Pengantar

Dokumen ini diambil dari Tugas Akhir saya yang berjudul **Penerapan Teknik Support Vector Machine untuk Pendeteksian Intrusi pada Jaringan**. Dokumen ini dibuat hanyak untuk memberikan teori SVM dalam bahasa Indonesia karena penulis belum menemukan tutorial SVM dalam bahasa Indonesia. Selain itu, tutorial dalam bahasa Inggris yang ditemukan penulis juga tidak cukup mudah dipahami. Semoga dokumen ini dapat membantu dalam mempelajari SVM.

September 2007,

Krisantus Sembiring
x_sanctus@yahoo.com
S1 Teknik Informatika, Sekalah Teknik Elektro dan Informatika, ITB

Daftar Isi

Support Vector Machine

Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi-fungsi linier dalam sebuah ruang fitur (*feature space*) berdimensi tinggi, dilatih dengan algoritma pembelajaran yang didasarkan pada teori optimasi dengan mengimplementasikan *learning bias* yang berasal dari teori pembelajaran statistik [CHR00]. Teori yang mendasari SVM sendiri sudah berkembang sejak 1960-an, tetapi baru diperkenalkan oleh Vapnik, Boser dan Guyon pada tahun 1992 dan sejak itu SVM berkembang dengan pesat. SVM adalah salah satu teknik yang relatif baru dibandingkan dengan teknik lain, tetapi memiliki performansi yang lebih baik di berbagai bidang aplikasi seperti *bioinformatics*, pengenalan tulisan tangan, klasifikasi teks dan lain sebagainya [CHR01].

Structural Risk Minimization (SRM)

Proses pembelajaran pada SVM bertujuan untuk mendapatkan hipotesis berupa bidang pemisah terbaik yang tidak hanya meminimalkan *empirical risk* yaitu rata-rata *error* pada data pelatihan, tetapi juga memiliki generalisasi yang baik Generalisasi adalah kemampuan sebuah hipotesis untuk mengklasifikasikan data yang tidak terdapat dalam data pelatihan dengan benar. Untuk menjamin generalisasi ini, SVM bekerja berdasarkan prinsip SRM.

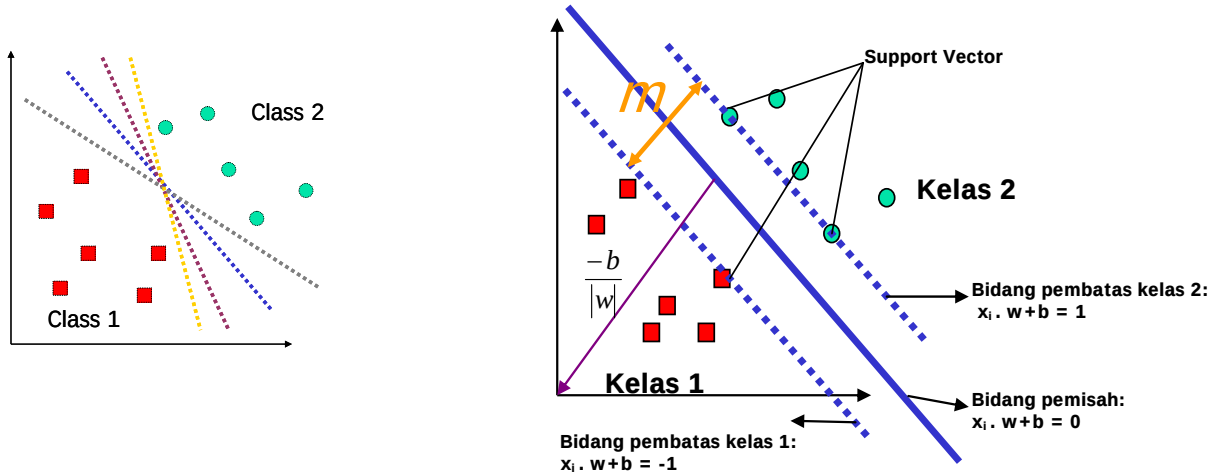
SRM bertujuan untuk menjamin batas atas dari generalisasi pada data pengujian dengan cara mengontrol ”kapasitas” (fleksibilitas) dari hipotesis hasil pembelajaran. Untuk mengukur kapasitas ini digunakan dimensi Vapnik-Chervonenkis (VC) yang merupakan properti dari ruang hipotesis $\{f(\alpha)\}$. Nilai dari dimensi VC ini, berdasarkan teori pembelajaran statistik akan menentukan besarnya nilai kesalahan hipotesis pada data pengujian. Lebih jelasnya, besar kesalahan pada data pengujian/ *actual risk* $R(\alpha)$ dengan probabilitas sebesar $1-\eta, 0 \le \eta \le 1$, pada dataset yang terdiri dari n data dapat dilihat pada persamaan (2.1). $R_{emp}(\alpha)$adalah kesalahan pada data pelatihan dan h adalah dimensi VC.

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\left(\frac{h\left(\log\left(\frac{2l}{h} \right) + 1 \right) - \log\left(\frac{\eta}{4} \right)}{l} \right)} \qquad (2.1)$$

Nilai VC *confidence* (nilai elemen kedua pada ruas kanan (2.1)), ditentukan oleh hipotesis/ fungsi hasil pembelajaran [BUR98]. Jadi, prinsip SRM adalah menemukan subset dari ruang hipotesis yang dipilih sehingga batas atas *actual risk* dengan menggunakan subset tersebut diminimumkan. SRM bertujuan untuk meminimumkan *actual risk* dengan cara meminimumkan kesalahan pada data pelatihan dan juga VC *confidence*. Namun, implementasi SRM tidak dilakukan dengan meminimumkan persamaan (2.1) karena dimensi VC dari ruang hipotesis $\{f(\alpha)\}$ sulit untuk dihitung dan hanya terdapat sedikit model hipotesis yang diketahui bagaimana cara menghitung dimensi VC-nya [OSU97]. Selain itu, walaupun dimensi VC dapat dihitung, tidak mudah meminimumumkan persamaan (2.1). Implementasi SRM pada SVM menggunakan fungsi linier dan akan dijelaskan pada bagian selanjutnya.

SVM pada *Linearly Separable Data*

Linearly separable data merupakan data yang dapat dipisahkan secara linier. Misalkan $\{x_1,...,x_n\}$adalah dataset dan $y_i \in \{+1,-1\}$ adalah label kelas dari data x_i. Pada gambar 1 dapat dilihat berbagai alternatif bidang pemisah yang dapat memisahkan semua data set sesuai dengan kelasnya. Namun, bidang pemisah terbaik tidak hanya dapat memisahkan data tetapi juga memiliki margin paling besar.



Gambar 1 Alternatif bidang pemisah (kiri) dan b*idang pemisah* terbaik dengan margin (m) terbesar (kanan)

Adapun data yang berada pada bidang pembatas ini disebut *support vector*. Dalam contoh di atas, dua kelas dapat dipisahkan oleh sepasang bidang pembatas yang sejajar. Bidang pembatas

pertama membatasi kelas pertama sedangkan bidang pembatas kedua membatasi kelas kedua, sehingga diperoleh:

$$\begin{aligned} x_i.w+b \geq +1 \quad for \quad y_i = +1 \\ x_i.w+b \leq -1 \quad for \quad y_i = -1 \end{aligned} \qquad (2.1)$$

w adalah normal bidang dan b adalah posisi bidang relatif terhadap pusat koordinat. Nilai margin (jarak) antara bidang pembatas (berdasarkan rumus jarak garis ke titik pusat) adalah $\frac{1-b-(-1-b)}{w} = \frac{2}{|w|}$. Nilai margin ini dimaksimalkan dengan tetap memenuhi (2.1). Dengan mengalikan b dan w dengan sebuah konstanta, akan dihasilkan nilai margin yang dikalikan dengan konstanta yang sama. Oleh karena itu, konstrain (2.1) merupakan *scaling constraint* yang dapat dipenuhi dengan *rescaling* b dan w. Selain itu, karena Memaksimalkan $\frac{1}{|w|}$ sama dengan meminimumkan $|w|^2$ dan jika kedua bidang pembatas pada (2.1) direpresentasikan dalam pertidaksamaan (2.2),

$$y_i(x_i.w+b)-1 \geq 0 \qquad (2.2)$$

maka pencarian bidang pemisah terbaik dengan nilai margin terbesar dapat dirumuskan menjadi masalah optimasi konstrain, yaitu

$$\begin{aligned} &\min \frac{1}{2}|w|^2 \\ &s.t \ \ y_i(x_i.w+b)-1 \geq 0 \end{aligned} \qquad (2.3)$$

Persoalan ini akan lebih mudah diselesaikan jika diubah ke dalam formula *lagrangian* yang menggunakan *lagrange multiplier*. Dengan demikian permasalahan optimasi konstrain dapat diubah menjadi:

$$\min_{w,b} L_p(w,b,\alpha) \equiv \frac{1}{2}|w|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i.w+b) + \sum_{i=1}^{n} \alpha_i \qquad (2.4)$$

dengan tambahan konstrain, $\alpha_i \geq 0$ (nilai dari koefisien *lagrange*). Dengan meminimumkan Lp terhadap w dan b, maka dari $\frac{\partial}{\partial b} L_p(w,b,\alpha) = 0$ diperoleh (2.5) dan dari $\frac{\partial}{\partial w} L_p(w,b,\alpha) = 0$ diperoleh (2.6).

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad (2.5)$$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad (2.6)$$

Vektor w sering kali bernilai besar (mungkin tak terhingga), tetapi nilai α_i terhingga. Untuk itu, formula *lagrangian* Lp (*primal problem*) diubah kedalam *dual problem* $L_{D.}$ Dengan mensubsitusikan persamaan (2.6) ke L_P diperoleh *dual problem* L_D dengan konstrain berbeda.

$$L_D(\alpha) \equiv \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j x_i . x_j \quad (2.7)$$

$\min_{w,b} L_p = \max_{\alpha} L_D$. Jadi persoalan pencarian bidang pemisah terbaik dapat dirumuskan sebagai berikut:

$$\max_{\alpha} L_D \equiv \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j x_i . x_j \qquad (2.8)$$
$$s.t. \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0$$

Dengan demikian, dapat diperoleh nilai α_i yang nantinya digunakan untuk menemukan w. Terdapat nilai α_i untuk setiap data pelatihan. Data pelatihan yang memiliki nilai $\alpha_i > 0$ adalah *support vector* sedangkan sisanya memiliki nilai $\alpha_i = 0$. Dengan demikian fungsi keputusan yang dihasilkan hanya dipengaruhi oleh *support vector*.

Formula pencarian bidang pemisah terbaik ini adalah pemasalahan *quadratic programming*, sehingga nilai maksimum global dari α_i selalu dapat ditemukan. Setelah solusi pemasalahan *quadratic programming* ditemukan (nilai α_i), maka kelas dari data pengujian x dapat ditentukan berdasarkan nilai dari fungsi keputusan:

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i x_i . x_d + b, \qquad (2.9)$$

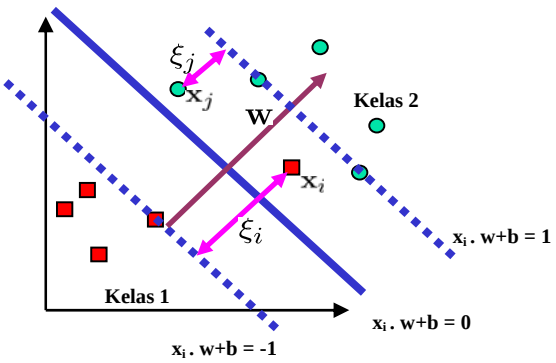
x_i adalah *support* vector, ns = jumlah *support vector* dan x_d adalah data yang akan diklasifikasikan.

SVM pada *Nonlinearly Separable Data*

Untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier formula SVM harus dimodifikasi karena tidak akan ada solusi yang ditemukan. Oleh karena itu, kedua bidang pembatas (2.1) harus diubah sehingga lebih fleksibel (untuk kondisi tertentu) dengan penambahan variabel ξ_i ($\xi_i \geq 0, \forall_i$: $\xi_i = 0$ jika x_i diklasifikasikan dengan benar) menjadi $x_i.w + b \geq 1 - \xi_i$ untuk kelas 1 dan $x_i.w + b \leq -1 + \xi_i$ untuk kelas 2. Pencarian bidang pemisah terbaik dengan dengan penambahan variabel ξ_i sering juga disebut *soft margin hyperplane*. Dengan demikian formula pencarian bidang pemisah terbaik berubah menjadi:

$$\begin{aligned} &\min \frac{1}{2}|w|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \\ &s.t. y_i (w.x_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \qquad (2.10)$$

C adalah parameter yang menentukan besar penalti akibat kesalahan dalam klasifikasi data dan nilainya ditentukan oleh pengguna. Bentuk persoalan (2.10) memenuhi prinsip SRM, dimana meminimumkan $\frac{1}{2}|w|^2$ ekivalen dengan meminimumkan dimensi VC dan meminimumkan $C\left(\sum_{i=1}^{n} \xi_i\right)$ berarti meminimumkan error pada data pelatihan [OSU97].



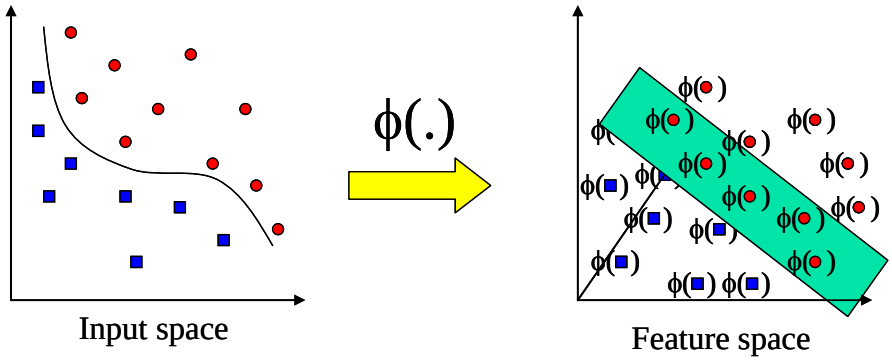
Gambar 2 *Soft margin hyperplane*

Selanjutnya, bentuk *primal problem* sebelumnya berubah menjadi:

$$\min_{w,b} L_p(w,b,\alpha) \equiv \frac{1}{2}|w|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) - \sum_{i=1}^{n} \alpha_i \{y_i (x_i.w + b) - 1 + \xi_i\} - \sum_{i=1}^{n} \mu_i \xi_i \qquad (2.11)$$

Pengubahan Lp ke dalam *dual problem*, menghasilkan formula yang sama dengan persamaan (2.6) sehingga pencarian bidang pemisah terbaik dilakukan dengan cara yang hampir sama dengan kasus dimana data dapat dipisahkan secara linier, tetapi rentang nilai α_i adalah $0 \geq \alpha_i \geq C$. *Instance* yang memiliki nilai $\alpha_i = C$ disebut *bounded support vector*.

Metode lain untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier adalah dengan mentransformasikan data ke dalam dimensi ruang fitur (*feature space*) sehingga dapat dipisahkan secara linier pada *feature space*.

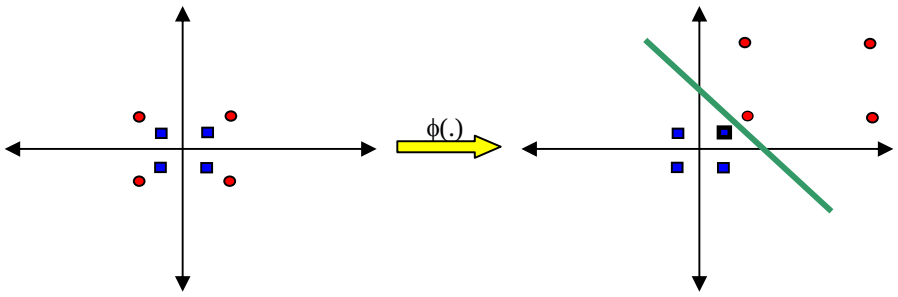


Gambar 3 Transformasi dari *vektor input* ke *feature space*

Caranya, data dipetakan dengan menggunakan fungsi pemetaan (transformasi) $x_k \rightarrow \phi(x_k)$ ke dalam *feature space* sehingga terdapat bidang pemisah yang dapat memisahkan data sesuai dengan kelasnya (gambar 3). Misalkan terdapat data set yang datanya memiliki dua atribut dan dua kelas yaitu kelas positif dan negatif. Data yang memiliki kelas positif adalah $\{(2,2),(2,-2),(-2,2),(-2,-2)\}$, dan data yang memiliki kelas negatif $\{(1,1),(1,-1),(-1,1),(-1,-1)\}$. Apabila data ini digambarkan dalam ruang dua dimensi (gambar 4) dapat dilihat data ini tidak dapat dipisahkan secara linier. Oleh karena itu, digunakan fungsi transformasi berikut:

$$\phi(x_1, x_2) = \begin{Bmatrix} \sqrt{x_1^2 + x_2^2} > 2 \rightarrow (4 - x_2 + |x_1 - x_2|,\ 4 - x_1 + |x_1 - x_2|) \\ \sqrt{x_1^2 + x_2^2} \leq 2 \rightarrow (x_1, x_2) \end{Bmatrix} \quad (2.12)$$

Data sesudah transformasi adalah $\{(2,2),(6,2),(6,6),(2,6)\}$ untuk kelas negatif, dan $\{(1,1),(1,-1),(-1,1),(-1,-1)\}$ untuk kelas positif. Selanjutnya pencarian bidang pemisah terbaik dilakukan pada data ini.



Gambar 4 Contoh transformasi untuk data yang tidak dapat dipisahkan secara linier

Dengan menggunakan fungsi transformasi $x_k \rightarrow \phi(x_k)$, maka nilai $w = \sum_{i=1}^{ns} \alpha_i y_i \phi(x_i)$ dan fungsi hasil pembelajaran yang dihasilkan adalah

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i \phi(x_i)\phi(x_d) + b \qquad (2.13)$$

Feature space dalam prakteknya biasanya memiliki dimensi yang lebih tinggi dari vektor input (*input space*). Hal ini mengakibatkan komputasi pada *feature space* mungkin sangat besar, karena ada kemungkinan *feature space* dapat memiliki jumlah *feature* yang tidak terhingga. Selain itu, sulit mengetahui fungsi transformasi yang tepat. Untuk mengatasi masalah ini, pada SVM digunakan "*kernel trick*". Dari persamaan (2.11) dapat dilihat terdapat *dot product* $\phi(x_i)\phi(x_d)$. Jika terdapat sebuah fungsi kernel K sehingga $K(x_i, x_d) = \phi(x_i).\phi(x_d)$, maka fungsi transformasi $\phi(x_k)$ tidak perlu diketahui secara persis. Dengan demikian fungsi yang dihasilkan dari pelatihan adalah

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i K(x_i, x_d) + b \ (x_i = support\ \ vector). \qquad (2.14)$$

Syarat sebuah fungsi untuk menjadi fungsi kernel adalah memenuhi *teorema Mercer* yang menyatakan bahwa matriks kernel yang dihasilkan harus bersifat *positive semi-definite*. Fungsi kernel yang umum digunakan adalah sebagai berikut:

a. *Kernel Linier*

$$K(x_i, x) = x_i^T x \qquad (2.15)$$

b. *Polynomial kernel*

$$K(x_i, x) = (\gamma . x_i^T x + r)^p, \gamma > 0 \qquad (2.16)$$

c. *Radial Basis Function* (RBF)

$$K(x_i, x) = \exp\left(-\gamma |x_i - x|^2\right), \gamma > 0 \quad (2.17)$$

d. *Sigmoid kernel*

$$K(x_i, x) = \tanh\left(\gamma x_i^T x + r\right) \quad (2.18)$$

Menurut tutorial pada [HSU04] fungsi kernel yang direkomendasikan untuk diuji pertama kali adalah fungsi kernel RBF karena memiliki performansi yang sama dengan kernel linier pada parameter tertentu, memiliki perilaku seperti fungsi kenel *sigmoid* dengan parameter tentu dan rentang nilainya kecil [0,1].

Multi Class SVM

SVM saat pertama kali diperkenalkan oleh Vapnik, hanya dapat mengklasifikasikan data ke dalam dua kelas (klasifikasi biner). Namun, penelitian lebih lanjut untuk mengembangkan SVM sehingga bisa mengklasifikasi data yang memiliki lebih dari dua kelas, terus dilakukan. Ada dua pilihan untuk mengimplementasikan *multi class* SVM yaitu dengan menggabungkan beberapa SVM biner atau menggabungkan semua data yang terdiri dari beberapa kelas ke dalam sebuah bentuk permasalah optimasi. Namun, pada pendekatan yang kedua permasalahan optimasi yang harus diselesaikan jauh lebih rumit. Berikut ini adalah metode yang umum digunakan untuk mengimplementasikan *multi class* SVM dengan pendekatan yang pertama:

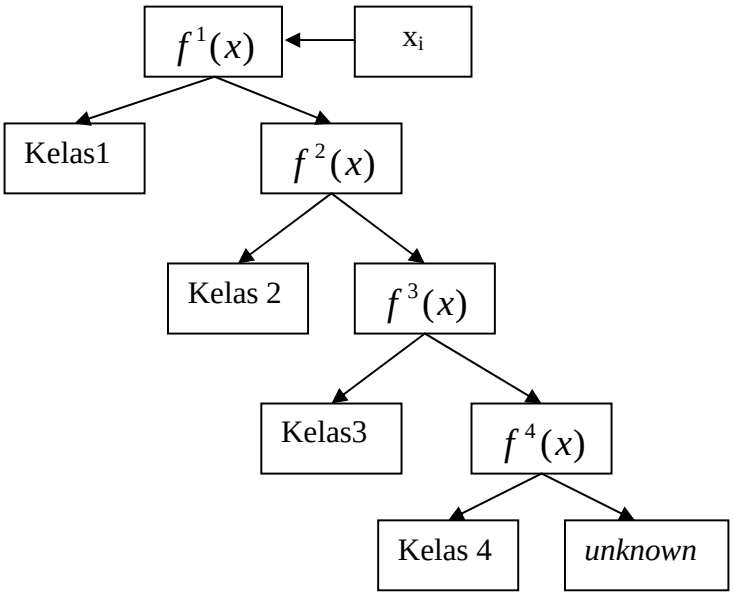
Metode "one-against-all"

Dengan menggunakan metode ini, dibangun k buah model SVM biner (k adalah jumlah kelas). Setiap model klasifikasi ke-i dilatih dengan menggunakan keseluruhan data, untuk mencari solusi permasalahan (2.16). Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner seperti pada tabel 1 dan penggunannya dalam mengklasifikasi data baru dapat dilihat pada 5.

$$\min_{w^i,b^i,\xi^i} \frac{1}{2}\left(w^i\right)^T w^i + C\sum_t \xi_t^i$$
$$s.t\ \left(w^i\right)^T \phi(x_t) + b^i \geq 1 - \xi_t^i \rightarrow y_t = i, \qquad (2.16)$$
$$\left(w^i\right)^T \phi(x_t) + b^i \geq -1 + \xi_t^i \rightarrow y_t \neq i,$$
$$\xi_t^i \geq 0$$

Tabel 1 Contoh 4 SVM biner dengan metode *One-against-all*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Bukan kelas 1	$f^1(x) = (w^1)x + b^1$
Kelas 2	Bukan kelas 2	$f^2(x) = (w^2)x + b^2$
Kelas 3	Bukan kelas 3	$f^3(x) = (w^3)x + b^3$
Kelas 4	Bukan kelas 4	$f^4(x) = (w^4)x + b^4$



Gambar 5 Contoh klasifikasi dengan metode O*ne-against-all*

Metode ”one-against-one”

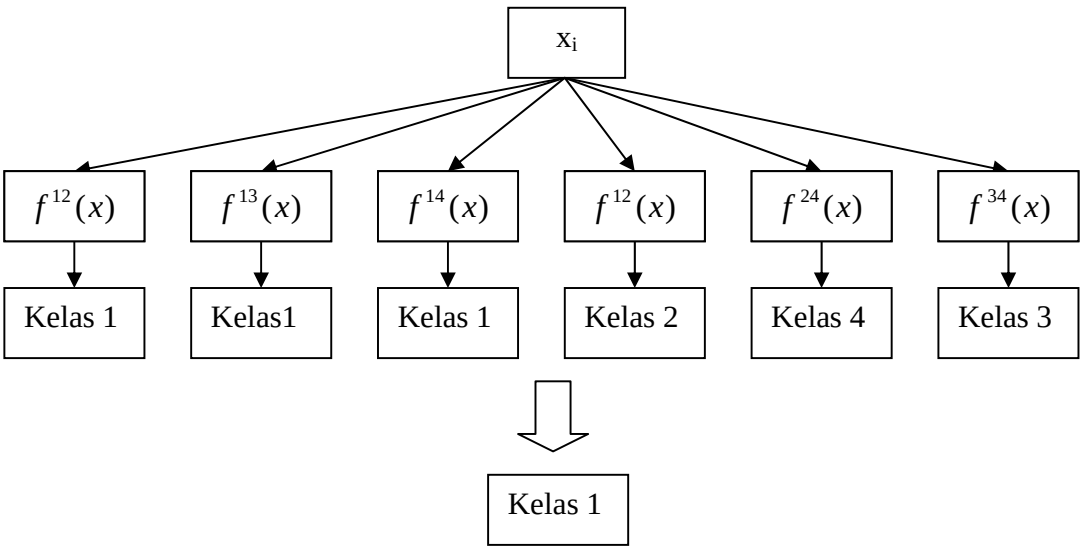
Dengan menggunakan metode ini, dibangun $\frac{k(k-1)}{2}$ buah model klasifikasi biner (k adalah jumlah kelas). Setiap model klasifikasi dilatih pada data dari dua kelas. Untuk data pelatihan dari kelas ke-i dan kelas ke-j, dilakukan pencarian solusi untuk persoalan optimasi konstrain sebagai berikut:

$$\begin{aligned}
&\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum_t \xi_t^{ij} \\
&s.t\ \left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i, \\
&\left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq -1 + \xi_t^{ij} \rightarrow y_t = j, \\
&\xi_t^{ij} \geq 0
\end{aligned} \qquad (2.17)$$

Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan $k(k-1)/2$ model klasifikasi selesai dibangun. Salah satunya adalah metode voting [HSU02].

Tabel 2 Contoh 6 SVM biner dengan metode *One-against-one*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Kelas 2	$f^{12}(x) = (w^{12})x + b^{12}$
Kelas 1	Kelas 3	$f^{13}(x) = (w^{13})x + b^{13}$
Kelas 1	Kelas 4	$f^{14}(x) = (w^{14})x + b^{14}$
Kelas 2	Kelas 3	$f^{23}(x) = (w^{23})x + b^{23}$
Kelas 2	Kelas 4	$f^{24}(x) = (w^{24})x + b^{24}$
Kelas 3	Kelas 4	$f^{34}(x) = (w^{34})x + b^{34}$



Gambar 6 Contoh klasifikasi dengan metode O*ne-against-one*

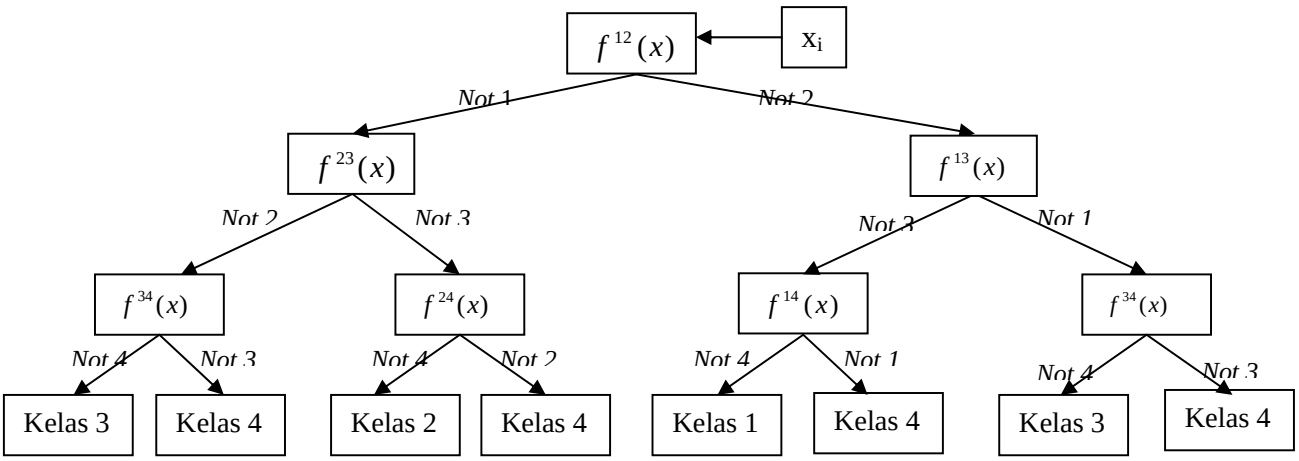
Jika data x dimasukkan ke dalam fungsi hasil pelatihan ($f(x) = \left(w^{ij}\right)^T \phi(x) + b$) dan hasilnya menyatakan menyatakan x adalah kelas i, maka suara untuk kelas i ditambah satu. Kelas dari data x akan ditentukan dari jumlah suara terbanyak. Jika terdapat dua buah kelas yang jumlah suaranya sama, maka kelas yang indeksnya lebih kecil dinyatakan sebagai kelas dari data. Jadi pada pendekatan ini terdapat $k(k-1)/2$ buah permasalahan *quadratic programming* yang masing-masing memiliki $2n/k$ variabel (n adalah jumlah data pelatihan). Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. Oleh karena itu, digunakan 6 buah SVM biner seperti pada tabel 2 dan contoh penggunaanya dalam memprediksi kelas data baru dapat dilihat pada gambar 6.

Metode DAGSVM (Directed Acyclic Graph Support Vector Machine)

Pelatihan dengan menggunakan metode ini sama dengan metode *one-against-one*, yaitu dengan membangun $\frac{k(k-1)}{2}$ buah model klasifikasi SVM biner. Akan tetapi, pada saat pengujian digunakan *binary directed acyclic graph*. Setiap node merupakan model SVM biner dari kelas ke-i dan kelas ke-j. Pada saat memprediksi kelas data pengujian, maka hipotesis dievaluasi mulai dari simpul akar, kemudian bergerak ke kiri atau ke kanan tergantung nilai output dari hipotesis.

Tabel 3 Contoh 6 SVM biner dengan metode DAGSVM

$y_i = 1$	$y_i = -1$	Hipotesis
Bukan Kelas 2	Bukan Kelas 1	$f^{12}(x) = (w^{12})x + b^{12}$
Bukan Kelas 3	Bukan Kelas 1	$f^{13}(x) = (w^{13})x + b^{13}$
Bukan Kelas 4	Bukan Kelas 1	$f^{14}(x) = (w^{14})x + b^{14}$
Bukan Kelas 3	Bukan Kelas 2	$f^{23}(x) = (w^{23})x + b^{23}$
Bukan Kelas 4	Bukan Kelas 2	$f^{24}(x) = (w^{24})x + b^{24}$
Bukan Kelas 4	Bukan Kelas 3	$f^{34}(x) = (w^{34})x + b^{34}$



Gambar 7 Contoh klasifikasi dengan metode DAGSVM

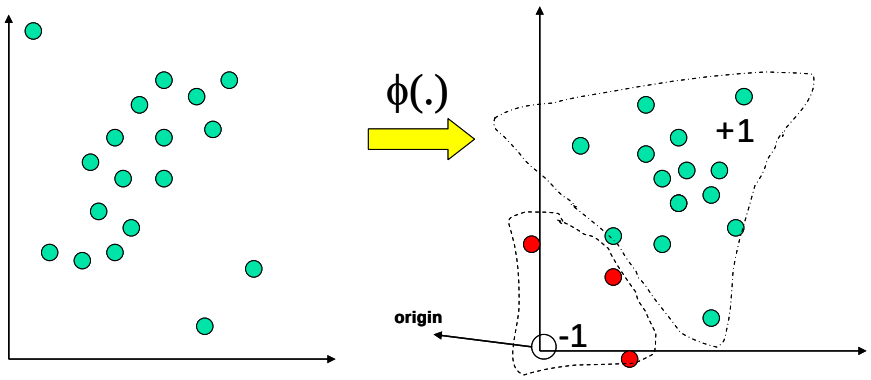
One Class SVM

One Class SVM adalah pengembangan dari SVM yang diusulkan oleh [SCH01], yang dapat digunakan untuk permasalahan *density estimation*. Dengan menggunakan teknik ini, SVM dapat digunakan pada dataset yang tidak memiliki label. Teknik O*ne Class* SVM mengidentifikasi *outlier* diantara contoh data positif dan menggunakannya sebagai contoh data negatif.

Persoalan *One Class* SVM ini dapat dirumuskan sebagai berikut: misalkan terdapat dataset yang memiliki *probability distribution* P dalam *feature space* dan kita ingin mengestimasi subset S pada *feature space* sehingga probabilitas sebuah data pengujian yang diambil dari P terletak di luar S, dibatasi oleh sebuah nilai v. Solusi dari permasalahan ini diperoleh dengan mengestimasi sebuah fungsi yang bernilai positif pada S dan negatif pada komplemen S. Dengan kata lain fungsi tersebut bernilai +1 pada sebuah area ”kecil” yang memuat hampir semua data dan bernilai -1 jika berada di luar area tersebut.

$$f(x) = \begin{cases} +1, & if\ x \in S \\ -1, & if\ x \notin S \end{cases} \qquad (2.18)$$

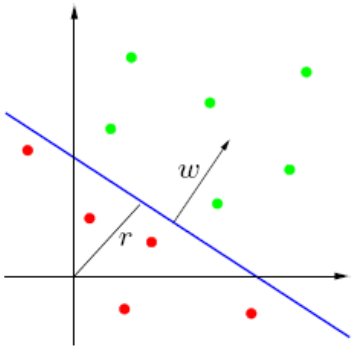
Prinsip dari teknik ini adalah mentransformasikan vektor input ke dalam *feature space* dengan menggunakan fungsi kernel, origin dianggap sebagai satu-satunya data negatif. Kemudian, dengan menggunakan ”*relaxation parameter*”, data yang bukan *outlier* dipisahkan dari origin. Selanjutnya prinsip kerja algoritma ini sama saja dengan klasifikasi biner pada SVM dengan tujuan mencari bidang pemisah terbaik yang memisahkan data dari origin dengan margin terbesar.



Gambar 8 Transformasi ke *feature space*

Persoalan pencarian *bidang pemisah* ini secara matematis adalah persamaan (2.19), dimana ξ_i adalah penalti terhadap data anomali yang terletak pada sisi *bidang pemisah* yang salah (sisi tempat data normal) dan v adalah parameter yang mengatur *trade off* antara memaksimalkan margin dari origin dan mencakup sebagian besar data pada daerah yang dibuat bidang pemisah dengan rasio *outlier* yang terdapat pada data pelatihan (seperti parameter C pada SVM untuk klasifikasi).

$$\min \frac{1}{2}|w|^2 + \frac{1}{vn}\sum_{i=1}^{n}\xi_i - r$$
$$s.t.\ \left(w.\phi(x_i)\right) \geq r - \xi_i, \qquad (2.19)$$
$$\xi_i \geq 0$$



Gambar 9 *One Class* SVM

Dengan menggunakan *lagrange multiplier* dan fungsi kernel maka persoalan di atas dapat dirumuskan sebagai berikut:

$$\min \sum_{i=1,j=1}^{n}\alpha_i\alpha_j K(x_i, x_j)$$
$$s.t.\sum_{i=1}^{n}\alpha_i = 1, \qquad (2.20)$$
$$0 \leq \alpha_i \leq \frac{1}{vn}$$

Dari hasil pelatihan, akan diperoleh nilai parameter α_i, kemudian nilai r dapat dihitung dari persamaan (2.21). Hasil dari proses pembelajaran adalah sebuah fungsi (2.22).

$$r = \sum \alpha_j K(x_i, x_j) \qquad (2.21)$$
$$f(x) = \sum_i \alpha_i K(x_i, x) - r,\ x_i = \textit{support vector} \qquad (2.22)$$

Algoritma Pelatihan SVM

Pelatihan pada SVM bertujuan untuk mencari solusi permasalahan optimasi konstain yang telah dijelaskan diatas (2.7,2.10,2.16,2.17,2.20). Untuk jumlah data yang sangat sedikit persoalan ini dapat diselesaikan dengan mudah. Misalkan $x_1 = 0, x_2 = 1$ dengan kelas $y \in -1,1$. Dengan mensubstitusi nilai x dan y ke persamaan (2.7) diperoleh:

$$\frac{1}{2}\alpha_1^2 - (\alpha_1 + \alpha_2)$$

$$= \frac{1}{2}[\alpha_1 \ \alpha_2]\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} - [1 \ 1]\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \quad \text{(Dalam notasi matriks)}$$

dengan konstrain: $\alpha_1 - \alpha_2 = 0, \alpha_1 \geq 0, \alpha_2 \geq 0$, maka:

$\frac{1}{2}\alpha_1^2 - 2\alpha_1 = 0$, diperoleh $\alpha_1 = \alpha_2 = 2$.

Bagaimana jika jumlah data yang diproses sangat besar? Berbagai teknik optimasi telah banyak dikembangkan, yang pada dasarnya secara interatif mencari solusi maksimum dari fungsi objektif. Akan tetapi, teknik-teknik ini memerlukan data disimpan pada memori dalam bentuk matriks kernel [CHR00]. Hal ini akan mengakibatkan kompleksitas data pelatihan meningkat dengan bertambahnya ukuran matriks sehingga penggunaan teknik ini dibatasi oleh jumlah data yang dapat diproses. Untuk dataset yang lebih besar digunakan teknik yang didasarkan pada metode '*working set*'.

Algoritma Decomposition

Metode yang banyak digunakan pada SVM saat ini adalah *decomposition* (misalnya *Sequential Minimal Optimization* (SMO), LibSVM, SVMLight) [LIN05] yang bekerja berdasarkan prinsip '*working set*'. Metode ini hanya mengubah beberapa *multiplier* α_i dalam jumlah tertentu pada setiap iterasi, sementara nilai yang lain bernilai tetap. *Working set* merupakan kumpulan variabel yang sedang dioptimasi pada *current iteration*. Jadi, prinsip *decomposition* adalah mengoptimasi masalah global dengan hanya menggunakan sebagian kecil data pada satu saat.

Teknik dekomposisi secara matematis dapat direpresentasikan dalam notasi matriks. Misalkan $\alpha = (\alpha_1, ...\alpha_l)^T$, $y = (y_1, ...y_l)^T$, $Q_{ij} = y_i y_j K(x_i, x_j)$, dan e merupakan vektor dengan jumlah elemen sebanyak l (jumlah data pelatihan) dan semuanya bernilai 1. Maka SVM *dual problem* dapat dituliskan sebagai berikut:

$$\begin{aligned} &\max_{\alpha} e^T\alpha - \frac{1}{2}\alpha^T Q\alpha \\ &s.t. 0 \leq \alpha_i \leq C, i = 1...l \\ &y^T\alpha = 0 \end{aligned} \qquad (2.23)$$

Misalnya vektor α dibagi menjadi α_B yang menyatakan variabel yang dimasukkan ke dalam working set, dan α_N merupakan variabel sisanya. Selanjutnya matrix Q dapat dipartisi menjadi $Q = \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$, dimana setiap bagiannya ditentukan oleh himpunan indeks B dan N. SMO menggunakan *working set* berelemen dua sehingga pencarian solusi optimal dapat dilakukan secara analitis seperti contoh sederhana diatas. Hal ini tentunya akan mengakibatkan jumah iterasi semakin bertambah, akan tetapi karena waktu yang dibutuhkan dalam setiap iterasi sangat kecil maka waktu total pelatihan menjadi lebih singkat.

Berikut ini adalah algoritma *decomposition* dengan menggunakan *working set* berelemen dua yang digunakan pada LibSVM:

1. Temukan variabel awal α^1 yang *feasible*, set k=1 (α^1 adalah vektor berisi semua nilai α).
2. Jika α^k merupakan titik stationer dari (2.23) , berhenti. Jika tidak, tentukan *working set* $B = \{i, j\}$
3. Jika $\alpha_{ij} \equiv K_{ij} + K_{jj} - 2K_{ij} > 0$

   selesaikan *sub-problem* α_B:

$$\min_{\alpha_i, \alpha_j} \frac{1}{2} \begin{bmatrix} \alpha_B^T & \left(\alpha_N^k\right)^T \end{bmatrix} \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} - \begin{bmatrix} e_B^T & e_N^T \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix}$$

$$= \frac{1}{2} \alpha_B^T Q_{BB} \alpha_B + (-e_B + Q_{BN} \alpha_N^k)^T \alpha_B + kons \tan$$

$$= \frac{1}{2} [\alpha_i \ \alpha_j] \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \left(-e_B + Q_{BN} \alpha_N^k\right)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + kons \tan \quad (2.24)$$

$$s.t. 0 \le \alpha_t \le C, t \in B$$

$$y_B^T \alpha_B^T = -y_N^T y_N^k$$

   Jika tidak, selesaikan:

$$\min_{\alpha_i,\alpha_j} \frac{1}{2}[\alpha_i \ \alpha_j] \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \left(-e_B + Q_{BN}\alpha_N^k\right)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix}$$

$$+ \frac{\tau - a_{ij}}{4}\left(\left(\alpha_i - {}_i^k\right)^2 + \left(\alpha_i - {}_j^k\right)^2\right)$$

s.t konstrain dari (2.20)

4. Set α_B^{k+1} sebagai solusi optimal dari *subproblem* pada langkah 3 dan $\alpha_N^{k+1} \equiv \alpha_N^k$. *Set* $k \leftarrow k+1$, lanjutkan ke langkah kedua.

α^k merupakan titik stationer dari (2.23), jika dan hanya jika $m(\alpha) \le M(\alpha)$ yang merupakan Karush-Kuhn-Tucker (KKT) *condition* [CHA01]. Dimana:

$$\begin{aligned} m(\alpha) &\equiv \max_{i \in I_{up}(\alpha)} -y_i \nabla f(\alpha)_i \\ M(\alpha) &\equiv \min_{i \in I_{low}(\alpha)} -y_i \nabla f(\alpha)_i \\ I_{up}(\alpha) &\equiv \left\{ t \mid \alpha_t < C, y_t = 1 \ \textit{atau} \ \alpha_t > 0, y_t = -1 \right\} \\ I_{low}(\alpha) &\equiv \left\{ t \mid \alpha_t < C, y_t = -1 \ \textit{atau} \ \alpha_t > 0, y_t = 1 \right\} \\ \nabla f(x) &\equiv Q\alpha + e \end{aligned} \qquad (2.25)$$

Untuk mentoleransi kondisi berhenti terdapat parameter tambahan ε ($m(\alpha) - M(\alpha) \le \varepsilon$), yang idealnya bernilai 0. Akan tetapi, dalam prakteknya dapat digunakan nilai yang lebih besar (misalnya pada LibSVM digunakan nilai *default* $\varepsilon = 10^{-3}$), sehingga jumlah iterasi menjadi lebih kecil.

Hal yang penting diperhatikan agar algoritma *decomposition* berjalan lebih cepat adalah pemilihan *working set* yang akan mengkibatkan solusi permasalahan global (2.19) lebih cepat dicapai. Nilai α yang dimasukkan ke dalam *working set* adalah varibel yang paling melanggar *KKT condition*. *Pseudocode* algoritma *decomposition* diatas beserta pemilihan *working set* dapat dilihat pada lampiran B.

Shrinking dan Caching

Strategi lainnya untuk mempercepat algoritma *decomposition* adalah *shrinking* dan *caching* yang pertama kali diperkenalkan oleh Joachim. *Shrinking* merupakan strategi heuristik yang memperkecil permasalahan pencarian solusi untuk persoalan optimasi diatas dengan mengabaikan beberapa *bounded support vector* ($\alpha = C$). Hal ini dapat dilakukan karena umumnya nilai *bounded support vector* setelah beberapa iterasi dapat diidentifikasi dan bernilai tetap sampai akhir iterasi [LIN05]. Akan tetapi, apabila solusi permalsahan dengan menerapkan shrinking bukan solusi optimal untuk (2.23) maka optimasi dilanjutkan dengan menggunakan keseluruhan variabel.

Seperti yang dilihat diatas *algoritma decomposition* melakukan iterasi sampai solusi persamaan (2.19) ditemukan. Dalam setiap iterasi nilai elemen matriks Q_{ij} digunakan. Agar tidak perlu dilakukan perhitungan ulang nilai Q_{ij} maka nilai Q_{ij} yang baru digunakan disimpan di memori sehingga waktu komputasi yang dibutuhkan menjadi jauh lebih singkat. Strategi ini disebut dengan *caching*.

Estimasi Parameter

Akurasi model yang akan dihasikan dari proses pelatihan dengan SVM sangat bergantung pada fungsi kernal serta parameter yang digunakan. Oleh karena itu performansinya dapat dioptimasi dengan mencari (mengestimasi) parameter terbaik. Ada beberapa cara yang dapat dilakukan antara lain *cross validation* (mudah digunakan), *leave-one-out* (akurat tetapi membutuhkan biaya komputasi yang tinggi) , dan $\xi\alpha$ -estimator [QUA02]. Cara yang ketiga merupakan modifikasi cara kedua yang diusulkan oleh Joachim [QUA02].

k-folds cross-validation dapat digunakan untuk menentukan nilai parameter C dan parameter kernel yang tidak overfit data pelatihan [HSU04]. Dengan metode ini, data yang diambil secara acak kemudian dibagi menjadi k buah partisi dengan ukuran yang sama. Selanjutnya, dilakukan iterasi sebanyak k. Pada setiap iterasi digunakan sebuah partisi sebagai data pengujian, sedangkan k-1 partisi sisanya digunakan sebagai data pelatihan. Jadi akan dicoba berbagai nilai parameter dan parameter terbaik ditentukan melalui *k-folds cross-validation*.

Pada tutorial [HSU04] dianjurkan supaya pada setiap iterasi digunakan penambahan parameter secara exponensial. Contohnya untuk kernel RBF nilai parameter yang digunakan pada setiap iterasi adalah $C = 2^{-5}, 2^{-3}, ..., 2^{15}, \gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$. Kemudian ditemukan nilai parameter terbaik misalnya $C = 2^{3}, \gamma = 2^{-5}$. Selanjutnya dilakukan pencarian parameter pada rentang nilai yang lebih kecil sehingga dapat parameter yang baik $C = 2^{3,25}, \gamma = 2^{-5,-25}$. Setelah nilai parameter terbaik ditemukan pelatihan dilakukan dengan menggunakan keseluruhan data. Pencarian nilai parameter ini disebut juga *grid search*.

SVM pada Imbalanced Dataset

Imbalanced dataset adalah dataset yang memiliki contoh kelas negatif (kelas mayoritas) jauh lebih banyak daripada contoh kelas positif (kelas minoritas) misalnya pada aplikasi *fraud detection* rasio *imbalance* dapat mencapai 100:1 dan bahkan bisa mencapai 100000:1 pada aplikasi lain. Banyak teknik klasifikasi yang akurasinya menurun ketika diterapkan pada dataset dengan *imbalance ratio* yang tinggi karena umumnya teknik tersebut didesain untuk mengeneralisasi data pelatihan dan menghasilkan hipotesis paling sederhana yang sesuai dengan data tersebut [AKB04]. Akan tetapi, hipotesis yang dihasilkan sering kali memprediksi hampir semua *instance* sebagai kelas yang menjadi mayoritas dalam data pelatihan. Selain itu data dari kelas minoritas mungkin dianggap *noise* sehingga mungkin saja diabaikan oleh teknik klasifikasi [AKB04].

SVM sendiri merupakan salah satu teknik yang tidak terlalu sensitif terhadap *imbalanced dataset* karena hipotesis yang dihasilkan hanya dipengaruhi oleh sebagian data yaitu data yang menjadi *support vector*. Ada beberapa alasan mengapa performansi SVM mungkin menurun jika diterapkan pada *imbalanced* dataset yaitu data dari kelas minoritas terletak jauh dari bidang pemisah yang ideal, kelemahan pada *soft-margin hyperplane* (jika nilai parameter C kecil maka SVM cenderung mengklasifikasikan data sebagai kelas mayoritas), dan rasio *support vector* yang tidak seimbang [AKB04].

Pendekatan yang populer untuk mengatasi masalah ini adalah memberikan bias kepada teknik klasifikasi sehingga lebih memperhatikan *instance* dari kelas minoritas. Hal ini dapat dilakukan melalui pemberian penalti yang lebih besar jika terjadi kesalahan dalam mengklasifikasikan kelas minoritas, dibandingkan jika salah mengklasifikasikan kelas mayoritas. Dengan demikian formula SVM diubah menjadi:

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}|w|^2 + C_+ \sum_{y_i=1} \xi_i + C_- \sum_{y_i=-1} \xi_i \\ &s.t.\ \ y_i\left(wx_i + b\right) \geq 1 - \xi_i, \\ &\xi_i \geq 0, i = 1,...,n \end{aligned} \qquad (3.1)$$

Implementasi modifikasi ini pada prinsipnya sama dengan implementasi SVM sebelumnya, penggunaan C_+ atau C_- hanyalah kasus khusus bergantung pada kelas *instance* data. Adapun perubahan utama terdapat pada pengubahan nilai *lagrange mulplier* selama pelatihan [CHA01]. Selain modifikasi pada teknik klasifikasi, dapat juga dilakukan pendekatan pada level data seperti melakukan *oversampling* kelas minoritas atau *undersampling* kelas mayoritas sehingga dihasilkan *balanced dataset* [AKB04].

Pemilihan Atribut penting (*feature selection*)

Pada pelatihan dengan menggunakan SVM tidak terdapat mekanisme untuk mengetahui atribut yang penting atau yang kurang penting. Atribut yang kurang penting umumnya tidak mempengaruhi efektifitas teknik klasifikasi. Oleh karena itu, jika atribut yang kurang penting ini dibuang maka efisiensi teknik klasifikasi akan meningkat. Efektifitas teknik klasifikasi juga dapat meningkat jika atribut yang kurang penting ini ternyata menjadi *noise*.

Pada SVM salah satu cara yang sederhana untuk mengetahui atribut yang penting adalah seperti yang dilakukan pada [MUK02B]. Pemilihan atribut penting dilakukan dengan mengevaluasi efektifitas dan efisiensi teknik klasifikasi setelah sebuah fitur dihilangkan. Dari hasil pengujian setelah fitur ini dihilangkan dapat diketahui sebuah fitur penting atau tidak dari perbedaan efisiensi dan efektifitas. Cara yang lain adalah dengan menggunakan f-*score* seperti pada [CHE03]. F-*score* adalah teknik sederhana untuk mengukur tingkat diskriminasi dua buah vektor

bilangan desimal. Jika terdapat vektor data pelatihan $x_k, k = 1,...,m$. Jika jumah data kelas positif adalah n_+ dan n_- adalah jumlah data kelas negatif. Maka F-*score* atribut ke i adalah :

$$F(i) \equiv \frac{\left(\bar{x}_i^{(+)} - \bar{x}_i\right)^2 + \left(\bar{x}_i^{(-)} - \bar{x}_i\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,i}^{(+)} - \bar{x}_i^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,i}^{(-)} - \bar{x}_i^{(-)}\right)^2} \quad (3.2)$$

dimana $\bar{x}_i, \bar{x}_i^{(+)}, \bar{x}_i^{(-)}$ adalah rata-rata dari nilai atribut keseluruhan data, data positif dan data negatif. $x_{k,i}^{(+)}$ adalah fitur ke-i dari data kelas positif ke-k dan sebaliknya $x_{k,i}^{(-)}$ adalah fitur ke-i dari date kelas negatif ke-k. Semakin besar nilai F(i), maka atribut ini dapat dianggap lebih diskriminatif (lebih penting).

Incremental Training dengan SVM

Pada SVM hanya support vector (data yang berada di perbatasan antar kelas) yang mempengaruhi fungsi keputusan hasil pelatihan. Karakteristik ini, dapat menyelesaikan masalah utama SVM yang menjadi lambat ketika dilatih dengan menggunakan data dalam jumlah yang sangat besar. Dengan demikian, kita dapat memilih kandidat support vector sebelum pelatihan, sehingga mengurangi jumlah data pelatihan dan pada akhirnya mengurangi kebutuhan penggunaan memori [KAT06].

Jika terdapat data pelatihan baru dan model hasil pelatihan yang lama ingin diperbaharui, maka dari data pelatihan lama cukup diambil sebagian data yang menjadi kandidat *support vector*. Kandidat *support vector* adalah data yang memenuhi $y(x) f(x) <= 1$. Selain itu, data yang memenuhi $y(x) f(x) > 1$ tetapi dekat dengan $y(x) f(x) = 1$ dapat menjadi support vector [KAT06]. Oleh karena itu kita dapat menentukan apakah sebuah data merupakan kandidat support vector dengan cara memeriksa data yang memenuhi :

$$y(x) f(x) <= \beta + 1 \quad (3.4)$$

$$y(x) f(x) \geq 1 \quad (3.5)$$

dimana β adalah parameter yang ditentukan pengguna. Jika data tidak memenuhi persamaan (3.3) maka data tersebut dihapus. Akan tetapi jika semua data memenuhi (3.4) maka fungsi

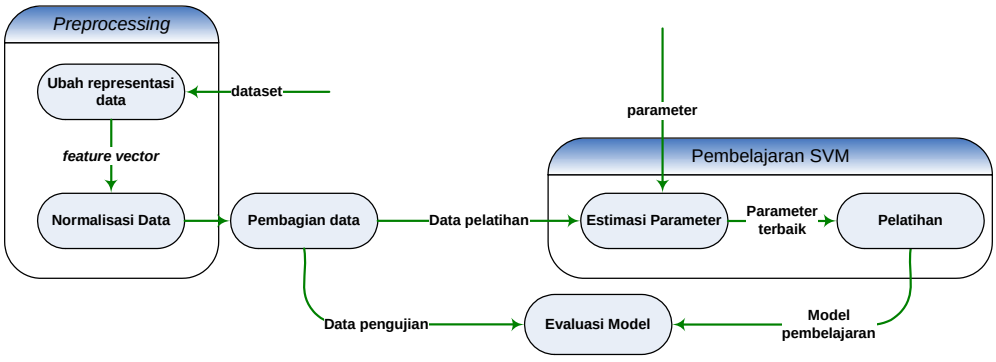
keputusan setelah pelatihan tidak berubah sehingga hanya data yang memenuhi (3.3) yang perlu ditambahkan pada pelatihan berikutnya.

Misalkan kita mempunyai dataset awal Xa dan dataset tambahan Xb. Algoritma untuk *incremental training* [KAT06] adalah sebagai berikut:

a. Lakukan pelatihan SVM dengan menggunakan Xa
b. Tambahkan dataset Xb sehingga $X_a = X_a \cup X_b$
c. If for $x \in X_a$, (3.4) tidak dipenuhi, buang data x $X_a = X_a - \{x\}$
d. If for $x \in X_a$, (3.5) tidak dipenuhi, lakukan pelatihan ulang SVM
e. Ulangi langkah b, c dan d.

Pembelajaran Dengan SVM

Penggunaan SVM baik dalam bentuk *supervised* maupun *unsupervised* pada prinsipnya menyelesaikan sebuah permasalahan *quadratic programming*. Oleh karena itu, proses pembelajarannya hampir sama dan tahapannya dapat dilihat pada gambar III-3. Akan tetapi untuk *unsupervised learning* dengan SVM data pelatihan dan data pengujian adalah data yang sama. Selain itu, untuk proses pelatihannya dapat juga hanya menggunakan sebagian data dari data pengujian sehingga proses waktu pelatihan menjadi lebih singkat, tetapi hal ini mungkin menurunkan akurasi pada tahap pengujian.



Gambar 10 Pembelajaran dengan SVM

Preprocessing Data

Data KDDCUP 99 memiliki atribut yang tipe nilainya *real* dan *discrete*. Agar *dataset* ini dapat diproses oleh SVM maka dataset tersebut harus direpresentasikan sebagai vektor dari bilangan *real*. Cara yang umum digunakan untuk data diskrit yang tidak terurut (data berupa kategori) adalah dengan memakai *1-of-c encoding*, dimana c adalah banyaknya kemungkinan nilai pada input tersebut [SAR02]. Misalnya untuk nilai atribut *protocol_type* : tcp,udp,icmp. Ketiganya tidak tepat jika di-*encode* ke nilai real yang berurutan karena tidak memiliki urutan besar-kecil. Pendekatan yang biasa dilakukan adalah memakai vektor berdimensi 3, bernilai 0 dan 1. Pada contoh ini tcp direpresentasikan sebagai [001], udp direpresentasikan sebagai [010] dan icmp direpresentasikan sbg. [100]. Vektor berdimensi 3 ini dalam implementasinya direpresentasikan dalam 3 *binary* atribut (masing-masing merepresentasikan satu elemen vektor). Cara ini akan digunakan pada tugas akhir ini.

Selain mengkonversi nilai atribut , normalisasi / *scaling* (perubahan rentang dari nilai atribut) atribut yang bernilai bilangan real umumnya juga diperlukan agar proses perhitungan lebih mudah dan atribut yang rentang nilainya besar tidak mendominasi atribut yang rentang nilainya lebih kecil. Normalisasi umumnya memberikan hasil yang lebih baik [HSU04]. Rentang nilai atribut yang dianjurkan untuk digunakan adalah [0,1] atau [-1,+1] [HSU04].
Normalisasi dalam rentang nilai [0,1] menghasilkan akurasi yang sama dengan [-1,+1] jika menggunakan fungsi kernel RBF, tetapi waktu komputasi yang dibutuhkan mungkin berbeda. Karena nilai atribut data pada KDD CUP 99 banyak yang bernilai 0, maka digunakan rentang normalisasi [0,1] karena waktu yang akan dibutuhkan menjadi lebih singkat dibandingkan dengan penggunaan rentang nilai [-1,+1]. Alasannya, jika dinormalisasi ke dalam rentang nilai [-1,+1] banyak nilai 0 akan berubah menjadi -1 sehingga perhitungannya menjadi lebih lambat. Adapun rumus normalisasi ke dalam rentang [0,1] adalah persamaan (3.1) dimana $x_{\min}$ dan $x_{\max}$ adalah nilai minimum dan nilai maksimum atribut x.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad (3.1)$$

Pemilihan dan Estimasi Parameter Terbaik

Sebelum pelatihan dilakukan, terlebih dahulu harus ditentukan nilai parameter C beserta fungsi kernel dan parameternya untuk pembelajaran *supervised* SVM. Untuk *One Class* SVM parameter yang harus ditentukan adalah parameter nu (v) beserta fungsi kernel dan parameternya. Menurut penelitian [LAS05], dataset KDD CUP paling baik diproses oleh algoritma *nonliniear* sehingga pada Tugas Akhir ini akan diterapkan SVM *nonliniear*. Untuk itu fungsi kernel yang akan digunakan adalah fungsi kernel RBF karena digunakan pada penelitian [MUK02,LAS04,LAS05].

Pada tugas akhir ini estimasi parameter terbaik akan dilakukan dengan mengunakan *k-folds crossalidation* dan *grid search*. Pendekatan ini mudah digunakan pada dataset yang jumlahnya ribuan. Akan tetapi, dataset yang akan digunakan pada Tugas Akhir ini ukurannya sangat besar, maka akan digunakan subset dari dataset yang diambil secara acak, kemudian dilakukan *grid search* pada subset tersebut. Setelah ditemukan parameter terbaik, dilakukan *grid search* pada rentang nilai yang lebih kecil untuk memperoleh parameter terbaik.

Daftar Referensi

[AKB04] Akbani, Rehan et. Al. 2004. *Applying Support Vector Machine to Imbalanced Datasets.* 2004. *Proceedings of* ECML-04.

[BUR98] Burges, Christopher. *A Tutorial On Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, 2(2):955-974. 1998.

[CHA01] Chang, Chih-Chung, Chih-Jen Lin. *LIB*SVM*: A Library for Support Vector Machines*. 2001.

[CHE03] Chen, Yi-Wei dan Chih-jen Lin. *Combining SVM with Various Features Selection Strategies*. 2003. Department of Computer Science, National Taiwan University

[CHR00] Christianini, Nello dan John S. Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000

[CHR01] Christianini, Nello. *Support Vector and Kernel Machines*. ICML tutorial, 2001.

[FAN05] Fan, Rong-En. et. al. 2005. *Working set Selection using Second Order Information for Training Support Vector Machines*. *Journal of Machine Learning Research* 6 1889-1918, 2005.

[HSU02] Hsu, Chih-Wei, Chih-Jen Lin. *A Comparison of Methods for Multi-class Support Vector Machines*. IEEE Transactions on Neural Networks, 13(2):415-425.2002.

[HSU04] Hsu, Chih-Wei et al. *A Practical Guide to Support Vector Classification.* Department of Computer Science and Information Engineering, National Taiwan University. 2004.

[LAS04] Laskov, Pavel et. Al. *Intrusion detection in unlabeled data with quarter-sphere Support Vector machines*. Fraunhofer-FIRST.IDA. 2004.

[KAT06] Katagiri, Shinya dan Shigeo Abe. *Incremental Training of Support Vector Machines Using Hyperspheres*. Graduate School of Science and technology, Kobe University. 2006.

[LIN05] Lin, Chih-Jen. 2005. *Optimization, Support Vector Machines, and Mahine Learning*. *http://www.csie.ntu.edu.tw/%7Ecjlin/papers/rome.pdf*. Diakses tanggal 10 Januari 2007.

[MUK02B] Mukkamala, S. et al. *Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines*. 2003.

[OSU97] Osuna, Edgar E. et. al.1997. *Support Vector machines: Training and Applications.* MIT, 1997.

[QUA02] Quang, Anh tran et al.*Evolving Support Vector Machine Parameters*. Slide presentasi ICML. Tsinghua University, 2002.

[SAR02] Sarle, Warren et.al. 2002. Neural Network FAQ. ftp://ftp.sas.com/pub/neural/FAQ2.html Diakses tanggal 14 Desember 2006.

[SCH01] Schölkopf, Bernhard. *Estimating the Support of High-Dimensional Distribution*. *Neural Computation* 13,1443-1471. MIT. 2001.

[VIS05] Visa, Sofia dan Anca Ralescu. *Issues in Mining Imbalanced Data Sets- A Review Paper*. 2005.

[WWW06B] LIBSVM FAQ. *http://www.csie.ntu.edu.tw/%7Ecjlin/libsvm/faq.htm*. Diakses tanggal 16 November 2006.