



Лекция 3

Линейная регрессия

Содержание



1. Знакомимся с параметрическими алгоритмами и линейной регрессией
2. Улучшаем алгоритм
3. Разбираем задачу рекомендаций (семинар)

Часть 1

Линейная регрессия





Решаем задачу обучения с учителем

Функционал качества

$$Q(a, X, Y) = \frac{1}{N} \sum_{i=1}^N L(a, x_i, y_i), x_i \in X, y_i \in Y$$

Принцип минимизации эмпирического риска:

$$a^* = \underset{A}{\operatorname{argmin}} Q(a, X_{train}, Y_{train}), A \text{ — семейство алгоритмов.}$$

Формула обучения:

Learning = Representation + Evaluation + Optimization

Источник: homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Параметризация



Функций $a(x)$ которые идеально описывают обучающую выборку бесконечно много. Нужно сузить функциональное пространство перебора. **Параметризуем** искомую функцию модель $a(x, w)$ описывается вектором весов w .

Тогда задача превращается в поиск весов:

$$w^* = \underset{w}{\operatorname{argmin}} Q(w, X_{train}, y_{train})$$

Примеры:

$$a(x, w) = x_1 w_1 + x_2 w_2$$

$$a(x, w) = x_1 x_2 x_3 w_1$$

$$a(x, w) = 1[x_1 < w_1]$$

Representation



Пусть объект описывается D признаками f_1, f_2, \dots, f_D .

Тогда модель:

$a(x, w) = w_0 + \sum_{j=1}^D f_j w_j$ называется **линейной моделью**,

где w — D -мерный вектор признаков w_1, w_2, \dots, w_D

Далее будем считать, что в векторе признаков есть тождественно равный единице признак f_0 , тогда формула упростится до:

Representation: $a(x, w) = \sum_{j=0}^D f_j w_j = \mathbf{x} \cdot \mathbf{w}$

Параметры модели интерпретируемы. w_i — значение, на которое изменится предсказание, если признак f_i увеличить на единицу.

Какая гипотеза лежит в основе линейной модели?



Если целевая переменная вещественное число, то такую модель называют **линейной регрессией**.

$$Q(a, X, Y) = \frac{1}{N} \sum_{i=1}^N L(a, x_i, y_i), x_i \in X, y_i \in Y$$

Функции потерь:

1. **Квадратичная** (Q — MSE)

$$L(a, x, y) = (a(x) - y)^2$$

2. **Абсолютная** (Q — MAE)

$$L(a, x, y) = |a(x) - y|$$

3. **Логарифмическая** (Q — MSLE)

$$L(a, x, y) = (\log(a(x) + 1) - \log(y + 1))^2$$

4. **Абсолютная-процентная** (Q — MAPE) $L(a, x, y) = \frac{|a(x) - y|}{y}$

Optimization



$$w^* = \operatorname{argmin}_w \frac{1}{N} \sum_{i=1}^N L(a(x_i, w), y_i) = \operatorname{argmin}_w \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i \cdot \mathbf{w}, y_i), x_i \in X, y_i \in Y$$



Как найти минимум?

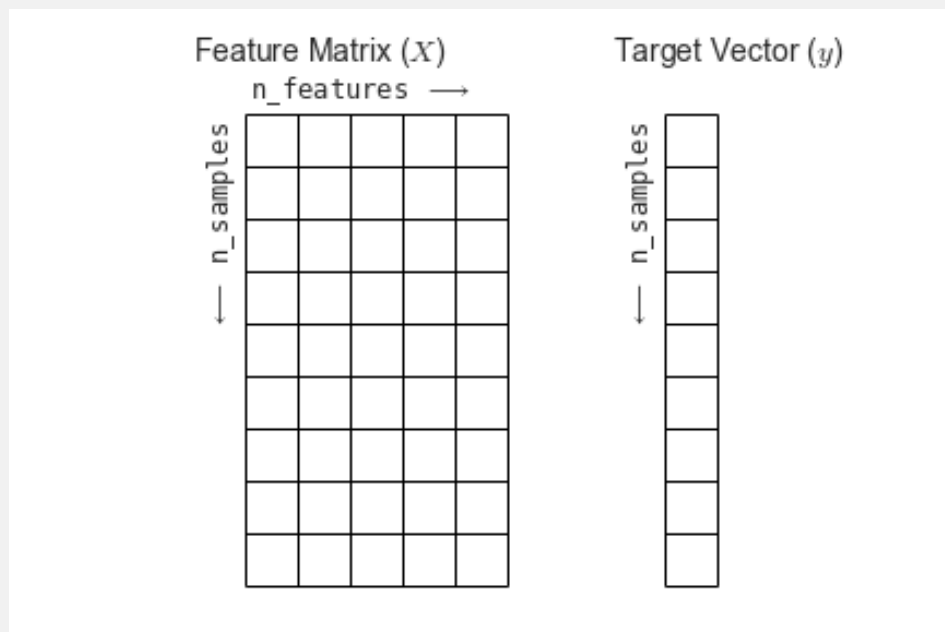
Optimization



$$Q(X, \mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (x_i \cdot \mathbf{w} - y_i)^2 = \frac{1}{N} ||X \cdot \mathbf{w} - \mathbf{y}||^2$$

X — матрица (N, D) , \mathbf{w} — вектор весов $(D, 1)$, \mathbf{y} — вектор ответов $(N, 1)$

$X \cdot \mathbf{w}$ — вектор предсказаний $(N, 1)$



Источник: jakevdp.github.io/PythonDataScienceHandbook

Точное решение



$\nabla_{\mathbf{w}} Q(\mathbf{w})$ — градиент, вектор частных производных. Необходимое условие минимума — градиент равен нулю.

$$Q(\mathbf{w}) = \frac{1}{N} ||X \cdot \mathbf{w} - \mathbf{y}||^2 = \frac{1}{N} (X \cdot \mathbf{w} - \mathbf{y})^T (X \cdot \mathbf{w} - \mathbf{y})$$

Два правила векторного дифференцирования:

$$\nabla_{\mathbf{w}} \mathbf{w}^T \mathbf{c} = \nabla_{\mathbf{w}} \mathbf{c}^T \mathbf{w} = \mathbf{c}; 2) \nabla_{\mathbf{w}} (\mathbf{w}^T C \mathbf{w}) = (C + C^T) \mathbf{w}$$

$$\nabla_{\mathbf{w}} Q(\mathbf{w}) = \nabla_{\mathbf{w}} (\mathbf{w}^T X^T X \mathbf{w} - \mathbf{w}^T X^T \mathbf{y} - \mathbf{y}^T X \mathbf{w} + \mathbf{y}^T \mathbf{y}) = (X^T X + X^T X) \mathbf{w} - 2X^T \mathbf{y} = 0$$

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$$

Дома покажите, что это действительно минимум!



Недостатки точного решения:

- Можно написать только для очень редких функций потерь
- Обращение матрицы — кубическая сложность
- Матрица $X^T X$ может быть плохо обусловленной, если есть ЛЗ признаки

$$\text{rank}(X) = \text{rank}(X^T X)$$



$$w^* = \operatorname{argmin}_w \frac{1}{N} \sum_{i=1}^N L(a(x_i, w), y_i)$$

- Нулевого порядка: жадные алгоритмы, динамическое программирование, золотое сечение, метод парабол, имитация отжига, генетические алгоритмы и т.д.
- Первого порядка: градиентный спуск, метод сопряженных градиентов, квазиньютоновские методы и т.д.
- Второго порядка: ньютоновские методы.

В курсе разбираем градиентный спуск.

Градиентный спуск



Антиградиент функции показывает направления наискорейшего убывания функции.

Ищем минимум $Q(w)$

1. Выбрать начальную длину шага α_0 , начальное приближение w_0

2. $w_{new} = w_{old} - \alpha \nabla_w Q(w_{old})$

3. $\alpha = f(k), k = k + 1$

4. Повторять (2), (3) до сходимости $Q(w)$ или w

$$f(k) = \alpha_0, f(k) = \frac{\alpha_0}{k}, f(k) = \frac{\alpha_0}{k^p}, \dots$$

Можно качество на валидации использовать как критерий останова



В задачах машинного обучения, оптимизируемая функция имеет специальный вид:

$$Q(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(\mathbf{w}, \mathbf{x}_i, y_i)$$

$$\nabla_{\mathbf{w}} Q(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \nabla_{\mathbf{w}} L(\mathbf{w}, \mathbf{x}_i, y_i)$$

Стохастическая оптимизация



1. Выбрать начальный шаг α_0 , начальное приближение \mathbf{w}_0 , размер батча n
2. Выбрать случайно $\{j_1, i_2, \dots, j_n\}$
3. Оценить градиент $\nabla_{\mathbf{w}} Q^*(\mathbf{w}_{old}) = \frac{1}{n} \sum_{j=1}^n \nabla_{\mathbf{w}} L(\mathbf{w}_{old}, \mathbf{x}_j, y_j)$
4. $\mathbf{w}_{new} = \mathbf{w}_{old} - \alpha \nabla_{\mathbf{w}} Q^*(\mathbf{w}_{old})$
5. $\alpha = f(k), k = k + 1$
6. Повторять (2 - 5) до сходимости

Обычно перемешивают всю выборку случайно.

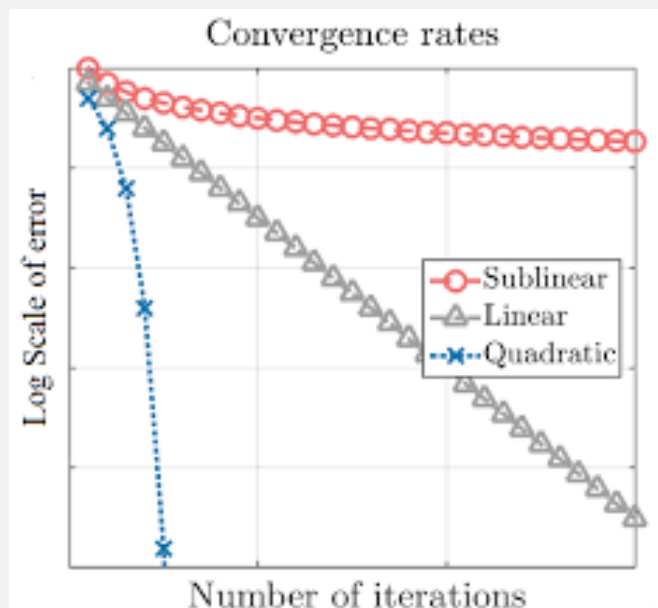
Когда прошли все выборку, то говорят, что прошла **одна эпоха**

- $n = N$ — градиентный спуск (Gradient Descent, GD)
- $n = 1$ — стохастический градиентный спуск (Stochastic Gradient Descent, SGD)
- $n > 1, n < N$ — мини-батч градиентный спуск (Mini-Batch Gradient Descent, MBGD)

Сходимость



Градиентный спуск имеет линейную скорость сходимости.
Стохастический градиентный спуск — сублинейную.



Источник: niaohe.ise.illinois.edu/IE598_2016/pdf/IE598-lecture8-gradient%20descent.pdf

Пример



$$Q(X, \mathbf{w}) = \frac{1}{2N} \sum_{i=1}^N (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2$$

$$L(\mathbf{w}, \mathbf{x}_i, y_i) = \frac{1}{2} (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, \mathbf{x}_i, y_i) = (\mathbf{x}_i \cdot \mathbf{w} - y_i) \mathbf{x}_i$$

Делаем SGD:

$\mathbf{w}_{new} = \mathbf{w}_{old} - \alpha (\mathbf{x}_i \cdot \mathbf{w}_{old} - y_i) \mathbf{x}_i$, где i случайно от 1 до N

Сложность предсказания $O(D)$

Сложность обучения одного шага $O(D)$

Делаем GD:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \alpha \sum_{i=1}^N (\mathbf{x}_i \cdot \mathbf{w}_{old} - y_i) \mathbf{x}_i$$

Сложность предсказания $O(D)$

Сложность обучения одного шага $O(ND)$

Стохастическая оптимизация



Почему это вообще работает?

Для SGD

$$j \sim U(1, N), \nabla_w Q^*(w_{old}) = \nabla_w L(w_{old}, x_j, y_j)$$

$\nabla_w L(w_{old}, x_j, y_j)$ равновероятно принимает N значений

Матожидание величины $\nabla_w L(w_{old}, x_j, y_j)$ по определению мат. ожидания

$$\frac{1}{N} \sum_{j=1}^N \nabla_w L(w_{old}, x_j, y_j)$$

То есть в среднем мы как раз идем в сторону антиградиента!



Линейная регрессия невероятно популярный алгоритм машинного обучения. Плюсы алгоритма:

- Быстро учится
- Быстро предсказывает
- Легко интерпретируется
- Легко хранить в памяти
- Легко применять с дифференцируемой функцией потерь

Весомый минус - не способен учитывать нелинейные зависимости в данных.

Часть 2

Улучшаем модель





$$a(x, w) = \sum_{j=0}^D f_j w_j = \mathbf{x} \cdot \mathbf{w}$$

Параметры модели интерпретируемы. w_i — значение, на которое изменится предсказание, если признак f_i увеличить на единицу.

Категориальные признаки кодируем:

- One-hot кодирование - категориальный признак с k значениями превращаем в k бинарных признаков!
- Кодирование через целевую переменную (нельзя включать переменную самого объекта)
- Кодирование через вещественные признаки

Правильные признаки



$$a(x, w) = \sum_{j=0}^D f_j w_j = \mathbf{x} \cdot \mathbf{w}$$

Для вещественных признаков применяем нелинейные функции - возводим в степень, берем синус и т.д

Учитываем взаимодействия:

- Пару вещественных перемножаем, делим и т.д.
- Для пары бинарных используем логические операции

Невозможно сделать правильное признаковое пространство без понимания самой задачи!

Нормализация данных



Масштаб признаков важен для скорости сходимости

$$f(x, y, z) = (x - 1)^2 + (y - 1)^2 + (z - 1)^2$$

Стартуем из (0,0,0) с шагом 0.5

$$(x, y, z) = (0,0,0) - 0.5(-2, -2, -2) = (1,1,1)$$

А если так:

$$f(x, y, z) = 1000(x - 1)^2 + 200(y - 1)^2 + (z - 1)^2$$

- Стандартизация $f_j = \frac{f_j - \text{mean}(f_j)}{\text{std}(f_j)}$

- Min-max нормализация $f_j = \frac{f_j - \min(f_j)}{\max(f_j) - \min(f_j)}$



Хотим еще сузить функциональное пространство $a(x)$, чтобы улучшить обобщающую способность. Наложим доп. штраф, если решение удаляется от нашего представления о правильном решении.

$$Q_r(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

$R(\mathbf{w})$ - регуляризатор, α - параметр регуляризации.

? Как подобрать параметр α ?



Большие веса — признак переобучения

Базовые регуляризаторы, штрафующие большие значения весов:

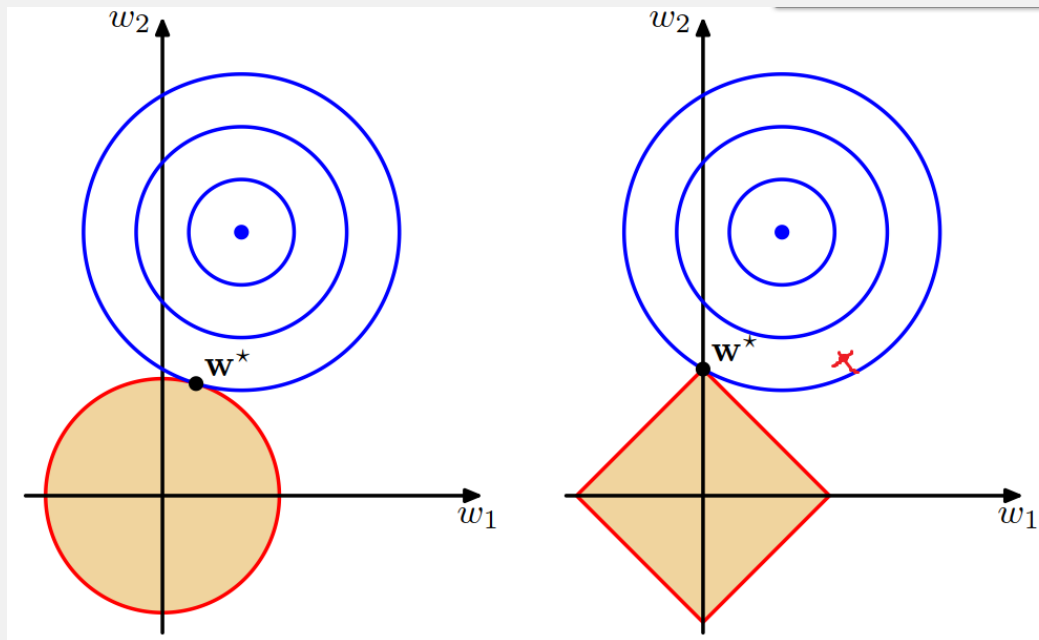
- L_1 регуляризация (Lasso регрессия).
 $R(\mathbf{w}) = \sum_{j=1}^D |w_j|$; С ней Q_r не будет гладким!
- L_2 регуляризация (Ridge регрессия).
 $R(\mathbf{w}) = \sum_{j=1}^D w_j^2$.

Дома вывести формулу точного для L_2 регуляризации

$$\mathbf{w} = (X^T X + \lambda \cdot I)^{-1} X^T \mathbf{y}$$



Можно показать, что L_1 приводит к занулению весов, то есть автоматически отбирает важные признаки!



Источник: Bishop

Big data и линейная регрессия



Стохастическая оптимизация позволяет не хранить данные в оперативной памяти! Взяли батч, обновили веса и сразу же его забыли. Позволяет легко обучаться на терабайтах данных. Рекомендую ознакомиться с библиотекой

Vowpal Wabbit

Online learning — обучение, когда прецеденты поступают потоком.



$$Q(X, \mathbf{w}) = \frac{1}{2N} \sum_{i=1}^N (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2$$

$$L(\mathbf{w}, \mathbf{x}_i, y_i) = \frac{1}{2} (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, \mathbf{x}_i, y_i) = (\mathbf{x}_i \cdot \mathbf{w} - y_i) \mathbf{x}_i$$

Чему равен градиент по j компоненте в точках $x_{i,j} = 0$?

Разреженный формат данных - для каждого объекта храним словарь вида

{номер ненулевого признака : значение}

Приведите пример задачи с разреженными данными

Разреженные данные



Самый частый пример разреженных данных — тексты. Объект — текст, признак — наличие слова в тексте.

Делаем one hot кодирование, получаем пространство размера словарь слов.

Вместо бинарных признаков часто делают TF-IDF преобразование

$TF(t) = (\text{сколько раз } t \text{ встречался в тексте}) / (\text{длина текста})$.

$IDF(t) = \log_e(\text{число текстов} / \text{число текстов с } t)$.



**Спасибо за
внимание!**