Introduction to Machine Learning

Introduction

People often conflate machine learning with artificial intelligence. While the two are closely intertwined, they are not synonymous. Artificial intelligence is concerned with developing synthetic intelligence at some level. At the more basic, and more practical level, it is concerned with expert systems. At a more advanced, and more speculative level, AI is about synthetic consciousness.

Machine learning is concerned with algorithms that improve their performance over time based on input. In other words, the algorithm learns to perform a particular task better over time. This is not generalized intelligence, or even expertise. It is improving the performance of a single, specific task. While that may not sound as exciting as expert systems and artificial consciousness, it is actually something that is working today. There are many real world applications of machine learning. Machine learning is used in facial recognition, natural language processing, analyzing medical images, and many other tasks.

Machine learning involves a range of mathematical techniques including statistics. However, it is important to keep in mind that machine learning is not merely the application of statistics. Simply applying statistics to a given problem may indeed yield results, but that is not machine learning.

Basics of Machine Learning

The concept of machine learning, as was stated in the beginning of this tutorial, is to provide an algorithm a means to improve its performance over time. Most machine learning algorithms can be divided into one of two categories. The first are called supervised machine learning, the second are unsupervised machine learning. The names are a bit misleading. It is not about a human directly supervising every step. It is really about the outcome. In supervised machine learning we know what the desired outcome is, and we are training the algorithm to accurately meet a specific goal.

A great application of supervised machine learning is found in a common laboratory assignment used in machine learning courses. Students are given a dataset of images of birds. The task is to train an algorithm to recognized bird species by analyzing features such as color of plumage, beak shape/size, etc. The goal is already known. The student can easily identify a robin, hawk, blue jay, etc. The task is to train the computer algorithm to do the same. Unsupervised machine learning is often used when we don't know what is actually in the data. We want the algorithm to find specific patterns or clusters. It will still require a human to analyze the results in order to divine their meaning, but the algorithm can inform us of what patterns exist.

When utilizing machine learning, there are various terms one must be familiar with. This is applicable regardless of the type of algorithm or the purpose of the machine learning project. The first such term is domain. A domain is the area in which the machine learning project is applied. For example, when applying machine learning to diagnosing neurological disorders from brain scans, the domain in neurology. One need not be an expert in the domain in order to implement a machine learning project, but having at least some knowledge of the area is essential.

Another term you will encounter frequently is model. This term has a wide range of definitions depending on the application. For example, in Microsoft Azure, a model is defined as a file that has been trained to recognize certain types of patterns[1]. Another common definition is

[1] https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model

that a machine learning model is the output of an algorithm that includes both data and the prediction algorithm[2]. Yet another definition is that a machine learning model is the mathematical representation of the output of the training process[3]. While these varied definitions may seem divergent, even contradictory, they are actually getting to the same point. Once you have completed training an algorithm, it should then be able to accomplish whatever task it has been trained for. The output of training is the machine learning model. Thus, it does indeed combine data and the prediction algorithm. And it is the output of the training process. Furthermore, in the Azure world, it is represented as a file.

Optimization is also commonly encountered in machine learning. Optimization, mathematically, is a process of minimizing some loss function. Loss functions describe discrepancy between predictions of a given model and the actual data found in the field. Optimization, in and of itself, is not machine learning. However, optimization can utilize machine learning. Furthermore, many machine learning algorithms incorporate optimization.

Supervised Algorithms

With supervised machine learning you know the outcome you wish to achieve. For example, you know that you want to identify birds. Each data point that is input has features and associated labels. The goal of a supervised machine learning algorithm is to use those features and map the input to the appropriate output labels. This is done by working with training data until the algorithm achieves sufficient accuracy. The loss function in the algorithm measures the accuracy of the algorithm. All supervised machine learning algorithms share specific steps:

1. Determine the type of data in the training set and gather a training set of data.

---

[2] https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/
[3] https://www.javatpoint.com/machine-learning-models

2. Determine the input features that will be used to evaluate the input data.
3. Choose or design an algorithm for machine learning
4. Run the training set and evaluate the accuracy.

One area of research in machine learning is to determine which algorithms have the greatest accuracy with specific types of data. For example, what algorithms are best at identifying a tumor based on MRI scans of the brain.

Supervised machine learning algorithms also must confront the bias variance problem. Bias errors originate from erroneous assumptions in the algorithm. In statistics, bias refers to the difference between an expected value and the true value of a parameter being estimated. A zero bias means the algorithm is completely unbiased. Variance errors are caused by the algorithm being too sensitive to small fluctuations in the training set. Variance errors usually lead to overfitting the algorithm to the training data set. The problem is that generally speaking, increasing bias decreases variance and vice versa. Figure 9.1 illustrates this concept.

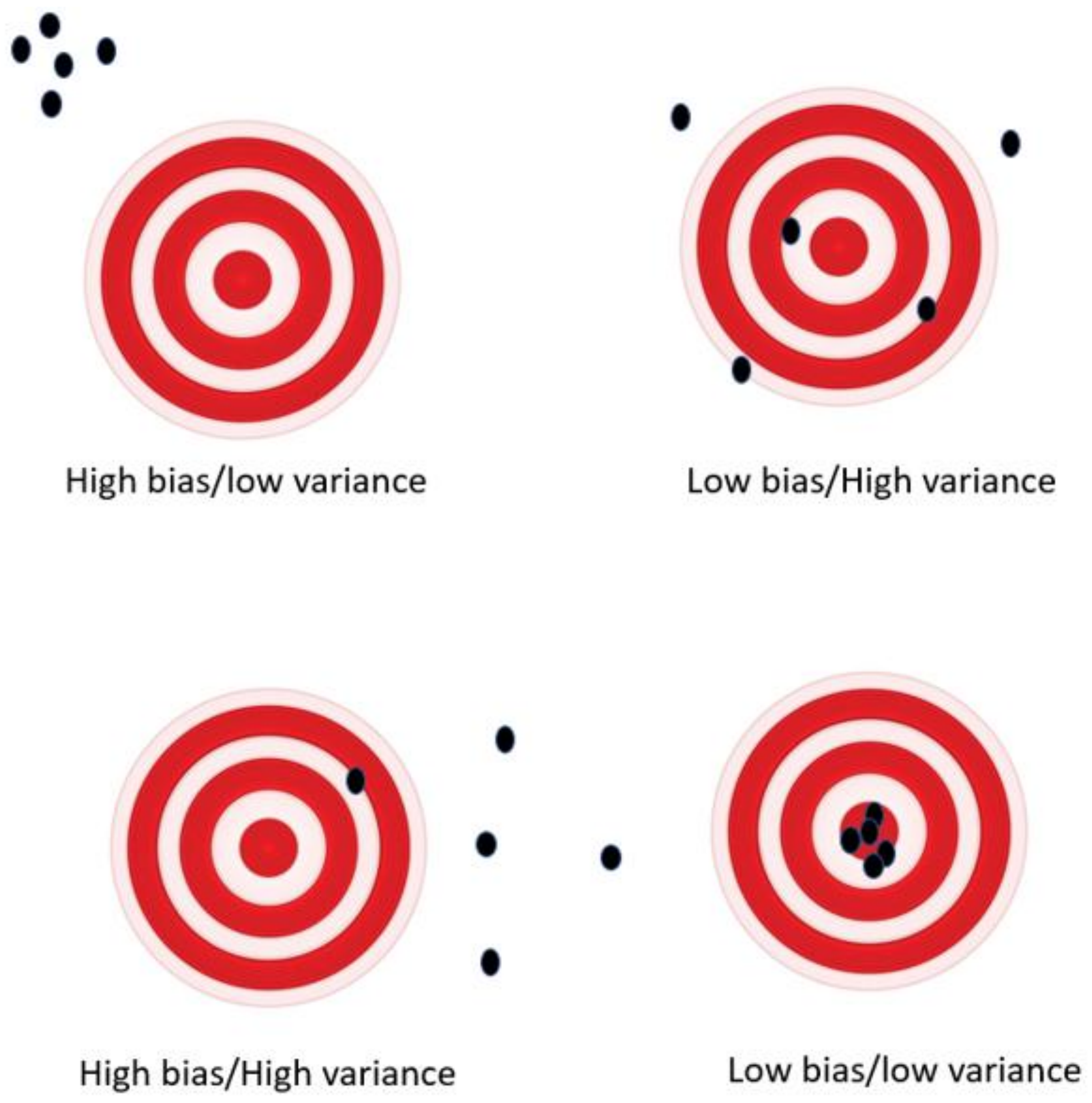


Figure 9.1 Variance and Bias

As can be clearly seen in figure 9.1, the ideal is to have both low bias and low variance. Any other scenario will lead to some type of error.

Unsupervised Algorithms

With unsupervised machine learning, one does not know that result being sought. Consider the example of supervised machine learning using identification of birds. The classifications are known in advance, and a human being can determine if the results are correct. In unsupervised machine learning, the goal is to determine what patterns exist in the data, without a prior determination of what the classifications might be.

Clustering

One common approach to unsupervised machine learning includes clustering methods. The goal of the algorithm is to group the input data in such a manner that items in the same group have more similarities to each other than they do to items in other groups. The human operating the algorithm may not have any idea what those groups will be before the algorithm is executed. This can be one of the most interesting applications of machine learning. Clustering can reveal patterns that the human operating the algorithm may not have even suspected.

There are several different clustering models that can be used. Connectivity models are based on distance. That means that each input set is a vector, and the algorithm determines the distance between input vectors to determine clusters. Another common approach is the centroid model. In this approach each cluster has a single mean vector which is the centroid for that cluster.

Graph models are also important. In graph model clustering, graph cliques are the basis for clustering the input data. A clique is a subset of vertices in a graph such that every two distinct vertices in the clique are adjacent. These models utilize graph theory to cluster data.

Density models are a bit simpler. These models simply look for regions of density in the data space. Such regions are then determined to be clusters, and data is grouped by cluster. This approach is used in algorithms such as DBSCAN.

Anomaly Detection

Anomaly detection methods are used in data analysis as well as machine learning. The concept is fairly simple. The algorithm seeks any outliers or anomalies in the data. An outlier or anomaly is any data point that deviates from the rest of the data enough to consider it may not be related. In statistics, outliers are sometimes removed from the data. However, when using anomaly detection machine learning algorithms, the idea is to find the outliers.

Local outlier factor (LOF) is perhaps the most common algorithm for anomaly detection. This algorithm uses the concept of local density. LOF compares the local density of an object with that of its neighboring data points. If a data point has a lower density than its neighbors, then it is considered an outlier

Specific Algorithms

K-Nearest Neighbor

The k-nearest neighbors' algorithm (KNN) is a non-parametric method used on both regression and classification. For the purposes of malware development, it would be most useful as a classification method for improving target acquisition. When applying k-NN classification the output is membership in a given class. Therefore, classes are predetermined. The simplest model would be target and non-target. However, that flat taxonomy can be expanded to include classifications of likely target and likely non-target. The k-nearest neighbor algorithm is essentially determining the K most similar instances to a given "unseen" observation. Similarity being defined according to some distance metric between two data points. A common choice for the distance is the Euclidean distance as shown in equation 9.1.

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

(eq 9.1)

The algorithm functions by iterating through the entire dataset and computing the distance between x and each training observation. Then the conditional probability for each class is estimated using the function shown in equation 9.2.

$$P(y=j|X=x)=\frac{1}{K}\sum_{i\in A} I(y^{(i)}=j)$$

(eq 9.2)

Naïve Bayes

Naive Bayes classifiers are essentially classifiers that work on probabilities applying Bayes theory. These algorithms are well established and have been studied for decades. They are widely used for categorizing text. For example, spam filters utilize these algorithms. Conditional probabilities in Bayes theorem are often represented by the formula shown here. The C are the classes being examined. This is shown in equation 9.3.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

(eq 9.3)

Naive Bayes is a relatively simple technique for classifying. Class labels are assigned to problem instances. These are represented as vectors of feature values. The features are independent variables in the formula. This is an appropriate modality for training weaponized malware based on selected feature sets for the malware.

There are a number of variations on naive Bayes. Among those variations are the Gaussian naive based which is often used when dealing with continuous data. Each class is distributed according to a Gaussian distribution. Multinomial naive Bayes is used when certain events are generated by a multinomial. The specific selection of a particular version of the Naïve Bayes algorithm will be dependent on the operational needs and the goals of the training and modeling.

Gradient Descent

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks.

Ultimately this algorithm was designed to find the minimum of a function. Starting at the top of the mountain, we take our first step downhill in the direction specified by the negative gradient. Next we recalculate the negative gradient (passing in the coordinates of our new point) and take another step in the direction it specifies. We continue this process iteratively until we get to the bottom of our graph, or to a point where we can no longer move downhill–a local minimum. image source. The size of these steps is called the learning rate. With a high learning rate, one can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. Put more concisely, Gradient descent is an optimization algorithm used to find the values of parameters of a function (f) that minimizes a cost function.

Support Vector Machines

Support vector machines (SVM) also called support vector networks are supervised machine learning algorithms often used for classification. Each data point is viewed as a n-dimensional vector (i.e., a vector of n numbers). The SVM creates a hyperplane or set of hyperplanes that can be used for tasks such as classification. This, of course, necessitates defining a hyperplane. Hyperplanes are a concept borrowed form geometry. In geometry, a hyperplane is a subspace with a dimension that is one less than that of the ambient space. Mathematically, ambient space is the space surrounding some mathematical object along with the object itself. If you consider a 3 dimensional space, then any hyperplanes would be 2 dimensional.

In SVM's the goal is to determine if there is a hyperplane that separates the pints in the input vector. This allows the classification of the data. Consider figure 9.2.

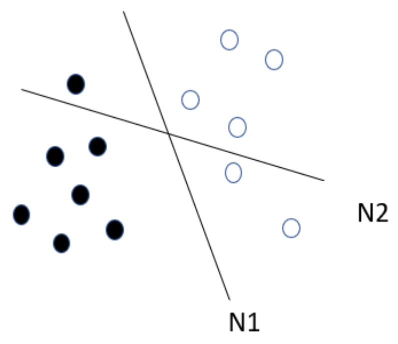


Figure 9.2 Hyperplanes

In figure 9.2, N1 effectively separates the data so that the distance from it to the nearest datapoint is maximized. N2 does not accomplish this goal. Therefore, N1 is used to classify the data. If an appropriate hyperplane exists, as it does in figure 9.2, it is called the maximum margin hyperplane. Put another way "The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points."[4]

Support vector machines are used in image classification, recognizing handwriting, and even classifying proteins. The name stems from the fact that the support vectors are the data points which are closest to the hyperplane. These points define the separating line between classes of data points. Margins are the term for a gap between the two lines that are closest to the class points[5]. Support vector machines are supported in the scikit library[6], which you can use in Python to create machine learning algorithms.

[4] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
[5] https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python
[6] https://scikit-learn.org/stable/modules/svm.html

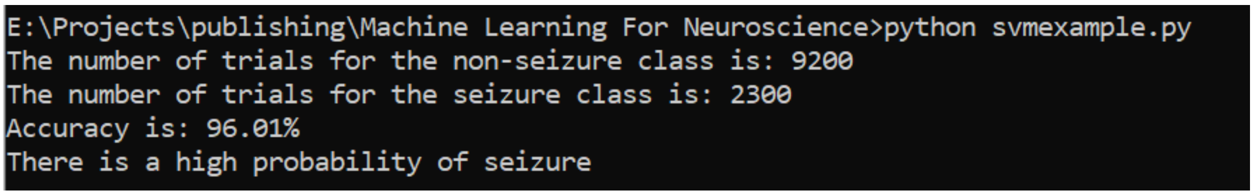
When working with support vector machines, another important concept is the kernel. A kernel transforms an input from the dataspace into the form needed. The two most common types are linear kernels and polynomial kernels. A linear kernel uses a dot product between any two input vectors (if you don't know what a dot product is, it is from linear algebra). A polynomial kernel can be used to distinguish nonlinear input spaces, including curved input spaces. There are other types of kernels such as the radial basis function kernel. The radial basis function kernel is used when there is no prior knowledge regarding the input dataset.

The following code uses a support vector machine to predict the probability of an epileptic seizure. The test data can be downloaded from https://www.kaggle.com/code/yatindeshpande/seizure-prediction-using-svm/data

```
#import libraries

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g., pd.read_csv)
import matplotlib.pyplot as pyplot
import seaborn as sn
import warnings
# note that this next line is importing the support vector machine
# so, most of the work is done for you.
# SVC is the classifer
# see https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score

#import the data. You can download the CSV from
#https://www.kaggle.com/code/yatindeshpande/seizure-prediction-using-svm/data
#you will also find a similar algorithm at that location
EData = pd.read_csv('EpilepsyData/EpilepsyData.csv')
EData = EData.drop(columns = EData.columns[0])
EData.head()
cols = EData.columns
tgt = EData.y
tgt[tgt > 1] = 0
ax = sn.countplot(tgt,label="Count")
non_seizure, seizure = tgt.value_counts()
```

```
print('The number of trials for the non-seizure class is:', non_seizure)
print('The number of trials for the seizure class is:', seizure)
EData.isnull().sum().sum()

Y = EData.iloc[:,178].values
Y.shape
Y[Y>1]=0
Y
X = EData.iloc[:,1:178].values
X.shape

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# if you don't provide a kernel type
# rbf is used by default.
clf = SVC(kernel='poly', degree=8)
clf.fit(X_train, y_train)
y_pred_svc = clf.predict(X_test)
acc_svc = round(clf.score(X_train, y_train) * 100, 2)

print("Accuracy is:",(str(acc_svc)+'%'))
new_input1 = [EData.iloc[6, :177]]

new_output = clf.predict(new_input1)


if new_output==[1]:
   print('There is a high probability of seizure')
else:
   print('There is a low probability of seizure')
```

If you execute the code as written you should see the result shown in figure 9.3



```
E:\Projects\publishing\Machine Learning For Neuroscience>python svmexample.py
The number of trials for the non-seizure class is: 9200
The number of trials for the seizure class is: 2300
Accuracy is: 96.01%
There is a high probability of seizure
```

Figure 9.3 SVM

Note the line of code that states:

```
   clf = SVC(kernel='poly', degree=8)
```

This is choosing a kernel type. If you don't choose any it will use the radial basis function. To use the default, you would simply write:

```
      clf = SVC()
```

If you wish to use the sigmoid kernel try this line

```
        clf = SVC(kernel='sigmoid')
```

It is a good practice to try different kernels to compare the performance. There is a reason that there are a range of kernels to choose from. Some kernels are more appropriate for particular applications and particular datasets. Unfortunately, this part of machine learning is a bit of an art, and requires trial and error.

Feature Extraction

Many algorithms will require you to extract the features of interest from the dataset being used. There are several algorithms one can use to do this, the most common are discussed in this section.

PCA

Principle component analysis (PCA) is a very widely used technique. It is particularly useful for large data sets that have a high number of features. PCA is essentially a statistical technique for reducing the dimensionality of a dataset. The principle components for a set of points in a real coordinate space are a series of p unit vectors where the i-th vector is the direction of a line that best fits the data. If this description seems a bit vague to you, it is based on a fundamental linear algebra understanding.

PCA is not necessarily appropriate for all applications. The following quote defines three situations wherein PCA is appropriate[7].

Do you want to reduce the number of variables, but aren't able to identify variables to completely remove from consideration?

Do you want to ensure your variables are independent of one another?

Are you comfortable making your independent variables less interpretable?

Principle component analysis finds lines and planes in the K-dimensional space that approximate the data as closely as possible. Closely is defined using least squares. Least squares is a method to find the line that best fits the data. A line or plane that is the least squares approximation of a set of data points makes the variance of the coordinates on the line or plane as great as possible. Fortunately, PCA is actually build in to sklearn.decomposition. The following code will demonstrate PCA.

```
#import your modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn.decomposition import PCA

#create random dots to illustrate
rng = np.random.RandomState(1)
X = np.dot(rng.rand(2, 2), rng.randn(2, 100)).T
plt.scatter(X[:, 0], X[:, 1])
plt.axis('equal')
plt.show()

#you should expiriment with different
#numbers of components
pca = PCA(n_components=4)
pca.fit(X)
print(pca.components_)
print(pca.explained_variance_)
```

[7] https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

```
def draw_vector(v0, v1, ax=None):
    ax = ax or plt.gca()
    arrowprops=dict(arrowstyle='->',
                    linewidth=2,
                    shrinkA=0, shrinkB=0)
    ax.annotate('', v1, v0, arrowprops=arrowprops)

# plot data
plt.scatter(X[:, 0], X[:, 1], alpha=0.2)
for length, vector in zip(pca.explained_variance_, pca.components_):
    v = vector * 3 * np.sqrt(length)
    draw_vector(pca.mean_, pca.mean_ + v)
plt.axis('equal');
plt.show()
```

When you execute the code, you will see the following three images, figures 9.4 to 9.6.

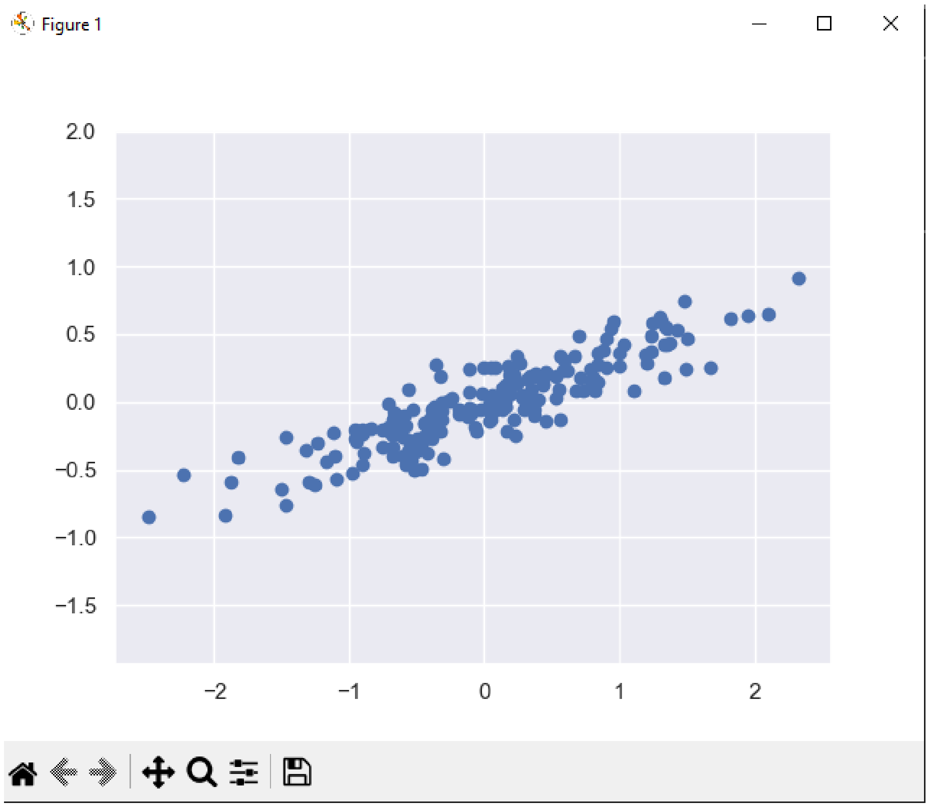


Figure 9.4 Data before PCA

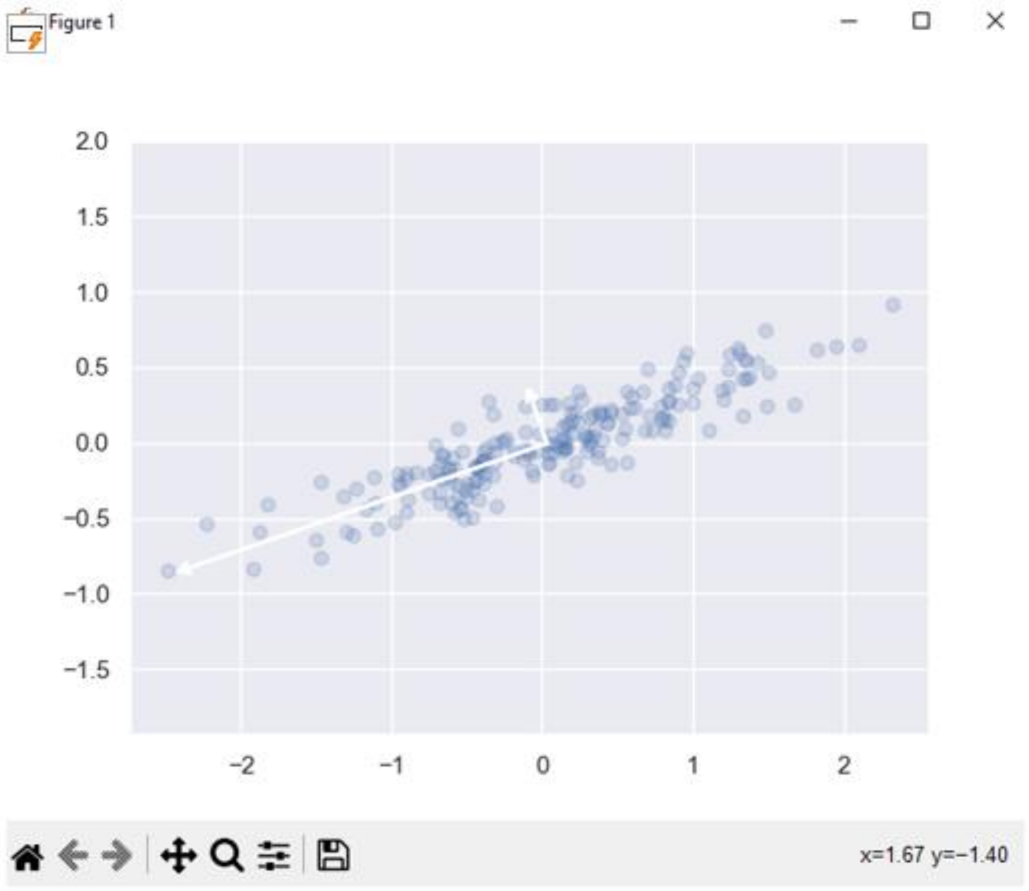
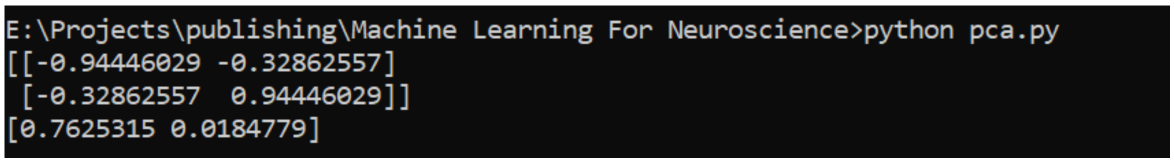


Figure 9.5 Data after PCA



```
E:\Projects\publishing\Machine Learning For Neuroscience>python pca.py
[[-0.94446029 -0.32862557]
 [-0.32862557  0.94446029]]
[0.7625315 0.0184779]
```

Figure 9.6 PCA Variance Data

As you can see, sklearn.decomposition includes the PCA algorithm for you. You could implement the algorithm with even less information than is provided in this section. However, it is always best to have a general understanding of what is being done, even if you have a tool that automates the action for you.

Artificial Intelligence

As was pointed out at the beginning of this tutorial, people often conflate artificial intelligence with machine learning. This section will briefly touch on the subject of artificial intelligence.

General Intelligence

Actual synthetic consciousness is not something that has been developed, or that is even close to being developed. However, more limited forms of machine intelligence can and have been developed. One such form of machine intelligence are expert systems. Expert systems utilize a knowledge base to make inferences from data. In many cases these systems will use machine learning algorithms, such as the ones we have discussed in this tutorial, as part of their functionality. The primary difference between simply applying machine learning and having an expert system is the breadth of applicability. A machine learning algorithm might learn to recognize tumors in MRI brain scans. An expert system can take that information and along with its knowledge base, make inferences regarding prognosis and treatment.

Medical expert systems are the systems of most interest in text focused on machine learning for neuroscience. A journal article from 1987 describes medical expert systems, and this description should aid you in understanding how expert systems work[8]:

[8] https://journal.chestnet.org/article/S0012-3692(15)42851-X/fulltext

"A medical expert system is a computer program that, when well-crafted, gives decision support in the form of accurate diagnostic information or, less commonly, suggests treatment or prognosis. Diagnostic, therapeutic, or prognostic advice is given after the program receives information (input) about the patient, usually via the patient's physician. Expert systems have characteristics which make them dissimilar from other kinds of medical software. One of these characteristics is that the sequence of steps used by the expert system in coming to a diagnostic or therapeutic conclusion often is designed to mimic clinical reasoning. Also, the sequence of steps is, in many expert systems, available to the physician using the system. Because clinical medicine often does not deal in certainty, expert systems may have the capability of expressing conclusions as a probability. It is generally agreed that expert system software must contain a large number of facts and rules about the disease or condition in question in order to deliver accurate answers. It has been estimated that two general internal medicine textbooks and three specialty textbooks would require 2 million rules.

Because large amounts of data are needed, in the recent past, expert systems were only feasible when used with large, expensive computers. With the advent of more powerful microcomputers and more efficient microcomputer languages, expert systems could now be available to any physician with a microcomputer."

In general, to be considered an expert system, the system must exhibit several characteristics. The first, and probably the most obvious, is a high level of expertise. The system should be able to make decisions on par with those of human experts in the field. In the case of neurology, which means an artificial system whose decisions are comparable to that of trained neurologists. The system should also be reliable and flexible. Perhaps most importantly, the system should not be prone to errors, as a human would be.

Synthetic Consciousness

Before one can effectively address the question synthetic consciousness, it is first instructive to examine the basis for biological consciousness. While one position of this paper is that synthetic consciousness need not necessarily be analogous to human consciousness, briefly examining human consciousness does provide a useful starting point for researching artificial consciousness. There have been a range of ideas posited regarding consciousness ranging from the philosophical to the biological. Of particular interest for developing synthetic consciousness is the orchestrated objective reduction theory posited by Roger Penrose and Stuart Hameroff[9]. Orchestrated objective reduction (often called Orch OR) is the hypothesis that biological consciousness is an emergent property of quantum activity within the microtubules of neurons. This is a divergence from classical neurological theories that postulate biological consciousness is an emergent property of the computations performed by neurons[10]. Essentially, classical neurobiology and cognitive science see neuron activity reaching a threshold of complexity that leads to consciousness emerging. Conversely Orchestrated Objective Reduction sees consciousness as an emergent property derived from quantum activities within the microtubules of the neurons themselves.

[9] Hameroff, S., & Penrose, R. (2014). Consciousness in the universe: A review of the 'Orch OR'theory. Physics of life reviews, 11(1), 39-78.
[10] Fingelkurts, A. A., Fingelkurts, A. A., & Neves, C. F. (2013). Consciousness as a phenomenon in the operational architectonics of brain organization: criticality and self-organization considerations. Chaos, Solitons & Fractals, 55, 13-31.

Essentially Hameroff proposed that microtubules were suitable candidates for quantum processing. Microtubules contain hydrophobic pockets that can contain delocalized electrons. Hameroff further posited that these electrons can become quantumly entangled and would form a Bose-Einstein condensate. Penrose and Hameroff' s theories have been widely criticized in the neuroscientific community. Among the criticisms have been arguments that a biological system cannot avoid quantum de-coherence due to the environment of the biological system. However, advances in quantum computing have cast doubt on this criticism as researchers have achieved quantum states for several seconds at room temperature[11]. Furthermore, evidence suggests that plants routinely use quantum-coherent electron transport mechanisms as part of photosynthesis[12].

Penrose and Lucas also argued that Gödel's theorem dictates that no process or algorithm can deterministically predict its outcome. Penrose further posited that this meant no algorithmic process could lead to consciousness. This was the issue that led Penrose to explore quantum behavior as the substrate for human consciousness, due to the non-deterministic nature of quantum interactions. In addition to forming a theoretical model of the basis for human consciousness, Penrose and Lucas were stating that true artificial intelligence is not possible from an algorithmic process.

Many neuroscientists disagree with Penrose, Hameroff, and Lucas. Instead, the prevailing opinion in neuroscience is that consciousness is an emergent property that is predicated on

---

[11] Neumann, P., Kolesov, R., Naydenov, B., Beck, J., Rempp, F., Steiner, M., ... & Pezzagna, S. (2010). Quantum register based on coupled electron spins in a room-temperature solid. Nature Physics, 6(4), 249.

[12]
Lambert, N., Chen, Y. N., Cheng, Y. C., Li, C. M., Chen, G. Y., & Nori, F. (2013). Quantum biology. Nature Physics, 9(1), 10. Laureys, S., Gosseries, O., & Tononi, G. (Eds.). (2015). The neurology of consciousness: cognitive neuroscience and neuropathology. Academic Press.

meeting a certain threshold of neurological complexity. In this view, consciousness will emerge when the neurology reaches a particular level of complexity. There is a body of evidence that supports this view. Primarily, comparative studies of neuro anatomy and physiology among diverse species shows at least some association between the complexity of the brain and the level of consciousness. Neurological complexity is more than just the number of neurons in the system. It also involves the connectivity between neurons. The complexity of the system in its entirety appears to have at least some correlation to consciousness.

Yet another neurobiological view of consciousness was posited by Nobel Laurate Francis Crick. Crick posited that consciousness in inextricably associated with how the brain uses short-term memory processes to facilitate sensory input[13]. Crick focused primarily on visual input, but his work is applicable to any sensory input. His focus was on the neurological correlates of visual processing. Not merely detecting an object, but processing that sensory input.

What all of this research really means is that we do not yet have a clear understanding of the origin of human consciousness. Without that, it would be quite difficult to create synthetic consciousness. Before one can even consider research into synthetic consciousness, it is necessary to first define consciousness. Blackmore[14] states that consciousness has no generally accepted definition in science or in philosophy. This statement, while accurate, does not address the problem encountered in artificial intelligence research. Blackmore's commentary is more applicable to a complete definition of consciousness for the purposes of cognitive science and psychology. For the purposes of furthering research into artificial consciousness, it is not necessary to derive a broadly applicable, generally accepted definition of consciousness. What is

[13] Crick, F., & Koch, C. (2003). A framework for consciousness. Nature neuroscience, 6(2), 119.
[14] Blackmore, S. (2013). Consciousness: an introduction. Routledge.

required is a minimalistic definition that is operationally effective. While cognitive scientists may explore a range of definitions of consciousness[15], for artificial intelligence research it is only necessary to select a single, operationally viable, elementary definition of consciousness. For the purposes of this paper, a simple definition of consciousness is espoused. That definition is simply self-awareness[16] . Therefore, synthetic consciousness would be operationally defined as any artificial device or software that is self-aware. This is a minimalistic definition and does not attempt to address issues regarding emotions, other related cognitive functions or philosophical questions.

Self-awareness provides a simple definition that avoids a range of philosophical issues as well as being basic enough to be acceptable as a minimalistic operational starting point. There certainly are researchers that would add to that definition, but as a minimal definition that can be broadly accepted and operationally effective, self-awareness is an operative definition. However, even with this simple definition, there is still an issue of how to recognize and measure self-awareness. Identifying self-awareness depends on a reliable methodology. The Turing test has long been accepted as a mechanism for identifying artificial intelligence. While this has been accepted for decades, recent advances in software cast some doubt on the reliability of that testing modality. It is certainly possible now for software to be programmed to simulate a level of dialog that is difficult to differentiate from human, self-directed dialog. It should also be noted that the nature of the Turing test has an implicit assumption that consciousness must at least, at a

[15] Bermúdez, J. L. (2014). *Cognitive science: An introduction to the science of the mind*. Cambridge University Press.

[16] Pope, K. (Ed.). (2013). The stream of consciousness: Scientific investigations into the flow of human experience. Springer Science & Business Media.

superficial level, appear analogous to human consciousness. This is an understandable definition, given that human consciousness is the model that researchers have at hand. However, that model entails a number of complexities that are not necessary to detect consciousness. It also assumes human consciousness is the only model for consciousness. Therefore, the Turing test may not be the appropriate modality for detecting synthetic consciousness.

Exercises

Lab 1 Detecting Parkinson's

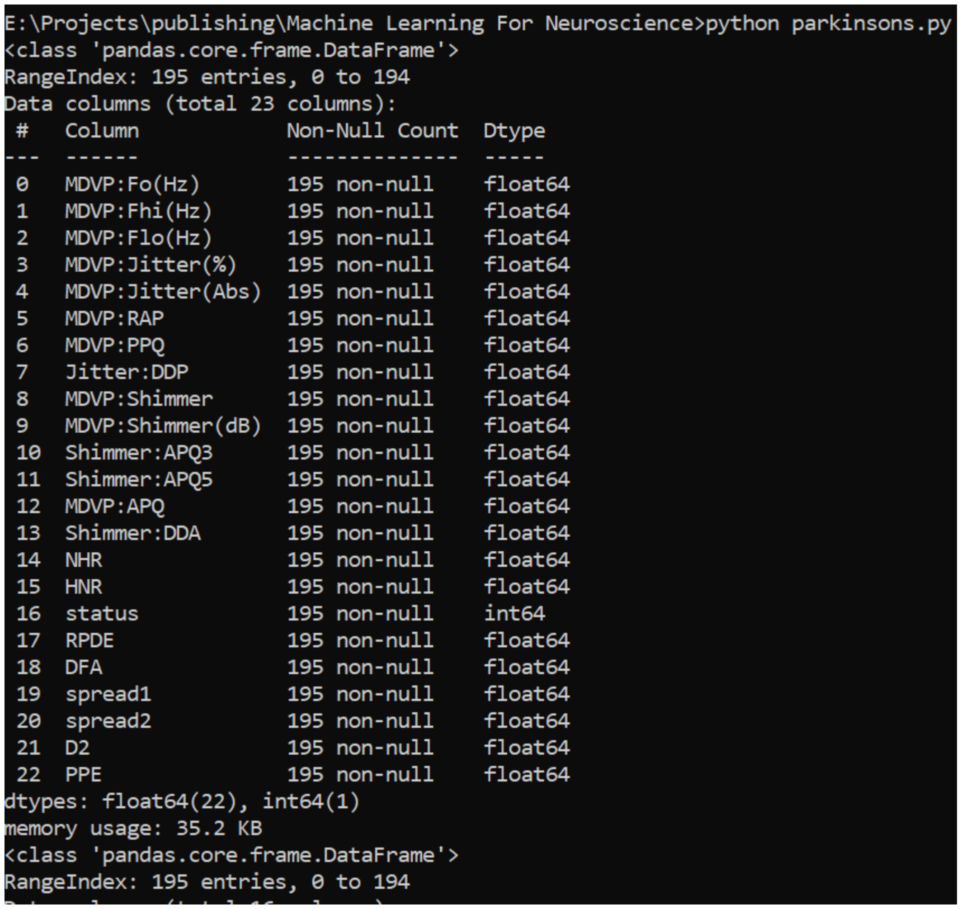
First you will download the dataset from Kaggle https://www.kaggle.com/code/vuppalaadithyasairam/feature-selection-xgboost-97-4-test-acc/data . You will also find similar Python scripts there.

Now you will code the following script

```
import numpy as np #
import pandas as pd # data processing, CSV file I/O
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

#read in data. Remember when you download you need
#to change the file name to one word with .csv not
#.xls
data=pd.read_csv('parkinsons/Parkinssondisease.csv')

#now display the data we have
data.head()
data=data.drop(['name'],axis=1)
data.info()


corr=data.corr()
cor_target = abs(corr["status"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.3]
relevant_features
data=data.drop(['MDVP:Fhi(Hz)','MDVP:Jitter(%)','MDVP:RAP','MDVP:PPQ','Jitter:DDP','NHR','DFA
        '],axis=1)
data.info()
for x in data.columns:
   data[x]= (data[x]-data[x].min())/(data[x].max()-data[x].min())
data.head()
y=data['status']
```

```
x=data.drop(['status'],axis=1)

X_train,X_test,y_train,y_test= train_test_split(x,y,test_size=0.2,stratify=y)


svc_model=XGBClassifier()
svc_model.fit(X_train,y_train)


predictions= svc_model .predict(X_train)
percentage=svc_model.score(X_train,y_train)
res=confusion_matrix(y_train,predictions)
print("Training confusion matrix")
print(res)
predictions= svc_model .predict(X_test)
train_percentage=svc_model.score(X_train,y_train)
test_percentage=svc_model.score(X_test,y_test)
res=confusion_matrix(y_test,predictions)
print("Testing confusion matrix")
print(res)
# check the accuracy on the training set
print(svc_model.score(X_train, y_train))
print(svc_model.score(X_test, y_test))
print(f"Train set:{len(X_train)}")
print(f"Train Accuracy={train_percentage*100}%")
print(f"Test set:{len(X_test)}")
print(f"Test Accuracy={test_percentage*100}%")
```

When you execute this you should see output like you see here:



```
E:\Projects\publishing\Machine Learning For Neuroscience>python parkinsons.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   MDVP:Fo(Hz)       195 non-null    float64
 1   MDVP:Fhi(Hz)      195 non-null    float64
 2   MDVP:Flo(Hz)      195 non-null    float64
 3   MDVP:Jitter(%)    195 non-null    float64
 4   MDVP:Jitter(Abs)  195 non-null    float64
 5   MDVP:RAP          195 non-null    float64
 6   MDVP:PPQ          195 non-null    float64
 7   Jitter:DDP        195 non-null    float64
 8   MDVP:Shimmer      195 non-null    float64
 9   MDVP:Shimmer(dB)  195 non-null    float64
 10  Shimmer:APQ3      195 non-null    float64
 11  Shimmer:APQ5      195 non-null    float64
 12  MDVP:APQ          195 non-null    float64
 13  Shimmer:DDA       195 non-null    float64
 14  NHR               195 non-null    float64
 15  HNR               195 non-null    float64
 16  status            195 non-null    int64
 17  RPDE              195 non-null    float64
 18  DFA               195 non-null    float64
 19  spread1           195 non-null    float64
 20  spread2           195 non-null    float64
 21  D2                195 non-null    float64
 22  PPE               195 non-null    float64
dtypes: float64(22), int64(1)
memory usage: 35.2 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
```

Much of the output may be new to you. Some of it is specific to Parkinson's diagnosis. Some terms are defined here for you. A more comprehensive discussion of the Parkinson's specific terms can be found at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5434464/

D2: Correlation dimension, *a measure of the dimensionality of the space occupied by a set of random points. This comes from chaos theory.*

RPDE: Recurrence period density entropy. THis is used in dynamical systems to determine periodicity.

Shimmer:DDA: Dysphonic Voice Pattern Analysis

Shimmer:APQ3: This Is a three point amplitude perturbation quotient. It Is analyzing voice patterns.