Support Vector Machine

Dr. rer. nat. Hendri Murfi

Intelligent Data Analysis (IDA) Group

Departemen Matematika, Universitas Indonesia – Depok 16424

Telp. +62-21-7862719/7863439, Fax. +62-21-7863439, Email. hendri@ui.ac.id

Machine Learning

Input	Model	Output
x_1 x_2 : x_D	$y(\mathbf{x},\mathbf{w})$	*Klasifikasi, regresi, clustering, dll*

- *Preprocessing*: ekstraksi fitur dan representasi data, misal dalam bentuk vektor $\mathbf{x}_i = (x_1, x_2, .., x_D)^T$
- *Learning*: pemilihan model dan penentuan parameter model, misal **w**, berdasarkan data pelatihan (*training data*)
- *Testing*: pengujian metode dengan data penguji (*testing data*) yang tidak sama dengan data pelatihan, sehingga didapat nilai estimasi untuk kapabilitas generalisasi dari model.

Learning

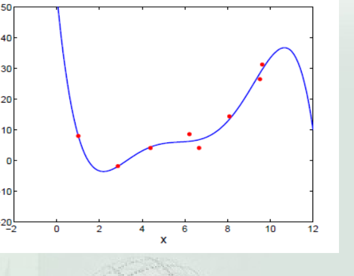
Diberikan data pelatihan $\boldsymbol{x}_i$, *i = 1 sd N,* dan/atau $\boldsymbol{t}_i$, *i = 1 as N*

- Supervised Learning. Data pelatihan disertai target, yaitu {$\mathbf{x}_i$, $\mathbf{t}_i$}, i = 1 sd N. Tujuan pembelajaran adalah membangun model yang dapat menghasilkan output yang benar untuk suatu data input, misal untuk pengklasifikasian (*classification*) dan regresi (*regression*)

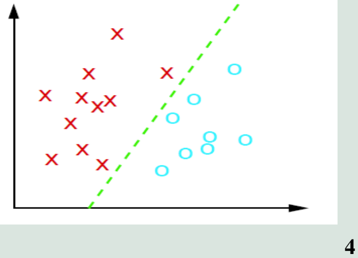
- Unsupervised Learning. Data pelatihan tidak disertai target, yaitu $\mathbf{x}_i$, i = 1 sd N. Tujuan pembelajaran adalah membagun model yang dapat menemukan variabel tersembunyi pada data pelatihan. Selanjutnya, variabel tersembunyi tersebut dapat digunakan untuk kebutuhan pemodelan (*latent variable models*)

Supervised Learning

- Regresi
  - Nilai output $\boldsymbol{t}_i$ bernilai kontinu (riil)
  - Bertujuan memprediksi output dari data baru dengan akurat



- Klasifikasi
  - Nilai output $\boldsymbol{t}_i$ bernilai diskrit (kelas)
  - Bertujuan mengklasifikasi data baru dengan akurat



Model Linear

- Model yang umum digunakan untuk menyelesaikan masalah klasifikasi dan regresi adalah model linear, yaitu model yang merupakan kombinasi linear dari fungsi basis:

$$y(\mathbf{x},\mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{x})$$

  Dimana $\mathbf{x} = (x_1, x_2, ..., x_D)^T$ adalah variabel input, dan $\mathbf{w} = (w_0, w_1, ..., w_D)^T$ adalah parameter, $\phi(\mathbf{x})$ adalah *fungsi basis*, *M* adalah jumlah total parameter dari model

- Biasanya, $\phi_0(\mathbf{x}) = 1$, sehingga w_0 berfungsi sebagai bias

- Ada banyak pilihan yang mungkin untuk fungsi basis $\phi(\mathbf{x})$, misal fungsi linear, fungsi polinomial, fungsi gaussian, fungsi sigmoidal, dll

Model Linear

Kutukan Dimensi

- Model linear memiliki sifat-sifat yang penting baik dari aspek komputasi maupun analitik. Penggunaan model linear dengan pendekatan parametrik pada metode klasik memiliki keterbatasan pada aplikasi praktis disebabkan oleh kutukan dimensi *(curse of dimensionality)*

$$y(x,w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

1

$$y(\mathbf{x},\mathbf{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D}\sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{D} w_{ijk} x_i x_j x_k$$

D^3

untuk model beorde M, maka pertumbuhan jumlah parameter w proposional dengan D^M

Model Linear

Pendekatan Alternatif

- Pendekatan alternatif adalah membuat fungsi basis adaptif terhadap data pelatihan dengan jumlah fungsi basis ditentukan didepan.
- Dengan kata lain, menggunakan bentuk **parametrik tidak linear** dimana nilai-nilai parameter adaptif terhadap data pelatihan selama proses training.
- Contoh metode yang menggunakan pendekatan ini adalah *neural networks* (NN).

Model Linear

Pendekatan Nonparametrik

- Pendekatan lain adalah pendekatan nonparametrik, yaitu menetapkan data pelatihan sebagai pusat-pusat fungsi basis. Selanjutnya, memilih sebagian dari fungsi-funsgi basis tersebut selama proses pelatihan untuk menjadi fungsi-fungsi basis dari model final.
- Dasarnya adalah bahwa data real biasanya memiliki sifat mulus, artinya perubahan sedikit pada data input hanya akan memberikan sedikit perubahan pada output
- Fungsi basis yang banyak digunakan pada pendekatan nonparametrik ini adalah fungsi kernel.

Metode Kernel

Fungsi Kernel : Definisi

- Fungsi kernel adalah suatu fungsi *k* yang mana untuk semua vektor input ***x*,*z*** akan memenuhi kondisi

$$k(\mathbf{x},\mathbf{z}) = \phi(\mathbf{x})^T\phi(\mathbf{z})$$

  dimana $\phi(.)$ adalah fungsi pemetaan dari ruang input ke ruang fitur

- Dengan kata lain, fungsi kernel adalah fungsi perkalian dalam (*inner product*) pada ruang fitur.

Metode Kernel

Fungsi Kernel : Contoh

- Salah satu contoh fungsi kernel yang banyak digunakan adalah *Gaussian radial basis function* (RBF), yaitu:

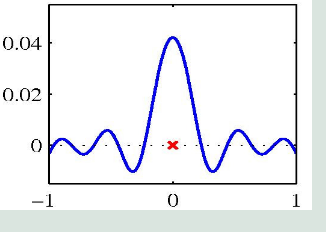
$$k(\mathbf{x},\mathbf{x}') = \phi(||\mathbf{x}-\mathbf{x}'||) = exp(-||\mathbf{x}-\mathbf{x}'||^2 / 2s^2)$$

  dimana **x**' adalah „inti“ yang dipilih dari data pelatihan.

- Contoh: fungsi basis dengan pusat/inti x' = 5 dan bobot w = 0.04 dapat digambarkan sbb:

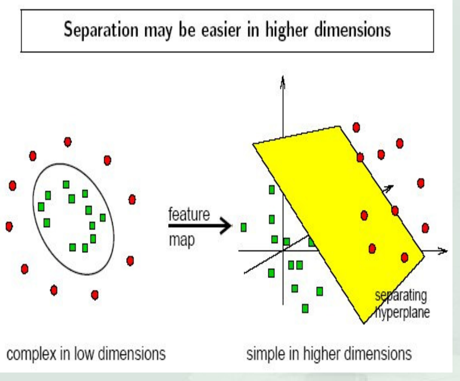
$$w * k(x,5) = 0.04 * \phi(||x-5||)$$

$$= 0.04 * exp(-||x-5||^2 / 2s^2)$$



Metode Kernel

Fungsi Kernel : Keuntungan



- Fungsi kernel memungkinkan kita untuk mengimplementasikan suatu model pada ruang dimensi lebih tinggi (ruang fitur) tanpa harus mendefinisikan fungsi pemetaan dari ruang input ke ruang fitur
- Sehingga, untuk kasus yang *non-linearly separable* pada ruang input, diharapkan akan menjadi *linearly separable* pada ruang fitur
- Selanjutnya, kita dapat menggunakan *hyperplane* sebagai *decision boundary* secara efisien

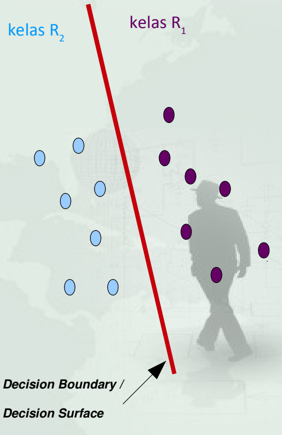
Metode Kernel

Fungsi Kernel : Penggunaan

- Secara umum, ada dua cara penggunaan metode kernel pada *machine learning*, yaitu:
  - Penggunaan langsung, yaitu fungsi kernel digunakan sebagai fungsi basis dari model machine learning tersebut, contoh: *radial basis function networks*
  - Penggunaan tidak langsung melalui *kernel trick*, yaitu merepresentasikan suatu model kedalam representasi dual yang mengandung *inner product* dari fungsi pemetaan, contoh: *kernel linear regression*, *kernel Perceptron*, ***support vector machine***, dll

Support Vector Machine

Two-Class Classification Problem



- Diberikan data pembelajaran $\{\mathbf{x}_n, t_n\}$, $n = 1$ sd N, yang diklasifikasikan dalam dua kelas, yaitu kelas R_1 $(t_n = +1)$ dan kelas R_2 $(t_n = -1)$
- Permasalahan: bagaimana menentukan *decision boundary* yang dapat mengklasifikasikan data dengan benar

Support Vector Machine

Bentuk Umum

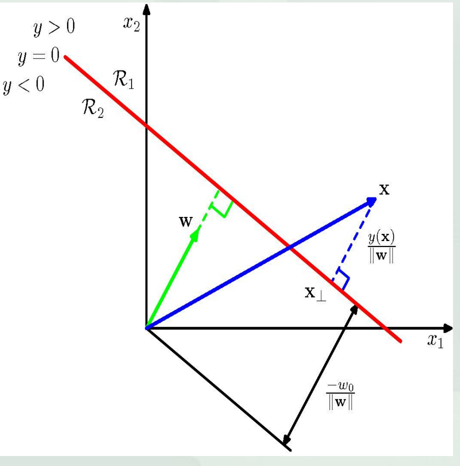
- *Support Vector Machine (SVM)* menggunakan model linear sebagai *decision boundary* dengan bentuk umum sbb:

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$$

dimana $\mathbf{x}$ adalah vektor input, $\mathbf{w}$ adalah parameter bobot, $\phi(\mathbf{x})$ adalah fungsi basis, dan b adalah suatu bias

Support Vector Machine

Hyperplane



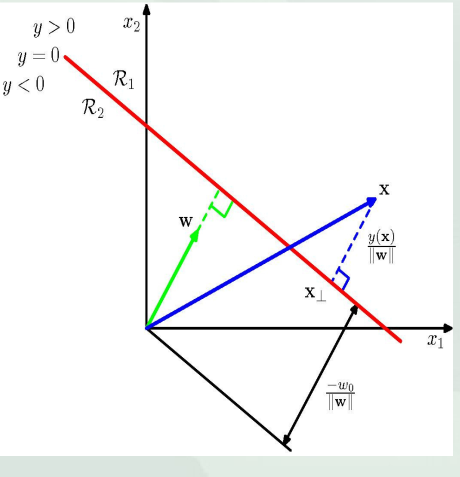
- Bentuk model linear yang paling sederhana untuk *decision boundary* adalah:

  $$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

  Dimana $\mathbf{x}$ adalah vektor input, $\mathbf{w}$ adalah vektor bobot dan w_0 adalah bias.
- Sehingga, *decision boundary* adalah $y(\mathbf{x})=0$, yaitu suatu *hyperplane* berdimensi $(D-1)$
- Suatu vektor input $\mathbf{x}$ akan diklasifikasikan ke kelas 1 (R_1) jika $y(\mathbf{x})\geq 0$, dan kelas 2 (R_2) jika $y(\mathbf{x})<0$

Support Vector Machine

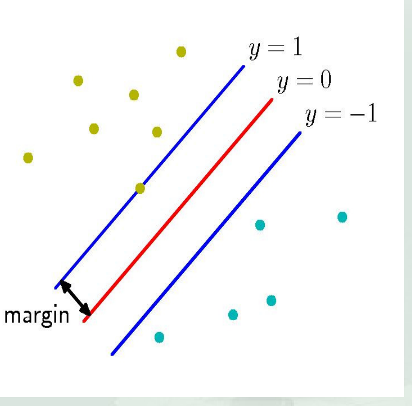
Sifat-Sifat Hyperplane



- Jika $\mathbf{x}_A$ dan $\mathbf{x}_B$ terletak pada *decision boundary* (DS), maka $y(\mathbf{x}_A)=y(\mathbf{x}_B)=0$ atau $\mathbf{w}^T(\mathbf{x}_A-\mathbf{x}_B)=0$, sehingga **w** tegak lurus terhadap semua vektor di DS. Dengan kata lain **w** menentukan orientasi dari DS
- Jarak titik awal ke DS adalah $-w_0/||\mathbf{w}||$. Dengan kata lain w_0 menentukan lokasi DS.
- Jarak sembarang vektor **x** ke DS dan searah **w** adalah $y(\mathbf{x})/||\mathbf{w}||$

Support Vector Machine

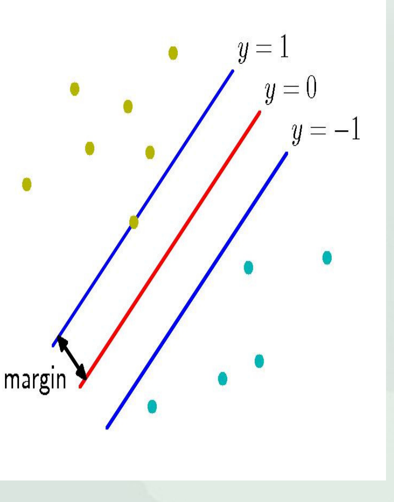
Maximum Margin



- Untuk menentukan *decision boundary* (DB), yaitu suatu model linear atau *hyperplane* y(**x**) dengan parameter **w** dan b, SVM menggunakan konsep *margin* yang didefiniskan sebagai jarak terdekat antara DB dengan sembarang data training
- Dengan memaksimumkan *margin*, maka akan didapat suatu DB tertentu

Support Vector Machine

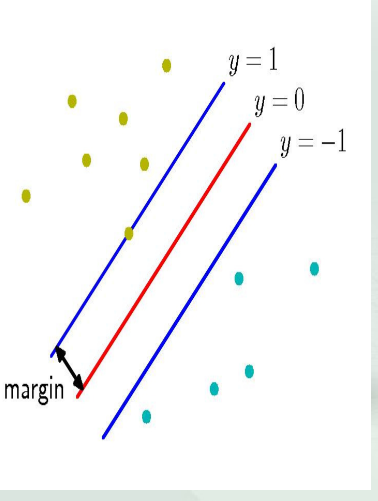
Maximum Margin



- Kenapa maksimum ?
- Berdasarkan intuisi, margin maksimum adalah pilihan yang aman karena jika terjadi sedikit kesalahan pada data maka akan memberikan kemungkinan terkecil terjadi kesalahan klasifikasi
- Berdasarkan teori, yang merupakan basis dari metode SVM, maksimum margin akan memberikan kapabilitas generalisasi terbaik (VC theory, 1960-1990)

Support Vector Machine

Maximum Margin



- Asumsikan semua data terklasifikasi dengan benar (*linearly sparable*), maka jarak antara sembarang data $\mathbf{x}_n$ ke *decision boundary* (DB) adalah:

$$\frac{t_n y(x_n)}{\|w\|} = \frac{t_n(w^T \phi(x_n)+b)}{\|w\|}$$

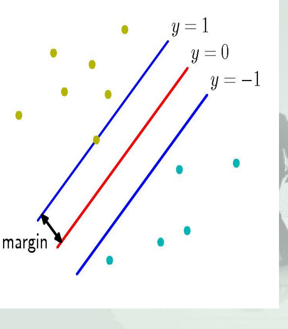
- Margin adalah jarak antara DB dan data terdekat, sehingga memaksimumkan margin dapat dideskripsikan sbb:

$$\underset{w,b}{arg\,max} \left\{\frac{1}{\|w\|} \min_n [t_n(w^T \phi(x_n)+b)]\right\}$$

Support Vector Machine

Maximum Margin

- Solusi langsung dari masalah optimasi sebelumnya akan sangat kompleks, sehingga perlu dikonversi ke masalah yang ekivalen yang lebih mudah diselesaikan



- Salah satu metode adalah menggunakan bentuk kanonik dari DB, yaitu:

$$t_n(w^T \phi(x_n)+b) = 1$$

untuk data yang terdekat ke DB. Selanjutnya, semua data pembelajaran akan memenuhi kondisi berikut ini:

$$t_n(w^T \phi(x_n)+b) \geqslant 1$$

Support Vector Machine

Quadratic Programming

- Selanjutnya, masalah optimasi sebelumnya dapat disederhanakan menjadi memaksimum 1/||w|| yang ekivalen dengan meminimumkan $||w||^2$
- Sehingga masalah optimasi-nya menjadi (bentuk *primal*):

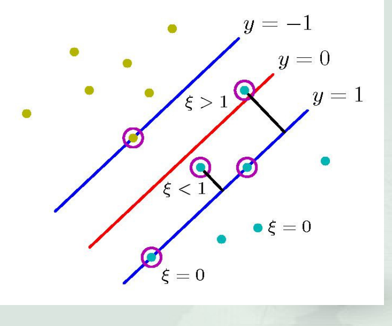
$$\arg\min_{w,b} \frac{1}{2}||w||^2$$

$$s.t. \quad t_n(w^T \phi(x_n)+b) \geqslant 1, \quad n=1,..,N$$

- Dengan kata lain, penentuan nilai parameter **w** dan b menjadi masalah pemrograman kuadrat (*quadratic programming*), yaitu meminimumkan suatu fungsi kuadrat dengan syarat suatu pertidaksamaan linear

Support Vector Machine

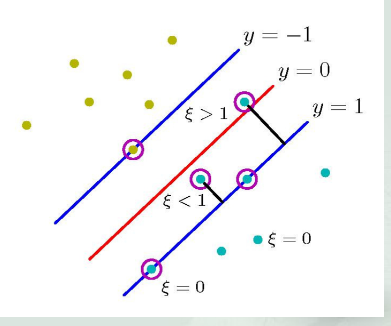
Soft Margin: Landasan



- Diberikan data pembelajaran $\{\mathbf{x}_n, t_n\}$, $n = 1$ sd N, $t_n \in \{-1,+1\}$.
- Pada formulasi masalah sebelumnya, semua data diasumsikan *linearly separable* pada ruang fitur $\phi(\mathbf{x})$.
- Dalam aplikasi praktis, kondisi tersebut sering tidak terpenuhi bahkan setelah data di transformasi ke ruang fitur $\phi(\mathbf{x})$.
- Masalah ini dipecahkan dengan memodifikasi formulasi masalah sebelumnya dengan cara membuat margin lunak (soft margin) yang memungkinkan beberapa data pada „posisi yang salah“

Support Vector Machine

Soft Margin: Formulasi Masalah



- Untuk merealisasikan *soft margin* ini, diperkenalkan variabel *slack*, $\xi_n \geq 0$, n=1, ..., N, dengan satu variabel *slack* untuk masing-masing data pembelajaran
- Variabel *slack* tersebut bernilai $\xi_n = |t_n - y(\mathbf{x}_n)|$
- Sehingga, $\xi_n = 0$ adalah data yang terletak pada *margin* dan sisi yang benar, $0 < \xi_n \leq 1$ adalah data yang terletak didalam margin pada sisi yang benar, dan $\xi_n > 1$ adalah data yang terletak pada sisi yang salah

Support Vector Machine

Soft Margin: *Quadratic Programming*

- Bentuk primal dari masalah optimasi sebelumnya (*hard margin*) adalah:

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2$$
$$s.t. \quad t_n(w^T \phi(x_n) + b) \geqslant 1, \quad n = 1, .., N$$

maka bentuk primal dari masalah optimasi untuk *soft margin* adalah:

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n$$
$$s.t. \quad t_n(w^T \phi(x_n) + b) \geqslant 1 - \xi_n, \quad n = 1, .., N$$
$$\xi_n \geqslant 0$$

dimana parameter C > 0 akan mengkontrol *trade-off* antara pinalti variabel *slack* dan *margin*

Support Vector Machine

Bentuk Dual : *Lagrange Multipliers*

Untuk menyelesaikan pemrograman kuadrat tersebut, cara yang umum digunakan adalah mencari bentuk dual dengan menggunakan perkalian Lagrange (*Lagrange multipliers*) $a_n \geq 0$, dengan satu pengali Lagrange untuk setiap kendala, untuk membentuk fungsi Lagrangian (*Lagrangian function*) sbb:

$$L(w,b,a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x_n)+b)-1+\xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

Support Vector Machine

Bentuk Dual : *Lagrange Multipliers*

$$L(w,b,a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x_n)+b)-1+\xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

- Dengan menurunkan *L(**w**, b, **a**)* terhadap ***w*** dan *b* sama dengan nol, maka:

$$\frac{\partial L}{\partial w} = w - \sum_{n=1}^{N} a_n t_n \phi(x_n) = 0 \quad \rightarrow \quad w = \sum_{n=1}^{N} a_n t_n \phi(x_n)$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{N} a_n t_n = 0 \quad \rightarrow \quad 0 = \sum_{n=1}^{N} a_n t_n$$

$$\frac{\partial L}{\partial \xi_n} = C - a_n - \mu_n = 0 \quad \rightarrow \quad a_n = C - \mu_n$$

Support Vector Machine

Bentuk Dual : Fungsi Kernel

- Substitusikan hasil turunan tsb ke persamaan Lagrangian, yaitu:

$$L(w,b,a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x_n)+b)-1+\xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

sehingga menjadi:

$$\tilde{L}(a) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_n) - \sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_n) - b\sum_{n=1}^{N}\sum_{m=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n$$

$$= \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_m)$$

$$= \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$

Support Vector Machine

Bentuk Dual

- Sehingga, bentuk *dual* dari masalah *soft margin* maksimum adalah:

$$\arg\max_a \ \tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$

$$s.t. \quad 0 \leqslant a_n \leqslant C, \quad n=1,..,N$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- Bentuk dual yang dihasilkan juga berupa pemrograman kuadrat dengan kendala yang lebih sederhana (*bound-constrained optimization*).

- Terlihat bahwa masalah pemrogramman kuadrat dari *hard margin* dan *soft margin* adalah identik, kecuali bagian kendalanya yang sedikit berbeda

Support Vector Machine

Solusi Bentuk Dual : Algoritma

- Ada beberapa algoritma dan perangkat lunak yang telah dikembangkan untuk memecahkan masalah optimisasi dari SVM, antara lain:
  - SMO [2]
  - LibSVM [3] (http://www.csie.ntu.edu.tw/~cjlin/libsvm)
  - SVM^{Light} [4] (http://svmlight.joachims.org)

Support Vector Machine

Solusi Bentuk Dual : Nilai Bobot

- Misal solusi dari pemrograman kuadrat bentuk dual tersebut adalah $\boldsymbol{a}^*$, maka:

$$w = \sum_{n=1}^{N} a_n^* t_n \phi(x_n)$$

$$y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} a_n^* t_n \phi(x_n)^T \phi(x) + b = \sum_{n=1}^{N} a_n^* t_n k(x, x_n) + b$$

- Hanya data dengan $a_n^* > 0$ (*support vectors*) yang berperan pada model *decision boundary* diatas, sehingga dapat ditulis menjadi:

$$y(x) = \sum_{m \in S} a_m^* t_m k(x, x_m) + b$$

dimana *S* adalah himpunan indeks dari *support vectors*

Support Vector Machine

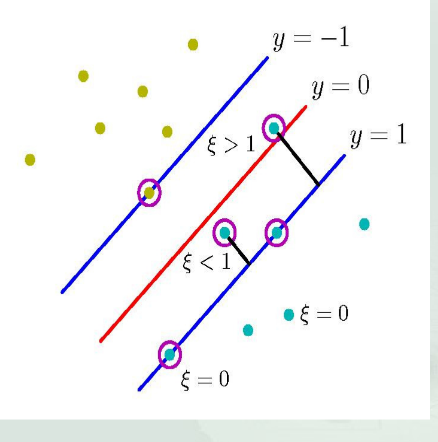
Solusi Bentuk Dual : Kondisi *Karush-Kuhn-Tucker*

Karush-Kuhn-Tucker (KKT) conditions adalah kondisi-kondisi yang menyatakan bahwa solusi yang diperoleh adalah optimal, atau pada solusi tersebut variabel dan kendala dari bentuk dual bertemu, adalah sbb:

$$
\begin{aligned}
a_n &\geqslant 0 && [KKT-S1] \\
t_n y(x_n) - 1 + \xi_n &\geqslant 0 && [KKT-S2] \\
a_n\{t_n y(x_n) - 1 + \xi_n\} &= 0 && [KKT-S3] \\
\mu_n &\geqslant 0 && [KKT-S4] \\
\xi_n &\geqslant 0 && [KKT-S5] \\
\mu_n \xi_n &= 0 && [KKT-S6]
\end{aligned}
$$

Support Vector Machine

Solusi Bentuk Dual : *Support Vectors*



- *Support vectors* adalah data training dengan nilai $a_n^* > 0$, maka dari [KKT-S3]

$$t_n y(x_n) = 1 - \xi_n$$

- Jika $a_n^* < C$, maka berdasarkan hasil turunan bahwa $a_n = C - \mu_n$ maka $\mu_n > 0$. Selanjutnya, berdasarkan [KKT-S6] $\mu_n \xi_n = 0$ maka $\xi_n = 0$. Dengan kata lain adalah data training yang terletak pada hyperplane
- Jika $a_n^* = C$, maka $\mu_n = 0$. Dari [KKT-S6] maka $\xi_n \neq 0$. Dengan kata lain adalah data training yang terletak didalam margin, baik yang terklasifikasi dengan benar ($\xi_n \leq 1$) atau salah ($\xi_n > 1$)

Support Vector Machine

Solusi Bentuk Dual : Nilai Bias

- Selanjutnya, *b* dapat dicari dengan cara sbb:

$$t_n y(x_n) = 1$$

$$t_n\left(\sum_{m \in S} a_m^* t_m k(x_n, x_m) + b\right) = 1 \rightarrow b = \frac{1}{N_S} \sum_{n \in S}\left(t_n - \sum_{m \in S} a_m^* t_m k(x_n, x_m)\right)$$

dimana *S* adalah himpunana indeks dari *support vectors,* dan N_S adalah jumlah semua *support vectors*

Support Vector Machine

Prediksi Data Baru

- Misal diberikan data ***z***, maka prediksi dari data tersebut ditentukan berdasarkan fungsi *y(**x**)*, yaitu:
  - Jika *y(**z**) ≥ 0* maka ***z*** adalah kelas C_1 (t = +1)
  - Jika *y(**z**) < 0* maka ***z*** adalah kelas C_2 (t = -1)

Support Vector Machine

Multi-Class Classification

- Muti-class classification adalah masalah klasifikasi yang memiliki jumlah kelas lebih dari 2. Sementara SVM standar didesain untuk masalah two-class classification. Ada beberapa teknik yang memungkinkan penggunaan SVM standar two-class untuk masalah multi-class, misal K kelas, yaitu:
  - *One-vs-The Rest*, yaitu membangun *K* buah SVM, dimana model ke-k, yaitu $y_k(\mathbf{x})$, dilatih dengan menggunakan data dari kelas C_k sebagai sampel positip (+1) dan data dari kelas yang lain sebagai sampel negatip (-1). Contoh: $y_1(\mathbf{x})$ akan memisahkan antara kelas 0 dan kelas-kelas lainnya (1,2,3,4)
  - *One-vs-One*, yaitu membangun *K(K-1)/2* buah SVM yang merupakan semua kemungkinan pasangan kelas, selanjutnya suatu data penguji akan diklasifikasikan ke kelas yang menang paling banyak. Contoh: $y_1(\mathbf{x})$ akan memisahkan kelas 0 dan kelas 1, $y_2(\mathbf{x})$ akan memisahkan kelas 1 dan kelas 2, dst

Support Vector Machine

Black Box

- Pada pembahasan SVM pada presentasi ini, beberapa bagian masih bersifat *black box*, yaitu:
  - ***VC Theory***, yaitu teori yang mendasari metode margin maksimum yang menunjukkan bahwa margin maksimum akan memberikan generalisasi error terkecil [1]
  - Algoritma penyelesaian masalah pemrograman kuadrat (bentuk dual dari *soft margin*), misal algoritma SVM^{Light} [2], SMO [3], LibSVM [4]

Referensi

(1) C. H. Bishop. *Pattern Recognition and Machine Learning*, Springer, 2006 (Bab 7.1, Bab 7.1.1, Bab 7.1.3, Appendix E)

(2) J. C. Platt. *Fast training of support vector machines using sequential minimal optimization*. In B. Schoelkopf, C. J. C. Burges, and A. J. Smola (Eds), Advances in Kernel Methods – Support Vector Learning, pp. 185-208, MIT Press, 1999

(3) R. E. Fan, P. H. Chen, C. J. Lin. *Working set selection using second order information for training SVM*. Journal of Machine Learning Research 6, 1889-1918, 2005

(4) T. Joachim. *Making Large-Scale SVM Learning Practical*. In B. Schoelkopf, C. J. C. Burges, and A. J. Smola (Eds), Advances in Kernel Methods – Support Vector Learning, pp. 169-184, MIT Press, 1999