

# Linear Models of Classification

Andres Ponce

March 28, 2020

The goal with **classification** is "to take an input vector  $x$  and to assign it to one of  $K$  discrete classes.

If we want to classify a certain input vector, we would have to map it to a certain region of a plane, defined by certain boundaries.<sup>1</sup>

If there are  $D$  variables we are trying to optimize, then our answer will exist in a  $D$  dimensional space.

## Discriminants

**Linear** discriminants can be of the form<sup>2</sup>

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

We also have a **decision boundary**, where we know which of the  $K$  categories our input belongs to given our discriminant function, and a certain cutoff value  $C_x$ .

We can also use a Bayesian approach to determine classification, if we calculate  $p(C_k|x)$ <sup>3</sup>

How do we extend a linear discriminant to other classes? We can have **One-vs-the-rest** or **one-vs-one** qualifiers.

## One-vs-the-rest

With this type of classifier functions, we try to distinguish those inputs on a binary case by case basis. That is, we try to one at a time build a classifier that knows points in  $C_i$  from those points in other classes. However, when two points "not in their respective classes" fall in the same area, this area would be defined differently by the areas we're actually testing.

## $K$ -class discriminant

If we use  $K$  linear functions of the form

$$y_k(x) = w_k^T(x) + w_{k0}$$

Then point  $x$  goes in class  $k$  iff  $y_k(x) > y_j(x) \forall j \neq k$ <sup>4</sup>

## Least Squares for classification

Similar to the original least squares definition, we want to minimize the difference between the data points and the final values.

<sup>1</sup> Is this similar to **linear programming**, where we had to optimize a linear combination of the parameters?

<sup>2</sup> Here,  $w$  is the parameters, and  $w_0$  is the bias.

<sup>3</sup> The probability that given input  $x$ , it belongs in class  $k$ .

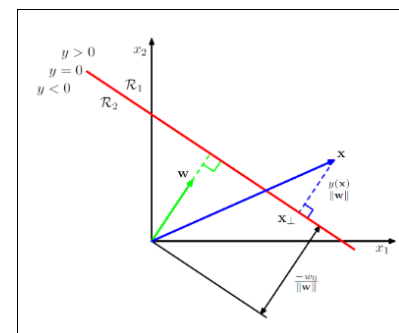


Figure 1: The dotted green and blue lines are the distance between the arbitrary point and the decision boundary.

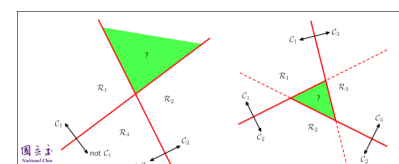


Figure 2: When a point is not in a certain region, it can be ambiguous where it falls, if the same point can be thought of differently by each region.

<sup>4</sup> essentially we describe every region by its own equation, and assign the points to the one that fits the best.

The actual formula for the least squares classification is

$$y(x) = \tilde{W}^T \tilde{x}$$

However, the least squares solution usually does not have the best performance, and is sensitive to **outliers**.

### Fisher's Linear Discriminant: 2 classes

**Fisher's Linear Discriminant** can help with dimensionality reduction at the moment of solving a K-dimensionality problem. With these kinds of problems, we want to find a linear combinations of elements that maximizes the distance between data points not in the same class.

The **mean vectors** between two classes is

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n$$

and likewise for  $m_2$ . Then, we want a function that maximizes the distance between the two functions. We would like

$$m_2 - m_1 = w^t(m_2 - m_1)$$

to be maximized. We also want to minimize the variance within a single class.

So, there are a couple ways to carry out classification:

- Determine a threshold: We can find a point  $x$  such that if  $y(x) \geq -y_0$ ,  $C_1$ , else  $C_2$ .
- use the nearest-neighbor rule: When we project a training sample onto a one-dimensional space, we can sample some of the training data a certain distance away, and classifying the point based on the number of training points belonging to a certain class.

### Perceptron Algorithm

The **perceptron algorithm** works by having a linear combination of the terms  $y(x) = f(w^T \phi(x))$ . We also want  $f$  to have a value either 1 or -1. Another parameter  $t$  can either be set to -1 or 1, and during the training phase we usually set it to 1 for  $C_1$  and -1 for  $C_2$ .

The goal of the perceptron then becomes to make  $w^T \phi(x_n) t_n > 0$ , since this means that the point  $x_n$  was correctly classified. In classification problems, we only have a binary choice: whether the point was classified into the correct class. Therefore, the error function to judge our model should be a function of all the points that were actually *misclassified*. One such function might be

$$E_p(w) = - \sum_{n \in M} w^T \phi_n t_n$$

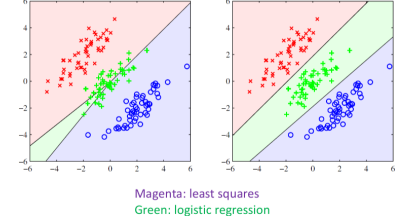


Figure 3: Logistic regression can sometimes have better performance than ordinary least squares.

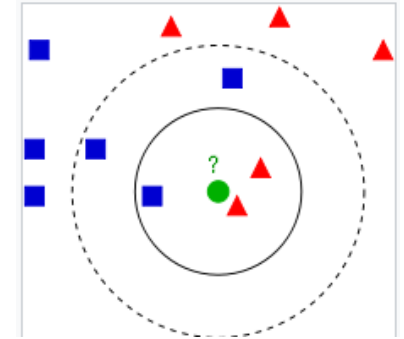


Figure 4: If we take the sample space to be the solid colored line, we say the green sample point is a red triangle, since the closest points are mostly red triangles (2-1). Using the dashed circle, we would classify it as a blue square for the same reason.