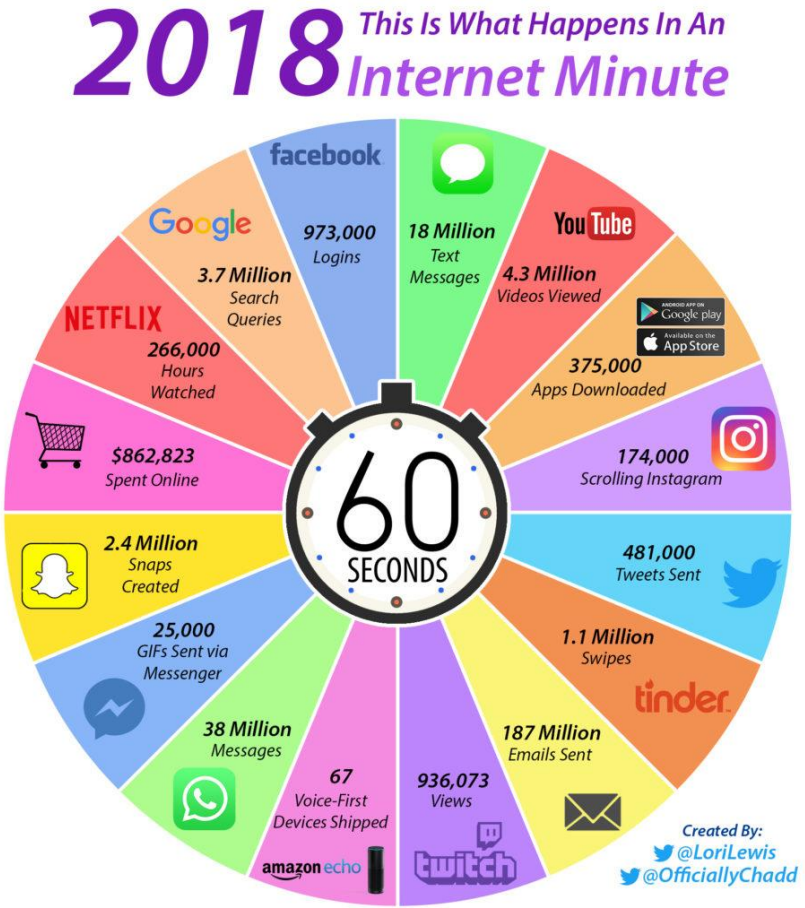
Обнаружение изменений в потоке данных с непрерывным распределением

Нагаев Артур, 444 группа
Научный руководитель:к.т.н, доц. Ю.В. Литвинов
Рецензент: Руководитель команды аналитики ООО “Интеллиджей Лабс” Поваров Н. И.

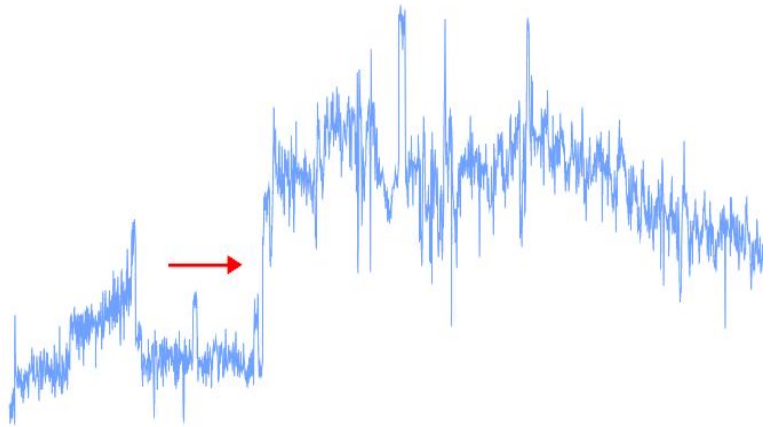
Поток данных

- Транспортные средства
- Финансовые учреждения
- Веб-сайты
- Игры
- ...

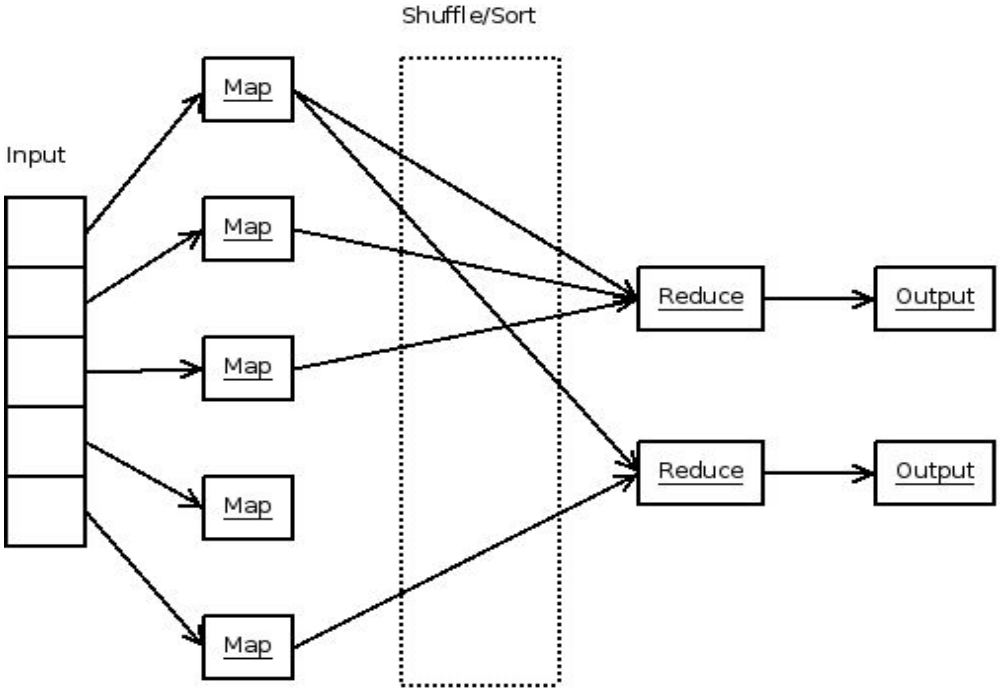


Изменение концепции

- Анализ текстов
- Автономные транспортные системы
- Мониторинг сетей
- Энергопотребление



Обнаружение изменения концепции в распределенных системах



Постановка задачи

- Изучить существующие технологии обнаружения изменений концепции в потоке данных.
- Реализовать фреймворк для обработки и обнаружения изменений в потоковых данных.
- Провести эксперименты, сравнить распределенный и нераспределенный случаи и провести анализ результатов.

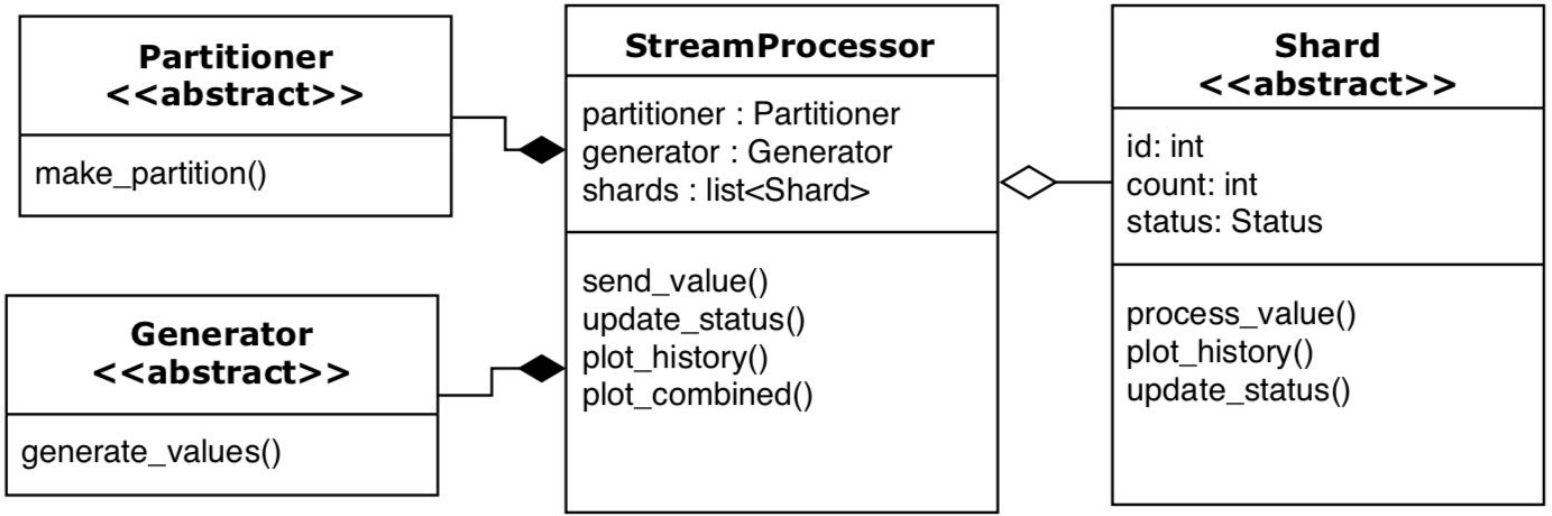
Методы обнаружения изменения потока данных

- Основанные на скользящих окнах (sliding windows)
    - Фиксированный размер окна
    - Параллельное использование нескольких окон
    - Окна с затухающими весами
- Основанные на статистическом последовательном анализе
    - Последовательный критерий отношения вероятностей (SPRT)
    - Кумулятивная сумма (CUSUM)
    - Экспоненциальные методы скользящих средних (EWMA)

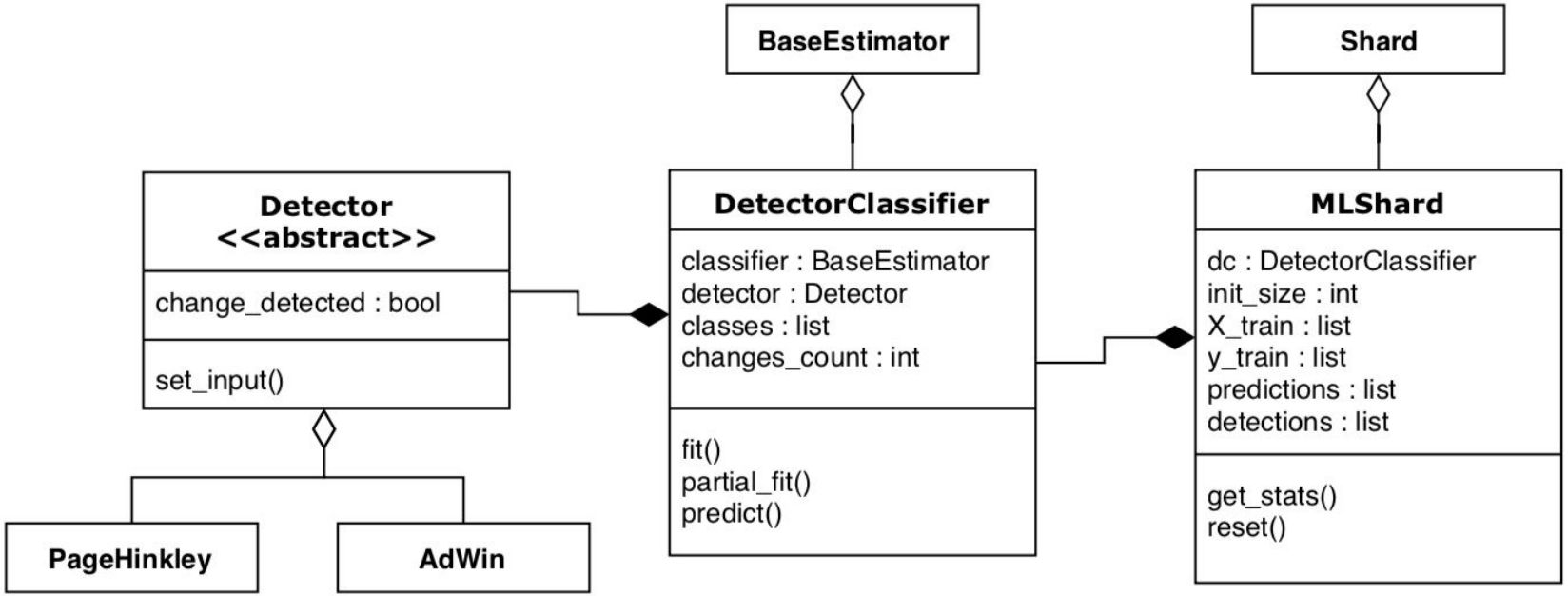
Методы распределенного обнаружения изменения потока данных

- Обнаружение изменение на одном reduce-элементе
- Micro-Cluster Nearest Neighbour (MC-NN)
- Online Map-Reduce Drift Detection Method (OMR-DDM)

Реализация


Partitioner
<<abstract>>
make_partition()
Generator
<<abstract>>
generate_values()
StreamProcessor
partitioner : Partitioner
generator : Generator
shards : list<Shard>
send_value()
update_status()
plot_history()
plot_combined()
Shard
<<abstract>>
id: int
count: int
status: Status
process_value()
plot_history()
update_status()

Реализация

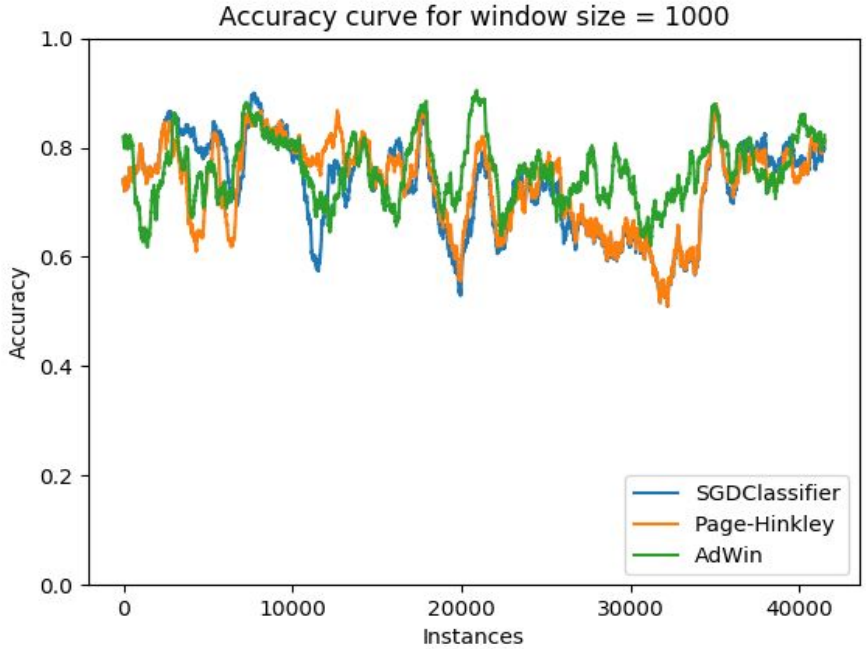

BaseEstimator
Shard
Detector
<<abstract>>
change_detected : bool
set_input()
DetectorClassifier
classifier : BaseEstimator
detector : Detector
classes : list
changes_count : int
fit()
partial_fit()
predict()
MLShard
dc : DetectorClassifier
init_size : int
X_train : list
y_train : list
predictions : list
detections : list
get_stats()
reset()
PageHinkley
AdWin

Эксперименты: данные

- Набор: elecNormNew
- 45311 экземпляров
- 2 класса (26075 ‘down’, 19236 ‘up’)
- 6 признаков

Эксперименты: модель

- Классификаторы: GaussianNB, SGDClassifier
- Детекторы: None, Page-Hinkley, Adaptive Window
- Метрика: Mean accuracy for sliding window



Эксперименты: результаты

Таблица 1: Сравнение детекторов

Модель	Детектор	Mean accuracy	Число изменений
GaussianNB	None	0.718	0
GaussianNB	PageHinkley	0.736	25
GaussianNB	ADWIN	0.75	139
SGDClassifier	None	0.84	0
SGDClassifier	PageHinkley	0.844	2
SGDClassifier	ADWIN	0.846	14

Таблица 2: Сравнение результатов для различного числа обработчиков

Число обработчиков	1	2	3	4	5	6	7	8
Mean accuracy	0.846	0.844	0.841	0.839	0.837	0.834	0.832	0.83
Время (сек)	15.97	8.34	6.61	5.23	4.29	3.87	3.66	3.43

Вывод

- Повторное обучение модели в местах изменения концепции позволяет улучшить её качество.
- Не смотря на то, что качество моделей, используемых параллельно, уменьшается с ростом числа обработчиков, можно значительно сократить время обработки благодаря естественной их масштабируемости.

Результаты

- Изучены существующие технологии обнаружения изменений концепции в потоке данных.
- Реализован фреймворк для обработки и обнаружения изменений потоковых данных.
- Проведены эксперименты, проведено сравнение распределенного и нераспределенного случаев, а также проведен анализ результатов.

Дополнительные слайды

- Page Hinkley

$$M_n = \frac{x + M_{n-1} \cdot (n-1)}{n}$$

$$S_n = S_{n-1} \cdot \alpha + (x - M_n - \delta)$$

Дополнительные слайды

- Adaptive window

Algorithm 1 ADWIN

1: Инициализировать окно W
2: **for each** $t > 0$ **do**
3: $W \leftarrow W \cup x_t$
4: **repeat**
5: Исключить элемент из W
6: **until** $|\hat{\mu}W_0 - \hat{\mu}W_1| < e$ для каждого разделения W на $W_0 \cup W_1$
7: **end for**
return $\hat{\mu}W$