#h1 Chapter 11 - The Geometry of Machine Learning

#h2 The Space of Data

#h3 11.1 - Introduction

#pg Machine learning is an advanced topic. Building successful algorithms requires a decent understanding of mathematics, statistics, probability theory, and computer science and a healthy dose of common sense. This chapter is not an exhaustive list of pseudo code and formulas. Instead, we aim to use it to lay the foundation for a working intuition of how machine learning algorithms work 'under the hood' and achieve their incredible feats of prediction (and deception). As stated before, my approach to machine learning has always been through the perspective of geometry. The closeness between data points conveys similarity; boundaries signal the edge between different classes; we can imagine the transformations and projections as revealing new insights into the regularities we can utilize by the algorithms we build. Imagining how input data might be organized and distributed in its state space helps me understand its structure and how to approach it to build a successful machine learning algorithm. For those familiar with linear algebra, feel free to skip ahead. This chapter will primarily cover matrix multiplication and transforms. For those new to these topics or new to their application to machine learning, stick around.

#h3 11.2 - Preamble

#pg Before we start our discussion on the geometry of our data, let us first go over some of the standard nomenclature used in machine learning as it pertains to data. Data science and machine learning borrow concepts from many fields, from statistics, mathematics, computer science, and experimental science in general. Therefore, there are many ways to refer to the same thing. For example, in empirical science, we test the effect of a controlled manipulation of a set of independent variables on the behavior of a dependent variable. In statistics and econometrics, we occasionally also refer to variables as regressors. In machine learning, these concepts map neatly onto inputs (independent variables or regressors) and outputs (dependent variables) of a machine learning algorithm. As we will see later in this chapter, the inputs to a machine learning model are typically multidimensional. Multidimensional in this context means that we have m dimensions for each of the m number of independent variables/regressors, with each dimension represented by a column in our input data matrix.

#h3 11.3 - Points

#pg Before we start thinking about entire data sets, with many examples and dimensions, let us begin by considering the atomic element of space: the point. At the most fundamental level, our data consists of one-dimensional points. And each point represents measured or observed values of an attribute of interest. For example, a patient has an age, an event might have a timestamp, and a disease could be present or absent. We strive for these different points to be as independent as possible to avoid over-specifying our models. For example, if we include speed in mph, we no longer need an additional value representing speed, but in km/h. This example is perhaps a little on the nose, but sometimes these dependencies are more subtle. For example, the body mass index (BMI) is computed as a person's weight (in kg), divided by their height (in meters) squared. In some sense, BMI is redundant if we have included height and weight as variables. But since BMI is a nonlinear derivative of height and weight, it is not redundant for linear models. In general, correlations between input variables can wreak havoc on our models by introducing statistical ambiguity, ill-conditioned matrices, and unreliable models.

#br

#im ../assets/equations/011/png/011-01.png 50 16 Equation 11.1 - A single point x, taking on a value of 1

#h3 11.4 - Vectors

#pg Vectors are an ordered list or collection of measurements. Note that vector size and length are two different things. Vector size reflects the number of elements in the collection (n=2), whereas vector length is the distance between the endpoint of the vector and the origin.

#br

#im ../assets/equations/011/png/011-02.png 50 32 Equation 11.2 - A vector x, of size 2 and length 1

#h3 11.5 - Matrices

#pg Vectors combine into the core data structure used in machine learning: the matrix. A matrix is a logical extension of a vector, which itself is a logical extension of a point. We describe a matrix by the number of rows n and columns m. At each cell, we find a measurement or observed value indexed by the position along the i-th row and the j-th column. And for mathematical convenience, a vector is a matrix with n rows (or columns) and just 1 column (or rows). Row vectors represent all measurements belonging to the same measurement, for example, datapoint. For example, each row represents a unique patient in a dataset containing demographic data. The values of the vector capture the different measurements or observables for that patient (age, race, height, etc.). For an image data set, each row consists of a single image, in which each vector element represents a pixel value. For an NLP (Natural Language Processing) data set, each row can encode a document, with each vector element representing the presence of a particular word in that document.

#br

#im ../assets/equations/011/png/011-03.png 75 128 Equation 11.3 - matrix X, with 4 rows and 2 columns

#br

#pg In contrast, the column vectors represent the values across examples for a single measurement. For example, a column vector of age is a vector containing an age for each data point in our patient demographic data set. Or it represents the gray values at an (x, y) coordinate across all images or the presence or absence of the word 'technology' in each document in our NLP data set. Once our data takes on this numerical form in which we neatly define our inputs and outputs as matrices, the same algebraic principles and methods apply, regardless of what data type gave rise to it. The next chapter will introduce essential techniques embedded in many machine learning algorithms, from linear regression to deep belief neural networks.

#h3 11.6 - The correct number of rows and columns

#pg We often assess the usefulness of a data set by its n, or the number of unique rows/data points. The larger the n, the better. But we should take caution in what constitutes a single data point. Imagine training a model to forecast the Dow Jones index from the fluctuations we have observed in the past. So, we chop up the time series of length q into smaller segments of length p, randomly taking a bit of data between two points from the entire time series. That would give you as many training examples as you can take p different segments from a time series with length q. While this is true, the problem with this approach is that the training examples are not independent. Even when they do not overlap, the values of one segment are likely to be a function of all earlier segments. The correct number of column vectors to include in your machine learning model is not always set in stone, but some general principles apply. First, each column vector included in the input data is a potential additional source of predictive power. If you do not include age in a model of cancer risk, your model cannot use this vital piece of input information and therefore is likely to underperform. However, each additional column vector also adds a degree of freedom. Additional degrees of freedom allows for a greater opportunity for our machine learning model to explain the variance, having more regressors to account for it. If the regressors are indeed predictive, this is a desired property. But it also can artificially inflate how well our model is doing if we do not validate that with some additional data. With many regressors, little idiosyncratic variations in our training data might be accounted for by these additional regressors. But those variations only exist in our training data and are not representative of the data the model will encounter in production. Unrepresentative training data leads to the problem of overfitting. While your model might look like it is improving during training, it is deteriorating when tested on new data not used during training. There is a general rule or heuristic that data scientists use: ideally, the number of data points is at least 2, as large as the number of dimensions or vectors. Otherwise, the mapping becomes an ill-conditioned problem, the algebraic version of trying to derive meaning where there is none. We denote this as the m << n rule (m should be smaller than).

#h3 11.7 - Formalizing machine learning

#pg After collecting data, we find ourselves in possession of matrix A with n number of rows of n examples and m number of columns of regressions, variables, or dimensions. The first step in building a machine algorithm is to divide matrix A into a set of input variables and output variables (the columns of A). In machine learning, we typically consider most column vectors to be inputs of our machine learning algorithms, and one vector represents the output or the thing we are trying to predict. We typically refer to this new matrix of inputs as X, and the vector of outputs as y. In formal mathematical terms, this then defines a machine learning model where our predicted outcome y is a function of input X, parametrized by θ. Machine learning aims to find the parameters \theta, our input data X, and our function f (the machine learning model itself) to best predict y.

#br

#im ../assets/equations/011/png/011-04.png 50 32 Equation 11.4 - Machine Learning, formalized as function f operating on data X, and parameters theta to produce predicted output y.

#h3 11.8 - The Geometry of Machine Learning

#pg Now that we have formalized the structure of our data, we can begin to describe some of the fundamental mathematical concepts that underlie most, if not all machine learning. With these concepts, we form an intuition on how various machine algorithms utilize the geometry of data to learn optimal parameters, safeguard against outliers and differences between data sources and types, predict outcomes from input and define success metrics around these predictions. The key idea behind this geometric interpretation is that our data can be considered a set of points occupying an N-dimensional space. The relative position of individual data points creates a structure that machine learning models can exploit. What we mean by relative structure varies on the machine learning task at hand. As we have seen in the previous module, we can classify these tasks by the types of mappings we are trying to approximate. In regression, we assume that the points fall along some parameterized function explaining a reasonable amount of the variance observed. In contrast, in classification, we assume that we can find a hyperplane (a plane extending in n dimensions) that creates a clear decision boundary between points belonging to different classes. In clustering, we seek to find a naturally occurring grouping of data points, as measured by within and between group distances of points. Finally, in manifold learning, we assume that points fall upon a hypersurface or manifold representing a subset of the original space along which our data varies smoothly in some aspect.

#h3 11.9 - Conclusion

#pg It is hard, if not impossible, to imagine or visualize anything beyond 3 dimensions. Yet we can still argue that the geometric point of view allows for an intuitive understanding of what our algorithms optimize for. The geometric interpretation of Equation 11.4 is that we seek to find a specific mapping of our inputs onto outputs to minimize our prediction error or some other success metric. The function f dictates the complexity of this mapping, and the weights θ specify one mapping out of an infinite number of mappings allowed by the complexity of f that (hopefully) provides the most optimal of all mappings. We transform our data with a large degree of freedom for highly nonlinear algorithms such as deep belief neural networks. We can move, rotate, warp, twist, bend, flip, and curve our data. In contrast, linear algorithms, such as linear regression, allow only rigid body and affine transforms. This restriction limits how well a linear algorithm can be applied to a data set and how much it requires nonlinear mappings. We can best understand how machine algorithms learn and apply these mappings by examining the linear case in the next chapter.

#h3 11.10 - References

#bs

#be

#be

#bp Strang, G. (2021). Introduction to Linear Algebra (5th ed.). Wellesley-Cambridge Press.

#bp Shilov, G. E. (2012). Linear Algebra. Dover Publications.

#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 Thompson, D. W. (1992). Canto: On growth and form (J. T. Bonner, Ed.). Cambridge University Press.

#bp Ellenberg, J. (2022). Shape: The hidden geometry of information, biology, strategy, democracy, and everything else. Penguin.

#be