The Types of Machine Learning Algorithms Explained

§

Made with Obsidian

Type blog | Category machine-learning | Technologies Python | Website Post Link

§

Machine Learning is a discipline aimed at designing, understanding and applying computer programs that learn from experience for the purpose of modelling, prediction or control. It has gained significant momentum in recent years because it has forever changed how we look at problem-solving; previous to Machine Learning, if we wanted to tackle a problem, we had to hard-code a solution that would consider every variable and parameter. This becomes increasingly challenging as the amount of generated data also increases. ML algorithms are able to analyze all this data and (*sometimes*) come up with more accurate answers faster than any hard-coded method.

With this new discipline come a variety of models that can be used for different applications; from product recommendations to image, speech & text recognition to sentiment analysis to DNA sequencing to global warming modelling and temperature prediction; the potential applications are vast, and almost every field of study has at least one problem where ML models can help come up with more efficient solutions. ML is here to stay, and it's helpful to know the generalities of the technology to understand its potential applications.

In this article, we will discuss the different Machine Learning algorithm types. We will also qualitatively compare different Machine Learning models to grasp a generalized understanding. We will conclude this segment with some next steps for those interested in diving deeper into this exciting discipline.

We'll be using Python scripts which can be found in the Blog Article Repo.

§

Table of Contents

Machine Learning vs traditional programming

What's so special about it? Why was it such an important breakthrough in how we think of problem-solving? To explain this, we have to refer to how traditional computation is done.

Traditional computer programming has been around for more than a century, with the first known computer program dating back to the mid 1800s. It refers to any manually created program that uses input data and executes to produce the output; a programmer creates the program, but without anyone programming the logic, one has to manually formulate rules (*this is what we'll be referring to as hard-coding*).

For example, we can define a function that will do a specific task based on a set of logical rules which we ourselves define. In this case, our function accepts a DataFrame object as input, evaluates each row of the `age` column, and performs a mathematical operation based on each `age` value:

Code

```python
# Import modules
import pandas as pd

# Example 1: Hard-coded function

# Declare entries
p0 = ['Saha', 'Howard', 22, 'Secretary', 1300]
p1 = ['John', 'Moore', 32, 'Illustrator', 3000]
p2 = ['Laszlo', 'Kreizler', 35, 'Alienist', 3500]
p3 = ['Willem', 'Van Bergen', 31, 'Pampered Child', 50000]
p4 = ['Libby', 'Hatch', 30, 'Nurse', 1100]
p5 = ['Lucifer', 'Morningstar', 35, 'Club Owner', 100000]
p6 = ['Chloe', 'Decker', 34, 'Detective', 1400]
p7 = ['Dan', 'Espinoza', 36, 'Detective', 1300]
p8 = ['Mazikeen', 'Smith', 35, 'Bounty Hunter', 20000]
p9 = ['Laura', 'Palmer', 22, 'Student', 100]
p10 = ['Dale', 'Cooper', 36, 'FBI Agent', 1800]
p11 = ['Harry', 'S. Truman', 34, 'Sheriff', 1300]
p12 = ['Bobby', 'Briggs', 23, 'Student', 80]
p13 = ['Gordon', 'Cole', 55, 'FBI Agent', 2000]

# Create list of lists
people_list = [p0, p1, p2, p3, p4,
               p5, p6, p7, p8,
               p9, p10, p11, p12,
               p13
               ]

# Declare DataFrame
df = pd.DataFrame(people_list,
                  columns = ['Name', 'Surname', 'Age', 'Job Position', 'Salary'])

# Declare function
def myFun(df):
    '''
    Parameters
    ----------
    df : Pandas DataFrame
        Contains 3 columns: Name, Surname, Age, Job Position, Salary (monthly in USD).

    Returns
    -------
    top_jobs : List
        Contains the top 7 paid job positions for a person between
        25 and 40 years old, sorted by importance.
    '''

    df_top = (df.query("`Age` >= 25 and `Age` <= 40").
              groupby('Job Position')['Salary'].mean().
              reset_index(name ='Average Salary').
              nlargest(7, 'Average Salary').
              sort_values(by='Average Salary', ascending=False).
              reset_index(drop=True)
              )

    return df_top
```

```
# Call function with df
print(myFun(df))
```

Output

```
     Job Position  Average Salary
0      Club Owner        100000.0
1  Pampered Child         50000.0
2   Bounty Hunter         20000.0
3        Alienist          3500.0
4     Illustrator          3000.0
5       FBI Agent          1800.0
6       Detective          1350.0
```

Apart from the fact that being a club owner or a pampered child could be the key in life, we can see that this method implementation worked because we defined logical rules denoted by our `query` statements. These were explicitly stated and required a logical reasoning behind curtains.

This was also feasible because we had a very limited set of rules: `age` and `Average Salary`, which we used as features in order to return an output.

The problems surface when we have an extensive set of features we need to consider, are not normalized, or even worst, we don't know which ones could be used leveraged to get the insight we're looking for.

In Machine Learning, the algorithm automatically formulates the rules from its input, and outputs a model which we can use to perform predictions, classifications, or control.

These algorithms can go from extremely simple to extremely complex; it all depends on what we're trying to solve, the input dimensions, and the complexity of our data.

It's important to remember that, as grandiose as it may seem, ML is just another tool in a very extensive toolbox; it's easy to assume that ML can be applied to every possible problem and outperform any traditional algorithm, but the truth is, it does not substitute traditional programming entirely, nor does it do magic.

As with any tool, it has its applications and limitations, and is based on rigorous mathematical and statistical theory. In fact, Machine Learning can be though of as a combination of probability theory, statistics & optimization. We'll see why in a moment.

§

Machine Learning algorithm types

Machine Learning algorithms can be classified by how they work and what type of data they accept as input. There are 3 main types and some other subclassifications:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised Learning

Supervised Learning uses labeled datasets to train algorithms that to classify data or predict outcomes accurately. We can think of a labeled dataset as one with tags that identify what the data represents.

If we look at our previous example, we used a DataFrame with 5 columns:

Code

```
print(df.columns)
```

Output

```
Index(['Name', 'Surname', 'Age', 'Job Position', 'Salary'], dtype='object')
```

This means that our data is labeled; it has identifiable tags which we can use to make sense of the information it presents.

More specifically, in the Machine Learning context, a label is the specific vector we're trying to predict or use to classify. This depends on the problem we're trying to solve.

For example, we could use our DataFrame `df` to train a model which predicts a person's Salary based on other attributes such as `Age` and `Job Position`. These attributes would be called features, and the target to predict, in this case `Salary`, would be our label. We could solve this problem either by regression or classification; these are the two main subclasses of Supervised Learning.

Regression

Regression is most commonly known since it's taught very early on math courses and has multiple applications; from market analysis to financial analysis to trend forecasting. It requires us to have continuous data points.

The most common and simple type of regression is **Linear Regression** (*LR*), which consists of fitting a straight line to a set of value pairs (x, y) in order to predict unseen values of a dependent variable y given an independent variable value x. This method uses the general equation of a straight line to fit data:

$$y = m \cdot x + b$$

Where:

- y is the dependent variable.
- m is the slope or gradient (*how steep the line is*).
- x is the independent variable.
- b is the y-intercept (*the value of y when $x = 0$*)

Which can be written in terms of Machine Learning notation:

$$h(x) = \theta_0 + \theta_1 x$$

Where:

- $h(x)$ is the label.
- θ_0, θ_1 are weights.
- x is the input feature.

Once we have the basic straight line form, we can start by using random weights and evaluating how well this line fits our data. This is done by using an error function, the most common in Linear Regression being a variation of the Mean Squared Error (*MSE*) function called the **Squared Error Cost Function**.

We can first express the simpler MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where:

- n are the number of data points.
- Y_i is the observed i^{th} value.
- $\hat{Y}_i$ is the predicted i^{th} value.

And then translate to the Squared Error Cost Function:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - Y^{(i)})^2$$

Where:

- m are the number of data points.
- $h_\theta(x^{(i)})$ is the predicted i^{th} value.
- $Y^{(i)}$ is the observed i^{th} value.

What Linear Regression does, is minimize this function by changing the weights iteratively. This is done by calculating the partial derivatives of $J(\theta_0, \theta_1)$ with respect to θ_0 and θ_1:

$$\frac{d}{d\theta_0}J(\theta_0, \theta_1) = \theta_0 + \theta_1 x - Y^{(i)}$$

$$\frac{d}{d\theta_1}J(\theta_0, \theta_1) = (\theta_0 + \theta_1 x - Y^{(i)})x$$

Where θ_0 and θ_1 are updated simultaneously on each iteration.

This method we just defined is called **Gradient Descent**, a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. It's widely used in Machine Learning to optimize multiple models.

In the end, we're left with a combination of parameters (θ_0, θ_1) which best fit our straight line to the data points.

Of course, Linear Regression, as its name suggests, is used to predict data which presents linear correlation. There are other methods for predicting non-linear continuous data such as Polynomial Regression.

Classification

As its name suggests, this subcategory of Supervised Learning aims to classify

Semi-Supervised Learning

Unsupervised Learning

Active Learning

Transfer Learning

Reinforcement Learning

§

Conclusions

We have reviewed ...

§

References

- IBM, What is supervised learning?
- IBM, What is unsupervised learning?
- Machine Learning Mastery, What Is Semi-Supervised Learning?
- Towards Data Science, Active Learning in Machine Learning
- Analytics Vidhya, Commonly used Machine Learning Algorithms

§

Copyright