Tutorial Support Vector Machine

Budi Santosa

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*. Baik para ilmuwan maupun praktisi telah banyak menerapkan teknik ini dalam menyelesaikan masalah-masalah nyata dalam kehidupan sehari-hari. Baik dalam masalah gene expression analysis, finansial, cuaca hingga di 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 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. 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

$$g(x) := sgn(f(x)) \tag{1}$$
$$\text{dengan } f(x) = w^T x + b,$$

dimana x, $w \in \Re^n$ and $b \in \Re$. Masalah klasifikasi ini bisa dirumuskan sebagai berikut: kita ingin menemukan set parameter (w, b) sehingga $f(x_i) =< w, x > +b = y_i$ untuk semua i. 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 $+1$ ($wx_1 + b = +1$) dan $wx_2 + b = -1$ hyperplane-pendukung dari kelas -1 ($wx_2 + b = -1$), margin antara dua kelas dapat dihitung dengan mencari jarak antara kedua hyperplane-pendukung dari kedua kelas. Secara spesifik, margin dihitung dengan cara berikut $(wx_1 + b = +1) - (wx_2 + b = -1) \Rightarrow w(x_1 - x_2)) = 2 \Rightarrow \left(\frac{w}{\|w\|}(x_1 - x_2)\right) = \frac{2}{\|w\|}$. Gambar 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 2). Fungsi f_1 mempunyai margin yang lebih besar dari pada fungsi f_2. Setelah mene-

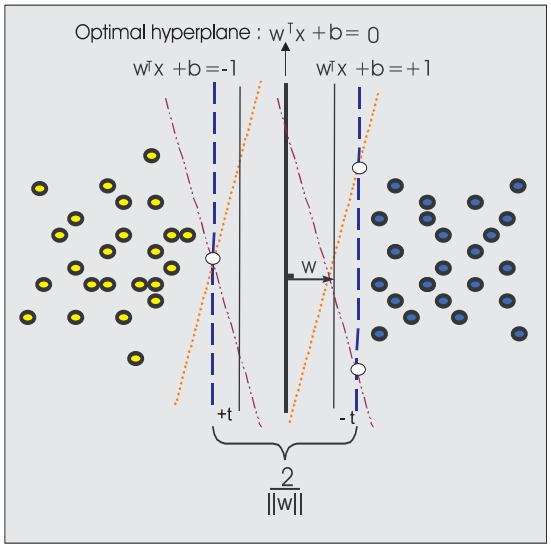

Figure 1: Mencari fungsi pemisah yang optimal untuk obyek yang bisa dipisahkan secara linier

mukan dua fungsi ini, sekarang 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. Dengan menggunakan f_1, kita akan kelompokkan data baru ini di kelas -1 yang berarti kita benar mengelompokkannya. Sekarang coba kita 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.

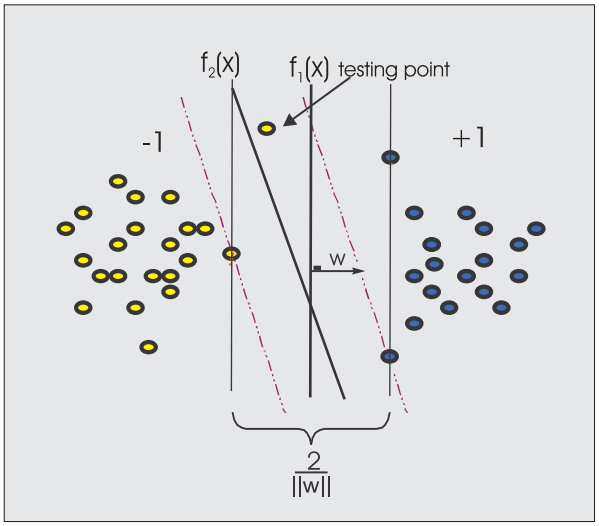

Figure 2: Memperbesar margin bisa meningkatkan probabilitas pengelompokkan suatu data secara benar.

2 Formulasi Matematis

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

$$\min \frac{1}{2} \|w\|^2 \tag{2}$$

Subject to

$$y_i(wx_i + b) \geq 1, \ i = 1, .., \ell,$$

dimana x_i adalah data input, y_i adalah keluaran dari data x_i, w, b adalah parameter-parameter yang kita cari nilainya. Dalam formulasi di atas, kita ingin meminimalkan *fungsi tujuan* (obyektif function) $\frac{1}{2}\|w\|^2$ atau memaksimalkan kuantitas $\|w\|^2$ 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

$$\begin{aligned} \min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\ell} t_i \\ \text{Subject to} \\ y_i(wx_i + b) + t_i \geq 1 \\ t_i \geq 0,\ i = 1, .., \ell, \end{aligned} \tag{3}$$

dimana t_i adalah variabel *slack*. Dengan formulasi ini kita ingin memaksimalkan margin antara dua kelas dengan meminimalkan $\|w\|^2$ [4]. 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, $\frac{1}{\|w\|}$. 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 (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 (3) menjadi *fungsi Lagrangian* :

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{\ell} \alpha_i \left[y_i(w^t x_i + b) - 1 \right] \tag{4}$$

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 (4) akan menghasilkan

$$w = \sum_{i=1}^{\ell} \alpha_i y_i x_i \tag{5}$$

Penerapan kondisi optimalitas 2 pada fungsi Lagrangian (4) akan menghasilkan

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \tag{6}$$

Menurut duality theorem [1]:

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

$$\Phi(w_o) = J(w_o, b_o, \alpha_o) = \min_w J(w, b, \alpha)$$

Untuk mendapatkan problem dual dari problem kita, kita jabarkan persamaan (4) sebagai berikut:

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{\ell} \alpha_i y_i w^T x_i - b \sum_{i=1}^{\ell} \alpha_i y_i + \sum_{i=1}^{\ell} \alpha_i \tag{7}$$

Menurut kondisi optimalitas ke dua dalam (6), term ketiga sisi sebelah kanan dalam persamaan di atas sama dengan 0. Dengan memakai nilai-nilai w di (5), kita dapatkan

$$w^T w = \sum_{i=1}^{\ell} \alpha_i y_i w^T x_i = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{8}$$

maka persamaan 7 menjadi

$$Q(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j x_i^T x_j \tag{9}$$

Selanjutnya kita dapatkan formulasi dual dari problem (3):

$$\begin{aligned} \max \quad & \sum_{i=1}^{\ell} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{Subject to} \quad & \sum_{i=1}^{\ell} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i, i = 1, ..\ell, \end{aligned} \tag{10}$$

Dengan dot product $x_i x_j$ sering diganti dengan simbol K. K adalah matrik kernel yang dijelaskan dalam bagian 3. Formulasi (10) adalah quadratic programming (QP) dengan pembatas (constraint) linier. Melatih SVM ekuivalen dengan menyelesaikan problem *convex optimization*. Karena itu solusi dari SVM adalah unik (dengan asumsi bahwa k adalah positive definite) dan global optimal. Hal ini berbeda dengan solusi neural networks [4] yang ekuivalen dengan problem *nonconvex optimization* dengan akibat solusi yang ditemukan adalah *local optima*. Ambil $f(x) = \sum_{i=1}^{\ell} y_i \alpha_i^* k(x_i, x) + b^*$.

Fungsi pemisah optimal adalah $g(x) = sign(\sum_{i=1}^{\ell} y_i \alpha_i^* k(x, x_i)) + b^*$, dimana $\alpha_i^*, i = 1, .., \ell$ adalah solusi optimal dari problem (10) dan b^* dipilih sehingga $y_i f(x_i) = 1$ untuk sembarang i dengan $C > \alpha_i^* > 0$ [2]. Data x_i dimana $\alpha_i^* > 0$ dinamakan *support vector* dan menyatakan data training yang diperlukan untuk mewakili fungsi keputusan yang optimal. Dalam gambar 1, sebagai contoh, 3 titik berwarna putih menyatakan *support vector*. Untuk mengatasi masalah ketidaklinieran (nonlinearity) yang sering terjadi dalam kasus nyata, kita bisa menerapkan metoda kernel. Metoda kernel [5] memberikan pendekatan alternatif dengan cara melakukan mapping data x dari input space ke *feature space* F melalui suatu fungsi φ sehingga $\varphi : x \mapsto \varphi(x)$. Karena itu suatu titik x dalam input space menjadi $\varphi(x)$ dalam feature space.

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 3. Data ini sulit dipisahkan secara linier. Metoda kernel [5] 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 $\varphi : x \mapsto \varphi(x)$. Karena itu data x di input space menjadi $\varphi(x)$ di feature space.

Sering kali fungsi $\varphi(x)$ tidak tersedia atau tidak bisa dihitung. tetapi *dot*

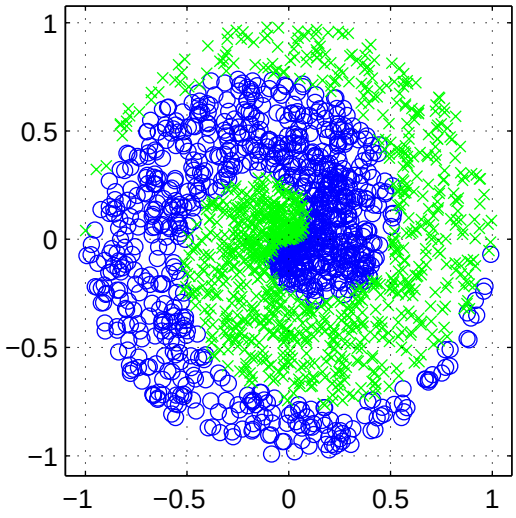

Figure 3: Data spiral yang menggambarkan ketidaklinieran

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, pembatas (constraint) perlu diekspresikan dalam bentuk dot product dari vektor data x_i. Sebagai konsekuensi, pembatas 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 untuk menggantikan dot product $<\varphi(x), \varphi(x)'>$. Kemudian di feature space, kita bisa membuat suatu fungsi pemisah yang linier yang mewakili fungsi nonlinear di input space. Gambar 4 mendeskrisikan 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 [5].

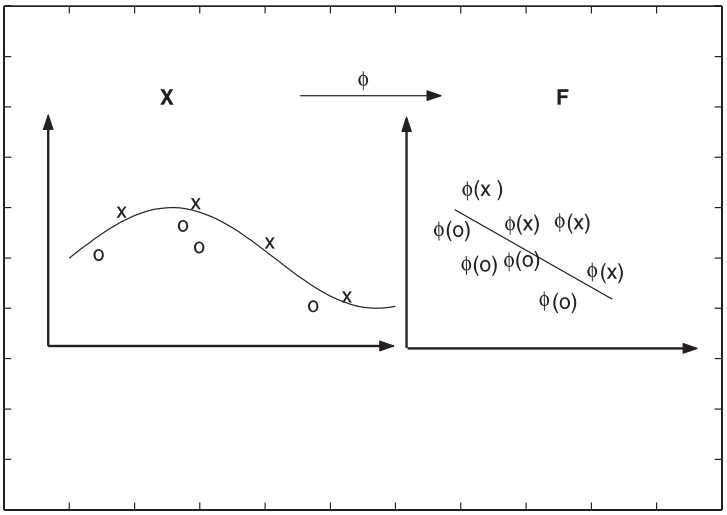

Figure 4: Suatu kernel map mengubah problem yang tidak linier menjadi linier dalam space baru

Fungsi kernel yang biasanya dipakai dalam literatur SVM [4]:

- linier: $x^T x$,
- Polynomial: $(x^T x_i + 1)^p$,
- Radial basis function (RBF): $exp(-\frac{1}{2\sigma^2}\|x - x_i\|^2)$,
- Tangent hyperbolic (sigmoid): $tanh(\beta x^T x_i + \beta_1)$, dimana $\beta, \beta_1 \in \Re$

Fungsi kernel mana yang harus digunakan untuk subtitusi dot product di feature space sangat bergantung pada data. Biasanya metoda cross-validation [3] 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 [6] 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 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.

4 Implementasi

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 1). Karena ini problem linier, kernelisasi tidak diperlukan. Menggunakan data di Tabel 1, kita

Table 1: AND Problem

x_1	x_2	y
1	1	1
-1	1	-1
1	-1	-1
-1	-1	-1

dapatkan formulasi masalah optimisasi sebagai berikut:

$$\min \frac{1}{2}(w_1^2 + w_2^2) + C(t_1 + t_2 + t_3 + t_4) \tag{11}$$

$$\begin{aligned} &\text{Subject to} \\ w_1 + w_2 + b + t_1 &\geq 1 \\ w_1 - w_2 - b + t_2 &\geq 1 \\ -w_1 + w_2 - b + t_3 &\geq 1 \\ w_1 + w_2 - b + t_4 &\geq 1 \\ t_1, t_2, t_3, t_4 &\geq 0 \end{aligned}$$

Karena fungsi AND adalah kasus klasifikasi linier, maka bisa dipastikan nilai variable slack $t_i = 0$. Jadi Kita bisa masukkan nilai $C = 0$. Setelah menyelesaikan problem optimasi di atas didapat solusi

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

Persamaan fungsi pemisahnya adalah

$$f(x) = x_1 + x_2 - 1.$$

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 5.

Dalam kasus XOR (lihat Tabel 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 $\sum_{i=1}^{\ell} \alpha_i y_i \varphi(x_i)$. Kemudian kita perlu melakukan kernelisasi sehingga kita bisa mendapatkan fungsi linier di dalam *feature space*. Untuk itu kita gunakan

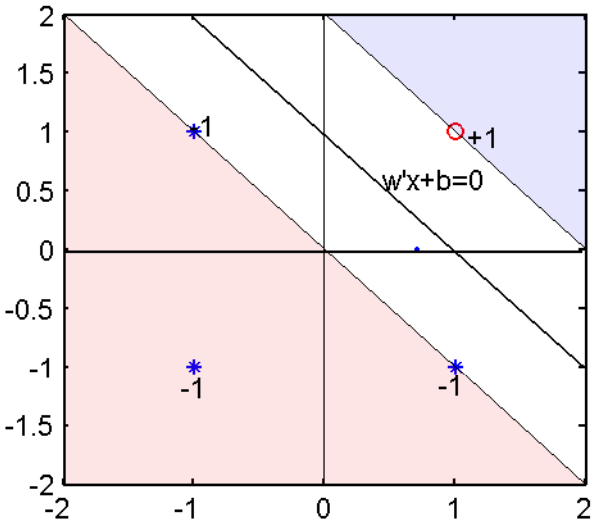

Figure 5: Ilustrasi bagaimana data dipisahkan dalam kasus AND.

fungsi polynomial kernel pangkat 2 yang didefinisikan sebagai $K(x_i, x_i) = (x_i x_i' + 1)^2$ [4]. Dengan kernel ini, kita bisa menghitung matriks kernel K dengan dimensi ℓ x ℓ, dimana ℓ adalah banyaknya data. Memakai data di Tabel 1, sebagai contoh, bisa dihitung $K(1,1)$ dan $K(1,2)$, sebagai berikut:

$$\begin{aligned}
x_1 &= [1 \quad 1]; \\
x_2 &= [-1 \quad 1]; \\
x_1 x_1' &= [1 \quad 1][1 \quad 1]' = 2; \quad (x_1 x_1' + 1)^2 = 9 \\
x_1 x_2' &= [1 \quad 1][-1 \quad 1]' = 0; \quad (x_1 x_2' + 1)^2 = 1
\end{aligned}$$

Dengan prosedur yang sama untuk semua nilai x_i, kita dapatkan nilai ma-

Table 2: XOR Problem

x_1	x_2	y
1	1	-1
-1	1	1
1	-1	1
-1	-1	-1

triks K sebagai berikut:

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix} \tag{12}$$

Dengan menggunakan matriks K sebagai pengganti dot product $x_i x_j$ dalam persamaan (10), maka kita dapatkan formulasi berikut

$$\begin{aligned} \max\ \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ +9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2) \\ \text{Subject to} \\ -\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0 \\ \alpha_i \geq 0 \end{aligned} \tag{13}$$

Dalam fungsi tujuan, persamaan (13), term kedua kita kalikan dengan $y_i y_j$. Persamaan (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:

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$$

Hasil ini menunjukkan bahwa semua data dalam contoh ini adalah *support vector*.

Setelah kita training SVM, 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 scara grafis dalam Gambar 6. Perlu dijelaskan di sini bahwa nilai w tidak selalu bisa diekspresikan secara eksplisit sebagai kombinasi antara α, y dan $\varphi(x)$, karena dalam banyak kasus $\varphi(x)$ tidak diketahui atau sulit dihitung, kecuali dalam kasus kernel linier dimana $\varphi(x) = x$.

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

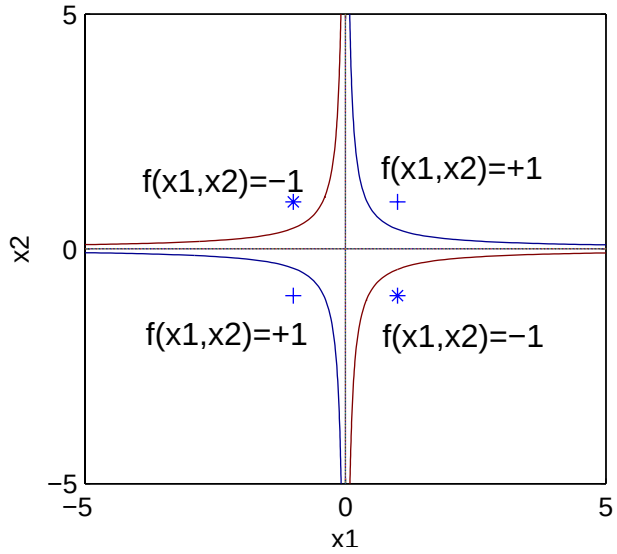

Figure 6: Ilustrasi bagaimana data dipisahkan dalam kasus XOR.

3. Tentukan fungsi tujuan programa kuadratik $\frac{1}{2}xHx + f'x$

4. Selesaikan masalah QP dan temukan solusi α dan b

5 Neural Networks (ANN) dan SVM

Ada beberapa persamaan dan perbedaan antara ANN dan SVM. Proses training pada kedua klasifier pada prinsipnya bertujuan mencari fungsi pemisah antara dua set data dari dua kelas di ruang vektor. Yang membedakan kedua buah model ini adalah prinsip yang dipakai untuk menemukan fungsi klasifier pemisah (*separating hyperplane*) tersebut.

- Pada ANN, fungsi klasifier dinyatakan dalam *primal form*, yaitu persamaan yang melibatkan weights/bobot dinyatakan dengan $\hat{y} = wx + w_0$. Weights ini sebenarnya tidak lain merepresentasikan posisi hyper-

plane itu pada *primal space*. Adapun pada SVM, tujuan dari proses training adalah untuk mencari posisi optimal dari hyperplane itu di *dual space*. Dalam hal ini kriteria yang dipakai adalah *margin* yaitu jarak dari separating hyperplane ke masing-masing kelas. Hasil perhitungan menunjukkan bahwa hyperplane yang terbaik dicapai dengan memaksimalkan nilai margin. Posisi ini tercapai jika hyperplane itu terletak tepat di tengah-tengah, memisahkan kedua buah kelas. Pada SVM, fungsi klasifikasi dinyatakan dalam *dual form*, yaitu dinyatakan sebagai fungsi dari data training dan Lagrange multiplier α, bukan weights. Fungsi ini tidak dinyatakan secara eksplisit.

- Proses training pada ANN berjalan dengan mengoreksi nilai weights secara berulang, sedemikian hingga kedua buah kelas dapat dipisahkan secara sempurna oleh hyperplane itu. Weight yang diperoleh adalah hasil akhir dari proses pembelajaran pada ANN. Proses training pada SVM sebagaimana dijelaskan diatas, bertujuan mencari data mana dari training set yang *paling informatif* dalam rangka membentuk fungsi pemisah. Subset dari training set yang paling informatif inilah hasil akhir dari proses training tersebut. Data ini disebut dengan ***Support Vektor***. Di ruang vektor, support vektor ini adalah data dari kedua buah kelas yang terletak paling dekat dengan hyperplane pemisah yang dipotong oleh *supporting hyperplane*.

- Untuk problem nonlinier, ANN biasanya melibatkan desain multilayer (hidden layer) agar bisa menemukan hyperplane pemisah secara nonlinier. Dalam hal ini proses pembelajaran yang populer antara lain back-propagasi. SVM pada prinsipnya adalah klasifier linier. Tetapi SVM justru unggul dalam klasifikasi untuk problem nonlinier. Dalam

SVM,pertama, data diproyeksikan ke ruang vektor baru, sering disebut *feature space*, berdimensi lebih tinggi sedemikian hingga data itu dapat terpisah secara linier. Selanjutnya, di ruang baru itu, SVM mencari hyperplane optimal dengan cara yang sama sebagaimana dijelaskan di atas, yaitu bekerja sebagai klasifier linear. Ini didukung oleh teori Cover, yang menyatakan bahwa suatu ruang vektor dapat ditransformasikan ke ruang vektor baru yang pada probabilitas tinggi dapat terpisah secara linier, jika memenuhi dua syarat : (i) transformasi itu nonlinier (ii) dimensi ruang vektor yang baru itu cukup tinggi. Fungsi klasifikasi pada SVM dinyatakan sebagai *dual form*, yang melibatkan dot product antara data pada training set. Dengan demikian, proses pencarian hyperplane optimal di "ruang baru" melibatkan dot product dari data yang sudah diproyeksikan ke "ruang baru" tersebut. Di sinilah muncul konsep "kernel trick", yaitu menghitung nilai dot product dua buah data di "ruang baru" itu, secara implisit. Dengan menggunakan fungsi kernel, kita tidak perlu mengetahui secara detil wujud transformasi ke ruang dimensi yang lebih tinggi. Bagaimana cara mapping satu per satu suatu titik sehingga berada pada suatu titik pada dimensi yang lebih tinggi tidak perlu diketahui. Dot product dua titik di ruang baru bisa dihitung dengan fungsi kernel. Umumnya data akan diubah ke dimensi ruang yang jauh lebih tinggi daripada dimensi aslinya. Dan, semua itu cukup dilakukan dengan menggunakan fungsi kernel. Yang penting di sini adalah data bisa pada data dipisahkan secara linear pada dimensi ruang yang lebih tinggi.

- ANN sering mengalami fenomena overfitting karena overtrained. SVM tidak mengalami masalah ini karena training perlu dilakukan sekali

saja dan akan mendapatkan solui optimal.

- SVM hanya memerlukan parameter kernel (tergantung fungsi kernel biasanya 1-2 parameter) dan satu lagi parameter C yang memberi pinalti pada titik data yang akan diklasifikasikan secara salah. Jadi hanya memerlukan dua atau tiga parameter saja. Sedangkan pada ANN, banyak hal yang harus diatur nilai parameternya seperti jumlah hidden layer, neuron untuk hidden layer, metode untuk training. dll.

- Dalam perceptron (ANN), solusi untuk w(bobot) dan b(bias) merupakan *local minimum* dari cost function(biasanya nonlinear) yangg cukup untuk memisahkan data ke dalam kelas yang sesuai dengan cara iterative. Sedangkan dalam SVM solusi w dan b (yang dinyatakan dengan Lagrange multiplier dalam α) adalah global optimal dari quadratic programming yang merupakan formulasi matematik dari SVM. Dalam ANN setiap kali running, bisa mencapai local mimimum yg berbeda (w dan b berbeda), dalam SVM selalu converge ke solusi yang sama. Sehingga dalam ANN kita harus running desain kita beberapa kali untuk mendapatkan solusi terbaik atau diambil w dan b rata-rata. Sedangkan dalam SVM dengan sekali running sudah cukup karena solusi yang didapat akan selalu sama untuk pilihan kernel dan parameter yang sama.

- ANN akan menggunakan fungsi aktivasi (transfer function) yang bisa linier ataupun nonlinier dalam rangka mendapatkan bobot dari klasifier yang mampu memisahkan data ke dalam kelas yang sesuai. Ini mirip dengan pemakaian kernel function dalam SVM yang tujuannya membuat data menjadi linearly seprable di "feature space".

6 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

$$\frac{1}{2}x'Hx + f'x \tag{14}$$

$$\begin{aligned} &\text{subject to} \\ &\alpha^T Y = 0, \\ &\alpha_i \geq 0 \\ &\alpha_i \leq c, \\ &i = 1, .., \ell \end{aligned} \tag{15}$$

dimana

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

$$Z = \begin{bmatrix} y_1 x_1 \\ . \\ . \\ y_\ell x_\ell \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ . \\ . \\ y_\ell \end{bmatrix},$$

dimana x adalah vektor data example, y adalah label. 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)
%              nsv    - number of support vectors
%              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);
    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));
% Compute the number of 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
if nobias(ker) ~= 0
  % find b0 from average of support vectors on margin
  % SVs on margin have 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('No support vectors on margin-cannot compute bias.\n');
  end
end
```

7 Latihan

1. Hitunglah matrik kernel K untuk masalah XOR dengan kernel RBF,$\sigma = 1$.

2. Dalam formulasi dual SVM, berapa jumlah multiplier/variabel yang dicari untuk problem berukuran m x n?

3. Untuk problem dengan ukuran m x n berapa ukuran matrik K yang dihasilkan dalam formulasi dual?

4. Apa pentingnya variabel slack dalam formulasi SVM?

References

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

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

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

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

[5] B. Schölkopf and A.J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, Massachusetts, 2002.

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