Векторное квантование. Алгоритм Линде-Бузо-Грея

5 января 2023 г. 1:28

Говоря о сжатии данных с потерями, мы принимаем во внимание ошибки, т.е. искажение данных.

Существует понятие "мера искажения", показывающее различие между множеством сообщений на выходе источника и аппроксимирующим множеством.

За аппроксимацию входного вектора **x** вектором **y** отвечает квантование.

Виды квантования:

- *Скалярное*, если входные символы обрабатываются независимо.
  - Равномерное.
  - Неравномерное.
- *Векторное*, если входные символы обрабатываются блоками.

Скалярное квантование	Векторное квантование
Аппроксимирующие значения $Y = \{y_i\} \subset X$	Кодовая книга $B = \{\boldsymbol{b}_i\} \subset X^n$
$\lvert Y \rvert$=M	$\lvert B \rvert$=M
$R = H(Y) \leq \log M$	$R = \frac{H(B)}{n} \leq \frac{\log M}{n}$
Кванты Δ_i	Решающие области S_i

размер
забрать

В случае векторного квантования для **x** выбирается b_i (аппроксимирующий **x** вектор) такой, что при $i \neq j$ даст наилучший результат (формулой записывается так: $\|x - b_i\|^2 \leq \|x - b_j\|^2$ при $i \neq j$, эта формула определяет границы разрешающей области S_i от S_j). Т.е. b_i ближе к **x**, чем b_j.

Чтобы найти искажение:

$$D = \frac{1}{n} \sum_j \sum_{\boldsymbol{x} \in S_j} \|\boldsymbol{x} - \boldsymbol{b}_j\|^2 \, p(\boldsymbol{x})$$

1/n, т.к. в среднем на символ,
суммируем по j, т.к. по всем разрешающим областям
суммируем по всем **x**, принадлежащим разрешающим областям
Внутри этого всего:
p(x) - вероятность, что такой x вообще возникнет
$\|x - b_j\|^2$ - квадратичная мера искажения

Если мы хотим минимизировать искажение, то вводим допущение, что в каждой разрешающей области иксы присутствуют примерно равномерно ($p\left(\mathbf{x}\right) = \frac{1}{|S_j|}$). Ищем производную функции искажения по b_j, приравниваем к нулю (т.е. ищем такое b_j, которое бы давало минимум искажения), тогда:

$$\boldsymbol{b}_j = \frac{\sum\limits_{\boldsymbol{x} \in S_j} \boldsymbol{x}}{|S_j|}, \quad j = 1, ..., M.$$

Эта идея лежит в основе алгоритма Линдо-Бузе-Грея, который выполняет поиск ближайших (или наилучших) векторов, которые входят в кодовую книгу. Алгоритм минимизирует искажение при заданном M (M - кол-во векторов, которыми мы хотим аппроксимировать K векторов).
Т.е. на входе K векторов, и мы хотим их аппроксимировать M векторами (M - размер кодовой книги, по-другому)
Также есть доп.параметры:
T - ограничивает сложность, т.к. алгоритм итеративный, T позволяет ограничить эти итерации
Эпсилон - точность выполнения

Как работает алгоритм:
Имеется два искажения: одно искажение на предыдущем шаге, одно - на текущем
Если в какой-то момент искажение меньше, чем эпсилон, или достигается ограничение на итерации T, то алгоритм заканчивает работу
В противном случае он выполняется.
В нем используется формула, позволяющая вычислить b, ближайший к x.
Т.е. вычисляется b, ближайший к x. Вычисляется искажение. Обновляется значение s, которое суммирует все x, которые у него попали (попали в b ?). И накапливает значение N, показывающее, сколько этих x. Это делается K раз (K - кол-во векторов на входе)
После этого происходит цикл от 1 до M, который вычисляет очередной b как частное s/N.
В конце искажение делится на K (D/K), поскольку мы ищем среднее искажение.

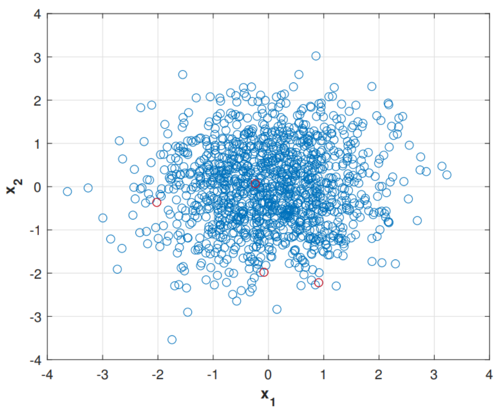
Input: Векторы $(\boldsymbol{x}_1, ..., \boldsymbol{x}_K)$, $\boldsymbol{x}_i \in X^n$, объём книги M, ε, T

$D \leftarrow \infty$, $D_c \leftarrow \max\limits_{i=1,...,K} \|\boldsymbol{x}_i\|^2$.
Выбрать начальные $\boldsymbol{b}_i$, $i = 1, ..., M$..
while $D - D_c > \varepsilon$ and $t < T$ **do**
  for j=1,...,M **do**
    $N_j \leftarrow 0$, $\boldsymbol{s}_j \leftarrow \boldsymbol{0}$
  end for
  $D_c \leftarrow D$, $D \leftarrow 0$.
  for i=1,...,K **do**
    Найти $\boldsymbol{b}_j$, ближайший к $\boldsymbol{x}_i$.
    $D \leftarrow D + \|\boldsymbol{x}_i - \boldsymbol{b}_j\|^2$.
    $\boldsymbol{s}_j \leftarrow \boldsymbol{s}_j + \boldsymbol{x}_i$.
    $N_j \leftarrow N_j + 1$.
  end for
  for j=1,...,M **do**
    $\boldsymbol{b}_j = \boldsymbol{s}_j / N_j$.
  end for

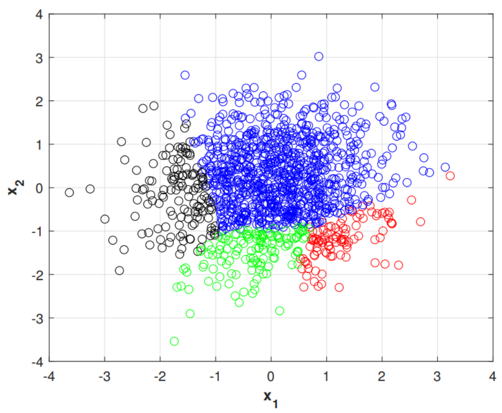
$D \leftarrow D/K$.
$t \leftarrow t + 1$.
end while

Пусть у нас имеется некоторое множество точек, которое мы хотим аппроксимировать всего четырьмя точками, которые минимизируют искажения для всего множества. В результате алгоритмом будут выбраны четыре стартовые точки (помечены красным на 1 картинке), и они будут представлять четыре разрешающие области (черную, синюю, красную и зеленую). На первой итерации заметна явная неравномерность, некоторым областям принадлежит больше точек. После 10 итераций точки сместятся и образуют четыре равномерные разрешающие области.
(При этом D заметно менялось на первых 5-6 итерациях, далее ошибка практически не изменялась)

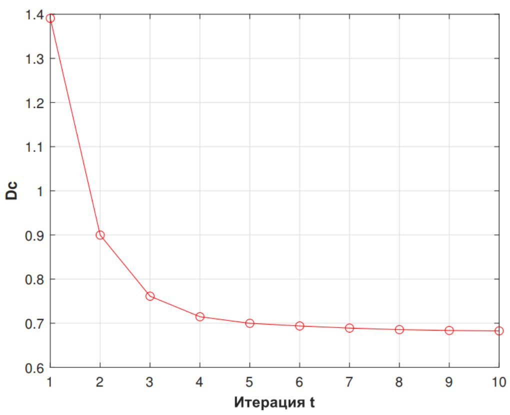
Алгоритм Линде-Бузо-Грея. Итерация $t = 1$



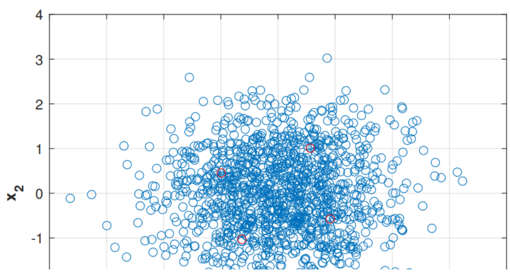
Алгоритм Линде-Бузо-Грея. Итерация $t = 1$

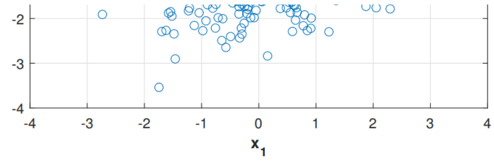


Алгоритм Линде-Бузо-Грея. Изменение D на итерации t

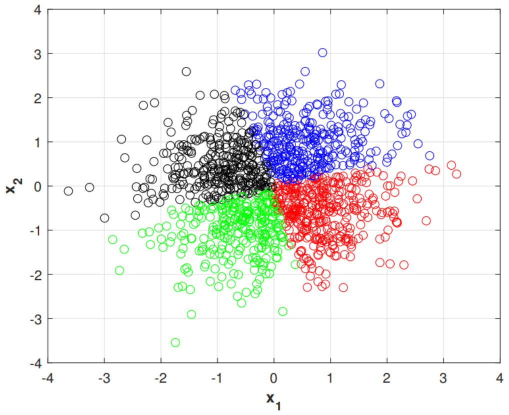


Алгоритм Линде-Бузо-Грея. Итерация $t = 10$





Алгоритм Линде-Бузо-Грея. Итерация $t = 10$



(тут получилось 4 области, в каждой из них есть одна конкретная точка, которая будет выдаваться вместо остальных, в нее входящих - та самая аппроксимация)