Week 5
Linear Algorithms

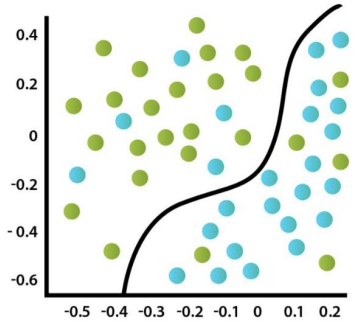
Content

- Linear Algorithms Introduction
- Linear Regression
- Logistic Regression
- Example: Applications
- Python: Linear Algorithms Practice

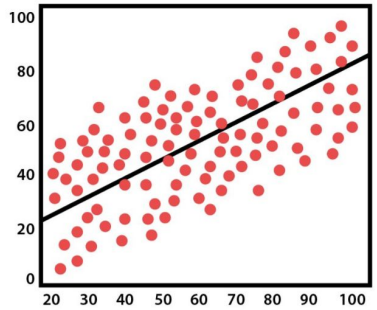
Recap: Supervised Learning

Supervised machine learning models try to solve Regression Problems and Classification Problems

- **Classification**
  - Classification is used when our goal is to categorize data into predefined classes or labels. It's like sorting things into different buckets based on their characteristics.
  - Algorithms
    - **Logistic Regression**
    - Decision Trees
    - Random Forests
    - Support Vector Machines (SVM)
- **Regression**
  - Regression models predict continuous values (real numbers). They're ideal for problems where the outcome is not discrete but lies on a continuous scale.
  - Algorithms
    - **Linear Regression**
    - Polynomial Regression
    - Support Vector Regression (SVR)



CLASSIFICATION



REGRESSION

Linear Algorithms Introduction

Linear Regression Example: House Price Prediction (correlation matrix)

Suppose you're a real estate agent, and you want to predict house prices based on features like square footage, number of bedrooms, and number of bathrooms. You collect data on various houses, including their features and actual selling prices. By using linear regression, you can create a model that estimates the price of a house based on these features. The equation might look like this:

$$\text{Price} = \beta 0 + \beta 1 \cdot \text{Square Footage} + \beta 2 \cdot \text{Bedrooms} + \beta 3 \cdot \text{Bathrooms} + \ldots + \beta p \cdot \text{Other Features}$$

HS No.	Price	Square Footage	Bedrooms	Bathrooms	Other Features
1	23455678	1908	3	2	.....
2	34560912	2480	2	1	.....
3	98537102	7800	4	2	.....
4	37865712	2901	2	1	.....
5	67901638	5678	4	2	.....
6	99075192	8012	5	3	.....
7	45619203	3201	3	2	.....

Linear Algorithms Introduction

Linear Regression Example: House Price Prediction

- Price = a0 · Square Footage + b0
- Price = a1 · Bedrooms + b1
- Price = a2 · Bathrooms + b2
- ……

A simple linear regression model: $y = a \cdot x + b$

$$\text{Price} = \beta 0 + \beta 1 \cdot \text{Square Footage} + \beta 2 \cdot \text{Bedrooms} + \beta 3 \cdot \text{Bathrooms} + \ldots + \beta p \cdot \text{Other Features}$$

Linear Algorithms Introduction

Logistic Regression Example: University Acceptance

Imagine you're a college admissions officer. You want to predict the probability that a student will get accepted into a certain university based on their GPA and ACT score. Since acceptance is a categorical outcome (either accepted or not accepted), you'd use logistic regression. The equation for logistic regression is:

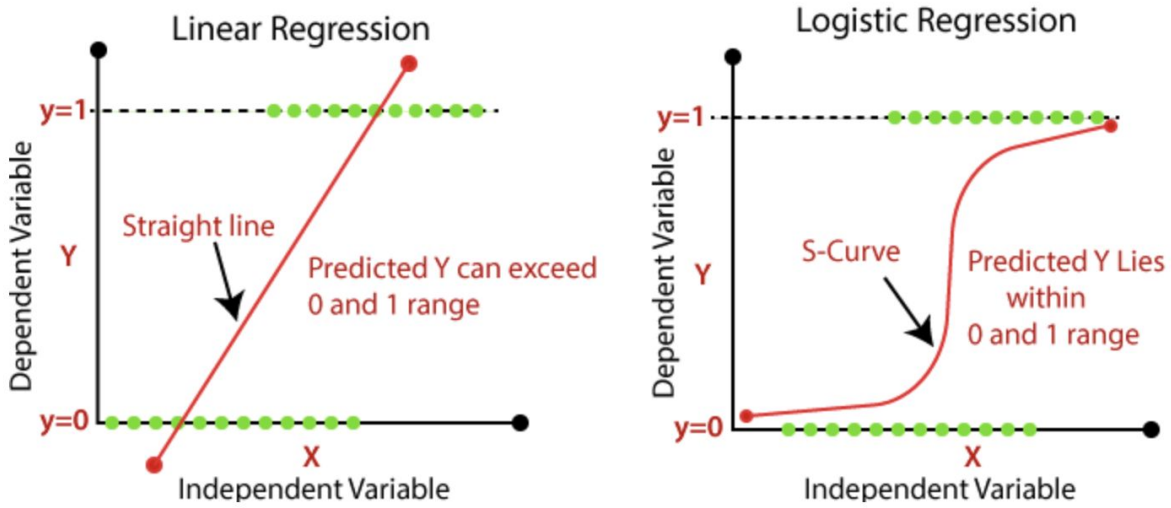
$$p(X) = \frac{e^{\beta_0 + \beta_1 \cdot \text{GPA} + \beta_2 \cdot \text{ACT Score} + \ldots}}{1 + e^{\beta_0 + \beta_1 \cdot \text{GPA} + \beta_2 \cdot \text{ACT Score} + \ldots}}$$

Logistic regression is used for **binary classifications.**
Logistic regression predicts **categorical target variable**.

Student ID	GPA	ACT Score	Accepted
1	3.5	28	1
2	3.8	32	1
3	3.2	25	0
4	3.9	34	1
5	3.6	30	1
…	…	…	…

Linear Algorithms Introduction

The difference between linear regression and logistic regression.

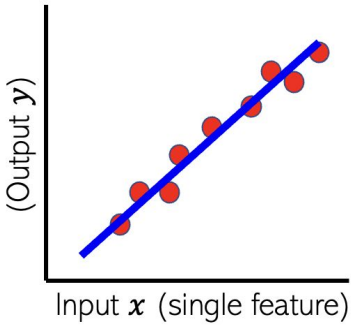


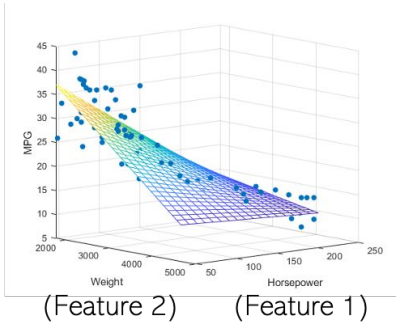
Linear Regression

Overview of Linear Regression

Linear Regression is a supervised machine learning algorithm that models the linear relationship between a **dependent** variable (also known as the target variable) and one or more **independent** features (also called predictor variables).

It aims to find the **best-fitting** straight line (or hyperplane in higher dimensions) that **predicts** the continuous output variable based on the input features.





Linear regression is like fitting a line or (hyper) plane to a set of points.

The line/plane must also predict outputs of the unseen (test) inputs well.

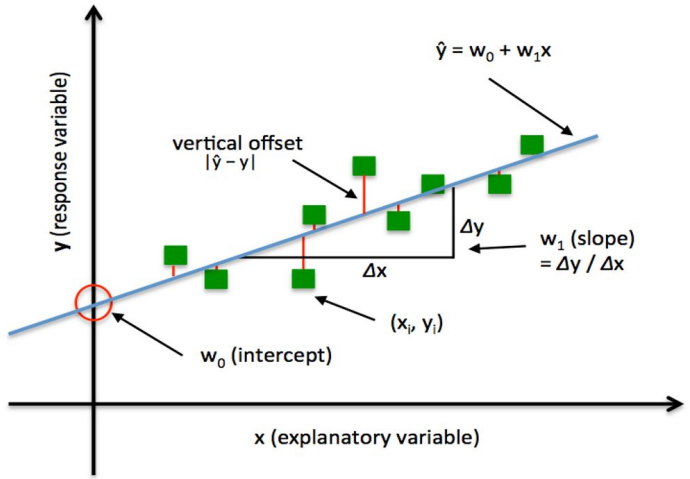
Linear Regression

Simple Linear Regression

- The base model for all statistical machine learning.
- There's only one independent feature.
- The equation for simple linear regression is:

$$y = \omega_0 + \omega_1 x + \varepsilon$$

  - **y** represents the dependent variable, the value we are trying to predict.
  - **x** represents the independent variable, a feature data variable.
  - ω_0 is the intercept (where the line crosses the y-axis).
  - ω_1 is the slope of the line.
  - **ε** represents random error (due to noise or unexplained factors).



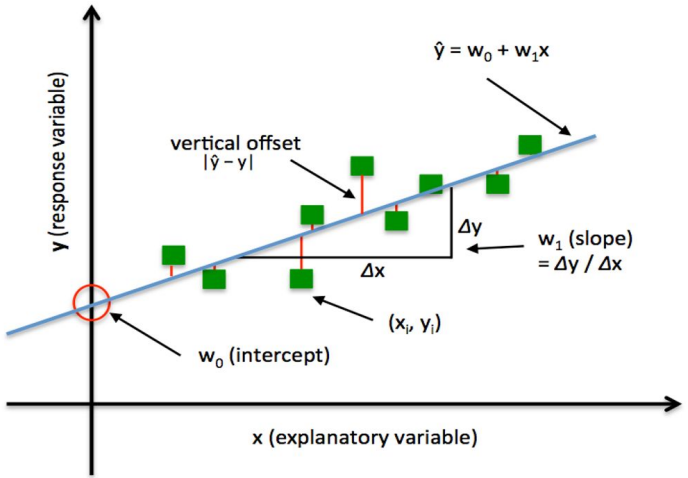
Linear Regression

Solving the regression problem

- Goal: find $\{\boldsymbol{\omega_0}, \boldsymbol{\omega_1}\}$ that minimize deviations from the predictor line

$$\arg\min_{w_0,w_1} \sum_{i}^{n} (y_i - w_0 - w_1 x_i)^2$$

- Q: How do we do it?
  - Iterate over all possible w values along the two dimensions?
    - Yes and No
    - No, we can do this in closed form with just plain calculus
    - Yes, same, but smarter… (See next slide)



Linear Regression

Multiple Linear Regression

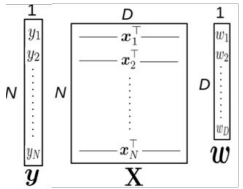
- There's only one independent feature.
- The equation for simple linear regression is: $y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots\ldots + \omega\square x\square + \varepsilon$
  - Input: $x_i \in \mathbb{R}^n, \ i = 1, \ldots, m$
  - Output: $y_i \in \mathbb{R}$
  - Model Parameters: $\boldsymbol{\omega} \in \mathbb{R}^k$
  - Predicted Output: $\hat{y}_i \in \mathbb{R}$

- More generally
  - **Given**: Training data with N input-output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}, \boldsymbol{x_n} \in \mathbb{R}^D, y_n \in \mathbb{R}$
  - **Goal**: Learn a model to predict the output for new test inputs

$$y_n \approx f(\boldsymbol{x}_n) = \boldsymbol{w}^\top \boldsymbol{x}_n \quad (n = 1,2, \ldots, N)$$

  - Let's write the total error or "loss" of this model over the training data as

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(y_n, \boldsymbol{w}^\top \boldsymbol{x}_n)$$



$\ell(y_n, \boldsymbol{w}^\top \boldsymbol{x}_n)$ measures the prediction error or "loss" or "deviation" of the model on a single training input $(\boldsymbol{x}_n, y_n)$

Goal of learning is to find the **w** that minimizes this loss + does well on test data

Linear Regression

Loss functions

- Want a model that performs “well” on the data we have

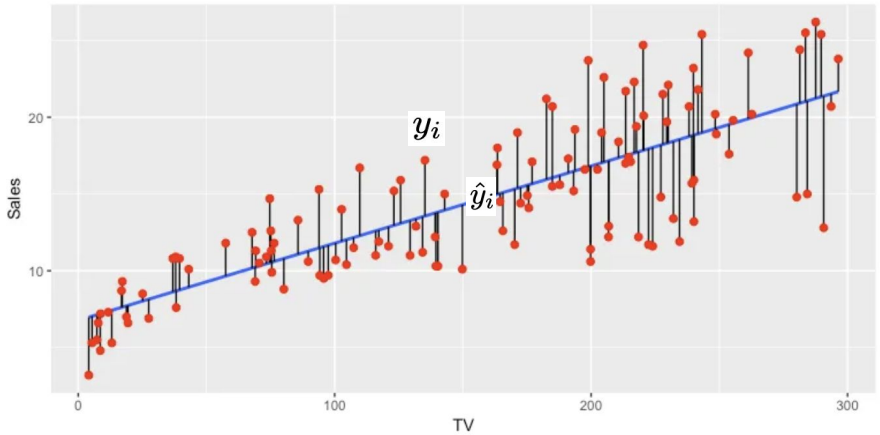
$$\hat{y}_i \approx y_i, \quad \forall i$$

- We measure “closeness” of $\hat{y}_i$ and y_i using loss function

$$\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$$

- Example: squared loss

$$\ell(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$$



Linear Regression

Parameter estimation via calculus

- Set the partial derivatives to zero

$$\frac{\partial \epsilon^2}{\partial w_0} = \sum_i^n -2(y_i - w_0 - w_1 x_i) = 0$$
$$\frac{\partial \epsilon^2}{\partial w_1} = \sum_i^n -2x_i(y_i - w_0 - w_1 x_i) = 0$$

- Simplifying

$$w_0 = \bar{y} - w_1 \bar{x}$$
$$w_1 = \frac{n\sum_i^n x_i y_i - \sum_i^n x_i \sum_i^n y_i}{n\sum_i^n x_i x_i - \sum_i^n x_i \sum_i^n x_i}$$

Define the line of best fit line as $\hat{Y}_i = a + Bx_i$

Cost function: Commonly, the cost function is computed as the **average** (mean or median) of the individual losses.

$$S = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

$$S = \sum_{i=1}^{n} (Y_i - a - Bx_i)^2$$

Goal: To minimize our cost function, S, we must find where the first derivative of S is equal to 0 with respect to **a** and **B.** The closer **a** and **B** are to 0, the less the total error for each point is.

Linear Regression

Finding model parameters, and optimization

- Given sum of costs over all input/output pairs, θ: model parameters

$$J(\theta) = \sum_{i=1}^{m} \ell(\hat{y}_i, y_i) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2$$

- Write our objective formally as $\underset{\theta}{\text{minimize}} \quad J(\theta)$
- Q: How do we optimize a function?
  - Gradient descent
  - Search algorithm: Start with an initial guess for θ. Keep changing θ (by a little bit) to reduce J(θ)
  - Gradient flow

Linear Regression

Gradient Descent

- Search algorithm: Start with an initial guess for θ. Keep changing θ (by a little bit) to reduce J(θ)

$$J(\theta) = \sum_{i=1}^{m} \ell(\hat{y}_i, y_i) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2$$

Gradient descent: $\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$, for all j

$$\begin{aligned}
\frac{\partial J}{\partial \theta_j} &= \frac{\partial \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2}{\partial \theta_j} = \sum_{i=1}^{m} \frac{\partial (\theta^T \phi(x_i) - y_i)^2}{\partial \theta_j} \\
&= \sum_{i=1}^{m} 2(\theta^T \phi(x_i) - y_i) \frac{\partial (\theta^T \phi(x_i) - y_i)}{\partial \theta_j} \\
&= \sum_{i=1}^{m} 2(\theta^T \phi(x_i) - y_i) \phi(x_i)_j
\end{aligned}$$

- Repeat until "convergence": $\theta_j = \theta_j - \alpha \sum_{i=1}^{m} 2(\theta^T \phi(x_i) - y_i) \phi(x_i)_j, \text{ for all } j$
- Example: Stochastic gradient descent

Linear Regression

Gradient Descent

Let's write $J(\theta)$ a little more compactly using matrix notation; define

$$\Phi \in \mathbb{R}^{m \times k} = \begin{bmatrix} — & \phi(x_1)^T & — \\ — & \phi(x_2)^T & — \\ & \vdots & \\ — & \phi(x_m)^T & — \end{bmatrix}, \quad y \in \mathbb{R}^m = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

then

$$J(\theta) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2 = \|\Phi\theta - y\|_2^2$$

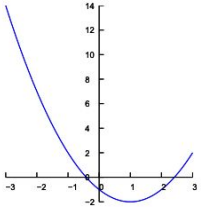
($\|z\|_2$ is ℓ_2 norm of a vector: $\|z\|_2 \equiv \sqrt{\sum_{i=1}^m z_i^2} = \sqrt{z^T z}$)

Called least-squares objective function

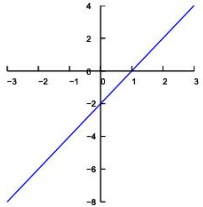
Linear Regression

Gradient Descent

- Q: How do we optimize a function? 1-D case ($\theta \in \mathbb{R}$):



$J(\theta) = \theta^2 - 2\theta - 1$



$\frac{dJ}{d\theta} = 2\theta - 2$

$$\theta^\star \text{ minimum} \implies \left.\frac{dJ}{d\theta}\right|_{\theta^\star} = 0$$
$$\implies 2\theta^\star - 2 = 0$$
$$\implies \theta^\star = 1$$

- Multi-variate case: $\theta \in \mathbb{R}^k$, $J : \mathbb{R}^k \to \mathbb{R}$

$$\text{Generalized condition:} \quad \nabla_\theta J(\theta)|_{\theta^\star} = 0$$

Linear Regression

Gradient Descent

$\nabla_\theta J(\theta)$ denotes *gradient* of J with respect to θ

$$\nabla_\theta J(\theta) \in \mathbb{R}^k \equiv \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \\ \vdots \\ \frac{\partial J}{\partial \theta_k} \end{bmatrix}$$

Some important rules and common gradient

$$\nabla_\theta(af(\theta) + bg(\theta)) = a\nabla_\theta f(\theta) + b\nabla_\theta g(\theta), \quad (a, b \in \mathbb{R})$$
$$\nabla_\theta(\theta^T A\theta) = (A + A^T)\theta, \quad (A \in \mathbb{R}^{k\times k})$$
$$\nabla_\theta(b^T\theta) = b, \quad (b \in \mathbb{R}^k)$$

Optimizing least-squares objective

$$\begin{aligned} J(\theta) &= \|\Phi\theta - y\|_2^2 \\ &= (\Phi\theta - y)^T(\Phi\theta - y) \\ &= \theta^T\Phi^T\Phi\theta - 2y^T\Phi\theta + y^Ty \end{aligned}$$

Using the previous gradient rules

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta(\theta^T\Phi^T\Phi\theta - 2y^T\Phi\theta + y^Ty) \\ &= \nabla_\theta(\theta^T\Phi^T\Phi\theta) - 2\nabla_\theta(y^T\Phi\theta) + \nabla_\theta(y^Ty) \\ &= 2\Phi^T\Phi\theta - 2\Phi^Ty \end{aligned}$$

Setting gradient equal to zero

$$2\Phi^T\Phi\theta^\star - 2\Phi^Ty = 0 \iff \theta^\star = (\Phi^T\Phi)^{-1}\Phi^Ty$$

known as the normal equations

Linear Regression

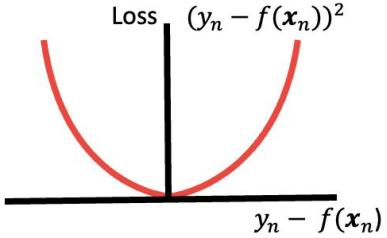
Alternative loss functions

- Many possible loss functions for regression problems

Note: Choice of loss function usually depends on the nature of the data. Also, some loss functions result in easier optimization problem than others.

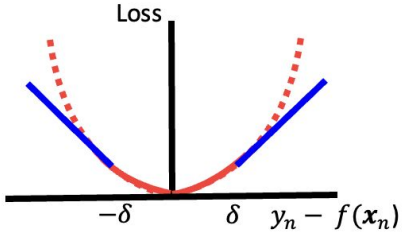
Squared loss

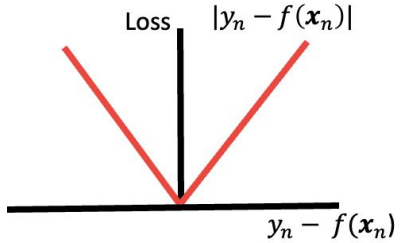
Very commonly used for regression. Leads to an easy-to-solve optimization problem



Huber loss

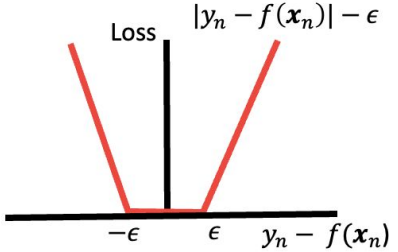
Squared loss for small errors (say up to δ); absolute loss for larger errors. Good for data with outliers.





Absolute loss

Grows more slowly than squared loss. Thus better suited when data has some outliers (inputs on which model makes large errors)



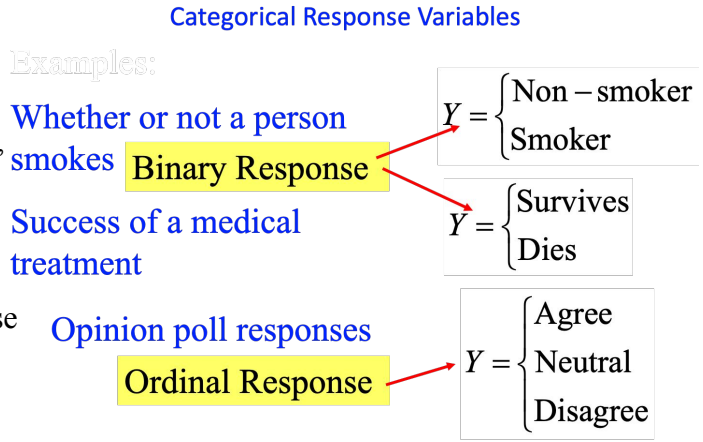
ϵ-insensitive loss (a.k.a. Vapnik loss)

Zero loss for small errors (say up to ϵ); absolute loss for larger errors.

Logistic Regression

Why use logistic regression?

- Limited Dependent Variables
  - There are many important research topics for which the dependent variable is "limited."
  - The dependent variable may be discrete, and could be binomial or multinomial. That is, the dependent variable is limited. In such cases, we need a different approach.
  - Discrete dependent variables are a special case of limited dependent variables. The Logit model we look at here is a discrete dependent variable model. Such models are also often called qualitative response (QR) models.
- For example: voting, morbidity or mortality, and participation data is not continuous or distributed normally.
- Binary logistic regression is a type of regression analysis where the dependent variable is a dummy variable: coded 0 (did not vote) or 1(did vote)



Logistic Regression

Why can't we use linear regression for binary outcomes?

- **Logistic regression is the type of regression we use for a binary response variable that follows a Bernoulli distribution.**
  - The relationship between X and Y is not linear.
  - The response Y is not normally distributed.
  - The variance of a Bernoulli random variable depends on its expected value p_x.
  - Fitted value of Y may not be 0 or 1, since linear models produce fitted values in (-∞, +∞)

Review of Bernoulli Distribution

- $Y \sim Bernoulli(p)$ takes values in $\{0, 1\}$,
  - e.g. a coin toss
- $Y = 1$ for a success, $Y = 0$ for failure,
- $p =$ probability of success, i.e. $p = P(Y = 1)$,
  - e.g. $p = \frac{1}{2} = P(\text{heads})$
- Mean is p, Variance is $p(1 - p)$.

Bernoulli probability density function (pdf):

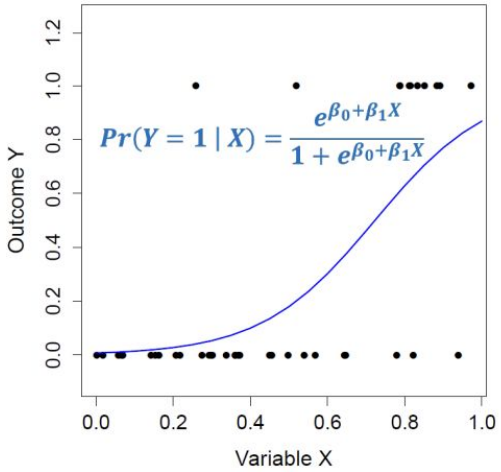
$$f(y; p) = \begin{cases} 1 - p & \text{for } y = 0 \\ p & \text{for } y = 1 \end{cases}$$
$$= p^y (1 - p)^{1-y}, \quad y \in \{0, 1\}$$

Logistic Regression

Logistic regression model

- **A regression model for binary data**
  - Instead of modeling Y, model P(Y = 1|X), i.e. probability that Y = 1 conditional on covariates.
  - Use a function that constrains probabilities between 0 and 1.
- **Regression model**
  - Let Y be a binary outcome and X a covariate/predictor.
  - We are interested in modeling px = P(Y = 1|X = x), i.e. the probability of a success for the covariate value of X = x. Define the logistic regression model as

$$logit(p_X) = \log\left(\frac{p_X}{1-p_X}\right) = \beta_0 + \beta_1 X$$



$\log\left(\frac{p_X}{1-p_X}\right)$ is called the **logit** function

$p_X = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$

$\lim_{x\to-\infty} \frac{e^x}{1+e^x} = 0$ and $\lim_{x\to\infty} \frac{e^x}{1+e^x} = 1$, so $0 \le p_x \le 1$.

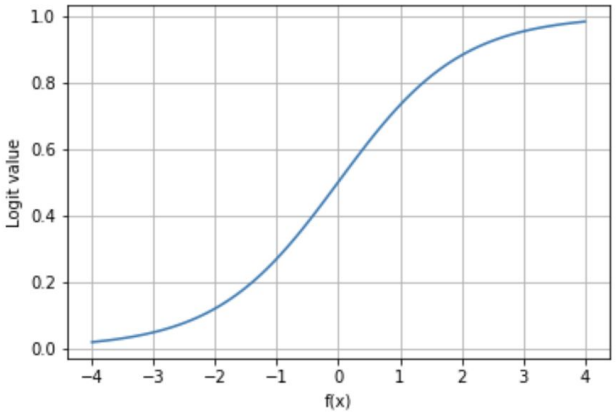
Logistic Regression

The Logistic Function

$$y = \frac{1}{1 + e^{-f(x_1, x_2, \ldots, x_n)}} \in (0, 1)$$

where

$$f(x_1, x_2, \ldots, x_n) = a_0 + a_1 x_1 + \ldots + a_n x_n \in (-\infty, +\infty)$$



Sigmoid Function

Logistic Regression

Likelihood equations for logistic regression

- Assume $Y_i|X_i \sim Bernoulli(p_{X_i})$ and
  $f(y_i|p_{x_i}) = p_{x_i}^{y_i} \times (1-p_{x_i})^{1-y_i}$

  Binomial likelihood: $\mathcal{L}(p_x|Y,X) = \prod_{i=1}^{N} p_{x_i}^{y_i}(1-p_{x_i})^{1-y_i}$
- Binomial log-likelihood:

  $\ell(p_x|Y,X) = \sum_{i=1}^{N} \left\{ y_i \log\left(\frac{p_{x_i}}{1-p_{x_i}}\right) + \log(1-p_{x_i}) \right\}$

  Logistic regression log-likelihood:

  $\ell(\beta|X,Y) = \sum_{i=1}^{N} \left\{ y_i(\beta_0 + \beta_1 x_i) - \log(1+e^{\beta_0+\beta_1 x_i}) \right\}$

  No closed form solution for Maximum Likelihood Estimates of β values.

  Numerical maximization techniques required.

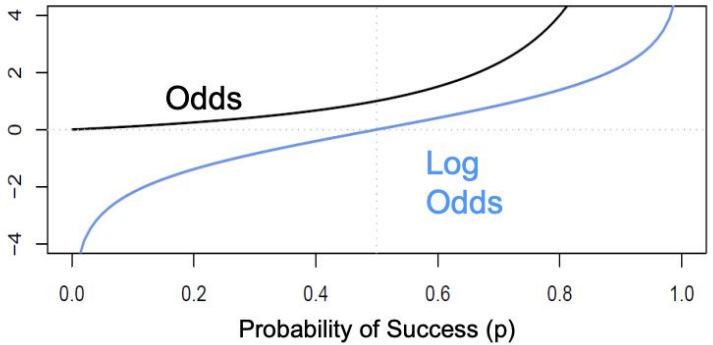
Logistic Regression

Logistic regression terminology

- Let p be the probability of success. Recall that

$$logit(p_X) = \log\left(\frac{p_X}{1-p_X}\right) = \beta_0 + \beta_1 X$$

  - Then $\frac{p_X}{1-p_X}$ is called the odds of success,
  - $\log\left(\frac{p_X}{1-p_X}\right)$ is called the log odds of success.



Logistic Regression

- **Another motivation for logistic regression**
  - Let p be the probability of success. Recall that I Since p $\in$ [0, 1], the log odds is log[p/(1 $\neq$ p)] $\in$ (-∞, +∞).
  - So while linear regression estimates anything in (-∞, +∞),
  - logistic regression estimates a proportion in [0, 1].

- **Review of probabilities and odds**

Measure	Min	Max	Name
$P(Y=1)$	0	1	"probability"
$\frac{P(Y=1)}{1-P(Y=1)}$	0	∞	"odds"
$\log\left[\frac{P(Y=1)}{1-P(Y=1)}\right]$	$-\infty$	∞	"log-odds" or "logit"

The odds of an event are defined as

$$\text{odds}(Y=1) = \frac{P(Y=1)}{P(Y=0)} = \frac{P(Y=1)}{1-P(Y=1)} = \frac{p}{1-p}$$

$$\Rightarrow p = \frac{\text{odds}(Y=1)}{1+\text{odds}(Y=1)}.$$

Logistic Regression

- **Review of odds ratio**
  - Odds Ratios (OR) can be useful for comparisons.
  - Suppose we have a trial to see if an intervention T reduces mortality, compared to a placebo, in patients with high cholesterol. The odds ratio is

$$OR = \frac{\text{odds(death|intervention T)}}{\text{odds(death|placebo)}}$$

  - The OR describes the benefits of intervention T:
    - OR< 1: the intervention is better than the placebo since odds(death|intervention T) < odds(death|placebo)
    - OR= 1: there is no dierence between the intervention and the placebo
    - OR> 1: the intervention is worse than the placebo since odds(death|intervention T) > odds(death|placebo)

		Outcome status +	Outcome status −
Exposure status	+	a	b
	−	c	d

$$\text{OR} = \frac{\text{Odds of being a case given exposed}}{\text{Odds of being a case given unexposed}} = \frac{\frac{a}{a+b}/\frac{b}{a+b}}{\frac{c}{c+d}/\frac{d}{c+d}} = \frac{a/c}{b/d} = \frac{ad}{bc}.$$

Logistic Regression

- **Interpretation of logistic regression parameters**
  - β_0 is the log of the odds of success at zero values for all covariates.
  - $\frac{e^{\beta_0}}{1+e^{\beta_0}}$ is the probability of success at zero values for all covariates
  - Interpretation of $\frac{e^{\beta_0}}{1+e^{\beta_0}}$ depends on the sampling of the dataset
    - Population cohort: disease prevalence at X = x
    - Case-control: ratio of cases to controls at X = x
  - Slope β_1 is the increase in the log odds ratio associated with a one-unit increase in X:

$$\log\left(\frac{p_X}{1-p_X}\right) = \beta_0 + \beta_1 X$$

$$\beta_1 = (\beta_0 + \beta_1(X+1)) - (\beta_0 + \beta_1 X)$$

$$= \log\left(\frac{p_{X+1}}{1+p_{X+1}}\right) - \log\left(\frac{p_X}{1-p_X}\right) = \log\left\{\frac{\left(\frac{p_{X+1}}{1-p_{X+1}}\right)}{\left(\frac{p_X}{1-p_X}\right)}\right\}$$

and e^{β_1}=OR!.

    - If $\beta_1 = 0$, there is no association between changes in X and changes in success probability (OR= 1).
    - If $\beta_1 > 0$, there is a positive association between X and p (OR> 1).
    - If $\beta_1 < 0$, there is a negative association between X and p (OR< 1).
  - Interpretation of slope —1 is the same regardless of sampling.

Example: Applications

- **Linear Regression**
  - **Financial Forecasting**: Predicting stock prices using historical data.
  - **Real Estate**: Estimating house prices based on features like square footage, bedrooms, and bathrooms.
  - **Economics**: Modeling annual income based on weekly hours worked and years of education.
  - **Market Trends**: Analyzing trends and relationships in business data.
  - **Scientific Research**: Investigating relationships between variables.
- **Logistic Regression**
  - **Healthcare**: Predicting the likelihood of a patient developing a specific condition based on risk factors.
  - **Admissions**: Determining the probability of a student getting accepted into a university based on GPA and test scores.
  - **Business Risk Assessment**: Assessing the risk of certain events occurring.

Note: Linear regression is used to predict continuous outcomes based on one or more predictor variables. Logistic regression is used for binary classification tasks with categorical outcomes (e.g., yes/no, win/lose).

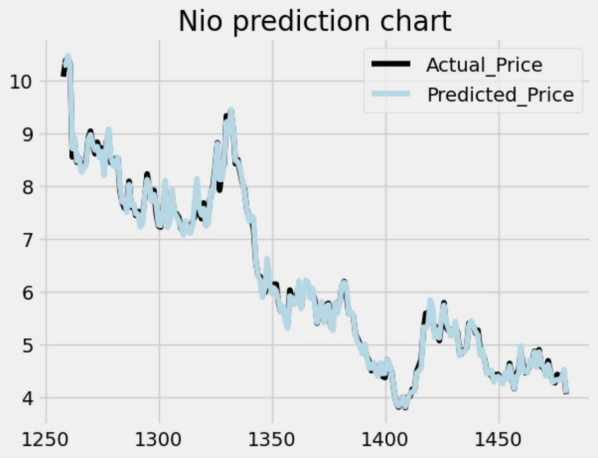
Python: Linear Algorithms Practice

- **Stock Prediction using Linear Regression Algorithm in Python (See Notebook)**
  - **Dataset: yfinance**
    - yfinance is an open-source Python library.
    - It provides easy access to financial data from Yahoo Finance.
    - You can retrieve data for stocks, bonds, currencies, and cryptocurrencies.
    - yfinance allows you to download historical market data.

Python: Linear Algorithms Practice

- **Stock Prediction using Linear Regression Algorithm in Python (See Notebook)**

```
from sklearn.linear_model import LinearRegression # type: ignore
from sklearn.metrics import confusion_matrix, accuracy_score # type: ignore
regression = LinearRegression()
regression.fit(train_x, train_y)
print("regression coefficient",regression.coef_)
print("regression intercept",regression.intercept_)
```

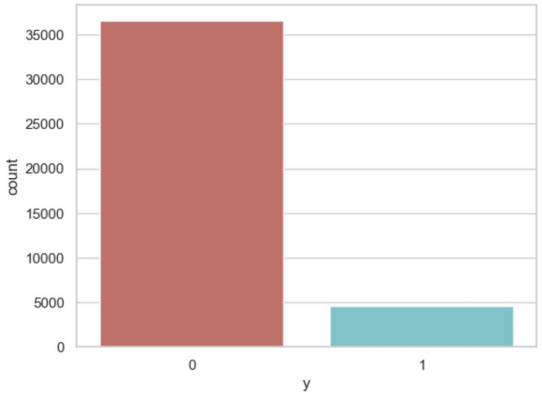


Python: Logistic Regression Practice

- **Predicting whether a customer will subscribe to Term Deposits using Logistic Regression Algorithm in Python (See Notebook)**
  - **Dataset: bank.csv**
    - The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution.
    - The classification goal is to predict if the client will subscribe (1/0) a term deposit (variable y).
    - This dataset provides the customer information. It includes 41188 records and 21 fields.

Bank client data:

- Age (numeric)
- Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
- Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)
- Education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
- Default: has credit in default? (categorical: 'no', 'yes', 'unknown')
- Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')
- Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')



Python: Logistic Regression Practice

- **Predicting whether a customer will subscribe to Term Deposits using Logistic Regression Algorithm in Python (See Notebook)**

Logistic Regression Model Fitting

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

✓ 0.2s

```
LogisticRegression()
```

```
y_pred = logreg.predict(X_test)

print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

✓ 0.0s

```
Accuracy of logistic regression classifier on test set: 0.90
```