COMPUTER VISION

IS THE SCIENCE OF TEACHING COMPUTERS HOW TO "SEE"

COMPUTER VISION

IS THE SCIENCE OF TEACHING COMPUTERS HOW TO "SEE"

BY THIS WE MEAN : BUILDING SYSTEMS THAT CAN

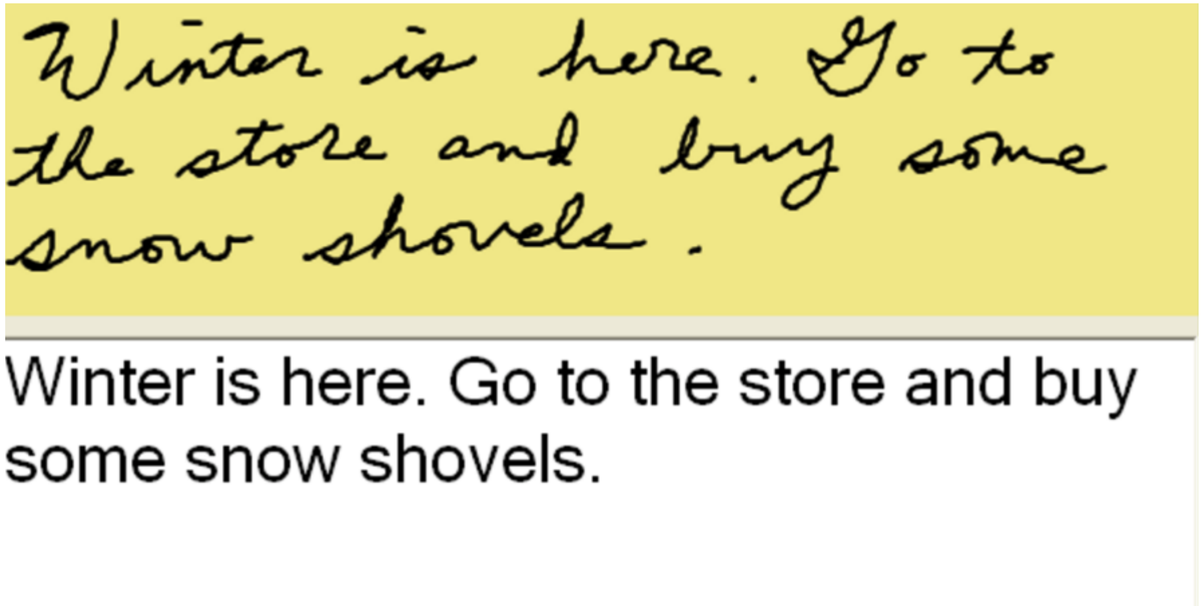
UNDERSTAND AND EXTRACT INFORMATION FROM IMAGES OR VIDEOS

TAKE DECISIONS BASED ON THIS INFORMATION

COMPUTER VISION

IS THE SCIENCE OF TEACHING COMPUTERS HOW TO "SEE"

HANDWRITING RECOGNITION

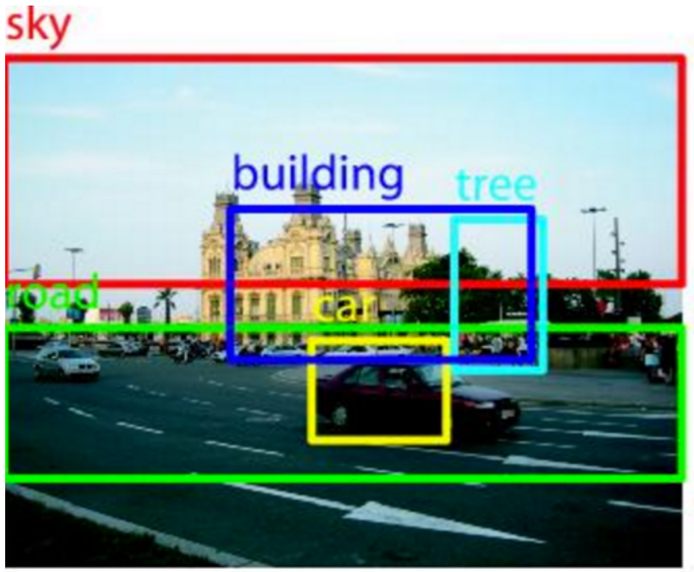


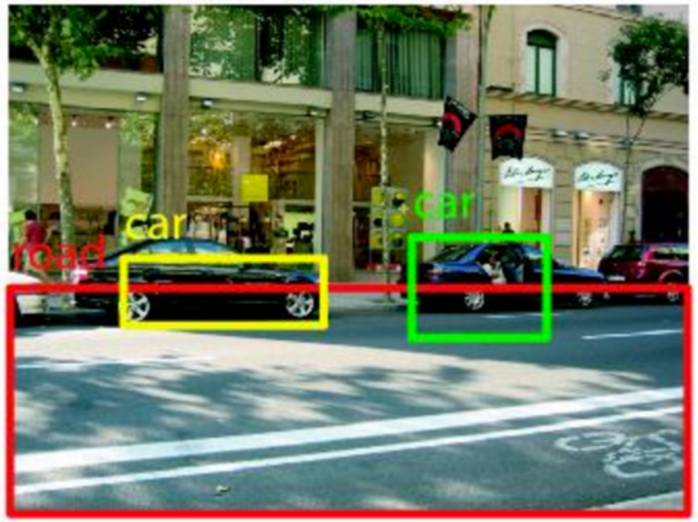
COMPUTER VISION

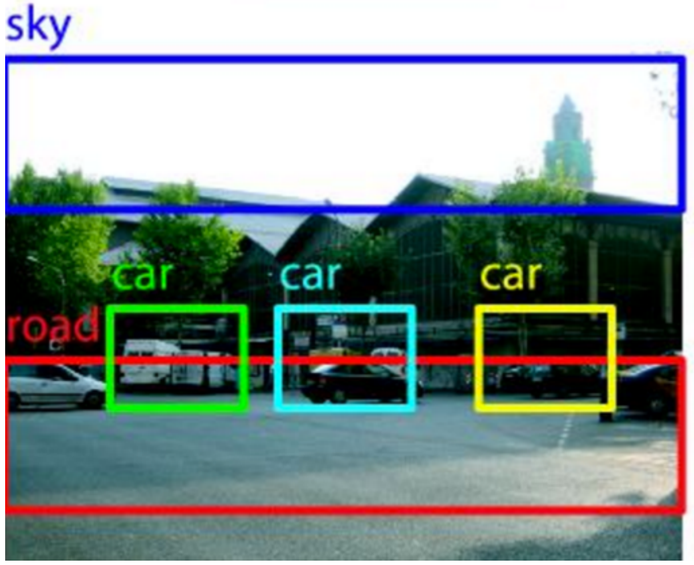
IS THE SCIENCE OF TEACHING COMPUTERS HOW TO "SEE"

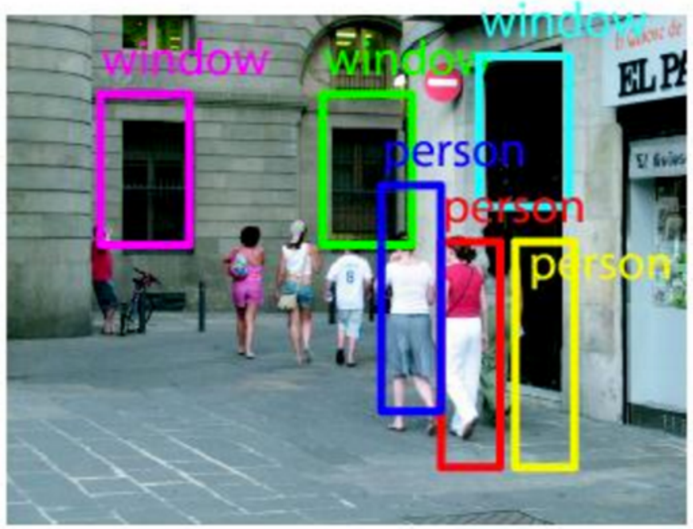
HANDWRITING RECOGNITION

OBJECT RECOGNITION









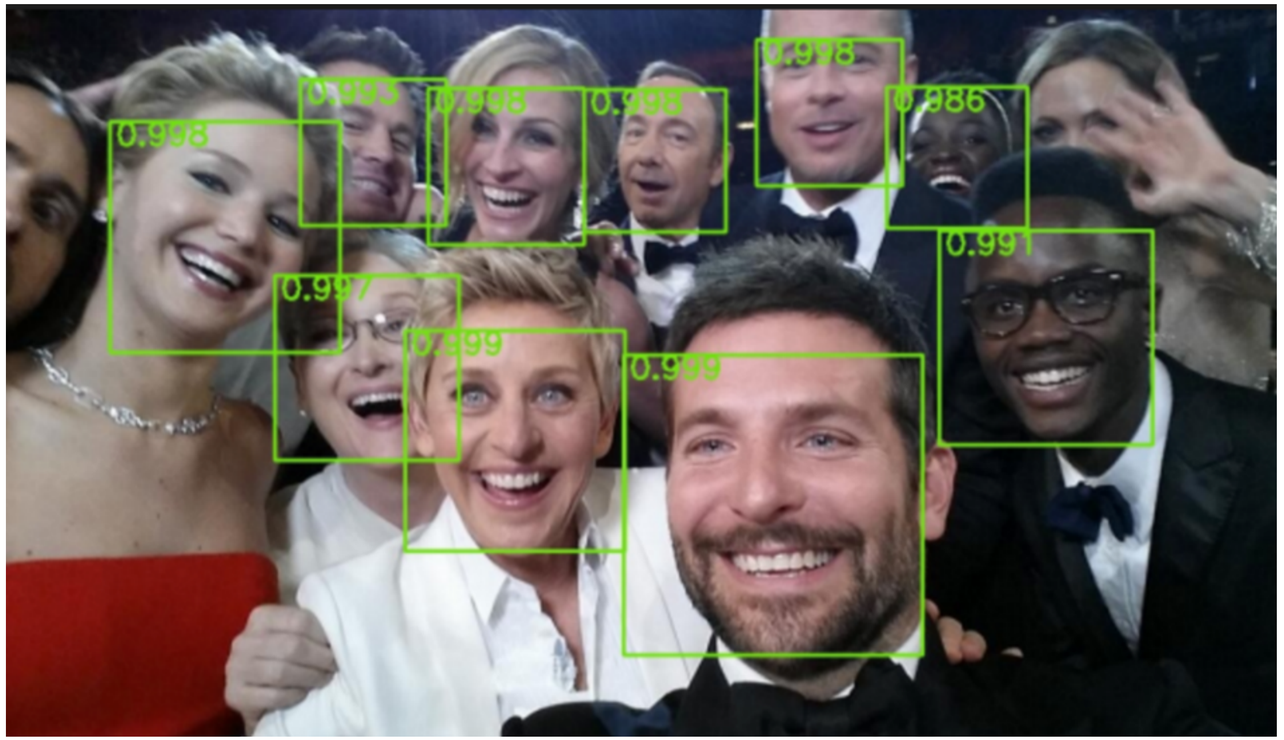
COMPUTER VISION

IS THE SCIENCE OF TEACHING COMPUTERS HOW TO "SEE"

HANDWRITING RECOGNITION

OBJECT RECOGNITION

FACE DETECTION



COMPUTER VISION

IS THE SCIENCE OF TEACHING COMPUTERS HOW TO "SEE"

HANDWRITING RECOGNITION

FACE DETECTION

OBJECT RECOGNITION

ARE ALL APPLICATIONS OF COMPUTER VISION

COMPUTER VISION MAKES AMAZING THINGS POSSIBLE!

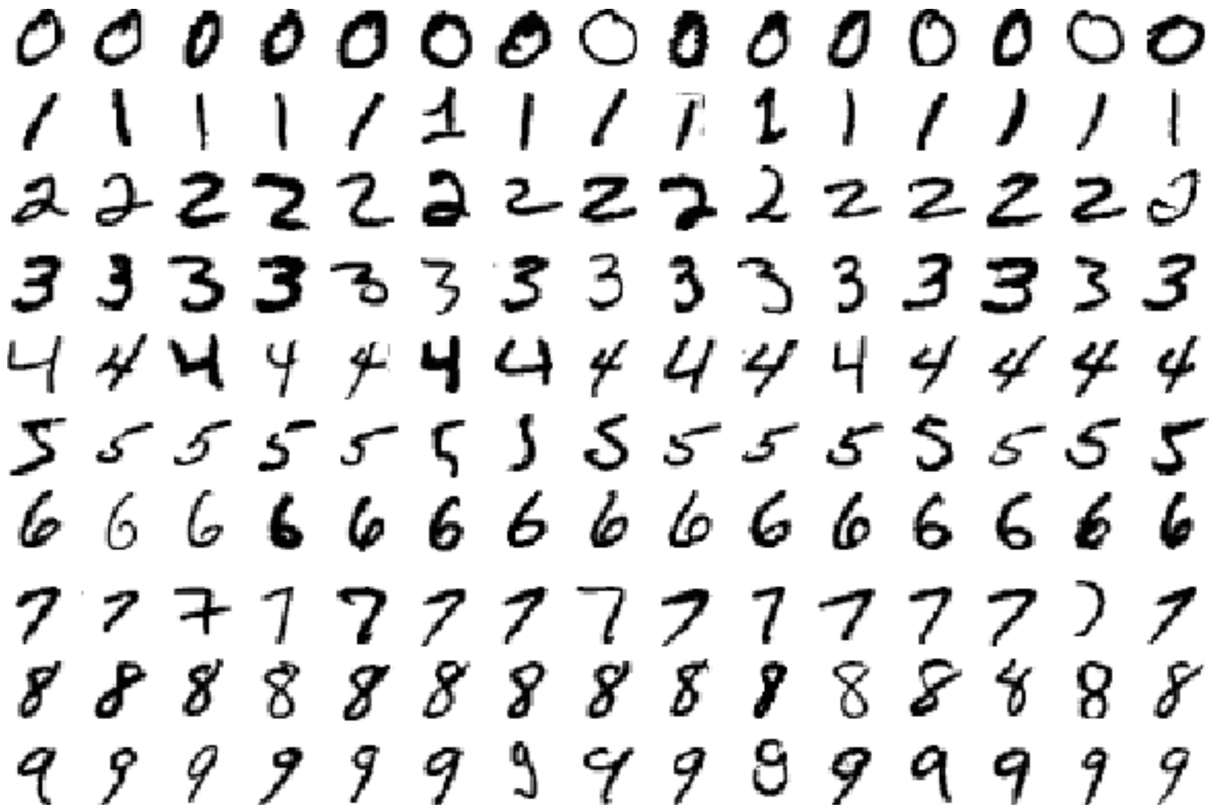
SELF - DRIVING CARS FOR EXAMPLE!

ONE OF THE STARTER PROBLEMS IN COMPUTER VISION IS

HANDWRITTEN DIGIT RECOGNITION

HANDWRITTEN DIGIT RECOGNITION IS USED IN BANK CHEQUE PROCESSING APPLICATIONS FOR EXAMPLE

THESE APPLICATIONS ARE THEREFORE REQUIRED TO HAVE VERY HIGH ACCURACY!



ONE OF THE STARTER PROBLEMS IN COMPUTER VISION IS

HANDWRITTEN DIGIT RECOGNITION

THIS CAN BE SEEN AS A
CLASSIFICATION PROBLEM

YOU ARE GIVEN AN IMAGE OF A
HANDWRITTEN DIGIT

CLASSIFY IT AS ONE OF
THE DIGITS BETWEEN 0-9

HANDWRITTEN DIGIT RECOGNITION

YOU ARE GIVEN AN IMAGE OF A HANDWRITTEN DIGIT

CLASSIFY IT AS ONE OF THE DIGITS BETWEEN 0-9

MNIST IS A DATABASE OF HANDWRITTEN DIGIT IMAGES

IT HAS ~60000 IMAGES OF DIGITS WRITTEN BY ~500 DIFFERENT WRITERS

THESE IMAGES CAN BE USED TO TRAIN A DIGIT CLASSIFIER

HANDWRITTEN DIGIT RECOGNITION

MNIST IS A DATABASE OF HANDWRITTEN DIGIT IMAGES

THESE IMAGES CAN BE USED TO TRAIN A DIGIT CLASSIFIER

TO TRAIN A CLASSIFIER YOU'LL NEED TO

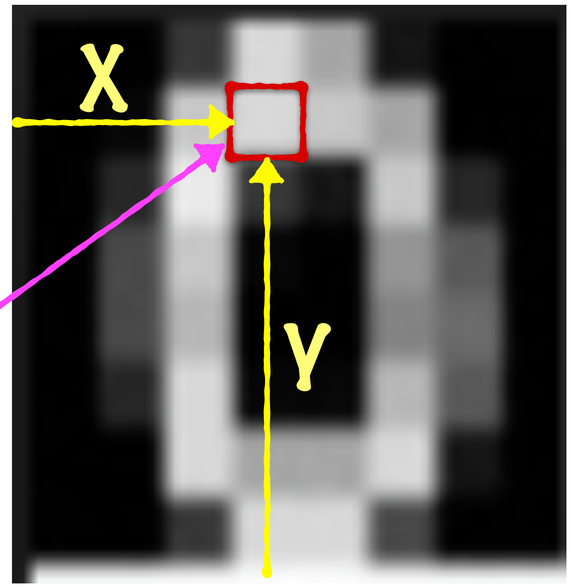
1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

HANDWRITTEN DIGIT RECOGNITION

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

AN IMAGE IS MADE UP OF **PIXELS** EACH PIXEL CAN BE REPRESENTED BY A TUPLE OF NUMBERS



THE X CO-ORDINATE OF THE PIXEL

THE Y CO-ORDINATE OF THE PIXEL

THE COLOR OF THE PIXEL

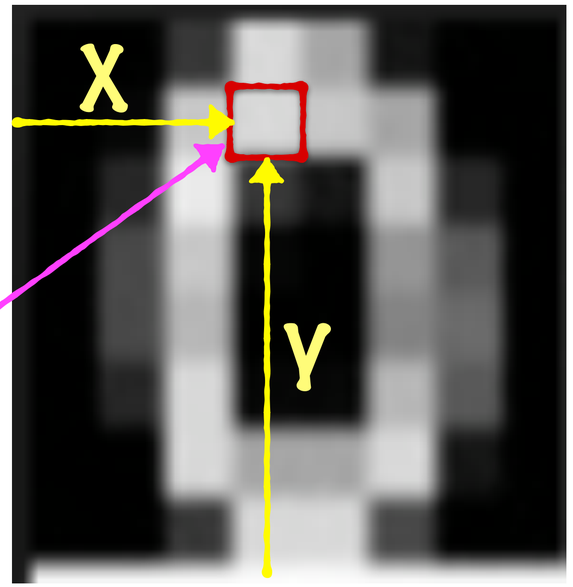
THE COLOR COULD BE 0/1 (BINARY)

IF THE IMAGE IS BLACK AND WHITE

HANDWRITTEN DIGIT RECOGNITION

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

AN IMAGE IS MADE UP OF **PIXELS** EACH PIXEL CAN BE REPRESENTED BY A TUPLE OF NUMBERS



THE X CO-ORDINATE OF THE PIXEL

THE Y CO-ORDINATE OF THE PIXEL

THE COLOR OF THE PIXEL

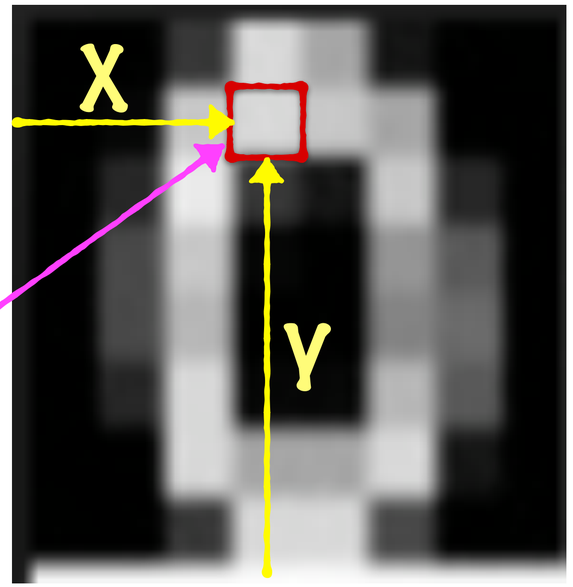
THE COLOR COULD BE A VALUE BETWEEN 0 AND 1

IF THE IMAGE IS GRAYSCALE

HANDWRITTEN DIGIT RECOGNITION

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

AN IMAGE IS MADE UP OF **PIXELS** EACH PIXEL CAN BE REPRESENTED BY A TUPLE OF NUMBERS



THE X CO-ORDINATE OF THE PIXEL

THE Y CO-ORDINATE OF THE PIXEL

THE COLOR OF THE PIXEL

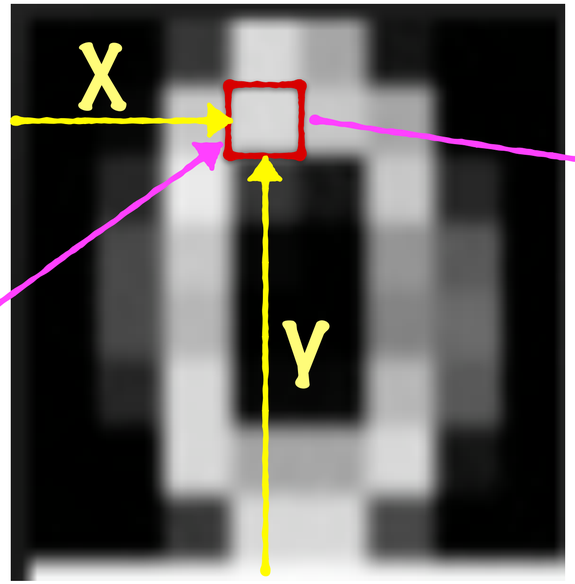
THE COLOR COULD BE A TUPLE OF 3 NUMBERS IE R, G, B INTENSITIES

IF IT'S A FULL-COLOR IMAGE

HANDWRITTEN DIGIT RECOGNITION

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

AN IMAGE IS MADE UP OF **PIXELS**



A PIXEL

THE X CO-ORDINATE OF THE PIXEL

THE Y CO-ORDINATE OF THE PIXEL

THE COLOR OF THE PIXEL

ANY (GRAYSCALE) IMAGE CAN BE REPRESENTED AS A 2-D ARRAY

0	0	0	0	0	0
0	1	1	1	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0

HANDWRITTEN DIGIT RECOGNITION

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

ANY (GRAYSCALE) IMAGE CAN BE REPRESENTED AS A 2-D ARRAY

0	0	0	0	0	0
0	1	1	1	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0

YOU COULD USE THIS DIRECTLY AS THE FEATURE VECTOR FOR A CLASSIFICATION ALGORITHM

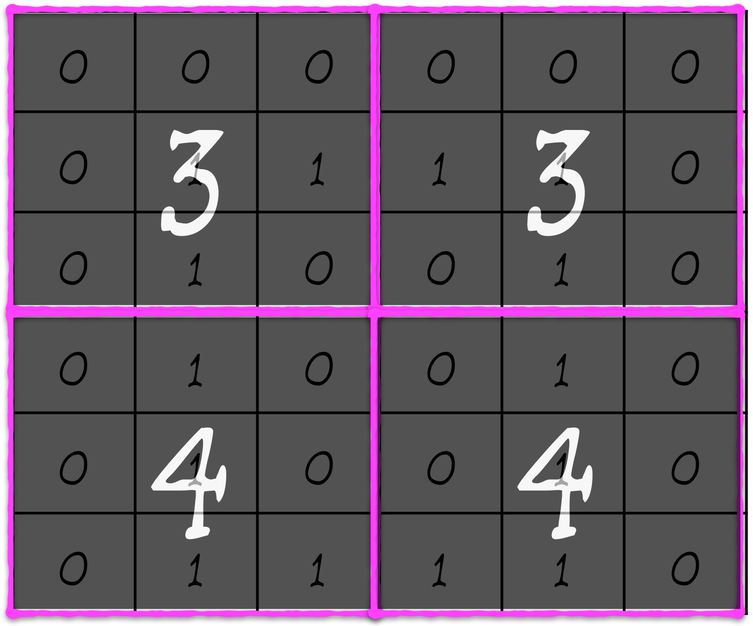
HOWEVER, IF THE IMAGE IS FULL COLOR WITH MILLIONS OF PIXELS - THE FEATURE VECTOR COULD END UP HAVING TOO MANY DIMENSIONS

USING FEATURE EXTRACTION TECHNIQUES, A SMALL NUMBER OF FEATURES CAN BE CREATED TO REPRESENT THE IMAGE

HANDWRITTEN DIGIT RECOGNITION

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

ANY (GRAYSCALE) IMAGE CAN BE REPRESENTED AS A 2-D ARRAY



USING FEATURE EXTRACTION TECHNIQUES A SMALL NUMBER OF FEATURES CAN BE CREATED TO REPRESENT THE IMAGE

ONE SIMPLE METHOD COULD BE

DIVIDE THE IMAGE INTO A NUMBER OF ZONES

COUNT THE NUMBER OF BLACK PIXELS IN EACH ZONE

HANDWRITTEN DIGIT RECOGNITION

MNIST IS A DATABASE OF HANDWRITTEN DIGIT IMAGES

THESE IMAGES CAN BE USED TO TRAIN A DIGIT CLASSIFIER

TO TRAIN A CLASSIFIER YOU'LL NEED TO

1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

HANDWRITTEN DIGIT RECOGNITION

MNIST IS A DATABASE OF HANDWRITTEN DIGIT IMAGES

THESE IMAGES CAN BE USED TO TRAIN A DIGIT CLASSIFIER

TO TRAIN A CLASSIFIER YOU'LL NEED TO

✓ 1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

HANDWRITTEN DIGIT RECOGNITION

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

THERE ARE MANY DIFFERENT CHOICES FOR THE CLASSIFICATION ALGORITHM

SUPPORT VECTOR MACHINES

LOGISTIC REGRESSION

RANDOM FORESTS

ALL OF THESE HAVE BEEN SEEN TO PERFORM WELL (WITH HIGH ACCURACY) ON THE MNIST DATABASE

THESE DAYS A POPULAR CHOICE FOR THE CLASSIFICATION ALGORITHM IS TO USE

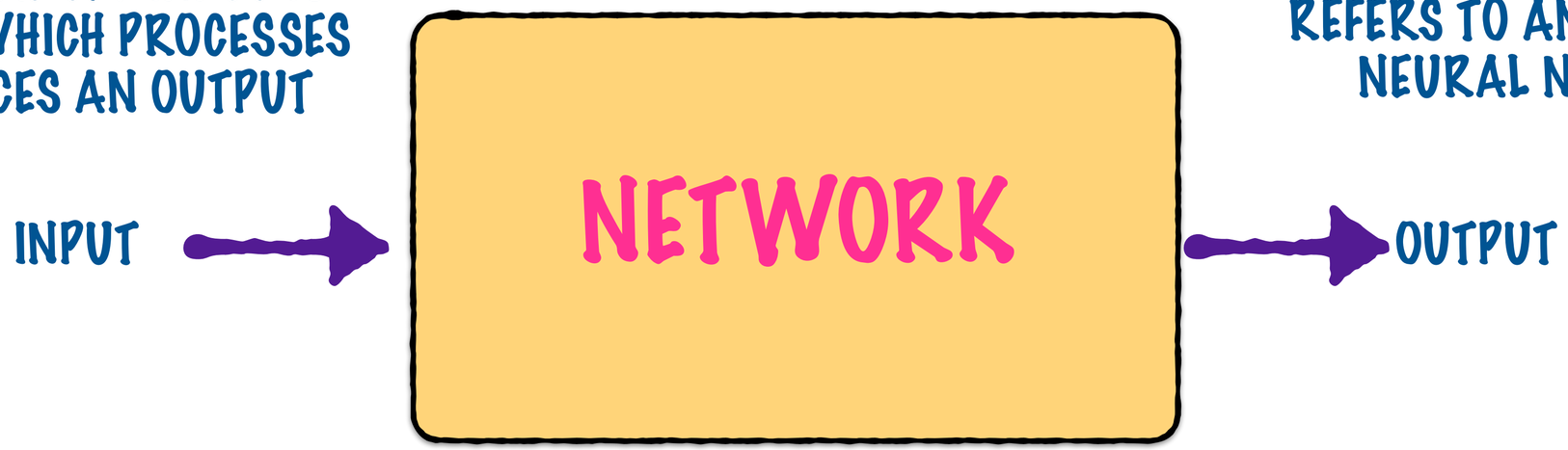
A DEEP LEARNING NETWORK

HANDWRITTEN DIGIT RECOGNITION

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

A DEEP LEARNING NETWORK

THE BASIC IDEA IS THAT THE INPUT IS FED TO A NETWORK WHICH PROCESSES IT AND THEN PRODUCES AN OUTPUT



THE NETWORK HERE USUALLY REFERS TO AN ARTIFICIAL NEURAL NETWORK

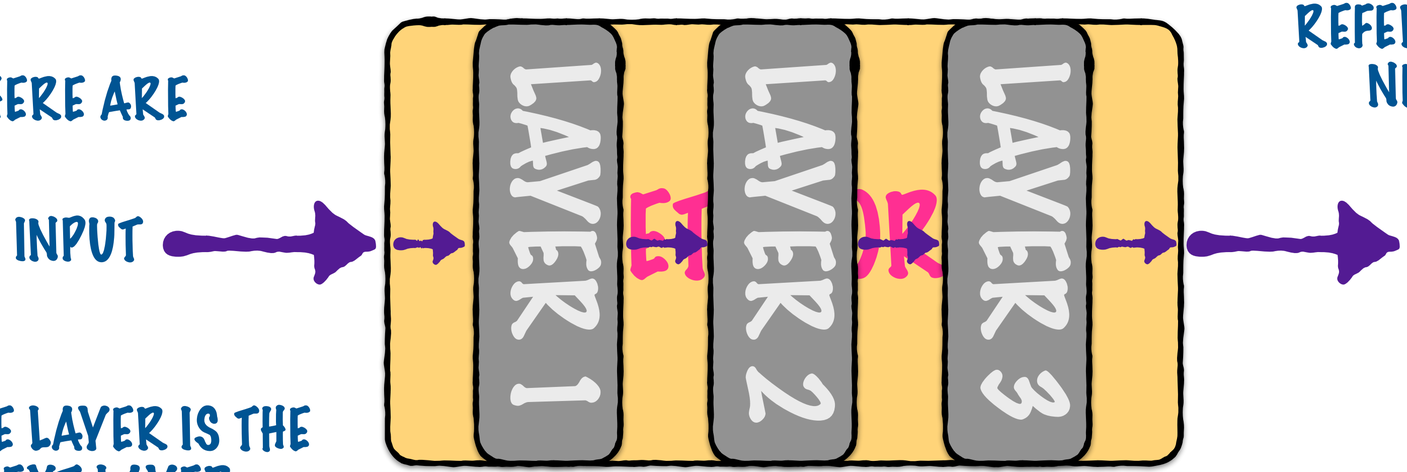
HANDWRITTEN DIGIT RECOGNITION

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

A DEEP LEARNING **NETWORK**

THE NETWORK HERE USUALLY REFERS TO AN ARTIFICIAL NEURAL NETWORK

INSIDE THE NETWORK THERE ARE **LAYERS**



THE OUTPUT OF ONE LAYER IS THE INPUT TO THE NEXT LAYER

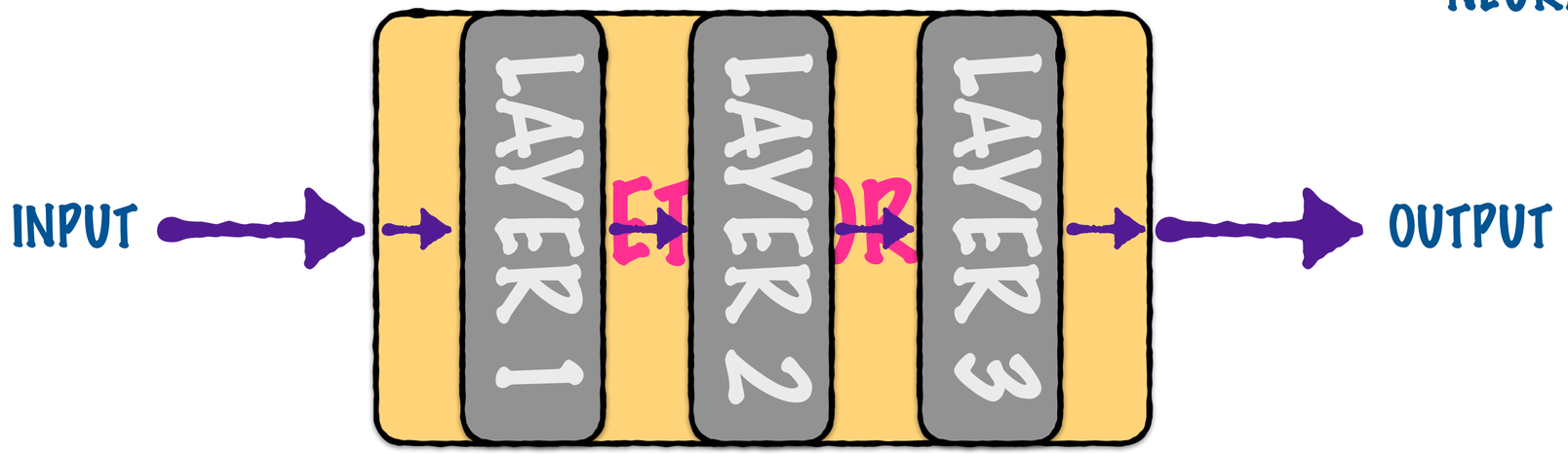
HANDWRITTEN DIGIT RECOGNITION

2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

A DEEP LEARNING NETWORK

DEEP JUST REFERS TO THE FACT THAT THERE ARE MANY LAYERS INSIDE THE NETWORK

THE NETWORK HERE USUALLY REFERS TO AN ARTIFICIAL NEURAL NETWORK



LET'S START WITH THE SIMPLEST POSSIBLE ARTIFICIAL NEURAL NETWORK

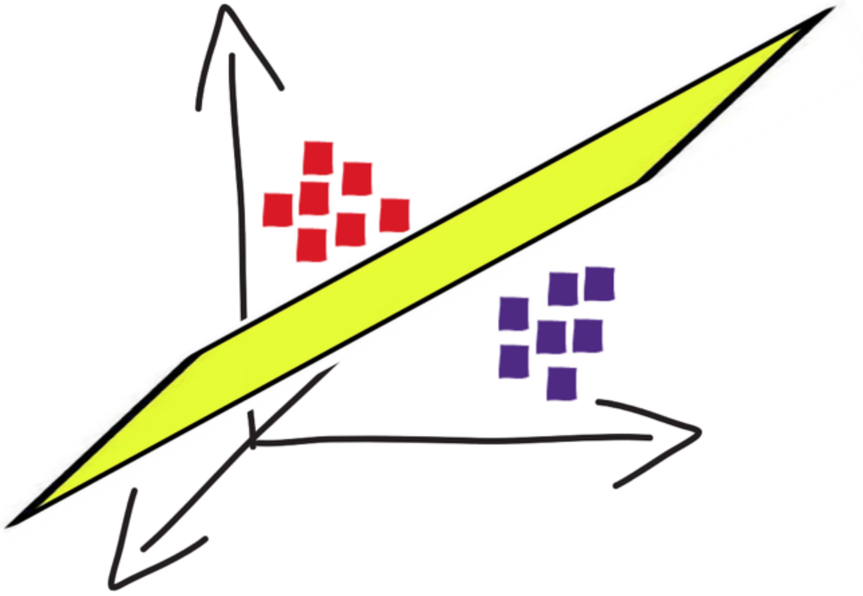
A PERCEPTRON

ARTIFICIAL NEURAL NETWORKS ARE ANALOGOUS TO SUPPORT VECTOR MACHINES AND OTHER LEARNING TECHNIQUES –

INDEED, THE FIRST AND PROTOTYPICAL EXAMPLE OF AN ARTIFICIAL NEURAL NETWORK IS

A PERCEPTRON

WHICH IS BASICALLY A BINARY CLASSIFIER – SIMILAR TO A SUPPORT VECTOR MACHINE, BUT LESS SOPHISTICATED



A PERCEPTRON IS A SPECIFIC ALGORITHM FOR DETERMING SOME HYPERPLANE THAT SEPARATES DATA OF TWO CATEGORIES

THIS SPECIFIC ALGORITHM IS A DIFFERENT WAY TO GET SOME – ANY – HYPERPLANE THAT SEPARATES THE POINTS

THE SUPPORT VECTOR MACHINE FINDS THE "BEST" SUCH HYPERPLANE, NAMELY THE MAXIMUM MARGIN HYPERPLANE

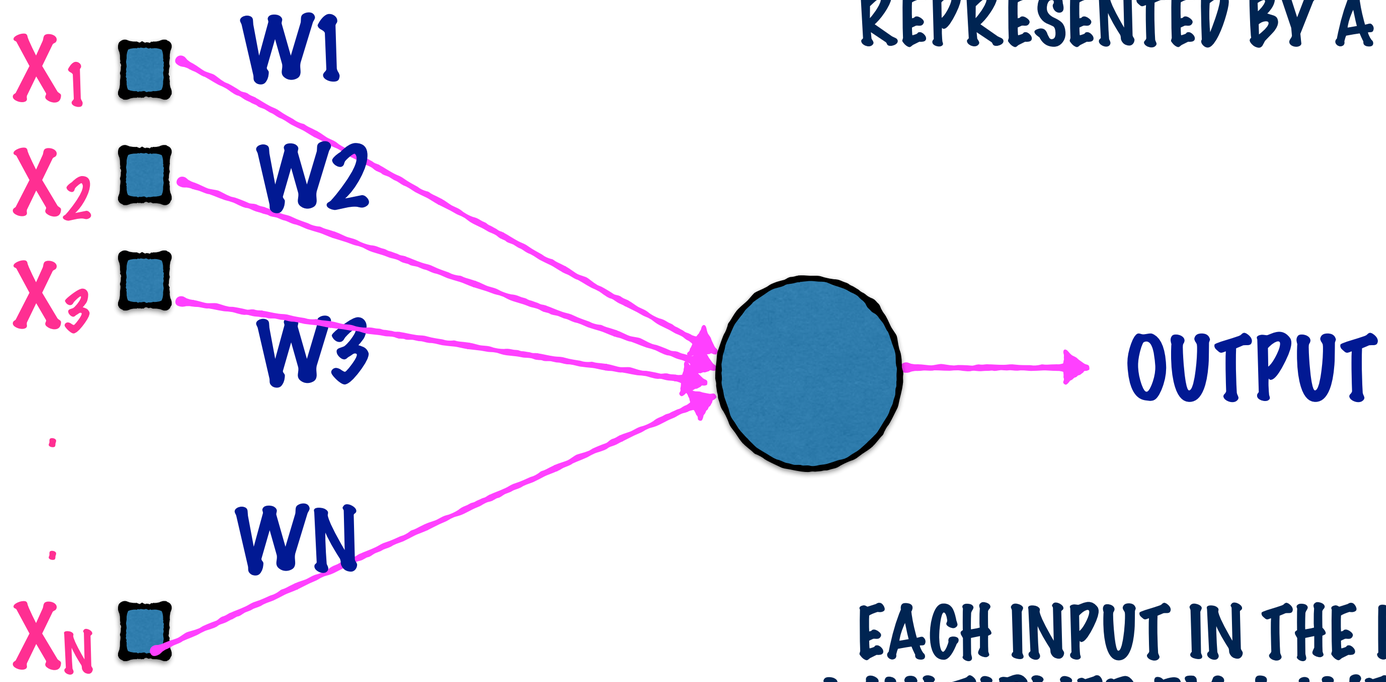
WHILE THE PERCEPTRON WILL MERELY ATTEMPT TO FIND ONE SUCH HYPERPLANE

LET'S START WITH THE SIMPLEST POSSIBLE ARTIFICIAL NEURAL NETWORK

A PERCEPTRON

HERE IS HOW A PERCEPTRON CAN BE REPRESENTED AS A NETWORK

FEATURE VECTOR (INPUT)



EACH INPUT IN THE FEATURE VECTOR IS REPRESENTED BY A NODE

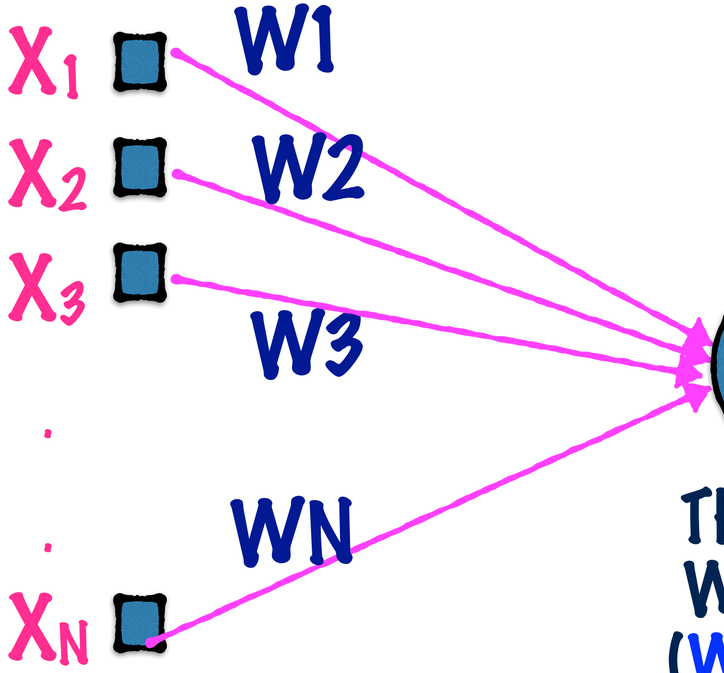
EACH INPUT IN THE FEATURE VECTOR IS MULTIPLIED BY A WEIGHT AND FED TO AN OUTPUT NODE

LET'S START WITH THE SIMPLEST POSSIBLE ARTIFICIAL NEURAL NETWORK

A PERCEPTRON

HERE IS HOW A PERCEPTRON CAN BE REPRESENTED AS A NETWORK

FEATURE VECTOR (INPUT)

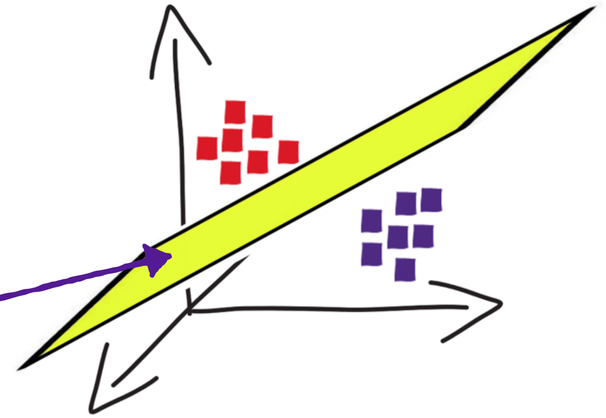


INSIDE THE OUTPUT NODE A FUNCTION IS APPLIED -

IF W1*X1+W2*X2+.....+WN*XN > D,
THEN OUTPUT =1
ELSE OUTPUT =0

OUTPUT

THIS IS EQUIVALENT TO CHECKING WHICH SIDE OF THE HYPERPLANE (W1*X1+W2*X2+.....+WN*XN = D) THE POINT REPRESENTED BY THE INPUT LIES

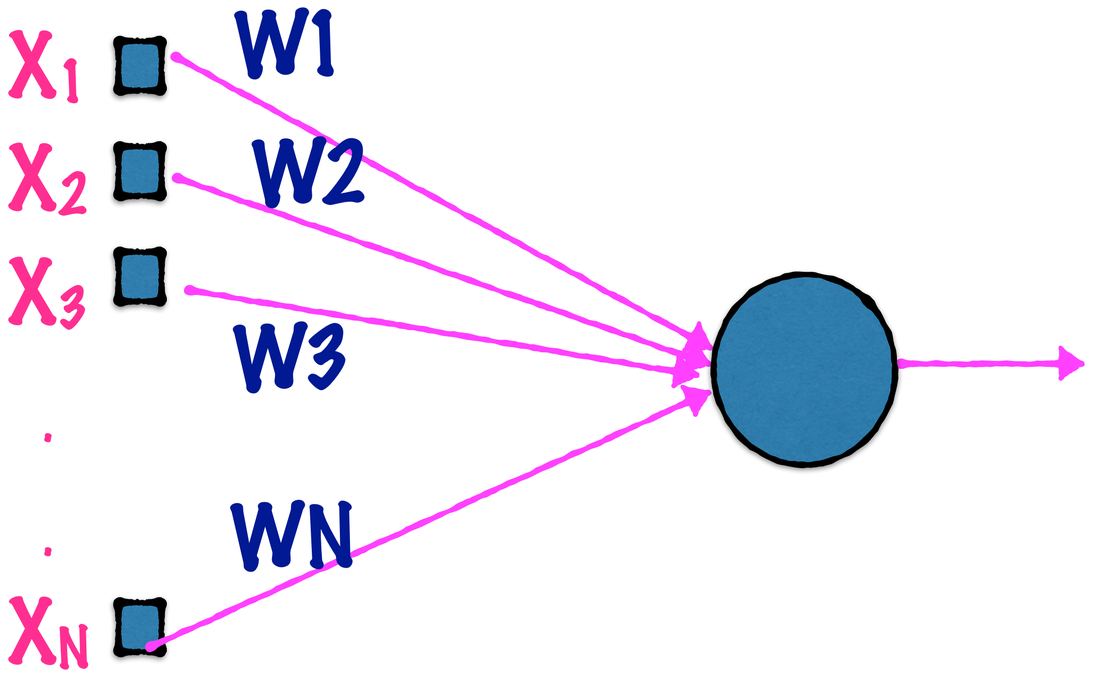


LET'S START WITH THE SIMPLEST POSSIBLE ARTIFICIAL NEURAL NETWORK

A PERCEPTRON

HERE IS HOW A PERCEPTRON CAN BE REPRESENTED AS A NETWORK

FEATURE VECTOR (INPUT)



WHEN YOU TRAIN A PERCEPTRON

THE OBJECTIVE IS TO FIND THE WEIGHTS W1, W2, ...ETC

BY FINDING AN APPROPRIATE SET OF WEIGHTS, THE PERCEPTRON CAN FIND A HYPERPLANE THAT SEPARATES THE POINTS INTO TWO CLASSES

2 QUESTIONS ARISE :

WHAT IF THERE ARE MORE THAN 2 CLASSES ?

WHAT IF THE BOUNDARY THAT SEPARATES THE CLASSES IS NON-LINEAR (IE NOT A PLANE)?

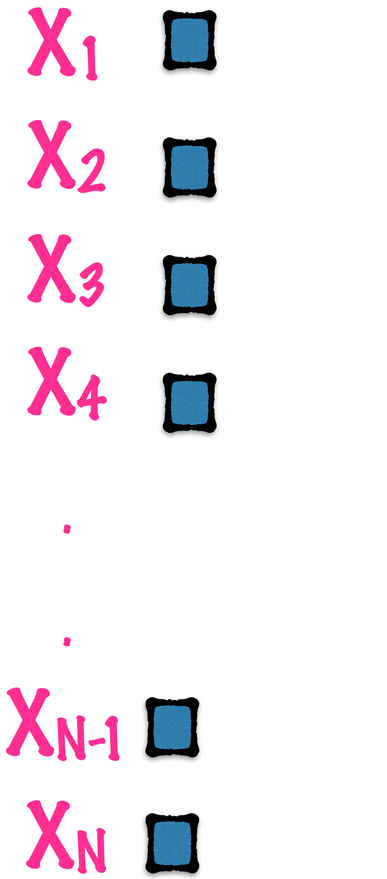
2 QUESTIONS ARISE :

WHAT IF THERE ARE MORE THAN 2 CLASSES ?

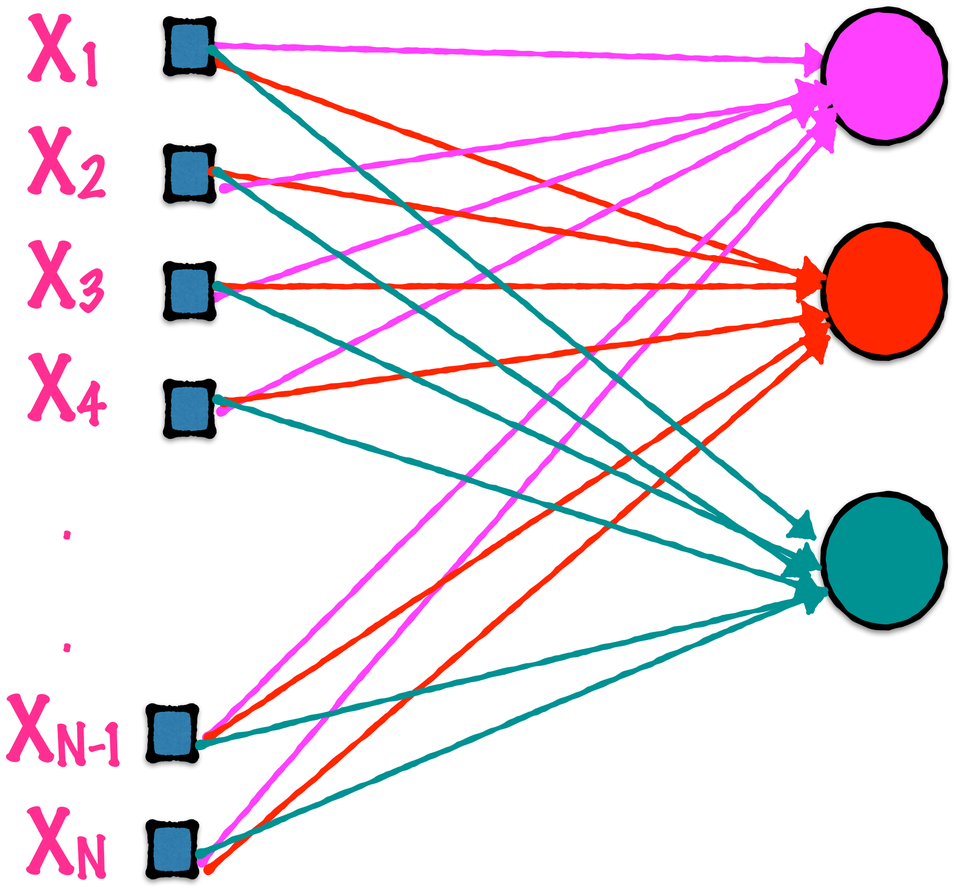
WHAT IF THE BOUNDARY THAT SEPARATES THE CLASSES IS NON-LINEAR (IE NOT A PLANE)?

BY BUILDING UPON THE IDEA OF A PERCEPTRON - WE CAN BUILD NETWORKS THAT SOLVE BOTH THESE PROBLEMS

AS BEFORE EACH INPUT IN THE FEATURE VECTOR IS REPRESENTED BY A NODE



EACH INPUT IS MULTIPLIED BY A WEIGHT AND FED INTO A NODE THAT PROCESSES THEM



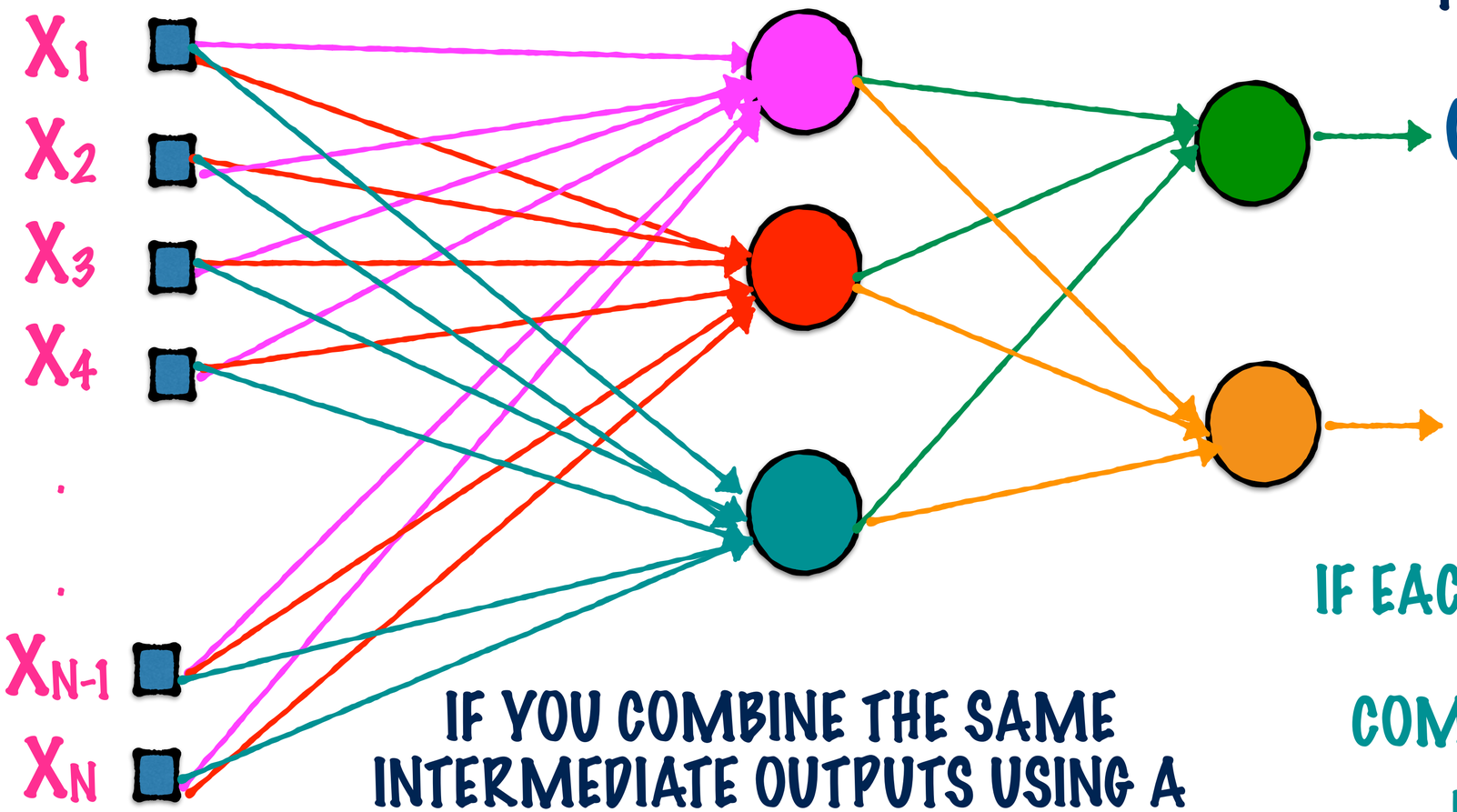
IN THE PERCEPTRON, YOU HAD ONLY 1 SUCH NODE

NOW WE ADD MORE NODES

EACH NODE WILL TAKE IN THE INPUTS MULTIPLIED BY A DIFFERENT SET OF WEIGHTS

EACH NODE WILL PRODUCE AN OUTPUT

EACH OF THE OUTPUTS IN THE INTERMEDIARY NODES CAN BE COMBINED WITH ANOTHER SET OF WEIGHTS TO GET A FINAL OUTPUT

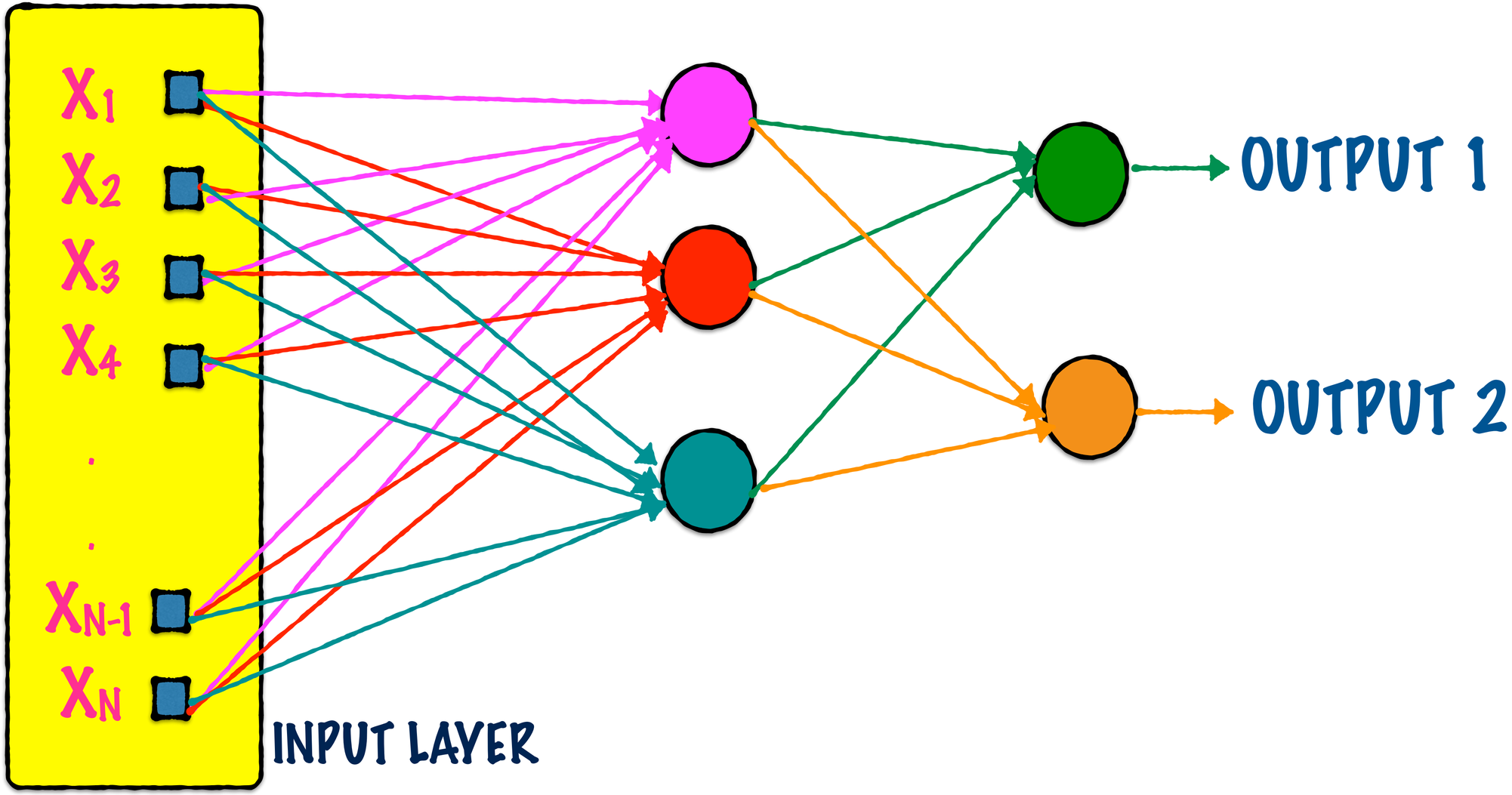


IF YOU COMBINE THE SAME INTERMEDIATE OUTPUTS USING A DIFFERENT SET OF WEIGHTS, YOU CAN GET A DIFFERENT FINAL OUTPUT

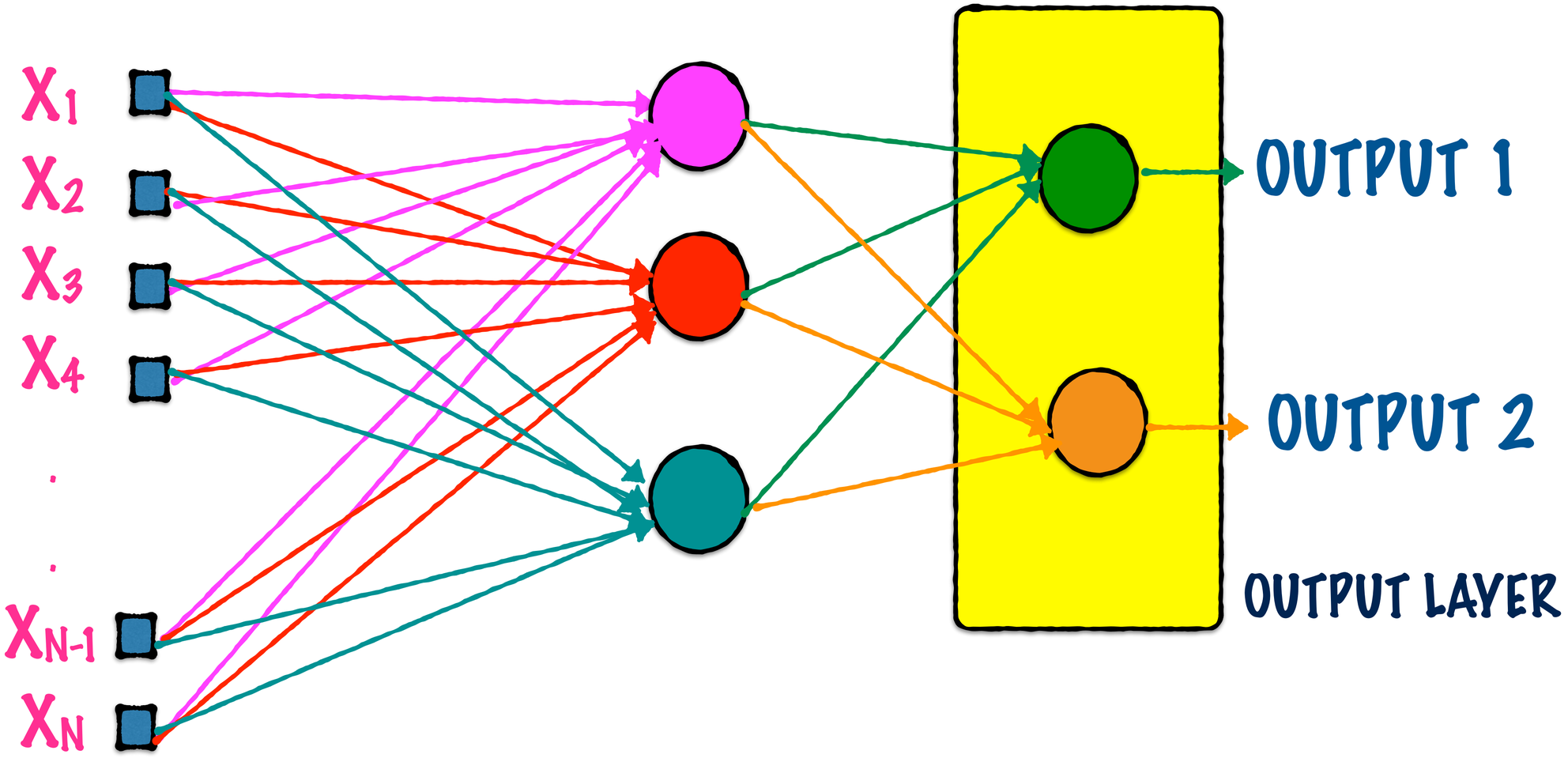
IF EACH OF THESE OUTPUTS IS 1/0 WE HAVE 4 POSSIBLE COMBINATIONS 00, 01, 10, 11

EACH CAN REPRESENT A DIFFERENT CLASS THAT WE CAN CLASSIFY THE OUTPUT AS

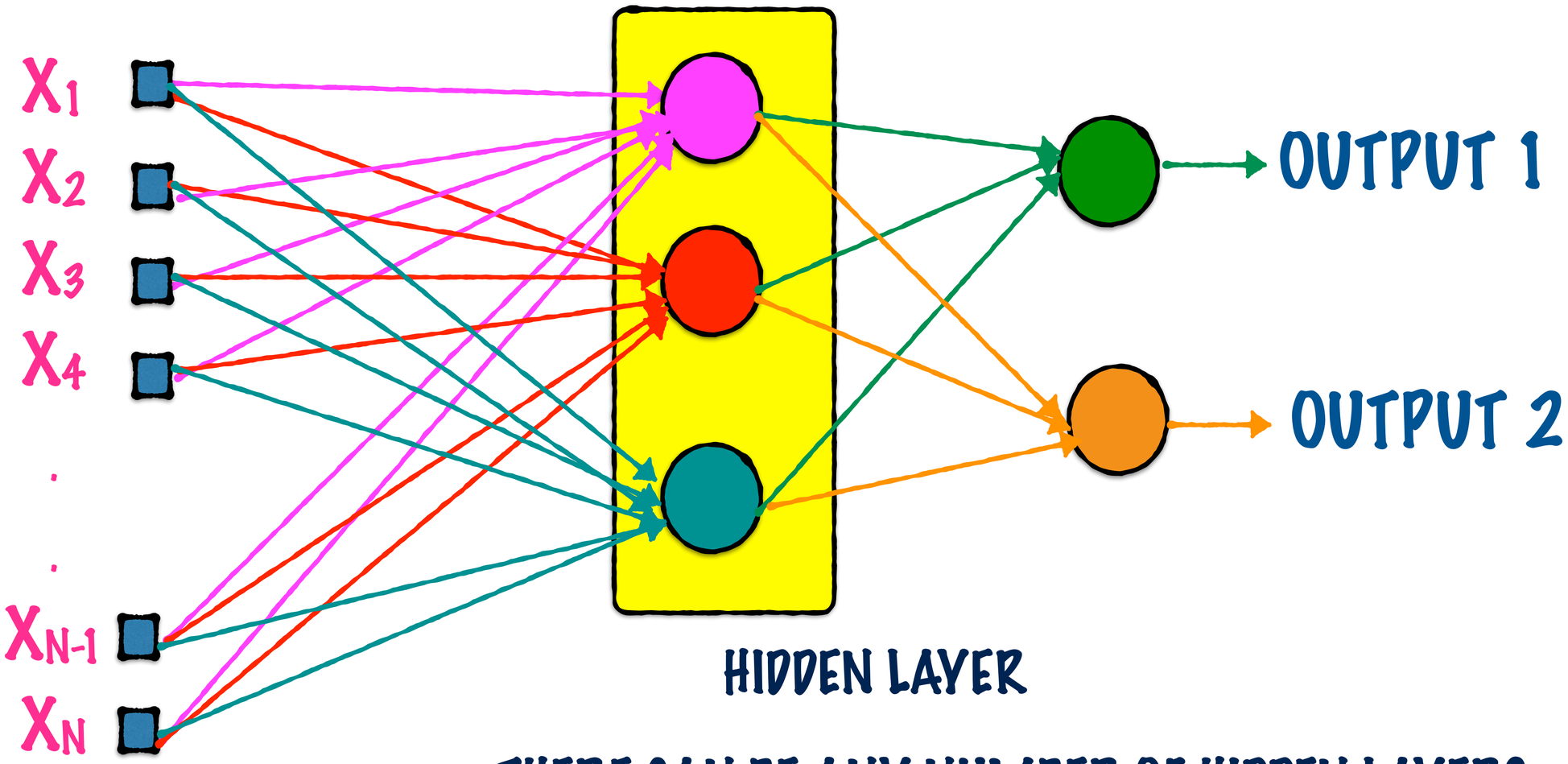
EACH SET OF NODES IS CALLED A LAYER



EACH SET OF NODES IS CALLED A LAYER



EACH SET OF NODES IS CALLED A LAYER

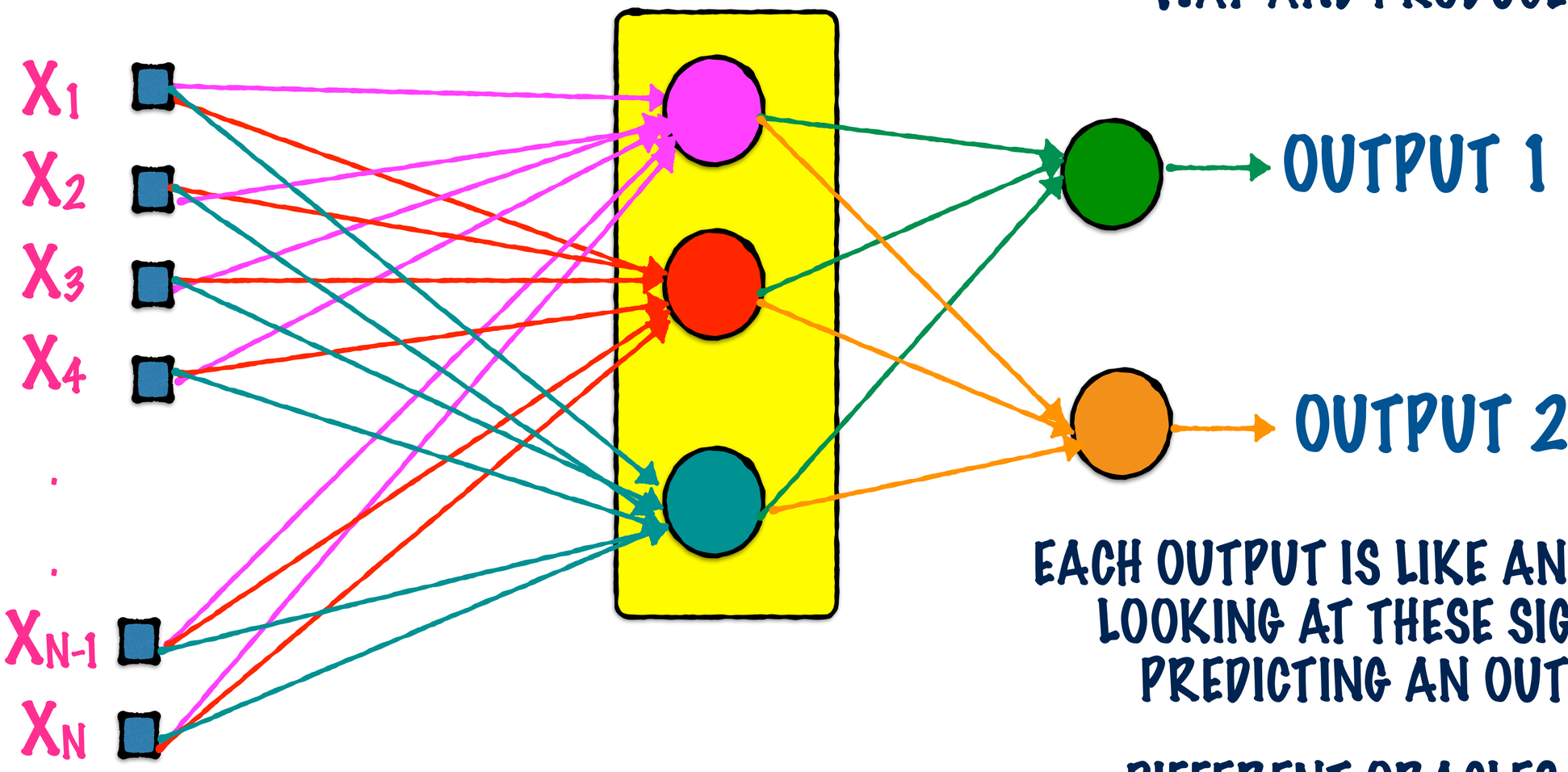


HIDDEN LAYER

THERE CAN BE ANY NUMBER OF HIDDEN LAYERS,
EACH ONE'S OUTPUTS FEEDING INTO THE NEXT

THINK OF AN ORACLE LOOKING AT TEA LEAVES

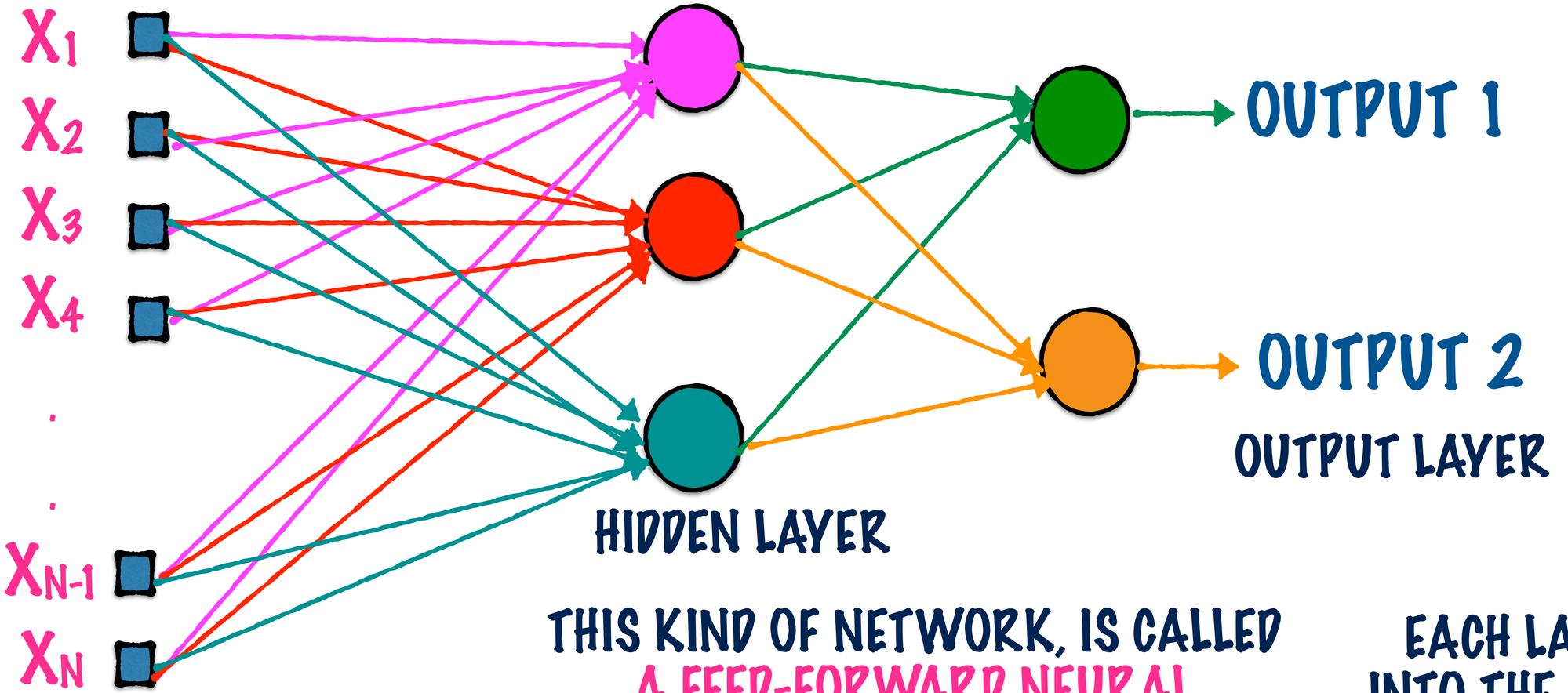
EACH NODE IN THE HIDDEN LAYER "INTERPRETS" THE INPUTS IN SOME WAY AND PRODUCES A SIGN



EACH OUTPUT IS LIKE AN "ORACLE"- LOOKING AT THESE SIGNS AND PREDICTING AN OUTCOME

DIFFERENT ORACLES INTERPRET THE SIGNS IN DIFFERENT WAYS

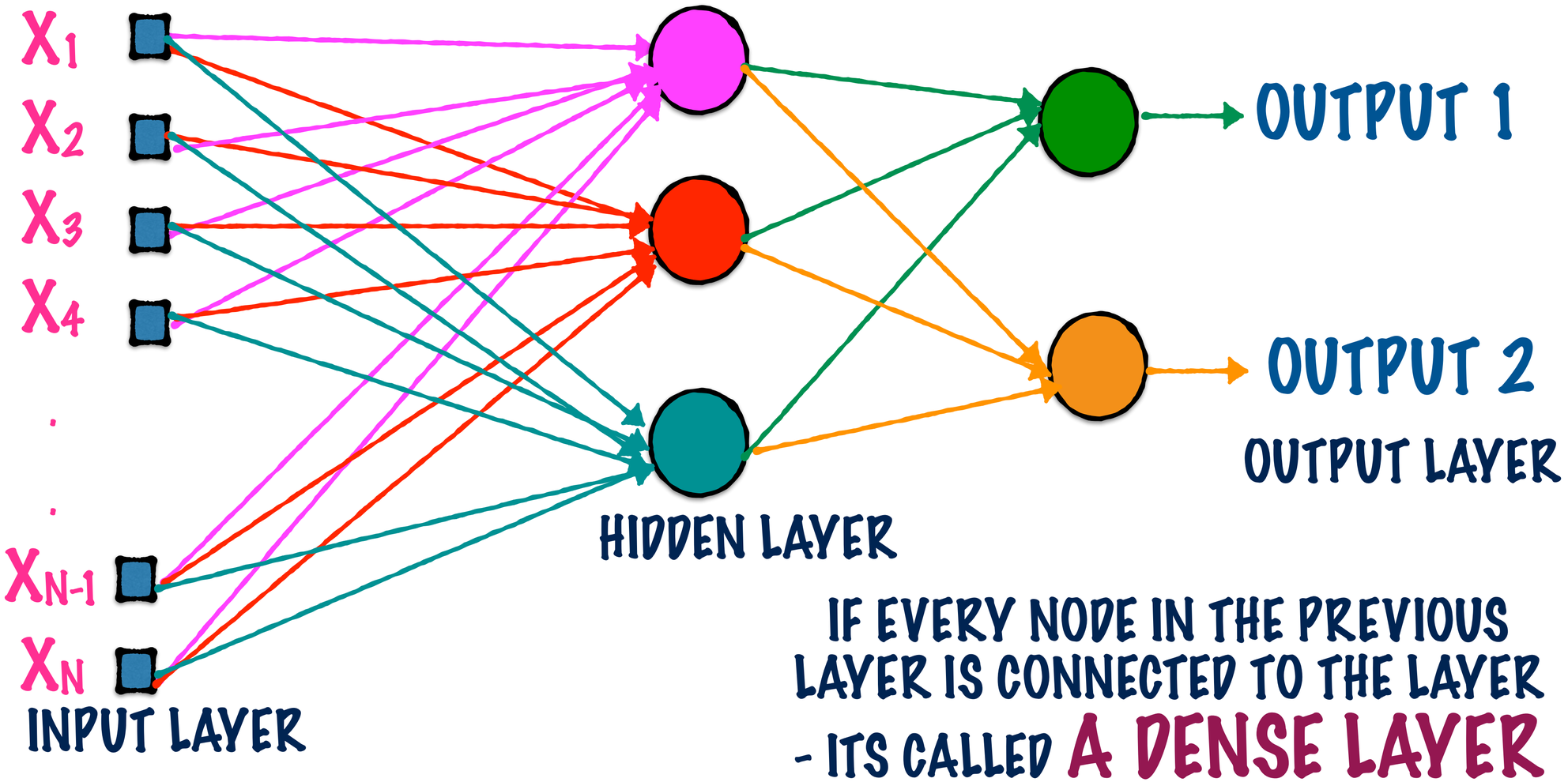
EACH SET OF NODES IS CALLED A LAYER



THIS KIND OF NETWORK, IS CALLED A FEED-FORWARD NEURAL NETWORK

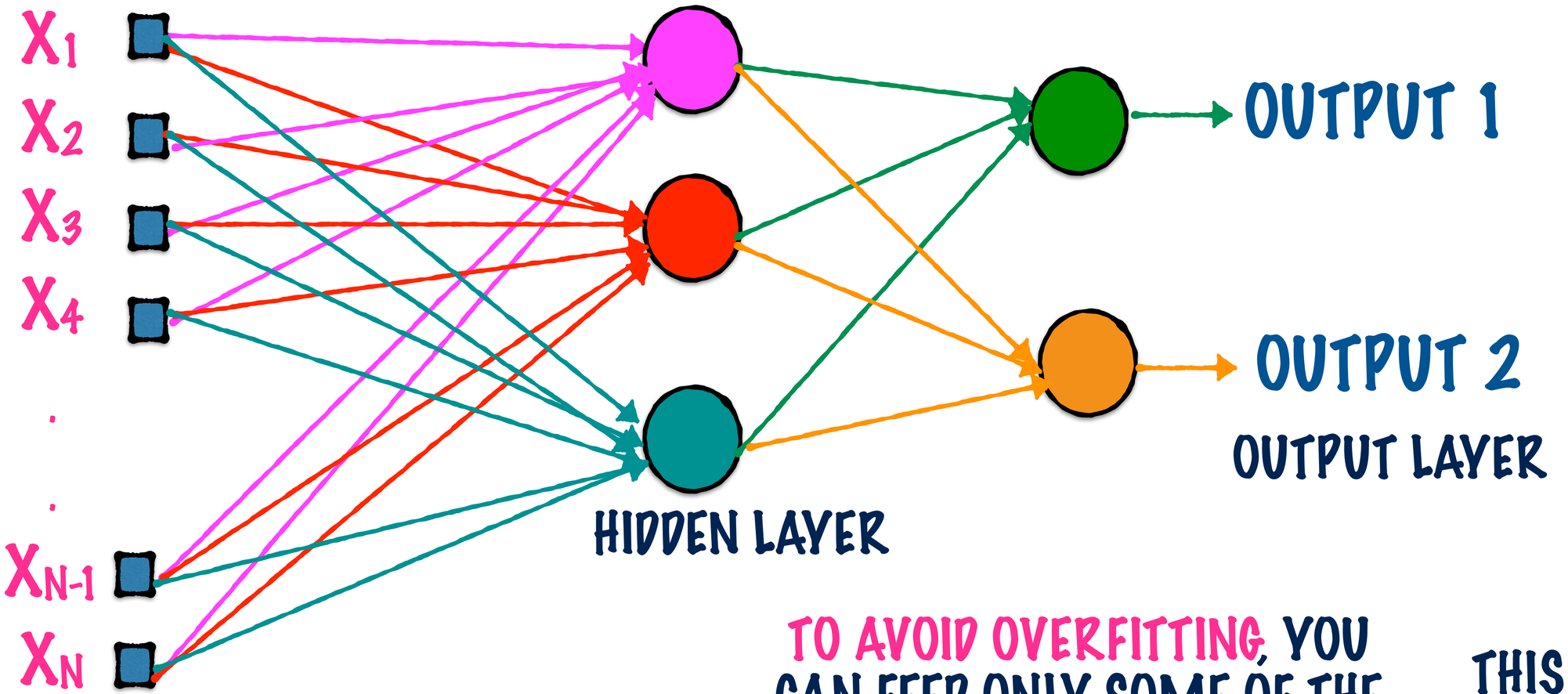
EACH LAYER FEEDS INTO THE NEXT LAYER

EACH SET OF NODES IS CALLED A LAYER



IF EVERY NODE IN THE PREVIOUS LAYER IS CONNECTED TO THE LAYER - ITS CALLED A DENSE LAYER

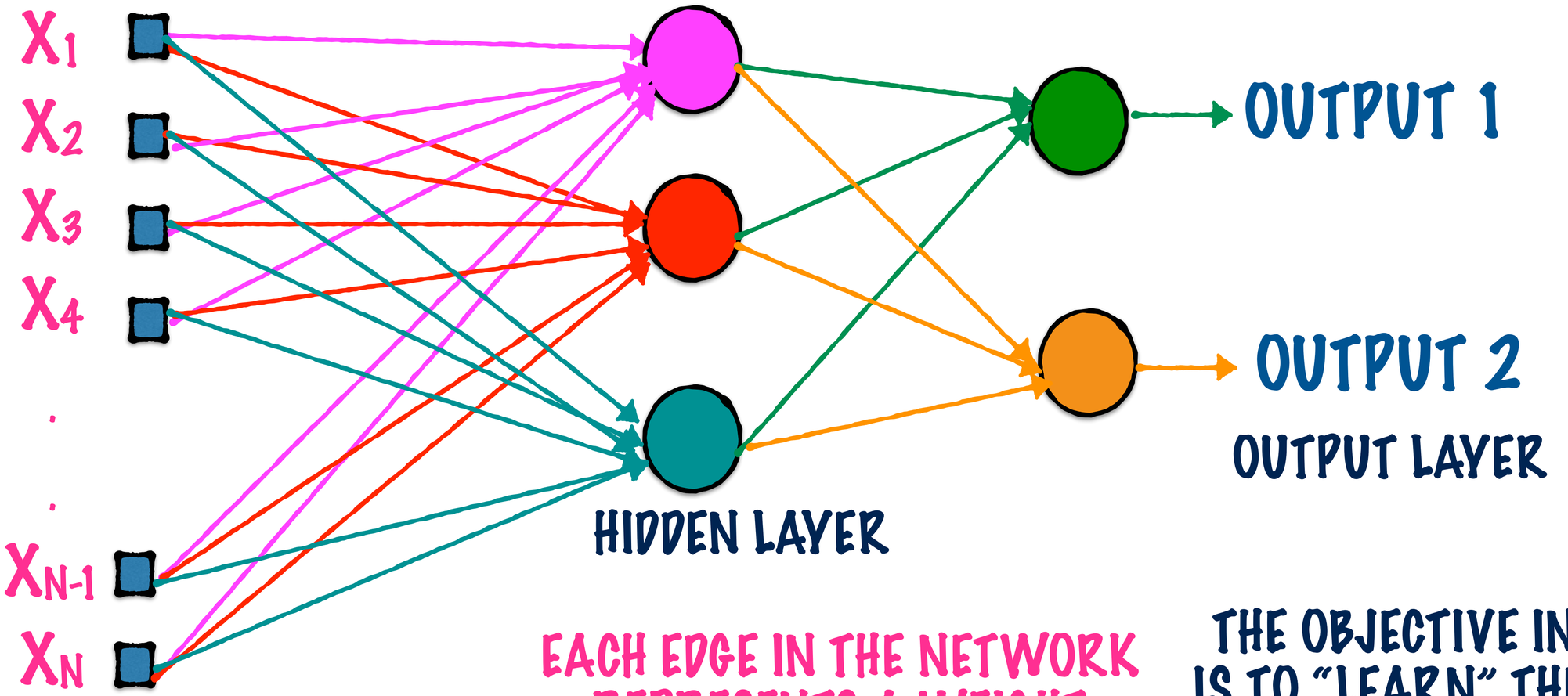
EACH SET OF NODES IS CALLED A LAYER



TO AVOID OVERFITTING, YOU CAN FEED ONLY SOME OF THE INPUTS FROM ONE LAYER TO EACH NODE IN THE NEXT LAYER

THIS IS CALLED DROPOUT

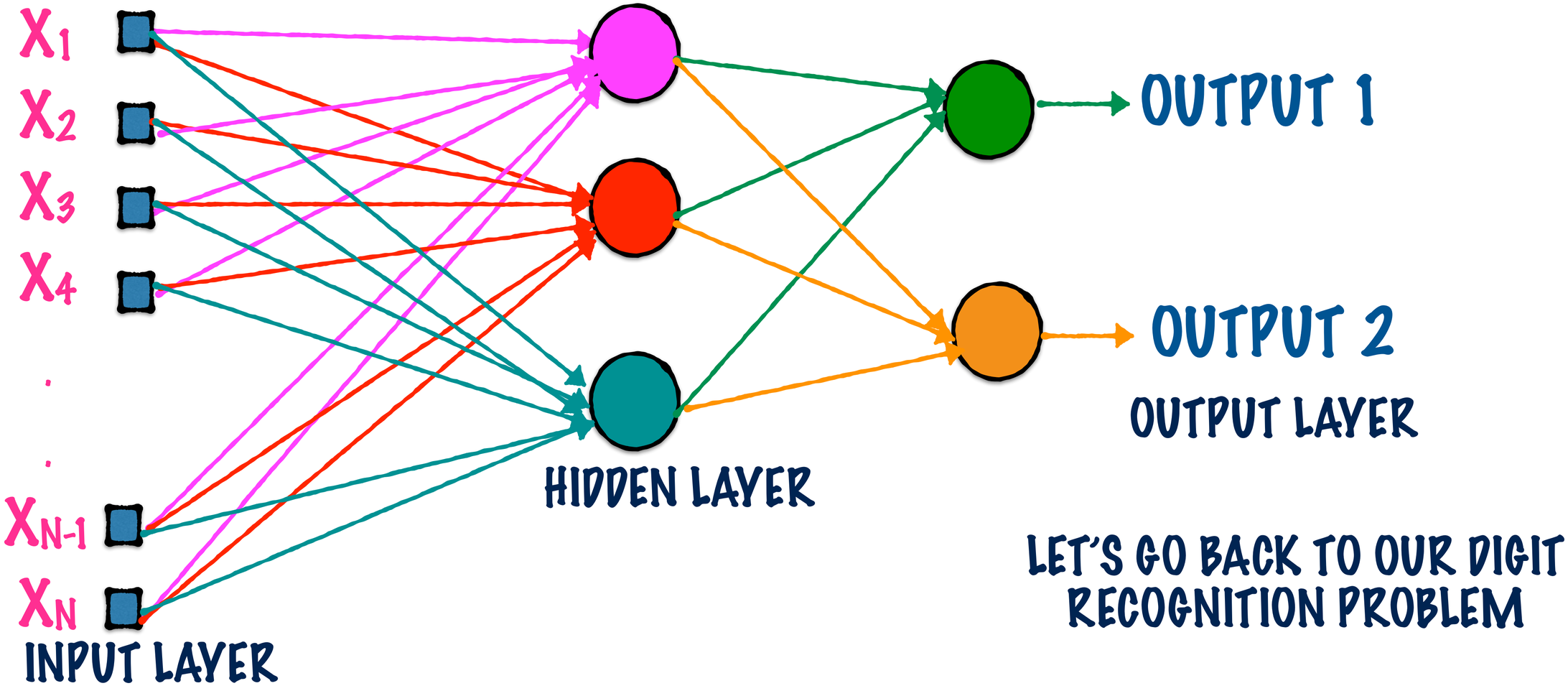
EACH SET OF NODES IS CALLED A LAYER



EACH EDGE IN THE NETWORK REPRESENTS A WEIGHT

THE OBJECTIVE IN TRAINING IS TO "LEARN" THE VALUES OF THESE WEIGHTS

EACH SET OF NODES IS CALLED A LAYER



LET'S GO BACK TO OUR DIGIT RECOGNITION PROBLEM

HANDWRITTEN DIGIT RECOGNITION

MNIST IS A DATABASE OF HANDWRITTEN DIGIT IMAGES

THESE IMAGES CAN BE USED TO TRAIN A DIGIT CLASSIFIER

TO TRAIN A CLASSIFIER YOU'LL NEED TO

✓ 1) REPRESENT EACH IMAGE IN THE TRAINING SET AS A FEATURE VECTOR (A TUPLE OF NUMBERS)

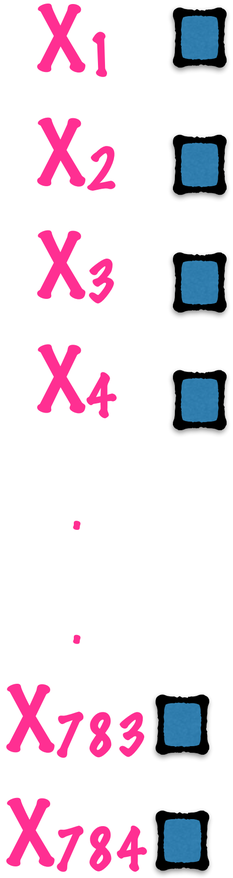
✓ 2) FEED ALL THE FEATURE VECTORS ALONG WITH THEIR LABELS (0-9) TO A CLASSIFICATION ALGORITHM

LET'S WALK THROUGH THE COMPLETE TRAINING PROCESS NOW

WE HAVE A BUNCH OF IMAGES OF HANDWRITTEN DIGITS

EACH HAPPENS TO BE A 28x28 GRAYSCALE IMAGE

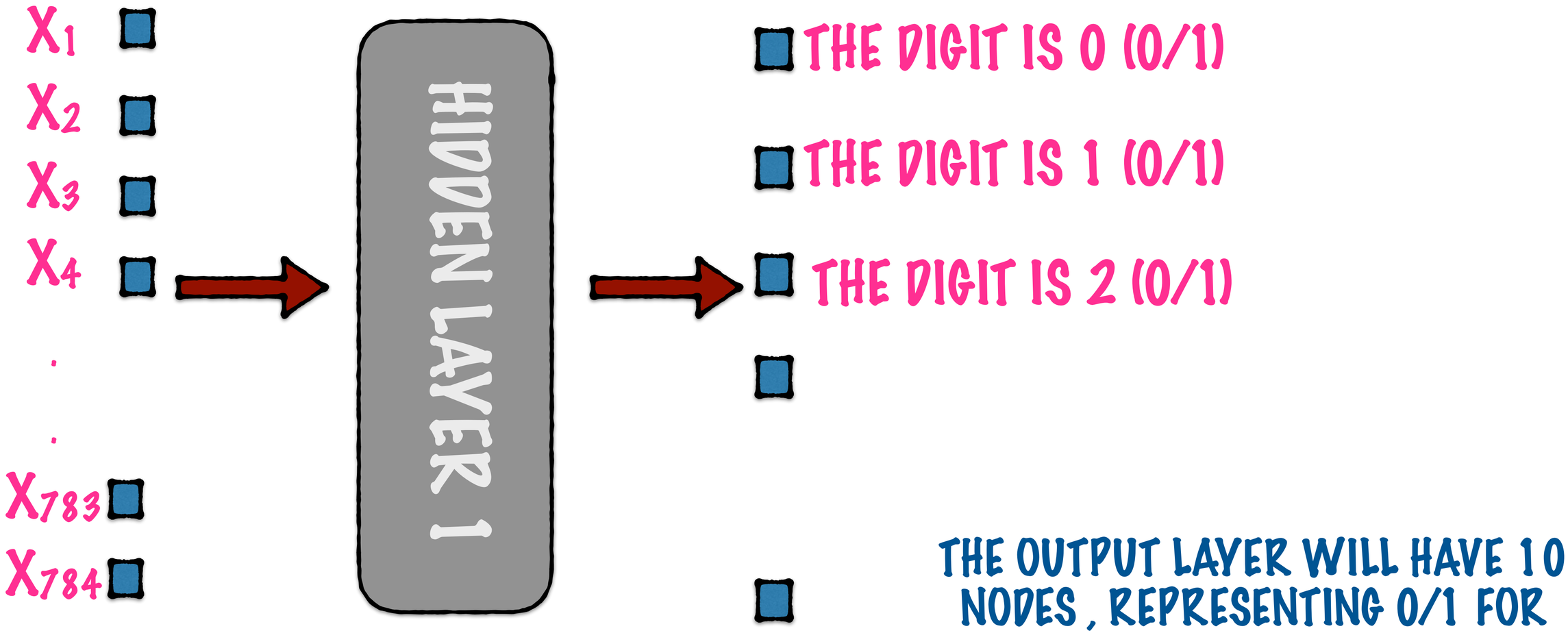
EACH IMAGE CAN BE REPRESENTED AS A TUPLE OF 784 NUMBERS



WE'LL START WITH AN INPUT LAYER OF 784 NODES

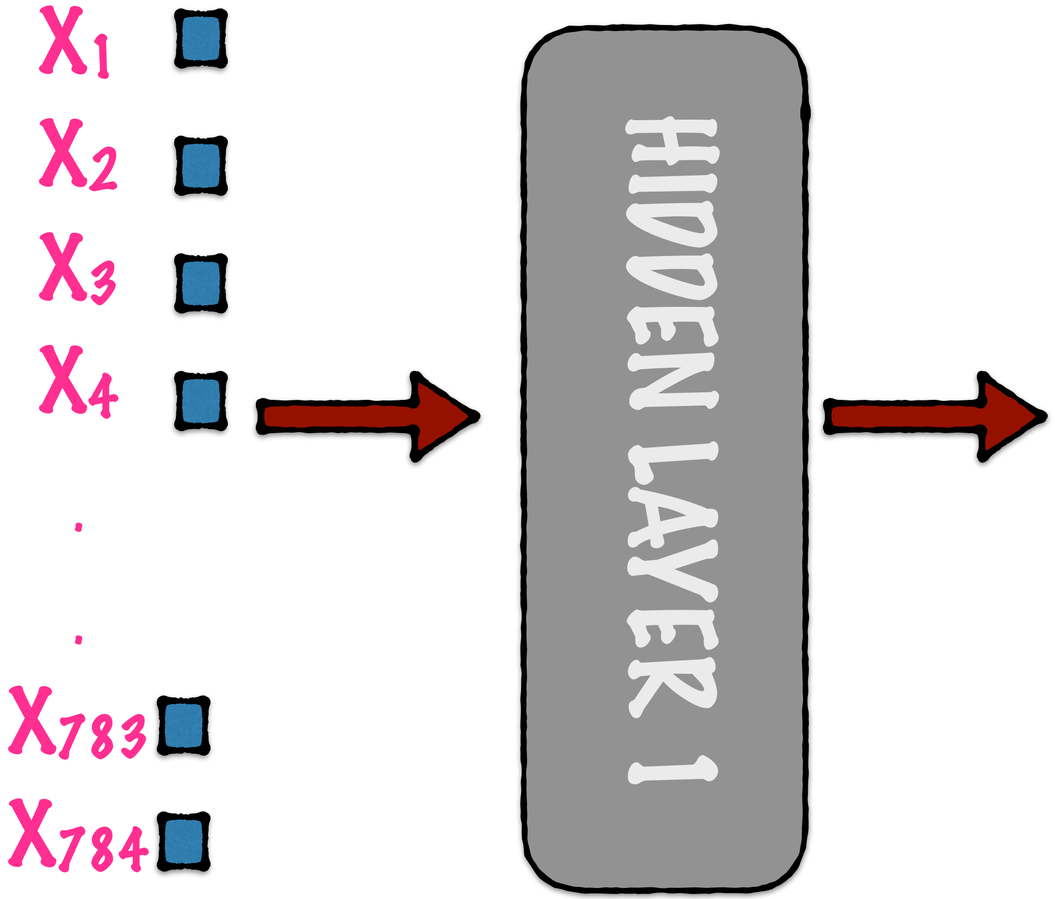
LET'S WALK THROUGH THE COMPLETE TRAINING PROCESS NOW

FEED THESE INPUTS TO A HIDDEN LAYER (OR MORE THAN 1 HIDDEN LAYER)



THE OUTPUT LAYER WILL HAVE 10 NODES , REPRESENTING 0/1 FOR EACH DIGIT

LET'S WALK THROUGH THE COMPLETE TRAINING PROCESS NOW



DURING TRAINING, THE OBJECTIVE IS TO FIND THE WEIGHTS FOR EACH LAYER, WHICH MINIMIZE THE ERROR ON THE TRAINING SET

1) INITIALIZE SOME RANDOM WEIGHTS

2) FIND THE OUTPUT

3) EVALUATE SOME ERROR FUNCTION
EX: (DESIRED OUTPUT - CURRENT OUTPUT)

4) ADJUST THE WEIGHTS BY A SMALL INCREMENT

5) REPEAT STEPS 2-4 UNTIL THE ERROR IS MINIMIZED (OR A CERTAIN NUMBER OF TIMES)