Linear Models of Classification

Andres Ponce

April 3, 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.[1]

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

[1] Is this similar to **linear programming**, where we had to optimize a linear combination of the parameters?

Discriminants

Linear discriminants can be of the form[2]

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

[2] Here, w is the parameters, and w_0 is the bias.

We also ahve 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 $p(C_k|x)$[3]

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

[3] The probability that given input x, it belongs in class k.

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 repsective classes" fall in the same area, this area would be defined differently by the areas we're actually testing.

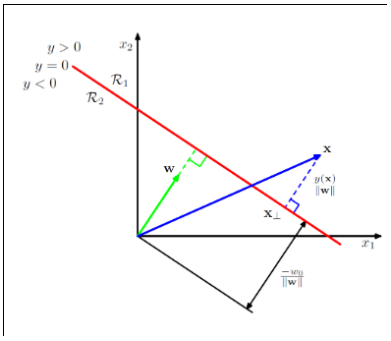

Figure 1: The dotted green and blue lines are the distance between the arbitrary point and the decision boundary.

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 $y_k(x) > y_j(x) \forall j \neq k$ [4]

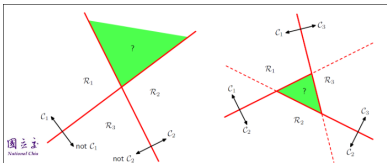

Figure 2: When a point is not in a certain region, it can be ambiguous where it falls, if the same point can be thought of differently by each region.

[4] essentially we describe every region by its own equation, and assign the points to the one that fits the best.

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.

The actual formula for the least squares classification is

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

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

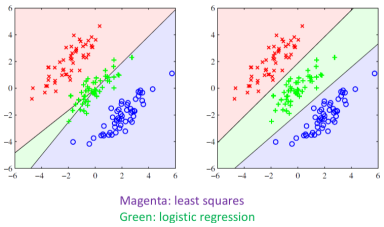

Figure 3: Logistic regression can sometimes have better performance than ordinary least squares.

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 lienar combinations of elements that maximizes the distance between data points not in the same class.

That is, we want to reduce the number of dimensions from D down to 1; then we want to select a threshold t, which serves to mark where one point would be classified as C_1 or C_2. After we have clearly separated the different classes in a reduced space, we would like to maximize the distance between the points of all classes, so as to minimize the variance between points of the same class and thus maximize the distance between points of different classes.

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 coupple 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.

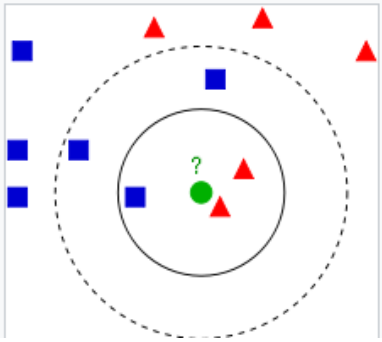

Figure 4: If we take the sample space to be the solid colored line, we say the green sample point is a red triangle, since the closest points are mostly red triangles(2-1). Using the dahsed circle, we would classify it as a blue square for the same reason.

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

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

where M is the set of points that were misclassified. Then, the weight vector is updated depending on whether we correctly classified the data or not. Add ϕ_n to w if x_n belongs to C_1 and was misclassified, and subtract ϕ_n from w if x_n belongs to C_2 and was misclassified.

Generative Models

In these types of mdoels, we try and figure out $p(C_k|x)$ by calculating $p(x|C_k)$ and $p(C_k)$ and solving for Bayes' Theorem.[5]

However, this was only the case for 2 classes; we can try to generalize to more classes. The basic formula still very much resembles Bayes' Theorem.[6]

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)}$$

With this type of model, we also have to set several parameters, π, μ_k. [7]

[5] The main idea of the gernative model remains to find the *joint probability* both the target variable and our input variable, i.e. $p(t, x)$

[6] basically the probability that the input vector belongs to class C_k over the probability that it belongs to any of the other classes.

[7] We can use the maximum likelihood to identify the parameters, if we recall by maximizing the log of the likelihood function.

Different models

So far, we have two different models for estimating the parameters.

- **Probabilistic Generative Model**: As we just saw, with this model we try and estimate the posterior probability by computing the class priors and trying to solve Bayes' Theorem.
- **Probabilistic Discriminative Models**: Here we also try and solve for the posterior probability of an input belonging to a certain class, but we generalize the linear model (like with Fisher's and Perceptron) to solve for the required parameters directly. There are a couple advantages to using this type of model, for example there are less parameters and usually has better performance.

After this, the slides mostly just discussed how to find the parameters used in these and related equations.