# Task 0 – Introduction to Machine Learning

We hope that you have installed all software/libraries. If not, please do that before trying out anything we have here.

In this document, you will learn about **Machine Learning** and related terminologies in brief.

A better understanding of the concepts will only come when you read more about it and perform tasks. For these, we are providing you with the resources. These may or may not be exhaustive. You are free to read from the web or books.

The resources will be categorized as **recommended** reading notes which we strictly recommend you to read and some **optional** topics to enhance your knowledge further.

When we say recommended readings, we don't mean you should try to cram as much of it before looking at anything else. Rather just skim through it first and then as and when required, do read carefully. So, one way to learn these concepts without overwhelming yourself is by doing an ad hoc reading. Learn enough to do only the sub-task at hand. Then again, if required go back to the recommended notes or this document and understand the concept. When done, go back and try to do the task. In short, do an on demand learning. Again, see what works best for you and then adjust.

Most of this guide will contain to-the-point content. We will try to skip textbook definitions and talk in plain english for better and easy understanding of concepts. Without any further ado, let's start !

**What is Machine Learning ?**

In simple words, Machine Learning is all about learning from data. It is the idea that there are generic algorithms that can tell you something interesting about a set of data without you having to write any custom code specific to the problem. Instead of writing code, you feed data to the generic algorithm and it builds its own logic based on the data (source). Machine Learning uses knowledge mostly from computer science, statistics, probability, linear algebra and calculus.

**Why Machine Learning ?**

Its not easy to program features for tasks that are very dynamic and take into consideration a lot of things. For example, take the task of classifying "**Apples**" and "**Oranges**". You might feel the task is very easy for humans to do but at the same its difficult for a computer. If you were to program this by conventional ways, you will need to consider all the different features and conditions for both fruits required to classify them correctly. For example, color, shape, dimensions, etc. Furthermore, the values for these features vary a lot over multiple samples.

But there's a better way. What if we could replicate how we humans learn ? What if we could only give labelled images of "**Apples**" and "**Oranges**" to the computer and it learns to classify them after it sees lots of them ? This is what Machine Learning does and that is why its so cool.

We are sitting on top of lots of data. Also, unlike two decades ago we have faster computers and GPUs now. These and a lot of factors are responsible for making **Machine Learning** and **Deep Learning** a big hit.

**What is Deep Learning ?**

Deep Learning is a branch of Machine Learning just like Machine Learning is part of Artificial Intelligence. In Deep Learning, we mostly make use of Neural Networks, which are generally more than 2-3 layers big. Most of the state of art research related to Machine Learning is happening in Deep Learning with Neural Networks.

Check this out:

1. Video 2 Video (https://github.com/NVIDIA/vid2vid)
2. Everybody Dance Now (https://github.com/nyoki-mtl/pytorch-EverybodyDanceNow)

**Resources for Machine Learning:**

The resources related to Math covered below is mostly high-school and early engineering level topics like linear algebra, calculus and probability. Based on your knowledge and understanding of math, you may or may not want to skip them. What we recommend is to read carefully the brief notes and just skim through bigger ones for a start. Note that you need to get an intuition behind the mathematics and not mug up the formulae or concepts. This will later help you understand algorithms better. Later when really required you can dive deep into specific math topic or carefully revise the needed topics.

1. Introduction to Machine Learning (for brief understanding of ML and its application)
    a) What is Machine Learning ? from Google AI Cloud (**recommended**) (link)
    b) Machine Learning for beginners (**recommended**) (link)
    c) Why Machine Learning matters ? (link)
2. Math for Machine Learning
    a) Linear Algebra (in no particular order, also not necessarily exculsive)
        i. Notes from CS229 course (theoretical; for quick revision) (link)
        ii. An Intuitive Guide to Linear Algebra by Better Explained (link)
        iii. Basic Linear Algebra from Towards Data Science on Medium (link)

iv. Videos by 3Blue1Brown (**recommended but not necessary for Task 0**) (link)

v. Linear Algebra with Numpy from Towards Data Science on Medium (link)

b) Calculus

i. Videos by 3Blue1Brown (**recommended but not necessary for Task 0**) (link)

ii. Notes from CS229 course (theoretical; for quick revision and quick review; calculus starts towards the end) (link)

c) Probability

i. Notes from CS229 course (theoretical; for quick revision) (link)

ii. Probability by Khan Academy (link)

**Types of Machine Learning Algorithms**

Machine Learning algorithms are classified into three categories.

**1. Supervised Learning**

In Supervised Learning, you give labelled data to the algorithm. So let's say you are classifying "**Dogs**" and "**Cats**" images, each of the images will be labelled as **Dogs** and **Cats**. The algorithm will learn from labelled data and later classify unlabelled examples that it gets as an input.

**2. Unsupervised Learning**

In unsupervised learning, you only give the data to the algorithm but no labels at all. For example, you are given a set of images and you want to cluster these images based on the similarity of images. This is an unsupervised learning task.

**3. Reinforcement Learning**

In Reinforcement Learning, for every example or data sample, you get time delayed labels called **rewards** for the actions that you take. So there are no correct or incorrect labels but only how well you did the task. These rewards are called **positive** or **negative** reinforcements. For example, consider the task of balancing a stick on a robot. The robot will move back and forth, and try to balance the stick. At every instance, the action you take (move back or forth) only gives you a reward either positive or negative, which tells you how good or bad was the action that you took.

**Resources:**

1. Types of Machine Learning methods (link)

**Machine Learning in Real Life:**

Let’s see what does it mean to apply machine learning algorithm. You can find what particular terms mean just below.

**The flow:**

**1. The Problem Statement**

Definition of the task or problem at hand. Carefully read it and see if the task can be easily solved or will require machine learning. If latter, go ahead with machine learning.

**2. Search for datasets**

Look on the web for datasets. Contact people who you think might have this dataset after your research on the web. The datasets could be SQL databases, CSV files, Excel files, images, etc. Depending upon the type of task, whether supervised, unsupervised or reinforcement learning, you will have labels associated with the datasets.

**3. Data pre-processing**

Here, you try to clean the dataset obtained. For example, you **replace fields in rows with no or garbage values with 0 or average across that particular field/column** in an excel sheet.

Normalizing the data is done so that all input variables have the same treatment in the model and the coefficients of a model are not scaled with respect to the units of the inputs.

For instance, consider that you have a model that measures the aging of paintings based on room temperature, humidity and some other variables. If humidity is measured in litres per cubic metre and temperature in degrees Celsius, the coefficients in the model will be scaled accordingly and may have a high value for humidity and a low value for temperature (say). If you scale the variables, such as by using **$(x-\mu)/\sigma$** or any other technique, variability in output due to unit change in input variables will be modeled more realistically (source). We then divide the data into Train and Test set.

**4. Building the model**

Here, we specify the model architecture we want to use. For example, logistic regression or SVM or neural networks, etc.

Normally, when building the model we do this:

1. We use simplest model we can use.

2. See the performance by adjusting the hyper-parameters.

3. The model will mostly underfit since its the least complex model.
4. We will select another model and start again from step 2.

We do this until we have a model which doesn't overfit and gives the most accuracy. An alternative to all this trial-and-error could be to use models that other people have used for similar problems and found out good accuracies.

**5. Training and evaluating the model**

Here we train the model. We make a forward pass with data on the model and a backward pass to adjust weights. We do this couple of times till we have good training and testing accuracy.

**Common Machine Learning terminologies:**

We will now look at some terminologies common to most machine learning applications.

**1. Dataset**

Dataset is the data used for machine learning application. For example, set of images used for classifying "**Dogs**" and "**Cats**". An excel spreadsheet of employee salaries, work experience and other details used for predicting salaries. Normally, tabular data is available as a CSV file and images can be present in folders with folders having the names of classes (like a folder of **Dog** and **Cat**).

**2. Model Architecture and Model Weights**

The term **model** is used equivocably multiple times. A model can mean a model architecture or the model weights. Model architecture is the type of algorithm that we apply on our data. A model architecture uses random weights in the beginning but as it is trained on data, it learns better weights.

**3. Training and Testing Phases**

Training phase is when we are improving our model by changing the model weights. We do this by showing new data to our model and correcting model weights to improve prediction or classification task.

Testing phase is when we are only predicting and classifying, in short when the machine learning application is on field.

**Note:** Sometimes it's possible for the application to be trained even when on field.

**4. Train and Test Data**

Our learning models require a unit that can quantify the overall performance of our model on our dataset. Our training phase relies heavily on our dataset and on our defined parameters. However, using the entire dataset would mean that our goal seems to be fitting our model to our dataset alone, which was never the objective in the first place. Exhaustively utilizing the dataset and tweaking the parameters would possibly result in an ideal model and not a prediction model.

One good way to avoid this would be to split our dataset into two parts of some defined ration - a training set and testing set. We never show test dataset to our model and train our model by using only the train dataset. This allows us to evaluate our model accuracy on unseen data.

## 5. Weights and Biases

Weights are model weights. These are just floating point numbers (normally). During training these numbers are changed and towards the end a good collection of these numbers are obtained. Biases are also floating point numbers which are similarly set during training.

For example, $\boldsymbol{y = Wx + B}$

Here, $\boldsymbol{W}$ is weight and $\boldsymbol{B}$ is bias. Using these weights we calculate result. This exact example is example of linear regression where in we fit a line on our training data.

The training data is just a table, with lots of ***x1***, ***x2***, ***x3…*** and **only one *y***. The ***x's*** are called **training variables** and ***y*** is called **prediction variable**. If there is only one ***x*** we call it **linear regression in one variable**. For **2 or more than 2 *x's***, we call it **multivariate linear regression**.

## 6. Forward pass and Backward pass

Forward pass is calculating the predictions. Backward pass involves resolving the error in predictions by changing weights. Forward pass happens in both training and testing phase. But backward pass only happens during training.

## 7. Batch, Mini-batch and Epoch

A batch is collection of entire training data samples or testing data samples. Mini-batch is subset of this training data. An epoch is running forward and backward pass on whole of the training data.

## 8. Error and Loss

Error is the difference between observed prediction and actual prediction of the algorithm. There are different ways to calculate this error. Loss is defined as average error over all examples. Loss can be either **training loss** or **testing loss**. Training loss is calculated over all or mini-batch of training examples. In short, loss is a parameter that explains accuracy of our model. The less the loss, better the model.

### 9. Optimization Algorithm

You need to find better weights such that the average error or loss over entire dataset becomes very low. So how do you find these sets of weights (weights are just matrix of numbers) ? You find it with an optimization algorithm. One very naive thing you can do is keep trying different sets of weights and check which one gives least loss. But this will take very long for you to find correct weights. We need an optimization algorithm to solve this problem.

### 10. Overfitting (Variance) and Underfitting (Bias)

Overfitting happens when your model just remembers all the data that you have. This is a common trait to see, if your model is complex and data is small in quantity. When this happens the training loss reduces while testing loss keeps increasing. Underfitting is when your model cannot fit the training data. This happens if model is not complex enough to model the data properly. To know more, follow this link.

### 11. Regression and Classification

In Regression, the output variable can take any real values. For example, predicting house prices. Since house prices can be any real number within a range, it is a regression problem. One thing to notice is that the number itself has some value in case of regression. For example, a lower number for house price suggest that the price is low.

In Classification, the output variable can take categorical value. For example, in predicting house prices one can say if the house price is low, medium, high, very high. These labels can be represented by numbers 1, 2, 3, 4. Note that these numbers are just classes and there is not much significance in their value.

**Resources:**

1. Andrew Ng’s course on Intro to Machine Learning on Coursera (link) upto section 3 (which is logistic regression and regularization) (**recommended but not necessary for Task 0**) or CS229 course upto convex optimization (link) (**recommended but not necessary for Task 0**).

2. Machine Learning series for beginners (link). This covers everything a beginner might have trouble with like terminologies, algorithms, etc. Chapters are similar to Andrew Ng’s course so it can also act as a refresher if you had taken that course earlier (**recommended but not necessary for Task 0**; only first few chapters for Task 0).

3. “Machine Learning is Fun” series by Adam Geitgey on Medium (link). This might act as a superset for the above one. This doesn’t delve into ultimate specifics but is good for understanding. Also covers non-trivial things like face recognition, etc. (**recommended but not necessary for Task 0**).

**Curse of dimensionality:**

In machine learning, there is a serious requirement of data. Let's say you have just a single attribute to predict a label, you should have data for that attribute such that the data sufficiently covers the space of that attribute. For example, if the range of attribute goes from 0 to 100 in real values and if you have values only upto 50 in test set, your algorithm won't work properly. Thus, the data points should cover the space effectively.

If there are two attributes i.e. two dimensions, now you need enough data to cover this 2-dimensional space to predict. Thus, you can see that the requirement of data increases by power of dimensions.

So, if the requirement of data for 1-dimension is **D** points, for **P**-dimensions, it is $\mathbf{D}^{\mathbf{P}}$ points.

**Some common Machine Learning algorithms:**

1. **Nearest Neighbour Classifier**

   **Nearest neighbour classifier** is a **supervised learning technique** which is used for **predicting class labels**.

   In nearest neighbour, every data sample (a row of data) is going to become a point in space. There are two sets of data, train and test data. The train data is like our knowledge base. The test data is the data that you wish to provide predictions for. Now this test data sample will be converted into a data point in N-dimensional space where N is the number of attributes in the data sample. The point from our train set which is most near to it is considered and label of that point is predicted as the label of the test data. So, there are actually no parameters being learned.

   **K-NN** is a variant of nearest neighbour algorithm which considers maximum of **K** nearest data points to classify a label. Here, **K** needs to be selected by the user which makes **K** a hyper-parameter. So for example if **K = 5** and you have to predict **cats** and **dogs**, if **K** nearest points have labels like **{cat, dog, dog, cat, dog}** then the answer is **dog**.

2. **Linear Regression**

   **Linear Regression** is a **supervised learning technique** which is used for **predicting values**.

   In linear regression, we try to predict a parameter (lets call it ***Y***), when we are given several other parameters (one or many, lets call them ***X1***, ***X2***, ***X3***). ***Y*** could be either a linear function or non-linear function of some of the given parameters.

   In linear regression, we try to fit estimate the parameter ***Y*** with a linear function

   $Y = W*x + b$. Here ***W*** and ***b*** are **weights** and **biases** respectively.

If **X** is just a **single value** or **scalar**, we call it **linear regression in one variable** where we try to fit a function ***Y* = *W*x* + *b***.

If **X** is a **vector**, we call it **multi-variate linear regression** where we try to fit a function ***Y* = *w1*x1* + *w2*x2* + *w3*x3* ... + *b***.

**3. Logistic Regression**

**Logistic Regression** is a **supervised learning technique** which is used for **predicting classes**.

The simplest logistic regression classifier works for **binary classification** tasks. We proceed like we normally would for linear regression task but in the end **squeeze output** to be in the range **0** to **1**. We do this by using a **sigmoid function**. The output of sigmoid function determines the confidence of **YES** or **NO** for a class, cause this is binary classification.

Logistic Regression has a variant called **One-vs-All** that can also be used for multiclass classification also.

**Resources:**

1. A Tour of the Top 10 Algorithms for Machine Learning Newbies by Towards Data Science on Medium (link).
2. A summarized list of things required for Machine learning. Good when you have seen them already and are about to code (link).