Pattern Classification

All materials in these slides were taken from *Pattern Classification (2nd ed)* by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000

with the permission of the authors and the publisher
Chapter 5: Linear Discriminant Functions (Sections 5.1-5-3)

- Introduction
- Linear Discriminant Functions and Decisions Surfaces
- Generalized Linear Discriminant Functions
Introduction

• In chapter 3, form of the underlying p.d.f was assumed to be known; parametric case; training samples used to estimate density parameters

• In chapter 4, form of the density was not known; training samples used to estimate the density function (non-parametric methods).

• Now, suppose we only know form of the discriminant function; while the assumed form may not be optimal, this approach is very simple to use

• Discriminant functions can either be linear in $x$ or linear in some given set of functions of $x$ (nonlinear in $x$)
Linear Discriminant Functions

- A discriminant function that is a linear combination of input features can be written as

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

- Sign of the function value gives the class label.
- Weight vector
- Bias or Threshold weight.

Figure 5.1: A simple linear classifier having \( d \) input units, each corresponding to the values of the components of an input vector. Each input feature value \( x_i \) is multiplied by its corresponding weight \( w_i \); the output unit sums all these products and emits a +1 if \( w^T x + w_0 > 0 \) or a -1 otherwise.
Linear Discriminant Functions

• A two-category classifier with a discriminant function of the form \( g(x) \) uses the following rule:
Decide \( \omega_1 \) if \( g(x) > 0 \) and \( \omega_2 \) if \( g(x) < 0 \)
\( \iff \) Decide \( \omega_1 \) if \( w^t x > -w_0 \) and \( \omega_2 \) otherwise

If \( g(x) = 0 \) \( \Rightarrow \) \( x \) can be assigned to either class
• Equation \( g(x) = 0 \) defines the decision surface that separates points assigned to the category \( \omega_1 \) from points assigned to the category \( \omega_2 \).

• When \( g(x) \) is linear, the decision surface is a hyperplane.

• Algebraic measure of the distance from \( x \) to the hyperplane.

• Problem of finding LDF will be formulated as a problem of minimizing a criterion function.
Decision Surface

- $g(x) = 0$ gives the decision surface.
- If two points $x_1$ and $x_2$ are on the decision surface, then

$$\mathbf{w}^t \mathbf{x}_1 + w_0 = \mathbf{w}^t \mathbf{x}_2 + w_0$$

$$\mathbf{w}^t (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

- The weight vector $\mathbf{w}$ is normal to the decision surface.
- The discriminant function $g(x)$ gives the distance of the point $x$ from the decision surface.
Figure 5.2: The linear decision boundary $H$, where $g(x) = w^t x + w_0 = 0$, separates the feature space into two half-spaces $R_1$ (where $g(x) > 0$) and $R_2$ (where $g(x) < 0$).

\[ g(x) = w^t x + w_0 = r \|w\| \]

\[ r = \frac{g(x)}{\|w\|} \]
In summary, a linear discriminant function divides the feature space by a hyperplane decision surface. The orientation of the surface is determined by the normal vector \( w \) and the location of the surface is determined by the bias. The discriminant function \( g(x) \) is proportional to the signed distance from \( x \) to the hyperplane.

\[
x = x_p + \frac{r \cdot w}{\| w \|} \quad (\text{since } w \text{ is colinear with } x - x_p \text{ and } \frac{w}{\| w \|} = 1)
\]

Since \( g(x) = 0 \) and \( w^t \cdot w = \| w \|^2 \)

Therefore \( r = \frac{g(x)}{\| w \|} \)

In particular \( d(0, H) = \frac{w_0}{\| w \|} \)
Multi-category Case

- There is more than one way to devise multi-category classifiers employing LDF (see Fig. 5.3): (i) \( w_i \) vs. rest, \( i = 1, \ldots, c \) (ii) pairwise LDF requiring \( c(c-1)/2 \) LDF. Both these approaches lead to “undefined” regions in the feature space.

- We define \( c \) linear discriminant functions

\[
g_i(x) = w_i^t x + w_{i0} \quad i = 1, \ldots, c
\]

and assign \( x \) to \( \omega_i \) if \( g_i(x) > g_j(x) \) \( \forall j \neq i \); in case of ties, the classification is undefined. This is called a linear machine.

- A linear machine divides the feature space into \( c \) decision regions, with \( g_i(x) \) being the largest discriminant if \( x \) is in the region \( R_i \).

- For two contiguous regions \( R_i \) and \( R_j \), the boundary that separates them is a portion of hyperplane \( H_{ij} \) defined by:

\[
g_i(x) = g_j(x) \iff (w_i - w_j)^t x + (w_{i0} - w_{j0}) = 0
\]
FIGURE 5.3. Linear decision boundaries for a four-class problem. The top figure shows $\omega_i/\overline{\omega_i}$ dichotomies while the bottom figure shows $\omega_i/\omega_j$ dichotomies and the corresponding decision boundaries $H_{ij}$. The pink regions have ambiguous category assignments. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.
• It is easy to show that the decision regions for a linear machine are convex
• This restriction limits the flexibility and accuracy of the classifier
Generalized Linear Discriminant Functions

• Recall the Linear Discriminant (2-category case)

\[ g(x) = w_0 + \sum_{i=1}^{d} w_i x_i \]

• \( g(x) \) positive implies class 1
• \( g(x) \) negative implies class 2

• Generalized Linear Discriminant
  • Add additional terms involving the products of features
  • For example,
    • Given: \([x_1, \; x_2, \; x_3]\)
    • Make it: \([x_1, \; x_2, \; x_3, \; x_1x_2, \; x_2x_3, \; x_1x_2x_3]\) by adding products of features.
  • Learn a discriminant function that is linear in the new feature space
Quadratic Discriminant Function

- Quadratic Discriminant Function
  - Obtained by adding pair-wise products of features

\[
g(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j
\]

- \( g(x) \) positive implies class 1; \( g(x) \) negative implies class 2
- \( g(x) = 0 \) represents a hyperquadric, as opposed to hyperplanes in linear discriminant case
- Adding more terms such as \( w_{ijk} x_i x_j x_k \) results in polynomial discriminant functions
Generalized Discriminant Function

• A *generalized linear discriminant* function can be written as

\[
g(x) = \sum_{i=1}^{\hat{d}} a_i y_i(x)
\]

- Dimensionality of the augmented feature space.
- Setting \( y_i(x) \) to be monomials results in polynomial discriminant functions.
- Weights in the augmented feature space. Note that the function is linear in \( a \).

• Equivalently,

\[
g(x) = a^t y
\]

\[
a = [a_1, a_2, \ldots, a_{\hat{d}}]^t \quad y = [y_1(x), y_2(x), \ldots, y_{\hat{d}}(x)]^t
\]

also called the augmented feature vector.
Phi Function

• The discriminant function $g(x)$ is not linear in $x$, but is linear in $y$
• The mapping $y = [y_1(x), y_2(x), ..., y_d(x)]^t$ takes a $d$-dimensional vector $x$ and maps it to a $\hat{d}$-dimensional space. The mapping $y$ is called the phi-function.
• When the input patterns $x$ are non-linearly separable in the input space, mapping them using the right phi-function maps them to a space where the patterns are linearly separable.
• Unfortunately, the curse of dimensionality makes it hard to capitalize this in practice. A complete QDF involves $(d + 1)(d+2)/2$ terms; for modest values of $d$, say $d = 50$, this requires many terms.
Quadratic Discriminant Function

Figure 5.5: The mapping $y = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $y$ space into regions corresponding to two categories, and this in turn gives a non-simply connected decision region in the one-dimensional $x$ space.
Figure 5.6: The two-dimensional input space $x$ is mapped through a polynomial function $f$ to $y$. Here the mapping is $y_1 = x_1$, $y_2 = x_2$ and $y_3 = x_1 x_2$. A linear discriminant in this transformed space is a hyperplane, which cuts the surface. Points to the positive side of the hyperplane $\hat{H}$ correspond to category $\omega_1$, and those beneath it $\omega_2$. Here, in terms of the $x$ space, $\mathcal{R}_1$ is a not simply connected.
Two-category Linearly Separable Case

- Let \( y_1, y_2, \ldots, y_n \) be a set of \( n \) examples in augmented feature space, which are linearly separable.
- We need to find a weight vector \( a \) such that
  - \( a^t y > 0 \) for examples from the positive class.
  - \( a^t y < 0 \) for examples from the negative class.
- “Normalizing” the input examples by multiplying them with their class label (replace all samples from class 2 by their negatives), find a weight vector \( a \) such that
  - \( a^t y > 0 \) for all the examples (here \( y \) is multiplied with class label)
- Such a weight vector is called a separating vector or a solution vector
Solution Region

Solution vector, if it exists in not unique. How do we constrain the solution? Find a minimum-length weight vector s.t. $a^T y > b$, where $b$ is a positive constant called margin.

Figure 5.8: Four training samples (black for $\omega_1$, red for $\omega_2$) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been “normalized” — i.e., changed in sign. Now the solution vector leads to a plane that places all “normalized” points on the same side.
The Perceptron Criterion Function

- Goal: Find a weight vector $\mathbf{a}$ such that $\mathbf{a}^\top \mathbf{y} > 0$ for all the examples (assuming it exists).
- Mathematically, this can be expressed as finding a weight vector $\mathbf{a}$ that minimizes the no. of samples misclassified.
  - Function is piecewise constant (discontinuous, and hence non-differentiable) and is difficult to optimize.
- **Perceptron Criterion Function**:

$$J_p(\mathbf{a}) = \sum_{\mathbf{y} \in \mathcal{Y}} (-\mathbf{a}^\top \mathbf{y})$$

Find an $\mathbf{a}$ that minimizes this criterion.

The criterion is proportional to the sum of distances from the misclassified samples to the decision boundary.

Now, the minimization is mathematically tractable, and hence it is a better criterion fn. than no. of misclassifications.
Different Learning Criteria

- Number of patterns misclassified.
- Squared error.
- Squared error with “margin”.

Figure 5.11: Four learning criteria as a function of weights in a linear classifier. At the upper left is the total number of patterns misclassified, which is piecewise constant and hence unacceptable for gradient descent procedures. At the upper right is the Perceptron criterion (Eq. 16), which is piecewise linear and acceptable for gradient descent. The lower left is squared error (Eq. 32), which has nice analytic properties and is useful even when the patterns are not linearly separable. The lower right is the square error with margin (Eq. 33). A designer may adjust the margin $b$ in order to force the solution vector to lie toward the middle of the $b = 0$ solution region in hopes of improving generalization of the resulting classifier.
Gradient Descent

• Perceptron criterion function $J(a)$ can be minimized using gradient descent.

• Gradient Descent Procedure:
  • Start with an arbitrarily chosen weight vector $a(1)$
  • Compute the gradient vector
  • The next value $a(2)$ is obtained by moving in the direction of the steepest descent, i.e. along the negative of the gradient.
  • In general, the $k+1$-th solution is obtained by

\[
a(k + 1) = a(k) - \eta(k) \nabla J(a(k))
\]
Gradient Descent Algorithm

Algorithm 1 (Basic gradient descent)

1. `begin` initialize a, criterion θ, η(·), k = 0
2. `do` k ← k + 1
3. a ← a − η(k) ∇J(a)
4. `until` η(k) ∇J(a) < θ
5. `return` a
6. `end`
Optimization Algorithm

• Use gradient descent to find $\mathbf{a}$
  • Move in the negative direction of the gradient iteratively to reach the minima.

• The gradient vector is given by,

$$\nabla J_p = \sum_{\mathbf{y} \in \mathcal{Y}} (-\mathbf{y})$$

• Starting from $\mathbf{a} = 0$, update $\mathbf{a}$ at each iteration $k$ as follows:

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y}$$

Directions of update
Learning rate or Step size: Magnitude of update
Figure 5.12: The Perceptron criterion, $J_p$, is plotted as a function of the weights $a_1$ and $a_2$ for a three-pattern problem. The weight vector begins at 0, and the algorithm sequentially adds to it vectors equal to the “normalized” misclassified patterns themselves. In the example shown, this sequence is $y_2, y_3, y_1, y_3$, at which time the vector lies in the solution region and iteration terminates. Note that the second update (by $y_3$) takes the candidate vector farther from the solution region than after the first update (cf. Theorem 5.1. (In an alternate, batch method, all the misclassified points are added at each iteration step leading to a smoother trajectory in weight space.)
Batch Perceptron

- Uses all the samples to compute the gradient direction.

Algorithm 3 (Batch Perceptron)

1. begin initialize $a$, $\eta(\cdot)$, criterion $\theta$, $k = 0$
2. do $k \leftarrow k + 1$
3. $a \leftarrow a + \eta(k) \sum_{y \in \mathcal{Y}_k} y$
4. until $\eta(k) \sum_{y \in \mathcal{Y}_k} y < \theta$
5. return $a$
6. end
Fixed-increment Single Sample Perceptron

• Computes the gradient using a single sample
  • Also called perceptron learning in an online setting
• For large datasets, this is much more efficient compared to batch descent (O(n) vs O(1) gradient computation in batch vs single-sample)

Algorithm 4 (Fixed-increment single-sample Perceptron)

```
begin initialize a, k = 0

do k ← (k + 1) mod n

if y_k is misclassified by a then a ← a - y_k

until all patterns properly classified

return a

end
```
Perceptron Convergence Theorem

- If training samples are linearly separable, then the sequence of weight vectors given by Algorithm 4 (Fixed-increment single-sample Perceptron) will terminate at a solution vector
Non-separable Behavior

• Perceptron algorithm modifies the solution vector $\mathbf{a}$ whenever an error is encountered
  • Converges only if an error-free solution exists, since no error is made after reaching the solution
  • Otherwise, the updates never cease, since no solution is error-free

• Any set of fewer than $2^d$ examples is linearly separable. Large training sets are usually not linearly separable

• Heuristic modifications:
  • Averaging the solutions obtained at all the steps
  • Reducing the step-size to zero as iterations progress
Minimum-Squared Error Procedures

- Perceptron criterion function focused only on errors. Mean-squared error (MSE) procedures involve all the samples.
- Using matrix notation for convenience

\[
\begin{pmatrix}
Y_{10} & Y_{11} & \cdots & Y_{1d} \\
Y_{20} & Y_{21} & \cdots & Y_{2d} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{n0} & Y_{n1} & \cdots & Y_{nd}
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_d
\end{pmatrix} =
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_n
\end{pmatrix}
\]

or

\[
Ya = b.
\]

- MSE Criterion: Minimize the sum of squared differences between Ya and b:

\[
J_s(a) = \|Ya - b\|^2 = \sum_{i=1}^{n} (a^t y_i - b_i)^2.
\]
Optimizing the MSE Criterion

• While a gradient search can be used to solve MSE, a closed form solution can be obtained in many cases.

• Computing the gradient gives:

\[ \nabla J_s = \sum_{i=1}^{n} 2(a^t y_i - b_i) y_i = 2Y^t (Ya - b) \]

• Setting the gradient to zero,

\[ Y^t Ya = Y^t b \]

• The solution for \( a \) can be obtained uniquely if \( Y^t Y \) is non-singular.

\[ a = (Y^t Y)^{-1} Y^t b = Y^\dagger b \]

Pseudo-inverse of the rectangular pattern matrix \( Y \).