Eberhard Karls Universität
Tübingen

Introduction to Machine Learning

Supervised Learning

Julia Erdei

supervised by
Dr. Akash Ashirbad Panda

27th April 2022

Contents

1 Machine Learning

1.1 Motivation for Machine Learning

Before jumping into the technicalities, the question needs to be answered as to why Machine Learning is as important and powerful as we see. Traditional programs have been around for a while, so why do data scientists even make the effort to study and develop self learning programs? To properly answer that question we should have a look at easily solvable problems. The Fibbonaci sequence or the factorial function can easily be implemented even by beginners of programming through the traditional way as "any manually created program that uses input data and runs on a computer to produce the output"[10]. In other words we simply tell the program quite directly what to do and explicitly state every step of its actions. But what happens, if the problem is not as straight forward?
The more complex a problem gets, the more traditional programs fail at exceeding their goals. There will be problems if

- The data is too large or complex to make clear conditions or that there are too many and it becomes inefficient to produce them manually.
- There are changes in the description of the problem or the data. This would lead to the need of changing the entire program which is very time consuming.
- The topic of the problem is specific. The programmer would have to be an expert in the field and know all of the details, which again takes a lot of time depending on the subject.

These are all issues where Machine Learning can contribute in reaching an outcome more effectively or efficiently. The high demand of research around it and the amount of applications in this field show the power and necessity of this kind of programming.

1.2 Definition of Machine Learning

Even though Machine Learning is a highly researched field, the definition of it is rather broad and the distinction to other similar fields is not necessarily very clear. Therefore, some well-known definitions are shown in the following:

- The area of Machine Learning deals with the design of programs that can learn rules from data, adapt to changes, and improve performance with experience [4]

- Machine learning (ML) is a branch of artificial intelligence (AI). Algorithms can recognize patterns and laws in data sets and develop solutions from them. Simply put, knowledge is generated from experience. [14]
- A program or system that builds (trains) a predictive model from input data. The system uses the learned model to make useful predictions from new (never-before-seen) data drawn from the same distribution as the one used to train the model.[7]

I would describe Machine Learning as programs and algorithms that solve problems by analyzing data and are able to recognize patterns and laws in the data, thus being able to adapt to changes by themselves.

1.3 History of Machine Learning

1949
"The Organization of Behavior" by Donald Hebb introduces the term "Machine Learning"
1950-1959
"Turing Test" by Alan Turing
Arthur Samuel develops program for playing checkers (first computer learning programm)
Frank Rosenblatt creates perceptron (first neural network)
1960-1969
Nearest neighbor algorithm - beginning of pattern recognition
Discovery of multilayers
1970-1979
Development of Backpropagation
Students develop "Stanford Cart"
1980-1989
Terry Sejnowski invents NetTalk
1990-1999
IBM's Deep Blue beats the world champion at chess
Term "Boosting" is coined - e.g., "AdaBoost"
Long short-term memory (LSTM) by Jürgen Schmidhuber and Sepp Hochreiter
2000-2009
Face Recognition Grand Challenge in 2006
Geoffrey Hinton coins the term “deep learning”
2010-2019
Google Brain is developed

Facebook develops DeepFace
Google's artificial intelligence algorithm beats a professional player at the Chinese board game Go
Present
Instruction fine-tuning observed by Google[6][13]

1.4 Common Applications of Machine Learning

Although there is an endless amount of possibilities to use Machine Learning, the following applications are the most commonly used.

- Face recognition
- Video-surveillance
- Sentiment analysis
- Email-classification and spam detection
- Virtual assistant
- Fraud detection
- Social media
- Medical diagnosis

1.5 Overview of Machine Learning types

Machine Learning algorithms can be divided into three major algorithm types. These are Supervised, Unsupervised and Reinforcement Learning. While the focus in this presentation lies on Supervised Learning, it is still necessary to briefly outline their characteristics.

- Supervised Learning is task-driven and one of the most basic parts of Machine Learning.
- Unsupervised Learning is driven by data and able to discover hidden structures in datasets.
- Reinforcement focuses on leaning from errors and is based on the psychological concept of conditioning.

1.6 Comparison to Unsupervised and Reinforcement Learning

The defining differences between the three big branches in Machine Learning are the usage of the data observed by the programs. Basically, Supervised Learning uses data, which has labeled input vectors and labeled output vectors. I will specify this in the following section. Unsupervised Learning only uses labeled input vectors and Reinforcement Learning does not require any labels. Apart from that there's also a mixture of Supervised and Unsupervised Learning called Semi-supervised Learning which party uses labeled and unlabeled output vectors.

2 Supervised Learning

2.1 Definition of Supervised Learning

Being a subcategory of Machine Learning, Supervised Learning describes algorithms and programs which use self-learning properties. As already mentioned, the training data consists of labeled input vectors $i_m := \left(x_{m,1}, ...x_{m,n}\right)^t$ and labeled output vectors $o_m := \left(x_{m,1}, ..., x_{m,n}\right)^t$. The algorithm will analyze the properties of the input data and categorize new data points in context of the already examined training data.

2.2 Example of Supervised Learning

Since today is the 4th of May I couldn't help but use an example inspired by the franchise Star Wars. Imagine this: Jedi master Plo Koon wants to add new clone troopers to his already employed platoons. He wants to have effective troops so he'd like to sort the new troopers into the two platoons by their characteristics. Since it's a bit too difficult for himself he asks his droid R7-F5 for help. In the following segments we will discover how R7-F5 uses Machine Learning to solve this problem.

Figure 1: A visual for our example

2.3 Classification and Regression

There are two main groups of problems handled by Supervised Learning
Classification
This subgroup of problems tries to predict the affiliation of data points to already existing classes of data. The algorithm would try to divide the training data into groups and assign a new data point to any of the groups by its characteristics. The example I described before would be a classification problem.
Regression
Regression on the other hand tries to find a function to approximate the data points and their properties. This could for example be important to solve numerical problems. [3]Concerning our example we could create a regression problem if we wanted to find out how the properties of the clones in the platoons change over time or how fast they change.

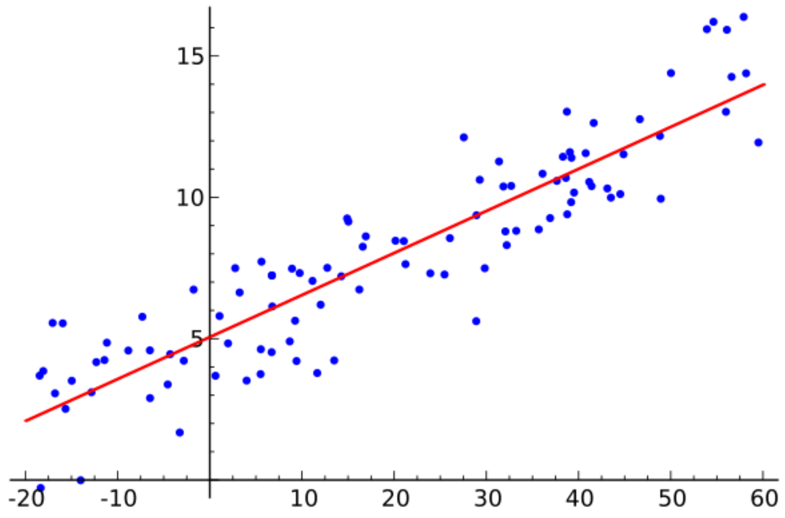

Figure 2: Development of shooting skills of clone troopers in drills of 20 rounds over the course of 80 days

2.4 Generative and Discriminative Models

One is able to differentiate between two types of models commonly used. Generally we try to estimate the propability of $P[Y|X]$ while we can label Y as the instance of a new data point and $X = (X_1, ..., X_n)^t$ as the prior data points.
Generative Models
These models use Bayes as method to make predictions and calculate the proximity $P[Y|X]$ as stated before.
The Bayes theorem states:

$$P[X|Y] = \frac{P[Y] \cdot P[Y|X]}{P[X]}$$

This roughly translates to

$$\text{posterior} = \frac{\text{prior} \times \text{likelyhood}}{\text{evidence}}$$

Common examples are

- Naive Bayes (and generally Bayesian Networks)
- Hidden Markov Models

Discriminative Models
They center around finding the boundary between one class and another. This boundary sometimes isn't very simple to choose, which is why the model tries to create a "decision boundary" to be able to decide the class as effectively as possible. Common examples are

- Logistic Regression
- Support Vector Machine
- Decision Trees

Discriminative Models have the benefit of being more stable considering outliers, data points which seem to go against the rules. On the other hand, generative models are less likely to classify data points incorrectly. Basically, discriminative models focus more on creating and predicting labels of the data and generative models rather try to explain how the data was generated. [17][2]

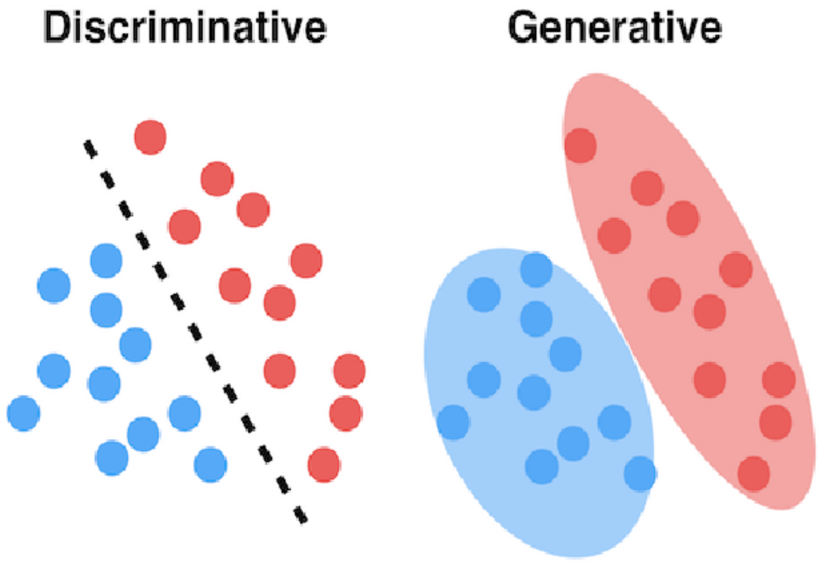

Figure 3: Discriminative vs. Generative Model

2.5 Challenges with Models

Applying models and to the data underly certain challenges and common pitfalls. The next section will be discussing different so called trade-offs that try to minimize any issues that come with dealing with data.

2.5.1 Under- and Overfitting

Overfitting
This happens if a model is trained too well and ends up too complex for the data it uses.
Underfitting
In this case the model doesn't learn enough from the data and ends up being too generalizing.
In both cases the algorithm ends up being unreliable and unable to make proper predictions. In conclusion it should be balanced between these two extremes.

2.5.2 Bias-Variance Trade-off

Another tricky issue with programming is creating an effective error prediction. To explore this we should remember two important terms from probability theory:

1. Bias describing the difference between the prediction of the model and the actual occurring value

2. Variance describing the variability of model prediction for a value telling us the spread of our data

Models with high bias end up paying too little attention to the training data, thus being a case of underfitting. On the other hand the ones with high variance are too fitted to the training data, resulting in overfitting. The program works very well with the training data but fails to make predictions for new data. To prevent a high bias or variance we need to consider a so called trade-off, balancing both to the minimum.[18]

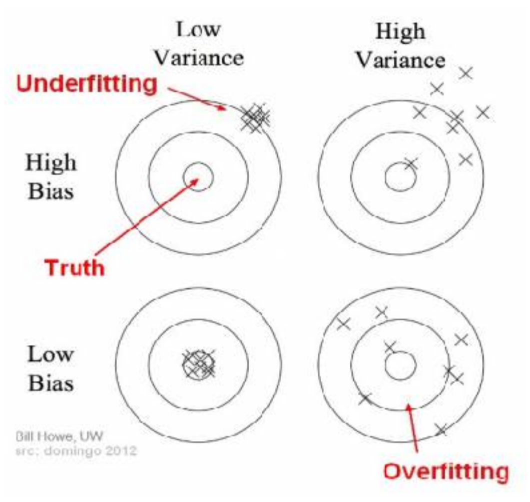

Figure 4: Comparison of high and low bias and variance

2.6 Precision-Recall Trade-off

To examine the accuracy of programs, we need to look at their precision and their recall to estimate how precise their predictions are. Precision tries to answer the question what proportion of positive identifications actually were correct. We can define this the following way

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$$

Recall on the other hand attends to the question what proportion of actual positives was identified correctly. This can be described as

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$$

The higher both of them are, the more accurate the program is going to be.[8]

3 Most common Algorithms in Supervised Learning

3.1 K-Nearest Neighbor(KNN)

As the most simple model, KNN can solve both classification and regression problems. It analyses the position of the vectors of the training data and tries to predict the classification of a new data point by comparing its position to the earlier examined dataset.
There are three steps to complete, when implementing a program working with this principle:

- We need to calculate the euclidean distance between the new data points and the ones in the dataset.
- Find the $k \in \mathbb{N}$ nearest neighbors of the new data points.
- Make the predictions based on these k neighbors.

In finished code, this could look something like this:

Listing 1: 1st step

```
for x in xrange(a):
    # calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
        distance = 0.0
        for i in range(len(row1)-1):
                distance += (row1[i] - row2[i])**2
        return sqrt(distance)
```

Listing 2: 2nd step

```
# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
        distances = list()
        for train_row in train:
                dist = euclidean_distance(test_row, train_row)
                distances.append((train_row, dist))
        distances.sort(key=lambda tup: tup[1])
        neighbors = list()
        for i in range(num_neighbors):
                neighbors.append(distances[i][0])
        return neighbors
```

Listing 3: 3rd step

```
# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
        neighbors = get_neighbors(train, test_row, num_neighbors)
        output_values = [row[-1] for row in neighbors]
        prediction = max(set(output_values), key=output_values.count)
        return prediction
```

Naturally, by decreasing the number of neighbors k the predictions will get less accurate and by increasing k the model will get more precise. The algorithm itself is very simple and can be used for a variety of problems, but has the disadvantage of becoming slower the bigger the dataset is.[9][5]
If we look at the example of our clone troopers, we notice we could easily solve this task by applying KNN. The following graph shows the idea behind that. After participating in a shooting drill, the platoons are cathegorized

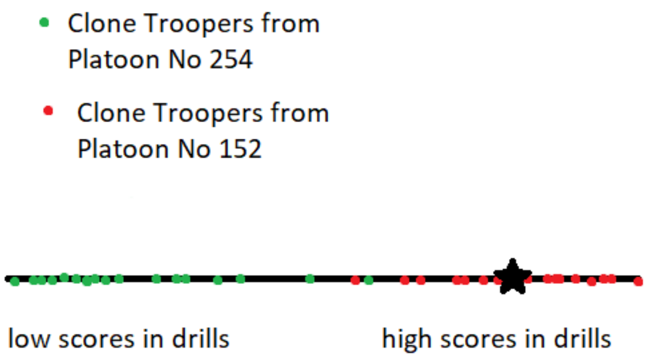
Clone Troopers from Platoon No 254
Clone Troopers from Platoon No 152
low scores in drills
high scores in drills

Figure 5: Solution of our earlier example

by their performance. Obviously, the new trooper marked by the star would be put in platoon No 152 based on this algorithm.

3.2 Naive Bayes Classifier

As the name says this model is based on the Bayes Theorem explained earlier and focuses on classification. It assumes the independence of classifying features. Because of this we can simplify the following equation

$$P[Y|X] = P[x_y|Y] \cdot P[x_2|Y, x_1] \cdot ... \cdot P[x_n|Y, x_1, ..., x_{n-1}] \cdot P[Y]$$

to

$$P[Y|X] = \frac{P[X|Y] \cdot P[Y]}{P[X]}$$

There are three types of NBCs:

- The Gaussian classifier, working with normal distribution.
- The multinomial classifier, which is used for discrete counts.
- The Bernoulli classifier, used for datasets with binary vectors.

As beneficial as the assumption of independence is, it is often quite unrealistic to find such a set.[2]

3.3 Hidden Markov Model

This type of models uses Markov chains with unobserved (hidden) states. Markov chains are a concept from probability theory, describing graphs with probabilities indicating how likely a state is in context of the past states.
The goal of the model is to learn about a Markov chain X by analyzing their hidden states Y. Additionally, the probability distribution of Y must not depend on the history of X according to that time. The HMM is a generative model similar to the Naive Bayes Model mostly used for classification. [19]

3.4 Decision Trees

This model can be used for both regression and classification problems. It visualizes how the decision making of the program works. The dataset is divided in smaller and smaller groups, until a tree-like structure forms. The leaves represent the decisions the program makes in the end and the nodes the features of the dataset that are used to classify.
A model made of several uncorrelated decision trees is called a random forest. This model is also commonly used for both types of problems solved by Supervised Learning.
Some disadvantages of decision trees are them being prone to a high variance and overfitting and the possibility of creating biased trees if some classes are more dominant in the dataset. [16]

3.5 Neural Networks

Roughly explained, neural networks are developed in a way that tries to simulate the way our brain works. They can analyze sensory data like sound, text of images trough clustering or some kind of machine perception. [15]

3.6 Kernel Methods

These are models using the kernel trick, which is solving a non-linear problem with a linear classifier. A common example are Support Vector Machines. They classify groups by placing a hyperplane between the data points. [1]

3.7 Linear and Logistic Regression

Linear regression assumes that there's a linear connection between independent and dependent predictors, thus handling regression problems while logistic regression works with binary classification.

4 Additional Thoughts on the Topic

4.1 Pros and Cons of Supervised Learning

Pros:

- Supervised learning are based on easily understandable algorithms.
- After the entire training is completed, you don't necessarily need to store the training data in your computer's memory. Instead, you can keep the decision boundary as a mathematical formula.
- Its strength are classification programs, which can be solved efficiently.

Cons:

- There are very complex tasks that cannot be solved by Supervised Learning
- Supervised learning naturally cannot give the programmer new and unknown information of the training data.
- There needs to be a big selection of good examples from each class, for the program to work precisely.

- The training of the program requires a lot of computation time.

[12]

4.2 Common Problems and Possible Solutions in Machine Learning

- Getting the right quality and quantity of data is very difficult but essential for effective programs.
- Non-representative training data will surely lead to too little generalization. The training data has to cover all classes, otherwise there will be viable information missing.
- There is still a lack of skilled experts. The solution to this is very simple: getting more people into studying and working with self-learning programs.
- A big problem are the ethics around programming. Common questions are, what kind of data is being used and who takes responsibility in case of system failure. Since all of the data is being produced or maintained by people, it is difficult or even impossible to find unbiased data. This can be an issue if for example the data handles personal information of people since it can support racial, gender or religious prejudiced. A possible solution would be to bring this issue more into the light of the public eye. The more people know about this danger, the more they can willingly work against that.

[11]

References

[1] Afonja, Tejumade. *Kernel functions.* 2018. URL: https://towardsdatascience.com/kernel-function-6f1d2be6091 (visited on 26/04/2022).

[2] Analytics Vidhya. *Learn naive Bayes algorithm: Naive Bayes classifier examples.* 2021. URL: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ (visited on 26/04/2022).

[3] Aunkofer, Benjamin. *Maschinelles Lernen: Klassifikation vs regression.* 2018. URL: https://data-science-blog.com/blog/2017/12/20/maschinelles-lernen-klassifikation-vs-regression/ (visited on 11/04/2022).

[4] Avrim Blum. "Machine learning theory". In: *Carnegie Melon Universit, School of Computer Science* 26 (2007).

[5] Brownlee, Jason. *Develop K-nearest neighbors in python from scratch.* 2019. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (visited on 26/04/2022).

[6] DFoote, Keith D. *A brief history of machine learning.* 2020. URL: https://digitaleweltmagazin.de/die-top-10-anwendungsfaelle-fuer-maschinelles-lernen-in-unternehmen/ (visited on 10/04/2022).

[7] Dumont, Andreas and Möller, Dirk and Pohlschroeder, Peter and Antova, Galina. *Die top 10 anwendungsfälle für Maschinelles Lernen in Unternehmen.* 2020. URL: https://digitaleweltmagazin.de/die-top-10-anwendungsfaelle-fuer-maschinelles-lernen-in-unternehmen/ (visited on 10/04/2022).

[8] Google. *Classification: Precision and recall.* 2020. URL: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall (visited on 25/04/2022).

[9] Harrison, Onel. *Machine learning basics with the K-nearest neighbors algorithm.* 2019. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (visited on 26/04/2022).

[10] insightsoftware. *Machine learning vs. traditional programming.* 2020. URL: https://insightsoftware.com/blog/machine-learning-vs-traditional-programming/ (visited on 10/04/2022).

[11] javatpoint. *Issues in machine learning - javatpoint.* 2018. URL: https://www.javatpoint.com/issues-in-machine-learning (visited on 12/04/2022).

[12] Joy, Ashwin. *Pros and cons of supervised machine learning.* 2019. URL: https://pythonistaplanet.com/pros-and-cons-of-supervised-machine-learning/ (visited on 12/04/2022).

[13] Marr, Bernard. *A short history of machine learning – every manager should read.* 2021. URL: https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/?sh=de7efb615e78 (visited on 10/04/2022).

[14] Microsoft. *Microsoft Erklärt: Was ist machine learning?* 2020. URL: https://news.microsoft.com/de-de/microsoft-erklaert-was-ist-machine-learning-definition-funktionen-von-ml/#:~:text=Machine\%20Learning\%20(ML)\%20ist\%20ein,wird%20Wissen%20aus%20Erfahrungen%20generiert (visited on 10/04/2022).

[15] Pathmind. *A beginner's guide to Convolutional Neural Networks.* 2020. URL: https://wiki.pathmind.com/neural-network (visited on 26/04/2022).

[16] Prashant Gupta. *Decision trees in machine learning.* 2021. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (visited on 26/04/2022).

[17] Rufai, Aminah Mardiyyah. *Generative vs. Discriminative models in machine learning.* 2020. URL: https://betterprogramming.pub/generative-vs-discriminative-models-d26def8fd64a?gi=515961fe3348 (visited on 20/04/2022).

[18] Singh, Seema. *Understanding the bias-variance tradeoff.* 2018. URL: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229 (visited on 25/04/2022).

[19] Verma, Yugesh. *A guide to hidden markov model and its applications in NLP.* 2021. URL: hhttps://analyticsindiamag.com/a-guide-to-hidden-markov-model-and-its-applications-in-nlp/ (visited on 26/04/2022).