

Predictive Machine Learning

Classification & Regression

These types of algorithms take two inputs

$X \rightarrow$ The features of different samples

$Y \rightarrow$ The value of a target quantity for each sample

The goal is to output a **Model** that can predict the target quantity from features of the sample.



In this section, we will briefly talk about some of the well-known algorithms for this type of machine learning.

But before that, let's talk about Regression & classification and their differences.

For reg. Y is a (bunch of) continuous variable : $Y \in \mathbb{R}$ or $Y \in \mathbb{R}^n$.

But for classification Y is discrete and categorical.

It means that we want the model to be able to determine the type of something.

Examples

Reg.

- Prediction of temperature from Black-Body radiation.
- Prediction of correlation in a state.

Clf.

- Is a state entangled?
- Is an object a conductor?
- Is a news fake?

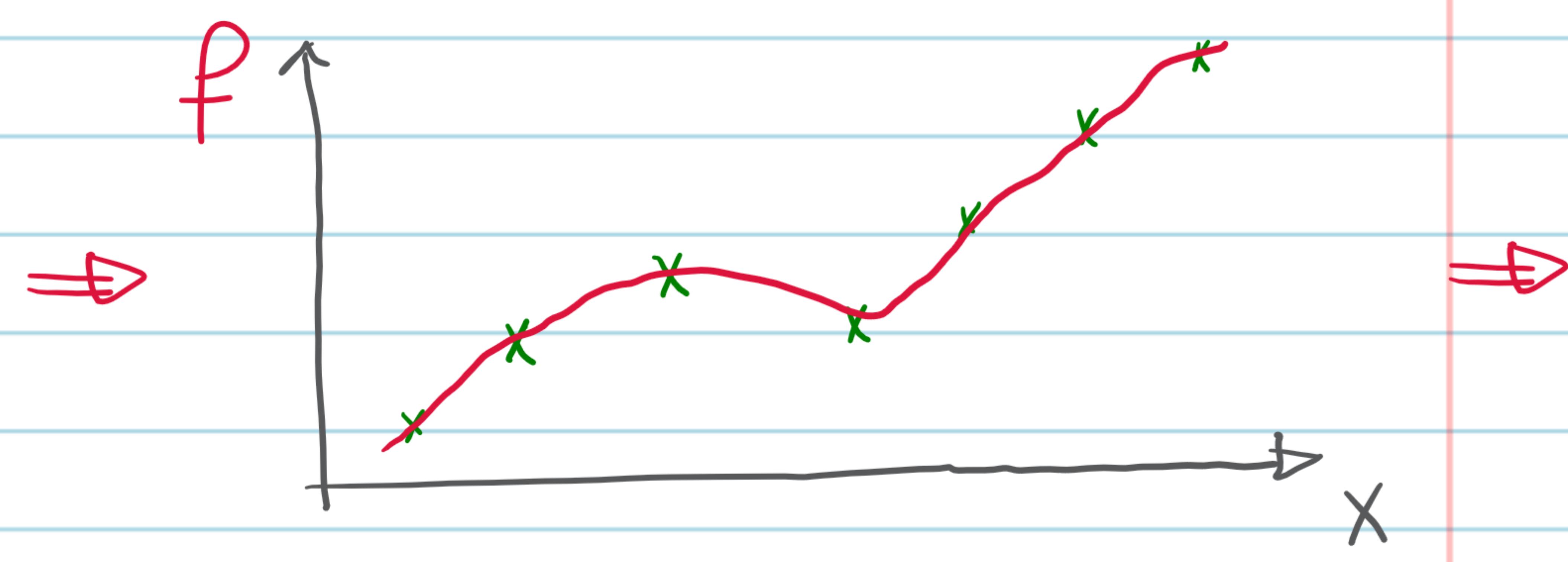
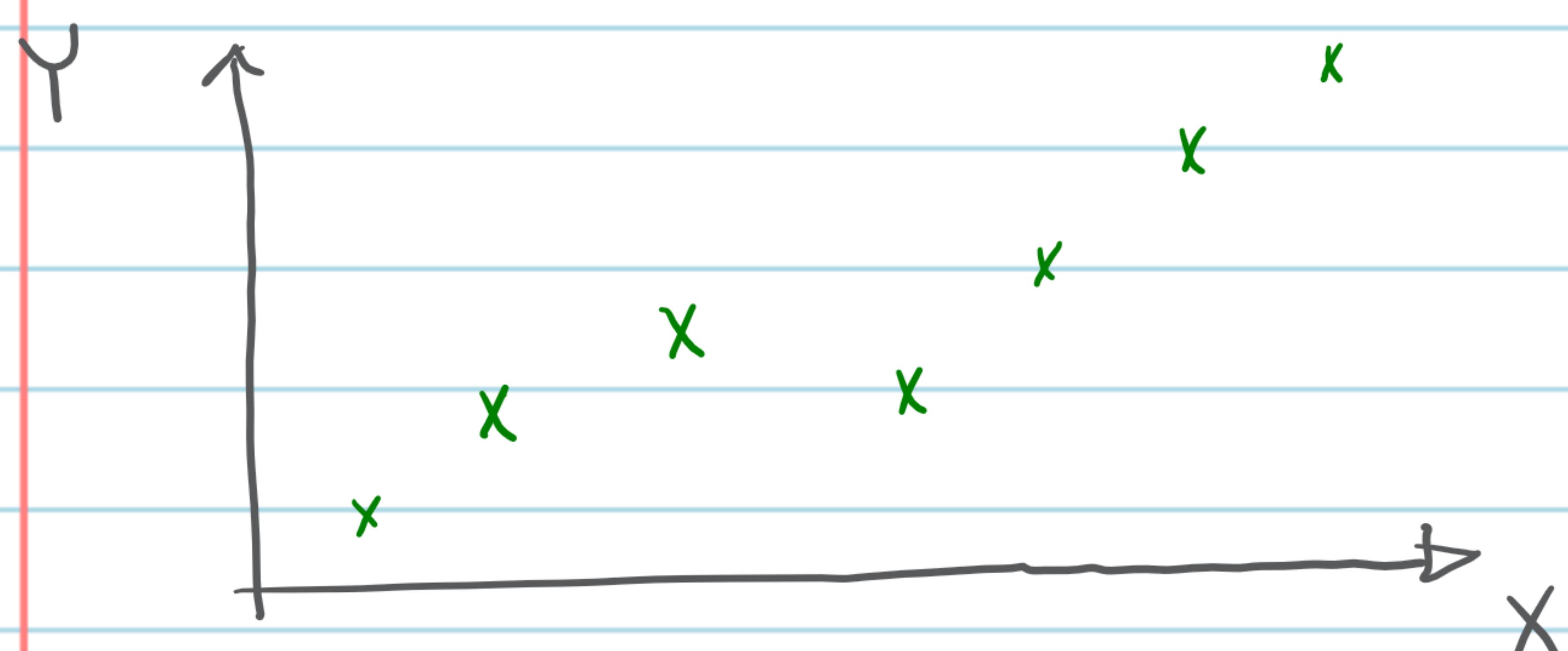
Reg. $\vec{X}^{(1)} = (x_1^1, x_2^1, \dots, x_{n_f}^1) \rightarrow Y^{(1)}$

$$\vec{X}^{(2)} \rightarrow Y^{(2)}$$

$$\vec{X}^{(n_s)} \rightarrow Y^{(n_s)}$$

$\Rightarrow \text{Model} = f : \underline{Y^{(i)} \simeq f(\vec{X}^{(i)})}$

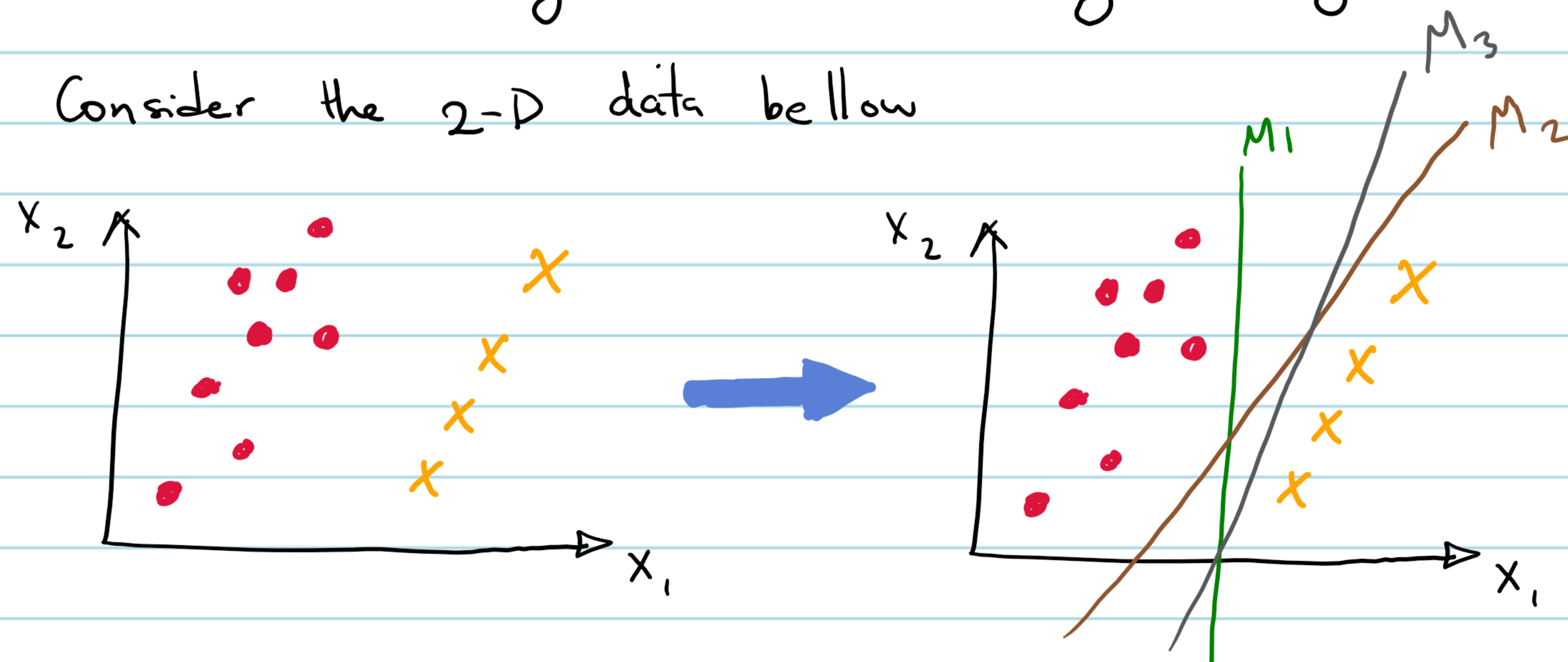
2-D example: $n_f = 1$



Classification (CLF)

This has similarity to both Clustering & Reg.

Consider the 2-D data below



The idea is to find a model that, based on the value of X , can predict whether $y = X$ or $y = \bullet$.

Coming from Reg., one approach could be to find a function f such that:

$$f(\vec{x}) > 0 \rightarrow \times$$

$$f(\vec{x}) < 0 \rightarrow \bullet$$

$\underline{f(x)=0 \rightarrow \text{Decision Boundary}}$

There are other ways that we'll talk about.

Polynomial models

Probably the simplest models we can consider are Poly(n) functions.

We talked about this in the introduction.

For regression

$$f(\vec{x}) = \vec{X} \cdot \vec{W} = w_1 x_1 + w_2 x_2 + \dots$$

$$x: [n_s, n_p] \quad \vec{W}: [n_p, 1]$$

$$\vec{X} \cdot \vec{W} = \vec{Y} \Rightarrow \underbrace{\vec{X}^\top \cdot \vec{X}}_{\vec{X}^\top \cdot \vec{X}} \cdot \vec{W} = \vec{X}^\top \cdot \vec{Y}$$

$$\vec{W} = (\vec{X}^\top \cdot \vec{X})^{-1} \cdot \vec{X}^\top \cdot \vec{Y} \rightarrow ?$$

For Poly(n) we can take \vec{X} & generate

$$\tilde{X} = [\vec{x}^n, \vec{x}^{n-1}, \dots, \vec{x}, 1].$$

Note that if $n_s < n_p$ or $n_s < \frac{n_p \times n}{\text{w/o cross terms}}$ this
(linear)

would not give good results.

The solution

$$\vec{W} = (\vec{X}^\top \cdot \vec{X})^{-1} \cdot \vec{X}^\top \cdot \vec{Y}, \text{ while simple has a big flaw.}$$

Consider a noise data & a poly ($n \gg$)

This model starts to fit the noise. See the nb.

But why is that a bad thing?

The trend is the general pattern & extends to other samples

→ Prediction Power

but not the noise.

What are our expectations from a good model?

* To have accurate prediction

* To generalize well.

* Is easy to train.

* Is easy to use.

We will go through this list over the next few sessions and specify them.

To start with, we need to have a notion of accuracy.

$$\text{i.e. } d(Y, \tilde{Y}) = 0$$

Some examples : $d(Y, \tilde{Y}) = \frac{1}{n_s} \sum_i (Y_i - \tilde{Y}_i)^2$

$$d(Y, \tilde{Y}) = \frac{1}{n_s} \sum_i |Y_i - \tilde{Y}_i|$$

$$d(Y, \tilde{Y}) = \dots$$

Based on the first expectation, to train a model

means to tune the parameters of the model such that it would minimizes $d(Y, \tilde{Y})$.

This applies to both Reg. & Clf.

Other techniques

We'll study some other techniques for classification.

But most of these techniques can be used for reg. too.

Logistic Reg.

$$f(\vec{x}) = \text{Sig}(\vec{x} \cdot \vec{w}) = \text{Sig}(w_0 + w_1 x_1 + w_2 x_2 + \dots)$$

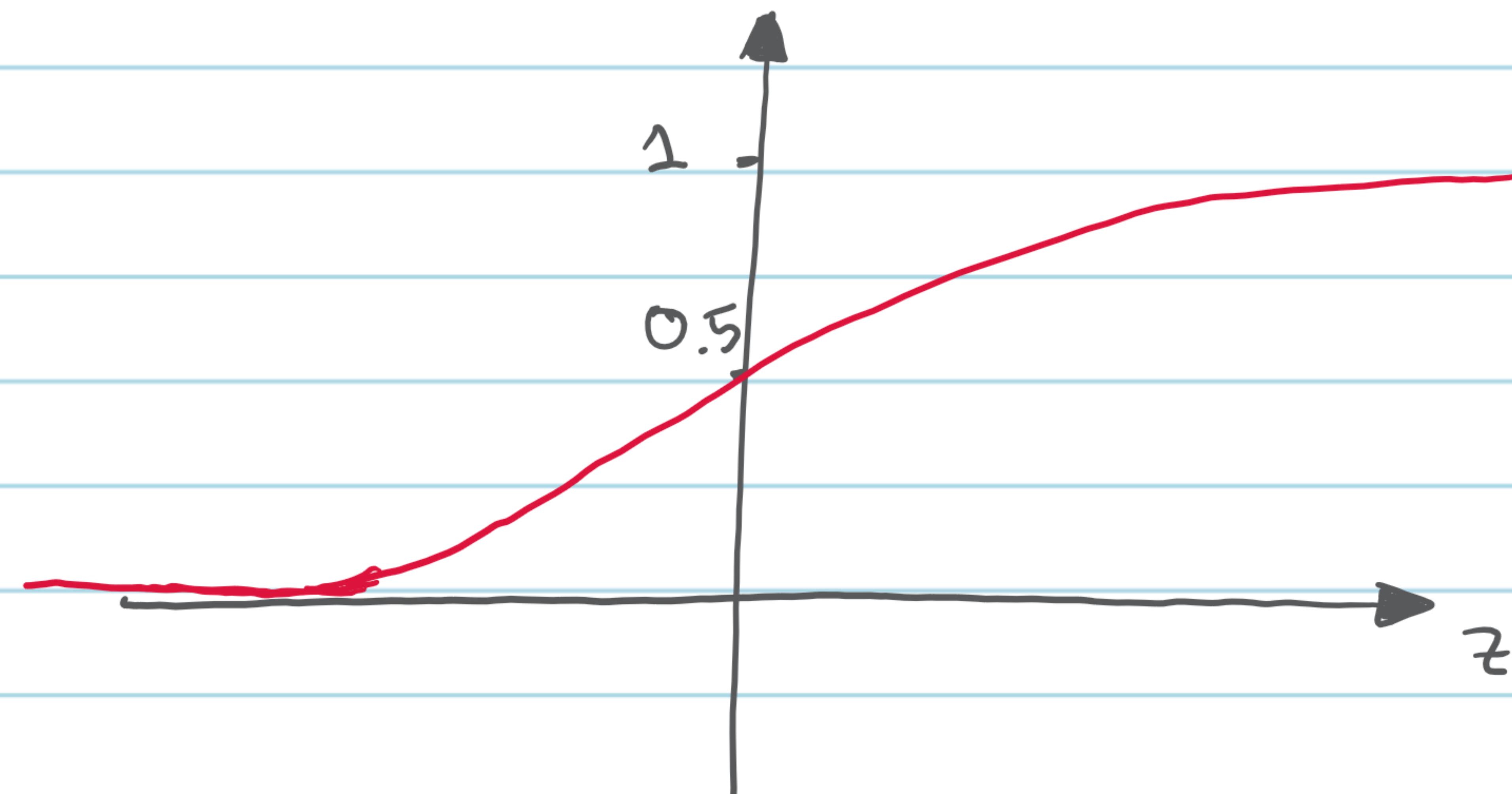
Sigmoid

$$\sigma(z) = \frac{e^z}{1 + e^z}$$



Gibbs distribution,
probability of

activation / excitation



This is giving a probability for the outcome.

If $\Pr \geq 1/2 \rightarrow \text{Class } 1$

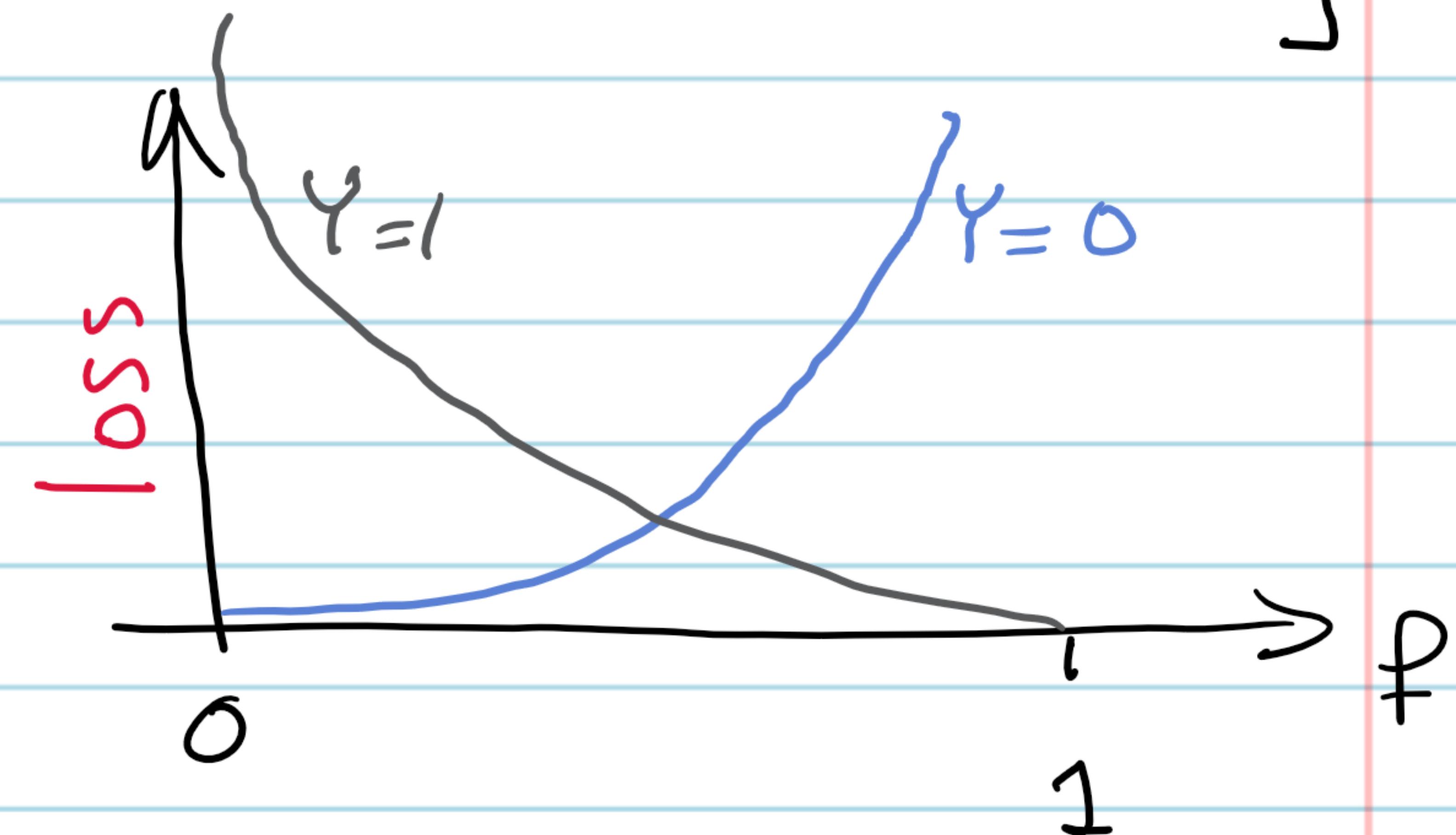
$\Pr < 1/2 \rightarrow \text{Class } 0$

For LR we usually use cross-entropy for the loss function:

$$\ell(x, y) = \frac{-1}{n_s} \sum_i \left[y_i \log(f(x_i)) + (1-y_i) \log(1-f(x_i)) \right]$$

$$y_i = 1 \rightarrow \frac{-1}{n_s} \log(f(x_i))$$

$$y_i = 0 \rightarrow \frac{-1}{n_s} \log(1-f(x_i))$$



Question: What's the advantage of using the cross-entropy as a loss for LR?

SVM

Support Vector machines (SVM) have some similarities

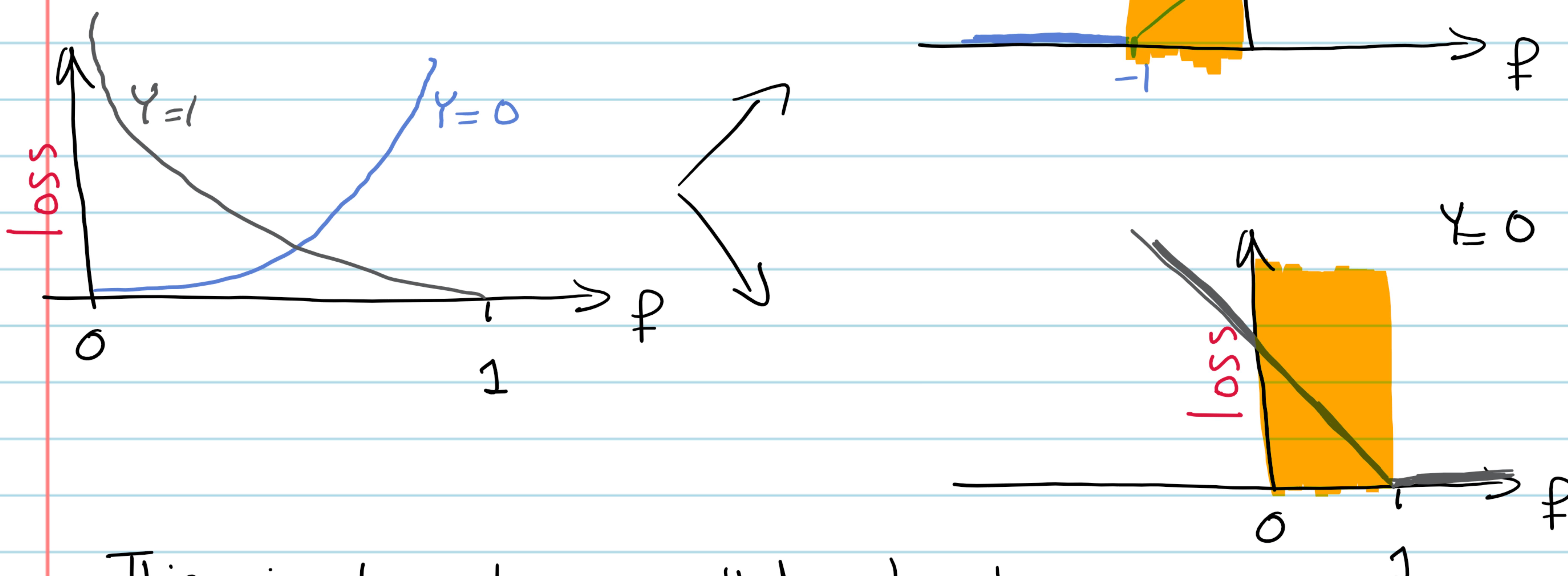
to LR. It is a linear model, i.e.

$$f(x) = \vec{x} \cdot \vec{w} = w_0 + w_1 x_1 + w_2 x_2 + \dots$$

but for the loss, we use something similar to the cross-entropy:

$$f(x) \geq 0 \rightarrow \text{class 0}$$

$$f(x) < 0 \rightarrow \text{class 1}$$



This is to make sure that not only $f(x) > 0$ gives the class, but it is the most confident model that does this. \rightarrow Max margin.

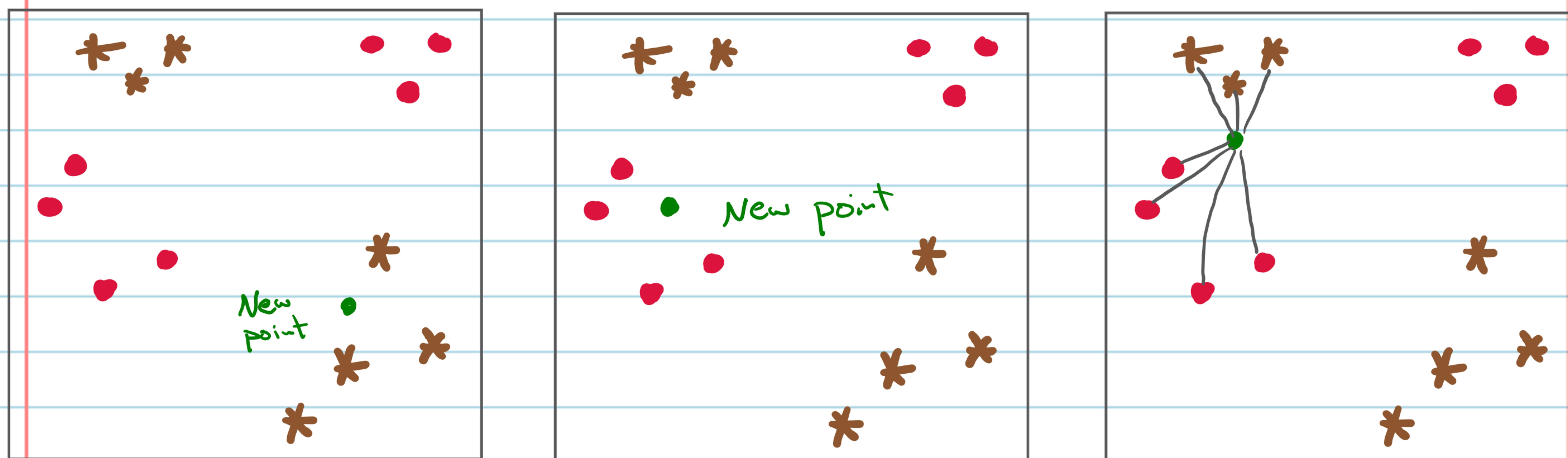
This is known as the "Hinge" function:

$$l_{\text{Hinge}}(x, y) = \max(0, 1 - \vec{y} \cdot \vec{f}(x))$$

KNN

This is an "instance-based" algorithm.

Basically, it memorizes all the samples & given a new sample, it looks at the k closest neighbors of the new sample.



*

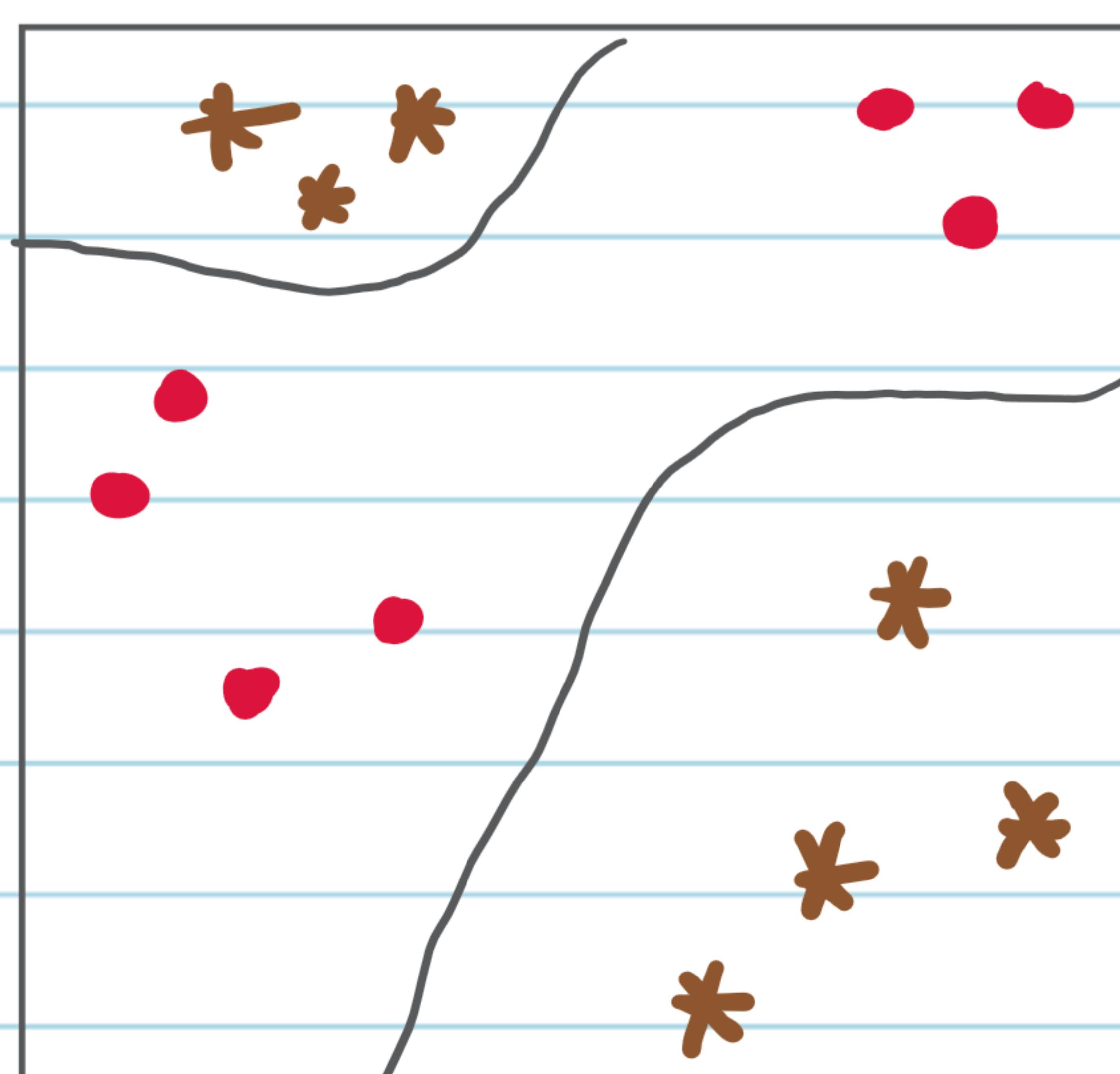
*

$k=1$ \rightarrow *

$k=3 \rightarrow$ ● or *

$k=7 \rightarrow$ ●

Remark: The decision boundary would depend on k and the algorithm for voting.



* Fast to train

* Slower to predict

* Requires a metric $d(\vec{x}_1, \vec{x}_2)$

Question: Can you think of a situation where majority voting is not good?