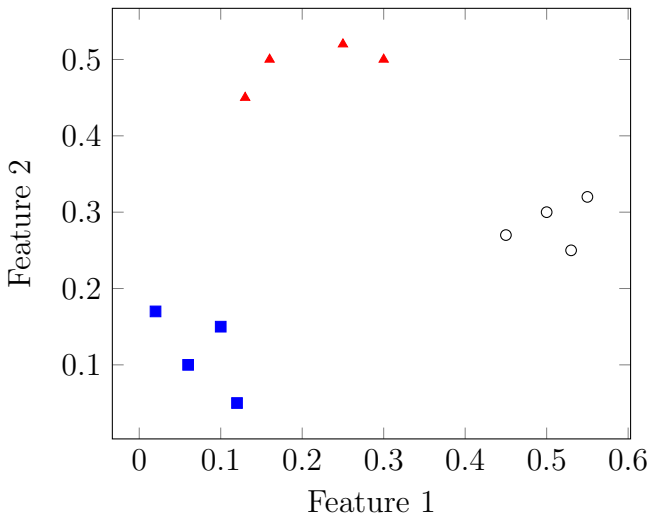
Multiclass Classification

Multiclass

So far we have learned about how to do binary classification. We could do pattern recognition and answer questions such as is this customer a good/bad customer, is this email a spam or a ham.

There is another type of question that is sort of related to binary classification; it is called multiclass classification. This is the question of the type: is this image a digit 0,1,2,3,4,5,6,7,8 or 9? There are more than two answers. An example of such problem is illustrated in the figure below



Some algorithm such as neural network and decision tree can be modified to do multiclass classification. We will cover neural network later. But, the algorithm we have learn so far can only do binary classification. Luckily, we can combine a bunch of binary classifiers to make a multiclass classifier.

One VS All

The first thing most people would try is to do One VS All. The idea is very simple. Consider the problem of trying to classify data in to class A, B, C and D. We could train 4 logistic regression model to model these four probabilities.

Classifer	$+1$ VS -1
Classifier $\bar{A}$	A VS B, C, D
Classifier $\bar{B}$	B VS A, C, D
Classifier $\bar{C}$	C VS A, B, D
Classifier $\bar{D}$	D VS A, B, C

Each classifier is trained to do one class(+1) vs all other classes(-1); hence the name one vs all.

Once we train these four classifiers. We can classify an unknown point by getting the output from the four classifiers. We can intepret each output as probability of belonging to a class. For example, classifier $\bar{A}$ gives the probability of the unknown data being of class A, classifier $\bar{B}$ gives the probability of the unknown point being of class B etc. Then, it is clear that we should pick the one with the highest probability. For instance, if the four classifiers above give the result as shown in the following table:

Classifer	Train for	Output
Classifier $\bar{A}$	A VS B, C, D	0.2
Classifier $\bar{B}$	B VS A, C, D	0.3
Classifier $\bar{C}$	C VS A, B, D	0.7
Classifier $\bar{D}$	D VS A, B, C	0.3

It is clear that we should classify this unknown point as belonging to class C.

ECOC

It turns out that one vs all is just one particular algorithm of a larger class of multiclass classification algorithm called Error-correcting output code. To explain this let us start by restate one vs all algorithm in another angle.

Now consider the problem of four classes again. This time we are going to explain the idea of one vs all using the idea of template. First we train 4 classifiers:

	Classifier $\bar{A}$	Classifier $\bar{B}$	Classifier $\bar{C}$	Classifier $\bar{D}$
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

This table is call *code table* the reason for the name will become clear in the next section.

First, let us interpret the column of this table. The column tells us how to train each classifer. For example, the first column has 1 in row A, 0 in row B, 0 in row C and 0 in row D. This tells us to train the first classfier to separate between A(as class +1) and B,C,D as (class -1).

Once we have all the classifiers, we can intepret the row as the perfect answer for each class. For example, the first row is for class A. If we have a data sample of class A and all the classifer does perfect job, then we expect that the first classifier will give out probability 1 and all the other classifiers will give the probability of zero.

These perfect answer row can be used as template for the unknown point. Concretely, suppose we have a new point and the 4 classifiers give the output of $[0.2, 0.3, 0.7, 0.3]$. To assign a class to this unknown data point we can just "compare" this output vector to the template then pickout the closest class. We can use the sum of square of the diffrence between the unknown output and the template for each classifier. For instance, to calculate how close $[0.2, 0.3, 0.7, 0.3]$ is to $[1, 0, 0, 0]$, we calculate $(1-0.2)^2+(0-0.3)^2+(0-0.7)^2+(0-0.3)^2 = 1.31$. An example is shown below:

	Classifier $\bar{A}$	Classifier $\bar{B}$	Classifier $\bar{C}$	Classifier $\bar{D}$	Distance
Unknown	0.2	0.3	0.7	0.3	
A	1	0	0	0	1.31
B	0	1	0	0	1.11
C	0	0	1	0	0.31
D	0	0	0	1	1.1

Once we calculate the distance of the output for the unknown point and the template ouput for each class, all that is left to do is to pick the closest class. In this case, it is class C.

In summary, the idea of one vs all matrix can be described as training multiple classifiers each to solve different binary problem. Then, once we get the ouput from all the classifiers, we can just compare the answer to the template of perfect answer for each class. Then, pick the class with the lowest distance.

Exhaustive Code Table

One vs all works but it is not robust against classifier making mistake. We could build a classifer that could recover its own mistake.

Supposed we want to send a message between 2 parties through a lossy channel. The message we sent is a binary one(yes/no). We could send 1 bit to represent our message 1 for yes and 0 for no. But since the communication channel is noisy the message can be altered in between so we can try to be a bit more clever by sending not just one bit but 3 bits: 111 for yes and 000 for no. You many think that this is redundant and is a wasted of bandwidth. However, we just did something quite amazing.

Supposed we send the messsage yes 111 to the receiving party but during the transit the middle bit get flipped. So, the receiving party receive 101 instead of the perfect 111. Even though the receiving pary did not get the perfect message one can still recover the true message if the message is not too screwed up(this has exponentially smaller probability). So to recover the message once could just compare what we receive

101 against 111 and 000. Then we calculate the distance between what we get and the template whichever one is closer is more likely to be what the sender meant. This is the reason for the name error correct output code.

This idea of using redundant data so that we can recover from mistake can be applied to our problem of multiclassification. The idea is to train a bunch of extra classifers doing different things. The more redundancy we have the better we can recover from mistakes. However, we do not want to do duplicate work.

Let us count that maximum number of classifer we can train for the problem of n classes. We just need to look at each column of the code table and making sure there i) there is no duplicate/complementary duplicate and ii) each classifer actually sepearte something. We can count this by considering each column as a binary string of length n. First, all the possible binary string of length of n is 2^n. However, this contains complementary duplicate. As an example the column that reads 1100 and 0011 are actually the same classifier, we switch what we call class +1 and class -1; they are solving the same binary problem. We only want to count this one once. So, we need to divide 2^n by 2. So far, we have 2^{n-1} classifier left. The last thing we need to get rid off is the binary string with all 1 or all 0. Since we already getrid of complementary duplicate we only need to subtract one off from what we have left. So, given a problem of n classes we can train at most $2^{n-1} - 1$ different classifers. The code table with the maximum number of the column is called exhaustive code table. The exhaustive code table for 4 classes is shown below.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
A	1	0	0	0	1	1	1
B	0	1	0	0	1	0	0
C	0	0	1	0	0	1	0
D	0	0	0	1	0	0	1

To assign the class we just do the same thing, get all the output and compare it to the template. Then we assign whichever class that has the closest answer. If your number of class is small and you have computing power, you should always use exhaustive code table instead of one vs all code table. This is much more robust against a classifier making mistake.