Не шуми!

Алексей Горностаев и Дамир Салахутдинов

1. Постановка задачи

В этом разделе необходимо ввести все обозначения, формализовать постановку задачи и объяснить, почему формализация именно такая.

От нас требовалось восстановить исходный сигнал, который должен удовлетворять следующим требованиям: похож на испорченный сигнал является гладким, то есть следующее значение и слабо отличается от предыдущего А также для проверки масштабируемости используемого метода решения научиться генерировать испорченные схожим образом сигналы произвольной длины.

Определление понятий:

Под сигналом (испорченным) мы понимем последовательность значений $\{y_i\}_{i=1}^N$.

Под шумом мы понимаем подпоследовательность значений $\{y_i\}_{i=k}^{k+m}$ такую, что $m \leq 0.008N$ и

$$\forall i \in [k, k+m] : \|y_{k-1} - y_i\| > \epsilon,\ \|y_i - y_{k+m+1}\| > \epsilon.$$

(Если y_{k-1} не существует тогда условие $\|y_{k-1} - y_i\| > \epsilon$ отсутствует, при не существовании y_{k+m+1} отсутствует второе условие)

Точкой разрыва в сигнале без шумов мы называем точку i такую, что $\|y_i - y_{i+1}\| > \epsilon$.

Непрерывной частью сигнала без шумов мы называем последовательность значений $\{y_i\}_{i=k}^{k+m}$, где

· 1) либо y_k первая точка, либо y_{k-1} точка разрыва,

· 2) y_{k+m} либо точка разрыва либо последняя точка,

· 3) $\forall i \in [k+m-1]:\ y_i$ - не точка разрыва.

Будем говорить, что полином pol_k апроксимирует k-ю непрерывную часть сигнала с точностью err, если

$$\sum_{i=lst(k)+1}^{lst(k+1)} (y_i - pol_k(i))^2 < err,$$

где lst - массив точек разрыва.

Исходным сигналом сигнала $\{y_i\}_{i=1}^N$ будем называть последовательность значений $\{pol_{k(i)}(i)\}_{i=1}^N$, где k выбирается в зависимости от того в какой непрерывной части находится точка i.

Формулировка задачи: на вход дается испорченый сигнал, нужно при заданых ϵ, err получить исходный сигнал, путем удаления шумов, поиска непрерывных частей и апроксимации их полиномами.

Объяснение: формализация именно такая, поскольку при достаточно малых err исходный сигнал будет похож на испорченный (условие 1), то есть сохранятся разрывы в тех точках, в которых они были и новые значения сигнала не будут сильно отличатся от прежних. Также полином является гладкой функцией и таким образом исходный сигнал будет гладким (условие 2), если соседние значения сигнала будут отличатся слишком сильно, всегда можно будет увеличить количество точек и посчитать их значения, подстановкой в полином.

2. Описание алгоритмов решения задачи

· Удаляем шумы (функция delNoize).

· Ищем разрывы (функция findGap).

· Для всех непрерывных частей k создаем функцию

$$f_k(w) = \sum_{i=lst(k)+1}^{lst(k+1)} (y_i - pol_k(i))^2,$$

где w - коэффициенты полинома. Создаем апроксимирующие полиномы для каждей части одним ис следующих способов:

I. Явное решение(функция explicitSolver):

$f_k(w)$ является сильно выпуклой и положительной, следовательно она имеет один минимум. Его можно найти напрямую, взяв градиент, приравняв его к нулю и решив получившееся уравнение .

II. Градиентный спуск(функция GCSolver):

Находим минимум функции $f_k(w)$ при помощи алгоритма градиентного спуска .

III. Увеличение степени полинома + scipy.optimize.minimize (polyGain):

Выставляем степень полинома 0.

Пока

$$\max_i \{y_i - pol_k(i)\} > err$$

увеличиваем степень полинома на 1, пересчитываем функцию $f_k(w)$ и ищем при помощи функции minimize ее минимум.

3. Роли

.

Горностаев: delNoize, findGap, polyGain

Салахутдинов: explicitSolver, advancedGenerator

Вместе написали GCSolver