Machine Learning - Lecture 5

Linear Discriminant Functions

04.11.2013

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de

Many slides adapted from B. Schiele
Announcements

- L2P is working
  - Yay!
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
Recap: Mixture of Gaussians (MoG)

- "Generative model"

\[ p(j) = \pi_j \]

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

"Weight" of mixture component

Mixture component

Mixture density
Recap: Estimating MoGs - Iterative Strategy

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

\[ f(x) \]

**ML for Gaussian #1**

Assumed known \( 1 \ 1 \ 1 \ 1 \)

\[ h(j = 1 | x_n) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \]

\[ h(j = 2 | x_n) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \]

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

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

**ML for Gaussian #2**

\[ 22 \ 2 \ 2 \ j \]

\[ h(j = 1 | x_n) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \]

\[ h(j = 2 | x_n) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \]

Slide credit: Bernt Schiele
Recap: Estimating MoGs - Iterative Strategy

• Assuming we knew the mixture components...

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

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

\[ j \]

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

\[ p(j = 1 | x_n) > p(j = 2 | x_n) \]
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.
Recap: EM Algorithm

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

Slide adapted from 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
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 \( C_k \) if
    \[ y_k(x) > y_j(x) \quad \forall j \neq k \]

- 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**
  
  \[ y_1(x) > y_2(x) \]
  
  \[ \iff y_1(x) - y_2(x) > 0 \]
  
  \[ \iff y(x) > 0 \]

- **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 \( X = \{x_1, \ldots, x_N\} \)
    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. \( t_n = (0, 1, 0, 0, 0)^T \)
Linear Discriminant Functions

• 2-class problem
  
  \( y(x) > 0 : \text{Decide for class } C_1, \text{ else for class } C_2 \)

• In the following, we focus on linear discriminant functions

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

- weight vector
- “bias” (= 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: $w$
  - Offset: $\frac{-w_0}{\|w\|}$

$$y(x) = w^T x + w_0$$

$y > 0$

$y = 0$

$y < 0$
Linear Discriminant Functions

- **Notation**
  - \( D \): Number of dimensions

\[
\begin{align*}
\mathbf{x} &= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}, & \mathbf{w} &= \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \\
y(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + w_0 \\
&= \sum_{i=0}^{D} w_i x_i + w_0 \\
&= \sum_{i=0}^{D} w_i x_i \quad \text{with } x_0 = 1 \text{ constant}
\end{align*}
\]
Extension to Multiple Classes

• Two simple strategies

**One-vs-all classifiers**

**One-vs-one classifiers**

- How many classifiers do we need in both cases?
- What difficulties do you see for those strategies?
Extension to Multiple Classes

• Problem
  - Both strategies result in regions for which the pure classification result ($y_k > 0$) is ambiguous.
  - In the one-vs-all case, it is still possible to classify those inputs based on the continuous classifier outputs $y_k > y_j \forall j \neq k$.

• Solution
  - We can avoid those difficulties by taking $K$ linear functions of the form
    \[ y_k(x) = w_k^T x + w_{k0} \]
    and defining the decision boundaries directly by deciding for $C_k$ iff $y_k > y_j \forall j \neq k$.
  - This corresponds to a 1-of-K coding scheme
    \[ t_n = (0, 1, 0, \ldots, 0, 0)^T \]
Extention to Multiple Classes

- **K-class discriminant**
  - Combination of K linear functions
    \[ y_k(x) = w_k^T x + w_{k0} \]
  - Resulting decision hyperplanes:
    \[ (w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0 \]

- 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 \( p(x | w_i) \) are unimodal.

Image source: C.M. Bishop, 2006
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
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
  \[ \tilde{W} = [\tilde{w}_1, \ldots, \tilde{w}_K] = \begin{bmatrix} w_{10} & \cdots & w_{K0} \\ w_{11} & \cdots & w_{K1} \\ \vdots & \ddots & \vdots \\ w_{1D} & \cdots & w_{KD} \end{bmatrix} \]

  - The output will again be in 1-of-$K$ notation.
    \[ t = [t_1, \ldots, t_k]^T. \]

B. Leibe
General Classification Problem

• Classification problem
  
  ➢ For the entire dataset, we can write

  \[ Y(\tilde{X}) = \tilde{X}\tilde{W} \]

  and compare this to the target matrix \( T \) where

  \[
  \tilde{W} = [\tilde{w}_1, \ldots, \tilde{w}_K]
  \]

  \[
  \tilde{X} = \begin{bmatrix}
  x_1^T \\
  \vdots \\
  x_N^T
  \end{bmatrix}
  \]

  \[
  T = \begin{bmatrix}
  t_1^T \\
  \vdots \\
  t_N^T
  \end{bmatrix}
  \]

  ➢ Result of the comparison:

  \[ \tilde{X}\tilde{W} - T \]

  Goal: Choose \( \tilde{W} \) such that this is minimal!
Least-Squares Classification

• Simplest approach
  - Directly try to minimize the sum-of-squares error
  - We could write this as

\[
E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_{kn})^2
\]

\[
= \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (w_k^T x_n - t_{kn})^2
\]

- 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...)
Least-Squares Classification

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

\[ \sum_{i,j} a_{i,j}^2 = \text{Tr}\{A^T A\} \]

\[ \frac{\partial Z}{\partial X} = \frac{\partial Z}{\partial Y} \frac{\partial Y}{\partial X} \]

\[ \frac{\partial}{\partial A} \text{Tr}\{A\} = I \]
Least-Squares Classification

- Minimizing the sum-of-squares error

\[ \frac{\partial}{\partial \tilde{W}} E_D(\tilde{W}) = \tilde{X}^T (\tilde{X} \tilde{W} - T) \overset{!}{=} 0 \]

\[ \tilde{X} \tilde{W} = T \]

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

\[ = \tilde{X}^\dagger T \] \text{ “pseudo-inverse”}

- We then obtain the discriminant function as

\[ y(x) = \tilde{W}^T \tilde{x} = T^T \left( \tilde{X}^\dagger \right)^T \tilde{x} \]

⇒ 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”.

Image source: C.M. Bishop, 2006
Problems with Least-Squares

• 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!
  \[ \Rightarrow \] Least-squares is the wrong probabilistic tool in this case!

Image source: C.M. Bishop, 2006
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

- 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 \( x \).
Generalized Linear Models

- Consider 2 classes:

\[
p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}
\]

\[
= \frac{1}{1 + \frac{p(x|C_2)p(C_2)}{p(x|C_1)p(C_1)}}
\]

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

with \( a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)} \)
Logistic Sigmoid Activation Function

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

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

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

- Neural network ("single-layer perceptron")

Slide credit: Bernt Schiele
Relationship to Neural Networks

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

- Multi-class perceptron
Logistic Discrimination

- If we use the logistic sigmoid activation function...

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

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

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

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

- **Classical counterexample: XOR**
Linear Separability

• Even if the data is not linearly separable, a linear decision boundary may still be “optimal”.
  - Generalization
  - E.g. in the case of Normal distributed data (with equal covariance matrices)

• Choice of the right discriminant function is important and should be based on
  - Prior knowledge (of the general functional form)
  - Empirical comparison of alternative models
  - Linear discriminants are often used as benchmark.
Generalized Linear Discriminants

- Generalization
  - Transform vector $\mathbf{x}$ with $M$ nonlinear basis functions $\phi_j(\mathbf{x})$:
    \[
y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj} \phi_j(\mathbf{x}) + w_{k0}
    \]
  - Purpose of $\phi_j(\mathbf{x})$: 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
  \[
y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj} \phi_j(\mathbf{x}) \quad \text{with} \quad \phi_0(\mathbf{x}) = 1
  \]
Generalized Linear Discriminants

• Model

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

- \( K \) functions (outputs) \( y_k(x; w) \)

• 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.

- In the following, we will not go into details about neural networks in particular, but consider generalized linear discriminants in general...
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: $T = \{t_1, \ldots, t_N\}$

- Error function (least-squares error) of linear model

\[
E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_{kn})^2
\]

\[
= \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2
\]

Slide credit: Bernt Schiele
Gradient Descent

• Problem
  - The error function can in general no longer be minimized in closed form.

• Idea (Gradient Descent)
  - Iterative minimization
  - Start with an initial guess for the parameter values $w_{kj}^{(0)}$.
  - Move towards a (local) minimum by following the gradient.

  $w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(w)}{\partial w_{kj}} |_{w(\tau)}$

  $\eta$: Learning rate

  - This simple scheme corresponds to a 1st-order Taylor expansion (There are more complex procedures available).
Gradient Descent - Basic Strategies

- “Batch learning”

\[ w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(w)}{\partial w_{kj}} \right|_{w^{(\tau)}} \]

\( \eta \): Learning rate

- Compute the gradient based on all training data:

\[ \frac{\partial E(w)}{\partial w_{kj}} \]
Gradient Descent - Basic Strategies

- “Sequential updating”

\[ E(w) = \sum_{n=1}^{N} E_n(w) \]

\[ w^{(\tau+1)}_{k,j} = w^{(\tau)}_{k,j} - \eta \left. \frac{\partial E_n(w)}{\partial w_{k,j}} \right|_{w^{(\tau)}} \]

\( \eta \): Learning rate

- Compute the gradient based on a single data point at a time:

\[ \frac{\partial E_n(w)}{\partial w_{k,j}} \]
Gradient Descent

- Error function

\[ E(w) = \sum_{n=1}^{N} E_n(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2 \]

\[ E_n(w) = \frac{1}{2} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2 \]

\[ \frac{\partial E_n(w)}{\partial w_{kj}} = \left( \sum_{\tilde{j}=1}^{M} w_{k\tilde{j}} \tilde{\phi}_{\tilde{j}}(x_n) - t_{kn} \right) \phi_j(x_n) \]

\[ = (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \]
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

\[
\delta_{kn} = y_k(x_n; w) - t_{kn}
\]

⇒ Simply feed back the input data point, weighted by the classification error.
Gradient Descent

- Cases with differentiable, non-linear activation function

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

- Gradient descent

\[
\frac{\partial E_n(w)}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n)
\]

\[
w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(x_n)
\]

\[
\delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn})
\]
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 2\textsuperscript{nd} 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