# Chapter 1

# Linear Regression

## 1.1 Problem Setup

Assume a dataset $\{\mathbf{X}, \mathbf{Y}\}$ where

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

The discrete or continuous independent variable (a.k.a. input, predictor, covariate) $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the continuous dependent variable (a.k.a. output, target, label, response variable) $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, $n$ is the number of training examples and $d$ is the number of features.

The goal is to use a supervised learning approach for a regression application by learning a model hypothesis $h_\beta(\mathbf{x}^{(i)})$ which predicts $\hat{y}^{(i)}$ given $\mathbf{x}^{(i)}$, assuming the relationship between the two is linear.

## 1.2 Model

For simplicity we can assume our input features are univariate ($d = 1$). We can use a simple linear model to describe the relationship between the $i^{\text{th}}$ input and output using the following equation:

$$\boxed{h_\beta(\mathbf{x}^{(i)}) = \beta_0 + \beta_1 x_1^{(i)} + \cdots + x_d^{(i)} = \beta_0 + \sum_{j=1}^{d} \beta_j x_j^{(i)}}$$

where $i = (1, \dots, n)$, $j = (1, \dots, d)$, $\beta_0$(a.k.a. bias or intercept) represents the average output when all $\mathbf{X} = 0$, $\beta_j$ (a.k.a. coefficient or slope) represents the average effect of one-unit increase of a certain feature $j$ on $\mathbf{Y}$

Note that the term $\beta_0$ makes the function affine.

## 1.3 Loss Function

In order to qunaitfy how well our simple model performs, we must define a function that quanitifies how "well" our model predictions $\hat{y}$ matched the actual output $y$. We call this our "loss" function. One way define our loss is to calculate the distance between the two values. We will call this distance the residual $r^{(i)}$, where

$$r^{(i)} = y^{(i)} - (\beta_0 + \beta_1 x^{(i)})$$

We would like the residuals to be close to zero.

## 1.4 Cost Function

To find how well our model as a whole performs, we would like to find the average loss over all our examples. We can do so by summing the absolute values of our residuals and dividing the sum by the number of examples. This is called Mean Absolute Error (MAE). The caveat is, absolute values are non-differentiable functions, which are not desirable for training. We can instead choose to average the sum of squared residuals instead, keeping the function differentiable, with an added benefit of penalizing larger losses over smaller ones. This gives us the Mean Squared Error (MSE).

$$\boxed{\mathcal{E} = \frac{1}{n}\sum_{i=1}^{n} r_i^2}$$

Note: Dividing the sum of squared residuals by $n$ gives us a biased estimate of the variance of the unobserved errors. Do remove the bias, divide the sum of squared residuals by $\text{df} = n - d - 1$ where df is the degrees of freedom, and $d$ is the number of parameters being estimated (excluding the intercept).

## 1.5 Training

Now that we have a model, along with a function to evaluate the fit, we would like to find the optimal values of our parameters $\beta$ that would minimize the cost. Fortunately, for linear regression problems, there is a direct, closed-form solution that we can employ. We will also take a look at optimizing our parameters using Gradient Descent and Maximum Likelihood Estimation.

### 1.5.1 Direct Solution

Note: Recall from single variable calculus that (assuming a function is differentiable) the minimum $x^*$ of a function $f$ has the property that the derivative $\frac{\mathrm{d}f}{\mathrm{d}x}$ is zero at $x = x^*$. Note that the converse is not true: if $\frac{\mathrm{d}f}{\mathrm{d}x} = 0$, then $x^*$ might be a maximum or an inflection point, rather than a minimum. But the minimum can only occur at points that have derivative zero.

We will use this information to find the parameters that minimize our cost in the case of a univariate distribution. We can take the partial derivatives of the cost function with respect to the parameters $\beta_0$ and $\beta_1$, set the derivatives to zero, then solve for our estimated parameters $\hat{\beta}_0$ and $\hat{\beta}_1$.

Taking the partial derivative w.r.t. $\hat{\beta}_0$:

$$\begin{aligned}
\frac{\delta \mathcal{E}}{\delta \hat{\beta}_0} &= \frac{\delta}{\delta \hat{\beta}_0} \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - \hat{y}^{(i)} \right)^2 \\
&= \frac{\delta}{\delta \hat{\beta}_0} \frac{1}{n} \sum_{i=1}^{n} \left[ y^{(i)} - \left( \hat{\beta}_0 + \hat{\beta}_1 x^{(i)} \right) \right]^2 \\
&= \frac{-2}{n} \sum_{i=1}^{n} \left( y^{(i)} - \hat{\beta}_0 - \hat{\beta}_1 x^{(i)} \right) \\
&= \frac{-2}{n} \left( \sum_{i=1}^{n} y^{(i)} - n\hat{\beta}_0 - \hat{\beta}_1 \sum_{i=1}^{n} x^{(i)} \right) \\
&= -2 \left( \frac{n}{n} \hat{\beta}_0 + \hat{\beta}_1 \frac{1}{n} \sum_{i=1}^{n} x^{(i)} - \frac{1}{n} \sum_{i=1}^{n} y^{(i)} \right) \\
&= -2 \left( \hat{\beta}_0 + \hat{\beta}_1 \bar{x} - \bar{y} \right)
\end{aligned}$$

Setting the partial derivative to 0 and solving for $\hat{\beta}_0$:

$$\begin{aligned}
&-2 \left[ \hat{\beta}_0 + \hat{\beta}_1 \bar{x} - \bar{y} \right] = 0 \\
&\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}
\end{aligned}$$

Taking the partial derivative of $\hat{\beta}_1$:

$$\begin{aligned}
\frac{\delta \mathcal{E}}{\delta \hat{\beta}_1} &= \frac{\delta}{\delta \hat{\beta}_1} \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - \hat{y}^{(i)} \right)^2 \\
&= \frac{\delta}{\delta \hat{\beta}_1} \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - \left( \hat{\beta}_0 + \hat{\beta}_1 x^{(i)} \right) \right)^2 \\
&= \frac{-2}{n} \sum_{i=1}^{n} (x^{(i)}) \left( y^{(i)} - \hat{\beta}_0 - \hat{\beta}_1 x^{(i)} \right) \\
&= \frac{-2}{n} \sum_{i=1}^{n} \left( x^{(i)} y^{(i)} - (\bar{y} - \hat{\beta}_1 \bar{x}) x^{(i)} - \hat{\beta}_1 x^{(i)^2} \right) \\
&= \frac{-2}{n} \left( \sum_{i=1}^{n} x^{(i)} y^{(i)} - \bar{y} \sum_{i=1}^{n} x^{(i)} + \hat{\beta}_1 \bar{x} \sum_{i=1}^{n} x^{(i)} - \hat{\beta}_1 \sum_{i=1}^{n} x^{(i)^2} \right)
\end{aligned}$$

Setting the partial derivative to 0 and solving for $\hat{\beta}_1$:

$$\left(-\sum_{i=1}^{n} x^{(i)} y^{(i)} + \bar{y}\sum_{i=1}^{n} x^{(i)} - \hat{\beta}_1 \bar{x}\sum_{i=1}^{n} x^{(i)} + \hat{\beta}_1 \sum_{i=1}^{n} x^{(i)^2}\right) = 0$$

$$\hat{\beta}_1 \left(\bar{x}\sum_{i=1}^{n} x^{(i)} - \sum_{i=1}^{n} x^{(i)^2}\right) = \bar{y}\sum_{i=1}^{n} x^{(i)} - \sum_{i=1}^{n} x^{(i)} y^{(i)}$$

$$\hat{\beta}_1 = \frac{\bar{y}\sum_{i=1}^{n} x^{(i)} - \sum_{i=1}^{n} x^{(i)} y^{(i)}}{\bar{x}\sum_{i=1}^{n} x^{(i)} - \sum_{i=1}^{n} x^{(i)^2}}$$

$$= \frac{n\bar{y}\bar{x} - \sum_{i=1}^{n} x^{(i)} y^{(i)}}{n\bar{x}^2 - \sum_{i=1}^{n} x^{(i)^2}} = \frac{\sum_{i=1}^{n} x^{(i)} y^{(i)} - n\bar{y}\bar{x}}{\sum_{i=1}^{n} x^{(i)^2} - n\bar{x}^2}$$

To simplify:

$$\sum_{i=1}^{n} x^{(i)} y^{(i)} - n\bar{y}\bar{x} = \sum_{i=1}^{n} (x^{(i)} - \bar{x})(y^{(i)} - \bar{y}) := \mathrm{S_{XY}}$$

$$\sum_{i=1}^{n} x^{(i)^2} - n\bar{x}^2 = \sum_{i=1}^{n} (x^{(i)} - \bar{x})^2 := \mathrm{S_{XX}}$$

Thus,

$$\hat{\beta}_1 = \frac{\mathrm{S_{XY}}}{\mathrm{S_{XX}}}$$

$$\hat{\beta}_0 = \bar{y} - \frac{\mathrm{S_{XY}}}{\mathrm{S_{XX}}}\bar{x}$$

We can expand this approach to multivariate training examples. To do so, we shall look at using a linear algebra appraoch to find a closed-form solution to linear regression. TODO:

### 1.5.2 Gradient Descent

We can also use an iterative approach for optimization of our parameters. This approach is called gradient descent. The basic algorithm of gradient descent is as follows:

- Initialize all parameters $\beta$ to some random value
- Repeat until convergence:
  - Choose a new value for $\beta$ to reduce $\mathcal{E}(\beta)$ toward the direction of steepest descent.
  - Simultaneously update for $j = (0, \ldots, d)$

The gradient descent step is formulated by the following function:

$$\beta_j := \beta_j - \alpha \frac{\delta}{\delta \beta_j} \mathcal{E}(\beta)$$

For simple linear regression, let's assume $\beta_0$ is included in $\beta_j$. We can assume this if we assume $x_0 = 1$. This will keep consistency in our formulas. So, for each $j$

$$\begin{aligned}
\frac{\delta \mathcal{E}}{\delta \beta_j} &= \frac{\delta}{\delta \beta_j} \frac{1}{2n} \sum_{i=1}^{n} \left( \hat{y}^{(i)} - y^{(i)} \right)^2 \\
&= \frac{\delta}{\delta \beta_j} \frac{1}{2n} \sum_{i=1}^{n} \left( \sum_{k=0}^{d} \beta_k + x_k^{(i)} - y^{(i)} \right)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)}) \left( \sum_{k=0}^{d} \beta_k + x_k^{(i)} - y^{(i)} \right)
\end{aligned}$$

Thus, for all training examples:

$$\beta_j := \beta_j - \alpha \sum_{i=1}^{n} x_j^{(i)} \left( \sum_{k=0}^{d} \beta_k + x_k^{(i)} - y^{(i)} \right)$$

Repeat the above until convergence and update for $j = (0, 1, \ldots, d)$ Note that $\mathcal{L}(\beta)$ is a quadratic equation and $\alpha$ is an empirical value. The above equation is also called batch gradient descent. All data is summed at every step, which is very ineffective for big datasets. Alternatively, you can use Stochastic Gradient Descent (SGD):

Repeat {
  For $i = 1$ to $n$ {
  $\beta_j := \beta_j - \alpha \left[ h_\beta x^{(i)} - y^{(i)} \right] x_j^{(i)}$
  }
}

The above takes derivative of one example at a time. The gradient takes a bit more noisy approach toward minima and it never converges (unlike batch gradient descent).

### 1.5.3 Probabilistic Solution

## 1.6 Evaluation

# Chapter 2

# Linear Regression (Regularization)

# Chapter 3

# Logistic Regression