МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ярославский государственный университет им. П.Г.Демидова» (ЯрГУ)

Кафедра компьютерной безопасности и

математических методов обработки информации

Курсовая работа

Алгоритм динамической кластеризации многомерных данных методом *k*-средних

(Специальность 10.05.01 Компьютерная безопасность)

Научный руководитель

\_\_\_\_\_\_\_\_\_\_\_ Н. П. Федотова

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Студент группы КБ-51

\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Комаров

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Ярославль 2019 г.

**1 Введение**

**2 Основные понятия кластеризации и метода *k*-средних**

**2.1 Кластеризация**

**2.2 Метод *k*-средних**

**3 Динамическая кластеризация**

**3.1 Построение *e*-coreset для k-средних**

**3.2 Применение sensitivity sampling (метода выборки чувствительности)**

**3.3 Метод случайного сдвига сеток**

**3.4 Порядок работы алгоритма**

**4 Результат работы алгоритма**

**5 Заключение**

**6 Список литературы**

**Приложения**

**Введение**

В своей работе я рассматриваю проблему кластеризации *k*-средних в настройке динамической потоковой передачи, где точки из дискретного евклидова пространства {1, 2, … , ∆}*d* могут быть динамически вставлены или удалены из набора данных. Для этой задачи предоставляется однопроходный алгоритм c пространственной сложностью Õ(*k* · poly(*d*, log ∆)), где *k* - целевое число центров, Õ(*f*) это O(*f \** logO(1)(*f*)). Проблема описана в статье исследователей Wei Hu, Zhao Song, Lin F. Yang, Peilin Zhong. Это первый из известных алгоритмов динамического потока данных для *k*-средних с полиномиальной сложностью и почти оптимальный по *k*.

**Понятие кластеризации и метода *k-*средних**

**Кластеризация**

Кластеризация — задача группировки множества объектов на подмножества (кластеры) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию.

Задача кластеризации относится к классу задач обучения без учителя.

Пусть *X* — множество объектов, *Y* — множество идентификаторов (меток) кластеров. На множестве *X* задана функция расстояния между объектами *p*(*x*,*x*′). Дана конечная обучающая выборка объектов *Xm* = {*x1*,…,*xm*} ⊂ *X*. Необходимо разбить выборку на подмножества (кластеры), то есть каждому объекту *xi* ∈ *Xm* сопоставить метку *yi*∈*Y*, таким образом чтобы объекты внутри каждого кластера были близки относительно метрики *p*, а объекты из разных кластеров значительно различались.

Алгоритм кластеризации — функция *a:X → Y*, которая любому объекту *x ∈ X* ставит в соответствие идентификатор кластера *y ∈ Y*

Множество *Y* в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки объектов из обучающей выборки *yi* изначально не заданы, и даже может быть неизвестно само множество *Y*.

Решение задачи кластеризации объективно неоднозначно по ряду причин:

Не существует однозначного критерия качества кластеризации. Известен ряд алгоритмов, осуществляющих разумную кластеризацию "по построению", однако все они могут давать разные результаты. Следовательно, для определения качества кластеризации и оценки выделенных кластеров необходим эксперт предметной области;

Число кластеров, как правило, заранее не известно и выбирается по субъективным критериям. Даже если алгоритм не требует изначального знания о числе классов, конкретные реализации зачастую требуют указать этот параметр

Результат кластеризации существенно зависит от метрики. Однако существует ряд рекомендаций по выбору метрик для определенных классов задач.

**Метод *k-*средних**

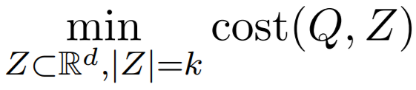
Метод k-средних является одним из самых важных подходов к кластеризации. Этот метод был описан в 1957 Стюартом Ллойдом (S. Lloyd), и широко изучается уже более 60 лет.

Основная идея алгоритма *k*-средних заключается в том, что данные произвольно разбиваются на кластеры, после чего итеративно перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

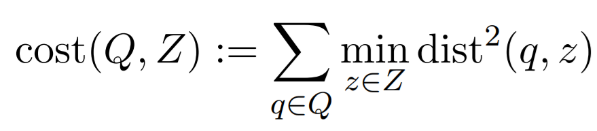
Цель алгоритма заключается в разделении *n* наблюдений на *k* кластеров таким образом, чтобы каждое наблюдение принадлежало ровно одному кластеру, расположенному на наименьшем расстоянии от наблюдения.

Для набора точек Q ⊂ ℝ*d* метод k-средних требует набор из центров Z ⊂ ℝ*d* так, что сумма квадратов расстояний между точками и их ближайшими центрами минимизируется.

Иными словами, алгоритм пытается решить

 ,

где cost(*Q, Z*) – оценочная функция:



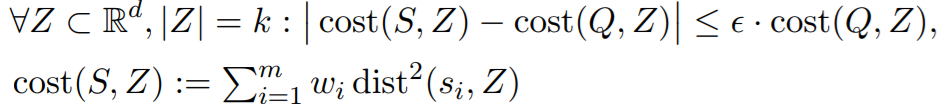
**Динамическая кластеризация**

При работе с большими наборами данных возникает проблема, что объем входных данных слишком велик, чтобы хранить его в памяти. Стандартной моделью в таком случае является потоковая модель, где точки добавляются и обрабатываются по порядку, а одновременно обрабатывается только полезная малая часть от всей информации. Такая модель была описана С. Мутукришаном (S. Muthukrishnan) в 2005 году. В динамических потоках данных точки из дискретного пространства {1, 2, … , ∆}*d* могут быть добавлены или удалены из набора данных.

Стандартный подход к решению кластеризации k-методами является поддержка небольшого набора точек (coreset), который относительно k-центра аппроксимирует форму целого набора для того же k-центра. В результате такого подхода, в конце потока требуется найти решение на приближенном малом наборе данных, который в свою очередь подходит и для всего набора.

**Coreset для *k*-средних.**

Даны *Q* ⊆ [∆]*d* – набор точек, *k* ∈ ℕ+ , *e* ∈ (0, 0.5) – параметр ошибки, набор взвешенных пар точек *S=*{(*s1, w1*),(*s2, w2*),. . . ,(*sm*,*wm*)}⊂ [∆]*d* × ℝ>0 является *e-*coreset для *Q*, где *wi* - вес s*i*, если S удовлетворяет требованиям

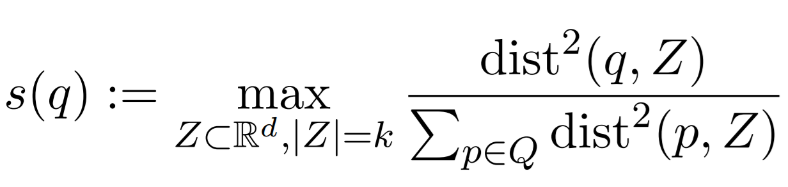


Следовательно, целью является нахождение эффективного метода поддержки coreset над потоком данных c наименьшей пространственной сложностью.

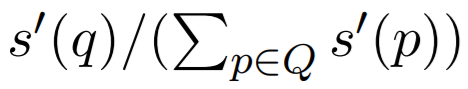
Предположим, что существует набор точек *Q* ⊆ [∆]*d*, полученный из потока операций вставки/удаления в модели динамического потока. Пусть *L =* log∆. Для данных *e* ∈ (0, 0.5) существует алгоритм, который в один проход по потоку данных возвращает k-средний *e*‑coreset *S* для *Q*. Более того, алгоритм использует Õ(*k* · poly(*d*, *L*, *e*-1)) бит.

**Sensitivity sampling**

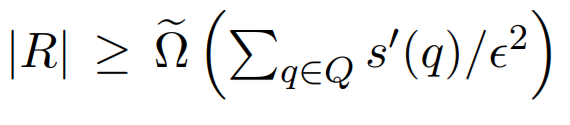
Алгоритм работает с применением технологии выборки чувствительности (sensitivity sampling). Для набора *Q* ⊆ [∆]*d*, чувствительность каждой точки *q* ∈ *Q* определена как:



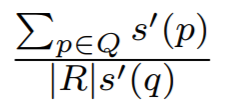
Где *s*(*q*) показывает, насколько удаление точки *q* скажется на оценочной функции. Ключевой результат был показан в работе Фельдмана и Лангберга (D. Feldman, M. Langberg). Утверждается, что, имея хорошую верхнюю границу чувствительности для каждой точки, существует метод выборки для построения *e*-coreset. В особенности, когда известно, что верхняя граница *s'*(*q*) *≥ s*(*q*) для каждой *q* ∈ *Q*, томожно выбрать *q* с вероятностью



Пусть R – набор независимых одинаково распределенных случайных величин, полученный из *s'*(*q*).



и каждому выбору *q* присвоен вес



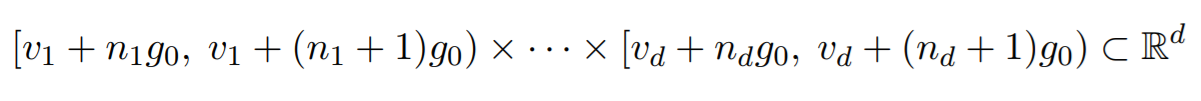
В таком случае с высокой вероятностью можно сказать, что *R* – *e*-coreset для *Q.*

Ключевой особенностью в оценке чувствительности является следующее. Возможно существование области с очень плотным расположением точек. Оценка чувствительности для каждой точки в этой области будет низкой, в силу того, что точку можно заменить на любую другую из этой области. Следовательно, проблема нахождения верхней границы для точки сводится к тому, чтобы определить к какой из областей будет относится эта точка, и можно ли считать эту область плотной. Интуитивно полагается, что оценка чувствительности для точки сводится к размеру плотной области, к которой эта точка принадлежит.

**Метод случайного сдвига сеток**

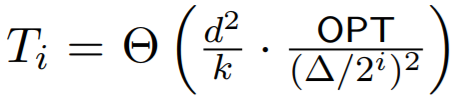
Можно сделать это предположение более формальным используя структуру иерархической сетки (Рис. 1).

Пусть *g*0 = ∆. Выберем вектор *v* случайно распределенный на [0, ∆]*d*. Разбиение пространства ℝ*d* в прямоугольную систему координат G0 с длиной стороны g0, такое что вершина этой системы падает на *v*. Сетка G0 может быть рассмотрена как бесконечный набор раздельных ячеек, где каждая ячейка C ∈ G0 может быть выражена как

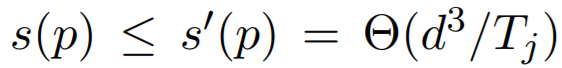


Для некоторых (*n1, n2,* …, *nd*) ∈ ℤ*d*

Верхний уровень сетки (0-вой) состоит из ячеек, которые представляют собой *d-*пространственные кубы с длинной стороны ∆. Каждая ячейка в уровне *i*-1 делится на 2*d* ячеек на уровне *i*, длина ячейки на уровне *i* равна ∆/2*i*. Будем считать каждую ячейку на уровне *i* тяжелой, если она содержит



точек в *Q*, где OPT – оптимальная оценка для *k*-средних. Так как *Ti > Ti-1*, если ячейка на уровне *i* является тяжелой, ее прародитель на уровне *i*-1 тоже тяжелая ячейка. Таким образом набор всех тяжелых ячеек на уровнях формирует дерево. Теперь для точки *p* ∈ *Q* обозначим ячейку *ci*(*p*), которая содержит эту точку. Обозначим *j* как индекс наименьшего уровня, где *cj*(*p*) уже не является тяжелой ячейкой; затем покажем, что верхняя граница для *s*(*p*) определяется исключительно на основе индекса *j*:



Чтобы обеспечить малое количество тяжелых ячеек, в начале используется случайный сдвиг (Рис. 2). Полученная сумма верхних границ будет достаточно мала, чтобы удовлетворять требованиям.

Трудностью в применении метода выборки чувствительности в динамической настройке является:

1. Изначально неизвестное значение OPT, которое изменяется в процессе обновления данных
2. Одновременное вычисление верхних границ и выборка точек в ограниченной памяти.

**Порядок работы алгоритма**

Сперва предположим, что в начале алгоритма известно значение OPT. Алгоритм выделяет ключевую роль использованию структуры *k*-набора данных для вычисления уникальных элементов в динамическом потоке. Такой набор данных гарантирует, что количество уникальных элементов в некоторой степени предопределено; алгоритм возвращает все уникальные элементы и их частоту или сообщение о неудачном исполнении. Для выполнения выборки чувствительности необходимо динамически отслеживать какие из ячеек являются тяжелыми.

Выборка состоит из двух этапов:

1. Первый образец строится для уровня *i* (с соответствующей вероятностью для каждого уровня)
2. Равномерно выбирается точка из всех связанных с *i* уровнем (все точки *p*, где *ci*(*p*) не является тяжелой ячейкой, а *ci-1*(*p*) является тяжелой)

Для того, чтобы сделать равномерную выборку, мы также поддерживаем для каждого уровня случайное подмножество точек.

Из-за того, что OPT неизвестно, для алгоритма предлагается несколько решений. Одним из таких решений является параллельный запуск нескольких копий процесса выборки для разных предположенных OPT: 1, 2, 4, …, ∆*d* \**d*∆*.* В случае, когда память закончилась, алгоритм выборки возвращает сообщение о неудачной попытке. Следуя из того, что хотя бы одна копия алгоритма будет успешно выполнена, на выходе получается нужный *e*‑coreset.

Решение *k*-средних является NP-сложной задачей даже в случаях, когда *k*=2 или *d*=2. Наиболее успешным на практике алгоритмом является изначальное решение, описанное Ллойдом. Из-за NP-сложности, было предпринято множество попыток оптимизировать алгоритмы аппроксимации. В решениях, при котором *k* или *d* константы, аппроксимация может быть достигнута за полиномиальное время. Также существует эффективное решение для потоков, которые могут принимать данные, но не удалять. Наибольший интерес вызывают именно динамические модели, с доступными вставкой и удалением данных. В работе Бравермана (Braverman) удалось построить coreset размера *O*(*k\**poly(*d,*log∆)) в динамическом потоке, но используемый метод опирается на использование *k-*медианы и не может быть применим к *k-*средним.

**Результат работы алгоритма**

В работе использовались некоторые из описанных методов для оптимизации работы алгоритма. Алгоритм реализован на языке Python. Для визуализации данных были использованы инструменты Jupyter Notebook, matplotlib, scikitlearn. В приложениях к работе выведены метрики алгоритма после его практического использования на данных.

**Заключение**

В данной работе представлен один из способов построения динамической модели на основе метода *k*-средних, использующий метод выборки чувствительности. Совокупность описанных методов сводит сложность алгоритма по памяти к полиномиальной. Пониженные требования алгоритма к памяти позволяют одновременно охватывать больший объем данных, что положительно сказывается на эффективности модели.

**Список литературы**

Machine Learning A Probabilistic Perspective Kevin P. Murphy

Pattern Recognition And Machine Learning Christopher M. Bishop   
  
Solving k-means on High-dimensional Big Data   
[https://arxiv.org/pdf/1502.04265.pdf](https://vk.com/away.php?to=https%3A%2F%2Farxiv.org%2Fpdf%2F1502.04265.pdf&cc_key=)   
  
Nearly Optimal Dynamic k-Means Clustering for High-Dimensional Data Wei Hu, Zhao Song, Lin F. Yang, Peilin Zhong  
[https://arxiv.org/pdf/1802.00459v2.pdf](https://vk.com/away.php?to=https%3A%2F%2Farxiv.org%2Fpdf%2F1802.00459v2.pdf&cc_key=)   
  
Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. [https://arxiv.org/pdf/1410.6801.pdf](https://vk.com/away.php?to=https%3A%2F%2Farxiv.org%2Fpdf%2F1410.6801.pdf&cc_key=)   
  
Shanmugavelayutham Muthukrishnan. Data streams: Algorithms and applications.   
  
Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In Proceedings of the   
thirty-sixth annual ACM symposium on Theory of computing, стр 373–380.    
  
Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. 569-578   
  
Stuart Lloyd. Least squares quantization in pcm. 129-137,   
  
Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sumof-squares clustering. Machine learning, 75(2):245–248

Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. arXiv preprint arXiv:1612.00889, 2016.

Приложение А  
Рисунок 1

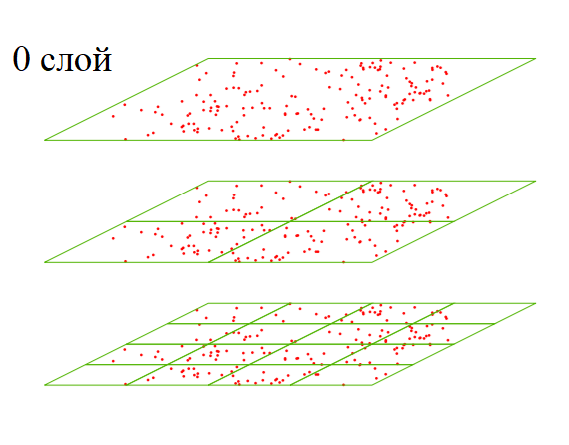
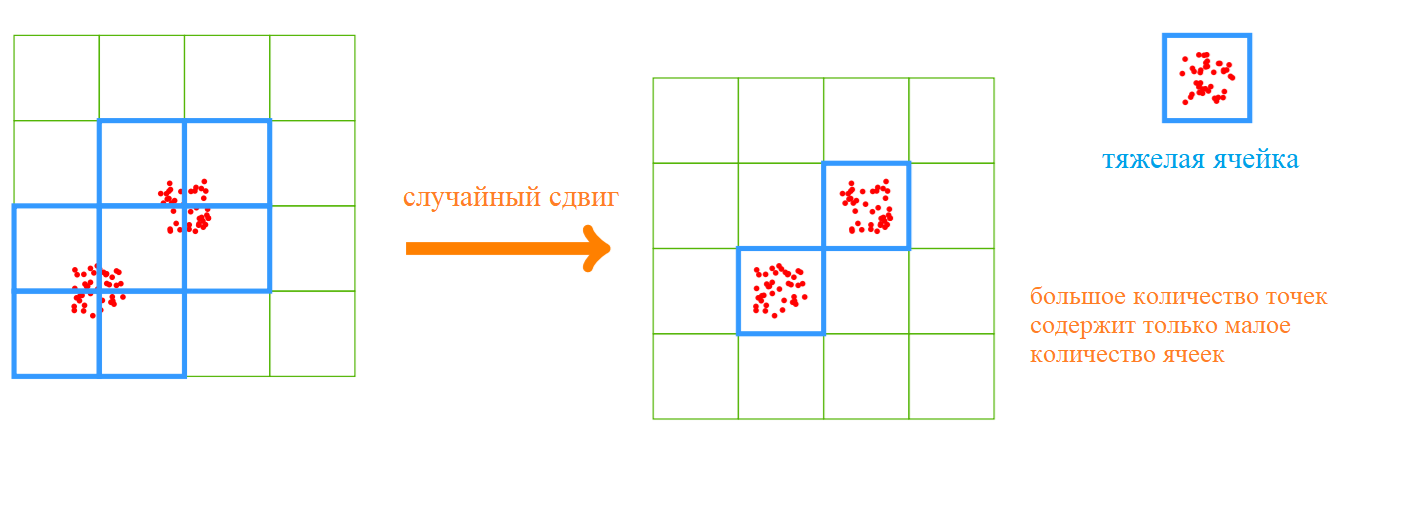
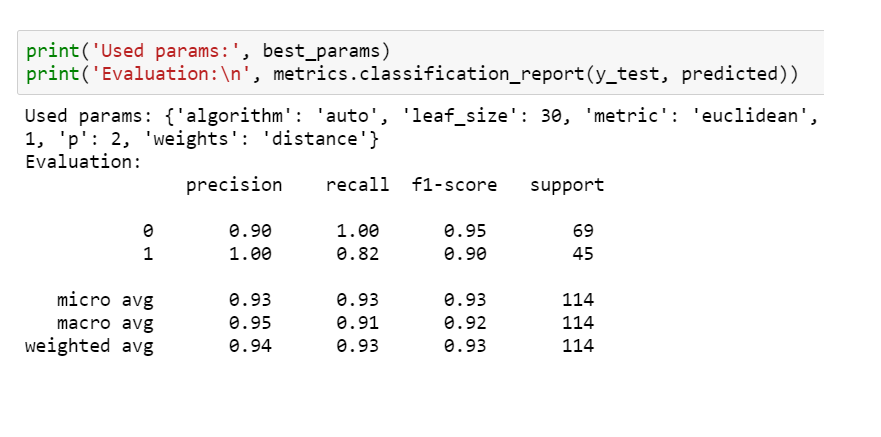


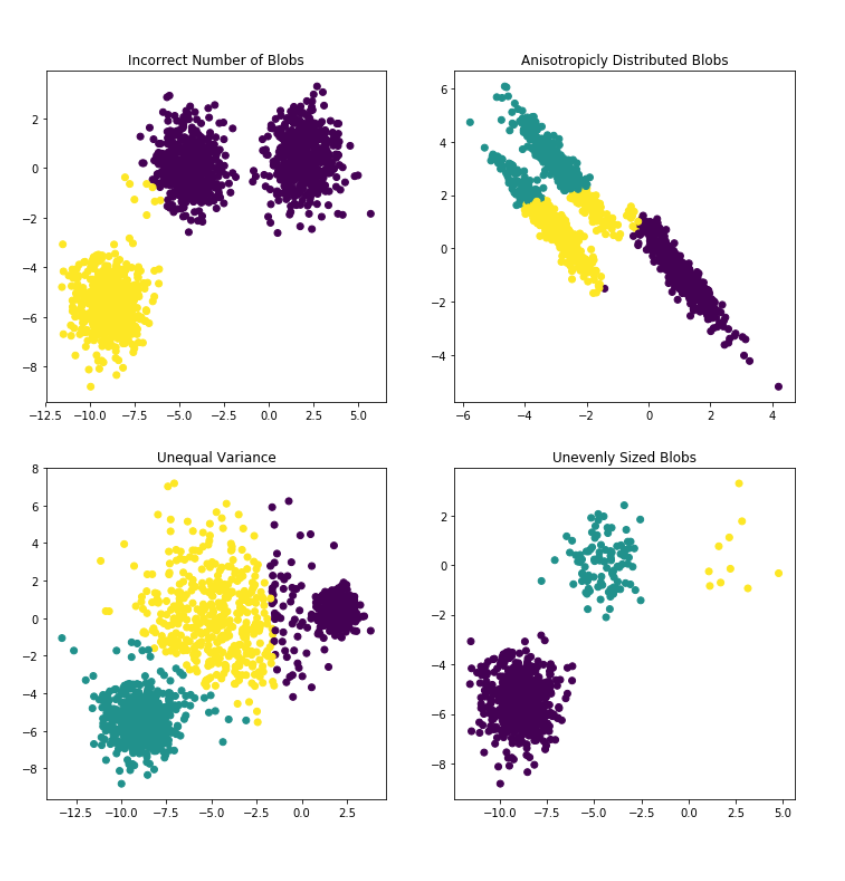
Рисунок 2



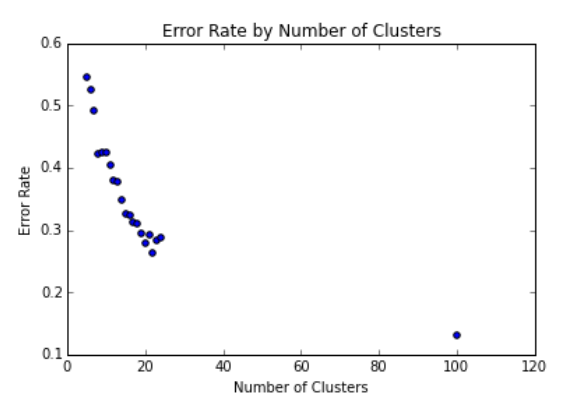
Приложение Б



Приложение В



Приложение Г



Приложение Д

