## II.4 Support Vector Machine

Support Vector Machine (SVM) merupakan salah satu metode *machine learning* untuk *pattern recognition*, yang dalam penelitian ini adalah *image recognition*. Algoritma SVM pertama kali dikembangkan oleh Vladimir Vapnik (Bishop,2006). Konsep dasar SVM ialah mengklasifikasikan data menjadi dua kelas yang berbeda dengan cara membuat membuat *hyperplane* atau suatu bidang yang merupakan fungsi klasifikator antara dua kelas dengan menggunakan konsep memaksimumkan margin.

Bentuk umum klasifikator SVM ialah sebagai berikut :

y = *sign*(wTx + b) (II.20) Keterangan :

y merupakan nilai target pada setiap vector baris

x Rn merupakan vector yang dimensinya bergantung dari *n* banyaknya vitur

w Rn vector yang menjadi parameter bobot

b bias atau eror berupa skalar

Hyperplane yang dihasilkan SVM dapat menklasifikasikan data menjadi dua kelas, yaitu kelas positf dan kelas negative yang dimodelkan sebagai berikut :

wTxi + b ≥1, untuk yi bernilai 1 (II.21)

wTxi + b ≥1, untuk yi bernilai -1 (II.22)

dapat dirumuskan sebagai berikut

yi (wTxi + b) ≥1 (II.23)

Misal dibuat dua bidang yang sejajar dengan hyperplane, untuk kelas B+ dapat dirumuskan sebagai berikut,

B+ : wTxi + b = 1 (II.24)

Untuk kelas B- dapat dirumuskan sebagai berikut,

B- : wTxi + b = -1 (II.25)

Margin adalah jarak antara bidang B+ dan B-

X1

Hyperplane

yi = 1

B+ : wTxi + b = 1

B- : wTxi + b = -1

yi = -1

||W||

X2

Gambar II.3 *Hyperplane*

Untuk mendapatkan jarak dari bidang B+ dengan data yang berlabel positif, dapat dilakukan dengan mencari jarak minimum antara data yang mempunyai target positif terdekat dengan bidang B+. Sehingga secara matematis kita dapat meminimumkan xTx dengan syarat wTxi + b 1. Dengan pengali Lagrange (λ) didapatkan bentuk.

L = xTx + λ (-wTx – b + 1) (II.26)

Kemudian turunkan fungsi L terhadap x sehingga didapatkan hasil

= 0 = xT- λwT

xT = λwT x = λw (II.27)

subtitusikan x pada persamaan (II.27) ke dalam persamaan bidang (II.24) B+ : wTxi + b = 1, sehingga menjadi

wT λw + b = 1

λ = (II.28)

Subtitusikan λ pada persamaan (II.28) kedalam **x** pada persamaan (II.27) menjadi

x = = (II.29)

sehingga didapat persamaan untuk xTx sebagai berikut :

xTx = (II.30)

Jadi, jara terdekat bidang B+ ke bidang pemisah adalah

S1 = ||x|| = = = (II.31) Sedangkan untuk mendapatkan jarak dari bidang B- dengan data yang berlabel negative, dapat dilakukan dengan mencari jarak minimum antara data yang mempunyai target negative terdekat dengan bidang B-. Sehingga secara matematis kita dapat meminimumkan xTx dengan syarat wTxi + b -1. Dengan pengali Lagrange (λ) didapatkan bentuk.

L = xTx + λ (wTx – b + 1) (II.32) Lalu turunkan fungsi L terhadap x sehingga didapatkan

= 0 = xT+ λwT

xT =- λwT x = - λw (II.33) subtitusikan x pada persamaan (II.33) ke dalam persamaan bidang (II.25) B- : wTxi + b = -1, sehingga menjadi

-wT λw + b = -1

λ = (II.34)

Subtitusikan λ pada persamaan (II.3.4) kedalam **x** pada persamaan (II.33) menjadi

x = = (II.35)

sehingga didapat persamaan untuk xTx sebagai berikut :

xTx = (II.36)

Jadi, jarak terdekat bidang B- ke *hyperline* adalah

S1 = ||x|| = = =

Untuk mencari margin yang sesuai, diperlukan jarak yang maksimum dari data yang terdekat dengan *hyperline* dari masing – masing kelas. Dari persamaan (II.31 dan II.36) dapat dibentuk,

=

(II.37)

Sehingga persamaan dapat dibentuk menjadi,

(II.38)

Dengan syarat :

min wTxi + b = 1 , yi = +1

max wTxi + b = -1 , yi = -1

sehingga akan menghasilkan fungsi klasifikator sebagai berikut :

y = *sign*(wTx + b) (II.39)

Masalah (II.38) juga dapat ditulis sebagai berikut :

atau (II.40)

Dengan syarat :

yi (wTxi + b) ≥1 (II.41) Dalam pembentukan fungsi klasifikator, ada beberapa yang tidak dapat diklasifikasikan secara benar atau permasalahan optimasinya bersifat tidak layak. Untuk menghindari masalah ini, ditambahkan variable *slack* ( ≥ 0) pada masalah optimasi (II.40) sehingga menjadi :

(II.42) Dengan syarat : yi (wTxi + b) ≥1 - i  (II.43)

i ≥ 0 , (II.44)

Dimana C akan mengontrol *tradeoff* antara variable slack dan margin.

Permasalahan (II.42) merupakan bentuk *quadratic programing*. Untuk menyelesaikan pemograman kuadrat tersebut, cara yang umum digunakan adalah mencari bentuk dual dengan menggunakan pengali Lagrange *(Lagrange multipliers)*. Dari permasalahan diatas, bentuk *Lagrange*-nya adalah sebagai berikut :

L (w, α, λ) = ||w||2 + C i+ i (1 - yi (wTxi + b - i ) + - i )

= ||w||2 + C i + i - i yi (wTxi + b) - i i - i i (II.45)

Dengan kendala variable dualnya adalah sebagai berikut

αi ≥ 0 , i ≥ 0 (II.46)

Pada bentuk dual (II.45), αi dan i adalah pengali *Lagrange.*

Soulusi dari masalah optimasi (II.45) dapat ditentukan melalui mencari *saddle point*-nya dengan cara menghitung turunan parsial dari *L* yang sama dengan 0 sebagai berikut :

* Turunan terhadap **w :**

= w - i yi xi = 0

w = i yi xi (II.47)

* Turunan terhadap *b* :

= i yi  = 0 (II.48)

* Turunan terhadap i :

= αi - i  = 0 (II.49)

Dengan syarat i ≥ 0, αi ≥ C.

Dengan mensubtitusikan turunan parsial L (II.47), (II.48) dan (II.49) ke masalah dual (II.45) menghasilkan :

Maks L = (iyixi) (iyixi) + C i + i - iyi ((iyixi) xi + b) - ii - ii

= i yi αj yj (xi . xj) -  i yi αj yj (xi . xj) - b i yi + i + C i - i i - i i

= i - i yi αj yj (xi . xj) (II.50)

Dengan syarat

0 ≤ αi ≤ C, I = 1, …, N (II.51)

i yi = 0 (II.52)

Meskipun telah menggunakan variable *slack* tidak semua data merupakan data yang dapat dipisahkan secara linier oleh *hyperlane*.Oleh Karena itu untuk mengatasi masalah ini digunakan fungsi kernel yang berfungsi untuk memetakan data ke dimensi yang lebih tinggi, sehingga diharapkan data menjadi bersifat linearly separable. Fungsi kernel Radial Basis Function (RBF) secara umum direkomendasikan sebagai pilihan utama, sehingga pada penelitian ini akan digunakan fungsi kernel RBF. Hal ini dikarenakan 3 hal yaitu :

1. Fungsi kernel RBF dapat memetakan secara tidak linear data ke ruang dimensi yang lebih tinggi, sehingga diharapkan dapat menangani kasus di mana relasi antara kelas dan fitur yang tidak linear.
2. Fungsi kernel RBF menggunakan lebih sedikit parameter
3. Fungsi kernel RBF memiliki kesulitan numerik yang lebih sedikit dibandingkan dengan fungsi kernel lain.

Dalam pemakaiannya, data xi akan dipetakan dengan fungsi Ф(xi) dan setiap perkalian (xi . xj) akan dihitung menggunakan K (xi . xj) yang memiliki bentuk sebagai berikut

*K (xi . xj)* = exp (-|| xi - xj||2); > 0, (II.53)

Dengan menambahkan fungsi kernel RBF ke dalam persamaan (II.50), maka bentuk, maka bentuk masalah dual yang baru adalah sebagai berikut :

maks L = i - i yi αj yj K(xi . xj) (II.54)

dengan syarat

0 ≤ αi ≤ C, i = 1, …, N (II.55)

i yi = 0 (II.56)

Permasalahan SVM dalam bentuk dual Lagrange (II.54) dapat diselesaikan dengan menggunakan metode Sequential Minimal Optimization (SMO) (Platt, 1998). Jika solusi dari permasalahan (II.54) adalah α\*, maka **w** dapat dicari dengan subtitusi α\* kedalam persamaan (II.47) sebagai berikut :

w = yi xi  (II.57)

Dengan mensubtitusikan persamaan (II.57) kedalam persamaan (II.20) sehingga fungsi klasifikator (*hyperplane*) adalah persamaan dengan bentuk sebagai berikut:

y = *sign*(wTx + b)

y = *sign* (yi x + b)

y = *sign* (yi K(xi . xj) + b) (II.58)

Pada persamaan (3.8) hanya data dengan > 0 (*support vectors*) yang akan berperan pada model persamaan fungsi klasifikator (*hyperplane*) atau dengan kata lain

y = *sign* (yi K(xi . xj) + b) (II.59)

dengan S adalah himpunan indeks dari *support vectors.*

Bentuk dual dari persamaan (II.59) memenuhi KKT sebagai berikut :

αi ≥ 0, [KKT – 1]

yi(wTxi + b) -1 + i ≥ 0, [KKT – 2]

αi(yi(wTxi + b) – 1 + i ) = 0, [KKT – 3]

i ≥ 0, [KKT – 4]

i ≥ 0, [KKT – 5]

ii = 0 [KKT – 6]

Karena *support vectors* adalah data training dengan αi ≥ 0, maka KKT – 3 menjadi

yi(wTxi + b) -1 + i = 0

yi(wTxi + b) = 1 – i (II.60)

Dari syarat persamaan (II.55) dan dari hasil turunan *L* terhadap i pada persamaan (II.49), akan didapat :

1. Bila< C maka i > 0, dan berdasarkan KKT – 6 maka I = 0 yang berarti data training terletak pada *hyperplane*
2. Bila = C maka i = 0, dan berdasarkan KKT – 6 maka i ≠ 0 yang berarti *data training* terletak di dalam margin, baik terklasifikasi dengan benar (i ≤ 1) maupun salah (i >1).

Dari fakta tersebut, maka b dapat dicari dengan persamaan berikut :

yi(wTxi + b) = 1

yiyi K(xi . xj) + b) = 1

b = i  yi K(xi . xj) (II.61)

dengan S adalah himpunan indeks dari *support vector*, dan Ns adalah banyaknya data train yang menjadi *support vectors.*

**II.4.1 Fungsi Kernel**

Pada umumnya masalah dalam domain dunia nyata jarang bersifat linear separable. Kebanyakan bersifat non linear. Oleh karena itu untuk menyelesaikan masalah non linear, SVM dapat dimodifikasikan dengan memasukkan fungsi kernel. Dalam non linear SVM, data dipetakan oleh fungsi Ф () ke ruang vector yang berdimensi lebih tinggi. Pada ruang vector yang baru ini, hyperplane yang memisahkan dua kelas tersebut dapat di konstruksikan.

Hal ini sejalan dengan teori Cover yang menyatakan “Jika suatu transformasi bersifat non linear dan dimensi dari feature space cukup tinggi, maka data pada input space dapat dipetakan ke feature space yang baru, dimana pattern-pattern tersebut pada probabilitas tinggi dapat dipisahkan secara linear” (Cover, 1965)

Hyperplane

Gambar II.4 Kernel pada SVM

Pada gambar diatas dapat dilihat bahwa pada kelas kuning dan pada kelas merah yang berada pada input space berdimensi dua tidak dapat dipisahkan secara linear. Selanjutnya ditunjukkan bahwa fungsi Ф memetakan tiap data pada input space tersebut ke ruang vector baru yang berdimensi lebih tinggi (dimensi 3), dimana kedua kelas dapat dipisahkan secara linear oleh sebuah hyperplane. Pemetaan ini dilakukan dengan cara menjaga topologi data, dalam artian dua data yang berjarak dekat pada input space, sebaliknya dua data yang berjarak jauh pada feature space.

**II.4.2 Multi class SVM**

*Hyperplane* yang dihasilkan SVM hanya bisa mengklasifikasikan dua kelas. Sedangkan pada kenyataannya akan banyak ditemukan kasus yang lebih dari dua kelas. Oleh karena itu dapat digunakan *Multiclass* SVM untuk mengklasifikasikan permasalahan yang memiliki lebih dari dua kelas. Terdapat beberapa pendekatan yang dapat menyelesaikan masalah *Multiclass* dengan menggunakan SVM, diantaranya dalah *One Vs All* (OVA) dan *One Vs One* (OVO).

**II.4.2.1 *One Vs All* (OVA)**

Memisahkan permasalahan yang ditemui dari *N* kelas. Sehingga akan dibuat *N decision boundary. Decision boundary* yang dihasilkan merupakan hasil dari pencarian *hyperplane* dari kelas ke *i* dengan kelas sisa yang lainnya. Berikut dilakukan simulasi dengan menggunakan tiga kelas.

A

A

A

A

B

B

B

B

C

C

C

C

Gambar 2.4 Contoh pendekatan OVA

Pada gambar diatas terdapat tiga kelas yaitu kelas A, kelas B, dan kelas C. karena menggunakan metode OVA akan dibuat tiga *decision boundary* dari tiga kelas tersebut. Setiap kelas dicari *decision boundary* dengan kelas sisanya.

A

A

A

A

B

B

B

D1

B

C

C

C

C

Gambar 2.5 *Decision boundary D1*pada pendekatan OVA

Pada gambar 2.5 dibuat *decision boundary* dari kelas A dengan kelas sisanya (kelas B dan C). Kelas A dilabelkan sebagai positif, sedangkan kelas sisanya dilabelkan menjadi negative, sehingga D1 merupakan decision boundary yang dihasilkan dari kelas A dan kelas sisanya (kelas B dan kelas C).

A

A

A

A

B

B

B

D2

B

C

C

C

C

Gambar 2.6 *Decision boundary D2* pada pendekatan OVA

Selanjutnya, sesuai dengan Gambar 2.6, dibuat *decision boundary* dari kelas B dengan kelas sisanya (kelas A dan kelas C). Kelas B dilabelkan sebagai positif, sedangkan kelas sisanya dilabelkan menjadi negatif, sehingga D2 merupakan *decision boundary* yang dihasilkan dari kelas B dan kelas sisanya (kelas A dan kelas C).

B

B

A

A

A

A

D3

B

B

c

c

c

c

Gambar 2.7 *Decision boundary D3* pada pendekatan OVA

Lalu yang terakhir adalah mencari *decision boundary* yang terakhir. Sesuai dengan gambar 2.7, dibuat *decision boundary* dari kelas C dengan kelas sisanya (kelas A dan kelas B). kelas C dilabelkan sebagai positif, sedangkan kelas sisanya dilabelkan menjadi negative, sehingga D3 merupakan *decision boundary* yang dihasilkan dari kelas C dan kelas sisanya (kelas A dan kelas B).

A

A

A

A

B

B

B

D2

D1

B

D3

C

C

C

C

Gambar 2.7 *Decision boundary D3* pada pendekatan OVA

**II.6.2.2 *One Vs One* (OVO)**

Misalkan permasalahan yang ditemui *N* kelas. Sehingga akan dibuat *N(N-1)/2 Decision Boundary, decision boundary* yang dihasilkan merupakan hasil pencarian *hyperplane* dari setiap kelas yang lainnya. Berikut dilakukan simulasi dengan menggunakan tiga kelas.

C

C

C

B

B

B

A

A

B

A

A

C

Gambar 2.8 contoh pendekatan OVO

Pada Gambar 2.8 terdapat tiga kelas, yaitu kelas A, elas B dan kelas C. Karena menggunakan metode OVO dibuat tiga *decision boundary* dari tiga kelas tersebut. Setiap kelas dicari  *decision boundary* dengan masing-masing kelas lainnya.

D1

B

B

A

A

B

A

A

B

Gambar 2.9 *Decision Boundary D1* pada pendekatan OVO

Sesuai dengan Gambar 2.9, dibuat *decision boundary* dari kelas A dengan kelas B. kelas A dilabelkan sebagai positif, sedangkan kelas B dilabelkan menjadi negative, sehingga *D1*merupakan *decision boundary* yang dihasilkan kelas A dan elas B.

B

B

B

B

D2

C

C

C

C

Gambar 2.10 *Decision Boundary D2* pada pendekatan OVO

Lalu selanjutnya, sesuai Gambar 2.10, dibuat *decision boundary* dari kelas B dengan kelas C. Kelas B dilabelkan sebagai positif, sedangkan kelas C dilabelkan menjadi negative, sehingga *D2* merupakan *decision boundary* yang dihasilkan dari kelas B dan kelas C.

A

A

C

C

C

A

A

D3

Gambar 2.11 *Decision Boundary D3* pada pendekatan OVO

Lalu yang terakhir sesuai dengan Gambar 2.11, dibuat *decision boundary* dari kelas A dengan kelas C. Kleas A dilabelkan sebagai positif, sedangkan kelas C dilabelkan menjadi negative *D3*merupakan *decision boundary* yang dihasilkan dari kelas A dan kelas C.

A

D1

B

B

A

A

B

B

A

D3

C

C

D3

C

C

Gambar 2.12 Hasil pendekatan OVO

Sehingga metode OVO menghasilkan area seperti Gambar 2.12, yang dibentuk oleh *decision boundary D1, D2 dan D3.*

**II.4.3**

**Contoh penggunaan SVM**

Misal diberikan *data training* ,} ϵ kelas (-1) dan ϵ kelas (+1), atau dapat digambarkan dengan vitur x1 dan x2 sebagai berikut :

x2

(2,3)

Keterangan :

* berarti yi = - 1

berarti yi = +1

3

(1,1)

1

X1

2

1

## Gambar 3.3 contoh representasi geometric dari data training dalam contoh

Perhatikan bahwa elemen terdekat dari masing – masing kelas adalah elemen Untuk membuat *hyperlane*, maka ambil satu titik yang pasti dilewati yaitu **w** = , yang akan mengimplikasikan vector bobot **w =** .

Ambil ϵ kelas (-1) maka dari persamaan (II.25) yaitu B- : wTxi + b = -1 diperoleh :

wTxi + b = -1 () + b = -1

+ + b = -1

b = -1 - 3 (II. 53)

Ambil ϵ kelas (+1) maka dari persamaan (II.24) yaitu B+ : wTxi + b = 1 diperoleh :

wTxi + b = +1 () + b = 1

+ + b =1

b = 1 - 8 (II. 54)

Dari persamaan (II. 53) dan (II. 54) didapatkan

-1 - 3 = b = 1 - 8

5 = 2

= (II. 55)

Dengan mensubtitusikan persamaan (II. 55) ke persamaan (II. 53), maka diperoleh

b = -1 – 3 = - (II. 56)

dari persamaan (II. 55), maka vector bobot *hyperlane*  menjadi

**w =**  , ) = ( , ) (II. 57)

Sehingga *hyperlane* akan berbentuk sebagai berikut

y (x1, x2) = wTxi + b

y (x1, x2) = ( ) ( ) -

y (x1, x2) = x1 + x2 -

sehingga didapatkan *hyperplane* sebagai berikut

y (x1, x2) = 2x1 + 4x2 = 11