding codebook vectors that minimize the total reconstruction error.
- Here the approach is probabilistic and we look for the component density parameters that maximize the likelihood of the sample.
- Using the mixture model of equation , the log likelihood given the sample $X = \{xt\}t$ is

$$\begin{aligned} \mathcal{L}(\Phi|\mathcal{X}) &= \log \prod_t p(\mathbf{x}^t|\Phi) \\ &= \sum_t \log \sum_{i-1}^{k} p(\mathbf{x}^t|G_i) P(G_i) \end{aligned}$$

- Where Φ includes the priors $P(Gi)$ and also the sufficient statistics of the component densities $p(xt|Gi)$.

- Unfortunately, we cannot solve for the parameters analytically and need to iterative optimization.
- The *expectation-maximization* algorithm (Dempster, Laird, and Rubin 1977; Redner and Walker 1984) is used in maximum likelihood estimation where the problem involves
- Two sets of random variables of which one, *X*, is observable and the other, *Z*, is hidden.
- The goal of the algorithm is to find the parameter vector Φ that maximizes the likelihood of the observed values of *X*, $L(\Phi|X)$.
- But in cases where this is not feasible, we associate the extra *hidden variables Z* and express the underlying model using both, to maximize the likelihood of the joint distribution of *X* and *Z*, the *complete* likelihood $Lc(\Phi|X, Z)$.
- Since the *Z* values are not observed, we cannot work directly with the complete data likelihood *Lc*; instead, we work with its expectation, *Q*, *X* and the current parameter values Φl, where *l* indexes iteration.
- This is the *expectation* (E) step of the algorithm. Then in the *maximization* (M) step, we look for the new parameter values, $\Phi l+1$, that maximize this. Thus

$$\text{E-step} \quad : \quad Q(\Phi|\Phi^l) = E[\mathcal{L}_c(\Phi|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \Phi^l]$$
$$\text{M-step} \quad : \quad \Phi^{l+1} = \arg\max_{\Phi} Q(\Phi|\Phi^l)$$

- In the E-step we estimate these labels given our current knowledge of components, and in the M-step we update our component knowledge given the labels estimated in the E-step.
- These two steps are the same as the two steps of *k*-means; calculation of (E-step) and re-estimation of *mi* (Mstep).