Chapter 9: Machine Learning

“…The growth in algorithmic modeling applications and methodology has been rapid. It has occurred largely outside statistics in a new community—often called machine learning—that is mostly young computer scientists.”

Leo Breiman, Statistical Modeling: The Two Cultures

There are two types of people within information security, those who are completely intimidated by machine learning and those who know machine learning largely solved the spam problem and are completely intimidated by machine learning.

It’s easy to be intimidated when machine learning is described as “a type of artificial intelligence that provides computers with the ability to learn without being explicitly programmed”.[[1]](#footnote-1) How can a computer do anything with being explicitly programmed? Or better yet, this rather well known definition from Tom Mitchell in his 1997 book:

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

Are you clear now on what Machine learning is? At the off chance that things aren’t clear, we are going to just dive right into machine learning with a rather simplified example. We will start this chapter off with some data and then not explicitly program the computer to learn from experience E and perform a task T, sound good?

Detecting Malware

Let’s assume that you have been able to record memory and processor usage on all of your systems. With some effort, you have been able to inspect almost 250 of the computers, discovering that some of the systems are infected with malware and some are operating normally (without malware). But you’ve got 445 other systems that haven’t been inspected and you’d like to save some time and use the data you have to determine if the other 445 systems you have are infected or not.[[2]](#footnote-2) First start by loading up the data on the hosts you know about and inspecting it.

memproc <- read.csv("data/memproc.csv", header=T)

summary(memproc)

## host proc mem state

## crisnd0004: 1 Min. :-3.1517 Min. :-3.5939 Infected: 53

## crisnd0062: 1 1st Qu.:-1.2056 1st Qu.:-1.4202 Normal :194

## crisnd0194: 1 Median :-0.4484 Median :-0.6212

## crisnd0203: 1 Mean :-0.4287 Mean :-0.5181

## crisnd0241: 1 3rd Qu.: 0.3689 3rd Qu.: 0.2413

## crisnd0269: 1 Max. : 3.1428 Max. : 3.2184

## (Other) :241

You can see there are 53 hosts identified as “infected” and 194 identified as “normal”. Also, notice that both the processor data and the memory information has been normalized (see the discussion of z-score in Chapter 5). But that will keep the numbers on the same scale. Scaling the variables like this is important when comparing across variables in some machine learning approaches. In order to explore this a bit more, let’s plot this data comparing the processor data to the memory and differentiate based on the malware state.

gg <- ggplot(memproc, aes(proc, mem, color=state))

gg <- gg + scale\_color\_brewer(palette="Set2")

gg <- gg + geom\_point(size=3) + theme\_bw()

print(gg)

Figure 9.1 Processor and Memory across systems [FILENAME 793725c06f001]

Notice how the infected systems appear to generally use more processor and memory? Perhaps you could develop an algorithm to classify this data just based on the relative location of the known hosts. But before you get too far, you’ll want to do a little planning. First you’ll want to figure out what machine learning algorithm you will want to apply, and then you should figure out how to test if the algorithm is any good. In a real problem, you would probably want to try several different algorithms and features, but we will discuss that later in this chapter.

Developing a machine learning algorithm

Did this title make you get a little worried? Did you have flashes of fear that we’ll start talking about mathematical formulas and make you say things like “sub i of x”? Don’t get too worried, we will keep this as light as we can and we will start by demystifying the word “algorithm”. Anytime you see the word algorithm try to mentally replace it with “a series of instructions” because that’s all an algorithm is. You’ll want to develop a series of instructions for the computer on how to inspect and understand the data (so it can learn about it) and then how to apply that learning to the systems we don’t know about.

Do you see how you are not explicitly programming? Even though you are absolutely writing a program for the computer, you will not be explicitly writing the decision criteria and that’s the difference. Your series of instructions (the algorithm) will explicitly tell the computer how to inspect the data and how it should build up its decision criteria from the data, but it will not tell the computer the decision criteria directly. Compare that to the traditional approach of programming firewall and intrusion detection/prevention systems. Humans try to think up what’s best and then explicitly program the rules the machines should follow. There is a limit to that approach and unfortunately our security systems reached that limit years ago. With machine learning, you are asking the computer to look through the data and identify and apply the differences and subtleties that you would never be able to find by yourself.

Looking back at the data in Figure 9.1, you’ll want to create a series of instructions to learn about the processor and memory usage on the normal hosts and compare them to the processor and memory usage on the infected hosts. Once the machine has some notion of a difference between the two sets, you can give it some instructions on how to apply that information to the unclassified systems. Remember the goal here is to have the computer give its best guess on whether or not a system is infected with malware or not. Let’s create a short algorithm that is easy to understand and easy to follow:

1. “Train” an algorithm:
   1. Calculate the average (mean) processor and memory usage for infected systems.
   2. Calculate the average (mean) processor and memory usage for normal systems.
2. Make a prediction using processor and memory usage for an unknown system:
   1. If the processor and memory usage are closer to the average infected machine, label it as infected.
   2. If the processor and memory usage are closer to the average normal machine, label it as normal.

Congratulations! You have written your first machine learning algorithm and now the computers are one step closer to world domination with this extra bit of artificial intelligence! Notice the choice of wording in the first step, you’ll want to “train” the algorithm. That’s the term used to describe when the machine is learning from the data, it’s being “trained” by the data.

Validating the Algorithm

Before you go off thinking that this algorithm is helpful, you should probably make sure. You’ll want some way to test how accurate this algorithm is at predicting infected systems. Rather than using all of this data to train the algorithm, how about you hold back some of the data to test how accurate the algorithm can predict? The process of “making sure” an approach is helpful is one of the strong suits of machine learning. It has evolved just as much (if not more so) in computer science as it has statistics and there is a strong element of pragmatism in the field. There are many techniques of validating algorithms, but we will just keep this simple and split the original data into two datasets. In reality, you would probably create multiple datasets from the original data, and train the data over multiple iterations (and validations). Cross-validation, bootstrapping and boosting are all very helpful approaches that have been developed to do that validation.

Once we split our data into two groups, we will call the training data, since we’ll use that to train the algorithm, and the second we’ll call the test data, since we’ll use that to test. To split the data randomly (which is important to get representation), make use of the sample() command. In this case, you will be pulling a random sample of the indexes of the original data and using that to split into the train and test data. While there’s no definitive rule on where to make the split (different techniques split in different ways), we will simply take one-third for the test data and train the algorithm on the other two-thirds.

# make this repeatable

set.seed(1492)

# get how many in the overall sample

n <- nrow(memproc)

# set the test.size to be 1/3rd

test.size <- as.integer(n/3)

# randomly sample the rows for test set

testset <- sample(n, size=test.size)

# now split the data into test and train

test <- memproc[testset, ]

train <- memproc[-testset, ]

Now you can train the algorithm on the train data and verify how good it is with test data. Please keep in mind, just splitting the data like this once to measure its accuracy is better than just assuming the algorithm is good, but as we’ve mentioned there are plenty of robust methods for validation.

Implementing the Algorithm

We stated earlier that the first step in training this algorithm was to calculate the average (mean) for the infected processor and memory usage and the normal processor and memory usage. This is done by subsetting the rows on the state field (so only infected or normal is returned) and subsetting the columns for just the proc and mem fields. That reduced data can be passed directly into colMeans(), which will compute the means on the columns and return a named vector with two elements.

inf <- colMeans(train[train$state=="Infected", c("proc", "mem")])

nrm <- colMeans(train[train$state=="Normal", c("proc", "mem")])

# view the results

print(inf)

## proc mem

## 1.152025 1.201779

print(nrm)

## proc mem

## -0.8701412 -0.9386983

The differences between the means here is not small and this may do relatively okay for how simple this algorithm is. With the algorithm now trained and ready to predict, the next step is to create a predict.malware() function. This will take in a single proc and mem value and calculate how far those are from the means that we generated during the training. What is the best way to calculate distance? Think back to geometry class and the Pythagorean theorem: a2 + b2 = c2, where a and b are the two sides of the triangle and c is the hypotenuse. In our case, a is the difference between the trained proc mean and the test proc value and b is the difference between the trained mem mean and the test mem value. Once we get the two distances, we simply compare them and which ever is closer (smaller) is the one we will predict.

predict.malware <- function(proc, mem) {

# set up infected comparison

inf.a <- inf['proc'] - proc

inf.b <- inf['mem'] - mem

# pythagorean distance c = sqrt(a^2 + b^2)

inf.dist <- sqrt(inf.a^2 + inf.b^2)

# repeat for normal systems

nrm.a <- nrm['proc'] - proc

nrm.b <- nrm['mem'] - mem

nrm.dist <- sqrt(nrm.a^2 + nrm.b^2)

# assign a label of the closest (smallest)

ifelse(inf.dist<nrm.dist,"Infected", "Normal")

}

Feel free to pass in a few values and see how it does if you’d like. But at this point, everything is ready to run against the test data. To pass in the test data you can use the apply function with the first argument being the test data set, the second argument being a “1” to denote to apply over the rows (instead of a “2” for columns), and then we pass in the function. In this case, the function takes in a variable called row, which is character vector in this case. Since it is reduced to a vector, and there is a character in the vector (state and host), the whole vector is converted to a character vector. Therefore, the first thing the function does is convert the two values we want back to numeric values.

prediction <- apply(test, 1, function(row) {

proc <- as.numeric(row[['proc']])

mem <- as.numeric(row[['mem']])

predict.malware(proc, mem)

})

type="note"

Don’t try this (algorithm) at home

This is a great time to point out that this is a very basic algorithm and it’s only for discussion purposes. This is a very simplified version of what’s known as a maximal margin classifier and for it to work well, the data must be able to be separated by a hyperplane (or a straight line in our example since we only have two variables). This is not something we would recommend implementing on real data simply because real data is generally not nice enough to split into two convenient groups.

Once the test data is run through that code, we have a set of predictions and the ability to compare to the real values (see the power of this method?). To look at how well it did, you’ll want to look at the proportion of correctly predicted results on the test data. You can calculate that by taking the number of correct predictions and divide it by the total number of predictions.

sum(test$state==prediction)/nrow(test)

## [1] 0.8780488

This very simple algorithm predicted almost 88% of the values correctly, which is probably more a statement about how segregated the data is than the strength of the algorithm. But overall, 88% is pretty good for your first machine learning algorithm, congratulations! The results are also pictured in figure 9.2. The way this classifier works is it creates a line halfway between the two means and perpendicular to an intersecting line. Anything above the line is predicted as infected, anything below is predicted to be normal. The misclassified values are clearly marked on the graphic, you can see how any normal systems above the line are mislabeled as well as any infected systems below the line.

Figure 9.2 Predictions from our algorithm [FILENAME 793725c06f002]

Benefiting from Machine Learning

Now that we’ve walked through a rather simple (perhaps too simple) example, you should have a basic understanding of the mental shift machine learning brings. Rather than focusing on rule sets and signatures, machine learning can shift the focus towards continual adaptation based on computers learning directly from the data. Hopefully, the days of thresholds and regular expressions rules are behind us.

Before we can talk about the benefits of machine learning, we should talk about the two types of machine learning algorithms: *supervised* and *unsupervised* and which approach to use is determined more by the type of data you have than any preference. **Supervised algorithms** require that the training set have known samples just like the opening example of this chapter. The data in that example was from collected from hosts that were identified as either infected with malware or not. Another example is the Zero Access data in Chapter 5, where we knew how many infections in each state and county and we could correlate that with other data about the states and counties.

**Unsupervised algorithms** are usually applied to data when what you’d like to know is unknown. For an example of this, think of the recommendation systems at Amazon or Netflix. Those systems begin with a history of movie rentals or purchases and apply unsupervised learning to group similar people (their habits actually) based on patterns in the data. This enables them to recommend products that other people like you have purchased. Given the unsupervised nature of these approaches, it is difficult to definitively prove something with unsupervised methods, but that’s not what these are designed to do. Unsupervised methods enable you to discover relationships and explore the data like no other approach and as we’ll see, we can discover some interesting things with unsupervised learning methods.

Answering Questions with Machine Learning

What types of questions can machine learning answer? What sort of problems can it solve? The opening example in this chapter already introduced the concept of classification, when we tried to determine if the hosts were infected or not. **Classification** is the process of identifying the category something belongs in, or which label should be applied. Classification always begins with a list of possible categories and known data that describes those categories (so they are supervised algorithms). Many of the challenges within information security revolve around a single classification problem, “Is this malicious or not?” Mechanisms exist to authenticate and authorize users, but do their actions match that of a normal user or a malicious user? Is this HTTP request valid or is the source attempting something they shouldn’t be? These are all questions that classification algorithms are best at tackling.

What if what you are more interested in is forecasting a quantity? Machine learning (and classical statistics) offers methods to do **quantitative prediction**. The overall approach may make people with a strong engineering background a bit uneasy thinking that prediction is impossible. But relax, nobody is claiming that the precise future is hidden in the data, however you can use the data to make a pretty good estimate. Given a set of observations and the outcome that resulted (so again, these are supervised methods), we can build predictive models make estimates for known future states. Think back to the linear regression analysis we did in Chapter 5. If by some strange turn of events another state appears with 6 million people, that regression analysis using just population would predict just under 5,000 Zero Access infections in that state. While that example isn’t exactly practical, you could use the technique to estimate bandwidth usage next month, or even forecast the size of the next DDoS attack.

Sometimes the end result isn’t prediction of a quantity or category. Sometimes we just want to know about the variables we observe and how they contribute and interact. For these cases we want to apply methods for **inference**. Inferential methods allow you to describe your environment. How important are these variables? Are data around processor and memory usage the best predictors of an infected machine? For example linear regression enables you to toss multiple variables into a single analysis and see how each of them contributes to the outcome and the quantitative relationships. Both supervised and unsupervised methods support inference about the variables and it’s an important part of any model or algorithm.

The last application of machine learning is for **exploration and discovery**. This is an area that unsupervised algorithms truly excel but supervised methods can also support exploration. Sometimes you may find yourself just sitting on a mound of data and you’d like to know what sort of relationships or patterns exists in the data. Using methods like mutidimensional scaling or principal component analysis will help you explore and gain perspectives of the data that just isn’t possible with simple descriptive statistics.

Selecting Features

One of the less talked about things within machine learning is how you’ll go about selecting the data to collect and include in any analysis. The variables that you collect and include are called *features*. Within classic statistics they are also called explanatory, independent or predictor variables. But within machine learning these variables are referred to as the features of the algorithm. Within classic statistics

variable importance

variable selection

Identifying which features to include often becomes more of an art than a science. Luckily there is plenty of feedback to know when feature selection stinks. Although there are some formal methods to determine which features to include, most of the time you’ll probably just pick out what sounds good (doesn’t that sound like most risk assessment methodologies?) but then verify how the model performs.

. Therefore the categories were “infected” and “normal”. Another example may be the types of assets on your network. Perhaps the categories there would be user devices, servers and network devices and you’d like to classify the handful of unknown systems that are on the network into one of those categories.

Examples would be determining if something is infected with malware or not, perhaps you want to classify systems on the network as user desktops, servers or networking equipment.

In order to classify something, It is usually a short list of defined directory server, the approach is similar. Begin with a set of features which are the data we are collecting, may also be called independent or explanatory variables among other things.

, often called the features, perhaps Given a set of attributes about refered to as features within machine learning (or independent or explanatory variables)

independent or explanatory variable

We can look at what’s in th

It’d be horribly inefficient to go back to the old way of looking for malware and

test is the algorithms predictive accuracy. Can you think of any way to do that? One way would be to run the algorithm on all the data we know about and then start looking at the unknown systems while also looking for malware in the slower method we have. That would work, but it’s horribly inefficient.

Most applications in statistics have methods to determine how trustworthy the model or algorithm performs, and machine learning is no different. IWhat’s especially nice about machine learning is the methods for estimating p

See? Now isn’t that simp’ll have to give it some logic to decide which one is which also. very simple algorithm that simply look ats where the unknown host is on the

the title, creating a machine learning algorithm is just a fancy way to say “create some a set of steps for the computer to follow”. Anytime you see the word “algorithm” you should mentally just substitute “

I’m not sure about you, but I didn’t think I have ever generated an algorithm in y life. easy to get all wrapped up in generating an algorithm, but just think of

been described as programming the computer to program itself or teaching the computer to learn from data itself. It can be overwhelming.

This is an impressive challenge seeing as how we still can’t get most developers to validate their input.

If you’ve never heard of machine learning before Machine learning has some rather intimidating descriptions.

Classic statistical methods have deep roots within mathematics and most of the techniques were developed largely without the benefit of computers. The term “computer” originally referred to people who were employed to sit at a desk and calculate (using manual methods) all day long. Statisticians made enormous strides in learning from data in the human computer world and the processes they developed were

Generations of statisticians were taught using pencil and paper and wonderfully even though they there were massive strides during the first half of the twentieth century Statistics evolved generations of statisticians developed their complex computations using pencil and paper Even as computers became ubiquitous, classical trained statisticians simply used computers to automate their manual methods. They saw were leveraged as a way to automate and speed up the methods developed for pencil and paper.

But there was a small faction of computer users that started to use the massive increase in computational power to develop new methods of data analysis. Many of them were trained engineers and wouldn’t be considered statisticians. these analytic methods have evolved not just because of the increase in processing power. The variety of data was evolving, the quantity of data being collected was exploding and the problems faced were evolving as well. New methods were needed and t

But computers also brought about larger sample sizes. No more was data limited to what we could manually collect. Some of the classic methods began to break down in this new world

But it wasn’t just the processing power that evolved. the problems and data evolved as well.

Even as computers grew in popularity, statisticians for the most part, just migrated their manual methods used them to make applying the mathematics easier and faster.

The concept is quite simple, although a little intimidating; rather than

Every description and definition of machine learning makes the field sound quite intimidating. We

and directly out of some sci-fi plot. The evil genius creates a computer system Either we’re attempting to give the computer the ability to learn from it’s own

There are three types of people in the world. Those who have no idea what machine learning is, those who know machine learning is and that is has largely solved their spam problem and those who are completely intimidated by the concept of machine learning.

If you are reading this book, you probably have at least some exposure to the concept of “machine learning”. Most likely you’ve even some exposure to the term “machine learning”. You may have even read up on , If you haven’t heard much about the topic of machine learning, the

Context: a definition.

practical nad pragmatic

enabled by computers and processing.

machine learning: think of spam filtering. “wow that’s really hard” - demystifying it.

Example - gmail automatically sorts incoming mail.

examples in IDS, tipping point uses some ML to figure to stuff out

We will want to cover:

* model selection, cross validation, bootstrapping and train/test data
* prediction versus inference
* parametric versus not parametric
* supervised versus unsupervised

Trying to cover machine learning in one chapter is like a grown man trying to slide into a newborn’s onesie, some things just aren’t going to fit. We will talk through what it is, and so on. But there is a pretty big difference between applying machine learning and applying machine learning well.

* What’s possible with machine learning?
  + quantitative prediction and inference about variables
  + classification and clustering
* Side-bar (perhaps): parametric versus non-parametric
* What can we (in infosec) do with ML?

What is Machine Learning?

Machine learning has the ability to overpower even the quickest learner, so we will be approaching this at a high level.

If you believe the hype, machine learning is a miracle solution that will cure all our uncertainty if we can either figure out the right levers to flip or hire someone who knows. The premise is simple enough; we feed all the data into a computer and have it “learn” all about that data. Then we can sit down with a cup of coffee and have the machine share its wisdom with us… right? We need to start out by putting that hype aside.

. . restore us to a full head of hair.

a beautifully complex collection of seemingly mystical incantations that it able to extract knowledge and truth from the data no matter how messy. Unfortunately, nothing could be further from the truth. Machine learning is for the next evolution of hackers.

We’ve mentioned this concept of classical statistics. Classic statistics was developed prior to computers. The goal was to apply mathematics in order to help people understand the world from the data. Then as computers evolved, their natural statistical use to speed up the labor intensive processes of applied mathematics. But the purpose was still to help people understand the data.

Some definitions of machine learning make the field sound rather intimidating to the uninitiated. Hearing about “neural networks” and attempting to program a computer to program itself may seem a bit daunting, but honestly,

Machine learning has a good solid foundation in computer science with a dash of statistics. This is a benefit because there is a huge amount of practicality within machine learning and we want to focus on, and be able to distinguish, things that work from things that don’t work.

Supervised: start with data, and make two data sets, train and test, go through feature selection (and probably iterate here). go through various models/algorithms and see how we do

“continues to improve as the amount of data increases” (and as your experience with those algorithms improves)”

Need: Algorithms, EC2 and data

feature selection is more of an artistic process. - what is interesting about the data? what is relevant?

Why Machine Learning?

The truth about machine learning and statistics is that we do what works. so the real trick isn’t just learning how to do statistics, the real trick is learning how to tell the difference between a helpful or misleading solutions. Most of the time, we have to be content with more of an educated guess that it’s better than alternative methods.

I was reading an explanation of causation versus correlation and a weight loss study that found correlation between weight loss and three very clean attributes. People weighed themselves every day, they started each day with a good breakfast and they had the ability to exercise at home.

Machine learning can offer several benefits:

**Quantitative prediction**: This concept may make people with a strong engineering background a bit uneasy thinking that prediction is impossible. But relax, nobody is claiming that the precise future is hidden in the data, however you can use the data to make a pretty good estimate. Given a set of observations and the outcome that resulted, we can build predictive models make estimates for known future states. Think back to the linear regression analysis we did in Chapter 5. If by some strange turn of events another state appears with 6 million people, that regression analysis using just population would predict just under 5,000 Zero Access infections in that state. While that example isn’t exactly practical, you could use the technique to estimate bandwidth usage next month, or even forecast the size of the next DDoS attack.

While quantitative prediction is great when you want to estimate the value of something, much of real life is spent trying to figure out what something is. This is where **classification** comes in. For example, you see valid logins but are they authorized or not?

Classification techniques are supervised techniques.

In other words, given information about something a set of possible categorwe want to assign a label to something or classify sClassification is used to So given a But we can’t just magically know Given a set of observations about known things, can we Classification methods are developed much of the work over the past few decades in Machine Learning Many recent trends means you’ll want a method to classify

Much of the supervised learning algorithms focus on problems with classification and it’s the work horse of machine learning. Often times you won’t want to know the *value* of something, but if it *is* something. You’ve collected data about all the logons, are they all from the people they are claiming to be? Are all of these valid logons the person they claim to be? Is this machine what it claims to be?

**inference**: While prediction and classification are helpful, sometimes you’ll just want to make some inferences about the phenomena you’re observing. more about the variables and the envi

This is really hwere we see a lot of benefit from regression models.

discovery and exploration:

1 2 3 4 5 6 7

0123456789012345678901234567890123456789012345678901234567890123456789012345678

**This is the red font**

**var <- c(“This is a test of the code”)**

define algorithm (algorithm versus model)

You’ll hear the word “algorithm” tossed around quite a bit with machine learning. This term may be intimidating, but don’t let it be. Algorithms sounds like they may be confusing and complex, but they are really just a series of steps to take on the data. Algorithms do not have to be fancy and are really just Don’t be intimidated by this because it’s just a fancy way to talk about a summary of steps. The slight difference is that there are actually two algorithms, the algorithm you will tell the computer and the algorithm the computer will build based on the data it is fed. It may get a little confusing as rarely is there any distinction between the two.

partial quotient

long division

Supervised Learning

There are two main classes of machine learning algorithms: supervised and unsupervised. Supervised learning is used when you have known data that can guide (or supervise) the algorithm creation. For example, if you would like to use supervised learning to tell the difference between a system infected with malware and a system acting normally, you would need data from systems in those known states.

You would have have to collect when a system is infected with malware, you would need to have data from systems known to be infected a system with malware. We could approach this with supervised learning if we had data from systems we know were infected with malware and system not infected.

classification

Machine learning will bring about a shift in thinking in the security industry away from absolutes and the outdated concepts of rule sets and signatures and a shift towards indicators, rumors and probability. Gone are the days of thresholds and simple regular expressions.

1. <http://whatis.techtarget.com/definition/machine-learning> retrieved 10/5/2013 [↑](#footnote-ref-1)
2. Please keep in mind that this is a contrived demonstration of a machine learning approach, for a much better application of machine learning for malware detection, see Bilge, Leyla, et al. [↑](#footnote-ref-2)