Motivation
Linear Discriminants
Multi-class classification using linear discriminants
Learning discriminants
Perceptron approach
Minimum squared error approach

SYDE 372 - Winter 2011
Introduction to Pattern Recognition

Discriminant Functions: Part I

Alexander Wong

Department of Systems Design Engineering
University of Waterloo
Outline

1. Motivation
2. Linear Discriminants
3. Multi-class classification using linear discriminants
4. Learning discriminants
5. Perceptron approach
6. Minimum squared error approach
Motivation

So far, the approach to the labeled sample problem is:

- Use the given samples to obtain a class description consisting of either a distance metric or probability density function
- Derive decision rule from description (e.g., MICD and MAP rules)

The decision rule in turn specifies a decision boundary in feature space.

For example, MICD rule and MAP rule have decision surfaces of the form:

\[ g(x) = x^T Q_0 x Q_1 x + Q_2 = 0 \]  

(1)
The function $g(x)$ is a discriminant function.

The two-class decision rule can be written as:

$$A \quad g(x) > 0 \quad (2)$$

$$B \quad g(x) < 0$$

A positive value for $g(x)$ means that the pattern $x$ belongs to class A, while a negative value for $g(x)$ means that the pattern $x$ belongs to class B.
Motivation

Idea: What if we take an alternative approach?

- Assume a particular form for the discriminant functions (e.g., hyperplane)
- Use the given samples to directly estimate the parameters of the discriminant functions
- Given discriminant functions, decision rules and decision surfaces are defined

What we basically want to do is learn the discriminant functions directly from the samples.
A linear discriminant function can be expressed as:

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

where \( \mathbf{w} \) is the weight vector and \( w_0 \) is a threshold.

If we set \( g(x) \), we have the equation for a hyperplane, with the decision surface defined for a linear classifier.
A more explicit way of expressing $g(x)$ to emphasize its linear nature is:

$$g(x) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_0$$  \hspace{1cm} (4)

$$g(x) = \sum_{i=1}^{n} w_i x_i + w_0$$  \hspace{1cm} (5)

e.g., For a two-class case $x = (x_1, x_2)$, the linear discriminant $g(x_1, x_2)$ can be written as:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$  \hspace{1cm} (6)

Just a straight line equation!
Consider the two class problem with discriminant
\[ g(x) = w^T x + w_0. \]

The decision rule can be defined as:
- \( x \in c_1 \) if \( g(x) > 0 \)
- \( x \in c_2 \) if \( g(x) < 0 \)

The decision surface, defined by \( g(x) = 0 \), is a hyperplane with the following properties:
- The unit normal vector is \( \frac{w}{|w|} \), since for any two vectors \( x_1 \) and \( x_2 \):
  \[
  g(x_1) = g(x_2) = 0 \tag{7}
  \]
  \[
  w^T x_1 + w_0 = w^T x_2 + w_0 \tag{8}
  \]
  \[
  w^T (x_1 - x_2) = 0 \tag{9}
  \]

This shows that \( \frac{w}{|w|} \) is normal to any vector lying in the plane so that \( \frac{w}{|w|} \) is the unit normal.
The decision surface, defined by $g(x) = 0$, is a hyperplane with the following properties:

- The distance between any $x$ and the hyperplane is $\left| \frac{g(x)}{w} \right|$.
- When $g(x) > 0$, $x$ is said to lie on the positive side of the plane, the side which $w$ points to.
- When $g(x) < 0$, $x$ is said to lie on the negative side of the plane.
Linear Discriminants: Visualization

$g(x) = 0$

$w_0/|w|$

$|g(x)/|w||$
Motivation

Linear Discriminants
Multi-class classification using linear discriminants
Learning discriminants
Perceptron approach
Minimum squared error approach

Linear Discriminants: Example

Suppose that we are given two classes that are linearly separable with the following discriminant function:

\[ g(x) = 4x_1 + 3x_2 - 5 \]  

and the following decision rule:

- \( x \in c_1 \) if \( g(x) > 0 \)
- \( x \in c_2 \) if \( g(x) < 0 \)

For the unit normal vector of the decision boundary and its distance from the origin. Plot the boundary indicating \( w \) and \( w_0 \).

Classify the following patterns:

- \( x_1 = (1, 3) \), \( x_2 = (2, -1) \), \( x_3 = (1, -3) \)
Linear Discriminants: Example

- If we were to rewrite $g(x)$ in vector form, we end up with:

$$g(x) = [4 \ 3] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 5 > 0$$

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

- Therefore, given this vector form, we know that:

$$w = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$
Therefore, the unit normal vector can be computed as:

\[
\frac{\mathbf{w}}{|\mathbf{w}|} = \left[ \begin{array}{c} 4 \\ 3 \end{array} \right] / (\sqrt{4^2 + 3^2}) \quad (14)
\]

\[
\frac{\mathbf{w}}{|\mathbf{w}|} = \left[ \begin{array}{c} 4 \\ 3 \end{array} \right] / (5) \quad (15)
\]

\[
\frac{\mathbf{w}}{|\mathbf{w}|} = \left[ \begin{array}{c} 4/5 \\ 3/5 \end{array} \right] \quad (16)
\]
Linear Discriminants: Example

The distance from the origin to the plane is given by:

\[ d = \frac{g(x)}{|w|} = \frac{g(0, 0)}{\sqrt{4^2 + 3^2}} \]  \hspace{1cm} (17)

\[ d = \frac{4(0) + 3(0) - 5}{\sqrt{4^2 + 3^2}} \]  \hspace{1cm} (18)

\[ d = \frac{-5}{5} \]  \hspace{1cm} (19)

\[ d = 1 \]  \hspace{1cm} (20)
Motivation
Linear Discriminants
Multi-class classification using linear discriminants
Learning discriminants
Perceptron approach
Minimum squared error approach

Linear Discriminants: Example

\[ 4x_1 + 3x_2 - 5 = 0 \]

Region 1
Region 2

\[ w_0 / |w| \]

\[ \frac{5}{3} \]

\[ \frac{5}{4} \]
Linear Discriminants: Example

To classify patterns, we plug our patterns into the decision rule:

- For $x_1 = (1, 3)$, $g(1, 3) = 4(1) + 3(3) - 5 = 13 - 5 = 8 > 0$, so class=$c_1$
- For $x_1 = (2, -1)$,
  $g(2, -1) = 4(2) + 3(-1) - 5 = 5 - 5 = 0 = 0$, so class=$c_1$ or $c_2$
- For $x_1 = (91, -3)$,
  $g(91, -3) = 4(91) + 3(-3) - 5 = 355 - 5 = 350 > 0$, so class=$c_1$
Multi-class classification using linear discriminants

- So far, we’ve only talked about defining decision regions for the two class problem
- How do we handle the situation where we have multiple classes \((k > 2)\)?
- Solution: Use multiple linear discriminants to separate the different classes!
- Three possible strategies:
  - Strategy 1: A linear discriminant can be found for each class which separates it from all other classes:
    - \(g_i(x) > 0\) if \(x \in c_i, \ i = 1, \ldots, k\)
    - \(g_i(x) < 0\) if otherwise
Multi-class classification using linear discriminants: Strategy 1

Quite a few indeterminant areas.
Multi-class classification using linear discriminants

- Three possible strategies:
  - Strategy 2: A linear discriminant can be found for every pair of classes (i.e., classes are pairwise separable):
    - $g_{ij}(x) > 0$ for all $j \neq i$ if $x \in c_i$
Three possible strategies:

- **Strategy 3**: Each class has its own discriminant function:
  \[ g_i(x) > g_j(x) \text{ for all } j \neq i \text{ if } x \in c_i \]
Multi-class classification using linear discriminants

- Of the three strategies, only the last strategy avoids producing indeterminant regions.
- Therefore, adopting this strategy, the general $k$ class linear discriminant classifier can be defined as:

\[ x \in c_i \text{ iff } g_i(x) > g_j(x) \text{ for all } j \neq i \quad (21) \]

with \( g_i(x) = w_i^T x + w_{i0}, \) \( i = 1, \ldots, k \)

- Based on this classifier, the decision boundary between \( c_i \) and \( c_j \) is given by \( g_i(x) = g_j(x) \):

\[ g_i(x) - g_j(x) = (w_i - w_j)^T x + w_{i0} - w_{j0} = 0 \quad (22) \]

- This is a hyperplane with normal vector \((w_i - w_j)\)!
Learning discriminants

- The question now is: how do we build such a $k$ class linear discriminant classifier?
- Suppose that we are given a set of labeled samples for each of the classes which are assumed to be **linearly separable** in an appropriate feature space.
- The goal is to learning appropriate discriminant functions $g_i(x)$ directly from the labeled samples.
- Focusing on the two-class problem, the problem of learning the discriminant is to find a weight vector $a$ such that:
  - $g(x) = a^T y > 0$ when $y$ (and $x$) is a member of class $c_1$
  - $g(x) = a^T y < 0$ when $y$ (and $x$) is a member of class $c_2$
Learning discriminants

- If the classes are linearly separable in the original feature space \((x)\), we have: 
  \[
  y = \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n \\
  1
  \end{bmatrix}
  \quad \text{and} \quad
  a = \begin{bmatrix}
  w_1 \\
  \vdots \\
  w_n \\
  w_0
  \end{bmatrix}
  \]

- In \(y\) space, the decision surface is a hyperplane which contains the original and has normal vector \(\frac{a}{|a|}\).

- Given labeled samples \(\{y_1, y_2, \ldots, y_N\}\), the goal is to find \(a\) (the solution vector) such that:
  - \(a^T y_i > 0\) for all \(y_i \in c_1\)
  - \(a^T y_i < 0\) for all \(y_i \in c_2\)
One way to simplify the problem a bit is to perform normalization by replacing all $y_i$ by $-y_i$ for all $y_i \in c_2$.

By doing so, we can change our goal to finding $a^T y_i > 0$ for all $i$! The solution remains the same!
Learning discriminants

Before normalization

After normalization
So how do we find a solution vector $a$ that satisfies our classification criteria?

Trial and error and exhaustive search strategies are impractical for the general $N$ sample $n$ dimensional problem.

A much more efficient strategy is to use iterative methods that use a criterion function which is minimized when $a$ is the solution vector.
Learning discriminants

Here, we will use gradient descent optimization strategies to find $\mathbf{a}$:

Let $J(\mathbf{a})$ be the criterion function.

The weight vector at $k + 1$ ($\mathbf{a}_{k+1}$) is computed based on the weight vector at $k$ ($\mathbf{a}_k$) and the gradient of the criterion function ($\nabla J(\mathbf{a})$).

Since $\nabla J(\mathbf{a})$ indicates the direction of maximum change, we wish to move in the opposite direction, which is the direction of steepest descent:

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \rho_k \nabla J(\mathbf{a})$$  \hspace{1cm} (23)

where $\rho_k$ is the step size (which dictates the rate of convergence).
Here, we will discuss two types of gradient descent approaches for learning discriminants:

- **Perceptron approach**: guide convergence based on sum of distances of misclassified samples to decision boundary
- **Minimum Squared Error approach**: guide convergence based on sum of squared error

Each comes in different varieties!

- Non-sequential: update based on all samples at the same time
- Sequential: update based on one sample at a time
The perceptron criterion may be interpreted as the sum of distances of the misclassified samples from the decision boundary.

\[
J_p(a) = \sum_{y \in Y(a)} (-a^T y) \tag{24}
\]

where \( Y \) is the set of misclassified samples due to \( a \):

\[
Y(a) = (y_i \text{ such that } a^T y_i \leq 0) \tag{25}
\]

\( a \) is the solution vector when \( J_p(a) = 0 \).
Perceptron approach

- The gradient of $J_p(a)$ can be written as:

$$\nabla J_p(a) = \sum_{y \in Y(a)} (-y)$$  \hspace{1cm} (26)

- This gives us the weight update formula as:

$$a_{k+1} = a_k + \rho_k \nabla J_p(a)$$  \hspace{1cm} (27)

$$a_{k+1} = a_k - \rho_k \sum_{y \in Y(a)} (-y)$$  \hspace{1cm} (28)

$$a_{k+1} = a_k + \rho_k \sum_{y \in Y(a)} y$$  \hspace{1cm} (29)
Motivation
Linear Discriminants
Multi-class classification using linear discriminants
Learning discriminants
Perceptron approach
Minimum squared error approach

Perceptron approach

- **Step 1:** Set an initial guess for the weight vector \( \mathbf{a}_0 \) and let \( k = 0 \)
- **Step 2:** Based on \( \mathbf{a}_k \), construct the classifier and determine the set of misclassified samples \( Y(\mathbf{a}) \). If there are no misclassified samples, stop here since we have arrived at the solution. Otherwise, continue to Step 3.
- **Step 3:** Compute a scalar multiple of the sum of misclassified samples \( \rho_k \sum_{y \in Y(\mathbf{a})} (y) \)
- **Step 4:** Determine \( \mathbf{a}_{k+1} \) as

  \[
  \mathbf{a}_{k+1} = \mathbf{a}_k + \rho_k \sum_{y \in Y(\mathbf{a})} (y) \quad (30)
  \]

- **Step 5:** Go to Step 2.
Variations on the Perceptron approach

- Fixed-increment: $\rho_k = 1$, constant step size.
- Variable-increment: $\rho_k \propto 1/k$, decreases as number of iterations increases to avoid over-shooting solution.
- Single sample correction: Treat samples sequentially, change weight vector with each misclassification.
Motivation
Linear Discriminants
Multi-class classification using linear discriminants
Learning discriminants
Perceptron approach
Minimum squared error approach

Sequential Perceptron approach

- **Step 1:** Set an initial guess for the weight vector \( \mathbf{a}_0 \) and let \( k = 0 \).

- **Step 2:** Based on \( \mathbf{a}_k \), construct the classifier and determine the set of misclassified samples \( Y(\mathbf{a}) \). If there are no misclassified samples, stop here since we have arrived at the solution. Otherwise, continue to Step 3.

- **Step 3:** Compute a scalar multiple of the \( k^{th} \) misclassified sample \( \rho_k \mathbf{y}^k \).

- **Step 4:** Determine \( \mathbf{a}_{k+1} \) as

\[
\mathbf{a}_{k+1} = \mathbf{a}_k + \rho_k \mathbf{y}^k
\]  

(31)

- **Step 5:** Go to Step 2.
Suppose that we are given the following data: $y_1 = (4, -1)$, $y_2 = (2, 1)$ belong to class $c_1$, and $y_3 = (5/2, -5/2)$ belong to class $c_2$.

Let the initial guess be $a_0 = (0, 0)$ and $\rho_k = 1$ for all $k$.

Use the standard perceptron approach to learn a solution vector $a$.

Use the sequential perceptron approach to learn a solution vector $a$. 

Alexander Wong  SYDE 372 - Winter 2011
Perceptron approach: Example

- **Step 1:** To simplify the problem, normalize by replacing $y_i$ by $-y_i$ for all $y_i \in c_2$
  - Therefore, in this case, $y_3 = (-5/2, 5/2)$.

- **Step 2:** Based on $a_0$, construct the classifier and determine the set of misclassified samples $Y(a)$.
  - $a_0^T y_1 = a_0^T y_2 = a_0^T y_3 = 0$
  - Since we arrive at our solution when $a_i^T y_i > 0$ for all $i$, all three cases fail!
  - Therefore, set of misclassified samples are $y_0, y_2, y_3$
Perceptron approach: Example

- **Step 3:** Determine $\underline{a}_1$ as
  \[
  \underline{a}_1 = \underline{a}_0 + \sum_{i=1}^{3} (y_i) = \left(\frac{7}{2}, \frac{5}{2}\right)
  \]

- Let’s go back to step 2!

- **Step 2:**
  - $\underline{a}_1^T \underline{y}_1 = \left[\frac{7}{2}, \frac{5}{2}\right] [4 \ - 1]^T = 23/2 > 0$ (correct)
  - $\underline{a}_1^T \underline{y}_2 = \left[\frac{7}{2}, \frac{5}{2}\right] [2 \ 1]^T = 19/2 > 0$ (correct)
  - $\underline{a}_1^T \underline{y}_3 = \left[\frac{7}{2}, \frac{5}{2}\right] [-5/2 \ 5/2]^T = -5/2 < 0$ (misclassified)
  - Therefore, set of misclassified samples is $\underline{y}_3$
Perceptron approach: Example

- **Step 3:** Determine $a_2$ as

$$a_2 = a_1 + (y_3)$$

$$a_2 = [7/2 \ 5/2] + [-5/2 \ 5/2] = [1 \ 5]$$

- Let’s go back to step 2!

- **Step 2:**
  - $a^T_2 y_1 = [1 \ 5][4 \ -1]^T = -1 < 0$ (misclassified)
  - $a^T_2 y_2 = [1 \ 5][2 \ 1]^T = 7 > 0$ (correct)
  - $a^T_2 y_3 = [1 \ 5][-5/2 \ 5/2]^T = 10 > 0$ (correct)
  - Therefore, set of misclassified samples is $y_1$
**Perceptron approach: Example**

- **Step 3:** Determine $\underline{a}_3$ as

$$
\underline{a}_3 = \underline{a}_2 + (\underline{y}_1)
$$

$$
\underline{a}_3 = \begin{bmatrix} 1 & 5 \end{bmatrix} + \begin{bmatrix} 4 & -1 \end{bmatrix} = \begin{bmatrix} 5 & 4 \end{bmatrix}
$$

- Let’s go back to step 2!

- **Step 2:**
  - $\underline{a}_3^T \underline{y}_1 = [5 \ 4][4 \ -1]^T = 16 > 0$ (correct)
  - $\underline{a}_3^T \underline{y}_2 = [5 \ 4][2 \ 1]^T = 14 > 0$ (correct)
  - $\underline{a}_3^T \underline{y}_3 = [5 \ 4][-5/2 \ 5/2]^T = -5/2 < 0$ (misclassified)
  - Therefore, set of misclassified samples is $\underline{y}_3$
Motivation
Linear Discriminants
Multi-class classification using linear discriminants
Learning discriminants
Perceptron approach
Minimum squared error approach

Perceptron approach: Example

**Step 3:** Determine \( \mathbf{a}_4 \) as

\[
\mathbf{a}_4 = \mathbf{a}_3 + (\mathbf{y}_3)
\]

\[
\mathbf{a}_4 = \begin{bmatrix} 5 & 4 \end{bmatrix} + \begin{bmatrix} -5/2 & 5/2 \end{bmatrix} = \begin{bmatrix} 5/2 & 13/2 \end{bmatrix}
\]

Let’s go back to step 2!

**Step 2:**

\[
\mathbf{a}_4^T \mathbf{y}_1 = [5/2 \ 13/2][4 \ -1]^T = 7/2 > 0 \text{ (correct)}
\]

\[
\mathbf{a}_4^T \mathbf{y}_2 = [5/2 \ 13/2][2 \ 1]^T = 23/2 > 0 \text{ (correct)}
\]

\[
\mathbf{a}_4^T \mathbf{y}_3 = [5/2 \ 13/2][-5/2 \ 5/2]^T = 10 > 0 \text{ (correct)}
\]

Therefore, there are no misclassified samples and we stop!
Perceptron approach: Example

- Let’s use the sequential perceptron approach!
- Determine $a_1$ as
  \[
  a_1 = a_0 + y_1 = (4, -1)
  \] (39)
- Check if $y_2$ is correctly classified
  \[
  a_1^T y_2 = [4 - 1][2 1]^T = 7 > 0 \quad \text{(correct)}
  \]
- Check if $y_3$ is correctly classified
  \[
  a_1^T y_3 = [4 - 1][-5/2 5/2]^T = -25/2 < 0 \quad \text{(misclassified)}
  \]
- Since $y_3$ is misclassified, we compute $a_2$ as
  \[
  a_2 = a_1 + y_3 = (3/2, 3/2)
  \] (40)
Perceptron approach: Example

- Check if $y_1$ is correctly classified
  - $a_2^T y_1 = [3/2 \ 3/2][4 \ -1]^T = 15/2 > 0$ (correct)
- Check if $y_2$ is correctly classified
  - $a_2^T y_2 = [3/2 \ 3/2][2 \ 1]^T = 9/2 > 0$ (correct)
- Check if $y_3$ is correctly classified
  - $a_2^T y_3 = [3/2 \ 3/2][-5/2 \ 5/2]^T = 0$ (misclassified)
- Since $y_3$ is misclassified, we compute $a_3$ as
  \[
  a_3 = a_2 + y_3 = (-1, 4)
  \] (41)
Perceptron approach: Example

- Check if $y_1$ is correctly classified
  - $a_3^Ty_1 = [-1 \ 4][4 \ -1]^T = 0$ (misclassified)
- Since $y_1$ is misclassified, we compute $a_4$ as
  \[
  a_4 = a_3 + y_1 = (3, 3)
  \] (42)

- Repeat until solution is reached!
- This sequential form can be viewed as reinforcement learning for machine learning.
- By combining perceptron classifiers until multi-layered networks, what we end up with are what we commonly refer to as neural networks!
One issue with the perceptron approach is that if the classes are not linearly separable, the learning procedure will never stop since there will always be misclassified samples!

One way around this is to terminate after a fixed number of iterations, but the resulting weight vector may or may not be appropriate for classification.

Solution: What if we use a different criterion that will converge even if there are misclassified samples?

The minimum squared error criterion provides a good compromise in performance for both separable and non-separable problems.
Minimum squared error approach

- Instead of solving a set of inequalities:
  \[ a^T y_i > 0, \ i = 1, \ldots, N \]  
  \hspace{1cm} (43)

- we can obtain a solution vector for a set of equations:
  \[ a^T y_i = b_i, \ i = 1, \ldots, N \]  
  \hspace{1cm} (44)

- Let the error vector \( e \) be defined as:
  \[
  e = \begin{bmatrix}
  y_1^T \\
  \vdots \\
  y_i^T \\
  \vdots \\
  y_N^T
  \end{bmatrix} \begin{bmatrix}
  a_1 \\
  \vdots \\
  a_i \\
  \vdots \\
  a_n
  \end{bmatrix} - \begin{bmatrix}
  b_1 \\
  \vdots \\
  b_i \\
  \vdots \\
  b_N
  \end{bmatrix} = Y a - b
  \]  
  \hspace{1cm} (45)
Instead of finding a solution \( \mathbf{a} \) that gives no misclassifications, which could be impossible if it is not a linearly separable problem, we want to find a solution \( \mathbf{a} \) that minimizes \( |e|^2 \).

This gives us the following sum of squared error criterion function:

\[
\nabla J_s(\mathbf{a}) = |\mathbf{e}|^2 = |Y\mathbf{a} - \mathbf{b}|^2 = \sum_{i=1}^{N} (a^T y_i - b_i)^2
\]

(46)
The gradient of $J_s(a)$ can be written as:

$$\nabla J_s(a) = Y^T(Ya_k - b)$$  \hspace{1cm} (47)

This gives us the weight update formula as:

$$a_{k+1} = a_k + \rho_k \nabla J_p(a)$$  \hspace{1cm} (48)

$$a_{k+1} = a_k - \rho_k Y^T(Ya_k - b)$$  \hspace{1cm} (49)
**Minimum squared error approach**

- **Step 1:** Set an initial guess for the weight vector \((a_0)\) and let \(k = 0\)

- **Step 2:** Determine \(a_{k+1}\) as

\[
a_{k+1} = a_k - \rho_k Y^T(Ya_k - b) \tag{50}
\]

- **Step 3:** If convergence reached, stop. Otherwise, go to Step 2.
Sequential variant of minimum squared error approach:

- **Step 1:** Set an initial guess for the weight vector \( \mathbf{a}_0 \) and let \( k = 0 \)
- **Step 2:** Determine \( \mathbf{a}_{k+1} \) as
  \[
  \mathbf{a}_{k+1} = \mathbf{a}_k - \rho_k (\mathbf{b}_k - \mathbf{a}_k^T \mathbf{y}_k^k) \mathbf{y}_k^k
  \]  
  (51)
- **Step 3:** If convergence reached, stop. Otherwise, go to Step 2.
How do we set up parameters (i.e., $\rho_k$, $b$) for the MSE approach?

Typically, $\rho_k$ decreases with $k$ (e.g., $\rho_k/k$) to obtain convergence.

In terms of $b$, useful settings include:

- Setting $b$ as a vector of ones.
- Setting the first $N_1$ of the $N$ components to $N/N_1$ and the rest to $N/N_2$, where $N_1$ and $N_2$ are the number of samples in each class (e.g., if there are 10 samples in class 1 and 3 samples in class 2, then the first 10 components of $b$ are set to 13/10 and the rest are set to 13/3.)