Machine Learning W12 Tutorial

COMP30027 | Sandy Luo

Please spend 10-15 mins completing:

- FEIT Small Classes survey
- ESS survey

Overview

GMM

Concept, E-M steps

Semi-supervised Learning

Concept, code

01

Concept

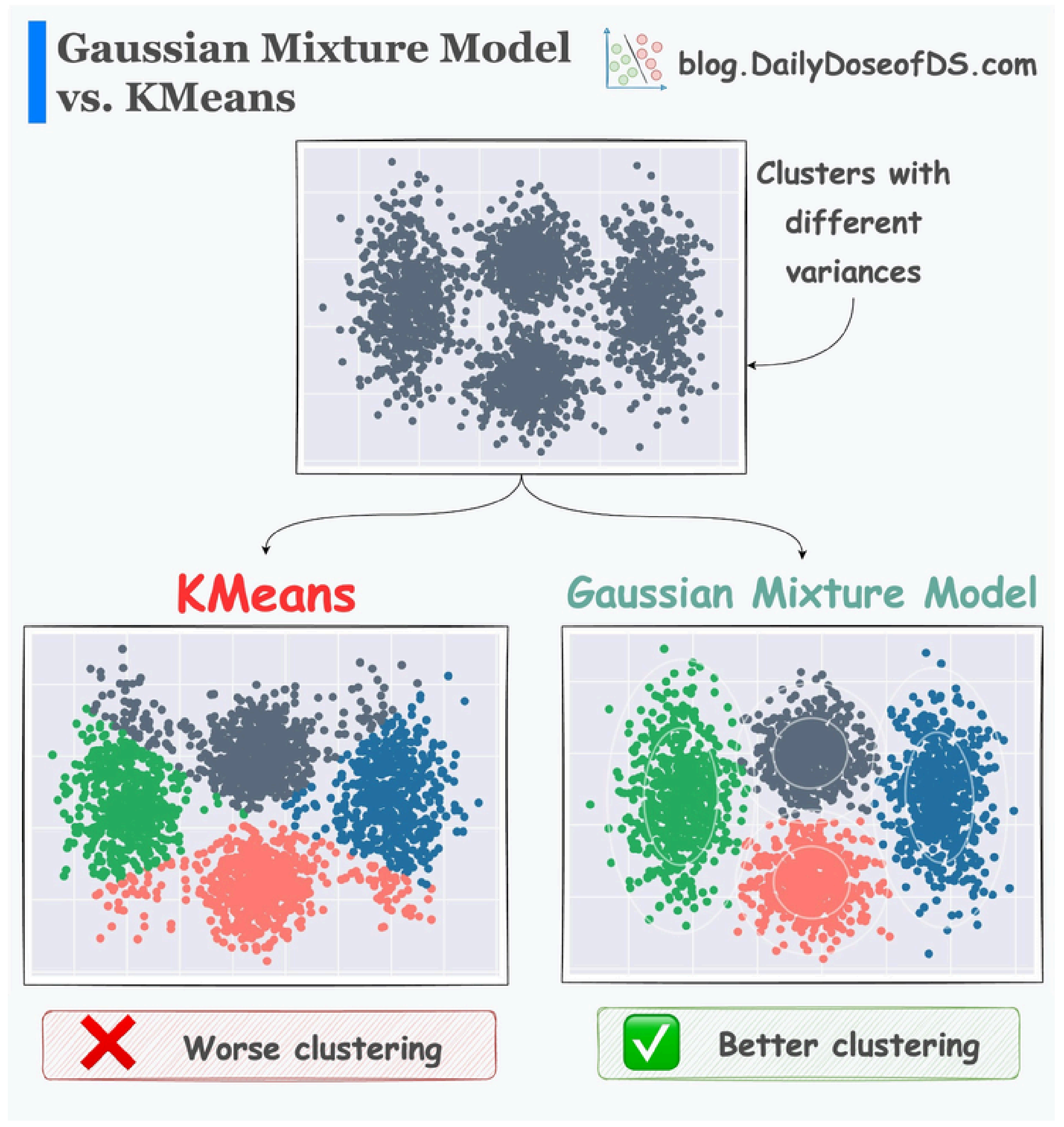
02

Expectation Maximisation

Gaussian Mixture Model

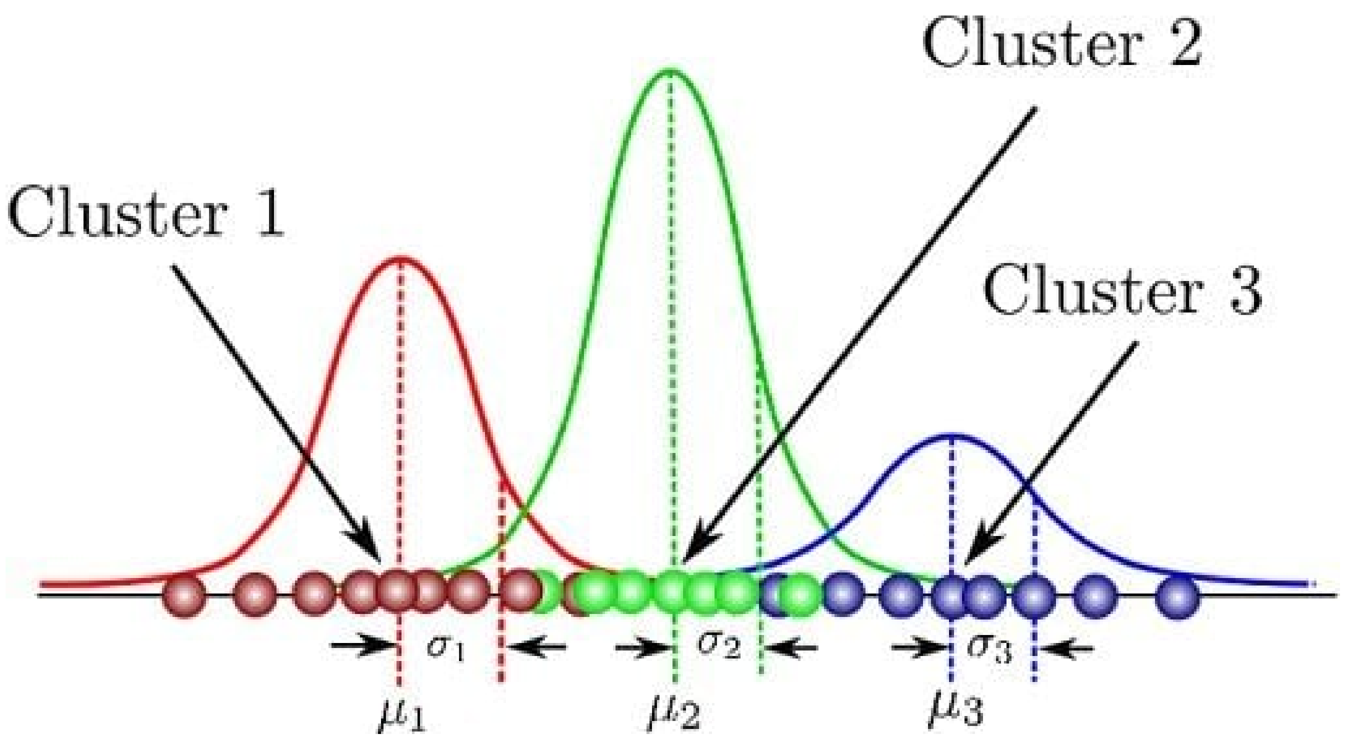
Gaussian Mixture Model

- Generalisation of soft k-means
  - Soft k-means is a probabilistic version of k-means
  - Treats each instance is a member of all clusters with some probability (softmax)



Gaussian Mixture Model

- Represents a distribution as composed of *k* Gaussian distributions



Expectation Maximisation

- Parameter estimation iterative algorithm
    - Guaranteed “positive” hill-climbing characteristics
    - Used to estimate (hidden) parameter values, e.g. cluster membership

- Log likelihood gives an estimate of how good a cluster model is
    - Guaranteed to increase on each iteration of the EM iteration
    - If the change in log likelihood falls below a predefined value, we consider the estimate to have converged

Q1(a):

What is the logic behind the EM algorithm, when used for clustering?

- **Explain the significance of the "E" step, and the "M" step.**

Q1(a):

What is the logic behind the EM algorithm, when used for clustering?

- Explain the significance of the "E" step, and the "M" step.

- EM:
    - Iterative algorithm used in unsupervised ML tasks such as clustering
    - Start with a random / uniform guess, then progressively improve guess by evaluating the expected likelihood on the given data.
- Idea: estimate parameters of a statistical model when part of data is missing
    - For clustering, missing data refers to the “class” of data points

Q1(a):

What is the logic behind the EM algorithm, when used for clustering?

- Explain the significance of the "E" step, and the "M" step.

1. Expectation step:
   - Compute the probabilities of each data point belonging to each cluster based on the current estimate of the cluster parameters.
   - Assign weighted labels to the training data → calculate the log-likelihood function

Q1(a):

What is the logic behind the EM algorithm, when used for clustering?

- Explain the significance of the "E" step, and the "M" step.

2. Maximisation step:

- Updates the estimates of the model parameters based on the expected values of the missing data computed in the E-step.
- For clustering, update the re-estimates of the cluster parameters based on the expected number of data points assigned to each cluster.

Q1(b):

What is the logic behind the EM algorithm, when used for clustering?

- **What happens in the "E" and "M" steps of the GMM method?**

Q1(b):

What is the logic behind the EM algorithm, when used for clustering?

- **What happens in the "E" and "M" steps of the GMM method?**

- **In the GMM method, each cluster is modeled as a Gaussian distribution** w/:
  - Mean (μ): centre of the cluster
  - Covariance matrix (Σ): shape, orientation, and spread of the cluster
- **The probability of a data point belonging to a cluster = probability density of the point under the corresponding Gaussian distribution.**

Q1(b):

What is the logic behind the EM algorithm, when used for clustering?

- **What happens in the "E" and "M" steps of the GMM method?**

1. <u>Expectation</u> step in GMM method:

- **Estimate how much each distribution is responsible for generating each point**
    - The expected no. of points assigned to each cluster based on current estimates
- For each data point, compute the probability of the point belonging to each cluster, given current estimates of the cluster parameters (mean & sd).
    - AKA ***responsibility*** (gamma) of the cluster for that point, computed w/ Bayes' rule

Q1(b):

What is the logic behind the EM algorithm, when used for clustering?

- **What happens in the "E" and "M" steps of the GMM method?**

2. <u>Maximisation</u> step:

- **Updates the estimates of the cluster parameters based on the *responsibilities***
  - Maximises the likelihood of the data given the expected no. of points in clusters
- For each cluster, computes the new estimate of the cluster parameters (mean, sd) to improve the log-likelihood of the data given the cluster assignments.

Q1(b):

What is the logic behind the EM algorithm, when used for clustering?

- What happens in the "E" and "M" steps of the GMM method?

- **Keep iterating b/w “Expectation” and “Maximisation” steps until convergence**
  - e.g., the log likelihood cannot be further improved

- **Final estimates of cluster parameters are used to assign each data point to a cluster**

01

Self-training

02

Active Learning

Semi-Supervised Learning

Q2(a):

What is the main assumption of self-training?

Self-Training

- **Method to improve model with unlabelled data**
- Assume we have:
    a. Trained model via supervised learning, *M*
    b. Unlabelled data

1. Find instances in the unlabelled data that *M* can confidently label
2. Label those instances with *M*
3. Treat the now-labelled instances as additional training data

Q2(a):

What is the main assumption of self-training?

Assumption:

- **Similar instances are likely to have the same label**
    - *“Find instances in the unlabelled data that M can* ***confidently*** *label”*
    - **Common measure of “confident” = similarity of data > defined threshold**

- Find very similar unlabelled instances to our labelled data, assign the same label, then add them to the "labelled" training dataset

Q2(b):

What is the main assumption of active learning?

Active Learning

- **Method to improve model with more labelled data (effectively)**
- Assume we have:
    a. Trained model via supervised learning, *M*
    b. Unlabelled data
    c. Human annotators to label new data, *oracles*

1. Find instances in the unlabelled data that *M* is the least sure about
2. Ask humans to label those data
3. Treat the now-labelled instances as additional training data

Q2(b):

What is the main assumption of active learning?

Assumption:

- **Instances that the model is the most unsure about are the most informative**
    - Reason why we focus on instances that we are most uncertain about
    - Assumes that having correct labels for these instances will be most helpful for learning the correct class boundaries

Q3(a):

Describe the rationale and key principles behind the query-by-committee (QBC) algorithm.

Q3(a):

Describe the rationale and key principles behind the query-by-committee (QBC) algorithm.

QBC:

- More complex strategy for when there's multiple classifiers
- **Train multiple classifiers on a labelled dataset, use each to predict on unlabelled data, and select instances w/ highest disagreement b/w classifiers**
- Diagreement can be measured by entropy

Q3(a):

Describe the rationale and key principles behind the query-by-committee (QBC) algorithm.

- **Rationale: diversity in the committee is crucial for achieving better accuracy**
  - Assume that all classifiers learn smth different, so can provide different info
- QBC uses an equation, which captures vote entropy, to determine the instance that our active learner would select first:

$$x_{VE} = \underset{x}{\operatorname{argmax}}\left(-\sum_{y_i} \frac{V(y_i)}{C} log_2 \frac{V(y_i)}{C}\right)$$

- x: an instance
- y_i: set of possible class labels
- V(y_i): number of votes a given label receives
- C: number of classifiers.

Q3(b):

$$x_{VE} = \underset{x}{\operatorname{argmax}}\left(-\sum_{y_i} \frac{V(y_i)}{C} log_2 \frac{V(y_i)}{C}\right)$$

The table below shows the predicted class labels (A, B, or C) from four classifiers (C1 - C4) for three instances (1-3). Use QBC to determine the instance that an active learner would select first in this scenario.

Instance	C1 pred	C2 pred	C3 pred	C4 pred
1	B	C	A	B
2	B	B	B	B
3	A	C	A	C

Q3(b):

$$x_{VE} = \underset{x}{\operatorname{argmax}}\left(-\sum_{y_i} \frac{V(y_i)}{C} log_2 \frac{V(y_i)}{C}\right)$$

Calculate the vote entropy for each instance:

Instance	C1 pred	C2 pred	C3 pred	C4 pred
1	B	C	A	B
2	B	B	B	B
3	A	C	A	C

Q3(b):

$$x_{VE} = \underset{x}{\operatorname{argmax}}\left(-\sum_{y_i} \frac{V(y_i)}{C} log_2 \frac{V(y_i)}{C}\right)$$

Calculate the vote entropy for each instance:

Instance	C1 pred	C2 pred	C3 pred	C4 pred
1	B	C	A	B
2	B	B	B	B
3	A	C	A	C

Instance 1:

- - ((1/4)*log_2(1/4) + (2/4)*log_2(2/4) + (1/4)*log_2(1/4)) = 1.5

Instance 2:

- - ((4/4)*log_2(4/4) + (0/4)*log_2(0/4) + (0/4)*log_2(0/4)) = 0

Instance 3:

- - ((2/4)*log_2(2/4) + (0/4)*log_2(0/4) + (2/4)*log_2(2/4)) = 1

Q4:

Consider a naive Bayes model trained using the following familiar weather dataset:

ID	Outlook	Temp	Humid	Wind	PLAY
A	S	H	N	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

Suppose that you made additional observations of days and their features. But you don't have the label for the PLAY in these days:

ID	Outlook	Temp	Humid	Wind	PLAY
G	O	M	N	T	?
H	S	M	H	F	?

How could you incorporate this information into your naive Bayes model without manually annotating the labels?

Q4:

Consider a naive Bayes model trained using the following familiar weather dataset:

ID	Outlook	Temp	Humid	Wind	PLAY
A	S	H	N	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

More unlabelled data,

no manually labelling

→ self training!

Suppose that you made additional observations of days and their features. But you don't have the label for the PLAY in these days:

ID	Outlook	Temp	Humid	Wind	PLAY
G	O	M	N	T	?
H	S	M	H	F	?

How could you incorporate this information into your naive Bayes model without manually annotating the labels?

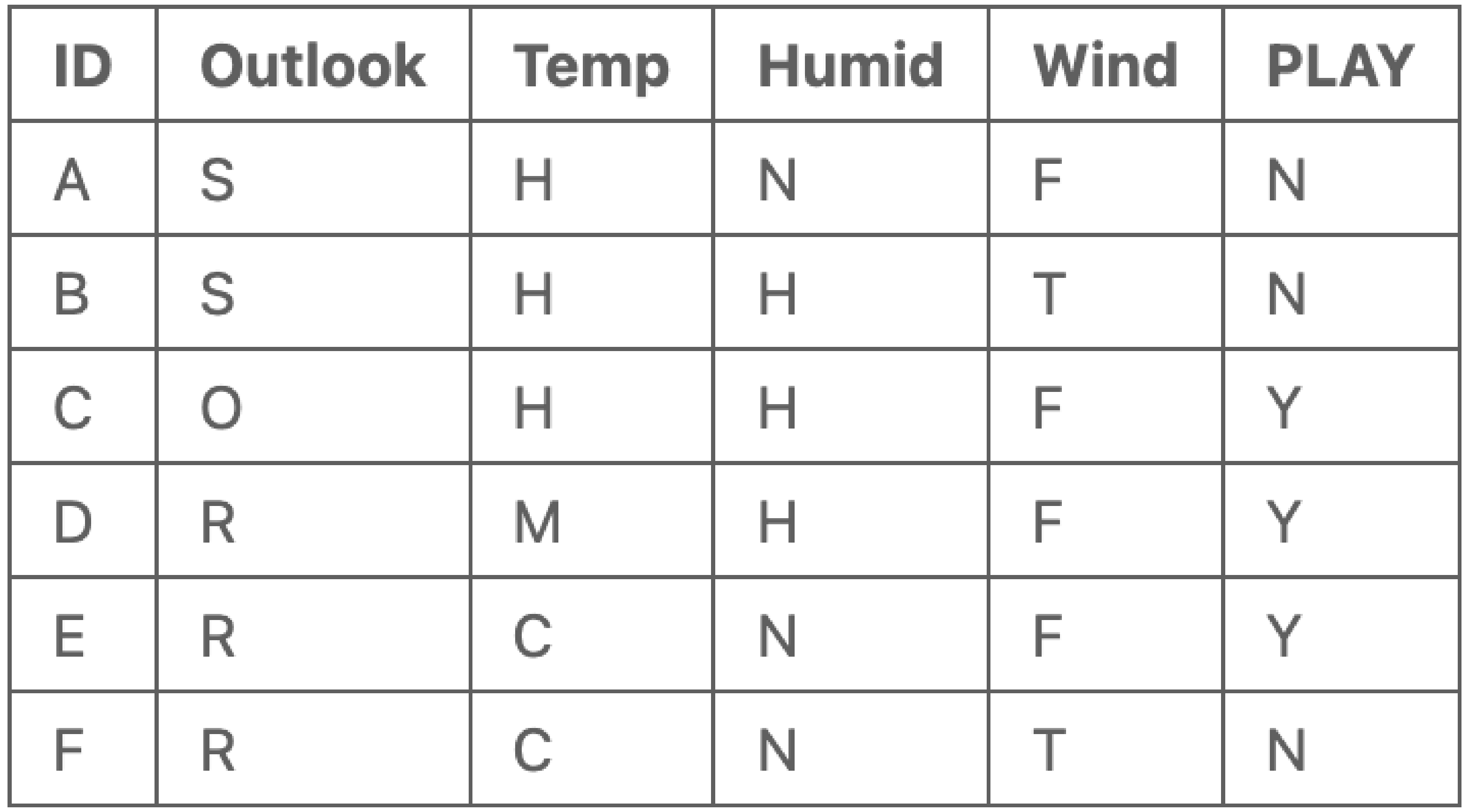
Q4:

Self-training steps:

1. Train the learner on the currently labelled instances
2. Use the learner to predict the labels of the unlabeled instances
3. Where the learner is very confident, add newly labelled instances to the training set
4. Repeat until all instances are labelled, or no new instances can be labelled confidently

Q4: Step 1

Example: Train a naive Bayes classifer using Laplace smoothing (alpha = 1)



ID	Outlook	Temp	Humid	Wind	PLAY
A	S	H	N	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

Q4: Step 2

Use the learner to predict the labels of the unlabeled instances

Instance G

$$N: P(N) \times P(Outlook = O|N) \times P(Temp = M|N) \times P(Humid = N|N) \times P(Wind = T|N)$$
$$= \frac{1}{2}(\frac{1}{6})(\frac{1}{6})(\frac{3}{5})(\frac{3}{5}) = 0.005$$
$$Y: P(Y) \times P(Outlook = O|Y) \times P(Temp = M|Y) \times P(Humid = N|Y) \times P(Wind = T|Y)$$
$$= \frac{1}{2}(\frac{2}{6})(\frac{2}{6})(\frac{2}{5})(\frac{1}{5}) = 0.004$$

Instance G will be classified as N.

Instance H

$$N: P(N) \times P(Outlook = S|N) \times P(Temp = M|N) \times P(Humid = H|N) \times P(Wind = F|N)$$
$$= \frac{1}{2}(\frac{3}{6})(\frac{1}{6})(\frac{2}{5})(\frac{2}{5}) = 0.007$$
$$Y: P(Y) \times P(Outlook = S|Y) \times P(Temp = M|Y) \times P(Humid = H|Y) \times P(Wind = F|Y)$$
$$= \frac{1}{2}(\frac{1}{6})(\frac{2}{6})(\frac{3}{5})(\frac{4}{5}) = 0.013$$

Instance H will be classified as Y.

Q4: Step 3

- Add confidently-classified instances to the training dataset
  - Naive Bayes → can use the posterior probability of the maximum class, ratio of probabilities (in a binary task) or entropy over class label as a measure of "confidence."
  - e.g. could define a "confident" classification as a classification with posterior probability > 0.01.

- Using this definition, instance H has been confidently classified and should be added to the training dataset with the label Y.

Q4: Step 4

- Repeat until all instances are labelled, or no new instances can be labelled confidently

- Retrain the NB model using the new training dataset (A-F + H)
    - With the new model, instance G will be classified as follows:

$$N: \frac{3}{7}(\frac{1}{6})(\frac{1}{6})(\frac{3}{5})(\frac{3}{5}) = 0.0042$$

$$Y: \frac{3}{7}(\frac{2}{7})(\frac{3}{7})(\frac{2}{6})(\frac{1}{6}) = 0.0038$$

- Less than threshold (<0.01), instance G still cannot be confidently classified, so the self-training algorithm will terminate at this iteration.