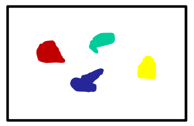
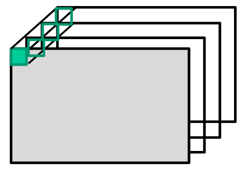
INF 4300
17.10.16
Classifier evaluation
KNN-classification
Kmeans-clustering

Anne Solberg (anne@ifi.uio.no)

- x_i – feature vector for pixel i
- ω_i - The class label for pixel i
- K – the number of classes given in the training data


Mask with training pixels


Multiband image with
n spectral channels or features

$$p(\mathbf{x}\mid\omega_s)=\frac{1}{(2\pi)^{n/2}\left|\boldsymbol{\Sigma}_s\right|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_s)^t\boldsymbol{\Sigma}_s^{-1}(\mathbf{x}-\boldsymbol{\mu}_s)\right]$$

From last week: Discriminant functions for the normal density

- When finding the class with the highest probability, these functions are equivalent:

$$g_i(\mathbf{x})=P(\omega_i\mid\mathbf{x})=\frac{p(\mathbf{x}\mid\omega_i)P(\omega_i)}{p(\mathbf{x})}$$
$$g_i(\mathbf{x})=p(\mathbf{x}\mid\omega_i)P(\omega_i)$$
$$g_i(\mathbf{x})=\ln p(\mathbf{x}\mid\omega_i)+\ln P(\omega_i)$$

- Let us now look at $g_i(\mathbf{x})=\ln p(\mathbf{x}\mid\omega_i)+\ln P(\omega_i)$
- With a multivariate Gaussian we get:

$$g_i(\mathbf{x})=-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)-\frac{d}{2}\ln 2\pi-\frac{1}{2}\ln\left|\Sigma_i\right|+\ln P(\omega_i)$$

- Let ut look at this expression for some special cases:

Case 1: $\Sigma_j=\sigma^2 I$

- Now we get an equivalent formulation of the discriminant functions:

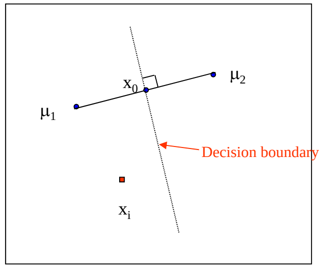
$$g_i(\mathbf{x})=\mathbf{w}_i^t\mathbf{x}+w_{i0}$$
$$\text{where } \mathbf{w}_i=\frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } w_{i0}=-\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i+\ln P(\omega_i)$$

- An equation for the decision boundary $g_i(\mathbf{x})=g_j(\mathbf{x})$ can be written as

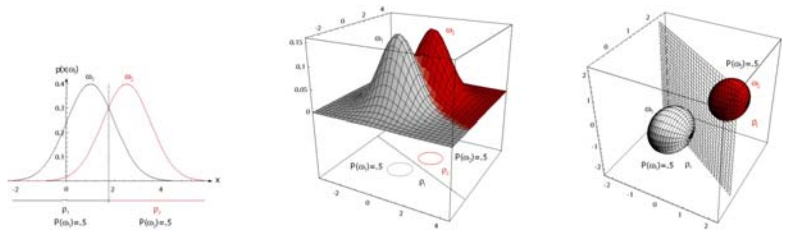
$$\mathbf{w}^t(\mathbf{x}-\mathbf{x}_0)=0$$
$$\text{where } \mathbf{w}=\boldsymbol{\mu}_i-\boldsymbol{\mu}_j$$
$$\text{and } x_0=\frac{1}{2}(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)-\frac{\sigma^2}{\left\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right\|^2}\ln\left[\frac{P(\omega_i)}{P(\omega_j)}\right](\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)$$

- $\mathbf{w}=\mu_i-\mu_j$ is the vector between the mean values.
- This equation defines a hyperplane through the point x_0, and orthogonal to **w**.
- If $P(\omega_i)=P(\omega_j)$ the hyperplane will be located halfway between the mean values.
- Proving this involves some algebra, see the proof at https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

- The discriminant function (when $\Sigma_j=\sigma^2 I$) that defines the border between class 1 and 2 in the feature space is a straight line.
- The discriminant function intersects the line connecting the two class means at the point $x_0=(\mu_1-\mu_2)/2$ (if we do not consider prior probabilities).
- The discriminant function will also be normal to the line connecting the means.



A simple model, $\Sigma_j=\sigma^2 I$



- The distributions are spherical in *d* dimensions.
- The decision boundary is a generalized hyperplane of *d-1* dimensions
- The decision boundary is perpendicular to the line separating the two mean values
- This kind of a classifier is called a linear classifier, or a linear discriminant function
  - Because the decision function is a linear function of ***x***.
- If $P(\omega_i)= P(\omega_j)$, the decision boundary will be half-way between μ_i and μ_j

Case 2: Common covariance, $\Sigma_j= \Sigma$

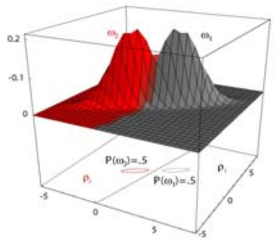
- If we assume that all classes have the same shape of data clusters, an intuitive model is to assume that their probability distributions have the same shape
- By this assumption we can use all the data to estimate the covariance matrix
- This estimate is common for all classes, and this means that also in this case the discriminant functions become linear functions

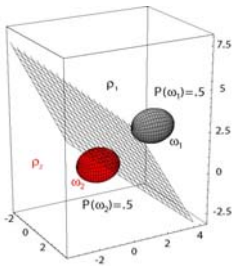
$$g_j(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

$$= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

Common for all classes, no need to compute
Since ***x***$\mathbf{x}^T\mathbf{x}$ is common for all classes, $g_j(\mathbf{x})$ again reduces to a linear function of ***x***.

Common covariance, $\Sigma_j= \Sigma$





- The classes can be described by hyperellipsoides in *d* dimensions.
- All hyperellipsoids have the same orientation.
- The decision boundary will again be a hyperplane.
- Because ***w***$= \Sigma^{-1}(\mu_i-\mu_j)$ is generally not in the direction of $\mu_i-\mu_j$, the hyperplane will not be perpendicular to the line between the means.
- Consider a point x_0 on the line $\mu_i-\mu_j$, defined by the prior probabilities:
  - If $P(\omega_i)= P(\omega_j)$, x_0 will be half way between the means.
  - The separating hyperplane will *intersect* the line at x_0

Case 3:, Σ_j=arbitrary

- When all classes are modeled as having different *shapes,* the discriminant functions cannot be simplified
- This means that the discriminant functions will be *quadratic* functions
- Decision boundaries will be hyperquadrics and assume any of the general forms:
  - hyperplanes, pairs of hyperplanes, hyperspheres, hyperellisoides, hyperparaboloids, hyperhyperboloids...

Case 3:, Σ_j=arbitrary

- The discriminant functions will be quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + wi_0$$

$$\text{where } \mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}, \qquad \mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$$

$$\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i{}^t \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i)$$

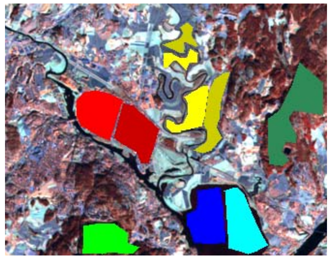
- The decision surfaces are hyperquadrics and can assume any of the general forms:
  - hyperplanes
  - hypershperes
  - pairs of hyperplanes
  - hyperellisoids,
  - Hyperparaboloids,..
- The next slides show examples of this.
- In this general case we cannot intuitively draw the decision boundaries just by looking at the mean and covariance.

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)
  - Mixtures of Gaussians
  - Other probability density functions (t-distribution, exponetial distributions).
  - Neural networks
  - Support vector machines
  - Ensembles of simple classifiers
    ADAboost
    Random forest/decision trees
  - kNN (k-Nearest-Neighbor) classification

Using masks to train and test

• Training mask: a mask where regions to train each class are marked using different pixel values, e.g. class label=1 for class 1, 2 for class 2 etc.
•Test mask: a similar mask as training, but to estimate classifier accuracy only.



Training a classifier

- Obtain as many ground truth samples for each class as possible
  - If visual inspection is reliable, experts can mark training regions interactively.
  - For remote sensing, go out in the field and collect field samples (or use images from a different sensor)
  - For symbol recognition, mark a set of symbols manually.
  - For medical applications, use e.g. tissue samples or interpretations made by experts.
- Divide the ground truth into a training set and a test set.
- Use feature extraction and feature selection/evaluation to determine the best set of features.
- Decide if a linear or quadratic classifier is needed.

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

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

Estimating μ_s and Σ_s

- For each class, compute μ_s (and Σ_s) either with a for-loop on each feature, or use a vector implementation.

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{M_s}\sum_{m=1}^{M_s} \mathbf{x}_m,$$

where the sum is over all training samples belonging to class s

$$\hat{\Sigma}_s = \frac{1}{M_s}\sum_{m=1}^{M_s}(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)^t$$

where the sum is over all training samples belonging to class s

$$\sigma_{ij,s}{}^2 = \frac{1}{M_s}\sum_{m=1}^{M_s}(x_{m,i} - \hat{\mu}_{i,s})(x_{m,j} - \hat{\mu}_{j,s})^t$$

for the covariance between feature i and j for class s

Training

```
for i=1:N
  for j=i:M
    if mask(i,j)>==K
      increment nof. Samples in class K
      store the feature vector f(i,j) in a vector of training samples from class K
   end
end
end

For class k=1:K
    compute mean(k) and sigma(k)
```

Classifying new data

- For each sample, compute the posterior probabilities for each class.

$$P(\omega_s \mid x) \propto p(x \mid \omega_s) P(\omega_s)$$

$$= \left[\frac{1}{(2\pi)^{P/2} |\Sigma_s|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_s)^t \Sigma_s^{-1} (x-\mu_s) \right] \right] P(\omega_s)$$

- Classify the sample to the class with the highest posterior probability.
- Evaluate the performance of the classifier on a different dataset.
- We can also produce images of the posterior probability for each class.

Validating classifier performance

- Classification performance is evaluated on a different set of samples with known class - the test set.
- The training set and the test set must be independent!
- Normally, the set of ground truth pixels (with known class) is partionioned into a set of training pixels and a set of test pixels of approximately the same size.
- This can be repeated several times to compute more robust estimates as average test accuracy over several different partitions of test set and training set.
  - By selecting e.g. 10 random partitions of the set of samples into a training set and a test set.

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

Confusion matrix - cont.

Alternatives:

•Report nof. correctly classified pixels for each class.

•Report the percentage of correctly classified pixels for each class.

•Report the percentage of correctly classified pixels in total.

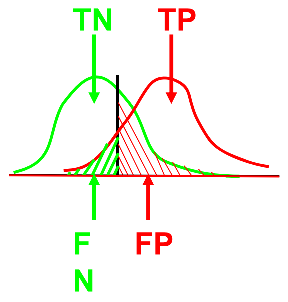
•Why is this not a good measure if the number of test pixels from each class varies between classes?

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

True / False positives / negatives

- **True positive (TP):** Patient has cancer and test result is positive.
- True negative (TN): A healthy patient and a negative test result.
- **False positive (FP):** Healthy patient that gets a positive test result.
- **False negative (FN):** Cancer patient that gets a negative test result.

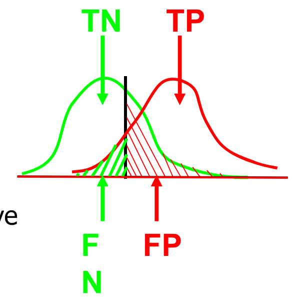
E.g., testing for cancer
No cancer | Cancer



- *Good to have: **TP** & **TN***
- *Bad to have: **FP** (but this will probably be detected)*
- *Worst to have: **FN** **(may go un-detected)***

Sensitivity and specificity

- ***Sensitivity:***
  the portion of the data set that tested positive out of all the positive patients tested:
  - **Sensitivity = TP/(TP+FN)**
  - The probability that the test is positive given that the patient is sick.
  - Higher sensitivity means that fewer decease cases go undetected.
- ***Specificity:***
  the portion of the data set that tested negative out of all the negative patients tested:
  - **Specificity = TN/(TN+FP)**
  - The probability that a test is negative given that the patient is not sick.
  - Higher specificity means that fewer healthy patients are labeled as sick.



Bayes classification with loss functions

- In cases where different classes have different importance (e.g. sick/healthy), we can incorporate this into a Bayesian classifier if we consider the loss.
- Let $\lambda(\alpha_i|\omega_j)$ be the loss if we decide class α_i if the true class is ω_j.
- The risk of deciding class α_i is then: $R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x})$
- To minimize the overall risk, compute $R(\alpha_i|x)$ for i=1…c and choose the class for which $R(\alpha_i|x)$ is minimum.

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 probability– 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 d-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 d elements

$\hat{\Sigma}_s$ has d(d+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 d
  - 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 d 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.

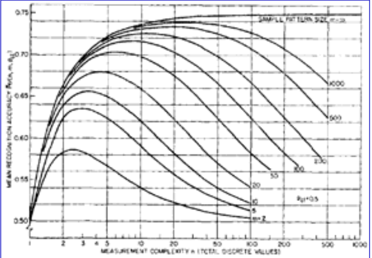


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

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

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.

How do we beat the ”curse of dimensionality”?

- Use regularized estimates for the Gaussian case
  - Use diagonal covariance matrices
  - Apply regularized covariance estimation
- Generate few, but informative features
  - Careful feature design given the application
- Reducing the dimensionality
  - Feature selection – select a subset of the original features (more in INF5300)
  - Feature transforms – compute a new subset of features based on a linear combination of all features (next week)
    - Example 1: Principal component transform
      - Unsupervised, finds the combination that maximized the variance in the data.
    - Example 2: Fisher’s linear discriminant
      - Supervised, finds the combination that maximizes the distance between the classes.

Regularized covariance matrix estimation

- Case 1 :Diagonal covariance matrix.
- Case 2: Common covariance matrix
- Let the covariance matrix be a weighted combination of a class-specific covariance matrix Σ_k and a common covariance matrix Σ (estimated from training samples for all classes) :

$$\Sigma_k(\alpha) = \frac{(1-\alpha)n_k\Sigma_k + \alpha n\Sigma}{(1-\alpha)n_k + \alpha n}$$

  where $0 \le \alpha \le 1$ must be determined, and n_k and n is the number of training samples for class k and overall.
- Alternatively:

$$\Sigma_k(\beta) = (1-\beta)\Sigma_k + \beta I$$

  where the parameter $0 \le \beta \le 1$ must be determined.
- The effect of these are that we can use a quadratic classifier even if we have little training data/ill-conditioned Σ_k
- We still have to be able to compute Σ_k, but the only the regularized/more robust $\Sigma_k(\alpha)$ or $\Sigma_k(\beta)$ must be inverted.

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:

$$n = \frac{D!}{(D-d)!\,d!} \qquad d!=1*2*..*d$$

- 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".

Distance measures used in feature selection

- In feature selection, each feature combination must be ranked based on a criterion function.
- Criteria functions can either be distances between classes, or the classification accuracy on a validation test set.
- If the criterion is based on e.g. the mean values/covariance matrices for the training data, distance computation is fast.
- Better performance at the cost of higher computation time is found when the classification accuracy on a validation data set (different from training and testing) is used as criterion for ranking features.
  - This will be slower as classification of the validattion data needs to be done for every combination of features.

Distance measures between classes

- How do be compute the distance between two classes:
  - Distance between the closest two points?
  - Maximum distance between two points?
  - Distance between the class means?
  - Average distance between points in the two classes?
  - Which distance measure?
    - Euclidean distance or Mahalanobis distance?
- Distance between K classes:
  - How do we generalize to more than two classes?
  - Average distance between the classes?
  - Smallest distance between a pair of classes?

Class separability measures

- How do we get an indication of the separability between two classes?
  - Euclidean distance between class means $|\mu_r - \mu_s|$
  - Bhattacharyya distance
    - Can be defined for different distributions
    - For Gaussian data, it is

$$B = \frac{1}{8}(\mu_r - \mu_s)^T \left(\frac{\Sigma_r + \Sigma_s}{2}\right)^{-1} (\mu_r - \mu_s) + \frac{1}{2}\ln \frac{\left|\frac{1}{2}(\Sigma_r + \Sigma_s)\right|}{\sqrt{|\Sigma_r||\Sigma_s|}}$$

  - Mahalanobis distance between two classes:

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$
$$\Sigma = N_1\Sigma_1 + N_2\Sigma_2$$

Examples of feature selection - Method 1 - Individual feature selection

- Each feature is treated individually (no correlation/covariance between features is consideren)
- Select a criteria, e.g. a distance measure
- Rank the feature according to the value of the criteria C(k)
- Select the set of features with the best individual criteria value
- Multiclass situations:
  - Average class separability or
  - C(k) = min distance(i,j) - worst case $\longleftarrow$ Often used

- Advantage with individual selection: computation time
- Disadvantage: no correlation is utilized.

Method 2 - Sequential backward selection

- Select l features out of d
- Example: 4 features x_1,x_2,x_3,x_4
- Choose a criterion C and compute it for the vector $[x_1,x_2,x_3,x_4]^T$
- Eliminate one feature at a time by computing $[x_1,x_2,x_3]^T$, $[x_1,x_2,x_4]_T$, $[x_1,x_3,x_4]^T$ and $[x_2,x_3,x_4]^T$

- Select the best combination, say $[x_1,x_2,x_3]^T$.

- From the selected 3-dimensional feature vector eliminate one more feature, and evaluate the criterion for $[x_1,x_2]^T$, $[x_1,x_3]_T$, $[x_2,x_3]^T$ and select the one with the best value.
- Number of combinations searched:
  1+1/2((d+1)d-l(l+1))

Method 3: Sequential forward selection

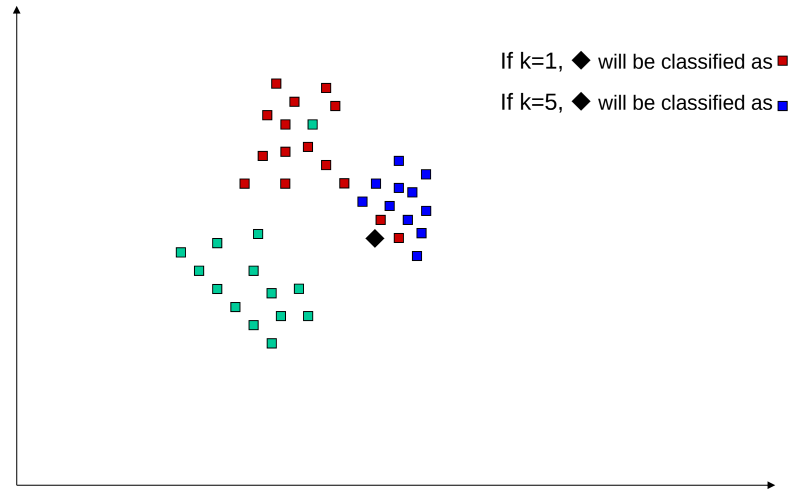
- Compute the criterion value for each feature. Select the feature with the best value, say x_1.
- Form all possible combinations of features x1 (the winner at the previous step) and a new feature, e.g. $[x_1,x_2]^T$, $[x_1,x_3]^T$, $[x_1,x_4]^T$, etc. Compute the criterion and select the best one, say $[x_1,x_3]^T$.
- Continue with adding a new feature.
- Number of combinations searched: ld-l(l-1)/2.
  - Backwards selection is faster if l is closer to d than to 1.

An alternative classifier

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 (measured 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



If k=1, ◆ will be classified as 🟥

If k=5, ◆ will be classified as 🟦

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_j 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*
- *Classification performance should **always** be computed on the test data set.*

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

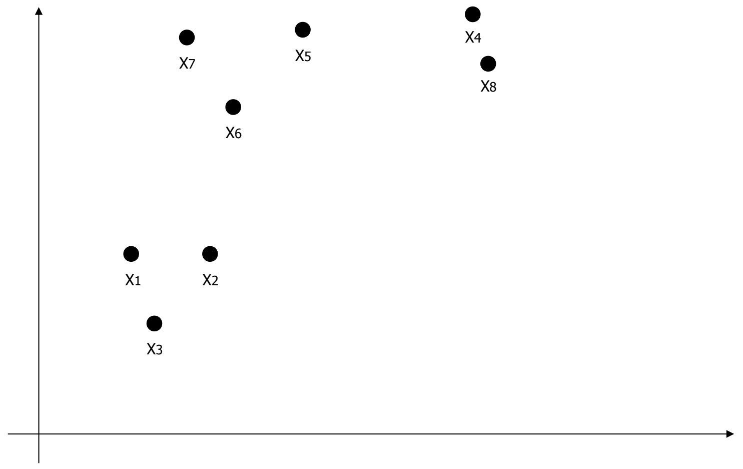
1. Start with assigning K cluster centers
   - k random data points, or the first K points, or K equally spaces points
   - For k=1:K, Set μ_k equal to the feature vector x_k for these points.
2. Assign each object/pixel x_i in the image to the closest cluster center using Euclidean distance.
   - Compute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

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

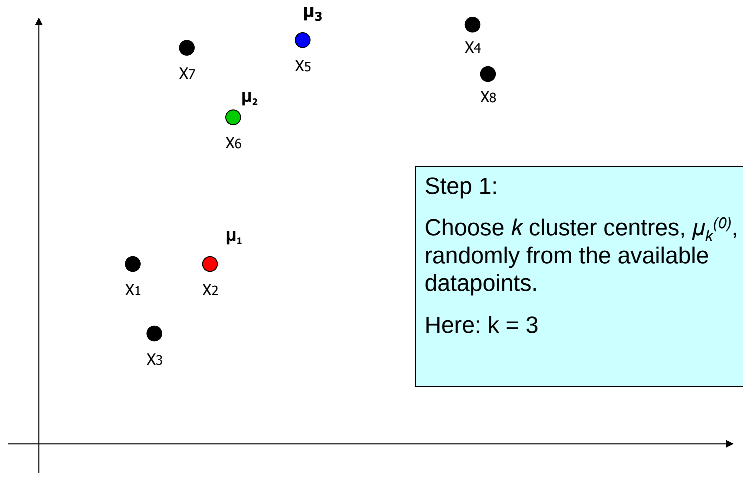
   - Assign x_i to the closest cluster (with minimum *r* value)
3. Recompute the cluster centers based on the new labels.
4. Repeat from 2 until #changes<limit.

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

k-means example



k-means example

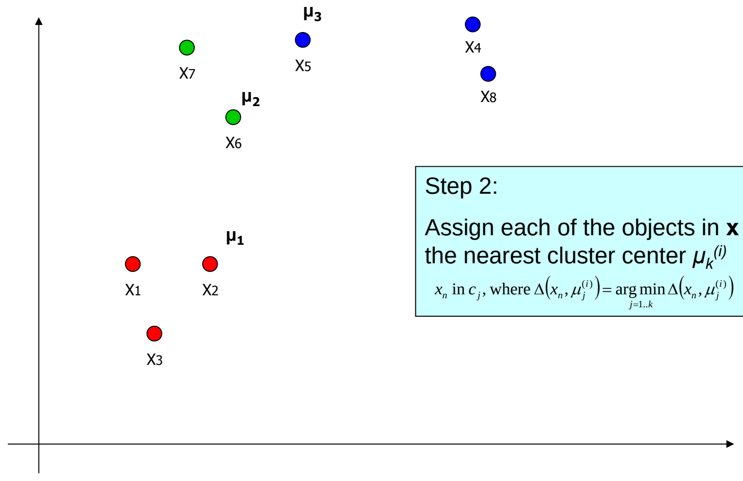


Step 1:

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

Here: k = 3

k-means example

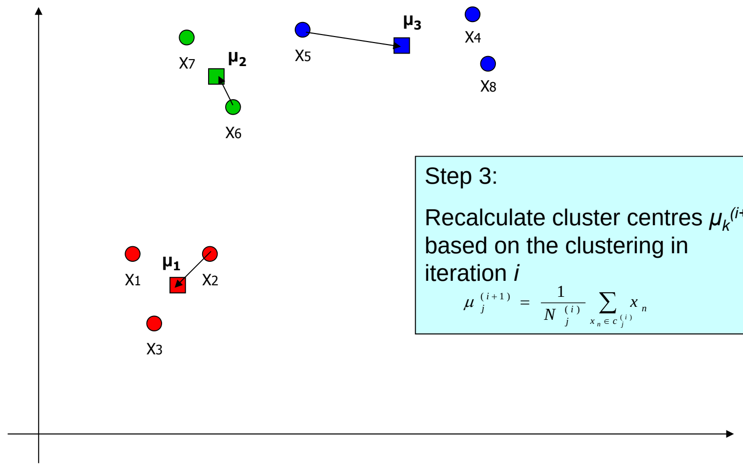


Step 2:

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

x_n in c_j, 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

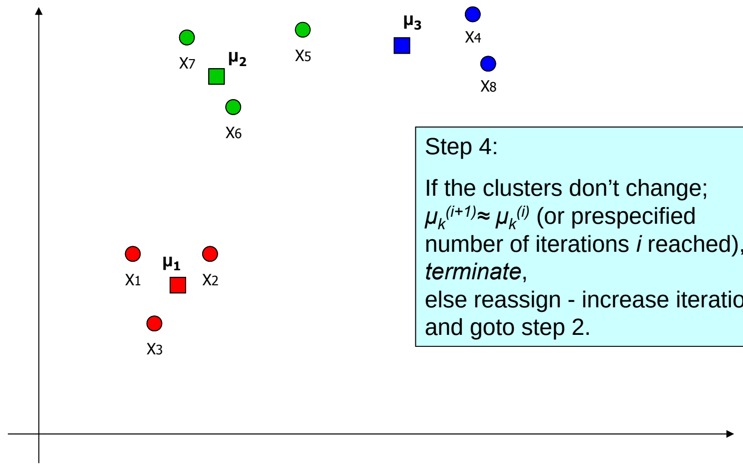


Step 3:

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

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

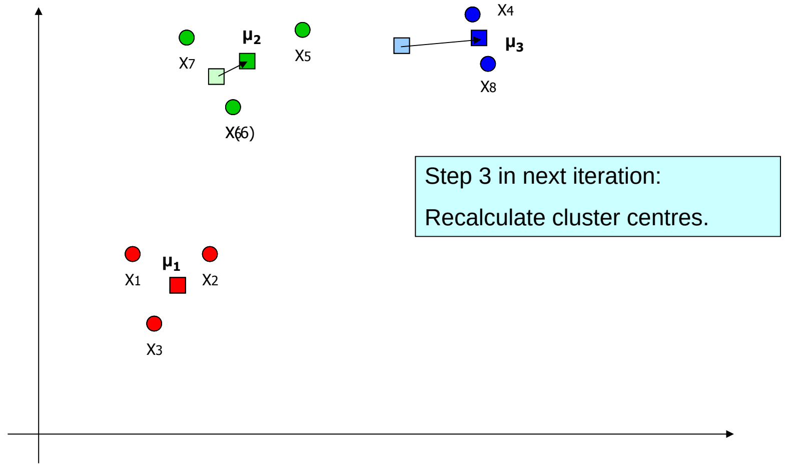
k-means example



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.

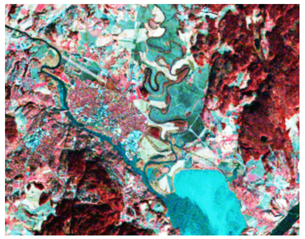
k-means variations

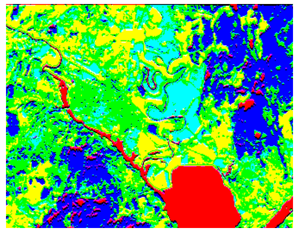
- The generic algorithm has many improvements
  - ISODATA – allow for merging and splitting of clusters
    - Among other things, this seeks to improve an initial ”bad” choice of *k*
  - k-medians is another variation
  - k-means optimizes a probabilistic model

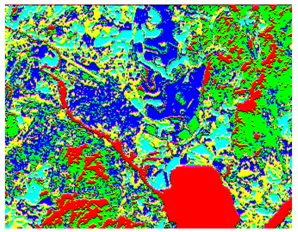
How do we determine k?

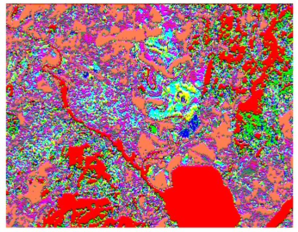
- The number of natural clusters in the data rarely corresponds to the number of information classes of interest.
- Cluster validity indices can give indications of how many clusters there are.
- Use cluster merging or splitting tailored to the application.
- Rule of thumb for practical image clustering:
  - 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

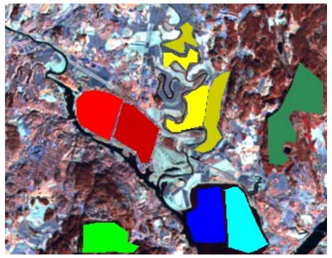
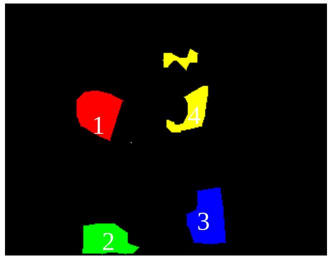
A classification example

Landsat image with 6 spectral bands
The 6 bands will be the features
Training areas and test areas shown in mask

Upper part: RGB-false color image created from bands 4,5 and 6 with training and test regions overlaid.

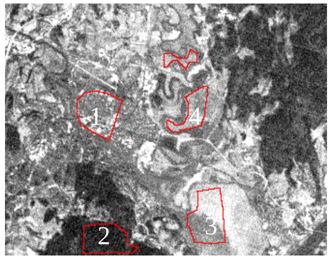
Lower part: image of training regions only

•

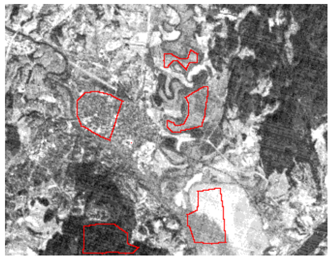
Visual inspection of feature 1

Class 2 (forest) seems to be well separated,
Maybe also class 1 (urban)



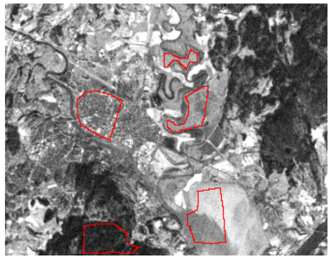
Visual inspection of feature 2

Class 2 (forest) seems to be well separated



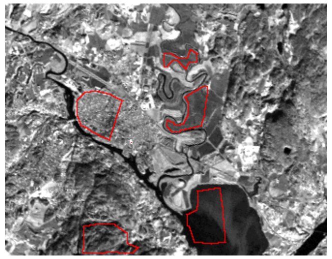
Visual inspection of feature 3

Class 2 (forest) seems to be well separated,
Class 1 (urban) seems to be well separated



Visual inspection of feature 4

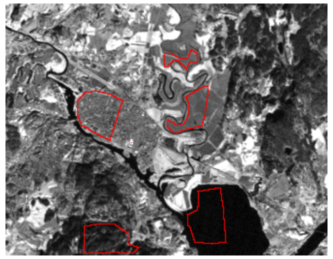
Class 1 (water) seems to be well separated,
Maybe also class 4 (agricultural)



Visual inspection of feature 5

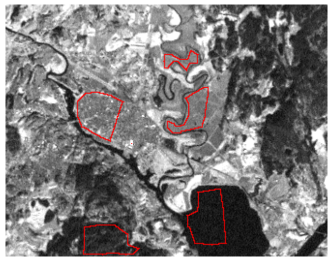
Water and forest appears similar
- but the variance might be
different

Urban and agricultural appears
similar – but the variance might
be different

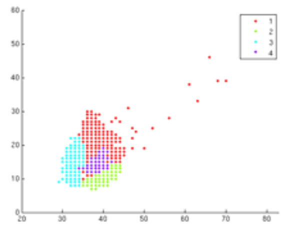


Visual inspection of feature 6

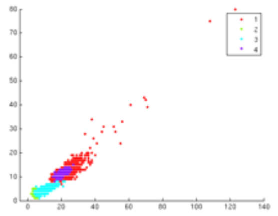
Seems similar to feature 5,
but with better contrast



Selected scatter plots (gscatter)

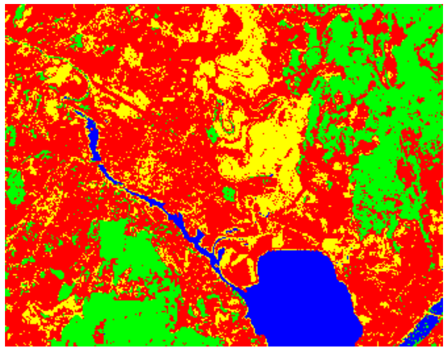


Scatterplot between feature 1 and 4



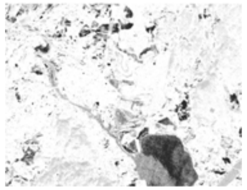
Scatterplot between feature 5 and 6

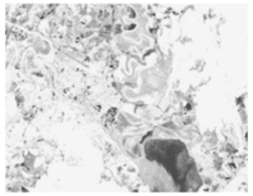
Classified images



The entire image classified to the most probable class

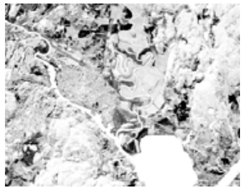
Display the posterior probabilities as images

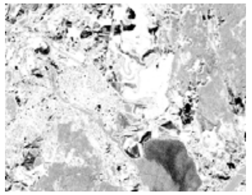

Posterior probability for class urban


Posterior probability for class forest

Dark values:
Probabilities close to 0

Bright values:
Probabilities close to 1


Posterior probability for class water


Posterior probability for class agricultural

Confusion matrix for the training set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1340	2	0	310
Class 1	43	1253	0	2
Class 3	0	0	1738	0
Class 4	131	3	0	1266

Accuracy per class:
Class1: 81%
Class2: 96%
Class3: 100%
Class4: 90%

Averaged over all classes: 91.7%

Confusion matrix for the test set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1474	3	1	251
Class 1	513	2311	0	0
Class 3	14	0	1953	0
Class 4	213	2	0	1390

Accuracy per class:
Class1: 85%
Class2: 81%
Class3: 98%
Class4: 86%

Averaged over all classes: 87.5%

Learning goals for this lecture

- Understand how different measures of classification accuracy work:
  - Confusion matrix
  - Sensitivity/specifity/TP/TN/FP/FN
  - Average classification accuracy
- Be familiar with the curse of dimensionality and the importance of selecting few, but good features
- Know simple forward and backward feature selection.
- Understand kNN-classification
- Understand the difference between supervised and unsupervised classification
- Understand the Kmeans-algorithm.

Next week

- Dimensionality reduction by linear feature transforms
  - Create new features in a lower-dimensional space from a linear combination of the input features
  - Principal component transform
  - Fisher's linear discriminant transform