Занятие 5. Логистическая регрессия

Колмагоров Евгений
ml.hse.dpo@yandex.ru

11 ноября 2024

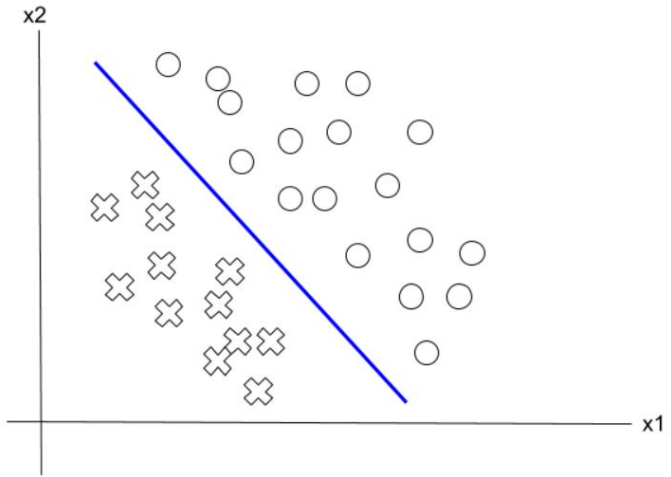
План лекции

1. Логистическая регрессия
2. Вывод функции ошибки через ММП
3. Кодирование признаков
4. Стратегии поиска гиперпараметров

Напоминание предыдущего материала

Решаем задачу бинарной классификации методом линейной регрессии:

$$a(x, w) = sign(\sum_{i=0}^{d} w_i x_i)$$



Уверенность алгоритма на некотором объекте x_i оцениваем через отступ M_i:

$$M_i = y_i \cdot a(x_i, w) = y_i \cdot (w, x_i)$$

Напоминание предыдущего материала

Из предыдущей лекции производим обучение алгоритма, через минимизацию верхних оценок L_i:

$$Q(a, X) = \frac{1}{N} \sum_{i=0}^{N} I[M_i < 0] \leq \frac{1}{N} \sum_{i=0}^{N} L(x_i, y_i) = \hat{Q}(a, X)$$

$$\hat{Q}(a, X) \to min_w$$

Одной из таких L была логистическая функция:

$$L(M_i) = log(1 + e^{-M_i}) = log(1 + e^{-y_i \cdot <w, x_i>})$$

Попробуем обосновать, почему была использована именно такая функция потерь

Проблема интерпретация ответов

Вспомним из предыдущей лекции, что принятие решения происходило по следующей формуле:

$$a(x, w) = sign(\sum_{i=0}^{d} w_i x_i)$$

Какую интерпретацию несут значения ответов (w_i, x) при фиксированном w?

Что означает величина этого произведения, например, 100, 1000 и тд.? Есть ли верхняя граница?

Более понятная интерпретация

Попробуем построить алгоритм бинарной классификации *a(x, w)*, таким образом, чтобы его ответ был равен вероятности принадлежности объекта к положительному классу:

$$a(x_i, w) = p(y = 1 | x_i, w)$$

Вопрос: Какими необходимыми условиям должна обладать функция ***a(x$_i$, w)****, чтобы она могла нести вероятностный смысл?*

Зачем это может быть нужно?

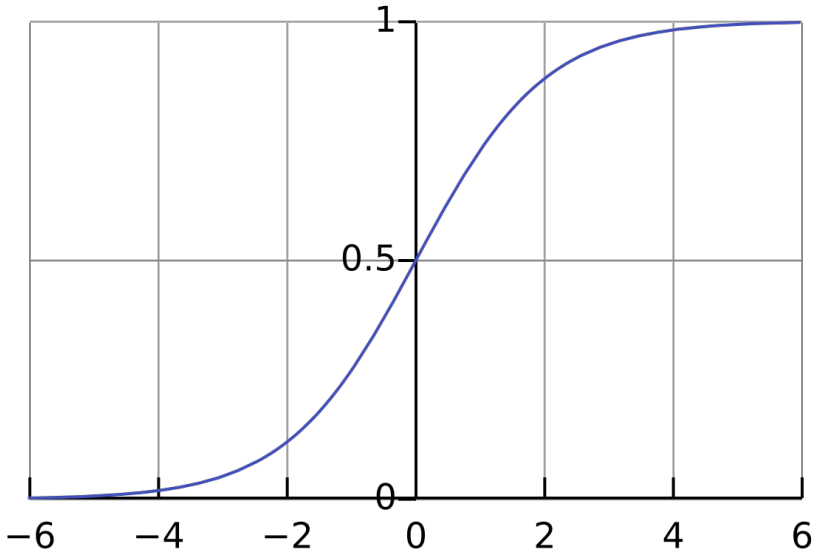
Имея вероятности принадлежности некоторого объекта x к заданному классу $p(y=1|\ x)$, можно проводить проводить более явный экономический подсчёт.

Например, если мы знаем, что некоторый пользователь с вектором признаков x купит показываемый товар с вероятностью p. То можно оценить ожидаемую выручку, как pNC, где N - кол-во таких пользователей, а C - цена товара.

Функция сигмоиды

Рассмотрим следующую функцию

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Представленная функция носит название сигмоиды, и обладает несколькими полезными свойствами

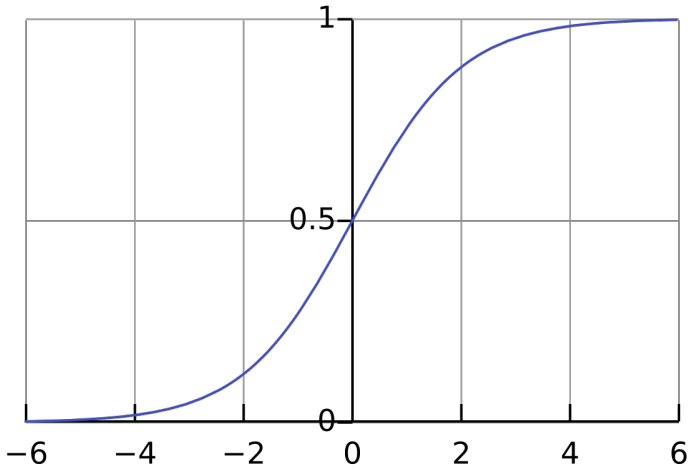
Несколько полезных свойств сигмоиды

1) Область допустимых значений σ(z): *[0, 1]*
2) Интерпретируемая вероятность противоположного события *σ(-z) = 1 - σ(z)*
3) Существование обратной функции

$$z(s) = ln\frac{s}{1-s}$$, где s -значение сигмоиды

4) Удобно дифференцируема

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$



Убедитесь в справедливости всех этих свойств

Достаточно ли введение сигмоиды над ответами модели, чтобы было выполнено свойство $a(x, w) = p(y=1 | x, w)$?

Подбор вектора весов w

Так как ответ алгоритма зависит от вектора весов $\boldsymbol{w}$, то необходимо, чтобы функция вероятности принадлежности некоторого объекта к положительному классу зависела от имеющейся выборки $\boldsymbol{X}$

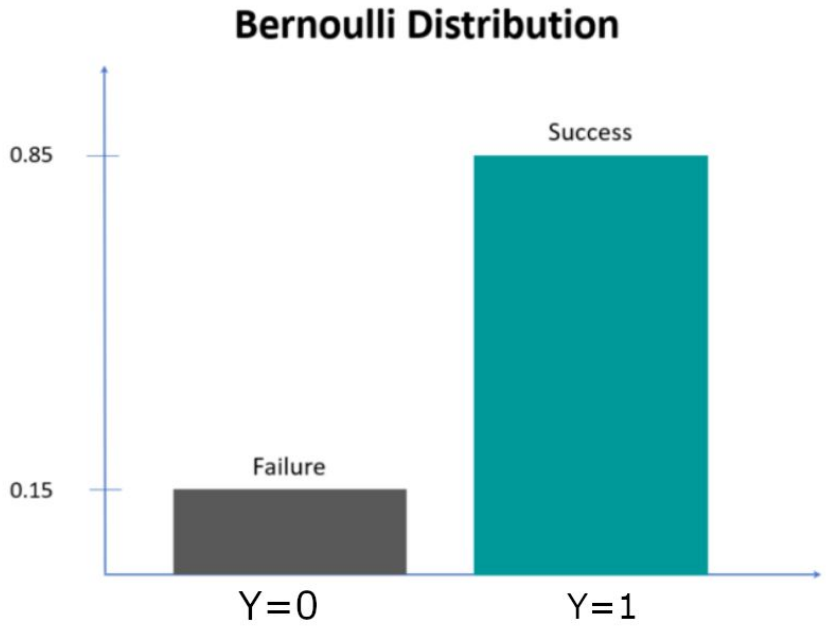
$$a(x_i, w) = \frac{1}{1+e^{-(w,\, x_i)}}$$

Попробуем ещё подобрать веса так, чтобы ответ алгоритма имел вероятностный смысл

Распределение бернулли

Так как рассматривается бинарная классификация, то распределение ответов p(y = 1 | x) для некоторого объекта x будет из распределения бернулли

$$\begin{cases} p, & \text{для } y = 1 \\ 1 - p, & \text{для } y = 0 \end{cases}$$



Метод максимального правдоподобия

Попробуем найти веса w, таким образом, чтобы они наилучшим образом объясняли имеющиеся данные (X, y).

Функция, которая описывает имеющиеся данные называется функцией правдоподобия L. Как правило стоит задача найти вектор весов таким образом, чтобы функция правдоподобия имела максимальное значение:

$$P(y_1, \ldots, y_n | x_1, \ldots, x_n, \mathbf{w}) \rightarrow max_w$$

Правдоподобие для одного объекта

Так как метка y_i для некоторого объекта x_i из бернуллиевской распределения, то функция правдоподобия приобретает следующий вид:

$$p(y_i|x_i, \mathbf{w}) = p(y_i = 1|x_i, \mathbf{w})^{y_i} \cdot p(y_i = 0|x_i, \mathbf{w})^{1-y_i}$$

Нетрудно заметить, что запись выше в точности совпадает с исходным определением распределения бернулли

$$\begin{cases} p, & \text{для } y = 1 \\ 1 - p, & \text{для } y = 0 \end{cases}$$

Функция правдоподобия для нескольких объектов

Воспользуемся тем, что каждый из объектов x_i независим друг от друга, поэтому факторизуем функцию совместного правдоподобия:

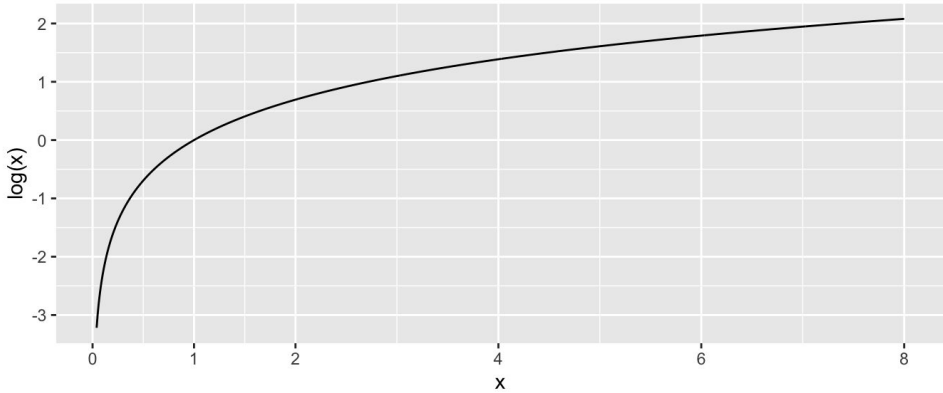
$$P(y_1, \dots, y_n | x_1, \dots, x_n, \mathbf{w}) = p(y_1 | x_1, \mathbf{w}) \cdot \dots \cdot p(y_n | x_n, \mathbf{w}) =$$
$$= \Pi_{i=1}^{n} p(y_i | x_i, \mathbf{w}) \rightarrow max_w$$

Вопрос: Какие могут быть вычислительные проблемы в формуле выше?

Перейдём к логарифму

При большом количестве объектов n произведение вероятностей может оказаться слишком маленьким числом, которое непредставимо в памяти компьютера. Поэтому применим логарифмическое преобразование:

$$logP(y_1, \dots, y_n | x_1, \dots, x_n, \mathbf{w}) =$$
$$= \log \prod_{i=1}^{n} p(y_n | x_n, \mathbf{w}) =$$
$$= \sum_{i=1}^{n} log\, p(y_i | x_i, \mathbf{w}) \to max_w$$



Вопрос: почему добавление логарифма законно?

Вернёмся к исходной задаче

Логарифмическая функция правдоподобия для бернуллиевской распределения:

$$L(X,w) = logP(y_1,\dots,y_n|x_1,\dots,x_n,\mathbf{w}) =$$
$$= \sum_{i=1}^{n}[y_i \cdot log\, p(y_i=1|x_i,\mathbf{w}) + (1-y_i)log\,(1-p(y_i=1|x_i,\mathbf{w}))] \to max_w$$

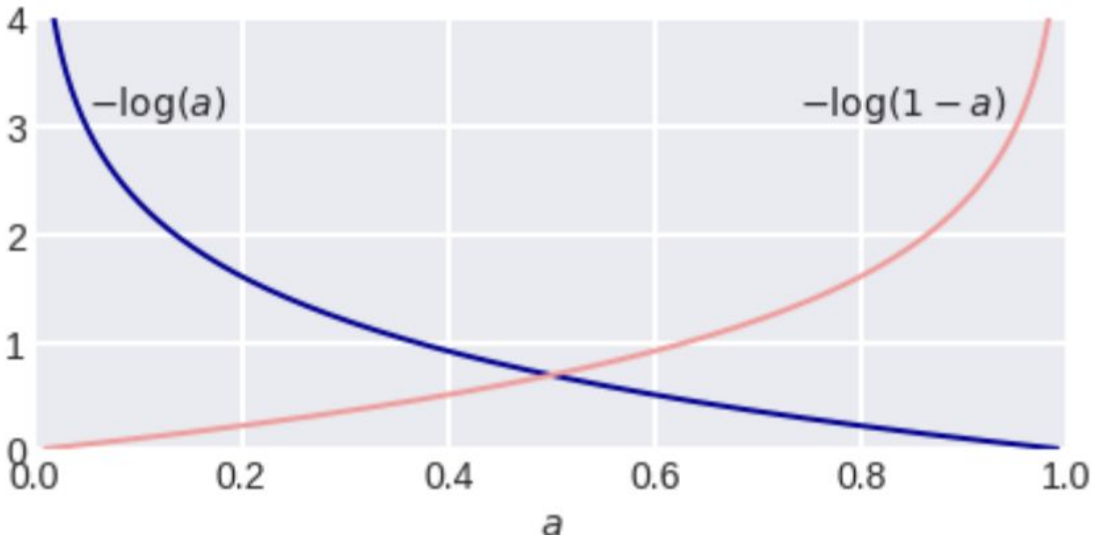
Перейдём от задачи максимизации к задаче минимизации умножив на -1 и вспомним, что ищем решение алгоритма в виде $a(x_i, w) = p(y_i = 1 | x_i, \mathbf{w})$

$$-\sum_{i=1}^{n}[y_i \cdot log\, a(x_i,w) + (1-y_i)log\,(1-a(x_i,w))] \to min_w$$

Логистическая функция потерь

Взяв среднее по всей выборке, получим функционал ошибки logloss:

$$logloss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot log\, a(x_i, w) + (1 - y_i) log\, (1 - a(x_i, w))]$$



Логистическая ошибка на одном объекте

Оптимизация константного решения

Пусть алгоритм принятия решения имеет константное значение.

Найдём оптимальное решение среди константных моделей.

Пусть:

- a - ответ алгоритма
- n_1 - объектов положительного класса
- n_2 - объектов отрицательного класса
- $n = n_1 + n_2$ - всего объектов в выборке

Оптимизация константного решения

Тогда функция потерь примет следующий вид:

$$logloss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot log\, a + (1 - y_i) log\, (1 - a)]$$

Так как в выборке n_1 положительных и n_2 отрицательных объектов, то можно представить сумму выше, как

$$logloss = -\frac{n_1}{n} log\, a - \frac{n_2}{n} log\, (1 - a)$$

Так как ищем оптимальное решение для a, то возьмём производную по a и приравняем её к 0:

$$\frac{\partial logloss}{\partial a} = -\frac{n_1}{n}\frac{1}{a} + \frac{n_2}{n}\frac{1}{1-a} = 0$$

Оптимизация константного решения

Выразив a из полученного выражения получим, что

$$a^* = \frac{n_1}{n}$$

Что в точности соответствует вероятности встретить объект первого класса в выборке

Диапазоны значений

Logloss потенциально неограниченная функция, так как логарифм способен принимать сколько угодно большие значения.

Но как правило не имеет смысла рассматривать значения функционала, которые хуже константного, поэтому принимается следующий диапазон

$$logloss \in [0, -\frac{n_1}{n} log \frac{n_1}{n} - \frac{n_2}{n} log \frac{n_2}{n}]$$

Логистическая регрессия

В логистической регрессии ответы алгоритма формируются по следующей формуле:

$$a(x_i, w) = \sigma(x \cdot w) = \frac{1}{1+e^{-w \cdot x}}$$

А вектор весов w формируется через оптимизацию функционала logloss:

$$w^* = argmin_w \frac{1}{n} \sum_{i=1}^{n} [-y_i log\, a(x_i, w) - (1 - y_i) log(1 - a(x_i, w))]$$

Вопрос: как будем оптимизировать logloss?

Шаг градиентного спуска для логистической регрессии

Посчитаем чему равен градиент логистической функции:

$$\nabla_w L(a, X) = -\frac{1}{n} \sum_{i=1}^{n} [\nabla_w y_i log\, \sigma(w \cdot x_i) + \nabla_w (1 - y_i) log\, (1 - \sigma(w \cdot x_i))]$$

$$\nabla_w L(a, X) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \frac{1}{\sigma(w \cdot x_i)} \cdot \sigma(w \cdot x_i)(1 - \sigma(w \cdot x_i))x_i +$$

$$+(1 - y_i) \frac{1}{1 - \sigma(w \cdot x_i)} \cdot -\sigma(w \cdot x_i)(1 - \sigma(w \cdot x_i))x_i]$$

Приведём подобные и получим:

$$\nabla_w L(a, X) = -\frac{1}{n} \sum_{i=1}^{n} [(y_i - \sigma(w \cdot x_i))x_i]$$

Шаг градиентного спуска для логистической регрессии

Шаг градиентного спуска:

$$w^{k+1} = w^k + \frac{\eta}{n} \sum_{i=1}^{n} [(y_i - \sigma(w \cdot x_i))x_i]$$

Минуточку…..

Но ведь в начале лекции логистическая функция имела совсем иной вид:

$$L(M_i) = log(1 + e^{-M_i}) = log(1 + e^{-y_i \cdot <w, x_i>})$$

Какая связь между тем, что было в начале и тем, что мы вывели через формулу правдоподобия?

Разные формы записи

Утверждение. Логарифмическая функция потерь может быть записана в виде:

$$L(a, X) = \sum_{i=1}^{n} \left[1 + e^{-y_i(w, x_i)}\right]$$

$$\text{logloss}(a, y) = \begin{cases} -\log a, & y = +1, \\ -\log(1-a), & y = -1, \end{cases} = \begin{cases} -\log\left(\dfrac{1}{1+\exp(-w^{\mathrm{T}}x)}\right), & y = +1, \\ -\log\left(\dfrac{\exp(-w^{\mathrm{T}}x)}{1+\exp(-w^{\mathrm{T}}x)}\right), & y = -1, \end{cases} =$$

$$= \begin{cases} -\log\left(\dfrac{1}{1+\exp(-w^{\mathrm{T}}x)}\right), & y = +1, \\ -\log\left(\dfrac{1}{\exp(+w^{\mathrm{T}}x)+1}\right), & y = -1, \end{cases} = \begin{cases} \log\left(1+\exp(-w^{\mathrm{T}}x)\right), & y = +1, \\ \log\left(1+\exp(+w^{\mathrm{T}}x)\right), & y = -1. \end{cases}$$

Смысл скалярного произведения

Обратная функция для сигмоиды представляет собой следующую формулу:

$$z(s) = ln\frac{s}{1-s}$$

Так как выполнены следующие соотношения:

$$\begin{cases} s = \sigma(w \cdot x_i) = p(y = 1|x, w) \\ 1 - s = 1 - p(y = 1|x, w) = p(y = 0|x, w) \end{cases}$$

Тогда скалярное произведение приобретает следующий смысл:

$$z(s) = w \cdot x = ln\frac{p(y=1|x,\mathbf{w})}{p(y=0|x,\mathbf{w})}$$

Смысл скалярного произведения

Величина

$$z(s) = w \cdot x = ln \frac{p(y{=}1|x,\mathbf{w})}{p(y{=}0|x,\mathbf{w})}$$

Называется логарифмом ***отношения шансов (log odds)***, и как можно заметить из формулы может принимать любое значение

Скоринговые функции

Logloss относится скоринговым функциям.

Определение: Функция L(y, a) называется скоринговой, если выполняется следующее равенство

$$p(y = 1|a, x_i) = argmin_a \mathbb{E}_y[L(y, a)],\ y \sim Be(p)$$

Помимо logloss есть ещё и другие…

Скоринговая функция MSE

В качестве L(y, a), можно взять функцию MSE, тогда

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

Посчитаем математическое ожидание:

$$\mathbb{E}_y[L(y, a)] = p(1 - a)^2 + (1 - p)a^2$$

Скоринговая функция MSE

Найдём оптимальный алгоритм a, взяв производную и приравняем её к 0:

$$\frac{\partial \mathbb{E}_y[L(y,a)]}{\partial a} = -2p(1-a) + 2(1-p)a = 0$$

Откуда следует что

$$a^* = p$$

Скоринговая функция MSE

Найдём оптимальный алгоритм a, взяв производную и приравняем её к 0:

$$\frac{\partial \mathbb{E}_y[L(y,a)]}{\partial a} = -2p(1-a) + 2(1-p)a = 0$$

Откуда следует что

$$a^* = p$$

Вопрос: почему тогда опять не воспользоваться функцией ошибки MSE?

MSE в качестве скоринговой функции

В качестве скоринговой функции MSE уступает LogLoss по нескольким причинам:

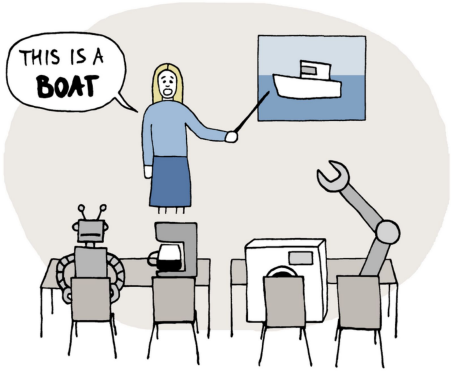
- Слабый штраф за большую ошибочные ответы. Например, на предсказании 1, когда верно значение 0, максимальная величина ошибки будет 1, в то время как у LogLoss $+\infty$
- Особенности оптимизации, функционал ниже не является всюду выпуклой функцией по z

$$\frac{1}{n}\sum_{i=1}^{n}(\sigma(z)-y)^2$$

Выводы по логистической регрессии

- Логистическая функция ошибки является естественным функционалом, который выводится из функции правдоподобия целевой переменной из бернулиевсого распределения
- Ответы алгоритма *a(x, w)*, должны принимать значения от 0 до 1. При этом необязательно использовать для этого сигмоиду, есть аналогичные ей функции
- LogLoss и MSE являются скоринговыми функциями, но предпочтительней использовать LogLoss, так как это более естественная функция для задачи классификации

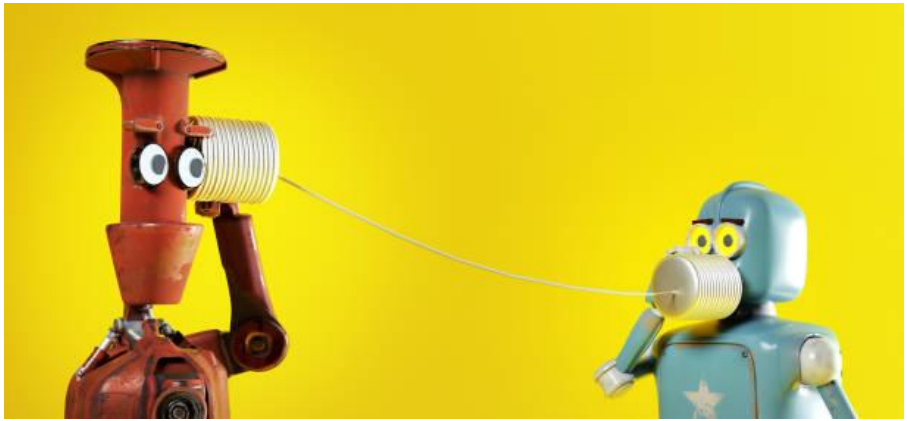
Ещё пара способов кодирования признаков



Ещё об одном способе кодирования признаков

На предыдущих занятиях было рассмотрено два способа кодирования признаков:

- One-hot encoding
- Кодирование немонотонных факторов



В данной части посмотрим на ещё один способ кодирования переменной на основе **счётчиков**

Mean Target Encoding

Определение: Счётчик (mean target encoding) – это вероятность получить значение целевой переменной для данного значения категориального признака.

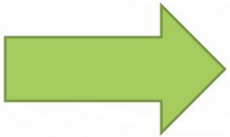
Пример

Для каждого уникального значения: Moscow, Tver, Klin некоторого признака посчитаем вероятность получить целевую переменную

	feature	target
0	Moscow	0
1	Moscow	1
2	Moscow	1
3	Moscow	0
4	Moscow	0
5	Tver	1
6	Tver	1
7	Tver	1
8	Tver	0
9	Klin	0
10	Klin	0
11	Tver	1

Пример

	feature	target
0	Moscow	0
1	Moscow	1
2	Moscow	1
3	Moscow	0
4	Moscow	0
5	Tver	1
6	Tver	1
7	Tver	1
8	Tver	0
9	Klin	0
10	Klin	0
11	Tver	1



	feature	feature_mean	target
0	Moscow	0.4	0
1	Moscow	0.4	1
2	Moscow	0.4	1
3	Moscow	0.4	0
4	Moscow	0.4	0
5	Tver	0.8	1
6	Tver	0.8	1
7	Tver	0.8	1
8	Tver	0.8	0
9	Klin	0.0	0
10	Klin	0.0	0
11	Tver	0.8	1

Счётчики в бинарной классификации

В случае бинарной классификации счётчики можно задать формулой

$$Counter = \frac{Goods}{Goods + Bads} = mean(Target)$$

- Goods - число единиц в столбце Targets
- Bads - число нулей в том же столбце

Счётчики для более общего случая

- Пусть целевая переменная y принимает значения от 1 до K
- Закодируем категориальную перменную f(x) следующим способом

  $counts(u, X) = \sum_{(x,y)} I[f(x) = u]$, т.е. посчитаем как часто встречается значение u для данного признака

  $success_k(u, X) = \sum_{(x,y)} I[f(x) = u] \cdot I[y = k],\ k = 1, .., K$, посчитаем как часто достигается класс k на данном значении признака u

$$mean\_target_k(u, X) = \frac{success_k(u,X)}{count(u,X)} \approx p(y = k|u)$$

Вопрос: когда такой способ кодирования переобучит алгоритм?

Счётчики для более общего случая

- Пусть целевая переменная y принимает значения от 1 до K
- Закодируем категориальную перменную f(x) следующим способом

$counts(u, X) = \sum_{(x,y)} I[f(x) = u]$, т.е. посчитаем как часто встречается значение u для данного признака

$success_k(u, X) = \sum_{(x,y)} I[f(x) = u] \cdot I[y = k],\ k = 1, .., K$, посчитаем как часто достигается класс k на данном значении признака u

$$mean\_target_k(u, X) = \frac{success_k(u,X)}{count(u,X)} \approx p(y = k|u)$$

Вопрос: когда такой способ кодирования переобучит алгоритм?
Ответ: когда в данных много редких категорий

Добавка сглаживания

Счётчик со сглаживанием:

$$\frac{mean\_target_k(u,X)\cdot n_{rows}+global\_mean\cdot\alpha}{n_{rows}+\alpha}$$

- n_{rows} – количество строк в данной категории
- α – параметр регуляризации

Счётчики: опасность переобучения

Вычисляя счётчики, мы закладываем в признаки информацию о целевой переменной посчитанной по обучающей выборке и, тем самым увеличиваем риск переобучиться!

Пример вычисления счётчика

Можно вычислять счётчики так

city	target
Moscow	1
London	0
London	2
Kiev	1
Moscow	1
Moscow	0
Kiev	0
Moscow	2

Вычисляем счетчики по этой части

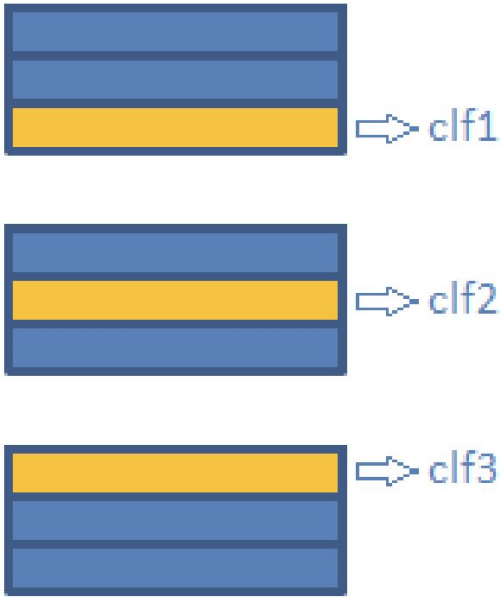
Кодируем признак вычисленными счётчиками и обучаемся по этой части

Пример вычисления счётчика

Более продвинутый способ (по кросс-валидации):

1. Разбиваем выборку на m частей X_1, …, X_m
2. На каждой части на X_i значения признаков вычисляются по оставшимся частям:

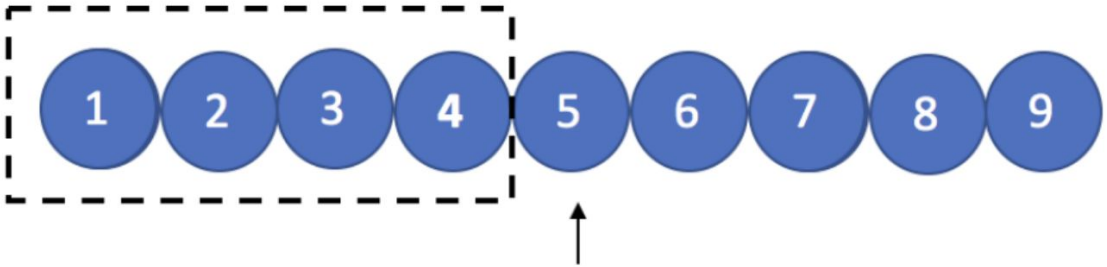
$$x \in X_i => f_k(x) = target\_mean_k(x, X \backslash X_k)$$



Пример вычисления счётчика

Ещё одним продвинутым примером вычисления счётчика является expanding mean подход.

Суть схемы заключается в том, чтобы пройти по отсортированному в определенном порядке датасету и для подсчета счетчика для строки m использовать строки от 0 до m-1.



Running mean calculation.
Numbers are assigned randomly to each observation. Only 1-4 are used to find encoding for 5

Борьба с переобучением в счётчиках

- Сглаживание
- Вычисление по кросс-валидации
- Expanding mean
- Добавление случайных шумов

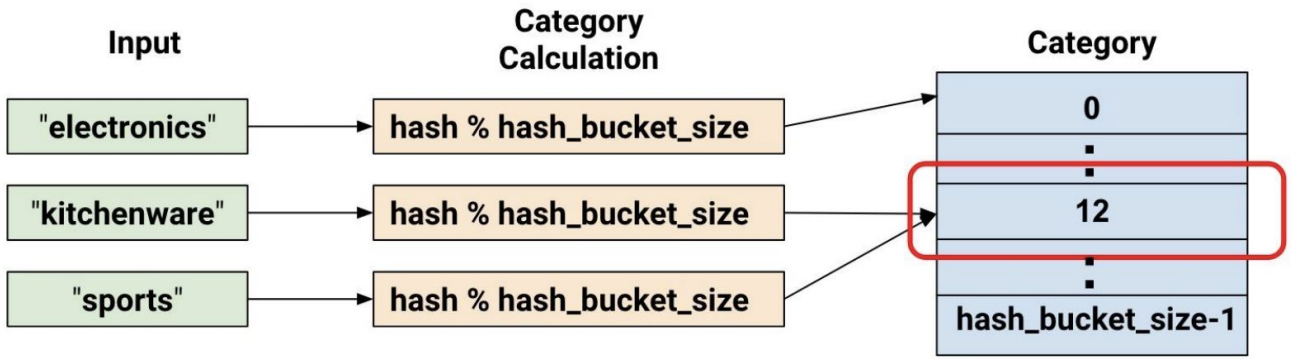
Хэширование признаков

- Если у категориального признака слишком много значений, скажем, миллион, то после применения One-hot кодировки мы получим миллион новых столбцов.

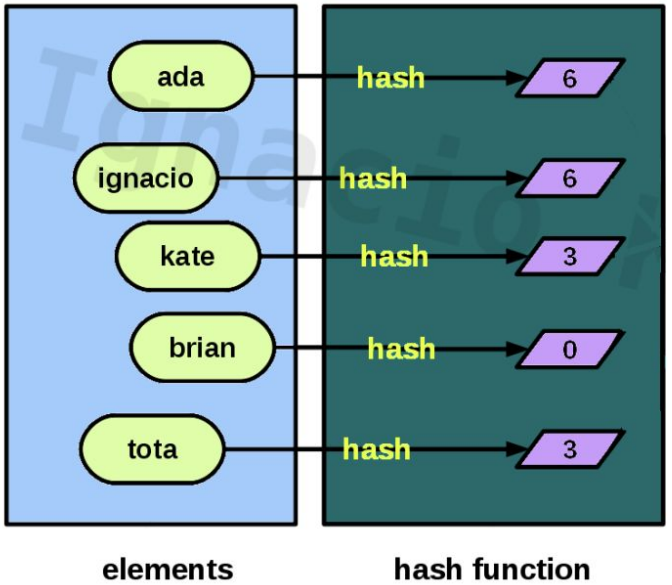
- Хэширование развивает идею one-hot кодирования, но позволяет получать любое заранее заданное число новых числовых столбцов после кодировки.

Алгоритм Хэширования

1. Для каждого значения признака вычисляем значение некоторой функции – хэш-функции (hash)
2. Задаем ***hash_bucket_size*** – итоговое количество различных значений категориального признака
3. Берем остаток: ***hash % hash_bucket_size*** – тем самым кодируем каждое значение признака числом от 0 до ***hash_bucket_size-1***
4. Дальше к полученным числам применяем ***One Hot Encoding***



Пример работы



0	1	2	3	4	5	6
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	1	0	0	0
1	0	0	0	0	0	0
0	0	0	1	0	0	0

Хэширование

- Хэширование – это способ кодирования категориальных данных, принимающих множество различных значений, показывающий хорошие результаты на практике.

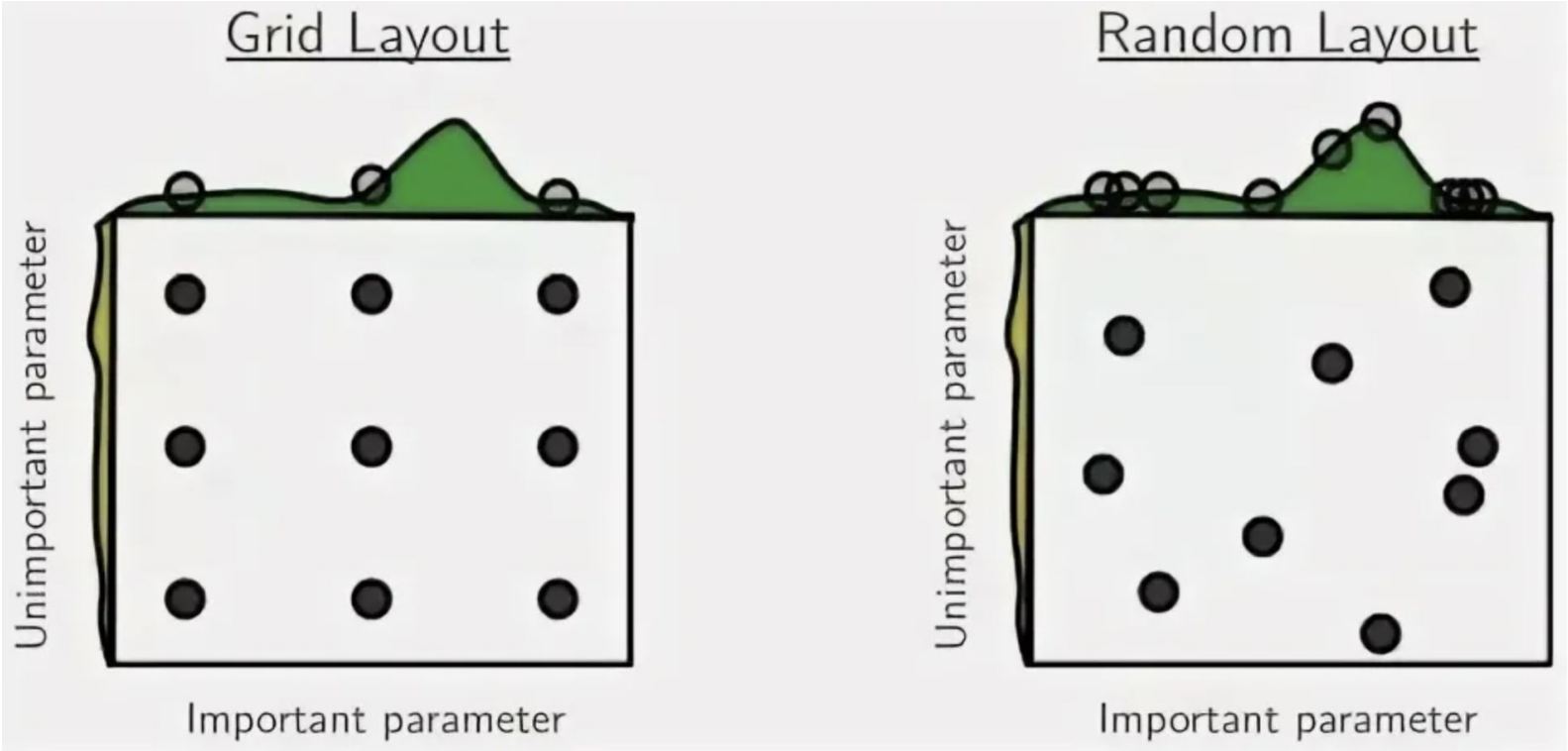
- Хэширование позволяет закодировать любое значение категориального признака (в том числе то, которого не было в тренировочной выборке)

Поиск гиперапарметров

При построении алгоритмов машинного обучения используются два вида параметров:

- Обучаемые, которые получаются в процессе оптимизации функционала ошибки
- Гиперпараметры, которые контролируют процесс оптимизации, и то какими будут веса модели в его процесс

Стандартные стратегии поиска



Байесовская оптимизация