# Регрессия как задача машинного обучения

## Линейная регрессия с одной переменной

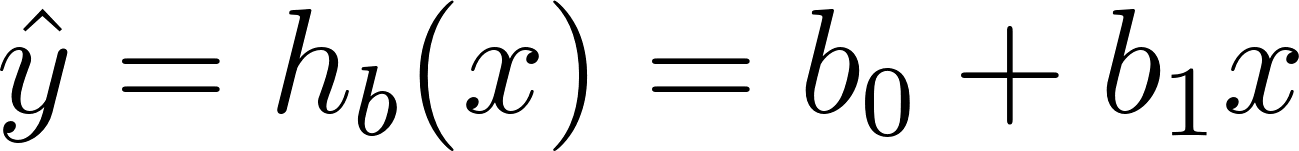
### Представление модели

Напомним, что в задачах регрессии мы принимаем входные переменные и пытаемся подогнать выход на непрерывную ожидаемую функцию результата. Линейная регрессия с одной переменной также известна как «парная линейная регрессия».

Одномерная линейная регрессия используется, когда вы хотите предсказать одно выходное значение *y*, зависящее от одного входного значения *x*. Мы проводим обучение с учителем, это означает, что мы уже имеем представление о том, какие значения выходной переменной соответствуют некоторым значениям входной переменной.

**Функция гипотезы**

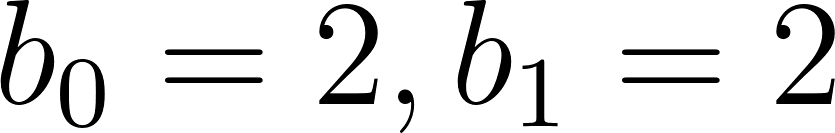
Наша прогностическая функция (функция гипотезы, модель) имеет общий вид:

[](about:blank)

Обратите внимание, что это похоже на уравнение прямой. в данном случае, мы пытаемся подобрать функцию *h(x)* таким образом, чтобы отобразить данные нам значения *x* в данные значения *y*.

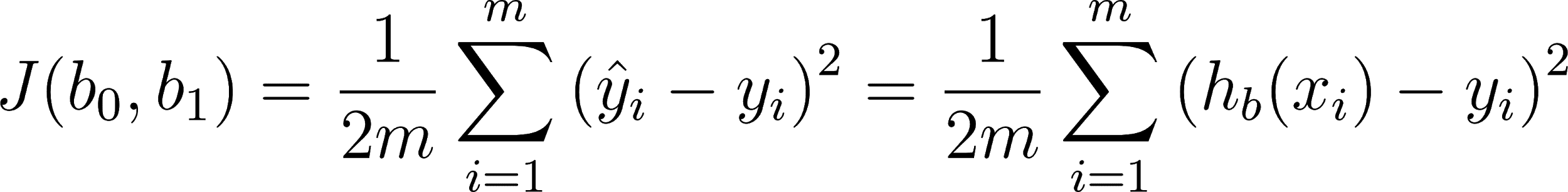
Допустим, мы имеем следующий обучающий набор данных:

| x | y |
| --- | --- |
| 0 | 4 |
| 1 | 7 |
| 2 | 7 |
| 3 | 8 |

Мы можем составить случайную гипотезу с параметрами [](about:blank). Тогда для входного значения *x=1, y=4*, что на 3 меньше данного. Задача регрессии состоит в нахождении таких параметров функции гипотезы, чтобы она отображала входные значения в выходные как можно более точно, или, другими словами, описывала линию, наиболее точно ложащуюся в данные точки на плоскости (x, y).

### Функция ошибки

Мы можем измерить точность нашей функции гипотезы, используя функцию ошибки. Для этого требуется средняя (фактически чуть усложненная версия среднего арифметического) всех результатов вычисления гипотезы с входами x по сравнению с фактическим выходом y.

[](about:blank)

По сути своей, это половина среднего квадрата разницы между прогнозируемым и фактическим значением выходной переменной.

Эту функцию называют «функцией квадрата ошибки» или «среднеквадратичной ошибкой» (mean squared error, MSE). Среднее значение уменьшено вдвое для удобства вычисления градиентного спуска, так как производная квадратичной функции будет отменять множитель 1/2.

Теперь мы можем конкретно измерить точность нашей предсказывающей функции по сравнению с правильными результатами, которые мы имеем, чтобы мы могли предсказать новые результаты, которых у нас нет.

Если мы попытаемся представить это наглядно, наш набор данных обучения будет разбросан по плоскости x-y. Мы пытаемся подобрать прямую линию, которая проходит через этот разбросанный набор данных. Наша цель - получить наилучшую возможную линию. Лучшая линия будет такой, чтобы средние квадраты вертикальных расстояний рассеянных точек от линии были наименьшими. В лучшем случае линия должна проходить через все точки нашего набора данных обучения. В таком случае значение J будет равно 0.

### Метод градиентного спуска

Таким образом, у нас есть наша функция гипотезы, и у нас есть способ оценить, насколько хорошо конкретная гипотеза вписывается в данные. Теперь нам нужно подобрать параметры в функции гипотезы. Вот где приходит на помощь метод градиентного спуска.

Представьте себе, что мы нарисуем нашу функцию гипотезы на основе ее параметров b0 и b1 (фактически мы представляем график функции стоимости как функцию оценок параметров).

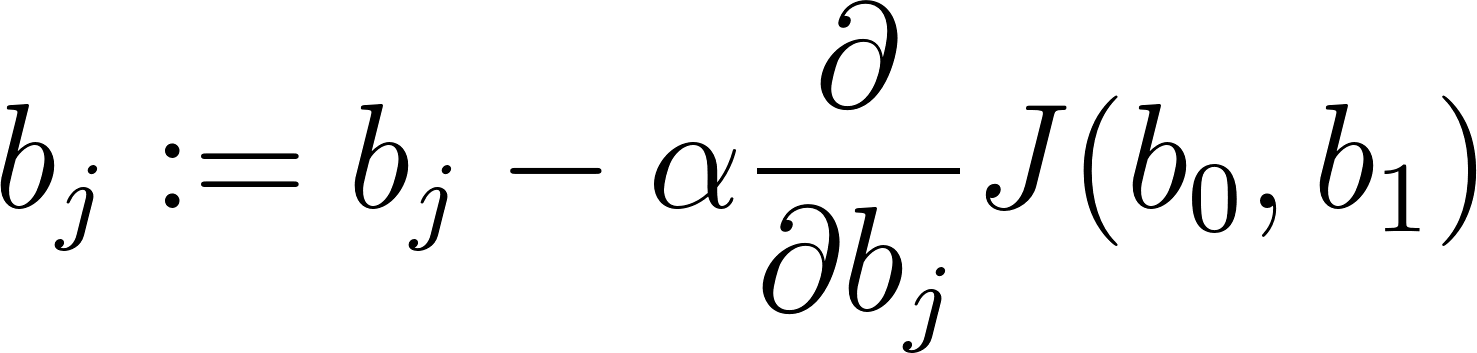
Отложим b0 на оси x и b1 на оси y, с функцией стоимости на вертикальной оси z. Точки на нашем графике будут результатом функции стоимости, используя нашу гипотезу с этими конкретными параметрами.

Мы будем знать, что нам удалось подобрать оптимальные параметры, когда наша функция стоимости находится в самом низу на нашем графике, то есть когда ее значение является минимальным.

То, как мы это делаем - это используя производную (касательную линию к функции) нашей функции стоимости. Наклон касательной является производной в этой точке, и это даст нам направление движения в сторону самого крутого уменьшения значения функции. Мы делаем шаги вниз по функции стоимости в направлении с самым крутым спусками, а размер каждого шага определяется параметром α, который называется скоростью обучения.

Алгоритм градиентного спуска:

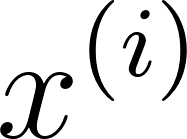
повторяйте до сходимости:

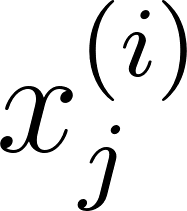
[](about:blank)

где j=0,1 - представляет собой индекс номера признака.

## Линейная регрессия с несколькими переменными

Линейная регрессия с несколькими переменными также известна как «множественная линейная регрессия». Введем обозначения для уравнений, где мы можем иметь любое количество входных переменных:

[](about:blank) - вектор-столбец всех значений признаков i-го обучающего примера;

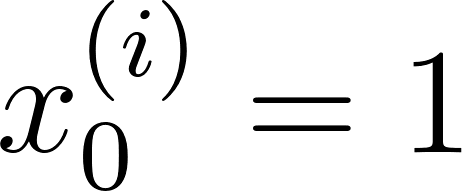
[](about:blank) - значение j-го признака i-го обучающего примера;

*m* - количество примеров в обучающей выборке;

*n* - количество признаков;

*X* - матрица признаков;

*b* - вектор параметров регрессии.

Заметим, что в будущем для удобства примем, что [](about:blank) для всех i. Другими словами, мы для удобства введем некий суррогатный признак, для всех наблюдений равный единице. это сильно упростит математические выкладки, особенно в матричной форме.

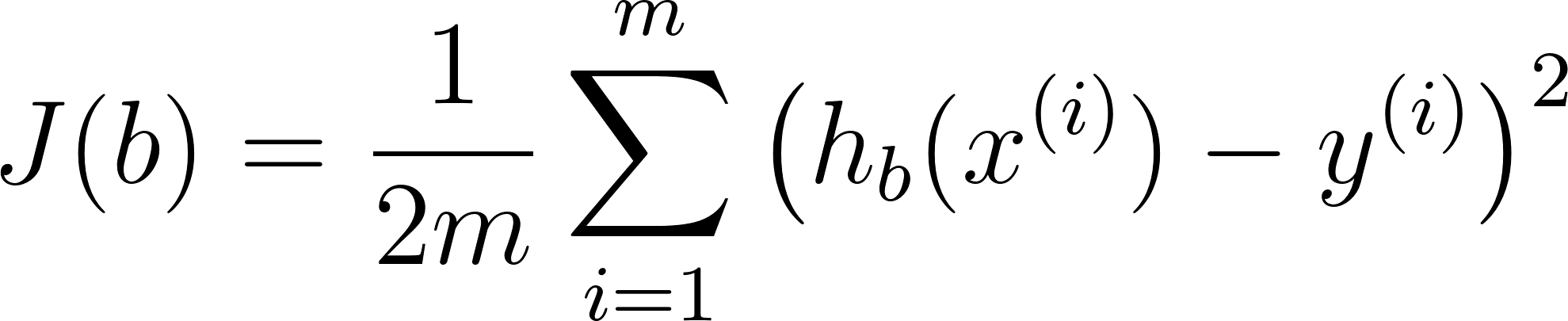
Теперь определим множественную форму функции гипотезы следующим образом, используя несколько признаков:

[](about:blank)

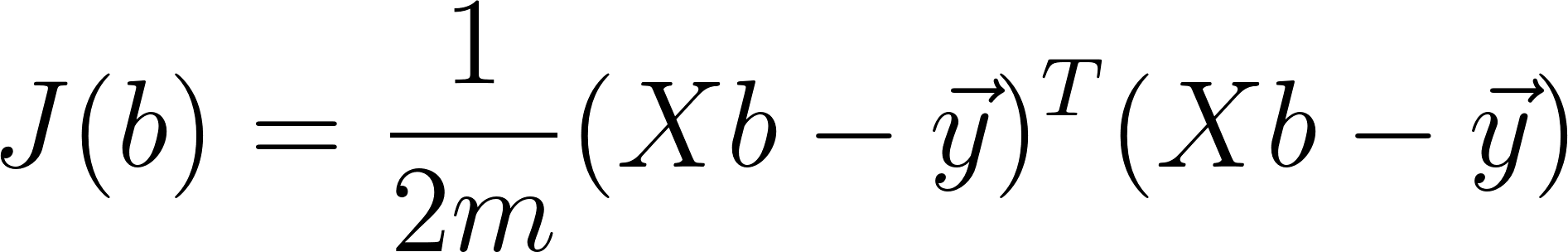
Используя определение матричного умножения, наша многопараметрическая функция гипотезы может быть кратко представлена в виде: *h(x) = B X*.

### Функция ошибки

Для множественной регрессии функция ошибки от вектора параметров b выглядит следующим образом:

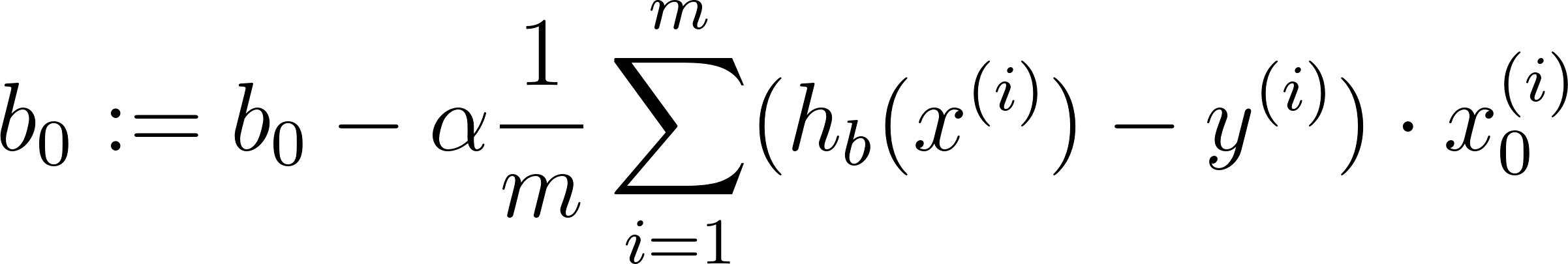
[](about:blank)

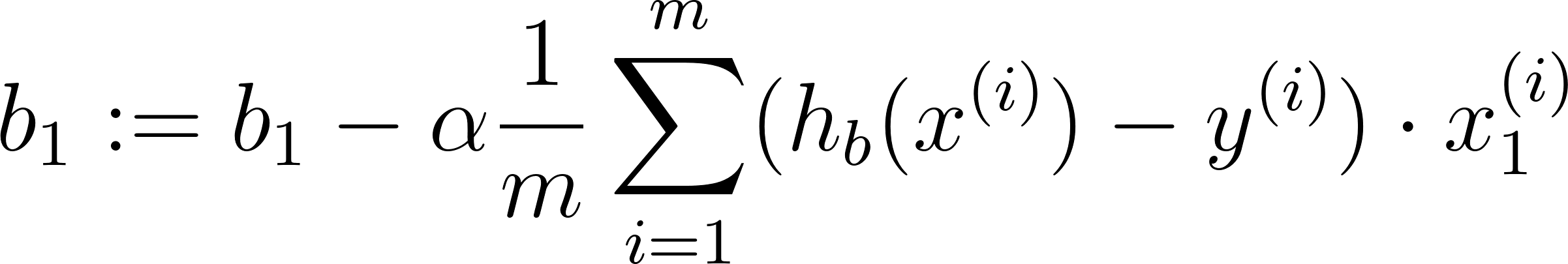
Или в матричной форме:

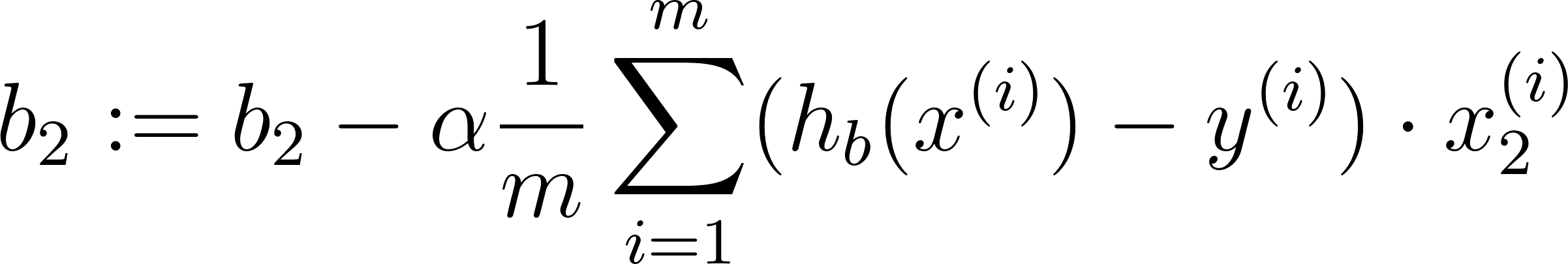
[](about:blank)

Метод градиентного спуска для множественной регрессии определяется следующими уравнениями:

повторять до сходимости:

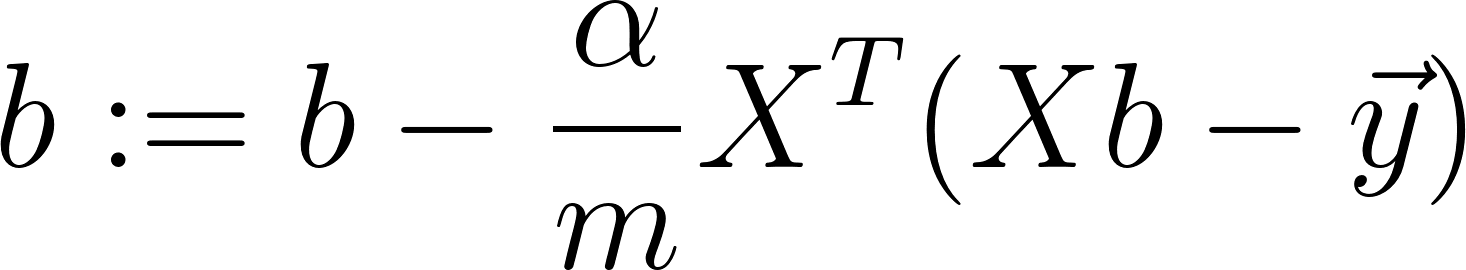
[](about:blank)

[](about:blank)

[](about:blank)

[](about:blank)

Или в матричной форме:

[](about:blank)

### Нормализация признаков

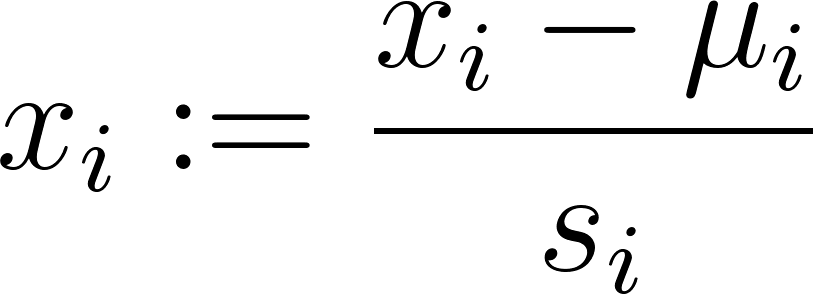
Мы можем ускорить сходимость метода градиентного спуска, получив каждое из наших входных значений примерно в том же диапазоне. Это связано с тем, что b будет быстро сходиться на малых диапазонах и медленно на больших диапазонах, и поэтому будет колебаться неэффективно до оптимального, если переменные очень неравномерны.

Способ предотвратить это - изменить диапазоны наших входных переменных, чтобы они были примерно одинаковыми. В идеале

-1 ≤ x ≤ 1 или же -0,5 ≤ x ≤ 0,5.

Это не точные требования; мы только пытаемся ускорить процесс. Цель состоит в том, чтобы получить все входные переменные в примерно один из этих диапазонов, дать или взять несколько.

Два метода для этого - масштабирование признаков и нормализация по среднему. Масштабирование признаков заключается в делении входных значений на размах выборки (то есть максимальное значение минус минимальное значение) входной переменной, в результате чего новый диапазон составляет всего 1. Нормализация по среднему включает в себя вычитание среднего значения входной переменной из значений для этой входной переменной, в результате чего новое среднее значение для этой переменной равно нулю. Чтобы реализовать оба этих метода, отрегулируйте свои входные значения, как показано в этой формуле:

[](about:blank)

Где [](about:blank) - среднее значение признака i, а [](about:blank) - стандартное отклонение этого признака.

### Советы по методу градиентного спуска

1. Отладка градиентного спуска.

Сделайте график с количеством итераций по оси x. Теперь построим функцию стоимости J(b) по числу итераций градиентного спуска. Если J(b) когда-либо возрастает, то, вероятно, вам необходимо уменьшить α.

2. Автоматический тест сходимости.

Объявите сходимость, если J(b) уменьшится меньше чем на E на одной итерации, где E - некоторое небольшое значение, такое как 10-3. Однако на практике трудно выбрать это пороговое значение.

Было доказано, что если скорость обучения α достаточно мала, то J(b) будет уменьшаться на каждой итерации.

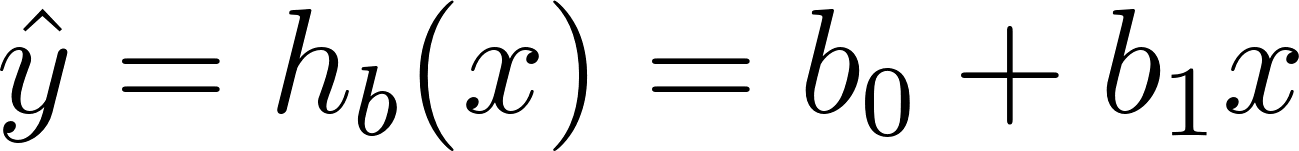
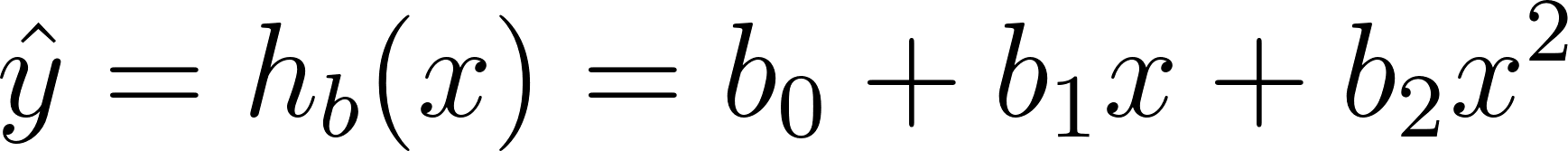
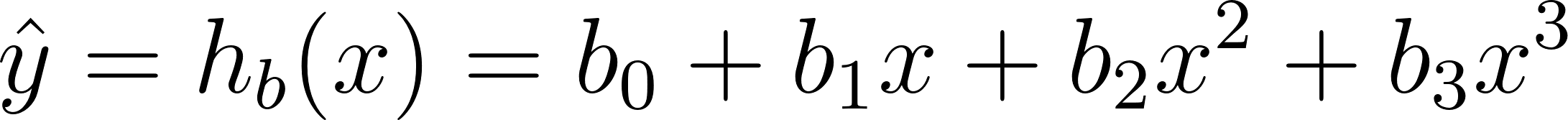
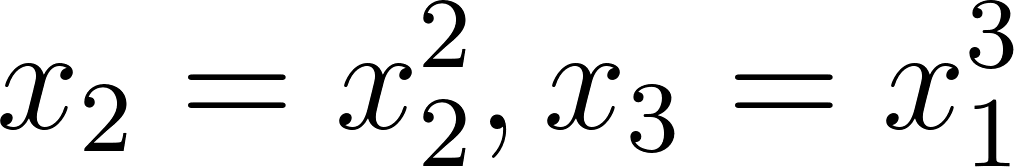
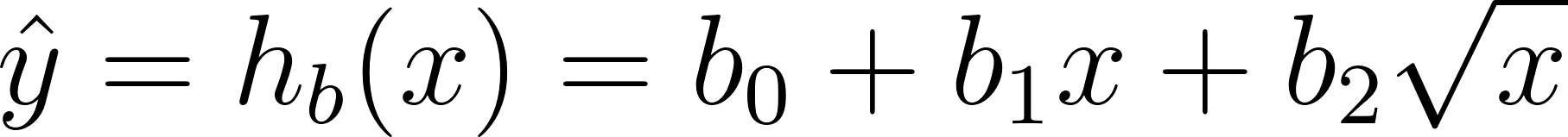
3. Суррогатные признаки

Мы можем улучшить наши функции и форму нашей функции гипотез несколькими способами. Мы можем объединить несколько признаков в один. Например, мы можем объединить x1 и x2 в новый признак x3, взяв x1⋅x2.

## Полиномиальная регрессия

Наша функция гипотезы не обязательно должна быть линейной (прямой), если это не соответствует данным.

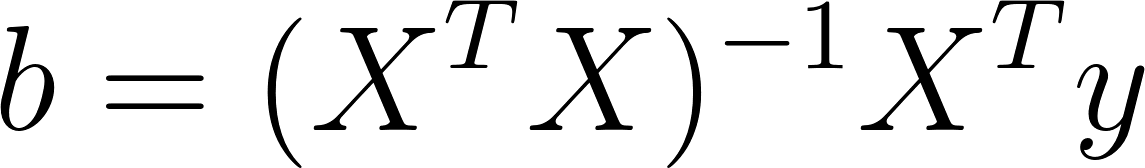
Мы можем изменить поведение или кривую нашей функции гипотезы, сделав ее квадратичной, кубической или квадратной корневой функцией (или любой другой формой).

Например, если наша функция гипотезы [](about:blank), то мы можем добавить еще один признак, основанный на x1, получив квадратичную функцию [](about:blank) или кубическую функцию [](about:blank). В кубической функции мы по сути ввели два новых признака: [](about:blank). Точно таким же образом, мы можем создать, например, такую функцию: [](about:blank).

Одна важная вещь, о которой следует помнить, заключается в том, что если вы выбираете свои функции таким образом, масштабирование признаков становится очень важным. Например, если x имеет диапазон 1 - 1000, тогда диапазон x2 становится 1 - 1000000, а диапазон x3 становится 1 - 1000000000.

### Нормальное уравнение

«Нормальное уравнение» - это метод нахождения оптимальных параметров регрессии без итераций:

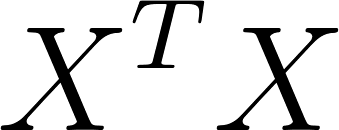
[](about:blank)

Нет необходимости выполнять масштабирование признаков, если мы решаем регрессию с помощью нормального уравнения.

Метод решения через нормальное уравнение имеет ряд преимуществ по сравнению с методом градиентного спуска:

1. Нет необходимости в нормализации признаков;
2. Не нужно выбирать скорость обучения;
3. Не требует вычисления частных производных функции ошибки;

Однако, у него есть и недостатки:

1. Имеет асимптотику O(n3) по сравнению с O(n2) у градиентного спуска. Поэтому довольно медленно работает при больших n.
2. Требует вычисления обратной матрицы. В некоторых случаях матрица [](about:blank) может быть вырожденной, что затруднит использование нормального уравнения.