# Linear Regression: History

- A very popular technique.
- Rooted in Statistics.
- Method of Least Squares used as early as 1795 by Gauss.
- Re-invented in 1805 by Legendre.
- Frequently applied in **astronomy** to study the large scale of the universe.
- Still a very useful tool today.

Carl Friedrich Gauss

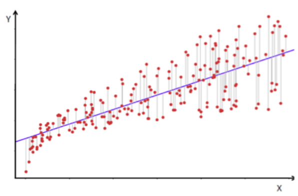
# Linear Regression

**Given:** Training data: $(x_1, y_1), \ldots, (x_n, y_n)$, $x_i \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

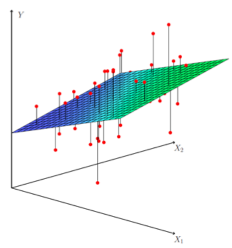
| | | | | | |
|---|---|---|---|---|---|
| example $x_1 \rightarrow$ | $x_{11}$ | $x_{12}$ | $\ldots$ | $x_{1d}$ | $y_1 \leftarrow$ label |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| example $x_i \rightarrow$ | $x_{i1}$ | $x_{i2}$ | $\ldots$ | $x_{id}$ | $y_i \leftarrow$ label |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| example $x_n \rightarrow$ | $x_{n1}$ | $x_{n2}$ | $\ldots$ | $x_{nd}$ | $y_n \leftarrow$ label |

**Task:** Learn a regression function: $f\colon \mathbb{R}^d \rightarrow \mathbb{R}$, $f(x) = y$

**Linear Regression:** A regression model is said to be linear if it is represented by a linear function.




$d = 1$, line in $\mathbb{R}^2$




$d = 2$, hyperplane is $\mathbb{R}^3$

线性回归是一条直线/平面(超平面)

**Linear Regression Model:**

$$f(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \quad \text{with} \quad \beta_j \in \mathbb{R}, \quad j \in \{1, \ldots, d\}$$

$\beta$'s are called parameters or coefficients or weights.

**Learning the linear model $\rightarrow$ learning the $\beta$'s**

**Estimation with Least squares:**

Use least square loss: $loss(y_i, f(x_i)) = (y_i - f(x_i))^2$

We want to minimize the loss over all examples, that is minimize the *risk or cost function R*:

$$R = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

**A simple case with one feature ($d$ = 1):** $f(x) = \beta_0 + \beta_1 x$

**We want to minimize:** $R = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$

$$R(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

**Find $\beta_0$ and $\beta_1$ that minimize:**

$$R(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

**Find $\beta_0$ and $\beta_1$ that minimize:** $argmin_\beta(\frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2)$

**Minimize:** $R(\beta_0, \beta_1)$, that is: $\frac{\partial R}{\partial \beta_0} = 0 \qquad \frac{\partial R}{\partial \beta_1} = 0$

$$\frac{\partial R}{\partial \beta_0} = 2 \times \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \times \frac{\partial}{\partial \beta_0}(y_i - \beta_0 - \beta_1 x_i)$$

$$\frac{\partial R}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \times (-1) = 0$$

$$\beta_0 = \frac{1}{n}\sum_{i=1}^{n} y_i - \beta_1 \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\frac{\partial R}{\partial \beta_1} = 2 \times \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \times \frac{\partial}{\partial \beta_1}(y_i - \beta_0 - \beta_1 x_i)$$

$$\frac{\partial R}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \times (-x_i) = 0$$

$$\beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} \beta_0 x_i$$

**Plugging $\beta_0$ and $\beta_1$:**

$$\beta_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{1}{n}\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\sum_{i=1}^{n} x_i \sum x_i}$$

9

**With more than one feature:**

$$f(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j$$

**Find the $\beta_j$ that minimize:**

$$R = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij})^2$$

**Let's write it more elegantly with matrices!**

## Matrix representation

Let $X$ be an $n \times (d+1)$ matrix where each row starts with a 1 followed by a feature vector.

Let $y$ be the label vector of the training set.

Let $\beta$ be the vector of weights (that we want to estimate!).

$$X := \begin{pmatrix} 1 & x_{11} & \cdots & x_{1j} & \cdots & x_{1d} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{i1} & \cdots & x_{ij} & \cdots & x_{id} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nj} & \cdots & x_{nd} \end{pmatrix} \qquad y := \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} \qquad \beta := \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_j \\ \vdots \\ \beta_d \end{pmatrix}$$

## Normal Equation

We want to find ($d$+1) $\beta$'s that minimize $R$. We write $R$:

$$R(\beta) = \frac{1}{2n}||(y - X\beta)||^2$$

$$R(\beta) = \frac{1}{2n}(y - X\beta)^T(y - X\beta)$$

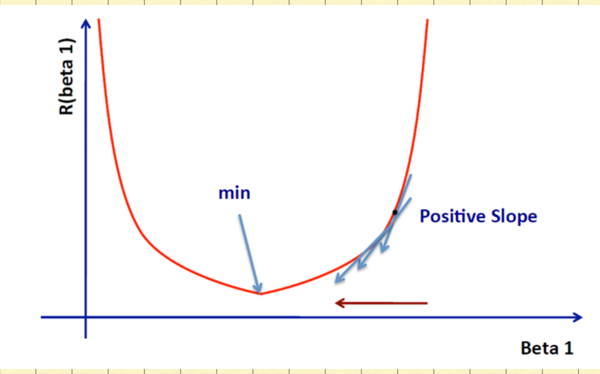
We have that: $\frac{\partial^2 R}{\partial \beta} = \frac{1}{n}X^T X$ $\qquad \frac{\partial R}{\partial \beta} = -\frac{1}{n}X^T(y - X\beta)$

is positive definite which ensures that $\beta$ is a minimum. We solve:

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

The unique solution is: $\boxed{\beta = (X^T X)^{-1} X^T y}$

# Gradient descent




Gradient Descent is an optimization method.

Repeat until convergence:

Update **simultaneously** all $\beta_j$ for ($j$ = 0 and $j$ = 1)

$$\beta_0 := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} R(\beta_0, \beta_1)$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} R(\beta_0, \beta_1)$$

**$\alpha$ is a learning rate.**

In the linear case:
$$\frac{\partial R}{\partial \beta_0} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) \times (-1)$$

$$\frac{\partial R}{\partial \beta_1} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) \times (-x_i)$$

Repeat until convergence:

Update **simultaneously** all $\beta_j$ for ($j$ = 0 and $j$ = 1)

$$\beta_0 := \beta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)$$

$$\beta_1 := \beta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)(x_i)$$

# Pros & cons

**Analytical approach: Normal Equation**

+ No need to specify a convergence rate or iterate.
- Works only if $X^T X$ is invertible
- Very slow if d is large $O(d^3)$ to compute $(X^T X)^{-1}$

**Iterative approach: Gradient Descent**

+ Effective and efficient even in high dimensions.
- Iterative (sometimes need many iterations to converge).
- Needs to choose the rate $\alpha$.

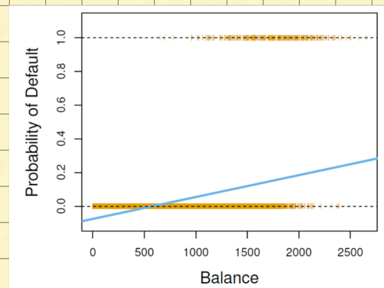
# Practical considerations

1. **Scaling**: Bring your features to a similar scale, e.g., $x_i := \frac{x_i - \mu_i}{stdev(x_i)}$
2. **Learning rate**: Don't use a rate that is too small or too large.
3. **R should decrease** after each iteration.
4. **Declare convergence** if it start decreasing by less $\epsilon$
5. When $X^T X$ is not **invertible**?
   a) Too many features as compared to the number of examples (e.g., 50 examples and 500 features)
   b) Features linearly dependent: e.g., weight in pounds and in kilo.

# Classification

- We can't predict Credit Card Default with any certainty. Suppose we want to predict how likely is a customer to default. That is output a probability between 0 and 1 that a customer will default.
- It makes sense and would be suitable and practical.
- In this case, the output is real (regression) but is bounded (classification).

$$P(y|x) = P(\text{default} = \text{yes}|\text{balance})$$




$$y = f(x) = \beta_0 + \beta_1 x$$

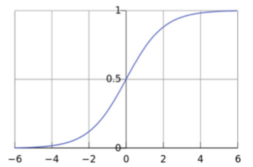
$$\text{Default} = \beta_0 + \beta_1 \times \text{Balance}$$

We want $0 \le f(x) \le 1$; $f(x) = P(y = 1|x)$

We use the sigmoid function: $g(z) = \dfrac{e^z}{1+e^z} = \dfrac{1}{1+e^{-z}}$ 激活函数

$g(z) \to 1$ when $z \to +\infty$

$g(z) \to 0$ when $z \to -\infty$




# Logistic Regression

$$g(\beta_0 + \beta_1 x) = \frac{e^{(\beta_0+\beta_1 x)}}{1 + e^{(\beta_0+\beta_1 x)}}$$

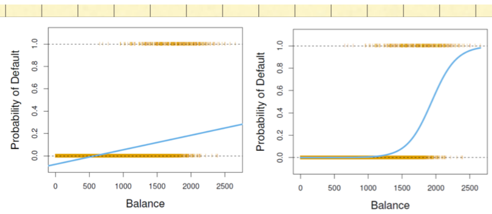
$$\boxed{\text{New } f(x) = g(\beta_0 + \beta_1 x)}$$

In general:

$$f(x) = g\left(\sum_{j=1}^{d} \beta_j x_j\right)$$

In other words, cast the output to bring the linear function quantity between 0 and 1.

Note: One can use other S-shaped functions.




Logistic regression is not a regression method but a classification method!

逻辑回归是分类方法

How to make a prediction?

- Suppose $\beta_0$ = -10.65 and $\beta_1$ = 0.0055. What is the probability of default for a customer with $1,000 balance?

$$P(default = yes|balance = 1000) = \frac{1}{1 + e^{10.65 - 0.0055*1000}}$$

$$P(default = yes|balance = 1000) = 0.00576$$

- To predict the class:

$$\text{If} g(z) \ge 0.5 \text{ predict } y = 1 \quad (z \ge 0)$$

$$\text{If} g(z) < 0.5 \text{ predict } y = 0 \quad (z < 0)$$

How to find the $\beta$'s?

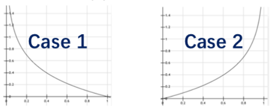
$$R(\beta) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(f(x) - y)^2$$

$$Loss = \frac{1}{2}(f(x) - y)^2$$

- Remember, $f(x)$ is now the logistic function so the $(f(x) - y)^2$ is not the quadratic function we had when $f$ was linear.
- Cost/risk is a complicated non-linear function!
- Many local optima, hence Gradient Descent will not find the global optimum!
- We need a different function that is convex.

New Convex function: $Cost(f(x), y) = \begin{cases} -log(f(x)) & \text{if } y = 1 \\ -log(1 - f(x)) & \text{if } y = 0 \end{cases}$

1. If $y = 1$ if the prediction $f(x) = 1$ then cost $= 0$
2. If $y = 1$ if the prediction $f(x) = 0$ then cost $\rightarrow \infty$
3. If $y = 0$ if the prediction $f(x) = 0$ then cost $\rightarrow 0$
4. If $y = 0$ if the prediction $f(x) = 1$ then cost $= \infty$




Nice convex functions!

Let's combine them in a compact function (because $y = 0$ or $y = 1$!):

$$Loss(f(x), y) = -ylogf(x) - (1 - y)log(1 - f(x))$$

$$R(\beta) = -\frac{1}{n}[\sum_{i=1}^{n} ylogf(x) + (1 - y)log(1 - f(x))]$$

# Gradient Descent

Repeat {

Simultaneously update for all $\beta$'s

$$\beta_j := \beta_j - \alpha\frac{\partial}{\partial\beta_j}R(\beta)$$

}

After some calculus:

Repeat {

Simultaneously update for all $\beta$'s

$$\beta_j := \beta_j - \alpha\sum_{i=1}^{n}(f(x) - y)x_j$$

}

Note: Same as linear regression BUT with the new function $f$.