Tutorial Support Vector Machines

Budi Santosa
Profesor di Teknik Industri, ITS
Kampus ITS, Sukolilo Surabaya
E-mails: budi_s@ie.its.ac.id

1. Ide Dasar Support Vector Machine

Support vector machine (SVM) adalah suatu teknik yang relatif baru (1995) untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi, yang sangat populer belakangan ini. SVM berada dalam satu kelas dengan ANN dalam hal fungsi dan kondisi permasalahan yang bisa diselesaikan. Keduanya masuk dalam kelas supervised learning, dimana dalam implementasinya perlu adanya tahap training dan disusul tahap testing.

Baik para ilmuwan maupun praktisi telah banyak menerapkan teknik ini dalam menyelesaikan masalah-masalah nyata dalam kehidupan sehari-hari. Penerapannya antara lain dalam masalah gene expression analysis, prediksi finansial, cuaca hingga bidang kedokteran. Terbukti dalam banyak implementasi, SVM memberi hasil yang lebih baik dari ANN, terutama dalam hal solusi yang dicapai. ANN menemukan solusi berupa *local optimal*, sedangkan SVM menemukan solusi yang global optimal. Tidak heran bila kita menjalankan ANN, solusi dari setiap training hampir selalu berbeda. Hal ini disebabkan solusi local optimal yang dicapai tidak selalu sama. SVM selalu mencapi solusi yang sama untuk setiap running. Dalam teknik ini, kita berusaha untuk menemukan fungsi pemisah (klasifier) yang optimal yang bisa memisahkan dua set data dari dua kelas yang berbeda Vapnik (1995).

Teknik ini menarik orang dalam bidang data mining maupun machine learning karena performansinya yang meyakinkan dalam memprediksi kelas suatu data baru. Kita akan memulai pembahasan dengan kasus klasifikasi yang secara linier bisa dipisahkan. Dalam hal ini fungsi pemisah yang kita cari adalah fungsi linier. Fungsi ini bisa didefinisikan sebagai

$$g(x) := sgn(f(x)) \tag{8.1}$$

dengan $f(x) = w^T x + b$,

$x, w \in R^n$ and $b \in R$. Masalah klasifikasi ini bisa dirumuskan sebagai berikut: kita ingin menemukan set parameter (w, b) sehingga $\text{sgn}(f(x_i)) = \text{sgn}(< w,x > + b) = y_i$ untuk semua *i*. Fungsi sgn digunakan untuk mengelompokkan semua nilai di atas 0 menjadi +1 dan dibawah 0 menjadi -1. Dalam teknik ini kita berusaha menemukan *fungsi pemisah* (klasifier/hyperplane) terbaik diantara fungsi yang tidak terbatas jumlahnya untuk memisahkan dua macam obyek. *Hyperplane* terbaik adalah *hyperplane* yang terletak di tengah-tengah antara dua set obyek dari dua kelas. Mencari hyperplane terbaik ekuivalen dengan memaksimalkan margin atau jarak antara dua set obyek dari kelas yang berbeda. Jika $wx_1 + b = +1$ adalah *hyperplane-pendukung* (supporting hyperplane) dari kelas +1, dan $wx_2 + b = -1$ adalah hyperplane-pendukung dari kelas -1, margin antara dua kelas dapat dihitung dengan mencari jarak antara kedua hyperplane-pendukung dari kedua kelas. Secara spesifik, margin dihitung dengan cara berikut

$wx_1 + b = +1$

$wx_2 + b = -1$

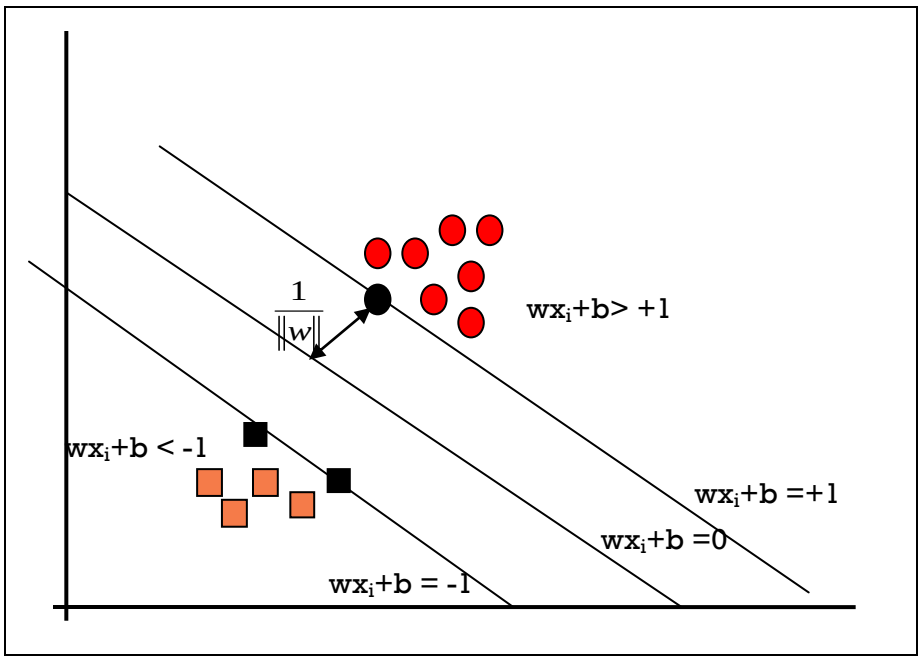
$w(x_1 - x_2) = 2$

$$\rightarrow \frac{w(x_1 - x_2)}{\| w \|} = \frac{2}{\| w \|}.$$

.

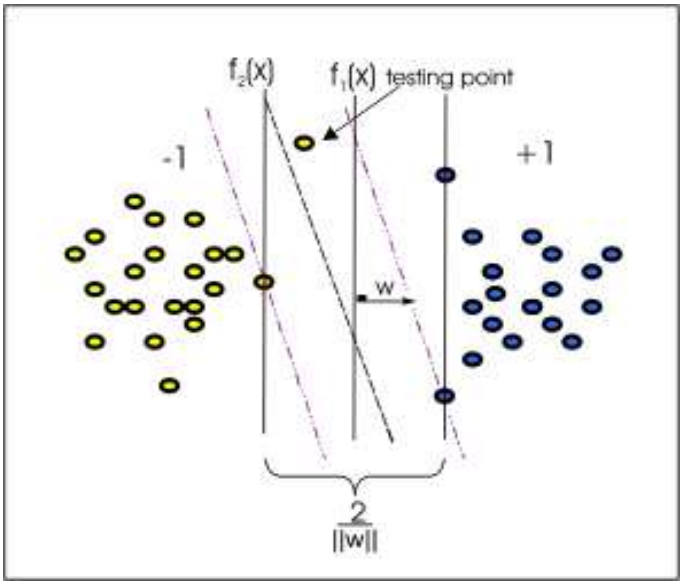
Gambar 8.1 memperlihatkan bagaimana SVM bekerja untuk menemukan suatu fungsi pemisah dengan margin yang maksimal. Untuk membuktikan bahwa memaksimalkan margin antara dua set obyek akan meningkatkan probabilitas pengelompokkan secara benar dari data testing.

Pada dasarnya jumlah fungsi pemisah ini tidak terbatas banyaknya. Misalkan dari jumlah yang tidak terbatas ini kita ambil dua saja, yaitu $f_1(x)$ and $f_2(x)$ (lihat gambar 8.2). Fungsi f_1 mempunyai margin yang lebih besar dari pada fungsi f_2. Setelah menemukan dua fungsi ini, suatu data baru masuk dengan keluaran −1. Kita harus mengelompokkan apakah data ini ada dalam kelas −1 atau +1 menggunakan fungsi pemisah yang sudah kita temukan.



Gambar 8.1: Mencari fungsi pemisah yang optimal untuk obyek yang bisa dipisahkan secara linier

Dengan menggunakan f_1, kita akan kelompokkan data baru ini di kelas −1 yang berarti kita benar mengelompokkannya. Sekarang kita coba gunakan f_2, kita akan menempatkannya di kelas +1 yang berarti salah. Dari contoh sederhana ini kita lihat bahwa memperbesar margin bisa meningkatkan probabilitas pengelompokkan suatu data secara benar.



Gambar 8.2: Memperbesar margin bisa meningkatkan probabilitas pengelompokkan suatu data secara benar.

2. Formulasi Matematis

Secara matematika, formulasi problem optimisasi SVM untuk kasus klasifikasi linier di dalam *primal space* adalah

$$\min_{w,b} \ \frac{1}{2} \| w \|^2$$
$$\text{subject to} \qquad (8.2)$$
$$y_i \, (wx_i + b) \geq 1 \, , \qquad i = 1,...m$$

dimana x_i adalah data input, y_i adalah output yang nilainya =1 atau -1, w dan b adalah parameter yang kita cari nilainya. Dalam formulasi di atas, kita ingin meminimalkan fungsi tujuan (obyektif function), $\frac{1}{2} \| w \|^2$, atau memaksimalkan kuantitas $\frac{1}{\| w \|}$ atau $w^T w$ dengan memperhatikan pembatas $y_i(wx_i + b) \geq 1$. Bila output data $y_i = +1$, maka pembatas menjadi $(wx_i + b) \geq 1$.

Sebaliknya bila $y_i = -1$, pembatas menjadi $(wx_i + b) \leq -1$. Di dalam kasus yang tidak *feasible* (infeasible) dimana beberapa data mungkin tidak bisa dikelompokkan secara benar, formulasi matematikanya menjadi berikut

min

$$\min_{w,b,t} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m} t_i$$

$$\text{subject to} \qquad (8.3)$$

$$y_i (wx_i + b) + t_i \geq 1 \qquad t_i \geq 0 \quad i = 1,...m$$

dimana t_i adalah variabel slack (Haykin, 1999). Dalam formulasi ini kita berusaha meminimalkan kesalahan klasifikasi (misclassification error) yang dinyatakan dengan adanya variabel slack t_i, sementara dalam waktu yang sama kita memaksimalkan margin. Penggunaan variabel slack t_i adalah untuk mengatasi kasus ketidaklayakan (infeasibility) dari pembatas (constraints) $y_i(wx_i+b) \geq 1$ dengan cara memberi pinalti untuk data yang tidak memenuhi pembatas tersebut. Untuk meminimalkan nilai t_i ini, kita berikan pinalti dengan menerapkan konstanta ongkos *C*. Vektor *w* tegak lurus terhadap fungsi pemisah: $wx+b = 0$. Konstanta b menentukan lokasi fungsi pemisah relatif terhadap titik asal (origin). Problem (8.3) adalah *programa nonlinear*. Ini bisa dilihat dari *fungsi tujuan* (objective function) yang berbentuk kuadrat. Untuk menyelesaikannya, secara komputasi agak sulit dan perlu waktu lebih panjang. Untuk membuat masalah ini lebih mudah dan efisien untuk diselesaikan, masalah ini bisa kita transformasikan ke dalam dual space. Untuk itu, pertama kita ubah problem (8.3) menjadi *fungsi Lagrangian* :

$$J(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i [y_i (w^T x_i + b) - 1] \qquad (8.4)$$

dimana variabel *non-negatif* α_i, dinamakan *Lagrange multiplier*. Solusi dari problem optimisasi dengan pembatas seperti di atas ditentukan dengan mencari saddle point dari fungsi Lagrangian $J(w, b, \alpha)$. Fungsi ini harus diminimalkan terhadap variabel *w* dan *b* dan harus dimaksimalkan terhadap variabel α. Kemudian kita cari turunan pertama dari fungsi $J(w, b, \alpha)$ terhadap variabel *w* dan *b* dan kita samakan dengan 0. Dengan melakukan proses ini, kita akan mendapatkan dua kondisi optimalitas berikut:

1. Kondisi 1:

$$\frac{\partial J(w,b,\alpha)}{\partial w} = 0$$

2. Kondisi 2:

$$\frac{\partial J(w,b,\alpha)}{\partial b} = 0$$

Penerapan kondisi optimalitas 1 pada fungsi Lagrangian (8.4) akan menghasilkan

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{8.5}$$

Penerapan kondisi optimalitas 2 pada fungsi Lagrangian (8.4) akan menghasilkan

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{8.6}$$

Menurut duality theorem (Bertsekas, 1995):

1. Jika problem primal mempunyai solusi optimal, maka problem dual juga akan mempunyai solusi optimal yang nilainya sama

2. Bila w_o adalah solusi optimal untuk problem primal dan α_o untuk problem dual, maka perlu dan cukup bahwa w_o solusi layak untuk problem primal dan

$$\Phi(w_0) = J(w_0, b_0, \alpha_0) = \min_w J(w,b,\alpha)$$

Untuk mendapatkan problem dual dari problem kita, kita jabarkan persamaan (8.4) sebagai berikut:

$$J(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i y_i w^T x_i - b \sum_{i=1}^{m} \alpha_i y_i - \sum_{i=1}^{m} \alpha_i \tag{8.7}$$

Menurut kondisi optimalitas ke dua dalam (8.6), term ketiga sisi sebelah kanan dalam persamaan di atas sama dengan 0. Dengan memakai nilai-nilai *w* di (8.5), kita dapatkan

$$w^T w = \sum_{i=1}^{m} \alpha_i y_i w^T x_i = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{8.8}$$

maka persamaan 8.7 menjadi

$$Q(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{8.9}$$

Selanjutnya kita dapatkan formulasi dual dari problem (8.3):

$$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

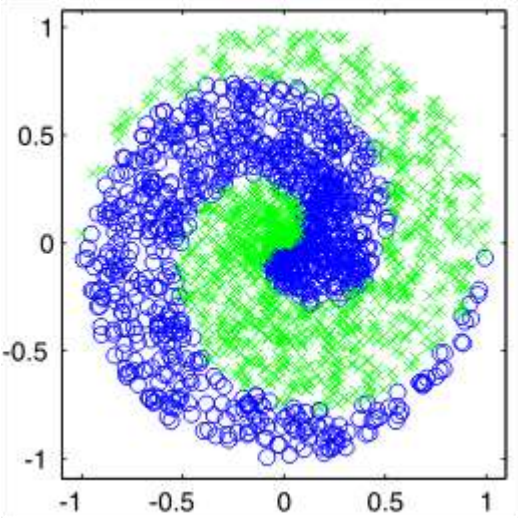
Subject to (8.10)

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ , \mathrm{i} = 1, ..\mathrm{m}$$

Formulasi (8.10) adalah quadratic programming (QP) dengan pembatas (constraint) linier. Melatih SVM ekuivalen dengan menyelesaikan problem *convex optimization*. Karena itu solusi dari SVM adalah unik dan global optimal. Hal ini berbeda dengan solusi neural networks (Haykin, 1999) yang ekuivalen dengan problem *nonconvex optimization* dengan akibat solusi yang ditemukan adalah *local optima*.

3. Metoda Kernel

Banyak teknik data mining atau machine learning yang dikembangkan dengan asumsi kelinieran. Sehingga algorithma yang dihasilkan terbatas untuk kasus-kasus yang linier. Karena itu, bila suatu kasus klasifikasi memperlihatkan ketidaklinieran, algorithma seperti perceptron tidak bisa mengatasinya. Secara umum, kasus-kasus di dunia nyata adalah kasus yang tidak linier. Sebagai contoh, perhatikan Gambar 8.3, data ini sulit dipisahkan secara linier. Metoda kernel (Scholkopf and Smola, 2002) adalah salah satu untuk mengatasinya. Dengan metoda kernel, suatu data x di input space dimapping ke feature space F dengan dimensi yang lebih tinggi melalui map φ sebagai berikut $\varphi : x \rightarrow \varphi(x)$. Karena itu data x di input space menjadi $\varphi(x)$ di feature space.



Gambar 8.3: Data spiral yang menggambarkan ketidaklinieran

Sering kali fungsi $\varphi(x)$ tidak tersedia atau tidak bisa dihitung, tetapi *dot product* dari dua vektor dapat dihitung baik di dalam *input space* maupun di *feature space*. Dengan kata lain, sementara $\varphi(x)$ mungkin tidak diketahui, *dot product* $< \varphi(x_1), \varphi(x_2) >$ masih bisa dihitung di *feature space*. Untuk bisa memakai metoda kernel, fungsi tujuan dan pembatas (constraint) perlu diekspresikan dalam bentuk dot product dari vektor data x_i. Sebagai konsekuensi, fungsi tujuan yang menjelaskan permasalahan dalam klasifikasi harus diformulasikan kembali sehingga menjadi bentuk dot product. Dalam feature space ini dot product $< . >$ menjadi $< \varphi(x), \varphi(x)' >$. Suatu fungsi kernel, $k(x, x')$, bisa digunakan untuk menggantikan dot product $< \varphi(x), \varphi(x)' >$. Kemudian di feature space, kita bisa menemukan suatu fungsi pemisah yang linier yang mewakili fungsi nonlinear di input space. Gambar 8.4 mendeskripsikan suatu contoh feature mapping dari ruang dua dimensi ke feature space dua dimensi. Dalam input space, data tidak bisa dipisahkan secara linier, tetapi kita bisa memisahkan di feature space. Karena itu dengan memetakan data ke feature space menjadikan tugas klasifikasi menjadi lebih mudah (Schölkopf and Smola, 2002).

Dengan memetakan setiap data ke feature space, maka formulasi 8.10, akan menjadi

$$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \varphi(x_i)^T \varphi(x_j)$$

Subject to (8.11)

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ , \mathrm{i} = 1, ..\mathrm{N}$$

Dalam formulasi 8.11, dot product antara $\varphi(x_i)^T \varphi(x_j)$ bisa diwakili oleh $k(x_i,x_j)$, dimana *k* adalah fungsi kernel. Selanjutnya sesudah kita melatih SVM dan ditemukan *alpha* dan *b* maka kita akan dapatkan $f(x) = \sum_{i=1}^{N} \alpha^*_i \ y_i k(x_i, x) + b^*$, fungsi pemisah optimal adalah $g(x) = sign(\sum_{i=1}^{N} \alpha^*_i \ y_i k(x_i, x) + b^*)$, dimana α^*_i, i = 1, .., N adalah *solusi optimal* dari problem (8.11) dan b^* dipilih sehingga $y_i f(x_i) = 1$ untuk sembarang i dengan $C > \alpha^*_i > 0$ (Cristianini and Shawe-Taylor, 2000).

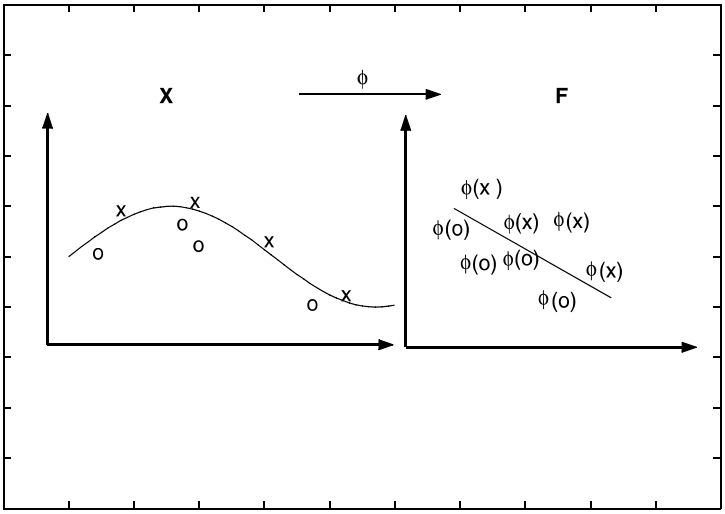
Data x_i, dimana $\alpha^*_i > 0$ dinamakan *support vector* dan menyatakan data training yang diperlukan untuk mewakili fungsi keputusan yang optimal. Dalam gambar 8.1, sebagai contoh, 3 titik berwarna hitam (2 kotak, 1 lingkaran) menyatakan *support vector*.

Secara lebih detail b^* dihitung dengan cara

$$b^* = \frac{1}{nsv}\sum_{sv}(y_{sv} - k(x_{sig}, x_{sv}) * \alpha_{sv} * y_{sig})$$

dimana *nsv* adalah banyaknya support vector, *sv* adalah indeks untuk support vector dan *sig* menunjukkan data yang berada dalam margin, dimana $0<\alpha_i<C$.

Untuk mengatasi masalah ketidaklinieran (nonlinearity) yang sering terjadi dalam kasus nyata, kita bisa menerapkan metoda kernel. Metoda kernel (Scholkopf and Smola, 2002) memberikan pendekatan alternatif dengan cara melakukan mapping data x dari input space ke *feature space F* melalui suatu fungsi φ sehingga $\varphi : (x) \rightarrow \varphi(x)$. Karena itu suatu titik x dalam input space menjadi $\varphi(x)$ dalam feature space.



Gambar 8.4: Suatu kernel map mengubah problem yang tidak linier menjadi linier dalam space baru.

Fungsi kernel yang biasanya dipakai dalam literatur SVM (Haykin, 1999):

(1) linier: $x^T x$,

(2) polynomial: $(x^T x + 1)^p$,

(3) radial basis function (RBF): $\exp(-\frac{1}{2\sigma^2} \| x - x \|^2)$,

(4) tangent hyperbolic (sigmoid) : $tanh(\beta_0 x^T x_i + \beta_1)$.

Dalam fungsi tersebut ada beberapa parameter yang nilainya ditentukan oleh user (pemakai). Misalnya dalam polynomial, *p* bisa diberi nilai 2, 3. atau nilai lain berupa bilangan bulat positif. Dalam kernel RBF, nilai σ bisa diberi nilai bilangan riil positif. Fungsi kernel mana yang harus digunakan untuk subtitusi dot product di feature space sangat bergantung pada data. Biasanya metoda *cross-validation* (Hastie et al. , 2001) digunakan untuk pemilihan fungsi kernel ini. Pemilihan fungsi kernel yang tepat adalah hal yang sangat penting. Karena fungsi kernel ini akan menentukan feature space di mana fungsi klasifier akan dicari. Sepanjang fungsi kernelnya legitimate, SVM akan beroperasi secara benar meskipun kita tidak tahu seperti apa map yang digunakan.

Fungsi kernel yang legitimate diberikan oleh Teori Mercer (Vapnik, 1995) dimana fungsi itu harus memenuhi syarat: kontinus dan positive definite. Lebih mudah menemukan fungsi kernel daripada mencari map φ seperti apa yang tepat untuk melakukan mapping dari *input space* ke *feature space*. Pada penerapan metoda kernel, kita tidak perlu tahu map φ apa yang digunakan untuk satu per satu data, tetapi lebih penting mengetahui bahwa dot produk dua titik di feaure space bisa digantikan oleh fungsi kernel. Selama ini orang lebih banyak menggunakan kernel RBF karena hanya ada satu parameter yang perlu dimasukkan. Bagaimana mencari nilai parameter kernel yang optimal menjadi obyek penelitian yang menarik.

Untuk ilustrasi bagaimana SVM bekerja, mari kita ikuti dua contoh berikut. Satu adalah contoh dimana data yang ada bisa dipisahkan secara linier. Untuk contoh ini kita gunakan problem *AND*. Contoh yang kedua adalah contoh untuk problem yang tidak bisa dipisahkan secara linier. Untuk contoh ini kita gunakan problem *Exclusive OR (XOR)*. Problem AND adalah klasifikasi dua kelas dengan empat data (lihat Tabel 8.1). Karena ini problem linier, kernelisasi tidak diperlukan.

Tabel 8.1: AND Problem

No	x_1	x_2	Y

1	1	1	1
2	-1	1	-1
3	1	-1	-1
4	-1	-1	-1

Menggunakan data di Tabel 8.1, kita dapatkan formulasi masalah optimisasi sebagai berikut:

$$min \ \frac{1}{2} (w_1^2 + w_2^2) + C (t_1 + t_2 + t_3 + t_4)$$

Subject to

$$w_1 + w_2 + b + t_1 \geq 1$$

$$w_1 - w_2 - b + t_2 \geq 1 \qquad (8.12)$$

$$-w_1 + w_2 - b + t_3 \geq 1$$

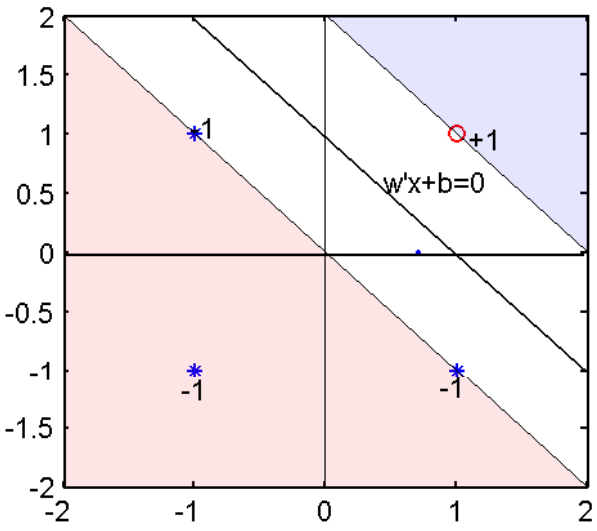
$$w_1 + w_2 - b + t_4 \geq 1$$

$$t_1, t_2, t_3, t_4 \geq 0$$

Karena fungsi AND adalah kasus klasifikasi linier, dimana data bisa dipisahkan secara linier maka bisa dipastikan nilai variabel slack $t_i = 0$. Jadi kita bisa masukkan nilai C = 0. Setelah menyelesaikan problem optimasi di atas didapat solusi :

$w_1 = 1$, $w_2 = 1$, $b = -1$.

Dengan nilai *w* dan *b* yang kita dapatkan ini, maka persamaan fungsi pemisahnya adalah $f(x) = w_1 x_1 + w_2 x_2 + b = x_1 + x_2 - 1$. Untuk menentukan output atau label dari setiap titik data/obyek kita gunakan fungsi $g(x) = sign(x)$. Dengan fungsi sign ini semua nilai $f(x) < 0$ diberi label −1 dan lainnya diberi label +1. Secara grafis fungsi pemisah ini diperlihatkan dalam Gambar 8.5.



Gambar 8.5: Ilustrasi bagaimana data dipisahkan dalam kasus AND.

Dalam kasus XOR (lihat Tabel 8.2), data tidak bisa dipisahkan secara linier. Untuk mengatasi masalah ketidaklinieran, kita perlu memformulasi SVM dalam *dual space*. Untuk itu kita perlu mengganti w dengan $w = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i)$. Kemudian kita perlu melakukan kernelisasi sehingga kita bisa mendapatkan fungsi pemisah linier di dalam *feature space*. Untuk itu kita gunakan fungsi kernel polynomial pangkat 2 yang didefinisikan sebagai $K(x_i, x_j) = (x_i x'_i + 1)^2$ (Haykin,1999).

Tabel 8.2: XOR Problem

No	x_1	x_2	Y
1	1	1	-1
2	-1	1	1
3	1	-1	1
4	-1	-1	-1

Dengan kernel ini, kita bisa menghitung matriks kernel K dengan dimensi *m* x *m*, dimana m adalah banyaknya data. Memakai data di Tabel 8.2, sebagai contoh, bisa dihitung K(1, 1) dan K(1, 2), sebagai berikut:

x^1 = [1 1]; (nilai x pada pengamatan ke-1)

x^2 = [−1 1]; (nilai x pada pengamatan ke-2)

$x^1x^{1'} = [1\ 1]\begin{bmatrix}1\\1\end{bmatrix} = 2;\ (x^1x^{1'} + 1)^2 = 9$

$x^1x^{2'} = [1\ 1]\quad\begin{bmatrix}-1\\1\end{bmatrix} = 0;\ (x^1x^{2'} + 1)^2 = 1$

Dengan prosedur yang sama untuk semua nilai x^i, kita dapatkan nilai matriks K sebagai berikut:

$$K = \begin{bmatrix}9 & 1 & 1 & 1\\ 1 & 9 & 1 & 1\\ 1 & 1 & 9 & 1\\ 1 & 1 & 1 & 9\end{bmatrix}$$

Dengan menggunakan matriks K sebagai pengganti dot product $\varphi(x_i)^T\varphi(x_j)$ dalam 8.11, maka kita dapatkan formulasi berikut

$$\max\ \alpha_1+\alpha_2+\alpha_3+\alpha_4 - \frac{1}{2}(9\alpha^2_1 - 2\alpha_1\alpha_2 - \alpha_1\alpha_3+\alpha_1\alpha_4+9\,\alpha^2_2 + 2\alpha_2\alpha_3 - \alpha_2\alpha_4+9\alpha^2_3 - 2\alpha_3\alpha_4 + 9\alpha^2_4)$$

Subject to

$$-\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0 \qquad (8.13)$$

$\alpha i \geq 0$

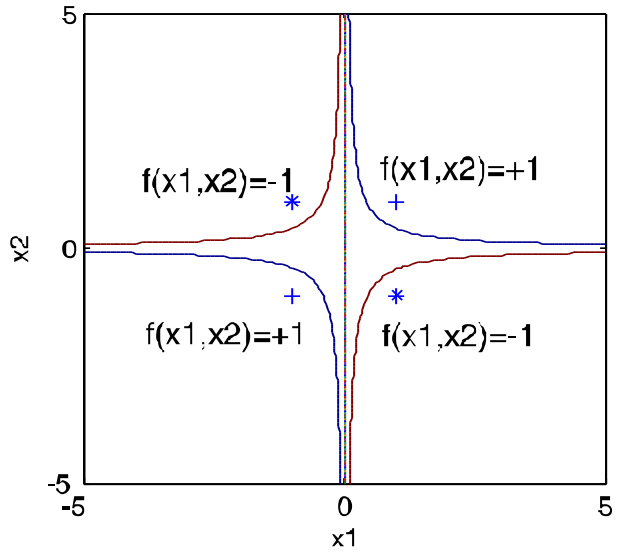
Dalam fungsi tujuan 8.13, term kedua kita kalikan dengan y_iy_j . Persamaan 8.13 memenuhi bentuk standar *programa kuadratik* (quadratic programming, QP). Sehingga masalah ini bisa diselesaikan dengan solver komersial untuk QP. Penyelesaian problem di atas dengan bantuan software akan memberi hasil sebagai berikut:

$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$

Hasil ini menunjukkan bahwa semua data dalam contoh ini adalah *support vector*. Karena nilai $\alpha \neq 0$. Setelah kita training SVM dan kita dapatkan nilai α, kita bisa menentukkan label untuk data testing dengan menggunakan fungsi pemisah sebagai berikut

$f(x) = y\,\alpha'\,K(x_{test}, x_{train}) + b,$

dimana vektor α dan konstanta b diketahui dari hasil training. Hasil ini dijelaskan secara grafis dalam Gambar 8.6. Perlu dijelaskan di sini bahwa nilai w tidak selalu bisa diekspresikan secara eksplisit sebagai kombinasi antara α, y dan φ(x), karena dalam banyak kasus φ(x) tidak diketahui atau sulit dihitung, kecuali dalam kasus kernel linier dimana φ(x) = x.



Gambar 8.6: Ilustrasi bagaimana data dipisahkan dalam kasus XOR.

Algoritma SVM untuk klasifikasi

Variabel dan parameter

$x = \{x_0, x_1, x_2, ..., x_m\}$: sampel training

$y = \{y_1, ..., y_m\} \subset \{\pm 1\}$: label data training

kernel: jenis fungsi kernel

par: parameter kernel

C: konstanta cost

$\alpha = [\alpha_1, ..., \alpha_m]$: Lagrange multiplier dan bias b

1. Hitung matriks kernel H
2. Tentukan pembatas untuk programa kuadratik, termasuk A_{eq}, b_{eq}, A dan b
3. Tentukan fungsi tujuan programa kuadratik $\frac{1}{2} xHx + f' x$
4. Selesaikan masalah QP dan temukan solusi α dan b

Contoh Hitungan

Kita perhatikan formulasi matematik SVM adalah

$$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ , \mathrm{i} = 1, ..\mathrm{m}$$

Atau

$$\min \ Q = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{N} \alpha_i$$

Subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ , \mathrm{i} = 1, ..\mathrm{m}$$

$$0 \le \alpha \le \mathrm{C}$$

Kemudian kita punya set data sederhana dengan 3 titik data sebagai berikut

X1	X2	Y
1	1	1
2	2	-1
3	3	-1

Menggunakan kernel linier, dengan data di atas kita dapatkan

$$\alpha y_i y_j x_i^T x \alpha^T = \frac{1}{2} [\alpha 1 \ \ \alpha 2 \ \ \alpha 3] \begin{bmatrix} 2 & -4 & -6 \\ -4 & 8 & 12 \\ -6 & 12 & 18 \end{bmatrix} \begin{bmatrix} \alpha 1 \\ \alpha 2 \\ \alpha 3 \end{bmatrix}$$

Setelah dikalikan didapatkan formulasi

Min $Q=\alpha_1^2+4\alpha_2^2+9\alpha_3^2-4\alpha_1\alpha_2-6\alpha_3\alpha_1+12\alpha_3\alpha_2-\alpha_1-\alpha_2-\alpha_3$

st

$\alpha_1-\alpha_2-\alpha_3=0$

Misalkan C=1, maka

$0 \le \alpha 1, \alpha 2, \alpha 3 \le 1$

Memasukkan konstrain ke dalam fugsi tujuan didapat fungsi Lagrange

$L= \alpha_1^2 +4\alpha_2^2 +9\alpha_3^2 -4\alpha_1\alpha_2-6\alpha_3\alpha_1+12\alpha_3\alpha_2- \alpha_1- \alpha_2- \alpha_3+u(\alpha_1- \alpha_2- \alpha_3)$

Untuk mencapai optimum dari fungsi L, kita turunkan sehingga didapatkan

1. $\frac{\partial L}{\partial \alpha_1} = 2\ \alpha_1 - 4\alpha_2 - 6\alpha_3 - 1 + u = 0$
2. $\frac{\partial L}{\partial \alpha_2} = -4\ \alpha_1 + 8\alpha_2 - 12\alpha_3 - 1 - u = 0$
3. $\frac{\partial L}{\partial \alpha_3} = -6\ \alpha_1 + 12\alpha_2 + 18\alpha_3 - 1 - u = 0$

4. $\alpha_1 - \alpha_2 - \alpha_3 = 0 \rightarrow \alpha_1 = \alpha_2 + \alpha_3$

Persamaan (2)&(3)&(4)

5. $\alpha_2 - 2\alpha_3 - u - 1 = 0$

(1)&(2)&(4)

6. $\alpha_2 - 10\ \alpha_3 - 2 = 0$

(1)&(3)

7. $\alpha_2 + 9\alpha_3 - 2 = 0$

Dari persaman (6) dan (7) didapat $-19\alpha_3 = 0$ maka $\alpha_3 = 0$

Kemudian didapat $\alpha_2 = 2$ dan $\alpha_1 = 2$, karena $\alpha <= 1$ maka bisa dilakukan normalisasi dengan membagi masing-masing nilai α dengan 2 (tanpa merubah persoalan aslinya), sehingga $\alpha_1 = 1$, $\alpha_2 = 1$ $\alpha_3 = 0$

Jadi data ketiga sebenarnya tdk menentukan hyperplane pemisah krn nilai $\alpha_3 = 0$.

Ini sesuai dgn dugaan kita bahwa titik (3,3) sebenarnya tidak terlalu penting, dengan 2 titik (1,1) dan (2,2) saja kita sudah bisa menemukan hyperplanenya.

8.7 Implementasi SVM dengan Matlab

Pada dasarnya, Matlab sampai versi terakhir (versi 7) ketika buku ini ditulis, tidak mempunyai toolbox khusus untuk implementasi SVM. Namun Matlab mempunyai solver *quadratic programming* yang merupakan solver kunci dalam implementasi

SVM. Seperti kita bahas sebelumnya bahwa formulasi matematika SVM adalah berbentuk *quadratic programming*. Ada banyak program komputer yang ditulis dalam berbagai bahasa maupun paket software yang ditujukan untuk implementasi SVM. Lihat website berikut http : //www.kernel−machines.org.

Ringkasan SVM dalam notasi matriks diberikan sebagai berikut

$$\frac{1}{2}\alpha' H \alpha + f'\alpha$$

subject to

$$\alpha^T Y = 0,\ 0 \le \alpha_i \le C,\ i = 1, .., m \qquad (8.14)$$

dimana

$$H = ZZ^T,\ f^T = (-1, ..., -1)$$

$$Z = \begin{bmatrix} y_1 x_1 \\ . \\ . \\ y_m x_m \end{bmatrix},\ Y = \begin{bmatrix} y_1 \\ . \\ . \\ y_m \end{bmatrix}$$

Berikut ini adalah Matlab code untuk Support Vector Classification.

```
function [alpha, b0] = svc(X,Y,ker,par,C)
%SVC Support Vector Classification
% Parameters: X - Training inputs
% Y - Training targets
% ker - kernel function
% par - parameter untuk kernel
% C - upper bound (non-separable case)/pinalti
% alpha - Lagrange Multipliers
% b0 - bias term
%fprintf('Support Vector Classification\n')
%fprintf('_____\n')
n = size(X,1);
if (nargin<5) C=Inf;, end
if (nargin<3) ker='linear';, end
% Construct the Kernel matrix
```

```
% fprintf('Constructing ...\n');
H = kernel(X',ker,par);%memanggil fungsi kernel
for i=1:n
for j=1:n
H(i,j) = Y(i)*Y(j)*H(i,j);
end
end
c = -ones(n,1);
% Set up the parameters for the Optimisation problem
vlb = zeros(n,1); % Set the bounds: alphas >= 0
vub = C*ones(n,1); % alphas <= C
x0 = zeros(n,1); % The starting point is [0 0 0 0]
A = Y';, b = 0; % Set the constraint Ax = b
% Solve the Optimisation Problem
fprintf('Optimising ...\n');
st = cputime;

 [alpha,lambda,how]=quadprog(H,c,[],[],A,b,vlb,vub,x0);
w2 = alpha'*H*alpha;
fprintf('|w0|^2 : %f\n',w2);
fprintf('Margin : %f\n',2/sqrt(w2));
fprintf('Sum alpha : %f\n',sum(alpha));
% Hitung banyaknya Support Vectors
svi = find( alpha > epsilon);
nsv = length(svi);
%fprintf('Support Vectors : %d (%3.1f%%)\n',nsv,100*nsv/n);
% Implicit bias, b0
b0 = 0;
% Explicit bias, b0
% Hitung b0 dari rata-rata support vectors pada margin
% SVs pada margin mempunyai alphas: 0 < alpha < C
svii = find( alpha > epsilon & alpha < (C - epsilon));
if length(svii) > 0
b0=(1/length(svii))*sum(Y(svii)                                    -
H(svii,svi)*alpha(svi).*Y(svii));
```

```
else
fprintf('Tidak ada support vectors pada margin-tidak bisa
menghitung bias.\n');
end
end
```

REFERENSI

Budi Santosa, Data Mining , Teknik Pemanfaatan Data untuk Keperluan bisnis, Graha Ilmu 2007

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 1999.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining , inference, and prediction*. Springer-Verlag, New York, 2001.

Simon Haykin. *Neural Network: A Comprehensive Foundation*. Prentice Hall, New Jersey, 1999.

B. Scholkopf and A.J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, Massachusetts, 2002.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag,1995.