Machine Learning – Lecture 5
Linear Discriminant Functions
23.10.2019

Recap: Mixture of Gaussians (MoG)
* "Generative model"

\[ p(j) = \pi_j \]

"Weight" of mixture component

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

Mixture density

Recap: Estimating MoGs – Iterative Strategy
* Assuming we knew the mixture components...

\[ f(x) \]

ML for Gaussian #1
\[ h(j = 1|x_n) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \]
\[ h(j = 2|x_n) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \]

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

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

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
    \[
    \gamma_j(x_n) \leftarrow \frac{\pi_j \mathcal{N}(x_n; \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; \mu_k, \Sigma_k)} \quad \forall j = 1, \ldots, K, \ n = 1, \ldots, N
    \]
  - M-Step: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[
    \hat{N}_j = \sum_{n=1}^{N} \gamma_j(x_n) = \text{soft number of samples labeled } j
    \]
    \[
    \bar{x}_j^{\text{new}} \leftarrow \frac{1}{N_j} \sum_{n=1}^{N} \gamma_j(x_n) x_n
    \]
    \[
    \bar{\mu}_j^{\text{new}} = \frac{1}{N_j} \sum_{n=1}^{N} \gamma_j(x_n) x_n
    \]
    \[
    \sum_{j=1}^{J} \gamma_j(x_n) (x_n - \mu_j^{\text{new}})^T (x_n - \mu_j^{\text{new}})
    \]

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

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

Discriminant Functions

- Bayesian Decision Theory
  - Model conditional probability densities \( p(x|C_k) \) and priors \( p(C_k) \)
  - Compute posteriors \( p(C_k|x) \) (using Bayes’ rule)
  - Minimize probability of misclassification by maximizing \( p(C|x) \)
- New approach
  - Directly encode decision boundary
    - Without explicit modeling of probability densities
    - Minimize misclassification probability directly.
Recap: Discriminant Functions

- Formulate classification in terms of comparisons
  - Discriminant functions \( y_1(x), \ldots, y_K(x) \)
  - Classify \( x \) as class \( C_i \) if \( y_i(x) > y_j(x) \) \( \forall j \neq k \)
- Examples (Bayes Decision Theory)
  \[
  y_k(x) = p(C_k|x) \\
  y_k(x) = p(x|C_k)p(C_k) \\
  y_k(x) = \log p(x|C_k) + \log p(C_k)
  \]

Discriminant Functions

- Example: 2 classes
  \[
  y_1(x) > y_2(x) \\
  \Leftrightarrow \ y_1(x) - y_2(x) > 0 \\
  \Leftrightarrow \ y(x) > 0
  \]
- Decision functions (from Bayes Decision Theory)
  \[
  y(x) = p(C_1|x) - p(C_2|x) \\
  y(x) = \ln \frac{p(x|C_1)}{p(x|C_2)} + \ln \frac{p(C_1)}{p(C_2)}
  \]

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 \( \mathbf{X} = \{x_1, \ldots, x_n\} \)
    - with target values \( \mathbf{T} = \{t_1, \ldots, t_n\} \).
  - \( \Rightarrow \) 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. \( t_n = (0, 1, 0, 0)^T \)

Linear Discriminant Functions

- 2-class problem
  - \( y(x) > 0 \): Decide for class \( C_1 \), else for class \( C_2 \)
  - In the following, we focus on linear discriminant functions
    \[
    y(x) = \mathbf{w}^T \mathbf{x} + w_0
    \]
    - weight vector \( \mathbf{w} \)
      - bias \( w_0 \) (threshold)
    - If a data set can be perfectly classified by a linear discriminant, then we call it **linearly separable**.

Linear Discriminant Functions

- Decision boundary \( y(x) = 0 \) defines a hyperplane
  - Normal vector: \( \mathbf{w} \)
  - Offset: \( -\frac{w_0}{||\mathbf{w}||} \)
  \[
  y(x) = \mathbf{w}^T \mathbf{x} + w_0
  \]
- Notation
  - \( D \): Number of dimensions
    \[
    \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \\
    \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}
    \]
    \[
    y(x) = \mathbf{w}^T \mathbf{x} + w_0
    = \sum_{i=1}^{D} w_ix_i + w_0
    = \sum_{i=0}^{D} w_ix_i \quad \text{with} \quad x_0 = 1 \quad \text{constant}
    \]
Extension to Multiple Classes

- Two simple strategies
  - One-vs-all classifiers
  - One-vs-one classifiers

Resulting decision hyperplanes:

\[
\begin{align*}
\text{Least:} & \quad w_k [x] + w_k0 > 0, \\
\text{Resulting decision hyperplanes:} & \quad (w_k - w_j)^T x + (w_k0 - w_j0) = 0.
\end{align*}
\]

- It can be shown that the decision regions of a linear discriminant are always connected and convex.
  - Convex means if \( x_k \) and \( x_j \) are both in \( R_k \), then any point \( x \) on the connecting line between \( x_k \) and \( x_j \) is also in \( R_k \).
  - This makes linear discriminant models particularly suitable for problems for which the conditional densities \( p(x|y_k) \) are unimodal.

Topics of This Lecture

- 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
      \[
y(x) = W^T x
      \]
    - where
      \[
      W = [\hat{w}_1, \ldots, \hat{w}_K] =
      \begin{bmatrix}
      w_{10} & \cdots & w_{K0} \\
      w_{11} & \cdots & w_{K1} \\
      \vdots & \ddots & \vdots \\
      w_{1D} & \cdots & w_{KD}
      \end{bmatrix}
      \]
  - The output will again be in 1-of-K notation.
    - We can directly compare it to the target values \( t \) of \( Y(x) = XW \) and compare this to the target matrix \( T \) where
      \[
      \hat{W} = [\hat{w}_1, \ldots, \hat{w}_K],
      \hat{X} = [\hat{x}_1, \ldots, \hat{x}_N],
      T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}
      \]

  - Result of the comparison:
    \[
    \hat{X} W - T
    \]
Simplest approach

- Directly try to minimize the sum-of-squares error
- We could write this as
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_{kn})^2 \]
  \[ = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (w^T x_n - t_{kn})^2 \]

- But let’s stick with the matrix notation for now...
- (The result will be simpler to express and we’ll learn some nice matrix algebra rules along the way...)

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!

Connection to neural networks

- Let’s formulate the linear discriminant functions
- \[ w = \mathbf{x}^T \]
- \[ \mathbf{y} = \mathbf{y}^T \]
- \[ \hat{y} = \mathbf{x}^T \]

Least squares is very sensitive to outliers!

- The error function penalizes predictions that are “too correct”.

Problems with Least Squares

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

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

The error function penalizes predictions that are “too correct”.

- Multi-class case
  - Let’s formulate the sum-of-squares error in matrix notation
  \[ E_D(W) = \frac{1}{2} \text{Tr} \left\{ (\mathbf{X}W - T)^T(\mathbf{X}W - T) \right\} \]
  - Taking the derivative yields
  \[ \frac{\partial}{\partial W} E_D(W) = \frac{1}{2} \text{Tr} \left\{ (\mathbf{X}W - T)^T(\mathbf{X}W - T) \right\} \]
  \[ = \frac{1}{2} \frac{\partial}{\partial W} \text{Tr} \left\{ (\mathbf{X}W - T)^T(\mathbf{X}W - T) \right\} \]
  \[ = \mathbf{X}^T(\mathbf{X}W - T) \]

Exact, closed-form solution for the discriminant function parameters.

\[ \hat{\mathbf{y}} = \mathbf{X}^T(\mathbf{X} \hat{\mathbf{W}} - T) \]

\[ \hat{\mathbf{W}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{T} \]

\[ \hat{\mathbf{W}} \text{ is the wrong probabilistic tool in this case!} \]
Generalized Linear Models

- Linear model
  \[ y(x) = w^T x + w_0 \]
- Generalized linear model
  \[ y(x) = g(w^T x + w_0) \]
  - \( g(\cdot) \) is called an activation function and may be nonlinear.
  - The decision surfaces correspond to \( y(x) = \text{const.} \iff w^T x + w_0 = \text{const.} \)
  - If \( g \) is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of \( x \).

Logistic Sigmoid Activation Function

\[ g(a) \equiv \frac{1}{1 + e^{-a}} \]

Example: Normal distributions with identical covariance

\[ p(x | a) \quad p(x | b) \]
\[ p(a | x) \quad p(b | x) \]

Normalized Exponential

- General case of \( K > 2 \) classes:
  \[ p(C_k | x) = \frac{p(x | C_k)p(C_k)}{\sum_j p(x | C_j)p(C_j)} \]
  \[ = \frac{1}{1 + \exp(-a)} \equiv g(a) \]
  \[ a_k = \ln \frac{p(x | C_k)p(C_k)}{p(x | C_j)p(C_j)} \]

   - 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”)

\[ \text{threshold} \quad w_0 \quad w_1 \quad w_2 \quad \text{weights} \]
\[ x_0 = 1 \quad x_1 \quad x_2 \quad \text{inputs} \quad \text{output} \]

- Multi-class case
  \[ y_k(x) = g \left( \sum_{i=0}^{D} w_{k,i} x_i \right) \quad \text{with} \quad x_0 = 1 \quad \text{constant} \]
- Multi-class perceptron
Logistic Discrimination

- If we use the logistic sigmoid activation function…
  \[ g(a) = \frac{1}{1 + \exp(-a)} \]
  \[ y(x) = g(w^T x + w_0) \]

… then we can interpret the \( y(x) \) as posterior probabilities!

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 \( y(x_n; w) \) 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

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.

Linear Separability

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

Classical counterexample: XOR

Generalized Linear Discriminants

- Generalization
  - Transform vector \( x \) with \( M \) nonlinear basis functions \( \phi_j(x) \):
    \[ y_k(x) = \sum_{j=1}^{M} w_{kj} \phi_j(x) + w_{k0} \]
  - Purpose of \( \phi_j(x) \): basis functions
  - Allow non-linear decision boundaries.
  - By choosing the right \( \phi_j \), every continuous function can (in principle) be approximated with arbitrary accuracy.

- Notation
  \[ y_k(x) = \sum_{j=0}^{M} w_{kj} \phi_j(x) \quad \text{with} \quad \phi_0(x) = 1 \]
Gradient Descent
• Learning the weights $w$:
  - $N$ training data points: $X = \{x_1, \ldots, x_N\}$
  - $K$ outputs of decision functions: $y_k(x; w)$
  - Target vector for each data point: $T = \{t_1, \ldots, t_N\}$

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

\[ \text{Gradient Descent} \]
\[ \text{Basic Strategies} \]
• “Batch learning”
  $$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(w)}{\partial w_{kj}}_{w^{(\tau)}}$$
  $$\eta: \text{Learning rate}$$

\[ \text{Gradient Descent} \]
• “Sequential updating”
  $$E(w) = \sum_{n=1}^{N} E_n(w)$$
  $$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E_n(w)}{\partial w_{kj}}_{w^{(\tau)}}$$
  $$\eta: \text{Learning rate}$$

\[ \text{Gradient Descent} \]
• Error function
  $$E(w) = \sum_{n=1}^{N} E_n(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2$$
  $$E_n(w) = \frac{1}{2} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2$$
  $$\frac{\partial E_n(w)}{\partial w_{kj}} = \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \phi_j(x_n)$$
  $$= (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$

\[ \text{Gradient Descent} \]
• Delta rule (LMS rule)
  $$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$
  $$= w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(x_n)$$
  where
  $$\delta_{kn} = y_k(x_n; w) - t_{kn}$$

\[ \Rightarrow \text{Simply feed back the input data point, weighted by the classification error.} \]
Gradient Descent

- Cases with differentiable, non-linear activation function

\[ y_k(x) = g(a_k) = g \left( \sum_{j=0}^{M} w_{kj} \phi_j(x_n) \right) \]

- Gradient descent

\[
\begin{align*}
\frac{\partial E_n(w)}{\partial w_{kj}} &= \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \\
w_{kj}^{(s+1)} &= w_{kj}^{(s)} - \eta \frac{\partial E_n(w)}{\partial w_{kj}} \\
\delta_{kn} &= \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn})
\end{align*}
\]

Summary: Generalized Linear Discriminants

- Properties
  - General class of decision functions.
  - Nonlinearity \( g(\cdot) \) and basis functions \( \phi_j \) allow us to address linearly non-separable problems.
  - Shown simple sequential learning approach for parameter estimation using gradient descent.
  - Better 2nd order gradient descent approaches available (e.g. Newton-Raphson).

- Limitations / Caveats
  - Flexibility of model is limited by curse of dimensionality
    - \( g(\cdot) \) and \( \phi_j \) often introduce additional parameters.
    - Models are either limited to lower-dimensional input space or need to share parameters.
  - Linearly separable case often leads to overfitting.
    - Several possible parameter choices minimize training error.

References and Further Reading

- More information on Linear Discriminant Functions can be found in Chapter 4 of Bishop’s book (in particular Chapter 4.1).

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