Supervised Learning

1 Introduction

Imagine we want to estimate the price of a housing unit in a specific city. The first factor that we may consider is the size of the unit. Therefore, other factors may also be included, such as its location in the city, its distance to train stations, its proximity to schools or to local stores. Likewise, its type also may influence the price. Examples of housing type include single-family house or a unit in a multi-unit dwelling.

Many other factors may influence the price. All these factors are generally called ***features*** or ***independent variables***. They may also know by other names such as ***attributes***, ***covariates***, and ***predictors***. The output (e.g., price) that we want to estimate is called ***output variable***, also known as ***response variable***, ***variates***, ***targets***, ***labels*** and more classically ***dependent variables***.

In machine learning, input and output variables may assume different types. They can be ***numeric*** variables representing numeric quantities like distance, area, or price; or ***categorical*** variable representing things or concepts. For instance, housing type is a categorical variable, as well as its location in a city. Numerical variables can be ***continuous*** or ***discrete***. When a output variable is categorical, its value is called ***class*** or ***label***. Categorical variables may also have ordering properties such as AAA and AA bonds in credit rating. In this case, these variables are called an ***ordinal*** variable.

Nominal variables are purely categorical variables with no ordinal relation. In practice, unless specified otherwise, "categorical" usually means "nominal" and vice-versa. In machine learning, a categorical variable is assumed to have a finite number of values. In this sense it can be regarded as discrete.

We normally use x to denote an input variable. If x is a vector, it is represented as $\vec{x}$, and its components can be accessed by subscripts $\vec{x}_i$. Quantitative outputs are denoted as y.

Supervised learning aims to find a way of expressing output variables $\hat{y}_i, \ldots, \hat{y}_n$ as a function of the input variables $\vec{x}_1, \ldots, \vec{x}_n$, where $\vec{x}_i$ is an element of the ***input space*** $\mathfrak{X}$ and each label $\hat{y}_i$ belongs to the ***output space*** $\mathfrak{D}$. Thus, the objective is to find a function:

$$f : \mathfrak{X} \mapsto \mathfrak{D}$$

such that $f(\vec{x}) \approx y$ for all pairs $(\vec{x}, y) \in \mathfrak{X}\ x\ \mathfrak{D}$.

In this case, f is called a ***predictor*** or ***prediction function***. It may be explicitly described by a mathematical model or it may be computed by an algorithm. Thus, we can loosely state the supervised learning task as a problem of finding a function that satisfies these criteria.

If the output value is continuous, i.e., $\mathfrak{D} = \mathbb{R}$ a ***predictor*** will be a ***regression function*** or simply a ***regressor***. When it is categorical, it is named a ***classifier***. If $y = \{0, 1\}$, the predictor will be a binary classifier, whereas when $y = \{1, 2, \ldots, C\},\ C > 2$, it will be a multi-class classifier.

Supervised learning requires data. As a result, we assume that we have a set of measurements $\mathscr{D} = \{(x^i, y^i)\}_{i=1}^p$, where $x_i^1, \ldots, x_i^p$ is the i-th data consisting of p features, and y^i is the i-th output value.

A predictor requires a function error, and depending on how we define it, we will have various supervised learning models. Consequently, when we find a predictor, we have to restrict the set of possible candidate functions, otherwise, we will have too many choose from, leading to a complex problem to deal with. The set of restricted functions is called ***hypothesis space*** denoted by $\mathcal{H}$ or $\mathcal{F}$. Hence, a machine learning problem presupposes a hypothesis space $\mathcal{H} = \{f | f : \mathfrak{X} \mapsto \mathfrak{D}\}$ and a search for a predictor in $\mathcal{H}$ that better fits the data. The process of finding a predictor is usually called ***learning*** or ***training***.

The goal of a predictor is not only to fit a given dataset at training time, but to continue to perform adequately in the future under the assumption that the new data will have the same probability distribution. Simply stating, we assume that the dataset $\mathscr{D} = \{(x^i, y^i)\}_{i=1}^p$ is a collection of ***independent and identically distributed (i.i.d)*** samples drawn from $\mathfrak{X}$ x $\mathfrak{D}$ according to a probability distribution $\mathbb{P}(x, y)$. In this case, $\mathbb{P}(x, y)$ denotes the joint probability distribution of $X \in \mathfrak{X}$ and $Y \in \mathfrak{D}$.

2 Regression Problem

A ***regression problem*** comprises the task of finding a ***predictor*** when the output variable is continuous. In most cases, the output space is assumed to be a Euclidean space $\mathbb{R}^m$, with a scalar output; i.e., $m = 1$.

Regression may be performed to (a) produce a trend line (or curve) that can help on visually summarize the data, (b) drive home a particular point about the data under study, or (c) to learn a model. In practice, many machine learning algorithms rely on linear predictors because of the ability to learn them efficiently in various cases. In addition, linear predictor are usually intuitive, easy to interpret, and fit the data reasonably well in many natural learning problems.

Data from regression problems come in the form of a training set P input/output pairs $\{(x^i, y^i)\}_{i=1}^p$, where x^i and y^i denote the ith input and output respectively. Each input x^i may be a column vector of length N.

$$\begin{bmatrix} x_{1,p} \\ \vdots \\ x_{n,p} \end{bmatrix}$$

Given a set of inputs x, we predict the output y via the model

$$\hat{Y} \approx \hat{\beta}_0 + \sum_{j=1}^{p} x_j \hat{\beta}_j$$

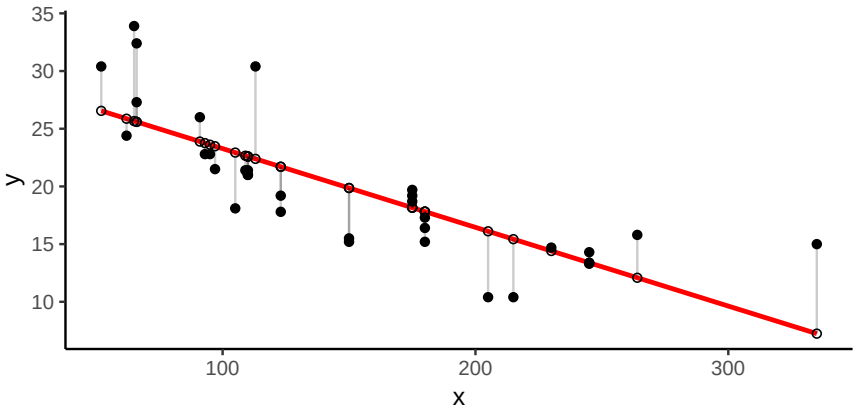


Figure 1: Illustration of the concept of least squares. The solid-red line minimizes the squared deviation between the observed data and the predicted value

The term $\hat{\beta}_0$ is the ***intercept***, also known in machine learning as the *bias*. If we include the constant variable 1 in $X \in \mathbb{R}^p$

$$X \hookleftarrow \begin{bmatrix} 1 & x_{1,1} & \cdots & x_p^1 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^n & \cdots & x_p^n \end{bmatrix}$$

and include $\hat{\beta}_0$ in the coefficients vector $\hat{\beta}$, we can write the linear model in the vector form as a product of an inner product.

$$\hat{Y} \approx x_p^T \hat{\beta}$$

2.1 Least Squares

Least squares is the most popular method to find the parameters of a hyperplane which best fits a regression training dataset. In this case, for the set of coefficients $\hat{\beta}$, this cost function computes the ***residual sum of squares (RSS)*** and the data, as illustrated in fig. 1, given a measure of how well the particular linear model fits the dataset. As a result, the best fitting hyperplane is the one that minimizes the residual sum of squares.

$$RSS(\beta) = \min_{\beta \in \mathbb{R}} \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$

The minimum of $RSS(\beta)$ always exists, but it may not be unique. Rewriting it using the matrix notation helps us to characterize it.

$$RSS(\beta) = \min_{\beta \in \mathbb{R}} (\boldsymbol{y} - \boldsymbol{\mathcal{X}}\beta)^T (\boldsymbol{y} - \boldsymbol{\mathcal{X}}\beta)$$

where $\boldsymbol{x}$ is a N x p matrix with each row representing an input vector, and $\boldsymbol{y}$ is an N-vector of the outputs in the training dataset. Differentiating w.r.t. β we obtain the ***normal equations***.

$$\boldsymbol{\mathcal{X}}^T (\boldsymbol{y} - \boldsymbol{\mathcal{X}}\beta) = 0$$

If $\boldsymbol{X}^T \boldsymbol{X}$ is nonsingular, then the unique solution is given by

$$\hat{\beta} = (\boldsymbol{\mathcal{X}}^T \boldsymbol{\mathcal{X}})^{-1} \boldsymbol{\mathcal{X}}^T \boldsymbol{y}$$

and the fitted value at the ith input x^i is $\hat{y}^i = \hat{y}(x^i) = x_i^T \hat{\beta}$.

2.2 Logistic Regression

Suppose that we want to solve a binary classification problem through the linear approach. In other words, we aim to model $y \in \{0, 1\}$ by employing a linear combination of the input variables. In this case, we can use a probabilistic model, where $\mathbb{P}(Y = y|X = \hat{x})$ is modeled as the combination of the input variables $\hat{x}$. Therefore, $\mathbb{P}(Y = y|X = \hat{x})$ must assume values between zero and one. Intuitively, this function is not linear when $\mathbb{P}(Y = 0|X = \hat{x})$ is close to 1. This means that $\hat{x}$ is probably negative and that a small perturbation of $\hat{x}$ may not affect this probability. Therefore, when $\mathbb{P}(Y = y|X = \hat{x})$ is close to 0.5 nothing opposes that a small perturbation of $\hat{x}$ affect this probability, as illustrated in fig. 2a. This is the reason to consider the ***logit transformation*** $\mathbb{P}(Y = y|X = \hat{x})$ as a linear combination of the input variables.

A logit function is defined as

$$logit : [0, 1] \mapsto \mathbb{R}$$

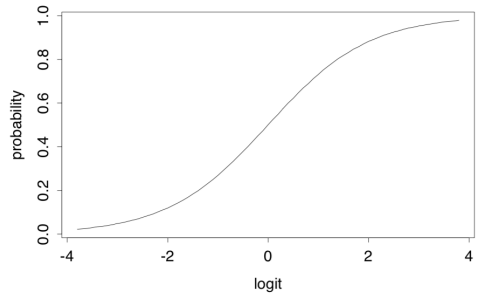
$$p \mapsto \frac{p}{1-p}$$

The ***logistic sigmoid*** function is at the heart of the logistic regression problem, depicted in fig. 2b, and mathematically defined as

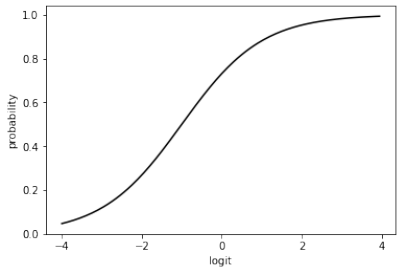
$$\sigma : \mathbb{R} \mapsto [0, 1]$$

$$u \mapsto \frac{1}{1+e^{-t}}$$

where t can take any real value.



(a) Logistic function



(b) Logistic function

This function was invented in the early 19th century by the mathematician Pierre Francois Verhuls, as a result of his pursuit to model how a population grows over time, taking into account the realistic assumption that regardless of the kind of organism under study, the system in which it lives has only a finite amount of resources.

If a dataset of n observations $\mathfrak{D} = \{\vec{x}^i, y^i\}_{i=1}^n$ and i.i.d, the likelihood of $\vec{\beta}$ is

$$\begin{aligned}
\log \mathbb{P}(\mathfrak{D}|\vec{\beta}) &= \log \prod_{i=1}^{n} \mathbb{P}(X = \vec{x}^i, Y = y^i|\vec{\beta}) \\
&= \log \prod_{i=1}^{n} \mathbb{P}(Y = y^i|\vec{x}^i, \vec{\beta}) + \log \prod_{i=1}^{n} \mathbb{P}(X = \vec{x}^i) \\
&= \sum_{i=1}^{n} \log \mathbb{P}(Y = 1|\vec{x}^i, \vec{\beta})^{y^i} (1 - \mathbb{P}(Y = 1|\vec{x}^i, \vec{\beta})^{1-y^i}) + c \\
&= \sum_{i=1}^{n} y^i \log \sigma(\vec{\beta}^T \vec{x}^i) + (1 - y^i) \log(1 - \sigma(\vec{\beta}^T \vec{x}^i)) + c
\end{aligned}$$

where c is a constant with respect to $\vec{\beta}$. Hence, the goal is to maximize the likelihood.

We consider a logistic regression the function $f : x \mapsto \sigma(\vec{beta}^T \vec{x})$, where the coefficients are computed as

$$argmax \sum_{i=i}^{n} y^i \log \sigma(\vec{\beta}^T \vec{x}^i) + (1 - y^i) \log(1 - \sigma(\vec{\beta}^T \vec{x}^i))$$

2.2.1 Checking regressors' errors

Mean square error (MSE) is a strategy to compute the errors associated to a predictor f. It is defined as:

$$MSE(f, \mathfrak{D}) = \frac{1}{N} \sum_{i=1}^{N} (f(x^i) - y^i)^2$$

This kind of error is generally called the ***empirical error***, ***in-sample error***, or ***empirical risk***. It is specific to the dataset (D), which means if the dataset changes, this kind of error will also change.

The intrinsic, or theoretical, error of f with respect to the probability $P(x, y)$ is defined as:

$$R(f) = \mathbb{E}|f(X) - Y|^2 = \int |f(x) - y|^2 P(x, y) dx dy$$

which is known as ***generalization error***, ***out-of-sample error***, or only ***risk***. Therefore, it is mostly a theoretical tools, as most of the machine learning problems are not concerned with finding out the probability distribution $P(x, y)$. The mean square error is an unbiased estimator of $R(f)$.

With the mean square error, we can also compute the ***root mean square error (RMSE)* and the ***coefficient of determination (R^2)***. The former indicates the standard deviation of the residuals, telling us how concentrated the data are around the line of the best fit; whereas the latter represents the fraction of response variance that is captured by the model.

$$RMSE = \sqrt{MSE}$$

$$R^2 = 1 - \frac{MSE}{Var(y)}$$

where *var* correspond to the variance of the output variable.

For the training dataset, R^2 is bounded between 0 and 1. Therefore, it can become negative for the test set. If $R^2 = 1$, the model fits the data perfectly, which correspond to a $MSE = 0$.