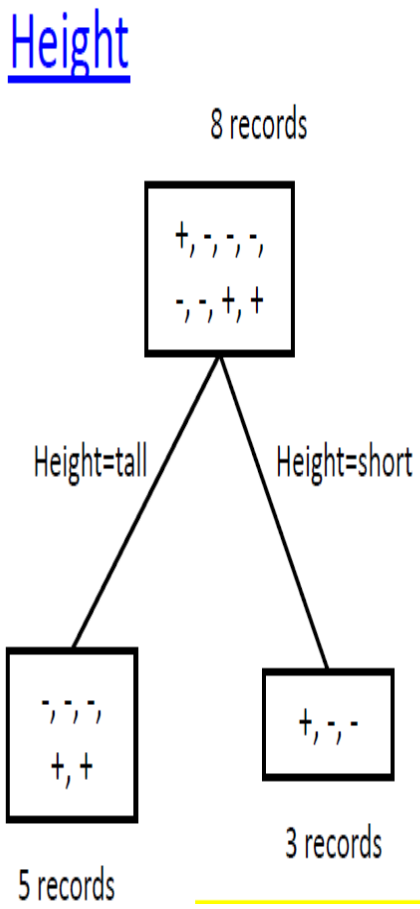
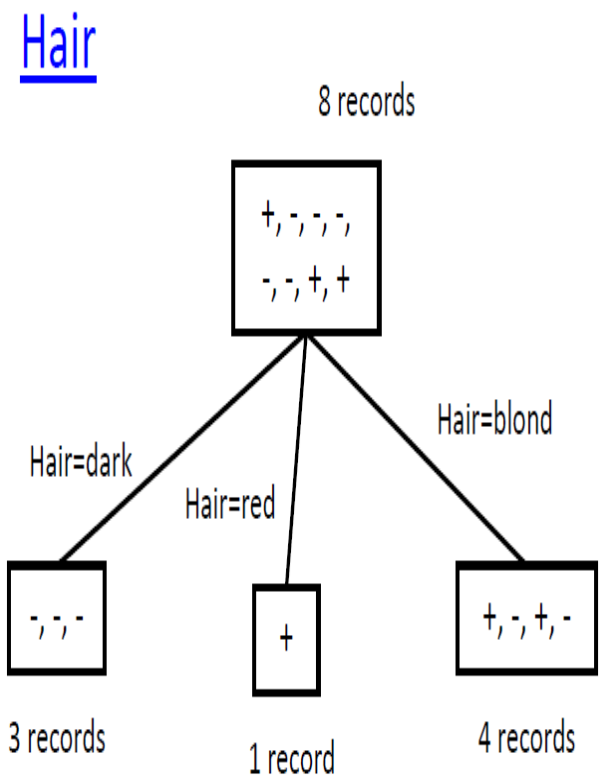
Use of decision tree & Clustering algorithms

Classification Algorithm: Decision tree algorithm

Which feature is a better one to split ?





ID	Height	Hair	Eye	Select?
1	Short	Blond	Blue	+
2	Short	Dark	Blue	–
3	Tall	Dark	Brown	–
4	Tall	Blond	Brown	–
5	Tall	Dark	Blue	–
6	Short	Blond	Brown	–
7	Tall	Red	Blue	+
8	Tall	Blond	Blue	+

How the decision tree works

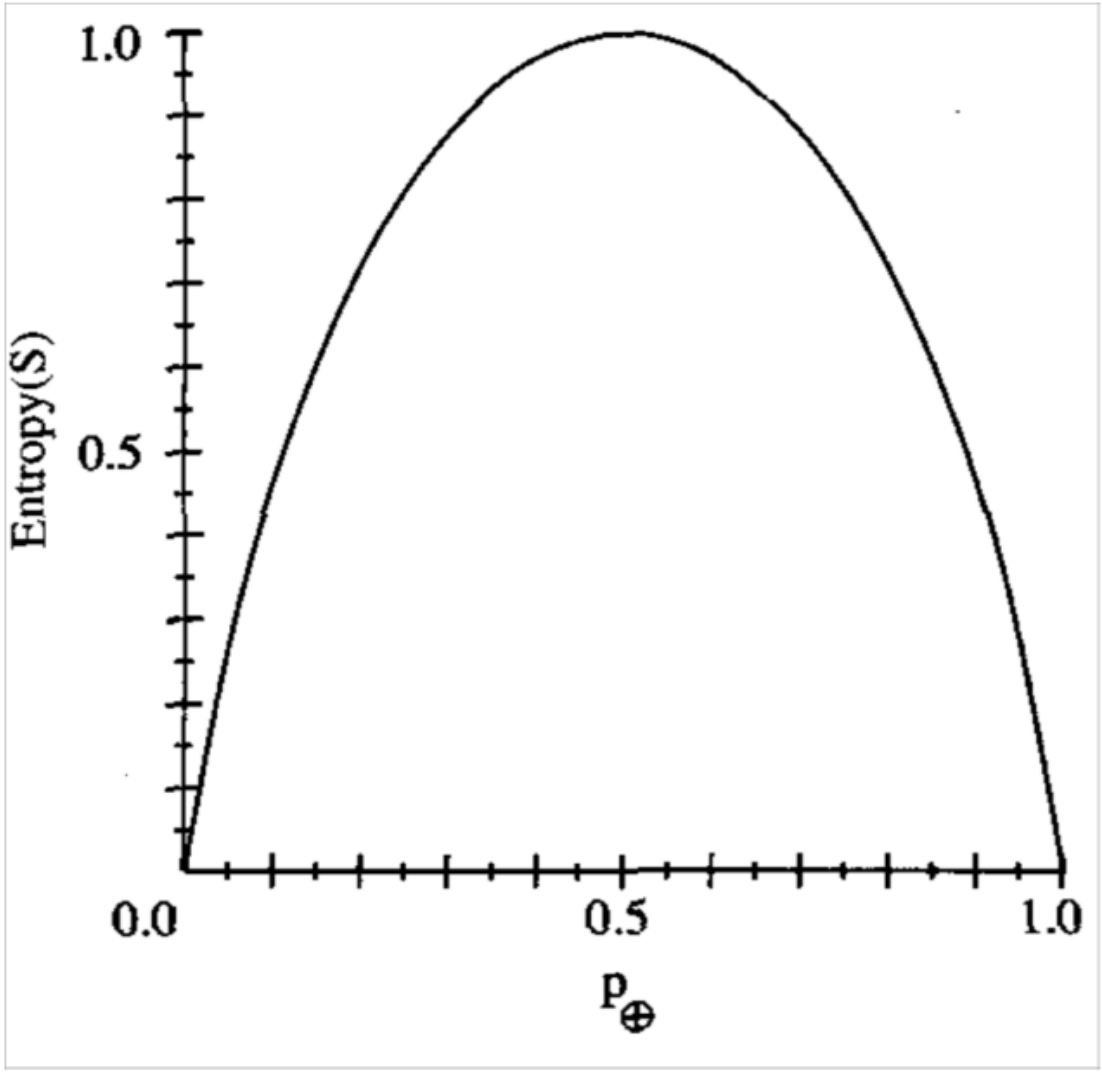
- In a decision tree, the idea is to split the data set based on **homogeneity** of data.
- The measure of impurity of a data set must be at a maximum when all possible classes are equally represented.
- The measure of impurity of a data set must be zero when only one class is represented.
- Measures such as **entropy** or **Gini index** easily meet these criteria and are used to build decision trees as described in the following sections. Different criteria will build different trees through different biases, for example, **information gain** favors tree splits that contain many cases, while **information gain ratio** attempts to balance this.

Impurity measurement with Gini

- A node's gini attribute measures its impurity: a node is "pure" (gini = 0) if all training instances it applies to belong to the same class.
- For example, since the depth-1 left node applies only to Iris-Setosa training instances, it is pure and its gini score is 0.

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

Quantifying Uncertainty: Entropy



Entropy: Impurity (Uncertainty) of a collection(S) of instances

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

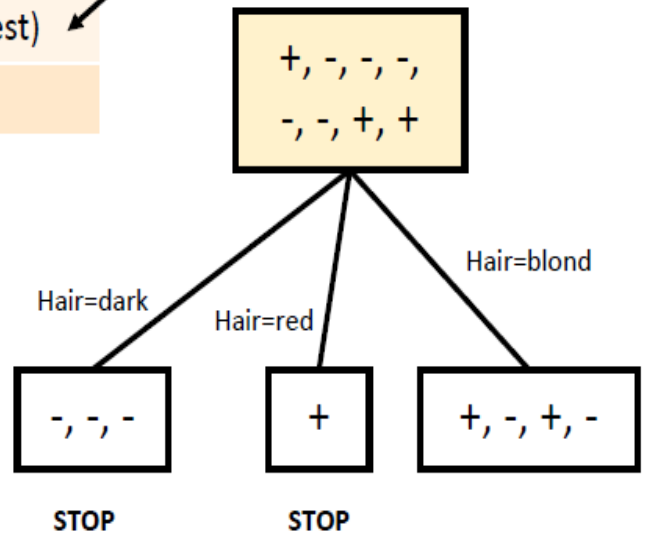
Use of entropy (Example)

ID	Height	Hair	Eye	Select?
1	Short	Blond	Blue	+
2	Short	Dark	Blue	–
3	Tall	Dark	Brown	–
4	Tall	Blond	Brown	–
5	Tall	Dark	Blue	–
6	Short	Blond	Brown	–
7	Tall	Red	Blue	+
8	Tall	Blond	Blue	+

- For the root node

Attribute	Gain
Height	0.954 – 0.951 = 0.003
Hair	0.954 – 0.5 = 0.454 (largest)
Eye	0.954 – 0.607 = 0.347

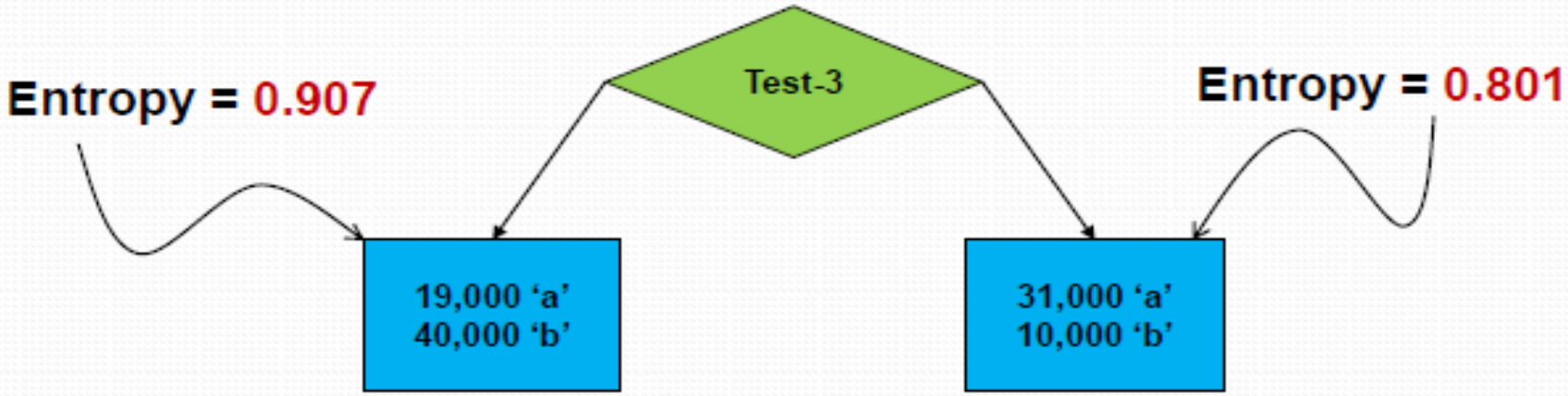
Hair is the best choice for classifying the root node



Another example

Finding a Good Test

- consider yet another test Test-3



- Expected entropy after Test-3:
  - (59/100) * 0.907 + (41/100) * 0.801 = 0.864

Pruning a decision tree: When to stop the splitting

- No attribute satisfies a minimum information gain threshold
- A maximal depth is reached
- Pruning is used so as to avoid overfitting
- Overfitting by a decision tree results not only in a difficult to interpret model, but also provides quite a useless model for unseen data.
- Two approaches of pruning: Pre-pruning and post-pruning

Pre-pruning and Post-pruning

- The above two stopping techniques mentioned above constitute what is known as **pre-pruning** the decision tree, because the pruning occurs before or during the growth of the tree.
- There are also methods that will not restrict the number of branches and allow the tree to grow as deep as the data will allow, and then trim or prune those branches that do not effectively change the classification error rates. This is called **post-pruning**.
- Post-pruning may sometimes be a better option because we will not miss any small but potentially significant relationships between attribute values and classes if we allow the tree to reach its maximum depth. However, one drawback with post-pruning is that it requires additional computations, which may be wasted when the tree needs to be trimmed back.

Decision Tree Algorithm

```
Algorithm build-tree(D, L) {

Begin
        If all records in D share the same label 'K'
                Return a leaf node with the label 'K';
        If ((L is empty) || (all records in D share the same attribute values)) {
                Let 'K' be the most common label in D;
                Return a leaf node with the label 'K';
        }
        For each attribute A in L do
                Compute the expected entropy achieved by using A as the test on D;
        Let A be the attribute which achieves the smallest expected entropy;
        Create a node N with label "A = ?";
        Assume A has n possible values a_1, ..., a_n
        Partition D into D_1, ..., D_n, where D_i contains all the records with A = a_i;
        For each a_i do {
                If D_i is empty {
                        Let 'K' be the most common label in D;
                        Attach a leaf node to N labeled 'K';
                }
                Else
                        Attach the sub-tree obtained by calling build-tree(D_i, L-{A}) to N;
        }
        Return the tree rooted at N;

End
```

Other classification algorithms

- KNN (K nearest neighbors)
- Naïve Bayesian algorithm
- SVM
- ANN
- Ensembling

Clustering algorithm

Clustering

- Clustering is an **unsupervised** machine learning task that automatically divides the data into clusters, or groupings of similar items. It does this without having been told what the groups should look like ahead of time. As we may not even know what we're looking for, clustering is used for knowledge discovery rather than prediction. It provides an insight into the natural groupings found within data.
- Clustering is guided by the principle that records inside a cluster should be very similar to each other, but very different from those outside.

Sample use cases of clustering

- The resulting clusters can then be used for action.
- For instance, you might find clustering methods employed in applications such as: Segmenting customers into groups with similar demographics or buying patterns for targeted marketing campaigns and/ or detailed analysis of purchasing behavior by subgroup
- Detecting anomalous behavior, such as unauthorized intrusions into computer networks, by identifying patterns of use falling outside known clusters
- Simplifying extremely large datasets by grouping a large number of features with similar values into a much smaller number of homogeneous categories

Clustering Definition

- Given a set of data objects, each having a set of attributes, and a similarity measure among them, find clusters such that
  - Objects in one cluster are more similar to one another.
  - Objects in separate clusters are less similar to one another.
- Typically, cluster analysis requires a user to define a similarity measure between records. Clustering is then performed based on the principle of maximizing the intra-cluster similarity and minimizing the inter-cluster similarity (*distance-based clustering*).

Clustering

- Finding similarity groups (called clusters) from data

Requires:

A clustering algorithm

-Partitional clustering

-Hierarchical clustering

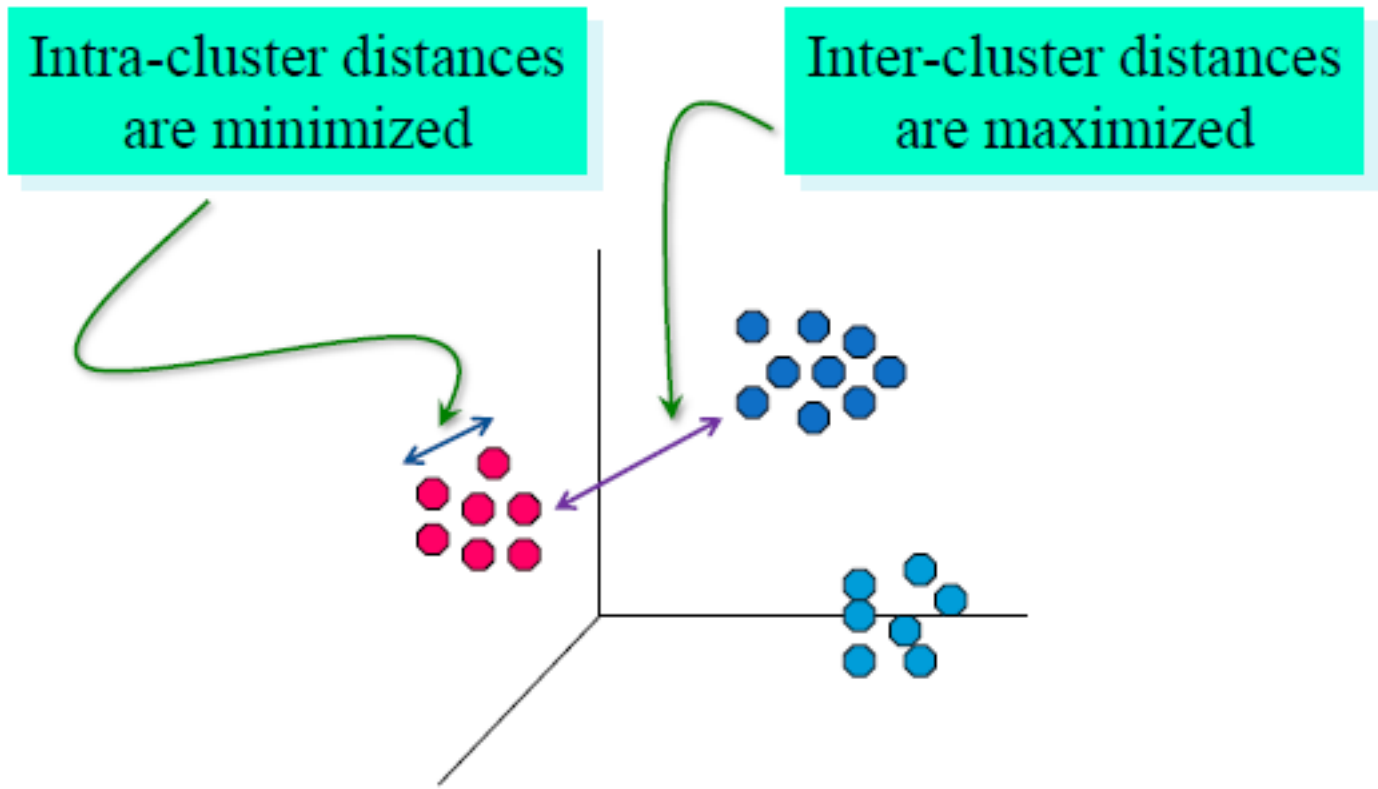
A distance function (similarity / dissimilarity)

Quality measure

-Inter-cluster similarity

-Intra-cluster similarity

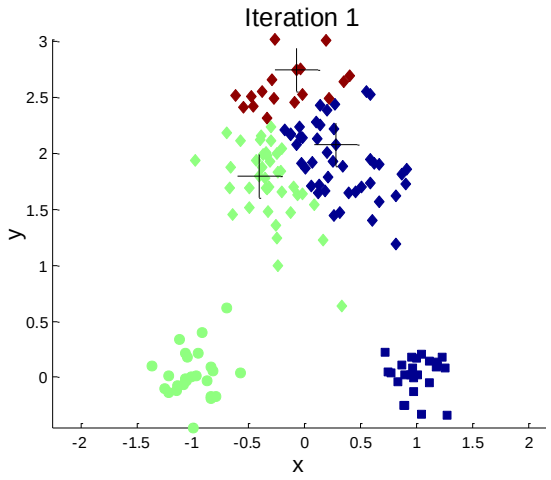
Illustrating Clustering

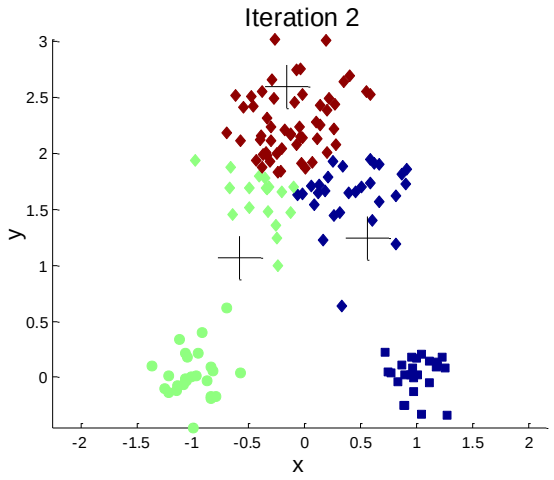


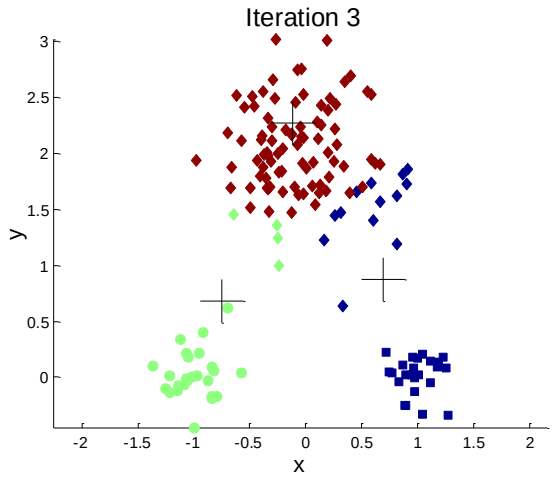
K-Means clustering approach

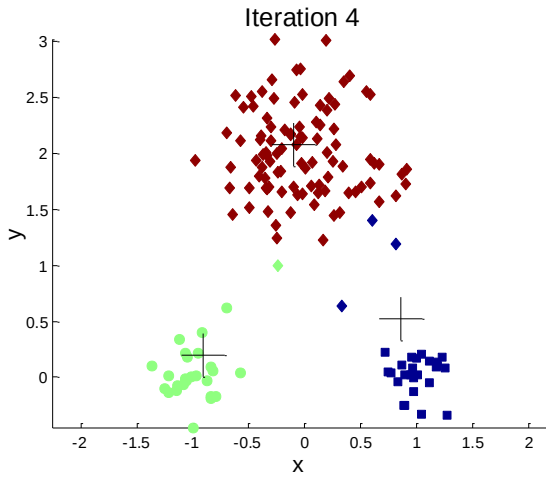
1. Choose a value of k
2. Select k objects in an arbitrary fashion. Use these as the initial set of k centroids
3. Assign each of the objects to the cluster for which it is nearest to the centroid
4. Recalculate the centroids of the k clusters
5. Repeat steps 3 and 4 until the centroids no longer move

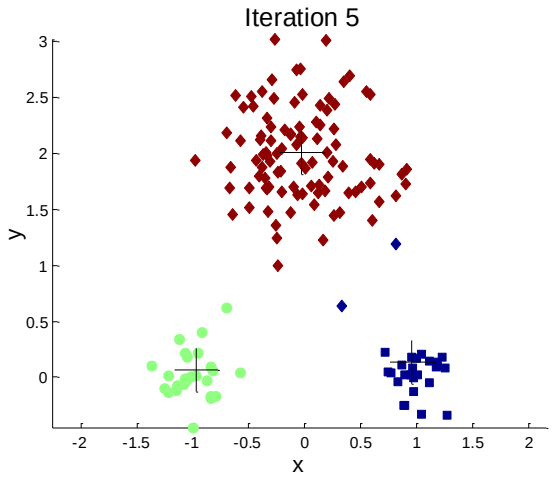
Importance of Choosing Initial Centroids

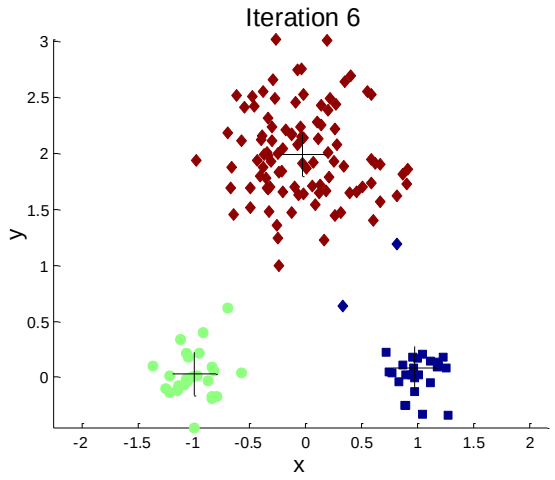












Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - *x* is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K