8001652 Introduction to Pattern Recognition. Lectures 10 and 11: Linear discriminant functions

Jussi Tohka

jussi.tohka@tut.fi

Institute of Signal Processing

Tampere University of Technology

Introduction

- With the Bayes classifier, we assumed that the underlying probability density functions were known.
- Therefore, we estimated the parameters for the density functions using the maximum-likelihood technique.
- But we still had to draw from a hat the parametric forms, or shapes if you will, of the density functions.
- We have already touched the concept of discriminant functions. They were useful tool for representing classifiers in simpler way that could be done with posterior probabilities.

Introduction

- Discriminant functions (DF)s were derived from posterior probabilities using operations that did not change the classification result. That is, the order of posterior probabilities of c classes.
- The decision rule given DFs $g_1, \ldots, g_c$ was

  Classify $\mathbf{x}$ to the class ω_i if $g_i(\mathbf{x}) \geq g_j(\mathbf{x})$ for all $j \neq i$.
- For the minimum-error-rate Bayes classifier one could always select $g_i(\mathbf{x}) = P(\omega_i|\mathbf{x})$, but we also saw that significant simplifications were possible to make.

Introduction

- This leads us to consider another training strategy.
- Instead of specifying the forms of underlying probability densities, specify forms of the discriminant functions.
- Then, during the training we estimate parameters of the discriminant functions.
- The estimation will be formulated as a problem of minimizing a criterion function - much in same way than with ML technique.

Introduction

- The criterion function can be e.g. training error - the average classification error in classifying the training samples.
- This criterion has some drawbacks and often some numerically easier criterion functions are considered. Also, a small training error does not necessarily convert to a small test error, which is more important quantity.

Linear discriminant functions (LDF)s

- A DF are said to be linear if it can written as

(1) $$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \sum_{i=1}^{d} w_i x_i + w_0.$$

- Here $\mathbf{w}$ is *the weight vector* and w_0 is the *bias* or *threshold weight*. These are the parameters that we want to estimate based on training data.

LDFs - the two-category case

- For a LDF, a two-category classifier implements the following decision rule

  Decide ω_1 if $g(\mathbf{x}) > 0$; otherwise decide ω_2.
- If $g(\mathbf{x}) = 0$, $\mathbf{x}$ can be assigned to either category and we are on the decision boundary or on the decision surface. Decision boundaries separate the decision regions for different classes.

LDFs - the two-category case

- Let us study the decision boundary H more carefully.
- $H = \{\mathbf{x} | \mathbf{w}^T\mathbf{x} + w_0 = 0\}$ that means H is a hyperplane.
- A hyperplane is a point if $d = 1$, a line if $d = 2$ and so on.
- Moreover if $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on H, then

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

  showing that $\mathbf{w}$ is a normal of the separating hyperplane H.

LDFs - the two-category case

- The discriminant function $g(\mathbf{x})$ gives an algebraic measure of the distance from $\mathbf{x}$ to the hyperplane.
- For this, write $\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}$, where $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ to H and r is the distance measure.
- Now $r = g(\mathbf{x})/||\mathbf{w}||$. (Proof straightforward)

LDFs - the multicategory case

- The decision rule given LDFs $g_1, \ldots, g_c$ was
  Classify $\mathbf{x}$ to the class ω_i if $g_i(\mathbf{x}) \geq g_j(\mathbf{x})$ for all $j \neq i$.
- Each $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$.
- The resulting classifier is called a linear machine.

LDFs - the multicategory case

- A linear machine (as all classifiers do) divides the feature space into decision regions $\mathcal{R}_1, \ldots, \mathcal{R}_c$.
- The boundary between $\mathcal{R}_i$ and $\mathcal{R}_j$ is a portion of the hyperplane
$$H_{ij} = \{\mathbf{x} | g_i(\mathbf{x}) = g_j(\mathbf{x})\}$$
- $(\mathbf{w}_i - \mathbf{w}_j)$ is normal to H_{ij} and the signed distance from $\mathbf{x}$ to H_{ij} is $(g_i(\mathbf{x}) = g_j(\mathbf{x}))/||\mathbf{w}_i - \mathbf{w}_j||$. Hence, it is the differences of weights that are important.

LDFs - the multicategory case

- A region R is convex if for all $\mathbf{r}_1, \mathbf{r}_2 \in R$ and $\lambda \in [0, 1]$ also $(1 - \lambda)\mathbf{r}_1 + \lambda\mathbf{r}_2 \in R$.
- That means: Take any two points from a convex region R and form a line between these two points. Then all the points in the formed line are also in R.
- Decision regions for a linear machine are convex. From which it follows that they are also simply connected.
- This highlights the limitations of linear machines.

Generalized LDFs

- Quadratic DFs:

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

- Generalized LDFs:

$$g(\mathbf{x}) = \sum_{i=1}^{m} a_i y_i(\mathbf{x}) = \mathbf{a}^T \mathbf{y},$$

  where y_is can be arbitrary functions of $\mathbf{x}$.
- Generalized LDFs are not linear in $\mathbf{x}$, but are linear in $\mathbf{y}$.

Generalized LDFs

- The discrimination power of generalized LDFs is huge, they can lead to very complicated decision regions.
- However, parameters for these are hard to estimate, especially if using a training set of a modest size.
- Examples of techniques that try to circumvent this problem are multilayer neural networks and support vector machines.
- We will not consider generalized LDFs further during this course.

Augmented feature vectors

- To simplify the analysis and algorithms for LDFs, we introduce some notation.
- We want to write LDF $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = \mathbf{a}^T \mathbf{y}$.
- This can be done by introducing *augmented feature vector*

$$\mathbf{y} = [1, x_1, \ldots, x_d]^T$$

  and *augmented weight vector*

$$\mathbf{a} = [w_0, w_1, \ldots, w_d]^T = [w_0, \mathbf{w}^T]^T.$$

Augmented feature vectors

- The transformation is simple but convenient.
- It converts the problem of having to find the weight vector and threshold to the problem of having to find value for the augmented weight vector.
- Addition of constant term does not change anything from the classification point of view. In particular, it preserves the relationships among the samples.

Gradients, Jacobians, Hessians

- Before proceeding, we need some further terminology.
- The gradient of $f : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \\ \vdots \\ \frac{df}{dx_d} \end{bmatrix}.$$

Gradients, Jacobians, Hessians

- The Jacobian (matrix) of $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^m$ is defined as

$$\mathbf{B}(\mathbf{x}) = \begin{bmatrix} \frac{dh_1}{dx_1} & \cdots & \frac{dh_1}{dx_d} \\ \vdots & \ddots & \vdots \\ \frac{dh_m}{dx_1} & \cdots & \frac{dh_m}{dx_d} \end{bmatrix}.$$

Gradients, Jacobians, Hessians

- The Hessian (matrix) of $f : \mathbb{R}^d \to \mathbb{R}$ is defined as the Jacobian of ∇f:

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{df_1}{dx_1 dx_1} & \cdots & \frac{df_1}{dx_1 dx_d} \\ \vdots & \ddots & \vdots \\ \frac{df_1}{dx_d dx_1} & \cdots & \frac{df_d}{dx_d dx_d} \end{bmatrix}.$$

Linearly separable case

- Consider a two-category classification problem.
- Suppose that we have a set of n samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$ some labeled ω_1 and some labeled ω_2.
- Note that all samples are augmented feature vectors.
- We want use these samples to determine the weights $\mathbf{a}$ in a linear discriminant function $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$.

Linearly separable case

- Suppose that there exists a solution for which the probability of error is very low.
- Then it is reasonable to try to find such $\mathbf{a}$ that all the training samples are classified correctly.
- i.e. $\mathbf{a}^T\mathbf{y}_i > 0$ if $\mathbf{y}_i$ is labeled ω_1 and $\mathbf{a}^T\mathbf{y}_i < 0$ if $\mathbf{y}_i$ is labeled ω_2.
- This suggests normalization: replace all feature vectors labeled ω_2 by their negatives.
- After normalization we can forget the labels, and look for a weight vector $\mathbf{a}$ such that $\mathbf{a}^T\mathbf{y}_i > 0$.

Linearly separable case

- A weight vector $\mathbf{a}$ such that $\mathbf{a}^T\mathbf{y}_i > 0$ all i is called *solution vector*.
- A solution vector - if exists - is not unique. The set of possible solution vectors, that are interpreted as points in $\mathbb{R}^d$, is called the solution region.
- To constrain solution vectors, we introduce *margin*, which is a positive constant b.
- That is we seek for the minimum length vector satisfying

$$\mathbf{a}^T\mathbf{y}_i > b, \forall i = 1, \ldots, n.$$

Solving inequalities

- To find a solution to the set of linear inequalities

$$\mathbf{a}^T\mathbf{y}_i > b,$$

  we define a criterion function $J(\mathbf{a})$ that is minimized if $\mathbf{a}$ is a solution.
- This kind of problem can be solved by gradient descent. The idea is very simple: Start with some vector $\mathbf{a}(1)$. Generate then $\mathbf{a}(2)$ by taking a small step in the direction of $-\nabla J(\mathbf{a}(1))$ and so on.
- Explanation: $-\nabla J(\mathbf{a}(k))$ is the direction of the steepest descent.

Basic gradient descent

1. Initialize: $\mathbf{a}(1)$, threshold θ, learning rate $\eta(k)$, and set $k \leftarrow 0$.
2. $k \leftarrow k + 1$
3. $\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k(1))$
4. If $|\eta(k)\nabla J(\mathbf{a}(k))| < \theta$ stop and return $\mathbf{a}(k)$, otherwise go to step 2.

The learning rate

- The learning rate can be set

$$\eta(k) = \frac{||\nabla J(\mathbf{a}(k))||^2}{\nabla J(\mathbf{a}(k))^T \mathbf{H} \nabla J(\mathbf{a}(k))},$$

where $\mathbf{H}$ is the Hessian at $\mathbf{a}(k)$.

The perceptron criterion function

- Consider now the problem of constructing a criterion function for solving the linear inequalities. Assume that the margin $b = 0$.
- The most obvious choice would be the number of samples misclassified by $\mathbf{a}$. However, this criterion is a piecewise constant function and poor candidate for a gradient search.
- The perceptron criterion function is defined by

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

where $\mathcal{Y}(\mathbf{a})$ is the of samples misclassified by $\mathbf{a}$, i.e. samples for which the inner product with $\mathbf{a}$ is negative.

The perceptron criterion function

- The gradient

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

- The update rule in gradient descent is

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

The perceptron algorithm

1. Initialize: $\mathbf{a}(1)$, threshold θ, learning rate $\eta(k)$, and set $k \leftarrow 0$.
2. $k \leftarrow k + 1$
3. $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{a}(k))} \mathbf{y}$.
4. If $|\eta(k) \nabla J(\mathbf{a}(k))| < \theta$ stop and return $\mathbf{a}(k)$, otherwise go to step 2.

The perceptron algorithm

- There are many modifications of this 'batch perceptron' algorithm in the course book. However, we will not deal them during these lectures.
- Bad feature of the perceptron algorithm is that it does not converge if the training samples are not linearly separable.
- If the training samples are linearly separable, the perceptron algorithm converges to a solution in a finite number of iterations.

Other criterion functions

- Relaxation:

$$J_r(\mathbf{a}) = 0.5 \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{a})} \frac{(\mathbf{a}^T\mathbf{y} - b)^2}{||\mathbf{y}||^2}$$

  , where b is the margin and the sum is over such samples that $\mathbf{a}^T\mathbf{y} < b$.
- Mean squared error MSE:

$$J_s(\mathbf{a}) = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b)^2$$

  .

Multicategory generalizations

- We have studied this far only procedures for finding the weight vector $\mathbf{a}$ for two-category problems. We now look how to generalize these procedures for multicategory tasks.
- Again we have n training samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$. A subset $\mathcal{Y}_1$ of these is labeled ω_1, $\mathcal{Y}_2$ is labeled ω_2.
- This set is said to be linearly separable if there exists a linear machine that classifies all these samples correctly. Then there exists a set of weight vectors $\mathbf{a}_1, \ldots, \mathbf{a}_c$ such that if $\mathbf{y}_k \in \mathcal{Y}_i$, then for all $i \neq j$

(2) $$\mathbf{a}_i^T\mathbf{y}_k > \mathbf{a}_j^T\mathbf{y}_k.$$

Kesler's construction

- The Eqs. (2) can be manipulated in such a way that the multicategory problem reduces to a two-categorory one. This manipulation is called Kesler's construction.
- This involves multiplication of dimensionality of the data by c and the number of samples by $c - 1$. Hence, it is not computationally attractive, but has some theoretical meaning.