Machine Learning Basics
Part 1

Johar M. Ashfaque

Deep learning is a specific kind of machine learning. To understand deep learning well, one must have a solid understanding of the basic principles of machine learning. We begin with a definition of what a learning algorithm is. We then proceed to describe how the challenge of fitting the training data diers from the challenge of finding patterns that generalize to new data. Most machine learning algorithms have settings called hyperparameters, which must be determined outside the learning algorithm itself; we discuss how to set these using additional data. Machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving condence intervals around these functions; we therefore present the two central approaches to statistics: frequentist estimators and Bayesian inference. Most machine learning algorithms can be divided into the categories of supervised learning and unsupervised learning; we describe these categories and give some examples of simple learning algorithms from each category. Most deep learning algorithms are based on an optimization algorithm called stochastic gradient descent. We describe how to combine various algorithm components, such as an optimization algorithm, a cost function, a model, and a dataset, to build a machine learning algorithm.

I. LEARNING ALGORITHMS

A machine learning algorithm is an algorithm that is able to learn from data. But what do we mean by learning? Following Mitchell, we have the following definition: "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". One can imagine a wide variety of experiences E, tasks T, and performance measures P, and we do not attempt to formally define what may be used for each of these entities. Instead, in the following sections, we provide intuitive descriptions and examples of the different kinds of tasks, performance measures, and experiences that can be used to construct machine learning algorithms.

A. The Task: T

Machine learning enables us to tackle tasks that are too difficult to solve with fixed programs written and designed by us. From a scientific and philosophical point of view, machine learning is interesting because developing our understanding of it entails developing our understanding of the principles that underlie intelligence. In this relatively formal denition of the word "task", the process of learning itself is not the task. Learning is our means of attaining the ability to perform the task. For example, if we want a robot to be able to walk, then walking is the task. We could program the robot to learn to walk, or we could attempt to directly write a program that specifies how to walk manually. Machine learning tasks are usually described in terms of how the machine learning system should process an example. An example is a collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process. We typically represent an example as a vector $x \in \mathbb{R}^n$ where each entry x_i of the vector is another feature. For example, the features of an image are usually the values of the pixels in the image.

Many kinds of tasks can be solved with machine learning. Some of the most common machine learning tasks include the following:

- Classification: In this type of task, the computer program is asked to specify which of k categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \to \{1, ..., k\}$. When $y = f(\mathbf{x})$, the model assigns an input described by vector $\mathbf{x}$ to a category identied by numeric code y. There are other variants of the classification task, for example, where f outputs a probability distribution over classes. An example of a classification

task is object recognition, where the input is an image (usually described as a set of pixel brightness values), and the output is a numeric code identifying the object in the image.

- Regression: In this type of task, the computer program is asked to predict a numerical value given some input. To solve this task, the learning algorithm is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$. This type of task is similar to classification, except that the format of output is different. An example of a regression task is the prediction of the expected claim amount that an insured person will make (used to set insurance premiums), or the prediction of future prices of securities. These kinds of predictions are also used for algorithmic trading.
- Machine Translation: In a machine translation task, the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language. This is commonly applied to natural languages, such as translating from English to French. Deep learning has recently begun to have an important impact on this kind of task.
- Anomaly Detection: In this type of task, the computer program sifts through a set of events or objects and ags some of them as being unusual or atypical. An example of an anomaly detection task is credit card fraud detection. By modeling your purchasing habits, a credit card company can detect misuse of your cards. If a thief steals your credit card or credit card information, the thief's purchases will often come from a different probability distribution over purchase types than your own. The credit card company can prevent fraud by placing a hold on an account as soon as that card has been used for an uncharacteristic purchase.

Of course, many other tasks and types of tasks are possible. The types of tasks we list here are intended only to provide examples of what machine learning can do.

B. The Performance Measure: P

To evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance. Usually this performance measure P is specific to the task T being carried out by the system. For tasks such as classification, we often measure the accuracy of the model. Accuracy is just the proportion of examples for which the model produces the correct output. We can also obtain equivalent information by measuring the error rate, the proportion of examples for which the model produces an incorrect output. Usually we are interested in how well the machine learning algorithm performs on data that it has not seen before, since this determines how well it will work when deployed in the real world. We therefore evaluate these performance measures using a test set of data that is separate from the data used for training the machine learning system. The choice of performance measure may seem straightforward and objective, but it is often difficult to choose a performance measure that corresponds well to the desired behavior of the system.

C. The Experience: E

Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process:

- Unsupervised Learning Algorithms: experience a dataset containing many features, then learn useful properties of the structure of this dataset. In the context of deep learning, we usually want to learn the entire probability distribution that generated a dataset, whether explicitly or implicitly. Some other unsupervised learning algorithms perform other roles, like clustering, which consists of dividing the dataset into clusters of similar examples.
- Supervised Learning Algorithms: experience a dataset containing features, but each example is also associated with a label or target. For example, the Iris dataset is annotated with the species of each iris plant. A supervised learning algorithm can study the Iris dataset and learn to classify iris plants into three different species based on their measurements.

Roughly speaking, unsupervised learning involves observing several examples of a random vector $\mathbf{x}$ and attempting to implicitly or explicitly learn the probability distribution $p(\mathbf{x})$, or some interesting properties of that distribution; while supervised learning involves observing several examples of a random vector $\mathbf{x}$ and an associated value or vector $\mathbf{y}$, then learning to predict $\mathbf{y}$ from $\mathbf{x}$, usually by estimating $p(\mathbf{y}|\mathbf{x})$.

Unsupervised learning and supervised learning are not formally defined terms. The lines between them are often blurred. Many machine learning technologies can be used to perform both tasks. For example, the chain rule of probability states that for a vector $\mathbf{x} \in \mathbb{R}^n$, the joint distribution can be decomposed as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1}).$$

D. An Example: Linear Regression

Our denition of a machine learning algorithm as an algorithm that is capable of improving a computer program's performance at some task via experience is somewhat abstract. To make this more concrete, we present an example of a simple machine learning algorithm: linear regression. As the name implies, linear regression solves a regression problem. In other words, the goal is to build a system that can take a vector $\mathbf{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output. The output of linear regression is a linear function of the input. Let $\hat{y}$ be the value that our model predicts y should take on. We dene the output to be

$$\hat{y} = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of parameters. Parameters are values that control the behavior of the system. In this case, w_i is the coefficient that we multiply by feature x_i before summing up the contributions from all the features. We can think of $\mathbf{w}$ as a set of weights that determine how each feature affects the prediction. If a feature x_i receives a positive weight w_i, then increasing the value of that feature increases the value of our prediction $\hat{y}$. If a feature receives a negative weight, then increasing the value of that feature decreases the value of our prediction. If a feature's weight is large in magnitude, then it has a large effect on the prediction. If a feature's weight is zero, it has no effect on the prediction.

We thus have a denition of our task T: to predict y from $\mathbf{x}$ by outputting

$$\hat{y} = \mathbf{w}^T \mathbf{x}.$$

Next we need a definition of our performance measure, P. Suppose that we have a design matrix of m example inputs that we will not use for training, only for evaluating how well the model performs. We also have a vector of regression targets providing the correct value of y for each of these examples. Because this dataset will only be used for evaluation, we call it the test set. We refer to the design matrix of inputs as $\mathbf{x}^{(test)}$ and the vector of regression targets as $\mathbf{y}^{(test)}$.

II. ESTIMATORS, BIAS AND VARIANCE

The eld of statistics gives us many tools to achieve the machine learning goal of solving a task not only on the training set but also to generalize.

A. Point Estimation

Point estimation is the attempt to provide the single "best" prediction of some quantity of interest. In general the quantity of interest can be a single parameter or a vector of parameters in some parametric model, such as the weights in our linear regression example but it can also be a whole function. To distinguish estimates of parameters from their true value, our convention will be to denote a point estimate of a parameter $\boldsymbol{\Theta}$ by $\hat{\boldsymbol{\Theta}}$. Let $\{x^{(1)}, ..., x^{(m)}\}$ be a set of m independent and identically distributed i.i.d. data points. A point estimator or statistic is any function of the data:

$$\hat{\boldsymbol{\Theta}}_m = g(x^{(1)}, ..., x^{(m)}).$$

The denition does not require that g return a value that is close to the true $\boldsymbol{\Theta}$ or even that the range of g be the same as the set of allowable values of $\boldsymbol{\Theta}$. This definition of a point estimator is very general and

would enable the designer of an estimator great flexibility. While almost any function thus qualifies as an estimator, a good estimator is a function whose output is close to the true underlying $\boldsymbol{\Theta}$ that generated the training data. For now, we take the frequentist perspective on statistics. That is, we assume that the true parameter value $\boldsymbol{\Theta}$ is xed but unknown, while the point estimate $\hat{\boldsymbol{\Theta}}$ is a function of the data. Since the data is drawn from a random process, any function of the data is random. Therefore $\hat{\boldsymbol{\Theta}}$ is a random variable.

B. Bias

The bias of an estimator is dened as

$$bias(\hat{\boldsymbol{\Theta}}_m) = \mathbb{E}(\hat{\boldsymbol{\Theta}}_m) - \boldsymbol{\Theta}$$

where the expectation is over the data (seen as samples from a random variable) and $\boldsymbol{\Theta}$ is the true underlying value of $\boldsymbol{\Theta}$ used to dene the data-generating distribution. An estimator $\hat{\boldsymbol{\Theta}}_m$ is said to be unbiased if $bias(\hat{\boldsymbol{\Theta}}_m) = 0$, which implies that $\mathbb{E}(\hat{\boldsymbol{\Theta}}_m) = \boldsymbol{\Theta}$. An estimator is said to be asymptotically unbiased if

$$\lim_{m\to\infty} bias(\hat{\boldsymbol{\Theta}}_m) = 0,$$

which implies that

$$\lim_{m\to\infty} \mathbb{E}(\hat{\boldsymbol{\Theta}}_m) = \boldsymbol{\Theta}.$$

1. The Bernoulli Distribution

Consider a set of samples $\{x^{(1)}, ..., x^{(m)}\}$ that are independently and identically distributed according to a Bernoulli distribution with mean θ:

$$P(x^{(i)}, \theta) = \theta^{x^{(i)}}(1-\theta)^{1-x^{(i)}}.$$

A common estimator for the θ parameter of this distribution is the mean of the training samples:

$$\hat{\theta}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}.$$

To determine whether this estimator is biased simply substitute into

$$bias(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta.$$

We obtain

$$\begin{aligned} bias(\hat{\theta}_m) &= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right] - \theta \\ &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left[x^{(i)}\right] - \theta \\ &= \frac{1}{m}\sum_{i=1}^{m} (\theta) - \theta \\ &= 0. \end{aligned}$$

Since $\text{bias}(\hat{\theta}) = 0$, we say that our estimator $\hat{\theta}$ is unbiased.

2. The Gaussian Distribution Estimator of the Mean

Now, consider a set of samples $\{x^{(1)}, ..., x^{(m)}\}$ that are independently and identically distributed according to a Gaussian distribution $p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2)$ where $i \in \{1, ..., m\}$. Recall that the Gaussian probability density function is given by

$$p(x^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(x^{(i)} - \mu)^2}{\sigma^2} \right).$$

A common estimator of the Gaussian mean parameter is known as the sample mean:

$$\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}.$$

To determine the bias of the sample mean, we are again interested in calculating its expectation:

$$\begin{aligned} bias(\hat{\theta}_m) &= \mathbb{E}\left[\frac{1}{m} \sum_{i=1}^{m} x^{(i)} \right] - \mu \\ &= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}\left[x^{(i)} \right] - \mu \\ &= \frac{1}{m} \sum_{i=1}^{m} (\mu) - \mu \\ &= 0. \end{aligned}$$

Thus we nd that the sample mean is an unbiased estimator of Gaussian mean parameter.

C. Variance and Standard Error

Another property of the estimator that we might want to consider is how much we expect it to vary as a function of the data sample. Just as we computed the expectation of the estimator to determine its bias, we can compute its variance. The variance of an estimator $\mathrm{Var}(\hat{\theta})$ where the random variable is the training set. Alternately, the square root of the variance is called the standard error, denoted $\mathrm{SE}(\hat{\theta})$. The variance, or the standard error, of an estimator provides a measure of how we would expect the estimate we compute from data to vary as we independently resample the dataset from the underlying data-generating process. Just as we might like an estimator to exhibit low bias, we would also like it to have relatively low variance. When we compute any statistic using a finite number of samples, our estimate of the true underlying parameter is uncertain, in the sense that we could have obtained other samples from the same distribution and their statistics would have been different. The expected degree of variation in any estimator is a source of error that we want to quantify.