Machine Learning. Regression Case
Part 1. Linear Regression

Machine Learning: Regression Case.

Dataset. Consider a collection of features $\mathbf{X} = \{X_1, \ldots, X_n\}$, such that $dom(X_i) \subseteq \mathbb{R}$ for all $i = 1 \ldots n$[1]. Consider also an additional feature Y, such that $dom(Y) \subseteq \mathbb{R}$

Let $\mathbf{D} = \{\mathbf{x'_1}, \ldots, \mathbf{x'_m}\}$ be a collection of *data points*, such that $(\forall j \in 1 \ldots m)(\mathbf{x'_j} \in dom(\mathbf{X} \times dom(Y))$. We write $\mathbf{D}$ as

$$\mathbf{D} = \left(\begin{array}{ccccc} X_1 & X_2 & \ldots & X_n & \\ \hline x_{11} & x_{12} & \ldots & x_{1n} & y_1 \\ x_{11} & x_{12} & \ldots & x_{1n} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{11} & x_{12} & \ldots & x_{1n} & y_m \end{array} \right)$$

We also write $\mathbf{x_i} = (x_{i1}, \ldots, x_{in})$.

Machine Learning Question. We treat $\mathbf{X}$ as the *independent* or *observed variables* and Y as the *dependent* or *target variable*.

Our goal can be expressed as such:

> Given the dataset $\mathbf{D}$ of data points, find a relationship between the vectors $\mathbf{x_i}$ of observed data and the values y_i of the dependent variable.

Regression. Representing the relationship between the independent variables $X_1, \ldots, X_n$ and the target variable Y (based on observations $\mathbf{D}$) as a function

$$f : dom(X_1) \times \ldots \times dom(X_n) \longrightarrow dom(Y)$$

[1] Later, we will relax this condition.

or, more generally, as

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

is called *regression modeling*, and the function f that represents this relationship is called the *regression function*.

Simple Linear Regression (Multivariate Case)

To save space, we discuss multivariate regression case directly.

Linear Regression. A regression function

$$f(x_1, \ldots, x_n) : \mathbb{R}^n \longrightarrow \mathbb{R}$$

is called a linear regression function iff it has the form

$$y = f(x_1, \ldots, x_n) = \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_n x_n + \epsilon.$$

The values $\beta_1, \ldots, \beta_n$ are the linear coefficients of the regression function, otherwise known as *feature loadings*. The value ϵ, otherwise known as the *intercept* is the value of the regression function on the vector $x_0 = (0, \ldots, 0) : f(0, 0, \ldots, 0) = \epsilon$.

The linear regression equation may be rewritten as

$$y = \mathbf{x}^T \beta + \epsilon$$

Finding the best regression function. How do we find the "right" $f(\mathbf{x})$?

We can use the dataset $\mathbf{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$ for guidance.

Assume for a moment that we already know the values for coefficients $\beta_1, \ldots, \beta_n$, and ϵ for some linear regression function $f()$. Then, this function can be used to predict the values $y'_1, \ldots, y'_m$ of the target variable Y on inputs $\mathbf{x_1}, \ldots, \mathbf{x_m}$ respectively:

$$\begin{gathered}
y'_1 = \mathbf{x_1}^T \beta + \epsilon = \beta_1 x_{11} + \beta_2 x_{12} + \ldots + \beta_n x_{1n} + \epsilon \\
y'_2 = \mathbf{x_2}^T \beta + \epsilon = \beta_1 x_{21} + \beta_2 x_{22} + \ldots + \beta_n x_{2n} + \epsilon \\
\ldots \\
y'_m = \mathbf{x_m}^T \beta + \epsilon = \beta_1 x_{m1} + \beta_2 x_{m2} + \ldots + \beta_n x_{mn} + \epsilon
\end{gathered}$$

or:

$$\mathbf{y}' = \mathbf{D}^T \beta + \mathbf{1}\epsilon.$$

Thus, for each vector $\mathbf{x_i}$ we have the regression prediction y'_i, and y_i: the true value of Y.

> Our goal is to *minimize the error of prediction*, i.e., to *minimize the overall differences between the predicted and the observed values*.

Error. Given a vector $\mathbf{x_i}$, the prediction error $error(\mathbf{x_i})$ is defined as:

$$error(\mathbf{x_i}) = y_i - y'_i = y_i - \mathbf{x_i}^T + \epsilon$$

Minimizing the error. There is a wide range of ways in which error can be considered "minimized". The *traditional* way of doing so is called the *least squares method.* In this method

> we minimize the *sum of squares of the individual errors of prediction.*

That is, we define the overall error of prediction $Error(\mathbf{D})$ as following:

$$Error(\mathbf{D}) = Error(\mathbf{x_1}, \dots \mathbf{x_n}) = \sum_{i=1}^{m} error^2(\mathbf{x_i}) =$$

$$= \sum_{i=1}^{m}(y_i - y'_i)^2 = \sum_{i=1}^{m}(y_i - (\mathbf{x_i}^T\beta + \epsilon))^2$$

Among all β vectors, we want to find the one that **minimizes**

$$Error(D)$$

.

If we fix the dataset

$$\mathbf{D}$$

and instead let β and ϵ vary, can the error function as a function of β, ϵ:

$$L(\beta, \epsilon) = \sum_{i=1}^{m}(y_i - (\mathbf{x_i}^T\beta + \epsilon))^2$$

We want to minimize

$$L(\beta)$$

subject to β, ϵ:

$$\min_{\beta,\epsilon} L(\beta, \epsilon)$$

Solving the minimization problem. $L(\beta)$ is minimized when the derivative of $L()$ is equal to zero. This can be expressed as the following $n + 1$ equations:

$$\frac{\partial L}{\partial \beta_1} = 0$$
$$\frac{\partial L}{\partial \beta_2} = 0$$
$$\dots$$
$$\frac{\partial L}{\partial \beta_n} = 0$$

$$\frac{\partial L}{\partial \epsilon} = 0$$

To simplify notation, let $\beta_0 = \epsilon$ and let $\beta' = (\beta_0, \beta_1, \beta_2, \ldots, \beta_n)$. Further, let $\mathbf{x_i'} = (1, x_{i1}, \ldots, x_{in})^T$.

$$\frac{\partial L}{\partial \beta_j} = \frac{\partial \left(\sum_{i=0}^{m} (y_i - (\mathbf{x_i}^T \beta))^2\right)}{\partial \beta_i} =$$

$$= 2 \sum_{i=0}^{m} x_{ij} (y_i - (\beta_0 x_{i0} + \ldots + \beta_n x_{in}) = 0$$

The system of linear equations generated can be described as:

$$\mathbf{D}^T (y - \mathbf{D}\beta') = 0$$

Its solution is

$$\hat{\beta'} = (\mathbf{D}^T \mathbf{D})^{-1} \mathbf{D}^T \mathbf{y}$$

This can be rewritten as

$$\hat{\beta'} = (\mathbf{D}^T \mathbf{D})^{-1} \mathbf{D}^T \mathbf{y} = \left(\sum_{i=1}^{m} \mathbf{x_i} \mathbf{x_i}^T \right)^{-1} \sum_{i=1}^{m} \mathbf{x_i} y_i$$

We can then estimate the values of y for $\mathbf{x_1}, \ldots, \mathbf{x_m}$:

$$\hat{y_i} = \mathbf{x_i}^T \hat{\beta} = \mathbf{x_i} \cdot (D^T D)^{-1} D^T \mathbf{y},$$

or:

$$\hat{\mathbf{y}} = D\hat{\beta} = D(D^T D)^{-1} D^T \mathbf{y} = P\mathbf{y}$$

for $P = D(D^T D)^{-1} D^T$.

Goodness-of-fit. You can test how good your regression model is by asking what percentage of variance the regression is responsible for. This is called a *goodness-of-fit* test, and the metric estimating the goodness of fit is called the R^2 metric, and is computed as follows:

$$R^2 = \frac{\sum_{i=1}^{m} (\hat{y_i} - \mu_y)^2}{\sum_{i=1}^{m} (y_i - \mu_y)^2}$$

Linear Regression with Categorical Variables

The specification for linear regression assumes that all variables $X_1, \ldots, X_n$ are numeric, i.e., $dom(X_i) \in \mathbb{R}$. Sometimes, however, some of the variables in the set $\mathbf{X}$ are categorical.

Nominal Variables Case. Let X_i be a nominal variable with the domain $dom(X_i) = \{a_1, \ldots, a_k\}$. We proceed as follows.

- Select one value (w/o loss of generality, a_k) as the *baseline*.
- For each value $a \in \{a_1, \ldots, a_{k-1}\}$ create a new numeric variable X_a.
- Let the new set of variables be $X' = (X - \{X_i\}) \cup \{X_{a_1}, \ldots X_{a_{k-1}}\}$
- represent values $a_1, \ldots, a_{k-1}$ as vectors

$$\mathbf{a_1} = (1, 0, 0, \ldots, 0)$$
$$\mathbf{a_2} = (0, 1, 0, \ldots, 0)$$
$$\ldots$$
$$\mathbf{a_{k-1}} = (0, 0, 0, \ldots, 1),$$

  and represent value a_k as the vector $\mathbf{a_k} = (0, \ldots, 0)$
- Replace each vector $\mathbf{x_j} = (x_1, \ldots, x_{i-1}, a_l, \ldots, x_n)$ with vector $\mathbf{x'_i} = (x_1, \ldots, x_{i-1}, \mathbf{a_l}, x_n)$.
- Perform linear regression from X' to Y.

Ordinal Variables Case. Let X_i be an ordinal variable with domain $dom(X) = \{1, 2, \ldots, k\}$[2]. We replace X_i as follows.

- Select one value (w/o loss of generality, a_k) as the *baseline*.
- For each value $j \in \{1, \ldots, k-1\}$ create a new numeric variable X_{ij}.
- Let the new set of variables be $X' = (X - \{X_i\}) \cup \{X_{i1}, \ldots X_{ik-1}\}$
- Represent values $1, \ldots, k-1$ of X_i as vectors

$$\mathbf{a_1} = (1, 0, 0 \ldots, 0)$$
$$\mathbf{a_2} = (1, 1, 0 \ldots, 0)$$
$$\ldots$$
$$\mathbf{a_{k-1}} = (1, 1, 1, \ldots, 1),$$

  and represent the value $X_i = k$ as the vector $\mathbf{a_k} = (0, 0, 0, \ldots, 0)$.
- Replace each vector $\mathbf{x_j} = (x_1, \ldots, x_{i-1}, l, \ldots, x_n)$ with vector $\mathbf{x'_i} = (x_1, \ldots, x_{i-1}, \mathbf{a_l}, x_n)$.
- Perform linear regression from X' to Y.

[2] Without loss of generality, we can represent all ordinal domains this way.

Basis Expansion

Given a set of features $X = \{X_1, \ldots, X_n\}$, and a dataset $\{(\mathbf{x_i}, y_i)\}$, we can construct a least-squares linear regression using the features $X_1, \ldots X_n$.

However, we can also transform features.

Polynomial basis expansion. Given a feature X_i, add features $X_i^2, \ldots, X_i^d$ to the list of features, and represent the output as

$$y = \beta_0 + x_1\beta_1 + \ldots + \beta_{i1}x_i + \beta_{i2}x_i^2 + \ldots + \beta id x_i^d + \beta_{i+1}x_{i+1} + \ldots + \beta_n x_n$$

This can be applied to **any** number of features from X.

Interactions. Let X_i and X_j be two independent variables in X. We can add a feature $X_{ij} = X_iX_j$ to the model. If we want to build a linear regression model that accounts for interactions between all pairs of features, the expression will look as follows:

$$y = \beta_0 + x_1\beta_1 + \ldots + x_n\beta_n + x_1x_2\beta 12 + \ldots + x_{i-1}x_i\beta_{(i-1)i} = \beta_0 + \sum_{i=1}^{n} x_i\beta_i + \sum_{i=1}^{n}\sum_{j=1}^{n} x_ix_j\beta_{ij}$$

General Expansion. Let $f_1(\mathbf{x}), \ldots, f_s(\mathbf{x})$ be efficiently computable functions over $dom(X_1) \times \ldots dom(X_n)$. We can consider the regression of the form:

$$y = \beta_0 + f_1(\mathbf{x})\beta_1 + f_2(\mathbf{x})\beta_2 + \ldots + f_s(\mathbf{x})\beta_s.$$

Note. These regressions are **non-linear** in X, but **they are linear** in β. Therefore, the standard least squares regression techniques will work.

Evaluation

How accurate is the regression model? This can be measured in a number of ways.

Training Error. Training error, or SSE is the target function we are optimizing:

$$SSE = \sum_{i=1}^{m}(y_i - \mathbf{x_i}^T\hat{\beta}')^2$$

Problem: overfit for more complex models.

Solution: Look for ways to penalize complexity.

Akaike Information Criterion (AIC). AIC chooses the model with the lowest value of

$$AIC = \sum_{i=1}^{m} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \ldots + \hat{\beta}_p x_{pi}))^2 + 2p$$

AIC adds a penalty for number of parameters p used in the model.

Bayesian Information Criterion (BIC). BIC chooses the model with the lowest value of

$$AIC = \sum_{i=1}^{m} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \ldots + \hat{\beta}_p x_{pi}))^2 + p \log(m)$$

Like AIC, BIC adds a penalty for number of parameters p used in the model, but it scales the penalty by $\log$ of the size of the dataset.