Machine Learning — Statistical Methods for Machine Learning

Introduction

Instructor: *Nicolò Cesa-Bianchi* version of March 5, 2019

Data inference is the study of methods that use data for making predictions about the future. Machine learning is a powerful tool that can be used to solve a wide range of data inference problems. For instance: categorization, clustering, time-series prediction, planning a sequence of actions. In this course we focus on machine learning systems whose goal is to learn functions that map data points $\boldsymbol{x}$ to labels y. Once learned, these functions can be used to classify documents or images, to predict which advertisement is more likely to be clicked on by a website visitor, to predict the yearly income of an individual based on lifestyle indicators, to diagnose a disease based on a patient's medical record, and so on.

Note that labels y implicitely defined in these examples are of two different kinds: symbolic labels, like the topics of a document, and numerical labels, like the income of an individual or the demand for a product. In the first case we have a categorization (or classification) problem with label set $\mathcal{Y}$ (for example, $\mathcal{Y} = \{\text{sport}, \text{politics}, \text{showbusiness}\}$). In the second case we have a regression problem, where the label set is contained in the reals $\mathbb{R}$. Classification mistakes are typically binary: if we predict the wrong label we make a mistake, irrespective of which wrong label we predicted. Regression mistakes, instead, measure some notion of distance between the predicted label and the correct label.

In order to measure the goodness of a prediction in a classification or regression problem we use a nonnegative **loss function** ℓ, measuring the discrepancy between the predicted label and the correct label. If the correct label for the data point $\boldsymbol{x}$ is y, the prediction $\widehat{y}$ is evaluated with $\ell(y, \widehat{y}) \geq 0$. In a classification problem, the most popular loss function is the zero-one loss:

$$\ell(y, \widehat{y}) = \begin{cases} 0 & \text{if } y = \widehat{y}, \\ 1 & \text{otherwise.} \end{cases}$$

Consider the problem of classifiying spam email using the label set $\mathcal{Y} = \{\text{spam}, \text{nonspam}\}$. In these cases, we may penalize a **false positive** mistake (i.e., a nonspam email wrongly classified as spam) more than a false negative mistake (i.e., a spam email wrongly classified as nonspam). For example,

$$\ell(y, \widehat{y}) = \begin{cases} 2 & \text{if } y = \text{nonspam and } \widehat{y} = \text{spam}, \\ 1 & \text{if } y = \text{spam and } \widehat{y} = \text{nonspam}, \\ 0 & \text{otherwise.} \end{cases}$$

In a regression problem, typical loss functions are the **absolute loss** $\ell(y, \widehat{y}) = |y - \widehat{y}|$ and the **quadratic loss** $\ell(y, \widehat{y}) = (y - \widehat{y})^2$. Note that these losses have only meaning when labels are numbers. Moreover, the value of the loss grows with the distance between y and $\widehat{y}$.

In some cases, it may be convenient to choose predictions from a set $\mathcal{Z}$ different from the label set $\mathcal{Y}$. For example, consider the problem of assigning a probability $\widehat{y} \in (0, 1)$ to the event $y =$ "it will rain tomorrow" (and, consequently, assigning probability $1 - \widehat{y}$ to the complementary event $y =$

"it will not rain tomorrow"). In this case, $\mathcal{Y} = \{\text{"rain"}, \text{"no rain"}\}$ and $\mathcal{Z} = (0, 1)$. Denoting these two events with 1 (for rain) and 0 (for no rain), we may use a loss function for regression, such as the absolute loss $\ell(y, \widehat{y}) = |y - \widehat{y}| \in (0, 1)$. In order to extend the range of the loss function, so to punish more harshly predictions that depart too much from reality, we may used instead the **logarithmic loss**,

$$\ell(y, \widehat{y}) = \begin{cases} \ln \frac{1}{\widehat{y}} & \text{if } y = 1 \text{ (rain)}, \\ \ln \frac{1}{1-\widehat{y}} & \text{if } y = 0 \text{ (no rain)}. \end{cases}$$

Note that, unlike the absolute loss, the logarithmic loss may take arbitrarily high values,

$$\lim_{\widehat{y} \to 0^+} \ell(+1, \widehat{y}) = \lim_{\widehat{y} \to 1^-} \ell(-1, \widehat{y}) = \infty \ .$$

In practice, this fact prevents the predictor from using predictions $\widehat{y}$ that are too certain, namely too close to zero or one.

Each data point $\boldsymbol{x}$ is typically stored as a database record. In many interesting cases, data can be encoded as vectors of real numbers, a representation that lends itself to being analyzed in geometrical terms. A vector representation is appropriate whenever the data consist of a set of homogeneous quantities, such as pixels in an image or word frequencies in a document. In other cases, a vector representation may not be natural at all. For example, the numerical fields of a medical record may be not directly comparable, such as age and ZIP code, or even take on symbolical values, such as gender. These data can be hardly viewed as points in an Euclidean space. In this course, we often assume that data can be represented as vectors of numbers, namely $\boldsymbol{x} \in \mathbb{R}^d$.

A **classifier** for a classification problem is a function $f : \mathcal{X} \to \mathcal{Y}$ (or $f : \mathcal{X} \to \mathcal{Z}$ if the predictions belong to a set $\mathcal{Z}$ different from $\mathcal{Y}$). When $\mathcal{Y}$ only contains two labels, for example $\mathcal{Y} = \{\text{spam}, \text{nonspam}\}$, we speak of a binary classification problem, and conventionally write $\mathcal{Y} = \{-1, +1\}$. Similarly, a regressor for a regression problem is a function $f : \mathcal{X} \to \mathbb{R}$. We use the term **predictor** to avoid specifying whether we are talking about classification or regression.

Even though several learning protocols exist, this course only considers the protocol of **learning by examples** (also known as supervised learning). An **example** is a pair $(\boldsymbol{x}, y)$ where $\boldsymbol{x}$ is a data point and y is the label associated with $\boldsymbol{x}$. In some cases, there is a unique true label for $\boldsymbol{x}$. This happens when y measures some objective property of the data point; for example, $\boldsymbol{x}$ is an individual and y is the individual's income in a given time interval. In some other cases, the label y is subjectively assigned by a human annotator; for example, the genre of a movie. Clearly, different annotators may have different opinions about a movie's genre, implying that the same data point may be associated with different labels.

A **training set** is a set[1] consisting of examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$. An **algorithm** for learning by examples receives a training set as input and outputs a predictor.

In order to estimate the predictive power of a predictor, which is what we are ultimately interested in, we typically use a **test set**. This is a (multi)set of examples $(\boldsymbol{x}'_1, y'_1), \ldots, (\boldsymbol{x}'_n, y'_n)$ that the

[1]Technically, it is a multiset as some examples may occur more than once. However, treating the training set as a multiset complicates the notation. For this reason, and without much loss of generality, we will mostly treat training sets (and test sets, introduced later on) as sets in the standard mathematical acception.

learning algorithm does not have access to. Training and test set are often collected together, through a single round of data collection and annotation. Partitioning the examples in training and test data is done afterward, typically using a random split.

The test set is used to compute the test error of predictors f,

$$\frac{1}{n}\sum_{t=1}^{n}\ell\big(y'_t, f(\boldsymbol{x}'_t)\big)\ .$$

The test error measures the behavior of the predictor "in the field", that is on examples not contained in the training set. Our objective is to develop a theory to guide us in the design of learning algorithms that generate predictor with low test error.

Since the only input to a learning algorithm is the training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$, a natural approach to the design of learning algorithms is to assume that the **training error**

$$\widehat{\ell}(f) = \frac{1}{m}\sum_{t=1}^{m}\ell\big(y_t, f(\boldsymbol{x}_t)\big)$$

of a predictor be correlated to its test error.

Let $\mathcal{F}$ be a given set of predictors. The **empirical risk minimizer** (ERM) denotes the learning algorithm that outputs the predictor in $\mathcal{F}$ minimizing the training error

$$\widehat{f} = \operatorname*{argmin}_{f\in\mathcal{F}}\widehat{\ell}(f)\ .$$

Unfortunately, this strategy may not always work well. For example, if the size of the training set is too small with respect to the cardinality of $\mathcal{F}$, it may be that $\mathcal{F}$ contains many predictors with low training error and high test error. In order to prevent this from happening, the training set should be large enough to help the algorithm discriminate the good predictors from the bad ones.

The situation where a learning algorithm tends to output predictors with low training error and large test error is known as **overfitting**. The control of overfitting is one of the key themes in machine learning. Overfitting is more likely to show up when data are affected by "noise".

Generally speaking, we say that a dataset is noisy when labels y are not deterministically associated with instances $\boldsymbol{x}$. Noise may occur for two (not mutually exclusive) reasons:

1. The labels are assigned by a human annotator who decides the "true" label for each data point. In this case, different annotators may have different opinions. This may happen —for instance— when labels denote the genre of a movie.

2. The information contained in the instance $\boldsymbol{x}$ is not sufficient to uniquely determine the label. For example, suppose $\boldsymbol{x}$ encodes measurements such as today's temperature, pressure, humidity, whereas $y \in \{-1, +1\}$ denotes whether tomorrow rains or not. It is quite possible that the same measured values lead to rain in some cases and to sun in other cases.

In the presence of noise, it is reasonable to predict the most popular label, i.e., the one that is most frequently associated with each given data point. Because in real-world datasets each data point

is unlikely to occur more than once, a learning algorithm can not reliably estimate frequencies of labels for individual data points. One can then constrain the learning algorithm to choose a "simple" predictor, hoping this restriction cause the algorithm to ignore the "anomalous" data points with infrequent labels.

When data are noise-free, labels y are deterministically associated with data points $\boldsymbol{x}$. Unfortunately, overfitting can occur even in this case. To see this, assume $\mathcal{Y} \equiv \{-1, +1\}$ and consider a toy problem with only five data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5$. Now, take some $\mathcal{F}$ containing a classifier $f : \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5\} \to \{0, 1\}$ for each of the $2^5 = 32$ possible binary labelings of the five points, and suppose we put three points in the training set and the two remaining points in the test set. No matter what are the true labels $y_1, \ldots, y_5$ of the five data points, we can always find four classifiers in $\mathcal{F}$ that have zero training error. Of these four classifiers with zero training error, only one has zero test error. However, the training set does not contain any information to help us choose this classifier over the others. Intuitively, the problem here is that $\mathcal{F}$ is "too big" with respect to the training set.

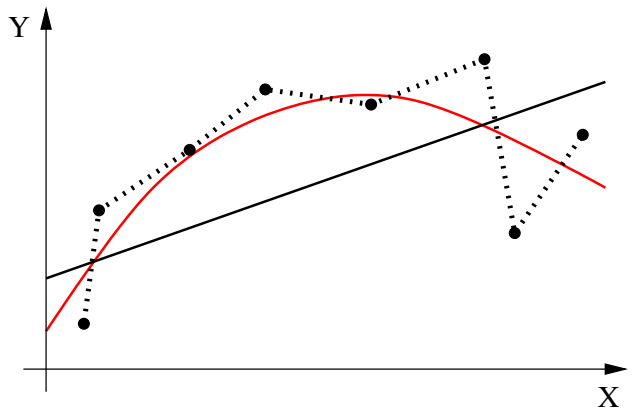

Figure 1: Control of overfitting in a one-dimensional regression problem. The training set contains the eight points $(x_t, y_t) \in \mathbb{R}^2$ depicted by black dots in the figure. The curves represent predictors of the form $f : \mathbb{R} \to \mathbb{R}$, each providing a different solution to the problem of predicting y given x. The red curve denotes the predictor minimizing the test error (the points in the test set are not shown here). The two black curves denote simpler (straight line) or more complex (dashed curve) predictors.

In case of regression problems, a simple overfitting example is shown by the dashed black curve in Figure 1. The black straight line shows an example of **underfitting**, which is the case when training error and test error are both high.