#h1 Chapter 8 - ML Algorithms - Algorithmic Objectives

#h2 Setting the stage

#h3 8.1 - Introduction

#pg In the previous chapter, we discussed how our choice of machine learning algorithm depends mainly on our model's input data. Is our data labeled, and if so, in what way? Is our data purely categorical or purely numerical? Or perhaps neither? For data that is numerical, was there a process of digitization that transformed the data from some raw format of direct measurements into some other more compact, abstract, or semantic form? In addition, we acknowledged that models could vary significantly in their complexity, with more complex models requiring more training data and computational resources (time and memory). And although the increase in complexity allows these models to outperform more conventional models, there is a risk of the complexity being too great. In this case, we will lose the ability to fully grasp what the model has learned and how it applies that knowledge to new inputs. These considerations are essential, regardless of what we ultimately want our machine learning models to do. You might think we should have started with that question rather than the one we discussed in chapter 7, but there is some logic in this order, trust me. In chapter 7, we discussed choices driven by the interaction between the input and design of our machine learning algorithms. Here, we focus on the interplay between our model's design and output. What is ultimately the algorithmic objective of our machine learning model? A sorting algorithm sorts, Newton's method is an algorithm that finds the square root of a number. But a machine learning algorithm can implement many different objectives, depending on our needs. Are we asking our algorithm to predict a number? Like the estimated accrued medical expenses, given someone's age? Are we classifying individuals as either at high or low risk for cancer? Perhaps we would rather have a probability of this risk instead. Or find a natural but unknown grouping between individuals. We could ask a machine learning model to generate synthetic patients based on a large sample of real patients and the patterns they share or do not share. We could even find your digital twin(s), the people most like you, across several variables we deem essential and predictive. If your digital twin responded well to a particular treatment, perhaps you will as well. Most textbooks will refer to these different objectives as classes, and algorithms fall within those classes. I find that not a very helpful or descriptive term. Besides, there are other ways we can create different types of machine learning algorithms, some of which we have already discussed—like complex versus simple, supervised vs. unsupervised, or linear vs. non-linear. So, I prefer to call the grouping of the different types of outputs an algorithm can produce particular algorithmic objectives. I hope the term will stick.

#h3 8.2 - Regression

#pg In regression, we want to predict a continuous-valued outcome. Requiring a continuous-valued output does not require the input data to be numerical and real-valued. The success metric of regression is a metric of the discrepancy between observed and predicted outputs, commonly measured as the sum of squared errors. It allows us to measure the amount of variance in our data, the amount of that variance that our regression model explains, and the amount of model left unexplained. The more variance we can explain with our model without overfitting, the better.

#im ../assets/figures/008/008-01.png 50 256 Figure 8.1 - Regression aims to predict a real-valued outcome from input data.

#h3 8.3 - Classification

#pg In classification, we aim to predict class membership. Most classification algorithms not only return a predicted class membership but also some confidence metric, typically in the form of a probability. We design most algorithms to perform binary classification and only binary classification. When we have classification tasks with more than two classes, we must consider how to approach them. When using algorithms designed to do binary classification, we can opt for a one-vs-rest or one-vs-one approach, two methods we will discuss in Chapter 15 on classification. For neural networks, the number of classes does not matter as much as requiring an additional step. By their very design, the output of neural networks takes on any number or dimensionality. Network output can be just a single node indicating the presence or absence of something. Or perhaps thousands of nodes, where each node activity provides evidence for a unique class (e.g., image classification). Or even a set of nodes that together form an entire image output identical in size to those used as inputs to the network, like a class of neural networks used to realistically color in footage recorded in black and white. Compared to regression, classification problems seem to provide a more intuitive performance measure. We can just look at how often we predict the correct class. While it is true that a classification algorithm with low to no accuracy adds little to no value, high classification accuracies can be highly misleading. If a specific class is rare, we can get high accuracy by simply never predicting its presence. Sure, we get it wrong for all instances of that class, but we get it right for most examples labeled a different class. There are numerous ways to counteract these effects, which we shall discuss in greater detail later.

#im ../assets/figures/008/008-02.png 50 256 Figure 8.2 - Classification aims to predict from input data as one of n classes it has been trained on.

#h3 8.4 - Clustering

#pg On the surface, clustering appears like categorization. However, the main difference is that clustering does not depend on ground truth labeling. In classification, the data point in our training set comes with class labels. We turn to clustering when they do not. The objective of clustering is to assign labels to our data such that they minimize some predefined property of clustering. In some algorithms, we define this as having the distance between points belonging to the same cluster be smaller than the distance between points belonging to different clusters. Intuitively, this is a reasonable assumption, but there are many data sets for which the clustering is readily apparent without meeting this constraint. In addition, it makes us think carefully about how we define distance and what distance to consider. However, other clustering algorithms use a grid, density, or distribution-based methods to define what constitutes a cluster and what does not.

#im ../assets/figures/008/008-03.png 50 256 Figure 8.3 - Clustering. In an unsupervised algorithm, clustering algorithms try to find clusters within a dataset based on the assumption of what constitutes a cluster and what does not. Points can cluster based on distance, density, distribution, and grid-based metrics.

#h3 8.5 - Manifolds

#pg With some imagination, we can state what regression models are to classification models and manifold models are to clustering models. In both classification and clustering models, we rely on some natural grouping of points belonging to the same class. The only difference is that we know what points go together beforehand with classification. In clustering, we need to derive this from the statistical properties (distance, density, etc.) of the individual data points relative to each other. However, in both cases, we hope that data points belong to some (known/unknown) grouping from a subspace in a large space of all possible combinations of input signals. All bees are small insects, have black and yellow wings, build hives, dance, and gather nectar for their colony. Suppose we have a sizeable N-dimensional space in which all dimensions are imaginable animal characteristics, such as its size, shape, set of behaviors, life span, and reproduction method. In that case, we expect all the different bees to occupy a little corner of that space, far away from elephants, Tasmanian devils, and portobello mushrooms. In regression and manifold learning, we are looking for a continuum along which points can vary. For example, we can measure all distinct facial features, like yaw angle, curvature, eye size, nose width, lips thickness, etc. But these can all vary continuously. Moreover, there is probably some direction in this N-dimensional space of features along which human observers will see an ordering between very female-looking faces, through gender-neutral looking faces, towards very stereotypically male-looking faces. We aim to find this axis using regression, how male/female is a particular face, given the values that serve as input: yaw angle, eye size, nose width, etc. Furthermore, in regression, we are specifically looking for this axis, as it is the output. We ask our algorithm to predict from a set of inputs. Manifold learning is similar because it also looks for axes along which our input data varies smoothly. Unlike regression, however, it isn't necessarily looking for a single direction. It can use as many directions through the original space as needed to organize all available data such that when we move along the surface formed by these directions (i.e., the manifold), we see a smooth transition between input data. Trying to imagine this for the first time is perhaps a little daunting.

#im ../assets/figures/008/008-04.png 50 256 Figure 8.4 - Manifolds. Sometimes our data is best described as a lower-dimensional subspace of the original data space. This brings together ideas we have seen in clustering, regression, and classification. All four objectives and their geometric interpretation are based on the notion that the relative position of data points in the data space is not random but structured. Data points organize themselves along a set of weighted axes that form a subspace of the original data space in which the points were defined. In clustering and classification, we think similar data points will find themselves closer, more connected, and within the area of higher probability densities. In regression, we assume that points align themselves with a weighted combination of the dimensions that describe them. In manifolds, we presume both aspects of being true.

#br

#pg We will discuss manifolds in greater detail later, but for now, the following example and illustration might provide an initial intuition. Imagine a pixel in a photograph. The standard way of representing the pixel will look by describing the percentage of red, blue, and green light it emits. We thus have a space of three dimensions, along our data varies. (0,0,0) represent black, (1,1,1) is total white and (1,0,0) is red. All other states (I am not saying color on purpose) are in that space. Notice that we have many reds, greens, yellow, and purples. Or at least things we would classify as those colors. We also can see that the points vary in intensity. We have dark blue, bright red, pale blues, and dark reds. It turns out that intensity or luminance is different than the color bit (which we refer to as 'hue'). Furthermore, not all hues of the same intensity are the same; they can also vary in saturation. It turns out we can find a 2D manifold that explains the relationship between hue and saturation but discards the effect of luminance. It is like we collapse the dimension of luminance to be 0. Our new 2D manifold lives in a subspace of the 3D pixel space that has two directions, one along which we see green turn into red through gray, and one in which we see purple turn into yellowish, also through gray. Those two dimensions tell us everything about hue and saturation that we need to know outside of luminance.

#h3 8.6 - Vectorization

#pg Vectorization is a helpful technique for turning categorical variables or other complex, non-numerical entities into numerical quantities that can be compared and operated on using mathematics. Conceptually it combines a few of the approaches we shall encounter later, so we will be brief in describing it here. In short, we typically define some metric of relatedness between class instances. For example, we might observe the words' cell', 'white', and 'blood' to occur quite frequently in the same paragraph of text. In contrast, the words' cell' and burger' are rarely observed together. By computing the probabilities for each word combination (cell and burger), we create a graph where the nodes represent different words. The links indicate the likelihood of observing them in the same document. We could stop here and vectorize each word as the vector of all its probabilities to other words. Such vectorization is sound in principle, just not very practical: 95% of all written English consists of around 10,000 different words. Therefore, we typically apply a dimensionality reduction approach to the graph, combining closely related words. Dimensionality reduction turns a graph of words into a graph of concepts, where we might be built one concept from the words (cancer, cell, blood, white, and research). Another concept might consist of the words (white, castle, burger, and king). We can now represent any word as a relative mixture of the membership to these two concepts.

#h3 8.7 - Forecasting

#pg Forecasting uses time series data to predict the dynamics for a period in the future. A few forecasting techniques exist, but all rely on fitting the observed data to a parameterized function of some complexity. We can use this function to interpolate and extrapolate to new, previously unobserved values. Interpolation fills in the gap between measurements, and extrapolation predicts data outside of our measurement domain.

#im ../assets/figures/008/008-05.png 50 256 Figure 8.5 - Forecasting is a particular case of regression. Suppose we find the correct equation to predict the fluctuations of a time series in the past. In that case, we can argue that we can extrapolate our function beyond the most recent data point and predict what future fluctuations might look like (with some margin of error).

#h3 8.8 - Link Prediction

#pg Link prediction allows us the estimate the probability of observing a previously unobserved link between two nodes within a graph. These nodes can represent anything entity of interest, like people, words, concepts, or images. Similarly, the links represent any relationship we can define between the entities, like friendship, co-occurrences, cause, effects, and similarity. In most cases, the classification is binary, yielding a probability associated with the link's existence or not. We can also predict the weight or strength of the predicted links with minor tweaks, should the observed links be weighted. Training a link prediction algorithm can be done utilizing vectorization (see above), using data consisting of positive examples (links already observed in the data) and negative examples (links not observed in the data).

#im ../assets/figures/008/008-06.png 50 256 Figure 8.6 - Link Prediction. If you have been on Facebook, you will have seen a use case for link prediction. Facebook suggests new friends based on the probability that there should have been a link between another person and me, given our mutual friends. But we can use it for less commercial purposes as well. Link prediction can help find connections between disparate scientific results that scientists have overlooked. And in biology, link prediction helps find interactions between proteins that can guide drug development.

#h3 8.9 - Recommendation

#pg Recommendation underlies many consumer services such as Netflix, Amazon, and Spotify. They use many so-called recommendation engines, each specializing in a domain or operating under different constraints. For example, Netflix might deploy a recommendation for a specific demographic, but it is not that meticulous about what it recommends. In contrast, Amazon might have a recommendation engine for high-end cosmetics that only recommends products to users if they have a high enough confidence level. Two main algorithms form the basis for most recommendation engines: matrix factorization and collaborative filtering. Both assume we start with a matrix of items by the user with each i,j element in the matrix representing the interaction between specific user i and specific item j. We use the word interaction because we can use several metrics here. Has the user viewed this item? Purchased this item? Has the user rated the item in some way? Did the user leave a review of the item? If so, what was the sentiment expressed in the review? Naturally, this matrix is sparse; most user have not interacted with most items. For example, there is a whole section of Korean teen movies I have never seen or rated on Netflix. Recommendation engines aim to fill these unknowns by comparing users to other users and items to other items. If two users show considerable overlap in the products they buy, it is reasonable to assume that a product purchased only b one user is relevant to the other user.

#h3 8.10 - Anomaly Detection

#pg Anomaly detection, by definition, is an unsupervised machine learning problem. Rather than being shown what constitutes an anomaly and what does not, we rely on detecting such data points because they depart from certain expectations about our data. In most cases, anomaly detection is outlier detection: associating a data point with a probability of observing it. If this probability is too low, flag it as an outlier or anomaly. Assigning a probability to observe a particular value is a statistical problem. The most common approach is to estimate the parameters of the underlying distribution from the entire data set. For example, it can fit most continuous signals to conform to the normal or gaussian distribution, parameterized by a. mean and standard deviation. With those estimates, we can now compute the probability of each data point under those assumptions. The further the observed data point is from the mean, the lower the probability of observing it. In engineering, this principle is reflected in the six-sigma rule: the idea that you do not have to worry about failures that have a probability of 6 standard deviations away from the mean. Because those points are genuinely anomalies: they have a 0.03% or less chance of occurring.

#im ../assets/figures/008/008-07.png 50 256 Figure 8.7 - Anomaly Detection. Detecting anomalies means assigning a probability to an event or measurement that can no longer be seen as nominal. We could look at observations that fall outside of some acceptable range. Still, more advanced anomaly detection algorithms will have a more detailed knowledge of the statistical processes in generating the observed data.

#h3 8.11 – Generative Models

#pg More recently, algorithms have become powerful enough to create rather than predict. A whole new class of algorithms can now generate content ranging from text, speech, music, and art to deep fakes. Deep fakes are worrisome, as this technique can generate audiovisual content that is already particularly hard to discern from actual footage. Presidents can be made to appear to be giving speeches that never happened, celebrities can be made to appear to endorse products they have never even heard of, and politicians can be made to express viewpoints that completely counter their ideologies. Speech synthesis has been around a bit longer, and with solutions like Google Assistant, it is safe to say that we all have already been fooled into thinking we were talking on the phone to an actual human being. Less nefarious, in some way, are the generative models that seem capable of capturing a particular style of an artist or art movement and applying that to an image of our choosing. Do you want van Gogh to paint a picture of your apartment building? No problem, machine learning has you covered. We will play with this later in the course.

#im ../assets/figures/008/008-08.png 50 256 Figure 8.8 - Generative models can create and synthesize new data with spectacular fidelity. The above 3D models are not based on any real human being. They are entirely computer generated.

#h3 8.12 - References

#bs

#be

#bp Norvig, P., Russell, S. J. (2016). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

#bp Wallisch, P., Lusignan, M., Benayoun, M., Baker, T. I., Dickey, A. S., & Hatzopoulos, N. (2014). MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB. Academic Press.

#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.

#be