Алгоритмы оптимизации

Подготовила:
Студентка ИА-42
Дяченко Каринэ

Оптимизация — модификация системы для улучшения
её эффективности.

Цель оптимизации - получение оптимальной системы.

НО истинно оптимальная система в процессе оптимизации достигается далеко не всегда.

«Преждевременная оптимизация — это корень всех бед». Тони Хоар

Нужно иметь:

- Озвученный алгоритм
- Работающий прототип

Оптимизация обычно обозначает модификацию кода и его настроек компиляции для данной архитектуры для производства более эффективного ПО.

Пример

- Задача

```
int i, sum = 0;
for (i = 1; i <= N; i++)
sum += i;
```

Для большей эффективности, если нет переполнения:

```
int sum = (N * (N+1)) / 2;
```

TRADEOFF

Компромиссы

Оптимизация в основном фокусируется на одиночном или повторном времени выполнения, использовании памяти, дискового пространства, пропускной способности или некотором другом ресурсе. Это обычно требует компромиссов — один параметр оптимизируется за счёт других.

Алгоритмы:

1. Алгоритм Витерби

алгоритм поиска наиболее подходящего списка состояний (называемого *путём Витерби*), который в контексте цепей Маркова получает наиболее вероятную последовательность произошедших событий.

Алгоритм декодирования кода, передаваемого по сетям с наличием шума

Алгоритм

Пусть у нас есть скрытая марковская модель с пространством состояний S, начальными вероятностями π_i нахождения в состоянии i и вероятностями $a_{i,j}$ перехода из состояния i в состояние j. Пусть мы наблюдаем на выходе $y_1, \ldots, y_T$. Тогда наиболее вероятная последовательность состояний $x_1, \ldots, x_T$ задается рекуррентными соотношениями:[2]

$$\begin{aligned} V_{1,k} &= \mathrm{P}(y_1 \mid k) \cdot \pi_k \\ V_{t,k} &= \mathrm{P}(y_t \mid k) \cdot \max_{x \in S} (a_{x,k} \cdot V_{t-1,x}) \end{aligned}$$

Здесь $V_{t,k}$ — это вероятность наиболее вероятной последовательности состояний, ответственная за появление первых t наблюдаемых символов, завершающаяся в состоянии k. Путь Витерби может быть найден при помощи указателей, запоминающих, какое состояние x появлялось во втором уравнении. Пусть $\mathrm{Ptr}(k, t)$ — функция, которая возвращает значение x, использованное для подсчета $V_{t,k}$, если $t > 1$, или k если $t = 1$. Тогда

$$\begin{aligned} x_T &= \arg\max_{x \in S}(V_{T,x}) \\ x_{t-1} &= \mathrm{Ptr}(x_t, t) \end{aligned}$$

Здесь мы используем стандартное определение arg max.

Сложность этого алгоритма равна $O(T \times |S|^2)$.

Предположения алгоритма:

- наблюдаемые и скрытые события должны быть последовательностью. Последовательность чаще всего упорядочена по времени.
- две последовательности должны быть выровнены: каждое наблюдаемое событие должно соответствовать ровно одному скрытому событию
- вычисление наиболее вероятной скрытой последовательности до момента t должно зависеть только от наблюдаемого события в момент времени *t*, и наиболее вероятной последовательности до момента t – 1.

Использование

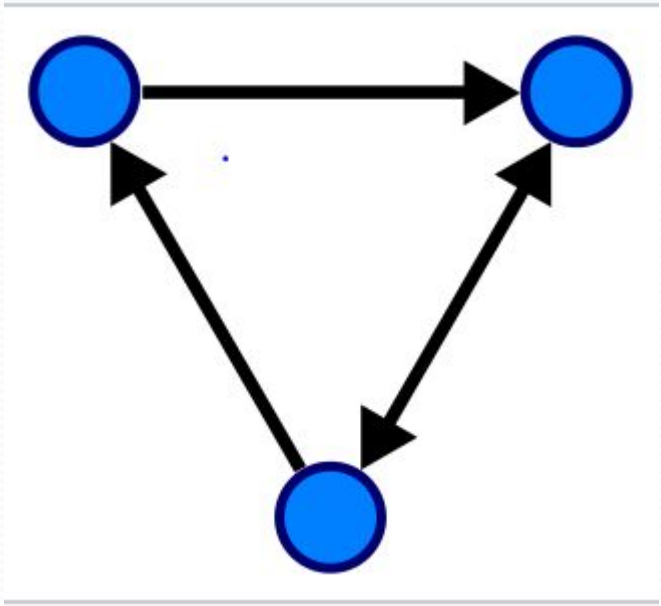
- Декодирование кода мобильных телефонов стандартов GSM и CDMA
- Dial-up модемы - сервис, позволяющий компьютеру, используя **модем** и телефонную сеть общего пользования, подключаться к другому компьютеру (серверу доступа) для инициализации сеанса передачи данных (например, для доступа в сеть Интернет).
- Беспроводные сети стандарта 802.11

Так же широко используется в:

- Распознавании речи
- Синтезе речи
- Компьютерной лингвистике
- Биоинформатике

2. Алгоритм Флойда-Уоршелла

алгоритм для нахождения кратчайших расстояний между всеми
вершинами взвешенного ориентированного графа.



Алгоритм

Пусть вершины графа $G = (V,\ E),\ |V| = n$ пронумерованы от 1 до n и введено обозначение d_{ij}^k для длины кратчайшего пути от i до j, который кроме самих вершин $i,\ j$ проходит только через вершины $1 \dots k$. Очевидно, что d_{ij}^0 — длина (вес) ребра $(i,\ j)$, если таковое существует (в противном случае его длина может быть обозначена как ∞).

Существует два варианта значения $d_{ij}^k,\ k \in (1,\ \dots,\ n)$:

1. Кратчайший путь между $i,\ j$ не проходит через вершину k, тогда $d_{ij}^k = d_{ij}^{k-1}$
2. Существует более короткий путь между $i,\ j$, проходящий через k, тогда он сначала идёт от i до k, а потом от k до j. В этом случае, очевидно, $d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1}$

Таким образом, для нахождения значения функции достаточно выбрать минимум из двух обозначенных значений.

Тогда рекуррентная формула для d_{ij}^k имеет вид:

d_{ij}^0 — длина ребра $(i,\ j)$;

$d_{ij}^k = \min(d_{ij}^{k-1},\ d_{ik}^{k-1} + d_{kj}^{k-1})$.

Алгоритм Флойда-Уоршелла последовательно вычисляет все значения $d_{ij}^k, \forall i,\ j$ для k от 1 до n. Полученные значения d_{ij}^n являются длинами кратчайших путей между вершинами $i,\ j$.

Сравнение с другими алгоритмами

Алгоритм Флойда — Уоршелла является эффективным для расчёта всех кратчайших путей в плотных графах, когда имеет место большое количество пар рёбер между парами вершин.

Из-за большого константного фактора времени выполнения преимущество при вычислениях над алгоритмом Флойда — Уоршелла проявляется только на больших графах.

3. Алгоритм динамической трансформации временной шкалы

алгоритм, позволяющий найти оптимальное соответствие между временными последовательностями. Впервые применен в распознавании речи, где использован для определения того, как два речевых сигнала представляют одну и ту же исходную произнесённую фразу. Впоследствии были найдены применения и в других областях.

Классический алгоритм

Рассмотрим два временных ряда Q длины n и C длины m[4]:

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n; \qquad (1)$$
$$C = c_1, c_2, \ldots, c_j, \ldots, c_m. \qquad (2)$$

Первый этап алгоритма состоит в следующем. Мы строим матрицу d порядка $n \times m$ (матрицу расстояний), в которой элемент $d_{i\ j}$ есть расстояние $d(q_i, c_j)$ между двумя точками q_i и c_j. Обычно используется евклидово расстояние: $d(q_i, c_j) = (q_i - c_j)^2$, или $d(q_i, c_j) = |q_i - c_j|$. Каждый элемент (i, j) матрицы соответствует выравниванию между точками q_i и c_j.

На втором этапе строим матрицу трансформаций (деформаций)D, каждый элемент которой вычисляется исходя из следующего соотношения:

$$D_{i\ j} = d_{i\ j} + min(D_{i-1\ j}, D_{i-1\ j-1}, D_{i\ j-1}). \qquad (3)$$

После заполнения матрицы трансформации, мы переходим к заключительному этапу, который заключается в том, чтобы построить некоторый оптимальный путь трансформации (деформации) и DTW расстояние (*стоимость пути*). Путь трансформации W — это набор смежных элементов матрицы, который устанавливает соответствие между Q и C . Он представляет собой путь, который минимизирует общее расстояние между Q и C. k-ый элемент пути W определяется как $w_k = (i, j)_k, \quad d(w_k) = d(q_i, c_j) = (q_i - c_j)^2$.

Таким образом:

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K; \qquad max(m, n) \leqslant K < m + n,$$

где K — длина пути.

Условия пути трансформации:

- Граничные условия
- Непрерывность
- Монотонность

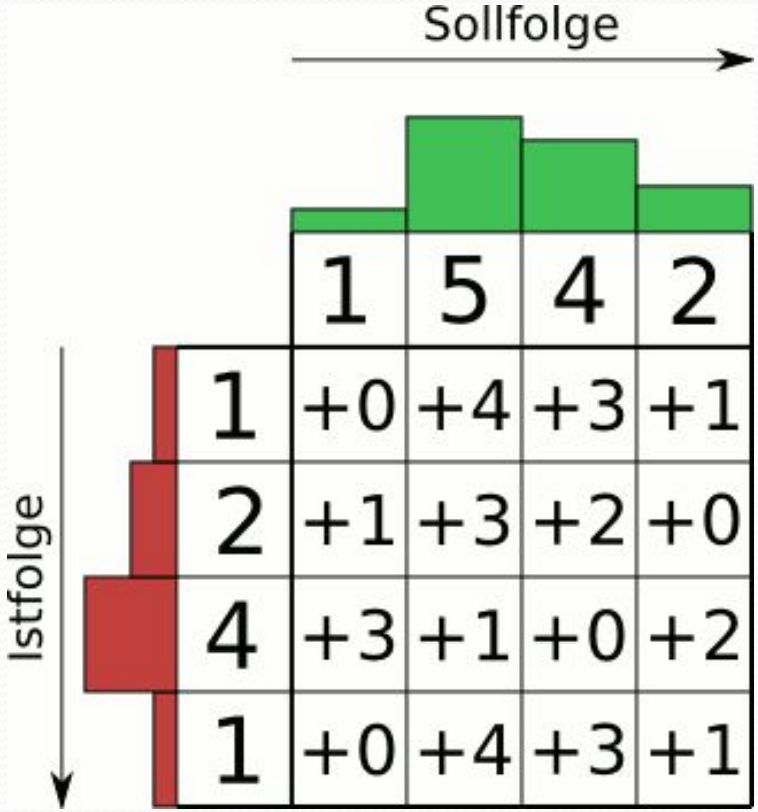
Хотя существует большое количество путей трансформации, удовлетворяющих всем вышеуказанным условиям, однако нас интересуют только тот путь, который минимизирует DTW расстояние (*стоимость пути*).

DTW расстояние (*стоимость пути*) между двумя последовательностями рассчитывается на основе оптимального пути трансформации с помощью формулы:

$$DTW(Q,C) = min\left\{\frac{\sum_{k=1}^{K} d(w_k)}{K}\right\}. \qquad (4)$$

K в знаменателе используется для учёта того, что пути трансформации могут быть различной длины.

Пространственная и временная сложность алгоритма — квадратичная, $O(nm)$, так как DTW алгоритм должен изучить каждую клетку матрицы трансформации.



Построение матрицы трансформаций и выбор оптимального пути трансформации

Разновидности DTW алгоритма

Различные доработки DTW алгоритма предназначены для ускорения его вычислений, а также для того, чтобы лучше контролировать возможные маршруты путей трансформации.

- Быстрый DTW-алгоритм
- Разреженный DTW-алгоритм

Быстрый

Этот алгоритм обладает линейной пространственной и временной сложностью. Быстрый DTW алгоритм использует многоуровневый подход с тремя ключевыми операциями:

- Уменьшение детализации
- Планирование
- Обработка

Разреженный

Основная идея данного метода состоит в том, чтобы динамически использовать наличие подобия и/или сопоставления данных для двух временных последовательностей.

Данный алгоритм имеет три особых преимущества:

- Матрица трансформации представляется с помощью разреженных матриц, что приводит к уменьшению средней пространственной сложности по сравнению с другими методами.
- Разреженный DTW алгоритм всегда выдает оптимальный путь трансформации.
- Так как алгоритм выдает оптимальное выравнивание, то он может быть легко использован в сочетании с другими методами.

Недостатки алгоритма

- Хотя алгоритм успешно используется во многих областях, он может выдавать неправильные результаты.
- Другая проблема заключается в том, что алгоритм может не найти очевидное выравнивание двух рядов вследствие того, что особая точка (пик, впадина, плато, точка перегиба) одного ряда расположена немного выше или ниже соответствующей ей особой точки другого ряда.

Области применения

- распознавание речи
- интеллектуальный анализ данных
- распознавание жестов
- робототехника
- медицина
- биоинформатика
- верификация подписи

Спасибо за внимание!