IN 5520
7.10.20
Introduction to classifiction

Anne Solberg (anne@ifi.uio.no)

- Introduction to classification

Based on handout from Pattern Recognition by Theodoridis

Lectures on zoom

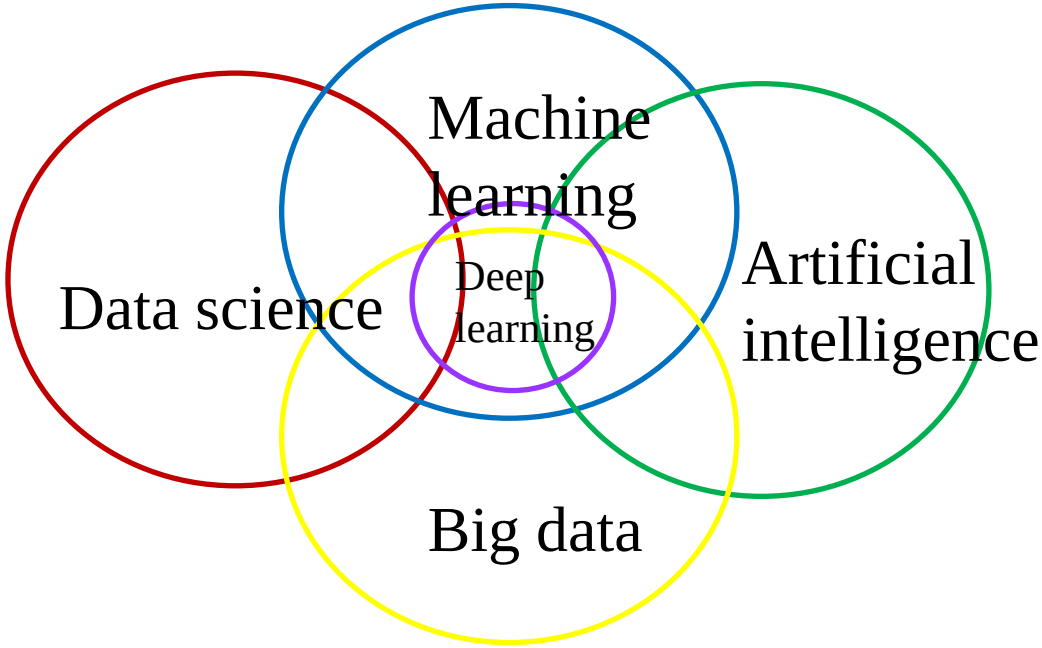
- The lectures will be recorded
- Don’t let this stop you to ask questions!!!
- During recording – ask your questions in chat
- At intervals marked «Questions?» I will repeat and answer these
- We take a short break in recording at halftime
  - This prevents the file to be too big
  - Ask oral questions during the break

- We are also open for oral questions after the lecture

Plan for this lecture:

- Introduction to machine learning/classification
- Explain the relation between thresholding and classification with 2 classes
- Background in probability theory
- Bayes rule
- K-nearest Neighbor classifier
- Classification with a Gaussian density and a single feature
  - Linear boundaries in feature space
- Minimum distance classifier
- Briefly: training and testing a classifier

Introduction to machine learning



The term **machine learning** is today used broadly and cover linear regression, statistical classification, neural networks ++

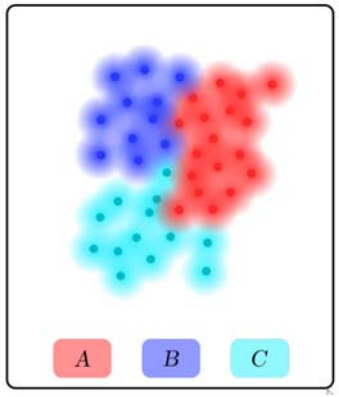
Keyworks: approaces that learn from data

Introduction to classification

- Given a feature vector **x** computed from a data set
- We want to predict the value of an unkown variable y given **x**.
- If y is continuous, the simplest model could be linear regression.
- If y is categorical, we want to assign **x** into a set of predefined classes c=1,...C, and y=c if **x** belongs to class c.
- A classification problem can be solved using supervised, unsupervised or reinforcement learning.

Unsupervised learning

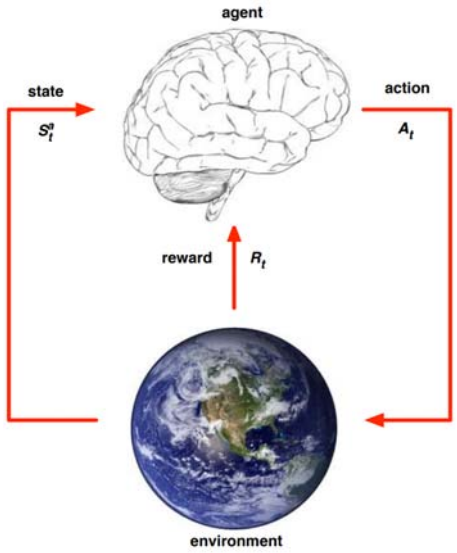
- Our training set consists of input $\mathbf{x}$ only:
- We do not have any labeled data. Our goal is to find an underlaying structure of the data.
- Examples:
  - Data clustering
  - Anomality detection
  - Signal generation
  - Signal compression



Variational autoencoder (latent space z)

Reinforcement learning

- Reinforcement Learning ~ Science of decision making
- In RL an agent learns from the experiences it gains by interacting with the environment.
- The goal is to maximize an accumulated reward given by the environment.
- An agent interacts with the environment via states, actions and rewards.
- Example: games like chess or black jack.



RL Course by David Silver

Supervised learning

- Given a training set with input $\mathbf{x}$ and desired output y:

- The goal is to create a function f that "approximates" this mapping:

- Hope that this generalizes well to unseen examples:

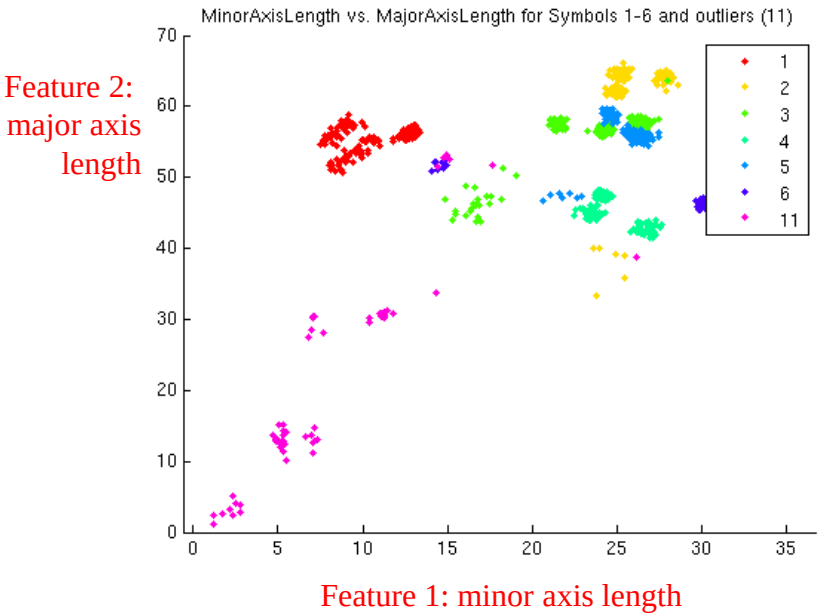
Examples:

  - Classification, regression, object detection,
  - Segmentation, image captioning.

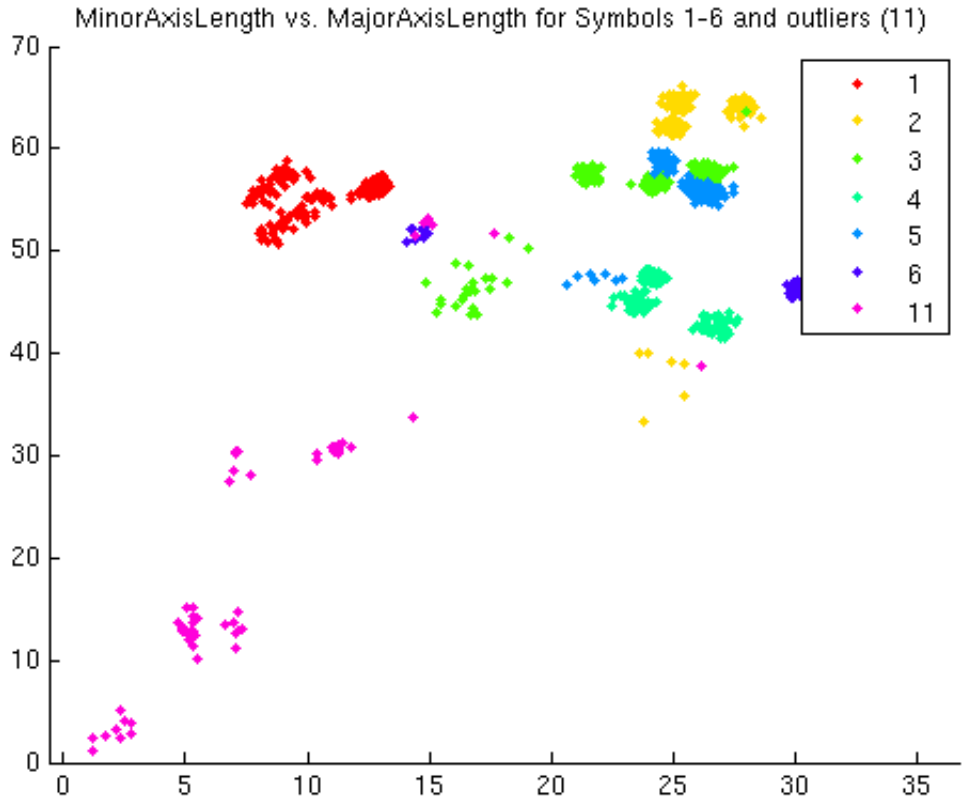
Supervised learning is normally used for image classification and the coming lectures are about it.

Scatter plots - reminder

- A 2D scatter plot is a plot of feature values for two different features. Each object's feature values are plotted in the position given by the features values, and with a class label telling its object class.
- A scatter plott visualize the space spanned by 2 or more features: called the **feature space**
- Matlab: gscatter(feature1, feature2, labelvector)
- Classification is done based on *more than two features*, but this is difficult to visualize.
- Features with good class separation show clusters for each class, but different clusters should ideally be separated.



Discriminating between classes in a scatter plot



Concepts in classification

- In the following lectures we will cover these topics related to classification:
  - Training set
  - Validation set
  - Test set
  - Classifier accuracy/confusion matrices.
  - Computing the probability that an object belongs to a class.
  - Bayes rule
  - Discriminant functions/Decision boundaries
  - Normal distribution, mean vector and covariance matrices
  - kNN classification
  - Support vector machines
  - Unsupervised classification/clustering
  - Dimensionality reduction

Training data set

- A set of known objects from all classes for the application.
- Typically a high number of samples from each class.
  - Approximately equal number of samples from each class if possible
- Used to **determine the parameters of the given classification algorithm**.
  - Example 1: fit a Gaussian density to the data
  - Example 2: find a Support Vector Model that fits the data
  - Example 3: find the weights in a neural net

Validation data set

- A set of known objects from all classes for the application.
- Typically a sufficiently high number of samples from each class.
- Used to **optimize hyperparameters**, e.g.
  - Compare feature selection algorithms
  - Compare classifiers
  - Compare network architectures

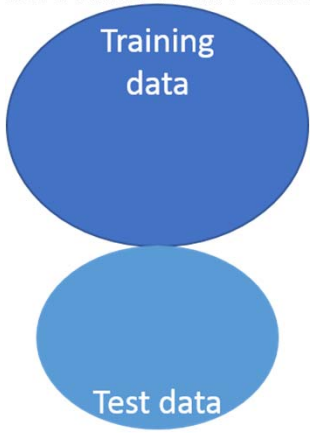
Test data set

- A set of known objects from all classes for the application.
- Typically a high number of samples from each class.
- Used ONCE to **estimate the accuracy of a trained model**.
- Should never optimize any parameters on the test data set.

Challenge: domain shift

- Training and test data from same distribution

Training and test data from different distributions

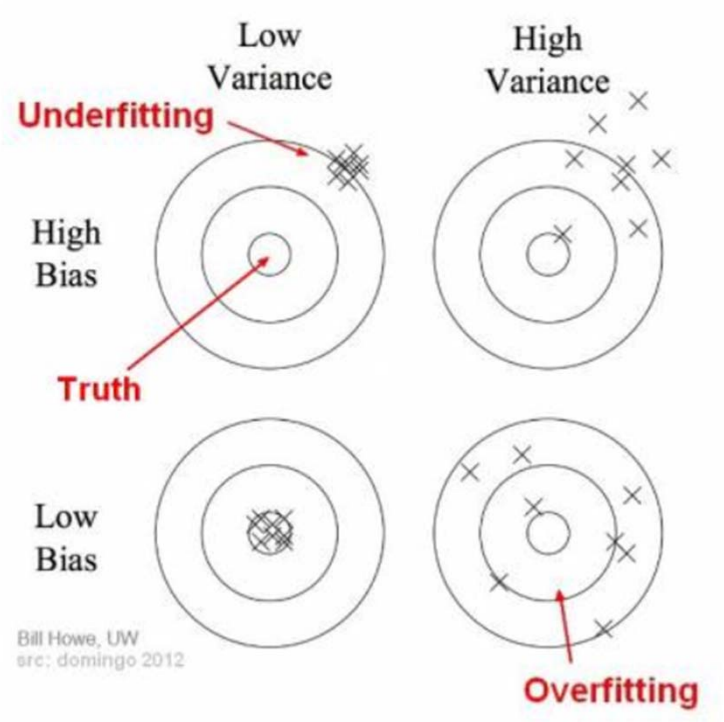


Generalization describes how well a classifiier is expected to perform on new data.

- Ohen ghoor!

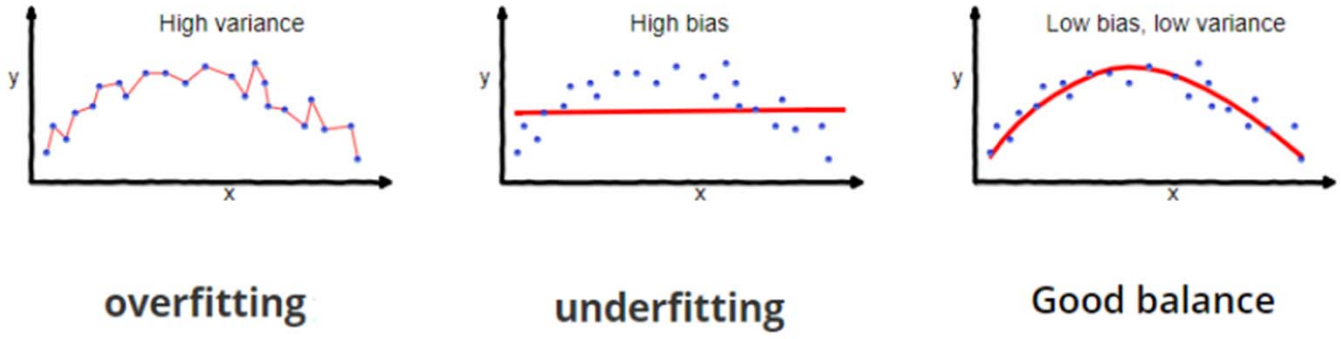
Bias-Variance tradeoff

- Given a model we fit to the data, e.g Y=f(X)+e
  Y: true values, X: measurements, e: error
  Estimate of Y: $\widehat{f(x)}$
- Bias: the difference between the average prediction and the correct value $E\left(\widehat{f(x)}\right) - f(x)$
- Variance: The variance of the predictions $\widehat{f(x)}$

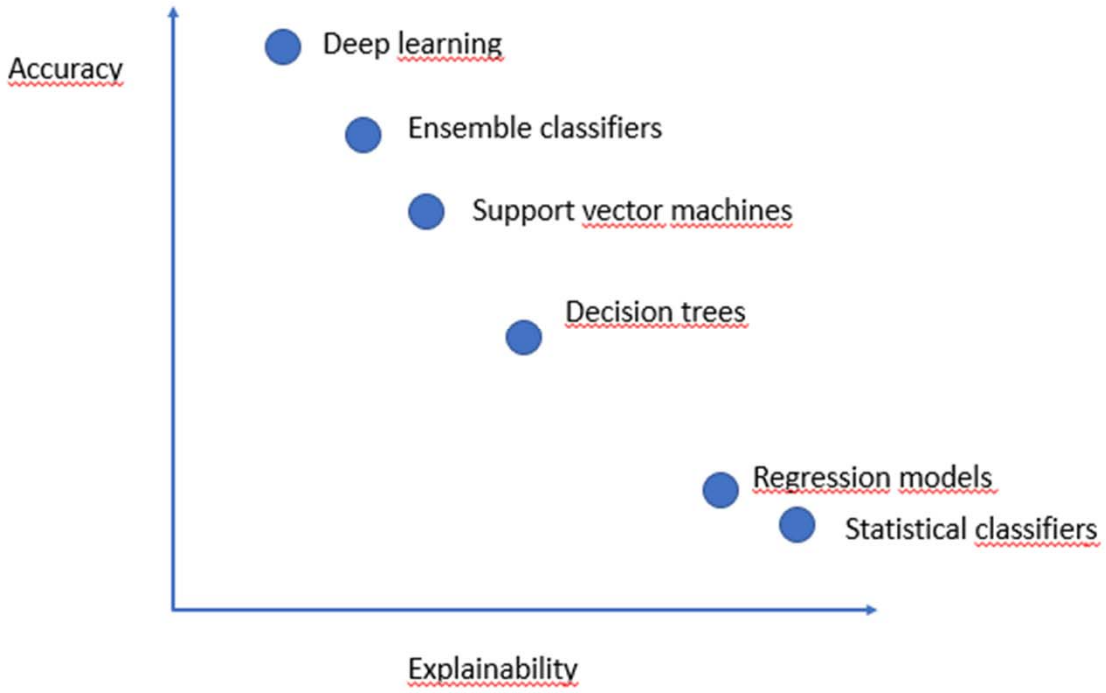


Underfitting and overfitting

- Underfitting: the model is not able to capture the underlying pattern of data
  - High bias, low variance
  - In our setting: high error rate on training data and validation data
- Overfitting: the model captures the noise with the underlying pattern of data
  - Low bias, high variance
  - In our setting: low error rate on training data, high error on validation data



Explaining the result



Always use the simplest model that can perform well on your validation data!

Explaining the decisions

- Where we are currently using deep learning:

- This is a cat

- Goal:

- This is a cat (p=0.87)
- It has fur, whiskers, claws.
- Most characteristic feature:


QUESTIONS
ANSWERS

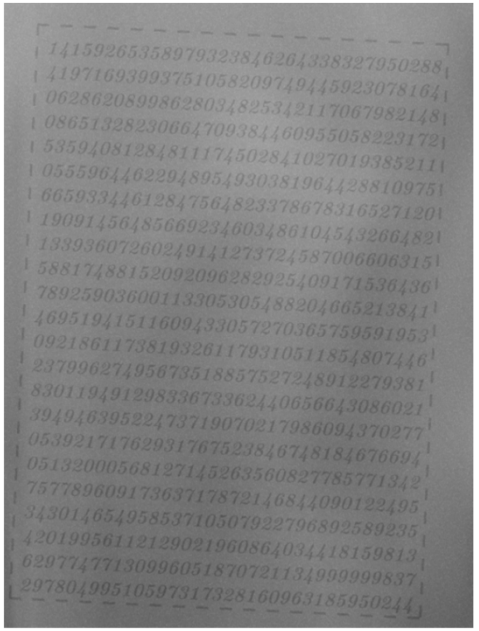
Thesholding as a binary classification problem

Gray level of foreground and background pixels modelled as probability density functions

From INF2310: Thresholding (Gonzalez and Woods 10.3)

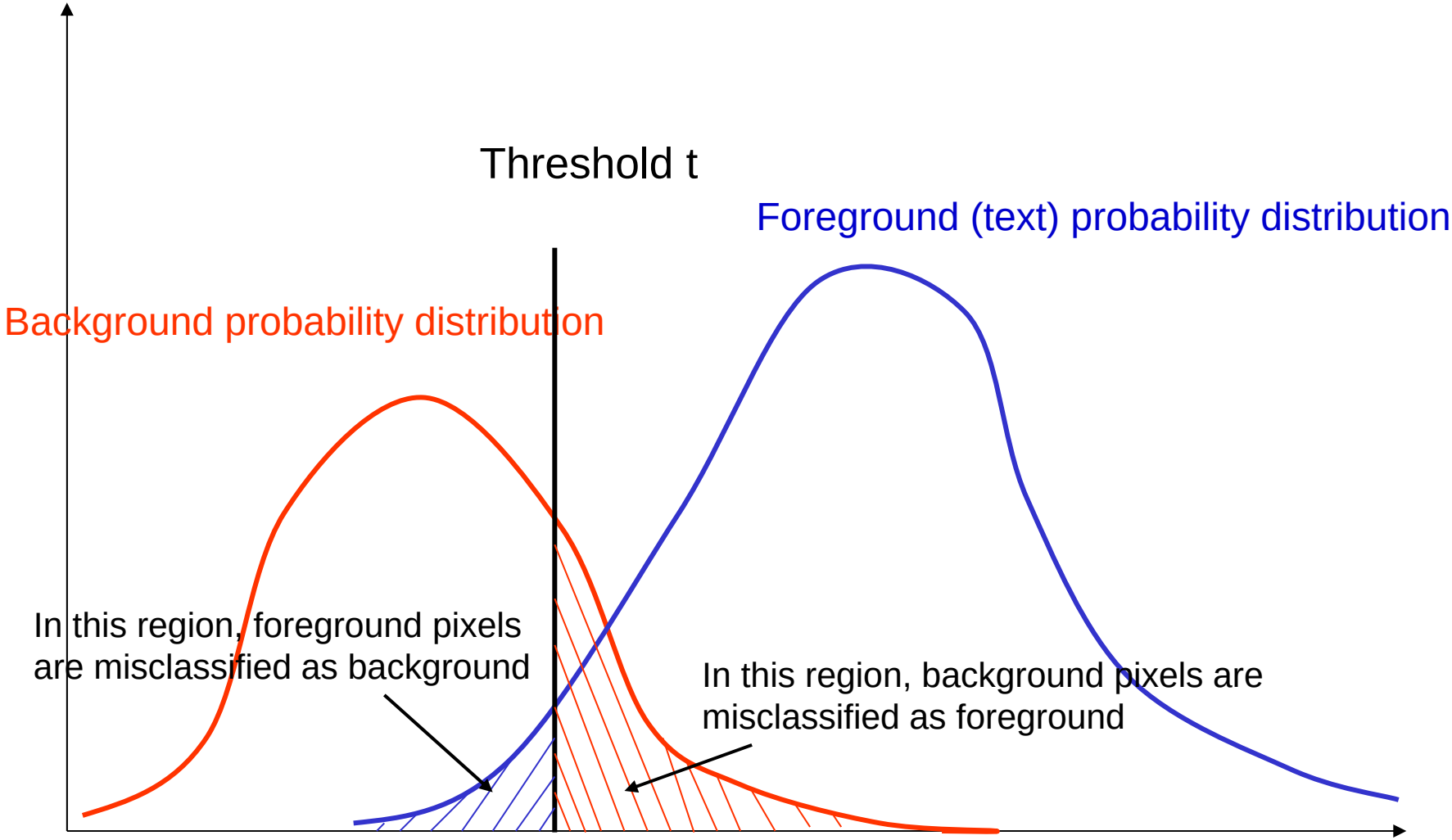
- Basic thresholding assigns all pixels in the image f(x,y) to one of 2 classes: foreground or background

$$g(x,y) = \begin{cases} 0 \text{ if } & f(x,y) \le T \\ 1 \text{ if } & f(x,y) > T \end{cases}$$



- g(x,y) is the thresholded image
- This can be seen as a 2-class classification problem based on **a single feature, the gray level.**
- The 2 classes are background and foreground, and the threshold T defines the border between them.

Classification error for thresholding

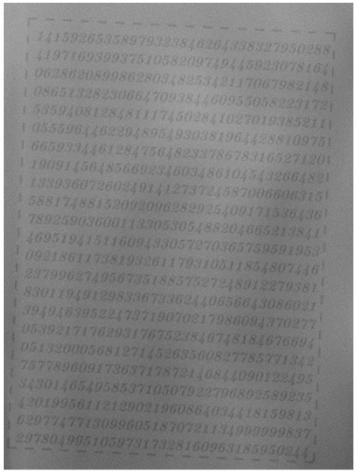


Histograms

- h(i)=number of pixels with gray level i
- $\sum_{i=0}^{NxM} h(i) = NxM\ for\ a\ NxM\ image$
- Normalized hisogram p(i)=h(i)/(NxM)
- For a normalized histogram
- $\sum_{i=0}^{NxM} p(i) = 1$

Classification error for thresholding

- We assume that b(z) is the normalized histogram for background *b(z)* and *f(z)* is the normalized histogram for foreground.
- The histograms are estimates of the probability distribution of the gray levels in the image.
- Let *F* and *B* be the prior probabilities for background and foreground(*B+F=1*)
- The normalized histogram for the image is then given by



$$p(z) = B \cdot b(z) + F \cdot f(z)$$

- The probability for misclassification given a treshold t is:

$$E_B(t) = \int_{-\infty}^{t} f(z)dz$$

$$E_F(t) = \int_{t}^{\infty} b(z)dz$$

Find T that minimizes the error

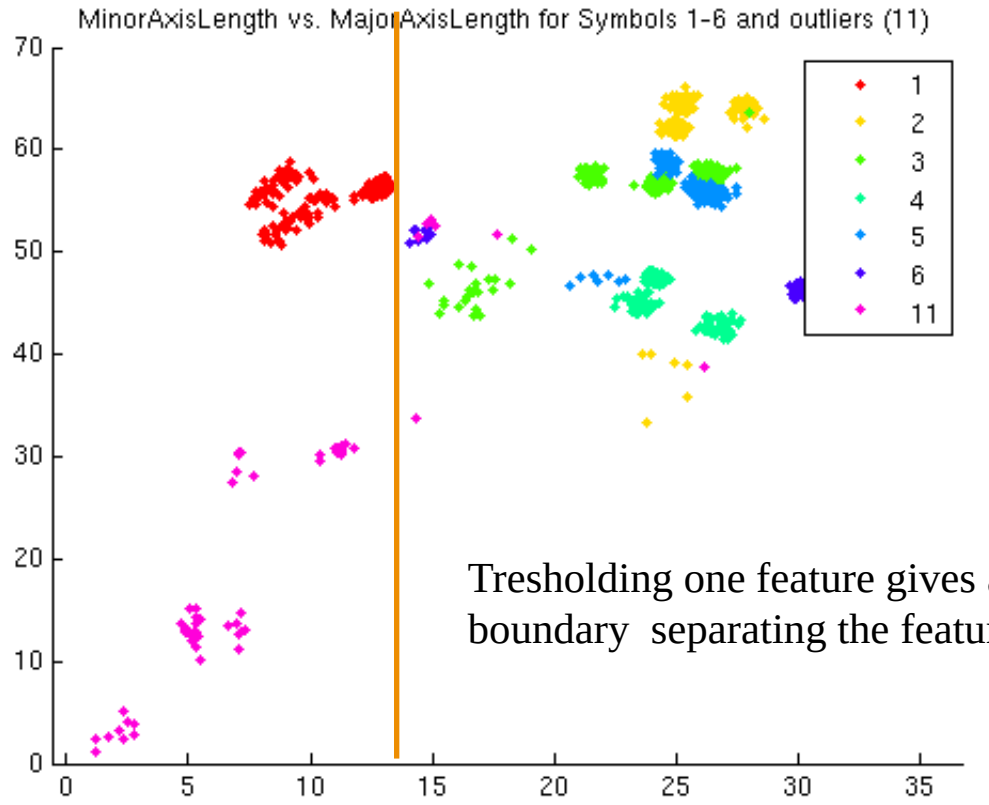
$$E(t) = F\int_{-\infty}^{t} f(z)dz + B\int_{t}^{\infty} b(z)dz$$

$$\frac{dE(t)}{dt} = 0 \Rightarrow F \cdot f(T) = B \cdot b(T)$$

Minimum error is achieved by setting T equal to the point where the probabilities for foreground and background are equal.

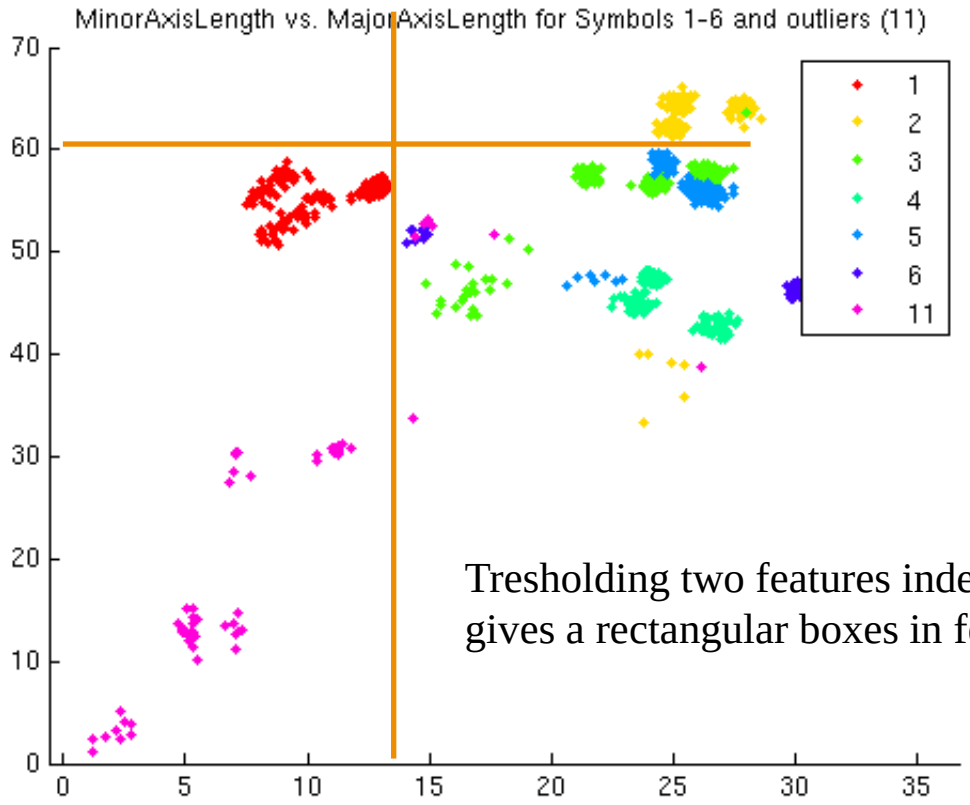
The locations in feature space where the probabilities are equal is called decision boundary in classification.
The goal of classification is to find the boundaries.

Partitioning the feature space using thresholding – 1 feature and 1 threshold



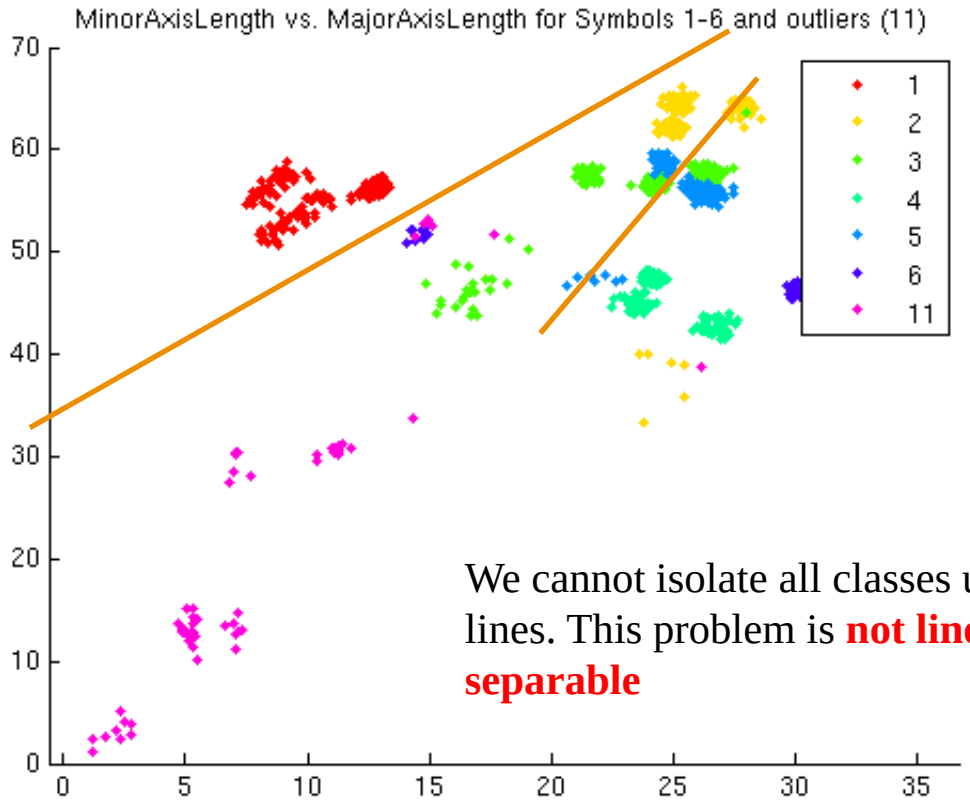
Tresholding one feature gives a linear boundary separating the feature space

Partitioning the feature space using thresholding – 2 features and 2 thresholds



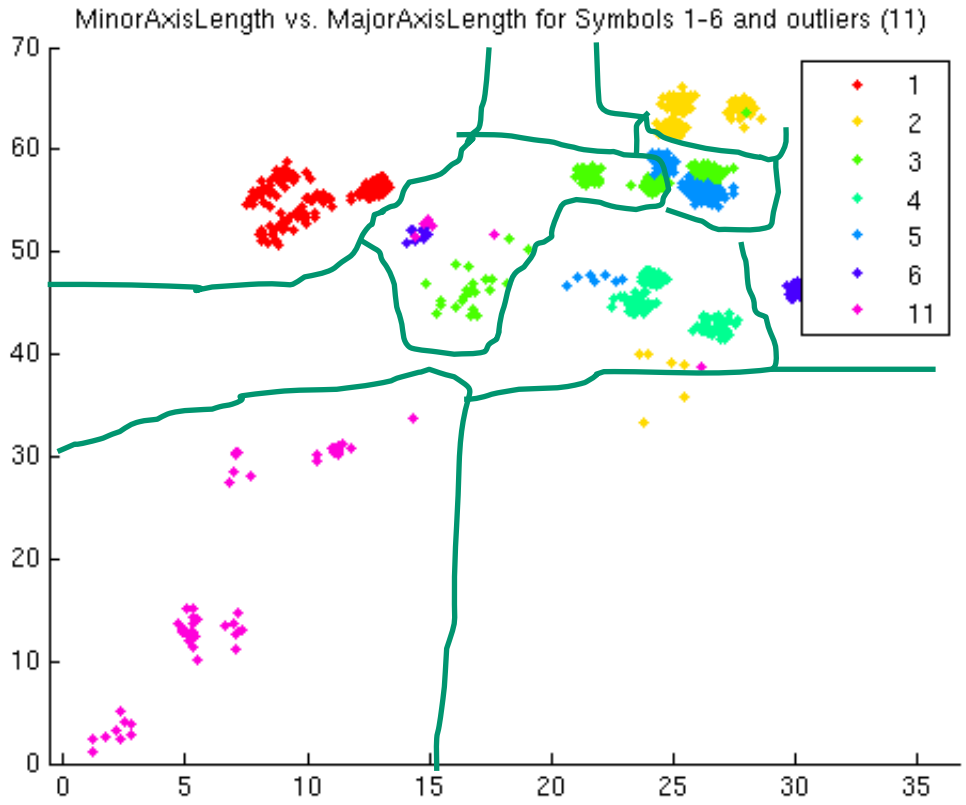
Tresholding two features independently gives a rectangular boxes in feature space

Can we find a line with better separation?



We cannot isolate all classes using straight lines. This problem is **not linearly separable**

The goal of classification: partitioning the feature space with smooth boundaries




QUESTIONS
ANSWERS

Types of classifiers

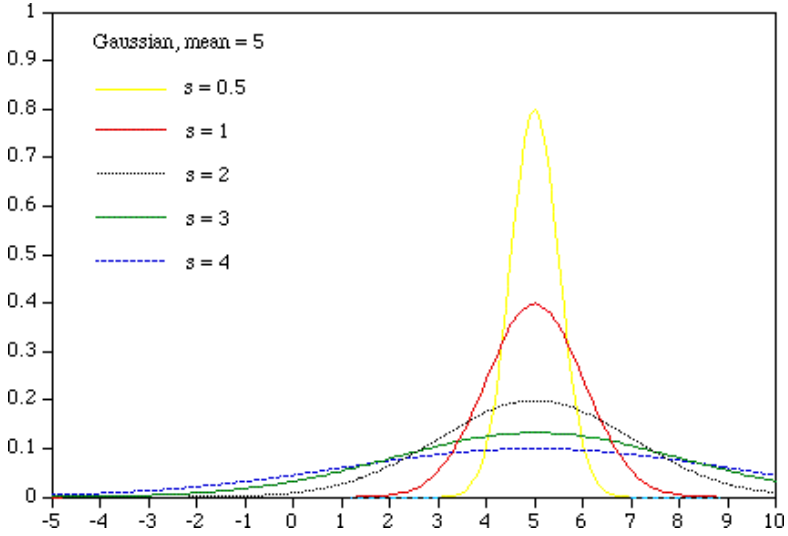
- Parametric probability distribution:
  - Assume a type of probability distribution, fit the parameters of the distribution to the data.
- Non-parametric probability distribution:
  - Estimate the distribution locally using e.g. Parzen windows
  - K Nearest Neighbor classifier
- Minimize loss function e.g. separation between classes
  - Support Vector Machines: maximize the distance between the closest points in two classes
  - Multilayer feed-forward network: minimize the cross entropy between the true labels and the estimated labels

Distributions, standard deviation and variance

- A univariate (one feature) Gaussian distribution (normal distribution) is specified given the mean value μ and the variance σ^2:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Variance σ^2, standard deviation σ



Two Gaussian distributions for thresholding a single feature

- Assume that *b(z)* and *f(z)* are Gaussian distributions, then

$$p(z) = \frac{B}{\sqrt{2\pi\sigma_B^{\,2}}} e^{-\frac{(x-\mu_B)^2}{2\sigma_B^{\,2}}} + \frac{F}{\sqrt{2\pi\sigma_F^{\,2}}} e^{-\frac{(x-\mu_F)^2}{2\sigma_F^{\,2}}}$$

- μ_B and μ_F are the mean values for background and foreground.
- σ_B^2 and σ_F^2 are the variance for background and foreground.

The 2-class classification problem summarized

- Given two Gaussian distributions *b(z)* and *f(z)*.
- The classes have prior probabilities F and B.
- Every pixel should be assigned to the class that minimizes the classification error.
- The classification error is minimized at the point where F f(z) = B b(z).

- What we will do now is to generalize to K classes and D features.

The goal of classification

- We estimate the decision boundaries (equivalent to the threshold for multivariate data) based on training data.
- Classification performance is always estimated on a separate ”test” data set.
  - We try to measure the generalization performance.
- The classifier should perform well when classifying new samples
  - Have lowest possible classification error.
- We often face a tradeoff between classification error on the training set and generalization ability when determining the complexity of the decision boundary.

Bayesian decision theory

- A fundamental statistical approach to pattern classification.
- Named after Thomas Bayes (1702-1761), an english priest and matematician.
- It combines prior knowledge about the problem with a probability distribution function.
- The most central concept (for us) is Bayes decision rule.

Bayes rule for a classification problem

- Suppose we have J, j=1,…J classes.
- ω_j is the class label for a pixel,
  x is the observed gray level/feature vector.
  - We can classify a single pixel, an object, or an entire image
- We can use Bayes rule to find an expression for the class with the highest probability:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)}$$

$$\text{posterior probability} = \frac{\textit{likelihood} \times \text{prior probability}}{\text{normalizing factor}}$$

- For thresholding, $P(\omega_j)$ is the prior probability for background or foreground. If we don't have special knowledge that one of the classes occur more frequent than other classes, we set them equal for all classes. ($P(\omega_j)=1/J$, j=1.,,,J).
- Small p means a probability distribution
- Capital P means a probability (scalar value between 0 and 1)

Bayes rule explained

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)}$$

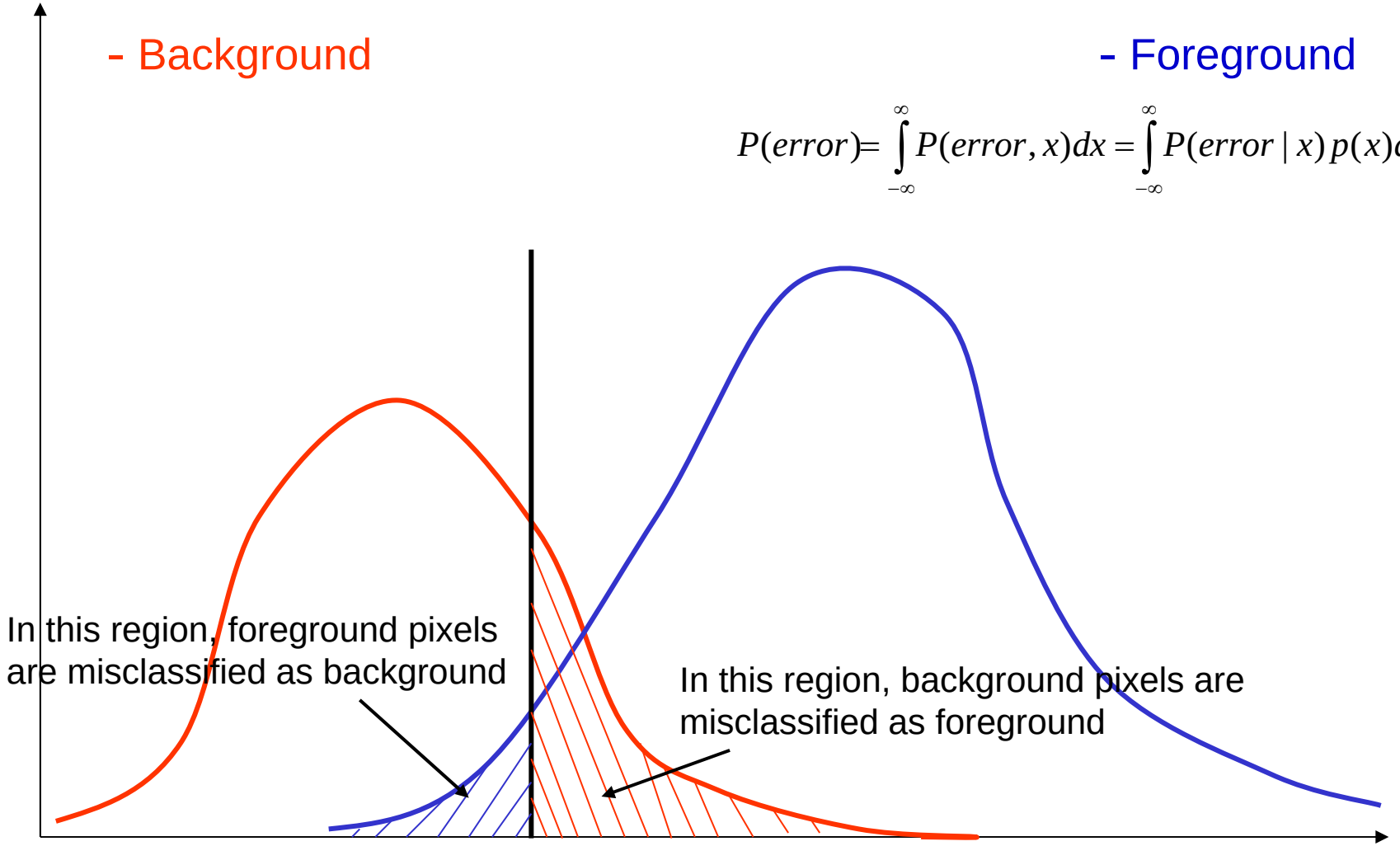
- $p(x|\omega_j)$ is the probability density function that models the likelihood for observing gray level x if the pixel belongs to class ω_j.
  - Typically we assume a type of distribution, e.g. Gaussian, and the mean and covariance of that distribution is fitted to some data that we know belong to that class. This fitting is called classifier training.
- $P(\omega_j|x)$ is the posterior probability that the pixel actually belongs to class ω_j. We will soon se that the the classifier that achieves the minimum error is a classifier that assigns each pixel to the class ω_j that has the highest posterior probability.
- p(x) is just a scaling factor that assures that the probabilities sum to 1.

Probability of error

- If we have 2 classes, we make an error either if we decide ω_1 if the true class is ω_2 , and if we decide ω_2 if the true class is ω_1.
- If $P(\omega_1|x) > P(\omega_2|x)$ we have more belief that x belongs to ω_1, and we decide ω_1.
- The probability of error is then:

$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) \text{ if we decide } \omega_2 \\ P(\omega_2 \mid x) \text{ if we decide } \omega_1 \end{cases}$$

Back to classification error for thresholding



Minimizing the error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\, p(x)dx$$

- When we derived the optimal threshold (INF 2310), we showed that the minimum error was achieved for placing the threshold (or *decision boundary* as we will call it now) at the point where

$$P(\omega_1|x) = P(\omega_2|x)$$

- This is still valid.

Bayes classification with J classes and D features

- How do we generalize:
  - To more the one feature at a time
  - To J classes
  - To consider loss functions (that some errors are more costly than others)

Bayes rule with J classes and d features

- If we measure d features, $\boldsymbol{x}$ will be a d-dimensional feature vector.
- Let $\{\omega_1,\ldots,\omega_J\}$ be a set of J classes.
- The posterior probability for class j is now computed as

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j)P(\omega_j)}{p(\mathbf{x})}$$

$$p(x) = \sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j)P(\omega_j)$$

- Still, we assign a pixel with feature vector x to the class that has the highest posterior probability:

$$\text{Decide } \omega_1 \text{ if } \mathrm{P}(\omega_1 \mid \mathbf{x}) \geq P(\omega_j \mid \mathbf{x}), \text{ for all } j \neq i$$

Feature space and decision regions

- The minimum error criterion (or a different classifier criterion) partitions feature space into disjoint region R_i.
- The **decision surface** separates the regions in multidimensional space.
- For minimum error, the equation for the decision surface is

$$P(\omega_j \mid \mathbf{x}) = P(\omega_k \mid \mathbf{x})$$

Exactly where the threshold was set in minimum error thresholding!

Discriminant functions

- The decision rule

$$\text{Decide } \omega_1 \text{ if } P(\omega_1 \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x}), \text{ for all } j \neq i$$

can be written as assign **x** to ω_1 if

$$g_i(\mathbf{x}) > g_j(\mathbf{x})$$

- The classifier computes J discriminant functions $g_i(\mathbf{x})$ and selects the class corresponding to the largest value of the discriminant function.
- Since classification consists of choosing the class that has the largest value, a scaling of the discriminant function $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$ will not effect the decision if f is a monotonically increasing function.
- This can lead to simplifications as we will soon see.

Equivalent discriminant functions

- The following choices of discriminant functions give equivalent decisions:

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

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

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

- The effect of the decision rules is to divide the feature space into c decision regions R_1,.....R_c.
- If $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$, then $\mathbf{x}$ is in region R_i.
- The regions are separated by decision boundaries, surfaces in features space where the discriminant functions for two classes are equal

The Gaussian density - univariate case (a single feature)

- To use a classifier we need to select a probability density function $p(x|\omega_i)$.
- The most commonly used probability density is the normal (Gaussian) distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right]$$

with expected value (or mean) $\mu = \mathrm{E}[x] = \int_{-\infty}^{\infty} x p(x) dx$

and variance $\sigma^2 = E\left[(x-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x) dx$

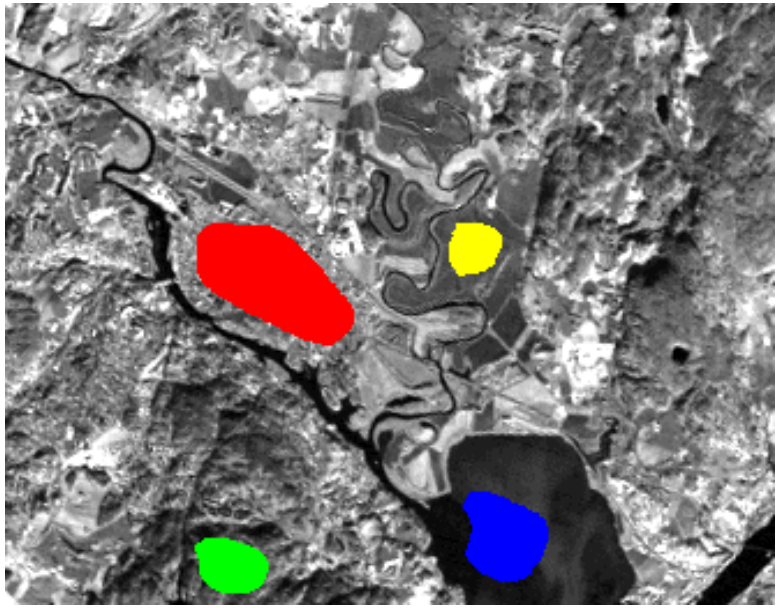

QUESTIONS
ANSWERS

Training data

- Collect image data with labels for each object type
  - Is the object a pixel, region, or image ? This varies.
- Each object category is assigned a number, called the class label.
- Example: Class 1: ‘dog’, class 2: ‘cat’, class 3: ‘bird’
- Integer (categorical) labels are used for implementation.

Example: image and training masks for pixel-based classification

The masks contain labels for the training data.
If label=1, then the pixel belongs to class 1 (red), and so on.
If a pixel is not part of the training data, it will have label 0.
A pixel belonging to class k will have value k in the mask image.
The mask is often visualized in pseudo-colors on top of the input image, where each class is assign a color.
We should have a similar mask for the test data.



How do we make training masks?

- Use a program that allows us to draw different shapes on top of an image, and save the result as a mask image
  - Matlab: e.g. drawpolygon:
    - pedestrian1 = imread('bike_001.png');
    - imshow(pedestrian1);
    - pedestrian1_ped=drawpolygon();
    - ped1 = createMask(pedestrian1_ped);
    - imwrite(ped1,"bike001_background.png");
- Depending on the task, we either draw:
  - Regions within each class (avoiding boundaries) for general classification
  - A larger Region-of-interest for region-based tasks like traffic sign segmentation
  - A detailed boundary for segmentation tasks

Training a univariate Gaussian classifier

- To be able to compute the value of the discriminant function, we need to have an estimate of μ_j and σ_j^2 for each class j.
- Assume that we know the true class labels for some pixels and that this is given in a mask image. The mask has N_k pixels for each class.
- Training the classifier then consists of computing μ_j and σ_j^2 for all pixels with class label j in the mask file.
- They are computed from training data as:
- For all pixels x_i with label k in the training mask, compute

$$\mu_k = \frac{1}{N_k} \sum_{i=i}^{N_k} x_i$$

$$\sigma_k^2 = \frac{1}{N_k} \sum_{i=i}^{N_k} (x_i - \mu_k)^2$$

Training a pixel-based classifier using masks

```
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)

$$\mu_k = \frac{1}{N_k} \sum_{i=i}^{N_k} x_i$$

$$\sigma_k^2 = \frac{1}{N_k} \sum_{i=i}^{N_k} (x_i - \mu_k)^2$$

How do to classification with a univariate Gaussian (1 feature)

- Decide on values for the prior probabilities, $P(\omega_j)$. If we have no prior information, assume that all classes are equally probable and $P(\omega_j)=1/J$.
- Estimate μ_j and σ_j^2 based on training data based on the formulae on the previous slide. (Training)
- For each pixel in a new image:

  For class j=1,....J, compute the discriminant function

$$P(\omega_j \mid x) = p(x \mid \omega_j)P(\omega_j) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp\left[-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma_j}\right)^2\right]P(\omega_j)$$

  Assign pixel x to the class C with the highest value of $P(\omega_j|x)$ by setting label_image(x,y)= C

The result after classification is an image with class labels corresponding to the most probable class for each pixel.

We compute the classification error rate from an independent test mask.

Minimum distance classification

- As we will see next week, if we assume that all features are uncorrelated and have unit variance, the Gaussian model reduces to
  - Assign the feature vector to the class with the closest mean value measured by Euclidean distance.
  - Compute the distance to all class means and choose the class with the closest mean.
  - Euclidean distance for a feature vector of dimension F:
  - $\sqrt{\sum_{f=1}^{F}\left(x_f - \mu_{f,K}\right)^2}$


QUESTIONS
ANSWERS

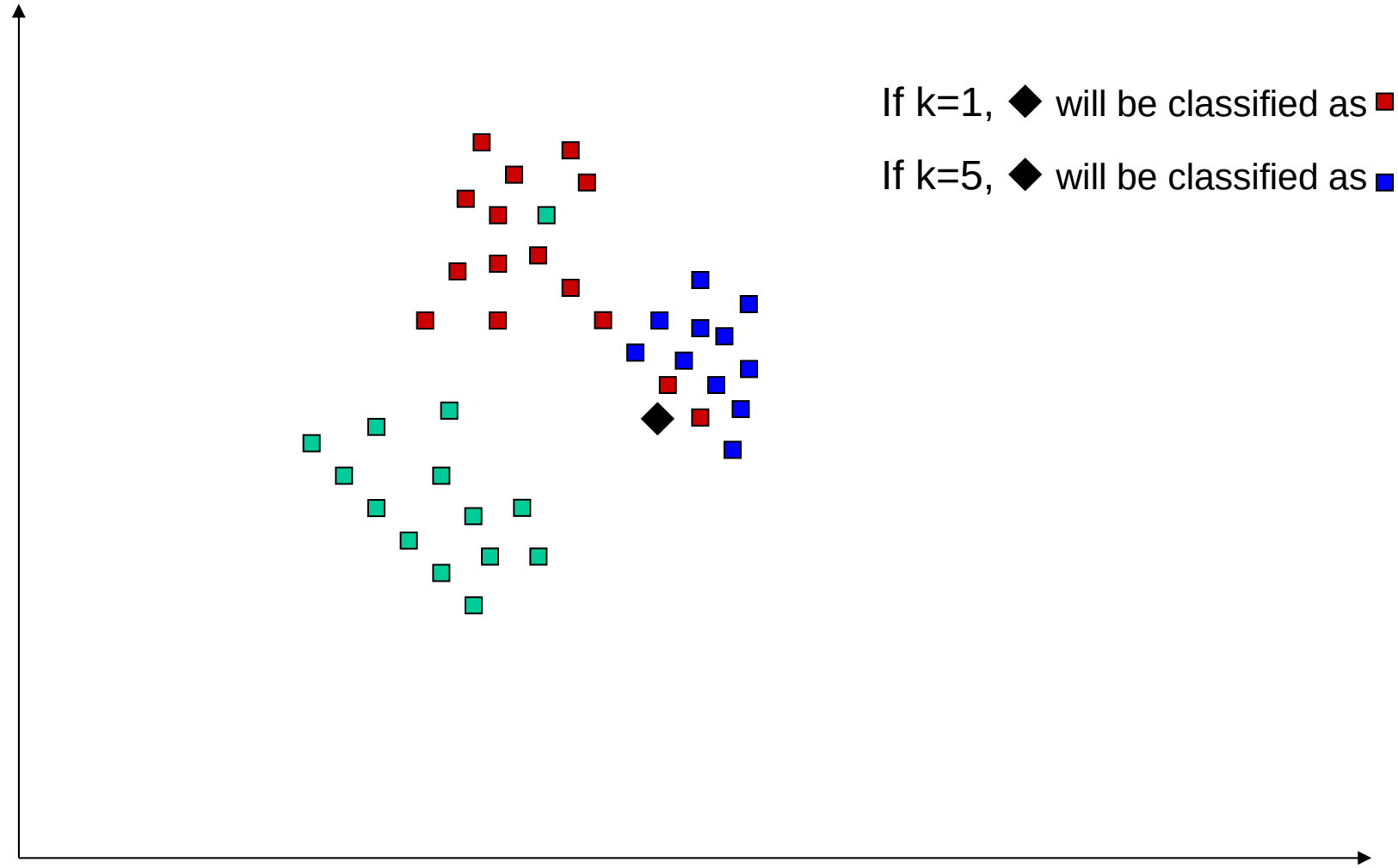
An alternative classifier - kNN

- Does not assume any particular probability density

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



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*
- *Classification performance should **always** be computed on the test data set.*
- *Find the best value of k on a validation data set.*

Estimating classification error

- A simple measure of classification accuracy can be to count the percentage of correctly classified pixels overall (averaged for all classes), or per. class. If a pixel has true class label k, it is correctly classified if ω_j=k.

- Estimate the classification error by classifying all pixels in the test set and count the percentage of wrongly classified pixels.

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 sets for training, validation and test.
- If the classifier has hyperparameters, we use a separate validation set to find the best value of them.
  More on this in a later lecture.

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

Using more than one feature

- The power of the computer lies in deciding based on more than 1 feature at a time
- A simple trick to do this is to assume that features i and j and independent, then p(i,j|c)=p(i|c)p(j|c)
- The joint decision based on D independent features is then:

$$P(\omega_j \mid x_1, x_2, \ldots x_D) = \frac{p(x_1 \mid \omega_j)\, p(x_2 \mid \omega_j) \cdots\ p(x_D \mid \omega_j) P(\omega_j)}{p(x_1, x_2, \ldots x_D)}$$

Upcoming lectures

- Multivariate Gaussian
- Classifier evaluation
- Support vector machine classification
- Feature selection/feature transforms
- Unsupervised classification

Thanks! Any more questions?