APLIKASI METODE *CROSS ENTROPY* UNTUK *SUPPORT VECTOR MACHINES*

BUDI SANTOSA[1] DAN **TIANANDA WIDYARINI**[2]
[1, 2] Jurusan Teknik Industri Institut Teknologi Sepuluh Nopember (ITS) Surabaya

E-mail: budi_s@ie.its.ac.id, tiananda_w@yahoo.com

ABSTRAK

Dukungan vektor mesin (SVM) adalah metode yang kuat untuk masalah klasifikasi. Dalam rumusan aslinya, bentuk ganda dari SVM harus diselesaikan dengan pemrograman kuadratik untuk mendapatkan solusi optimal. Kelemahan dari versi standar sebagai masalah klasifikasi semakin besar, waktu komputasi tinggi diperlukan. Palang entropi (CE) adalah metode optimasi yang baru ditemukan dengan dua prosedur utama: menghasilkan sampel data dengan distribusi yang dipilih, dan memperbarui parameter distribusi karena sampel elit untuk menghasilkan sampel yang lebih baik di iterasi berikutnya. Metode CE telah diterapkan di banyak masalah optimasi dengan hasil memuaskan. Dalam penelitian ini, CE diterapkan untuk memecahkan masalah optimasi Lagrangian SVM untuk waktu yang lebih cepat komputasi. Metode ini diuji di beberapa dataset dunia nyata untuk masalah klasifikasi. Hasil penelitian menunjukkan bahwa penerapan CE di SVM adalah sebanding dengan SVM standar dalam mengklasifikasikan dua data kelas dalam hal akurasi. Selain itu, metode ini dapat memecahkan masalah besar dataset klasifikasi SVM lebih cepat dari standar.

***Kata kunci**: dukungan mesin vektor, cross entropi, data mining, klasifikasi*

ABSTRACT

Support vector machines (SVM) is a robust method for classification problem. In the original formulation, the dual form of SVM must be solved by a quadratic programming in order to get the optimal solution. The shortcoming of the standard version is as the classification problem is getting larger, the high computing time is needed. Cross entropy (CE) is a newly discovered optimization method with two main procedures: generating data samples by a chosen distribution, and updating the distribution parameters due to elite samples to generate a better sample in the next iteration. The CE method has been applied in many optimization problems with satisfying result. In this research, CE is applied to solve the optimization problem of Lagrangian SVM for faster computational time. This method is tested in some real world datasets for classification problem. The results show that the application of CE in SVM is comparable to standard SVM in classifying two class data in terms of accuracy. In addition, this method can solve large datasets classification problem faster than standard SVM.

***Key words**: support vector machines, cross entropy, data mining, classification*

PENDAHULUAN

Masalah klasifikasi banyak dijumpai dalam kehidupan sehari-hari seperti dalam penentuan diterima atau tidaknya pengajuan kredit dalam bidang perbankan, hingga diagnosis suatu penyakit di bidang kedokteran. Klasifikasi merupakan salah satu bentuk peramalan yang memiliki nilai keluaran diskrit, dan bertujuan untuk menemukan suatu fungsi keputusan *f(x)* yang secara akurat memprediksi kelas dari data (Santosa, 2007). Pola data dipelajari dengan pendekatan *supervised learning* untuk memprediksi data berikutnya yang memiliki kemiripan. Dalam pendekatan ini, label keluaran telah dikelompokkan, sehingga fungsi pemisah antara label satu dengan lainnya dapat dicari dengan mempelajari data kelas-kelas yang telah ada untuk mengklasifikasi data baru. Data yang digunakan untuk melatih fungsi disebut data *training*, sedangkan data untuk menguji model disebut data *testing*. Dalam *data mining* dan *machine learning* telah dikembangkan berbagai metode klasifikasi seperti analisis diskriminan (*linear discriminant analysis*), *decision tree, Artificial Neural Networks*, hingga *Support Vector Machines* (SVM). Pada beberapa penelitian, metode SVM telah terbukti mampu melakukan klasifikasi dengan baik untuk berbagai kasus. Menurut Frieβ, *et al*., pencarian *hyperplane* dengan menggunakan programa kuadratik SVM memiliki kelemahan yakni proses komputasi yang berat, berakibat pada waktu komputasi yang panjang.

Seiring dengan perkembangan metode *data mining*, ditemukan pula beberapa algoritma heuristik

untuk penyelesaian masalah optimasi. Berdasarkan (de Boer *et al*, 2004) beberapa algoritma heuristik yang dikembangkan untuk penyelesaian masalah optimasi, misalnya *Simulated Annealing* (Aarts dan Korst, 1989), *Genetic Algorithm* (Goldberg, 1989), *Tabu Search* (1993), dan *Cross Entropy* (CE). CE merupakan algoritma baru yang telah diaplikasikan untuk penyelesaian optimasi kombinatorial dan optimasi *multi-extremal*, serta *rare event simulation*. Aplikasi metode CE untuk kasus *klustering* juga memberi hasil yang baik dibandingkan metode *klustering* seperti *K-means* dan *fuzzy K-means* (Kroese *et al*., 2004). Pengembangan algoritma SVM dengan berbagai metode optimasi mampu memberikan hasil yang lebih baik. Hal ini dapat dilihat pada metode Kernel-Adatron (Frieβ, *et al*.) yang merupakan adaptasi algoritma Adatron (Anlauf dan Biehl, 1989) untuk klasifikasi kernel SVM. Aplikasi Adatron mempercepat proses penemuan solusi optimal pada kasus klasifikasi SVM yang biasanya membutuhkan waktu komputasi yang panjang. SVM sendiri merupakan suatu bentuk masalah optimasi yang bertujuan memaksimisasi *margin* dari klasifier. Mannor, *et al*. (2005) telah mengembangkan metode CE untuk klasifikasi yang menggunakan pendekatan L_0-Norm dengan tujuan meminimasi jumlah *support vector*. Diyakini bahwa dengan *support vector* yang minimal, tingkat kesalahan klasifikasi dapat di-*generate* lebih baik. Karena bertujuan meminimasi *support vector*, pendekatan CE yang digunakan adalah untuk *combinatorial optimization*. Melalui pendekatan tersebut ribuan problem optimasi harus diselesaikan untuk mendapatkan nilai optimal.

Pada penelitian ini bertujuan untuk menerapkan CE dalam menyelesaikan masalah optimasi Lagrangian SVM, yang merupakan bentuk dual dari problem optimasi *Support Vector Machine*.

METODE

Metode yang digunakan dalam penelitian ini menggunakan *support vector machines, cross entropy, algoritma svm-ce* yang dijelaskan secara rinci sebagai berikut.

Support Vector Machines

Berdasarkan Gunn (1998), *Support Vector Machine* (SVM) merupakan metode yang populer karena menjanjikan kinerja yang baik. Konsep SVM adalah pencarian hyperplane terbaik yang berfungsi sebagai pemisah data dari dua kelas pada *input space*. *Hyperplane* pemisah terbaik adalah *hyperplane* yang terletak di tengah di antara dua set obyek dari dua kelas (Santosa, 2007). *Hyperplane* terbaik dapat dicari dengan memaksimalkan *margin* atau jarak dari dua set obyek dari dua kelas yang berbeda.

Berdasarkan Gunn (2004), *hyperplane* pemisah dalam bentuk kanonikal harus memenuhi konstrain berikut,

$$y_i \left[\langle w, x_i \rangle + b \right] \geq 1, \; i = 1,\ldots, m \quad \ldots\ldots (1)$$

dimana x_i adalah data input, y_i adalah *output* yang nilainya = 1 atau -1, w dan b adalah parameter yang kita cari nilainya sehingga *hyperplane* yang mampu memisahkan data secara optimal adalah *hyperplane* yang meminimasi,

$$\Phi(w) = \frac{1}{2}\|w\|^2 \quad \ldots\ldots (2)$$

Berdasarkan Gunn (2004), solusi masalah optimasi dari persamaan 2 di bawah konstrain persamaan 1 diberikan melalui *saddle point* dari fungsi Lagrange (Lagrangian) (Minoux, 1986).

$$\Phi(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i [y_i (w^T x_i + b) - 1] \quad \ldots (3)$$

dimana α adalah Lagrange *multiplier*. Fungsi Lagrangian tersebut harus diminimasi berdasarkan w, b dan dimaksimasi berdasarkan $\alpha \geq 0$. Persamaan 6 dapat ditransformasi ke dalam bentuk permasalahan dual, yang lebih mudah diselesaikan. Dual problem diberikan dengan,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} (\min_{w,b} \Phi(w, b, \alpha) \quad \ldots\ldots (4)$$

Maka, dual problem menjadi,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j (x_i, x_j) + \sum_{i=1}^{m} \alpha_i \quad \ldots\ldots (5)$$

dan dengan demikian solusi permasalahan dapat diselesaikan melalui,

$$\alpha^* = \arg \min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j (x_i, x_j) - \sum_{i=1}^{m} \alpha_i \quad \ldots\ldots (6)$$

dengan fungsi pembatas,

$$\sum_{i=1}^{\ell} y_i \alpha_i = 0$$
$$0 \leq \alpha_i \leq C \qquad i = 1,\ldots m \quad \ldots\ldots (7)$$

Penyelesaian persamaan 5 dengan pembatas 7 akan menentukan Lagrange *multipliers*, dan *hyperplane* pemisah yang optimal diberikan dengan,

$$w^* = \sum_{i=1}^{N} \alpha_i y_i x_i$$
$$b^* = -\frac{1}{2}\langle w^*, x_r + x_s \rangle \quad \text{..... (8)}$$

dimana x_r dan x_s adalah sembarang *support vector* dari tiap kelas yang memenuhi,

$$w^* = \alpha_r, \alpha_s > 0, y_r = -1, y_s = 1 \quad \text{..... (9)}$$

Fungsi klasifier yang dapat digunakan untuk memisahkan data adalah,

$$f(x) = \text{sgn}\left(\langle w^*, x \rangle + b\right) \quad \text{..... (10)}$$

Dalam kasus dimana titik-titik dari dua kelas tidak bisa dipisahkan maka perlu ditambahkan variabel *slack* ξ_i. Konstrain persamaan (1) dimodifikasi menjadi,

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, m. \quad \text{..... (11)}$$

dimana $\xi_i \geq 0$. *Hyperplane* pemisah yang optimal ditentukan oleh vektor w, yang meminimasi fungsi,

$$\Phi(w, \xi) \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \quad \text{..... (12)}$$

dimana C adalah nilai biaya yang ditentukan sebagai penalti kesalahan dan sedapat mungkin diminimasi, dengan fungsi pembatas persamaan 11. Dual problem dari fungsi Lagrangian didapat seperti persamaan 5 dengan solusi permasalahan seperti persamaan 6 dan fungsi pembatas persamaan 7. Solusi dari permasalahan minimasi tersebut mirip dengan pada kasus dimana pola data dapat dipisahkan secara linier, kecuali pada modifikasi dari batasan *Lagrange* multiplier. Nilai C harus ditentukan dari awal perhitungan.

Pada kasus ketika batasan linier tidak dapat memisahkan kelas data, SVM dapat memetakan vektor input x ke dimensi yang lebih tinggi. Permasalahan SVM dapat dimodifikasi dengan memasukkan fungsi kernel. Menurut Santosa (2007), dengan metode kernel (Scholkopf dan Smola, 2002), suatu data x di *input space* dipetakan ke *feature space* F dengan dimensi yang lebih tinggi. Pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data yang berjarak dekat pada *input space* akan berjarak dekat juga pada *feature space*, sebaliknya dua data yang berjarak jauh pada *input space* juga berjarak jauh pada *feature space*.

Beberapa fungsi kernel dasar yang biasa dipakai dalam literatur SVM (Hsu *et al*., 2008) dan (Gunn, 1998) adalah:

- Linear: $K(x_i, x_j) = x_i^T x_j$,
- Polynomial: $K(x_i, x_j) = \left(\langle x_i, x_j^T \rangle + 1\right)^d$
- Radial basis function (RBF):

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j^T\|^2}{2\sigma^2}\right)$$

Beberapa fungsi kernel memiliki kondisi khusus untuk penggunaannya. Gunn (1998) menyatakan bahwa sebagian besar fungsi kernel telah memiliki bias implisit. Normalisasi data juga dibutuhkan untuk beberapa fungsi Kernel. Chin (1999) telah meneliti bahwa Kernel *polynomial* lebih cocok digunakan pada masalah dimana data *training*-nya telah dinormalisasi.

Cross Entropy

Pada subbab ini, prinsip-prinsip metode *Cross Entropy* (CE) akan dikemukakan berdasarkan Kroese *et al*. (2004) dan de Boer *et al*. (2005) beserta referensinya. Metode CE diawali dengan algoritma *adaptive* untuk mengestimasi probabilitas dari *rare-event* pada jaringan stokastik yang kompleks yang melibatkan minimasi standar deviasi.

Ide utama dari metode CE untuk optimasi dapat dinyatakan sebagai berikut: misalnya terdapat suatu masalah untuk memaksimasi kinerja fungsi S(x) pada setiap x di dalam kumpulan χ dimana nilai maksimum yang didapat adalah γ^*,

$$\gamma^* = \min_{z \in Z} S(z) \quad \text{..... (13)}$$

Pertama, masalah deterministik (13) dirandomisasi dengan mendefinisikan *probability density function* (pdf) {f(-;v), $v \in V$ } pada set χ. Diasumsikan pdf memiliki densitas yang berdegenerasi pada x*, misalnya f(-;v*). Selanjutnya, (14) diasosiasikan sebagai masalah estimasi dalam bentuk,

$$l(\gamma) = P_u(S(X) \geq \gamma) = E_u I_{\{S(X) \geq \gamma\}} \quad \text{..... (14)}$$

Di sini, X adalah vektor random dengan pdf f(-;u), untuk beberapa $u \in V$ dan $\gamma \in R$.

Tujuan algoritma CE adalah untuk menghasilkan urutan solusi $\{(\gamma_t, v_t)\}$, yang memusat dengan cepat ke arah solusi optimal (γ^*, v*). Sebagai awalnya, v_0 harus

ditetapkan, dan ρ yang tidak terlalu kecil dipilih, misalnya $\rho = 10^{-2}$. Metode CE melibatkan prosedur iterasi, dimana tiap iterasi dapat dipecah menjadi dua fase. (1) Melakukan generate sampel data random (χ) berdasarkan mekanisme spesifik; (2) Memperbaharui parameter (v) dari mekanisme random berdasarkan data sampel elite untuk menghasilkan sampel yang lebih baik pada iterasi berikutnya.

Proses tersebut secara matematis dapat ditulis sebagai berikut (Kroese *et al*., 2004):

1. Memperbaharui γ_t secara adaptif. Untuk v_{t-1}, γ_t adalah (1-ρ) persentil dari S(X) di bawah v_{t-1}. $\hat{\gamma}_t$ dari γ_t dapat diperoleh dengan mengacak suatu sampel random $X_1,\ldots X_N$ dari $f(\cdot;v_{t-1})$, menghitung performansi $S(X_i)$ untuk tiap i, mengurutkan performansi tersebut dari kecil ke besar: $S_{(1)} \le \ldots \le S_{(N)}$, dan terakhir mengevaluasi sampel sebesar (1-ρ) persentil dengan,

$$\hat{\gamma}_t = S_{(\lceil (1-\rho)N \rceil)} \quad \ldots\ldots (15)$$

   dimana S(k) merupakan urutan ke-k dari statistik $\{S(X_i)\}$
2. Memperbaharui v_t secara adaptif. Untuk γ_t dan v_{t-1} yang telah ditetapkan, turunkan v_t dari solusi programa CE,

$$\max_v D(v) = \max_v \mathrm{E}_{v_{t-1}} I_{\{S(Z)\le\gamma t\}} \log f(Z;v) \quad \ldots\ldots (16)$$

Bagian stokastik dari (3.5) adalah: untuk $\hat{\gamma}_t$ dan $\hat{v}_{t-1}$ yang telah ditetapkan, turunkan $\hat{v}_t$ dari programa berikut,

$$\max_v \hat{D}(v) = \max_v \frac{1}{N}\sum_{\ell=1}^{N} I_{\{S(Z^{\ell})\le\hat{\gamma}t\}} \log f(Z^{(\ell)};v) \quad \ldots\ldots (17)$$

Dibandingkan dengan melakukan *update* parameter vektor v secara langsung dari solusi (17), versi berikut dapat digunakan,

$$\hat{v}_t = \theta\tilde{v}_t + (1-\theta)\hat{v}_{t-1} \quad \ldots\ldots (18)$$

dimana $\tilde{v}_t$ adalah parameter vektor yang didapatkan dari solusi (17), dan θ disebut sebagai parameter *smoothing*, dengan $0{,}7 < \theta < 1$. Untuk $\theta = 1$, akan didapatkan aturan *update* yang asli. Beberapa alasan untuk menggunakan aturan *smoothed* daripada aturan *update* yang biasa adalah: (a) untuk memuluskan nilai $\hat{v}_t$, (b) untuk mengurangi probabilitas bahwa beberapa komponen $\hat{v}_{t,i}$ dari $\hat{v}_t$ akan bernilai nol atau satu pada beberapa iterasi awal.

Algoritma utama CE untuk optimasi:

1. Nyatakan $\hat{v}_o = u$. Tetapkan t = 1.
2. Bangkitkan sampel random $X_1,\ldots,X_N$ dari fungsi densitas tertentu $f(\cdot; v_{t-1})$ dan hitung sampel (1-ρ) persentil $\hat{\gamma}_t$ dari performansi berdasarkan persamaan 15.
3. Gunakan sampel yang sama dan selesaikan program stokastik (16). Catat solusinya sebagai $\tilde{v}_t$.
4. Aplikasikan persamaan 18 untuk memuluskan vektor $\tilde{v}_t$.
5. Jika untuk beberapa $t \ge d$, misalnya d = 5, $\hat{\gamma}_t = \hat{\gamma}_{t-1} = \ldots = \hat{\gamma}_{t-d}$,

maka hentikan iterasi (nyatakan T sebagai iterasi terakhir). Namun jika tidak terpenuhi, tetapkan t = t + 1 dan ulangi iterasi mulai dari langkah 2.

Perlu dicatat bahwa *stopping criterion*, vektor pertama, ukuran sampel N dan nilai ρ harus dinyatakan secara spesifik dari awal. Parameter v yang di-*update* berdasarkan (19) hanya sejumlah (1-ρ)% sampel terbaik. Data ini dinamakan sampel elite.

Algoritma SVM-CE

Masalah pencarian garis *hyperplane* terbaik untuk klasifikasi data dua kelas dapat dipecahkan dengan menyelesaikan persamaan *Lagrangian* SVM. Nilai α pada lagrange SVM akan dicari dan di-*update* menggunakan metode CE untuk menghasilkan klasifier yang paling optimal.Fungsi tujuan yang akan dipecahkan untuk mendapatkan *hyperplane* pemisah optimal adalah fungsi Lagrangian SVM seperti pada persamaan 5.

CE menyelesaikan suatu permasalahan optimasi dengan mula-mula membangkitkan bilangan random dalam distribusi tertentu, kemudian dari sampel tersebut dipilih sampel terbaik untuk menjadi parameter bilangan random pada iterasi berikutnya. Pada penelitian ini, sampel yang dibangkitkan adalah nilai α pada lagrange SVM. Algoritma yang diusulkan dijelaskan pada langkah satu hingga sembilan.

Langkah 1: Hitung matriks Kernel dikalikan label keluaran (*output*) dari data *training*

Perkalian matriks ditunjukkan pada persamaan berikut

$$H = \sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \langle x_i, x_j \rangle$$

Langkah 2: Tetapkan parameter awal

Parameter yang diperlukan yaitu jumlah sampel random yang akan dibangkitkan (N), batasan nilai α yang diijinkan dalam pembuatan sampel (C), persentase sampel elite (ρ).

Langkah 3: *Generate* sampel α

Nilai Lagrange multiplier α_i, untuk i = 1 hingga m digenerate menggunakan distribusi tertentu sebanyak N, dengan batasan nilai $0 \le \alpha \le C$. Pada penelitian ini digunakan distribusi normal.

Langkah 4: Hitung solusi Lagrangian SVM

Untuk semua nilai α dari i = 1 hingga N, hitung solusi Lagrangian SVM $W(\alpha)$ seperti pada persamaan (1). Persamaan tersebut ekuivalen dengan

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \alpha_i' H \alpha_i$$

Langkah 5: Urutkan nilai Lagrangian SVM

Langkah 6: Ambil sampel elite

Sampel elite diambil dari $W(\alpha)$ sebanyak ρN dengan nilai $W(\alpha)$ terbesar setelah diurutkan.

Langkah 7: Lakukan *update* parameter distribusi

Pada penelitian ini, distribusi yang digunakan adalah distribusi normal dengan parameter rata-rata (μ) dan standar deviasi (σ^2). Jika I adalah indeks dari N^{elite} sampel elite dan θ adalah variabel *smoothing*,

$$\mu_{tj} = \theta(\sum_{i \in I} \alpha_{ij} / N^{elite}) + (1-\theta)\mu_{t-1}$$

$$\sigma_{tj}^2 = \theta\left(\sum_{i \in I} \left(\alpha_{ij} - \mu_{ij}\right)^2 / N^{elite}\right) + (1-\theta)\sigma_{t-1}^2$$

Pada penelitian ini, digunakan $\theta = 1$.

Langkah 8: Ulangi langkah 3 hingga 7

Pengulangan dilakukan hingga *stopping criterion* terpenuhi. *Stopping criterion* yang digunakan delam penelitian ini adalah ketika standar deviasi maksimum dari α telah kurang dari 10^{-8}.

Langkah 9: Tetapkan nilai α

Untuk distribusi normal, nilai α adalah rata-rata α (μ) yang diperoleh dari iterasi terakhir.

Untuk mendapatkan fungsi klasifier, hitung vektor w optimal dengkan menggunakan α yang didapatkan dari langkah 9 seperti persamaan (8). Cari label data *testing* menggunakan persamaan (10), yakni pembulatan hasil dari perkalian vektor w* dengan data *testing* x ditambah bias.

HASIL DAN PEMBAHASAN

Algoritma yang diusulkan diuji dengan beberapa set data. Dalam pengujian, data dipisahkan menjadi set *training* dan testing. Proses *training* dilakukan untuk menemukan parameter model, sedangkan proses testing digunakan untuk menguji performansi model. Data yang digunakan untuk pengujian terdiri atas 6 kumpulan dataset dari kasus nyata yaitu data Hepatitis, Iris, *Breast Cancer*, *Haberman's Survival*, *Credit Approval*, dan *Splice* (UCI Repository, 2009). Penjelasan data yang digunakan ada pada Tabel 1. Data Hepatitis digunakan untuk memprediksi apakah seorang pasien hepatitis akan meninggal (-1) atau selamat (+1) berdasarkan atribut yang dimilikinya seperti usia, jenis kelamin, ada tidaknya varises, dan lainnya. Setelah *missing value* dihapus, data yang tersisa berjumlah 80. Data *training* yang digunakan berjumlah 53, dipilih secara acak, dan data *testing* adalah 27 data sisanya. Data iris asli memiliki tiga kelas jenis bunga dengan total data sebanyak 150. Untuk uji klasifikasi pada penelitian ini, data yang digunakan dibatasi sebanyak dua kelas, karena itu hanya diambil 100 data, dimana terdapat 50 data dari kelas +1 dan sisanya dari kelas -1. Untuk *training* dan *testing* digunakan 2:1.

Data *Breast Cancer* digunakan untuk memprediksi diagnosis kanker payudara, apakah jinak (+1) atau ganas (-1). Adapun data *training* yang digunakan adalah sebanyak 479 dari 683 data *Breast Cancer*, dan sisanya untuk data *testing*. Data *Haberman's Survival* digunakan untuk memprediksi apakah seorang pasien yang telah menjalani operasi kanker payudara akan bertahan hidup atau tidak dalam kurun waktu lima tahun. Data *Haberman's Survival* yang menjadi data *training* adalah sebanyak 204 data, dan sisanya 102 data sebagai data *testing*. Data *Credit Approval* asli terdiri dari 690 data dengan 16 atribut yang terdiri dari beragam jenis data, yaitu data kontinu dan nominal. Data nominal diubah ke dalam bentuk numeris agar dapat diolah pada pengujian data. Data ini memiliki *range* yang beragam antar atribut. Setelah data yang mengandung *missing value* dihapus, data *Credit Approval* terdiri dari 653 data dan 16 atribut. Pengujian data *Credit Approval* dilakukan dengan menggunakan 435 data *training* dan 218 data *testing*.

Data *Splice* asli terdiri dari 3.175 data untuk membedakan *splice junction* atau persimpangan dari urutan DNA; exon/intron (EI) atau intron/extron (IE). Data masih mengandung kelas yang tidak termasuk ke dalam dua kelas yang telah ditentukan sehingga dilakukan penghapusan data untuk data dengan kelas selain EI dan IE. Kegiatan *preprocessing* berikutnya yaitu mengubah jenis data dari kategorial menjadi numeris. Setelah *preprocessing*, data *Splice* yang tersisa sebanyak 1527 digunakan untuk pengujian. Pengujian data *Splice* dilakukan menggunakan 1018 data *training* dan 509 data *testing*.

Performansi aplikasi CE untuk SVM (SVM-CE), akan dibandingkan dengan metode Kernel Adatron (KA) dan SVM standar. Parameter komputasi CE yang digunakan adalah N = 40 dan ρ sebesar 0,2 (20% sampel). Untuk iterasi pada SVM-CE, *stopping criterion* yang digunakan adalah nilai maksimum dari standar deviasi σ kurang dari 10^{-8}.

Tabel 1. Keterangan data

No.	Dataset	Jumlah Data	Data *Training*	Atribut
1.	Hepatitis	80	53	19
2.	Iris	100	66	4
3.	*Breast Cancer*	683	479	9
4.	*Haberman's*	306	204	3
5.	*Credit App.*	653	435	15
6.	*Splice*	1527	1018	60

Pada pengujian, biaya penalti SVM yang dikenakan adalah C= 2, fungsi Kernel yang digunakan adalah Kernel RBF dengan $\sigma = 2$, Kernel polinomial derajat 5, serta Kernel linier. Untuk Kernel polinomial dan linier, data dinormalisasi terlebih dahulu. Normalisasi dilakukan dengan mengurangi data tiap atribut dengan rata-ratanya, kemudian membagi hasil pengurangan tersebut dengan standar deviasinya. Hasil uji yang dibandingkan adalah waktu komputasi (CPU *Time*) serta jumlah persentase misklasifikasi yang didapatkan untuk data testing. Hasil rata-rata dua puluh kali replikasi pengujian data untuk tiap metode dan parameter ditunjukkan pada Tabel 2 hingga Tabel 8. Nilai minimum untuk waktu komputasi dan kesalahan klasifikasi tiap metode diperjelas dengan huruf bercetak tebal.

Tabel 2. Hasil uji data Hepatitis

Metode	Kernel	Waktu komputasi T (detik)	Misklasifikasi (%)
	RBF, $\sigma = 2$	0,38 ± 0,06	24,07 ± 6,41
SVM-CE	Poly, d = 5	0,30 ± 0,00	17,78 ± 5,84
	Linear	0,30 ± 0,00	38,52 ± 8,10
	RBF, $\sigma = 2$	0,00 ± 0,00	22,78 ± 6,50
KA	Poly, d = 5	0,00 ± 0,00	30,56 ± 9,53
	Linear	0,00 ± 0,00	40,74 ± 14,22
	RBF, $\sigma = 2$	0,18 ± 0,04	25,00 ± 6,99
SVM Std.	Poly, d = 5	0,10 ± 0,00	27,78 ± 5,82
	Linear	0,10 ± 0,00	30,00 ± 6,00

Untuk data hepatitis Kernel polinomial derajat 5 untuk SVM-CE memberikan tingkat misklasifikasi paling kecil dibandingkan dengan uji metode KA dan SVM standar.

Iris dari segi keakuratan klasifikasi, metode SVM-CE dapat disejajarkan dengan SVM standar karena mampu mengklasifikasi data tanpa kesalahan. Namun, waktu komputasi SVM-CE masih kalah dibandingkan dengan pengujian menggunakan SVM standar, dimana dari ketiga metode Kernel yang diaplikasikan, semuanya menghasilkan solusi dalam waktu hanya 0,1 detik. Metode KA menghasilkan solusi dalam waktu 0 detik, tercepat di antara ketiga metode yang dibandingkan.

Tabel 3. Hasil uji data Iris

Metode	Kernel	Waktu komputasi (detik)	Misklasifikasi (%)
	RBF, $\sigma = 2$	0,40 ± 0,02	0,00 ± 0,00
SVM-CE	Poly, d = 5	0,35 ± 0,05	0,00 ± 0,00
	Linear	0,31 ± 0,02	0,00 ± 0,00
	RBF, $\sigma = 2$	0,00 ± 0,00	0,00 ± 0,00
KA	Poly, d = 5	0,00 ± 0,00	1,52 ± 2,51
	Linear	0,00 ± 0,00	0,00 ± 0,00
	RBF, $\sigma = 2$	0,100 ± 0	0,00 ± 0,00
SVM Std.	Poly, d = 5	0,100 ± 0	0,30 ± 0,93
	Linear	0,100 ± 0	0,00 ± 0,00

Tabel 4. Hasil uji data *Breast Cancer*

Metode	Kernel	T (detik)	Misklasifikasi (%)
	RBF, $\sigma = 2$	13,02 ± 0,41	3,53 ± 1,35
SVM-CE	Poly, d = 5	12,66 ± 0,28	3,06 ± 0,79
	Linear	10,87 ± 0,24	3,11 ± 0,78
	RBF, $\sigma = 2$	12,63 ± 0,43	3,19 ± 1,08
KA	Poly, d = 5	12,31 ± 0,03	4,95 ± 1,91
	Linear	12,39 ± 0,09	3,14 ± 1,05
	RBF, $\sigma = 2$	21,44 ± 0,28	6,05 ± 1,79
SVM Std.	Poly, d = 5	25,86 ± 0,37	6,44 ± 1,66
	Linear	24,49 ± 0,50	3,01 ± 1,20

Untuk data *Breast Cancer*, fungsi Kernel terbaik pada pengujian SVM-CE adalah Kernel polinomial derajat 5 yaitu dengan hasil misklasifikasi sebesar 3,06%. Pada KA, misklasifikasi terkecil didapatkan dengan aplikasi Kernel linier yaitu sebesar 3,14%. Pada SVM standar, fungsi Kernel linier memberikan hasil misklasifikasi sebesar 3,01%. Melalui uji statistik, ketiga hasil tersebut tidak berbeda secara signifikan. Karena itu, akurasi algoritma SVM-CE dapat dibandingkan dengan KA dan SVM standar untuk mengklasifikasi data *Breast Cancer*. Dibandingkan dengan kedua pengujian data sebelumnya, dataset *Breast Cancer* memiliki jumlah data yang jauh lebih banyak, yaitu 683 data. Dapat disimpulkan bahwa untuk data berukuran besar, SVM-CE mampu menyelesaikan masalah klasifikasi lebih cepat daripada SVM standar. Pada titik ini, pemusatan sebaran titik ke arah titik optimal (berdasarkan sampel elite) lebih cepat dibandingkan dengan penyelesaian *quadratic programming* SVM standar.

Tabel 5. Hasil uji *Haberman's Survival*

Metode	Kernel	Waktu komputasi (detik)	Misklasifikasi (%)
SVM-CE	RBF, $\sigma = 2$	2,36 ± 0,08	30,88 ± 2,73
	Poly, d = 5	2,36 ± 0,15	34,90 ± 5,53
	Linear	2,13 ± 0,12	26,13 ± 2,99
KA	RBF, $\sigma = 2$	0,20 ± 0,00	30,29 ± 4,23
	Poly, d = 5	0,20 ± 0,00	37,79 ± 6,04
	Linear	0,20 ± 0,00	29,41 ± 5,72
SVM Std.	RBF, $\sigma = 2$	1,75 ± 0,05	31,52 ± 3,73
	Poly, d = 5	2,04 ± 0,19	43,23 ± 9,32
	Linear	1,84 ± 0,06	26,32 ± 4,16

Untuk data *Haberman's Survival* didapatkan CE-SVM dengan kernel linear memberi misklasifikasi yang minimum. Tingkat misklasifikasi yang besar pada pengujian data *Haberman's Survival* menunjukkan pola data cukup sulit diramalkan. Dengan model klasifikasi CE-SVM atau SVM standar menggunakan fungsi Kernel linier, kesalahan klasifikasi tersebut dapat diminimisasi, tapi masih terdapat sekitar 26% misklasifikasi.

Tabel 6. Hasil uji data *Credit Approval*

Metode	Kernel	Waktu komputasi (detik)	Misklasifikasi (%)
SVM-CE	RBF, $\sigma = 2$	11,60 ± 0,42	44,73 ± 2,47
	Poly, d = 5	10,23 ± 0,28	15,41 ± 1,63
	Linear	8,97 ± 0,28	13,62 ± 2,42
KA	RBF, $\sigma = 2$	10,40 ± 0,32	45,16 ± 2,58
	Poly, d = 5	10,33 ± 0,10	25,34 ± 4,23
	Linear	9,01 ± 0,03	18,42 ± 6,76
SVM Std.	RBF, $\sigma = 2$	32,50 ± 2,07	43,33 ± 2,17
	Poly, d = 5	18,64 ± 0,36	21,58 ± 1,78
	Linear	17,17 ± 1,67	13,72 ± 2,28

Dari segi waktu komputasi, dapat disimpulkan bahwa pada ukuran data seperti data *Credit Approval*, kinerja SVM-CE mampu melampaui SVM standar dengan akurasi yang baik.

Data *Splice* merupakan dataset dengan jumlah data terbanyak pada keenam pengujian yang dilakukan pada penelitian ini. Dilihat dari tingkat kesalahan klasifikasi, baik pada SVM-CE dan SVM standar, kesalahan paling sedikit terjadi dengan menggunakan Kernel polinomial derajat 5. Untuk metode KA, misklasifikasi terkecil didapatkan dari aplikasi Kernel RBF parameter 5.

Namun jika dilihat dari waktu komputasi, perbedaan cukup jauh terjadi antara SVM-CE dengan SVM standar. Waktu komputasi tercepat pada SVM-CE diperoleh dengan Kernel linier yakni 70,20 detik, sedangkan waktu komputasi tercepat SVM standar dengan Kernel RBF adalah 212,2 detik. Jika diinginkan kesalahan klasifikasi yang kecil yaitu dengan menggunakan Kernel polinomial, SVM-CE hanya membutuhkan waktu komputasi 75,56 detik. Sedangkan, SVM standar membutuhkan waktu komputasi 264,99 detik.

Tabel 7. Hasil uji data *Splice*

Metode	Kernel	Waktu komputasi (detik)	Misklasifikasi (%)
SVM-CE	RBF, $\sigma = 2$	78,01 ± 1,83	4,93 ± 0,81
	Poly, d = 5	75,56 ± 1,65	4,35 ± 0,76
	Linear	70,20 ± 1,68	5,20 ± 0,92
KA	RBF, $\sigma = 2$	133,48 ± 0,14	5,41 ± 1,15
	Poly, d = 5	144,22 ± 7,23	12,21 ± 2,03
	Linear	135,04 ± 0,27	10,00 ± 2,20
SVM Std.	RBF, $\sigma = 2$	212,20 ± 4,32	4,81 ± 0,93
	Poly, d = 5	264,99 ± 5,39	3,94 ± 0,81
	Linear	248,55 ± 5,79	5,05 ± 0,77

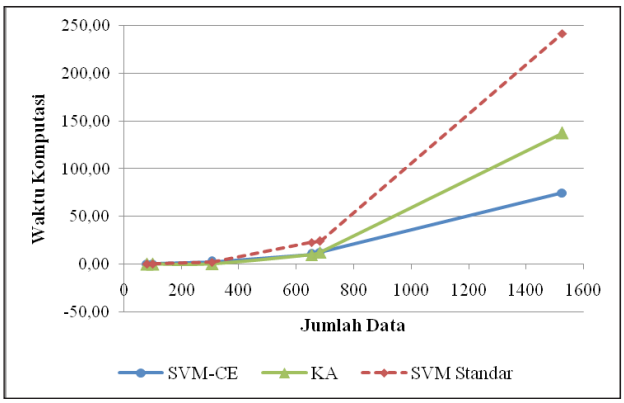
Tabel 8. Perbandingan *stopping criterion* uji data Hepatitis

Stopping criterion	Misklasifikasi	Waktu komputasi
*Standar deviasi tiap $\alpha \leq$ 10e-8	17,78%	0,3000
Standar deviasi tiap $\alpha \leq$ 10e-7	17,78%	0,2492
Standar deviasi tiap $\alpha \leq$ 10e-6	17,50%	0,2156
Standar deviasi tiap $\alpha \leq$ 10e-5	19,35%	0,1875
Standar deviasi tiap $\alpha \leq$ 10e-4	19,07%	0,1586
Standar deviasi tiap $\alpha \leq$ 10e-3	17,22%	0,1211
Standar deviasi tiap $\alpha \leq$ 10e-2	17,41%	0,0883

Karena waktu komputasi SVM-CE lebih besar daripada SVM standar, pengujian tambahan dilakukan untuk meneliti apakah dengan waktu komputasi lebih cepat SVM-CE masih dapat mengklasifikasi data dengan baik. Pengujian tambahan dilakukan dengan mengubah *stopping criterion* pada SVM-CE (Kernel polinomial derajat 5) yaitu memperbesar toleransi tingkat standar deviasi α untuk menjadi α optimal. Pada pengujian yang dilakukan sebelumnya, ketika tingkat standar deviasi α mencapai kurang dari 10^{-8} maka iterasi dihentikan. Tingkat standar deviasi kemudian akan dinaikkan menjadi 10^{-7} hingga 10^{-2}. bahkan dapat ditemui, untuk data hepatitis, tingkat kesalahan klasifikasi justru semakin kecil ketika toleransi standar deviasi diperbesar. Hal ini dapat dianalisa sebagai berikut: ketika nilai standar deviasi untuk melakukan *generate* sampel α_i masih besar, maka nilai α_i akan memiliki rentang yang besar (bervariasi sesuai distribusi normal). Karena nilai α optimal didapatkan dari rata-rata α_i pada iterasi terakhir,

jika nilai α_i masih bervariasi maka rata-ratanya akan di atas nol. Nilai α_i yang lebih besar dari nol menandakan bahwa data x_i menjadi *support vector*. Maka, semakin cepat iterasi dihentikan, titik data yang akan menjadi *support vector* semakin banyak. Kondisi ini dapat menyebabkan *overfitting* untuk klasifikasi berikutnya, yang berakibat data *testing* jatuh pada kelas yang salah, namun dapat juga menjadi batas klasifier yang baik jika data testing tepat masuk di dalam kelasnya. Pada pengujian, tidak diketahui secara pasti data mana yang menjadi data *testing* dan data *training* (acak), maka bisa jadi untuk pengujian klasifikasi dengan nilai misklasifikasi yang kecil, data *testing* memang merupakan data yang mudah untuk diklasifikasi.

Pada Gambar 1 ditunjukkan rata-rata waktu komputasi ketiga metode Kernel dari tiap metode dibandingkan dengan jumlah data uji pada keenam dataset. Dari Gambar 2 tersebut , dapat dilihat bahwa semakin besar jumlah data, kinerja SVM-CE akan semakin baik dari segi kecepatan waktu komputasi, dibandingkan dengan KA dan SVM standar.



Gambar 1. Perbandingan jumlah data vs waktu komputasi tiap metode

Perlu dicatat bahwa pada Gambar 2 waktu komputasi SVM CE diperoleh dari hasil uji dengan *stopping criterion maximum* standar deviasi 10e-8. Maka, jika nilai toleransi standar deviasi diatur menjadi lebih kecil, waktu komputasi yang dibutuhkan juga akan lebih cepat. Kecepatan komputasi SVM-CE masih di bawah SVM standar hingga jumlah data mencapai sekitar 300 data.

SIMPULAN

Metode *Cross Entropy* dapat digunakan untuk menyelesaikan problem *Support Vector Machine* dengan 6 dataset yang memiliki karakteristik berbeda, dimulai dari data berukuran kecil hingga besar, serta jumlah atribut sedikit hingga banyak. Semakin banyak data yang digunakan, maka semakin besar waktu yang dibutuhkan untuk penyelesaian masalah SVM baik dengan SVM standar atau SVM CE. Sedangkan banyaknya atribut pada data tidak berpengaruh. Dilihat dari hasil pengujian pada keenam dataset, metode SVM-CE yang dikembangkan terbukti mampu mengklasifikasi data dengan akurasi yang sebanding dengan SVM standar dan KA. Maka, metode ini dapat digunakan untuk penelitian berikutnya terutama pada data berukuran besar, dimana dibutuhkan kecepatan komputasi dengan akurasi yang tetap terjaga, karena metode ini telah terbukti mempunyai waktu komputasi yang cepat terutama untuk permsalahan dengan data yang besar.

DAFTAR PUSTAKA

Anlauf, J. K. dan Biehl, M., 1989. The AdaTron: an Adaptive Perceptron Algorithm. Europhysics Letters, Vol. 10(7), 687–692.

Chin, K.K., 1999. Support Vector Machines Applied to Speech Pattern Classification. Dissertation of Cambridge University Engineering Department. University of Cambridge.

De Boer, P. T., Kroese, D. P., Mannor, S. dan Rubinstein, R. Y. 2004. A Tutorial on the Cross-Entropy Method. Annals of Operations Research. Vol. 134(1), 19–67.

Frieβ, T., Cristianini, N., and Campbell, C., 1998. The Kernel-Adatron Algorithm: a Fast and Simple Learning Procedure for Support Vector Machines.

Gunn, S. R., 1998. Support Vector Machines for Classification and Regression. Technical Report. University of Southampton.

Kroese, D. P., Rubinstein, R. Y., Taimre, T., 2004. Application of the Cross-Entropy Method to Clustering and Vector Quantization. Journal of Global Optimization, 37: 137–157.

Osuna, E. E., Freund, R., dan Federico, G., 1997. Support Vector Macines: Training and Applications. A.I. Memo No. 1602.

Peleg, D., Mannor, S., dan Rubinstein, R. Y., 2005, The Cross-Entropy Method for Classification. Proceeding of the 22-nd International Conference on Machine Learning. Bonn, Germany.

Santosa, B., 2007. Data Mining: Teknik Pemanfaatan Data untuk Keperluan Bisnis. Graha Ilmu, Yogyakarta.

UCI Machine Learning, 2009,. *http://archive.ics.uci. edu/ml/machine-learning-databases/.*