Классическое машинное обучение

Минкин Даниэль

10 апреля 2025 г.

Содержание

1 Функции потерь

1.1 Задачи регрессии

1.1.1 MSE

$$MSE(f, X, y) = \frac{1}{N}\|f(X) - y\|^2_{euclidean} \tag{1}$$

Градиент для линейной регрессии

Раскроем наше выражение

$$\|Aw - b\|^2 = \langle Aw - b, Aw - b\rangle = \langle Aw, Aw\rangle - 2\langle Aw, b\rangle + \langle b, b\rangle \tag{2}$$

Возьмем дифференциал от данного выражения

$$[D_{w_0}(\|Aw - b\|)] = [D_{w_0}(\langle Aw, Aw\rangle)] - 2[D_{w_0}(\langle Aw, b\rangle)] + [D_{w_0}(\langle b, b\rangle)] \tag{3}$$

Мы можем исключить последее слагаемое

$$[D_{w_0}(\langle Aw, Aw\rangle)] - 2[D_{w_0}(\langle Aw, b\rangle)] \tag{4}$$

Рассмотрим дифференциал второго слагаемого

$$b^T A(w + \Delta w) - b^T Aw = b^T A\Delta w = \Delta w^T A^T b \tag{5}$$

Следовательно:

$$[D_{w_0}(\langle Aw, b\rangle)](\Delta w) = \Delta w^T A^T b \tag{6}$$

Рассмотрим дифференциал первого выражения, заметим, что по свойству симметричности произвдения и правилу дифференцирования умножения мы получим

$$[D_{w_0}(\langle Aw, Aw\rangle)] = 2\langle [D_{w_0}(Aw)], Aw_0\rangle \tag{7}$$

Легко показать, что

$$[D_{w_0}(Aw)] = A \tag{8}$$

Следовательно наше выражение приводимо к

$$[D_{w_0}(\langle Aw, Aw\rangle)] = 2\langle A, Aw_0\rangle = 2A^T Aw_0 \tag{9}$$

Приведем все это к одной формуле

$$2A^T Aw_0 - 2A^T b = 2A^T(Aw_0 - b) \tag{10}$$

А теперь поделим все на N, где N — размер выборки, так как это монотонное преобразование не зависящее от w, оно не влияет на дифференциал
Таким образом:

$$[D_{w_0}(MSE)] = \frac{2}{N} A^T (Aw_0 - b) \tag{11}$$

1.1.2 MAE

$$MAE(f, X, y) = \frac{1}{N} \langle \text{sign}(f(X) - y), f(X) - y \rangle \tag{12}$$

Градиент для линейной регрессии

Уберем деление на N, так как это не влияет на дифференциал

$$[D_{w_0}(MAE)] = \frac{1}{N} [D_{w_0}(\langle \text{sign}(Xw - y), Xw - y \rangle)] \tag{13}$$

Тогда

$$\begin{gathered}
[D_{w_0}(\langle \text{sign}(Xw - y), Xw - y \rangle)] \\
= \langle [D_{w_0}(\text{sign}(Xw - y))], Xw - y \rangle + \langle \text{sign}(Xw - y), [D_{w_0}(Xw - y)] \rangle \\
= \langle 0, Xw - y \rangle + \langle \text{sign}(Xw - y), X \rangle \\
= \text{sign}(Xw - y)^* X
\end{gathered}$$

Если Δw — вектор строка
Следовательно:

$$[D_{w_0}(MAE)](\Delta w) = \frac{1}{N} \text{sign}(Xw - y)^* X \Delta w \tag{14}$$

1.1.3 MAPE

$$MAPE(f, X, y) = \frac{1}{N} \text{sign}(f(X) - y)^* \cdot \text{diag}(\frac{1}{|y_1|}, \frac{1}{|y_2|}, \frac{1}{|y_3|} ...) \cdot (f(X) - y) \tag{15}$$

Градиент для линейной регрессии

По аналогии с MAE

$$[D_{w_0}(MAPE)](\Delta w) = \frac{1}{N} \text{sign}(f(X) - y)^* \cdot \text{diag}(\frac{1}{|y_1|}, \frac{1}{|y_2|}, \frac{1}{|y_3|} ...) X \Delta w \tag{16}$$

2 Регуляризация

Регуляризация — наложение штрафа на модель за обучаемые параметры. Регуляризация добавляется в качестве меры для борьбы с переобучением. Для каждой модели регуляризация описывается отдельно

3 Линейные модели

Линейные модели — класс моделей которые используют линейное преобразование для вектора входных фичей.

3.1 Регрессия

Формализуем задачу регрессии, пусть у нас есть вектор $\bar{x} \in \mathrm{R}^n$. Тогда предсказание может быть сделано с помощью такой формулы:

$$y_{pred} = \bar{x} \cdot \bar{w} + w_0 \tag{17}$$

Т.е мы ищем такой вектор $\bar{w} \in \mathrm{R}^{n+1}$, который будет выдавать наиболее близкие y_{pred} к y_{true}.

3.2 Классификация

В случае решения задачи классификации через линейные модели сначала делается предсказание как в случае регрессии, а после к результату предсказания применяется разделяющее правило (например в случае бинарной классификации правило может задаваться так: если $y_{pred} > 0$ мы относим объект к положительному классу — если нет, то к отрицательному)

В случае бинарной классификации с разделяющим правилом из примера уравнение регрессии задает гиперплоскость, которая разделяет исходное пространство: i.e $\sum_{i=1}^{n} w_i \cdot x_i + c > 0$ — плоскость в n-мерном пространстве, которая делит пространство на положительный и отрицательные классы

3.3 Общее

Несколько фактов:

- **При использовании OneHot Encoding-а мы можем избавиться от одной encoded фичи**. Все просто: пусть у нас есть веса для каждой закодированной фичи $w_1, w_2 ... w_n$, также у нас добавляется константа c к предсказанию по фичам. Давайте удалим последнюю фичу, тогда в случае если $x_1 = 0$ ... $x_{n-1} = 0$ нам нужно добавить к константе еще и w_n, иначе результат изменится, следовательно константа в новой модели должна быть равна $w_n + c$. Однако тогда нам нужно внести поправку в веса в случае если один из $x_i \mid i < n$ равен 1, чтобы результат остался таким же. Мы можем просто вычесть из старых весов w_n, за счет того, что мы добавили его к const ничего не изменится. Таким образом мы успешно исключили одну encoded фичу, оставив результаты предсказаний без изменений. **Важно заметить, что если мы работаем в модели без константы, то тогда данный подход не сработает**

- **Для более сложных зависимостей необходимо использовать новые фичи которые являются функциями от старых**. Т.е мы включаем в модель фичи задаваемые как $f(x_1, x_2...x_n)$

- Если между признаками есть приближённая линейная зависимость, коэффициенты в линейной модели могут совершенно потерять физический смысл

3.4 Оценка по МНК

3.4.1 Общее

В первую очередь опустим свободный член, так как можно считать, что у нас просто есть еще один признак, который всегда равен 1. Пусть функция потерь задается как Евклидова норма между предсказанными и истинными значениями. I.е мы решаем следующую задачу:

$$\|Xw - b\|_{euclidean} \to \min_w \tag{18}$$

где X — матрица размера (N, k), N – размер выборки, а k — кол-во фичей, т.е это матрица где в строки записаны вектора, по которым нужно сделать предсказания.

Однако нам также нужно сделать поправку на размер выборки, чтобы значения функции потерь можно было сравнивать между собой для разных выборок. Получается задача выгладит так:

$$\frac{\|Xw - b\|_{euclidean}}{N} \to \min_w \tag{19}$$

Функция потерь является функционалом, так как принимает на вход три значения — матрицу наблюдений, true значения предсказываемой перменной и функцию, которая возвращает некое значение по вектору наблюдений

Значение коэффициентов может быть получено через псевдообратную матрицу, по ее свойству:

$$\|XX^+b - b\|_{euclidean} \leq \|Xw - b\|_{euclidean} \tag{20}$$

для любого w

Так как деление на константу — монотонное преобразование, данное решение минимизирует нашу функцию потерь

> **Важно заметить:** псевдообратная матрица может быть использована и для других норм, однако тогда нам нужно перейти в евклидово пространство A с новым скалярным произведением, тогда псевдообратная матрица будет минимизировать норму задаваемую как $\sqrt{\langle u, u\rangle_A}$. Об этом будет рассказано позже

Таким образом решением данного СЛАУ будет

$$w_* = A^+ b \tag{21}$$

В случае если $N \geq k$

$$w_* = (X^T X)^{-1} X^T b \tag{22}$$

А в противном случае, когда $N < k$

$$w_* = X^T (X X^T)^{-1} b \tag{23}$$

Мы можем считать, что X — матрица полного столбцового или строчного ранга, зачастую у нас не будет полностью ЛЗ столбцов или строк, а даже если они и есть их можно исключить из-за бессмысленности

3.4.2 Невырожденность матрицы $X^T X$

В реальных задачах матрицах $X^T X$ или $X X^T$ являются невырожденными, однако нам нужно оценить насколько они "невырождены", так как у нас есть погрешность при вычислении детерминанта, например мы могли получить неотрицательное значение из-за логики работы чисел с плавающей точкой или же в изначальных данных содержится погрешность.

Число обусловленности

Число обусловленности — максимальное значение отношения относительного изменения функции и относительного измененииия арргмента

$$\mu(f, x) = \max_{\Delta x} \frac{\frac{\|f(x+\Delta x) - f(x)\|}{\|f(x)\|}}{\frac{\|\Delta x\|}{\|x\|}} \tag{24}$$

в малой окрестности Δx

Легко заметить, что чем больше значение $\mu(f, x)$, тем хуже так как наше решение может очень сильно измениться от малого приращения аргумента, а мы бы хотели иметь "стабильное" решение

Покажем на зависимость числа обусловленности задачи $f(A) = (A^T A)^{-1}$ от матрицы корреляции. Мы воспользуемся определением числа обусловленности:

$$\mu(f, A) = \max_{\Delta A} \frac{\frac{\|f(A+\Delta A) - (A^T A)^{-1}\|}{\|(A^T A)^{-1}\|}}{\frac{\|\Delta A\|}{\|A\|}} \tag{25}$$

Тут нам понадобится матричное дифференцирование, нам нужно найти дифференциал $(A^T A)^{-1}$, заметим, что это композиция двух функций. Во-первых, вспомним определение дифференциала, пусть задана функция $f : R^n \to R^m$ Тогда ее дифференциал может быть найден так:

$$f(x_0 + \Delta x) - f(x_0) = [D_{x_0} f](\Delta x) + o(\|\Delta x\|) \tag{26}$$

При этом дифференциал $[D_{x_0} f]$ — линейное отображение из R^n в R^m, т.е мы находим линейную часть приращения отображения

Найдем дифференциал $f(X) = X^T X$ где X - матрица (n, m) и $n > m$. Тогда:

$$\begin{aligned} f(X + \Delta X) - f(X) &= (X + \Delta X)^T (X + \Delta X) - X^T X \\ &= (X^T + (\Delta X)^T)(X + \Delta X) - X^T X \\ &= X^T X + X^T \Delta X + (\Delta X)^T X + (\Delta X)^T \Delta X - X^T X \\ &= X^T \Delta X + (\Delta X)^T X + (\Delta X)^T \Delta X \end{aligned}$$

Можно легко показать, что $[D_X f](\Delta X) = X^T \Delta X + (\Delta X)^T X$ является линейным оператором по ΔX. При этом $(\Delta X)^T \Delta X$ и есть $o(\|\Delta X\|)$.

Найдем дифференциал $g(X) = X^{-1}$. Мы будем использовать равенство $E = X^{-1} X$. Продифференцируем выражение с обеих сторон.

$$0 = [D_{X_0}(X^{-1} X)](H) \tag{27}$$

По правилам дифференцирования умножения мы получаем

$$0 = [D_{X_0}(X^{-1})](H) \cdot X_0 + X_0^{-1} \cdot [D_{X_0}(X)](H) \tag{28}$$

Таким образом

$$[D_{X_0}(X^{-1})](H) \cdot X_0 = -X_0^{-1} \cdot [D_{X_0}(X)](H) \tag{29}$$

Т.е все сводится к формуле

$$[D_{X_0}(X^{-1})](H) = -X_0^{-1} \cdot [D_{X_0}(X)](H) \cdot X_0^{-1} \tag{30}$$

При этом $[D_{X_0}(X)](H) = H$, таким образом итоговая формула имеет вид

$$[D_{X_0}(X^{-1})](H) = -X_0^{-1} \cdot H \cdot X_0^{-1} \tag{31}$$

Мы имеем дело с функцией $g(f(X))$, дифференциал такой функции представим как

$$[D_{f(X_0)}(g)]([D_{X_0}(f)](\Delta X)) \tag{32}$$

Следовательно $[D_{X_0}((X^T X)^{-1})](\Delta X)$, может быть найдено так

$$(X_0^T X_0)^{-1} \cdot (X_0^T \Delta X + (\Delta X)^T X_0) \cdot (X_0^T X_0)^{-1} \tag{33}$$

Таким образом в изначальной формуле числа обусловленности мы можем записать приращение функции как

$$\mu(A) = \max_{\Delta A} \frac{\frac{\|(A^TA)^{-1}\cdot(A^T\Delta A+(\Delta A)^TA)\cdot(A^TA)^{-1}\|}{\|(A^TA)^{-1}\|}}{\frac{\|\Delta A\|}{\|A\|}} \tag{34}$$

Оценим наше выражение сверху, сделаем несколько предположений на будущее, **пусть мы используем одно и тоже семейство норм для входного и выходного пространства и для этого семейства выполняется свойство субмультипликативности**, тогда для $(A^TA)^{-1}$ и $(A^T\Delta A + (\Delta A)^TA)$ можно будет использовать следующее мажорирование

$$\|(A^TA)^{-1}(A^T\Delta A + (\Delta A)^TA)(A^TA)^{-1}\| \leq \|(A^TA)^{-1}\|^2\|A^T\Delta A + (\Delta A)^TA\| \tag{35}$$

В итоге мы можем мажорировать наше выражение так:

$$\frac{\frac{\|(A^TA)^{-1}\cdot(A^T\Delta A+(\Delta A)^TA)\cdot(A^TA)^{-1}\|}{\|(A^TA)^{-1}\|}}{\frac{\|\Delta A\|}{\|A\|}} \leq$$
$$\frac{\|(A^TA)^{-1}\|\cdot\|A^T\Delta A + (\Delta A)^TA\|}{\frac{\|\Delta A\|}{\|A\|}}$$

Продолжая мажорирование мы получим следующую ситуацию

$$\frac{\|(A^TA)^{-1}\|\cdot\|A^T\Delta A + (\Delta A)^TA\|}{\frac{\|\Delta A\|}{\|A\|}} \leq$$
$$\frac{\|(A^TA)^{-1}\|\cdot(\|A^T\Delta A\| + \|(\Delta A)^TA\|)}{\frac{\|\Delta A\|}{\|A\|}}$$

Предположим, что норма выходного пространства устойчива к транспонированию, также вспомним условие про субмультипликативность, тогда наше выражение примет вид

$$\frac{\|(A^TA)^{-1}\|\cdot(\|A^T\Delta A\| + \|(\Delta A)^TA\|)}{\frac{\|\Delta A\|}{\|A\|}} =$$
$$2\cdot\frac{\|(A^TA)^{-1}\|\cdot\|A^T\Delta A\|}{\frac{\|\Delta A\|}{\|A\|}} \leq$$
$$2\cdot\frac{\|(A^TA)^{-1}\|\cdot\|A^T\|\cdot\|\Delta A\|}{\frac{\|\Delta A\|}{\|A\|}} =$$
$$2\cdot\frac{\|(A^TA)^{-1}\|\cdot\|A^T\|}{\frac{1}{\|A\|}} =$$
$$2\cdot\|(A^TA)^{-1}\|\cdot\|A\|^2$$

Таким образом, при определенных требованиях к семейству норм мы получаем, что

$$\max_{\Delta A} \frac{\frac{\|(A^TA)^{-1}\cdot(A^T\Delta A+(\Delta A)^TA)\cdot(A^TA)^{-1}\|}{\|(A^TA)^{-1}\|}}{\frac{\|\Delta A\|}{\|A\|}} \leq$$
$$2\cdot\|(A^TA)^{-1}\|\cdot\|A\|^2$$

Вместо точного максимума мы можем использовать оценку сверху для всех значений.

Покажем влияние на данный результат близости матрицы к "вырожденной". Применим сингулярное разложение к матрице A, если она имеет размеры (n,m), где $n>m$

$$A = U_{(n,n)}\Sigma_{(n,m)}V^T_{(m,m)} \tag{36}$$

Σ — диагональная матрица где на диагонали находятся сингулярные числа матрицы, а U и V — унитарные матрицы, тогда A^TA равно
$V\Sigma^TU^TU\Sigma V^T = V\Sigma^TE\Sigma V^T$. Тогда:

$$(A^TA)^{-1} = (V\Sigma^TE\Sigma V^T)^{-1} \tag{37}$$

При этом $\Sigma^TE\Sigma$ — матрица (m,m), если матрица A имеет размер (n,m), где $n>m$. Так как матрица A имеет m сингулярных значений и является матрицей полного столбцового ранга — матрица $\Sigma^T\Sigma$ является обратимой. Продолжим раскрывать наше выражение

$$(A^TA)^{-1} = V(\Sigma^T\Sigma)^{-1}V^T \tag{38}$$

Мажорируем наше старое выражение, оно будет равно

$$2\cdot\|(A^TA)^{-1}\|\cdot\|A\|^2 \leq 2\cdot\|(\Sigma^T\Sigma)^{-1}\|\cdot\|V\|^2\cdot\|A\|^2 \tag{39}$$

$\Sigma^T\Sigma$ — квадратная матрица на диагонали у которой находятся квадраты сингулярных значений матрицы, следовательно, обратная матрица к ней будет иметь вид $\mathrm{diag}(\frac{1}{\sigma_1^2},\ \frac{1}{\sigma_2^2}\ ...\ \frac{1}{\sigma_m^2})$

$$2\cdot\|(\Sigma^T\Sigma)^{-1}\|\cdot\|V\|^2\cdot\|A\|^2 = 2\cdot\|\,\mathrm{diag}(\frac{1}{\sigma_1^2},\ \frac{1}{\sigma_2^2}\ ...\ \frac{1}{\sigma_m^2})\|\cdot\|V\|^2\cdot\|A\|^2 \tag{40}$$

При этом сингулярные числа показывают близость матрицы к невырожденной, или же близость к матрице полного столбцового/строкового ранга, если матрица A имеет сильную зависимость между столбцами то $\min\{\sigma_1,\sigma_2...\}\to 0$, что ведет к увеличению числа обусловленности (сингулярные числа рассмотрю позже)

Второй подход к числу обусловленности

Мы можем сделать проще и рассмотреть число обусловленности матрицы A как линейного оператора. Рассмотрим уравнение:

$$Ax = b \tag{41}$$

Тогда решением в общем виде является

$$x = A^+ b \tag{42}$$

Рассмотрим число обусловленности

$$\mu(A, b) = \max_{\Delta b} \frac{\frac{\|A^+(b+\Delta b) - A^+ b\|}{\|A^+ b\|}}{\frac{\|b+\Delta b - b\|}{\|b\|}} = \max_{\Delta b} \frac{\frac{\|A^+ \Delta b\|}{\|A^+ b\|}}{\frac{\|\Delta b\|}{\|b\|}} \tag{43}$$

Перенеся деление

$$\max_{\Delta b} \frac{\|A^+ \Delta b\| \cdot \|b\|}{\|\Delta b\| \cdot \|A^+ b\|} = \max_{\Delta b}(\frac{\|A^+ \Delta b\|}{\|\Delta b\|}) \cdot (\frac{\|b\|}{\|A^+ b\|}) \tag{44}$$

Аналогично прошлому случаю затребуем свойство субмультипликативности от нормы, тогда мы можем мажорировать первый множитель с помощью $\|A^+\|$. Следовательно:

$$\mu(A, b) = \|A^+\| \frac{\|b\|}{\|A^+ b\|} = \|A^+\| \frac{\|Ax + \epsilon\|}{\|x\|} \tag{45}$$

где ϵ - вектор отклонений

$$\|A^+\| \frac{\|Ax + \epsilon\|}{\|x\|} \leq \|A^+\| \frac{\|Ax\| + \|\epsilon\|}{\|x\|} \tag{46}$$

Таким образом мы получаем

$$\|A^+\| \frac{\|Ax\|}{\|x\|} + \|A^+\| \frac{\|\epsilon\|}{\|x\|} \leq \|A^+\| \frac{\|A\|x\|}{\|x\|} + \|A^+\| \frac{\|\epsilon\|}{\|x\|} \tag{47}$$

Финальное выражение выглядит так:

$$\|A^+\|\|A\| + \|A^+\| \frac{\|\epsilon\|}{\|x\|} \tag{48}$$

В случае если используется сингулярная норма: Сингулярная норма матрицы — ее максимальное сингулярное значение, а псевдообратная матрица имеет обратные сингулярные значения к исходной матрице (легко показать), таким образом:

$$\|A\|_{spec} = \sigma_{max}(A) \tag{49}$$

и

$$\|A^+\|_{spec} = \frac{1}{\sigma_{min}(A)} \tag{50}$$

Таким образом мы получаем:

$$\mu(A,b) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} + \|A^+\| \frac{\|\epsilon\|}{\|x\|} \tag{51}$$

Если пренебречь вторым слагаемым, то мы получим то, что описано в учебнике ШАДА: “Пожертвовав математической строгостью, мы можем считать, что число обусловленности матрицы X – это корень из отношения наибольшего и наименьшего из собственных чисел матрицы $X^T X$“.

Пояснение: Сингулярные значения матрицы A — это квадратные корни из собственных значений матрицы $A^T A$

Intuition проблемы числа обусловленности

Матрица A — линейное отображение из одного пространства в другое, если в новом пространстве есть вектора с сильной линейной связью, то оно становится более сжатым, т.е непохожие вектора в исходном пространстве могут стать сильно более похожими в новом пространстве за счет того, что оно сжато в размерах, таким образом если мы немного изменим целевой вектор (из нового пространства), то вектор который его образовал может сильно отличаться от того, что мы получили в прошлый раз, так как за счет сжатия они лежат рядом в новом пространстве, однако не в старом, таким образом: число обусловленности — лишь следствие того, что новое пространство сжато

Для наглядности можно рассмотреть, двумерный случай, рассмотрим матрицу перехода в новое пространство

$$\begin{pmatrix} 1 & 1 \\ 1 & 1+\epsilon \end{pmatrix}$$

где $\epsilon > 0$

Мы видим, что $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ отобразится в $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, а $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ в $\left[\begin{smallmatrix}1\\1+\epsilon\end{smallmatrix}\right]$. При маленьких ϵ матрица остается невырожденной, однако пространство “сжимается“ делая $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ и $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ все менее отличимыми в новом пространстве

3.4.3 Переход к новому скалярному произвдениию

Мы хотим перейти в пространство с новым скалярным произведением, однако большинство компьютерных систем заточены под обычное Евклидово скалярное произведение, поэтому нам бы хотелось перейти в такой базис, чтобы операции в этом базисе с использованием “обычного“ скалярного произведения были равносильны операциям с использованием “нового“ скалярного произведения в исходном пространстве

Заметим, что любое произвольное скалярное произведение может быть задано как квадратичная форма, исходя из свойства линейности.

$$\langle u, v \rangle_G = u^T G v \tag{52}$$

Заметим, что G — положительно определена и симметрична, по свойству скалярного произведения. Мы можем найти матрицу перехода P таким образом, через разложение Холецкого.

$$G = P^T P \tag{53}$$

Тогда вектор k и матрица M в новом пространстве превращаются в Pk и PMP^T соответственно

Пример

Пусть мы работаем в новом пространстве и хотим умножить матрицу на вектор в новом пространстве. У нас есть $k' = Pk$ и $M' = PMP^T$, мы хотим, чтобы итоговый вектор имел вид:

$$Mk = \begin{bmatrix} \langle M_{1,*}, k \rangle_G \\ \langle M_{2,*}, k \rangle_G \\ \vdots \\ \langle M_{n,*}, k \rangle_G \end{bmatrix}$$

это равносильно:

$$Mk = \begin{bmatrix} M_{1,*}^T Gk \\ M_{2,*}^T Gk \\ \vdots \\ M_{3,*}^T Gk \end{bmatrix}$$

следовательно

$$[Mk]_{new\ dot\ product} = MGk \tag{54}$$

Рассмотрим линейный оператор MG, тогда в новом базисе данный оператор может быть представлен как $PMGP^{-1}$

$$PMGP^{-1} = PMP^T PP^{-1} = PMP^T \tag{55}$$

Проверим, что это работает

$$P^{-1}(PMP^T Pk) = P^{-1}(PMGk) = MGk \tag{56}$$

Таким образом

$$P^{-1}M'k' = MGk \tag{57}$$

Покажем, что это сработает и для умножения на вектор справа

$$k'^T M' {P^{-1}}^T = k^T P^T PMP^T {P^{-1}}^T = k^T GM \tag{58}$$

Таким образом мы выполнили поставленную перед нами задачу

3.5 Сложность точного решения

Алгоритмическая сложность точного решения для матрицы A размерами (N, D) задается как:

$$O(D^2N + D^3 + DN + D^2) \tag{59}$$

Из них мы тратим D^2N на перемножение матриц A^T и A, а D^3 — на обращение данной матрицы, DN — на умножение A^T на b и также тратим D^2 на умножение обратной матрицы на вектор A^Tb

Рассмотрим способы ускорения данных расчетов

- Во-первых легко заметить, что обращение матрицы имеет кубическую сложность, что плохо для задач с большим количеством фичей, рассмотрим итерационный алгоритм Шульца. Тогда $X_{k+1} = 2X_k - X_kAX_k = X_k(2E - AX_k)$, где A — исходная матрица, нужно заметить, что умножение A на X_k требует $O(D^3)$ итераций, однако на практике это не равносильно такому же количеству итераций, как и при обращении матрицы за счет возможности использования векторизации и распараллеливания расчетов, после нам требуется еще D^2 операций на вычитание и еще D^3 на перемножение итоговой матрицы с X_k. Также доп итерации тратятся на проверку сходимости, которая проводится через такую норму $\|X_kA - E\|$. Таким образом, при долгой сходимости данный способ может проигрывать точным алгоритмам по итеряциям
- Для симметричных матриц хорошо применимы методы Крылова, такие как Conjugate Gradient, MINRES, SYMMLQ, которые дают квадратичную сложность (слишком сложная тема, чтобы тут подробно раскрывать)
- Умножение матриц может быть распараллелено, например на GPU

3.6 Использование разложений для решения задачи

3.6.1 Построим QR разложение матрицы A

В данном разложении столбцы матрицы Q отнормированны и имеют единичную длину, i.e $Q^TQ = E$, а R — верхнетреугольная квадратная матрица, тогда:

$$w = (R^TQ^TQR)^{-1}R^TQ^Tb = (R^TR)^{-1}R^TQ^Tb = R^{-1}R^{T^{-1}}R^TQ^Tb \tag{60}$$

Таким образом итоговая формула будет иметь вид

$$w = R^{-1}Q^Tb \tag{61}$$

Для матрицы A размером (N, D) QR разложение будет иметь сложность сложность $O(ND^2 - \frac{D^3}{3})$, также мы затратим $O(D^3)$ на обращение R (мы можем сократить кол-во итераций за счет того, что R — верхнетреугольная матрица). Плюсами такого решения является то, что мы снижаем кол-во операций перемножения матриц, что повышает численную стабильность и снижает общее кол-во итераций. Суммарное кол-во итераций задается так:

$$O(ND^2 + D^3\frac{2}{3} + DN + D^2) \tag{62}$$

Мы получаем $O(DN)$ за счет перемножения $Q^T b$ и $O(D^2)$ за счет финального перемножения вектора R^{-1} и $Q^T b$. В итоге сложность примерно такая же как и в прошлом случае, однако часть операций структурно отличается от него, что может позволить ускорить алгоритм.

3.6.2 Построим сингулярное разложение матрицы A

Про сингулярное разложение уже было сказано выше. Однако в данном случае мы будем использовать усеченное сингулярное разложение, где матрица с сингулярными значениями является квадратной, а U и V — ортогональны по столбцам (i.e $U^T U = E$ и $V^T V = E$) тогда $A = U\Sigma V^T$. Тогда:

$$w = (V\Sigma U^T U\Sigma V^T)^{-1} V\Sigma U^T b = (V\Sigma\Sigma V^T)^{-1} V\Sigma U^T b \tag{63}$$

Раскроем обратную матрицу

$$w = V^{T^{-1}}\Sigma^{-1}\Sigma^{-1}V^{-1}V\Sigma U^T b = V^{T^{-1}}\Sigma^{-1}U^T b = V\Sigma^{-1}U^T b \tag{64}$$

Данное решение хорошо себя ведет в случае плохой обусловленности матрицы A

3.6.3 Заключение про точное решение

Можно увидеть, что точное решение является достаточно вычислительно сложным, как минимум из-за обращения матрицы, которое при большом количестве фичей будет вносить значительный вклад в итоговую сложность. Так же матрица A очень часто является плохо обусловленной в практических задачах.

3.7 Регуляризация

Зачем нам регуляризация? Рассмотрим влияние числа обусловленности на модель. Мы уже знаем, что число обусловленности влияет на стабильность модели. Однако

Во-первых, при добавлении $L2$ регуляризации в модель регрессии, она будет называться гребневая регрессия (Ridge regression). А в случае $L1$ — название будет лассо регрессия (Lasso regression).

Также заметим, что мы не будем регуляризовать вес, который соответствует константе, пока что опустим это, однако будем помнить об этом. Такая регуляризация просто не имеет смысла

3.7.1 Аналитическое решение задачи MSE с L2 регуляризацией

Нам нужно решить задачу

$$\min_w(X,y)\{\|Xw-y\|_2^2+\lambda\|w\|_2^2\} \tag{65}$$

Продифференцируем функцию $\|Xw-y\|_2+\lambda\|w\|_2$. Для этого мы представим нормы как скалярные произвдения

$$\langle Xw-y,Xw-y\rangle+\lambda\langle w,w\rangle \tag{66}$$

Мы уже знаем дифференциал первой части

$$2A^T(Aw-b)+[D_w(\langle w,w\rangle)] \tag{67}$$

$$[D_w(\langle w,w\rangle)]=2w \tag{68}$$

Следовательно нам нужно найти такой w, что

$$A^T(Aw-b)+\lambda w=0 \tag{69}$$

Это равносильно

$$A^TAw-A^Tb+\lambda w=0 \tag{70}$$

$$(A^TA+\lambda E)w-A^Tb=0 \tag{71}$$

Следовательно

$$(A^TA+\lambda E)w=A^Tb \tag{72}$$

Благодаря этому мы получаем, что

$$w=(A^TA+\lambda E)^{-1}A^Tb \tag{73}$$

Заметим, что для всех собственных значений $A^TA+\lambda E$ выполняется $\lambda_i(A^TA+\lambda E)\geq\lambda$. Так как для любого i верно $\lambda_i(A^TA)\geq 0$. Следовательно, данная матрица обратима

Заметим, что для $L1$ аналитического решения не существует, так как:

$$\begin{gathered}[D_w(\|w\|_1)]=[D_w(\langle w,\operatorname{sign}(w)\rangle)]\\ =\langle[D_w(w)],\operatorname{sign}(w)\rangle+\langle w,[D_w(\operatorname{sign}(w))]\rangle\\ =\langle[D_w(w)],\operatorname{sign}(w)\rangle+\langle w,0\rangle\\ =\langle[D_w(w)],\operatorname{sign}(w)\rangle=\langle 1,\operatorname{sign}(w)\rangle\end{gathered}$$

4 Эмбеддеры в Word2Vec

Цель эмбедера - отобразить некий нечисловой объект в многомерное числовое пространство, так чтобы похожие объекты в старом пространстве получали как можно более сонаправленные вектора в новом.

Существует два основных подхода к созданию эмбедеров в контексте Word2Vec, это

- CBOW — предсказывает слово по контексту. Она работает быстрее, но, в силу своей логики, хуже улавливает семантику редких слов.
- Skip-Gram — предсказывает контекст по слову. Из-за необходимости обрабатывать несколько выходов для каждого слова эта архитектура работает медленнее, зато лучше захватывает семантику редких слов.

Примечание Обе архитектуры основаны на концепции Bag of Words, которая утверждает, что вся информация содержится в совокупности слов без учёта их порядка. Это справедливо только для небольших окон (эмпирическое наблюдение). Проявление данной концепции можно будет наблюдать в используемых функциях потерь

Обе архитектуры изначально использовали softmax выходной слой для предсказния вероятности по всем словам и log-loss функцию потерь, предсказывая вероятности по всем словам словаря, однако более эффективным способом является Negative Sampling, который является надстройкой над оргинальной моделью

4.1 CBOW

Пусть в словаре n слов, каждое из которых имеет векторное представление. Рассмотрим множество окон K, полученных из текстового корпуса. Каждое окно соответствует целевому слову w_{target}. Для каждого $k \in K$ мы стремимся максимизировать:

$$P(w_{target} \mid w_{context}) \to \max$$

Здесь $w_{context}$ определяется как $\text{mean}_{w \in k}(w)$. Этот вектор аккумулирует информацию о контексте.

Вероятность $P(w_m \mid w_{context})$ наблюдать слово w_m в контексте слов из $w_{context}$ вычисляется как $\sigma(w_{target} \cdot w_{context})$.

Однако, мы не можем использовать такую функцию потерь, так как модель будет просто пытаться сделать векторы слова и контекста сонаправленными для каждого окна. К функции максимизации добавляются потери на случайных словах, не входящих в текущее окно контекста, это и есть Negative Sampling. Мы минимизируем вероятность для таких слов, что приводит к итоговой формуле:

$$P(w_{target} \mid w_{context}) - \prod_{w_{rand} \notin k | w_{rand} \neq w_{target}} P(w_{rand} \mid w_{context}) \to \max$$

Это грубая аппроксимация log-loss функции, учитывающей все слова словаря в оригинальной реализации.

С точки зрения оптимизации, формула эквивалентна:

$$\log(P(w_{target} \mid w_{context})) + \sum_{w_{rand} \notin k | w_{rand} \neq w_{target}} \log(P(w_{rand} \mid -w_{context})) \to \max$$

CBOW использует две матрицы: одна кодирует контекст, другая — целевое слово и негативные примеры. Это реализует принцип разделения ответственности: матрица контекста учится понимать контекст слова, а матрица целевых слов учится отражать семантику. Обоснование использования двух матриц можно найти в статьях по теме:

> Предполагается, что слова и контексты принадлежат разным словарям. Например, вектор слова "собака"отличается от вектора контекста "собака". Это связано с тем, что слова редко встречаются в своих собственных контекстах, следовательно модель должна присваивать низкую вероятность P(собака | собака), что требует малой величины $w \cdot w$. В оптимуме $\|w\| \to 0$, что неправильно.

Эта проблема скорее относится к оригинальной реализации. Negative Sampling частично решает данную проблему, так как минимизация вероятности выплняется не по всем словам вне окна, а по случайным словам, не входящим в окно, также нужно учитывать, что при семплировании мы не берем целевое слово, что положительно сказывается на решении данной проблемы

4.2 Skip-Gram

Логика этой модели похожа на CBOW, но с несколькими ключевыми отличиями. Мы работаем с вероятностями $P(w_{context} \mid w_{target})$, где $w_{context}$ — множество векторов контекста. С добавлением негативного семплирования задача оптимизации принимает вид:

$$\sum_{w_i \in k} \log(P(w_i \mid w_{target})) + \sum_{w_{rand} \notin k | w_{rand} \neq w_{target}} \log(P(w_{rand} \mid -w_{target})) \to \max$$

В Skip-Gram используется одна матрица для всех векторов. Это связано с тем, что эмбеддинг должен аккумулировать информацию о контексте: предсказание вероятности контекста следует непосредственно из вектора слова. Проблема описанная выше, которая обосновывает использование двух матриц, решается с помощью Negative Sampling, который не позволяет минимизировать $\|w\|$

4.3 Выравнивание эмбеддингов

Допустим мы используем эмбеддеры для перевода слов с одного языка на другой. У нас могут быть размечены слова переводы для различных слов. Мы хотим отобразить наши вектора в единое векторное пространство, чтобы семантически похожие слова внутри разных языков имели как можно более сонаправленные вектора в этом новом пространстве, так как это работает внутри одного языка. Есть два основных способа:

- MUSE (Multilingual Unsupervised and Supervised Embeddings)
- VecMap

Пусть у нас есть два словаря для двух разных языков W_c — словарь нашего языка, а W_f — словарь иностранных слов. Формально можно считать, что существует два линейных пространства: L_c, L_f, таких, что $\forall w_c \in W_c$: $w_c \in L_c$ и $\forall w_f \in W_f : w_f \in L_f$. Т.е каждый словарь опредлен над L_c и L_f соответственно. Цель выравнивания можно определить как $L_c \to L_f$.

Нам нужно привести словари к единому виду, так как они обучены на разных корпусах. Можно использовать общее подмножество слов. Например через явный перевод, или нахождение общих слов, таких как заимствованные. Мы создаем перекресный словарь по ним. Некоторые методы концентрируются на распределении векторов, не используя перекрестный словаь.

4.3.1 MUSE

Вот статья: https://arxiv.org/abs/1710.04087

Исходя из названия поддерживает как supervised так и unsupervised режимы

Unsupervised

Мы используем GAN для того, чтобы получить вектора с таким же распределением как и в целевом словаре. Однако в данном случае в качестве генератора выступает обыкновенная ортогональная матрица, которая учится обманывать дискриминатор. При этом пары “Слово-Перевод“ семплируются случайно, без смыслового сопоставления, так как наша цель — именно имитация распределения. Авторы оригинальной статьи использовали 50 тысяч самых часто встречающихся слов для обучения генератора для словарей размерами по 200 тысяч. Однако использование всех слов не приводило к значимому ухудшению результатов. Слова выбирались равномерно, так как семплирование пропорционально частоте не дало улучшений. После того как мы обучили GAN, мы приступаем к Procrustes analysis (анализ Прокруста), предварительно применив полученную матрицу ко всем эмбэддингам.

Рассмотрим задачу более подробно:

Наша цель — найти такую матрицу W, что

$$\begin{cases} \|WX - Y\|_F^2 \to \min \\ WW^T = E \\ W^T W = E \end{cases}$$

Аналитически задача решается так:

Нам дано $\text{trace}((WX - Y)^T (WX - Y))$, расскроем транспонирование и получаем $\text{trace}((X^T W^T - Y^T)(WX - Y)) = \text{trace}(X^T W^T WX - X^T W^T Y - Y^T WX + Y^T Y)$. Так как $W^T W = E$. Расскроем выражение $\text{trace}(X^T X - X^T W^T Y - Y^T WX + Y^T Y)$. Так как оптимизация выполняется по W, это равносильно $\text{trace}(-X^T W^T Y - Y^T WX)$. Следовательно

$$\begin{cases} \text{trace}(-X^T W^T Y - Y^T WX) \to \min \\ WW^T = E \\ W^T W = E \end{cases}$$

Мы можем преобразовать $\text{trace}(-X^T W^T Y - Y^T WX) \to \min$ в $\text{trace}(X^T W^T Y + Y^T WX) \to \max$. Так как $\text{trace}(A) = \text{trace}(A^T)$. Вся задача оптимизации сводится к $\text{trace}(Y^T WX) \to \max$. Это можно свести к $\text{trace}(WXY^T) \to \max$. Пусть $M = XY^T$. Тогда нам нужно оптмизировать $\text{trace}(WM)$. По теореме Eckart–Young–Mirsky, максимум будет достигаться в $W = UV^T$

Вот как происходит обучение в рамках Procrustes analysis: Во-первых, мы выбираем наиболее частые слова для данного этапа. Во-вторых, мы образуем синтетический словарь, на основе Cross-Domain Similarity Local Scaling (CSLS). Обсудим это подробнее:

CSLS

Мы не можем просто использовать NN, так как с таким подходом часто возникает ситуация, что для пары (x, y), где y — ближайший к x вектор, далеко не всегда выполняется, что x – ближайший к y вектор. Авторы ссылаются на статью, которая показывает, что в высокоразмерных пространствах это приводит к образованию “хабов“ и “антихабов“ — векторов, которые являются соседями сразу для многих или не практически не являются соседями ни для кого, соответственно. Для решения данной проблемы мы бдуем “штрафовать“ точки за то, что они являются хабами. Пусть мы будем использовать для этого K соседей. Определим пару функций: $N_{c'}(x)$ — функция, которая возвращает первые K соседей вектора x из словаря W'_c, который образован путем применения генератора ко всем эмбеддингам из W_c. Аналогичная функция для W_f — $N_f(x)$. Определим еще две функции:

$$r_{c'}(x) = \frac{1}{K} \sum_{y \in N_{c'}(x)} cos(x, y) \tag{74}$$

где cos — косинусная метрика. Также введем аналогчиную функцию для W_f — $r_{f'}(x)$.

Введем финальную функцию:

$$\text{CSLS}(x, y) = 2 \cdot cos(x, y) - r_{c'}(y) - r_f(x) \tag{75}$$

где $x \in W_{c'}$ и $y \in W_f$

Получается для каждого языка мы ищем соседей в противоположном. Такая фнукция сходства штрафует векторы, которые лежат в плотных областях. И по наблюдениям авторов метода использование CSLS дает значительное улучшение показателей.

Мы создаем синтетический словарь, ищя MNN (Mutal Nearest Neigh.) по CSLS на каждой итерации. Далее, для этого словаря мы ищем матрицу W по алгоритму выше и применяем ко всем эмбэддингам. Процесс повторяется итерационно. Согласно эмпирическим данным, обычно хватает от 5 до 10 итераций. Итоговая функция для трансформации вектора в новое пространство имеет вид

$$\prod_{i=0}^{n-1} W_{n-i} \cdot W_G \cdot x \tag{76}$$

где W_G - матрица генератора

Supervised

Если у нас есть перекрестный словарь, то мы переходим сразу к шагу Procrustes анализа. Однако нужно учитывать, что, несмотря на название, метод разрабатывался для unsupervised режима работы

4.3.2 VecMap

В первую очередь мы нормализуем эмбэддинги, а также центрируем каждую координату. После этого мы снова применяем нормализацию. Постром матрицы $M_X = X^T X$ и $M_Y = Y^T Y$ (тут предполагается, что X и Y — словари в которые по столбцам записаны эмбэддинги). Если пространства эмбэддингов полностью изометричны, то существует такая перестановка строк или столбцов что матрицы M_X и M_Y будут равны. Тут предполагается, что размеры словарей по которым будет производиться выравнивание равны, также как и размерности эмбеддингов. На практике изометрия выполняется приближённо, а поиск перестановок NP-полная задача.

Возьмем и независимо отсортируем каждую строку в M_X и в M_Y, получив матрицы sorted(M_X) и sorted(M_Y). При условии идеальной изометрии получившиеся матрицы будут равны с точностью до перестановки строк. Исходя из предположения, что изометрия приближенно выполняется мы можем искать ближайших соседей для строки из одной матрицы среди строк другой матрицы и считать, что найденный ближайший сосед является репрезентацией исходной строки в другом пространстве.

Важное замечание: если мы используем SVD для матриц эбэддингов (т.е $X = U\Sigma V^T$) то получим, что $M_X = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$. Авторы

оригинальной статьи заметили, что использование матрицы $\sqrt{M_X} = V\Sigma V^T$ более эффективно (в оригинале использовалась матрица $U\Sigma U^T$, так как эмбеддинги были записаны по строкам).

Процесс обучения мы начинаем с формирования небольшого словаря, например 25 слов. Для его формирования мы используем алгоритм описанный выше (в оригинальной статье использовались 4000 наиболее частых слов для создания начального словаря). Далее по известному алгоритму из Procrustes анализа находим ортгональную матрицу W, используя только эмбеддинги из словаря, которую после применяем ко всем эмбеддингам. Далее мы пытаемся добавить новое слово в словарь, для этого мы используем алгоритм из начала. Для этого мы ищем ближайшего соседа для каждой строки из корня sorted версии матрицы схожести одного языка в такой же матрице другого языка, пару наиболее похожих слов мы добавляем в словарь. После снова переходим к поиску матрицы W, итеративно повторяя весь процесс с новым словарем.

Однако такой сырой алгоритм нуждается в дополнениях. Вот некоторые из них:

- Dropout. Для каждого вектора при поиске его соседей, каждый сосед с вероятностью $1 - p$ игнорируется, т.е мы зануляем веса связи с вероятностью $1 - p$. В предложенной авторами метода реализации мы начинаем с $p = 0.1$ (сильный стохастический шум). Если целевая функция не улучшается 50 итераций подряд, увеличиваем p в 2 раза (аналог simulated annealing).
- Использование наиболее частых слов. Для обучения мы используем лишь первые 20000 наиболее частых слов в каждом языке.
- MNN + CSLS. Как и в MUSE, для поиска нового слова мы используем MNN + CSLS, вместо обычного NN.

5 Контекстуальные эмбэддеры

Нужно заметить, что сейчас статические эмбэддеры не используются, так как на рынке доминируют контекстуальные эмбэддеры, т.е вектор слова меняется в зависимости от контекста в котором оно находится

Примеры контекстуальных эмбэддеров:

- **ELMo**. Использует двунаправленные LSTM
- **BERT**. Использует трансформеры и учится на задачах маскирования. Маскирование похоже на CBOW, часть слов в окне маскируется и заменяется на токен маски. После этого модель пытается предсказать на основе контекста, что за слово скрыто за токеном маски.

(Информация по ним будет внесена позже)

6 ONNX

ONNX (Open Neural Network Exchange) - формат для представления моделей из разных фреймворков в унифицированном виде. Это позволяет передавать модели между ними, а также ONNX удобен для инференса