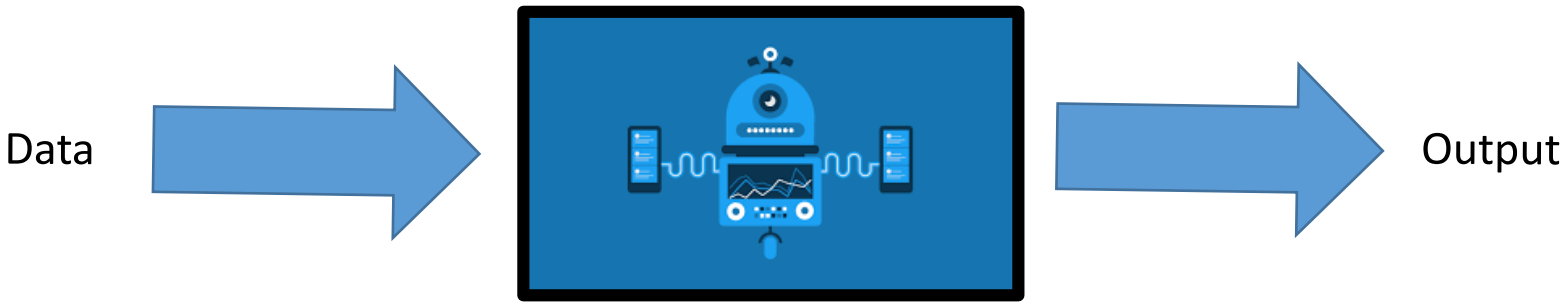
From traditional **Machine Learning** to modern day **Deep Learning**

Aakarsh Malhotra
Ph.D Scholar
IAB lab, IIIT-Delhi

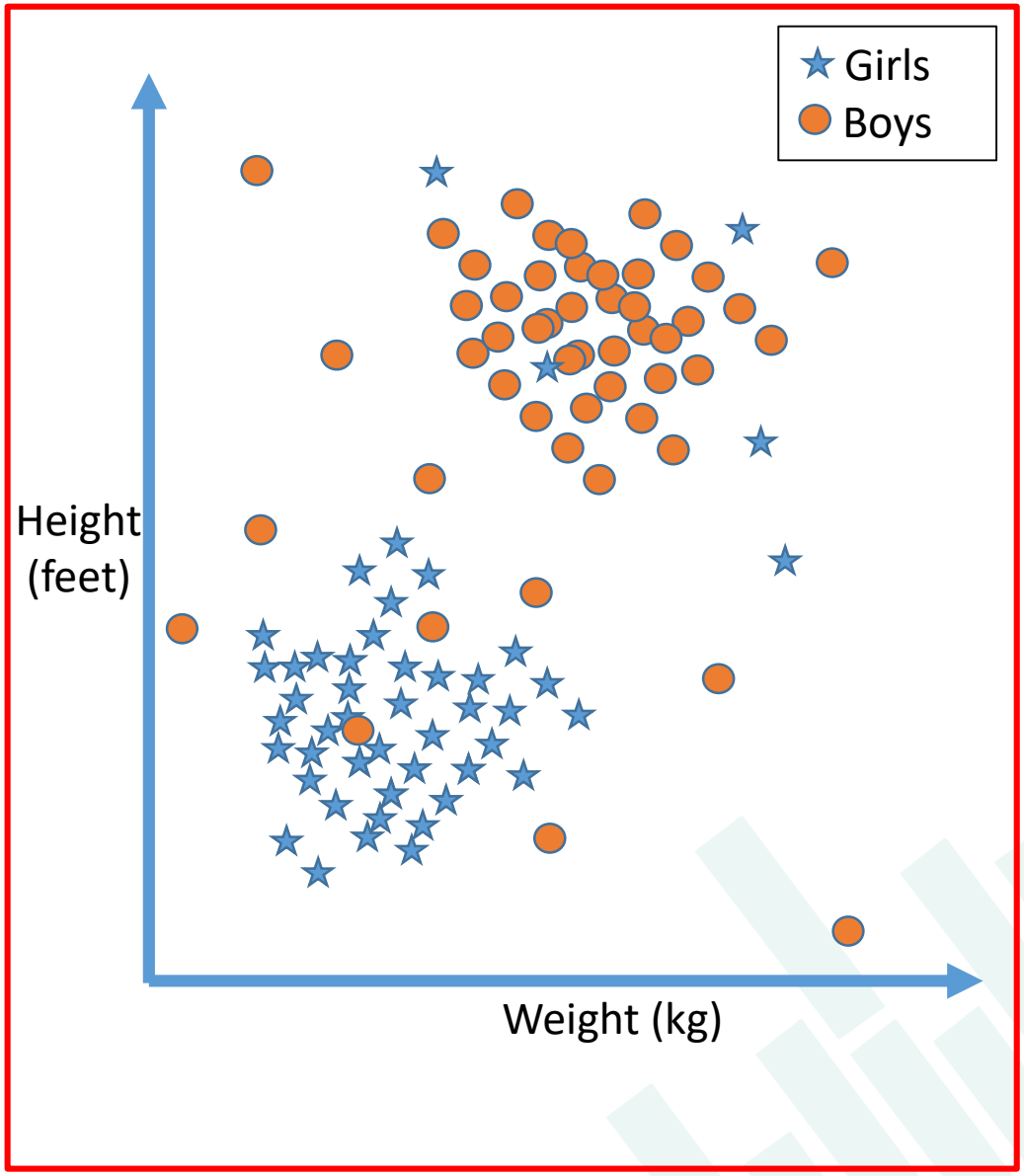
INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Machine learning?



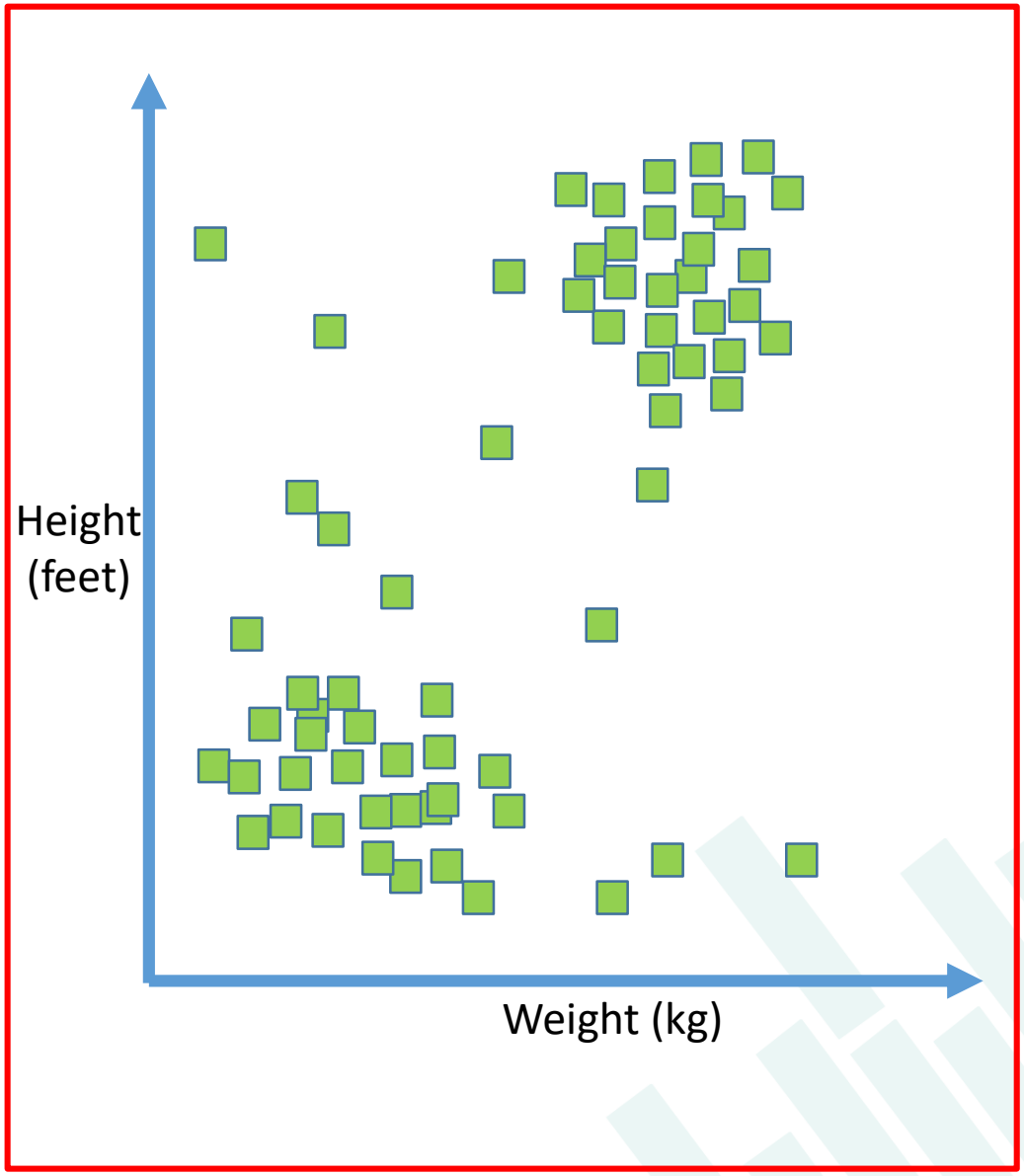
Output: Task specific!

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



Output: Task specific!

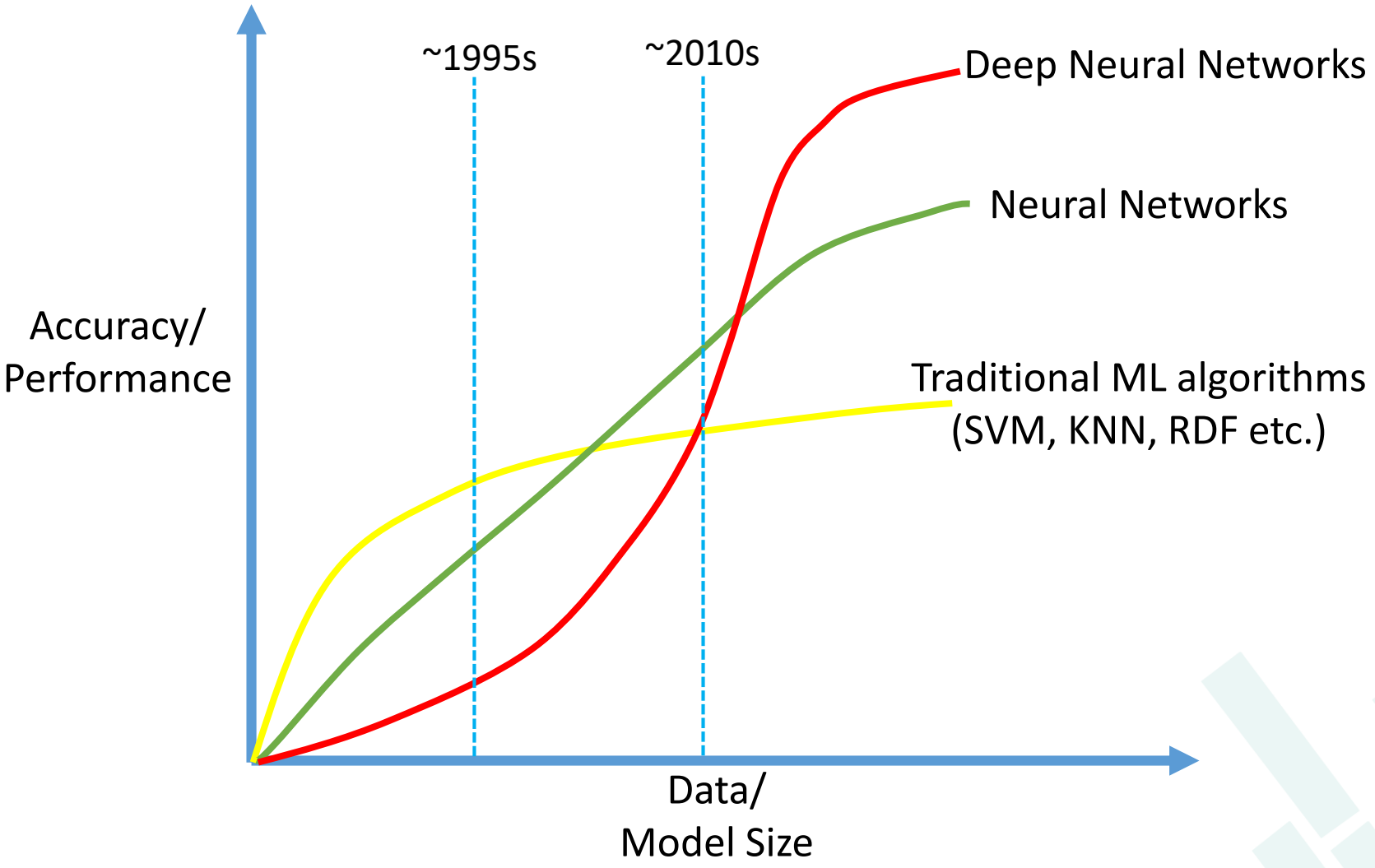
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



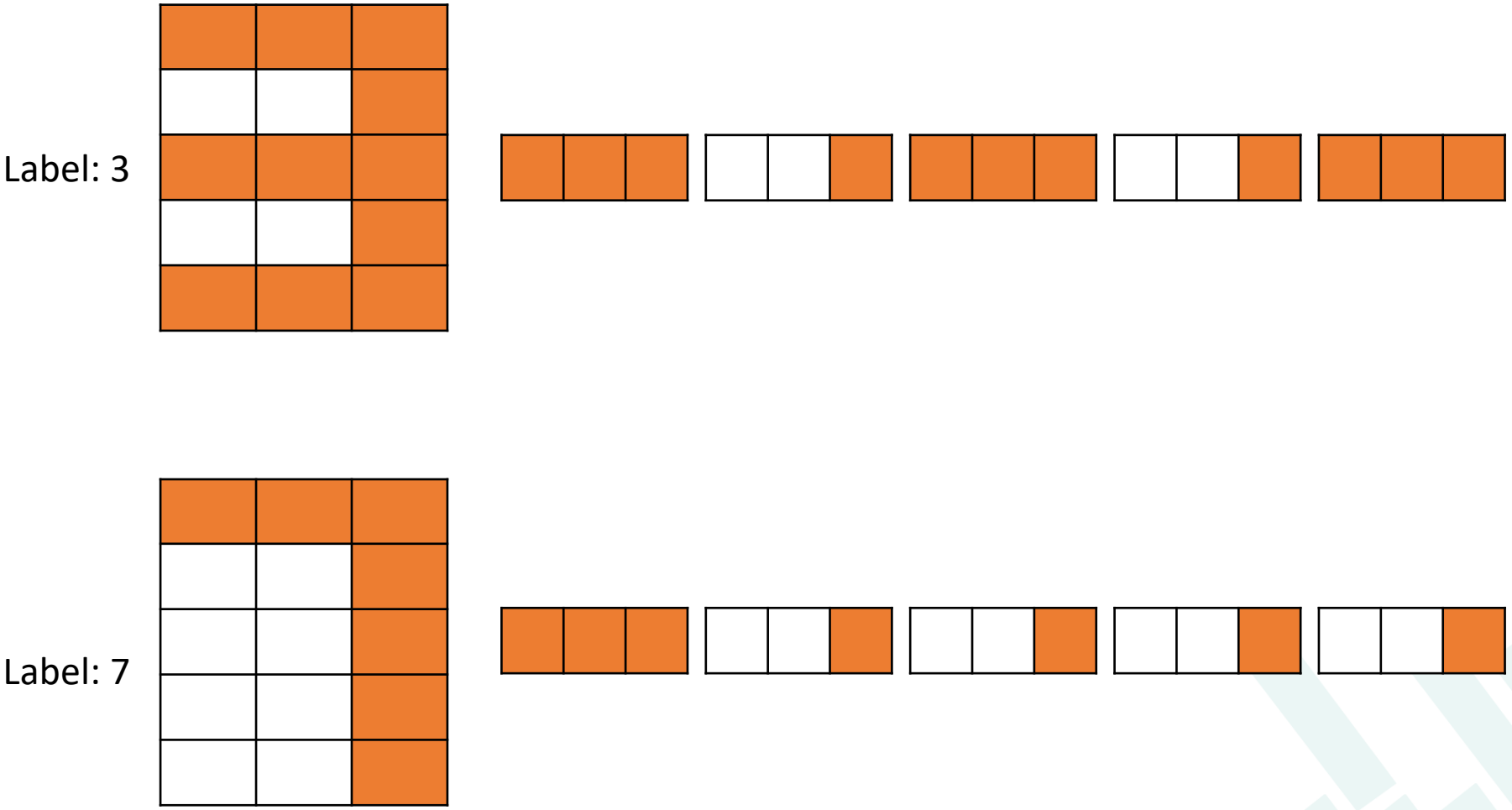
Output: Task specific!

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Dependency on data



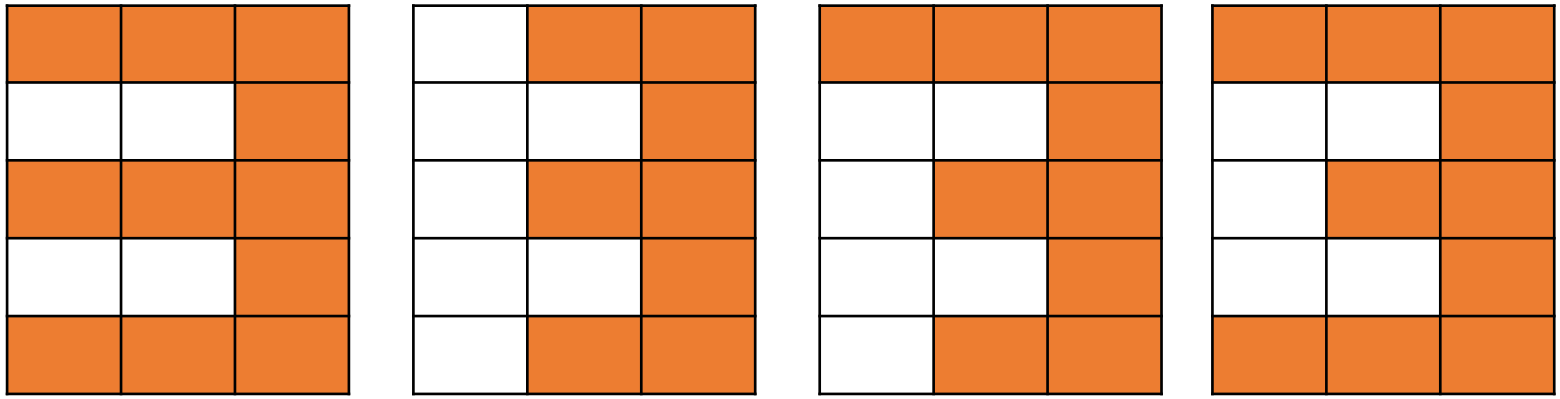
Images: How does algorithms see it?



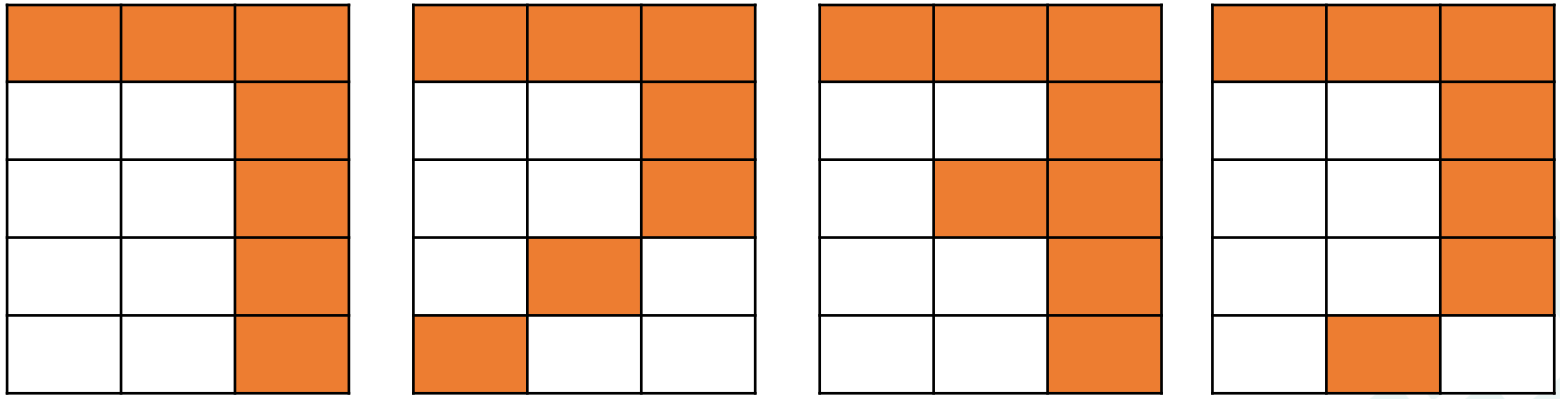
Images: How does algorithms see it?

If we have more data, we can learn more!

Label: 3



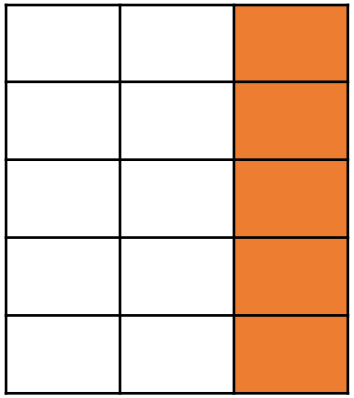
Label: 7



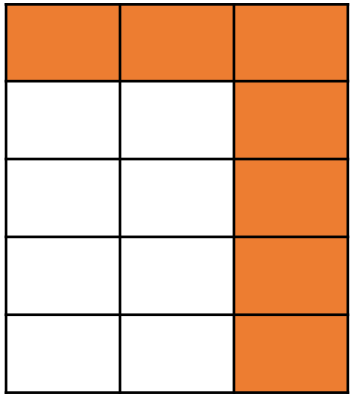
Images: How does algorithms see it?

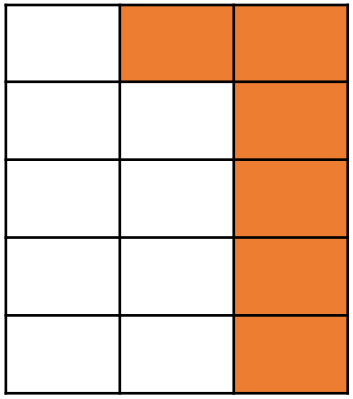
However, things can get difficult!

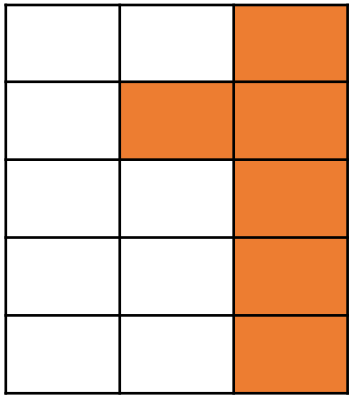
Label: 1



Label: 7







One or Seven?

Random Decision Forest (RDF)

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

What is RDF?

RDF: Random Decision Forests

- Random
- Decision
- Forests → Collection of trees!

- Decision Forests: Trees that take decision!
  aka: Decision Trees

Decision Tree

1 1 0 0 0 0 0

- We have people with:
  - Red or blue colored hair
  - Gender (F/M): 1 or 0
  - Hair short/long (underlined or not)
- Look at the data, 3 types of people:
  - Blue colored males with short hair
  - Red colored males with short hair
  - Blue colored females with long hair

Aim: Split into groups such that different groups are dissimilar.

From a Decision Tree to Forest

1 1 0 0 0 0 0

Color->Number->Underline

Color->Underline->Number

Underline->Number->Color

Underline->Color->Number

Number->Color->Underline

Number->Underline->Color

Underline->Number

Color->Number

Number->Underline

Color->Underline

Number->Color

Underline->Color

Number

Underline

Color

Different combinations and different number of trees learn to classify!

From a Decision Tree to Forest

1 1 0 0 0 0 0

Color->Number->Underline

Color->Underline->Number

Underline->Number->Color

Underline->Color->Number

Number->Color->Underline

Number->Underline->Color

Time for some randomness! Pick 10 trees…

Underline->Number

Color->Number

Number->Underline

Color->Underline

Number->Color

Underline->Color

Number

Underline

Color

Different combinations and different number of trees learn to classify!

From a Forest to RDF

1 1 0 0 0 0 0

Color->Number->Underline

Underline->Color->Number

Number->Color->Underline

Underline->Number->Color

Time for some randomness! Pick 10 trees…

Number->Underline

Number->Color

Color->Number

Underline->Color

Number

Color

Different combinations and different number of trees learn to classify!

From a Forest to RDF

OR 1 1 0 0 0 0 0

Underline->Color->Number

Underline->Number->Color

Number->Underline->Color

Time for some randomness! Pick 10 trees…

Number->Underline

Color->Number

Color->Underline

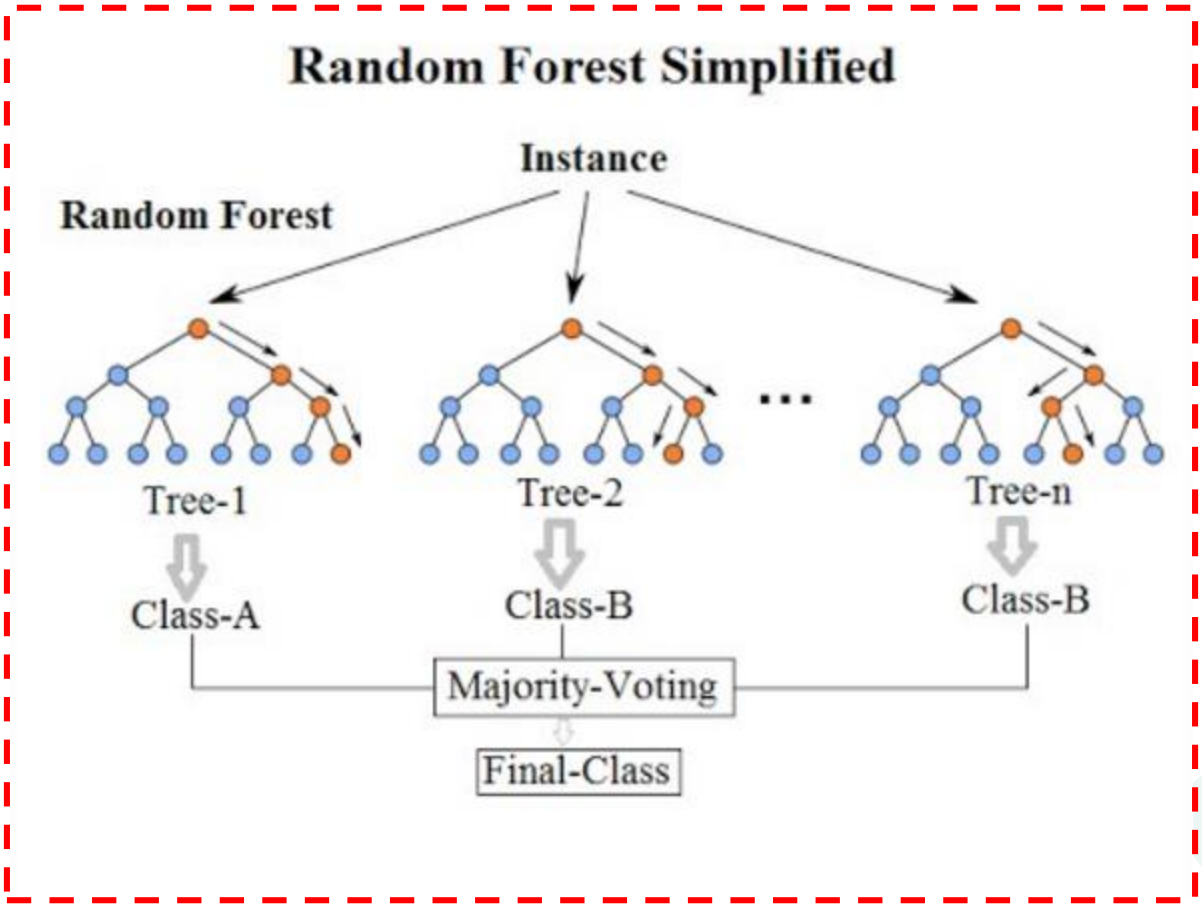
Underline->Color

Underline

Color

Different combinations and different number of trees learn to classify!

What is RDF?


Random Forest Simplified
Instance
Random Forest
Tree-1
Tree-2
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class

Let us code!

Part 1: Loading the data

```
from keras.datasets import mnist
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

(X_Train, Y_Train),(X_Test, Y_Test)=mnist.load_data()
```

```
fig = plt.figure()
for i in range(9):
  plt.subplot(3,3,i+1)
  plt.imshow(X_Train[i], cmap='gray')
  plt.title("Digit: {}".format(Y_Train[i]))
  plt.xticks([])
  plt.yticks([])
```

Let us code!

Part 2: Reshaping the data

- Input Image Dimension – 28x28
- Number of Images – 60,000
- Convert 2D image of dimension of 28x28 into one single vector of 1x784
- Do the above for each image and put these vectors in a matrix
- The output matrix will be of size 60,000 x 784

```
print(X_Train.shape)
nsamples_Tr, dimx, dimy = X_Train.shape
X_Train = X_Train.reshape((nsamples_Tr,dimx*dimy))
print(X_Train.shape)
print(Y_Train.shape)
print(X_Test.shape)
nsamples_Te, dimx, dimy = X_Test.shape
X_Test = X_Test.reshape((nsamples_Te,dimx*dimy))
print(X_Test.shape)
print(Y_Test.shape)
```

Let us code!

Part 3: Train RDF and predict!

```
rfc = RandomForestClassifier(n_estimators=10)
rfc.fit(X_Train, Y_Train)
```

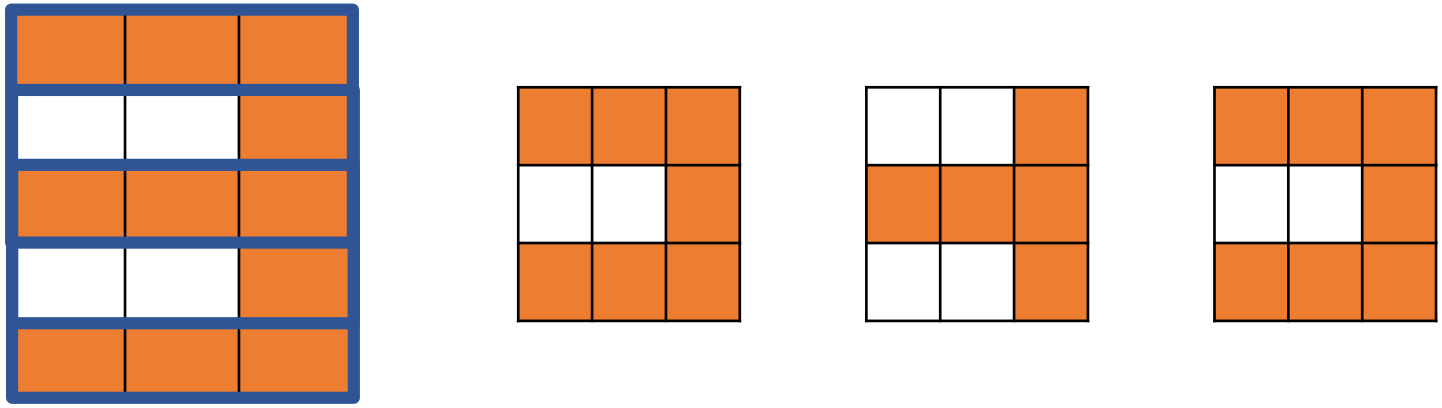
```
Accuracy=rfc.score(X_Test, Y_Test)*100
print(Accuracy)
Predictions=rfc.predict(X_Test)
print(Y_Test)
print(Predictions)
```

Convolutional Networks

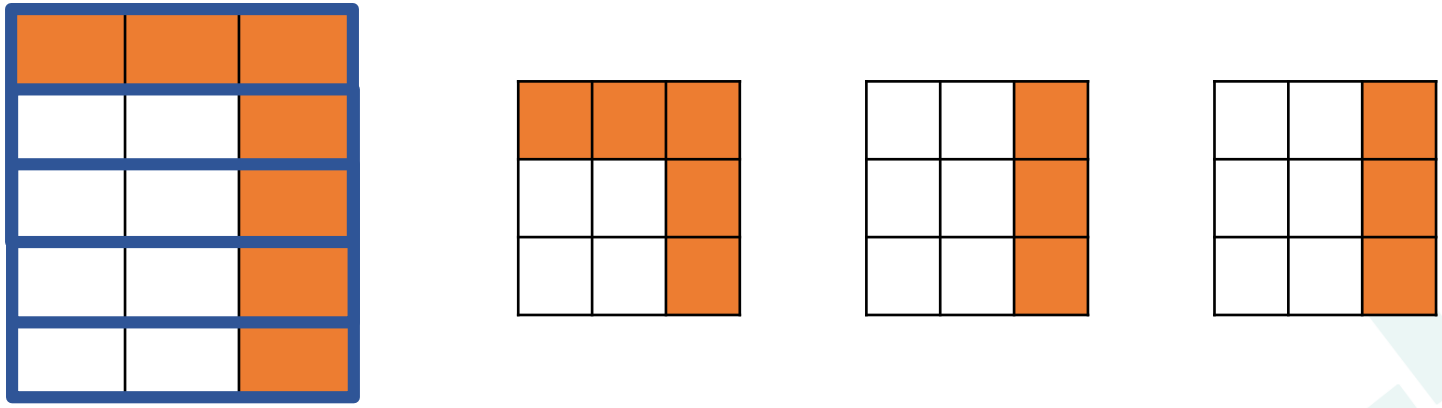
INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Images: How does algorithms see it?

Label: 3



Label: 7



What is Convolution?

- "A mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other"

f

1	2	2	3	3
4	4	5	3	2
1	3	4	5	1
4	1	2	0	1
7	2	1	3	4

*

g

1	0	-1
0	2	1
0	-1	0

=

What is Convolution?

- "A mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other"

f

1	2	2	3	3
4	4	5	3	2
1	3	4	5	1
4	1	2	0	1
7	2	1	3	4

*

g

1	0	-1
0	2	1
0	-1	0

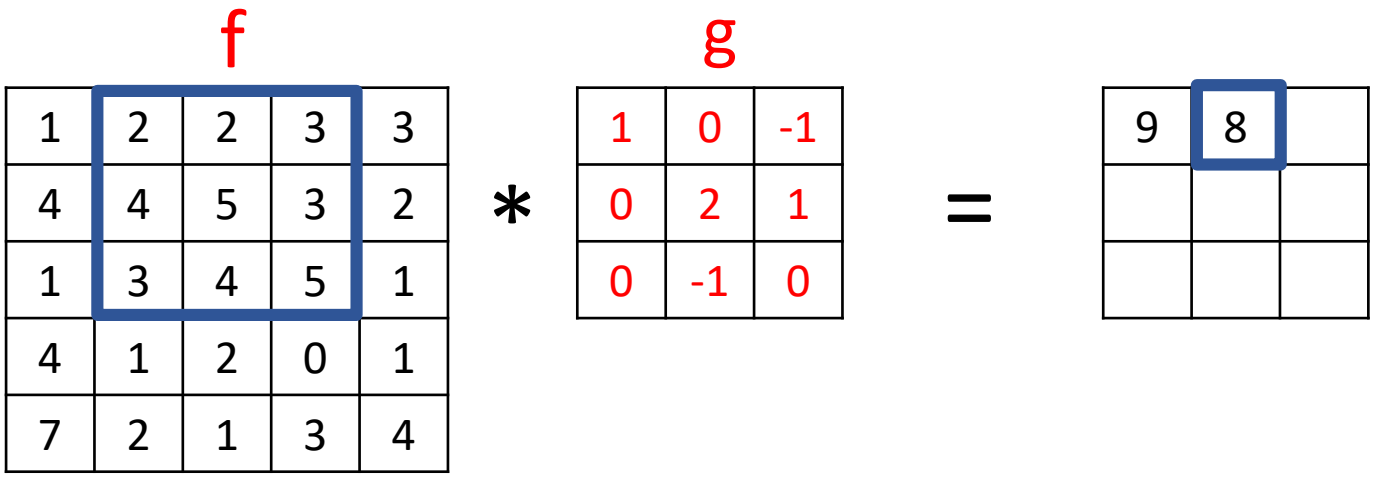
=

9		

Output = 1x1 + 2x0 + 2x(-1) + 4x0 + 4x2 + 5x1 + 1x0 + 3x(-1) + 4x0 = 9

What is Convolution?

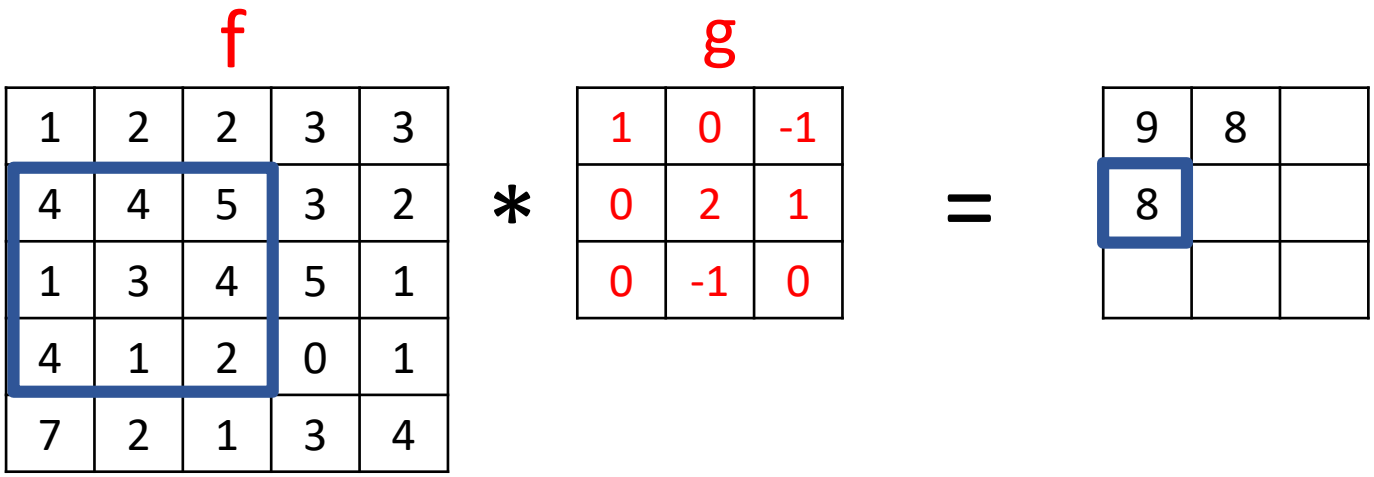
- "A mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other"



Output = 2x1 + 2x0 + 3x(-1) + 4x0 + 5x2 + 3x1 + 3x0 + 4x(-1) + 5x0 = 8

What is Convolution?

- "A mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other"



Output = 4x1 + 4x0 + 5x(-1) + 1x0 + 3x2 + 4x1 + 4x0 + 1x(-1) + 2x0 = 8

What is Convolution?

- "A mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other"

f

1	2	2	3	3
4	4	5	3	2
1	3	4	5	1
4	1	2	0	1
7	2	1	3	4

*

g

1	0	-1
0	2	1
0	-1	0

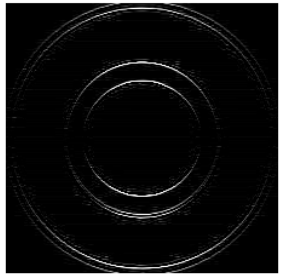
=

9	8	…
8	…	…
…	…	…

Significance of Convolution (in images)

- The filter (g) helps in interpreting particular details in image (f).

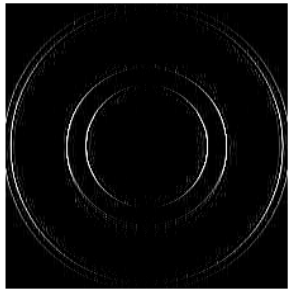
f



g=

-1	-1	-1
2	2	2
-1	-1	-1

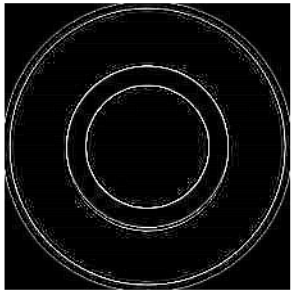
Horizontal filter



g=

-1	2	-1
-1	2	-1
-1	2	-1

Vertical filter



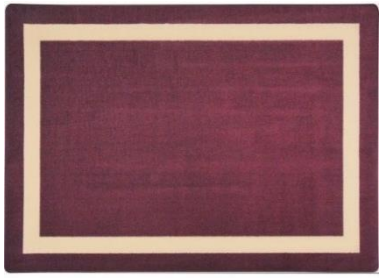
g=

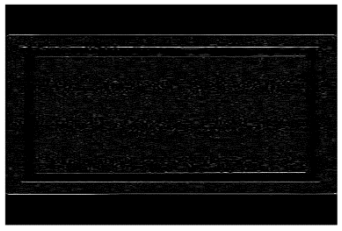
-1	-1	-1
-1	8	-1
-1	-1	-1

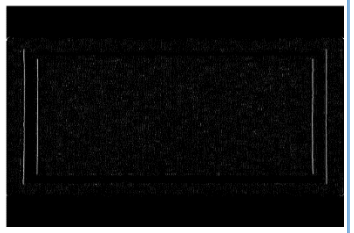
All edges (diag)

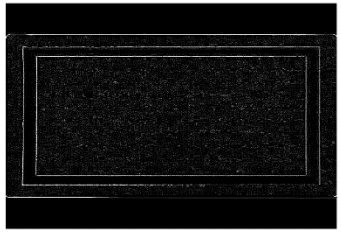
0.0113	0.0149	0.0176	0.0186	0.0176	0.0149	0.0113
0.0149	0.0197	0.0233	0.0246	0.0233	0.0197	0.0149
0.0176	0.0233	0.0275	0.0290	0.0275	0.0233	0.0176
0.0186	0.0246	0.0290	0.0307	0.0290	0.0246	0.0186
0.0176	0.0233	0.0275	0.0290	0.0275	0.0233	0.0176
0.0149	0.0197	0.0233	0.0246	0.0233	0.0197	0.0149
0.0113	0.0149	0.0176	0.0186	0.0176	0.0149	0.0113

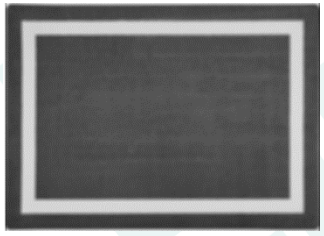
Gaussian Blur



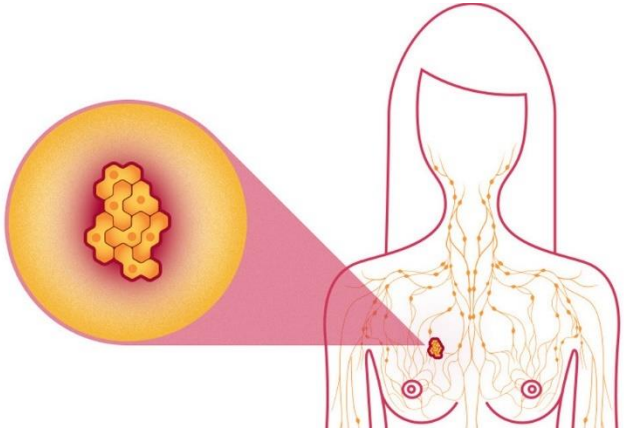








Complex objects?



$$g = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Deep learning models

- Tell the number of filters (g)
- Tell the shape of the filters (3x3, 5x5 etc.)
- Provide lots of data
- Let the modal learn these filters!

Let us code!

Part 1: Loading the data (and importing libraries)

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
```

```
batch_size = 128
num_classes = 10
epochs = 2
img_rows, img_cols = 28, 28
input_shape = (img_rows, img_cols, 1)

(X_Train, Y_Train),(X_Test, Y_Test)=mnist.load_data()
```

Let us code!

Part 2: Reshaping the data

- Input Image Dimension – 28x28
- Convert into matrix of size: 60,000 x 28 x 28 x 1

```
print(X_Train.shape)
print(X_Test.shape)
X_Train = X_Train.reshape(X_Train.shape[0], img_rows, img_cols, 1)
X_Test = X_Test.reshape(X_Test.shape[0], img_rows, img_cols, 1)
print(X_Train.shape)
print(X_Test.shape)
```

- A Labels – single value! -----> [1 x n_classes]
- Convert into matrix of size: 60,000 x 10

```
print(Y_Train[0])
Y_Train = keras.utils.to_categorical(Y_Train, num_classes)
Y_Test = keras.utils.to_categorical(Y_Test, num_classes)
print(Y_Train[0])
```

Let us code!

Part 3: Preprocessing the data

- Convert the data into range: [0,1]

```
X_Train = X_Train.astype('float32')
X_Test = X_Test.astype('float32')
X_Train /= 255
X_Test /= 255
```

Let us code!

Part 4: Create convolutional deep model

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
#model.summary()
```

Let us code!

Part 5: Train, learn, and predict!

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
model.fit(X_Train, Y_Train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(X_Test, Y_Test))
```

```
score = model.evaluate(X_Test, Y_Test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Thank you!!

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI