Learning Nonlinearities

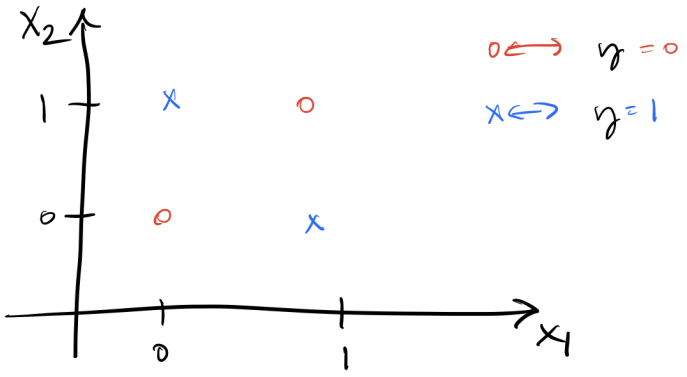
Introduction

Consider the XOR operator. In plain language, $y = \text{XOR}(x_1, x_2)$ lights up when exactly one of x_1 or x_2 is true, but not both. That is, if we have $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1\}$, $y = \text{XOR}(x_1, x_2)$ takes its value according to:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

The Classification Problem

Since $y \in \{0, 1\}$, we can think of this setup in terms of a classification problem with the predictors x_1 and x_2.

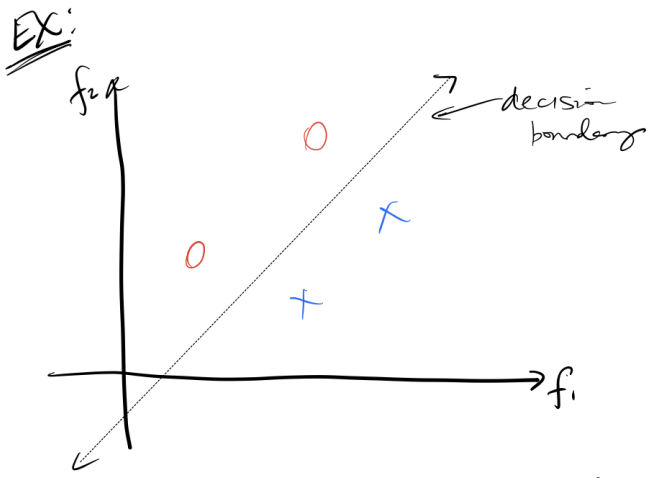


The big problem:

We cannot come up with a decision boundary that is linear in x_1 and x_2 which separates the classes!!

Feature Engineering

One idea is to try feature engineering. That is, construct $f_1 = \phi_1(x_1, x_2)$ and $f_2 = \phi_2(x_1, x_2)$ such that the classes are linearly separable in the space spanned by (f_1, f_2). We want to end up with something like:



But how should we pick $\phi_1(\cdot, \cdot)$ and $\phi_2(\cdot, \cdot)$?

Since this example is really simple, we can just think a bit and arrive through trial and error at some good choices.

Example

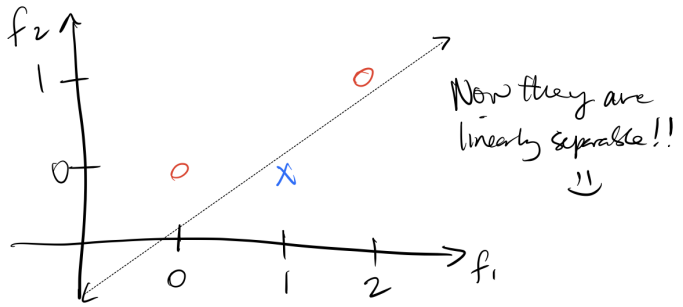
For example, take:

$$f_1 = \phi_1(x_1, x_2) = \max(x_1 + x_2, 0)$$
$$f_2 = \phi_2(x_1, x_2) = \max(x_1 + x_2 - 1, 0)$$

Then we have:

x_1	x_2	f_1	f_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

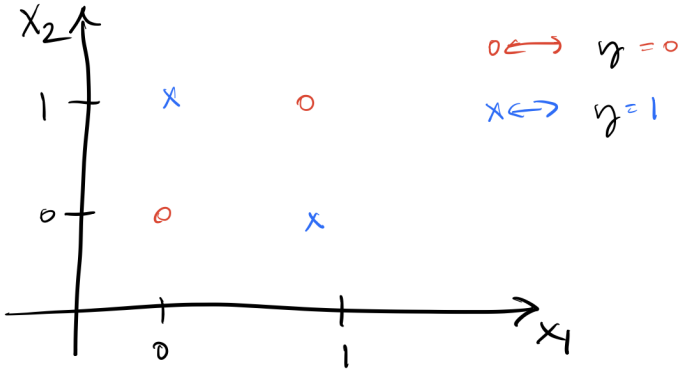
Visually, now we have:



Now they are linearly separable!! :)

Connection to Neural Networks/Representation Learning

This example of feature engineering also illustrates how neural networks work! Here is the idea:



If we take $W = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$ and the ReLU activation function,

$$\sigma(x) = ReLU(x) = x\mathbf{1}_{\{x>0\}}(x) = \max(x, 0)$$

then the constructed features are:

$$f_1 = \sigma(w_{10} + w_{11}x_1 + w_{12}x_2) = (x_1 + x_2)_+ = \phi_1(x_1, x_2)$$
$$f_2 = \sigma(w_{20} + w_{21}x_1 + w_{22}x_2) = (x_1 + x_2 - 1)_+ = \phi_2(x_1, x_2)$$

The last step is just to train a logistic regression on them and learn the $\hat{\beta}$ vector. Knowing the $\hat{\beta}$ coefficients is equivalent to knowing the decision boundary!

A Couple Things to Note

1. This problem was simple enough so that we could just guess the values of W. In general, we will have to learn them!

2. Sometimes classes are not linearly separable because of noise, sometimes because of a true structural nonlinearity (or both). In this case, we observe the values of $y = \text{XOR}(x_1, x_2)$ without noise, so we know it is a true structural nonlinearity. For structural nonlinearities, neural networks can help a lot; they can't do much about the noise problem, though.

3. The concept that the neural network is trying to learn how to represent the features f is one of the big ideas of neural networks. Sometimes you'll hear this referred to as "representation learning."