

Fundamentals of Machine Learning. Part 1. Linear Regression

Machine Learning: Regression Case.

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

Let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be a collection of *data points*, such that $(\forall j \in 1 \dots n)(\mathbf{x}_j \in \text{dom}(\mathbf{X}))$. Let $\mathbf{y} = \{y_1, \dots, y_n\}$ such that $(\forall j \in 1 \dots n)(y_j \in \text{dom}(Y))$. We write X as

$$\mathbf{X} = \begin{pmatrix} X_1 & X_2 & \dots & X_d \\ x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{pmatrix}$$

We also write $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$.

Supervised 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.

In supervised learning settings, we refer to \mathbf{X} as *training data*, and \mathbf{y} as *ground truth*. Without loss of generality, we often call the pair (\mathbf{X}, \mathbf{y}) *training set* or *training data*.

¹Later, we will relax this condition.

Regression vs. Classification. If $\text{dom}(Y) \subseteq \mathbb{R}$, then the supervised learning question above becomes a **regression problem**. That is, **regression** is the process of modeling dependencies between independent variables and one (or more) dependent variables when the dependent variables are **numeric**.

If $\text{dom}(Y) = \{c_1, \dots, c_k\}$, i.e., a finite set of categorical labels, then the supervised learning question becomes a **classification problem**. That is, **classification** is the process of modeling dependencies between independent variables and one dependent variable when the dependent variable is **categorical**².

General approach to supervised learning. In both **regression** and **classification** cases, the overall approach to the problem has the following components.

- **Model shape.** The relationship between the independent variables X_1, \dots, X_d and the target variable Y (based on observations \mathbf{X}) is represented as some function

$$f : \text{dom}(X_1) \times \dots \times \text{dom}(X_d) \longrightarrow \text{dom}(Y).$$

In regression cases, this function becomes

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

For the **classification** case, function f is called a **classifier** or **classification model**.

For the **regression** case, function f is called a **regressor** or **regression model**.

- **Optimization criterion.** An arbitrary function f from $\text{dom}(X_1) \times \dots \times \text{dom}(X_d)$ to $\text{dom}(Y)$ is most likely not going to be a *good representation* of the true relationship between the independent and dependent variables.

In order to successfully find an appropriate classification or regression model, we need a criterion that allows us to *select the best possible function* out of a universe of possible functions. This criterion typically relies (heavily) on the data contained in the dataset \mathbf{X} and the ground truth data \mathbf{y} .

The optimization criterion is generally specified as follows.

The classification/regression approach considers a family of models

$$f(\beta, \mathbf{x})$$

where $\mathbf{x} \in \text{dom}(X_1) \times \dots \times \text{dom}(X_d)$, and $\beta = (\beta_1, \dots, \beta_m)$ is a set of numeric **parameters** determining the final "shape" of the model.

Given a fixed vector \mathbf{x} , the model becomes a function of its parameters β :

$$f : \mathbb{R}^m \longrightarrow \text{dom}(Y)$$

²Note, that while *multiple regression*, i.e., the process of modeling multiple dependent variables in parallel, is an important part of regression modeling, there are virtually no classification scenarios where multiple dependent categorical variables have to be modeled in conjunction with each other. That is, for classification, whenever we have multiple class variables, we consider modeling each class variable as a separate independent problem.

An optimization criterion is essentially a condition $G(\beta)$ that depends on the vector $\beta = (\beta_1, \dots, \beta_m)$. Typically, this condition is expressed in a form of

$$G(\beta) \longrightarrow \min,$$

(i.e., we want to find values of β that minimize the value of function G), or

$$G(\beta) \longrightarrow \max,$$

(i.e., we want to find values of β that maximize the value of function G)

- **Optimization procedure/Model Fit.** The heart of any supervised learning process is the procedure of finding the appropriate model, typically called **model fit** procedure. This is also what is *typically* known as a *classification algorithm* or *regression algorithm*.

All supervised learning methods can be broken into two groups:

- **Methods with explicit model fit procedure.** These methods outline the shape of the model $f(\beta)$ and the optimization criterion $G(\beta)$ explicitly, and solve the optimization problem either in closed form to obtain the exact solution, or iteratively to obtain an approximate solution.

Methods with explicit model fit procedure include *linear regression*, *support vector machines for both regression and classification*, *logistic regression* (which is actually a classification method), and *neural networks for both regression and classification* among many others.

- **Methods with implicit model fit procedure.** These methods "hide" the exact function $f(\beta)$ and the optimization criterion $G(\beta)$, and instead concentrate on implementing a *supervised learning heuristic* via a set of algorithmic steps. It is possible to reverse-engineer the explicit shape of $f(\beta)$ and $G(\beta)$ for such methods, but these are typically more complex.

Methods with implicit model fit procedure include *K-nearest neighbors* and (to a point) any *decision tree classifiers* and *association rule classifiers*.

- **Predictor.** Once the optimal values $\hat{\beta}$ of parameters β are found by the model fit process/supervised learning algorithm, the function

$$f(\hat{\beta}) : \text{dom}(X_1) \times \dots \times \text{dom}(X_d) \longrightarrow \text{dom}(Y)$$

becomes the **predictor function** or *learned model* that can be used in applications that require actually making predictions.

(Note, some supervised learning applications call for *interpretive* approach, which means that while the shape of the learned model is very important for consideration, the model itself is not intended for making any predictions.)

We illustrate the supervised learning process on the case of *multivariate linear regression*.

Linear Regression (Multivariate Case)

To save space, we discuss multivariate regression case directly.

Linear Regression. A regression function

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

is called a linear regression function iff it has the form

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

The values β_1, \dots, β_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 $x_0 = (0, \dots, 0) : f(0, 0, \dots, 0) = \beta_0$.

The value ϵ represents *error*. The error is assumed to be independent of our data (vector (x_1, \dots, 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 $\mathbf{x}^T = (1, x_1, \dots, x_d)$ and $\beta = (\beta_0, \beta_1, \dots, \beta_d)$.

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

We can use the dataset $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ for guidance.

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

$$\begin{aligned}\hat{y}_1 &= \mathbf{x}_1^T \hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{12} + \dots + \hat{\beta}_d x_{1d} \\ \hat{y}_2 &= \mathbf{x}_2^T \hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_{21} + \hat{\beta}_2 x_{22} + \dots + \hat{\beta}_d x_{2d} \\ &\dots \\ \hat{y}_n &= \mathbf{x}_n^T \hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_{n1} + \hat{\beta}_2 x_{n2} + \dots + \hat{\beta}_d x_{nd}\end{aligned}$$

If we, without loss of generality assume that matrix

X

is

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

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:

$$error(\mathbf{x}_i) = y_i - y'_i = y_i - \mathbf{x}_i^T \beta = \epsilon_i$$

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:

$$\begin{aligned} Error(X) &= Error(\mathbf{x}_1, \dots, \mathbf{x}_n) = \sum_{i=1}^n error^2(\mathbf{x}_i) = \\ &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (\mathbf{x}_i^T \beta))^2 \end{aligned}$$

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:

$$\begin{aligned}\frac{\partial L}{\partial \beta_0} &= 0 \\ \frac{\partial L}{\partial \beta_1} &= 0 \\ \frac{\partial L}{\partial \beta_2} &= 0 \\ &\dots \\ \frac{\partial L}{\partial \beta_d} &= 0\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial \beta_j} &= \frac{\partial (\sum_{i=0}^m (y_i - (\mathbf{x}_i^T \beta))^2)}{\partial \beta_i} = \\ &= 2 \sum_{i=0}^m x_{ij} (y_i - (\beta_0 x_{i0} + \dots + \beta_n x_{in})) = 0\end{aligned}$$

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 $\mathbf{x}_1, \dots, \mathbf{x}_m$:

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

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 Likelihood estimator we obtained and at the least squares estimator, we will notice an 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 preferable.

Linear Regression with Categorical Variables

The specification for linear regression assumes that all variables X_1, \dots, X_n are numeric, i.e., $\text{dom}(X_i) \in \mathbb{R}$. 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 $\text{dom}(X_i) = \{a_1, \dots, a_k\}$. We proceed as follows.

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

$$\mathbf{a}_1 = (1, 0, 0, \dots, 0)$$

$$\mathbf{a}_2 = (0, 1, 0, \dots, 0)$$

...

$$\mathbf{a}_{k-1} = (0, 0, 0, \dots, 1),$$

and represent value a_k as the vector $\mathbf{a}_k = (0, \dots, 0)$

- Replace each vector $\mathbf{x}_j = (x_1, \dots, x_{i-1}, a_l, \dots, x_d)$ with vector $\mathbf{x}'_j = (x_1, \dots, x_{i-1}, \mathbf{a}_l, x_d)$.
- Perform linear regression from X' to \mathbf{y} .

Ordinal Variables Case. Let X_i be an ordinal variable with domain $\text{dom}(X) = \{1, 2, \dots, k\}$ ³. We replace X_i as follows.

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

$$\mathbf{a}_1 = (1, 0, 0, \dots, 0)$$

$$\mathbf{a}_2 = (1, 1, 0, \dots, 0)$$

...

$$\mathbf{a}_{k-1} = (1, 1, 1, \dots, 1),$$

and represent the value $X_i = k$ as the vector $\mathbf{a}_k = (0, 0, 0, \dots, 0)$.

- Replace each vector $\mathbf{x}_j = (x_1, \dots, x_{i-1}, l, \dots, x_d)$ with vector $\mathbf{x}'_j = (x_1, \dots, x_{i-1}, \mathbf{a}_l, x_d)$.
- Perform linear regression from X' to Y .

Basis Expansion

Given a set of features $\mathbf{X} = \{X_1, \dots, X_d\}$, and a dataset $\{(\mathbf{x}_i, y_i)\} = (X, \mathbf{y})$, we can construct a least-squares linear regression using the features X_1, \dots, X_d .

However, we can also transform features.

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

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

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

³Without loss of generality, we can represent all ordinal domains this way.

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:

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

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

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

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.

Linear Regression for Interpretive Purposes

Most of statistical analysis is *interpretive* rather than *predictive*. That is, the analyst is interested in a model that provides for a clear explanation of the dependencies between the independent variables and the dependent variable, but the analyst is not interested in using the model for predictive purposes.

In such cases, *predictive accuracy* as represented by least squares minimization criterion is of secondary importance to *model interpretability*. Typically, linear regression models that include *fewer* independent variables with *larger coefficients* $\hat{\beta}_i$ are preferable to models with more non-zero coefficients but of lower magnitudes, *even if that latter models have better predictive accuracy/lower error*.

To "force" the model fit process to find simpler models, we can modify the least squares optimization criterion in a variety of ways. Two such ways are briefly presented below.

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

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

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

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

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.

Maximum Likelihood Estimation for Linear Regression

The Linear Regression problem can also be expressed in terms of a maximum likelihood estimation problem.

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

$$\epsilon = (\epsilon_1, \dots, \epsilon_n) = ((y_1 - \mathbf{x}_1^T \beta), \dots, (y_n - \mathbf{x}_n^T \beta))$$

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 likelihood function can be represented as follows:

$$\begin{aligned} \ell(\beta, \sigma) &= -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{x}_i \beta)^2 = \\ &= -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (\mathbf{y} - X\beta)^T (\mathbf{y} - X\beta) \end{aligned}$$

We can maximize the log likelihood 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 $\beta = (\beta_0, \beta_1, \dots, \beta_d)$, 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:

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

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:

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

Solving for β we get

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

which yields the closed form solution for β :

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