Mixture Models and EM

Recap: Histograms

- Basic idea:
  - Partition the data space into distinct bins with widths \( \Delta_i \) and count the number of observations, \( n_i \), in each bin.
  - Often, the same width is used for all bins, \( \Delta_i = \Delta \).
  - This can be done, in principle, for any dimensionality \( D \).

Recap: Kernel Density Estimation

- Approximation formula:
  \[
  p(x) \approx \frac{K}{NV} \\
  \text{fixed } V \quad \text{determine } K \\
  \text{fixed } K \quad \text{determine } V
  \]

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

Recap: Mixture of Gaussians (MoG)

- “Generative model”
  \[
  p(x) = \sum_{j=1}^{M} p(x|\theta_j) p(j) \\
  p(j) = \pi_j \\
  \text{“Weight” of mixture component}
  \]

Course Outline

- Fundamentals (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation
- Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns
- Generative Models (4 weeks)
  - Bayesian Networks
  - Markov Random Fields
We can already see that this will be difficult, since the Maximum Likelihood procedure involves computing a complex gradient function (non-linear mutual dependencies). Optimization of one Gaussian depends on all other Gaussians! It is possible to apply iterative numerical optimization here, but in the following, we will see a simpler method.

Mixture of Gaussians

- **Multivariate Gaussians**
  \[ p(x|\theta) = \sum_{j=1}^{M} \pi_j p(x|\theta_j) \]
  \[ p(x|\theta_j) = \frac{1}{(2\pi)^{D/2} |\Sigma_j|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right\} \]

- **Maximum Likelihood**
  \[ \theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_M, \mu_M, \Sigma_M) \]

- **Minimum:**
  \[ E = -\ln p(x|\theta) = -\sum_{n=1}^{N} \ln p(x_n|\theta) \]

  - Let's first look at \( \mu_j \):
    \[ \frac{\partial E}{\partial \mu_j} = 0 \]

    - We can already see that this will be difficult, since
    \[ \ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left( \sum_{j=1}^{M} \pi_j N(x_n; \mu_j, \Sigma_j) \right) \]

      This will cause problems!

- **But...**
  \[ \mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) x_n}{\sum_{n=1}^{N} \gamma_j(x_n)} = \frac{\pi_j N(x_n; \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k N(x_n; \mu_k, \Sigma_k)} \]

  - I.e. there is no direct analytical solution!

\[ \frac{\partial E}{\partial \mu_j} = f(\pi_1, \mu_1, \Sigma_1, \ldots, \pi_M, \mu_M, \Sigma_M) \]

- Complex gradient function (non-linear mutual dependencies)
- Optimization of one Gaussian depends on all other Gaussians!
- It is possible to apply iterative numerical optimization here, but in the following, we will see a simpler method.

Mixture of Gaussians - 1st Estimation Attempt

- Minimization:
  \[ E = -\sum_{n=1}^{N} \ln \frac{1}{(2\pi)^{D/2} |\Sigma_j|^{1/2}} \exp \left\{ -\frac{1}{2} (x_n - \mu_j)^T \Sigma_j^{-1} (x_n - \mu_j) \right\} \]

  - **Parameters:**
    \[ \theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_M, \mu_M, \Sigma_M) \]

  - **Minimization:**
    \[ \frac{\partial E}{\partial \mu_j} = -\sum_{n=1}^{N} \frac{\partial}{\partial \mu_j} \ln p(x_n|\theta_j) \]
    \[ = -\sum_{n=1}^{N} \left( \Sigma_j^{-1} (x_n - \mu_j) \right) \frac{p(x_n|\theta_j)}{\sum_{k=1}^{M} p(x_n|\theta_k)} \]
    \[ = -\Sigma_j^{-1} \sum_{n=1}^{N} (x_n - \mu_j) \frac{p(x_n|\theta_j)}{\sum_{k=1}^{M} p(x_n|\theta_k)} = 0 \]

- **We thus obtain**
  \[ \mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) x_n}{\sum_{n=1}^{N} \gamma_j(x_n)} \]

  "responsibility" of component \( j \) for \( x_n \)
Mixture of Gaussians - Other Strategy

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

\[ f(x) \]

\[ \mu_1 = \frac{\sum_{n=1}^{N} h(j = 1|x_n)x_n}{\sum_{i=1}^{N} h(j = 1|x_n)} \]
\[ \mu_2 = \frac{\sum_{n=1}^{N} h(j = 2|x_n)x_n}{\sum_{i=1}^{N} h(j = 2|x_n)} \]

ML for Gaussian #1
ML for Gaussian #2

\[ p(j = 1|x) > p(j = 2|x) \]

Bayes decision rule: Decide \( j = 1 \) if

Chicken and egg problem - what comes first?

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

- K-Means optimizes the following objective function:
  \[
  J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|^2
  \]
  
  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.

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

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**
Clustering with "soft assignments"

- Expectation step of the EM algorithm
  \[ p(j|x) \]
  \[ p(1|x) \quad 0.99 \quad 0.8 \quad 0.2 \quad 0.01 \]
  \[ p(2|x) \quad 0.01 \quad 0.2 \quad 0.8 \quad 0.99 \]

- Maximization step of the EM algorithm
  \[ \mu_j = \frac{\sum_{n=1}^{N} p(j|x_n) x_n}{\sum_{n=1}^{N} p(j|x_n)} \]

**EM Algorithm**

- Expectation-Maximization (EM) Algorithm
  - **E-Step**: softly assign samples to mixture components
    \[ \gamma_j(x_n) \leftarrow \frac{p_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}{\sum_{j=1}^{K} p_j \mathcal{N}(x_n | \mu_j, \Sigma_j)} \quad \forall j = 1, \ldots, K, n = 1, \ldots, N \]
  - **M-Step**: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[ \hat{N}_j = \sum_{n=1}^{N} \gamma_j(x_n) = \text{soft number of samples labeled } j \]
    \[ \hat{\Sigma}^\text{new}_j = \frac{\hat{N}_j}{\sum_{n=1}^{N} \gamma_j(x_n) x_n (x_n - \hat{\mu}^\text{new}_j)} \]
    \[ \hat{\mu}^\text{new}_j = \frac{1}{\sum_{n=1}^{N} \gamma_j(x_n)} \sum_{n=1}^{N} \gamma_j(x_n) x_n \]

**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 \( \Sigma_j = \sigma^2 I \) (this also holds in general).
  - Assume component \( j \) is exactly centered on data point \( x \).

- Need to introduce regularization
  - Enforce minimum width for the Gaussians
    - E.g., instead of \( \Sigma^{-1} \), use \( (\Sigma + \sigma^2 I)^{-1} \)

- Convergence is relatively slow.

- 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.

**EM - Technical Advice (2)**

- EM is very sensitive to the initialization
  - Will converge to a local optimum of \( E \).
  - Convergence is relatively slow.

- 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 \( \mu \) to the corresponding cluster mean from k-Means.
    - Initialize \( \Sigma \) 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. \((E\text{-Step})\)
3. Adjust the centroids to be the means of the samples assigned to them. \((M\text{-Step})\)
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
  \( \Rightarrow \) Need to apply regularization in order to avoid singularities.
- EM for MoG is computationally expensive
  \( \Rightarrow \) Especially for high-dimensional problems!
  \( \Rightarrow \) More computational overhead and slower convergence than k-Means
  \( \Rightarrow \) Results very sensitive to initialization
  \( \Rightarrow \) Run k-Means for some iterations as initialization!
- Need to select the number of mixture components \( K \).
  \( \Rightarrow \) Model selection problem (see Lecture 16)

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

Applications

- Mixture models are used in many practical applications.
  \( \Rightarrow \) 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.
  \( \Rightarrow \) Learn a MoG to represent the class-conditional densities.
  \( \Rightarrow \) 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:

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