Pertemuan 7

KLASIFIKASI DENGAN SVM, SELEKSI DAN EVALUASI MODEL

1

Sejarah Support Vector Machine (SVM)

- SVM adalah algoritme *supervised learning* yang dikembangkan oleh Vladimir Vapnik dan diperkenalkan pertama kali pada tahun 1992 oleh Vapnik, Boser dan Guyon pada Computational Learning Theory (*COLT*) *1992* [*]
- (dikatakan Vladimir Vapnik telah mengemukan idenya pada tahun 1979 di salah satu papernya, tetapi mulai berkembang pada tahun 90-an)

*) Vikramaditya Jakkula : "Tutorial on Support vector machines" school of EECS Washington State University

Pendekatan dengan *Machine Learning*
Terminologi dasar

- Training set: N data $\{x_{1, \ldots, x_N}\}$
  - Digunakan untuk tuning parameter dari model
  - Kategori dari data *training set* telah diketahui sebelumnya
- Target vector: vektor unik t untuk tiap target
  - Merepresentasikan identitas dari data yang bersesuaian

Pendekatan dengan *Machine Learning*
Terminologi dasar

- Learned function: y(x)
  - Training phase (learning phase) → proses untuk menentukan y(x) berdasarkan *training data*
- Test set: data yang tidak terdapat pada *training set*
  - Setelah model selesai dilatih, model dapat menentukan kategori dari data baru
  - Kemampuan untuk mengkategorikan dengan benar data baru yang berbeda dengan data yang digunakan pada training set → GENERALISASI

Definisi Masalah :

- Diberikan sekumpulan n titik (vectors) :
  $x_1, x_2, \ldots\ldots x_n$ di mana x_i adalah vektor dengan panjang *m* dan masing-masing adalah anggota salah satu dari dua kelas yang memiliki label "+1" and "-1".

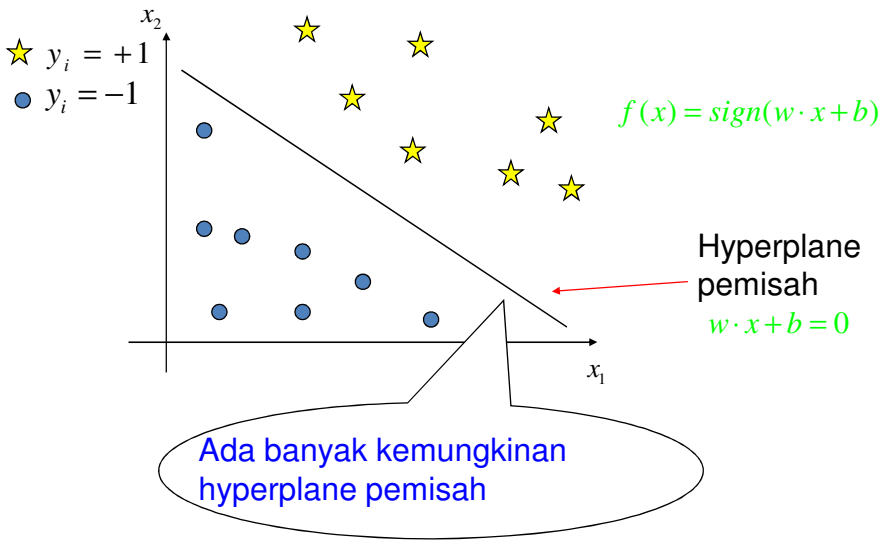
-Maka training set nya adalah :

$$(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$$
$$\forall i \;\; x_i \in R^m, \; y_i \in \{+1, -1\}$$

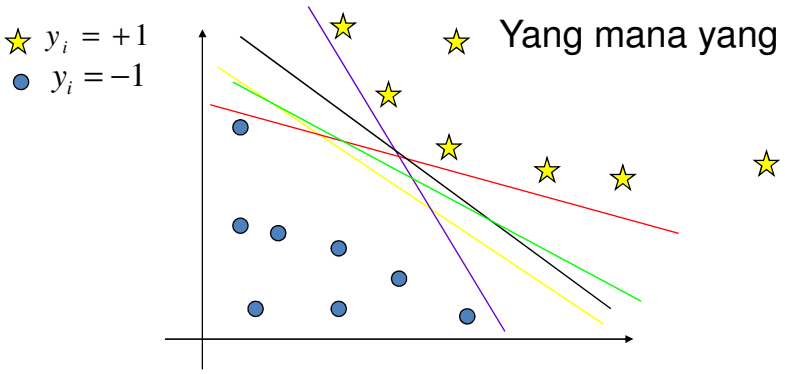
Fungsi keputusannya berupa
$f(x) = sign(w \cdot x + b)$

- Diinginkan untuk menemukan hyperplane $w \cdot x + b = 0$
Yang memisahkan titik-2 tersebut ke dalam dua kelas
"Positif" (kelas "+1") dan "Negatif" (kelas "-1").
(Diasumsikan titik-titik tersebut dapat dipisahkan secara linier)

Hyperplane pemisah



Hyperplane pemisah

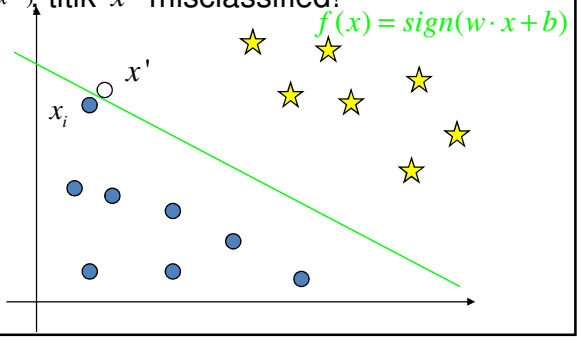


Ada banyak hyperplane pemisah yang mungkin

Memilih hyperplane pemisah:

- Misal dipilih *hyperplane* (di bawah ini) yang dekat dengan sampel x_i
- Selanjutnya misal terdapat titik baru x' yang merupakan kelas "-1" dan dekat dengan x_i . Dengan menggunakan fungsi klasifikasi $f(x)$, titik x' misclassified!

Generalisasi yang buruk

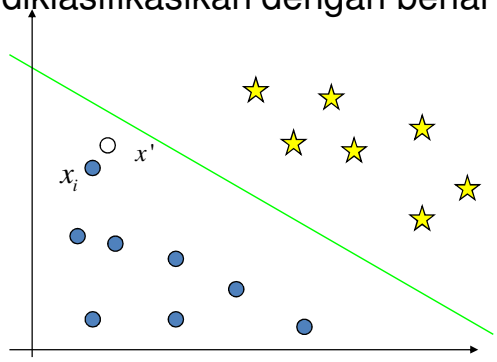


Memilih hyperplane pemisah:

Pendekatan SVM: Kasus Linear separable

- Hyperplane seharusnya sejauh mungkin dari titik sampel
- Dengan demikian data baru yang dekat dengan sampel data akan diklasifikasikan dengan benar

Generalisasi yang baik !



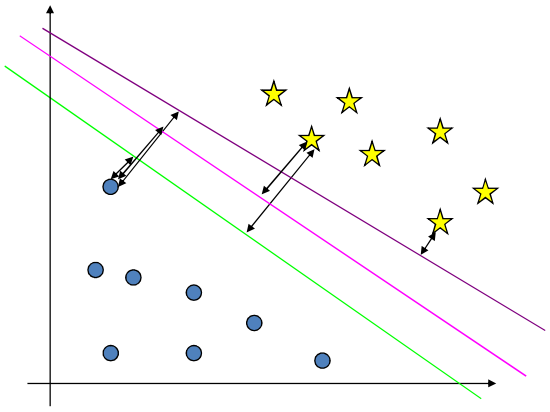
Memilih hyperplane pemisah

Pendekatan SVM: kasus Linear separable

- Ide dasar dari SVM adalah memaksimalkan distance (jarak) antara *hyperplane* dan titik sampel terdekat

Pada optimal hyper-plane:

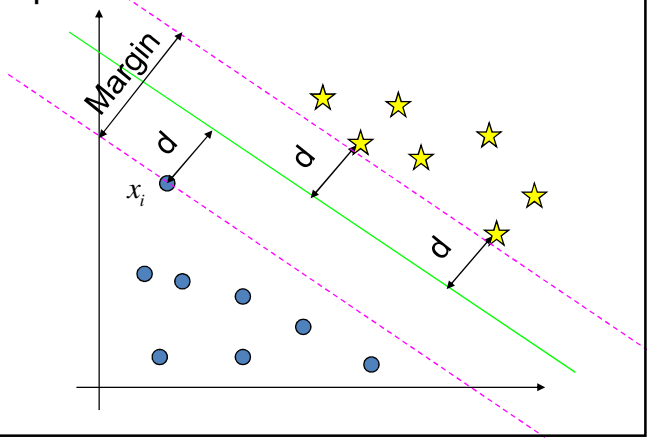
Jarak ke titik negatif terdekat = jarak ke titik positif terdekat

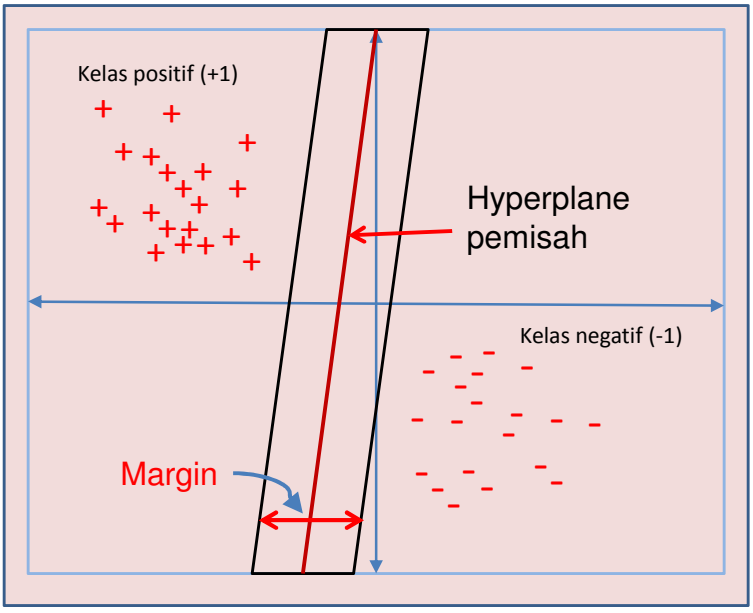


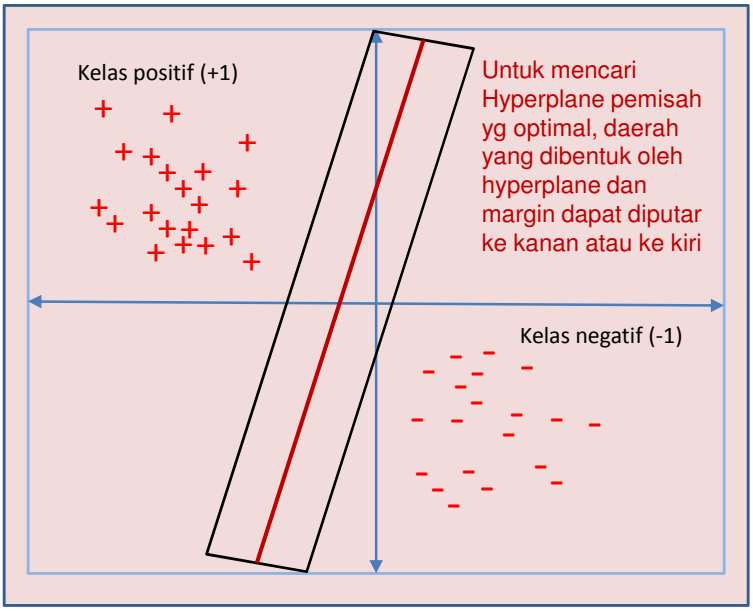
Memilih hyperplane pemisah:

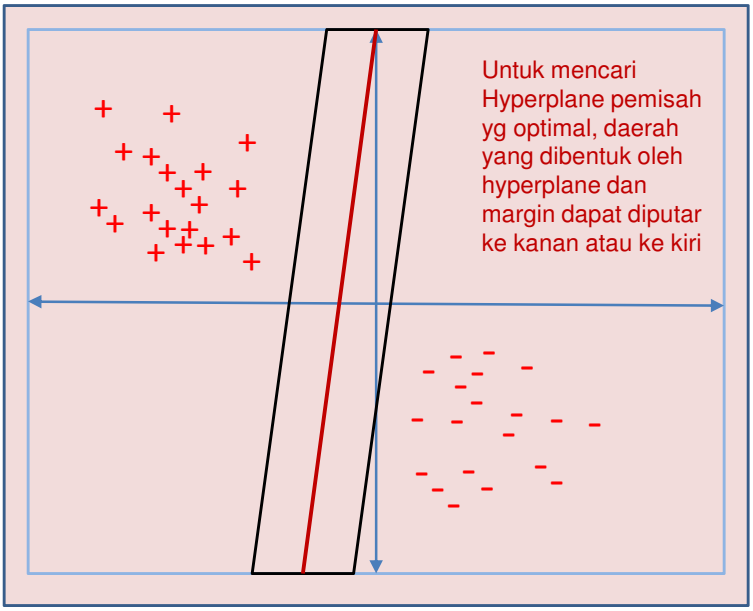
Pendekatan SVM: kasus Linear separable

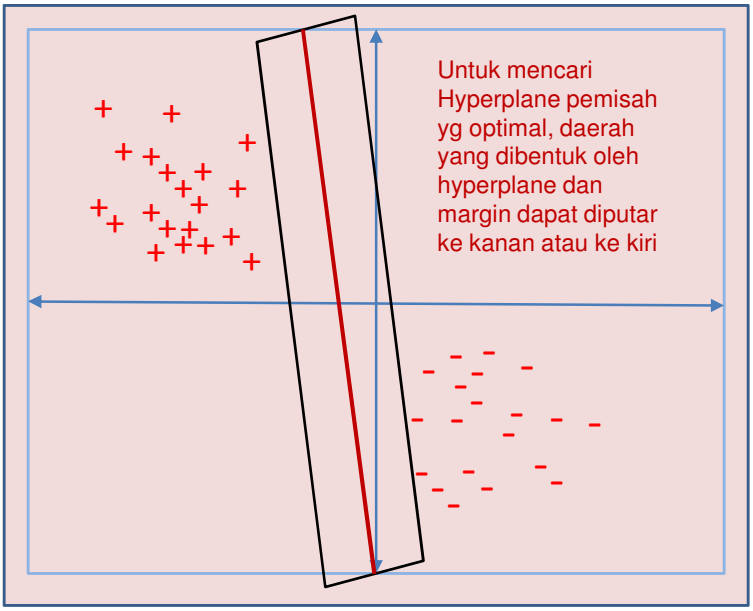
Tujuan SVM adalah memaksimalkan Margin yang besarnya dua kali jarak "d" antara hyperplane pemisah dan sampel terdekat.

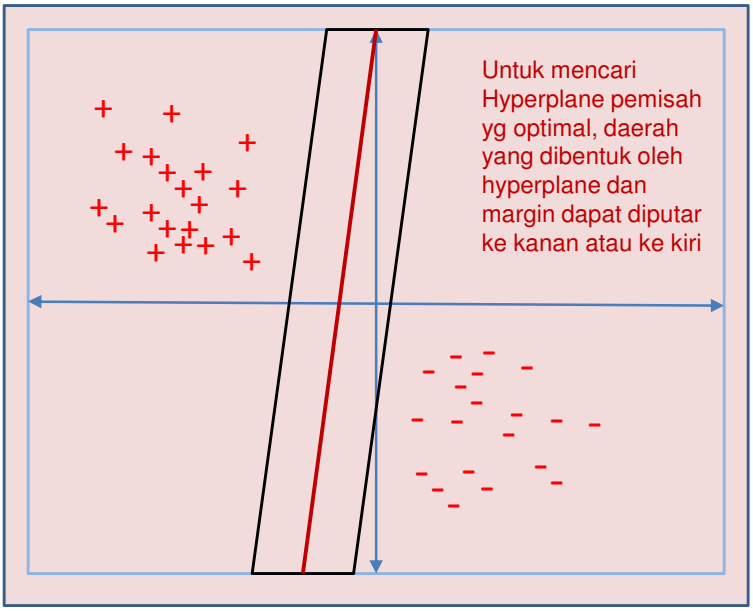


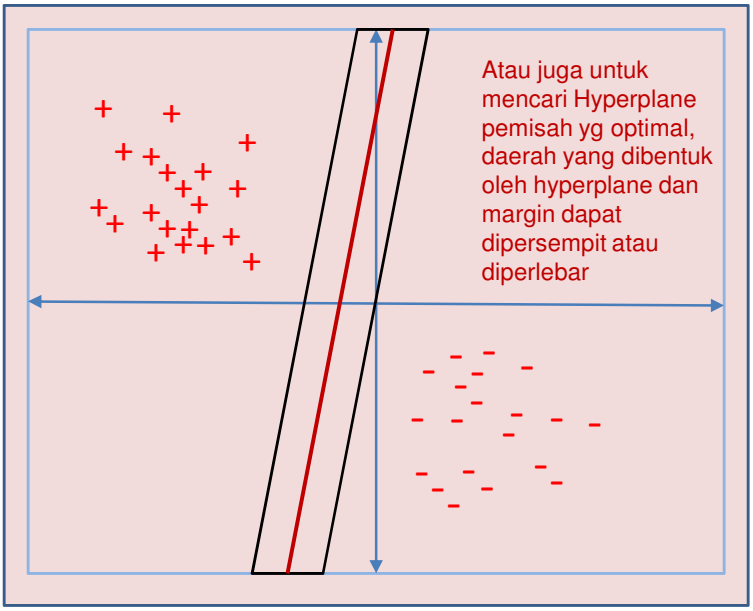


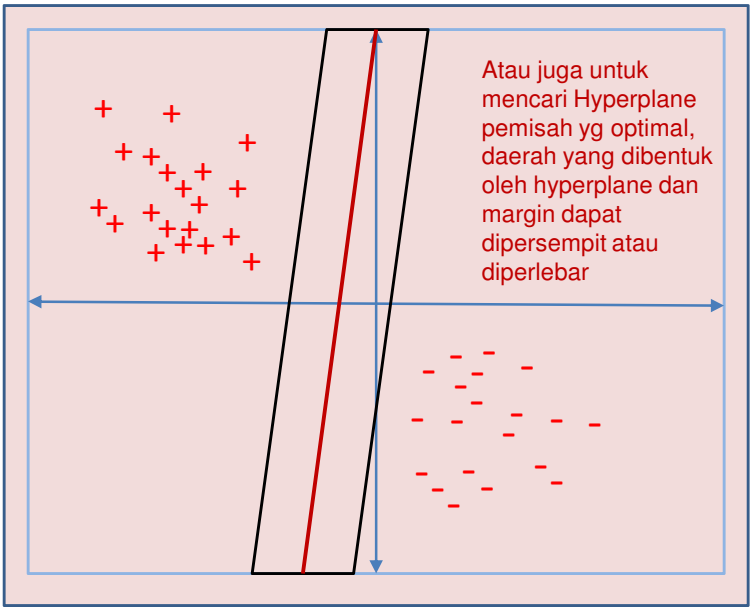

Kelas positif (+1)
Untuk mencari Hyperplane pemisah yg optimal, daerah yang dibentuk oleh hyperplane dan margin dapat diputar ke kanan atau ke kiri
Kelas negatif (-1)

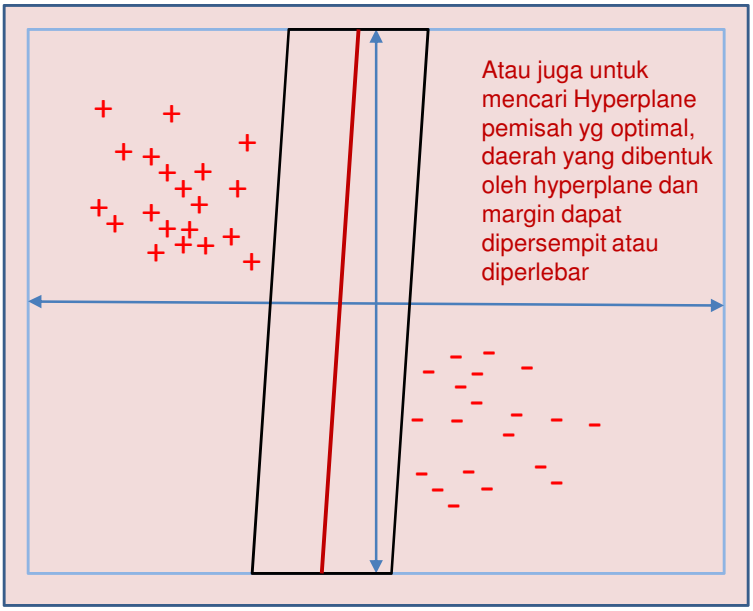

Untuk mencari Hyperplane pemisah yg optimal, daerah yang dibentuk oleh hyperplane dan margin dapat diputar ke kanan atau ke kiri

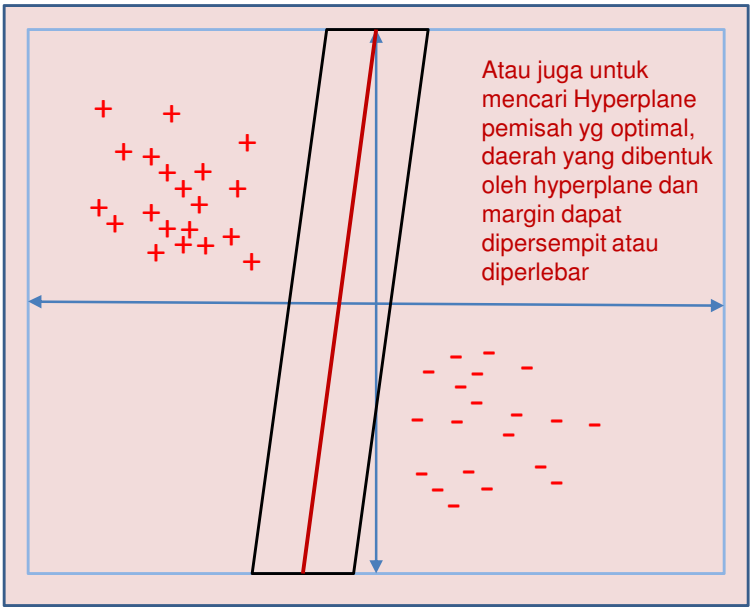

Untuk mencari
Hyperplane pemisah
yg optimal, daerah
yang dibentuk oleh
hyperplane dan
margin dapat diputar
ke kanan atau ke kiri

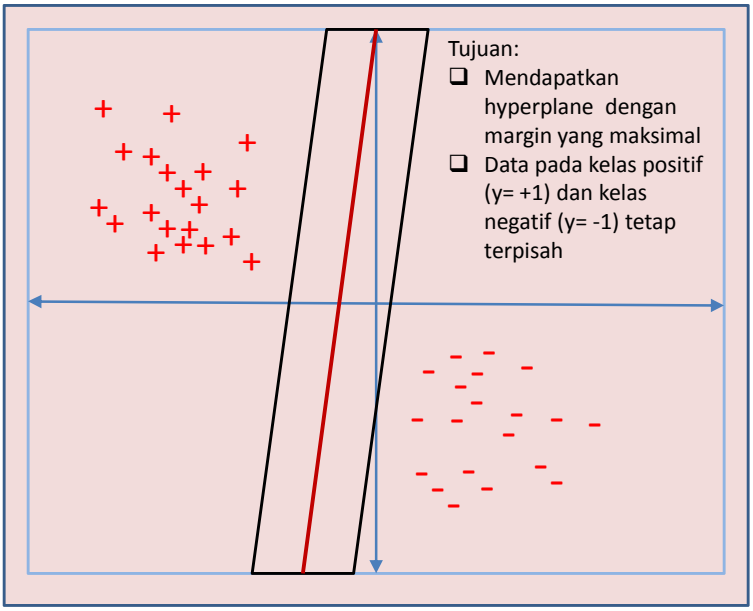

Untuk mencari
Hyperplane pemisah
yg optimal, daerah
yang dibentuk oleh
hyperplane dan
margin dapat diputar
ke kanan atau ke kiri

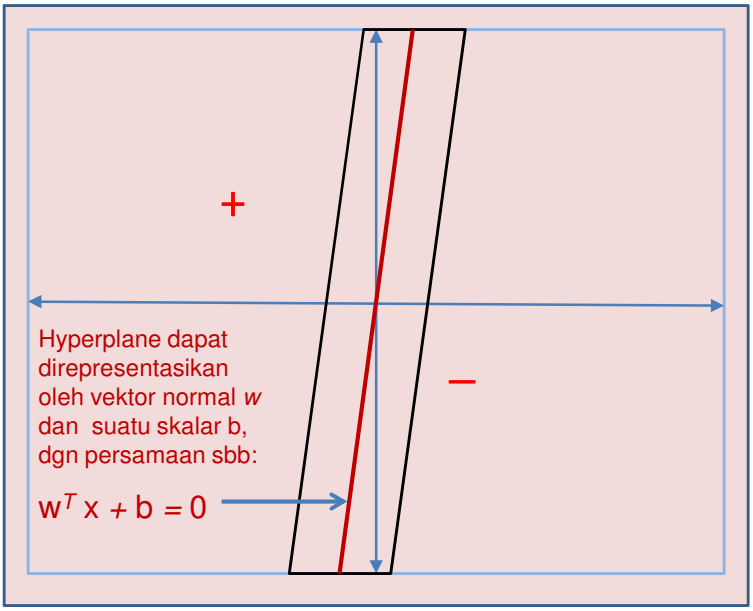

Atau juga untuk mencari Hyperplane pemisah yg optimal, daerah yang dibentuk oleh hyperplane dan margin dapat dipersempit atau diperlebar

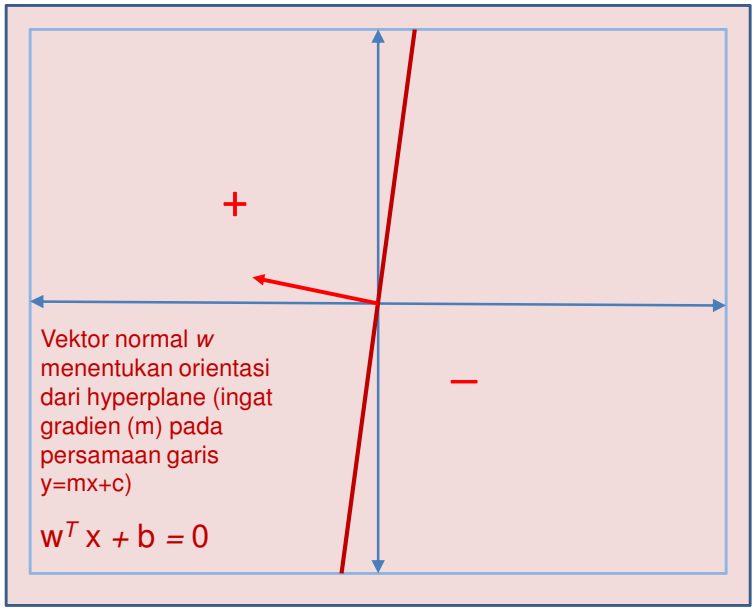

Atau juga untuk mencari Hyperplane pemisah yg optimal, daerah yang dibentuk oleh hyperplane dan margin dapat dipersempit atau diperlebar

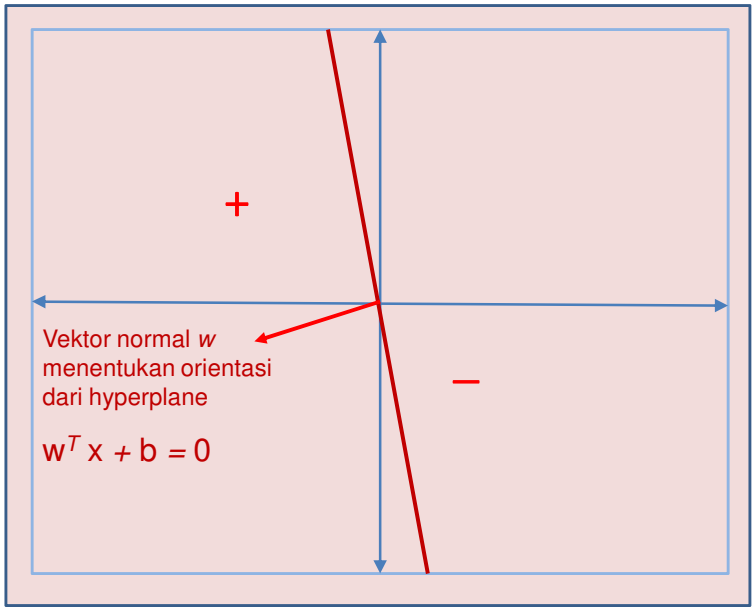

Atau juga untuk mencari Hyperplane pemisah yg optimal, daerah yang dibentuk oleh hyperplane dan margin dapat dipersempit atau diperlebar

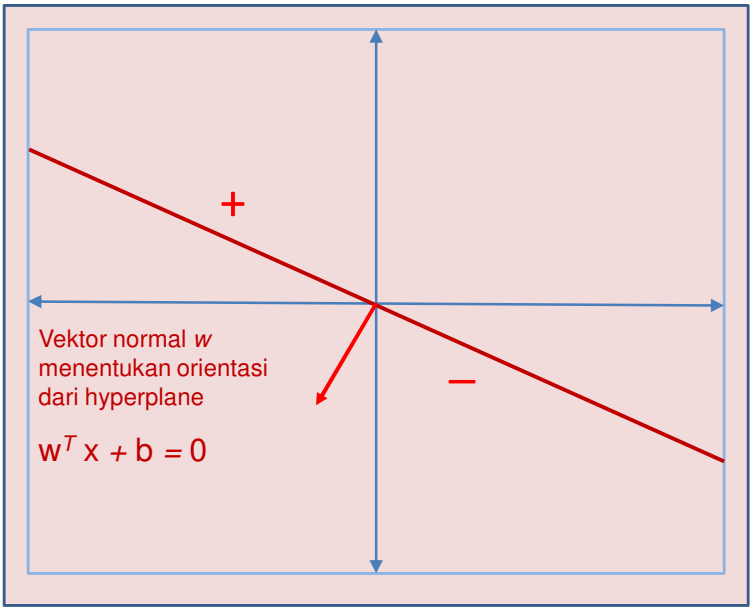

Atau juga untuk mencari Hyperplane pemisah yg optimal, daerah yang dibentuk oleh hyperplane dan margin dapat dipersempit atau diperlebar

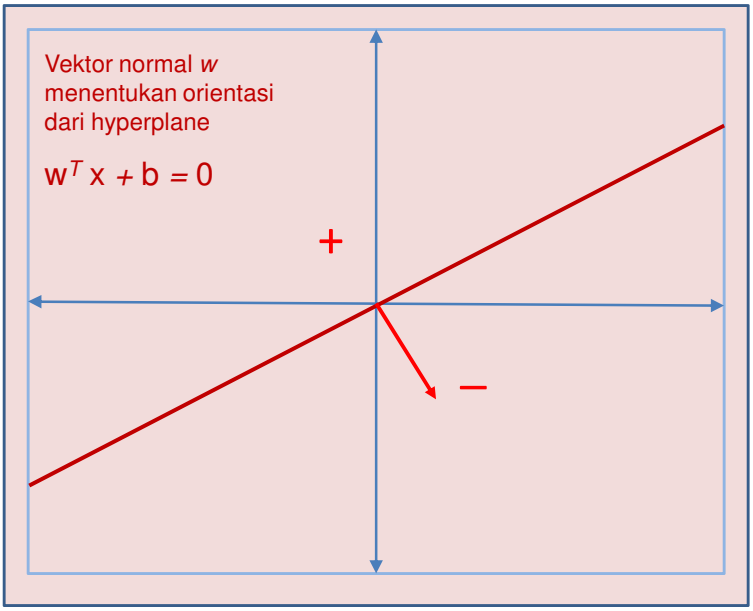

Tujuan:
❑ Mendapatkan hyperplane dengan margin yang maksimal
❑ Data pada kelas positif (y= +1) dan kelas negatif (y= -1) tetap terpisah

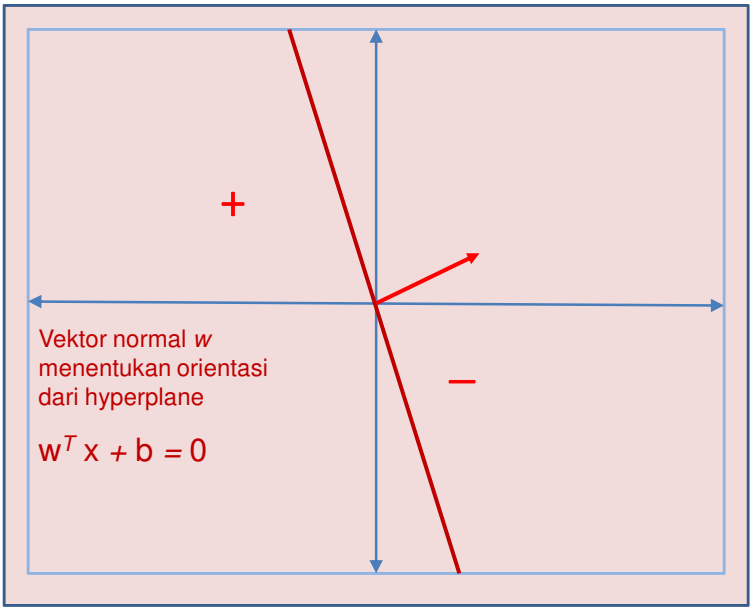

+
−
Hyperplane dapat direpresentasikan oleh vektor normal *w* dan suatu skalar b, dgn persamaan sbb:
$w^T x + b = 0$

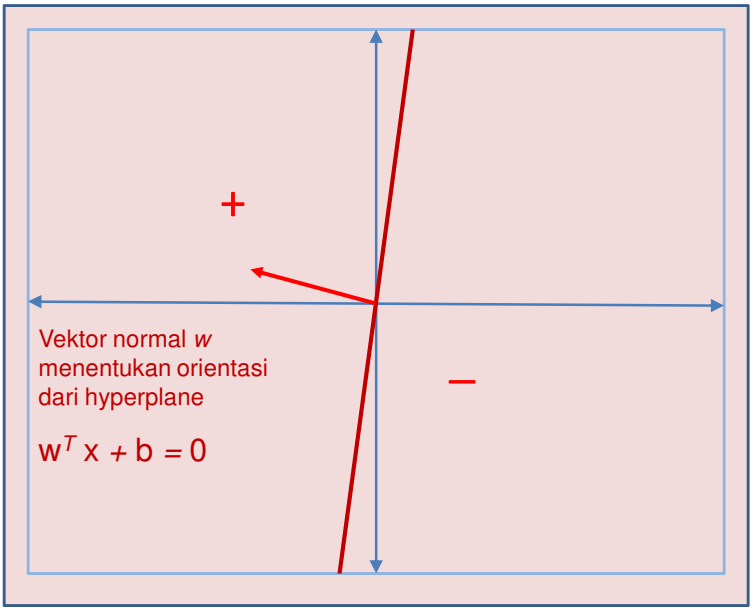

+
−
Vektor normal *w* menentukan orientasi dari hyperplane (ingat gradien (m) pada persamaan garis y=mx+c)
$w^T x + b = 0$

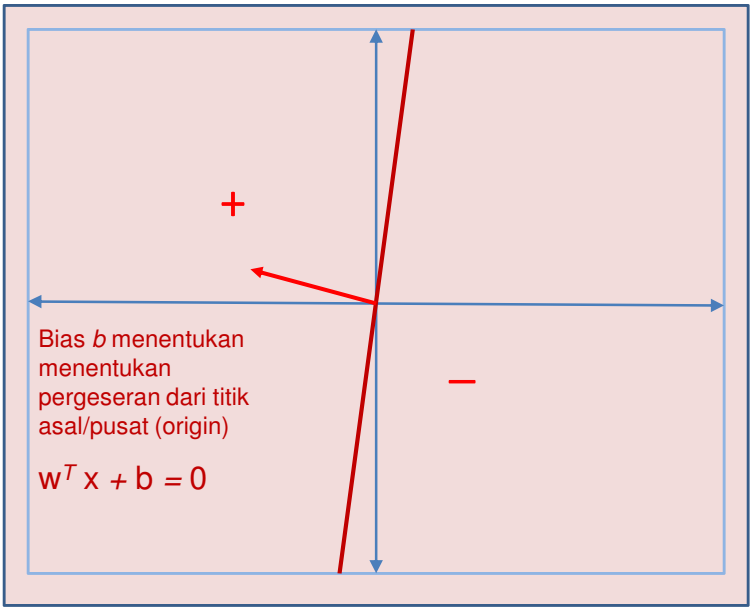

+
−
Vektor normal *w* menentukan orientasi dari hyperplane
$w^T x + b = 0$

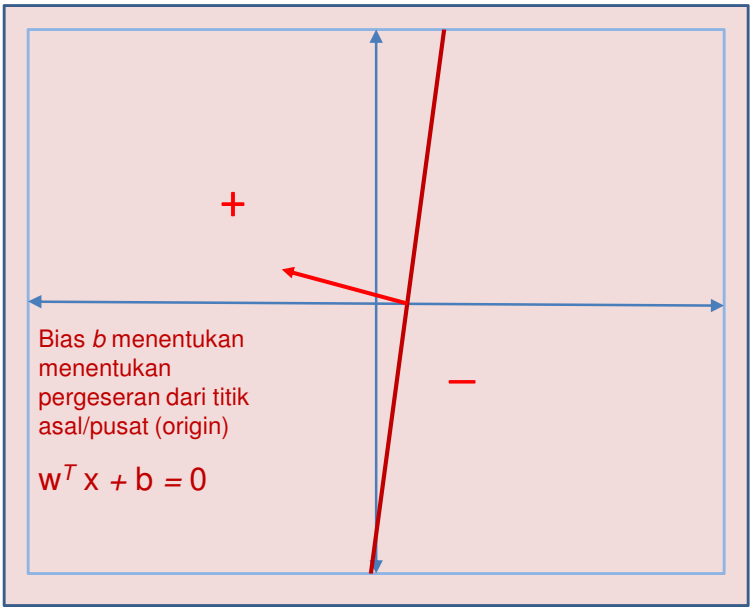

+
Vektor normal w
menentukan orientasi
dari hyperplane
$w^T x + b = 0$
−

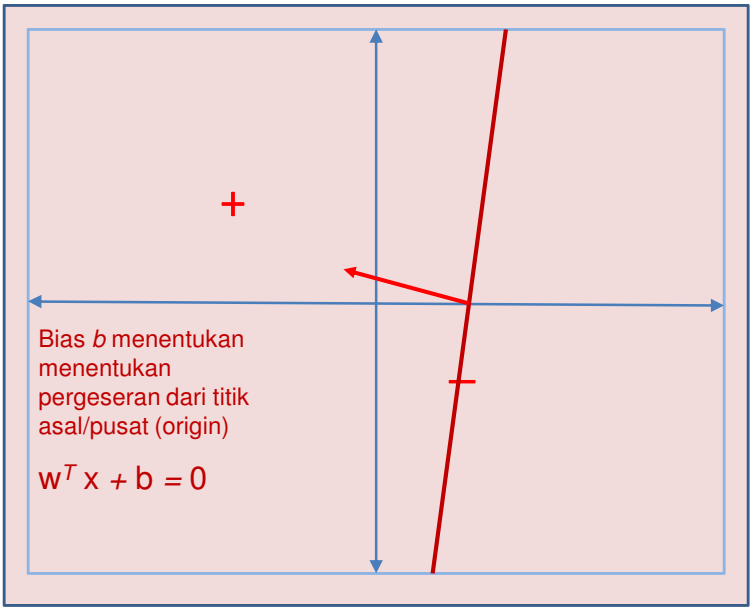

Vektor normal w
menentukan orientasi
dari hyperplane
$w^T x + b = 0$
+
−

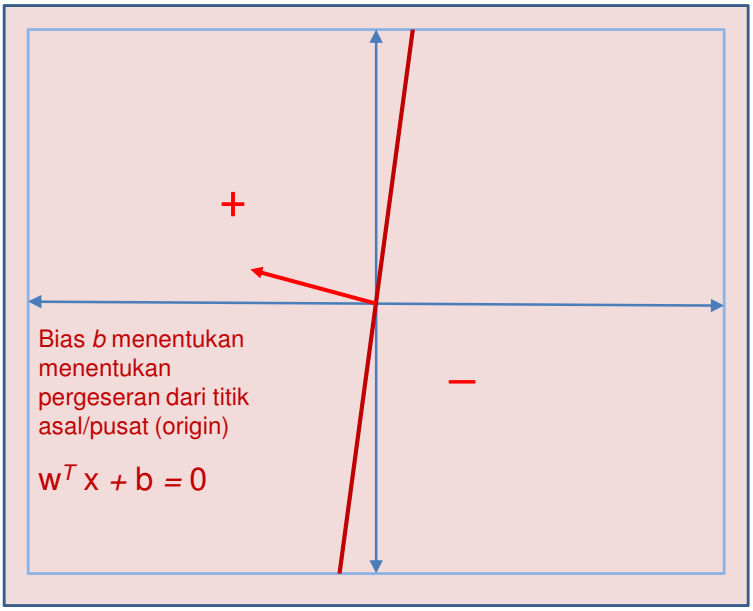

+
Vektor normal w
menentukan orientasi
dari hyperplane
$w^T x + b = 0$
–

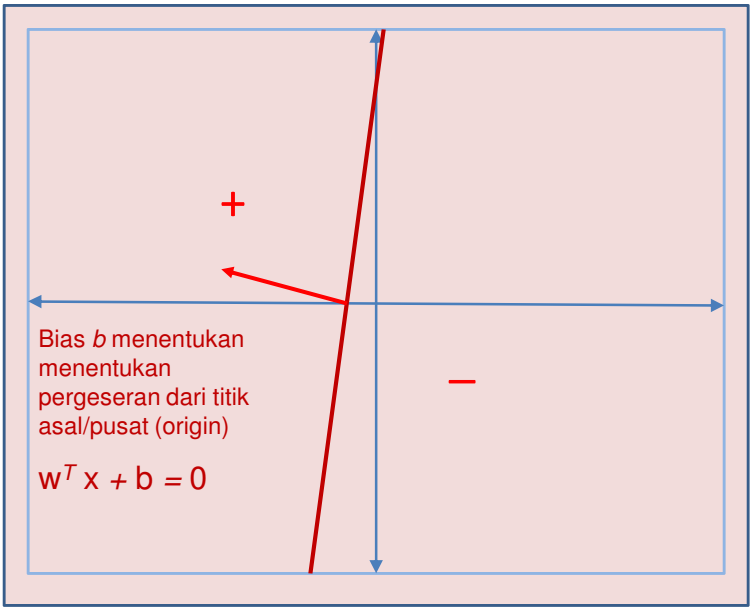

+
Vektor normal w
menentukan orientasi
dari hyperplane
$w^T x + b = 0$
–

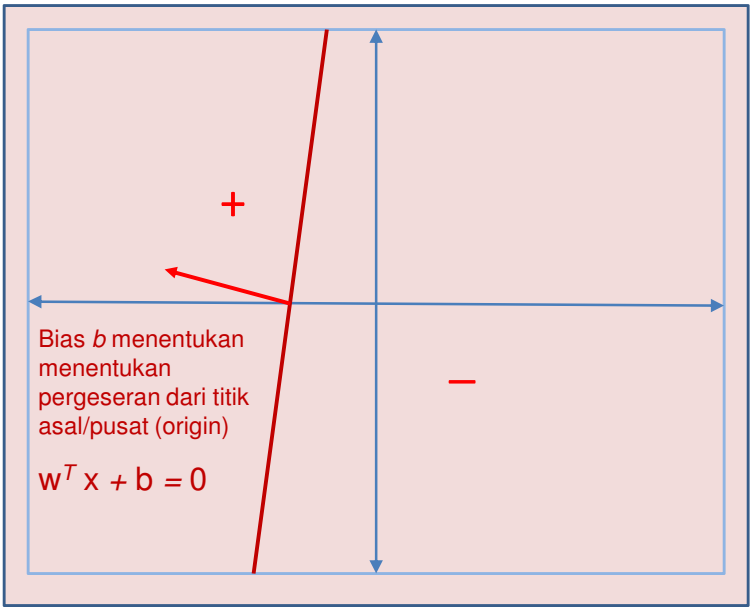

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

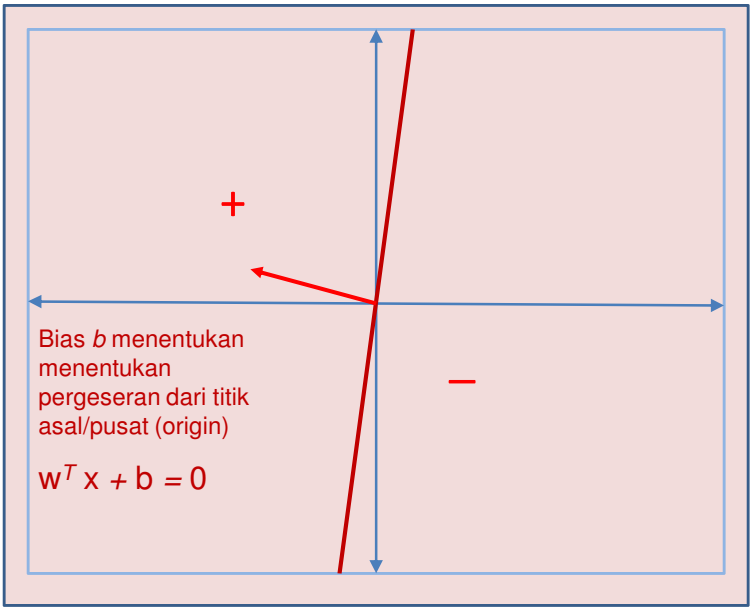

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

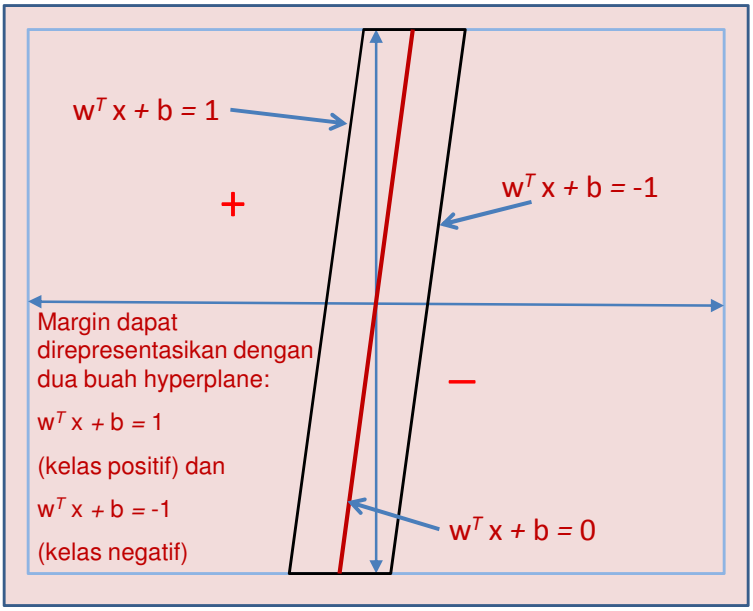

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

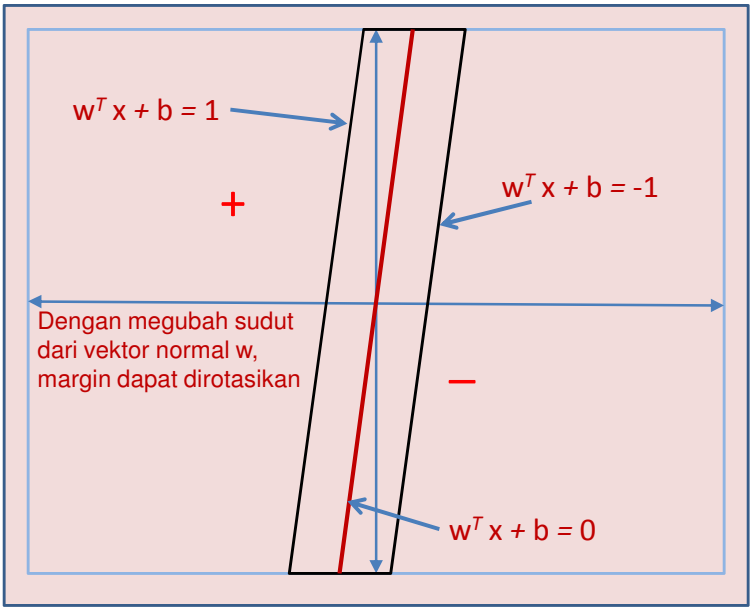

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

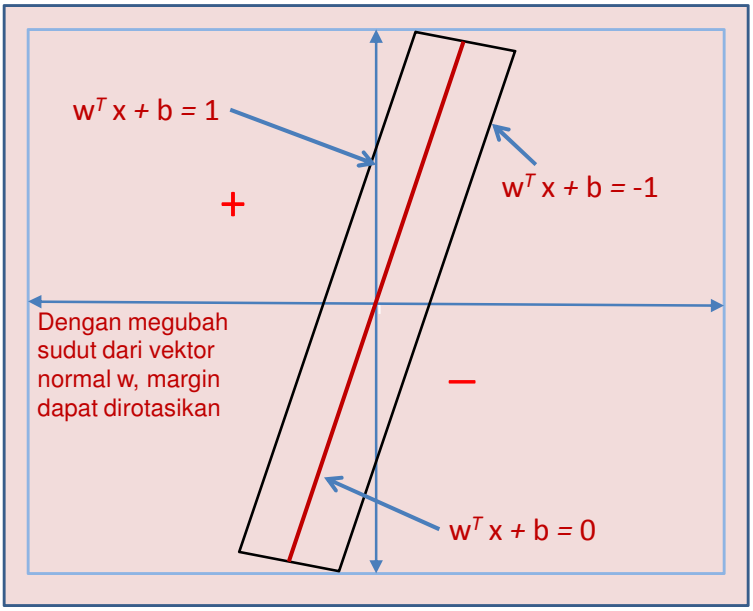

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

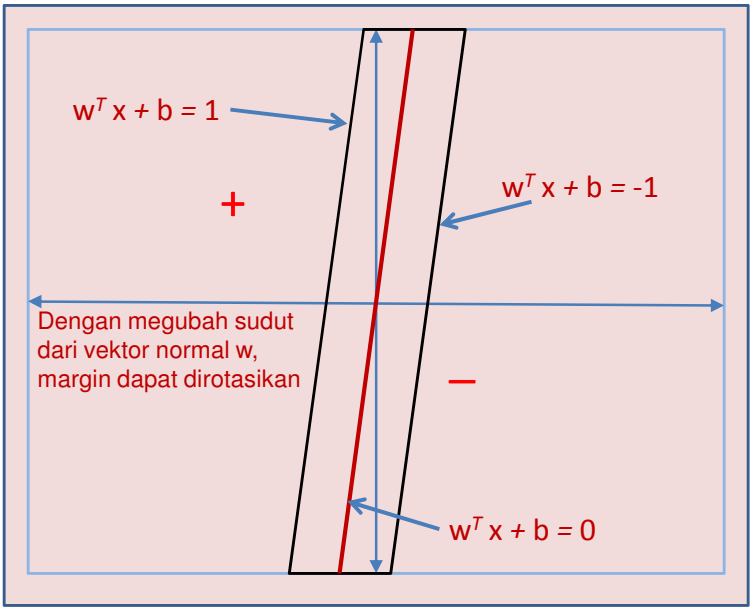

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

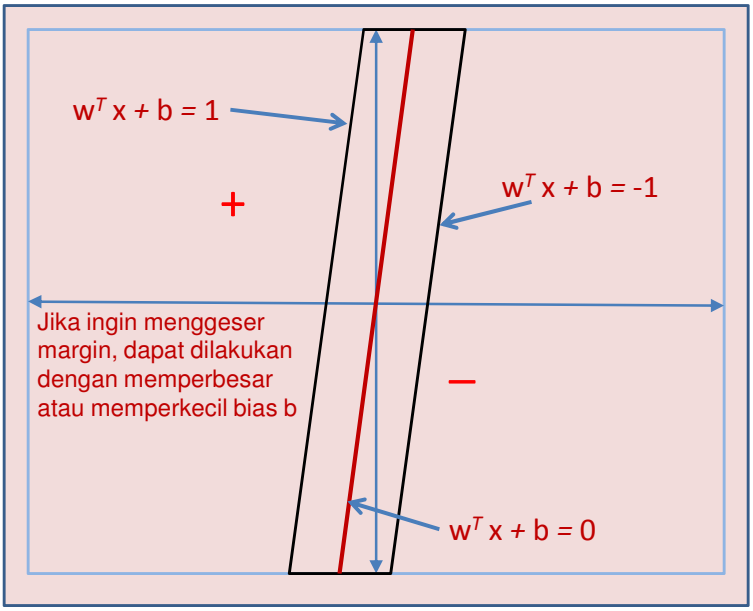

+
−
Bias *b* menentukan menentukan pergeseran dari titik asal/pusat (origin)
$w^T x + b = 0$

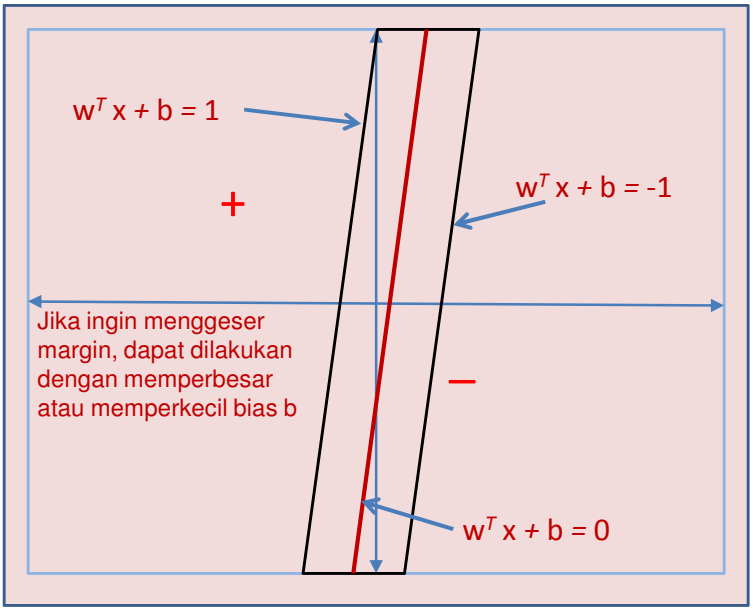

$w^T x + b = 1$
$w^T x + b = -1$
+
−
Margin dapat direpresentasikan dengan dua buah hyperplane:
$w^T x + b = 1$
(kelas positif) dan
$w^T x + b = -1$
(kelas negatif)
$w^T x + b = 0$

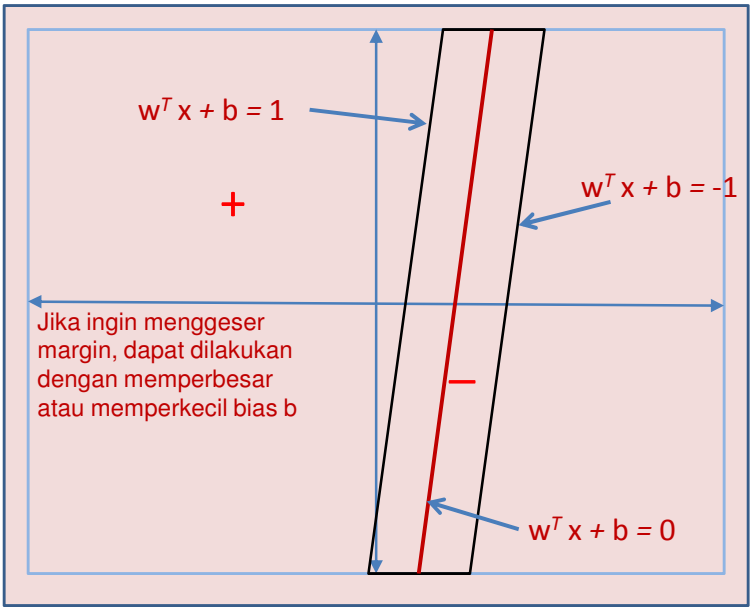

$w^T x + b = 1$
+
$w^T x + b = -1$
Dengan megubah sudut dari vektor normal w, margin dapat dirotasikan
–
$w^T x + b = 0$

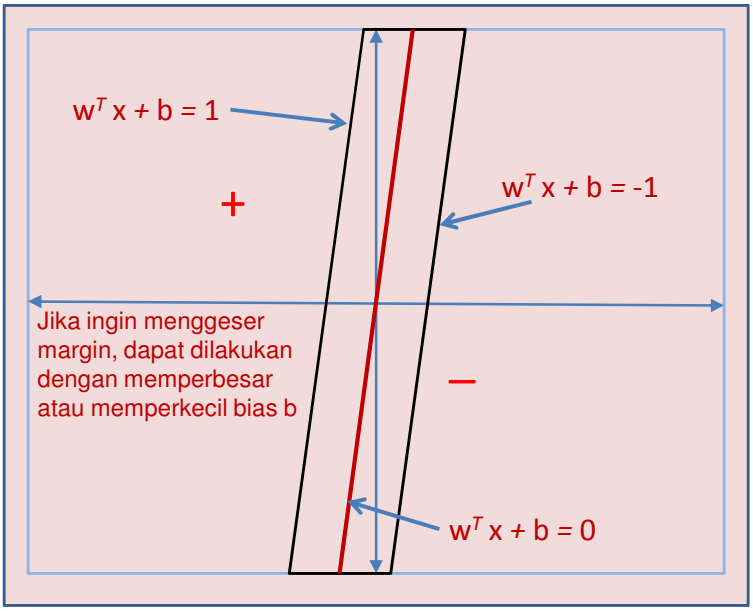

$w^T x + b = 1$
+
$w^T x + b = -1$
Dengan megubah sudut dari vektor normal w, margin dapat dirotasikan
–
$w^T x + b = 0$

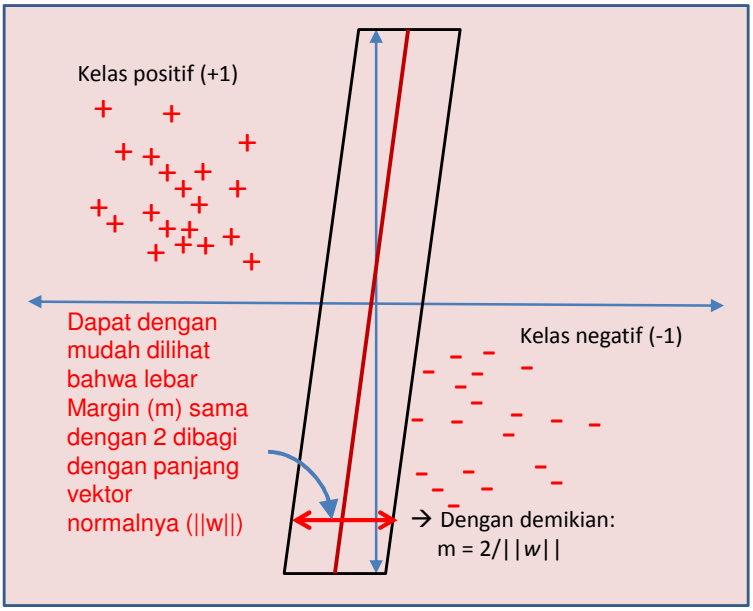

$w^T x + b = 1$
+
$w^T x + b = -1$
Dengan megubah sudut
dari vektor normal w,
margin dapat dirotasikan
–
$w^T x + b = 0$

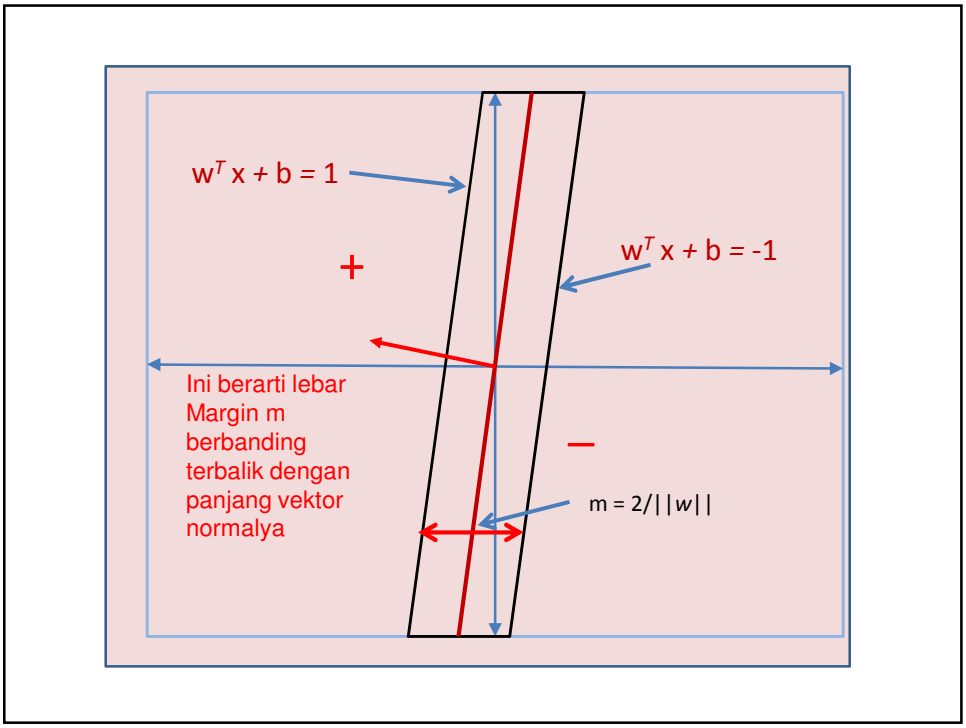

$w^T x + b = 1$
+
$w^T x + b = -1$
Jika ingin menggeser
margin, dapat dilakukan
dengan memperbesar
atau memperkecil bias b
–
$w^T x + b = 0$

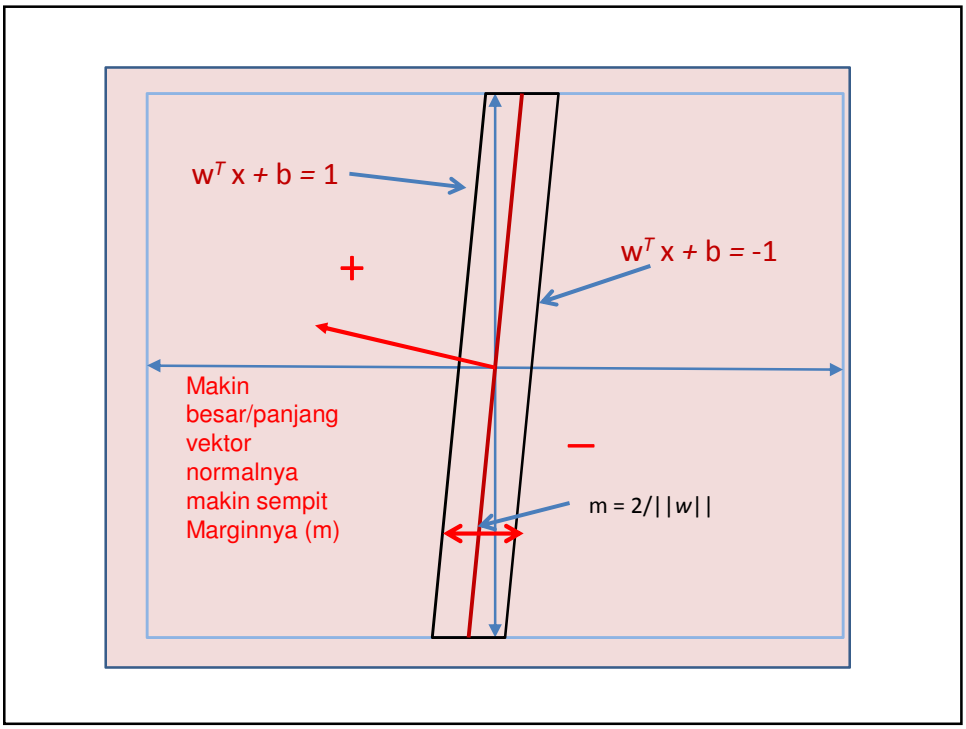

$w^T x + b = 1$
+
$w^T x + b = -1$
Jika ingin menggeser
margin, dapat dilakukan
dengan memperbesar
atau memperkecil bias b
–
$w^T x + b = 0$

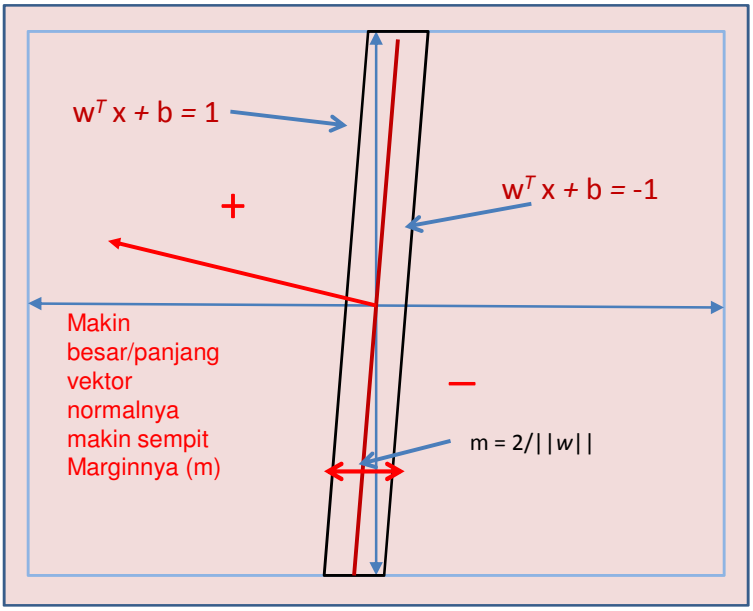

$w^T x + b = 1$
+
$w^T x + b = -1$
Jika ingin menggeser
margin, dapat dilakukan
dengan memperbesar
atau memperkecil bias b
–
$w^T x + b = 0$

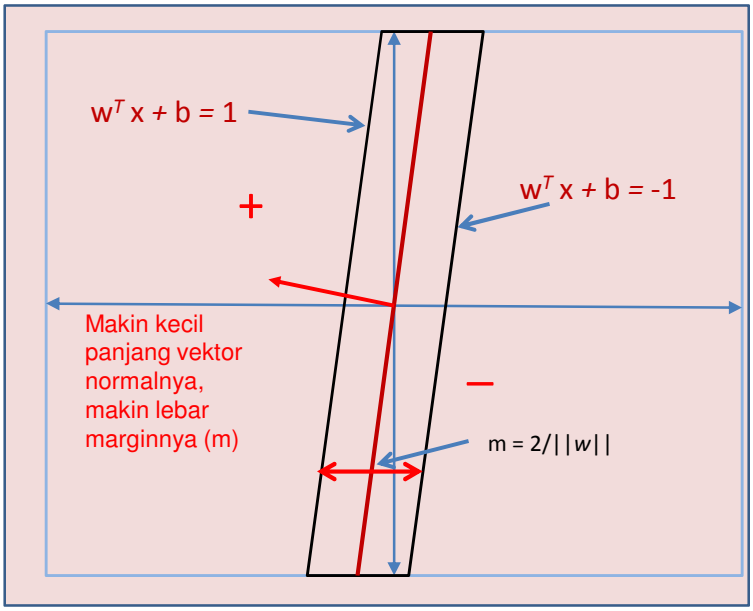

$w^T x + b = 1$
+
$w^T x + b = -1$
Jika ingin menggeser margin, dapat dilakukan dengan memperbesar atau memperkecil bias b
–
$w^T x + b = 0$

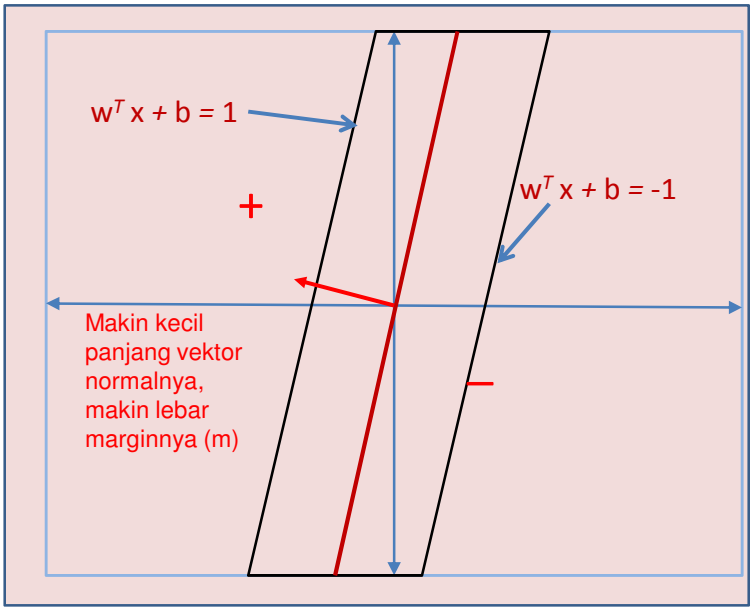

Kelas positif (+1)
Dapat dengan mudah dilihat bahwa lebar Margin (m) sama dengan 2 dibagi dengan panjang vektor normalnya (||w||)
Kelas negatif (-1)
→ Dengan demikian:
m = 2/||w||


$w^T x + b = 1$
+
$w^T x + b = -1$
Ini berarti lebar
Margin m
berbanding
terbalik dengan
panjang vektor
normalya
–
m = 2/||w||

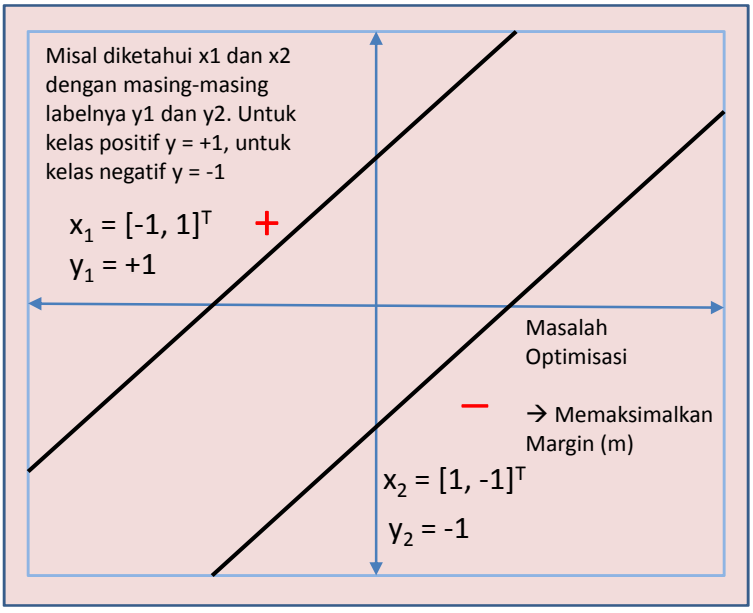

$w^T x + b = 1$
+
$w^T x + b = -1$
Makin
besar/panjang
vektor
normalnya
makin sempit
Marginnya (m)
–
m = 2/||w||


$w^T x + b = 1$
$w^T x + b = -1$
+
−
Makin
besar/panjang
vektor
normalnya
makin sempit
Marginnya (m)
m = 2/||w||


$w^T x + b = 1$
$w^T x + b = -1$
+
−
Makin kecil
panjang vektor
normalnya,
makin lebar
marginnya (m)
m = 2/||w||



Contoh Kasus



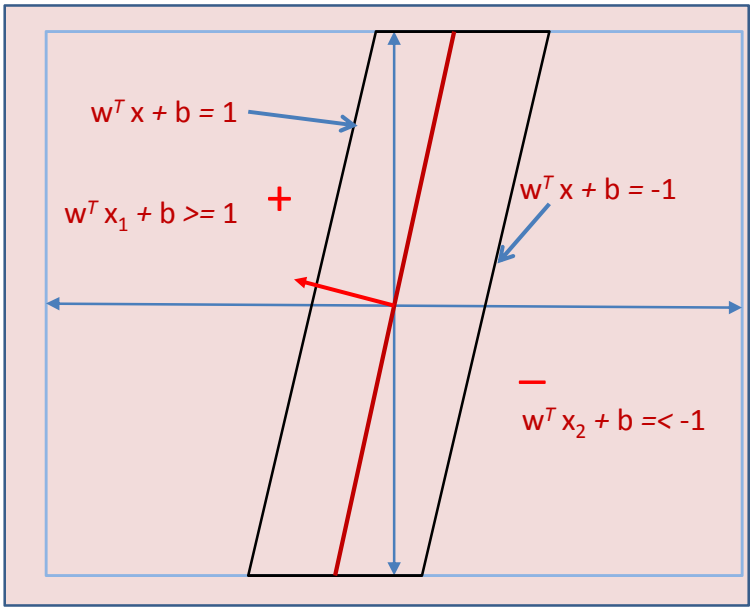
Memecahkan masalah optimisasi
Memaksimalkan Lebar Margin (m)

m = 2/||w||

Memaksimalkan lebar Margin (m)
max (2/||w||)

artinya panjang normal vektor ||w|| harus diminimalkan, sehingga:
max (2/||w||) → min (||w||/2)

Untuk menghilangkan akar kuadrat, vektor normal ||w|| harus dikuadratkan, sehingga:
max (2/||w||) → min (||w||/2)→ min (1/2 $||w||^2$)



Memecahkan masalah optimisasi

Memaksimalkan Lebar Margin (m)

max (2/||w||) → min (||w||/2)→ min (1/2 ||w||^2)
Konstrain:
Dari gambar sebelumnya:

- Untuk kelas positif : $w^T x_1 + b >= 1$
- Untuk kelas negatif: $w^T x_2 + b =< -1$

Selanjutnya kalikan tiap konstrain di atas dengan label yang sesuai, sehingga menjadi: $y_1(w^T x_1 + b) >= 1$ dan $y_2 (w^T x_2 + b) =< -1$

Sehingga diperoleh : min (1/2 ||w||^2) dengan konstrain:

$$y_i(w^T x_i + b) -1 >= 0$$

Ini adalah masalah Optimisasi.

Memecahkan masalah optimisasi

- Untuk memecahkan masalah optimisasi ini kalikan pertidaksamaan $y_i(w^T x_i + b) -1 >= 0$ dengan Lagrange Multipliers untuk membentuk the Primal-Dual Form:

$$\text{minimize} \quad L_p(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (x_i \cdot w + b) - 1)$$
$$s.t \quad \alpha_i \geq 0$$

Memecahkan masalah optimisasi

Sehingga the primal optimization problem menjadi :

$$\text{minimize} \quad L_p(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (x_i \cdot w + b) - 1)$$
$$s.t \quad \alpha_i \geq 0$$

Selanjutnya:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Memecahkan masalah optimasi

Mendapatkan Lagrangian Dual Problem.

Dengan mensubstitusi hasil di atas dengan primal problem dan melakukan beberapa manipulasi matematika diperoleh:

Lagrangian Dual Problem:

$$\text{maximaize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=0}^{n} \sum_{j=0}^{n} \alpha_i \alpha_j y_i y_j x_i^t x_j$$
$$s.t \quad \alpha_i \geq 0 \quad and \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ adalah sekarang variabel untuk tiap sample point x_i .

Lagrangian dual problem

- Dengan memasukkan ke persamaan Langrangian dual problem nilai $y_1 = 1$, $y_2 = -1$

dan $x_1 = [-1, 1]^T$, $x_2 = [1, -1]^T$

Diperoleh persamaan polinomial sederhana:

- $L_D = -\alpha_1^2 - \alpha_2^2 - 2\alpha_1\alpha_2 + \alpha_1 + \alpha_2$
- Untuk mencari Lagrange multiplier yang optimal:
- $\frac{\partial L}{\partial \alpha} = 0 \rightarrow \alpha_1 + \alpha_2 = ½$

  $\sum_{i=1}^{n} \alpha_i y_i = 0 \rightarrow \alpha_1 = \alpha_2$

Sehingga $\alpha_1 + \alpha_2 = ½ \rightarrow \alpha_1 = \alpha_2 = 1/4$

- Dengan demikian diperoleh

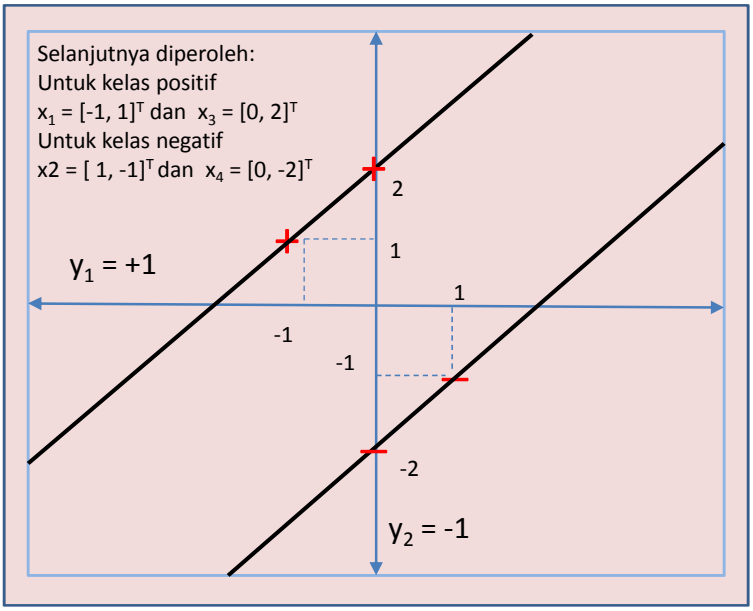
$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = [-1/2, 1/2]^T$$

dari $\alpha_i[y_i(w^T x_i + b) - 1] = 0$ diperoleh:

$b = 1/y_i - w^T x_i$

untuk setiap i dengan $\alpha_i \neq 0$

$\rightarrow$ b = 0 untuk setiap i

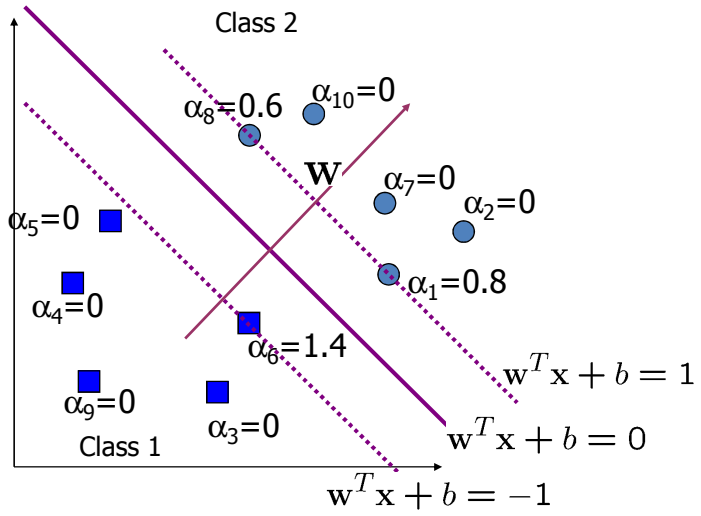


Karakteristik solusi

- Banyak α_i yang bernilai nol
  - **w** adalah linear combination dari sejumlah kecil data
  - Representasi Sparse
- $\mathbf{x}_i$ non-zero α_i disebut support vectors (SV)
  - decision boundary ditentukan oleh SV
  - Jika t_j (j=1, ..., s) adalah indeks dari support , maka dapat ditulis $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- Untuk testing dengan data baru **z**
  - Hitung $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$

    dan klasifikasikan **z** sebagai kelas 1 jika hasil penjumlahannya positive, dan kelas 2 jika sebaliknya

Intrepretasi Geometris

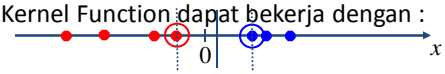


Bagaimana jika titik-titik sampel tidak dapat dipisahkan secara linier ?!

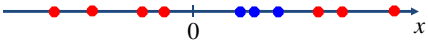
62

Non-linear SVMs

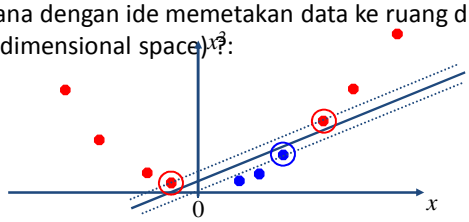
- Dengan dataset yang linearly separable (dengan beberapa noise), SVM tanpa Kernel Function dapat bekerja dengan :



- Tetapi dengan dataset yang tidak linear separable menjadi sulit !



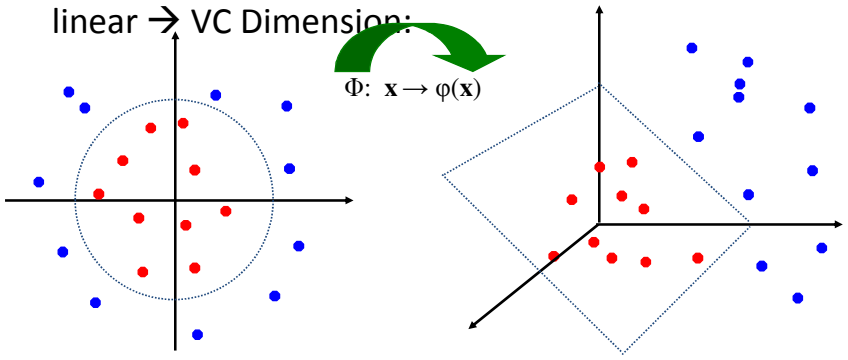
- Bagaimana dengan ide memetakan data ke ruang dimensi yang lebih tinggi (higher-dimensional space) ?:



63

Non-linear SVMs: Feature spaces

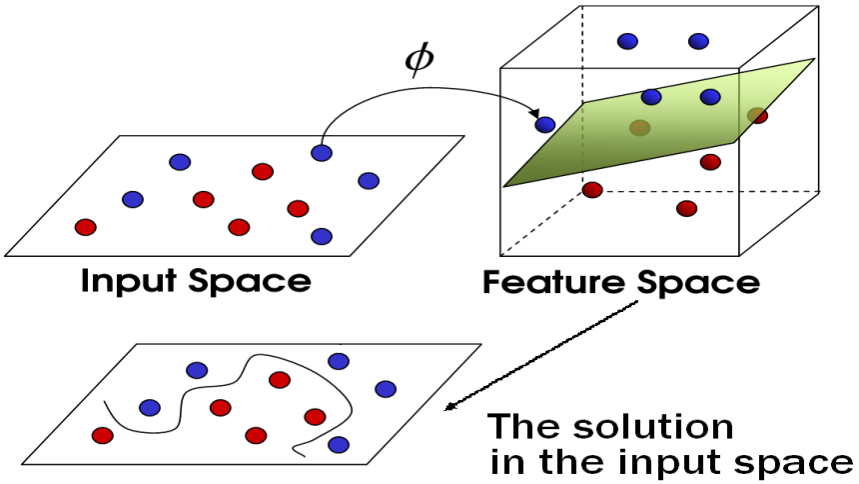
- Ide umumnya: original feature space dapat selalu dipetakan ke beberapa higher-dimensional feature space di mana training set dapat dipisahkan secara linear → VC Dimension:



64

Non Linear SVM :

Pemetaan data ke dalam dimensi yang lebih tinggi

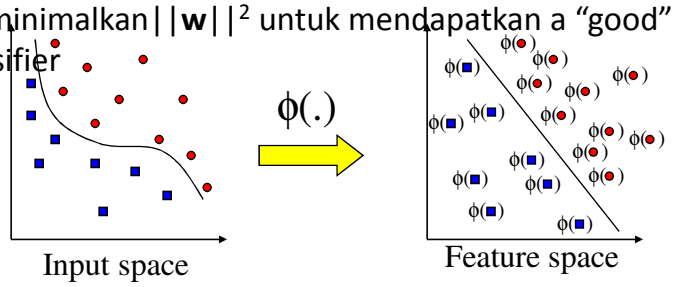


Perluasan ke Non-linear Decision Boundary

- Ide kuncinya: transformasikan $\mathbf{x}_i$ ke sebuah ruang dimensi yang lebih tinggi (higher dimensional space) → "**Dimensi spriritual , to make life easier ☺**"
  - Input space: ruang di mana $\mathbf{x}_i$ berada
  - Feature space: ruang dari $\phi(\mathbf{x}_i)$ hasil transformasi
- Kenapa perlu transformasi?
  - Linear operation pada feature space ekuivalen dengan non-linear operation pada input space
  - Klasifikasi dapat dilakukan dengan lebih mudah dengan suatu transformasi yang sesuai contoh: XOR

Extension to Non-linear Decision Boundary

- Masalah yang muncul dengan transformasi
  - Komputasi tinggi dan sulit untuk mendapatkan estimasi yang baik
- SVM memecahkan dua masalah ini secara simultan
  - Kernel tricks untuk komputasi yang efisien
  - Meminimalkan||**w**||^2 untuk mendapatkan a "good" classifier



The Kernel Trick:

Jika kita melihat lagi problem optimalisasi berikut

$$\text{maximize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i^t)\phi(x_j) \quad s.t\ \alpha_i \geq 0 \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

Dan fungsi keputusan :

$$f(\phi(x)) = sign(w^t \phi(x) + b) = sign(\sum_{i=1}^{n} \alpha_i y_i \phi(x_i^t)\phi(x) + b)$$

Tidak perlu memetakan secara eksplisit atau mengetahui dimensi dari space baru, karena hanya perlu menggunakan dot product dari vektor cirinya baik pada training maupun test.

The Kernel Trick:

Jadi hanya perlu menghitung $K(x_i, x_j)$ dan tidak perlu melakukan perhitungan dalam ruang berdimensi tinggi secara eksplisit → ini yang disebut Kernel Trick.

Contoh Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  – Closely related to radial basis function neural networks

- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  – It does not satisfy the Mercer condition on all κ and θ

- Research on different kernel functions in different applications is very active

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals

71

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

72

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All
- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

73

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
- Perfect score is 1.0
- Inverse relationship between precision & recall

$$recall = \frac{TP}{TP + FN}$$

- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F_β*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

74

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

75

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

76

Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

77

Estimating Confidence Intervals: Classifier Models M_1 vs. M_2

- Suppose we have 2 classifiers, M_1 and M_2, which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the 2 error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

78

Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom** (here, *k=10*)
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: M_1 & M_2 are the same
- If we can **reject** null hypothesis, then
  - we conclude that the difference between M_1 & M_2 is **statistically significant**
  - Chose model with lower error rate

79

Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

80

Terima kasih

"Jika anda memiliki problem kehidupan yang rumit (non linear), petakan ke dimensi yang lebih tinggi (spiritual), kembalikan kepada Allah SWT yang Memegang hidup anda. Insyaa Allah dengan demikian, permasalahan akan lebih mudah dipecahkan…." ☺

"Jadikanlah sabar dan sholat sebagai penolongmu" (QS 2:45)

81