Elements of Machine Learning & Data Science

Winter semester 2025/26

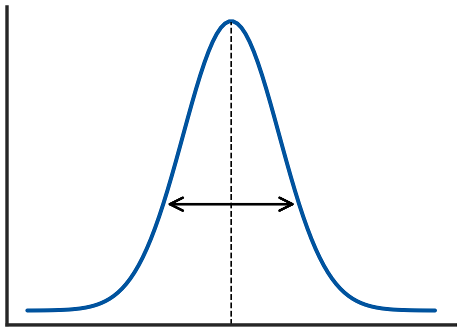
Lecture 10 – Probability Density Estimation

25.11.2025

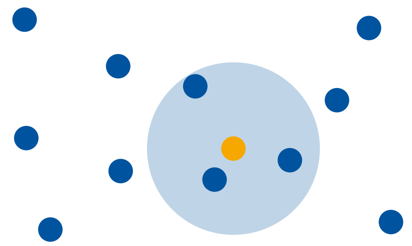
Prof. Bastian Leibe

Machine Learning Topics

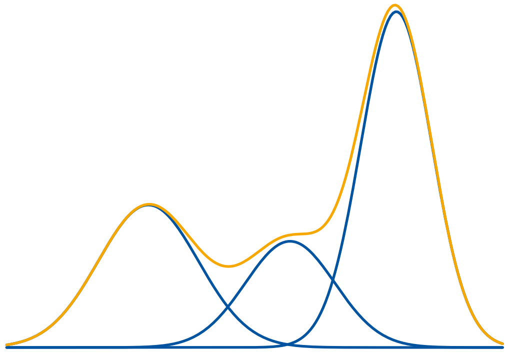
8. Introduction to ML
9. **Probability Density Estimation**
10. Linear Discriminants
11. Linear Regression
12. Logistic Regression
13. Support Vector Machines
14. Neural Network Basics



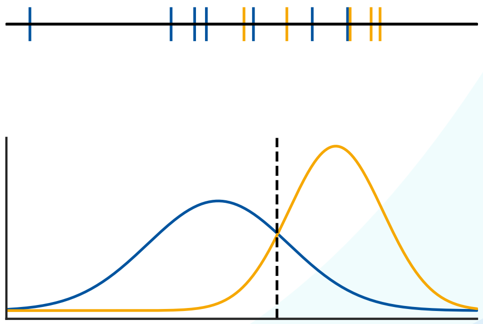
Parametric Methods & ML-Algorithm



Nonparametric Methods



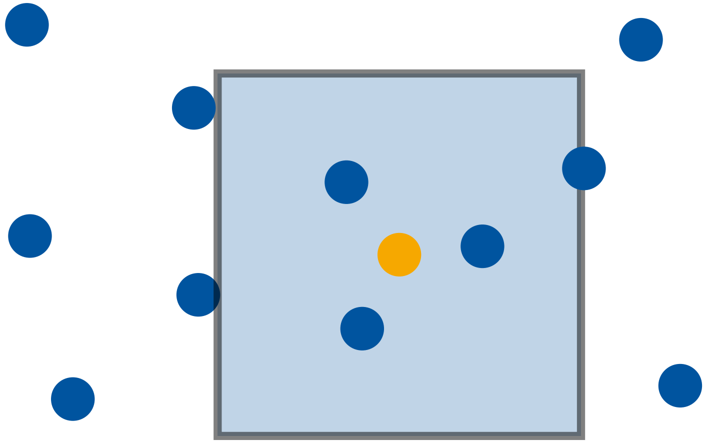
Mixtures of Gaussians & EM-Algorithm



Bayes Classifiers

Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. **Nonparametric Methods**
   a) Histograms
   b) **Kernel Methods & k-Nearest Neighbors**
4. Mixture Models
5. Bayes Classifier
6. K-NN Classifier



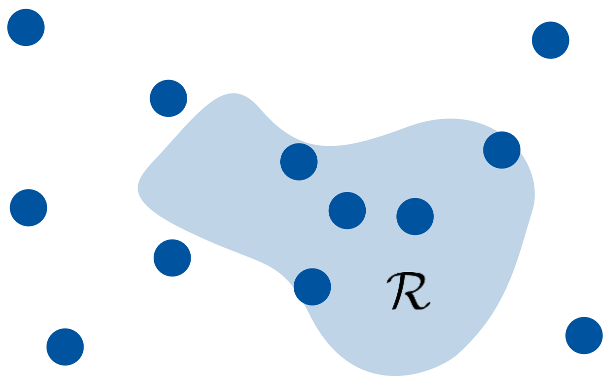
Kernel Methods and k-Nearest Neighbors

- Data point $\mathbf{x}$ comes from pdf $p(\mathbf{x})$.
  - Probability that $\mathbf{x}$ falls into small region $\mathcal{R}$:

$$P = \int_{\mathcal{R}} p(y)dy \approx p(\mathbf{x})V$$

- Estimate $p(\mathbf{x})$ from samples
  - Let K be the number of samples that fall into $\mathcal{R}$.
  - If the number of samples N is sufficiently large, we can estimate P as:

$$P = \frac{K}{N} \quad \Rightarrow \quad p(\mathbf{x}) \approx \frac{K}{NV}$$



$K = 5$
$N = 12$

For sufficiently small $\mathcal{R}$, $p(\mathbf{x})$ is roughly constant.

V : volume of $\mathcal{R}$.

$$p(\mathbf{x}) \approx \frac{K}{NV}$$

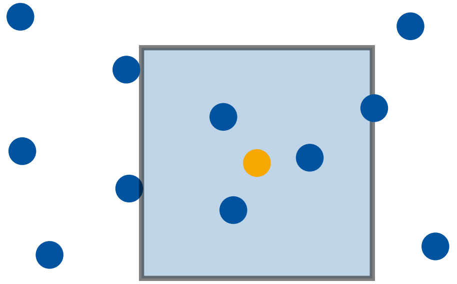
fixed V
determine K

fixed K
determine V

Kernel Methods

k-Nearest Neighbors

Example: Determine the number K of data points inside a fixed hypercube



Kernel Methods

- Hypercube of dimension D with edge length h :

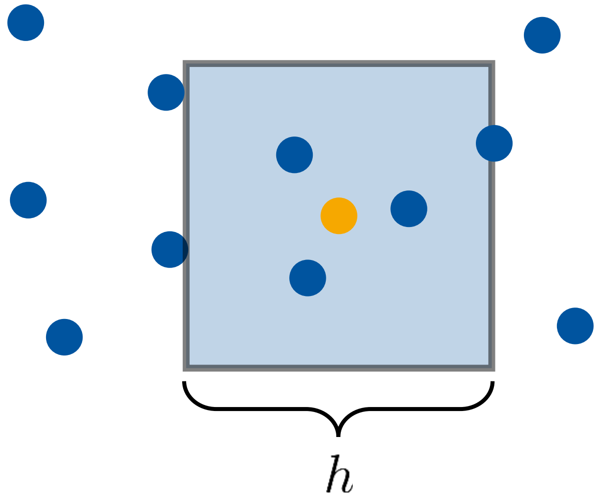
$$k(\mathbf{u}) = \begin{cases} 1, & \text{if } |u_i| \leq \frac{1}{2}h, \;\; i = 1, \ldots, D \\ 0, & \text{otherwise} \end{cases}$$

$$K = \sum_{n=1}^{N} k(\mathbf{x} - \mathbf{x}_n) \qquad V = \int k(\mathbf{u}) d\mathbf{u} = h^D$$

- Probability density estimate:

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{Nh^D} \sum_{n=1}^{N} k(\mathbf{x} - \mathbf{x}_n)$$

- This method is known as Parzen Window estimation.



- In general, we can use any kernel such that

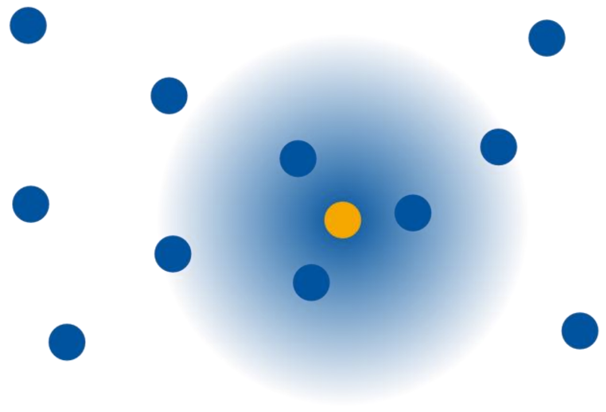
$$k(\mathbf{u}) \geq 0, \quad \int k(\mathbf{u}) d\mathbf{u} = 1$$

$$K = \sum_{n=1}^{N} k(\mathbf{x} - \mathbf{x}_n)$$

- Then, we get the probability density estimate

$$p(\mathbf{x}) \approx \frac{K}{NV} = \frac{1}{N} \sum_{n=1}^{N} k(\mathbf{x} - \mathbf{x}_n)$$

- This is known as Kernel Density Estimation.



E.g., a Gaussian kernel for smoother boundaries.

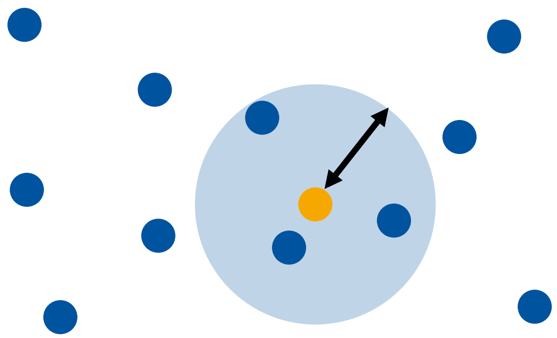
$$p(\mathbf{x}) \approx \frac{K}{NV}$$

fixed V
determine K

fixed K
determine V

Kernel Methods

k-Nearest Neighbors



Increase the volume V until the K next data points are found.

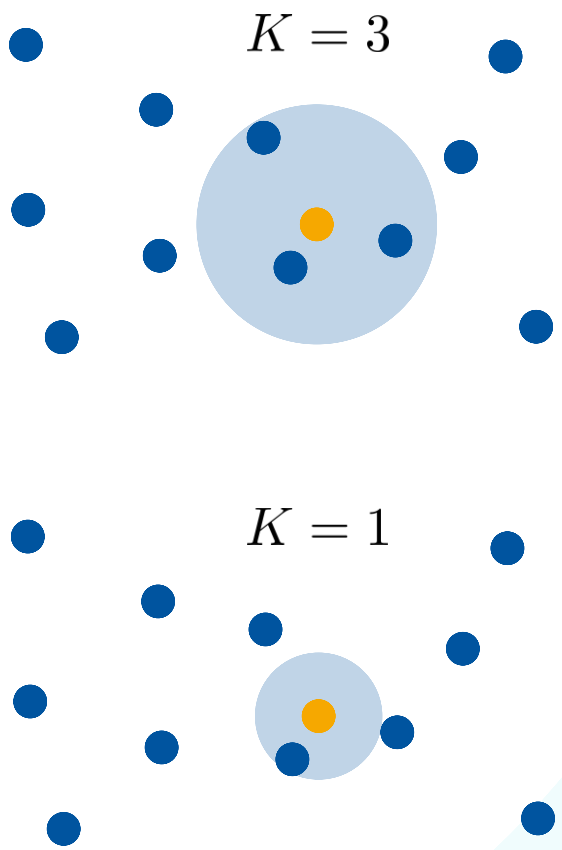
k-Nearest Neighbors

- Fix K, estimate V from the data.
- Consider a hypersphere centered on $\mathbf{x}$ and let it grow to a volume V^* that includes K of the given N data points.
- Then

$$p(\mathbf{x}) \approx \frac{K}{NV^*}$$

- Side note:
  - Strictly speaking, the model produced by k-NN is not a true density model, because the integral over all space diverges.
  - E.g. consider $K = 1$ and a sample exactly on a data point.

$$V^* = 0 \quad \Rightarrow \quad p(\mathbf{x}) \approx \frac{K}{N \cdot 0}$$



Advantages

- Very general. In the limit $(N \to \infty)$, every probability density can be represented.
- No computation during training phase
  - Just need to store training set

Limitations

- Requires storing and computing with the entire dataset.
  - Computational costs linear in the number of data points.
  - Can be improved through efficient storage structures (at the cost of some computation during training).
- Choosing the kernel size/K is a hyperparameter optimization problem.

Bias-Variance Tradeoff

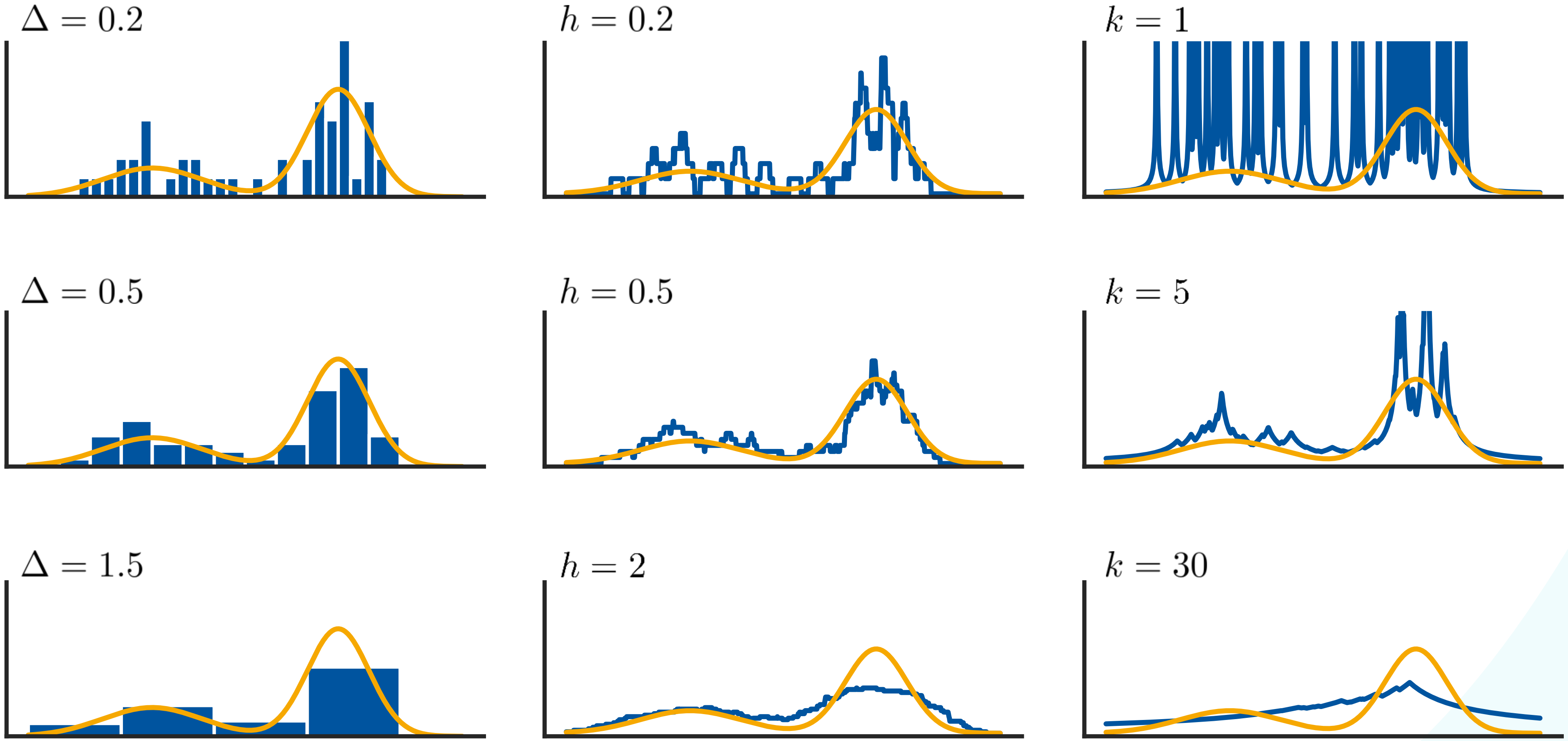
Not smooth enough

Too much variance

About ok

Too smooth

Too much bias



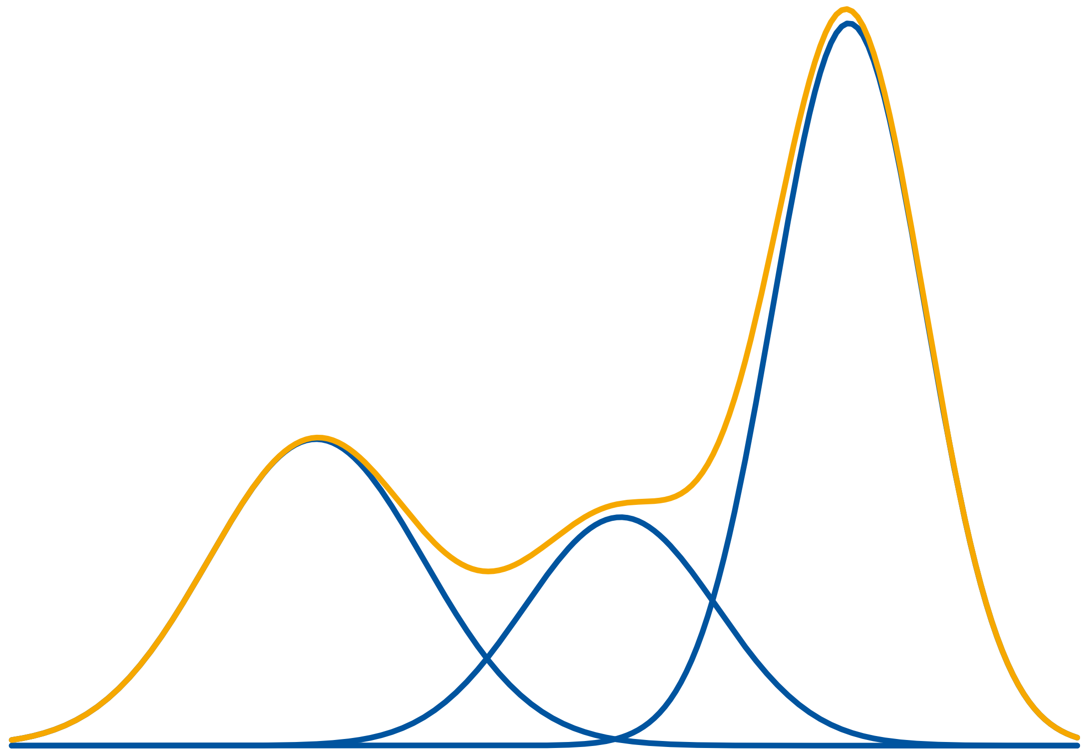
Histograms: Bin width Δ

Parzen Window: Kernel size h

k-NN: # of neighbors k

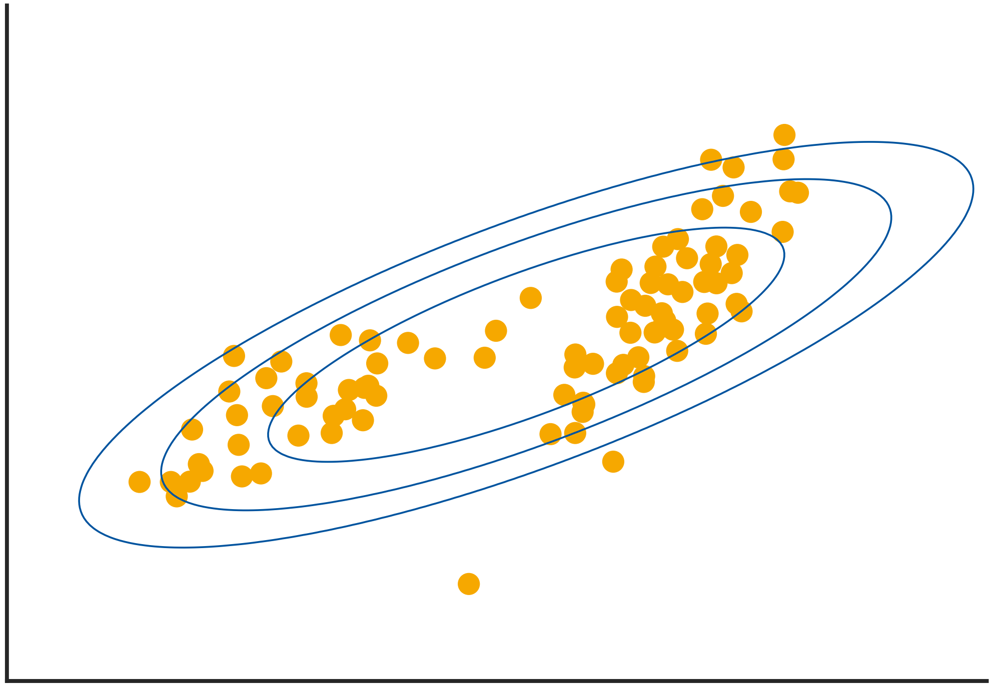
Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. **Mixture Models**
5. Bayes Classifier
6. K-NN Classifier



Mixture Models

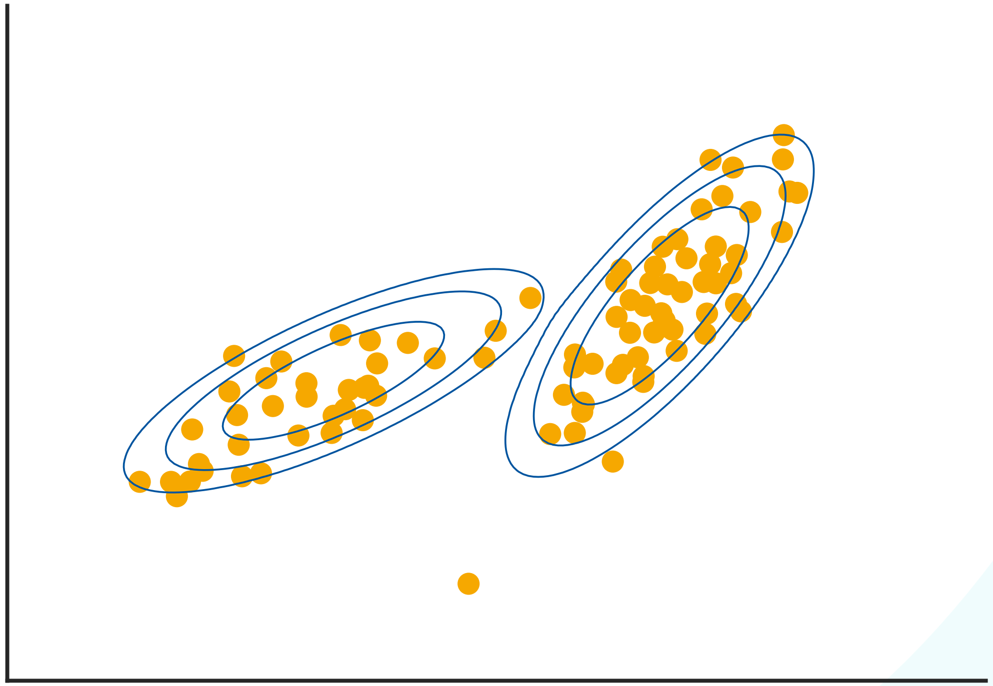
- Often, a single parametric representation is not enough.
- Struggle to fit multimodal data



Single Gaussian

Mixture Models

- Often, a single parametric representation is not enough.
- Struggle to fit multimodal data
- Mixture models combine multiple densities into a single distribution.
  - Improves modeling of multimodal data.



Mixture of two Gaussians

Mixture of Gaussians (MoG)

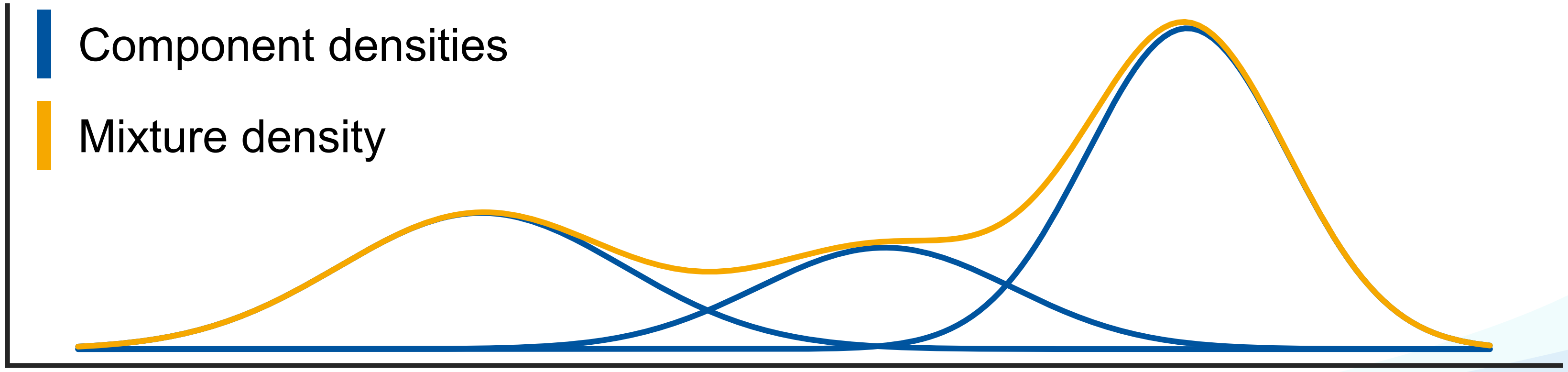
- This is the sum of M individual Normal distributions:

$$p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)$$

Likelihood of measurement x given mixture component j

Prior of component j

- In the limit, every smooth distribution can be approximated this way (if M is large enough).



- For Gaussians, the complete mixture model is given as:

$$p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)$$

$$p(x|\theta_j) = \mathcal{N}(x|\mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right)$$

$p(j) = \pi_j$ with $0 \leq \pi_j \leq 1$ and $\sum_{j=1}^{M} \pi_j = 1$

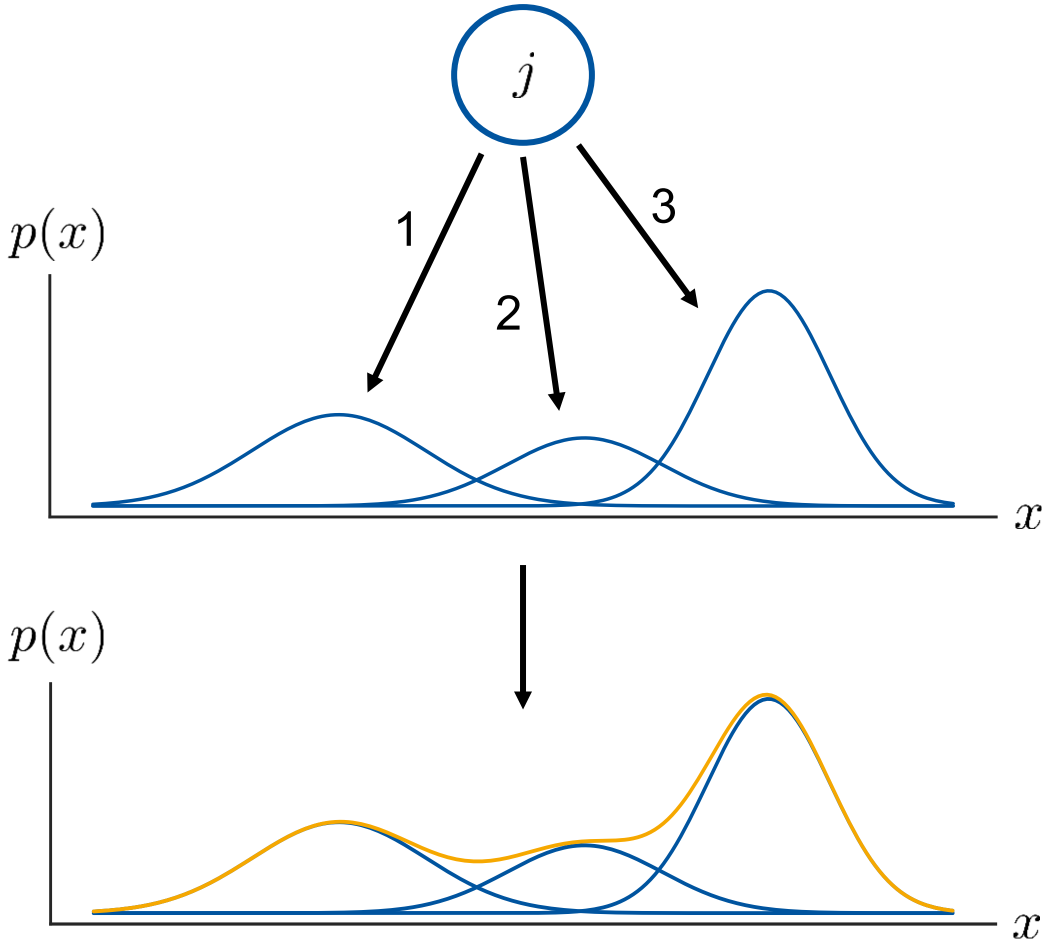
Note: this integrates to 1

$$\int p(x|\theta)dx = 1$$

Total parameters:

$$\theta = (\pi_1, \mu_1, \sigma_1, \ldots, \pi_M, \mu_M, \sigma_M)$$

- MoGs are generative models: we can easily sample from them.



$$p(j) = \pi_j$$

Sample the component using its “weight”

$$p(x|\theta_j)$$

Sample from the mixture component

$$p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)$$

The result is a sample from the mixture density

Learning a Mixture Model

- Apply Maximum Likelihood
  - Minimize $E = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(\mathbf{x}_n|\theta)$
  - Let's first look at $\boldsymbol{\mu}_j$:

$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = 0$$

  - We can already see that this will be difficult:

$$\ln p(\mathcal{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \boldsymbol{\pi}_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

Steps for Maximum Likelihood:
1. Express Likelihood $L(\theta)$
2. Apply negative logarithm to get $E(\theta)$
3. Take derivative, set to zero
4. Solve for parameters

Learning a Mixture Model

- Apply Maximum Likelihood
  - Minimize $E = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(\mathbf{x}_n|\theta)$
  - Let's first look at $\boldsymbol{\mu}_j$:

$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = 0$$

  - We can already see that this will be difficult:

$$\ln p(\mathcal{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \boldsymbol{\pi}_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

This will cause problems!

Steps for Maximum Likelihood:

1. Express Likelihood $L(\theta)$
2. Apply negative logarithm to get $E(\theta)$
3. Take derivative, set to zero
4. Solve for parameters

$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = -\sum_{n=1}^{N} \frac{\frac{\partial}{\partial \boldsymbol{\mu}_j} p(\mathbf{x}_n|\theta_j)}{\sum_{k=1}^{K} p(\mathbf{x}_n|\theta_k)}$$

$$= -\sum_{n=1}^{N} \left(\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j) \frac{p(\mathbf{x}_n|\theta_j)}{\sum_{k=1}^{K} p(\mathbf{x}_n|\theta_k)} \right)$$

$$= -\boldsymbol{\Sigma}^{-1} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu}_j) \underbrace{\frac{\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}}_{= \gamma_j(\mathbf{x}_n)} \stackrel{!}{=} 0$$

$$\boxed{\frac{\partial}{\partial \boldsymbol{\mu}_j} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j) \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

$= \gamma_j(\mathbf{x}_n)$

"responsibility" of component j for $\mathbf{x}_n$

We thus obtain $\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)} \qquad \gamma_j(\mathbf{x}_n) = \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- There is no direct analytical solution!

$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = f(\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \ldots, \pi_M, \boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M)$$

  - Complex gradient function (non-linear mutual dependencies)
  - Optimization of one Gaussian depends on all other Gaussians!
  - Standard solution: iterative optimization with EM algorithm

The EM Algorithm

- The Expectation-Maximization (EM) Algorithm alternates between two steps:
  - **E-Step**: softly assign samples to mixture components:

$$\gamma_j(\mathbf{x}_n) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \quad \forall j = 1, \ldots, K, n = 1, \ldots, N$$

  - **M-Step**: re-estimate parameters of each component based on the soft assignments:

$$\hat{N}_j \leftarrow \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n) \qquad \hat{\boldsymbol{\mu}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n$$

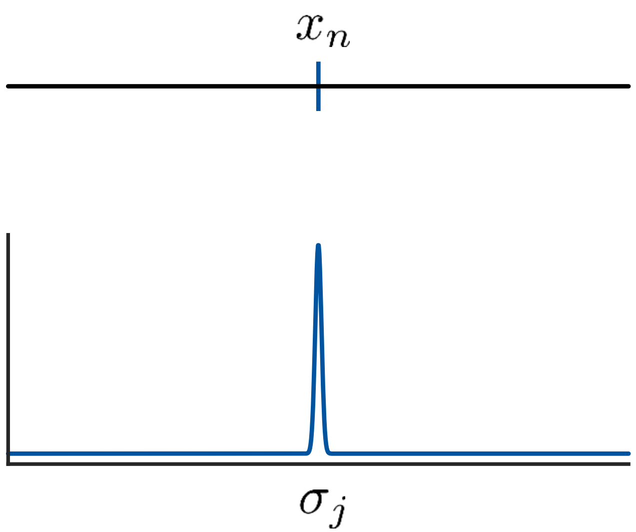
$$\hat{\pi}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N} \qquad \hat{\boldsymbol{\Sigma}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})^{\mathsf{T}}$$

Practical Advice

- When implementing EM, we need to take care to avoid singularities in the estimation!
  - Mixture components may collapse on single data points.
  - E.g. consider the case $\mathbf{\Sigma}_k = \sigma_k^2 \mathbf{I}$ (this also holds in general)
  - Assume component j is exactly centered on data point $\mathbf{x}_n$. This data point will then contribute a term in the likelihood function

$$\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n, \sigma_j^2 \mathbf{I}) = \frac{1}{\sqrt{2\pi}\sigma_j}$$

  - For $\sigma_j \to 0$, this term goes to infinity!
- We need to introduce regularization to avoid this.
  - Enforce minimum width for the Gaussians



Instead of $\mathbf{\Sigma}^{-1}$, use $(\mathbf{\Sigma} + \sigma_{\min}\mathbf{I})^{-1}$.

EM Algorithm – An Example

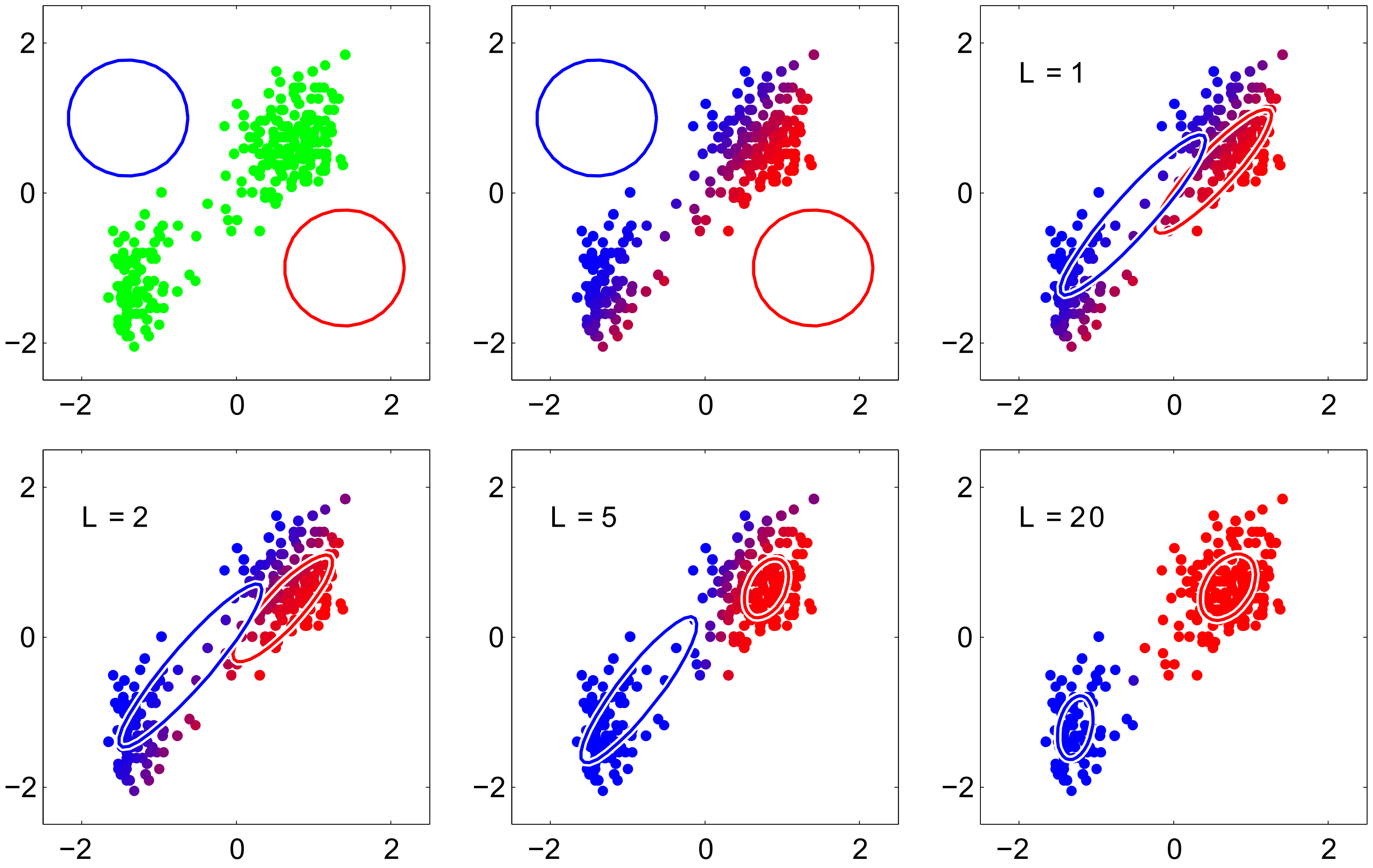


Image source: C.M. Bishop, 2006

Discussion: Mixture Models

Advantages

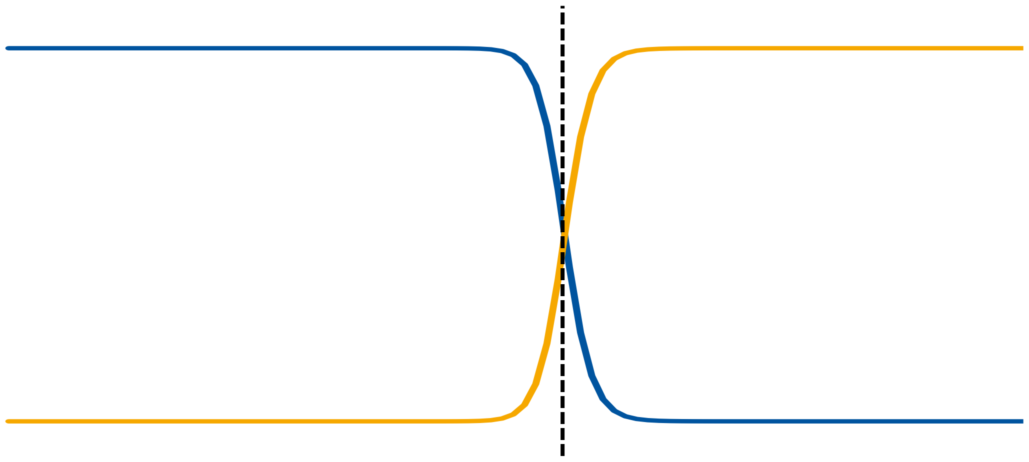
- Very general, can represent any continuous distribution.
- Once trained, is very fast to evaluate.

Limitations

- Need to apply regularization to avoid numerical instabilities.
- Choosing the right number of mixture components is hard.
- The EM algorithm is computationally expensive.
  - Especially for high-dim. problems.
  - Very sensitive to initialization.
  - *Practical Tip*: Run k-Means first and initialize clusters with k-Means result

Probability Density Estimation

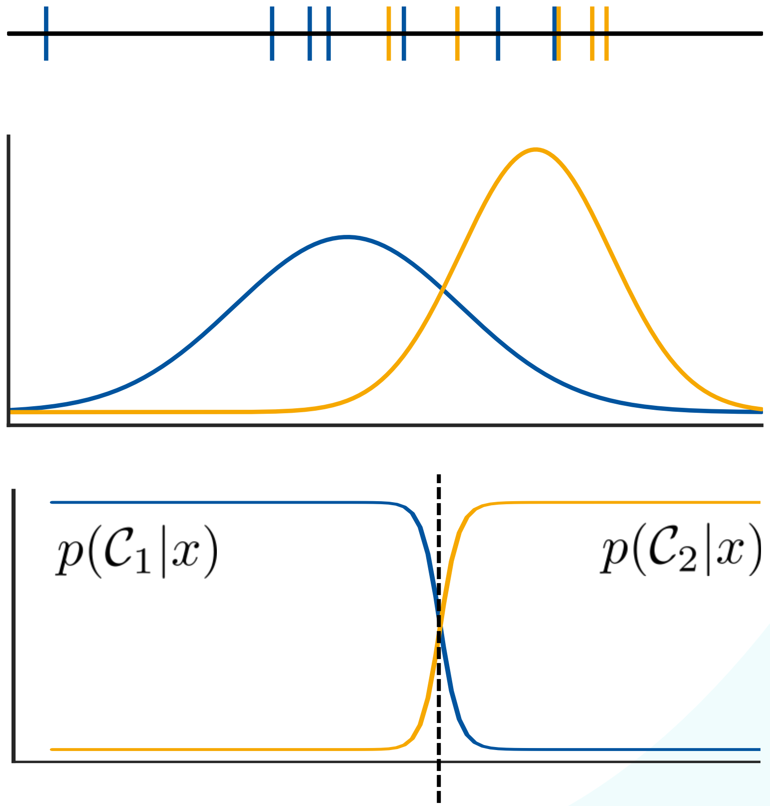
1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. Mixture Models
5. **Bayes Classifier**
6. K-NN Classifier



Bayes Classifier

- We know how to estimate probability densities from data.
- We can now use Bayes Decision Theory to build a classifier:
  - Estimate likelihoods & priors from data.
  - Calculate posterior with Bayes' Theorem.
  - Decide for class with highest posterior probability:

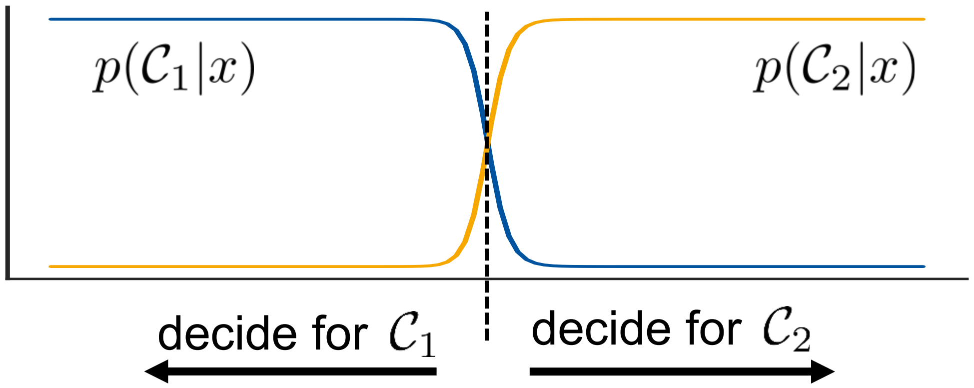
$$p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x)$$



Likelihood-Ratio Test

- Assume we want to classify an observation x into one of two classes $\mathcal{C}_1,\ \mathcal{C}_2$.
  - Decide for $\mathcal{C}_1$ if
    $$p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x)$$
  - This is equivalent to
    $$p(x|\mathcal{C}_1)p(\mathcal{C}_1) > p(x|\mathcal{C}_2)p(\mathcal{C}_2)$$
  - Which again is equivalent to
    $$\frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} > \boxed{\frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)}}$$

Decision threshold θ



$$\boxed{p(\mathcal{C}|x) = \frac{p(x|\mathcal{C})p(\mathcal{C})}{p(x)}}$$

Decision Functions

- We can find a decision function based on probability densities.
  - Determine class-conditional densities $p(x|\mathcal{C}_k)$ for each class individually.
  - Separately infer the prior class probabilities $p(\mathcal{C}_k)$.
  - Then use Bayes' theorem and/or the likelihood-ratio test.

Generative methods:

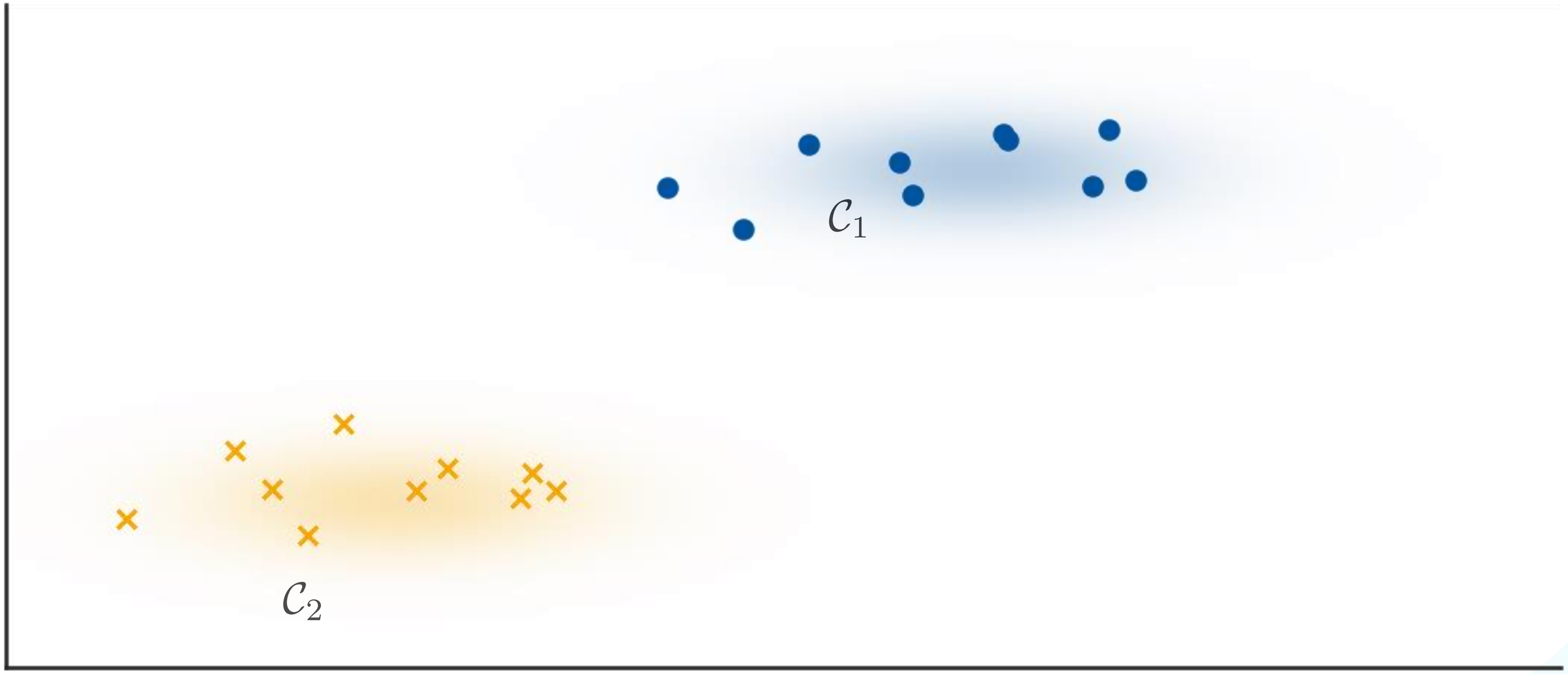
$$y_k(x) \propto p(x|\mathcal{C}_k)p(\mathcal{C}_k)$$

- Alternative: solve the inference problem of determining the posterior class probabilities directly.
  - Then use Bayes' decision theory to assign each new observation to its class.

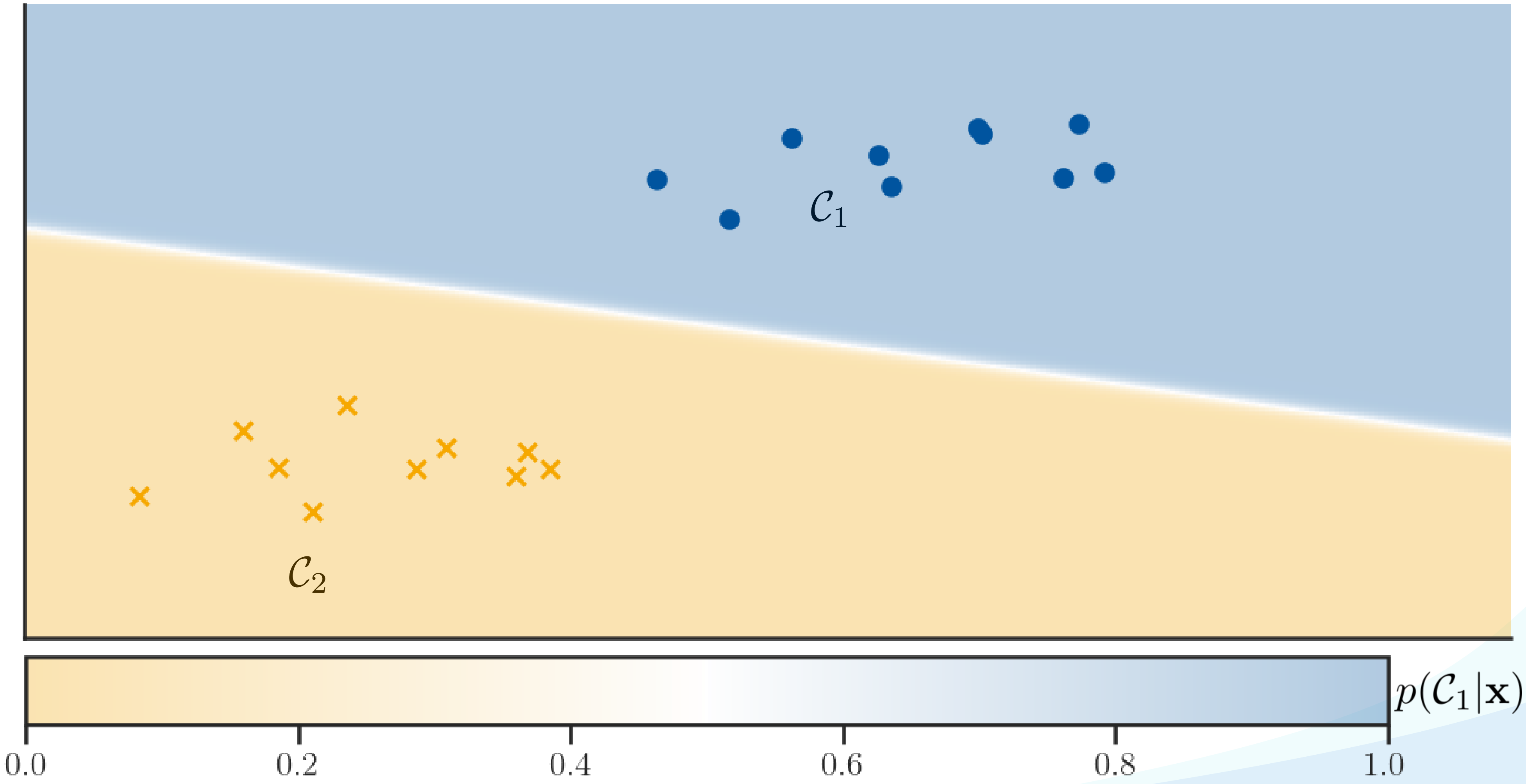
Discriminative methods:

$$y_k(x) = p(\mathcal{C}_k|x)$$

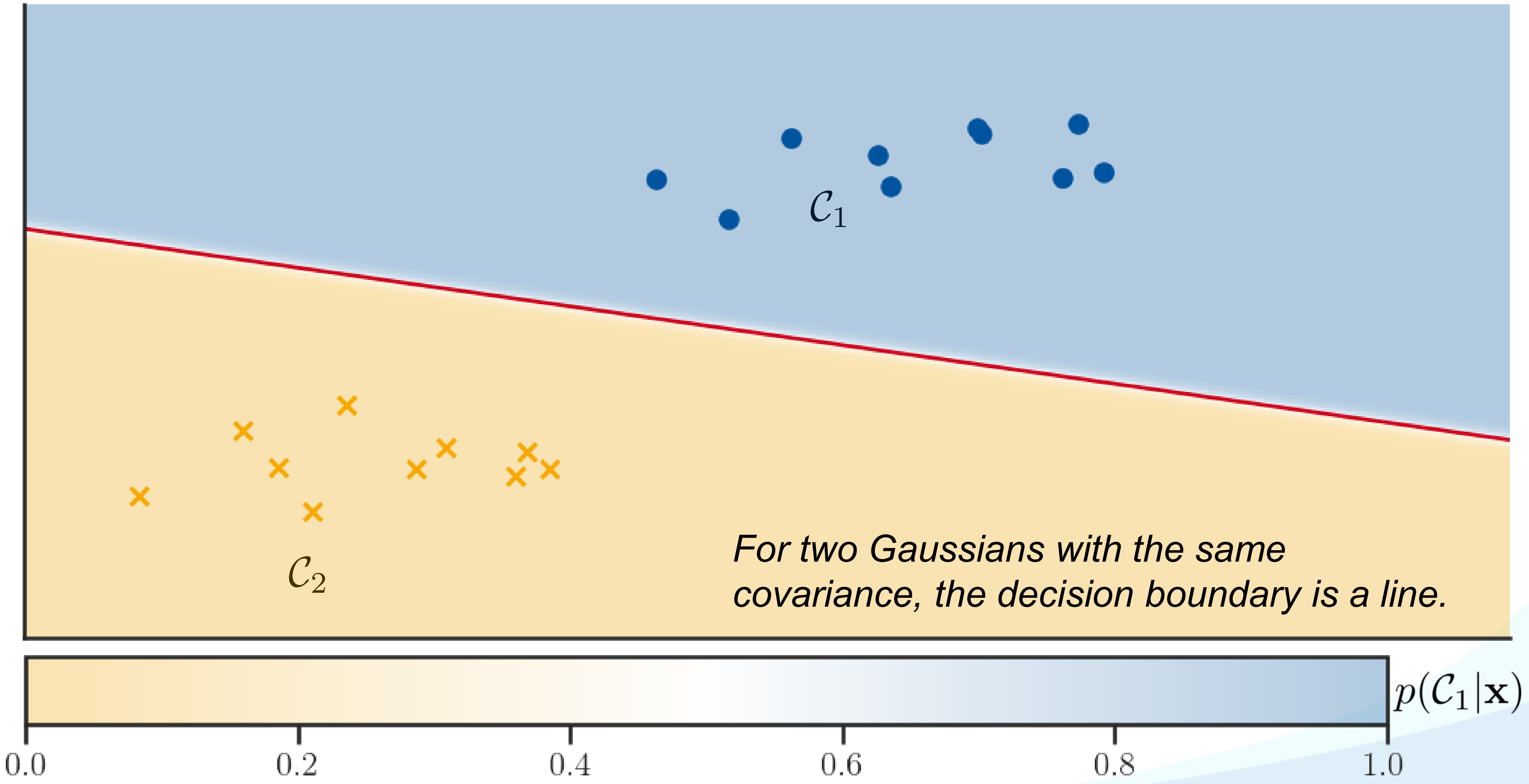
Example



Example



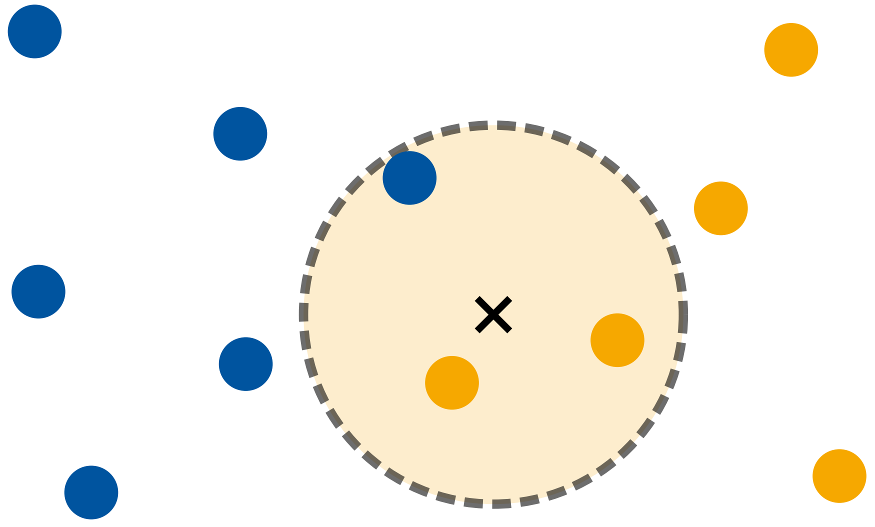
Example



$\mathcal{C}_1$
$\mathcal{C}_2$
For two Gaussians with the same covariance, the decision boundary is a line.
$p(\mathcal{C}_1|\mathbf{x})$
0.0
0.2
0.4
0.6
0.8
1.0

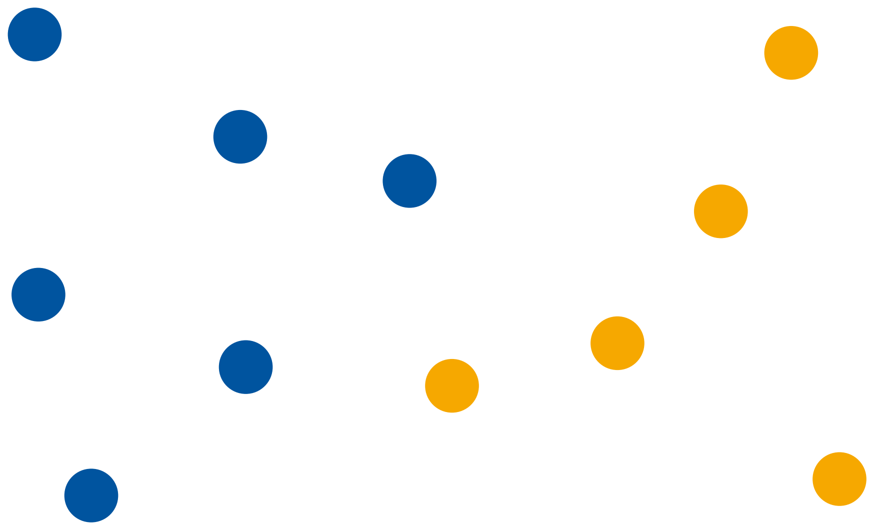
Probability Density Estimation

1. Probability Distributions
2. Parametric Methods
3. Nonparametric Methods
4. Mixture Models
5. Bayes Classifier
6. **K-NN Classifier**



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier

1. Determine the class-conditional densities

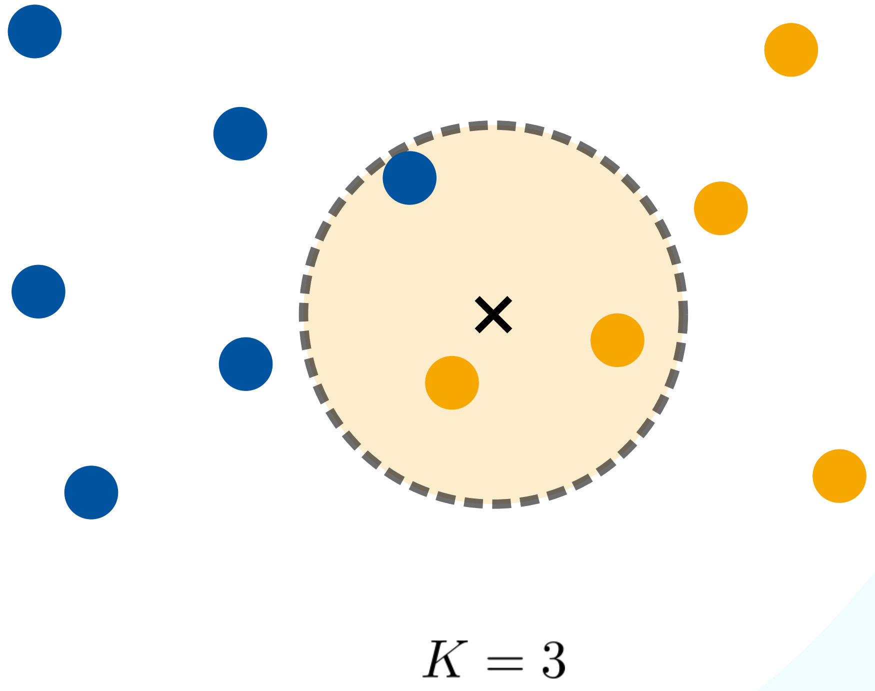
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \qquad p(\mathbf{x}) \approx \frac{K}{NV}$$

2. Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

3. Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)\frac{1}{p(\mathbf{x})}$$



K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
  1. Determine the class-conditional densities

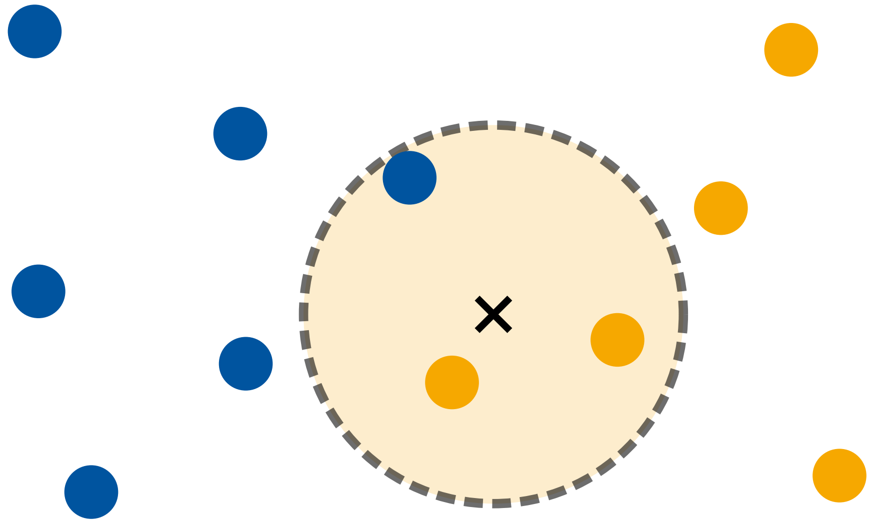
     $$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \qquad p(\mathbf{x}) \approx \frac{K}{NV}$$

  2. Determine the prior probabilities

     $$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

  3. Use Bayes' theorem to compute the posterior

     $$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} p(\mathcal{C}_j) \frac{1}{p(\mathbf{x})}$$



$K = 3$

K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
  1. Determine the class-conditional densities

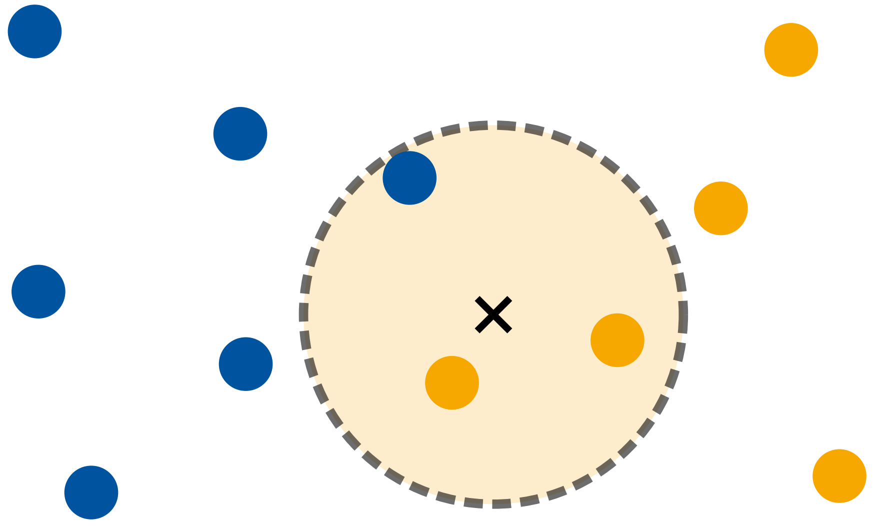
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \qquad p(\mathbf{x}) \approx \frac{K}{NV}$$

  2. Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

  3. Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} \frac{N_j}{N} \frac{1}{p(\mathbf{x})}$$



$K = 3$

K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
  1. Determine the class-conditional densities

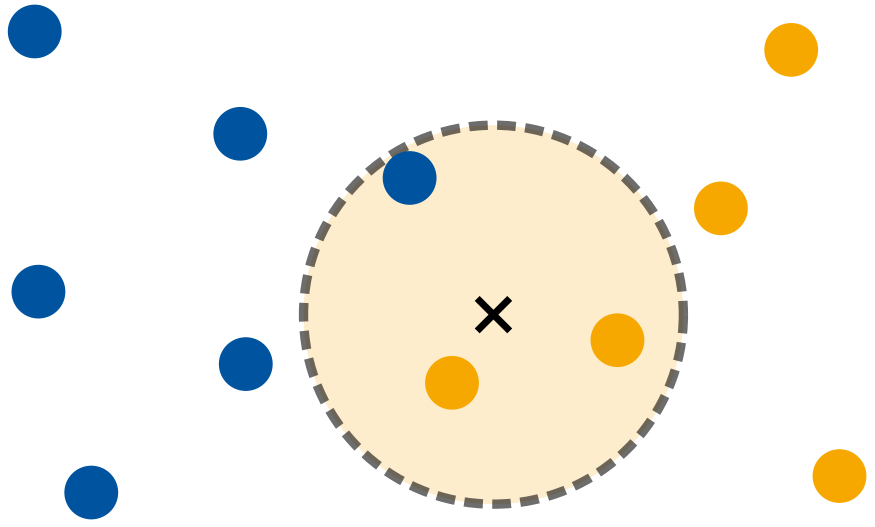
$$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \qquad p(\mathbf{x}) \approx \frac{K}{NV}$$

  2. Determine the prior probabilities

$$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

  3. Use Bayes' theorem to compute the posterior

$$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{N_j V} \frac{N_j}{N} \frac{NV}{K}$$



$K = 3$

K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
  1. Determine the class-conditional densities

     $$p(\mathbf{x}|\mathcal{C}_j) \approx \frac{K_j}{N_j V} \qquad p(\mathbf{x}) \approx \frac{K}{NV}$$

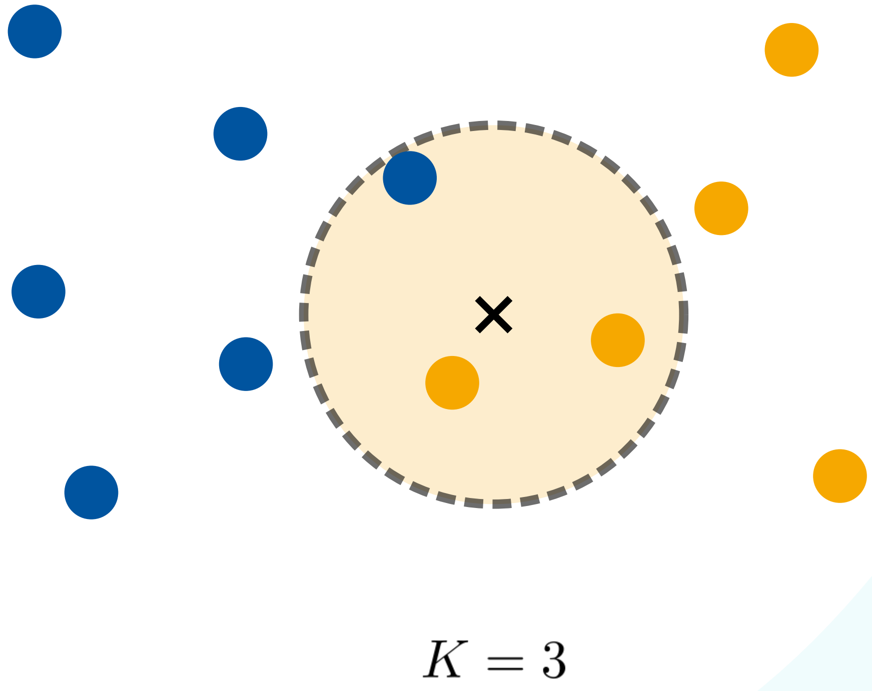
  2. Determine the prior probabilities

     $$p(\mathcal{C}_j) \approx \frac{N_j}{N}$$

  3. Use Bayes' theorem to compute the posterior

     $$p(\mathcal{C}_j|\mathbf{x}) \approx \frac{K_j}{\cancel{N_j}\cancel{V}} \frac{\cancel{N_j}}{\cancel{N}} \frac{\cancel{N}\cancel{V}}{K} = \frac{K_j}{K}$$

$\Rightarrow$ *Decide for the majority class among the neighbors.*

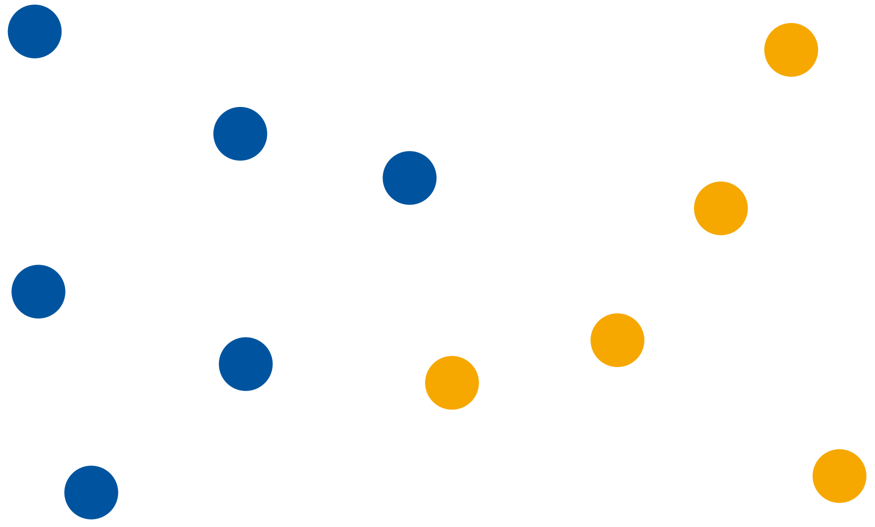


K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
- Algorithm

Given a new sample $\mathbf{x}$:

1. Find the K training samples with the smallest distance to $\mathbf{x}$.
2. Assign the majority label of those samples to $\mathbf{x}$.

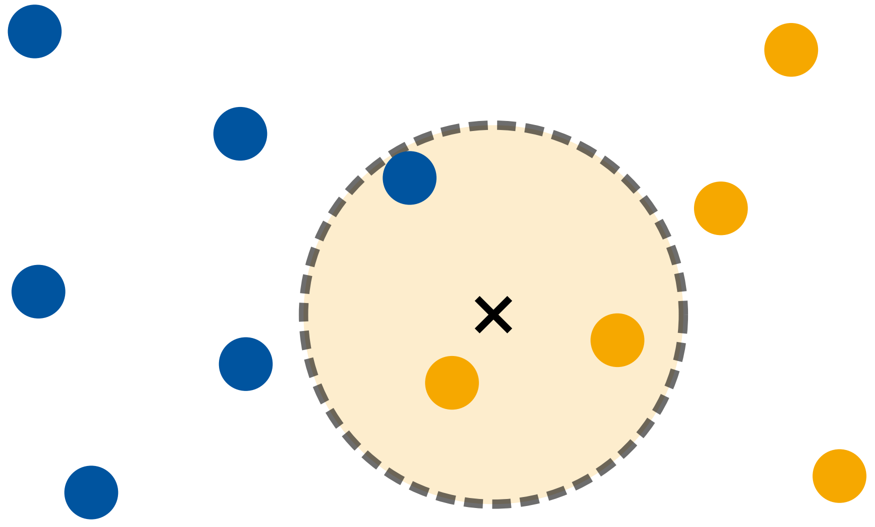


K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
- Algorithm

Given a new sample $\mathbf{x}$:

1. Find the K training samples with the smallest distance to $\mathbf{x}$.
2. Assign the majority label of those samples to $\mathbf{x}$.



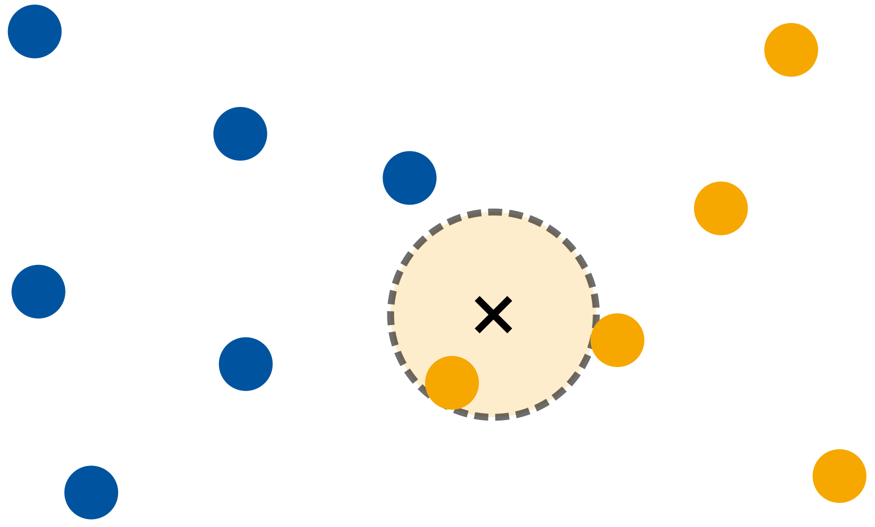
$K = 3$

K-NN Classifier

- Combine K-NN density estimation with Bayes Decision Theory: K-NN Classifier
- Algorithm

Given a new sample $\mathbf{x}$:

1. Find the K training samples with the smallest distance to $\mathbf{x}$.
2. Assign the majority label of those samples to $\mathbf{x}$.

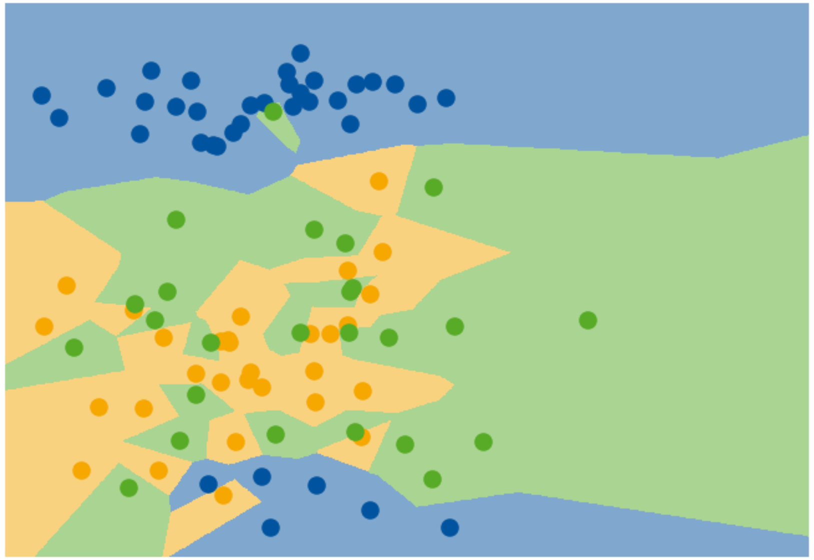


$K = 1$

- Special case: 1-NN Classifier.

Theoretical guarantee: Never worse than 2x the error of the optimal classifier!

Example



$K = 1$

$K = 3$

$K = 15$

Discussion: K-NN Classifier

Advantages

- Very simple, Bayes-optimal classifier.
- Needs no training.
- Can always be used as first estimate when working with a new dataset.
- Theoretical guarantees
  - Never worse than 2x optimal error

Limitations

- Requires storing the complete training set.
- Finding the k-nearest neighbors can become very expensive in high-dimensional spaces
- Theoretical optimality bound is often too loose to be of practical value.

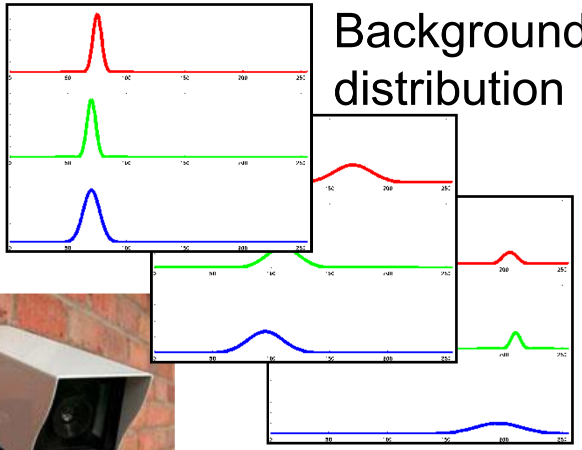
Application Example: Background Models for Tracking

- **Example: Object tracking in static surveillance cameras**
  - Want to know if anybody enters a forbidden area
  - Challenge: many possible moving objects

- Idea: Train background color model for each pixel
  - Initialize with an empty scene.
  - Learn “common“ appearance variation for each background pixel, e.g., by fitting a Gaussian distribution to the observed noise over several frames.

  - Evaluate the likelihood of observed pixel colors under this model.

  $\Rightarrow$ *Anything that cannot be explained by the background model is labeled as foreground (=object).*



Background color distribution per pixel

Application Example: Background Models for Tracking

- Problem: Outdoor scenes
  - Dynamic areas
  - Waving trees, rippling water, …
  - $\Rightarrow$ *More flexible representation needed here!*

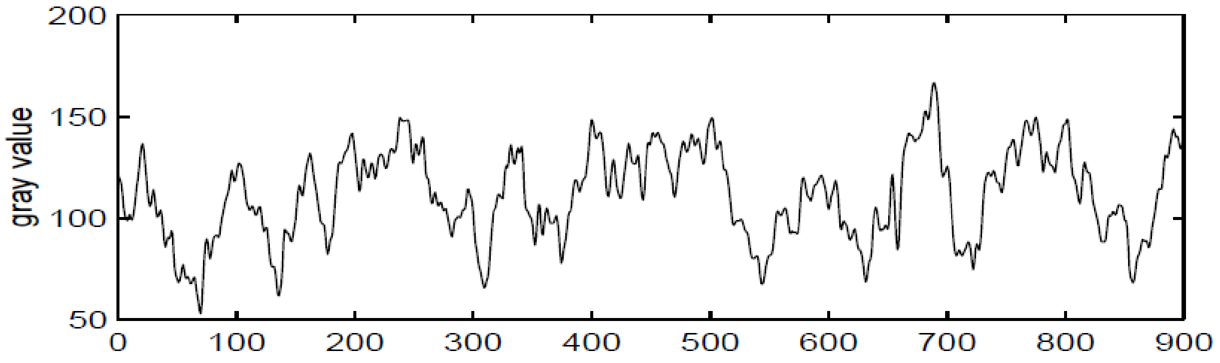


Image & Video source: A. Elgammal

- Idea:
  - Use Kernel Density Estimation using the observed pixel values over a temporal window to model the "background" distribution for each pixel.
  - Again, evaluate the likelihood of the observed pixel color under this background model to detect "foreground" objects.

A. Elgammal, D. Harwood, L.S. Davis, Non-parametric Model for Background Subtraction, ECCV 2000.

Application Example: Background Models for Tracking

- Results
  - Very robust foreground object detection in dynamic scenes

  - Automatic adaptation to varying weather conditions through temporal window

Video source: A. Elgammal

A. Elgammal, D. Harwood, L.S. Davis, Non-parametric Model for Background Subtraction, ECCV 2000.

Application Example: Image Segmentation

(a) input image (b) user input (c) inferred segmentation

- **Example: User assisted image segmentation**
  - User marks two regions for foreground and background.
  - Learn a MoG model for the color values in each region.
  - Use those models to classify all other pixels with a Bayes classifier (likelihood ratio test)

  $\Rightarrow$ Simple, but effective segmentation procedure

References and Further Reading

- More information about EM and MoG estimation is available in Chapter 2.3.9 and the entire Chapter 9 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006