Lecture Notes: Linear Classification with Discriminant Functions

Ted Meeds[1,2]

[1] Informatics Institute, University of Amsterdam
[2] The Centre for Integrative Bioinformatics, Vrije University
tmeeds@gmail.com

Abstract In this note we introduce linear classification using linear discriminant functions (analogous to learning regression models directly). This will be followed by taking a probabilistic view (logistic regression and generative classifiers).

1 Introduction to Linear Classification

The main difference from linear regression and linear classification is the type of target variable used for prediction. For regression, targets t_n were real-valued; for classification we have discrete targets corresponding to labels that have been associated with input vectors. For example, an image $\mathbf{x}$ is associated with a discrete digit labeled 0 through 9. The goal of classification is to **discriminate** test input vectors; that is, assign them a label from the discrete set. In some ways this makes the problem easier because mapping from input to output is simply a splitting of the data in input space. However, since the targets are no longer continuous, we cannot use the nice properties of Gaussians from regression, so although probabilistic classifiers are manageable, Bayesian classifiers are difficult, requiring approximations and are beyond the scope of the course.

1.1 Goal of classification

Given an input vector $\mathbf{x}$, assign it to **one** of K discrete classes (or labels). This is different from learning a map to multiple classes *simultaneously*. Each region of the input space maps to a class; these regions are called **decision regions**.

1.2 Data representation

As with linear regression, data come in pairs $\{t_n, \mathbf{x}_n\}_{n=1}^N$, $t_n \in \{0, 1\}$ for $K = 2$ classes, or $\{\mathbf{t}_n, \mathbf{x}_n\}_{n=1}^N$, where $\mathbf{t}_n = \begin{bmatrix} t_{n1} \\ t_{n2} \\ \vdots \\ t_{nK} \end{bmatrix}$ $\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}$. Note $\sum_k t_{nk} = 1$ and $t_{nk} \in \{0, 1\}$, i.e. only one class per data item. If $t_{nk} = 1$ then data item n has label k.

e.g. $\mathbf{t}_n = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, then $n \in \mathcal{C}_2$ (i.e. input n belongs to class 2). The class variable $\mathcal{C}_k$ can represent a set of indices, or just the value of class k. E.g. we might have a distribution over classes $p(\mathcal{C})$ of which $p(\mathcal{C}_k)$ is one value. Alternatively, for individual data vectors, we might write $c_n = k$ to mean the nth class label is k, i.e. $n \in \mathcal{C}_k$ (so k is the index of $\mathbf{t}_n$ that has value 1). These are choices made for convenience when working with the data, for instance using sets of indices from the training set that have the same labels.

Just like with regression, "linear" classification is linear in parameters $\mathbf{w}$. This corresponds to linear separating hyperplanes in $\boldsymbol{\phi}$-space, but if we use non-linear $\boldsymbol{\phi}$ then in input-space the decision boundaries will be non-linear.

1.3 Learning Discriminants

We will discuss three approaches to linear classification, analogous to approaches to linear regression:

1. Directly learn discriminative function by minimizing an objective (regression: learn $y(x, \mathbf{w})$ directly by minimizing sum-of-squared error).
2. Conditional probability models: i.e. learn $p(c_k|\mathbf{x})$ then assign $c = \arg\max_{c_k} p(c_k|\mathbf{x})$ (regression: learn $p(t|\mathbf{x}, \mathbf{w})$ assuming a Gaussian model and by maximizing log-likelihood).
3. Fully probabilistic model: i.e. $p(c, \mathbf{x})$. We did not do this for regression.
4. Bayesian linear classification. We will not cover this, but we could do this analytically for linear regression.

2 Discriminant Functions

A **discriminant** function maps an input vector to a class label. A **linear discriminant** uses hyperplanes as decision surfaces, that is linear mapping of inputs/features. Let $y(\mathbf{x})$ be a linear discriminant function:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

then we assign $\mathbf{x}$ to $\mathcal{C}_1$ is $y(\mathbf{x}) \geq 0$ and to $\mathcal{C}_0$ otherwise. The bias w_0 can also be though of the decision threshold (the value 0 is chosen arbitrarily, but is convenient).

The decision boundary is a $(D-1)$-dimensional hyperplane within the D-dimensional input space. Consider 2 points $\mathbf{x}_A$ and $\mathbf{x}_B$ along the decision boundary (i.e. $y(\mathbf{x}) = 0$):

$$\begin{aligned} y(\mathbf{x}_A) - y(\mathbf{x}_B) &= (\mathbf{w}^T\mathbf{x}_A + w_0) - (\mathbf{w}^T\mathbf{x}_B + w_0) \\ 0 &= \mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) \end{aligned}$$

therefore $\mathbf{w}$ is **orthogonal** to every vector along the decision surface. I.e. $\mathbf{w}$ is perpendicular to the decision surface.

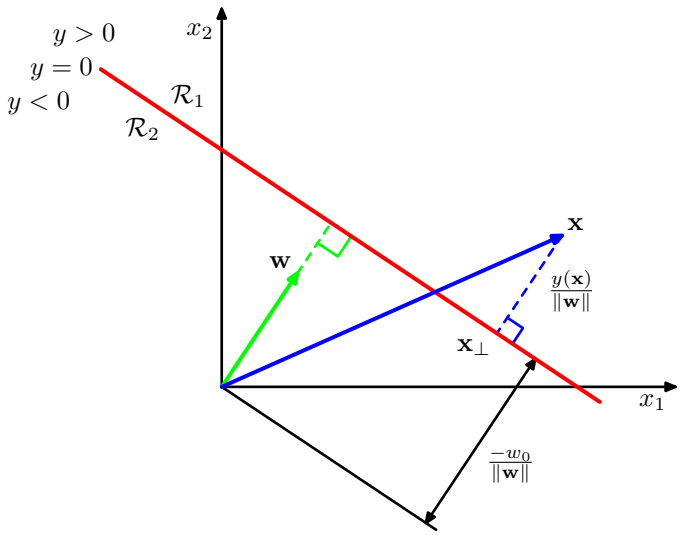


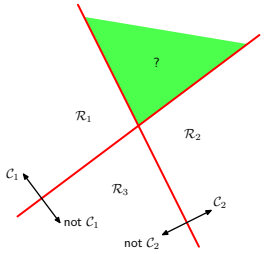
Figure 1. Geometry of linear discriminant (Bishop Figure 4.1).

3 Multiple classes

Once the number of classes goes beyond, special care must be taken to avoid ambiguous decisions. The solution is to treat the K classes with K linear functions.

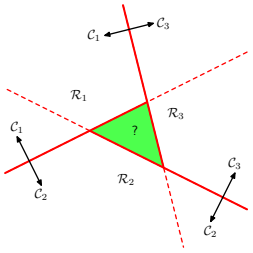
3.1 One-versus-rest

Build (K-1) classifiers of class k versus all the rest. Leads to ambiguous regions.



3.2 One-versus-one

Build K(K-1)/2 of binary classifiers. Leads to ambiguous regions.



3.3 K-class discriminant

Avoid the ambiguous regions by computing $y_k = \mathbf{w}_k^T\mathbf{x} + w_{k0}$, then assigning $\mathcal{C} = \arg\max_k y_k(\mathbf{x})$. Analogous geometry to two-class boundary. It can be shown that the regions are singly connected. Assume X_A and X_B are in region R_k, this means that $y_k(X_A) > y_j(X_A), \forall j \neq k$ (same for X_B). Now pick any $\tilde{X} = \lambda X_A + (1-\lambda)X_B$ (a point somewhere along a line between X_A and X_B):

$$\begin{aligned} \tilde{X} =& \lambda X_A + (1-\lambda)X_B \\ y_k(\tilde{X}) =& \lambda y_k(X_A) + (1-\lambda)y_k(X_B) \\ \geq& y_j(\tilde{X}) \qquad \forall j \neq k \end{aligned}$$

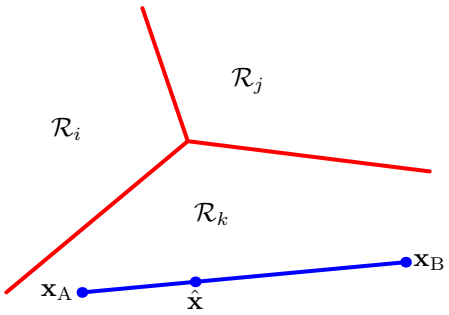


Figure 2. K class discriminants are singly connected. (Bishop Figure 4.3).

Bishop presents 3 ways of learning the linear discriminant: least-squares, Fisher's linear discriminant, and the perceptron. We will cover the first two.

4 Least-squares classification

Least-squares classification treats discrete targets as real-valued in exactly the same way as least-squares regression. The only difference is that in classification we

have multiply classes K. We will see later that this approach has serious drawbacks. By using a probabilistic approach, the targets are treated appropriately and correspondingly the solutions are better (next lecture). The motivation for using least-squares is to get a predictive function $\mathbb{E}[t|\mathbf{x}]$. The solution is useful to illustrate the negative properties an incorrect model assumptions creates.

First the sum-of-squares error for multiple classes:

$$E_D(\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\mathbf{w}_k^T\mathbf{x}_n - t_{nk}\right)^2$$

we have assumed an augmented $D+1$ representation for $\mathbf{x}$ to hide the bias terms.

To simplify we write $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K]$ where each column is the kth weight vector that has been augmented with w_{0k} (the kth bias). This means that $\mathbf{X}$ (N by $D+1$) has also been augmented with 1 (just like in linear regression). We can then write

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}^T\mathbf{x} \tag{1}$$

(K-dim prediction vector) and the objective function

$$E_D(\mathbf{W}) = \frac{1}{2}\,\mathrm{tr}\left[(\mathbf{XW}-\mathbf{T})^T(\mathbf{XW}-\mathbf{T})\right] \tag{2}$$

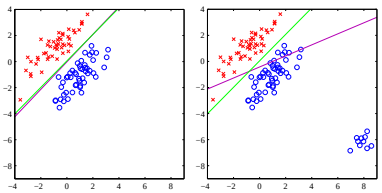
Note: the two ways of writing $E_D(\mathbf{W})$ give exactly the same value (look up trace operator in Matrix Cookbook). This version makes solving for $\mathbf{W}$ very simple:

$$\mathbf{W} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{T} \tag{3}$$

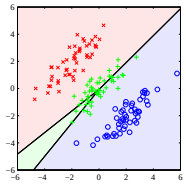
Note the actual decision is the same: arg max over $\mathbf{y}(\mathbf{x})$.

4.1 Problems with least-squares classification

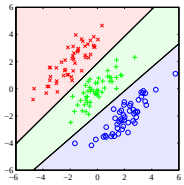
1. The least-squares solution is closed-form, and interestingly is constrained to sum to one, ie $\sum_k \mathbf{y}(\mathbf{x})_k = 1$ (the proof is tricky, but doable). Unfortunately, this does not mean that the predictions will in $[0, 1]$, let alone treatable as probabilities. We will see that y can be negative (problem: not positive, not probabilities).
2. Another important problem is the effect of new training points that are correctly classified, but are far from the decision boundary. Because we are forcing the classifier to be like regression, these 'too correct' examples (or outliers) will change the decision hyperplane to the detriment of the generalization error (problem: lack of robustness, sensitivity to outliers).



3. Problems with multiple classes.



These problems (mostly) disappear if we no longer make a Gaussian assumption and use the correct probabilistic assumptions. Logistic regression:



5 Fisher's linear discriminant

Fisher's linear discriminant (or more generally: linear discriminant analysis (LDA)) solves the classification problem indirectly, as a **dimensionality reduction** problem. Idea: find a linear project of the data onto a line (for binary classification), such that the distribution of the projected classes is maximally separated.

In this section we will keep to binary classification and therefore only need to find a single hyperplane $\mathbf{w}$. The form of the discriminant remains $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, however, the objective function which is optimized, changes to reflect the goal. More formally, we want to learn a projection that maximizes the squared distance between projected *means* and minimizes the within class sum-of-squares (**variance**). This is the **Fisher criterion** aka **Fisher objective function**:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{4}$$

where m_k is the mean projection for class k, and s_k^2 is the sum of squares of the predictions for class k.

5.1 Maximize projections of means

First we project the D-dim means for each class:

$$m_k = \frac{1}{N_k} \sum_{n \in C_k} \mathbf{w}^T\mathbf{x}_n \tag{5a}$$

$$= \mathbf{w}^T\mathbf{m}_k \quad = \mathbf{m}_k^T\mathbf{w} \tag{5b}$$

then the difference between projections is maximized:

$$m_2 - m_1 = (\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w} \tag{6a}$$

$$(m_2 - m_1)^2 = \mathbf{w}^T \underbrace{(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T}_{\mathbf{S}_B\text{: between class cov.}} \mathbf{w} \tag{6b}$$

$$= \mathbf{w}^T \mathbf{S}_B \mathbf{w} \tag{6c}$$

This is the numerator of the objective function (increasing it will maximize the project).

5.2 Minimize the sum-of-squares of projected data

The sum-squares for a class k is:

$$s_k^2 = \sum_{n \in C_k} (\mathbf{x}_n^T \mathbf{w} - m_k)^2 \tag{7a}$$

$$= \sum_{n \in C_k} (\mathbf{x}_n^T \mathbf{w} - \mathbf{m}_k^T \mathbf{w})^2 \tag{7b}$$

$$= \mathbf{w}^T \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{w} \tag{7c}$$

The sum over classes (in this case only 2) is:

$$s_1^2 + s_2^2 = \mathbf{w}^T \underbrace{\left[\sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T \right]}_{\mathbf{S}_W\text{: within class cov.}} \mathbf{w}$$

$$= \mathbf{w}^T \mathbf{S}_W \mathbf{w} \tag{8a}$$

(8b)

This is the denominator of the objective function (decreasing it will minimize the sum-of-squares).

5.3 Maximizing the Fisher objective

The Fisher objective becomes:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{9}$$

We want to maximize the objective with respect to $\mathbf{w}$, so again we take the derivative and set to 0. Note that we will be primary be interested in the *direction*

of $\mathbf{w}$ (there will be some constants involved that we will ignore below).

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{2\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} - \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{(\mathbf{w}^T\mathbf{S}_W\mathbf{w})^2}(2\mathbf{S}_W\mathbf{w}) \tag{10a}$$

$$0 = \underbrace{(\mathbf{w}^T\mathbf{S}_W\mathbf{w})}_{\text{constant } C_1}\mathbf{S}_B\mathbf{w} - \underbrace{(\mathbf{w}^T\mathbf{S}_B\mathbf{w})}_{\text{constant } C_2}\mathbf{S}_W\mathbf{w} \tag{10b}$$

$$C_2\mathbf{S}_W\mathbf{w} = C_3\mathbf{S}_B\mathbf{w} \tag{10c}$$

$$\mathbf{S}_W\mathbf{w} = C(\mathbf{m}_2 - \mathbf{m}_1) \quad \underbrace{(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w}}_{\text{const. dep. on mag. w } C_4} \tag{10d}$$

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1) \tag{10e}$$

The final *direction* $\mathbf{w}$ is therefore a **rotation** of the line $\mathbf{m}_2 - \mathbf{m}_1$; $\mathbf{w}$ is called Fisher's linear discriminant, but it only gives a direction of the projection. An actual algorithm requires selecting some threshold to distinguish the regions. One way is to fit a Gaussian to each projected class, then select based on the class-conditional Gaussian with the highest probability, $p(y|C_k)$ (where C_k now refers to belonging to the kth class.) Another way is to simply fine a threshold y_0 and choose C_1 if $\mathbf{w}^T\mathbf{x} > y_0$.

Note that there is an equivalence between least-squares classification and Fisher's if the targets for least-squares are set to $t = N/N_k$ (see Bishop).

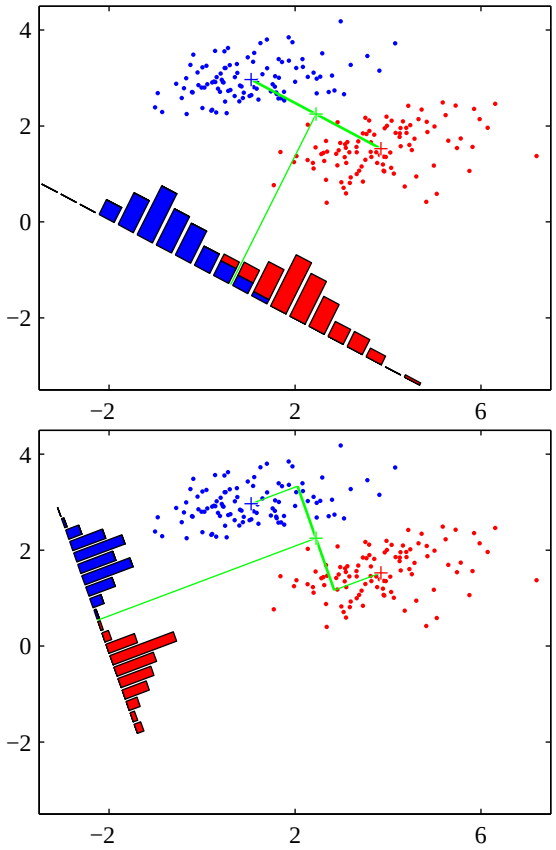


Figure 3. Fisher linear discriminant. (Bishop Figure 4.6).