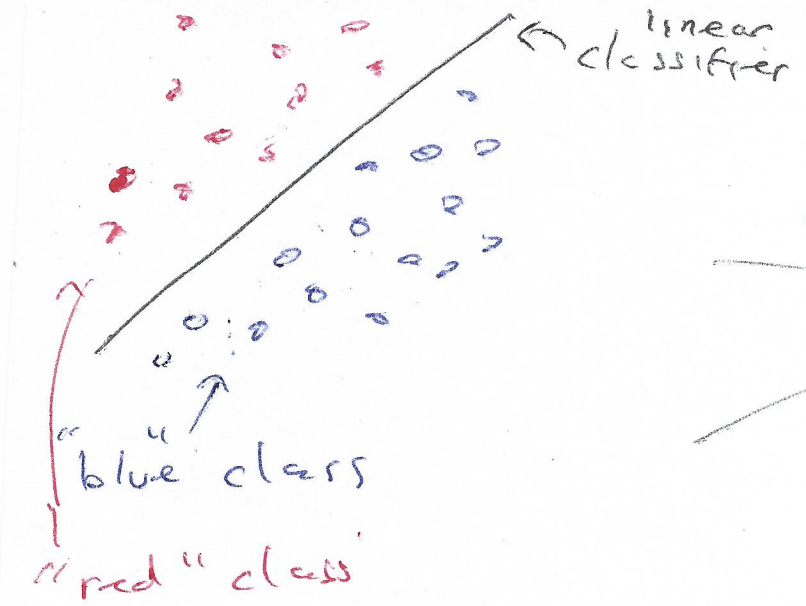
Linear Classification : (2 class)

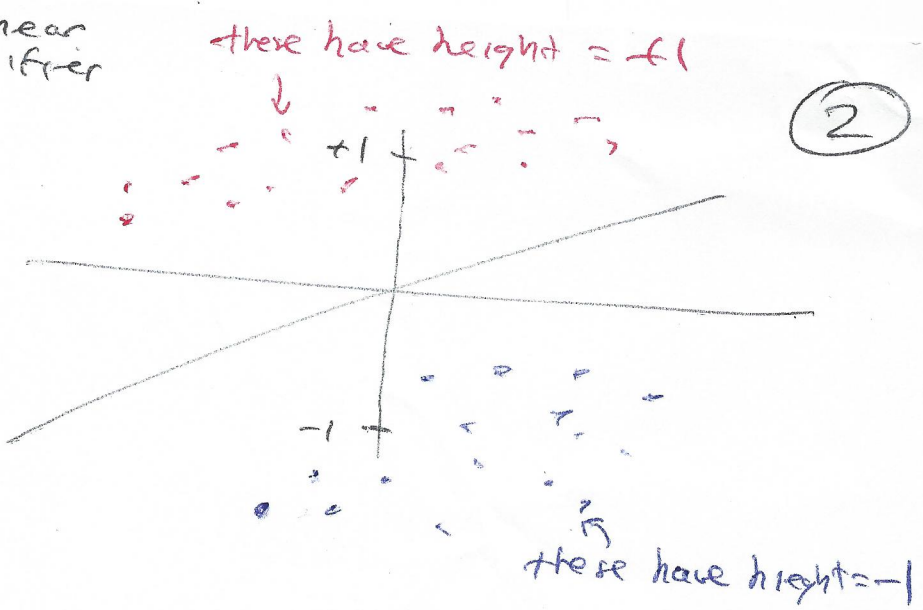
- How do we distinguish between classes of data? E.g., faces from non-faces in digital images, spam vs. non-spam messages → thats classification
- We begin by formalizing 2-class classificatin, multiclass will follow.
- More specifically, we begin with logistic regression → very popular method

Logistic regression → 2-class :

- From the level 1 notebook we saw how 2-class classification can be viewed in 2 ways :

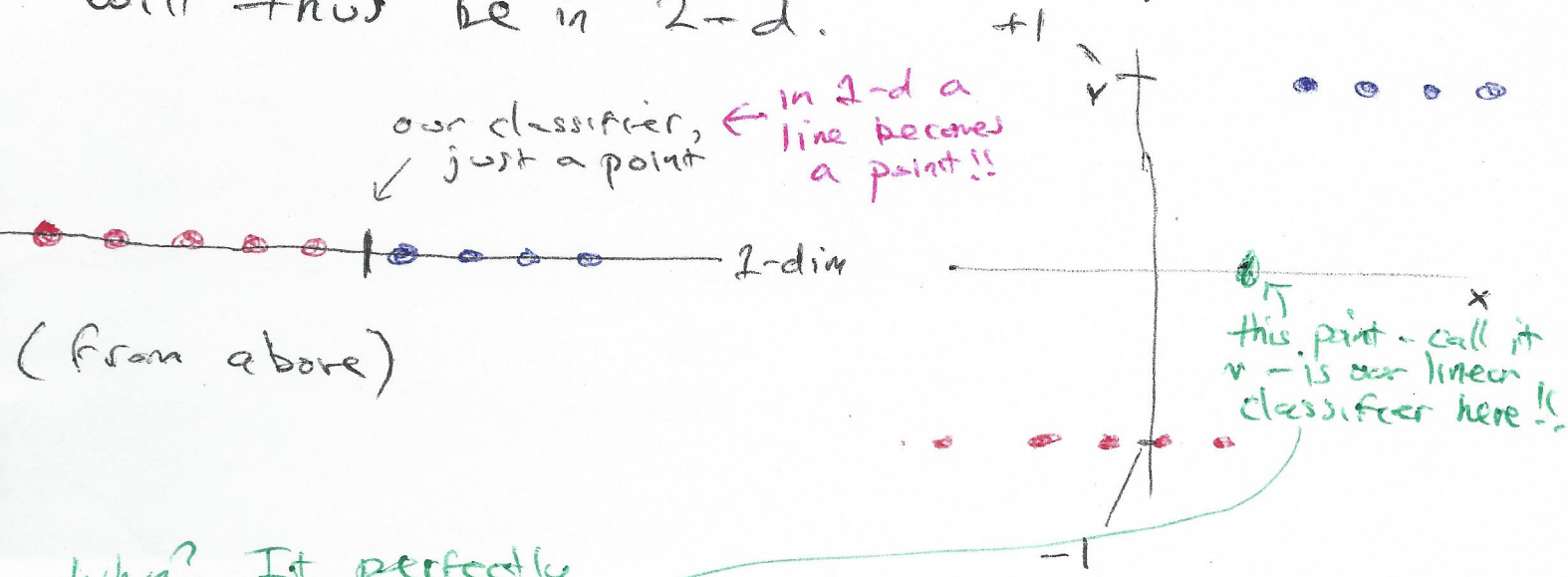


(From above)



(From the side, as a regression problem)

- Lets drop off an input dimension, and first tinker with a simpler setup. So instead of the "from above" view being in 2-d it will be 1-d, and the regression view will thus be in 2-d.



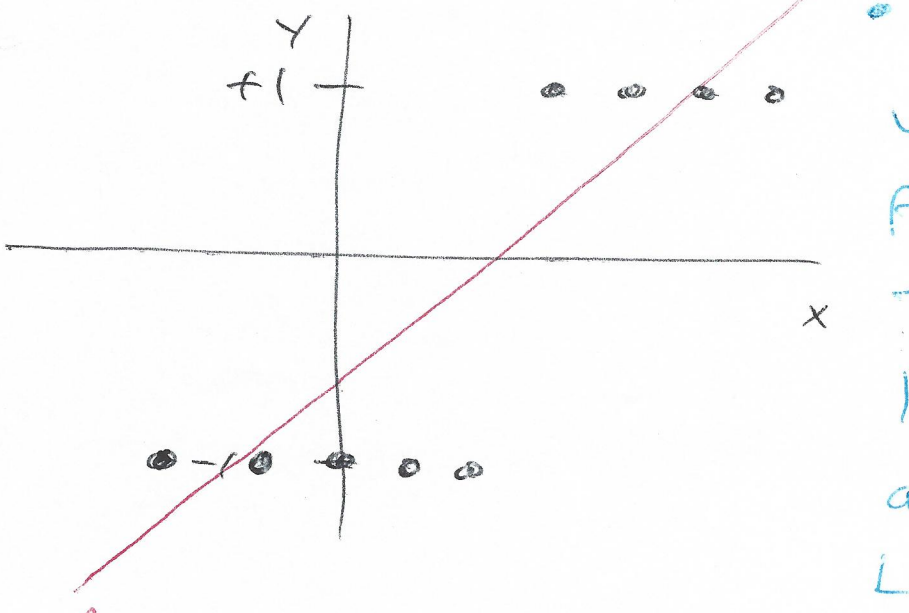
(From above)

(From the side, as a regression problem)

Why? It perfectly seperates the 2 classes. If $x_p < v$ then its output y_p is red
If $x_p > v$ then its output y_p is blue

- Lets stick with the "regression perspective" for a while, and see what we can conjur up.

- Lets remove the colors from our data, they're not real anyway, and are used just to distinguish the output values of the points when we view the problem "from above"



our linear regression fit.

- As a regression problem, we can go ahead and fit a line to this! Treating it as a linear regression problem, and minimizing the Least Squares cost, etc.,

- OK, whats the problem(s) with doing this?

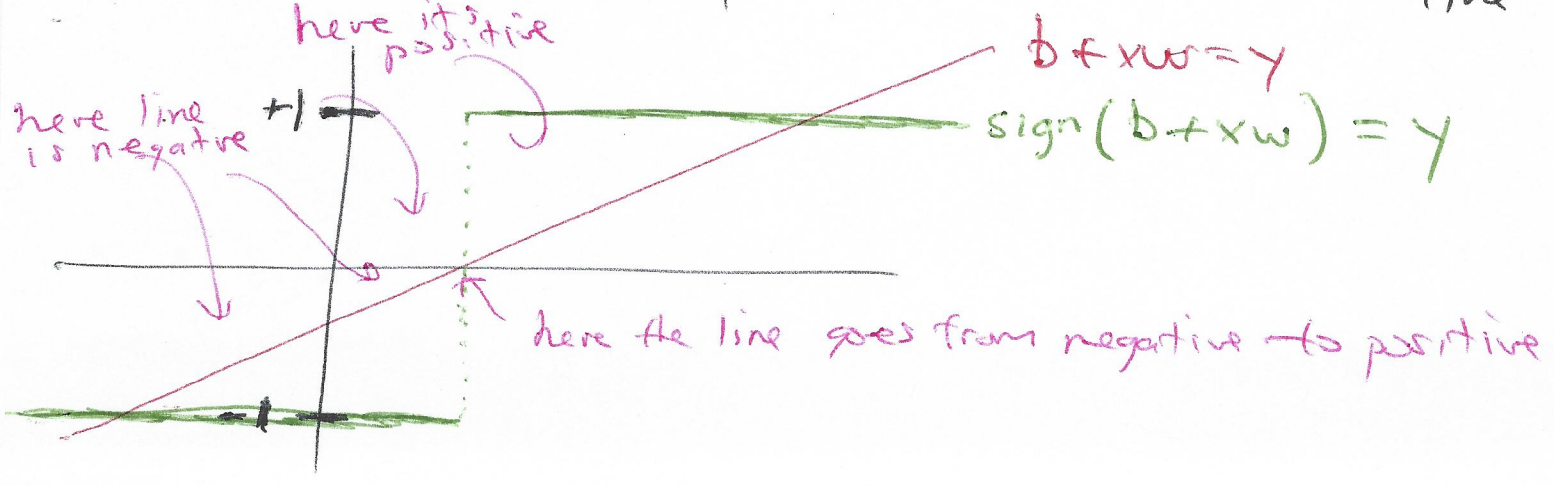
- In short, the data is clearly not distributed like a line, so a line is a bad model

- How about this then: since the output only takes on the values -1 or $+1$, we shove the line through the sign $(\cdot)$ function.

- $\text{sign}(\beta) = \begin{cases} +1 & \beta \geq 0 \\ -1 & \beta < 0 \end{cases}$

You just keep the numerical "sign" of the input.

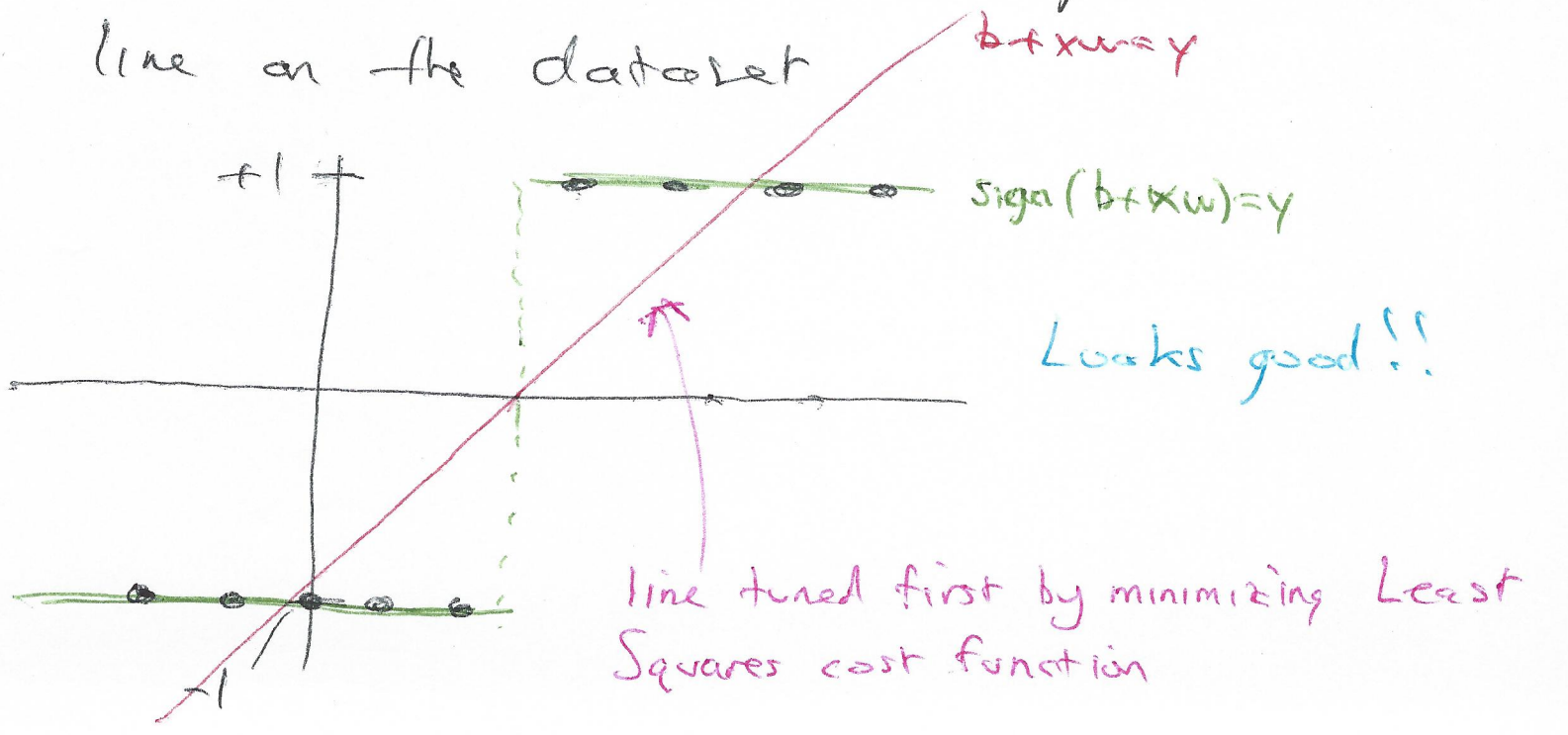
- Lets first apply sign$(\cdot)$ to a test line



- Ahhh, now sign(b+xw) looks a hell of a lot better, it only takes on the values −1, +1 → just like the data. Plus it "steps up" from −1 to +1 just like our ~~to~~ data.

- because of this "step up" the function sign(b+xw)=y is often referred to as a "step function"

- Lets do it to our linear-regression determined line on the data. Let



Can this work in general for any (nice) linear classification dataset? that is where the points from each class truly are seperated by a line (a point in our 1-d example)?

Lets do some experiments on a few toy datasets where we first fit a line to the data via linear regression (using gradient descent to minimize) followed by sign(·) the output best fit line.

To-do:

Get two 2-d logistic regression datasets

2d_logistic_data_v1.csv
2d_logistic_data_v2.csv

Do the following for each dataset

1) Use your linear ~~Regression~~ best fit functionality to fit a line to ~~the~~ data

2) Say your best fit line is given as

$$w_0 + w_1 x = y$$

Plot the sign(·) of this to your data - i.e. plot

$$\text{sign}(w_0 + w_1 x) = y$$

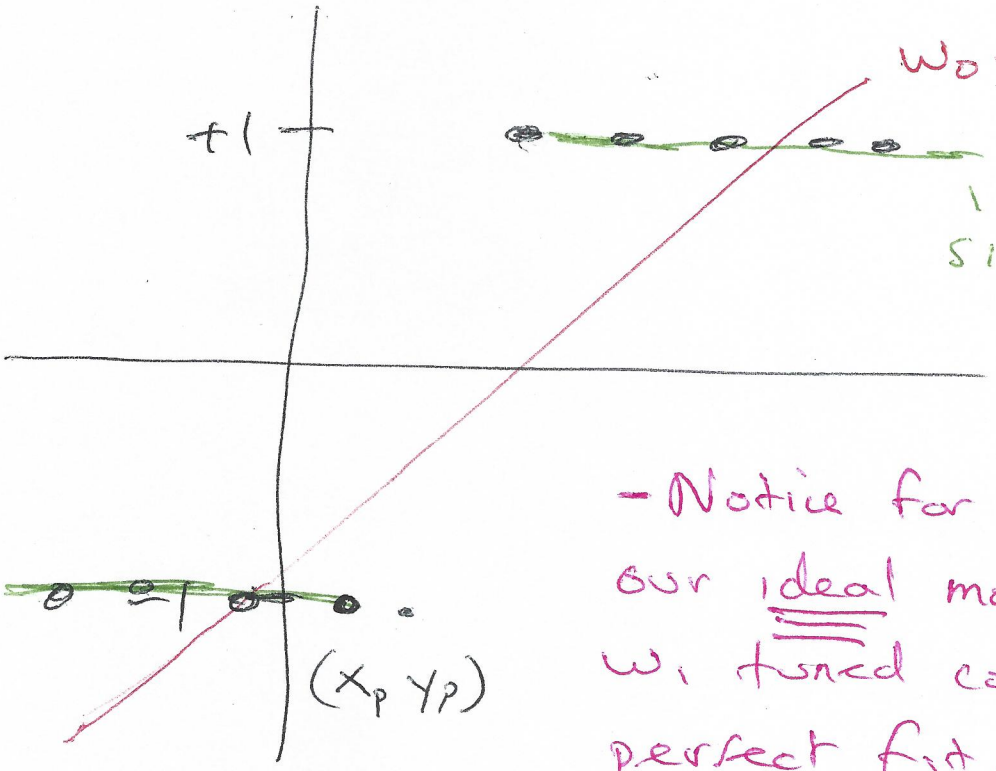
to see how well you fit to each dataset!

Looks like the first one works great, but the second one not so much!

-So it looks we can't take the easy way out! That is, we cannot just fit with linear regression and shove the resulting line through sign

---

- What can we do instead?



$\text{sign}(w_0 + w_1 x) = y$

- Notice for each point (x_p, y_p) our ideal model - if w_0 and w_1 tuned correctly - gives perfect fit, so

$$\text{sign}(w_0 + w_1 x_p) = y_p$$

ideal

- Using this ideal we can work backwards, and design a function that — cost

- Notice how the statement

$$\text{sign}(w_0 + w_1 x_p) = y_p$$

is the same as - since $y_p = \pm 1$ - if we multiply both sides by y_p

$$y_p \, \text{sign}(w_0 + w_1 x_p) = 1$$

- Notice also we can move the y_p inside of sign! So the above is

$$\text{sign}(y_p(w_0 + w_1 x_p)) = 1$$

- Alright! So this is true if the wieghts w_0, w_1 are tuned such that the prediction is correct. More generally

$$\text{sign}(y_p(w_0 + w_1 x_p)) = \begin{cases} +1 & \text{if prediction correct} \\ -1 & \text{if prediction incorrect} \end{cases}$$