INF 4300 – Classification III

Anne Solberg

6.11.13

The agenda today:

- More on estimating classifier accuracy
- Curse of dimensionality
- kNN-classification
- K-means clustering

Confusion matrices

- A matrix with the true class label versus the estimated class labels for each class

Estimated class labels

True class labels

	Class 1	Class 2	Class 3	Total # of samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

Binary classifier - performance

• Sensitivity and specificity are measures of performance for a binary classifier.

- •True positive (TP): a sick person classified as sick
- •False positive(FP): a healthy person classified as sick
- •True negative (TN): a healthy person classified as healthy
- •False negative (FN): a sick person classified as healthy

•Sensitivity =TP/(TP+FN) (”recall”)

- the portion of the data set that tested positive of all the positive patients tested.
- probability of positive test for positive instances.

• Specificity = TN/(TN+FP)

- the portion of the data set that tested negative of all the negative patients tested.
- probability of negative test for negative instances.

Binary classifier - performance

- Optimizing these will be a trade off: is correctly identifying a positive more important that falsely classifiying a true negative as positive. This is context dependent.

The curse of dimensionality

- Assume we have S classes and a n-dimensional feature vector.
- With a fully multivariate Gaussian model, we must estimate S different mean vectors and S different covariance matrices from training samples.

  $\hat{\mu}_s$ has n elements

  $\hat{\Sigma}_s$ has n(n-1)/2 elements
- Assume that we have M_s training samples from each class
- Given M_s, there is a maximum of the achieved classification performance for a certain value of n (increasing n beyond this limit will lead to worse performance after a certain).
- Adding more features is not always a good idea!
- A rule of thumb says that we need at least to have M_s>10n (for each class)
- If we have limited training data, we can use diagonal covariance matrices or regularization (i.e., a reduction in complexity).

Use few, but good features

- To avoid the ”curse of dimensionality” we must take care in finding a set of relatively few features.
- A good feature has high within-class homogeneity, and should ideally have large between-class separation.
- In practise, one feature is not enough to separate all classes, but a good feature should:
  - separate some of the classes well
  - Isolate one class from the others.
- If two features look very similar (or have high correlation), they are often redundant and we should use only one of them.
- Class separation can be studied by:
  - Visual inspection of the feature image overlaid the training mask
  - Scatter plots
- Evaluating features as done by training can be difficult to do automatically, so manual interaction is normally required.

Outliers and doubt

- In a classification problem, we might want to identify outliers and doubt samples
- We might want an ideal classifier to report
  - 'this sample is from class l' (usual case)
  - 'this sample is not from any of the classes' (outlier)
  - 'this sample is too hard for me' (doubt/reject)
- The two last cases should lead to a rejection of the sample!

Outliers

- Heuristically defined as "… samples which did not come from the assumed population of samples"
- The outliers can result from some breakdown in preprocessing.
- Outliers can also come from pixels from other classes than the classes in the training data set.
  - Example: K tree species classes, but a few road pixels divide the forest regions.
- One way to deal with outliers is to model them as a separate class, e.g., a gaussian with very large variance, and estimate prior probability from the training data
- Another approach is to decide on some threshold on the aposteriori – and if a sample falls below this threshold for all classes, then declare it an outlier.

Doubt samples

- Doubt samples are samples for which the class with the highest probability is not significantly more probable than some of the other classes (e.g. two classes have essentially equal probability).
- Doubt pixels typically occurr on the border between two classes (”mixels”)
  - Close to the decision boundary the probabilities will be almost equal.
- Classification software can allow the user to specify thresholds for doubt.

The training / test set dilemma

- Ideally we want to maximize the size of both the training and test dataset
- Obviously there is a fixed amount of available data with known labels
- A very simple approach is to separate the dataset in two random subsets
- For small sample sizes we may have to use another strategy: Cross-validation
- This is a good strategy when we have very few ”ground truth” samples.
  - Common in medicine where we might have a small number of patients with a certain type of cancer.
  - The cost of obtaining more ground truth data might be so high that we have to do with a small number of ground truth samples.

Crossvalidation / Leave – n - Out

- A very simple (but computationally complex) idea allows us us to "fake" a large test set
  - Train the classifier on a set of *N-n* samples
  - Test the classifier on the *n* remaining samples
  - Repeat n/N times (dependent on subsampling)
  - Report average performance on the repeated experiments as "test set" error
- An example with leave-1-out and 30 samples:
  - Select one sample to leave out
  - Train on the remaining 29 samples
  - Classify the one sample and store its class label
  - Repeat this 30 times
  - Count the number of misclassifications among the 30 experiments.
- Leave-n-Out estimation generally overestimates the classification accuracy.
  - Feature selection should be performed within the loop, not in advance!!!
- Using a training set and a test set of approximately the same size is better.

The covariance matrix and dimensionality

- Assume we have S classes and a n-dimensional feature vector.
- With a fully multivariate Gaussian model, we must estimate S different mean vectors and S different covariance matrices from training samples.

$\hat{\mu}_s$ has n elements

$\hat{\Sigma}_s$ has n(n-1)/2 elements

- Assume that we have M_s training samples from each class
- Given M_s, there is a maximum of the achieved classification performance for a certain value of n
  - increasing n beyond this limit will lead to worse performance.
- Adding more features is not always a good idea!
- Total number of samples given by a rule of thumb: **M>10 n S**
- If we have limited training data, we can use diagonal covariance matrices or regularization

The ”curse” of dimensionality

- In practice, the curse means that, for a given sample size, there is a maximum number of features one can add before the classifier starts to degrade.

- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.
- For a high dimensionality, we will need lots of training data to get the best performance.
- => ≈10 samples / feature / class.

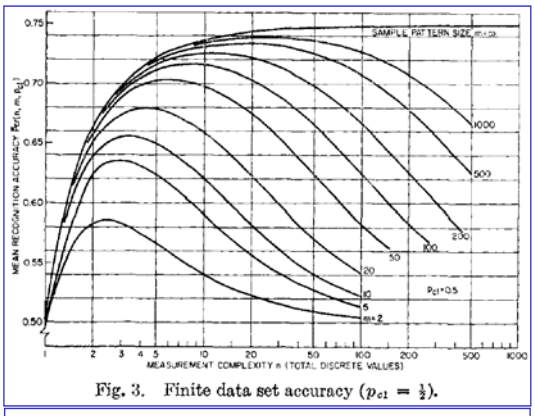


Fig. 3. Finite data set accuracy ($p_{ci} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.

How do we beat the ”curse of dimensionality”?

- Use regularized estimates for the Gaussian case
  - Use diagonal covariance matrices
  - Apply regularized covariance estimation (INF 5300)
- Generate few, but informative features
  - Careful feature design given the application
- Reducing the dimensionality
  - Feature selection (more in INF5300)
  - Feature transforms (INF 5300)

Exhaustive feature selection

- If – for some reason – you know that you will use d out of D available features, an exhaustive search will involve a number of combinations to test:

$$n = \frac{D!}{(D-d)!\,d!}$$

- If we want to perform an exhaustive search through D features for the optimal subset of the d ≤ m "best features", the number of combinations to test is

$$n = \sum_{d=1}^{m} \frac{D!}{(D-d)!\,d!}$$

- Impractical even for a moderate number of features!

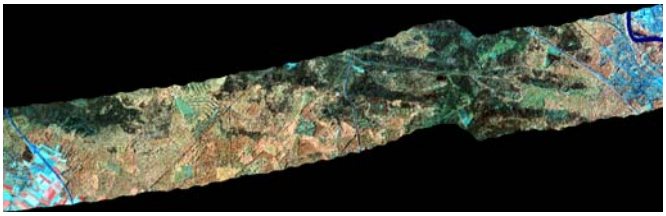
d ≤ 5, D = 100 => n = 79.374.995

Suboptimal feature selection

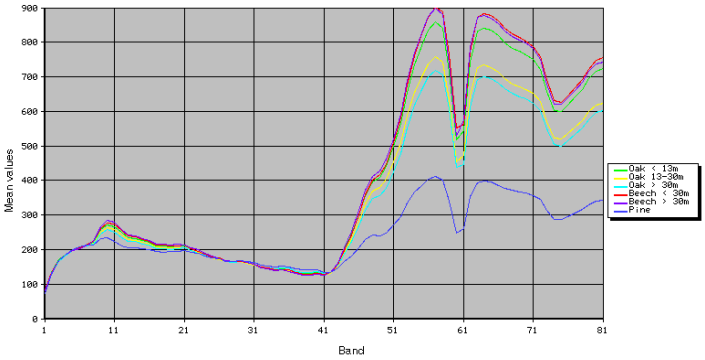
- Select the best single features based on some quality criteria, e.g., estimated correct classification rate.
  - A combination of the best single features will often imply correlated features and will therefore be suboptimal .
- "Sequential forward selection" implies that when a feature is selected or removed, this decision is final.
- "Stepwise forward-backward selection" overcomes this.
  - A special case of the "add - a, remove - r algorithm".
- Improved into "floating search" by making the number of forward and backward search steps data dependent.
  - "Adaptive floating search"
  - "Oscillating search".

Hyperspectral image example

- A hyperspectral image from France
- 81 features/spectral bands
- 6 classes (tree species)
- μ has 81 parameters to compute for each class
- Σ has 81*80/2=3240 parameters for each class.
- 1000 training samples for each class.
- Test set: 1000-2000 samples for each class.



3 of the 81 bands shown as RGB image

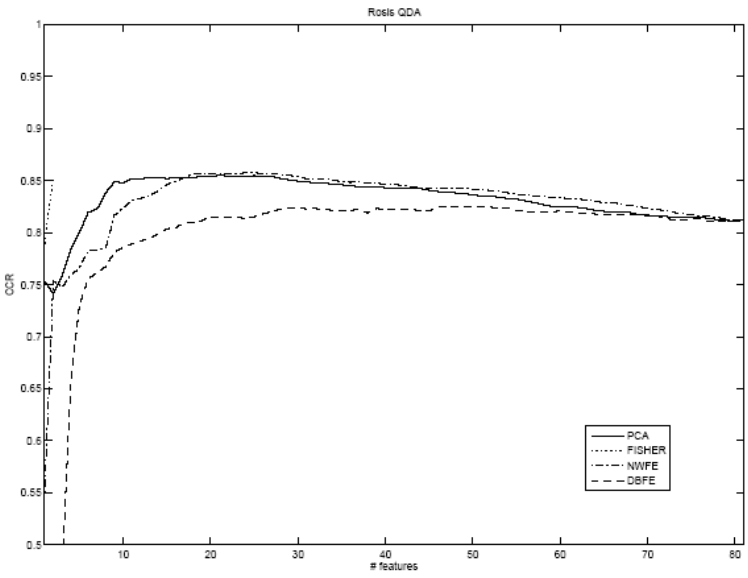


Plots of the 81 mean values for each class

Hyperspectral example

classification accuracy vs. nof. features on test set

classification accuracy



number of features used in a Gaussian classifier

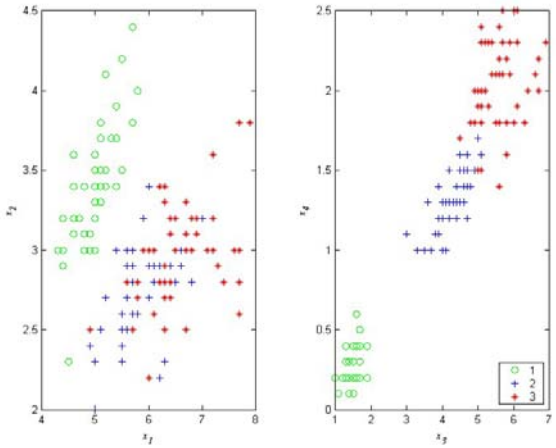
Each curve shows a different dimensionality reduction method.

Note that as we include more features, the classification accuracy first increases, then it starts to decrease.

Curse of dimensionality!

Exploratory data analysis

- For a small number of features, manual data analysis to study the features is recommended.
- Choose intelligent features.
- Evaluate e.g.
  - Error rates for single-feature classification
  - Scatter plots



Scatter plots of feature combinations

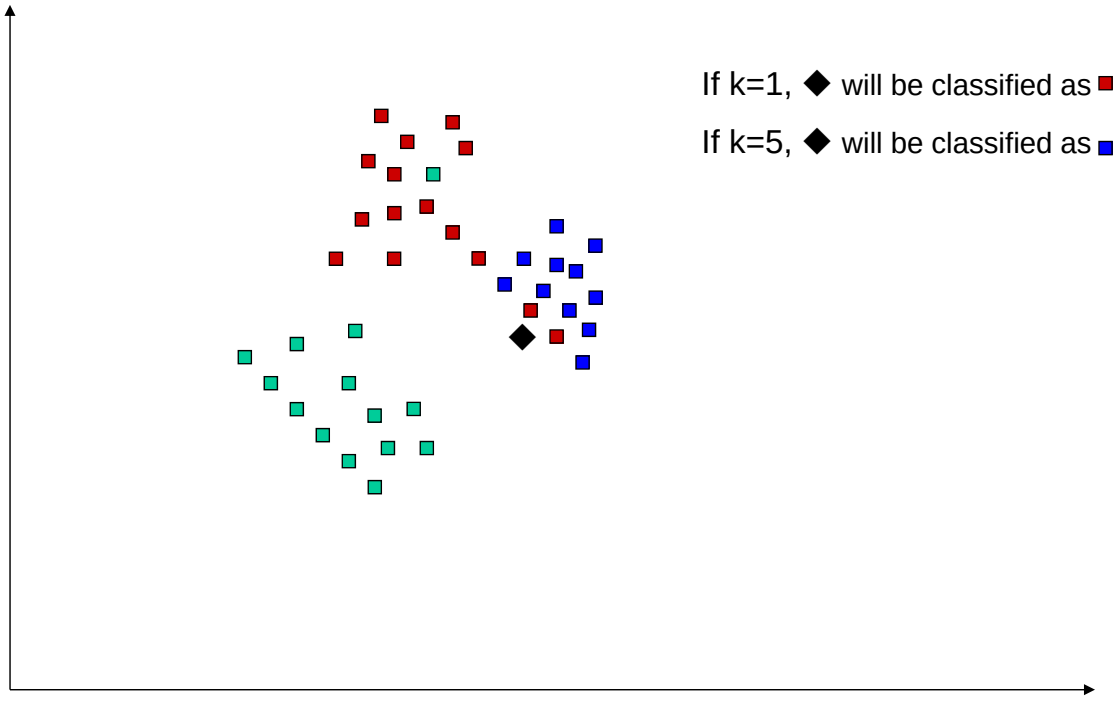
Is the Gaussian classifier the only choice?

- The Gaussian classifier gives linear or quadratic discriminant function.
- Other classifiers can give arbitrary complex decision surfaces (often piecewise-linear)
  - Neural networks
  - Support vector machines
  - kNN (k-Nearest-Neighbor) classification
  - Mixtures of Gaussians

k-Nearest-Neighbor classification

- A very simple classifier.
- Classification of a new sample x_j is done as follows:
  - Out of N training vectors, identify the k nearest neighbors (measure by Euclidean distance) in the training set, irrespectively of the class label.
  - Out of these k samples, identify the number of vectors k_i that belong to class ω_i , $i:1,2,....M$ (if we have M classes)
  - Assign x_j to the class ω_i with the maximum number of k_i samples.
- k should be odd, and must be selected a priori.

kNN-example



About kNN-classification

- If *k=1* (1NN-classification), each sample is assigned to the same class as the closest sample in the training data set.
- If the number of training samples is very high, this can be a good rule.
- If k->∞, this is theoretically a very good classifier.
- This classifier involves no ”training time”, but the time needed to classify one pattern x_i will depend on the number of training samples, as the distance to all points in the training set must be computed.
- ”Practical” values for *k*: *3<=k<=9*

Supervised or unsupervised classification

- Supervised classification
  - Classify each object or pixel into a set of *k* known classes
  - Class parameters are estimated using a set of training samples from each class.
- Unsupervised classification
  - Partition the feature space into a set of *k* clusters
  - *k* is not known and must be estimated (difficult)
- In both cases, classification is based on the value of the set of *n* features $x_1,....x_n$.
- The object is classified to the class which has the highest posterior probability.
- ”The clusters we get are not the classes we want”.

Unsupervised classification/clustering

- Divide the data into clusters based on similarity (or dissimilarity)
- Similarity or dissimilarity is based on distance measures (sometimes called proximity measures)
  - Euclidean distance, Mahalanobis distance etc.
- Two main approaches to clustering
  - hierarchical
    - divisive
    - agglomerative
  - non-hierarchical (sequential)
- Non-hierarachical methods are often used in image analysis

K-means clustering

- Note: K-means algorithm normally means ISODATA, but different definitions are found in different books
- K is assumed to be known

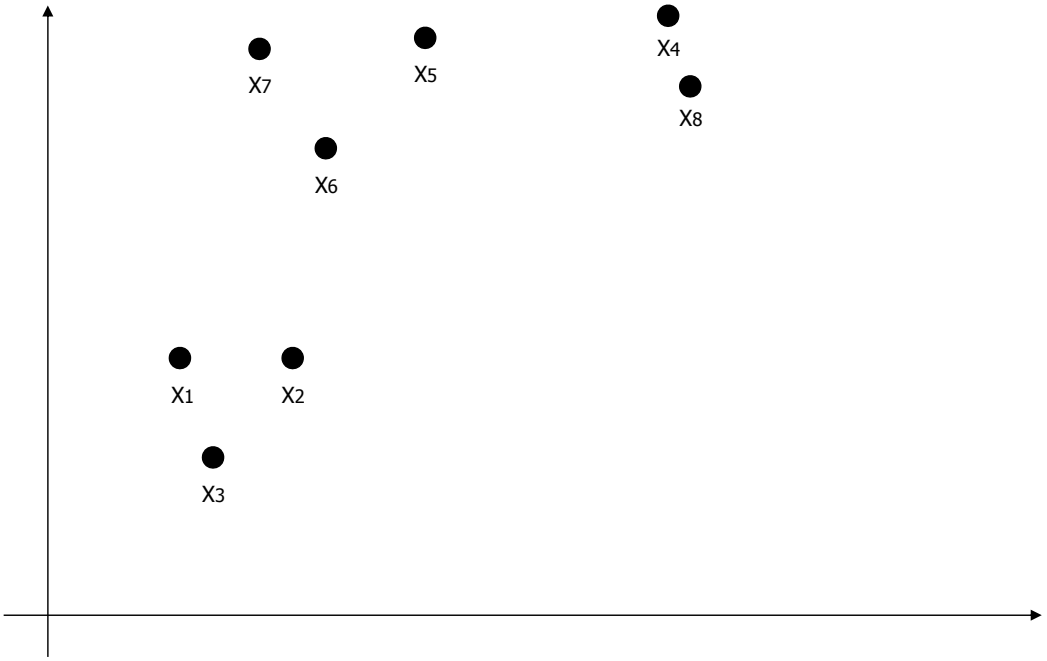
1. Start with assigning K cluster centers
   - k random data points, or the first K points, or K equally spaces points
   - For k=1:K, Set μ_k equal to the feature vector x_k for these points.
2. Assign each object/pixel x_i in the image to the closest cluster center using Euclidean distance.
   - Compute for each sample the distance r2 to each cluster center:

$$r^2 = (x_i - \mu_k)^T (x_i - \mu_k) = \|x_i - \mu_k\|^2$$

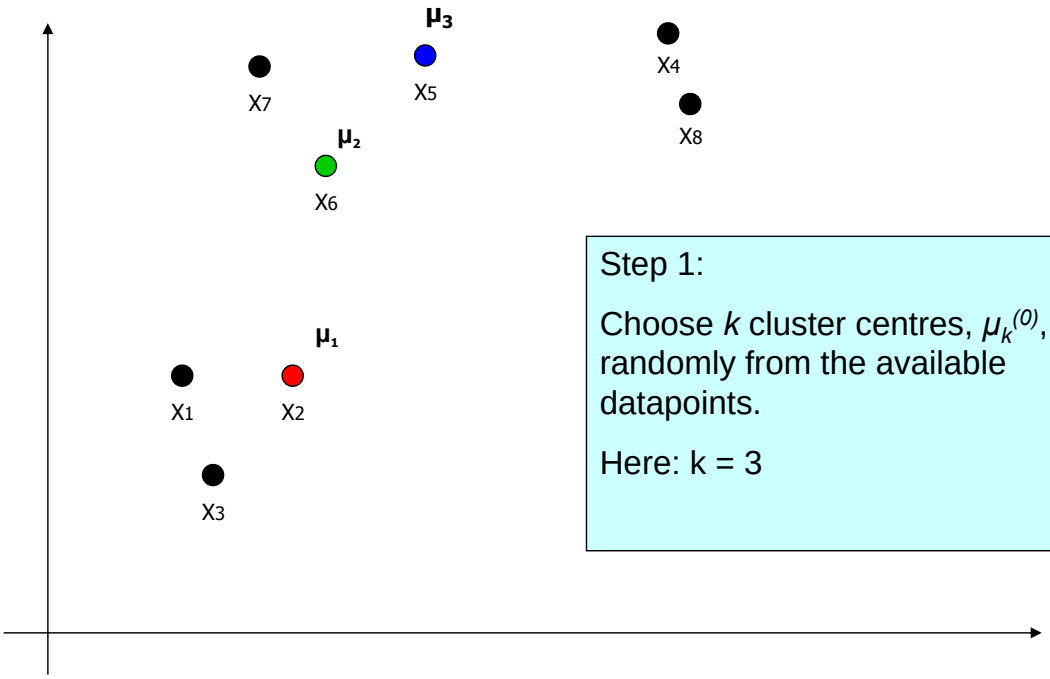
   - Assign x_i to the closest cluster (with minimum r value)
3. Recompute the cluster centers based on the new labels.
4. Repeat from 2 until #changes<limit.

ISODATA K-means: splitting and merging of clusters are included in the algorithm

k-means example



k-means example

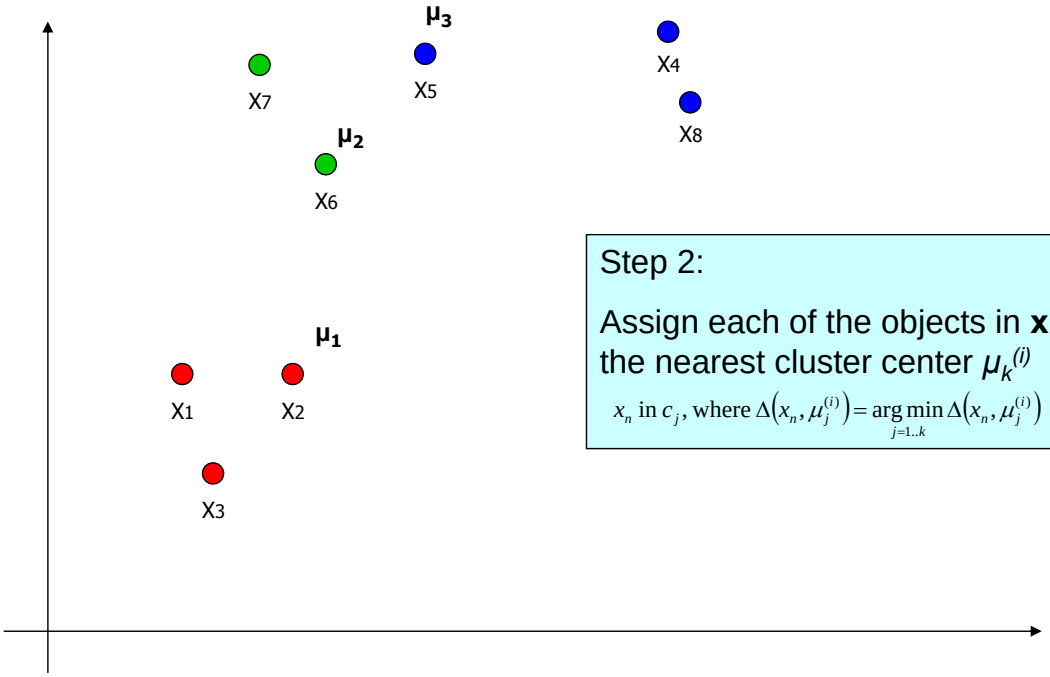


Step 1:

Choose k cluster centres, $\mu_k^{(0)}$, randomly from the available datapoints.

Here: k = 3

k-means example

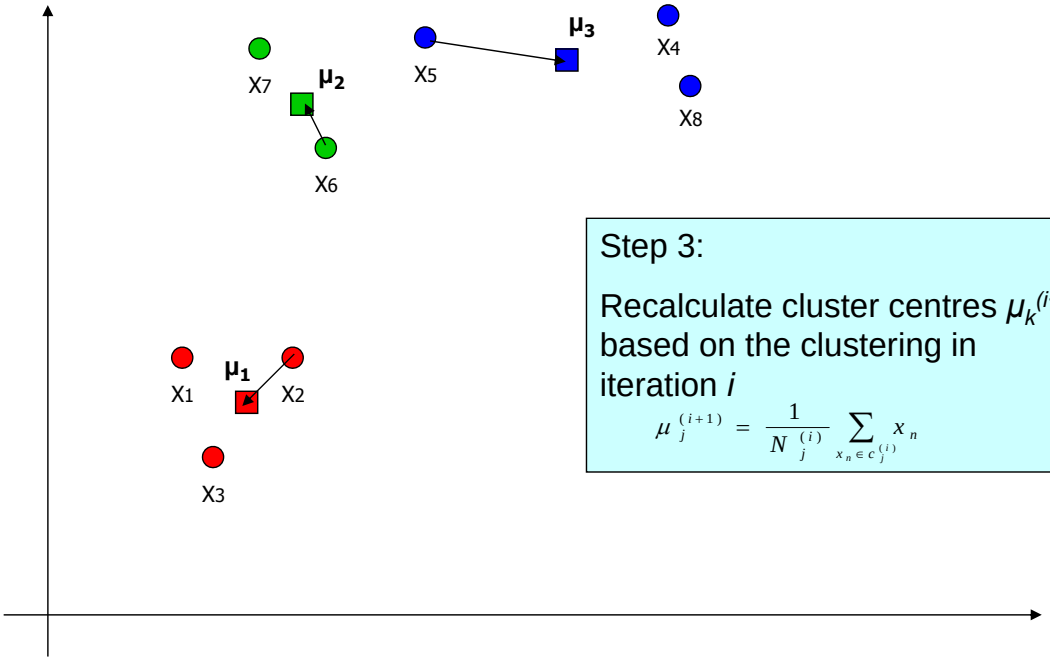


Step 2:

Assign each of the objects in **x** to the nearest cluster center $\mu_k^{(i)}$

$$x_n \text{ in } c_j, \text{ where } \Delta\left(x_n, \mu_j^{(i)}\right) = \arg\min_{j=1..k} \Delta\left(x_n, \mu_j^{(i)}\right)$$

k-means example

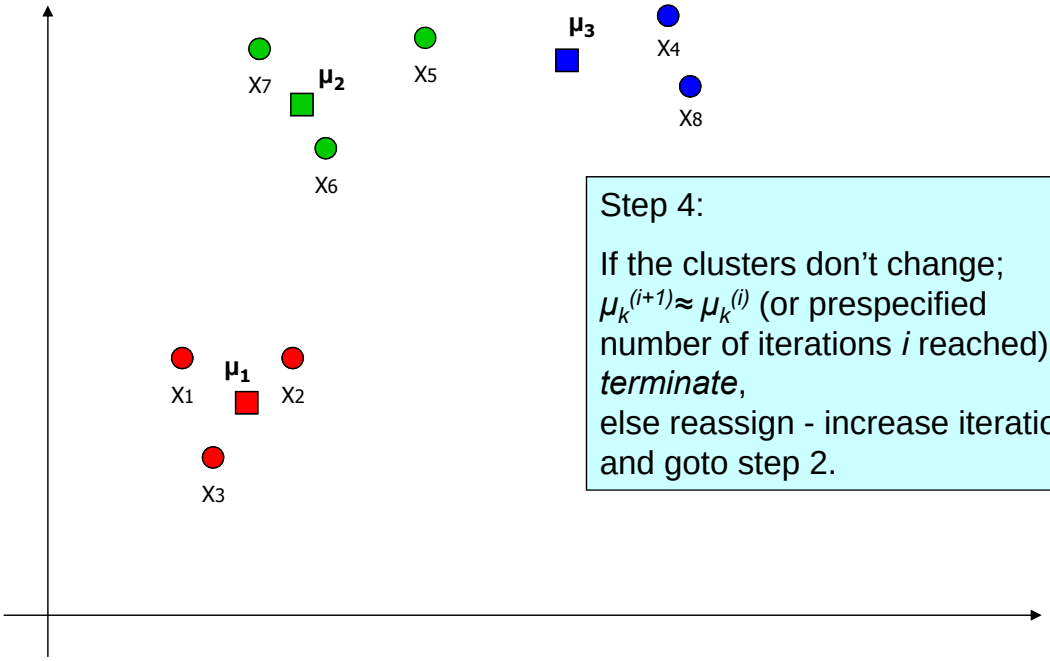


Step 3:

Recalculate cluster centres $\mu_k^{(i+1)}$ based on the clustering in iteration *i*

$$\mu_j^{(i+1)} = \frac{1}{N_j^{(i)}} \sum_{x_n \in c_j^{(i)}} x_n$$

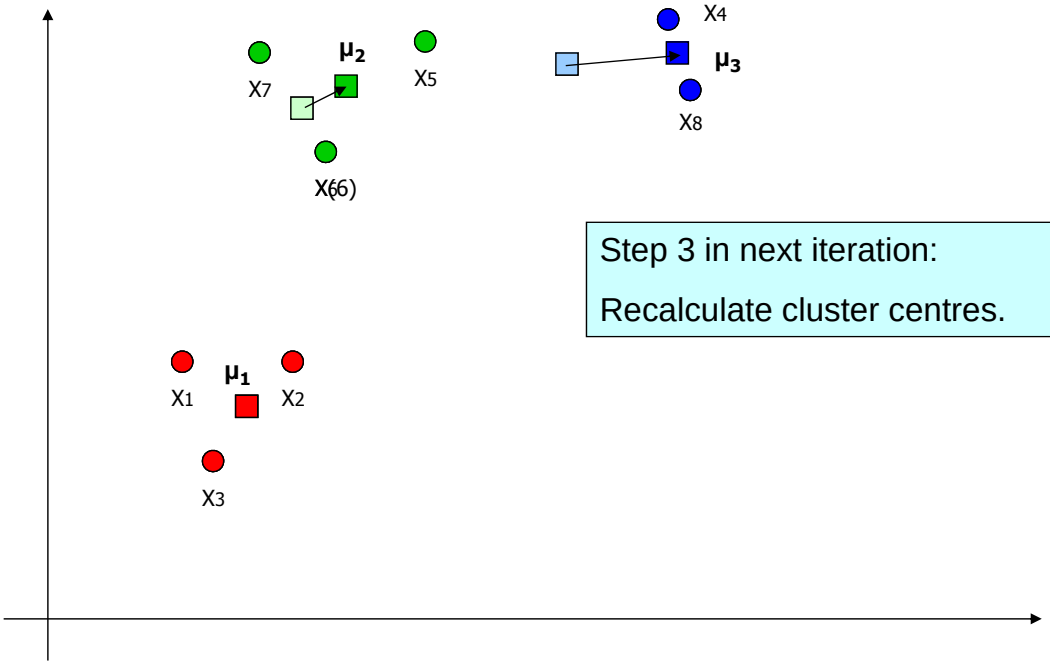
k-means example



Step 4:

If the clusters don't change; $\mu_k^{(i+1)} \approx \mu_k^{(i)}$ (or prespecified number of iterations *i* reached), *terminate*,
else reassign - increase iteration *i* and goto step 2.

k-means example



Step 3 in next iteration:

Recalculate cluster centres.

k-means variations

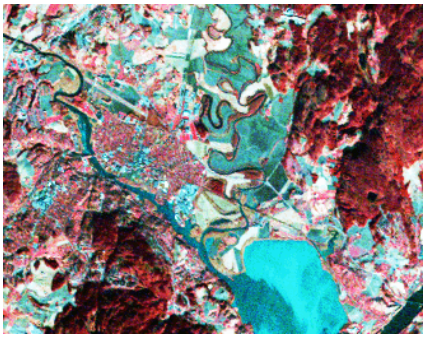
- The generic algorithm has many improvements
  - ISODATA – allow for merging and splitting of clusters
    - Among other things, this seeks to improve an initial ”bad” choice of *k*
  - k-medians is another variation
  - k-means optimizes a probabilistic model

How do we determine k?

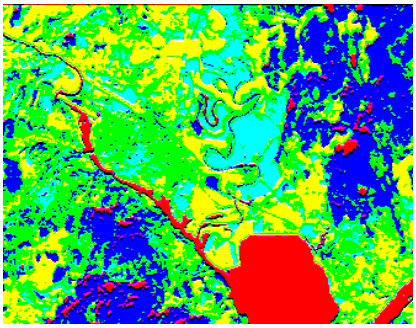
- The number of natural clusters in the data rarely corresponds to the number of information classes of interest.
- Cluster validity indices can give indications of how many clusters there are.
- Use cluster merging or splitting tailored to the application.
- Rule of thumb for practical image clustering:
  - start with approximately twice as many clusters as expected information classes
  - determine which clusters correspond to the information classes
  - split and merge clusters to improve.

Example: K-means clustering

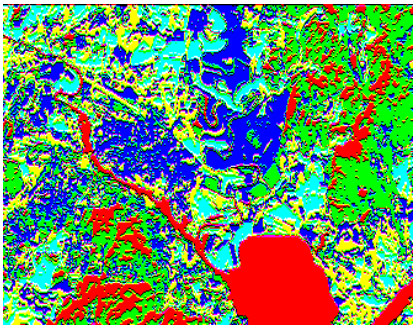
Original



Supervised
4 classes



Kmeans
K=5



Kmeans
K=10