# A History of Machine Learning

Modern day lay people, the media, even many residing in academia itself tend to view Artificial Intelligence (AI) and Machine Learning as one in the same. While an understandable misconception, it still proves problematic as society seeks to integrate and come to terms with the integration of technology in daily life and work. This paper offers a brief review of the goal and history of Machine Learning which in turn provides the context and foundation on which to build that understanding.

Simply stated, the ultimate goal of AI is the quest to build intelligent machines. Machine learning constitutes a subset of that endeavor, with distinct goals and objectives of its own. Essentially, machine learning focuses on creating systems that learn from their own environment. Machine learning appears in every day life in the form of computer algorithms that enable computers to not communicate with their environment, and respond to data entered by the user. Computers and systems that do not require explicit programming and possess the ability to change and improve their algorithms is the key, overriding aim in machine learning. Many argue such science exists already, while others claim the science still has far to grow. Regardless, that machine learning will eventually result in a system capable of altering its basic algorithm through experience and applying this to new environments garners no argument. Understanding the beginning and evolution of machine learning illustrates why achieving the ultimate end game has earned regard as a when, not an if.

IN THE BEGINNING

We saw the first drafts of the neural network in 1943 with Warren McCullock's and Walter Pitts' concept of a simplified brain cell, the McCullock-Pitts(MCP) Neuron. McCullock and Pitts described this cell as a binary system with multiple signals being input in the cell body, and ultimately those inputs correspond to whether or not an output signal is generated.

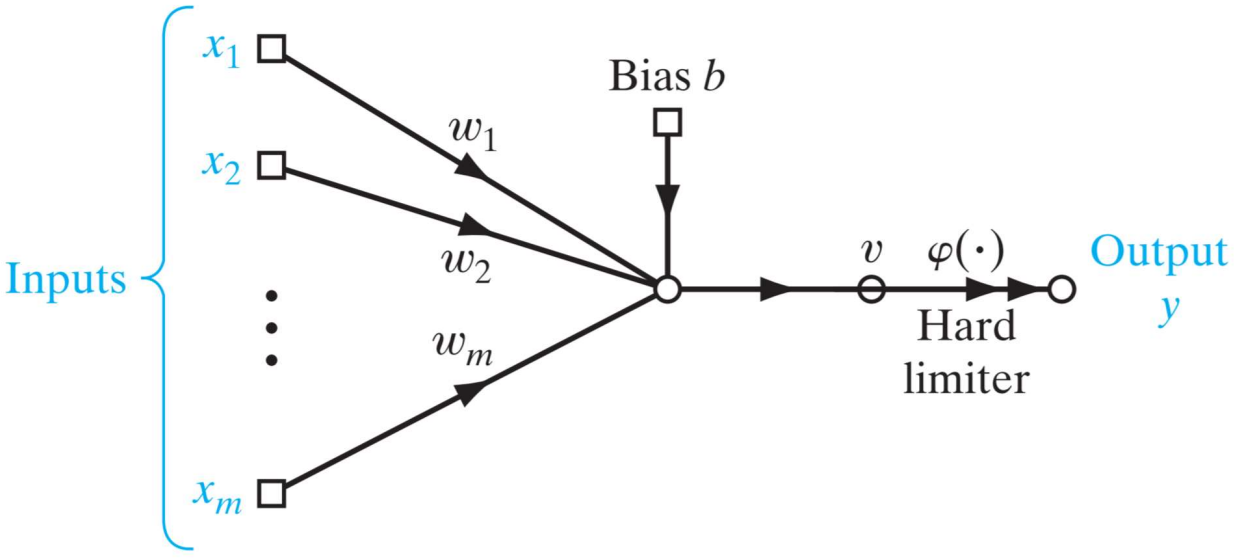
In 1946 the U.S. army constructed the first general purpose digital computer. Powered by vacuum tubes, the Electronic Numerical Integrator and computer, or ENIAC, covered 167 square meters, filling an entire 50-foot-long basement. For physicist John Mauchly and engineer J. Presper Eckert from the University of Pennsylvania, ENIAC represented their success in creating the first computer to utilize electronic processing speeds and essentially think for itself through the use of conditional branching and nested subroutines. Originally built to produce ballistic tables for the U.S. Army in World War II, ENIAC arrived post war and scientists instead used it for scientific computation until the beginning of the 1950's.

Mauchly and Eckert went on to form their own computer corporation, introducing the UNIVAC, the universal automatic computer, in 1951. The first commercially available computer in the U.S., Eckert-Mauchly produced and distributed just over 40 UNIVACs. In 1950, in response to the emergence of the giant brain ENIAC and rumors of the upcoming UNIVAC systems, Alan Turing created a test to determine whether a computer possesses true intelligence. In order to pass, the computer must convince a

human interfacing with the system that it is human as well; to date no system has successfully past the Turing test, though a few have falsely claimed as much.

Perhaps the most stunning success arose in 1955 with Arthur Samuels creation of the first computer learning program. Samuel worked on an IBM computer, writing an algorith for the game checkers. With each subsequent game and exposure to various moves and winning strategies, the computer incorporated these into the algorithm, demonstrating successive improvement. By the 1970s Samuel's system consistently beat expert human players.

Come 1957 and Frank Rosenblatt published his historic expansion of McCullock's and Pitt's theory, the perceptron. The perceptron was first algorithm considered to describe a neural network. It was built around the nonlinear MCP neuron described above. The MCP neuron consist of a linear function that is then limited by a threshold function.



The output y is then determined and is either +1 or -1 corresponding to the class label of the target variable. The aim of the perceptron algorithm, and that of neural networks, is to correctly classify data based on the input variables. Without going into detail, it worth noting that the perceptron algorithm is analogous to a special case of something known as the Bayes classifier in statistical theory. When an environment is gaussian (normally distributed) the Bayes classifier reduces to a linear classifier analogous to that of the perceptron. There are however subtle, but significant differences that won't be discussed here.

Almost a full decade later the well-known k-nearest neighbors algorithm was introduced by Thomas Cover and Peter Hart in their paper titled *Nearest Neighbor Pattern Classification.* The nearest neighbor algorithm assigned to an unlabeled observation the label of the nearest observation. The simple intuition behind it being that things that look alike must be alike. Of course, Hart and Cover realized that was not necessarily the case and aimed to classify the potential of misclassifying an observation. In statistics the Bayes risk measures the minimal probability of error given complete statistical information. As luck would have it, the two were able to prove that the nearest neighbor risk is bound above by a factor of 2 on the bayes risk. This simple, and useful bound help nearest neighbors become an oft used method. However, Hart and Cover excluded a class of probability distributions form their proof, and the general nearest-neighbors method risk bound was not proved until 1977 in Charles Stones' *Annals of Statistics*. In the next paper we will spend time classifying the use of kNN in supervised learning problems.

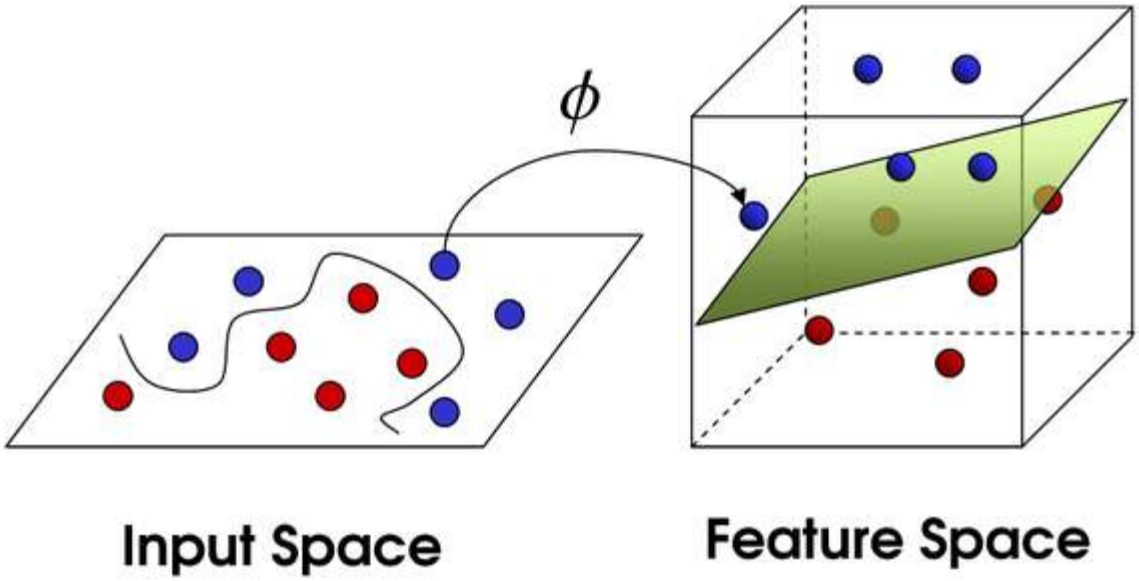
The AI winters are periods of time throughout the late 1960s to the late 1980s in wish a large portion of the AI community lost interest in AI and neural networks. The field experienced a lot of hype and ultimately when it was unable to deliver researcher grew frustrated, funding decreased, and commercial interest turned pessimistic.

During this time, in 1981, a statistical method called projection pursuit regression was introduce by Jerome H. Friedman and Werner Stuetzle. Project pursuit regression (PPR) is a statistical model that adapts additive models in that it first projects the data matrix of explanatory variables in the optimal direction before applying any smoothing methods to the features. The PPR method is incredibly general since forming nonlinear function of linear combinations generates a large class of models. In fact, for a large enough sample PPR can approximate any continuous function in R^p. However, such models are generally hard to interpret. The reason PPR is so interesting is that is in fact a more general analogue to a neural network in that both project the data in one direction and then apply a non linear transform of the data. The main difference is that PPR is more straightforward and general than a neural network with data driven transforms as opposed to fixed transforms in a neural network.

*Learning representations by back-propagating errors* was published in 1986 by David Rumelhart, Geoffrey Hinton and Ronald J Williams. The paper describes the procedure of backpropagation on neural networks. The procedure repeatedly parameterizes weights in the network in an effort tot minimize sum of squared errors (SSE) between the "desired" output layer and the actual one. Backpropagation utilized gradient descent by relating an associated cost (SSE) with each weight in each layer of

a multilayer neural network and minimizing the said cost starting with the top layer and minimizing each cost as we move 'backwards' through the network. This breakthrough allowed researchers to realistically explore multilayer neural networks. From a statistics perspective it is interesting to note that the neural network model is a general linear model, known as logistic regression, in the hidden units and all the unknown weight parameters are estimated by maximum likelihood estimators.

Come 1995 and Corinna Cortes and Vladimir Vapnik publish their work on support vector machines. A support vector classifier finds linear boundaries in the feature space and form class separations. The Support vector machine classifier is a generalization of this in which the feature space is enlarged and allowed to become very large, infinite in some cases. The resulting linear boundaries achieve a much better class separation and can form the non-linear boundaries in the original space.



Throughout the next decade, AI and neural networks saw another boost in popularity. With the discovery of Long short-term memory networks that expanded up recurrent of neural networks in 1997. Also in 1997, IBM's deep blue successfully defeated the world champion under standard regulations, the first of its kind to do so.

In 2015 Google's DeepMind, formerly just DeepMind, used stochastic decision methods know as partially observed Markov decision processes to approach reinforcement learning in Atari games in which their algorithm was able to beat Atari games with state of the art scores. This showed the incredible generality of machine learning, particularly reinforcement learning and how the incorporation of stochastic methods lends way to surprising and general results. The same principles of this algorithm led to Google's AlphaGo which beat the master go player in 2016. Particularly unique is that DeepMind used a Convolutional Network with pixeled game screens as input and a Q-value, the value we wish to maximize, as the output. From a probabilistic perspective the pixels on the screen served as the state-space for our decision process, and incorporating the partially observed state space allows agents to learn from trial and error. The subset of reinforcement learning, known as Q-learning is the field that finds the optimal policy to maximize the reward of an agent in the said environment.

Machine Learning has a large amount of buzz surrounding it, and with that buzz there is huge sense of mysticism surrounding it. However, in this paper we have explored a small portion of machine learning's evolution over that past 7 decades. While it is useful to simplify our explanations of concepts machine learning so that they remain approachable to general audiences. However, for practitioners of machine

learning it is of great importance to give respect to underlying theories of machine learning which are largely analogous, if not rooted, in statistical and probabilistic techniques. Properly acknowledging the connection between the analytic methods to minimize functions from R^p to R^q, the rigorous statistical theory lurking in learning theory, and the implementation of machine learning models will only improve the utility and effectiveness of our machine learning methods..

References

Cover, T., and P. Hart. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory*, vol. 13, no. 1, 1967, pp. 21–27., doi:10.1109/tit.1967.1053964.

Friedman, Jerome. "Projection-Pursuit Regression." *Multiple and Generalized Nonparametric Regression*, pp. 38–47., doi:10.4135/9781412985154.n4.

Hastie, Trevor, et al. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

Marr, Bernard. "A Short History of Machine Learnin." *Forbes*, Forbes Magazine, 8 Mar. 2016, www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/2/#3b34bb466b1b.

"Perceptron Algorithm, 1959; Rosenblatt." *SpringerReference*, doi:10.1007/springerreference_57806.

S, Siraj. "Deep Q Learning for Video Games - The Math of Intelligence #9." *YouTube*, YouTube, 11 Aug. 2017, www.youtube.com/watch?v=79pmNdyxEGo.

"Timeline of Machine Learning." *Wikipedia*, Wikimedia Foundation, 11 Apr. 2018, en.wikipedia.org/wiki/Timeline_of_machine_learning.

# Introduction to Supervised Learning

## Introduction

Machine learning is the art and science of giving computers the ability to learn and make decisions from data without being explicitly programmed. We describe the three branches of machine learning: Supervised learning involves analyzing labeled datasets and predicting the labels of future data points. Unsupervised learning is the art of uncovering patterns from unlabeled data. Reinforcement Learning: Agents interact with environment, learn how to optimize their behavior given a system of rewards and consequences. Draws inspiration from behavioral psychology and stochastic decision processes.

In Supervised learning we have several data points or samples called predictor variables. Our aim is to predict the target variable, given some predictor variables. The learning task is a classification task if the target variable is a discrete category and a regression if the target variable is continuous. We will often interchange between the terms feature, predictor variable, and independent variable. These terms describe the inputs into our machine learning model. The output of our model, or the thing we are trying to describe is referred as a target, response, and/or dependent variable.

For supervised learning we need labeled data for which the right output is known. The library *Scikit-learn* is a popular approach to supervised learning in python. Let us explore how supervised learning techniques are applied to real world data.

## Exploratory Data Analysis

We consider the iris data set from RL Fisher included in the *sklearn* package. We will use popular packages *pandas*, *numpy*, and *matplotlib*.

```
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
iris = datasets.load_iris()
print(type(iris))
print(iris.data.shape)
X = iris.data
y = iris.target
df = pd.DataFrame(X, columns=iris.feature_names)
print(df.head())

   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
```

```
4                  5.0                3.6                1.4
0.2
```

### k-Nearest Neighbors

We use the algorithm k-Nearest neighbors algorithm to predict the flower species. The idea of KNN is to predict the label a of a data point by using the surrounding k data points labels.

## Measuring Model Performance

In classification problems accuracy is a desired metric. We view accuracy as the proportion of correct predictions. A good question is question which data should be used to compute accuracy. We should not infer the accuracy from the training data as this will not generalize to other data. Rather, we should split data into a test and validation set, where we fit and train our model on the training data and make predictions on the test or validation set. Finally, we compare predictions with known labels to measure accuracy.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
                                                    random_state = 21,
                                                    stratify=y)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("test set predictions: \n {}".format(y_pred))
print(knn.score(X_test, y_test))

test set predictions:
 [2 1 2 2 1 0 1 0 0 1 0 2 0 2 2 0 0 0 1 0 2 2 2 0 1 1 1 0 0 1 2 2 0 0 1 2 2
 1 1 2 1 1 0 2 1]
0.9555555555555556
```

## Model Complexity:

It should be noted how the parameterization of k affects the overall model. In general models with a high k value tend to smoother, but can oversimplify the problem, while models with a small k are complex and can overfit the random effects on data. So, we aim to find a medium where we avoid overfitting, or underfitting, our data.