Алгоритм PPM-D

5 января 2023 г. 1:26

Контекстное адаптивное арифметическое кодирование
Его идея заключается в том, чтобы кодировать каждый символ, учитывая как можно больше предыдущих символов
t - кол-во предыдущих символов, учитываемых алгоритмом (контекст)

- Если распределение вероятностей вида $\hat{p}(x_{t+1}|x_1, ..., x_t)$ возможно оценить с достаточной точностью, то арифметический кодер может использовать эти вероятности для достижения максимального сжатия, т.е., среднюю длину кодового слова, близкую к $H(X|X^\infty)$.
- Каждое условие соответствует источнику без памяти, к которому может быть применено универсальное кодирование.
- Избыточность адаптивного кодирования приближается к нулю с ростом длины кодируемой последовательности. Поэтому в каждый контекст должно попасть достаточное количество символов, т.е., контекстов не должно быть слишком много.
- Эффективность такого кодирования сильно зависит от метода оценки вероятностей.

индексы от $t-d+1$ до t

Строка (как бы подстрока входной последовательности) $\boldsymbol{s} = \boldsymbol{x}_{t-d+1}^{t}$ длины $d \le D$, предшествующая x_{t+1}, является контекстом для x_{t+1}, если $\boldsymbol{s}$ уже появлялось в $\boldsymbol{x}_1^{t-1}$.
D - ограничение на память и на вычислительную сложность

Пример определения контекста двух букв (отмеченных красным)
Пример. THE_CAT_IN_THE_CAR_ATE_THE_RAT

t	буква	контекст
16	A	THE_C
27	R	THE_

PPM - prediction by partial matching - предсказание по частичному совпадению
Основные этапы кодирования символа x_{t+1}:

1. Выполняется поиск контекста $\boldsymbol{s} = \boldsymbol{x}_{t-d+1}^{t}$ наибольшей длины d, не превышающей D.
2. Для всех возможных значений символа x_{t+1} вычисляются оценки условных вероятностей символа при известном контексте $\boldsymbol{s}$.
3. Значение символа x_{t+1} кодируется арифметическим кодом в соответствии с вычисленной условной вероятностью.

сколько раз появлялось s в контексте s / сколько раз этот контекст s уже появлялся

- D – параметр алгоритма.
- Вероятность того, что символ $x_{t+1} = a$ после контекста $\boldsymbol{s} = \boldsymbol{x}_{t-d+1}^{t}$ может быть оценена как

$$\hat{p}_t(a|\boldsymbol{s}) = \frac{\tau_t(\boldsymbol{s}, a)}{\tau_t(\boldsymbol{s})}.$$

- Если буква для данного контекста не встречалась, т.е. $\tau_t(\boldsymbol{s}, a) = 0$, то арифметический кодер не сможет использовать $\hat{p}_t(a|\boldsymbol{s}) = 0$. Поэтому используется *esc*-символ.
- Если $\hat{p}_t(a|\boldsymbol{s}) = 0$, то передаётся *esc*-символ и контекст укорачивается на одну букву.

последовательность должна была уже до этого встречаться, тогда она может стать контекстом, поэтому > 0

Тот же алгоритм в ином представлении

1: Находим наибольшее d такое, что $\hat{p}_t(\boldsymbol{x}_{t-d+1}^{t}) > 0$, $d \le D$.
2: Выбираем контекст $\boldsymbol{s} \leftarrow \boldsymbol{x}_{t-d+1}^{t}$. если эта буква при нашем контексте s не встречалась...
3: **while** $\hat{p}_t(x_{t+1}|\boldsymbol{s}) = 0$ **do**
4: Кодируем *esc* в соответствии с $\hat{p}_t(esc|\boldsymbol{s})$.
5: Уменьшаем длину контекста: $d \leftarrow d - 1, \boldsymbol{s} \leftarrow \boldsymbol{x}_{t-d+1}^{t}$.
6: **end while** контекст нашелся (хотя бы длиной в 1 символ)
7: **if** $d > 0$ **then**
8: Кодируем x_{t+1} в соответствии с $\hat{p}_t(x|\boldsymbol{s})$.
9: **else** контекста не было, но символ появлялся
10: **if** $\hat{p}_t(x) > 0$ **then**
11: Кодируем x_{t+1} в соответствии с $\hat{p}_t(x)$
12: **else**
13: Передаём *esc* и кодируем x_{t+1} в соответствии с равномерным распределением на не встречавшихся в $\boldsymbol{x}_1^t$ буквах.
14: **end if**
15: **end if**

Алгоритм D:

$$\hat{p}_t(a|\boldsymbol{s}) = \frac{\tau_t(\boldsymbol{s},a)-1/2}{\tau_t(\boldsymbol{s})}; \quad \hat{p}_t(esc|\boldsymbol{s}) = \frac{M_t(\boldsymbol{s})}{2\tau_t(\boldsymbol{s})}, \tau_t(\boldsymbol{s},\boldsymbol{a}) > 0,$$

где $M_t(\boldsymbol{s})$ – число различных букв, появившихся в последовательности длины t вслед за контекстом $\boldsymbol{s}$.

Пример "последний алгоритм PPM"
Предположим, мы нашли контекст s, и оказалось, что для него нужно передавать esc-символ, т.е. ни разу не было буквы а за этим контекстом s.
Передаем esc-символ и сокращаем контекст на 1.
Далее:

После получения *esc* декодер всё еще не знает x_{t+1}, но он знает, какие символы не могут быть на этой позиции. Это символы, которые следовали за контекстом $\boldsymbol{s}$, когда он появлялся ранее. Это знание позволяет уточнять вероятность $\hat{p}(x_{t+1}|\boldsymbol{s}') = (s_{d-1}, ..., s_1)$.

Список исключаемых символов растёт, если *esc* снова передаётся.

Исключение некоторых символов увеличивает оценку вероятности для оставшихся символов, т.е., увеличивает сжатие.

Пример.

- Рассмотрим контекст $\boldsymbol{s}$ =''*кор*'' за которым следует буква ''*т*'', причем ''*корт*'' ранее не встречалось. При этом предположим, что ранее после $\boldsymbol{s}$ встречались ''*а*'' и ''*с*''.
- Передаём *esc* и укорачиваем контекст до $\boldsymbol{s}$ =''*ор*''.
- Тогда τ'_t(ор)=τ_t(ор)-τ_t(ора)-τ_t(орс). ← не учитываем