Machine Learning Course - CS-433

Classification

Oct 18, 2016

changes by Rüdiger Urbanke 2016

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Classification

Similar to regression, classification relates input variables $\mathbf{x}$ to the output variable y, but now y can only take discrete values, i.e., y is a categorical (or nominal) variable.

Binary classification

When y can only take two discrete values, it is called binary classification. There are many notations in use. Often we say that $y \in \{\mathcal{C}_1, \mathcal{C}_2\}$. The values $\mathcal{C}_i$ are called class labels or simply classes. Other common notations are $y \in \{-1, +1\}$ or $y \in \{0, 1\}$, although there may not necessarily be any ordering between the two classes.

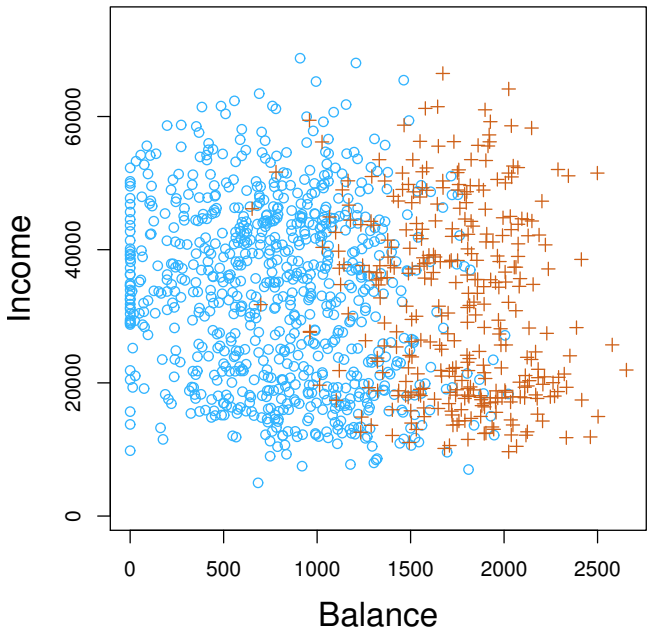
Multi-class classification

In a multi-class classification, y can take multiple discrete values i.e., $y \in \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{K-1}\}$ for a K-class problem. Again, there is no notion of ordering among these classes, but we may ignore this fact and may sometimes use the following notation for convenience: $y \in \{0, 1, 2, \ldots, K-1\}$.

Examples of classification problems

A credit card service must be able to determine whether or not a requested transaction is fraudulent. They might have at their disposal the users IP address (if the transaction happens on the web), past transaction history, and perhaps some other features.

An example is shown below on the *default* dataset from JWHT. We have at our disposal the annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not default are shown in blue.



To consider another example, a person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?

Classifier

A classifier will divide the input space into a collection of regions belonging to each class. The boundaries of these regions are calld decision boundaries. A classifier can be linear or nonlinear. This distinction is less strict than it might seem

at first. E.g., if you look at the classifier in the right-hand side of "Figure 4.1" you will see that the decision regions are non-linear (not straight lines). But in fact the classifier that led to these region is linear. We added some non-linear features to the original feature vector (think polynomial basis) before performing the linear classification. The decision boundaries appear non-linear since the plot is made in the original feature space and not the extended feature space.

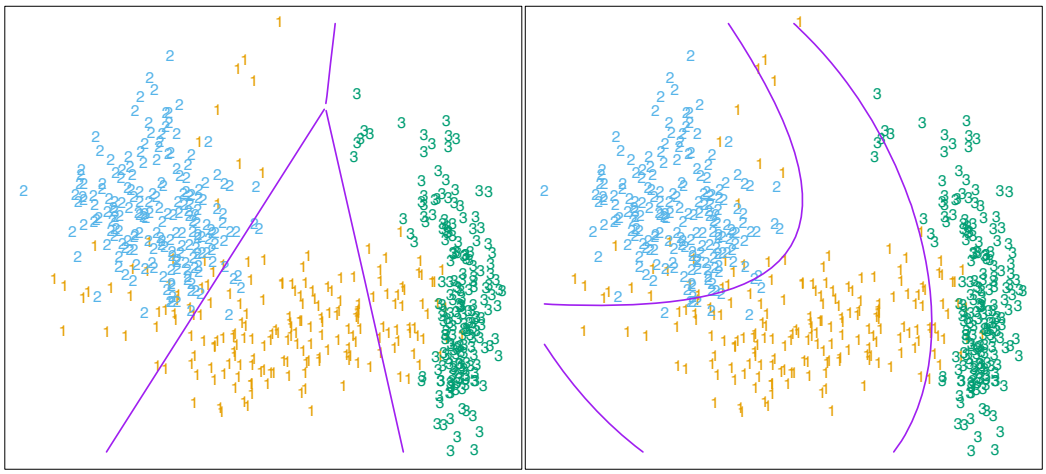


FIGURE 4.1. *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_1X_2, X_1^2, X_2^2$. *Linear inequalities in this space are quadratic inequalities in the original space.*

What is the aim of classification?

In some situations we are interested in classification in itself. This means, we are constructing a predictor based on a training set and are interested in applying this predictor to "new" incoming data. Consider e.g. the example of the credit card company that wants to predict if a customer will default or not.

But in some instances we are interested in more. We would like to "understand" the cause. Think e.g. of a decease prediction. Not only are we interested to know who is at risk but we would like to understand *why* somebody is at risk. So we are interested in the "interpretation" of the prediction. In particular for this second task it is important to have *simple* models and to have means of eliminating features that do not contribute significantly to the prediction.

Consider "Figure 4.12". In this data set various risk factors are given for a particular heart disease. These risk factos are blood pressure (sbp), tobacco usage, family history, obesity, alcohol consumption, and age. For each pair of these risk factors the figure shows how the cases (people who have this disease) separate from the controls (people who do not have the desease). Such plots can help to decide which risk factors should be included in a model and which might have little predictive power.

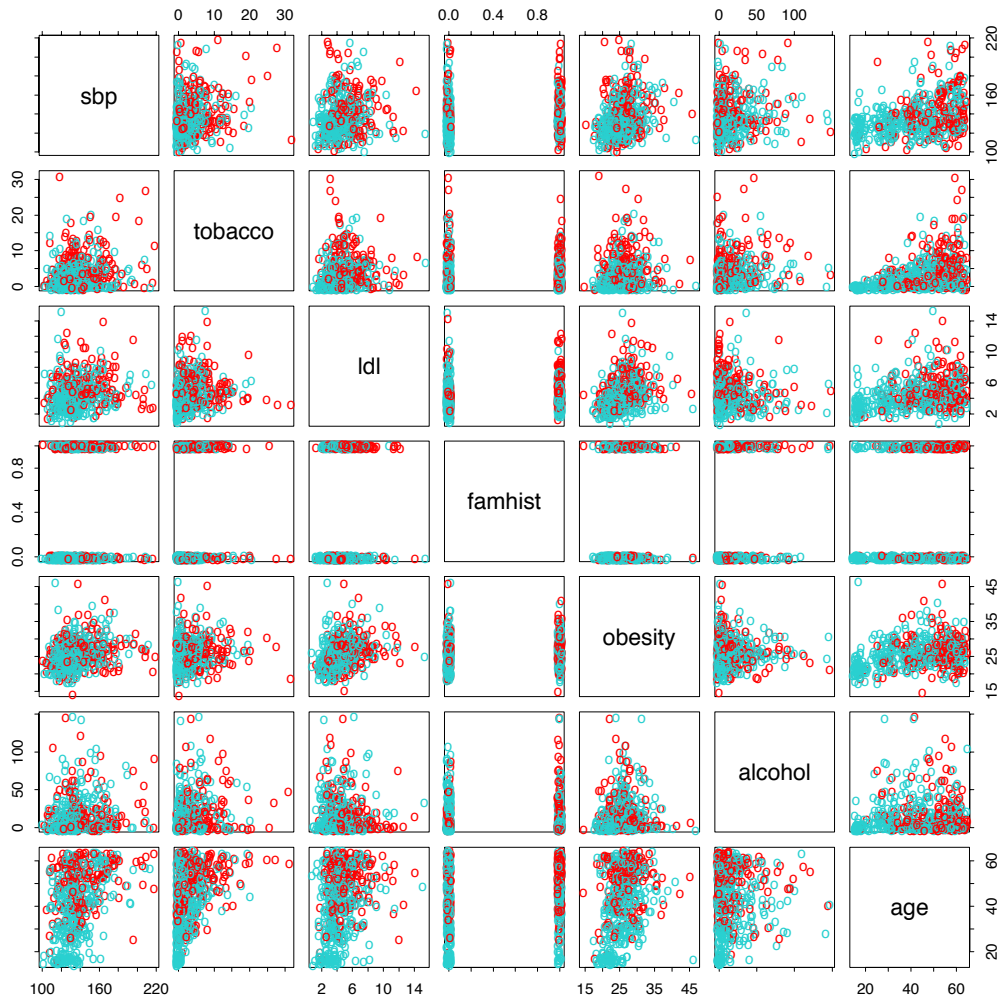


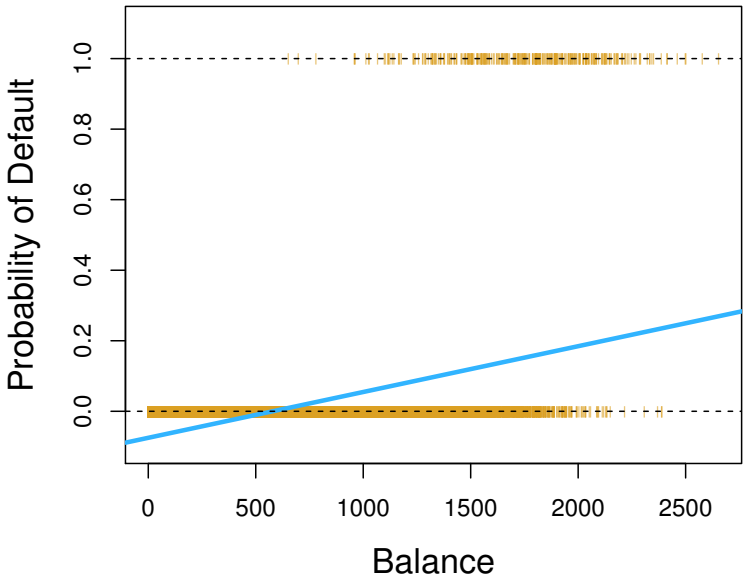
FIGURE 4.12. *A scatterplot matrix of the South African heart disease data. Each plot shows a pair of risk factors, and the cases and controls are color coded (red is a case). The variable* family history of heart disease (`famhist`) *is binary (yes or no).*

Classification as a special case of regression

From the very definition we see that classification is a *special case* of regression. It is a special case since the output is restricted to a small discrete set. So it might appear there

is not much to be discussed and we should simply apply our standard regression techniques to this special case.
E.g., we can assign $y = 0$ for $\mathcal{C}_1$ and $y = 1$ for $\mathcal{C}_2$ and, given a training set S_t, we can use (regularized) least-squares to learn a prediction function f_{S_t} for this regression problem. To convert the regression into a classification it is then natural to decide on class$\mathcal{C}_1$ if $f_{S_t}(\mathbf{x}) < 0.5$ and $\mathcal{C}_2$ if $f_{S_t}(\mathbf{x}) > 0.5$.
In the figure below this approach is applied to the credit-card default problem. To keep things simple we use as feature only the *balance* (as we have seen in a previous plot the second feature (the *income*) does not contain much information).
We think of $y = 0$ as "no default" and $y = 1$ as "default". The dots we see corresponds to the various data points and all dots are either on the line $y = 0$ or $y = 1$. The horizontal axis corresponds to the input $\mathbf{x}$, the *balance*.
In the figure the output y is labeled as *probability*. This is just a convenient way of interpretation. Since the desired label y is either 0 or 1 we can think of y as the probability of a default.
So let us now run a regression on the training data S_t. To keep things simple we run a linear regression and learn the linear function $f_{S_t}(\mathbf{x})$. The result is the blue curve that is indicated.
We might want to interpret the value $f_{S_t}(\mathbf{x})$ as the probability of a default and then assign a label depending on whether this "probability" is smaller or larger than 0.5. Of course, this "probability" can be negative or be larger than 1 so such an interpretation has to be taken with a grain of salt.
It is not hard to see that this method can lead to questionable

We can not just use lineal regression to classify discret results!



results. Even if the cases and controls are well separated, the general "position" of the line will depend crucially on how many points are in each class and where these points lie. E.g., if we add a few points with $y = 1$ and a very large balance this will shift the curve significantly even though only a few points changed. This is clearly not a desirable property.

Why does this happen? The quared loss function that we often use for regression problems is not a good match to our objective. We would like that the fraction of misclassified cases is small. But the mean-squared error is only very loosely related to this objective. In particular, the mean-squared error counts positive and negative deviations from the class label equally bad, although only one of them can potentially lead to a misclassification. If we do have a very small mean-squared error then indeed we can guarantee a small classification error but the opposite is not true – a regression function can have arbitrarily large mean-squared error even though the fraction of misclassified cases is arbitrarily small.

Property of mean-square error:
-> if error is small, then regression is guarantee to be good!
-> if error is large... we can't say nothing and it's the case for classification !

We therefore might have to work "much harder" than we should in bringing down the mean-squared error and so to have a guarantee on the misclassification error.
Based on the above observation, we see that classification is not just a special form of regression with a simple loss function like the mean-squared error.

Some basic ideas of how to perform classification

Many different approaches have been developed over the years of how to efficiently perform classification. Our aim right now is not to give an exhaustive account of all possible techniques. Rather, let us quickly discuss some basic ideas. We will come back and discuss several of those in much more detail in later lectures.

Nearest Neighbor

In some cases it is reasonable to postulate that inputs that are "close" are also likely have the same label attached. Here "close" might e.g. be measured by the Euclidean distance. If we believe that this assumption is good, then, given an input $\mathbf{x}$ and a training set S_t, we can look for that point $\mathbf{x}^*$ which is closest to $\mathbf{x}$ and an element of S_t and then output y^*, the label attached to $\mathbf{x}^*$.
The good point about such a classifier is that it might work well even in cases where the decision boundaries are very irregular (see the Figure 2.3 below). But, as we will discuss in a later lecture, such a scheme fails miserably in high di-

mensions since in this case the geometry renders the notion of "close by" meaningless.

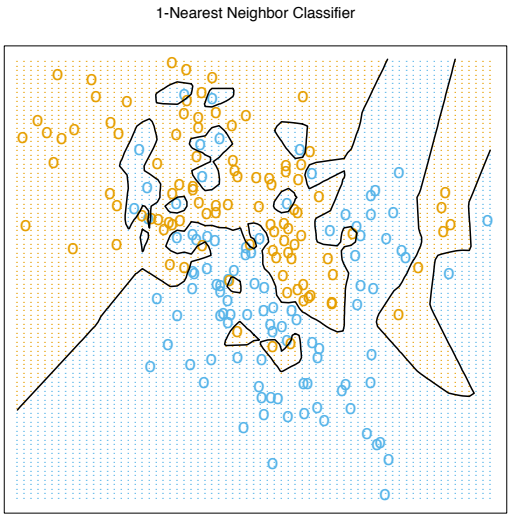


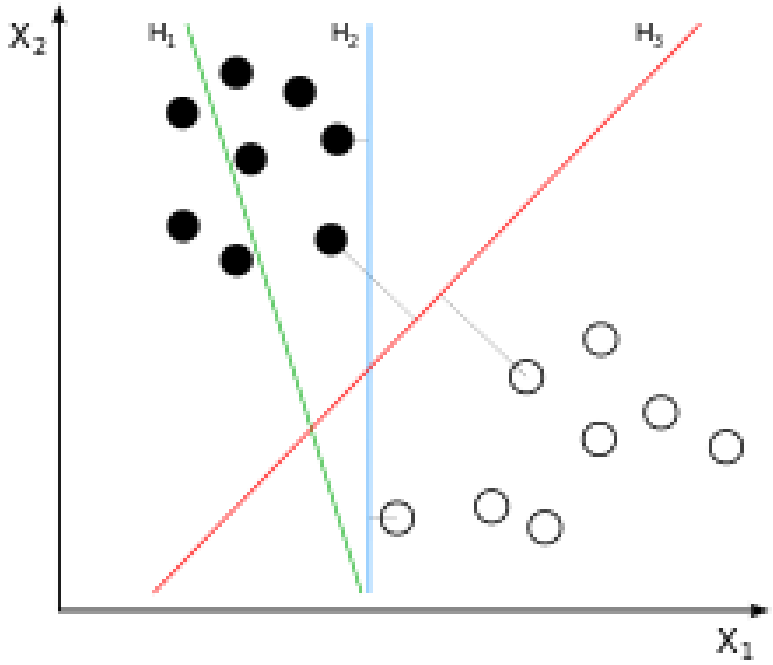
FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

There are many natural generalizations of this concept. Instead of using a single neighbor we can use lets say the k nearest neighbors or we can take a weighted linear combination of elements in our neighborhood. The latter idea leads to to *smoothing kernels.*

Linear decision boundaries

One starting point is to assume that decision boundaries are linear (hyperplanes). Consider e.g. a binary classification problem and look at schemes where the boundary between the two classes is a hyperplane. We can ask how to pick this boundary. To keep things simple, assume that there exists a "separating hyperplane" i.e., a hyperplane so that no point in the training set is misclassified.

In general, there might be many hyperplanes that do the trick (assuming there is at least one). So which one should we pick?
One idea is to pick a hyperplane so that the decision has as much "robustness/margin" with respect to the training set as is possible. I.e., if we slightly change the training set by "wiggling" the inputs we would like that the number of misclassifications stays low.
In the figure below (taken from Wikipedia) we see that H_1 does not separate the data, but H_2 and H_3 do. Between H_2 and H_3, H_3 is preferable, since it has a larger "margin." This idea will lead us to *support vector machines* (SVM).



The technique of SVM is not the only method that leads to linear decision regions. One very popular and powerful technique that leads to linear decision regions is called *logistic regression*. We will spend the whole next class to talk about it.

Non-linear decision boundaries

In many cases linear decision boundaries will not allow us to separate the data and non-linearities are needed. One option is to augment the feature vector with some non-linear functions. The *kernel trick* is a method of doing this in an efficient way.
Another way is to find an appropriate non-linear transform of the input so that the transformed input is then linearly separable. This is what is done when we are using *neural networks*.

Optimal classification for known generating model

It is instructive to think about how one could classify in an *optimal* fashion, i.e., how one could minimize the probability of misclassification, if the distribution of the generating model was known. To be concrete, assume that we know the joint distribution

$$p(\mathbf{x}, y)$$

and that y takes on elements in a discret set $\mathcal{Y}$.
Given the "observation" (input $\mathbf{x}$), what is the optimal choice for the class label, call it $\hat{y}(\mathbf{x})$. Our aim is to minimize the probability of misclassification, i.e., we want to pick the function $\hat{y}(\mathbf{x})$ in such a way that

$$\mathbb{P}\{\hat{y}(\mathbf{x}) = y\}$$

is maximized, where we think of the pair $(y, \mathbf{x})$ as a random variable with joint distribution $p(\mathbf{x}, y)$. Writing the criterion

explicitly, we have

$$\mathbb{P}\{\hat{y}(\mathbf{x}) = y\} = \int p(\mathbf{x}) \sum_{y \in \mathcal{Y}} p(y \mid \mathbf{x}) \mathbf{1}_{\{\hat{y}(\mathbf{x})=y\}} dx.$$

Note that the indicator function $\mathbf{1}_{\{\hat{y}(\mathbf{x})=y\}}$ takes on the value 1 for exactly one value of $y \in \mathcal{Y}$, namely that value of y that the function $\hat{y}(\mathbf{x})$ chooses, and it takes on the value 0 otherwise. Therefore we see that we maximize the expression, and hence the probability of correct classification, if we chose the decision rule

$$\hat{y}(\mathbf{x}) = \text{argmax}_{y \in \mathcal{Y}} p(y \mid x).$$

The above discussion is not realistic. We do not know the joint distribution $p(\mathbf{x}, y)$. But we could use such an approach by using the date to learn the distribution (perhaps by assuming that the distribution is Gaussian and then just fitting the parameters from the data).