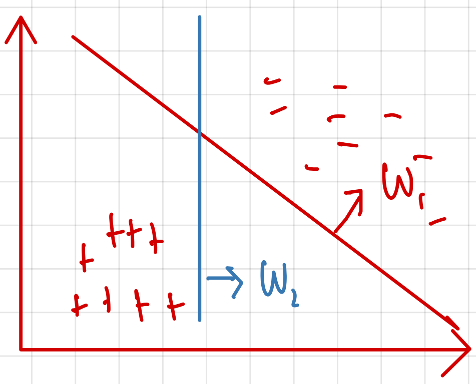
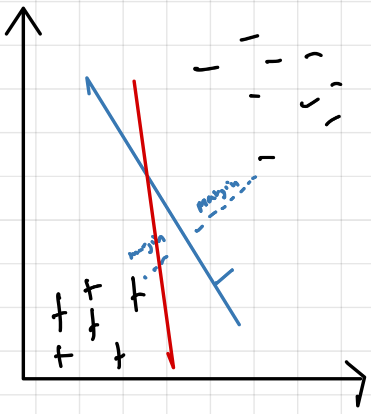
# Support vector machine (SVM)

- Extendtion ของ perceptron
  - ↳ perceptron : find a hyperphane if it exists
    (how many hyperplane ? )

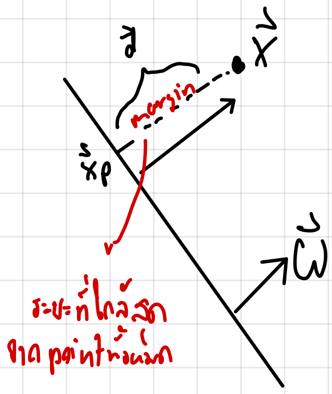



: many hyperplane but which one is the best?
↳ SVM

- SVM : find the maximum margin separating hyperplane : marginสูงสุด
  ↳ ระยะทางที่สั้นที่สุดจาก hyperplane ไปยัง point



- The margin ($\gamma$): is the closest distance from the hyperplane to the closest points in either classes




$\vec x_p = \vec x - \vec d$ ; $\vec x = \vec x_p + d$

$\vec w^T \vec x_p + b = 0$ [$\vec x_p$ lies on the hyperplane] [$x_p$ อยู่กับ hyperplane]

$\vec w^T(\vec x - \vec d) + b = 0$ ; $d = \alpha \vec w$ ; for some $\alpha$ ($\alpha$ เป็นตัวหด/ขยาย $\omega$ เพื่อให้มีความยาว)

$\vec w^T(\vec x - \alpha \vec w) + b = 0$

$$\alpha = \frac{w^T \vec x + b}{w^T w}$$

find distance

$$\Rightarrow \vec d = \left(\frac{\vec w^T \vec x + b}{\vec w^T \vec w}\right)\vec w$$

$$\Rightarrow \|\vec d\|_2 = \sqrt{\vec d^T \vec d} = \sqrt{(\alpha \vec w)^T(\alpha \vec w)}$$

$$= \sqrt{\alpha^2 w^T w}$$

$$= \alpha \sqrt{\vec w^T \vec w}$$

$$= \frac{\vec w^T \vec x + b}{\vec w^T \vec w}\sqrt{\vec w^T \vec w}$$

$$= \frac{\vec w^T x + b}{\sqrt{\vec w^T \vec w}}$$

find margin $\gamma = \min\limits_{\vec{x}} \frac{\vec{w}^T \vec{x} + b}{\sqrt{\vec{w}^T \vec{w}}} = \gamma(w,b) = \min\limits_{\vec{x}_i} \frac{w^T x_i + b}{\sqrt{w^T w}}$ ↳ $\|w\|_2$

## Maximum margin hyperplane:

$$\boxed{\vec{w}, b = \max_{\vec{w}, b} \underbrace{\left( \min_{\vec{x}} \frac{\vec{w}^T \vec{x} + b}{\sqrt{w^T w}} \right)}_{\text{margin term}}}$$

} ถ้าไม่ separate {+,-}

## - Maximum margin separating hyperplane:

$$\underline{\vec{w}, b = \max_{\vec{w}, b} \left( \min_{\forall \vec{x}_i} \frac{\vec{w}^T \vec{x}_i + b}{\sqrt{\vec{w}^T \vec{w}}} \right) \text{ st.}}$$

↖ objective function

$$\underline{\forall i,\ y_i (\vec{w}^T x_i + b) \geq 0}$$

↑ constraints

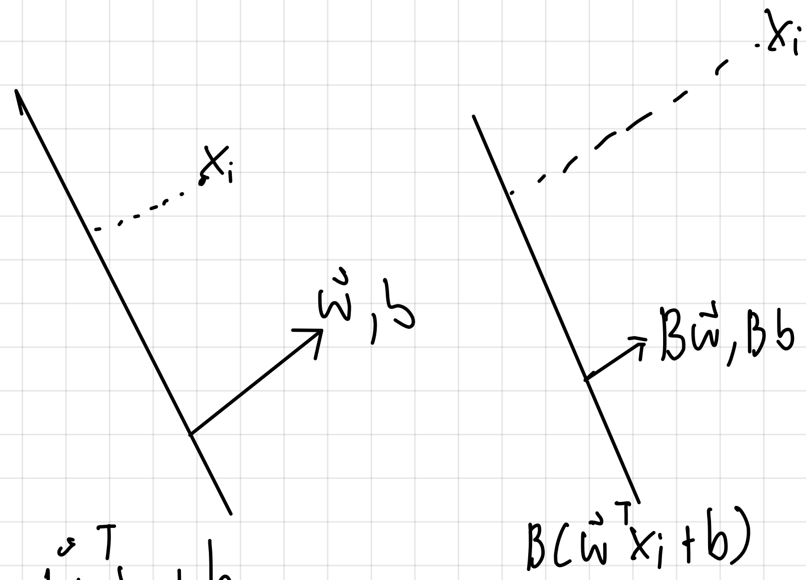
- Simplification of finding such $\vec{w}, b$: [Classifly all point] (แบ่ง 1,-1 ได้)

  - $\vec{w}, b = \max\limits_{w,b} \left( \frac{1}{\sqrt{\vec{w}^T \vec{w}}} \underbrace{\min_{\forall \vec{x}_i} \vec{w}^T \vec{x}_i + b}_{\text{margin}} \right)$ st. $\forall i,\ y_i (\vec{w}^T \vec{x}_i + b) \geq 0$

=> Because the hyperplane is scale invariant,

we can enforce

$$\min_{\forall x_i} |\vec{w}^T \vec{x}_i + b| = 1 \quad \text{(another constraint)}$$




$$\underbrace{\vec{w}^T x_i + b}_{\text{prediction}}$$

ระยะทางจาก $x_i$ ไปยัง $w$

prediction ก็อยู่ห่างจาก hyperplane

ถ้ามากค่า prediction จะมาก

=> $\boxed{\vec{w}, b = \max\limits_{w,b} \frac{1}{\sqrt{w^T w}}}$ st. $\min\limits_{\forall x_i} |\vec{w}^T \vec{x}_i + b| = 1$

$\forall i,\ y_i (\vec{w}^T \vec{x}_i + b) \geq 0$

} $\forall i,\ y_i (\vec{w}^T \vec{x}_i + b) \geq 1$

$$\boxed{\vec{w}^*, b^* = \min_{\vec{w}, b} \overbrace{\sqrt{\vec{w}^T \vec{w}}}^{\|w\|_2} = \min_{\vec{w}, b} \vec{w}^T \vec{w}}$$

↳ we can find them by using QCQP

$\|w\|_2 = \sqrt{w^T w}$

$\|w\|_2^2 = w^T w$

$w^T w \propto \sqrt{w^T w}$

Final Formulation:

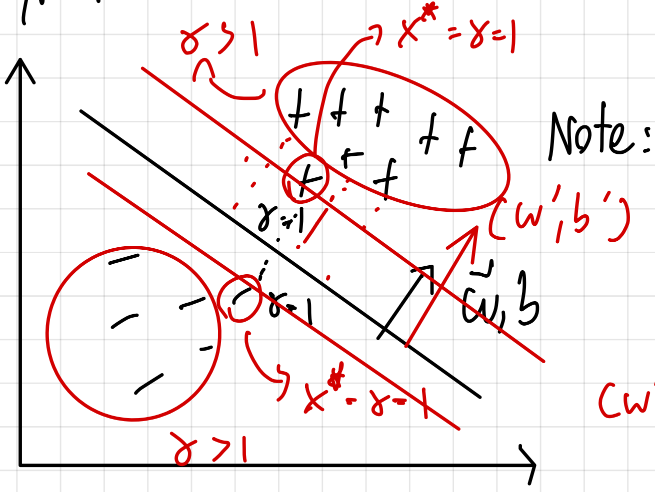
quadatic function

Goal of SVM is to find $\vec{w}, b$ according to the formulation

$$\boxed{\vec{w}, b = \min_{\vec{w}, b} \underbrace{w^T w}_{\text{quadatic function}} \text{ st. } \forall i\ \underbrace{y_i (\vec{w}^T \vec{x}_i + b) \geq 1}_{\text{linear inequalities}}}$$

To find $\vec{w}, b$ we can use Quadratic Programming Solver / QCQP)

Interpretation: find $\vec{w}, b$ where $\vec{w}$ is of minimum magnitude such that all points lie at least 1 unit away from the hyperplane on the correct side. [w is the simplest solution]




Note: There always exist $x_i$ st.

$|\vec{w}^T x_i + b| = 1$ (margin)

($x_i$ is the closest point)

$(w', b') = (Bw, Bb)$

The vector $\vec{w}$ (and $b$) supports the closest point.

Note $y_i(\underbrace{\vec{w}^T \vec{x}_i + b}) \geq 1$

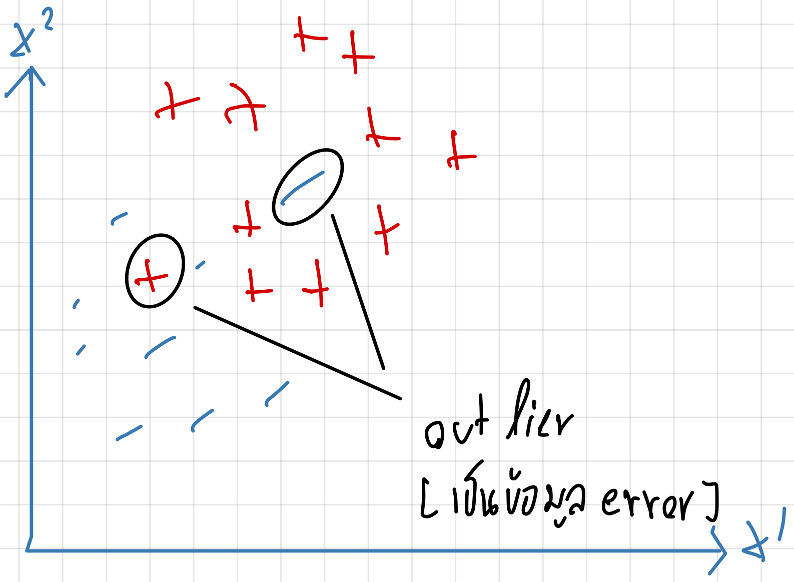
ต้องมีค่ามากกว่า 1

$\gamma = 1$ เสมอ [ค่าใกล้เส้น]

$$\gamma(w, b) = w^T x^* + b$$
$$\gamma'(w', b') = w'^T x^* + b'$$
$$= Bw'^T x^* + Bb$$
$$= B(w^T x^* + b)$$

Support Vector: The vector w and b supports the closest point $x^*$

## Problem Percepton [low Dimensional Data set]




$\therefore$ can't use percepton [infinte loop]

## Dealing with non-linearly separable data:

- IDEA: We may sacrifice some outlier(s) in order to place the hyperplane

# Softing the constraints [relax constraint]

$\forall i,\ y_i(w^T x_i + b) \geq 0.7$ [น่าจะ atleast 0.7]

$< 0$ [can classifly wrong side]

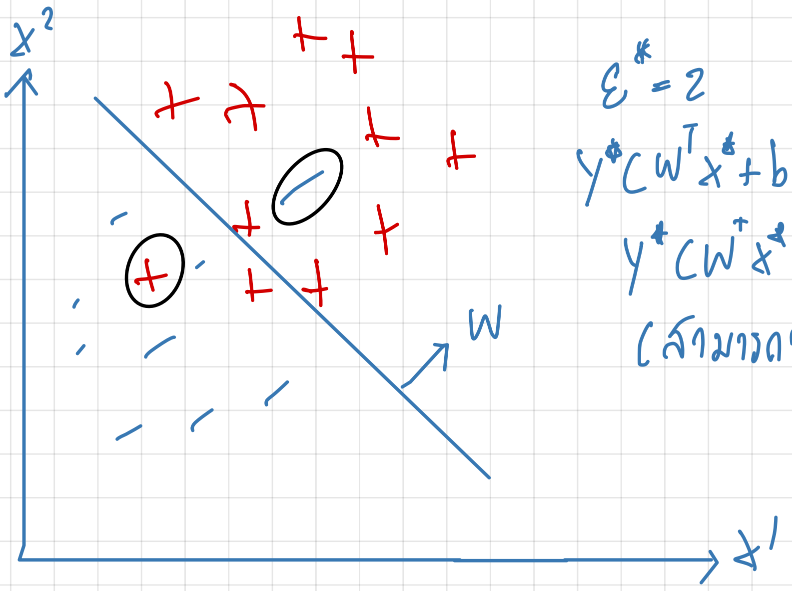
- create new variable for constraints.
- Fix outlier
  - we allow the constraints to be seften slightly with the introdection of slack (หย่อน) variable

$$\varepsilon_i \geq 0,\ \forall i$$

(\psi)

$$\forall i,\ y_i(w^T x_i + b) \geq 1 - \varepsilon_i$$ : corespanse ของแต่ละ $y_i$ ; $\varepsilon_i \geq 0$

($\varepsilon_i$ ไม่จำเป็นต้องเหมือนกัน)




$\varepsilon^* = 2$

$y^*(w^T x^* + b) < 0$

$y^*(w^T x^* + b) \geq -1$

(สามารถวางไว้ผิดฝั่งได้)

hyper parameter [หน้าที่เรา set]

จากเดิม -> New

$$w^*, b^*, \varepsilon^* = \min_{w,b,\varepsilon} \left[ w^T w + C \sum_{i=1}^{n} \varepsilon_i \right] \ ; \ C \geq 0$$

$\hookrightarrow \varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$

if set $C = \infty$ [no relax] $\forall i,\ y_i(w^T x_i + b) \geq 1 - \varepsilon_i$

if set $C = 0$ [free]

- The slack variable $\varepsilon_i$ allow $x_i$ to be closer the hyperplane or even be on the wrong side but there is a penalty (exchange cost) in the objective function for such slack.

<u>Penalty</u> $C \to +\infty$, svm will try to make all the points to be on carrect side.

$C \to 0$, svm will sacrifice some points

# Unconstrainted formulation:

- we set $\varepsilon_i$ as follows:

$$\varepsilon_i = \begin{cases} 1 - y_i(w^T x_i + b) & \text{if } y_i(w^T x_i + b) < 1 \\ 0 & \text{if } y_i(w^T x_i + b) \geq 1 \end{cases}$$

- In other words

$$\varepsilon_i = \max(1 - y_i(w^T x_i + b), 0)$$

- Hence, we can rewrite

$$\boxed{w^*, b^* = \min_{w,b} \Big( \underbrace{w^T w}_{\substack{\text{regularizer} \\ l_2}} + C \underbrace{\sum_{i=1}^{n} \max(\overbrace{1 - y_i(w^T x + b)}^{1 - h_{w,b}(x_i)}, 0)}_{\text{loss function}} \Big)}$$

(hinge loss)

svm with soft constrants

Many ML algorithms can be expressed via the optimatzation problem of the form

$$w^*, b^* = \min_{w} \Big( \underbrace{\frac{1}{n} \sum_{i=1}^{n} \underbrace{l(h_w(x_i), y_i)}_{\substack{\text{loss function} \\ \text{of } h() \text{ with} \\ w \text{ as parameter}}}}_{\text{average loss}} + \underbrace{\lambda r(w)}_{\text{regularizer}} \Big)$$

Similar

put $\lambda$ in regularizer and cancel $C$ cause $\lambda$, $C$ is balance in each other.

Note: regularizer like prior so it used for MAP

if $\lambda = \frac{1}{C}$

$$w^*, b^* = \min_{w,b} \Big( w^T w + C \sum_{i=1}^{n} \max(1 - y_i(w^T x + b), 0) \Big)$$

$$\boxed{= \min \Big( \underbrace{\lambda w^T w}_{l_2\text{-regularizer}} + \sum_{i=1}^{n} \underbrace{\max(1 - y_i(w^T x_i + b), 0)}_{\text{hinge loss}} \Big)}$$

if $C = \frac{1}{\lambda}$ $\quad \hookrightarrow \|w\|_2^2 = w^T w = (w^1)^2 + (w^2)^2 + \dots + (w^d)^2$

$$\min_{w,b} \Big( w^T w + C \sum_{i=1}^{n} \max(1 - y_i(w^T x + b), 0) \Big)$$

$n$

# Empirical risk minimization

- Many learning algorithms can be writen in a form of an Optimization problem with objective to minimize some loss function $l$ and a regularizer $r()$

$$w^* = \min_w \underbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{l(h_w(x_i), y_i)}_{\text{loss}}}_{\text{average loss}} + \underbrace{\lambda r(w)}_{\text{regularizer}}$$

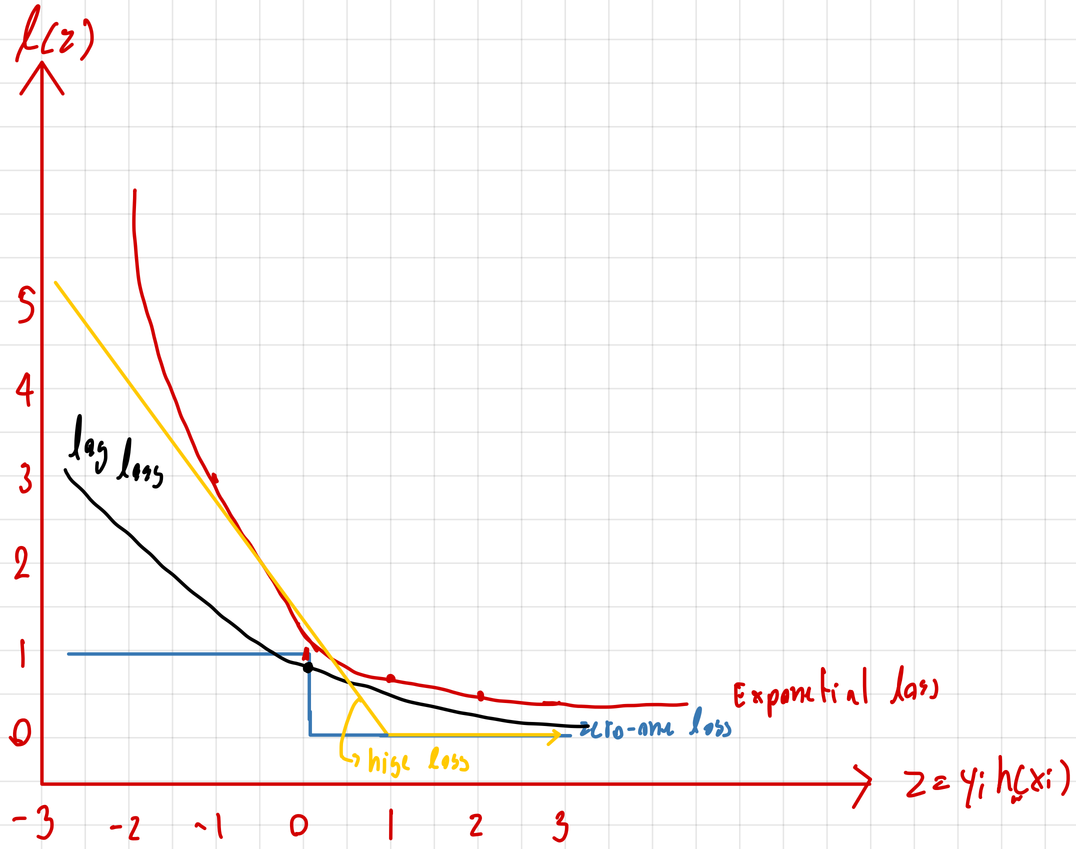
like pior

Example

$$w_{MAP} = \boxed{\arg\min_w \frac{1}{n}\underbrace{\sum_{i=1}^{n}(w^T x_i - y_i)^2}_{\text{sq-loss}} + \lambda \underbrace{w^T w}_{\text{regularizer}}}$$

## Commonly Used Binary Classification loss functions. (not for regression)

| loss $l(h_w(x_i, y_i))$ | Usage | Comments |
|---|---|---|
| Hige-loss $[\max(1-h_w(x_i)y_i, 0)]^P$ | - Standard [P=1]<br>- Hige less SVM (P=2) (differentiable | When used for standard SVM, the loss function denotes the size of the margin between the linear separator and its closest points in either class |
| log-loss $(\log(1+e^{-h_w(x_i)y_i})$ | - logistic regression | - One of the most popular loss functions in ML, since its output are well calibrated probs. |
| Exponential loss $e^{-h_w(x_i)y_i}$ | - Ada Boost | - This loss function is very aggressive it increases exponentialy with vale of $-h_w(x_i)y_i$ it's sensitive to noisy data |
| zero-one loss $\delta(\text{sign}(h_w(x_i) \neq y_i)$ | - Actual classification loss | - non continous cannot optimize in practice |




- Assume $y_i \in \{-1, 1\}$

hige loss
- miss when data in margin or high margin / loss will be linear

zero-one loss
if $z > 0$ loss = 0 : classifly
if $z < 0$ loss = 1 : miss classifly [count 1]

exponetina loss
if $z > 0$ loss approach to 0 : classify
if $z < 0$ loss will jump high loss and try to decrease it : miss classifly

log-loss
if $z < 0$ loss will likely linear : miss classifly

# Regression

| loss $l(h_w(x_i), y_i)$ | Comment |
|---|---|
| Squared loss $(h_w(x_i) - y_i)^2$ | - most popular regression loss function<br>- Estimate mean label<br>- Pros: Differentiable everywhere<br>- Cons: Sensitive to outliers / nosie |
| Absolute loss $\lvert h_w(x_i) - y_i \rvert$ | - Also, very paplar loss function<br>- estimetes median label<br>- Pros: less sensitive to noise<br>- Cons: Not differntiable at 0 |
| Huber loss $\frac{1}{2}(h_w(x_i) - y_i)^2$ if $\lvert h_w(x_i) - y_i \rvert < \delta$<br>- otherwise $\delta(\lvert h(x_i) - y_i \rvert - \frac{\delta}{2})$ | - A.k.A. smooth absolute loss<br>- Pros Best of "Both worlds" (squared + Absolute loss)<br>- Once-differentiable<br>- Takes on behavior of sq-loss when loss is small, and absolute loss when loss is large. |

| | |
|---|---|
| log - Cosh <br> $\log(\cosh(h(x_i)-y_i))$, <br> $\cosh(x) = \frac{e^x+e^{-x}}{2}$ | - Pros Similar to Huber loss, but twice differtiable every where |

## Regularizer

$$w^* = \min_w \underbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell(h_w(x_i), y_i)}_{\text{loss}}}_{\text{average loss}} + \underbrace{\lambda r(w)}_{\text{regularizer}}$$

like pior

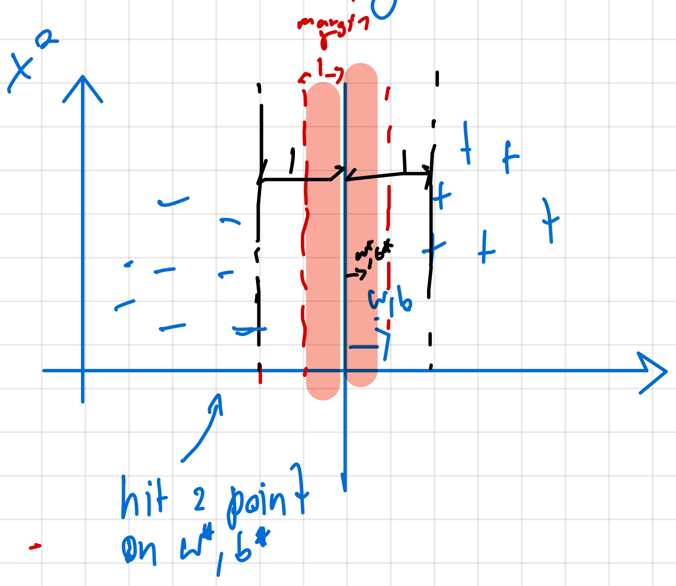
∴ data ที่เรามีมาจากการเก็บของ population
ทำให้ไม่รู้ว่า data นั้นมีการกระจายตัวปกติหรือไม่เลยต้องมี
ตัว regularizer (over fit ใน collect data แต่ไม่ใช่นอก data)
↳ เป็นตัวต้องไม่ให้ overfit

- without regularizer, we always end up with only minimizing loss function on traning data. This often leads to "over fitting".
- In general, regularizer corresponds to the notion of simplicity / complexity of the solution to the optimization problem

SVM: $\min_{w,b} w^T w \rightarrow$ regularizer

subject to $\underbrace{\forall i, y_i(w^T x_i + b) \geq 1}_{\text{loss}}$

# Maximum margin Solution $w^*, b^*$




norm $w^*, b^*$ ลดลงเมื่อ $\lambda$ เพิ่มขึ้น

$$\boxed{\min_{w} \frac{1}{n} \sum_{i=1}^{n} l(h_w(x_i), y_i) + \lambda r(w)}$$

for any value $\lambda \geq 0$

there exists $B \geq 0$

and vice versa

$$\Longleftrightarrow$$

$$\boxed{\begin{aligned} &\min_{w} \frac{1}{n} \sum_{i=1}^{n} l(h_w(x_i), y_i) \\ &\text{subject to } r(w) \leq B \end{aligned}}$$

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} l(h_w(x_i), y_i)$$

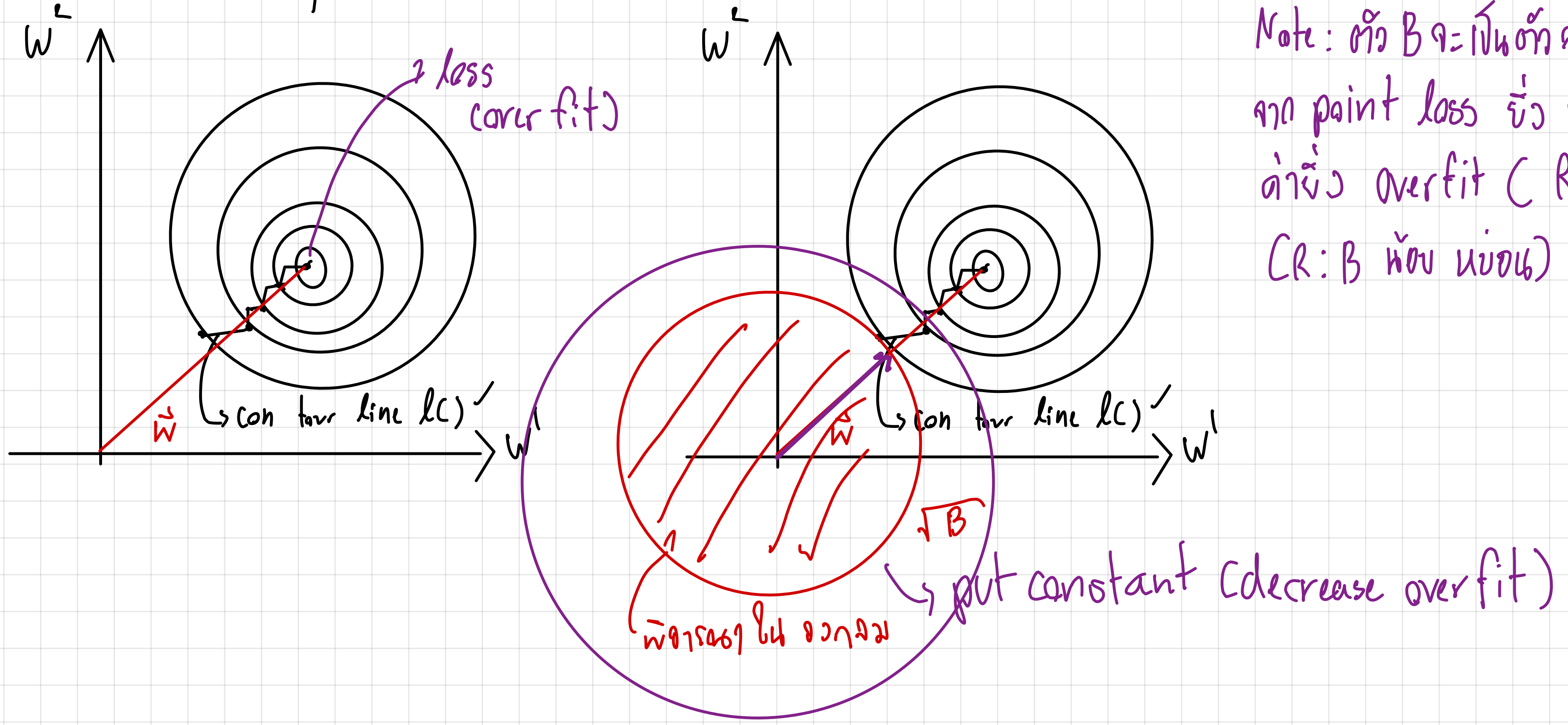
subject to $r(w) \leq B$

$$r(w) = w^T w = \|w\|_2^2 = (w^1)^2 + (w^2)^2 \leq B$$

## $l_2$ regularizer

พิจารณาตัว loss only

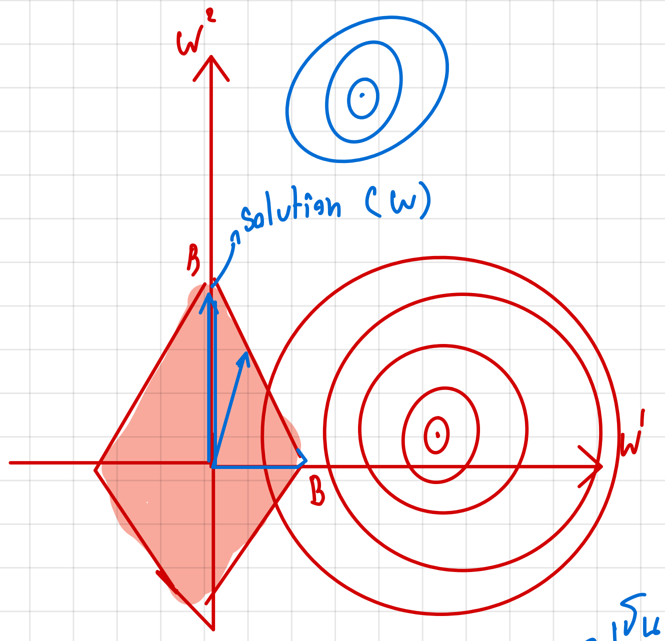
กรณีพิจารณา B ด้วย → $x^2 + y^2 \leq B$ → สมการวงกลม




Note: ตัว B จะเป็นตัวลดความ overfit จาก point loss ยิ่ง radius B มาก ค่ายิ่ง overfit (R: B มาก overfit) (R: B น้อย น้อยลง)

## $l_1$ - regularizer

$r(w) = \|w\|_1 = |w^1| + |w^2| \le B$ สมการเส้นตรง



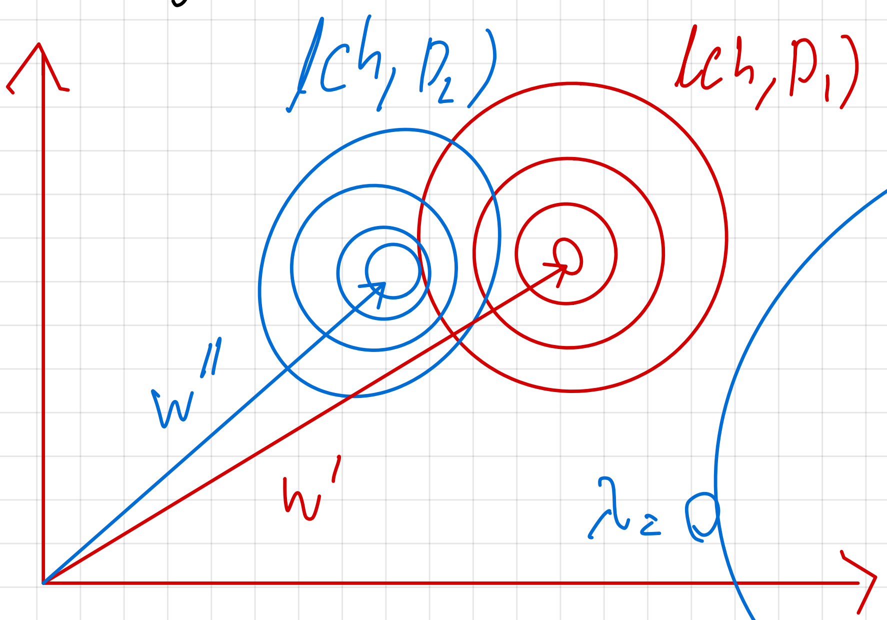

$\begin{bmatrix} w^1 = 0 \\ w^2 = 100 \end{bmatrix}$

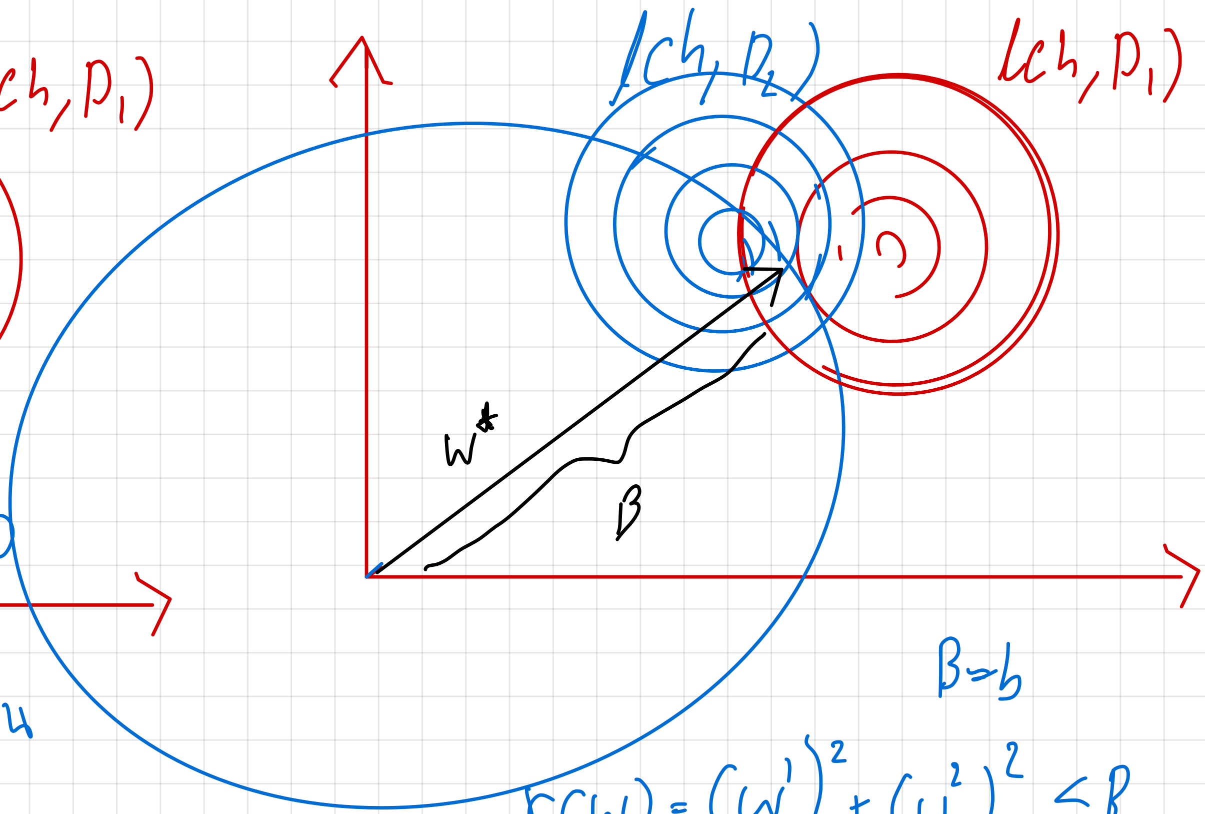
เป็นแค่ feature เดียว (Coordinate ที่สำคัญน้อยตัดออก)

w will be sparse solution

-set coordinate ที่ไม่สำคัญ ออก

## $l_2$ - Regularizer Review




∴ เปลี่ยน data set w เปลี่ยน

∴ w overfit

$r(w) = (w^1)^2 + (w^2)^2 \le \beta$

$w^*$ is optimal solution within the constraint

$w^*$ จะไม่อยู่ที่ minimum ของ loss

ถ้าเพิ่มค่า $\beta$ จะสามารถไปที่ minimum ของ loss ทั้ง $D_1, D_2$ ได้