Exercise 1 due Monday next week
- Bayes decision theory
- Maximum Likelihood
- Kernel density estimation / k-NN
⇒ Submit your results to Georgios until evening of 25.04.

Exercise modalities
- You can work in teams of up to 3 people.
- If you work in a team
  - Turn in a single solution
  - But put all names on it

\[ L(\mu) = p(X|\mu) = \prod_{n=1}^{N} p(x_n|\mu) \]
\[ E(\theta) = -\ln L(\theta) = - \sum_{n=1}^{N} \ln p(x_n|\theta) \]
\[ \frac{\partial}{\partial \theta} E(\theta) = - \sum_{n=1}^{N} \frac{p(x_n|\theta)}{p(x_n|\theta)} \frac{\partial p(x_n|\theta)}{\partial \theta} = 0 \]

Recap: Gaussian (or Normal) Distribution
- One-dimensional case
  - Mean \( \mu \)
  - Variance \( \sigma^2 \)

\[ \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\} \]

- Multi-dimensional case
  - Mean \( \mu \)
  - Covariance \( \Sigma \)

\[ \mathcal{N}(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} \exp \left\{ -\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right\} \]
Recap: Bayesian Learning Approach

- **Discussion**
  - Likelihood of the parametric form $\theta$ given the data set $X$.
  - Estimate for $x$ based on parametric form $\theta$.
  - Prior for the parameters $\theta$.
  - The more uncertain we are about $\theta$, the more we average over all possible parameter values.

$$p(x|X) = \frac{\int p(x|\theta) L(\theta) p(\theta) d\theta}{\int L(\theta) p(\theta) d\theta}$$

Normalization: integrate over all possible values of $\theta$

Topics of This Lecture

- **Probability Density Estimation**
  - General concepts
  - Gaussian distribution
- **Parametric Methods**
  - Maximum Likelihood approach
  - Bayesian vs. Frequentist views on probability
  - Bayesian Learning
- **Non-Parametric Methods**
  - Histograms
  - Kernel density estimation
  - $k$-Nearest Neighbors
  - $k$-NN for Classification
  - Bias-Variance tradeoff

Non-Parametric Methods

- **Non-parametric representations**
  - Often the functional form of the distribution is unknown

- **Estimate probability density from data**
  - Histograms
  - Kernel density estimation (Parzen window / Gaussian kernels)
  - $k$-Nearest-Neighbor

Histograms

- **Basic idea:**
  - Partition the data space into distinct bins with widths $\Delta$, and count the number of observations, $n_i$, in each bin.

  $$\hat{p}_i = \frac{n_i}{N \Delta}$$

  - Often, the same width is used for all bins, $\Delta = \Delta_i$.
  - This can be done, in principle, for any dimensionality $D$...

  ![Histogram example image](image-source: C.M. Bishop, 2006)

Summary: Histograms

- **Properties**
  - Very general. In the limit ($N \to \infty$), every probability density can be represented.
  - No need to store the data points once histogram is computed.
  - Rather brute-force

- **Problems**
  - High-dimensional feature spaces
    - $D$-dimensional space with $M^D$ bins/dimension will require $M^D$ bins!
    - Requires an exponentially growing number of data points
    - "Curse of dimensionality"
  - Discontinuities at bin edges
  - Bin size?
    - Too large: too much smoothing
    - Too small: too much noise

![Histogram example image](image-source: C.M. Bishop, 2006)
Statistically Better-Founded Approach

- Data point $x$ comes from pdf $p(x)$
  - Probability that $x$ falls into small region $R$
    \[ P = \int_R p(y)dy \]
- If $R$ is sufficiently small, $p(x)$ is roughly constant
  - Let $V$ be the volume of $R$
    \[ P = \int_R p(y)dy \approx p(x)V \]
- If the number $N$ of samples is sufficiently large, we can estimate $P$ as
  \[ P = \frac{K}{N} \Rightarrow p(x) \approx \frac{K}{NV} \]

Kernel Methods

- Parzen Window
  - Hypercube of dimension $D$ with edge length $h$
    \[ k(u) = \begin{cases} 
    1, & \text{if } u_i \cdot \frac{1}{2} \leq i \leq \frac{1}{2} \text{ for all } i = 1, \ldots, D \\
    0, & \text{else} 
    \end{cases} \]
  - "Kernel function"
    \[ K = \sum_{n=1}^{N} k(\frac{x-x_n}{h}) \quad V = \int k(u)du = h^d \]
  - Probability density estimate:
    \[ p(x) \approx \frac{K}{NV} = \frac{1}{Nh^D} \sum_{n=1}^{N} k(\frac{x-x_n}{h}) \]

Kernel Methods: Gaussian Kernel

- Gaussian kernel
  - Kernel function
    \[ k(u) = \frac{1}{(2\pi h^2)^{D/2}} \exp \left\{- \frac{||u||^2}{2h^2} \right\} \]
    \[ K = \sum_{n=1}^{N} k(x-x_n) \quad V = \int k(u)du = 1 \]
  - Probability density estimate
    \[ p(x) \approx \frac{K}{NV} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{D/2}} \exp \left\{- \frac{||x-x_n||^2}{2h^2} \right\} \]

Gauss Kernel: Examples

- not smooth enough
- about OK
- too smooth

Image source: C.M. Bishop, 2006
Kernel Methods

- In general
  - Any kernel such that
    \[ k(u) \geq 0, \quad \int k(u) \, du = 1 \]
  can be used. Then
    \[ K = \sum_{n=1}^{N} k(x - x_n) \]
  - And we get the probability density estimate
    \[ p(x) \approx \frac{K}{NV} = \frac{1}{NV} \sum_{n=1}^{N} k(x - x_n) \]

Statistically Better-Founded Approach

\[ p(x) \approx \frac{K}{NV} \]

- K-Nearest Neighbor
  - Increase the volume \( V \) until the \( K \) next data points are found.

K-Nearest Neighbor

- Nearest-Neighbor density estimation
  - Fix \( K \), estimate \( V \) from the data.
  - Consider a hypersphere centred on \( x \) and let it grow to a volume \( V \) that includes \( K \) of the given \( N \) data points.
  - Then
    \[ p(x) \approx \frac{K}{NV} \]
  - 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 \( x = x_j \).

k-Nearest Neighbor: Examples

- Not smooth enough
- About OK
- Too smooth

K acts as a smoother.

Summary: Kernel and k-NN Density Estimation

- Properties
  - Very general. In the limit (\( N \to \infty \)), every probability density can be represented.
  - No computation involved in the training phase
    \( \Rightarrow \) Simply storage of the training set
- Problems
  - Requires storing and computing with the entire dataset.
  - Computational cost linear in the number of data points.
  - This can be improved, at the expense of some computation during training, by constructing efficient tree-based search structures.
  - Kernel size \( K \) in K-NN?
    - Too large: too much smoothing
    - Too small: too much noise

K-Nearest Neighbor Classification

- Bayesian Classification
  \[ p(C_j|x) = \frac{p(x|C_j)p(C_j)}{p(x)} \]
- Here we have
  \[ p(x) \approx \frac{K}{NV} \]
  \[ p(x|C_j) \approx \frac{K_j}{N_jV} \]
  \[ p(C_j) \approx \frac{N_j}{N} \]

k-Nearest Neighbor classification
### Discussion

- The methods discussed so far are all simple and easy to apply. They are used in many practical applications.
- However...
  - Histograms scale poorly with increasing dimensionality.  
    - Only suitable for relatively low-dimensional data.
  - Both $k$-NN and kernel density estimation require the entire data set to be stored.  
    - Too expensive if the data set is large.
  - Simple parametric models are very restricted in what forms of distributions they can represent.  
    - Only suitable if the data has the same general form.
- We need density models that are efficient and flexible!  
  - Next topic...

### Topics of This Lecture

- **Mixture distributions**
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt
- **K-Means Clustering**
  - Algorithm
  - Applications
- **EM Algorithm**
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice
- **Applications**

### Mixture Distributions

- A single parametric distribution is often not sufficient  
  - E.g. for multimodal data

---

**K-Nearest Neighbors for Classification**

- Results on an example data set
- $K$ acts as a smoothing parameter.
- Theoretical guarantee
  - For $N \to \infty$, the error rate of the 1-NN classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

**Bias-Variance Tradeoff**

- Probability density estimation
  - Histograms: bin size?  
    - $\Delta$ too large: too smooth
    - $\Delta$ too small: not smooth enough
  - Kernel methods: kernel size?  
    - $h$ too large: too smooth
    - $h$ too small: not smooth enough
  - K-Nearest Neighbor: $K$?  
    - $K$ too large: too smooth
    - $K$ too small: not smooth enough
- This is a general problem of many probability density estimation methods
  - Including parametric methods and mixture models

**Topics of This Lecture**

- Mixture distributions
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt
- K-Means Clustering
  - Algorithm
  - Applications
- EM Algorithm
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice
- Applications

**K-Nearest Neighbors for Classification**

- $K$-Nearest Neighbors for Classification
  - Results on an example data set
  - $K$ acts as a smoothing parameter.
  - Theoretical guarantee
    - For $N \to \infty$, the error rate of the 1-NN classifier is never more than twice the optimal error (obtained from the true conditional class distributions).
Mixture of Gaussians (MoG)

- Sum of $M$ individual Normal distributions

\[ f(x) = \sum_{j=1}^{M} p(x|\theta_j)p(j) \]

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

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

Notes:
- The mixture density integrates to 1: $\int p(x)dx = 1$
- The mixture parameters are
  \[ \theta = (\pi_1, \mu_1, \sigma_1^2, \ldots, \pi_M, \mu_M, \sigma_M) \]
- Likelihood of measurement $x$ given mixture component $j$
  \[ p(x|\theta_j) = N(x|\mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi\sigma_j}} \exp \left\{ -\frac{(x-\mu_j)^2}{2\sigma_j^2} \right\} \]
  \[ p(j) = \pi_j \text{ with } 0 \cdot \pi_j \cdot 1 \text{ and } \sum_{j=1}^{M} \pi_j = 1. \]

Mixture of Multivariate Gaussians

- Multivariate Gaussians

\[ p(x|\theta_j) = \frac{1}{(2\pi)^{D/2} |\Sigma_j|^{1/2}} \exp \left\{-\frac{1}{2} (x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j) \right\} \]

- Mixture weights / mixture coefficients:

\[ p(j) = \pi_j \text{ with } 0 \cdot \pi_j \cdot 1 \text{ and } \sum_{j=1}^{M} \pi_j = 1 \]

- Parameters:

\[ \theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_M, \mu_M, \Sigma_M) \]
I.e. there is no direct analytical solution!

But...

I.e. there is no direct analytical solution!

Other strategy:

Assuming we knew the values of the hidden variable...

Assuming we knew the mixture components...

Bayes decision rule: Decide $j = 1$ if

$$p(j = 1|x_n) > p(j = 2|x_n)$$
Mixture of Gaussians - Other Strategy

- Chicken and egg problem - what comes first?

\[ f(x) \]

We don’t know any of those!

- In order to break the loop, we need an estimate for \( j \).
  - E.g. by clustering...

Topics of This Lecture

- Mixture distributions
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt
- K-Means Clustering
  - Algorithm
  - Applications
- EM Algorithm
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice
- Applications

K-Means Clustering

- Iterative procedure
  1. Initialization: pick \( K \) arbitrary centroids (cluster means)
  2. Assign each sample to the closest centroid.
  3. Adjust the centroids to be the means of the samples assigned to them.
  4. Go to step 2 (until no change)
- Algorithm is guaranteed to converge after finite #iterations.
  - Local optimum
  - Final result depends on initialization.

K-Means Clustering

\[ J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \]

where

\[ r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j ||x_n - \mu_j||^2 \\ 0 & \text{otherwise}. \end{cases} \]

In practice, this procedure usually converges quickly to a local optimum.
Example Application: Image Compression

Take each pixel as one data point.

Set the pixel color to the cluster mean.

K-Means Clustering

Summary K-Means

- **Pros**
  - Simple, fast to compute
  - Converges to local minimum of within-cluster squared error

- **Problem cases**
  - Setting k?
  - Sensitive to initial centers
  - Sensitive to outliers
  - Detects spherical clusters only

- **Extensions**
  - Speed-ups possible through efficient search structures
  - General distance measures: k-medoids

Credit Assignment Problem

“Credit Assignment Problem”

- If we are just given \( x \), we don’t know which mixture component this example came from:

  \[
  p(x|\theta) = \sum_{j=1}^{2} \pi_j p(x|\theta_j)
  \]

- We can however evaluate the posterior probability that an observed \( x \) was generated from the first mixture component:

  \[
  p(j = 1|x, \theta) = \frac{p(j = 1, x|\theta)}{p(x|\theta)}
  \]

  \[
  p(j = 1, x|\theta) = p(x|j = 1, \theta) p(j = 1) = p(x|\theta_1) p(j = 1)
  \]

  \[
  p(j = 1|x, \theta) = \frac{p(x|\theta_1) p(j = 1)}{\sum_{j=1}^{2} p(x|\theta_j) p(j)}
  \]

Topics of This Lecture

- Mixture distributions
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt

- K-Means Clustering
  - Algorithm
  - Applications

- EM Algorithm
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice

- Applications

Mixture Density Estimation Example

- Example

  Assume we want to estimate a 2-component MoG model

  \[
  p(x|\theta) = \sum_{j=1}^{2} \pi_j p(x|\theta_j)
  = \pi_1 p(x|\mu_1, \Sigma_1) + \pi_2 p(x|\mu_2, \Sigma_2)
  \]

  If each sample in the training set were labeled \( \pi \in \{1, 2\} \) according to which mixture component (1 or 2) had generated it, then the estimation would be easy.

  Labeled examples = no credit assignment problem.
Mixture Density Estimation Example

- When examples are labeled, we can estimate the Gaussians independently.
  - Using Maximum Likelihood estimation for single Gaussians.

- Notation
  - Let \( l \) be the label for sample \( x_i \).
  - Let \( N \) be the number of samples.
  - Let \( N_j \) be the number of samples labeled \( j \).
  - Then for each \( j \in [1,2] \) we set

\[
\tilde{\mu}_j = \frac{1}{N_j} \sum_{l_i = j} x_i,
\]

\[
\Sigma_j = \frac{1}{N_j} \sum_{l_i = j} (x_i - \tilde{\mu}_j)(x_i - \tilde{\mu}_j)^T.
\]

\[\text{(This also holds in general)}\]

EM Algorithm

- Expectation-Maximization (EM) Algorithm
  - **E-Step**: softly assign samples to mixture components
    \[
    \gamma_j(x_n) = \frac{\pi_j \mathcal{N}(x_n; \tilde{\mu}_j, \Sigma_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(x_n; \tilde{\mu}_k, \Sigma_k)} \quad \forall j = 1, \ldots, K, \quad n = 1, \ldots, N
    \]
  - **M-Step**: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[
    \tilde{N}_j = \sum_{i=1}^N \gamma_j(x_i) = \text{soft number of samples labeled } j
    \]
    \[
    \tilde{N}_j \tilde{\mu}_j^{\text{new}} = \frac{1}{N_j} \sum_{l_i = j} \gamma_j(x_i) x_i
    \]
    \[
    \tilde{N}_j \tilde{\Sigma}_j^{\text{new}} = \frac{1}{N_j} \sum_{l_i = j} \gamma_j(x_i)(x_i - \tilde{\mu}_j^{\text{new}})(x_i - \tilde{\mu}_j^{\text{new}})^T
    \]

- Of course, we don’t have such labels \( l_i \).
  - But we can guess what the labels might be based on our current mixture distribution estimate (credit assignment problem).

  - We get soft labels or posterior probabilities of which Gaussian generated which example:
    \[
    \gamma_j(x_i) = p(l_i = j|x_i, \theta) \quad \sum_{j=1}^2 \gamma_j(x_i) = 1 \quad \forall i = 1, \ldots, N
    \]

  - When the Gaussians are almost identical (as in the figure), then \( \gamma_j(x_i) = \gamma_1(x_i) \) for almost any given sample \( x_i \).
    \[\Rightarrow\text{Even small differences can help to determine how to update the Gaussians.}\]

EM - Technical 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 \( \Sigma = \sigma^2 I \) (this also holds in general)
  - Assume component \( j \) is exactly centered on data point \( x_i \). This data point will then contribute a term in the likelihood function

\[
\mathcal{N}(x_i; x_i, \sigma_j^2 I) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2\sigma_j^2}}
\]

  - For \( \sigma_j \to 0 \), this term goes to infinity!

  \[\Rightarrow\text{Need to introduce regularization}\]
  - Enforce minimum width for the Gaussians

EM - Technical Advice (2)

- EM is very sensitive to the initialization
  - Will converge to a local optimum of \( E \).
  - Convergence is relatively slow.
    \[\Rightarrow\text{Initialize with k-Means to get better results!}\]
  - k-Means is itself initialized randomly, will also only find a local optimum.
  - But convergence is much faster.

  - Typical procedure
    - Run k-Means \( M \) times (e.g. \( M = 10-100 \)).
    - Pick the best result (lowest error \( J \)).
    - Use this result to initialize EM.
      - Set \( \mu_k \) to the corresponding cluster mean from k-Means.
      - Initialize \( \Sigma_k \) to the sample covariance of the associated data points.
K-Means Clustering Revisited

- Interpreting the procedure
  1. Initialization: pick $K$ arbitrary centroids (cluster means)
  2. Assign each sample to the closest centroid. (E-Step)
  3. Adjust the centroids to be the means of the samples assigned to them. (M-Step)
  4. Go to step 2 (until no change)

Summary: Gaussian Mixture Models

- Properties
  - Very general, can represent any (continuous) distribution.
  - Once trained, very fast to evaluate.
  - Can be updated online.

- Problems / Caveats
  - Some numerical issues in the implementation
    - Need to apply regularization in order to avoid singularities.
  - EM for MoG is computationally expensive
    - Especially for high-dimensional problems!
    - More computational overhead and slower convergence than k-Means
    - Results very sensitive to initialization
    - Run k-Means for some iterations as initialization!
  - Need to select the number of mixture components $K$.
    - Model selection problem (see Lecture 10)

Applications

- Mixture models are used in many practical applications.
  - Wherever distributions with complex or unknown shapes need to be represented...

- Popular application in Computer Vision
  - Model distributions of pixel colors.
  - Each pixel is one data point in e.g., RGB space.
  - Learn a MoG to represent the class-conditional densities.
  - Use the learned models to classify other pixels.

Application: Background Model for Tracking

- Train background MoG for each pixel
  - Model “common” appearance variation for each background pixel.
  - Initialization with an empty scene.
  - Update the mixtures over time
    - Adapt to lighting changes, etc.

- Used in many vision-based tracking applications
  - Anything that cannot be explained by the background model is labeled as foreground (object).
  - Easy segmentation if camera is fixed.
Application: Image Segmentation

- 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.
- Simple segmentation procedure
  (building block for more complex applications)

Application: Color-Based Skin Detection

- Collect training samples for skin/non-skin pixels.
- Estimate MoG to represent the skin/non-skin densities
- Classify skin color pixels in novel images

Interested to Try It?

- Here’s how you can access a webcam in Matlab:

  ```matlab
  function out = webcam
  % uses "Image Acquisition Toolbox",
  adaptorName = 'winvideo';
  vidFormat = 'I420_320x240';
  vidObj1= videoinput(adaptorName, 1, vidFormat);
  set(vidObj1, 'ReturnedColorSpace', 'rgb');
  set(vidObj1, 'FramesPerTrigger', 1);
  out = vidObj1 ;

  cam = webcam();
  img=getsnapshot(cam);
  ```

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 (recommendable to read).

- Additional information
  - Original EM paper:
  - EM tutorial: