PEMBELAJARAN INCREMENTAL PADA SUPPORT VECTOR MACHINE DENGAN PENDEKATAN PRIMAL

Princea Praditia S[1], Anny Yuniarti[2], Rully Soelaiman[3]
Teknik Informatika, Fakultas Teknologi Informasi, ITS
email : princea_praditiaS@yahoo.com[1], anny@if.its.ac.id[2], rully@if.its.ac.id

ABSTRAKSI

Support Vector Machine (SVM) telah terbukti sangat efektif dalam meyelesaikan permasalahan klasifikasi pola. Berbagai teknik terus dikembangkan dalam mengembangkan SVM yang lebih optimal, dan kebanyakan dari algoritma SVM tersebut diimplementasikan pada dual karena dianggap permasalahan SVM dapat lebih mudah diselesaikan dengan menggunakan pendekatan dual. Dalam tugas akhir ini dijelaskan bahwa baik didekati dengan pendekatan primal maupun dual, Primal SVM maupun Dual SVM akan sama-sama menghasilkan solusi optimal. Tapi ada kekurangan pada Primal SVM yaitu pada saat data training datang secara berurutan, proses komputasinya akan sangat bergantung pada data sampel karena data training yang baru harus di latih ulang dari awal. Karena itu, diterapkan algoritma baru yang dikenal dengan Incremental SVM. Incremental SVM terbukti dapat menyelesaikan permasalahan Primal SVM dengan mengupdate nilai bobot vector dan nilai bias yang dihasilkan pada proses traning untuk setiap data datang secara berurutan, sehingga proses komputasinya tidak terlalu besar, dan Incremental SVM ini terbukti dapat menghasilkan tingkat keakuratan yang cukup tinggi bila dibandingkan dengan Dual SVM biasa.

Kata kunci : Support Vector Machine, Incremental Support Vector Machine, Primal Support Vector Machine, kalsifikasi pada SVM

1. PENDAHULUAN

Klasifikasi dan pengenalan pola memainkan peran utama dalam kebanyakan kecerdasan buatan modern dan permasalahan *computer science* lainnya. Keteraturan, struktur atau hubungan apapun yang melekat pada beberapa sumber data dapat dipelajari dengan mendeteksi pola-pola yang signifikan dalam data yang tersedia. Sebuah sistem dapat membuat prediksi tentang data yang baru masuk dari sebuah sumber, dalam arti sistem tersebut telah mendapat kekuatan untuk belajar dan mengeneralisasi sesuatu yang baru dari sumber data tersebut.

Salah satu alat untuk mengklasifikasi pola adalah *Support Vector Machine* (SVM). SVM pertama kali diperkenalkan oleh Vapnik [2] pada tahun 1995 dan mendapatkan perhatian dari banyak penelitian karena fiturnya yang menarik dan menjanjikan kinerja empiris yang efektif, meskipun usianya relatif masih sangat muda.

Konsep dasar SVM sebenarnya merupakan kombinasi harmonis dari *Statistical Learning Theory*, *Structural Risk Minimization* dan *reproducing kernel* yang digabungkan menjadi sebuah algoritma yang dinamakan *Support Vector Machine*. SVM juga menggabungkan teori-teori komputasi yang telah ada puluhan tahun sebelumnya, seperti optimisasi konveks dan *margin hyperplane*

Penulis mengembangkan pembelajaran Incremental pada Support Vector Machine untuk mengatasi kelemahan Support Vector Machine yang harus dilatih ulang pada saat adanya penambahan data yang datang secara berurutan. Dalam makalah ini penulis membuktikan bahwa penerapan pembelajaran Incremental pada Support Vector Machine mengurangi biaya komputasinya hampir separuh dari biaya komputasi yang dipakai oleh SVM biasa.

Pada pembahasan selanjutnya akan dibahas lebih dalam tentang Support Vector Machine, tentang

Support Vector Machine yang berjalan pada kasus data yang dapat dipisahkan secara linear dan kasus data yang tidak dapat dipisahkan secara linear, pada bab selanjutnya akan dibahas lebih dalam tentang penerapan pembelajaran Incremental pada Support Vector Machine sendiri, selanjutnya pada bab selanjutnya akan dibahas mengenai dekomposisi cholesky yang digunakan pada pembelajaran Incremental, seperti apa dekomposisi cholesky itu dan apa yang menjadikan dekomposisi cholesky diterapkan kedalam pembelajaran Incremental pada Support Vector Machine.

Makalah ini juga menampilkan hasil uji coba penulis dalam menerapkan pembelajaran Incremental terhadap tiga jenis dataset yaitu : Ionosphere, Transfusion dan liver. Dataset tersebut merupakan data yang diambil langsung dari UCI Machine Learning Repository.

2. SUPPORT VECTOR MACHINE

Bentuk dasar Support Vector Machine adalah sebuah alat klasifikasi pola yang mengklasifikasikan data kedalam dua kelas yang diberi label {-1,1}, SVM merupakan supervised task karena data masukkannya memiliki sebuah label yang mengikuti setiap datanya. Tujuan SVM adalah menemukan batas pemisah antara kelas satu dengan kelas lainnya, pemisah tersebut dinamakan *hyperplane*, dengan cara mencari jarak maksimal batas pemisah antara margin dan *hyperplane* tersebut. Prinsip dasar SVM adalah pemisahan secara linear, dan selanjutnya dikembangkan agar dapat bekerja pada permasalahan *non-linear* dengan memanfaatkan fungsi kernel pada ruang vektor berdimensi tinggi, kemudian data tersebut diproyeksikan ke dalam ruang vektor tersebut sehingga data tersebut terpisah secara linear, baru kemudian SVM menyelesaikan permasalahan dengan data yang sudah terpisah secara linear dalam ruang vector yang berdimensi tinggi tersebut.

2.1 SVM pada kasus data terpisah secara linear

Semisal ada data training {xi,yi} dengan i=1,…,s , yi ϵ {1,-1}, xi ϵ R^n dipisahkan oleh garis pemisah f(x) = <w.x>+b. Pada kasus ini hyperplane pemisah dapat memisahkan data secara linear dengan fungsi batasan <xi.w>+b $\geq$ 1 dan <xi.w>+b $\leq$ 1 berlaku untuk keseluruhan data training, fungsi batasan tersebut dapat disingkat menjadi sebuah pertidaksamaan yi(<xi.w>+b)-1 $\geq$ 0 untuk semua i. Jarak antara hyperplane pemisah dengan titik yang berada paling dekat dengan hperplane dinamakan margin. Semisal jarak yang memisahkan *hyperplane* optimal dengan *hyperplnae* pemisah adalah d_- dan d_+. Maka nilai margin tersebut ditentukan oleh besarnya $d_+ + d_-$ [1] dan margin maksimal dihasilkan adalah $2/\|w\|$, dimana $\|w\|$ merupakan norm dari bobot vector w. Hyperplane pemisah H1, H2 sejajar dan mempunyai garis normal yang sama, tidak ada titik yang berada diantara dua hyperplane pemisah tersebut, H1 dan H2 yang berpegaruh terhadap pemaksimalan nilai margin dapat dicari dengan meminimalkan $\|w\|^2$ dengan fungsi batas yi(<xi.w>+b)-1 $\geq$ 0.

Titik – titik yang berada pada hyperplane pemisah tersebut dinamakan support vector, dan nantinya support vektor ini akan memberi informasi tentang penentuan hyperplane pemisah pada saat proses pembelajaran dijalankan.

2.2 SVM pada kasus data tidak dapat terpisah secara linear

Ketika dimensi data menjadi terlalu besar dan data tidak lagi dapat terpisahkan oleh hyperplane pemisah secara linier biaya komputasi akan tak terhitung mahal nya ketika SVM dipaksakan dipisahkan dengan pemisah yang tidak linear. Akan tetapi hal ini bisa dicegah dengan memanfaatkan fungsi kernel yang dapat memapingkan data ke dalam sebuah ruang vektor berdimensi tinggi sehingga data tersebut dapat dipisahkan secara linear. Untuk mereduksi penggunaan biaya komptasi yang terlampau mahal diperkenalkanlah sebuah slack variabel ξ_i , i = 1,…,s oleh Vapnik [5] pada fungsi batasannya sehingga permasalahan fungsi objektifnya berubah menjadi

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{s}\xi_i \qquad (1)$$

$$\text{subject to} \quad yi(<w,\Phi(x_i)>+b) \geq 1-\xi_i \qquad (2)$$

$$\xi_i \geq 0$$

dengan C adalah parameter yang menetukan besar pinalti akibat kesalahan dalam klasifikasi data dan nilai parameter C ditentukan oleh pengguna.

Sedangkan $\Phi(x_i)$ merupakan fungsi yang memapingkan x ke dalam ruang vector berdimensi tinggi. Jika seumpama persamaan (1) ditulis kembali dalam bentuk persamaan unconstraint maka didapatkan

$$\|w\|^2 + C \sum_{i=1}^{s} L(yi, < w.\Phi(x_i) > + b) \quad (3)$$

dengan

$$L(y, < w.\Phi(x) > + b) = \max(0, 1 - y(< w.\Phi(x) > + b))^p$$

dan L merupakan fungsi loss dimana saat nilai p=1, fungsi loss tersebut termasuk *hinge loss* dan ketika nilai p=2, fungsi loss tersebut merupakan *kuadratik loss*.

Semisal didefinisikan sebuah non-linear SVM dengan fungsi kernel k dan sebuah *Reproducing Kernel Hilbert Space* (RKHS) H yang bersesuaian dengan fungsi kernel tersebut, persamaan (3) dapat dituliskan sebagai

$$\lambda\beta^T K\beta + \sum_{i=1}^{s} L(y_i, K_i^T \beta) \quad (4)$$

dengan memasukkan $w = \sum_{i=1}^{s} \beta_i \Phi(x_i)$ kedalam persamaan(3). Nilai λ=1/C dan K merupakan kernel matrik dan K_{ij} = < $\Phi(x_i)$, $\Phi(x_j)$ > = k(x_i,x_j) sedangkan Ki merupakan elemen pada baris ke-i pada matrik K. Penerapan lebih jelas dapat dilihat pada [3]

2.3 Fungsi kernel

Metode kernel melakukan pendekatan permasalahan dengan melakukan pemetaan data ke dalam dimensi tinggi, dimana setiap koordinat berkorespondesi terhadap satu fitur dari ítem data, kemudian mentransformasikan data kedalam sekumpulan point ruang Euclidean. Ada dua alasan mengapa menggunakan kernel, yaitu menjadikan sebuah algoritma pembelajar untuk bisa menemukan sebuah pola linear pada ruang Euclidean tersebut dan membuat data non-vektor mudah untuk diakses dalam proses pembelajaran. Ada beberapa jenis fungsi kernel yang biasa digunakan, yaitu *polynomial*, kernel gaussian, *sigmoid*.[4] Tapi fungsi kernel yang dipakai penulis adalah kernel Gaussian. Fungsi kernel Gaussian yang dipakai adalah

$$k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / 2\sigma^2) \quad (5)$$

dimana i,j merupakan indeks baris yang ada di dalam data training, σ merupakan nilai sigma yang dimasukan user.

3.PEMBELAJARAN INCREMENTAL PADA SUPPORT VECTOR MACHINE

Semisal ada fungsi optimisasi dibawah ini :
min

$$\frac{1}{2}\left[\|w\|^2 + b^2 + C\sum_{i=1}^{s} L(y_i, \langle w, \Phi(x_i)\rangle + b)\right] \quad (6)$$

dengan s adalah jumlah data *training* yang ada. Karena w adalah merupakan kombinasi linear dari

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (7)$$

Jika digabungkan persamaan (6) dengan persamaan (7) maka akan didapatkan persamaan optimasi baru
min

$$\frac{1}{2}\left[\lambda(b \quad \beta^T)\begin{pmatrix}1 & 0\\ 0 & K\end{pmatrix}\begin{pmatrix}b\\ \beta\end{pmatrix} + \sum_{i=1}^{s} L\left(y_i, (1 \quad (K_i)^T)\begin{pmatrix}b\\ \beta\end{pmatrix}\right)\right] \quad (8)$$

Pada metode newton , setiap langkah pada newton method meliputi proses update sebagai berikut :

$$\begin{pmatrix}b\\ \beta\end{pmatrix} \leftarrow \begin{pmatrix}b\\ \beta\end{pmatrix} - H^{-1}\nabla \quad (9)$$

dengan H dan ∇ merupakan turunan gradient dari persamaan (8) dan Hessian nya maka didapatkan

$$\begin{pmatrix}b\\ \beta\end{pmatrix} \leftarrow H^{-1}\nabla\begin{pmatrix}e^T\\ K\end{pmatrix} I_0 Y \quad (10)$$

$$= \begin{pmatrix}-1 & e^T\\ I_0 e & \lambda I + I_0 K\end{pmatrix}^{-1}\begin{pmatrix}-\lambda & e^T\\ 0 & K\end{pmatrix}^{-1}\begin{pmatrix}e^T\\ K\end{pmatrix} I_0 Y$$

persamaan (10) kemudian disederhanakan menjadi persamaan (11)

$$\begin{pmatrix}b\\ \beta_{sv}\end{pmatrix} = \begin{pmatrix}-1 & e_{sv}^T\\ e_{sv} & \lambda I_{sv} + K_{sv}\end{pmatrix}^{-1}\begin{pmatrix}0\\ Y_{sv}\end{pmatrix}, \quad (11)$$

Penerapan lebih jelas dapat dilihat pada [5].

Ketika sampel baru (x_{s+1}, y_{s+1}) ditambahkan pada *training* set T, maka akan muncul masalah dalam mendapatkan nilai y_{s+1} f(x_{s+1}). Jika sebuah value y_{s+1} f(x_{s+1}) tidak lebih kecil dari 1, maka nilai β_{s+1} yang berkoresponden diset menjadi nol. Jika value y_{s+1} f(x_{s+1}) lebih kecil dari 1, mska sampel (x

$_{s+1}$, y $_{s+1}$) adalah *support vector* dan indexnya ditambahkan pada SV set.

Jika sampel (x $_{s+1}$, y $_{s+1}$) adalah *support vector*, maka persamaan (11) berubah menjadi

$$\begin{pmatrix} -1 & e^T & 1 \\ e_{sv} & K_{sv}+\lambda I_{sv} & \bar{k}_{s+1} \\ 1 & (\bar{k}_{s+1})^T & k(x_{s+1},x_{s+1})+\lambda \end{pmatrix}\begin{pmatrix} b \\ \beta_{sv} \\ \beta_{s+1} \end{pmatrix}=\begin{pmatrix} 0 \\ Y_{sv} \\ y_{s+1} \end{pmatrix}, \quad (12)$$

Semisal didefinisikan $\bar{R}$

$$\bar{R}=\begin{pmatrix} -1 & e^T & 1 \\ e_{sv} & K_{sv}+\lambda I_{sv} & \bar{k}_{s+1} \\ 1 & (\bar{k}_{s+1})^T & k(x_{s+1},x_{s+1})+\lambda \end{pmatrix} \quad (13)$$

dan untuk menyelesaikan persamaan (13) tersebut, matrik yang berordo 3 x 3 tersebut didekomposisi dengan menggunakan dekomposisi *cholesky*. Untuk mendekomposisi $\bar{R}$, terlebih dahulu harus mendapatkan hasil dekomposisi dari matrik R

$$R=\begin{pmatrix} -1 & e_{sv}^T \\ e_{sv} & \lambda I_{sv}+K_{sv} \end{pmatrix} \quad (14)$$

dengan L adalah *cholesky factor* dari R, L berupa lower triangular matrik dan D adalah merupakan diagonal matriks dan R = L D L^T.

Sehingga matrik $\bar{R}$ dapat didapatkan dengan menggunakan persamaan $\bar{R}=\bar{L}\bar{D}\bar{L}^T$. Sedangkan $\bar{L}$ dan $\bar{D}$ didapatkan dari persamaan

$$\bar{L}=\begin{pmatrix} L & 0 \\ l^T & 1 \end{pmatrix} \quad \bar{D}=\begin{pmatrix} D & 0 \\ 0 & d \end{pmatrix} \quad (15)$$

dimana l dan d didapatkan dari persamaan (16)

$$\begin{cases} LDI=(1,(\bar{k}_{s+1})^T)^T \\ d=k(x_{s+1},x_{s+1})+\lambda-l^T DI \end{cases} \quad (16)$$

Setelah itu baru dapat diperoleh beta dan b baru jika $\bar{R}$ sudah dapat didefinisikan.

SV set dan NSV set akan berubah setiap iterasi berjalan, element yang ada pada SV set bisa berubah menjadi anggota NSV set ataupun element NSV set mungkin bisa berubah menjadi anggota SV set. Saat pada salah satu iterasi, anggota dari SV set berubah menjadi NSV set, nilai beta yang berkoresponden dengan data tersebut diset menjadi nol sehingga tidak mempengaruhi iterasi selanjutnya. Jika seumpama (k-1) merupakan posisi suatu data pada SV set pada saat dia berpindah, maka R dapat dituliskan sebagai berikut :

$$R=\begin{pmatrix} R(1:k-1,1:k-1) & R(1:k-1,k) & R(1:k-1,k+1:sv) \\ R(k,1:k-1) & R(k,k) & R(k,k+1:sv) \\ R(k+1:sv,k-1) & R(k+1:sv,k) & R(k+1:sv,k+1:sv) \end{pmatrix} \quad (17)$$

R(i : j , u : v) dinotasikan sebagai submatrix dimana tiap elemen diambil dari baris ke i sampai j, dan kolom u sampai v pada matrix R.

$$\bar{R}=\begin{pmatrix} R(1:k-1,1:k-1) & R(1:k-1,k+1:sv) \\ R(k+1:sv,k-1) & R(k+1:sv,k+1:sv) \end{pmatrix} \quad (18)$$

Asumsikan bahwa dekomposisi *cholesky* dari R dinotasikan sebagai R = LDL^T. Maka pada saat salah satu anggota SV set pada sebuah iterasi berpindah menjadi NSV set persamaan $\bar{R}$ dapat diformulakan sebagai berikut :

$$\bar{R}=\hat{L}D(\hat{L})^T \quad (19)$$

dimana $\hat{L}$ didapatkan dengan membuang baris dengan indeks k dari matriks L.

Pengimplementasian persamaan (19) untuk proses update *cholesky factor* kemudian membuang salah satu baris pada indeks ke k tersebut dinamakan algoritma decremental.

4.DEKOMPOSISI CHOLESKY PADA PEMBELAJARAN INCREMENTAL

Seluruh matrik definite positif dapat didekomposisi dengan menggunakan dekompisisi *cholesky*. Syarat utama untuk dekomposisi *cholesky* adalah inputan matriknya harus merupakan matrik definite positif. Dekomposisi *cholesky* mengkonstruksi sebuah matrik definite positif *lower triangular matrik* L yang unik dari inputan matrik A yang akan didekomposisi. Sehingga matrik A tersebut dapat difaktorkan sebagai berikut:

$$A = LL^T \text{ atau } A=LDL^T$$

Dekomposisi *cholesky* memfaktorkan sebuah matrik definite positif A kedalam sebuah matrik segitiga bawah L dan matrik segitiga atas L^T . Sedangkan matrik D merupakan matrik diagonal dengan ukuran yang sama seperti matrik A. Matrik L tersebut yang kemudian dinamakan *cholesky faktor* dan hubungan A = LL^T dinamakan *cholesky factorization.*

5. UJI COBA DAN EVALUASI

5.1 Penerapan Pembelajaran Incremental pada dataset UCI

Pada uji coba ini penerapan pembelajaran Incremental diujicobakan pada dataset UCI Machine Learning Repository yaitu dataset ionosphere, transfusion dan liver dengan spesifikasi tiga jenis dataset tersebut dapat dilihat pada Tabel 1. Penerapan pembelajaran Incremental akan diujicobakan pada tiga dataset pada Tabel 1 dengan diberikan nilai parameter σ^2= 0.5 dan λ = 0.5 terhadap perubahan data training, data testing, dan data tambahan yang dimasukan sebagai inputan dari aplikasi.

Penambahan perubahan data training, data testing dan data tambahan yang digunakan dapat dilihat masing-masing pada Tabel 2, Tabel 3 dan Tabel 4. Uji coba ini didasarkan pada tingkat akurasi, tingkat eror dan waktu komputasi yang dibutuhkan saat penerapan pembelajaran incremental diterapkan. Pada Tabel 2 menggambarkan perubahan tingkar akurasi dan pencatatan waktu komputasi pada dataset ionosphere. Pada Tabel 3 menggambarkan perubahan tingkar akurasi dan pencatatan waktu komputasi pada dataset liver. Pada Tabel 4 perubahan tingkar akurasi dan pencatatan waktu komputasi pada dataset transfusion.

Tabel 1 Spesifikasi dataset UCI

dataset	atribut	jumlah	kelas
ionosphere	34	351	2
transfusion	7	345	2
liver	5	748	2

5.2 Pengaruh Parameter σ^2 dan λ pada Pembelajaran Incremental

Pada uji coba ini penerapan pembelajaran Incremental diujicobakan dengan perubahan nilai σ^2 dan λ yang berubah-ubah. Perubahan nilai σ^2dilakukan dengan variasi nilai $\sigma^2 = 2^{-6} - 2^6$. Sedangkan perubahan nilai λ dilakukan dengan variasi nilai $\lambda = 2^{-5} - 2^6$. Hasil tingkat keberhasilan masing-masing nilai λ dapat dilihat pada Tabel 6. Hasil tingkat keberhasilan masing-masing nilai σ^2dapat dilihat pada Tabel 5. Uji coba ini ditujukan agar dapat menentukan pada range mana nilai σ^2dan nilai λ mendapatkan hasil akurasi terbaik serta apakah perubahan nilai σ^2dan nilai λ berpengaruh terhadap waktu komputasi saat proses pembelajaran Incremental diterapkan.

Tabel 2 Hasil uji dengan dataset ionosphere

Data yang digunakan	Hasil uji coba	ISVM	PSVM	DSVM
data training : 30%	akurasi	84.3902 %	84 %	66.8227 %
data tambahan : 30%	error rate	15,61	16,00	33,18
data uji : 40%	waktu komputasi	2.0160	5.8280	9.1090
data training : 30%	akurasi	77.8409 %	77 %	64.9773%
data tambahan : 20%	error rate	22,16	23,00	35,02
data uji : 50%	waktu komputasi	1.3750	5.7040	6.1560
data training : 30%	akurasi	96.2264 %	77.5094 %	65.0943 %
data tambahan : 40%	error rate	3,77	22,49	34,91
data uji : 30%	waktu komputasi	2.6400	6.1090	13
data training : 10%	akurasi	65.2264 %	33.5094 %	32.7400 %
data tambahan : 10%	error rate	34,77	66,49	67,26
data uji : 80%	waktu komputasi	0.5630	2.0030	0.7660
data training : 20%	akurasi	72.9858 %	66.9858 %	30.3318 %
data tambahan : 20%	error rate	27,01	33,01	69,67
data uji : 60%	waktu komputasi	0.9530	2.5603	2.7600

Tabel 3 Hasil uji dengan dataset liver

Data yang digunakan	Hasil uji coba	ISVM	PSVM	DSVM
data training : 30%	akurasi	73.7226 %	70 %	70.4526 %
data tambahan : 30%	error rate	26,28	30,00	29,55
data uji : 40%	waktu komputasi	1.8120	3.7229	7.5940
data training : 30%	akurasi	73.0930 %	70.0190 %	70.2558%
data tambahan : 20%	error rate	26,91	29,98	29,74
data uji : 50%	waktu komputasi	1.3280	3.2220	7.6870
data training : 30%	akurasi	71.8447%	69.4450%	69.8155%
data tambahan : 40%	error rate	28,16	30,56	30,18
data uji : 30%	waktu komputasi	1.4380	2.4840	11.6570
data training : 10%	akurasi	71.2727%	70.3420%	70.5030%
data tambahan : 10%	error rate	28,73	29,66	29,50
data uji : 80%	waktu komputasi	0.4530	1.2810	1.7190

Tabel 4 Hasil uji dengan dataset transfusion

Data yang digunakan	Hasil uji coba	ISVM	PSVM	DSVM
data training : 30%	akurasi	78.3333 %	66 %	70.3333 %
data tambahan : 30%	error rate	21,67	34,00	29,67
data uji : 40%	waktu komputasi	4.2500	9.2820	41.3120
data training : 30%	akurasi	81.8182 %	75.4011 %	78.8770%
data tambahan : 20%	error rate	18,18	24,60	21,12
data uji : 50%	waktu komputasi	3.0150	6.5460	29.5160
data training : 30%	akurasi	96.2264 %	77.5094 %	65.0943 %
data tambahan : 40%	error rate	3,77	22,49	34,91
data uji : 30%	waktu komputasi	2.6400	6.1090	13
data training : 20%	akurasi	82.5893%	78.6411 %	78.5714 %
data tambahan : 20%	error rate	17,41	21,36	21,43
data uji : 60%	waktu komputasi	3.9380	5.0630	14.9690
data training : 40%	akurasi	88.6667 %	57.3333 %	66.6667 %
data tambahan : 40%	error rate	11,33	42,67	33,33
data uji : 20%	waktu komputasi	7.3900	17.5603	99.7650

Pada Tabel 5 dapat dilihat bahwa range nilai σ^2 terbaik yang dapat dihasilkan pada saat proses klasifikasi perangkat lunak berkisar antara 2^{-6} sampai 2^5 karena tingkat akurasinya tinggi pada saat lamda nya terletak diantara antara 2^{-6} sampai 2^5.

Waktu komputasi yang tercatat pada saat nilai σ^2 berubah-ubah berkisar antara 2.9220 sampai 3.1410. Dapat dilihat pada Tabel 6, bahwa pengaruh parameter lamda tidak terlalu besar pengaruhnya pada waktu komputasi saat proses pembelajaran Incremental dijalankan. Waktu komputasi bergerak konstan diantara 2.9-3 detik.

Pada Tabel 6 dapat dilihat bahwa range lamda terbaik yang dapat dihasilkan pada saat proses klasifikasi perangkat lunak berkisar antara 2^{-3} sampai 2^5 karena tingkat akurasinya tinggi pada saat lamda nya terletak diantara antara 2^{-3} sampai 2^5.

Waktu komputasi yang tercatat pada saat nilai lamda berubah-ubah berkisar antara 2.7350 sampai 2.9060. Dapat dilihat pada Tabel 6, bahwa pengaruh parameter lamda tidak terlalu besar pengaruhnya pada waktu komputasi saat proses pembelajaran Incremental dijalankan. Waktu komputasi bergerak konstan diantara 2-3 detik.

Tabel 5 Hasil uji dengan perubahan nilai σ^2

σ^2	waktu komputasi	akurasi	jumlah sv
2^{-1}	3.1410	92.4528	193
2^{-2}	3.0160	82.0755	185
2^{-3}	2.9840	88.6792	178
2^{-4}	3.0470	94.3396	168
2^{-5}	2.9540	96.2264	158
2^{-6}	3.0150	96.2264	139
2^{0}	3.0780	91.5094	207
2^{1}	2.9220	75.4717	219
2^{2}	3.1560	75.4717	224
2^{3}	3	96.2264	234
2^{4}	3.0470	96.2264	245
2^{5}	2.9850	96.2264	245
2^{6}	3.0470	50	245

Tabel 6 Hasil uji dengan perubahan nilai λ

λ	waktu komputasi	akurasi	jumlah sv
2^{-1}	2.7810	92.4528	193
2^{-2}	2.8590	92.4528	211
2^{-3}	2.8120	89.6226	224
2^{-4}	2.8750	69.8113	242
2^{-5}	2.8600	66.9811	245
2^{0}	2.8590	95.2830	181
2^{1}	2.8280	95.2830	183
2^{2}	2.8900	96.2264	182
2^{3}	2.8280	96.2264	180
2^{4}	2.7350	96.2264	180
2^{5}	2.7970	96.2264	180
2^{6}	2.9060	50	180

6. KESIMPULAN

Kesimpulan yang diperoleh berdasarkan uji coba dan evaluasi yang telah dilakukan adalah sebagai berikut:

1. Kelemahan Support Vector Machine adalah pada saat ada penambahan data ke dalam data *training*, data *training* harus dilatih ulang dari awal sehingga biaya komputasinya tinggi, dengan diterapkannya pembelajaran Incremental pada Support Vector Machine data *training* tidak harus dilatih ulang dari awal sehingga biaya komputasinya tidak lagi tinggi dan tingkat keakurasiannya juga tidak kalah tinggi dengan Support Vector Machine yang biasa.
2. Penggunaan dekomposisi *cholesky* terbuki efektif dalam mereduksi hampir 50% dari biaya komputasi yang seharusnya bahkan dari data set yang besar dengan penambahan data yang secara berurut ditambahkan kedalam data *training*
3. Perubahan nilai parameter yaitu λ dan δ^2 juga akan mempengaruhi hasil akurasi yang dihasilkan pada saat proses klasifikasi. Sedangkan perubahan nilai λ dan δ^2 terhadap waktu komputasi sedikit berpengaruh dan dapat ditetapkan nilainya sebagai suatu konstanta.
4. Penerapan pembelajaran Incremental pada Support Vector Machine terbukti menaikan tingkat akurasi data sebanyak hampir 30-40% dari tingkat akurasi Support Vector Machine dual biasa.
5. Perolehan tingkat akurasi terbaik didapatkan pada saat λ berkisar antara 2^{-3} - 2^{5} dan δ^2 berkisar antara 2^{-6} - 2^{5}

8.REFERENSI

[1] Burges, Cristhoper J.C. 1998. “A Tutorial on Support Vector Machine for Pattern Recognition,” Bell Laboratories,Lucent Technologies. Data Mining and Knowledge Discovery,2,121-167(1998).

[2] Vapnik, V. 1995. “The Nature of Statistical Learning Theory,” Springer-Verlag, New York.

[3] Chapple, Oliver. 2006.”Training Support Vector Machine in the Primal,” MPI for Biological Cybernetics, 72076 T¨ubingen, Germany.

[4] Cristianini, Nello dan John S. Taylor. 2004. “Kernel Methods for Pattern Analysis,” Cambridge - University Press.

[5] Liang, Zhizheng dan YouFu Li. 2009. “Incremental support vector machine learning in the primal and applications,” Neurocomputing 72 (2009) 2249 – 2258, Elsevier.