Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

STUDI KINERJA ALGORITMA OPTIMASI PADA METODE *QUANTUM CLUSTERING* DENGAN *KERNEL ENTROPY COMPONENT ANALYSIS* UNTUK REDUKSI DIMENSI

RIANSYA PAMUSTI
NRP 05111440000175

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

Dosen Pembimbing II
Arya Yudhi Wijaya, S.Kom, M.Kom

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - KI141502

STUDI KINERJA ALGORITMA OPTIMASI PADA METODE *QUANTUM CLUSTERING* DENGAN *KERNEL ENTROPY COMPONENT ANALYSIS* UNTUK REDUKSI DIMENSI

RIANSYA PAMUSTI
NRP 05111440000175

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

Dosen Pembimbing II
Arya Yudhi Wijaya, S.Kom, M.Kom

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

PERFORMANCE STUDY OF OPTIMIZATION ALGORITHM ON QUANTUM CLUSTERING WITH KERNEL ENTROPY COMPONENT ANALYSIS FOR DIMENSIONAL REDUCTION

RIANSYA PAMUSTI
NRP 05111440000175

Supervisor I
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.

Supervisor II
Arya Yudhi Wijaya, S.Kom, M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

STUDI KINERJA ALGORITMA OPTIMASI PADA METODE *QUANTUM CLUSTERING* DENGAN *KERNEL ENTROPY COMPONENT ANALYSIS* UNTUK REDUKSI DIMENSI

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
RIANSYA PAMUSTI
NRP : 05111440000175

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr. Chastine Fatichah, S.Kom, M.Kom. .....
NIP: 19751220 200112 2 002 (pembimbing 1)

Arya Yudhi Wijaya, S.Kom, M.Kom .....
NIP: 19710428 199412 2 001 (pembimbing 2)

SURABAYA
JULI 2018

[Halaman ini sengaja dikosongkan]

STUDI KINERJA ALGORITMA OPTIMASI PADA METODE *QUANTUM CLUSTERING* DENGAN *KERNEL ENTROPY COMPONENT ANALYSIS* UNTUK REDUKSI DIMENSI

Nama Mahasiswa	**: RIANSYA PAMUSTI**
NRP	**: 05111440000175**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Dr. Chastine Fatichah, S.Kom, M.Kom.**
Dosen Pembimbing 2	**: Arya Yudhi Wijaya, S.Kom, M.Kom.**

ABSTRAK

Clustering merupakan proses partisi dataset menjadi beberapa bagian yang disebut dengan cluster. Data yang terdapat dalam satu cluster memiliki kemiripan karakteristik antar satu sama lainnya dan berbeda dengan cluster yang lain. Proses partisi tidak dilakukan secara manual melainkan dengan suatu algoritma clustering. Oleh karena itu, clustering sangat berguna untuk menemukan kelompok yang terdapat dalam dataset.

Penentuan jumlah cluster pada algoritma clustering merupakan tantangan tersendiri yang umumnya dilakukan degan uji coba secara empiris. Tugas Akhir ini mengimplementasikan algoritma clustering yang merupakan kombinasi dari algoritma kernel entropy component analysis (KECA) sebagai pre-processing dengan quantum clustering (QC) sebagai clustering. Dengan algoritma KECA, bisa diperoleh dimensi yang jauh lebih kecil dari data yang sesungguhnya. Setelah itu, QC digunakan untuk mengelompokkan data yang telah diproses oleh algoritma KECA. Algoritma QC bisa mendapat jumlah cluster secara otomatis tanpa mengetahui jumlah cluster yang sesungguhnya. Tugas Akhir ini diimplementasikan QC dalam lima macam algoritma optimasi untuk menemukan quantum potential minima.

Dataset yang digunakan dalam proses uji coba terdiri dari lima buah dataset buatan dan tiga buah dataset dari UCI. Hasil akhir yang didapat menunjukkan bahwa setiap algoritma optimasi menunjukkan hasil yang tidak jauh berbeda. Namun, hasil yang buruk selalu didapat saat menggunakan algoritma optimasi AdaGrad. Hasil penggunaan KECA-QC terbaik terdapat pada dataset Cluster in cluster dengan akurasi sempurna 100% dan hasil terburuk terdapat pada dataset Corners dengan akurassi 33.58%.

Kata kunci: Data mining, Clustering, Quantum Clustering, Kernel Entropy Component Analysis.

PERFORMANCE STUDY OF OPTIMIZATION ALGORITHM ON QUANTUM CLUSTERING WITH KERNEL ENTROPY COMPONENT ANALYSIS FOR DIMENSIONAL REDUCTION

Student's Name : **RIANSYA PAMUSTI**
Student's IDE : **05111440000175**
Department : **Informatika FTIK-ITS**
First Advisor : **Dr. Chastine Fatichah, S.Kom, M.Kom.**
Second Advisor : **Arya Yudhi Wijaya, S.Kom, M.Kom.**

ABSTRACT

Clustering is the process of partitioning dataset into subsets called clusters. Data within a cluster have similar characteristics between each other and are different from other clusters. Partitions are not done manually but with a clustering algorithm. Therefore, clustering is very useful to find unknown groups in the dataset.

Determining the number of clusters in the clustering algorithm is a challenge that is generally done by empirical testing. In this undergraduate theses, We just implement combination of quantum clustering (QC) algorithm with kernel entropy component analysis (KECA) for clustering dataset. With KECA algorithm we can get far smaller size of data, so it significantly reduce the computation for the clustering. After that, use the QC algorithm to cluster the data processed by KECA algorithm. QC algorithm can obtain number of cluster without knowing the real number of cluster. In this undergraduated theses QC was implemented using five optimization algorithm to find quantum potential minima.

The dataset used in this experiment consists of five artificial datasets and three from UCI datasets. The results show that every optimization algorithm obtain nearly similar results. However,

when the AdaGrad optimization algorithm used, poor results are always obtained. The best results of KECA-QC usage are found on the Cluster in cluster dataset with 100% perfect accuracy and the worst result is in the Corners dataset with 33.58% accuracy.

Kata kunci: Data mining, Clustering, Quantum Clustering, Kernel Entropy Component Analysis.

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيم

Alhamdulillahirabbil'alamin, segala puji penulis panjatkan kepada Allah SWT, yang selalu memberikan kemudahan dibalik kesulitan pada setiap hambanya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

STUDI KINERJA ALGORITMA OPTIMASI PADA METODE QUANTUM CLUSTERING DENGAN KERNEL ENTROPY COMPONENT ANALYSIS UNTUK REDUKSI DIMENSI

Pengerjaan Tugas Akhir in telah menjadi salah satu pengalaman yang paling berharga sepanjang hayat penulis. Pengerjaan Tugas Akhir ini mengimplementasikan berbagai macam ilmu yang telah didapatkan penulis selama menempuh perkuliahan di Informatika ITS.

Tugas Akhir ini dapat selesai karena bantuan dan dukungan dari berbagai pihak. Penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya dan kemudahan yang diberikannya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Syarif Pamungkas dan Reni Siti Sundari dan kedua saudara Resya Pamukti serta Risya Anisya Pamurni terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Informatika ITS hingga akhirnya menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua departemen Informatika ITS
4. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Informatika ITS.

5. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom selaku pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
6. Bapak Arya Yudhi Wijaya, S.Kom, M.Kom.Kom selaku pembimbing II Tugas Akhir yang telah memberikan bimbingan selama penulis menyelesaikan Tugas Akhir.
7. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan banyak sekali ilmu dan pengalaman selama penulis kuliah di Teknik Informatika
8. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
9. Rekan-rekan admin laboratorium Komputasi Cerdas dan Visi yang telah membantu pengerjaan Tugas Akhir ini dengan adanya peminjaman PC dan juga tempat yang nyaman.
10. Rekan-rekan TC14 terutama Ghazian, Galang, Luqman, Anwar, Dave yang banyak sekali saya tanya untuk penyelesaian Tugas Akhir ini.
11. Rekan-rekan soritia Rizal, Romi, Evan, Danis, Adiwinoto, Dwiyandono, Anggit, Ucup, Agun, Andre, Fito, Angga, Rochman, OjanGG, Bagas, Iqbal, Digoz yang sangat unik telah menemani penulis baik susah maupun senang selama perkuliahan di Informatika ITS dan terus memberikan semangat pada penulis untuk menyelesaikan Tugas Akhir ini.
12. Rekan-rekan kucing69 Fariz, Al, Askar, Rully, Oleo, Adis, Nody yang telah memberikan dukungan dan kebersamaan yang luar biasa selama penulis melakukan studi di ITS.

Penulis memohon maaf apabila terdapat kekurangan dan kesalahan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan kedepannya. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, April 2018

Riansya Pamusti

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran tugas akhir secara umum dapat dipahami.

1.1. Latar Belakang

Seiring berjalannya waktu, data kian menjadi hal yang sering kali kita temui dalam kehidupan sehari-hari karena kebutuhan terhadap data digital terus menerus berkembang menjadi skala yang semakin besar setiap harinya. Hal tersebut menjadi masalah baru dimana banyaknya data yang ada tidak diimbangi dengan kemampuan untuk mengolahnya. Oleh karena itu, dibutuhkan suatu pengelompokkan terhadap data yang semakin banyak tersebut agar dapat diolah sesuai dengan kebutuhan. Era perkembangan teknologi seperti sekarang ini membuat pengolahan tersebut perlu dieksekusi dengan komputer menggunakan suatu algoritma agar proses yang dilakukan lebih cepat. Algoritma yang dimaksud adalah algoritma clustering. *Clustering* merupakan proses pengelompokkan data menjadi beberapa kelas berdasarkan kemiripan data, data dengan kemiripan yang tinggi akan dikelompokkan pada kelompok yang sama[1].

Berawal dari permasalahan diatas, penulis mencoba menerapkan suatu metode yang diklaim oleh pencetusnya mampu melakukan *clustering* tanpa terlebih dahulu menentukan jumlah *cluster* yang ada pada suatu data dan dalam bentuk apapun menyesuaikan dengan data yang diuji. Metode tersebut bernama *quantum clustering* dengan *kernel entropy component analysis* (KECA-QC) yang terbagi menjadi dua tahap, tahap *preprocessing* dan tahap *clustering*. Tahap *preprocessing* menggunakan KECA dengan ide utama memetakan data asli menjadi data dengan fitur berdimensi besar yang kemudian dipilih beberapa komponen yang memiliki *renyi entropy* terbesar. Setelah proses tersebut data yang terkelompok pada

cluster yang berbeda kurang lebih akan berada di posisi yang berbeda pula, selain itu data asli pun tereduksi menjadi data dengan dimensi yang lebih kecil terutama jika yang digunakan adalah dataset yang berdimensi besar. Tahap *clustering* menggunakan algoritma *quantum clustering* (QC) yang mampu menemukan *cluster* dengan berbagai bentuk tanpa mengetahui jumlah *cluster* yang sebenarnya. Untuk mempercepat algoritma *quantum clustering* tradisional, dilakukan modifikasi saat perhitungan *wave function* dengan tidak dilakukan perhitungan pada seluruh dataset, namun hanya dengan statistik dari distribusi k data terdekat. Metode KECA-QC memiliki suatu proses untuk menemukan nilai minimum dari *quantum potential function* dengan menggunakan algoritma optimasi *gradient descent*. Disini penulis bermaksud untuk melakukan studi kinerja pada proses tersebut dengan menerapkan algoritma optimasi lain pada proses tersebut diantaranya, *gradient descent* dengan momentum dan *conjugate gradient*.

Hasil Tugas Akhir ini diharapkan dapat membuktikan bahwa algoritma yang dipakai dapat mengelompokkan data dengan berbagai bentuk tanpa mengetahui terlebih dahulu jumlah *cluster*, selain itu algoritma ini dapat memproses data berdimensi besar yang tentunya dengan tingkat akurasi yang tinggi dan diharapkan hasil tugas akhir ini dapat berguna bagi perkembangan teknologi informasi.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana menerapkan metode KECA-QC ?
2. Bagaimana menerapkan metode KECA-QC dengan menggunakan berbagai algoritma optimasi ?
3. Bagaimana akurasi dari algoritma KECA-QC pada berbagai algoritma optimasi dan berbagai macam dataset ?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki satu Batasan masalah, yaitu mplementasi algoritma menggunakan bahasa pemrograman Matlab R2017b.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Untuk menerapkan metode KECA-QC dengan berbagai algoritma optimasi.
2. Untuk dapat mengevaluasi akurasi dari algoritma KECA-QC pada berbagai algoritma optimasi dan berbagai macam dataset.

1.5. Manfaat

Pengerjaan tugas akhir ini diharapkan dapat menjadi alternatif untuk dapat melakukan *clustering* terhadap berbagai jenis dan ukuran dataset yang memiliki akurasi yang tinggi.

1.6. Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Langkah awal dalam mengerjakan tugas akhir adalah dengan menyusun proposal tugas akhir. Proposal ini berisi studi kinerja dari algoritma optimasi pada suatu metode yang bisa melakukan *clustering* pada data dengan bentuk apapun tanpa mengetahui terlebih dahulu jumlah *cluster* dan pada data dengan dimensi besar bernama KECA-QC.

2. Studi literatur

   Tahap ini akan membahas peninjauan terhadap studi literatur yang membahas KECA-QC dan berbagai ilmu yang terkait di dalamnya, terutama yang masih penulis kurang pahami seperti, perhitungan *eigenvalues* dan *eigenvectosr* dan penerapan algoritma optimas seperti, *gradient descent, momentum,* dan *conjugate gradient*. Studi literatur didapatkan dari buku, *internet*,

dan materi-materi kuliah yang berhubungan dengan metode yang akan digunakan.

3. Perancangan perangkat lunak

   Tahap ini akan berisi analisis terhadap apa yang sebenarnya dilakukan pada setiap tahap metode KECA-QC dan apa pengaruh diterapkannya berbagai algoritma optimasi pada KECA-QC.

4. Implementasi perangkat lunak

   Implementasi dilakukan dengan menggunakan bahasa pemrograman Matlab R2017b dengan beberapa bantuan *library* yang sudah ada pada Matlab R2017b.

5. Pengujian dan evaluasi

   Tahap pengujian dan evaluasi akan berisi pengujian hasil *clustering* terhadap metode KECA-QC dengan berbagai algoritma optimasi baik dari segi akurasi maupun dari segi kecepatan jalannya program.

6. Penyusunan buku Tugas Akhir.

   Tahap ini berisi penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. Bab I. Pendahuluan

   Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.

2. Bab II Tinjauan Pustaka

Bab ini berisi penjelasan tentang penelitian sebelumnya dan berbagai dasar teori yang berhubungan dengan KECA-QC.

3. Bab III Perancangan Perangkat Lunak

   Bab ini berisi penjelasan mengenai desain, perancangan, bahan, dan pemodelan proses yang digunakan dalam Tugas Akhir ini yang direpresentasikan dengan *pseudocode*.

4. Bab IV. Implementasi

   Bab ini merupakan pembangunan aplikasi dengan MATLAB sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I.

5. Bab V. Hasil Uji Coba dan Evaluasi

   Bab ini berisi penjelasan mengenai data hasil percobaan, pengukuran, visualisasi dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. Bab VI. Kesimpulan dan Saran

   Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan karya rancangan. Selain itu, pada bab ini diberikan saran-saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pembuatan aplikasi *clustering* dengan menggunakan metode KECA-QC. Penjelasan ini bertujuan untuk memberikan dasar teori yang mendasari pengembangan perangkat lunak.

2.1. *Clustering*

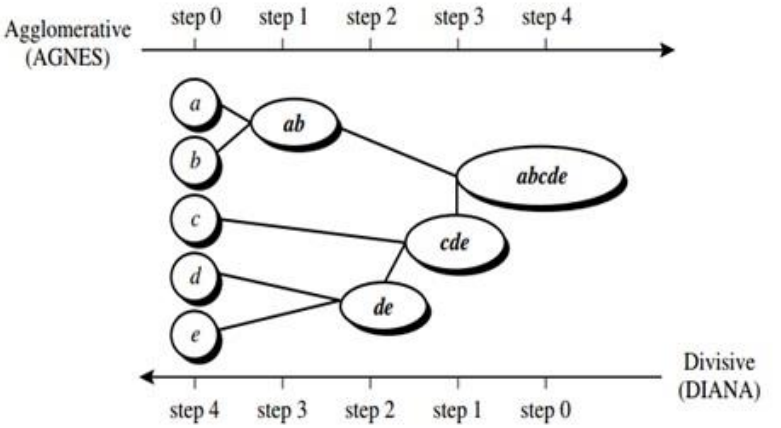
Clustering atau klasterisasi adalah metode pengelompokan data. C*lustering* adalah sebuah proses untuk mengelompokan data ke dalam beberapa *cluster* atau kelompok sehingga data dalam satu *cluster* memiliki tingkat kemiripan yang maksimum dan data antar cluster memiliki kemiripan yang minimum [1].

Clustering merupakan proses partisi satu set objek data ke dalam himpunan bagian yang disebut dengan *cluster*. Objek yang di dalam cluster memiliki kemiripan karakteristik antar satu sama lainnya dan berbeda dengan *cluster* yang lain. Partisi tidak dilakukan secara manual melainkan dengan suatu algoritma *clustering*. Oleh karena itu, *clustering* sangat berguna dan bisa menemukan group atau kelompokyang tidak dikenal dalam data. *Clustering* banyak digunakan dalam berbagai aplikasi seperti misalnya pada *business inteligence*, pengenalan pola citra, *web search*, bidang ilmu biologi, dan untuk keamanan (*security*). Di dalam *business inteligence*, *clustering* bisa mengatur banyak *customer* ke dalam banyaknya kelompok. Contohnya mengelompokan *customer* ke dalam beberapa *cluster* dengan kesamaan karakteristik yang kuat. *Clustering* juga dikenal sebagai data segmentasi karena *clustering* mempartisi banyak data set ke dalam banyak *group* berdasarkan kesamaannya. Selain itu *clustering* juga bisa sebagai *outlier detection*.

Metode *Clustering* secara umum dapat dibagi menjadi dua yaitu *hierarchical clustering* dan *partitional clustering* [1] Sebagai tambahan, terdapat pula metode *Density-Based* dan *Grid–Based* yang juga sering diterapkan dalam implementasi *clustering*.

2.1.1. *Hierarchical clustering*

Hierarchical clustering pengelompokkan data melalui suatu bagan yang berupa hirarki, dimana terdapat penggabungan dua grup yang terdekat disetiap iterasinya ataupun pembagian dari seluruh set data kedalam *cluster*. *Hierarchical clustering* dapat dipahami dengan melihat pada Gambar 2.1



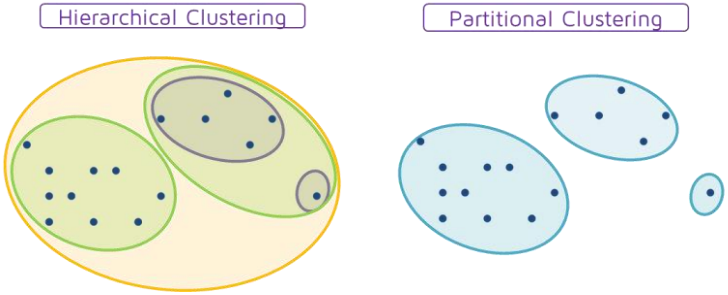
Sumber : socs.binus.ac.id

Gambar 2.1 Gambaran umum Hierarchical Clustering

2.1.2. *Partitional Clustering*

Partitional clustering yaitu data dikelompokkan ke dalam sejumlah *cluster* tanpa adanya struktur hirarki antara satu dengan yang lainnya. Metode *partitional clustering memiliki ciri dimana* setiap *cluster* memiliki titik pusat *cluster* (*centroid*) dan secara umum metode ini memiliki fungsi tujuan yaitu meminimumkan jarak (*dissimilarity*) dari seluruh data ke pusat *cluster* masing-masing.

Tentu sudah terlihat sangat terlihat berbeda dengan *Hierarchical Clustering* dimana pada *Partitional Clustering* tidak terdapat hirarki kedekatan antar kelompok melainkan langsung terbagi dengan jumlah kelompok yang pasti. Perbedaan antara *Hierarchical Clustering* dimana dan *Partitional Clustering* dapat dilihat pada Gambar 2.2



Sumber : quantdare.com

Gambar 2.2 Perbedaan Hierarchical dengan Partitional Clustering

2.2. *Kernel Entropy Component Analysis* dengan *Quantum Clustering* (KECA-QC)

KECA-QC merupakan metode baru yang memanfaatkan KECA sebagai *pre-processing* dan QC sebagai *clustering*. Algoritma ini diklaim dapat melakukan *clustering* dengan waktu yang cepat meskipun pada dataset berdimensi besar dan dapat melakukan *clustering* pada dataset berbentuk apapun tanpa mengetahui jumlah *cluster*. Algoritma ini juga telah dibandingkan dengan algoritma lain seperti k-means (KM), Ng-Jordan-Weiss (NJW), traditional QC, dan KECA-KM yang menunjukkan hasil bahwa KECA-QC menunjukkan hasil yang paling baik [2].

2.3. *Eigenvalues* dan *Eigenvectors*

Eigenvalue adalah sebuah bilangan skalar dan *eigenvector* adalah sebuah matriks yang keduanya dapat mendefinisikan matriks A. Matriks A adalah matriks bujur sangkar dengan ukuran nxn. Namun, tidak semua matriks bujur sangkar memiliki *eigenvalue* dan

eigenvector. Contoh, sebuah matriks 2x2 A = $\begin{pmatrix} 3 & 0 \\ -6 & 0 \end{pmatrix}$ dan sebuah matriks (atau vektor) $x_1 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$. Maka A$x_1$ = $\begin{pmatrix} -3 \\ 6 \end{pmatrix}$. Maka bisa dilihat bahwa Ax_1 merupakan perpanjangan/kelipatan dari x_1 dengan persamaan A$x_1 = \lambda x_1$ = $\begin{pmatrix} 3 & 0 \\ -6 & 0 \end{pmatrix}\begin{pmatrix} -1 \\ 2 \end{pmatrix}$ = $3\begin{pmatrix} -1 \\ 2 \end{pmatrix}$. Maka $x_1 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$ adalah *eigenvector* A dan $\lambda = 3$ adalah *eigenvalue*-nya [3].

2.4. *Gradient Descent*

Gradient descent adalah algoritma optimasi orde pertama untuk menemukan minimum lokal dari fungsi menggunakan *gradient descent*, diambil langkah sebanding dengan negatif dari gradien (atau perkiraan gradien) dari fungsi pada titik sekarang[6]. Hal tersebut dapat dicapai dengan melakukan iterasi pada persamaan

$$y_i(t + \Delta t) = y_i(t) - \eta(t)\nabla V(y_i(t)) \qquad \textbf{(2.1)}$$

Keterangan:

yi : adalah data awal (Φ_{keca}) dimana $y_i(0)=p_i$

$\boldsymbol{\eta(t)}$: merupakan kecepatan iterasi

∇V : merupakan gradien dari V(p)

Terdapat beberapa algoritma yang digunakan untuk memaksimalkan kinerja dari *gradient descent*, diantaranya adalah sebagai berikut:

a. Momentum
b. Nesterov Accelerated Gradient (NAG)
c. AdaGrad
d. RMSProp

2.4.1. Momentum

Momentum merupakan metode yang membantu mempercepat *gradient descent*. Hal ini dilakukan dengan menambah parameter γ. Perubahan posisi dapat dicari dengan persamaan iterasi dibawah ini

$$y_i(t + \Delta t) = y_i(t) - v(t + \Delta t) \quad \textbf{(2.2)}$$

$$v(t + \Delta t) = \gamma v(t) + \eta(t)\nabla V\big(y_i(t)\big) \quad \textbf{(2.3)}$$

Keterangan:

yi : adalah data awal (Φ_{keca}) dimana $y_i(0)=p_i$

$\boldsymbol{\eta}$: merupakan kecepatan iterasi

∇V : merupakan gradien dari V(p)

$\boldsymbol{v}$ **:** merupakan kecepatan momentum

γ **:** merupakan parameter momentum dengan nilai $\gamma < 1$

Intinya, saat menggunakan momentum dapat dianalogikan sebagai bola yang menuruni bukit dengan kecepatan yang terus bertambah. Hal yang sama terjadi pada pembaruan parameter. Masa momentum meningkat untuk dimensi yang titik puncaknya mengarah ke arah yang sama dan mengurangi pembaruan dimensi yang arah gradiennya berubah [4].

2.4.2. Nesterov Accelerated Gradient (NAG)

Bagian sebelumnya telah membahas bahwa algoritma optimasi momentum dapat dianalogikan sebagai bola yang menuruni bukit dengan kecepatan yang terus bertambah, tentunya hal tersebut memiliki kekurangan yaitu bola menuruni bukit dengan cepat namun akan terus melaju meskipun terdapat tanjakan. Dengan metode NAG ini analogi bola yang sedang menuruni bukit tersebut dapat turun dengan mengetahui akan adanya tanjakan sehingga bola bisa melambat agar tidak menaiki tanjakan.

Penerapan NAG memperhitungkan kecepatan sebelumnya untuk mengoreksi arah minimum lokal dengan rumus sebagai berikut

$$y_i(t + \Delta t) = y_i(t) - v(t + \Delta t) \quad \textbf{(2.4)}$$

$$v(t + \Delta t) = \gamma v(t) + \eta(t)\nabla V\big(y_i(t) - v(t)\big) \quad \textbf{(2.5)}$$

Keterangan:

yi : adalah data awal (Φ_{keca}) dimana $y_i(0)=p_i$

$\boldsymbol{\eta}$: merupakan kecepatan iterasi

$\boldsymbol{\nabla V}$ **:** merupakan gradien dari V(p)

$\boldsymbol{v}$ **:** merupakan kecepatan momentum

γ **:** merupakan parameter momentum dengan nilai $\gamma < 1$

2.4.3. AdaGrad

Cara lain untuk memperbaiki *Gradient Descent* selain dengan momentum adalah dengan memberikan kecepatan update yang berbeda pada tiap dimensi vektor parameter yi dan kemudian mampu beradaptasi berdasarkan indikator tertentu. Salah satu indikator yang dapat dipakai adalah besarnya perubahan nilai vektor *gradient* pada dimensi tertentu. Realisasi dari ide ini dikenal dengan metode *adaptive subgradient descent* (AdaGrad) yang dapat ditulis sebagai rumus berikut (Duchi et al, 2011) [5].

$$y(t+\Delta t) = y(t) - \frac{\eta(t)}{\sqrt{G(t)+\varepsilon}} \odot (g(t)) \qquad \textbf{(2.6)}$$

Keterangan:

y : adalah data awal (Φ_{keca}) dimana $y_i(0)=p_i$

$\boldsymbol{\eta(t)}$: merupakan kecepatan iterasi

$\boldsymbol{g(t)}$ **:** merupakan gradien dari V(p)

$\boldsymbol{G(t)}$ **:** merupakan matriks diagonal dimana setiap elemennya merupakan jumlah dari $g(t)$ dikuadratkan

$\boldsymbol{\varepsilon}$ **:** merupakan penambahan untuk menghindari pembagian dengan nol dimana biasanya ε = 1e-8

2.4.4. RMSProp

AdaGrad memiliki sebuah problem, yaitu pada waktu tertentu nilai $G(t)$ berpotensi sangat besar sehingga malah akan

memperlambat proses optimisasi. Untuk mengatasi hal ini, (Tieleman dan Hinton, 2012) melakukan sedikit modifikasi terhadap AdaGrad dengan menambahkan konstanta untuk mengatur besaran masing-masing dari g2tgt2 dan ntnt. Modifikasi ini menghasilkan nama algoritma baru yang disebut dengan RMSProp. Algoritma ini dapat ditulis sebagai berikut

$$y(t+\Delta t) = y(t) - \frac{\eta(t)}{\sqrt{E[g(t)^2]+\varepsilon}}(g(t)) \quad \textbf{(2.7)}$$

$$E[g(t)^2] = 0.9E[g(t-1)^2] + 0.1g(t)^2 \quad \textbf{(2.8)}$$

Keterangan:

y : adalah data awal (Φ_{keca}) dimana $y_i(0)=p_i$

$\boldsymbol{\eta(t)}$: merupakan kecepatan iterasi

$\boldsymbol{g(t)}$ **:** merupakan gradien dari V(p)

$\boldsymbol{E}$ **:** rata-rata

$\boldsymbol{\varepsilon}$ **:** merupakan penambahan untuk menghindari pembagian dengan nol dimana biasanya ε = 1e-8

2.5. Euclidean Distance

Adalah salah satu metode menghitung jarak antara dua objek (vector atau titik) pada sebuah ruang dimensi. Secara matematis dapat dihitung dengan persamaan (2.9) ataupun (2.10) sesuai dengan kebutuhannya:

a. Jarak pada ruang dimensi dua:

$$d(p,q) = \sqrt{(q1-p1)^2 + (q2-p2)^2} \quad \textbf{(2.9)}$$

b. Jarak pada ruang dimens-n:

$$d(p,q) = \sqrt{(q1-p1)^2 + (q2-p2)^2 + \cdots + (qn-pn)^2} \quad \textbf{(2.10)}$$

Keterangan:

$\boldsymbol{x, y}$ adalah titik pada ruang dimensi satu

$\boldsymbol{p, q}$ adalah vektor pada ruang dimensi 2 atau lebih

[Halaman ini sengaja dikosongkan]

.

BAB III
PERANCANGAN PERANGKAT LUNAK

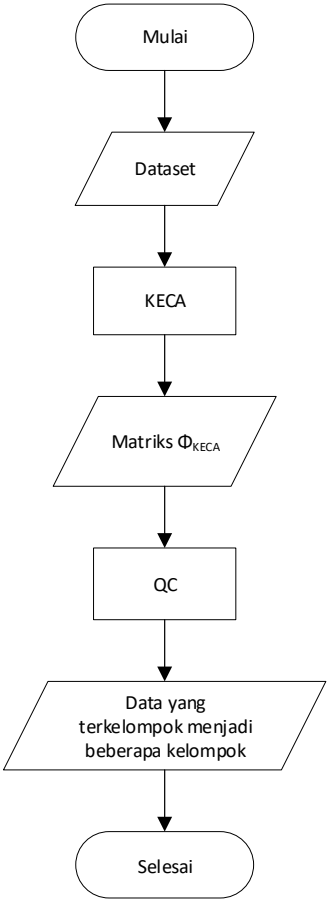
Bab ini menjelaskan mengenai rancangan sistem perangkat lunak yang akan dibuat. Perancangan yang dijelaskan meliputi data , proses, dan perhitungan metrik. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak. Proses terbagi menjadi dua bagian besar yaitu *pre-processing* dan *clustering. Pre-processing* yaitu tahap-tahap yang dilakukan pada data sebelum diproses lebih lanjut. *Clustering* ialah proses dimana data dikelompokkan berdasarkan kedekatan tertentu. Terdapat tiga metode perhitungan metrik yang digunakan untuk menguji hasil *clustering* diantaranya, *Jaccard Score*, *Minskowski Score*, dan *Cluster Accuracy*

3.1. Data

Sub bab ini akan menjelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan. Kumpulan data atau dataset yang digunakan pada penelitian ini didapatkan dari UCI *Machine Learning*. Selain data dari UCI, dilakukan juga pengujian menggunakan dataset buatan berukuran dua dimensi agar proses pengelompokkan yang dilakukan lebih bisa divisualisasikan prosesnya.

3.2. Proses

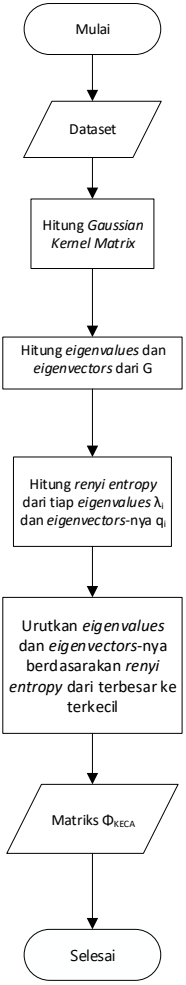
Tugas akhir yang dikerjakan merupakan penerapan dari algoritma KECA-QC terbagi menjadi dua tahap utama, yaitu tahap *pre-processing* dan tahap *clustering* yang secara sederhana dapat ditunjukkan pada diagram alir berikut ini.



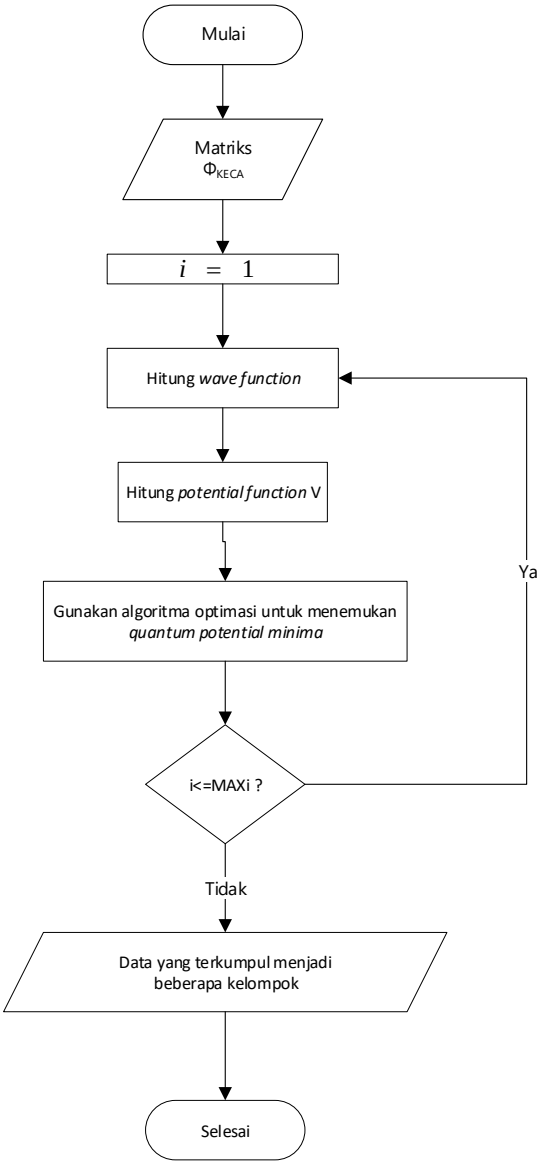
Gambar 3.1 Diagram Alir Sederhana KECA-QC

Diagram alir tersebut memperlihatkan bahwa metode KECA-QC memiliki dua proses utama yaitu KECA dan QC dimana KECA digunakan untuk *preprocessing* dataset yang kemudian dilanjutkan dengan *clustering* menggunakan QC dengan hasil data yang sudah terkelompok.

Untuk mengetahui proses metode KECA-QC yang lebih terperinci dimana terdapat beberapa perhitungan yang harus dilakukan pada masing-masing tahap KECA (*pre-processing*) maupun QC (clustering), berikut ini ditampilkan diagram alur KECA-QC secara rinci pada Gambar 3.2 dan pada Gambar 3.3.



Gambar 3.2 Diagram Alir KECA



Gambar 3.3 Diagram Alir QC

3.2.1. *Pre-Processing*

Tahap *preprocessing* pada awalnya berisi *Gaussian Kernel Matrix* dari dataset awal $X = \{x_1, x_2, \dots, x_N\}$ dengan persamaan

$$G(x_i, x_j) = e^{-||x_i - x_j||/2\sigma^2} \quad \textbf{(3.1)}$$

Keterangan:

$\boldsymbol{G(x_i, x_j)}$: matriks simetris dengan ukuran NxN

$\boldsymbol{e}$: basis logaritma alami

$\boldsymbol{x_n}$: nilai dari data ke-n

Lalu hitung *eigenvalues* dan *eigenvectors* dari G. Kemudian hitung *renyi entropy* dari tiap *eigenvalues* λ_i dan *eigenvectors*-nya q_i dengan rumus

$$r_i = (\sqrt{\lambda_i} q_i^T 1)^2 \quad \textbf{(3.2)}$$

Keterangan:

$\boldsymbol{r_i}$: *renyi entropy* dari tiap *eigenvalues* dan *eigenvectors*-nya

$\boldsymbol{\lambda}$: *eigenvalues* dari G

$\boldsymbol{q}$: *eigenvectors* dari G

1: vector berukuran Nx1 dengan masing-masing elemen bernilai 1

Setelah itu urutkan *eigenvalues* dan *eigenvectors*-nya berdasarkan *renyi entropy* dari terbesar ke terkecil. Terakhir, Bentuklah Matriks

$$\Phi_{keca} = \Lambda_{keca\_l}^{\frac{1}{2}} Q_{keca\_l}^T \quad \textbf{(3.3)}$$

Keterangan:

$\boldsymbol{\Phi_{keca}}$: matriks akhir keca

$\boldsymbol{\Lambda}$: matriks diagonal yang berisi *eigenvalues* dari G

$\boldsymbol{Q}$: *eigenvectors* dari G

Nilai l pada $\Lambda_{keca\_l}^{\frac{1}{2}}$ dan $Q_{keca\_l}^{T}$ harus memenuhi persamaan berikut

$$r_l - r_{l+1} = \max(r_i - r_{i+1}), \quad i = 1,2,\dots,N-1 \tag{3.4}$$

Keterangan:

$\boldsymbol{r}$: *renyi entropy*

3.2.2. *Clustering*

Tahap *clustering* menggunakan Φ_{keca} sebagai *input*. Pertama kita harus menghitung *wave function* dengan

$$\psi(p) = \sum_{p_{(k)} \epsilon \Gamma_k(p)} e^{-\left|\left|p - p_{(k)}\right|\right|^2 / 2\sigma^2} \tag{3.5}$$

Keterangan:

$\boldsymbol{e}$: basis logaritma alami

$\boldsymbol{p}$: nilai dari data

Dari *wave function,*kita dapat menghitung *potential function* dengan rumus

$$V(p) = E - \frac{d}{2} + \frac{1}{2x^2\Psi} \sum_{p_{(k)} \epsilon \Gamma_k(p)} \left|\left|p - p_{(k)}\right|\right|^2 \exp\left[-\frac{\left|\left|p - p_{(k)}\right|\right|^2}{2x^2}\right] \tag{3.6}$$

$$E = -min(-\frac{d}{2} + \frac{1}{2\sigma^2\psi}) \sum_{p_{(k)} \epsilon \Gamma_k(p)} \left|\left|p - p_{(k)}\right|\right|^2 exp\left[-\frac{\left|\left|p - p_{(k)}\right|\right|^2}{2x^2}\right] \tag{3.7}$$

Keterangan:

$\boldsymbol{d}$: dimensi dari data

$\boldsymbol{exp}$: basis logaritma alami

$\boldsymbol{p}$: nilai dari data

Dari hasil V(p) tersebut perlu dicari *quantum potential minima* sebagai pusat *cluster* dengan melakukan iterasi menggunakan algoritma optimasi yang sudah dijelaskan sebelumnya. Iterasi dilakukan sebanyak maxi, menurut paper referensi, 20 iterasi sudah cukup untuk menemukan *quantum potential minima* [2]. Hasil dari iterasi tersebut adalah terkumpulnya data menjadi beberapa kelompok. Terakhir, tetapkan setiap titik asli menjadi suatu *cluster* yang sama jika pada hasil iterasi, data tersebut terkumpul pada kelompok yang sama [3].

3.3. Perhitungan Metrik

Tahap ini berisikan beberapa metode untuk mengukur seberapa baik suatu algoritma *clustering* dalam mengelompokkan data yang akan diujikan. Berikut beberapa metode yang akan dilakukan pada penelitian ini:

a. *Jaccard Score*
b. *Minkowski Score*
c. *Cluster Accuracy*

3.3.1. Jaccard Score

Jaccard Score merupakan cara yang digunakan untuk mengukur tingkat kemiripan dari dua himpunan yang saling beririsan. Perhitungan kesamaan tersebut dapat dirumuskan seperti pada persamaan (3.1).

$$JS = \frac{n_{11}}{n_{11} + n_{01} + n_{10}} \quad \textbf{(3.1)}$$

Keterangan:

$\boldsymbol{n_{11}}$: pasangan benar dan dideteksi benar

$\boldsymbol{n_{01}}$: pasangan salah namun dideteksi benar

$\boldsymbol{n_{10}}$: pasangan benar namun dideteksi salah

Dari persamaan diatas dapat dilihat bahwa semakin besar nilai dari Jaccard Score menunjukkan semakin baiknya suatu algoritma dalam mengelompokkan data dengan nilai terbaik 1.

3.3.2. Minkowski Score

Minkowski Score mengukur seberapa banyak perbedaan antara dua himpunan yang saling beririsan. Perhitungan Minkowski Score dapat dirumuskan sebagai berikut.

$$MS = \sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}} \qquad \textbf{(3.2)}$$

Keterangan:

$\boldsymbol{n_{11}}$: pasangan benar dan dideteksi benar

$\boldsymbol{n_{01}}$: pasangan salah namun dideteksi benar

$\boldsymbol{n_{10}}$: pasangan benar namun dideteksi salah

Berbeda dengan Jaccard Score, Minkowski Score memiliki nilai yang semakin kecil saat suatu data dikelompokkan dengan semakin baik dengan nilai terbaik 0.

3.3.3. Cluster Accuracy

Cluster Accuracy atau kadang disebut *purity* merupakan perhitungan untuk mengukur akurasi dari *clustering* dengan menghitung sebanyak-banyaknya anggota kelompok yang dianggap sebagai bagian dari suatu kelompok adalah memang sebenarnya tergabung dalam kelompok yang sama.

$$CA = \frac{1}{N}\sum_{i=1}^{T} \max Confusion(i,j), \qquad \textbf{(3.1)}$$

Keterangan:

$\boldsymbol{T}$: Jumlah Solusi dari kelompok sebenarnya

$\boldsymbol{N}$: Jumlah semua data

Cluster Accuracy sama dengan Jaccard Score yang nilainya semakin besar saat pengelompokkan data semakin baik dengan nilai terbaik 1.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini akan membahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan sepenuhnya menggunakan bahasa Matlab.

4.1. Lingkungan implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan implementasi Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i7-2640M CPU @ 2.80 GHz Memori: 4.00 GB
Perangkat lunak	Sistem Operasi: Windows 10 Education 64-bit Perangkat Pengembang: Matlab

4.2. Implementasi

Sub-bab implementasi ini menjelaskan tentang bagaimana teori yang sebelumnya telah dipaparkan pada bab desain perangkat lunak, diterapkan dalam bentuk kode sumber dalam bahasa pemrograman matlab dan penjelasan mengenai step yang dilakukan pada kode sumber tersebut.

4.2.1. Implementasi Tahap Pre-processing

Sub bab ini menjelaskan tentang implementasi tahapan *pre-processing* yaitu menggunakan metode kernel entropy component analysis (KECA) dimana data dataset akan diproses menjadi matriks simetris yang kemudian akan direduksi dimensinya dengan metode yang sudah digunakan pada *paper* karya Yangyang Li.

Tahap paling awal adalah dengan mengubah data menjadi matriks simetris yang berisi gaussian kernel dengan cara menjalankan Kode Sumber 4.1. Setiap data pada matriks merupakan hasil perhitungan dengan rumus gaussian yang sangat dipengaruhi oleh variabel sigma. Sigma didapat dari perhitungan rata-rata jarak tiap data ke sejumlah k terdekat dari tiap data tersebut. Untuk fungsi perhitungan sigma terdapat pada Kode Sumber 4.2.

```
1.  function [K] = kernel(data,sigma)
2.      X = squareform(pdist(data));
3.      alpha = 1/(2*sigma^2);
4.      K = exp(-1*alpha*X.^2);
5.  end
```

Kode Sumber 4.1 Perhitungan *Gaussian Kernel Matrix*

```
1.  function [sigma] = ct_sigma(M,baris)
2.      sigma = sort(squareform(pdist(M)).^2,2);
3.      sigma = sum(sigma(:,2:51),2);
4.      sigma = sum(sigma,1)/(baris*49);
5.  end
```

Kode Sumber 4.2 Perhitungan Sigma

Kode Sumber 4.3 merupakan proses KECA . Tahap awal yang dilakukan adalah menghitung dan mengurutkan *eigen value* dan *eigen vector* berdasarkan *eigen value* dari yang terbesar ke terkecil dengan perhitungan *eigen value* dan *eigen vector* terdapat pada Kode Sumber 4.3 baris kedua dengan menggunakan fungsi yang sudah terdapat pada matlab yaitu fungsi eig. Untuk mengurutkan *eigen value* dan *eigen vector* tersebut digunakan fungsi sort_eigenvalues yang dipaparkan pada Lampiran 7.1. Kemudian data akan dihitung *entropy* nya yang kemudian entropy tersebut diurutkan dari yang terbesar hingga terkecil yang terdapat pada baris kelima dengan menmanggil fungsi ECA. Fungsi ECA yang dipanggil tersebut dapat dilihat pada Lampiran 7.2. Baris keenam hingga lima belas merupakan proses penentuan variabel l. Dimana l merupakan indeks yang memiliki nilai

selisih terbesar terhadap indeks setelahnya. Variabel l tersebut digunakan untuk menentukan jumlah dimensi yang tepat untuk hasil akhir dari KECA nantinya. Setelah itu, pada baris enam belas dan tujuh belas *eigen value* dan *eigen vector* diurutkan berdasarkan *entropy* yang sebelumnya sudah terurut pada pemanggilan fungsi ECA. Bentuk akhir dari proses KECA diproses pada perulangan yang terdapat pada baris delapan belas hingga dua puluh yaitu dengan mengalikan akar dari tiap *eigen value* dengan tiap *eigen vector nya*.

```
1.  function [Phi,d,E,sorted_entropy] =
    kernelECA(K,c,n,baris);
2.      [E,D] = eig(K);
3.      [D,E] = sort_eigenvalues(D,E);
4.      d = diag(D)';
5.      [sorted_entropy_index,sorted_entropy,entropy] =
    ECA(D,E);
6.
7.      l=0;
8.      max=-999999;
9.      for i = 1 : baris-1;
10.         temp=sorted_entropy(i)-sorted_entropy(i+1);
11.         if(temp>max);
12.             max=temp;
13.             l=i;
14.         end;
15.     end;
16.
17.     C=l;
18.     Es = E(:,sorted_entropy_index);
19.     ds = d(sorted_entropy_index);
20.     for i = 1 : C;
21.         Phi(:,i) = sqrt(ds(i)) * Es(:,i);
22.     end;
23. End
```

Kode Sumber 4.3 KECA

4.2.2. Implementasi Tahap Clustering

Kode Sumber 4.4 dan Kode Sumber 4.4 merupakan proses quantum clustering dengan menghitung *wave function* (P) yang terdapat dari baris pertama Kode Sumber 4.4 hingga baris sembilan Kode Sumber 4.5 kemudian dilanjutkan dengan menghitung *potential function* (V). Kode Sumber 4.4 baris kedua hingga kesepuluh merupakan kode untuk mempersiapkan matriks yang akan digunakan pada perhitungan *wave function* (P) maupun *potential function* (V).Kemudian pada baris sebelas dilakukan penentuan data mana saja yang termasuk k terdekat dari setiap data tersebut. Setelah itu, proses berlanjut ke Kode Sumber 4.5.

```
1.  function [V,P,E,dV] = qc (ri,q,r)
2.  [pointsNum,dims] = size(ri);
3.  calculatedNum=size(r,1);
4.  V=zeros(calculatedNum,1);
5.  dP2=zeros(calculatedNum,1);
6.  P=zeros(calculatedNum,1);
7.  singlePoint = ones(pointsNum,1);
8.  singleLaplace = zeros(50,1);
9.  singledV1=zeros(50,dims);
10. singledV2=zeros(50,dims);
11. [sorted_dri,sorted_dri_index] =
    sort(squareform(pdist(ri)).^2,2);
```

Kode Sumber 4.4 Quantum Clustering 1

Kode Sumber 4.5 terdapat perulangan untuk menghitung tiap *wave function* (P) dari data yang didalamnya terdapat tiga perulangan. Perulangan pertama digunakan untuk menghitung *gaussian* dari tiap nilai matriks pada data hasil KECA yang selanjutnya akan dijumlahkan menjadi *wave function* (P) untuk setiap data. Setelah didapat hasilnya, pada baris ke-22 kita dapat lihat bahwa *wave function* (P) yang memiliki nilai nol diganti untuk menghindari nilai tak hingga pada perthitungan kedepannya. Baris selanjutnya berisi *potential function* (V) yang sudah dapat dihitung karena *wave function* (P) sudah didapatkan. Setelah itu, dicari gradient dari V untuk

selanjutnya diproses saat pencarian minima lokal pada *gradient descent.*

```
1.  for point = 1:calculatedNum
2.      singlePoint = ones(pointsNum,1);
3.      singleLaplace = singleLaplace.*0;
4.      ri=ri(sorted_dri_index(point,:),:);
5.      D2=sum(((repmat(r(point,:),50,1)-
    ri(2:51,:)).^2)');
6.      singlePoint=exp(-q*D2)';
7.      for dim=1:dims
8.          singleLaplace = singleLaplace + (r(point,dim)-
    ri(2:51,dim)).^2.*singlePoint;
9.      end;
10.     for dim=1:dims
11.         singledV1(:,dim) = (r(point,dim)-
    ri(2:51,dim)).*singleLaplace;
12.     end;
13.     for dim=1:dims
14.         singledV2(:,dim) = (r(point,dim)-
    ri(2:51,dim)).*singlePoint;
15.     end;
16.     P(point) = sum(singlePoint);
17.     dP2(point) = sum(singleLaplace);
18.     dV1(point,:)=sum(singledV1,1);
19.     dV2(point,:)=sum(singledV2,1);
20. end;
21. P(find(P==0)) = min(P(find(P)));
22. V=-dims/2+q*dP2./P;
23. E=-min(V);
24. V=V+E;
25. for dim=1:dims
26.     dV(:,dim)=-q*dV1(:,dim)+(V-
    E+(dims+2)/2).*dV2(:,dim);end
27. dV(find(P==0),:)=0;
```

Kode Sumber 4.5 Quantum Clustering 2

Proses Quantum Clustering dilakukan berulang-ulang sebanyak dua puluh iterasi yang diproses di dalam gradient descent dan algoritma optimasi lainnya untuk mencapai minima lokal. Proses algoritma optimasi sendiri pada percobaan ini dilakukan dengan lima macam algoritma optimasi yang diimplementasikan ke dalam satu fungsi yang dicantumkan pada Kode Sumber 4.6.

Kode Sumber 4.6 dilakukan algoritma optimasi untuk menemukan minima lokal dari *potential funtion*. Perulangan yang berada antara baris ketujuh hingga baris ke-24 memiliki beberapa fungsi if yang menentukan algoritma optimasi apakah yang akan digunakan untuk menemukan lokal minima dari *potential function*. Fungsi if pada baris kesepuluh akan dikerjakan jika algoritma optimasi yang digunakan adalah *gradient descent* yaitu saat variabel opt bernilai 1. Jika variabel opt bernilai 2 maka algoritma optimasi yang akan digunakan adalah algoritma optimasi gradient descent dengan momentum yakni dengan mengerjakan fungsi if dari baris ke tiga belas hingga baris ke enam belas. Jika variabel opt bernilai 3, maka proses akan dilakukan sama seperti saat variabel opt bernilai 2, hanya kali ini akan dikerjakan juga proses yang terdapat dalam fungsi if pada baris ke tujuh belas. Algoritma yang digunakan saat variabel Ketika variabel opt bernilai 4, maka akan dijalankan proses pada baris ke empat yang mana merupakan algoritma optimasi yang dinamai AdaGrad. opt bernilai 3 tersebut bernama *Nesterov Accelerated Gradient* (NAG). AdaGrad memiliki kekurangan dimana konstanta yang digunakan untuk menghindari pembagian dengan nol akan terus menumpuk nilainya dan membuat pencarian lokal minima tidak terkendali. Untuk mengatasi masalah tersebut digunakan algoritma optimasi bernama RMSProp yang akan mengoreksi pencarian lokal minima pada setiap iterasi.

```
1.  function D=graddesc(xyData,q,steps,opt)
2.
3.  eta=0.1;
4.  D=xyData;
5.  tD=D;
6.  [V,P,E,dV] = qc (D,q,D);
7.
8.  for j=1:4
9.      for i=1:(steps/4)
10.         dV=normc(dV')';
11.         Eg=dV;
12.         if(opt==1);
13.         D=D-eta*dV;
14.         end;
15.         if(opt==4);
16.         D=D-(eta./((diag(sum(dV.^2,2))+
    0.00000001).^.5))*dV;
17.         end;
18.         if(opt==2 || opt==3)
19.         velo=0.3*(D-tD)-eta*dV;
20.         tD=D;
21.         D= D+velo;
22.           if(opt==3);
23.            D=D-0.3*(D-tD);
24.            end;
25.         end;
26.         [V,P,E,dV] = qc (D,q,D);
27.         if(opt==5)
28.         Eg=0.9*mean(Eg(:).^2)+0.1.*(dV.^2);
29.         D=D-(eta./((Eg+ 0.00000001).^.5)).*dV;
30.         end
31.     end;
32.     eta=eta*0.5;
33. end
```

Kode Sumber 4.6 Algoritma Optimasi

Kode Sumber 4.7 merupakan fungsi yang digunakan untuk perhitungan metrik dari metode KECA-QC yang telah diterapkan. Perhitungan metrik *Minkowski Measure* terdapat pada baris ke sepuluh, kemudian perhitungan metrik *Jaccard Measure* terdapat pada baris ke tiga belas. *Minkowski Measure* merupakan metode perhitungan metrik *clustering* yang memiliki nilai yang semakin kecil seiring dengan hasil dari pengelompokkan yang semakin baik karena metode ini menunjukkan tingkat kesalahan dari pengelompokkan, sehinnga semakin kecil nilainya maka semakin baik. Sementara *Jaccard Measure* merupakan irisan hasil pengelompokkan bersama kelompok sesungguhnya dibagi dengan dengan gabungan hasil pengelompokkan bersama kelompok sesungguhnya, sehingga semakin besar nilainya semakin baik.

```
1.  function
    [minkowski_measure,jacard_measure,purity,efficienc
    y,purity2,sNum]=clustMeasure(clust,realClust)
2.  pNum=length(clust);
3.  S=(repmat(clust,1,pNum)==repmat(clust',pNum,1));
4.  for i=1:(length(realClust)-1);
5.      t(realClust(i):(realClust(i+1)-1))=i;
6.  end
7.  l=length(clust);
8.  t(realClust(i+1):l)=i+1;
9.  T=(repmat(t',1,l)==repmat(t,l,1));
10. minkowski_measure=sqrt(sum(sum(T~=S))/sum(sum(T==1
    )));
11. S1=S*2-1;
12. TP=(T==S1);
13. jacard_measure =
    sum(sum(TP))/(sum(sum(T~=S))+sum(sum(TP)));
```

Kode Sumber 4.7 Perhitungan Metrik 1

Kode Sumber 4.8 memperlihatkan perhitungan metode *Cluster Accuracy*. Perhitungannya sederhana yaitu menjumlahkan

nilai maksimum dari setiap data yang terkelompok pada kelompok yang sama dan memang sebenarnya terdapat pada kelompok yang sama

.

```
1.  tNum=length(realClust);
2.  sNum=max(clust);
3.  kfrom=1;
4.  kto=0;
5.  maks=0;
6.  total=0;
7.  for i=1:l;
8.       for j=1:tNum;
9.            kto=kto+sum(t(:)==j);
10.           temp=sum(S(i,kfrom:kto));
11.           if(temp>maks);maks=temp;end
12.           kfrom=kto+1;
13.      end
14.      total=total+maks;
15.      kfrom=1;
16.      kto=0;
17.      maks=0;
18. end
19. cluster_accuracy=total/sum(sum(S==1));
```

Kode Sumber 4.8 Perhitungan Metrik 2

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini akan menjelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi. Evaluasi dilakukan dengan memvisualisasikan dataset yang berdimensi dua dan menghitung keakuratan dari seluruh dataset dengan menggunakan tiga macam perhitungan metrik untuk menguji suatu metode *clustering* yang digunakan.

5.1. Lingkungan Pengujian

Lingkungan uji coba yang akan digunakan adalah,

1. Perangkat Keras
   Prosesor: Intel® Core™ i7-2640M CPU @ 2.80 GHz
   Memori: 4.00 GB.
   Sistem Operasi: 64-bit .
2. Perangkat Lunak
   Sistem Operasi: Windows 10 Education.
   Perangkat Pengembang: Matlab.

5.2. Data Uji Coba

Data yang digunakan untuk studi kinerja dari metode KECA-QC menggunakan tiga buah data dari UCI dan juga lima buah data buatan . Seluruh data buatan berdimensi dua dengan jumlah data pada setiap dataset yang beragam. Untuk dataset UCI, dataset iris memiliki dimensi yang paling sedikit dibandingkan dengan dataset UCI lainnya yaitu hanya empat dimensi. Dataset WBC memiliki jumlah dimensi sembilan, dan dataset terakhir yaitu dazaset zoo memiliki data dengan dimensi berjumlah enam belas. Seluruh data pada dataset telah diurutkan terlebih dahulu agar mempermudah dalam melakukan analisis pada plot clustering yang terdapat pada Lampiran 7.18. Hasil sempurna pada plot *clustering* tersebut dapat dilihat dengan mudah yaitu saat hasil plot berbentuk diagonal.

5.3. Skenario dan Evaluasi Pengujian

Sub bab ini akan menjelaskan mengenai skenario dan evaluasi pengujian yang dilakukan pada implementasi metode KECA-QC untuk clustering pada data. Pengujian dilakukan dengan melalui berbagai macam algoritma optimasi untuk memaksimalkan kinerja dari *gradient descent*. Algoritma optimasi yang digunakan adalah sebagai berikut

1. *Gradient descent.*
2. *Gradient descent* dengan *momentum.*
3. *Gradient descent* dengan nesterov accelerated gradient.
4. *Gradient descent* dengan AdaGrad.
5. *Gradient descent* dengan RMSprop.

Untuk evaluasi dari pengujian menggunakan tiga metode dari pengukuran kinerja clustering berikut ini.

1. *Jaccard Measure (JM).*
2. *Minkowski Measure (MM).*
3. *Cluster Accuracy (CA).*

Sebelum melihat nilai yang didapat dengan menggunakan tiga macam perhitungan metrik diatas, terlebih dahulu akan diperhatikan apakah hasil akhir *clustering* mengelompokkan data sejumlah data yang ada sehingga setiap *cluster* hanya memiliki sedikit anggota yang mengakibatkan metrik dari *Cluster Accuracy* nyaris ataupun memang sempurna, namun hal tersebut tidak baik karena nilai sempurna tersebut tidak dapat memberikan pengetahuan mengenai *cluster* data tersebut. Oleh karena itu, jika terjadi hal demikian, nilai dari algoritma optimasi yang sesuai dengan kasus tersebut tidak akan diangap sebagai nilai yang baik dibandingkan dengan algoritma optimasi lain.

5.4. Hasil Uji Coba

Awal sub bab ini akan berisikan pengujian terlebih dahulu apakah memang benar bahwa KECA dapat digunakan untuk mereduksi dimensi dari suatu data. Oleh karena itu setelah ini akan ditunjukkan hasil reduksi dimensi dari KECA yang akan dipaparkan dalam bentuk tabel agar mudah dipahami. Berikut adalah tabel yang dimaksud.

Tabel 5.1 Hasil Reduksi dimensi KECA

Dataset	Dimensi Awal	Dimensi Akhir
Cluster in cluster	2	2
Crescent & Fullmoon	2	2
Corners	2	2
Half-kernel	2	2
Spiral	2	2
Zoo	**16**	**9**
Iris	**4**	**3**
WBC	**9**	**2**

Dari Tabel 5.1 diatas dapat dilihat bahwa KECA dapat dengan baik mereduksi dimensi dari suatu dataset, terutama pada dataset UCI. Dataset buatan tidak menghasilkan reduksi dimensi karena dimensi awal dari data buatan sudah memuat hal penting yang tidak dapat direduksi kembali.

Sub bab ini juga akan ditampilkan hasil uji coba berupa visualisasi dan perhitungan metrik berdasarkan plot hasil *clustering* yang terdapat pada lampiran 7.18.Hasil visualisasi yang ditampilkan hanya pada dataset buatan yaitu dataset yang berdimensi dua karena pada dataset UCI yang memiliki jumlah dimensi yang banyak akan sulit untuk divisualisasikan pada bidang kertas seperti ini.

5.4.1. Visualisasi Data

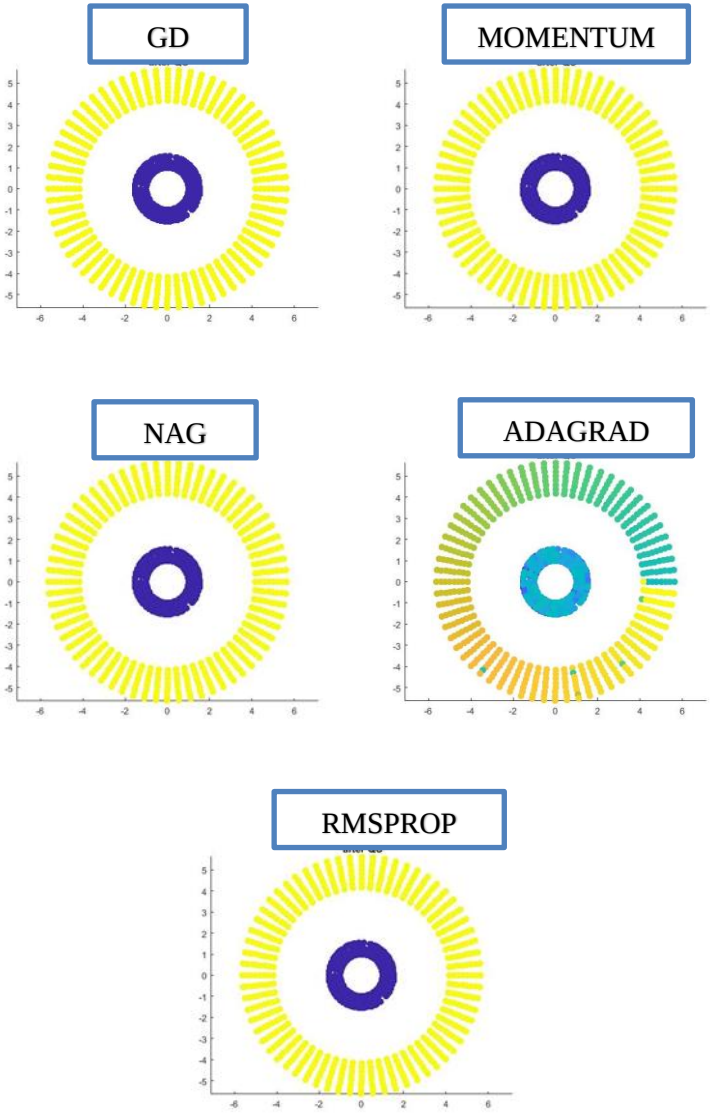
Bagian ini akan memaparkan visualisasi hasil akhir dari proses KECA-QC yang diterapkan pada data berdimensi dua atau tiga. Hal ini dilakukan dengan tujuan untuk mempermudah analisis dari kinerja algoritma KECA-QC dengan berbagai algoritma optimasi. Visualisasi yang ditampilkan pada bab ini hanyalah visualisasi hasil akhir dari KECA-QC, untuk detail proses berupa plot data berdasarkan koordinat ditampilkan pada Lampiran 7.3 hingga Lampiran 7.17. Lampiran tersebut mulanya akan ditunjukan keadaan dimana persebaran asli dari suatu dataset. Kemudian pada bagian selanjutnya akan dipaparkan visualisasi data yang didapatkan setelah tahap *pre-processing* menggunakan metode KECA dilakukan. Setelah itu, akan dipaparkan visualisasi data yang didapatkan dari data yang didapatkan pada tahap *processing* dengan mengolah data dari hasil proses KECA hingga hasil akhir proses *quantum clustering* menggunakan lima algoritma optimasi yang sebelumnya telah disebutkan. Visualisasi tersebut dilakukan kepada lima buah dataset buatan dengan berbagai bentuk yang masing-masing diberi nama *Cluster in cluster*, *Crescent & Fullmoon*, *Corners*, *Half-kernel*, dan *Spiral*.

Hasil akhir visualisasi data dapat dilihat dari Gambar 5.1 hingga Gambar 5.5 pada halaman berikutnya.

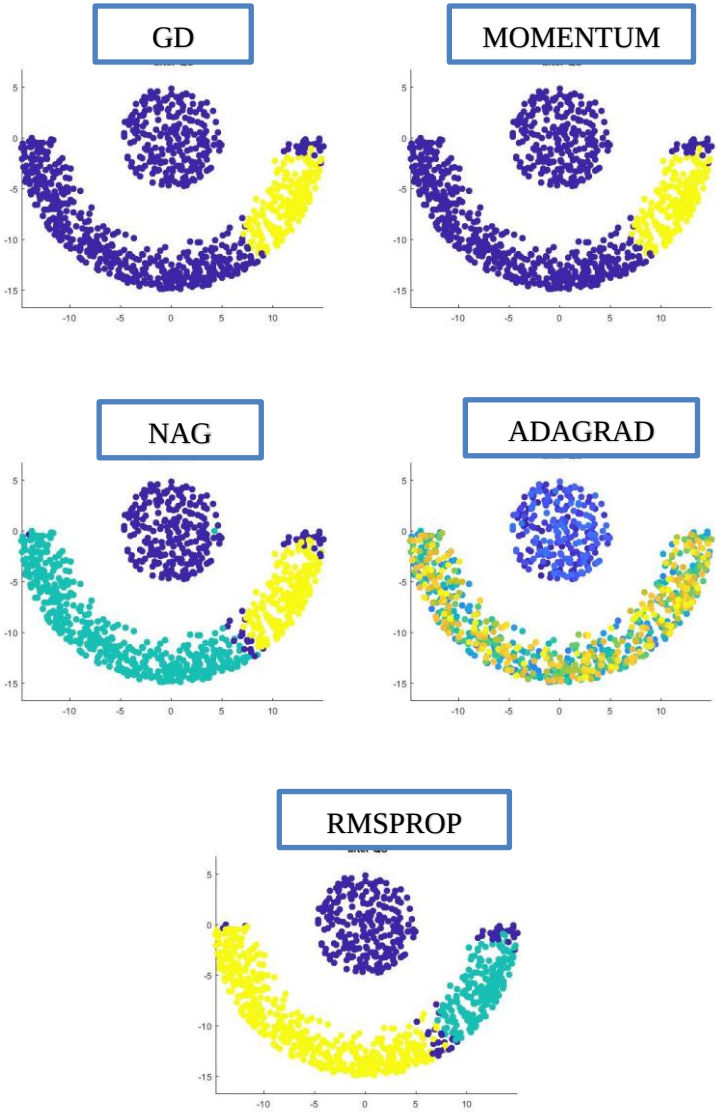
5.4.2. Perhitungan metrik

Sub bab ini akan ditampilkan hasil perhitungan metrik dari lima dataset buatan dan tiga dataset UCI berdasarkan plot *clustering* yang terdapat pada Lampiran 7.18.

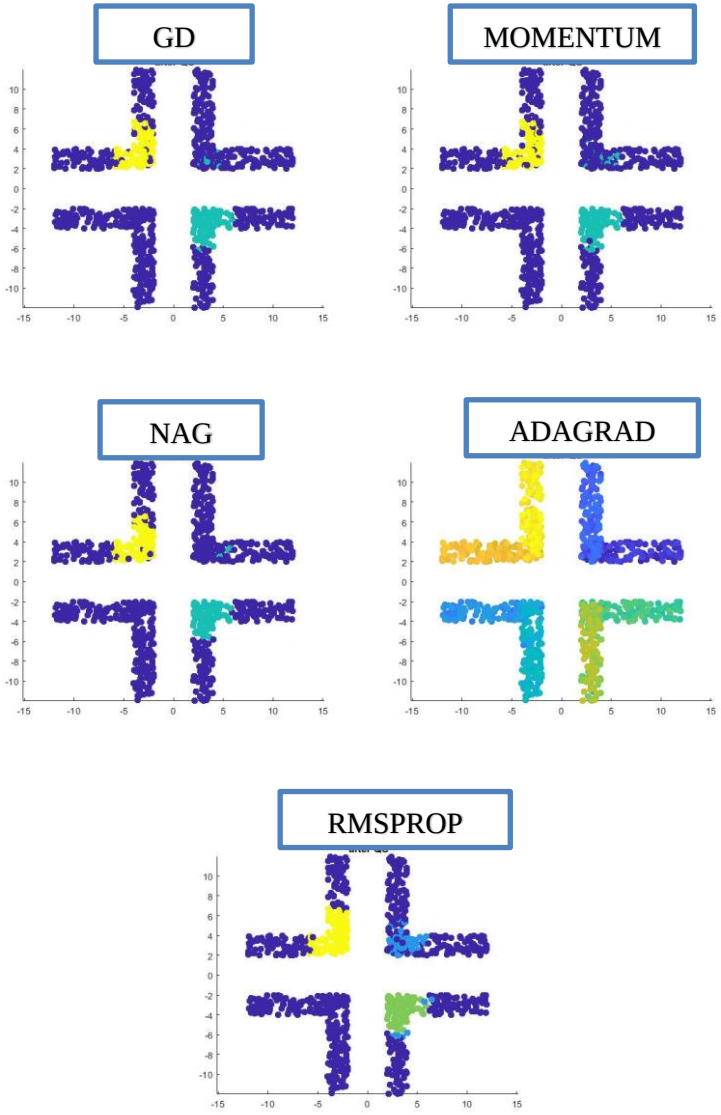
Hasil perhitungan metrik ditampilkan melalui dua tabel yang mana Tabel 5.1 menampilkan hasil perhitungan metrik dari dataset buatan dan Tabel 5.2 menampilkan hasil perhitungan metrik dari dataset UCI. Perhitungan metrik dilakukan dengan tiga macam metode yaitu *Jaccard Measure, Minkowski Measure,* dan *Cluster Accuracy* dalam persen dengan tetap memperhatikan jumlah cluster yang didapat apakah sesuai dengan jumlah cluster yang sesungguhnya ataukah tidak.



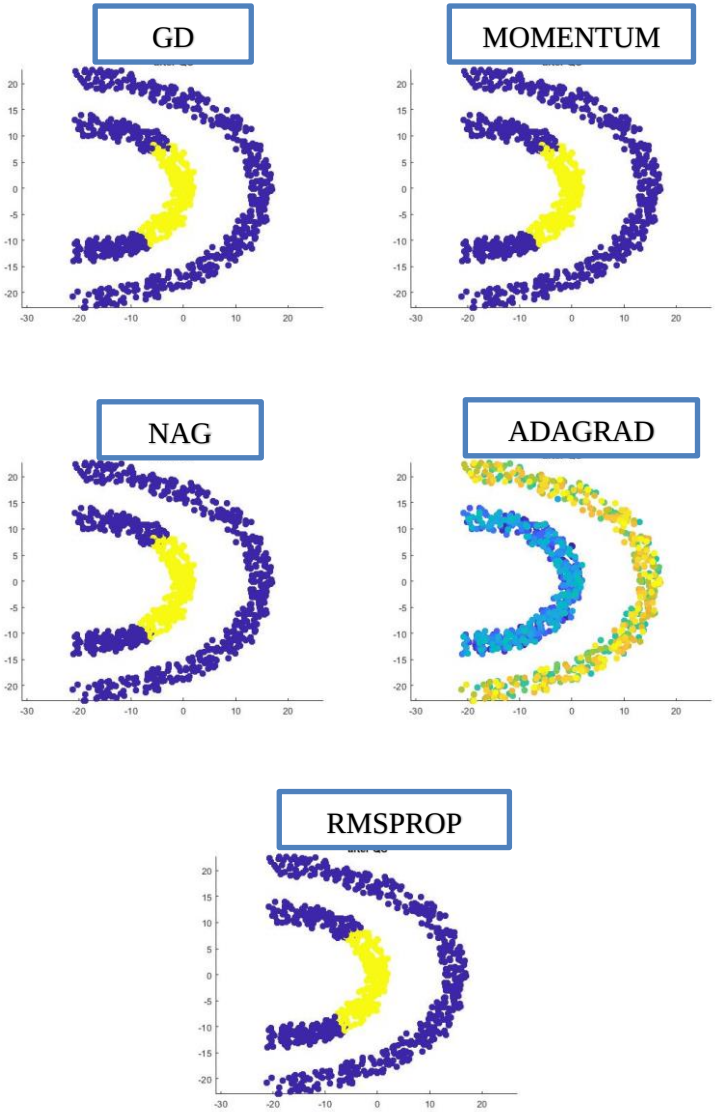
Gambar 5.1 Hasil akhir dataset *Cluster in cluster*



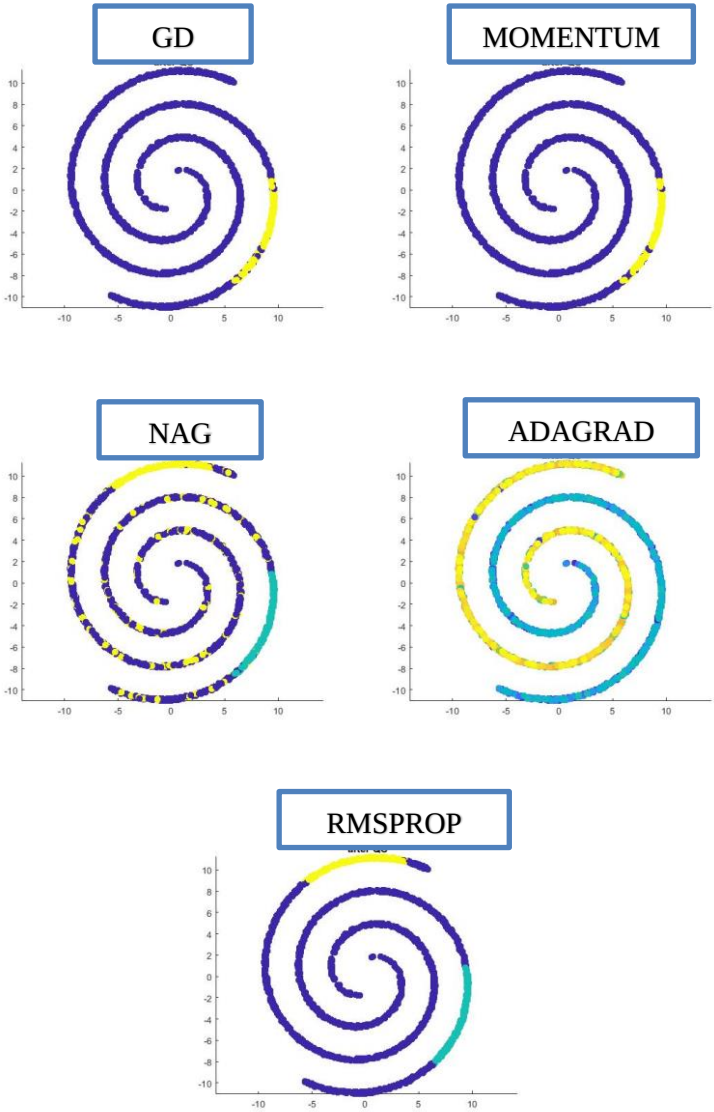
Gambar 5.2 Hasil akhir dataset *Crescent & Fullmoon*



Gambar 5.3 Hasil akhir dataset *Corners*



Gambar 5.4 Hasil akhir dataset *Half-kernel*



Gambar 5.5 Hasil akhir dataset *Spiral*

Tabel 5.2 Hasil Evaluasi KECA-QC pada dataset buatan

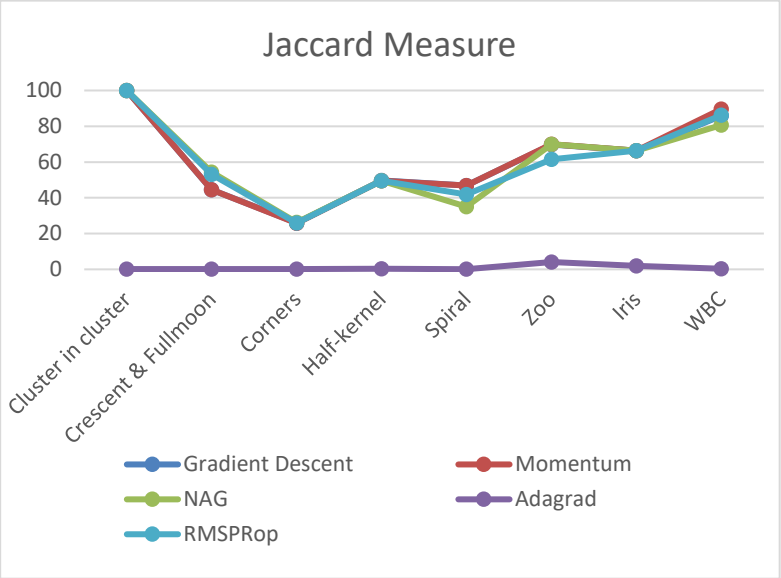
Dataset	Metrik	GD	MOMENTUM	NAG	AdaGrad	RMSprop
Cluster in cluster	JM (%)	**100**	**100**	**100**	0.20	**100**
	MM (%)	**0**	**0**	**0**	99.90	**0**
	CA (%)	**100**	**100**	**100**	100	**100**
	Cluster	**2/2**	**2/2**	**2/2**	1012/2	**2/2**
Crescent & Fullmoon	JM (%)	44.49	44.49	**54.29**	0.16	53.11
	MM (%)	89.16	89.16	**68.89**	99.92	69.94
	CA (%)	70.44	70.44	**96.26**	99.60	95.60
	Cluster	**2/2**	**2/2**	3/2	999/2	3/2
Corners	JM (%)	25.99	25.79	**26.25**	0.20	25.89
	MM (%)	145.6	145.8	145.4	99.90	**136.2**
	CA (%)	33.58	33.58	33.53	99.80	**38.25**
	Cluster	3/4	3/4	3/4	996/4	**4/4**
Half-kernel	JM (%)	**49.68**	49.55	49.61	0.29	49.61
	MM (%)	**87.17**	87.40	87.28	99.88	87.28
	CA (%)	**69.44**	69.22	69.33	87.46	69.33
	Cluster	**2/2**	**2/2**	**2/2**	997/2	**2/2**
Spiral	JM (%)	**46.86**	46.84	35.10	0.10	41.89
	MM (%)	98.42	98.37	**98.00**	99.95	98.22
	CA (%)	55.28	**55.38**	54.32	97.74	51.57
	Cluster	2/2	**2/2**	3/2	1942/2	3/2

Tabel 5.3 Hasil Evaluasi KECA-QC pada dataset UCI

Dataset	Metrik	GD	MOMENTUM	NAG	AdaGrad	RMSprop
Z00	JM (%)	69.70	69.89	**69.97**	4.11	61.48
	MM (%)	57.65	57.30	**57.23**	97.92	66.20
	CA (%)	93.66	**94.03**	**94.03**	100	89.73
	Cluster	9/7	11/7	10/7	101/7	8/7
Iris	JM (%)	**66.40**	**66.40**	**66.40**	2.00	**66.40**
	MM (%)	**64.06**	**64.06**	**64.06**	98.99	**64.06**
	CA (%)	**86.17**	**86.17**	**86.17**	100	**86.17**
	Cluster	**3/3**	**3/3**	**3/3**	150/3	**3/3**

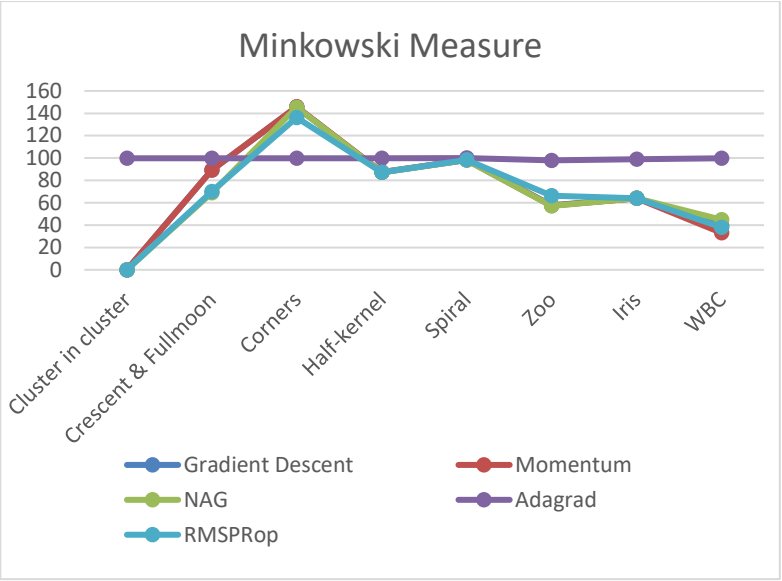
WBC	JM (%)	85.83	**89.69**	80.73	0.27	86.31
	MM (%)	39.26	**33.07**	44.84	99.87	37.98
	CA (%)	95.52	96.91	**97.47**	100	96.91
	Cluster	**2/2**	**2/2**	3/2	683/2	**2/2**

Berdasarkan hasil pada Tabel 5.1 dibuat gambar diagram garis agar proses analisis dapat dilakukan dengan lebih mudah..



Gambar 5.6 Diagram Jaccard Measure

Gambar 5.6 memperlihatkan bahwa hasil Jaccard Measure pada tiap algoritma optimasi tidak berbeda secara signifikan kecuali pada AdaGrad yang selalu mendapat hasil mendekati 0% pada dataset manapun. Hasil *Jaccard Measure* terbaik untuk dataset buatan terdapat pada dataset *Cluster in cluster* dan hasil yang terburuk terdapat pada dataset *Corners*. Dataset UCI secara garis besar mendapatkan hasil *Jaccard Measure* yang cukup baik pada semua dataset dengan hasil yang sangat baik terdapat dapa dataset WBC dan hasil yang paling rendah terdapat pada dataset Iris.

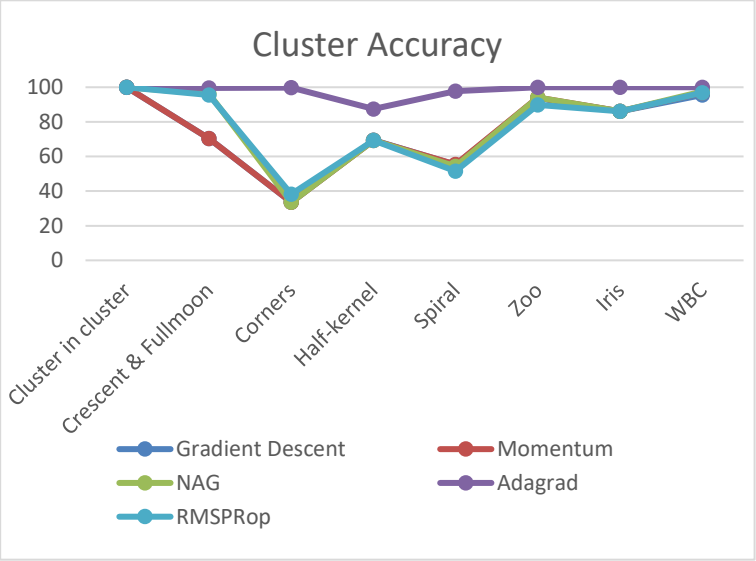


Gambar 5.7 Diagram Minkowski Measure

Perhitungan *Minkowski Measure* Gambar 5.7 menunjukkan nilai yang rendah bilamana hasil dari pengelompokkan data baik. Secara garis besar hasil yang didapat tidak jauh berbeda dengan hasil *Jaccard Measure* dimana pada setiap algoritma optimasi didapat hasil yang tidak jauh berbeda kecuali pada AdaGrad.

Gambar 5.8 memperlihatkan hasil *Cluster Accuracy* tertinggi selalu didapat oleh AdaGrad. Namun nilai yang tinggi tersebut tidak menunjukkan hasil yang baik karena nilai tersebut didapat AdaGrad dengan mengelompokkan data menjadi kemlompok yang yang sangat banyak, hampir sebanyak data yang ada. Sehingga setiap kelompoknya hanya memiliki satu atau beberaoa data saja. Hasil *Cluster Accuracy* lainnya tidak jauh berbeda pada tiap algoritma optimasi. Hanya ada sedikit perbedaan yang cukup mencolok pada dataset *Crescent & Fullmoon* dimana *Gradient Descent* dan *Momentum* mendapat akurasi yang lebih

buruk dibandungkan dengan algoritma lainnya meskipun kedua algoritma tersebut berhasil mengelompokkan data dengan jumlah yang sesuai dengan data sesungguhnya dari *Crescent & Fullmoon.*



Gambar 5.8 Diagram Cluster Accuracy

5.4.3. Perbandingan Metode Clustering

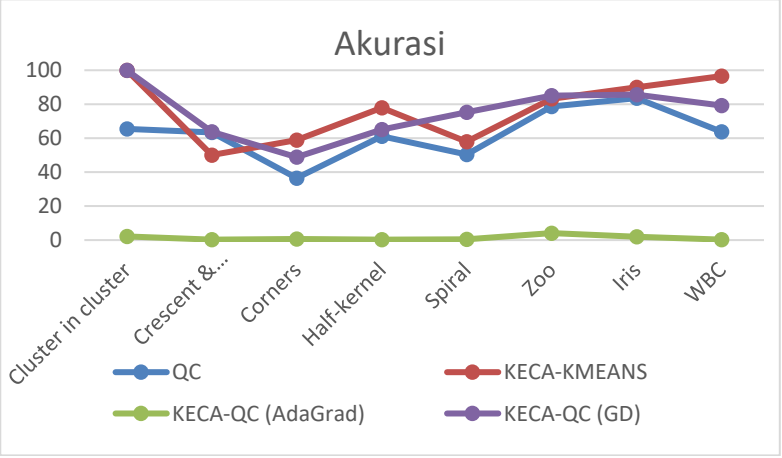
Sub bab sebelumnya telah berisikan hasil uji coba dari KECA-QC yang menunjukkan hasil yang tidak menunjukkan perbedaan yang berarti pada setiap algoritma optimasi kecuali pada AdaGrad yang menunjukkan hasil yang sangat buruk. AdaGrad mendapat hasil yang buruk namun mendapat metrik *Cluster Accuracy* yang baik, oleh karena itu dibawah ini akan dilakukan kembali perhitungan akurasi Adagrad dengan cara perhitungan terhadap *Ground Truth* ditampilkan dalam bentuk persen yang akan dibandingkan dengan *Gradient Descent* yang dianggap sebagai representasi dari algoritma optimasi lainnya karena dari sub bab sebelumnya dapat disimpulkan bahwa *Gradient Descent*

tidak memiliki hasil yang berbeda jauh dengan tambahan algoritma optimasi lainnya. Selain itu, sebagai pelengkap dibawah ini akan ditampilkan juga perbandingan KECA-QC dengan metode *clustering* lainnya.

Tabel 5.4 Perbandingan Akurasi Metode Clustering

Dataset	QC	KECA-KMEANS	KECA-QC (AdaGrad)	KECA-QC (GD)
Cluster in cluster	65.45 %	**100 %**	2.21 %	**100 %**
Crescent & Fullmoon	63.38 %	50.10 %	0.28 %	**63.71 %**
Corners	36.52 %	**58.90 %**	0.72 %	48.83 %
Half-kernel	61.04 %	**77.90 %**	0.28 %	65.00 %
Spiral	50.32 %	57.90 %	0.48 %	**75.21 %**
Zoo	78.74 %	83.17 %	4.11 %	**85.09 %**
Iris	83.57 %	**90.00 %**	2.00 %	85.56 %
WBC	63.79 %	**96.49 %**	0.27 %	79.15 %

Dari Tabel 5.4 diatas dapat diubah kedalam bentuk diagram garis yang akan mempermudah proses analisis yang akan dilakukan dari hasil uji coba perbandingan metode *clustering* tersebut yang dapat dilihat pada Gambar 5.9 dibawah ini.



Gambar 5.9 Diagram Akurasi Berbagai Metode Clustering

Gambar 5.9 menunjukkan bahwa KECA-QC dengan KECA-KMEANS cukup berimbang dimana masing-masing memiliki kelebihan pada dataset tertentu.

5.5. Analisis Hasil Uji Coba

Sub bab ini akan berisi analisis dari apa yang didapat dari uji coba yang dilakukan pada sub bab sebelumnya. Dari hasil uji coba yang telah dilakukan, dapat dilakukan beberapa analisis sebagai berikut :

1. Berdasarkan hasil visualisasi pada Lampiran 7.3 hingga Lampiran 7.17, metode KECA yang menggunakan rata-rata jarak dari K data terdekat sebagai dasar penentuan lebar gaussian dapat memisahkan data dengan cluster yang berbeda dengan cukup baik, namun quantum clustering yang diterapkan pada penelitian ini belum mampu memanfaatkannya dengan maksimal pada beberapa dataset karena quantum clustering sangat dipengaruhi oleh persebaran data yang tentunya setiap dataset memiliki persebaran yang beragam. Meski begitu, hasil yang didapatkan dari metode KECA dan QC ini sudah cukup baik terutama pada beberapa dataset.
2. Berdasarkan hasil pengujian pada kelima algoritma optimasi, hasil yang sangat mencolok terdapat pada algoritma optimasi AdaGrad yang hasilnya sangatlah buruk dimana data dikelompokkan sebanyak data yang ada, sehingga hampir setiap data hanya dikelompokkan sendiri dalam satu kelompok. Hal tersebut terjadi karena memang kelemahan dari AdaGrad adalah akumulasi dari jumlah gradien kuadrat pada penyebut. Dikarenakan pertambahan pada setiap iterasi bernilai positif, jumlah akumulasi terus bertambah pada setiap iterasi. Akhirnya menyebabkan *learning rate* terus menyusut menjadi sangat kecil sehingga tidak bisa lagi didapatkan pengetahuan dari algoritma AdaGrad ini.

3. Berdasarkan persamaan yang dimiliki setiap algoritma optimasi yang diuji, algoritma AdaGrad yang menunjukkan hasil yang sangat buruk memiliki persamaan yang mirip dengan RMSPROP. Berdasarkan hasil percobaan, RMSPROP menunjukkan hasil yang cukup baik. Itu menunjukkan bahwa kekurangan yang terdapat pada AdaGrad berhasil diatasi oleh algoritma RMSPROP. RMSPROP menggunakan rata2 kuadrat gradien sebelumnya dipadukan dengan gradien saat ini, berbeda dengan AdaGrad yang menjumlahkan kuadrat gradien.
4. Berdasarkan hasil pengujian pada algortima optimasi selain AdaGrad, penggunaan rata-rata jarak dari K data terdekat sebagai lebar dari gaussian berhasil mengelompokkan data tidak jauh dari jumlah kelompok yang seharusnya.
5. KECA-QC dan KECA-KMEANS memiliki kelebihan masing-masing bergantung pada datset yang digunakan.

BAB VI
KESIMPULAN DAN SARAN

Bab ini akan menjelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk kedepannya..

6.1. Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. KECA sudah mampu memisahkan data dengan baik namun Quantum Clustering masih kurang bisa mengelompokkan data tersebut dengan baik pada beberapa data.
2. Hasil yang didapat dari lima algoritma optimasi yang dipakai menunjukkan hasil yang tidak berbeda secara signifikan kecuali pada AdaGrad.
3. ADAGRAD selalu menunjukkan hasil yang buruk dengan mengelompokkan data sejumlah data yang ada.
4. Hasil terbaik penggunaan KECA-QC terbaik pada dataset *Cluster in cluster* dengan akurasi sempurna 100% dan hasil terburuk pada dataset *Corners* dengan akurassi 33.58%.

6.2. Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

1. Penggunaan metode lain pada bagian *processing* dapat dicoba untuk memaksimalkan kinerja dari KECA yang sudah cukup baik ataupun dilakukan pencarian untuk memaksimalkan kinerja dari quantum clustering.
2. Sebaiknya metode ini tidak diimplementasikan ataupun dikembangkan pada bahasa pemrograman python karena banyak sekali perhitungan matrix yang dilakukan sehingga menyebabkan waktu komputasi berlangsung lama, akan lebih baik menggunakan bahasa pemrograman MATLAB yang memang diperuntukkan untuk melakukan banyak perhitungan matematis.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] E. IRWANSYAH, "CLUSTERING," Binus University, 9 3 2017. [Online]. Available: https://socs.binus.ac.id/2017/03/09/clustering/. [Diakses 25 3 2018].

[2] Y. Li, Y. Wang, L. Jiao dan Y. Liu, "Quantum clustering using kernel entropy component analysis," *Neurocomputing,* pp. 36-48, 2016.

[3] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," 1994.

[4] S. Ruder, "Tentang Kami: AYLIEN startup," 19 Januari 2016. [Online]. Available: http://sebastianruder.com/optimizing-gradient-descent/. [Diakses 8 Juni 2017].

[5] M. Ghifary, "Optimisasi pada Deep Learning," 11 4 2017. [Online]. Available: https://ghif.github.io/aiml/2017/04/11/optimisasi-pada-deep-learning.html. [Diakses 25 3 2018].

[6] D. Horn dan A. Gottlieb, "The method of quantum clustering," *Proceeding of the Advances in Neural Information Processing Systems (NIPS) 14*, pp. 769-776, 2001.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

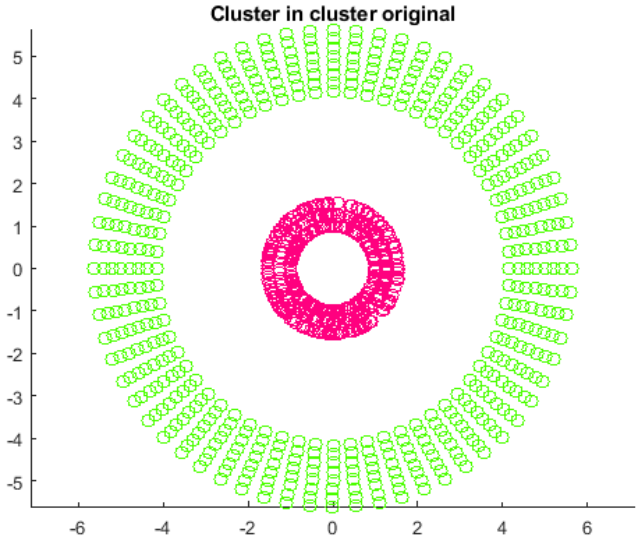
Lampiran 7.1 Pengurutan Eigen Value

```
1.  function [D,E] = sort_eigenvalues(D,E);
2.
3.      d = diag(D);
4.
5.      [d_sorted d_index] = sort(d,'descend');
6.
7.      D = zeros(length(d_sorted));
8.
9.      for i = 1 : length(d_sorted);
10.         D(i,i) = d_sorted(i);
11.     end;
12.
13.     E = E(:,d_index);
14.
15. end
```

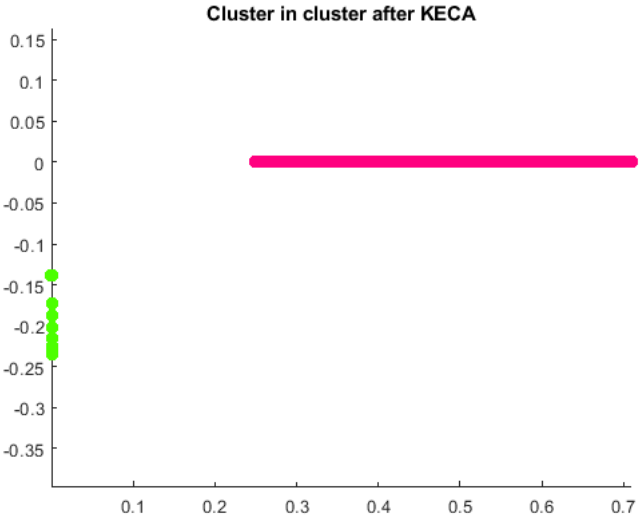
Lampiran 7.2 Pengurutan Eigen Value

```
1.  function
    [sorted_entropy_index,sorted_entropy,entropy] =
    ECA(D,E);
2.
3.      N = size(E,2);
4.
5.      entropy = diag(D)' .* (ones(1,N)*E).^2;
6.      [sorted_entropy,sorted_entropy_index] =
    sort(entropy,'descend');
7.  end
```

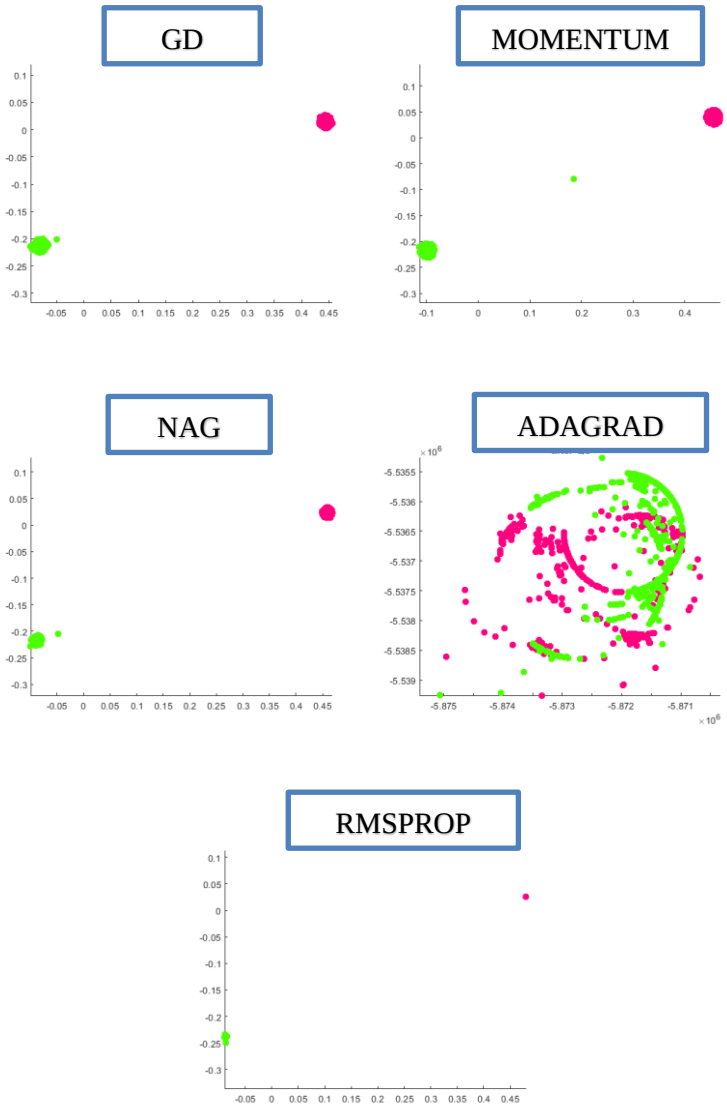
Lampiran 7.3 Dataset buatan *Cluster in cluster* sebelum KECA



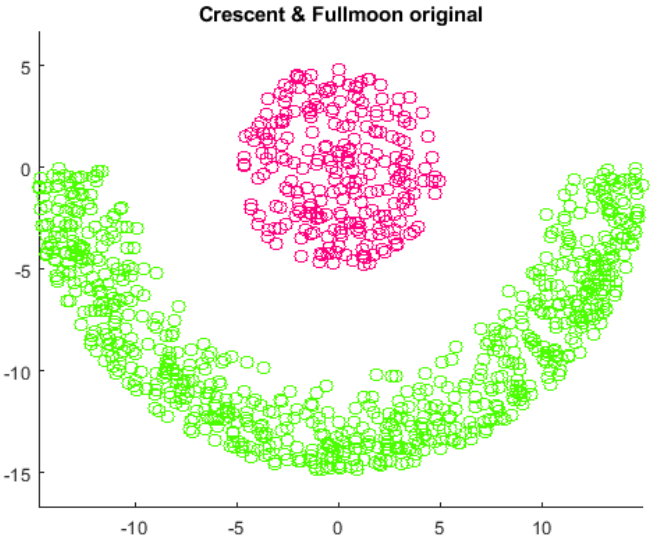
Lampiran 7.4 Dataset buatan *Cluster in cluster* setelah KECA



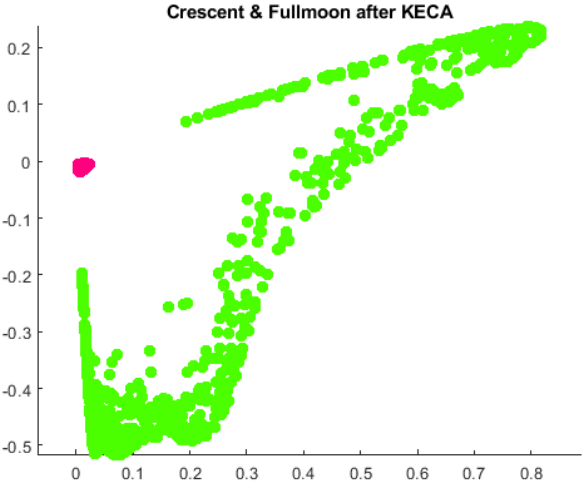
Lampiran 7.5 Dataset buatan *Cluster in cluster* setelah QC



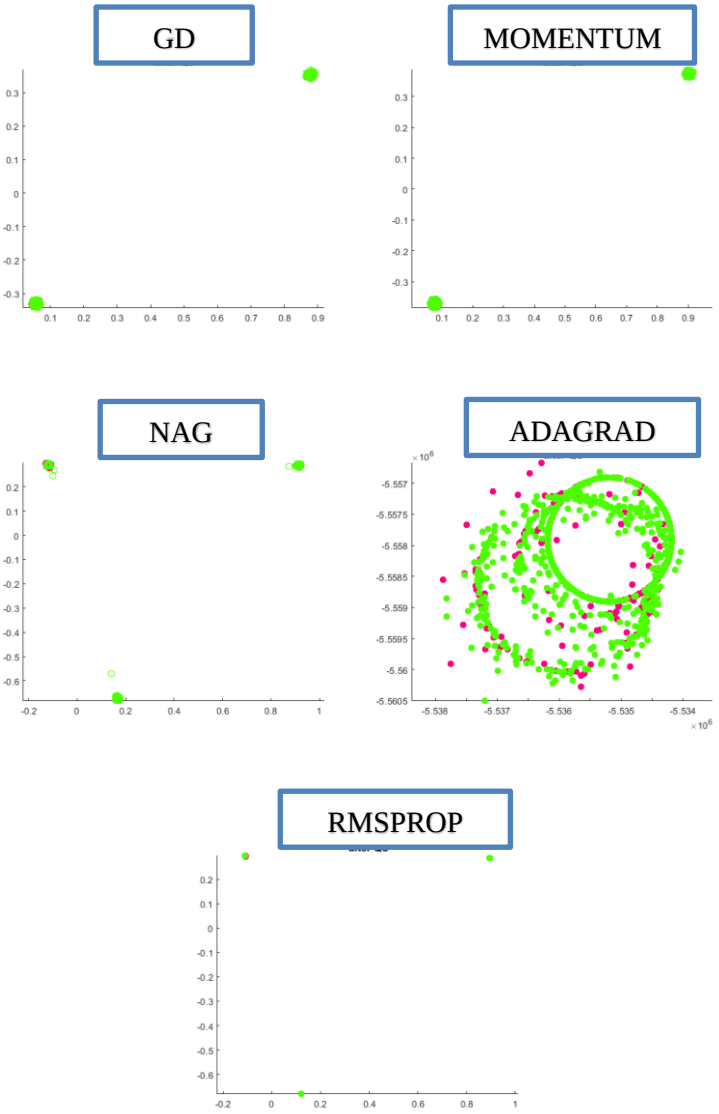
Lampiran 7.6 Data buatan *Crescent & Fullmoon* sebelum KECA



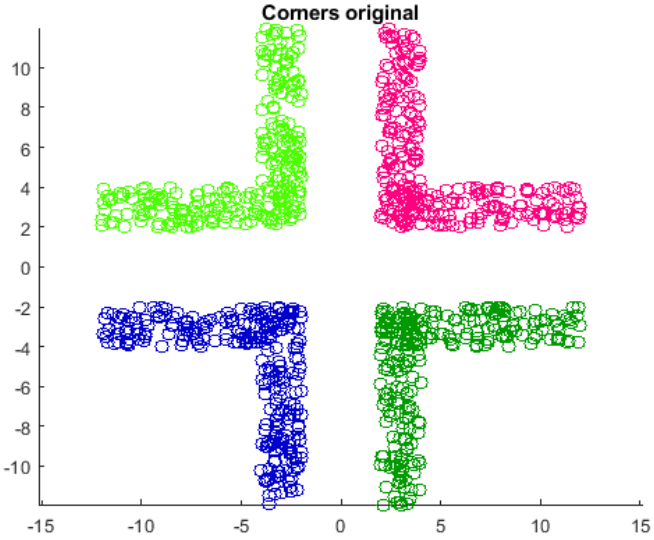
Lampiran 7.7 Data buatan *Crescent & Fullmoon* setelah KECA



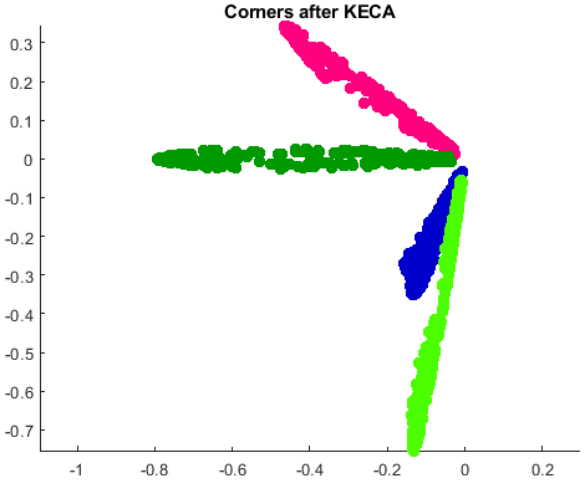
Lampiran 7.8 Dataset buatan *Crescent & Fullmoon* setelah QC



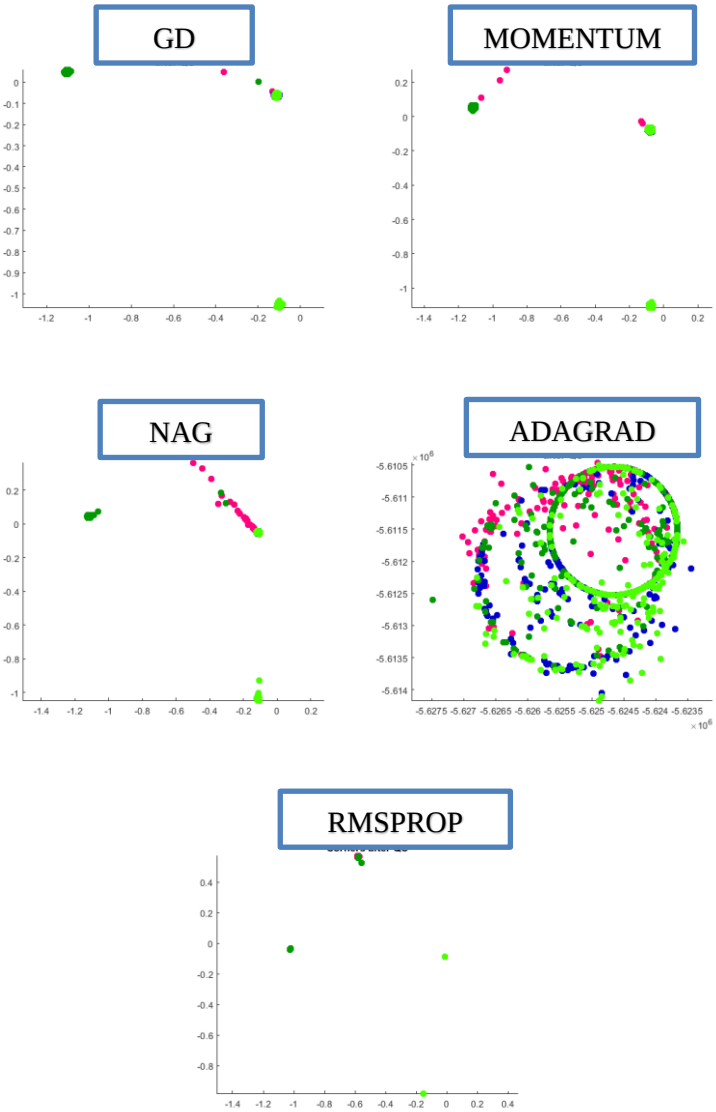
Lampiran 7.9 Data buatan *Corners* sebelum KECA



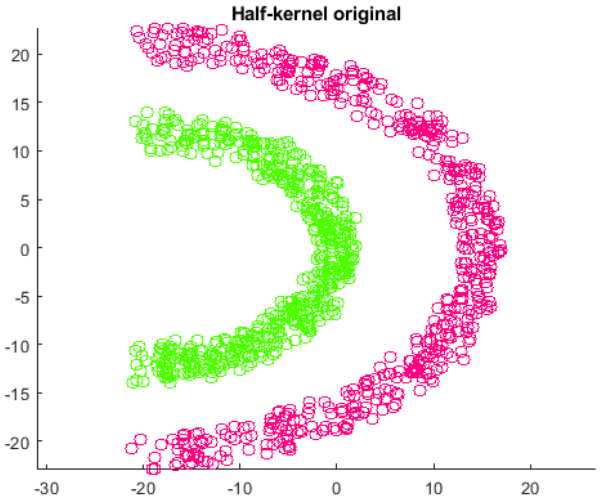
Lampiran 7.10 Data buatan *Corners* setelah KECA



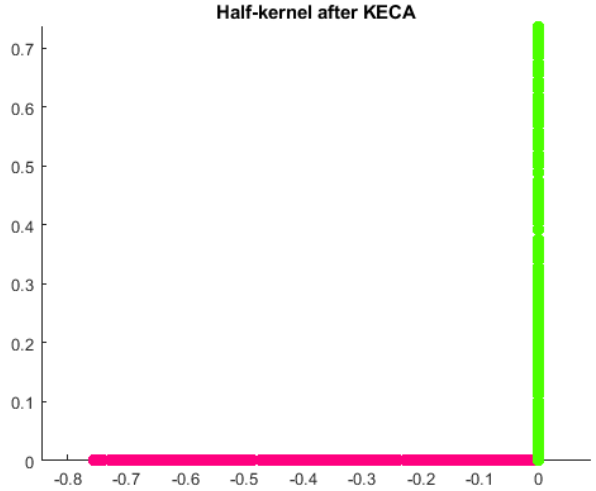
Lampiran 7.11 Dataset buatan *Corners* setelah QC



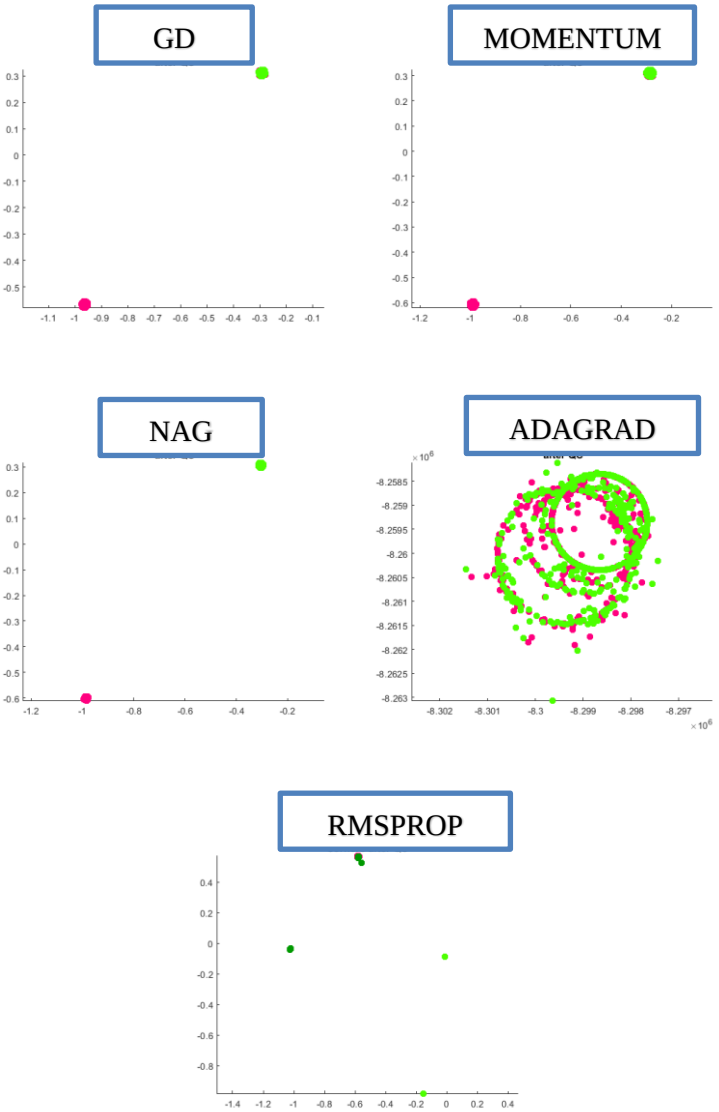
Lampiran 7.12 Dataset buatan *Half-kernel* sebelum KECA



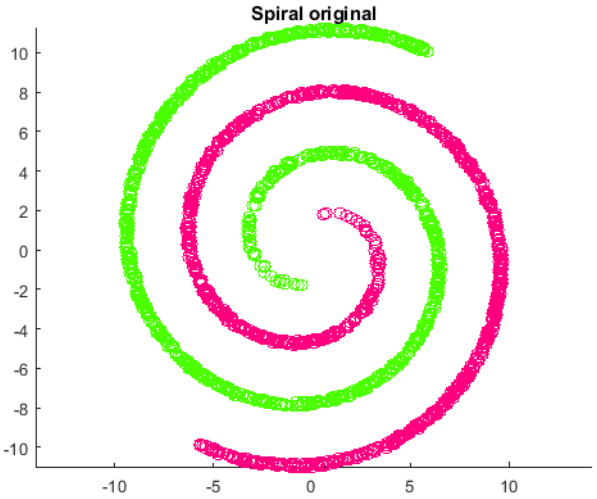
Lampiran 7.13 Dataset buatan *Half-kernel* setelah KECA



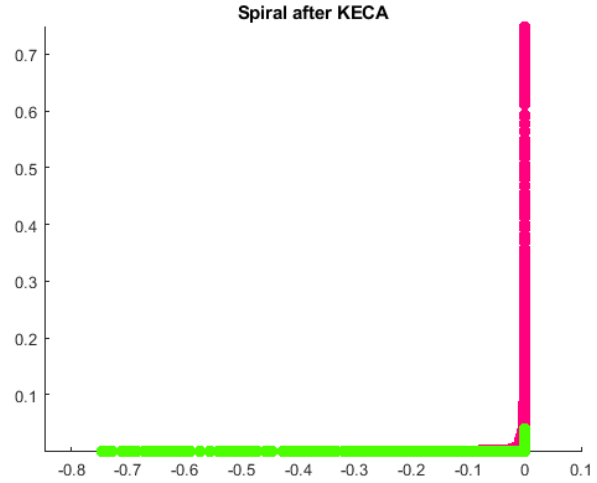
Lampiran 7.14 Dataset buatan *Half-kernel* setelah QC



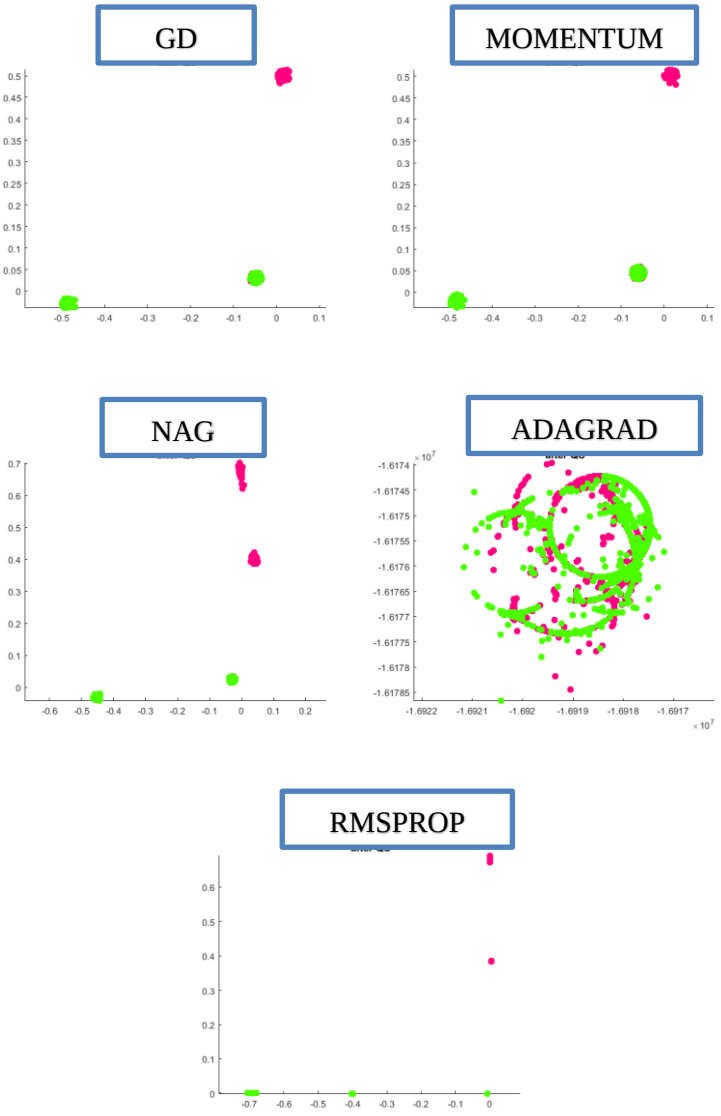
Lampiran 7.15 Dataset buatan *Spiral* sebelum KECA



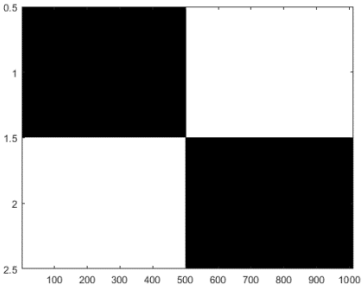
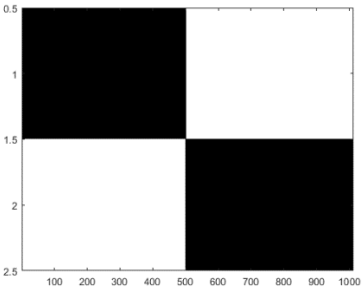
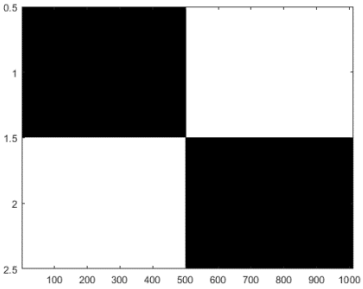
Lampiran 7.16 Dataset buatan *Spiral* setelah KECA

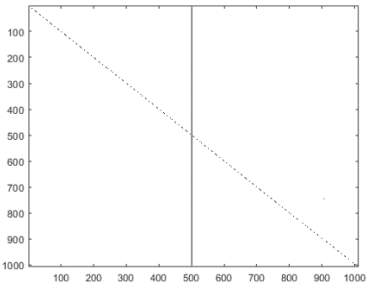
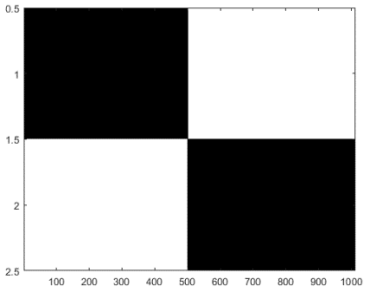
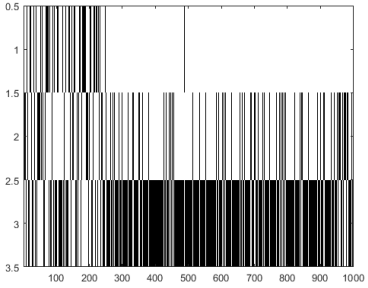


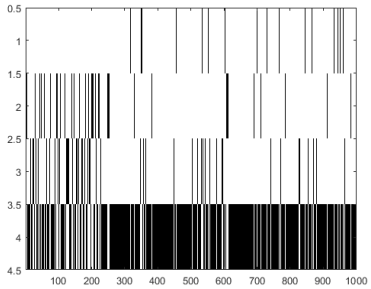
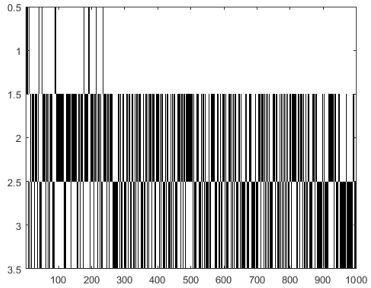
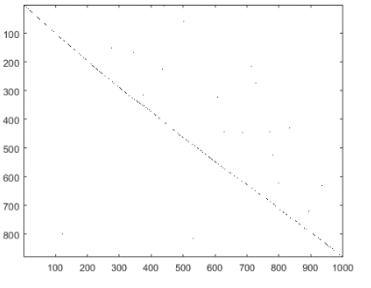
Lampiran 7.17 Dataset buatan *Spiral* setelah QC

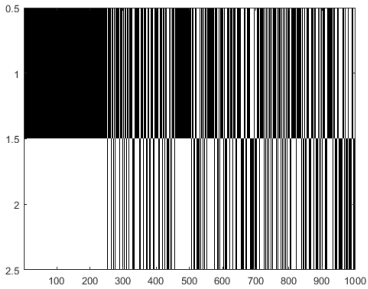
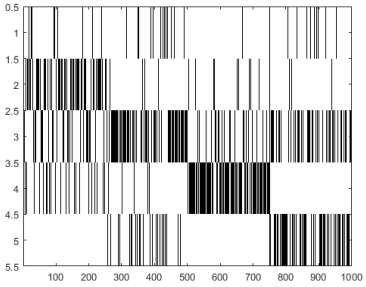
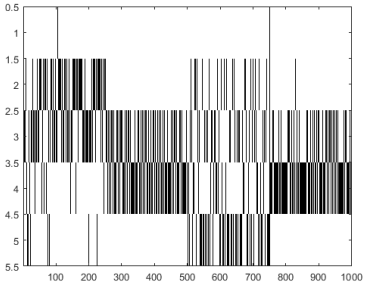


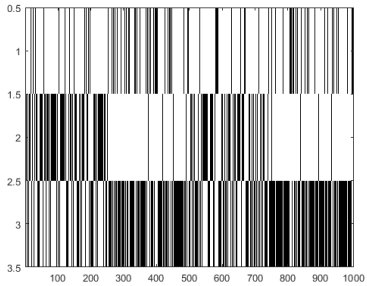
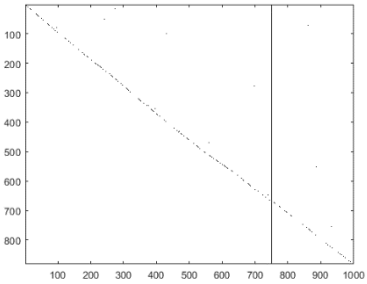
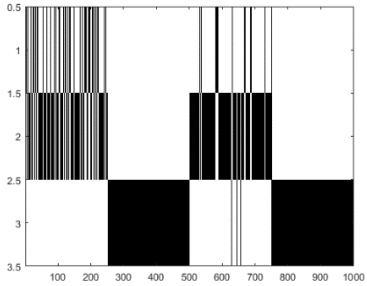
Lampiran 7.18 Hasil Plot *Clustering*

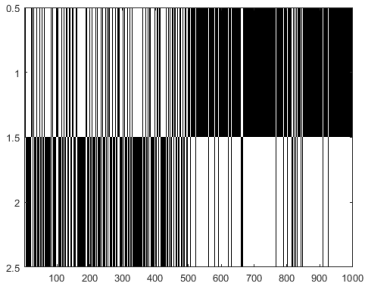
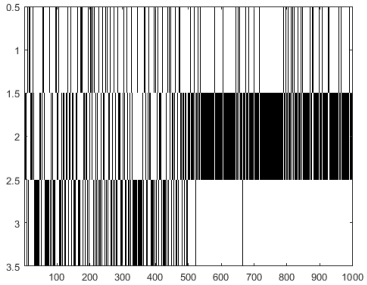
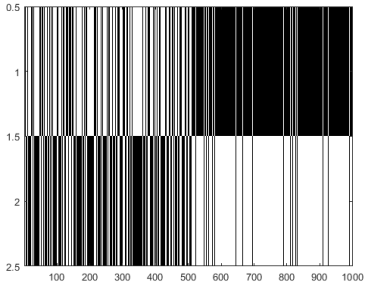
DATASET	OPTIMASI	PLOT *CLUSTERING*
Cluster in cluster	GD	
	MOMENTUM	
	NAG	

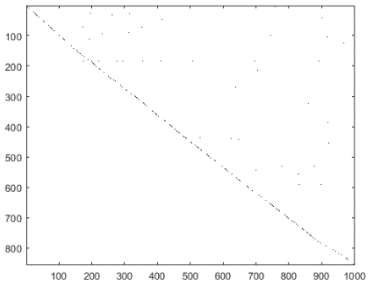
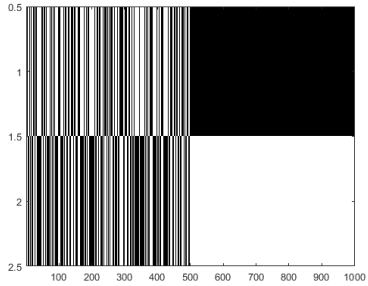
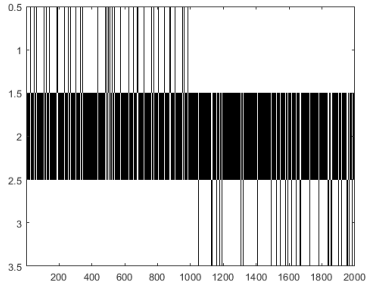
	ADAGRAD	
	RMSPROP	
Crescent & Fullmoon	GD	

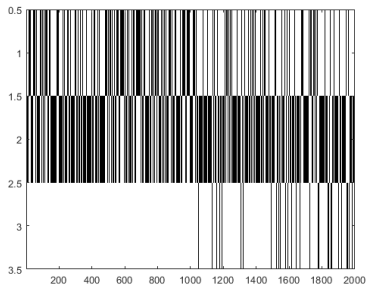
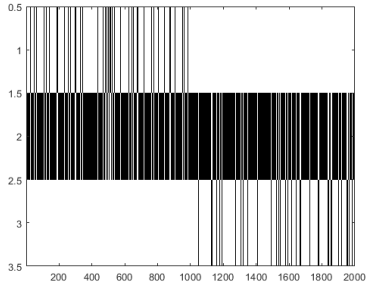
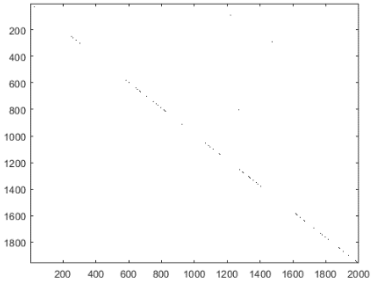
	MOMENTUM	
	NAG	
	ADAGRAD	

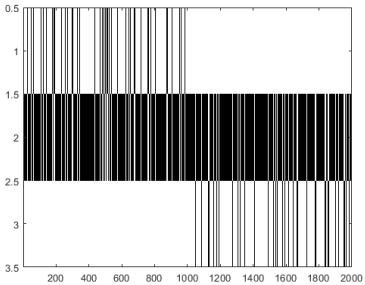
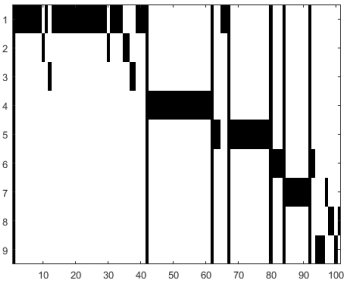
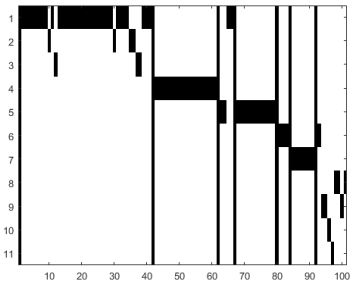
	RMSPROP	
Corners	GD	
	MOMENTUM	

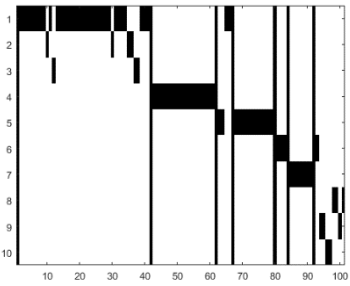
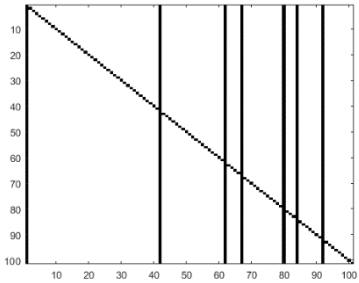
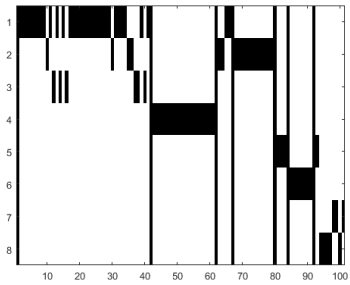
	NAG	
	ADAGRAD	
	RMSPROP	

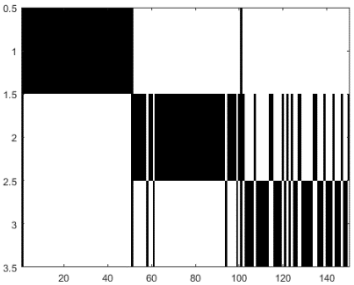
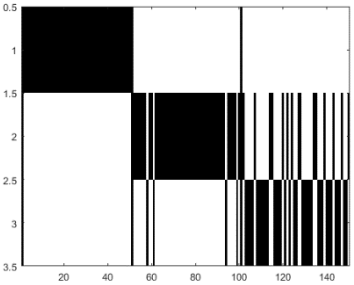
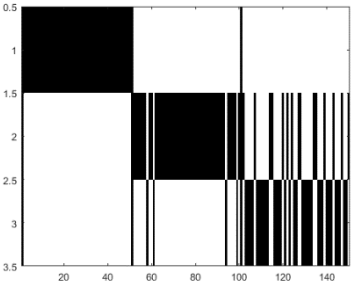
Half-kernel	GD	
	MOMENTUM	
	NAG	

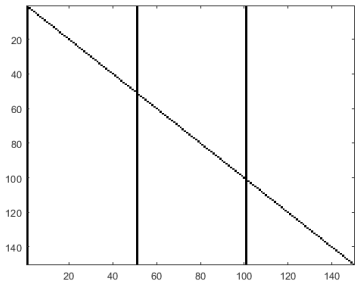
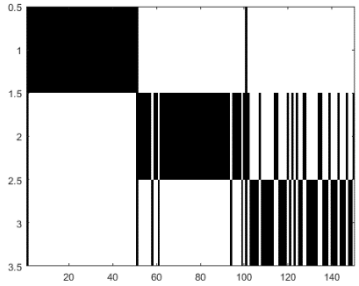
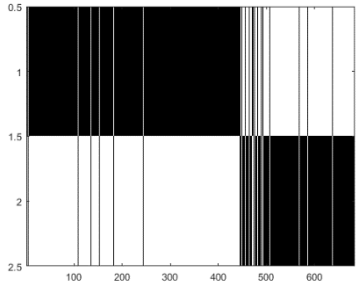
	ADAGRAD	
	RMSPROP	
Spiral	GD	

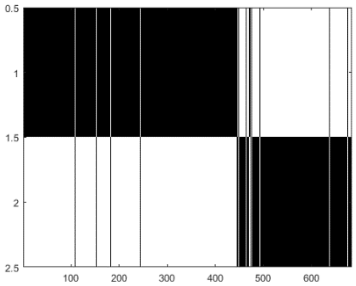
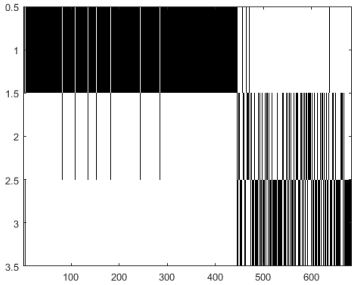
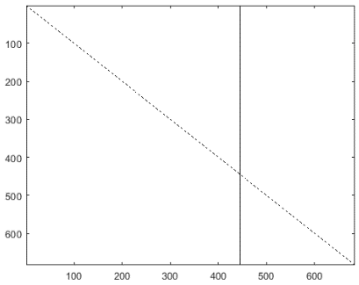
	MOMENTUM	
	NAG	
	ADAGRAD	

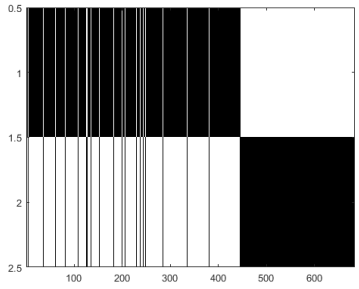
	RMSPROP	
Zoo	GD	
	MOMENTUM	

	NAG	
	ADAGRAD	
	RMSPROP	

Iris	GD	
	MOMENTUM	
	NAG	

	ADAGRAD	
	RMSPROP	
WBC	GD	

	MOMENTUM	
	NAG	
	ADAGRAD	

	RMSPROP	

Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Riansya Pamusti dilahirkan di Bandung 29 April 1996. Penulis adalah anak ketiga dari tiga bersaudara. Penulis menempuh pendidikan di SDPN Setiabudi Bandung (2002-2008), SMP Negeri 5 Bandung (2008-2011), dan SMA Negeri 3 bandung (2011-2014). Kemudian penulis melanjutkan studi di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama menempuh kuliah penulis aktif sebagai anggota Himpunan Mahasiswa Teknik Computer (HMTC) ITS pada departemen dalam negeri,anggota Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi (BEM-FTIf) ITS pada departemen *entrepreneurship*, dan Badan Pengurus Harian (BPH) dari *National Programming Contest* (NPC) Schematics ITS.