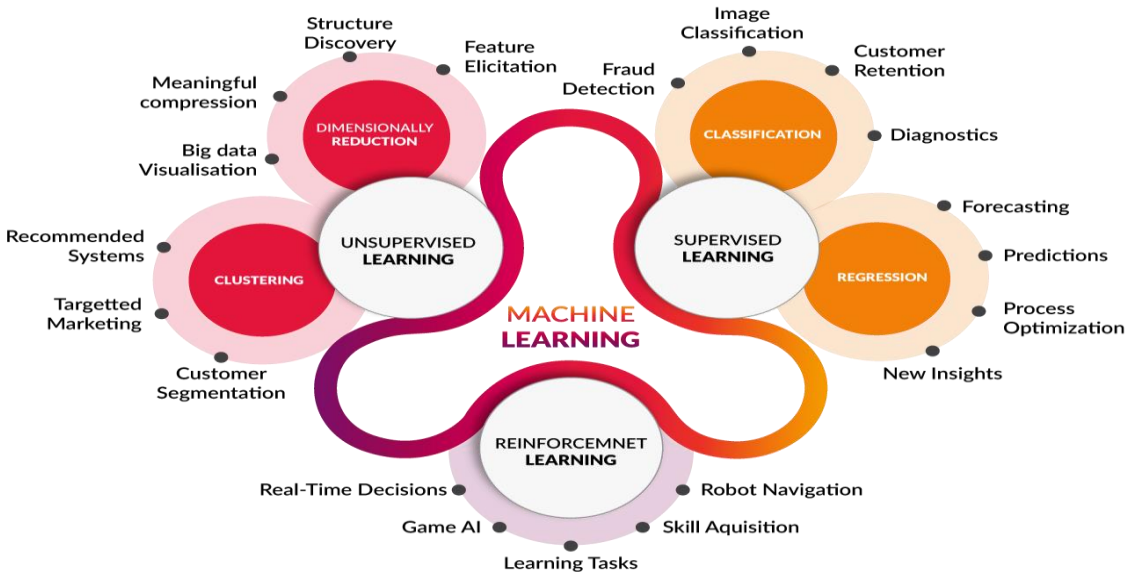
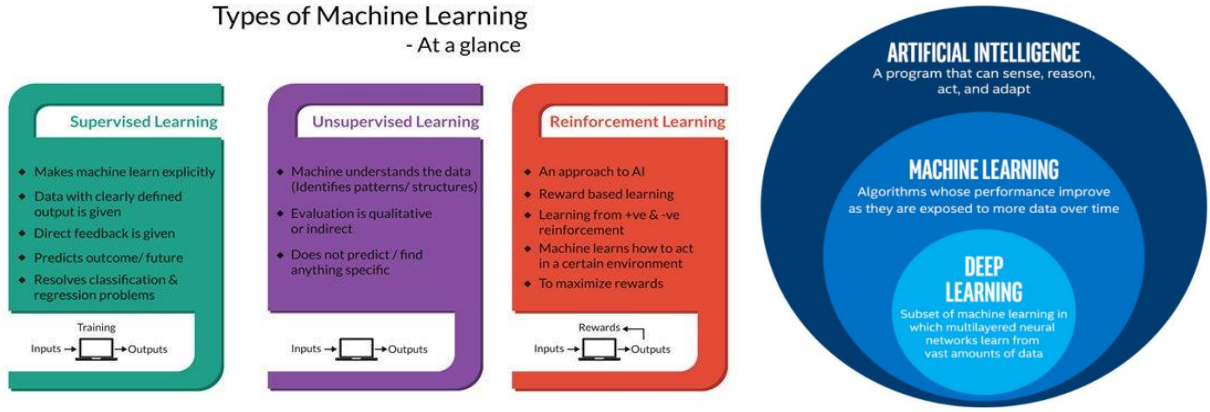
Summary

Computers are exceptionally precise; they can perform billions of well-defined operations accurately and without errors. Writing software consists of specifying precise rules on how to process information. This, however, stands in stark contrast to the real world, where the concepts and rules are changing, noisy, and even inconsistent. Machine learning helps us find patterns and make predictions based on data.

This week was essentially all about learning the different ways of applying supervised learning to any problem. Supervised learning captures the idea of learning from examples and build models that can generalize well. Neural networks are best approach to solve complex problems that are difficult to interpret using simple equation and find pattern that relates input variables to output variables by humans such as categorizing cats and dogs. They have multiple layers that repeats itself to achieve complexity, making them more non-linear. The number of hidden layers is arbitrary and can be adjusted for optimization by trial. Reinforcement learning takes a lot of computational power to let the model learn on its own.





Important Definitions:

- Overfitting is when a model performs really well on the given training examples but does not generalize beyond them (i.e. the green line)
- A model is a function that has tuneable settings (parameters)
- Learning is the process of finding the settings that make your tuneable box as similar as possible to the real one.

Machine learning

ML Learning types:

ML Type 1) Supervised **learning:** Predictive - Uses labelled data (input-output pairs of examples).

- Regression outputs are an infinite set of output, values usually real numbers such as temperature, grade etc.

  To **optimize** our predictive model, we can define a loss function to which we wish to minimize i.e. *Measure the error and tune our predictive model to minimize the error during training*

  - MSE: Mean squared error
  - MAE: Mean absolute error

- Classification outputs are as finite set of output values, usually small set of values or strings such as gender, rank, etc.
  Measuring errors in classification for **optimization** requires metrics based on the prediction of categories, not of real numbers.
  Most obvious metric is:
  accuracy:

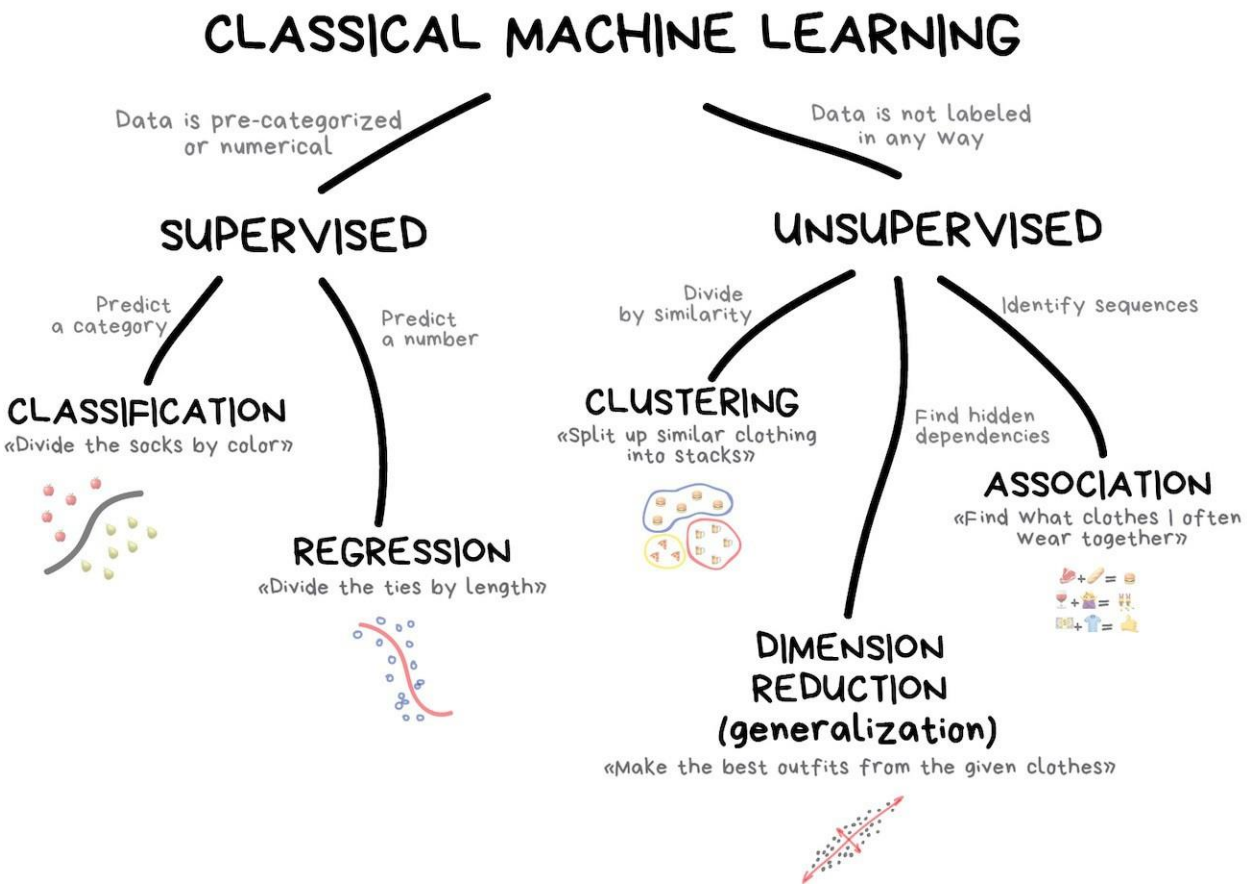
$$acc = \frac{\text{number of correctly predicted examples}}{\text{total number of examples}}$$

  Note: With binary classifications (true or false) one should consider if predictions are false-positives or false-negatives.

  Algorithm 1:
  For example, SVM can be used for binary classification or regression model, it can be linearly separable or non-linearly separable. For a linearly separable case: the SVM has a center line called Hyperplane, it has margins and support vectors. We can go for hard margin and soft margin, but soft margins are usually better. We can use the following kernels: rbf (best), linear, polynomial.

ML type 2) **Unsupervised learning:** Descriptive - No labelled data (no input-output pairs of examples).

(Next Week)

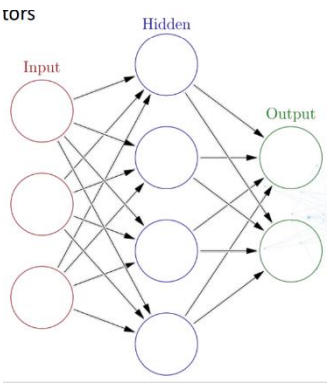


Deep learning

Deep learning is a subset of Machine learning and is used to achieve complex AI. It can be **supervised**, unsupervised, or even semi-supervised (Mostly all the data is unlabeled except few numbers of data set).

- As Supervised: Image classification, object detection, face recognition etc.
- As Unsupervised: Word embedding, image encoding into lower or higher dimensional etc.

    - Artificial neural networks can approximate complex functions. Internal structure of such networks provides multiple levels of abstraction.



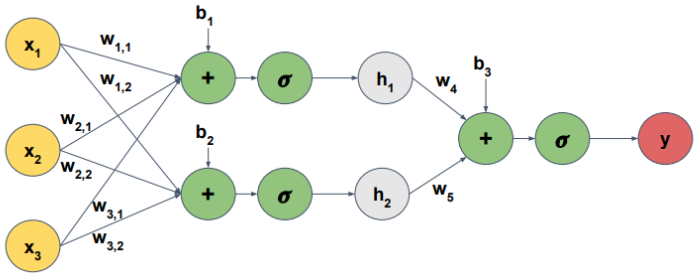
Example:

Neural Networks Terminology:

- Input Units/Layer - all of the input variables
- Weights - the numbers that you multiply variables by before adding things together
- Biases - numbers that you add independently of any inputs
- Hidden Units/Layers - intermediate variables between the inputs and outputs
- Activation Functions - a non-linear function that you apply to intermediate variables

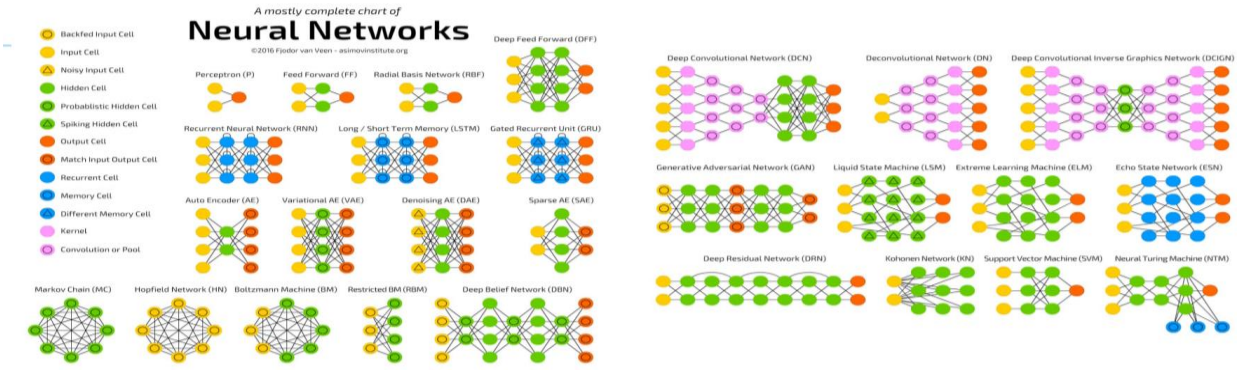
- Output Units/Layer - all of the output variables
- Depth = number of layers - the number of times we repeat the multiplication/addition/activation chain.

NN:



With neural networks, we have inputs weights and a bias variable. The input layer is fed into the model and repeats itself. There is an activation layer (sigma) which converts linear to non-linear model.

When a neural network has multiple layers (i.e is deep) then its call deep neural network and belongs to the branch name deep learning.



Neural networks are a complex model that requires complex data, but complex model overfits with few data!

Solution:

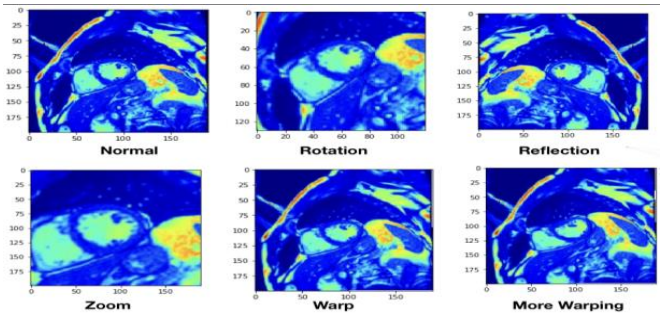
Neural networks require large amount of data to train and if there was insufficient amount of data, we need to augment the data, ie. Apply changes to inputs that doesn't affect the output by creating more versions of inputs (fake data) so our neural networks don't overfit.

Q) What if I don't have enough inputs to train my NN data?

Take the picture and apply valid changes such as:

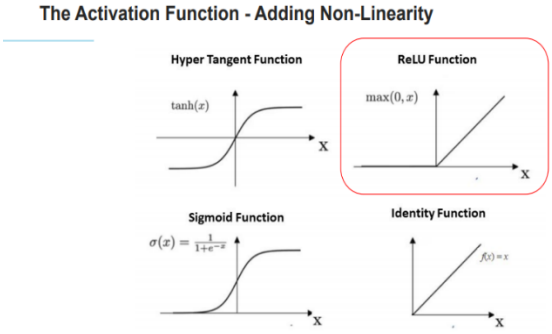
- Changing brightness of every picture.
- Zooming in and out.
- Shifting the picture to multiple directions.
- Changing the resolution of the picture.

Augmentation Example in medical field, we segment different augmentations of heart images in simulation:



Q) How to make the neural network a classification or regression problem?

- If we added an activation function --> it becomes a classification problem.
- The best activation function to use, that works the best all the time is the RelU function having the characteristic below.



<u>With python:</u>

We can use TensorFlow (Google) or pytorch (Facebook) frameworks which makes deep learning super easy to code and run.

Q) Why are neural networks better than regression?

Neural network is better because we can approximate any function

- **2-Layer Neural Network**

```python
import tensorflow as tf

# 2 Layer NN with 3 input units, 10 hidden units, 1 output unit, and a ReLU activation function
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_dim=3, activation='relu'),
    tf.keras.layers.Dense(1, activation='relu')
])
```

- **6-Layer Deep Neural Network**

```python
# 6 Layer NN with 512 input units, lots of hidden units/layers, 1 output unit,
# and different activation function
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(256, input_dim=512, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='sigmoid')
    tf.keras.layers.Dense(100, activation='relu')
    tf.keras.layers.Dense(100, activation='tanh')
    tf.keras.layers.Dense(1)
])
```

Q) How to optimize NN?

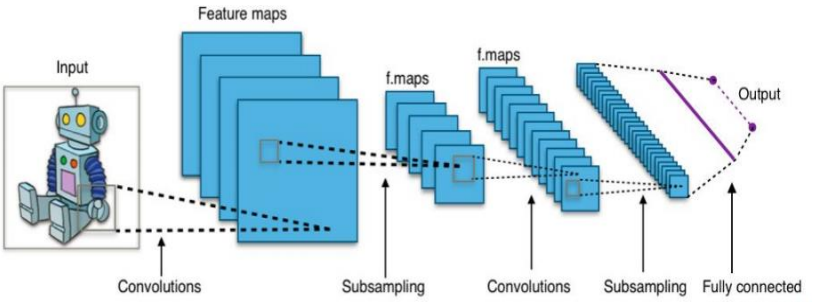
Remember how we optimized regression problems? (tuning?)

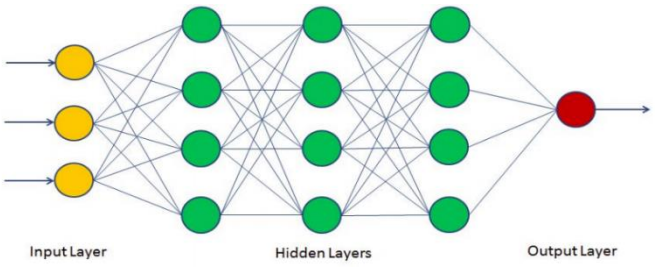
By using **gradient descend** to make how bad it fits as minimal as possible by setting the weights that is best bad.

<u>CNN VS RNN Algorithms:</u>

- <u>CNN</u>: is used for images. For example: Image classification of cats and dogs.
- <u>RNN</u>: is used for images that are related in time. For example: voice, video processing etc.

<u>CNN:</u>





Repeat it multiple times

The 2D vector image is first flattened to 1 D vector image in the convolutional layer. The convolutional layer takes patches of the images and filters the image. The image is down sampled in the pooling layer (i.e: maxpooling). Then it stiches back the picture in the fully connected layer. The layer repeats itself to achieve complexity. Number of hidden layers is arbitrary and we can down sample and remove neurons to make model perform better. This process is known as: dropout

ML Type 3) **Reinforcement Learning:**

- Previous action depends on next action.
- Sequential decision making.
- Feedback loop.
- Uses a lot of memory and training.