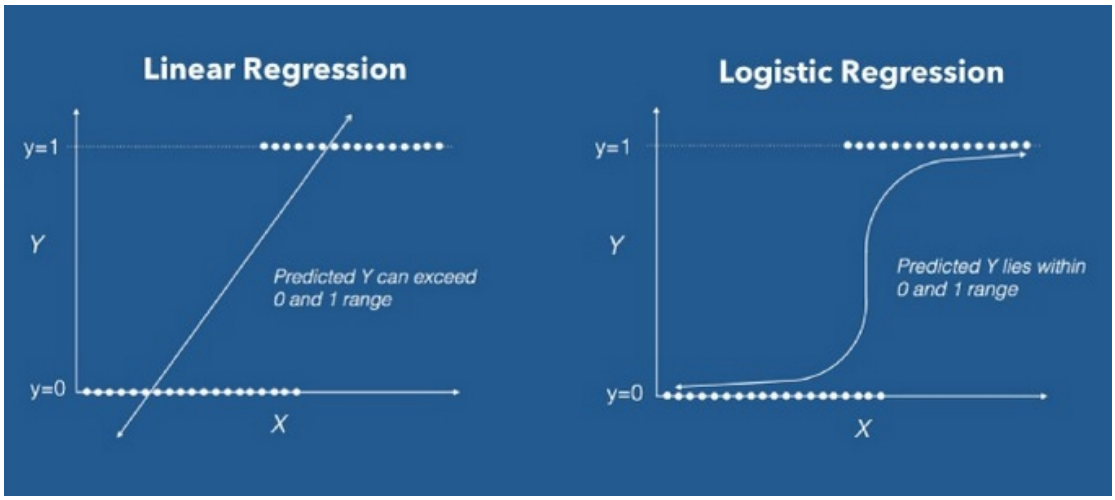
Basics of Logistics Regression

Логистическая регрессия — это алгоритм классификации, используемый для распределения наблюдений по дискретному набору классов. Некоторые из примеров проблем классификации: спам по электронной почте или не спам, онлайн-транзакции, мошенничество или не мошенничество, опухоль злокачественная или доброкачественная. Логистическая регрессия преобразует свои выходные данные, используя логистическую сигмовидную функцию, чтобы вернуть значение вероятности.

Какие бывают виды логистической регрессии:

1) Бинарный (например, опухоль злокачественная или доброкачественная)
2) Многолинейные функции failsClass (например, кошки, собаки или овцы)

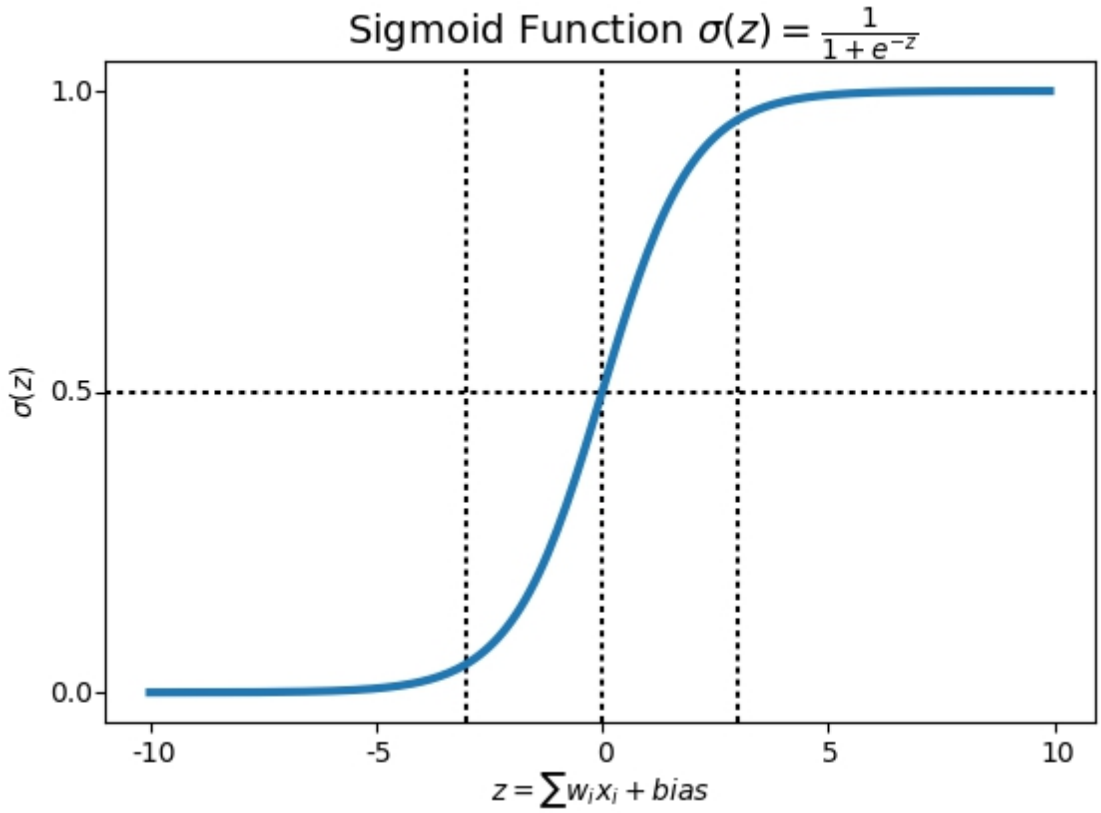
Логистическая регрессия — это алгоритм машинного обучения, который используется для задач классификации, это алгоритм прогнозирующего анализа, основанный на концепции вероятности.



Мы можем назвать логистическую регрессию моделью линейной регрессии, но логистическая регрессия использует более сложную функцию стоимости, эта функция стоимости может быть определена как «**сигмоидальная функция**» или также известна как «логистическая функция» вместо линейной функции.

Что такое сигмовидная функция?

Чтобы сопоставить прогнозируемые значения с вероятностями, мы используем сигмовидную функцию. Функция отображает любое реальное значение в другое значение от 0 до 1. В машинном обучении мы используем сигмоид для сопоставления прогнозов с вероятностями.



Гипотеза

При использовании линейной регрессии мы использовали формулу гипотезы, т.е.

$$h\Theta(x) = \beta_0 + \beta_1 X$$

Для логистической регрессии мы немного изменим его, т.е.

$$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$$

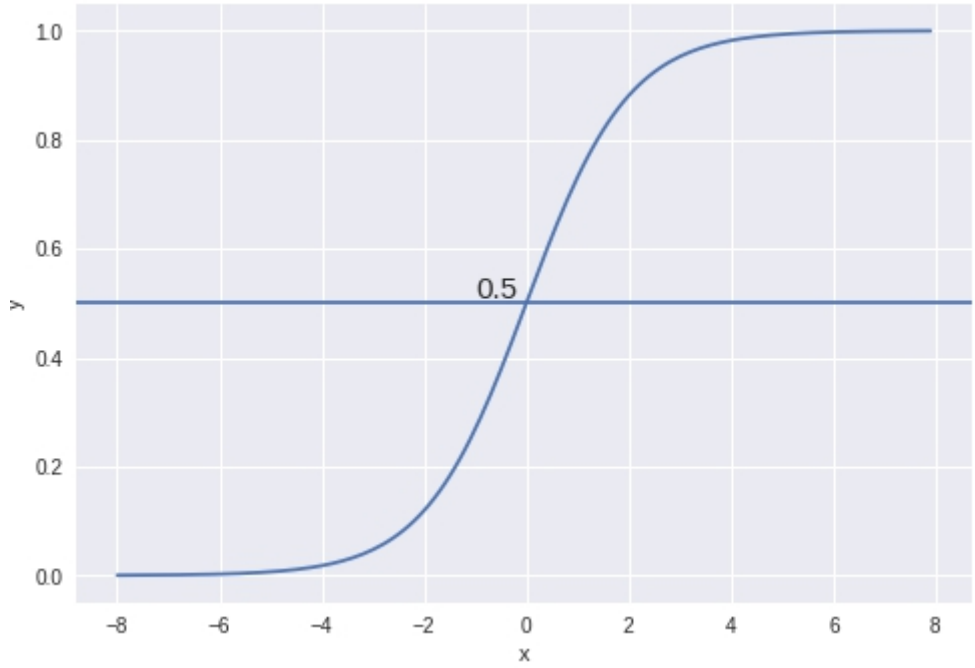
Мы ожидали, что наша гипотеза даст значения от 0 до 1.

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Граница принятия решения

Мы ожидаем, что наш классификатор даст нам набор выходных данных или классов на основе вероятности, когда мы пропустим входные данные через функцию прогнозирования и вернем оценку вероятности между 0 и 1.

Например, у нас есть 2 класса, возьмем их как кошек и собак (1 — собака, 0 — кошки). В основном мы решаем с пороговым значением, выше которого мы классифицируем значения в класс 1, а если значение опускается ниже порога, мы классифицируем его в классе 2.



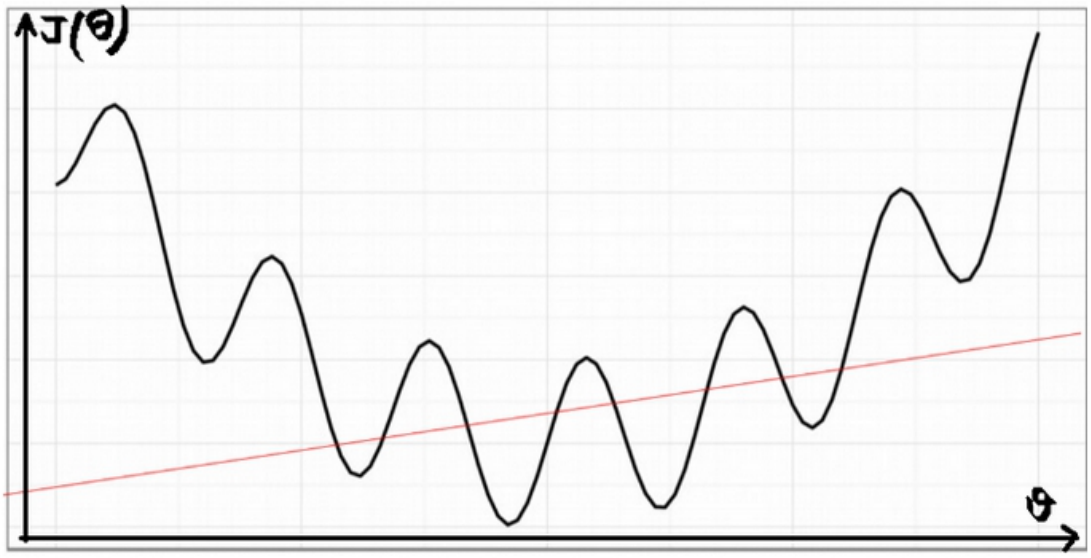
Как показано на приведенном выше графике, мы выбрали пороговое значение 0,5, если функция прогнозирования возвращает значение 0,7, то мы классифицируем это наблюдение как класс 1 (COC). Если бы наш прогноз вернул значение 0,2, мы бы классифицировали наблюдение как класс 2 (CAT).

Функция стоимости

Мы узнали о функции стоимости J(θ) в линейной регрессии, функция стоимости представляет цель оптимизации, т. е. мы создаем функцию стоимости и минимизируем ее, чтобы мы могли разработать точную модель с минимальной ошибкой.

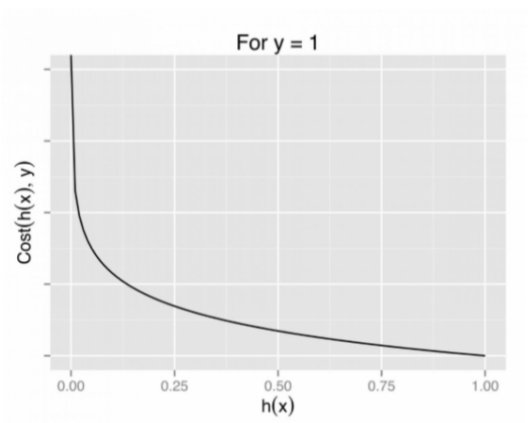
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

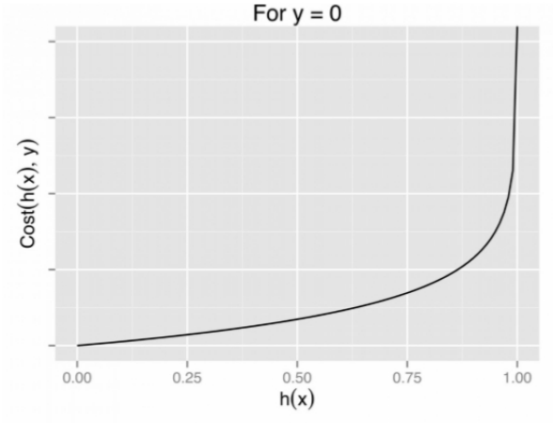
Если мы попытаемся использовать функцию стоимости линейной регрессии в «Логистической регрессии», то это будет бесполезно, поскольку в конечном итоге это будет невыпуклая функция со многими локальными минимумами, в которой будет очень трудно минимизировать стоимости и найти глобальный минимум.



Для логистической регрессии функция стоимости определяется как:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





Вышеупомянутые две функции могут быть сжаты в одну функцию, т.е.

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

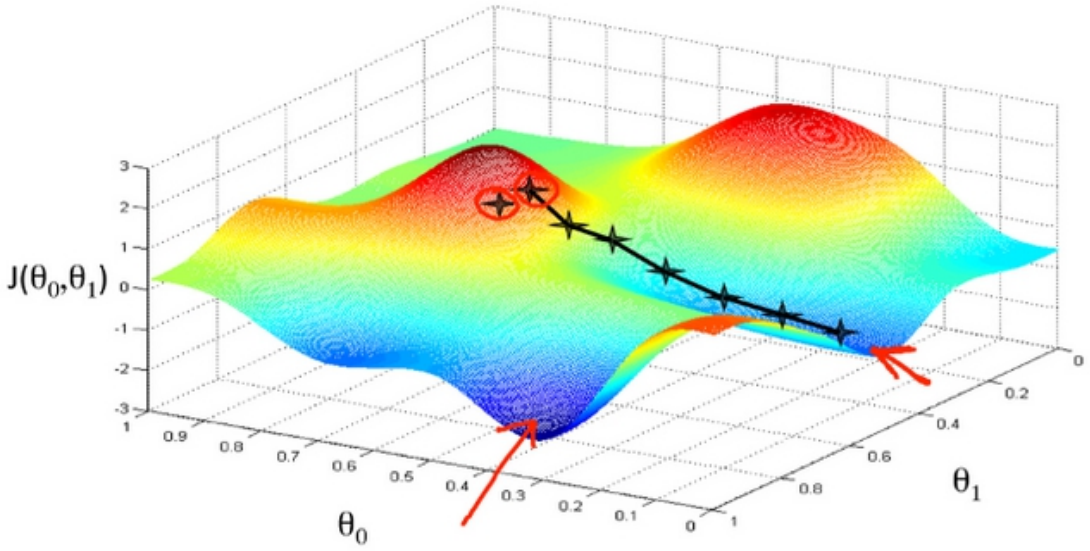
Градиентный спуск

Теперь возникает вопрос, как уменьшить себестоимость. Что ж, это можно сделать с помощью градиентного спуска. Основная цель градиентного спуска — минимизировать стоимость. т. е. мин J(θ).

Теперь, чтобы минимизировать нашу функцию стоимости, нам нужно запустить функцию градиентного спуска для каждого параметра, т.е.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Градиентный спуск имеет аналогию, в которой мы должны представить себя на вершине горной долины и остаться с завязанными глазами, наша цель - достичь подножия холма. Ощущение уклона местности вокруг вас — это то, что сделал бы каждый. Ну, это действие аналогично вычислению градиентного спуска, а выполнение шага аналогично одной итерации обновления параметров.


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1