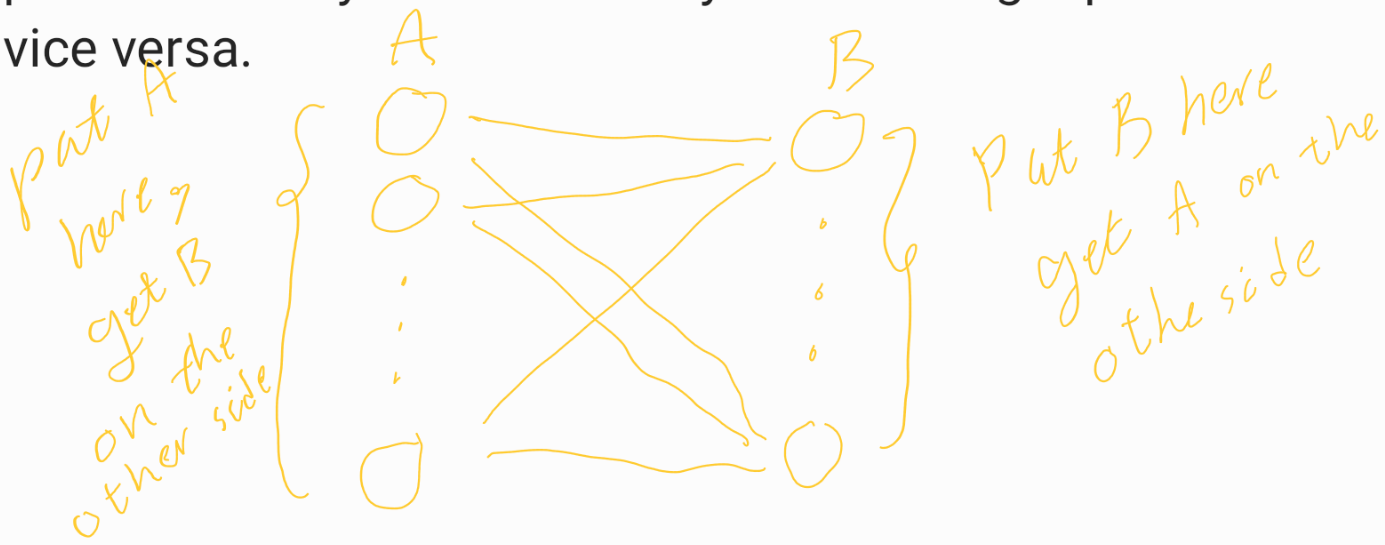
BAM: Bidirectional Associative Memory الف - ا

Is a type of associative memory which can relate a pair of patterns to each other. This means for example if it learnt to pair pattern A and B, if you present Pattern A, BAM will recall Pattern B and If you present pattern B, BAM will recall Pattern A. It also is robust to some level noise, so you may present a noisy Pattern A but you can still get pattern B and vice versa.


A
B
put A here, get B on the other side
Put B here get A on the other side

There is no self loop and weights are symmetric.

ب-۱

They are both examples of associative memories, from structure point of view, BSB is an Hopfiled network with self connection and assymetric weights.

From learning point of view, both try to minimze an energy function by learning the input pattern in an iterative way using Hebb learning rule.
But BSB has fewer memories and since it pushes the memories into corners of it box, they are more stable than Hopfiled stored memories.

From use case point of view, Hopfiled is useful for content-addressable memory but BSB which has clustering abilities is used for data representation and concept formation.

ج-۱

The problem with Hopfiled is it can stuck in local minima, boltzman tried to solve this issue by using simulated annealing which is technique to use high randomness at the beginning of the training and gradually reduce the randomness during the training. This way we can get out from local minima and find lower energy point in solution space.

الف-3

In RCE each input point draws a circle around itself, so having two circle with same center from same class means that the input examples responsible for generating those circles are actually the same. So it is not the case to have circles with same centers and same labels.

ب-3

Having circles with same centers from different class means that there are some input examples that have the exact same feature vector but different label. This is a case of inherent error (maybe inputs have wrong labels or few features). If our dataset has this type of error, we need to introduce a method to resolve this issue during training, other wise the training would not converge, because let's say we created the first circle, the next training data will come, now we have reduce the first sample radius to 0 (since it should not contains the second sample) so this is an issue which neads special attention if it happens during training.

ج-3

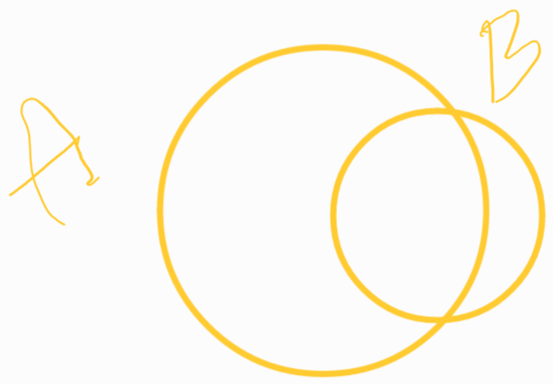
Yes this can happen, let's say we have two points with distance D between them, if we consider the initial circle radios to be D/2 then by introducing those sample inputs during training we have two circles that are touched.

د-3

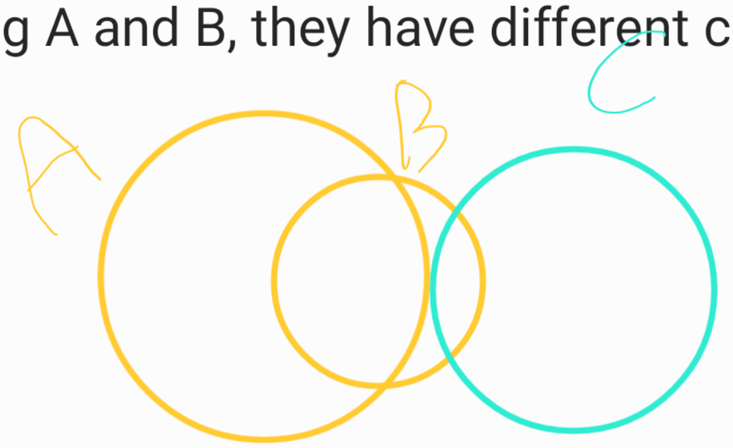
Like the previous example, this can happen two but this time we have different labels.

It can not happen if two circles are from different classes, because they adjust their radius to not contain samples from opposing classes. In case of same class circles, we can think of it in this way, let say we have three circle A,B,C. A and B have the same class but C is different.

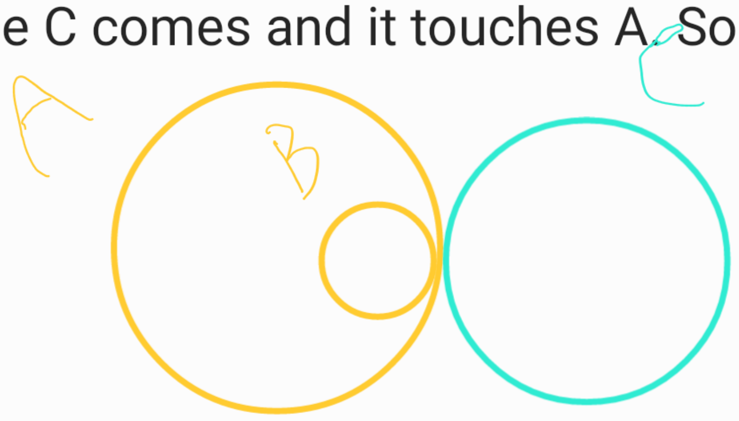
Q-3



Drawing A and B, they have different centers.



Sample C comes and it touches A. So B has to reduce radius.

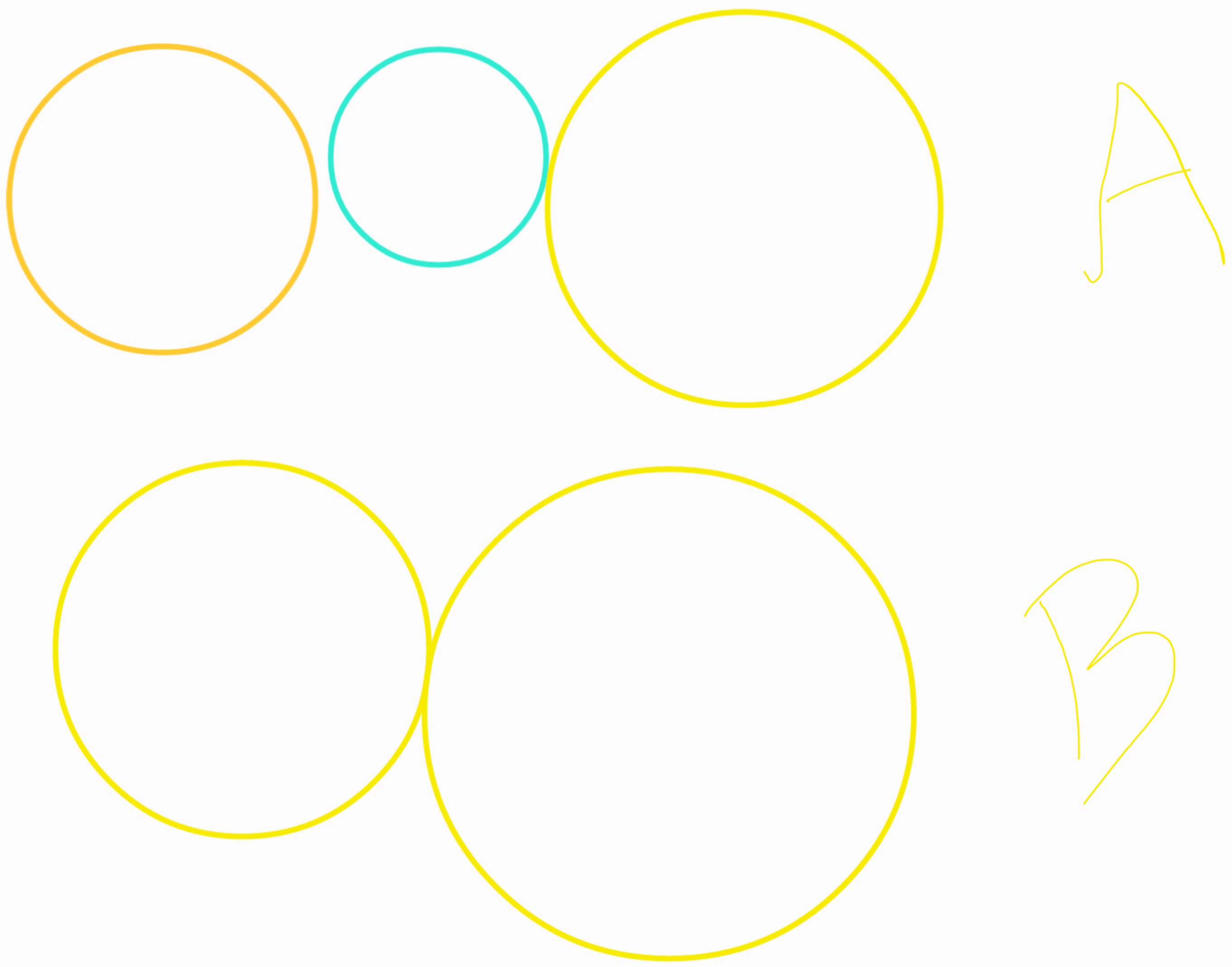


Now B is inside A.

In art network, noisy training set can cause problem because it may result in low similarity between same class data. this create many prototype vectors or if we limit the number of prototypes to a low number, net network may end up with lots of miss classification. this effect CAN cause grandmother node (having prototype for each sample input).

Grandmother node happens when we use large values for rho parameter. In this way each input becomes its own prototype and this reduce the robust Ness of the model. it limits the number of saved clusters and it does not mimic brain anymore.

RCE can super fit on training data if we don't have any inherent error in training samples (Same input, different class)
RCE can adjust the radius of circles, so in wrost case scenario each point has its own circle, so it can fit training input with desired accuracy.
But this is not the case for test inputs. because they may end up inside wrong class cirlce. This can happen if train data is not representive so we declared some wrong classes in the input space.



A is actual input space but during selection of training data, the blue circle went for test set, so now during training those yellow ones can cover the entire area of blue circle and when during the test we introduce test circle blue, it classified as yellow.