Lab 1

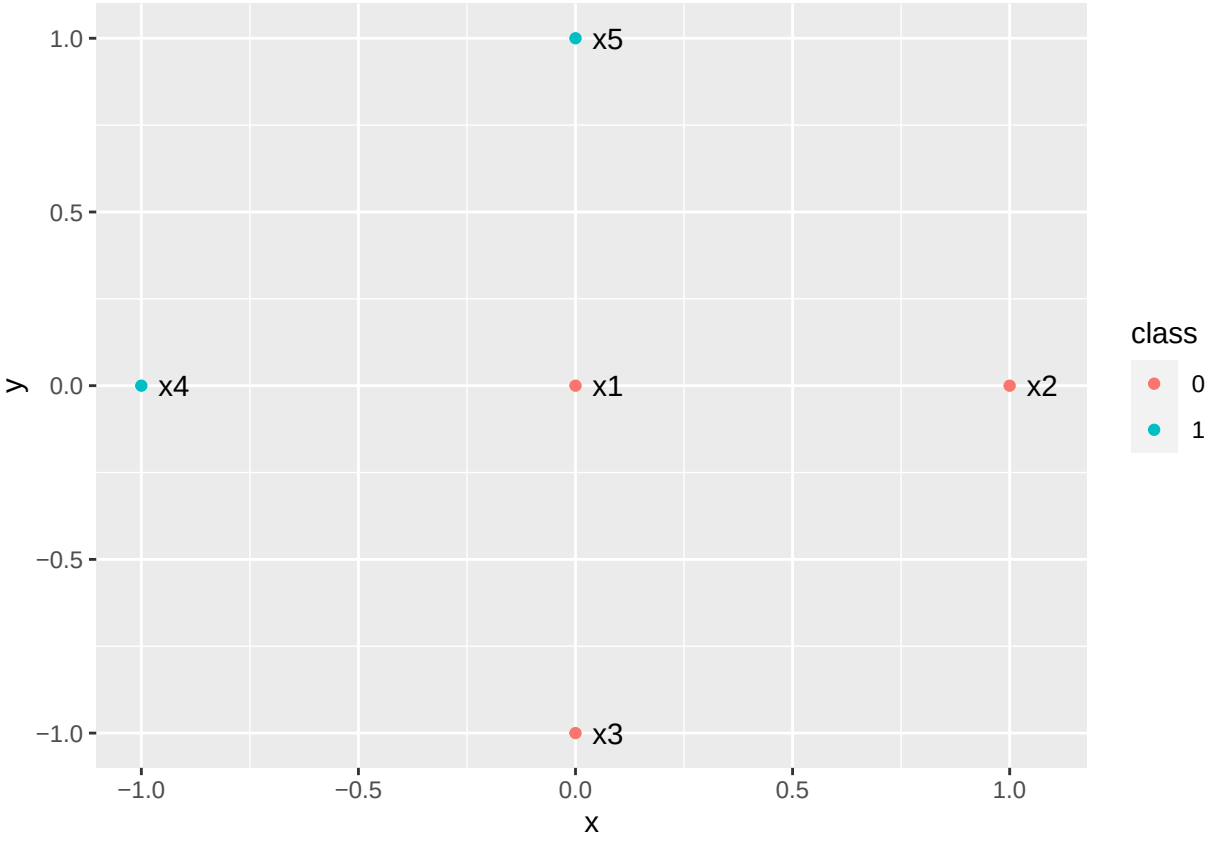
Hüseyin Anil Gündüz

2022-05-03

Welcome to the very first lab, in which we will have fun with logistic regression.

Exercise 1

Suppose you have five input points, $\mathbf{x}_1 = |0, 0|^T$, $\mathbf{x}_2 = |1, 0|^T$, $\mathbf{x}_3 = |0, -1|^T$, $\mathbf{x}_4 = |-1, 0|^T$ and $\mathbf{x}_5 = |0, 1|^T$, and the corresponding classes are $y_1 = y_2 = y_3 = 0$ and $y_4 = y_5 = 1$:



Consider a logistic regression model $\hat{y}_i = \sigma(\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2})$, with $\sigma(\cdot)$ the sigmoid function, $\sigma(x) = (1 + e^{-x})^{-1}$. What values for α_0, α_1 and α_2 would result in the correct classification for this dataset? A positive label is predicted when the output of the sigmoid is larger or equal than 0.5.

> **Note**: do not use any formulas or automated methods to find the answer. Think for yourself. A logistic regression classifier is nothing more than a hyper-plane separating points of the two classes. If necessary, review vectors, dot-products and their geometrical interpretation in linear algebra. This applies to the following exercises, too.

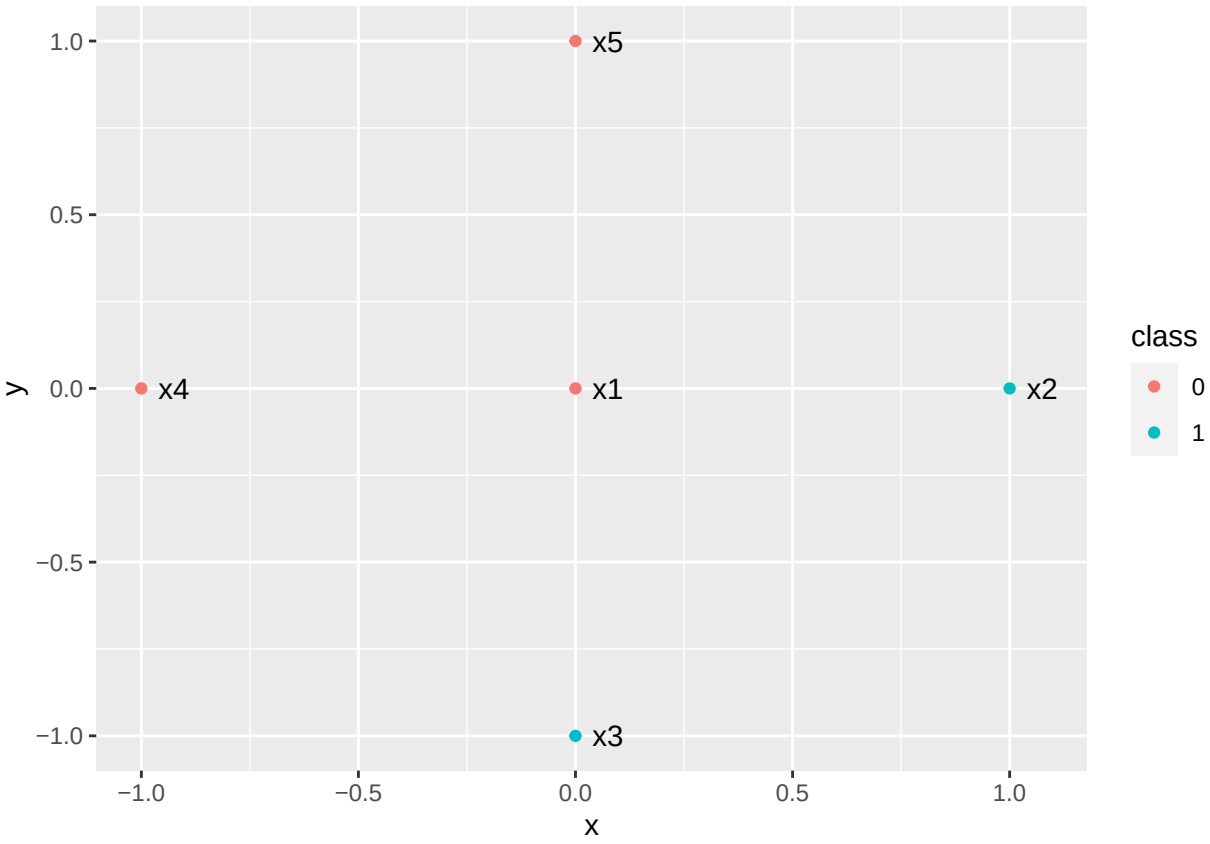
```r
a0 = (
  # TODO fill in the value for alpha 0
)

a1 = (
  # TODO fill in the value for alpha 1
)

a2 = (
  # TODO fill in the value for alpha 2
)

# the first column is always one and is used for the "bias"
xs = matrix(c(
  1, 0, 0,
  1, 1, 0,
  1, 0, -1,
  1, -1, 0,
  1, 0, 1
), ncol = 3, byrow = T)

sigmoid = function(x) {
  # TODO compute and return the sigmoid transformation on x
}

sigmoid(xs %*% c(a0, a1, a2))
```

You should make sure that the last two values are close to one, and the others are close to zero.

> **Note:** There are many valid parametrization that lead to a separating hyperplane. How would you prioritize between them?

Exercise 2

Continuing from the previous exercise, suppose now that $y_2 = y_3 = 1$ and $y_1 = y_2 = y_5 = 0$:



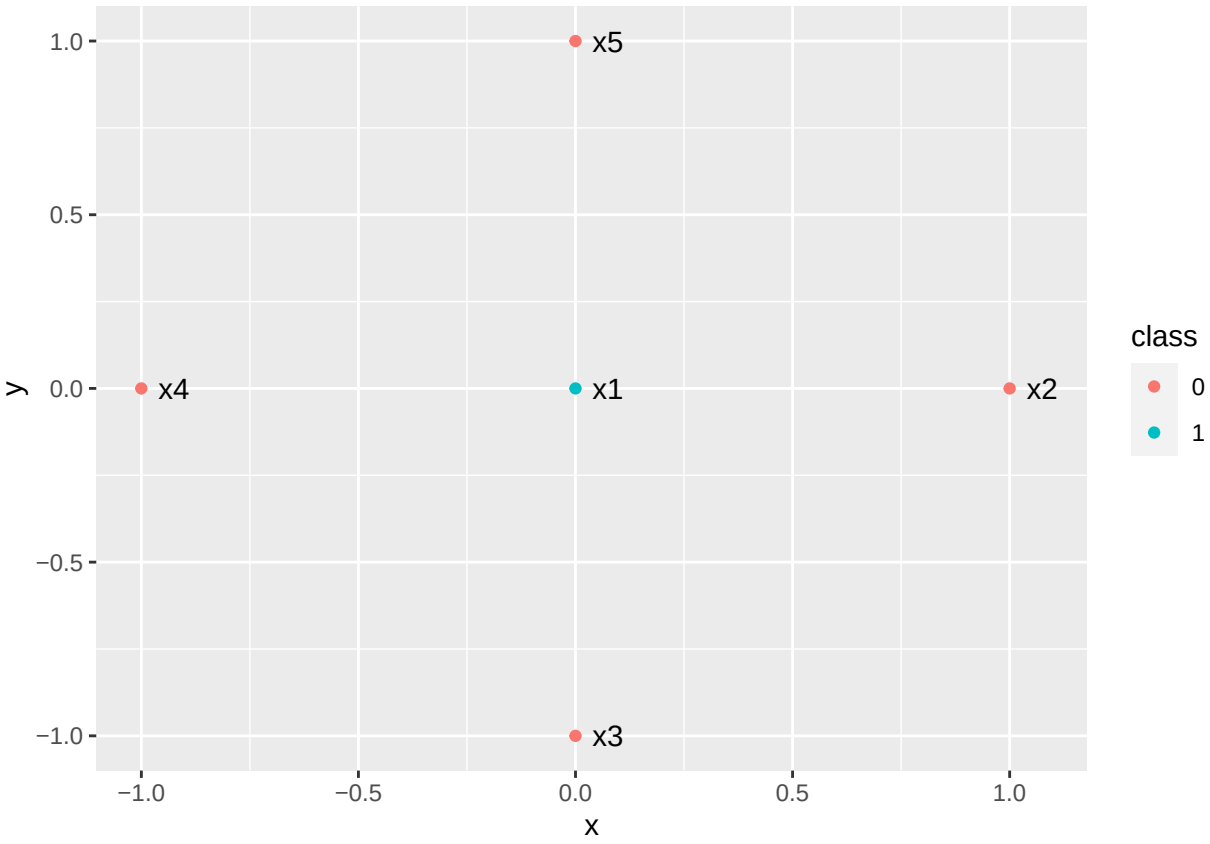
Consider the same logistic regression model above with coefficients β_0, β_1 and β_2, how would you need to set these coefficients to correctly classify this dataset?

```
b0 = (
  # TODO fill in the value for beta 0
)

b1 = (
  # TODO fill in the value for beta 1
)

b2 = (
  # TODO fill in the value for beta 2
)

sigmoid(xs %*% c(b0, b1, b2))
```

Make sure that the second and third elements are close to one, and the others close to zero.

Exercise 3

Finally, with the same data as before, suppose that $y_1 = 1$ and $y_2 = y_3 = y_4 = y_5 = 0$:



Clearly, logistic regression cannot correctly classify this dataset, since the two classes are not linearly separable (optional: prove it).

However, as we have shown in the previous exercises, it is possible to separate x_2 and x_3 from the rest, and x_4 and x_5 from the rest.

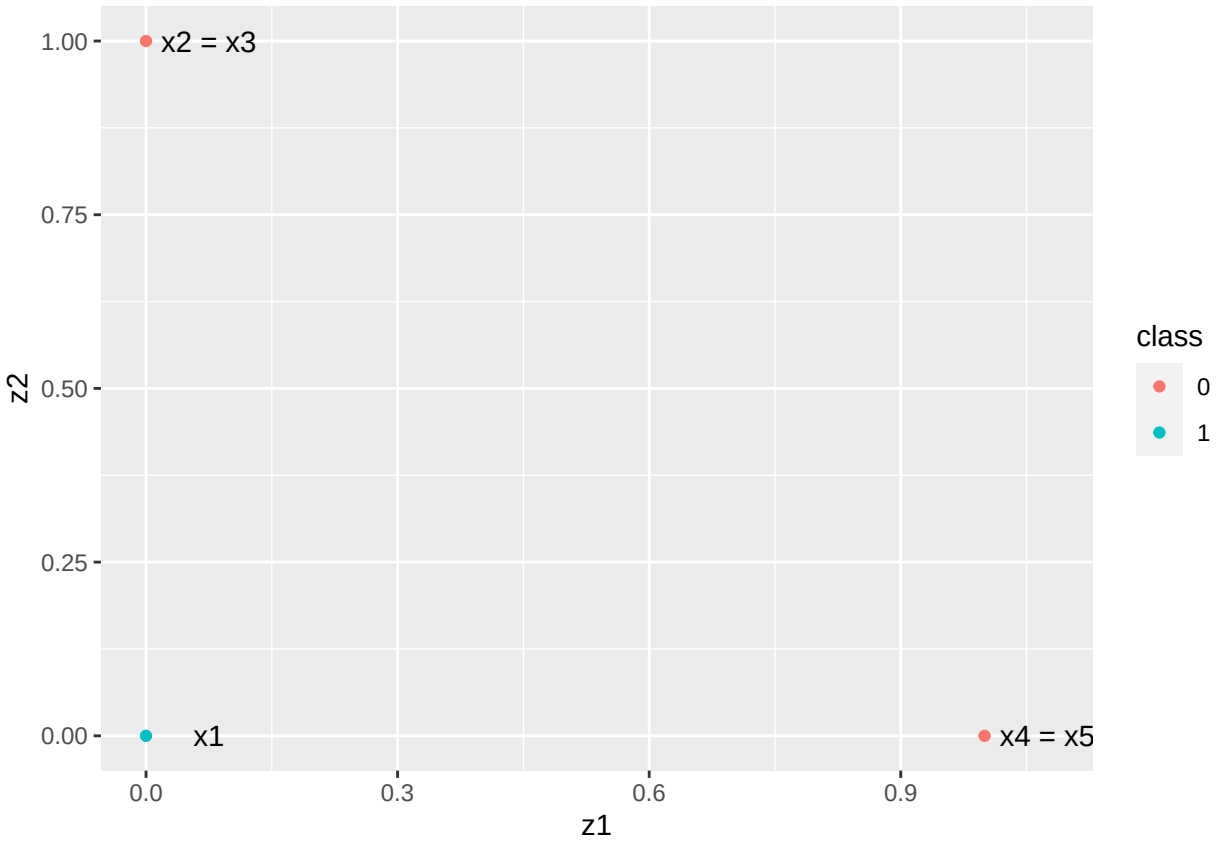
Can these two simple classifiers be composed into one that is powerful enough to separate x_1 from the rest?

Can we use their predictions as input for another logistic regression classifier?

Let $z_{i1} = \sigma(\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2})$ and $z_{i2} = \sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2})$ be the output of the two logistic regression classifiers for point i. Then, the dataset would become:

i	z_{i1}	z_{i2}	y
1	0	0	1
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0

In graphical form:



This sure looks linearly separable! As before, find the coefficients for a linear classifier $\hat{y}_i = \sigma(\gamma_0 + \gamma_1 z_{i1} + \gamma_2 z_{i2})$:

```
g0 = (
  # TODO fill in the value for gamma 0
)

g1 = (
  # TODO fill in the value for gamma 1
)

g2 = (
  # TODO fill in the value for gamma 2
)

zs = matrix(c(
  1, 0, 0,
  1, 0, 1,
  1, 0, 1,
  1, 1, 0,
  1, 1, 0
), ncol = 3, byrow = T)

sigmoid(zs %*% c(g0, g1, g2))
```

Make sure that the first element is close to one, and the others close to zero.

This big classifier can be summarized as follows:

```
z1 = sigmoid(xs %*% c(a0, a1, a2))
z2 = sigmoid(xs %*% c(b0, b1, b2))

yhat = sigmoid(g0 + g1 * z1 + g2 * z2)
yhat
```

And this is just what a neural network looks like! Each neuron is a simple linear classifier, and we just stack linear classifiers on top of linear classifiers. And we could go on and on, with many layers of linear classifiers.