Machine Learning.
Part 1. Linear Regression

Machine Learning: Regression Case.

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

Let $X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ be a collection of *data points*, such that $(\forall j \in 1 \ldots n)(\mathbf{x_j} \in dom(\mathbf{X})$. Let $\mathbf{y} = \{y_1, \ldots, y_n\}$ such that $(\forall j \in 1 \ldots n)(y_j \in dom(Y))$. We write X as

$$\mathbf{X} = \left(\begin{array}{cccc} X_1 & X_2 & \ldots & X_d \\ \hline x_{11} & x_{12} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nd} \end{array} \right)$$

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

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 X 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 $X_1, \ldots, X_d$ and the target variable Y (based on observations $\mathbf{X}$) as a function

[1] Later, we will relax this condition.

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

or, more generally, as

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

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

Linear Regression (Multivariate Case)

To save space, we discuss multivariate regression case directly.

Linear Regression. A regression function

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

is called a linear regression function iff it has the form

$$y = f(x_1, \ldots, x_d) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_d x_d + \epsilon.$$

The values $\beta_1, \ldots, \beta_n$ are the linear coefficients of the regression function, otherwise known as *feature loadings*.

The value β_0, otherwise known as the *intercept* is the value of the regression function on the vector $x_0 = (0, \ldots, 0) : f(0, 0, \ldots, 0) = \beta_0$.

The value ϵ represents *error*. The error is assumed to be independent of our data (vector $(x_1, \ldots x_n)$) and normally distributed with the mean of 0 and some unknown standard deviation σ^2:

$$\epsilon \sim N(0, \sigma^2)$$

The linear regression equation may be rewritten as

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

where $\mathbf{x}^T = (1, x_1, \ldots, x_d)$ and $\beta = (\beta_0, \beta_1, \ldots, \beta_d)$.

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

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

Assume for a moment that we already know the values for coefficients $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_n$ for some linear regression function $f()$. Then, this function can be used to predict the values $\hat{y}_1, \ldots, \hat{y}_n$ of the target variable Y on inputs $\mathbf{x_1}, \ldots, \mathbf{x_n}$: respectively:

$$\hat{y}_1 = \mathbf{x_1}^T \hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12} + \ldots + \beta_d x_{1d}$$
$$\hat{y}_2 = \mathbf{x_2}^T \hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22} + \ldots + \hat{\beta}_d x_{2d}$$
$$\ldots$$

$$\hat{y}_n = \mathbf{x_n}^T \hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_{n1} + \hat{\beta}_2 x_{m2} + \ldots + \hat{\beta}_d x_{nd}$$

If we, without loss of generality assume that matrix

$$\mathbf{X}$$

is

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1d} \\ 1 & x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nd} \end{pmatrix}$$

then we can rewrite the expressions above using the linear algebra notation:

$$\hat{\mathbf{y}} = X\hat{\beta}$$

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

> 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:

$$error(\mathbf{x_i}) = y_i - y_i' = y_i - \mathbf{x_i}^T = \epsilon_i$$

Maximum Liklihood Estimation. Under our assumptions, ϵ_i are independently identically distributed according to a normal distribution:

$$\epsilon_i \sim N(0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mathbf{x_i}^T \beta)^2}{2\sigma^2}}$$

The probability of observing the vector

$$\epsilon = (\epsilon_1, \ldots, \epsilon_n) = ((y_1 - \mathbf{x_1}^T \beta), \ldots (y_n - \mathbf{x_n}^T \beta))$$

can therefore be expressed as follows:

$$P(\epsilon|\beta, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mathbf{x_i}^T \beta)^2}{2\sigma^2}}$$

The log liklihood function can be represented as follows:

$$\ell(\beta, \sigma) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x_i}\beta)^2 =$$

$$-\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta)$$

We can maximize the log liklihood function by taking the partial derivatives of $\ell(\beta, \sigma)$ and setting the derivatives to 0.

For the purpose of prediction, we need to estimate the values $\beta = (\beta_0, \beta_1, \ldots, \beta_d)$, but we actually do not need an estimate for the variance σ^2.

We need σ^2 estimated though if we want to understand how well our linear regression model matches (fits) the observed data.

Estimating Regression Coefficients. The partial derivatives of $L(\beta, \sigma)$ on β_i parameters are:

$$\frac{\partial L}{\partial \beta_j} = \frac{1}{\sigma^2} \sum_{i=1}^{m} x_{ij}(y_i - \mathbf{x_i}^T \beta) = \frac{1}{\sigma^2} \sum_{i=1}^{n} x_{ij}(y_i - (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_n x_{id}))$$

If we set it to zero, we get

$$\sum_{i=1}^{m} x_{ij}(y_i - \mathbf{x_i}^T \beta) = 0$$

Generalizing for the entire β, we can get

$$\frac{\partial \ell}{\partial \beta} = \frac{1}{\sigma^2} \mathbf{X}^T (\mathbf{y} - \mathbf{X}\beta),$$

which, when set to a zero vector, gives us:

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

Solving for β we get

$$X^T \mathbf{y} = X^T X \beta,$$

which yields the closed form solution for β:

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}.$$

Least Squares Approximation. What type of an estimator did we just obtain?

Let us consider a numeric-analytical approach to predicting coefficients β. In predicting β, we want to minimize the error resulted in our prediction.

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 $Error(\mathbf{X})$ as following:

$$Error(X) = Error(\mathbf{x_1}, \ldots \mathbf{x_n}) = \sum_{i=1}^{n} error^2(\mathbf{x_i}) =$$

$$= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (\mathbf{x_i}^T \beta))^2$$

Among all β vectors, we want to find the one that **minimizes** $Error(X)$.

If we fix the dataset X and instead let β vary, the error function becomes a function of β:

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

We want to minimize

$$L(\beta)$$

subject to β:

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

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

$$\frac{\partial L}{\partial \beta_0} = 0$$
$$\frac{\partial L}{\partial \beta_1} = 0$$
$$\frac{\partial L}{\partial \beta_2} = 0$$
$$\cdots$$
$$\frac{\partial L}{\partial \beta_d} = 0$$

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

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

The system of linear equations generated can be described as:

$$X^T (y - X\beta') = 0$$

Its solution is

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

This can be rewritten as

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

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

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

or:

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

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

Conclusion. If we look at the Maximum Liklihood estimator we obtained and at the least squares estimator, we will notice and important fact – **it is the same estimator.**

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:

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

Tricks. We have two equations describing the solution for the estimates β.

The first equation is

$$X^T X \beta = X^T \mathbf{y}.$$

The second equation is

$$\beta = (X^T X)^{-} 1 X^T \mathbf{y}.$$

The advantage of the second equation is that it provides **a direct way of computing** β.

The disadvantage is the fact that the computation involves taking the inverse of a matrix.

The first equation is a linear equation system. While it does not give a closed form solution, it can be leveraged in actual computations, if you are relying on a linear algebra package (such as `NumPy`) for your linear algebra computations. In such a case, using the solver for a system of linear equations **rather than** a function that inverts a matrix is preferrable.

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 $\mathbf{X}' = (\mathbf{X} - \{X_i\}) \cup \{X_{a_1}, \ldots X_{a_{k-1}}\}$
- 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 $\mathbf{x_j} = (x_1, \ldots, x_{i-1}, a_l, \ldots, x_d)$ with vector $\mathbf{x'_i} = (x_1, \ldots, x_{i-1}, \mathbf{a_l}, x_d)$.
- Perform linear regression from X' to $\mathbf{y}$.

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 $\mathbf{x_j} = (x_1, \ldots, x_{i-1}, l, \ldots, x_d)$ with vector $\mathbf{x'_i} = (x_1, \ldots, x_{i-1}, \mathbf{a_l}, x_d)$.
- 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 $\mathbf{X} = \{X_1, \ldots, X_d\}$, and a dataset $\{(\mathbf{x_i}, y_i)\} = (X, \mathbf{y})$, we can construct a least-squares linear regression using the features $X_1, \ldots X_d$.

However, we can also transform features.

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

$$y = \beta_0 + x_1\beta_1 + \ldots + \beta_{i1}x_i + \beta_{i2}x_i^2 + \ldots + \beta_{im}x_i^m + \beta_{i+1}x_{i+1} + \ldots + \beta_d x_d$$

This can be applied to **any** number of features from $\mathbf{X}$.

Interactions. Let X_i and X_j be two independent variables in X. We can add a feature $X_{ij} = X_i X_j$ 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:

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

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

$$AIC = \sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \ldots + \hat{\beta}_d x_{di}))^2 + 2d$$

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

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

$$AIC = \sum_{i=1}^{d}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \ldots + \hat{\beta}_d x_{di}))^2 + d\log(n)$$

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