Boise State University

CS 496 - Independent Study

Machine Learning

# Machine Learning

*Author:*
Cory Owens

*Supervisor:*
Shane Panter

April 26, 2016

# Contents

# Introduction

According to Dr. Parag Kulkarni, "Machine learning is the study of methods for programming computers to learn." [1] With traditional algorithms, the program is given a repeatable process to apply for any and all problems to which it is applied. The solution has to be explicitly outlined for the program to work. With machine learning, the program is, instead, given a method to learn from data, and use that to derive its own method for solving associated problems.

Machine learning is therefore most useful in solving problems which may not be easily generalized, or which may have some degree of variation. Kulkarni highlights four such classes of problems:

1. Repetitive tasks with small variation but which require very high levels of precision. For example, automated manufacturing.

2. Problems which, for humans, the knowledge is tacit. For example, speech recognition and language understanding.

3. Rapidly changing problems. For example, recognition and filtering of spam email.

4. Applications which must be customized for each individual user. For example, a personal assistant program.

Machine learning algorithms can be categorized in two primary ways. The first is to categorize them by the types of data they deal with. There are four common categories:

1. Supervised Learning, where the machine is trained on labeled example data.

2. Unsupervised Learning, where the machine is given unlabeled data and asked to find underlying order and patterns.

3. Semisupervised Learning, where the machine is given a mix of labels and unlabeled data.

4. Reinforcement Learning, where the machine is tasked with producing a solution, that solution is then rated or graded and then incorporated back into the machine as data.

In section 1, this paper will discuss these data categories in more detail.

The second way to categorized machine learning algorithms is by the output or goal of the algorithm. Here, there are three common categories:

1. Classification or Decision machines, where the output is discrete.
2. Regression, where the output is continuous.
3. Clustering, where the output is some set of groupings of data.

In section 2, this paper will discuss these output categories in more detail.

There is a large number of machine learning algorithms. In section 3, this paper will discuss five of the most popular machine learning algorithms

1. Linear/Polynomial Regression
2. Logistic Regression
3. Decision Trees
4. Naive Bayes Classifiers
5. Neural Networks

# 1 Categorization by Data

One important way of categorizing machine learning algorithms is by the type of data they are given. This data can be prelabeled example data, often referred to as the "training set." In the case where a machine learning algorithm is given a labeled training set, the algorithm is classified as "supervised learning." The data can also be unlabeled, and the algorithm must then find hidden structure within the data for it to be useful. In the case where a machine learning algorithm is given unlabeled data, the algorithm is classified as "unsupervised learning." The data may be some mix of prelabeled and unlabeled data. In the case where a machine learning algorithm is given such a mixture of labeled and unlabeled data, the algorithm is classified as "semisupervised learning." Finally, in the case where the machine produces output, and that output is then rated before being reincorporated into the algorithm's data set, the algorithm is classified as "reinforcement learning". Each of these methods has its own strengths and weaknesses.

## 1.1 Supervised Learning

In supervised learning, where the machine learning algorithm is provided a labeled training set of data, the algorithm has the advantage of being able to build a starting knowledge base from existing data with the confidence that this data is accurate. This, of course, alludes to one challenge in using supervised learning: the selection of a training set. Not only must the data be accurate, but it is important to minimize variance and outliers to ensure the algorithm does not form a false hypothesis from the data. In some applications, even having a large enough set of appropriate training data may be difficult in itself. Supervised learning is commonly used in classification and regression, which will be discussed in the next section. An example of supervised learning would be to supply the machine learning algorithm with a set of housing data, including size, age, location, and price. The algorithm could then be given input of size, age, and location which could then be mapped to a predictive price. [2]

## 1.2 Unsupervised Learning

In unsupervised learning, where the machine learning algorithm is provided unlabeled data, the algorithm has to discover some underlying structure within the data. Unsupervised learning has the benefit that the user doesn't have to provide a curated training set. This is particularly useful for datasets too large for a human to realistically analyze, allowing unsupervised learning to discover patterns of which the user was entirely unaware. One drawback is that the user loses the ability to direct the learning process, and that no meaningful structure may even be discovered. Unsupervised learning is commonly used in clustering, which will be discussed in the next section. An example of unsupervised learning would be to supply the machine learning algorithm with an unlabeled set of botany data for flowers, including stem length, petal shape, and petal count. The machine could then find order within this data, identifying some distinct classes, even if the machine does not produce labels for these classes (it wouldn't know that it had just identified a family of orchids, just that these types of flowers are somehow related, based on the dataset).

## 1.3 Semisupervised Learning

In semisupervised learning, where the machine learning algorithm is provided a mix of labeled and unlabled data, the algorithm attempts to find underlying structure within unlabeled data, but with the guidance of a labeled training set. Of course, this implies the main advantage of semisupervised learning is in that it allows the user to guide the algorithm's exploration.

## 1.4 Reinforcement Learning

Reinforcement learning can be seen as either supervised learning (where the output rating and its reincorporation controlled by a human being) or unsupervised (where the machine explores and reincorporates results itself), but it is important to identify its differences from the more traditional supervised and unsupervised learning above. In reinforcement learning, the algorithm, typically, is not given a large set of data to work with. The machine is called upon to produce output. That output is rated, and then fed back to the machine. This mapping of an

output to a rating is added to a running dataset for the machine. Reinforcement learning is a balance between exploration of new output spaces and exploitation of previously successful outputs. An example of reinforcement learning would be a machine learning to play chess. Each game, the machine may build a strategy. That strategy is then played out against an opponent. If the opponent is a human, this could be seen as supervised learning. If the opponent is another machine, or even another instance of the same machine, this could be seen as unsupervised learning. Each move may be rated, or perhaps only the outcome of the game. This strategy and the resultant rating is then incorporated back into the machine before it starts its next game.

# 2 Categorization by Goals

Machine learning algorithms can also be classified by the output or end goal of the algorithm. When the input is mapped to a discrete set of classes, the algorithm is said to perform "classification." In such cases as the class set is binary, the algorithm is said to perform "decision." When the input is mapped to a continuous output, the algorithm is said to perform "regression." When the input is grouped together into unlabeled classes, the algorithm is said to perform "clustering."

## 2.1 Classification and Decision

Machine learning algorithms which perform classification ("classifiers") and decision ("deciders") are useful for taking input and matching it to a particular class within a set of classes. By definition, the output of a classifier is a discrete value. Classification is often performed with supervised learning, where the machine is provided a training set of example inputs and their associated classes, and the algorithm then produces a decision boundary between each class to predict the classification of future inputs. An example of a decider (classifying input between two possible classes) would be a machine designed to predict whether a tumor is malignant or benign. The machine would be provided a training set of tumor data, including size, location, and class (malignant or benign). The machine would then be tasked with classifying other tumors based on their size and location. [2]

## 2.2 Regression

Machine learning algorithms which perform regression are useful for taking input and predicting an output. The output of a regression algorithm is a continuous value. Regression is often performed with supervised learning, where the machine is provided a training set of examples, and the algorithm generates a closest fit function to predict the output value of future inputs. The example in the previous section of a machine with predicts housing prices by size and location would be an example of regression.

## 2.3 Clustering

Machine learning algorithms which perform clustering are useful for exploring data and finding hidden structure within large sets of data. The example in the previous section of a machine identifying families of flowers is an example of clustering.

# 3 Commonly Used Algorithms

The study of machine learning has produced a large number of useful algorithms. This section will discuss some of the most popular algorithms within machine learning.

## 3.1 Linear and Polynomial Regression

Linear and polynomial regression are classes of supervised machine learning algorithms which form a best fit function to their training data. This hypothesis is optimized to minimize a cost function. The optimization may be performed with a process known as "gradient descent", or by using calculus. [2] For a succinct example, let's describe univariate linear regression (linear regression with one variable).

Let $m$ be the number of training examples for the machine. Let $x$ be an input variable or feature. Let $y$ be the output variable or target variable. Then, $(x, y)$ would be one training example, with $(x^i, y^i)$ being the $i^{th}$ training example. The hypothesis function maps input $(x)$ to output $(y)$:

$$h_\theta(x) = \theta_0 + \theta_1(x)$$

where $\theta_i$ is the $i^{th}$ hypothesis parameter value. Linear regression aims to choose hypothesis parameters such that $h_\theta(x)$ is close to $y$ for the training example $(x, y)$. This is done by minimizing the cost function (also known as the square error function):

$$J(\theta_0, \theta_1) = (\frac{1}{2m}) \sum_{i=1}^{m} [(h_\theta(x^i) - y^i)^2]$$

Gradient descent is an iterative process which attempts to minimize the cost function my making incremental changes to the hypothesis parameters in the direction of the mathematical gradient. This gradient describes the "downhill" direction for the cost function. This process is described as:

repeat until convergence:

$$\theta_j :=\theta_j - \alpha[\frac{\partial}{\partial\theta_j}]J(\theta_0, \theta_1) \text{ for } j = 0 \text{ and } j = 1$$

where $\alpha$ is known as the "learning rate". The learning rate is a constant which dictates how large each incremental step is. With too low a learning rate, gradient descent may take a very long time. With too large a learning rate, gradient descent may not converge at all, as it overshoots the global minimum and actually begins to diverge. It is worth noting that both $\theta_0$ and $\theta_1$ must be updated simultaneously, so in implementation code a pair of intermediate variables would be needed.

As an alternative to gradient descent, calculus can be used through a method called the "normal equation." For brevity, the algorithm will not be described fully here. The advantage of the normal equation is that it is not an iterative approach, and thus there is no need to select a learning rate. Its disadvantage is that, because it has to compute an $n \times n$ matrix, it can be very slow if the training set is very large ($O(n^3)$).

This example describes univariate linear regression. Other variations exist for multivariate regression and polynomial regression, but for brevity they will not be described here.

## 3.2 Logistic Regression

Logistic regression is a supervised classification machine learning algorithm. It is similar in many ways to the regression algorithms above, save for the fact that the hypothesis maps input to a discrete class of outputs. This section will describe a binary classifier (a decider), which maps input to a negative class or a positive class. That is to say that $x$ is mapped to a $y$ in $\{0, 1\}$. The hypothesis produced by logistic regression utilizes a sigmoid function $g(z)$ (also known as a logistic function):

$$g(z) = \frac{1}{1 + e^{-z}}$$

This returns a probability that $y = 1$ on input $x$. The cost function is then described as:

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Again, this cost function is minimized using gradient descent or a calculus-based solution. One way of implementing a multi-class classifier with logistic regression is through one-vs-all classification. In one-vs-all classification, a multi-class problem is converted into multiple binary-class problems. For instance, a machine to classify animals may be made with logistic regression by first identifying the likelihood that the input is a dog vs all other types of animals. Then, it would identify the likelihood that the input is a cat vs all other types of animals. It would repeat this process for all classes upon which it has been trained and return the class with the highest likelihood.

## 3.3 Decision Trees

Decision trees are technically a data structure used by certain machine learning algorithms, but in practice the algorithm which builds up the decision tree simply takes on the name of its backing structure. For the rest of this section, the term decision tree will be used interchangeably to refer to either the backing data

structure or the machine learning algorithm which uses that data structure, as determined by context. Decision trees are a supervised learning classifier. The decision tree is built of two types of nodes: internal nodes (which test the value of a particular feature, branching according to the results), and leaf nodes (which predict a particular class). The basic algorithm to build up a decision tree is quite simple. Given a training set of data, if all examples are of a given set, return a leaf node predicting that class. Else, split the data by the best dividing attribute, and return a new internal node which splits on that attribute. The branches of this internal node are directed to new nodes formed by a recursive call of this function on the split portion of the data. [3] The psuedocode for a binary decision tree (a decision tree with two classes, 0 and 1):

```
GrowTree(S)
    if (y = 0 for all <x,y> in S) return new Leaf(0)
    else if (y = 1 for all <x,y> in S) return new Leaf(1)
    else
        xj = ChooseBestAttribute(S)
        S0 = all <x,y> in S with xj = 0
        S1 = all <x,y> in S with xj = 1
        return new Node(xj, GrowTree(S0), GrowTree(S1))
```

The definition of the ChooseBestAttribute method is somewhat more complex, requiring some information theory, essentially attempting to minimize entropy in each portion of the split. In pseudocode:

```
ChooseBestAttribute(S)
    bestJ = 0
    jEntropy = MAX_DOUBLE
    for i = 1 .. NUM_OF_ATTRIBUTES
        ent = Entropy(i, S)
        if (ent < jEntropy)
            bestJ = i
            jEntropy = ent
    return bestJ
```

Where Entropy(i, S) is an implementation of the mathematical function:

$$H(V) = \sum_{v=0}^{1} -P(H = v) lg P(H = v)$$

Decision trees become considerably more complex when considering non-boolean features. For discrete features, a decision may have mutli-way splits, or test for one value versus all others, or group values into disjoint subsets. For continuous features, a decision tree may have to consider a threshold split. Decision trees have some difficulties with noise in the data (attributes which really have no bearing on classification), attributes with many values (relative to the size of the dataset), and unknown attribute values. Decision trees also have some difficulty with data whose decision boundary is at some slope or curve, as the comparison splits on attributes are always parallel to data axes ("vertical" or "horizontal" in the two-dimensional sense. Still, decision trees remain one of the most popular machine learning algorithm for the ease with which they can be understood and implemented.

## 3.4 Naive Bayes

Bayesian methods use statistical inference to update the probability of a hypothesis based on additional data. The naive Bayes classifier, which utilizes these methods, is a supervised learning classifier. The naive Bayes classifier attempts to produce the most probable hypothesis function using the naive Bayes assumption applied to Bayesian methods:

$$v_{NB} = \underset{v_j \in V}{argmax} P(v_j) \prod_i P(a_i | v_j)$$

The algorithm is considered "naive" because it makes an assumption (that the probability of the attributes given the class are independent) which is patently false in many domains. Yet in spite of this, naive Bayes classifiers remain popular because they perform as well as other classifiers. One drawback of naive Bayes classifiers is that they often produce inaccurate probability estimates, even though they are effective at making correct classifications. Another problem for naive Bayes classifiers is that, for your training set, if no example of target class $v_j$ has attribute $a_i$, then for an input of $a_i$, the classifier will never predict $v_j$. This

is easily corrected by assuming each target class has some prior example of an attribute $a_i$. This is accomplished using an $m$-estimate of $P(a_i|v_j)$:

$$P(a_i|v_j) = \frac{n_c + mp}{n + m}$$

Where $n$ is the number of training examples for which $v = v_j$, $n_c$ is the number of examples for which $v = v_j$ and $a = a_i$, $p$ is the prior estimate for $P(a_i|v_j)$, and $m$ is the weight given to prior. [3]

## 3.5 Neural Networks

Neural networks are supervised learning classifiers. Neural networks came about in an attempt to model a machine learning algorithm after the human brain. Neural networks have many neuron-like threshold switching units and many weighted interconnections between units. They rely on a highly parallel, distributed process. [3] A simple "neuron" representation is the "perceptron," which has a set of inputs, or "dendrites," $\vec{x} = \{x_0, x_1, ...x_n\}$ with associated weights $\vec{w} = \{w_0, w_1, ...w_n\}$. It has a sum function, or "soma," $\sum_{i=0}^{n}(w_i x_i) = \vec{x} \cdot \vec{w}$. Finally, it has an output, or "axon," which is equal to 1 if the soma result is greater than 0, and $-1$ otherwise. The algorithm learns by adjusting the weights of its inputs. One simple method for this follows:

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Where $t = c(x)$ is the target value, $o$ is the output, and $\eta$ is a small constant (around 0.1) known as the "learning rate." [3] This learning rate serves the same purpose as the learning rate above in linear and polynomial regression. In fact, neural networks often implement gradient descent to optimize at this stage by minimizing the square error, as the above equation has one serious drawback. In a similar way to how decision trees form a linear decision boundary, perceptrons form a linear or planar decision boundary. If such a boundary can not be made, the optimization will never converge, as small adjustments are made to the weights each time the output is incorrect. Utilizing gradient descent, the process will eventually converge, minimizing the error over the training set.

# 4 Conclusion

In conclusion, machine learning is a considerably wide field within computer science, and its applications are growing by the day. In a world of immense amounts of data, machine learning is becoming increasingly relevant, being able to organize and analyze data beyond human capability. In a world where people expect a personalized interaction with technology, machine learning is useful for adapting software to the user's individual needs.

# References

[1] P. Kulkarni, "Introduction to reinforcement and systemic machine learning," in *Reinforcement and Systemic Machine Learning for Decision Making*, 1st ed. Wiley-IEEE Press, 2012, pp. 1–21.

[2] A. Ng. (2016) Machine learning. [Online]. Available: https://www.coursera.org/learn/machine-learning/

[3] P. Domingos. (2016) Machine learning. [Online]. Available: https://class.coursera.org/machlearning-001/lecture