Part 1: Neural Networks & Logic Gates

Understand how neural networks can be used to approximate boolean functions like AND, OR, XOR.

Write the weights of a neural network that can compute each of these boolean functions.

Understand why the ability to represent boolean functions is important, and what it shows about the way neural networks represent or approximate functions.

Understand the concept of latent space.

Part 2: Classification Metrics

Describe, calculate, and interpret classification metrics including the confusion matrix, accuracy, precision, recall, F1 score, ROC curve, PR curve, AUROC, and AUPR.

Compare classification metrics; understand strengths & weaknesses of each; understand when to apply each metric.

Part 1: Neural Networks & Logic Gates

Understand how neural networks can be used to approximate boolean functions like AND, OR, XOR.

Write the weights of a neural network that can compute each of these boolean functions.

Understand why the ability to represent boolean functions is important, and what it shows about the way neural networks represent or approximate functions.

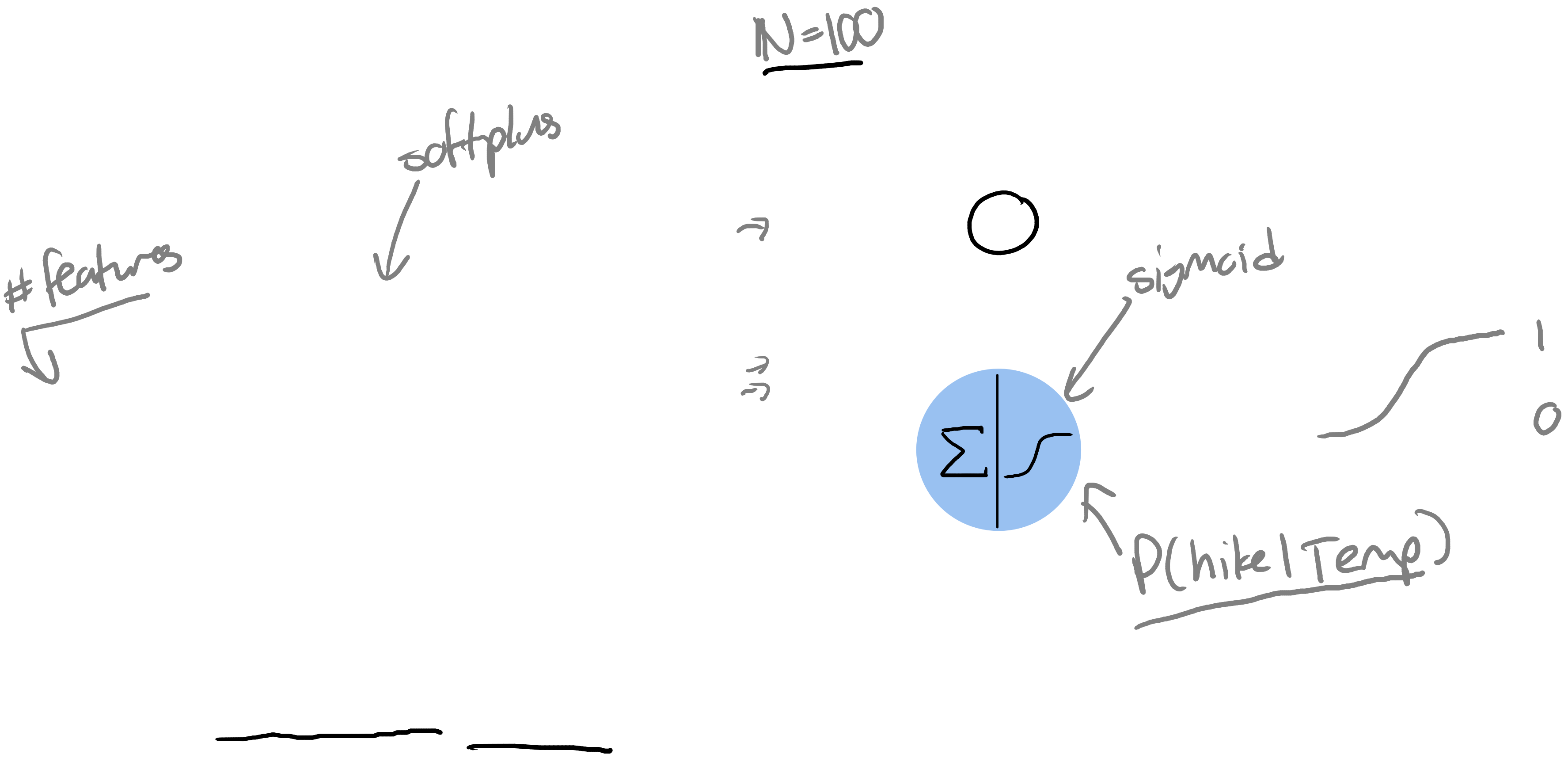
Understand the concept of latent space.

Part 2: Classification Metrics

Describe, calculate, and interpret classification metrics including the confusion matrix, accuracy, precision, recall, F1 score, ROC curve, PR curve, AUROC, and AUPR.

Compare classification metrics; understand strengths & weaknesses of each; understand when to apply each metric.

N=100



N=100
softplus
features
sigmoid
1
0
P(hike|Temp)

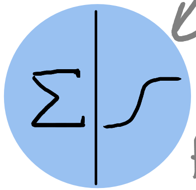
softplus

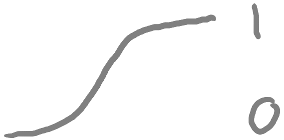
features

softplus(z)

$z = w^T x$

sigmoid





P(hike|Temp)

X $\quad$ Y

Temperature	Hike
14.2	0
10.9	0
13.1	0
25.6	1
24.8	1
38	0
40	0
32	0.5
33.2	0.4
16	0.3

softplus(z)

$z = w^T x$

Min-Max Scale the Temperature feature

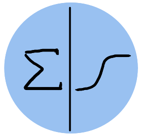
X_{sc} $\quad$ Y

Temperature	Hike
0.16	0
0.0	0
0.07560137	0
0.50515464	1
0.47766323	1
0.93127148	0
1.0	0
0.72508591	0.5
0.76632302	0.4
0.17525773	0.3

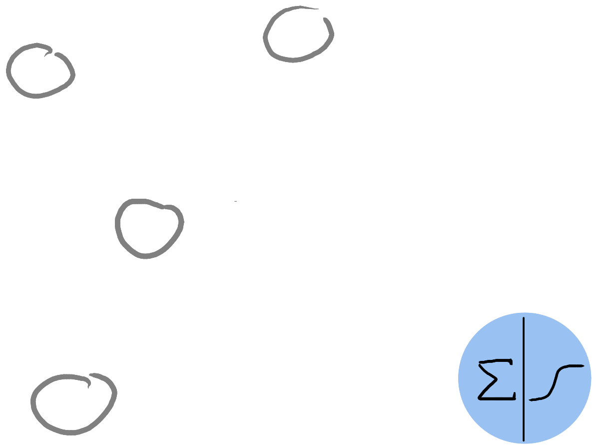
X Temperature	Y Hike
14.2	0
10.9	0
13.1	0
25.6	1
24.8	1
38	0
40	0
32	0.5
33.2	0.4
16	0.3

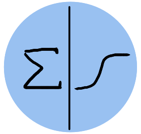
Min-Max Scale the Temperature feature $\longrightarrow$

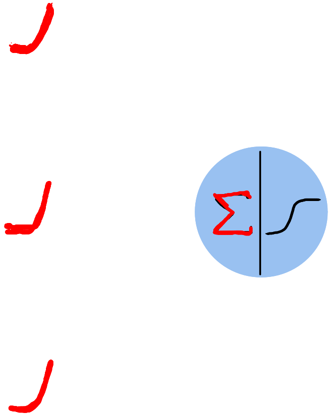
X_{sc} Temperature	Y Hike
0.16	0
0.0	0
0.07560137	0
0.50515464	1
0.47766323	1
0.93127148	0
1.0	0
0.72508591	0.5
0.76632302	0.4
0.17525773	0.3

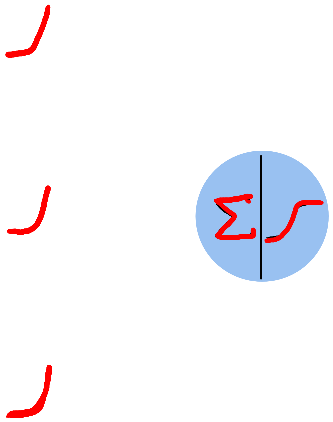



bias


Σ


Σ



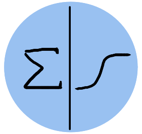


$$\hat{y}_1 = 0.627\ldots$$

How do we interpret this number?

What does it mean?

Is it good? bad?



$$P(\text{hike} \mid \text{Temp} = 14.2^\circ C)$$

$$0.16 = \text{scale}(14.2^\circ C)$$

For input x, $\hat{y} = \underline{NN}(\underline{scale}(x))$, where scale is the min-max scaling function and NN is the neural network

$x = 14.2°C$

$scale(x) = 0.16$

$\hat{y} = NN(0.16) = 0.627$

We interpret $\hat{y}$ as a prediction for $P(hike \mid Temp = x)$

We have a known value for $P(hike \mid Temp = 14.2°C)$, it's 0!

$\hat{y} = 0.62 \longleftrightarrow y = 0.0$

Not good!

No! random weights.

Did we expect good?

Next time...
  How to get weights that give better predictions (How to train neural networks via backpropagation)

Later in this lecture...
  How to say something more precise than "Not Good" (Classification Metrics)

Up next...
  What kinds of functions can neural networks represent?

For input x, $\hat{y} = \underline{NN}(\underline{scale}(x))$, where scale is the min-max scaling function and NN is the neural network

$x = 14.2°C$

$scale(x) = 0.16$

$\hat{y} = NN(0.16) = 0.627$

We interpret $\hat{y}$ as a prediction for $P(hike \mid Temp = x)$

We have a known value for $P(hike \mid Temp = 14.2°C)$, it's 0!

$\hat{y} = 0.62 \longleftrightarrow y = 0.0$

Not good!

No! random weights.

Did we expect good?

Next time...
How to get weights that give better predictions (How to train neural networks via backpropagation)

Later in this lecture...
How to say something more precise than "Not Good" (Classification Metrics)

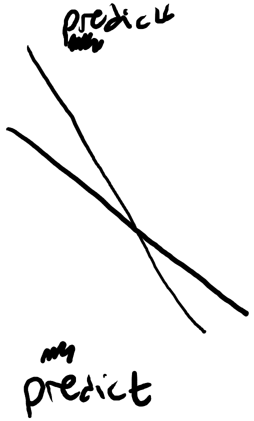
Up next...
What kinds of functions can neural networks represent?

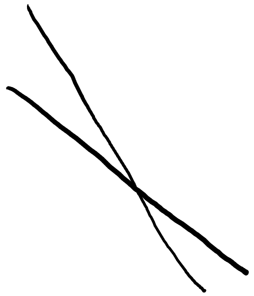
Solutions are non-unique

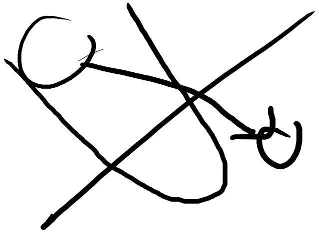
predict

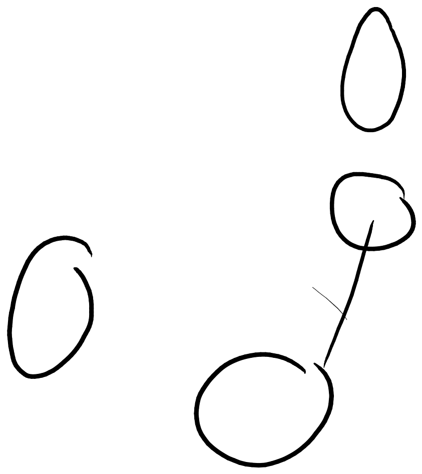
predict

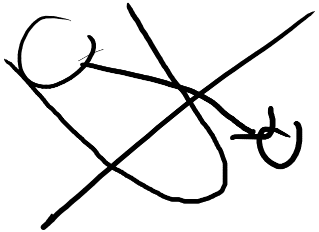
Solutions are non-unique

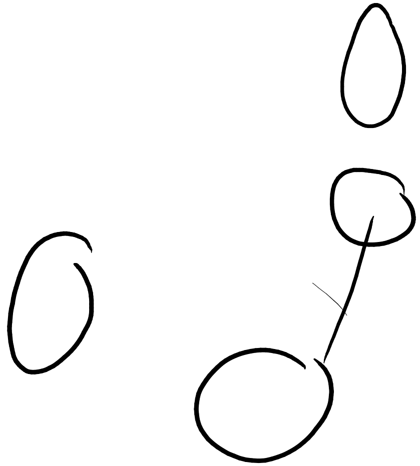

predict
predict



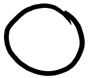




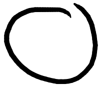




$$\frac{0}{0}$$



τ



π

MLP can represent non-linearly separable decision problems!

$=-2$

$=-2$

$=1$

bias

-4
-4
3

We engineered a neural network by hand, starting with intuitions about logical functions.

This is abnormal! We only did this to show that it's possible!

Normally we are much lazier, and have gradient descent do the work for us.

Will gradient descent find such a "neat" solution? Sometimes, but not always.

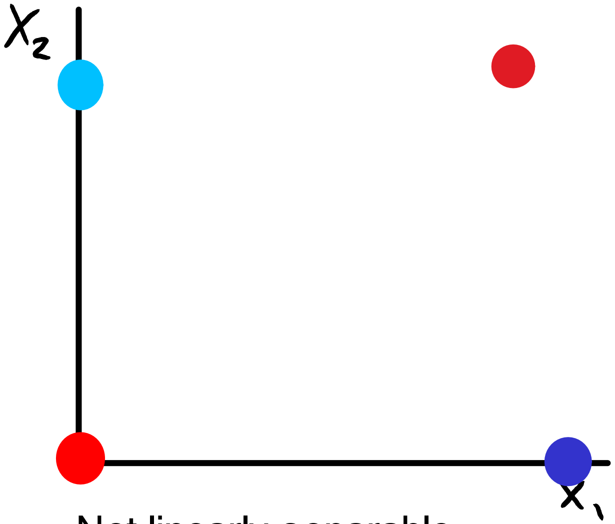
data facts $\xrightarrow{?}$ Best architecture

Truth Table for the network

x_1	x_2	$N1$	$N2$	(XOR) Nz
0	0	1	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

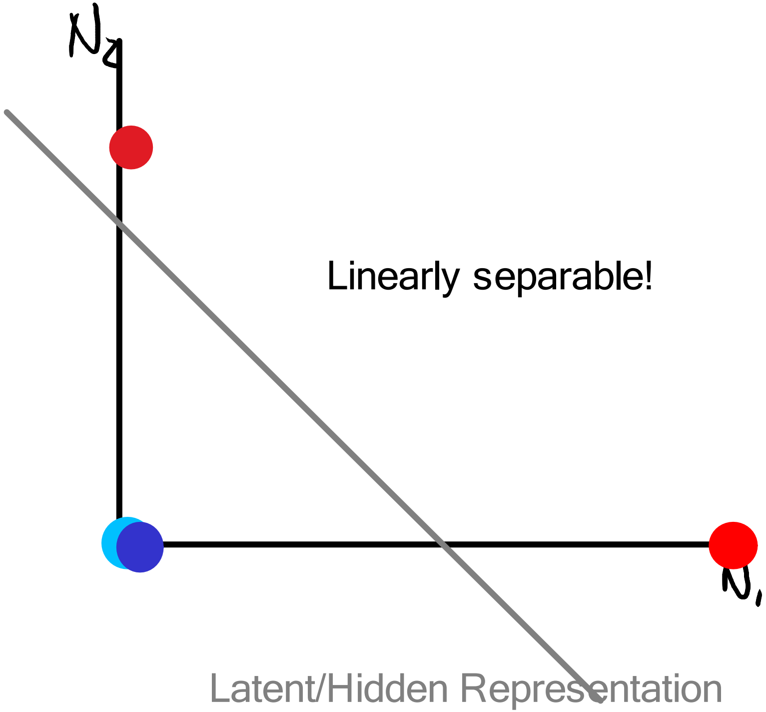
→Latent Representation

→Latent Space



Not linearly separable

Native/original representation



Latent/Hidden Representation

Types of prediction

		P	N ← predict
real data	P	TP	FN
	N	FP	TN

Dog?

TN

MLP can represent non-linearly separable decision problems!

=-2

=-2

=1

bias

-4
-4
3

We engineered a neural network by hand, starting with intuitions about logical functions.

$TP + TN + FP + FN = N$

This is abnormal! We only did this to show that it's possible!

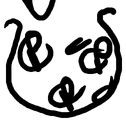
N := number of examples

Normally we are much lazier, and have gradient descent do the work for us.

ignore

P, P all the positives!

Will gradient descent find such a "neat" solution? Sometimes, but not always.



data

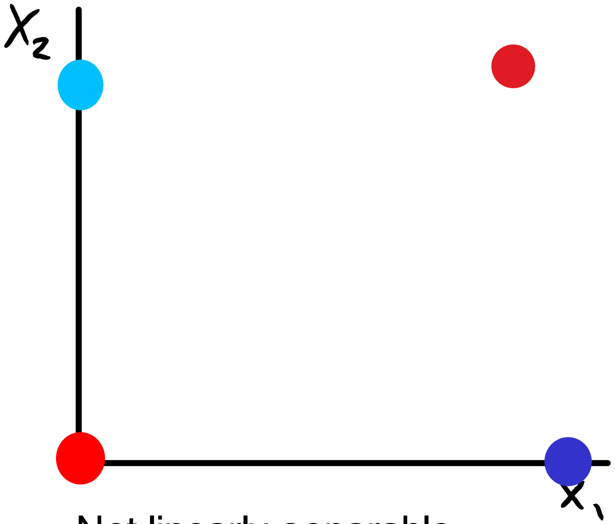
facts $\xrightarrow{?}$ Best architecture

Truth Table for the network

x_1	x_2	$N1$	$N2$	(XOR) Nz
0	0	1	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

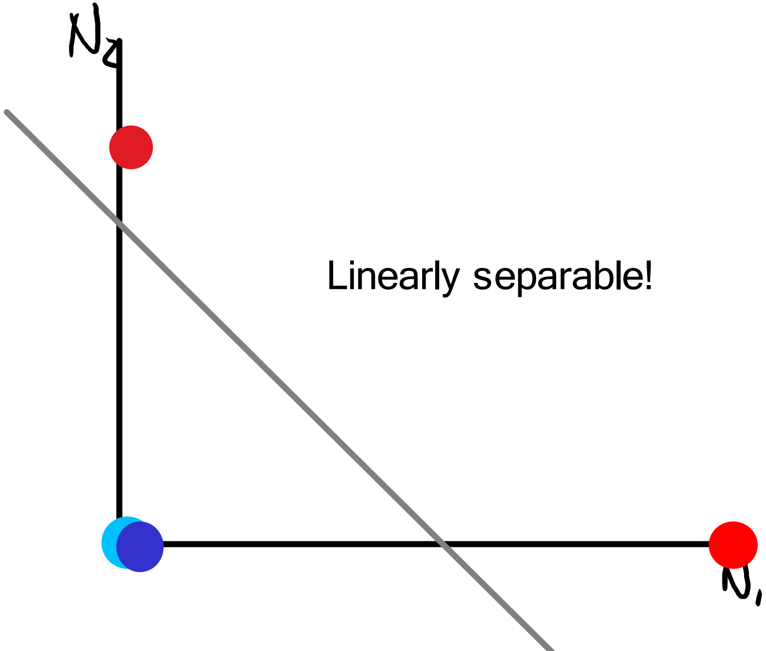
→ Latent Representation

→ Latent Space



Not linearly separable

Native/original representation



Latent/Hidden Representation

Types of prediction

		P'	N ← predict
real data	P	TP	FN
	N	FP	TN

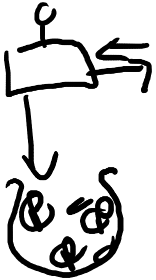
software → trojan

software → not trojan

TN

$$TP + TN + FP + FN = N$$

$N :=$ number of examples



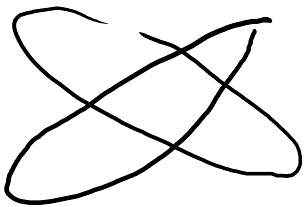
P, P give me all the positives!

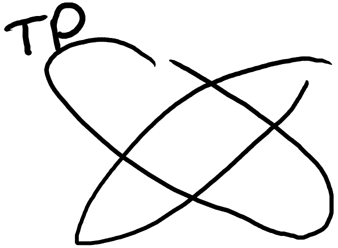
then

software → trojan

→ not trojan

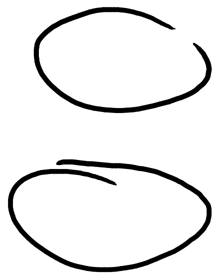
predicted B
actual A

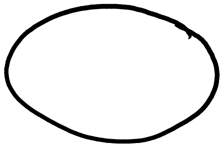




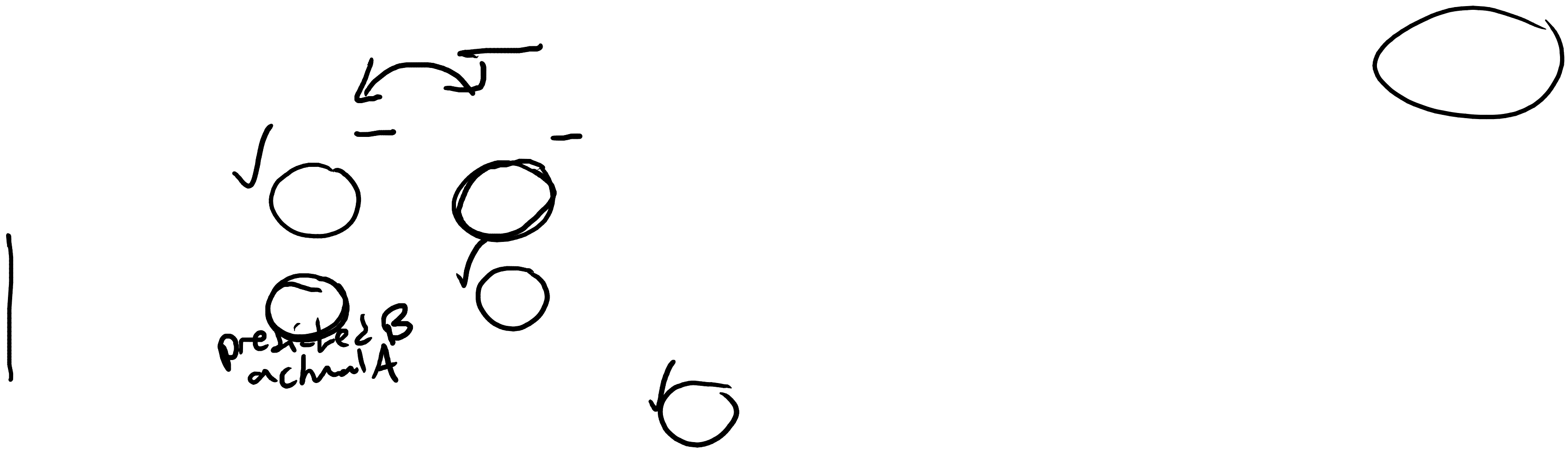
It depends!

$$FNR = \frac{FN}{FN + TN}$$



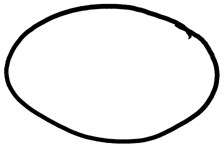


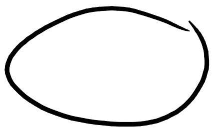
than

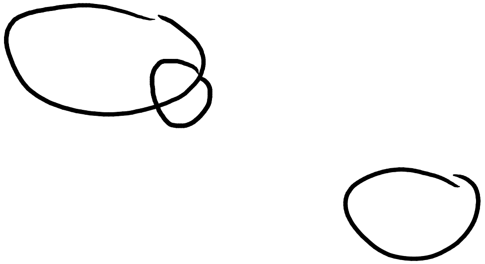


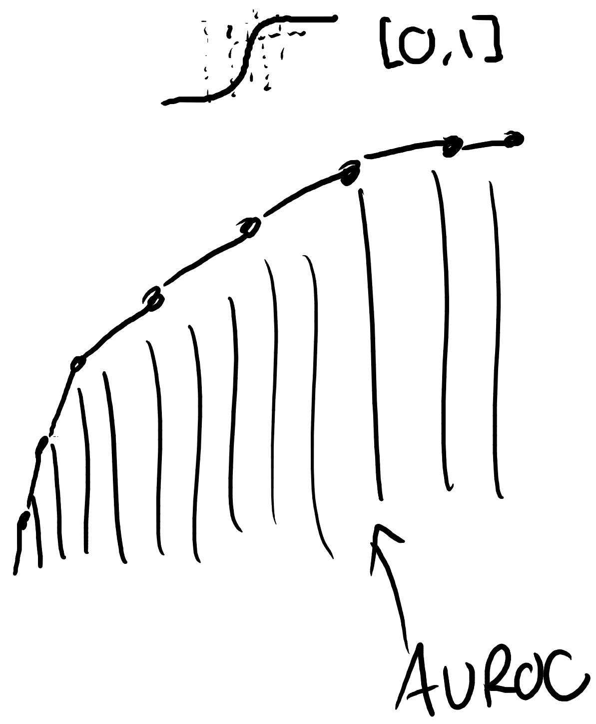
It depends!

$$FNR = \frac{FN}{FN + TN}$$








[0,1]
AUROC