Exercise 1: Logistic regression or deep learning?

Suppose you a set of five training points from two classes. Consider a logistic regression model $f(\mathbf{x}) = \sigma\left(\boldsymbol{\alpha}^\top \mathbf{x}\right) = \sigma(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2)$, with $\sigma(\cdot)$ the logistic/sigmoid function, $\sigma(c) = \frac{1}{1+\exp(-c)}$.

a) Which values for $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2)^\top$ would result in correct classification for the problem in Fig. 1 (assuming a threshold of 0.5 for the positive class)? Don't use any statistical estimation to answer this question – think in geometrical terms: you need a linear hyperplane that represents a suitable decision boundary.

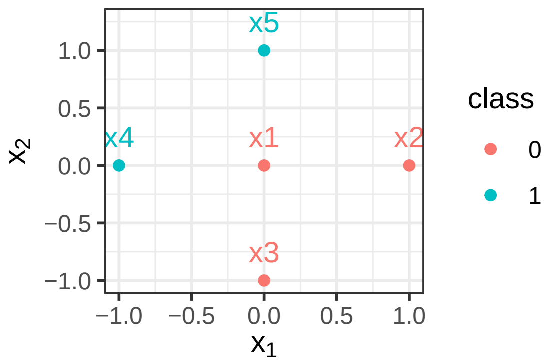

Figure 1: Classification problem I

b) Apply the same principle to find the parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)^\top$ for the modified problem in Fig. 2.

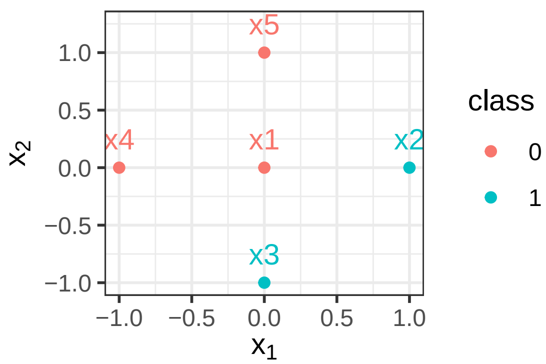

Figure 2: Classification problem II

c) Now consider the problem in Fig. 3, which is not linearly separable anymore, so logistic regression will not help us any longer. Suppose we had alternative coordinates $(z_1, z_2)^\top$ for our data points:

i	$z_1^{(i)}$	$z_2^{(i)}$	$y^{(i)}$
1	0	0	1
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0

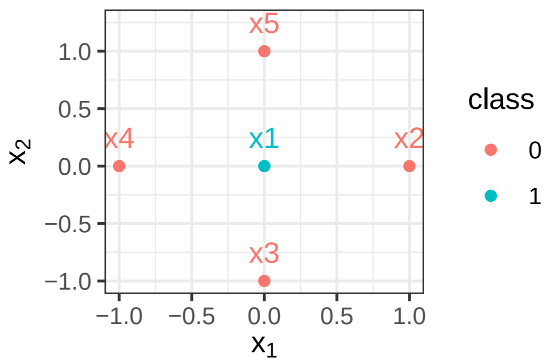

Figure 3: Classification problem III

i) Explain how we can use z_1 and z_2 to classify the dataset in Fig. 3.

ii) The question is now, of course, how we can get these z_1 and z_2 that provide a solution to our previously unsolved problem – naturally, from the data.
Perform logistic regression to predict z_1 and z_2 (separately), treating them as target labels to the original observations with coordinates $(x_1, x_2)^\top$. Find the respective parameter vectors $\boldsymbol{\gamma}, \boldsymbol{\phi} \in \mathbb{R}^3$.

iii) Lastly, put together your previous results to formulate a model that predicts the original target y from the original features $(x_1, x_2)^\top$.

d) Sketch the neural network you just created (perhaps without realizing it).

e) Explain briefly how the chain rule is applied to the computational graph such a neural network represents. Can you think of a use we can put the resulting gradients to?