#h1 Chapter 7 - ML Algorithms - Prior Considerations

#h2 Setting Expectations

#h3 7.1 - Introduction

#pg For someone new to machine learning, the amount of algorithm name-dropping in the field will be daunting. Some are supervised, and some are not (we will get to the difference between those two concepts soon). Others use images as input, whereas others work with text solely. But this grouping is a bit of an illusion. Indeed, we typically design algorithms to work on a specific type of data. However, the new generation of neural networks primarily provides a multipurpose framework for many different algorithmic objectives of interest. I use the term 'algorithmic objective' to refer to a natural grouping of the commonly used algorithms and the machine learning problems they solve. In most textbooks, people refer to these different objectives as different classes of machine learning algorithms. That is not technically incorrect, just a little nondescript. Besides, we can apply other natural groupings to machine learning algorithms. We have already mentioned supervised/non-supervised and input data types. Different dimensions along which algorithms can vary are complexity, whether they are strictly linear or allow for nonlinear relationships. Here, we want to focus on four high-level groupings that we feel should guide our decision-making process early in the process of choosing the appropriate machine learning algorithm

#im ../assets/figures/007/007-01.png 50 256 Figure 7.1 - Choosing the correct type of algorithm is not easy

#h3 7.2 - Supervised vs. Unsupervised - The amount of ground truth

#pg The most crucial distinction between machine learning algorithms is whether the algorithms expect some ground truth to learn against (supervised learning) or whether the algorithm needs to discover patterns in the data based on certain statistical or theoretical assumptions (unsupervised learning). It is usually clear whether you have some ground truth data early in building a machine learning solution. If you do, ensure that you have enough of it, that it is (largely) independent between data points, and is an appropriate and unbiased sample of the actual data your algorithm might encounter in the wild. If your data has no ground truth data, but you still expect some structure of interest in the data you do have, unsupervised machine learning algorithms are the way forward. In this scenario, spend some time thinking through the type of structure you expect to find in your data. Do the points cluster in some meaningful way? Do they organize themselves along some subspace of the original data space? Even something that appears as clear cut and dichotomous as supervised versus unsupervised learning has some middle ground, called reinforcement learning. This approach is because sense that the machine learning algorithm receives feedback on its current performance, but this feedback is less descriptive. For example, in regression (a supervised technique), we are given the discrepancy or error between prediction and actual observed values. In reinforcement learning, the feedback might be qualitative: the forecast was adequate or inadequate, correct, incorrect, helpful, or hurtful. We can use reinforcement learning when we simply do not know the true quantitative difference between predicted and actual outcomes. In addition, quantitative and imprecise feedback is common in biological adaptation and learning. It offers machine learning models greater freedom to find a more diverse set of (near) optimal solutions.

#h3 7.3 - Categorical, numerical, or something else altogether

#pg A common theme in machine learning is whether our data is strictly numerical or categorical. Our data is usually mixed, and things are not always clear-cut (a theme we will discuss in greater detail when we discuss scales. Most conventional algorithms still perform best with one type of data. For example, decision trees prefer categorical data, whereas regression methods will behave optimally with numerical data. However, clever recoding of the input data from categorical to numerical or vice versa, as well as minor tweaks to the algorithms, can make them more general. Whatever the driving force is, humans have a need and desire to divide things into categories, even though objects often fall along a continuum between two or more extremes.

#im ../assets/figures/007/007-02.png 50 256 Figure 7.2 - Amazon tribe members have the same three distinct cones that underlie human color vision, so in theory, they perceive color like all other humans. However, they have fewer names for them. In the western world, we use yellow, orange, red, purple, pink, blue, cyan, and green, but the tribe members only have a few proper color names. Even more interesting, their color categories do not overlap with ours. Evolutionary psychology suggests that the tribe members' primarily green and brown environment did not require them to have as many distinct categories of color as humans in settings where the color palette is more diverse.

#br

#pg However, as we have seen in the previous chapter, in some instances, our data is 1) not numerical or categorical in the strict sense, or 2) the data looks numerical but represents some digitized version of a more complex input pattern. The most obvious example of data being neither numerical nor categorical is text. Our inputs are letters, words, sentences, paragraphs, and complete books here. We will need to find some way to transform these words into quantities that allows us to operate on them numerically in a way that preserves something about the original form (i.e., word) that is informative for the task at hand. We shall see later, in the chapter explicitly addressing natural language processing, that there are different approaches to this transformation, depending on our use case. Sometimes, we can discard the word completely, noting and keeping track of some identifier so we can detect all occurrences of the same word. For other use cases, we can aim to create a numerical representation of each word that captures its similarity to other words. For example, in a Word2Vec (or word to vector) transform, each word is assigned a vector of n elements, such that two words we deem to be similar in meaning will be assigned vectors that are similar and distinct from vectors assigned to unrelated words. When we face the necessity of digitizing inputs, we should take care in digitizing our inputs that preserve the signal of interest. For example, you can record some audio, which on modern systems is likely to be sampled at some high frequency, say 44100Hz. That means a vector of 44100 elements represents a human vocalization of one second! However, human speech does not even get near 44100Hz in pitch frequency, not even the Bee Gees. So downsampling the audio to a more workable number of samples is warranted. In speech recognition, research typically goes even further. Rather than representing the audio waveform as a series of samples, they chop up the time series into known atomic units of human speech, dubbed phonemes. There are only a small number of different phonemes humans use, far less than the original 44100 samples. But these changes the nature of the data to some respect. Short phonemes are more like categorical data, whereas the original and resampled waveforms vectors are numerical. Thus, the machine learning algorithm used still needs to fit the bill, given our choice of representation.

#im ../assets/figures/007/007-03.png 50 256 Figure 7.3 - The waveform, spectrogram, and transcription of the word "Wikipedia" pronounced by an American English speaker. A spectrogram is a 2D dimensional plot where we compute the amplitude of each frequency over time. We can decompose this spectrogram into segments, each containing a single phoneme for human speech.

#h3 7.4 - Black box versus open box

#pg The general-purpose machinery we get from neural networks comes at a price. The network takes on multiple objectives, optimizing its weights to find the optimal transformation between input and output. These transformations become highly nonlinear and complex with an increasing number of nodes, edges, and layers. At some point, we will have a tough time understanding what the nodes in deeper layers are encoding. This lack of understanding is particularly pervasive in one type of architecture: convolutional neural networks. When we train these networks on copious amounts of image data, we will find that the early layers can be seen as image filters: detecting lines, edges, and gradients. However, the deeper we dive into the net, looking at what activates the nodes deep into the hidden layer, the harder it becomes to deduce meaning from the activation pattern of even a single node to many stimuli. A few years ago, the community was startled by a strange phenomenon. Most modern image classification algorithms correctly predict the label when presented with an image of an elephant. However, the networks changed their minds when we added a tiny amount of noise so small a human observer could not even tell. They sometimes predicted vastly different things only because we added a little noise. Biological systems are immune to such a small amount of noise. Humans could not even tell there was noise added! We must admit that we do not fully understand what has been learned, why it can so easily fail in the presence of noise, and that what we have built is not entirely analogous to biological vision. A black box, indeed. Neural networks becoming black boxes creates an issue for any governance on their performance and potential bias and limits our proper understanding of their inner workings. In stark contrast, linear methods, like linear regression, are easy to understand. Their prediction is no more than a linear weighted sum of all input signals. Such prediction allows us to make factual statements such as the risk of cancer increases positively related to BMI, where every single point increase in a person's BMI is mirrored by about a 5% increase in the risk of developing a specific cancer. These algorithms are limited in what they can explain and might overlook critical predictive sources of input signals simply because they do not fit our simple assumptions.

#im ../assets/figures/007/007-04.png 50 256 Figure 7.4 - Be wary of black boxes.

#h3 7.5 - Deterministic versus Probabilistic

#pg In most cases, it would be preferable if there is one and only one optimal set of parameters for predicting outputs from the input. This unique set of parameters would make our model deterministic: the model will always be the same after training, no matter how often we train it if the input data does not change. Linear regression can be deterministic if we use the closed-from implementation. Decision trees are also typically deterministic, although techniques like probabilistic pruning can introduce some randomness. However, most machine learning algorithms rely on iterative methods and use weight randomization techniques before starting a training regime. Random initial weights acknowledge the possible existence of local optima: place in the state space of parameters whether the model performs well, compared to any slight variations in its parameters. However, that is not to say that beyond this valley, there is not yet another deeper valley where the algorithm performance is even better. Local optima are a result of non-linearities in the models by design. The main workhorse of machine learning is the gradient descent algorithm, which is susceptible to such local optima. Many optimizations have been introduced to prevent algorithms from settling in these locally but not globally optimal minima.

#h3 7.6 - Conclusions

#pg Algorithms can vary along many other dimensions, such as model complexity, ease of implementation, and algorithmic complexity. Consider such differences when deciding which machine learning algorithm to use. As a rule of thumb, start with an algorithm that is easy to implement, debug, and easy to understand once learned. It might not be the most optimal of algorithms, but it provides solid sanity checks of both model and data and will guide your decision-making in adopting more complex variants.

#h3 7.7 - References

#bs

#be

#bp Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. United Kingdom: MIT Press.

#bp Newman, M. (2010). Networks: An Introduction. United Kingdom: OUP Oxford.

#bp Gegenfurtner, K and Sharpe, L. (2001). Color Vision: From Genes to Perception. United Kingdom: Cambridge University Press.

#be