ALGORITMA SHARED NEAREST NEIGHBOR BERBASIS DATA SHRINKING

Rifki Fahrial Zainal[1] Arif Djunaidy[2]

[1]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
[2]Jurusan Sistem Informasi , Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Email: rifkifz@cs.its.ac.id, adjunaidy@its.ac.id

Shared Nearest Neighbor (SNN) algorithm constructs a neighbor graph that uses similarity between data points based on amount of nearest neighbor which shared together. Cluster obtained from representative points that are selected from the neighbor graph. The representative point is used to reduce number of clusterization errors, but also reduces accuracy. Data based shrinking SNN algorithm (SSNN) uses the concept of data movement from data shrinking algorithm to increase accuracy of obtained data shrinking. The concept of data movement will strengthen the density of neighbor graph so that the cluster formation process could be done from neighbor graph components which still has a neighbor relationship. Test result shows SSNN algorithm accuracy is 2% until 8% higher than SNN algorithm, because of the termination of relationship between weak data points in the neighbor graph is done slowly in several iteration. However, the computation time required by SSNN algorithm is three times longer than SNN algoritm computational time, because SSNN algorithm constructs neighbor graph in several iteration.

Keywords: *Clusterization algorithm, data shrinking, data mining, shared nearest neighbor*

1 PENDAHULUAN

Klasterisasi berguna untuk menemukan kelompok data sehingga diperoleh data yang lebih mudah dianalisa. Walaupun sudah banyak algoritma klasterisasi yang dikembangkan, tetapi terdapat sebuah permasalahan yang selalu muncul. Permasalahan tersebut disebabkan karena data set yang memiliki dimensi besar. Sehingga tantangan utama dari algoritma klasterisasi selalu sama, yaitu bagaimana cara untuk menemukan klaster dengan ukuran, bentuk, dan kepadatan yang sama, bagaimana cara untuk mengatasi desau, dan bagaimana cara menentukan jumlah klaster.

1.1 Latar Belakang

Beberapa algoritma klasterisasi bergantung pada sebuah fungsi jarak sehingga obyek-obyek akan terletak pada klaster yang sama apabila obyek-obyek tersebut satu sama lain merupakan tetangga terdekat [1, 2]. Tetapi terdapat permasalahan dimensi *(curse of dimensionality)* yang menyatakan efisiensi dan efektifitas algoritma klasterisasi akan berkurang seiring dengan bertambahnya dimensi data.

Algoritma seperti DBSCAN [3], CURE [4], Chameleon [5], dan beberapa algoritma klasterisasi lainnya [6, 7, 8, 9, 10, 11] seringkali digunakan untuk mengatasi permasalahan klasterisasi untuk data dimensi kecil, tetapi data dimensi besar memberikan tantangan yang baru. Dalam data set berdimensi besar, jarak atau kesamaan diantara titik-titik data menjadi semakin sama sehingga mempersulit proses klasterisasi.

Algoritma-algoritma tersebut mengalami permasalahan dimensi karena pengukuran nilai kesamaan dari jarak antara titik data dalam data set. Pendekatan SNN merupakan cara yang terbaik untuk mengatasi permasalahan tersebut [12]. Setelah ketetanggaan terdekat dari semua titik data telah ditentukan, maka nilai kesamaan yang baru diantara titik-titik data ditentukan dari jumlah ketetanggaan yang dimiliki secara bersama-sama. Kekurangan utama dari algoritma SNN tersebut adalah dibutuhkannya sebuah nilai ambang batas untuk menentukan penggabungan atau pemisahan klaster. Bahkan dapat terjadi tidak adanya nilai ambang batas yang sesuai untuk beberapa data set.

Untuk mengatasi kekurangan tersebut, dikembangkan sebuah algoritma SNN berbasis kepadatan [13, 14]. Algoritma ini menggabungkan proses SNN dengan proses pembentukan klaster pada DBSCAN. Graph ketetanggaan yang dibentuk dalam proses SNN digunakan untuk menentukan titik-titik representatif. Algoritma SNN menggunakan proses pemilihan titik representatif tersebut untuk mengurangi kesalahan peletakan titik-titik data dalam klaster yang benar, tetapi proses tersebut juga memberikan sebuah kerugian.

Bila sebuah titik data memiliki nilai kepadatan yang lebih kecil dari nilai ambang batas kepadatan tertentu atau dengan kata lain tidak terpilih sebagai titik representatif, maka titik data tersebut akan diabaikan dalam proses pembentukan klaster. Titik-titik data tersebut akan mengurangi jumlah titik data yang diletakkan dalam klaster yang benar, atau dengan kata lain mengurangi akurasi hasil dari algoritma SNN.

1.2 Tujuan dan Kontribusi

Dalam penelitian ini akan dikembangkan sebuah algoritma klasterisasi baru yang menggabungkan konsep pergerakan data dari algoritma data shrinking [15, 16] ke dalam pembentukan graph ketetanggaan pada algoritma SNN. Pergerakan data ke arah pusat klaster akan memperbesar kepadatan pada graph ketetanggaan sehingga dapat mempermudah proses pembentukan klaster.

Tujuan utama dari penelitian ini adalah melakukan perbaikan algoritma SNN dengan memperbesar kepadatan graph ketetanggaan dari algoritma data shrinking. Penelitian ini tidak akan menggunakan algoritma data shrinking sebagai praproses untuk algoritma SNN berbasis kepadatan, tetapi memasukkan konsep pergerakan data dari algoritma data shrinking ke dalam algoritma SNN. Sehingga penelitian ini diharapkan dapat memberikan sebuah kontribusi baru dalam bentuk perbaikan akurasi algoritma klasterisasi SNN menggunakan konsep pergerakan data dari al-

goritma data shrinking.

1.3 Susunan Makalah

Susunan makalah penelitian ini mengikuti aturan dalam pembahasan berikut. Bagian 2 dan 3 secara berturut-turut akan membahas algoritma yang digunakan sebagai dasar dari penelitian beserta proses yang akan digunakan dalam pengembangan algoritma SNN berbasis data shrinking. Pengembangan algoritma itu sendiri akan dibahas pada bagian 4. Bagian 5 membahas mengenai uji coba yang dilakukan pada algoritma SNN berbasis data shrinking. Kompleksitas dari algoritma [17] akan dibahas pada bagian 6, kemudian dilanjutkan dengan pembahasan mengenai kesimpulan dan pengembangan lebih lanjut pada bagian 7.

2 ALGORITMA DATA SHRINKING

Data shrinking merupakan sebuah teknik pra-proses data yang mengoptimalkan strutur data dengan menggunakan hukum grafitasi. Algoritma ini terdiri dari tiga langkah utama, yaitu penyusutan data, deteksi klaster, dan seleksi klaster. Dalam tahap penyusutan data, titik-titik data akan digerakkan searah dengan gradien kepadatan mensimulasikan hukum grafitasi. Sehingga diperoleh klaster yang padat dan terpisah dengan baik.

Selanjutnya klaster akan dideteksi dengan menemukan komponen cell padat yang saling berhubungan. Kedua tahapan tersebut dilakukan dalam klasterisasi berbasis grid. Kemudian dalam langkah seleksi klaster, setiap klaster yang telah dideteksi akan dievaluasi dalam skala yang berbeda menggunakan pengukuran evaluasi klaster dan klaster terbaik akan dipilih sebagai hasil akhirnya.

2.1 Pra-proses Data Shrinking

Data set dalam proses algoritma data shrinking akan dibagi menjadi ruang-ruang berdasarkan ukuran skala yang telah ditentukan sebelumnya. Pergerakan titik dalam masing-masing ruang dalam data set akan diperlakukan sebagai satu badan yang bergerak sebagai satu kesatuan ke arah pusat data dari ruang tetangganya. Sehingga semua titik dalam berpartisipasi dalam gerakan yang sama. Misalnya data set pada awal iterasi adalah

$$\{X_1^i, X_2^i, \ldots, X_n^i\} \tag{1}$$

dan set dari ruang padat adalah:

$$DenseCellSet^i = \{C_1^i, C_2^i, \ldots, C_m^i\} \tag{2}$$

Diasumsikan masing-masing ruang padat memiliki titik sejumlah $n_1, n_2, \ldots, n_m$ dan data pusatnya adalah $\Phi_1, \Phi_2, \ldots, \Phi_m$. Untuk setiap ruang padat C_j, ruang padat disekitarnya adalah C_{jk}, dengan $k = 1, 2, \ldots, w$. Maka pusat data ruang sekitarnya adalah:

$$\frac{\sum_{k=1}^{w} n_{j_k} \times \Phi_{j_k}}{\sum_{k=1}^{w} n_{j_k}} \tag{3}$$

Pergerakan cell C_j pada iterasi ke-i adalah:

$$Movement(C_j^i) = \begin{cases} \Phi_j^s - \Phi_j & \text{jika } \|\Phi_j^s - \Phi_j\| \geq \\ & T_{mv} \times \frac{1}{k} \text{ dan} \\ & \sum_{k=1}^{w} n_{j_k} > n_j \,, \\ 0 & \text{jika sebaliknya.} \end{cases} \tag{4}$$

Jika jarak antara pusat ruang padat dengan pusat ruang disekitarnya tidak terlalu kecil dan ruang disekitarnya memiliki lebih banyak titik, maka ruang C_j akan dimanipulasi sehingga pusat ruang C_j bergerak ke arah pusat ruang disekitarnya. Bila tidak, maka ruang C_j akan tetap diam. Proses ini akan dilakukan terus-menerus hingga pergerakan rata-rata seluruh titik pada setiap iterasi tidak banyak berubah atau telah melampaui nilai ambang batas iterasi.

2.2 Deteksi dan Evaluasi Klaster

Langkah kedua dalam algoritma data shrinking adalah deteksi klaster. Karena proses penyusutan data menghasilkan klaster individual yang padat dan terpisah dengan baik, maka dapat digunakan berbagai algoritma deteksi klaster. Pada penelitian awal data shrinking, digunakan metode deteksi klaster berbasis skala. Untuk setiap ruang padat, ruang tetangganya dihubungkan dan dibentuk sebuah graph ketetanggaan. Kemudian dihitung nilai kerapatan sebelum dilakukan penyusutan data dari klaster yang dideteksi pada semua skala. Klaster yang memiliki kerapatan lebih dari nilai ambang batas tertentu akan dijadikan sebagai klaster hasil.

3 ALGORITMA SNN

Dalam beberapa kasus, teknik klasterisasi yang bergantung pada pendekatan standar ke arah kesamaan dan kepadatan tidak menghasilkan hasil klasterisasi yang diinginkan. Pendekatan SNN yang dikembangkan oleh Jarvis dan Patrick merupakan pendekatan tidak langsung terdapat kesamaan berdasarkan prinsip berikut:

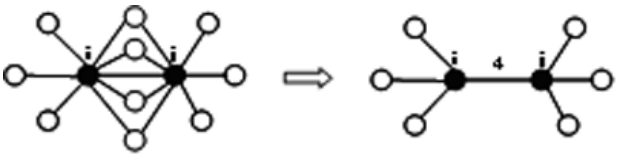
> Jika dua titik memiliki kesamaan terhadap titik yang sama banyak, maka kedua titik tersebut memiliki kesamaan satu sama lain, bahkan jika pengukuran kesamaan tidak menunjukkan kesamaan tersebut.

Ide kunci dari algoritma ini adalah mengambil jumlah dari titik-titik data untuk menentukan pengukuran kesamaan. Kesamaan dalam algoritma SNN didasarkan jumlah tetangga yang dimiliki secara bersama-sama selama kedua obyek terdapat dalam daftar tetangga terdekat masing-masing seperti diperlihatkan pada Gambar 1.

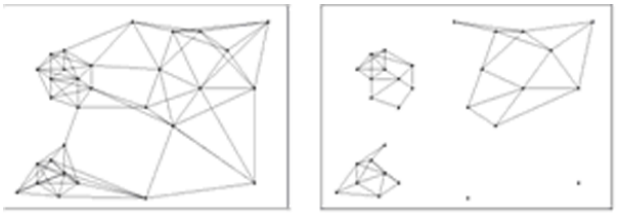
Proses kesamaan SNN sangat berguna karena dapat mengatasi beberapa permasalahan yang ditimbulkan dengan perhitungan kesamaan secara langsung. Karena mengikutsertakan isi dari sebuah obyek dengan menggunakan jumlah tetangga terdekat yang dimiliki secara bersama, SNN dapat mengatasi situasi yang mana sebuah obyek dekat dengan obyek lainnya yang berbeda kelas.

Algoritma ini bekerja baik untuk data berdimensi besar dan secara khusus bekerja baik dalam menemukan kluster padat. Tetapi klasterisasi SNN mendefinisikan klaster sebagai komponen-komponen graph ketetanggaan yang saling berhubungan. Sehingga pembagian klaster sangat bergantung pada sebuah hubungan antara obyek.

Untuk mengatasi permasalahan dari algoritma SNN dasar tersebut, maka diciptakan algoritma SNN berbasis kepadatan. Algoritma SNN ini mengaplikasikan titik representatif CURE dan DBSCAN dalam proses untuk memperoleh klaster. Sehingga efek desau dapat dikurangi dengan



Gambar 1: Komputasi Kesamaan dalam SNN



Gambar 2: Graph Ketetanggaan SNN

menggunakan titik reperesentatif. Algoritma SNN ini terdiri dari langkah:

1. Hitung nilai kesamaan dari data set
2. Bentuk daftar k tetangga terdekat masing-masing titik data untuk k data
3. Bentuk graph ketetanggaan dari hasil daftar k tetangga terdekat
4. Temukan kepadatan untuk setiap data
5. Temukan titik-titik representatif
6. Bentuk klaster dari titik-titik representatif tersebut

Pertama-tama, algoritma SNN menghitung nilai kesamaan masing-masing titik data. Nilai kesamaan tersebut kemudian diurutkan secara menanjak, sehingga diperoleh daftar k tetangga terdekat masing-masing titik data dengan tetangga terdekat terletak pada awal daftar. Daftar k tetangga terdekat tersebut hanya dibuat untuk k data saja untuk menghemat ruang memori yang dibutuhkan.

Kemudian graph ketetanggaan dapat dibentuk dari daftar k tetangga terdekat tersebut seperti ditunjukkan pada Gambar 2. Titik data digunakan sebagai *node* dalam graph ketetanggaan dengan bobot hubungan ketetanggaan sebagai *link* antar *node*. Jumlah tetangga terdekat yang dimiliki secara bersama-sama digunakan sebagai nilai bobot hubungan ketetanggaan tersebut. Misalkan terdapat titik data p dan q dengan tetangga terdekat masing-masing adalah $NN(p)$ dan $NN(q)$, maka nilai bobot hubungan ketetanggaan antara titik data p dan q adalah:

$$bobot(p,q) = jumlah(NN(p) \cap NN(q)) \quad (5)$$

Hasil graph ketetanggaan tersebut digunakan dalam pembentukan klaster pada tahap akhir klasterisasi. Algoritma SNN membentuk klaster menggunakan titik-titik representatif yang ditentukan berdasarkan nilai kepadatan masing-masing titik data. Nilai kepadatan dihitung dari jumlah bobot hubungan ketetanggaan yang bernilai lebih dari nilai ambang batas ketetanggaan.

Titik-titik data dengan nilai kepadatan yang lebih besar dari nilai ambang batas kepadatan akan ditentukan sebagai titik-titik representatif. Titik representatif yang saling berhubungan dan bobot hubungan ketetanggaannya lebih besar dari nilai ambang batas ketetanggaan akan digabungkan dalam klaster yang sama.

Titik-titik data yang tidak terpilih sebagai titik representatif akan diabaikan dalam proses pembentukan klaster. Algoritma SNN mengurangi jumlah klasterisasi yang salah dengan mengabaikan titik-titik data yang meragukan posisinya dalam sebuah klaster. Tetapi proses tersebut menyebabkan berkurangnya jumlah titik data yang mungkin dapat diklasterisasi dengan benar, sehingga akurasi dari algoritma SNN menjadi tidak optimal.

4 ALGORITMA SSNN

Akurasi klaster yang diperoleh algoritma SNN dapat diperbaiki dengan memperbesar kepadatan graph ketetanggaan yang dibentuk dalam algoritma SNN. Salah satu langkah untuk melakukan penguatan kepadatan tersebut adalah dengan menggunakan konsep pergerakan data ke arah pusat klaster dari algoritma data shrinking. Karena algoritma perbaikan ini akan membuat titik-titik data seakan-akan menyusut ke arah pusat klaster, maka algoritma ini diberi nama algoritma SNN berbasis data shrinking atau *Shrinking based Shared Nearest Neighbor* (SSNN).

Apabila titik-titik data dalam graph ketetanggaan digerakkan ke arah pusat klaster, maka bobot hubungan ketetanggaan untuk titik-titik data dalam klaster yang sama akan menjadi semakin besar dan bobot hubungan ketetanggaan untuk titik-titik data dalam klaster yang berbeda akan menjadi semakin kecil. Untuk dapat menerapkan konsep tersebut, maka graph ketetanggaan akan dibentuk dalam beberapa iterasi. Jumlah k tetangga terdekat yang digunakan untuk pembentukan graph ketetanggaan semakin besar dalam setiap iterasinya. Secara keseluruhan, algoritma SSNN terdiri dari langkah:

1. Hitung nilai kesamaan dari data set
2. Bentuk daftar k tetangga terdekat masing-masing titik data
3. Bentuk graph ketetanggaan dari hasil daftar k tetangga terdekat
4. Hitung nilai kedekatan dan putuskan bobot hubungan ketetanggaan yang lebih kecil dari nilai kedekatan tersebut
5. Ulangi langkah 3 dan 4 hingga nilai kedekatan lebih besar dari nilai ambang batas kedekatan
6. Bentuk klaster dari komponen graph ketetanggaan yang masih memiliki bobot hubungan ketetanggaan

Langkah 1 hingga 3 masih sama seperti algoritma SNN. Tetapi pada langkah 2 daftar k tetangga terdekat dibentuk untuk keseluruhan titik data dalam data set karena dibutuhkan untuk pembentukan graph ketetanggaan dalam beberapa iterasi. Langkah 3 dan 4 dilakukan dalam beberapa iterasi untuk menerapkan konsep pergerakan titik data ke arah pusat klaster.

Algoritma SSNN memasukkan urutan tetangga terdekat dalam daftar k tetangga terdekat dalam perhitungan bobot hubungan ketetanggaan, sehingga nilai bobot hubungan ketetanggaan akan memiliki nilai yang lebih akurat. Misalnya terdapat dua titik data i dan j, maka bobot hubungan ketetanggaan dari i dan j adalah: Dalam persamaan (6), k adalah ukuran dari daftar tetangga terdekat, m dan n adalah posisi dari tetangga terdekat yang terdapat di dalam daftar tetangga terdekat i dan j. Nilai k tersebut akan bertambah besar dalam setiap iterasinya sesuai dengan parameter *Move Points* (MP) untuk dapat mengatur perubahan graph ketetanggaan. Nilai kepadatan masing-masing titik data dihitung berdasarkan total bobot hubungan ketetanggaan yang dimiliki masing-masing titik data.

Titik data dalam klaster yang sama memiliki nilai kepadatan yang berdekatan. Algoritma SSNN menggunakan kondisi tersebut sebagai sebuah nilai kedekatan untuk menentukan bobot hubungan ketetanggaan yang akan diputus dan diabaikan dalam proses pembentukan graph ketetanggaan pada iterasi berikutnya. Dengan kata lain, bobot hubungan ketetanggaan yang tidak dekat dengan nilai kepadatannya akan diputus dan diabaikan.

Proses perhitungan nilai kedekatan dalam setiap iterasi tersebut berfungsi untuk mengatasi kekurangan dari algoritma SNN dasar yang terlalu bergantung pada sebuah nilai ambang batas bobot hubungan ketetanggaan. Proses iterasi akan dihentikan apabila nilai kedekatan telah mencapai sebuah nilai ambang batas kedekatan atau besar k tetangga terdekat yang digunakan sudah lebih besar dari jumlah data dalam data set. Pembatasan tersebut dilakukan untuk mencegah terjadinya perpecahan klaster.

Graph ketetanggaan yang diperoleh pada akhir iterasi akan memiliki kepadatan yang kuat sehingga tidak perlu dilakukan proses penentuan titik-titik representatif. Klaster dapat langsung dibentuk dari komponen graph ketetanggaan yang masih memiliki bobot hubungan ketetanggaan. Titik-titik data yang masih berhubungan akan dimasukkan dalam klaster yang sama, sehingga proses pembentukan klaster dapat dikerjakan dalam waktu komputasi yang lebih singkat.

5 UJI COBA

Untuk pengujian algoritma SSNN, digunakan lima buah data set dengan variasi total data dan dimensi yang diperoleh dari UCI *Machine Learning Repository*.

5.1 Hasil Uji Coba

Data set yang pertama, data tumbuhan iris, terdiri dari informasi ukuran bagian-bagian tumbuhan iris. Data ini memiliki 150 data dan 4 atribut, terbagi menjadi 3 klaster dan masing-masing terdiri dari 50 data. Data set Iris memiliki distribusi yang merata dengan posisi sebuah klaster yang terpisah dengan baik sedangkan 2 klaster sisanya saling berdekatan satu sama lainnya.

Data set Iris diuji dengan algoritma SSNN menggunakan 10 variasi parameter nilai k tetangga terdekat awal dengan set nilai {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, 10 variasi nilai ambang batas pergerakan data (MP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai ambang batas kedekatan ketetanggaan (CP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Untuk menghemat tempat, maka dalam makalah ini hanya ditampilkan hasil klasterisasi terbaik ketika pasangan parameter k tetangga terdekat, MP, dan CP bernilai (70, 0.9, 0.6 - 1) yang ditunjukkan pada Tabel 1.

Untuk pengujian dengan SNN, digunakan 10 variasi parameter nilai k tetangga terdekat dengan set nilai {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, 10 variasi nilai *topic* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai *merge* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik diperoleh ketika pasangan parameter bernilai (30, 0.8, 0.5) yang ditunjukkan pada Tabel 2.

Untuk data set yang kedua, akan digunakan data pengenalan buah anggur. Data ini merupakan hasil analisis kimia dari anggur yang tumbuh di tempat yang sama di Itali tetapi dikembangkan pada daerah yang berbeda. Dalam data ini terdapat 178 data, 13 atribut dan 3 buah klaster. Data set wine memiliki distribusi data yang tidak merata. Satu klaster memiliki 71 data sedangkan dua klaster sisanya hanya terdapat 48 dan 59 data.

Data set Wine diuji dengan algoritma SSNN menggunakan 10 variasi parameter nilai k tetangga terdekat awal dengan set nilai {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, 10 variasi nilai ambang batas pergerakan data (MP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai ambang batas kedekatan ketetanggaan (CP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik ketika pasangan parameter k tetangga terdekat, MP, dan CP bernilai (50, 0.1, 0.6 - 1) yang ditunjukkan pada Tabel 3.

Untuk pengujian dengan SNN, digunakan 10 variasi parameter nilai k tetangga terdekat dengan set nilai {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, 10 variasi nilai *topic* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai *merge* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Tabel 4 memperlihatkan hasil klasterisasi terbaik saat parameter bernilai (40, 0.8, 0.3).

Tabel 1: Hasil Pengujian Data Set Iris Menggunakan SSNN

	Setosa	Versicolor	Virginica
Total Data	50	50	50
Data Benar	50	40	39
Data Salah	0	10	11
Data Hilang	0	0	0
Akurasi (%)	100,00	80,00	78,00
Rerata Akurasi (%)		86,00	
Rerata Memori (kBytes)		620	
Waktu (det)		2,243	

Data set berikutnya adalah data set Optidigits atau digit pengenalan tulisan tangan berbasis penglihatan (*Optical Recognition of Handwritten Digits*). Secara keseluruhan data set ini memiliki 1797 data. Data set ini memiliki 64 atribut dan 10 klaster. Data set Optidigits memiliki distribusi data yang merata dengan jumlah data pada masing-masing klaster berkisar antara 177 hingga 182 data.

Data set Optdigits diuji dengan algoritma SSNN menggunakan 10 variasi parameter nilai k tetangga terdekat awal

Tabel 2: Hasil Pengujian Data Set Iris Menggunakan SNN

	Setosa	Versicolor	Virginica
Total Data	50	50	50
Data Benar	50	39	38
Data Salah	0	3	1
Data Hilang	0	8	11
Akurasi (%)	100,00	78,00	76,00
Rerata Akurasi (%)		84,67	
Rerata Memori (kBytes)		510	
Waktu (det)		5,529	

Tabel 3: Hasil Pengujian Data Set Wine dengan SSNN

	1	2	3
Total Data	59	71	48
Data Benar	57	45	30
Data Salah	2	26	18
Data Hilang	0	0	0
Akurasi (%)	96,61	63,38	62,50
Rerata Akurasi (%)		74,16	
Rerata Memori (kBytes)		898	
Waktu (det)		1,422	

dengan set nilai {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, 10 variasi nilai ambang batas pergerakan data (MP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai ambang batas kedekatan ketetanggaan (CP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik ketika pasangan parameter k tetangga terdekat, MP, dan CP bernilai (60.1, 0.6 - 1).

Pengujian dengan SNN menggunakan 10 variasi parameter nilai k tetangga terdekat dengan set nilai {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, 10 variasi nilai *topic* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai *merge* dengan set nilai merge. Hasil klasterisasi terbaik diperoleh ketika pasangan parameter bernilai (70, 0.5, 0.5). Hasil kedua algoritma tersebut ditunjukkan pada Tabel 5.

Data set yang keempat adalah data set Pendigits, atau digit pengenalan tulisan tangan berbasis pena (*Pen-Based Recognition of Handwritten Digits*). Data set ini terdiri dari dua sub data yaitu sub data untuk pelatihan pengenalan tulisan tangan dan sub data untuk pengujian pengenalan tulisan tangan. Untuk penelitian ini digunakan gabungan keduanya sebanyak 10.992 data. Data set Pendigits terdiri dari 16 atribut dan 10 klaster hasil. Data set ini memiliki distribusi data yang merata dengan jumlah data pada masing-masing klaster berkisar antara 1055 data hingga 1144 data.

Data set Pendigits diuji dengan algoritma SSNN menggunakan 5 variasi parameter nilai k tetangga terdekat awal dengan set nilai {300, 400, 500, 600, 700}, 10 variasi nilai ambang batas pergerakan data (MP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai ambang batas kedekatan ketetanggaan (CP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik ketika pasangan parameter k tetangga terdekat, MP, dan CP bernilai (700, 0.3, 0.5 - 1).

Pengujian dengan SNN menggunakan 5 variasi parameter nilai k tetangga terdekat dengan set nilai {300, 400, 500, 600, 700}, 10 variasi nilai *topic* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai *merge* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik diperoleh ketika pasangan parameter k tetangga terdekat, *topic*, dan *merge* bernilai (300, 0.5, 0.7). Hasil kedua algoritma tersebut ditunjukkan pada Tabel 6.

Tabel 4: Hasil Pengujian Data Set Wine dengan SNN

	1	2	3
Total Data	59	71	48
Data Benar	49	45	35
Data Salah	9	26	12
Data Hilang	1	0	1
Akurasi (%)	85,05	63,38	72,92
Rerata Akurasi (%)		73,12	
Rerata Memori (kBytes)		883	
Waktu (det)		8,125	

Tabel 5: Hasil Pengujian Data Set Optdigits

	Algoritma SSNN	Algoritma SNN
Rerata Akurasi (%)	79,80	78,96
Rerata Memori (kBytes)	41686	13913
Waktu (det)	286,592	443,472

Data yang terakhir adalah data pengenalan 26 huruf alfabet yang diambil dari 20 bentuk huruf yang berbeda. Data ini memiliki 20.000 data, 16 atribut dan 26 klaster. Data set *Letter-Recognition* memiliki distribusi data yang merata yang mana masing-masing klaster memiliki jumlah data berkisar antara 734 data hingga 813 data.

Data set *Letter-Recognition* diuji dengan algoritma SSNN menggunakan 5 variasi parameter nilai k tetangga terdekat awal dengan set nilai {200, 250, 300, 350, 400}, 10 variasi nilai ambang batas pergerakan data (MP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai ambang batas kedekatan ketetanggaan (CP) dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik ketika pasangan parameter k tetangga terdekat, MP, dan CP bernilai (400, 0.3, 0.5 - 1).

Pengujian dengan SNN menggunakan 5 variasi parameter nilai k tetangga terdekat dengan set nilai {200, 250, 300, 350, 400}, 10 variasi nilai *topic* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}, dan 10 variasi nilai *merge* dengan set nilai {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. Hasil klasterisasi terbaik diperoleh ketika pasangan parameter k tetangga terdekat, *topic*, dan *merge* bernilai (200, 0.5, 0.7). Hasil kedua algoritma tersebut ditunjukkan pada Tabel 7.

5.2 Analisis Hasil

Tabel 1 hingga 5 menunjukkan akurasi yang diperoleh algoritma SSNN lebih baik daripada akurasi yang diperoleh algoritma SNN. Perbedaan akurasi kedua algoritma tersebut sebesar 0,5% hingga 2 %. Hasil akurasi algoritma SSNN yang lebih baik daripada algoritma SNN tersebut disebabkan karena algoritma SSNN tidak menggunakan titik representatif untuk melakukan pembentukan klaster. Algoritma SSNN menggunakan proses pemutusan bobot hubungan ketetanggaan yang lemah pada graph ketetang-

Tabel 6: Hasil Pengujian Data Set Optdigits

	Algoritma SSNN	Algoritma SNN
Rerata Akurasi (%)	86,12	78,10
Rerata Memori (kBytes)	156529	76802
Waktu (det)	13270	4070

Tabel 7: Hasil Pengujian Data Set Letter-Recognition

	Algoritma SSNN	Algoritma SNN
Rerata Akurasi (%)	89,24	81,97
Rerata Memori (kBytes)	356529	176802
Waktu (det)	15741	5574

gaan dalam beberapa iterasi. Sehingga proses pembagian klaster dilakukan secara perlahan untuk mengurangi terjadinya data yang salah tanpa mengurangi kemungkinan data benar.

Tabel 6 dan 7 masih menunjukkan kondisi akurasi hasil yang lebih baik bagi algoritma SSNN. Perbedaan akurasi kedua algoritma pada pengujian data set Pendigits dan *Letter-Recognition* adalah sebesar 7% hingga 8%. Hal itu dikarenakan semakin besar total data dalam data set yang digunakan, maka semakin banyak iterasi yang bisa dilakukan oleh SSNN. Proses iterasi tidak berhenti karena nilai k tetangga terdekat yang digunakan lebih besar daripada total data yang diujikan. Proses pembentukan graph ketetanggaan berhenti karena telah mencapai nilai ambang batas kedekatan ketetanggaan yang dikehendaki, sehingga graph ketetanggaan yang diperoleh sudah dipetakan dengan lebih baik.

Untuk masalah waktu komputasi yang digunakan, pada Tabel 1 hingga 5 ditunjukkan bahwa algoritma SSNN membutuhkan waktu yang lebih kecil dibandingkan waktu yang dibutuhkan algoritma SNN. Tetapi pada Tabel 6 dan 7 diperoleh waktu komputasi yang dibutuhkan algoritma SSNN menjadi lebih besar 3 kali lipat waktu yang dibutuhkan algoritma SNN. Hal itu disebabkan pada pengujian data set yang memiliki total data besar (10.997 data dan 20.000 data), algoritma SSNN melakukan proses pembentukan graph ketetanggaan dalam iterasi yang lebih banyak daripada saat melakukan pengujian pada data set dengan total data yang lebih kecil (150 data hingga 1.797 data). Dalam setiap iterasi tersebut dilakukan pembentukan graph ketetanggaan dengan menggunakan nilai k tetangga terdekat yang bertambah besar dalam setiap iterasi, sehingga waktu komputasi menjadi semakin besar seiring dengan bertambah besar data set yang diujikan.

Tabel 1 hingga 7 memperlihatkan algoritma SSNN membutuhkan ruang memori yang lebih besar dibandingkan ruang memori yang dibutuhkan algoritma SNN. Penggunaan memori yang lebih besar tersebut disebabkan karena algoritma SSNN harus menyimpan keseluruhan daftar k tetangga terdekat untuk dapat membentuk graph ketetanggaan dalam beberapa iterasi. Sedangkan algoritma SNN hanya perlu menyimpan daftar k tetangga terdekat sebanyak k data saja untuk digunakan dalam pembentukan graph ketetanggaan.

Hasil pengujian algoritma SSNN pada data set Wine (Tabel 3) yang memiliki data yang tidak merata memperlihatkan kesulitan untuk memperoleh hasil yang terbaik terutama pada klaster yang memiliki jumlah data yang berbeda dengan klaster lainnya. Hal ini disebabkan karena algoritma SSNN melakukan proses pembentukan graph ketetanggaan dalam beberapa iterasi hingga dicapai kondisi optimal. Sebuah klaster dengan jumlah data yang besar akan mengakibatkan penggabungan klaster ketika dilakukan usaha optimasi klaster tersebut. Pada akhirnya, pengujian data set wine memberikan hasil yang terbaik ketika diperoleh hasil klasterisasi dengan distribusi data yang merata.

6 KOMPLEKSITAS ALGORITMA

Kompleksitas algoritma SSNN bergantung pada proses pembentukan graph ketetanggaan, yaitu sebesar $\Theta(n^2)$. Tetapi proses pembentukan graph ketetanggaan dilakukan dalam beberapa iterasi dengan nilai k tetangga terdekat yang semakin besar dalam setiap iterasinya. Karena proses iterasi tersebut, maka kompleksitas algoritma SSNN menjadi sebesar Akn^2. Sedangkan algoritma SNN memiliki kompleksitas sebesar kn^2.

Kompleksitas memori untuk algoritma SSNN adalah sebesar n^2 karena harus melakukan pembentukan graph k tetangga terdekat untuk seluruh titik data dalam data set. Sedangkan algoritma SNN membentuk daftar k tetangga terdekat untuk k data saja, sehingga kompleksitas memori algoritma SNN hanya sebesar kn saja.

7 KESIMPULAN DAN PENGEMBANGAN

Dalam penelitian ini telah berhasil dikembangkan sebuah algoritma klasterisasi berbasis data shrinking (SSNN) yang dapat menemukan klaster dengan bentuk, ukuran, dan kepadatan yang berbeda, dengan efisiensi yang lebih baik dari pada algoritma klasterisasi berbasis kepadatan (SNN). Akurasi yang diperoleh algoritma SSNN selalu lebih besar daripada algoritma SNN dengan perbedaan akurasi kedua algoritma berkisar antara 0,5% hingga 8%. Dari aspek efisiensi waktu dan memori, algoritma SSNN membutuhkan waktu komputasi dan ruang memori yang lebih besar dibandingkan waktu komputasi dan ruang memori yang dibutuhkan algoritma SNN.

Untuk pengembangan lebih lanjut, dapat dilakukan penelitian sebuah cara untuk melakukan pembentukan graph ketetanggaan pada titik-titik data dalam klaster yang sama saja, sehingga waktu komputasi dapat dikurangi.

DAFTAR PUSTAKA

[1] Tan, P.N., Steinbach, M., Kumar, V.: *Introduction to Data Mining*. Addison Wesley (2005)

[2] Han, J., Kamber, M.: *Data Mining: Concepts and Techniques*. 2nd edn. Morgan Kaufmann (2005)

[3] Hinneburg, A., Keim, D.A.: *An Efficient Approach to Clustering in Large Multimedia Databases with Noise*. In: ACM SIG Knowledge Discovery and Data Mining. (1998) 58–65

[4] Guha, S., Rastogi, R., Shim, K.: *CURE: An Efficient Clustering Algorithm for Large Databases*. In: SIG-

MOD '98: Proceedings of the 1998 ACM SIGMOD Intl. Conf. on Management of Data. (1998) 73–84

[5] Karypis, G., Han, E.H.S., Kumar, V.: *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*. IEEE Computer **32**(8) (1999) 68–75

[6] Guha, S., Rastogi, R., Shim, K.: *ROCK: A Robust Clustering Algorithm for Categorical Attributes*. In: ICDE '99: Proceedings of the 15th Intl. Conf. on Data Engineering. (1999) 512

[7] Zhang, T., Ramakrishnan, R., Livny, M.: *BIRCH: An Efficient Data Clustering Method for Very Large Databases*. SIGMOD Rec. **25**(2) (1996) 103–114

[8] Wang, W., Yang, J., Muntz, R.R.: *STING: A Statistical Information Grid Approach to Spatial Data Mining*. In: VLDB '97: Proceedings of the 23rd Intl. Conf. on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (1997) 186–195

[9] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: *OPTICS: Ordering Points to Identify the Clustering Structure*. SIGMOD Rec. **28** (1999) 49–60

[10] Ester, M., Kriegel, H.P., Jörg, S., Xu, X.: *A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. In: Proceedings of the Second Intl. Conf. on Knowledge Discovery and Data Mining, Morgan Kaufmann Publishers Inc. (1996) 226–231

[11] Seidl, T., Kriegel, H.P.: *Optimal Multi-step k-Nearest Neighbor Search*. In: SIGMOD '98: Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of data. (1998) 154–165

[12] Jarvis, R.A., Patrick, E.A.: *Clustering Using a Similarity Measure Based on Shared Near Neighbors*. IEEE Trans. Comput. **22**(11) (1973) 1025–1034

[13] L. Ertöz, M.S., Kumar, V.: *A New Shared Nearest Neighbor Clustering Algorithm and Its Applications*. In: Proc. Workshop on Clustering High Dimensional Data and its Applications. (2002)

[14] L. Ertöz, M. Steinbach, V.K.: *Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data*. In: Proc. of Second SIAM Intl. Conf. on Data Mining, San Francisco, CA, USA. (2003)

[15] Shi, Y., Song, Y., Zhang, A.: *A Shrinking-based Approach for Multi-dimensional Data Analysis*. In: VLDB '2003: Proceedings of the 29th Intl. Conf. on Very Large Data Bases. (2003) 440–451

[16] Shi, Y., Song, Y., Zhang, A.: *A Shrinking-Based Clustering Approach for Multidimensional Data*. IEEE Trans. on Knowl. and Data Eng. **17**(10) (2005) 1389–1403

[17] Levitin, A.V.: *Introduction to the Design and Analysis of Algorithms*. Addison Wesley (2002)

[HALAMAN INI SENGAJA DIKOSONGKAN]