Recap: Mixture of Gaussians (MoG)

- "Generative model"
- "Weight" of mixture component

\[ p(j) = \pi_j \]

\[ p(x|\theta_j) \]

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

Recap: Estimating MoGs – Iterative Strategy

- Assuming we knew the mixture components...

\[ f(x) \]

- ML for Gaussian #1
- ML for Gaussian #2

\[ h(j = 1|x_n) = 1 111 \]
\[ h(j = 2|x_n) = 0 000 \]

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

Recap: 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.

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
- Classification Approaches
  - Linear Discriminants
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns
- Deep Learning
  - Foundations
  - Convolutional Neural Networks
  - Recurrent Neural Networks
Recap: EM Algorithm

- Expectation-Maximization (EM) Algorithm
  - E-Step: softly assign samples to mixture components
    \[ \gamma_j(x_n) \leftarrow \frac{\pi_j N(x_n \mid \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \Sigma_k)} \quad \forall j = 1, \ldots, K; \quad n = 1, \ldots, N \]
  - M-Step: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[ \hat{N}_j \leftarrow \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n) \]
    \[ \hat{\mu}^{new}_j \leftarrow \frac{1}{N_j} \sum_{n=1}^{N_j} \gamma_j(x_n) x_n \]
    \[ \hat{\Sigma}^{new}_j \leftarrow \frac{1}{N_j} \sum_{n=1}^{N_j} \gamma_j(x_n) (x_n - \hat{\mu}^{new}_j)(x_n - \hat{\mu}^{new}_j)^T \]

Slide adapted from Bernt Schiele

Slide credit: Bernt Schiele

Topics of This Lecture

- Linear discriminant functions
  - Definition
  - Extension to multiple classes
- Least-squares classification
  - Derivation
  - Shortcomings
- Generalized linear models
  - Connection to neural networks
  - Generalized linear discriminants & gradient descent

Slide credit: Bernt Schiele

Learning Discriminant Functions

- General classification problem
  - Goal: take a new input \( x \) and assign it to one of \( K \) classes \( C_k \).
  - Given: training set \( X = \{ x_1, \ldots, x_N \} \) with target values \( T = \{ t_1, \ldots, t_N \} \).
  - Learn a discriminant function \( y(x) \) to perform the classification.

- 2-class problem
  - Binary target values: \( t_n \in \{ 0,1 \} \)

- K-class problem
  - 1-of-K coding scheme, e.g. \( \mathbf{t}_n = (0, 1, 0, 0, 0)^T \)

Slide credit: Bernt Schiele
Two simple strategies

- **One-vs-all classifiers**
  - Combine $K$ linear functions if
    - $y_k(x) = w_k^T x + w_{k0}$
  - Resulting decision hyperplanes:
    - $y_k(x) = w_k^T x + w_{k0}$
  - It can be shown that the decision regions of such a discriminant are always singly connected and convex.
  - This makes linear discriminant models particularly suitable for problems for which the conditional densities $p(x|y_i)$ are unimodal.

- **One-vs-one classifiers**
  - How many classifiers do we need in both cases?
  - What difficulties do you see for those strategies?
General Classification Problem

- Classification problem
  - Let's consider $K$ classes described by linear models
    \[ y_k(x) = w_k^T x + w_{k0}, \quad k = 1, \ldots, K \]
  - We can group those together using vector notation
    \[ y(x) = \tilde{W}^T \tilde{x} \]
    where
    \[ \tilde{W} = \begin{bmatrix} w_{10} & \ldots & w_{K0} \\ w_{11} & \ldots & w_{K1} \\ \vdots & \ddots & \vdots \\ w_{1D} & \ldots & w_{KD} \end{bmatrix} \]
  - The output will again be in 1-of-$K$ notation.
  \[ \Rightarrow \text{We can directly compare it to the target value } t = [t_1, \ldots, t_K]^T \]

Least-Squares Classification

- Simplest approach
  - Directly try to minimize the sum-of-squares error
  - We could write this as
    \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_{kn})^2 \]
    \[ = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (w_k^T x_n - t_{kn})^2 \]
    \[ = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left[ (w_k^T x_n - t_{kn})^2 \right] \]
    \[ \Rightarrow \text{But let's stick with the matrix notation for now...} \]
    \[ \Rightarrow \text{The result will be simpler to express and we'll learn some} \]
    \[ \text{nice matrix algebra rules along the way...} \]

- Of course, we need to solve for the discriminant function parameters.

Least-Squares Classification

- Multi-class case
  - Let's formulate the sum-of-squares error in matrix notation
    \[ E_D(\tilde{W}) = \frac{1}{2} \text{Tr} \left\{ (\tilde{W} \tilde{x} - T)^T (\tilde{W} \tilde{x} - T) \right\} \]
  - Taking the derivative yields
    \[ \frac{\partial}{\partial \tilde{W}} E_D(\tilde{W}) = \frac{1}{2} \text{Tr} \left\{ (\tilde{W} \tilde{x} - T)^T (\tilde{W} \tilde{x} - T) \right\} \]
    \[ = \frac{1}{2} \text{Tr} \left\{ \frac{\partial}{\partial \tilde{W}} (\tilde{W} \tilde{x} - T)^T (\tilde{W} \tilde{x} - T) \right\} \]
    \[ = \frac{1}{2} \text{Tr} \left\{ \tilde{X}^T (\tilde{W} \tilde{x} - T) \right\} \]
    \[ \Rightarrow \text{Exact, closed-form solution for the discriminant function parameters.} \]
Problems with Least Squares

- Least-squares is very sensitive to outliers!
  - The error function penalizes predictions that are "too correct".

- Another example:
  - 3 classes (red, green, blue)
  - Linearly separable problem
  - Least-squares solution: Most green points are misclassified!

- Deeper reason for the failure
  - Least-squares corresponds to Maximum Likelihood under the assumption of a Gaussian conditional distribution.
  - However, our binary target vectors have a distribution that is clearly non-Gaussian!
  - ⇒ Least-squares is the wrong probabilistic tool in this case!

Topics of This Lecture

- Linear discriminant functions
  - Definition
  - Extension to multiple classes
- Least-squares classification
  - Derivation
  - Shortcomings
- Generalized linear models
  - Connection to neural networks
  - Generalized linear discriminants & gradient descent

Generalized Linear Models

- Consider 2 classes:
  \[ p(C_1 | x) = \frac{p(x | C_1)p(C_1)}{p(x | C_1)p(C_1) + p(x | C_2)p(C_2)} \]
  \[ = \frac{1}{1 + \exp(-a)} \]
  \[ = 1 + \exp(-a) \]
  \[ = g(a) \]
  \[ \text{with } a = \ln \left( \frac{p(x | C_1)p(C_1)}{p(x | C_2)p(C_2)} \right) \]

Logistic Sigmoid Activation Function

\[ g(a) = \frac{1}{1 + \exp(-a)} \]

Example: Normal distributions with identical covariance
Normalized Exponential

- General case of $K > 2$ classes:
  \[
  p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)}
  \]
  \[
  = \frac{\exp(a_k)}{\sum_j \exp(a_j)}
  \]
  with $a_k = \ln p(x|C_k)p(C_k)$
  
  - This is known as the **normalized exponential** or **softmax** function
  
  - Can be regarded as a multiclass generalization of the logistic sigmoid.

Relationship to Neural Networks

- 2-Class case
  \[
  y(x) = g \left( \sum_{i=0}^{D} w_i x_i \right) \quad \text{with} \quad x_0 = 1 \quad \text{constant}
  \]

- Neural network ("single-layer perceptron")

Other Motivation for Nonlinearity

- Recall least-squares classification
  
  - One of the problems was that data points that are "too correct" have a strong influence on the decision surface under a squared-error criterion.

  \[
  E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_n)^2
  \]
  
  - Reason: the output of $y(x; w)$ can grow arbitrarily large for some $x_n$:
  
  \[
  y(x; w) = w^T x + w_0
  \]
  
  - By choosing a suitable nonlinearity (e.g. a sigmoid), we can limit those influences

  \[
  y(x; w) = g(w^T x + w_0)
  \]

Discussion: Generalized Linear Models

- Advantages
  
  - The nonlinearity gives us more flexibility.
  
  - Can be used to limit the effect of outliers.
  
  - Choice of a sigmoid leads to a nice probabilistic interpretation.

- Disadvantage
  
  - Least-squares minimization in general no longer leads to a closed-form analytical solution.
  
  ⇒ Need to apply iterative methods.

  ⇒ Gradient descent.
Learning in Neural Networks

Up to now: restrictive assumption
- Only consider linear decision boundaries

Classical counterexample: XOR

By choosing the right training data points:
- The error function can in general no longer be minimized in closed form.

Learning in Neural Networks
- K functions (outputs) $y_i(x; w)$
- Single-layer networks: $\phi_j$ are fixed, only weights $w$ are learned.
- Multi-layer networks: both the $w$ and the $\phi_j$ are learned.

We will take a closer look at neural networks from lecture 11 on. For now, let’s first consider generalized linear discriminants in general…

Generalized Linear Discriminants

Model
$$ y_k(x) = \sum_{j=0}^{M} w_{kj} \phi_j(x) = y_k(x; w) $$

- $K$ functions (outputs $y_i(x; w)$)
- Learning in Neural Networks
  - Single-layer networks: $\phi_j$ are fixed, only weights $w$ are learned.
  - Multi-layer networks: both the $w$ and the $\phi_j$ are learned.

We will take a closer look at neural networks from lecture 11 on. For now, let’s first consider generalized linear discriminants in general…

Error function (least-squares error) of linear model
$$ E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_{kn})^2 $$
$$ = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2 $$

Gradient Descent

- Learning the weights $w$:
  - $N$ training data points: $X = \{x_1, \ldots, x_N\}$
  - $K$ outputs of decision functions: $y_k(x_n; w)$
  - Target vector for each data point: $T = \{t_{1n}, \ldots, t_{Nn}\}$

- Error function (least-squares error) of linear model
$$ E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_{kn})^2 $$

- “Batch learning”
  $$ w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta \frac{\partial E(w)}{\partial w_{kj}} (w^{(r)}) $$
  $\eta$: Learning rate

- Compute the gradient based on all training data:
$$ \frac{\partial E(w)}{\partial w_{kj}} $$

Slide credit: Bernt Schiele

Slide credit: Bernt Schiele
Gradient Descent – Basic Strategies

- "Sequential updating"
  
  $$E(w) = \sum_{n=1}^{N} E_n(w)$$
  
  $$w_{k}^{(\tau+1)} = w_k^{(\tau)} - \eta \frac{\partial E_n(w)}{\partial w_{k}} |_{w^{(\tau)}}$$
  
  - Learning rate
  
  - Compute the gradient based on a single data point at a time:
    
    $$\frac{\partial E_n(w)}{\partial w_{k}}$$

- Several possible parameter choices minimize training error.

- Limitations / Caveats
  
  - Delta gradient descent approaches available
    
    $$50$$
  
    - Often introduce additional parameters.
      
      - More information on Linear Discriminant Functions can be
        found in Chapter 4 of Bishop’s book (in particular Chapter 4.1).