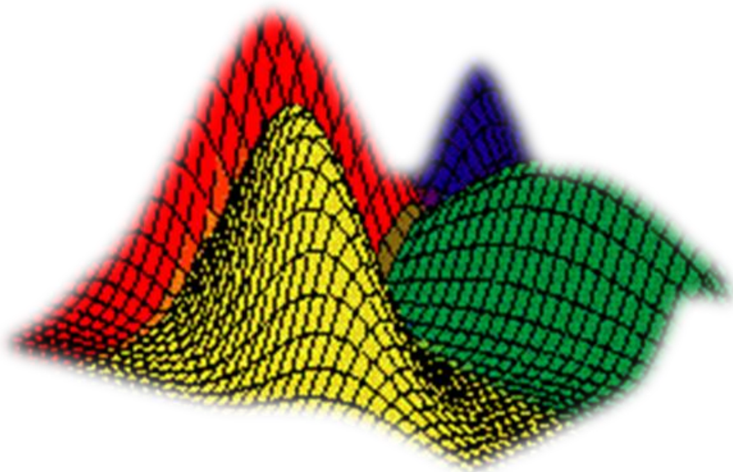
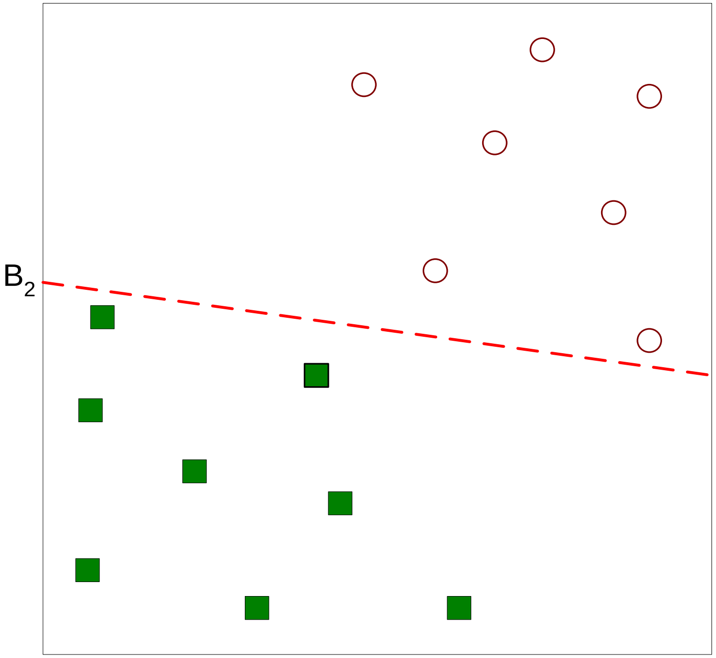


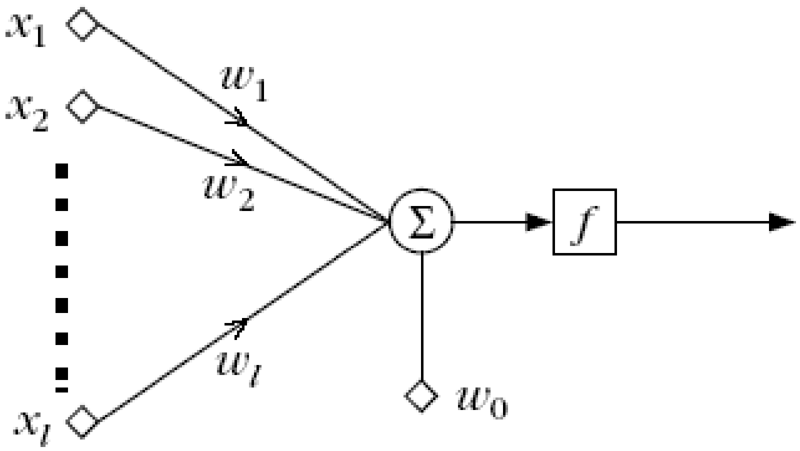
CSE 473
Pattern Recognition

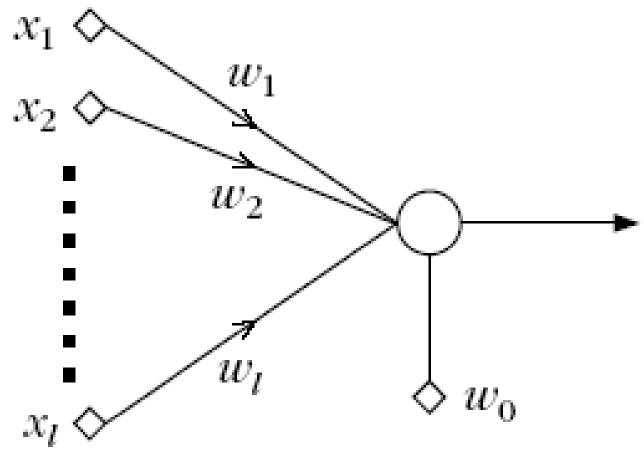
Linear Classifier: Introduction

- Classifies linearly separable patterns
- Assume proper forms for the discriminant functions
- may not be optimal
- very simple to use



The Perceptron

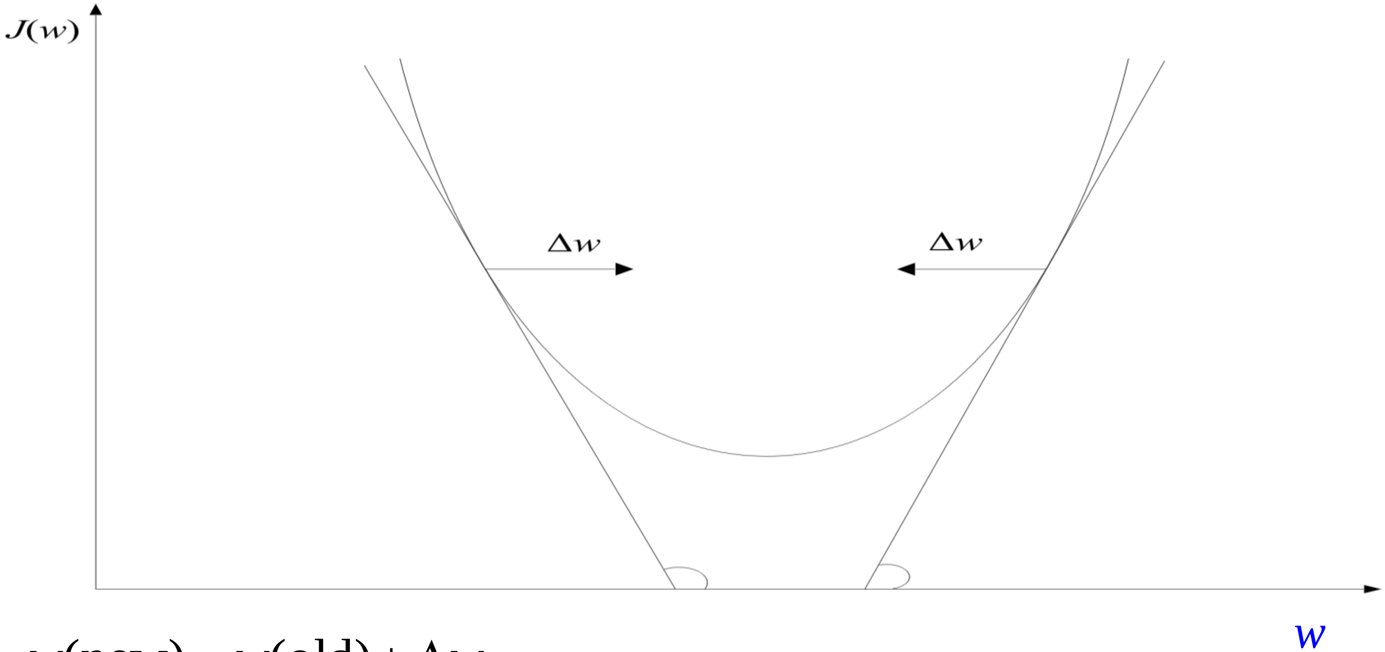




$w_i's$ synapses or synaptic weights

w_0 threshold

- This structure is called perceptron or neuron
- a learning machine that learns from the training vectors



$$\underline{w}(\text{new}) = \underline{w}(\text{old}) + \Delta \underline{w}$$

$$\Delta \underline{w} = -\mu \frac{\partial J(\underline{w})}{\partial \underline{w}} \Big| \underline{w} = \underline{w}(\text{old})$$

- Wherever valid

$$\frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}} \left(\sum_{\underline{x} \in Y} \delta_x \underline{w}^T \underline{x} \right) = \sum_{\underline{x} \in Y} \delta_x \underline{x}$$

- $$\boxed{\underline{w}(t+1) = \underline{w}(t) - \rho_t \sum_{\underline{x} \in Y} \delta_x \underline{x}}$$

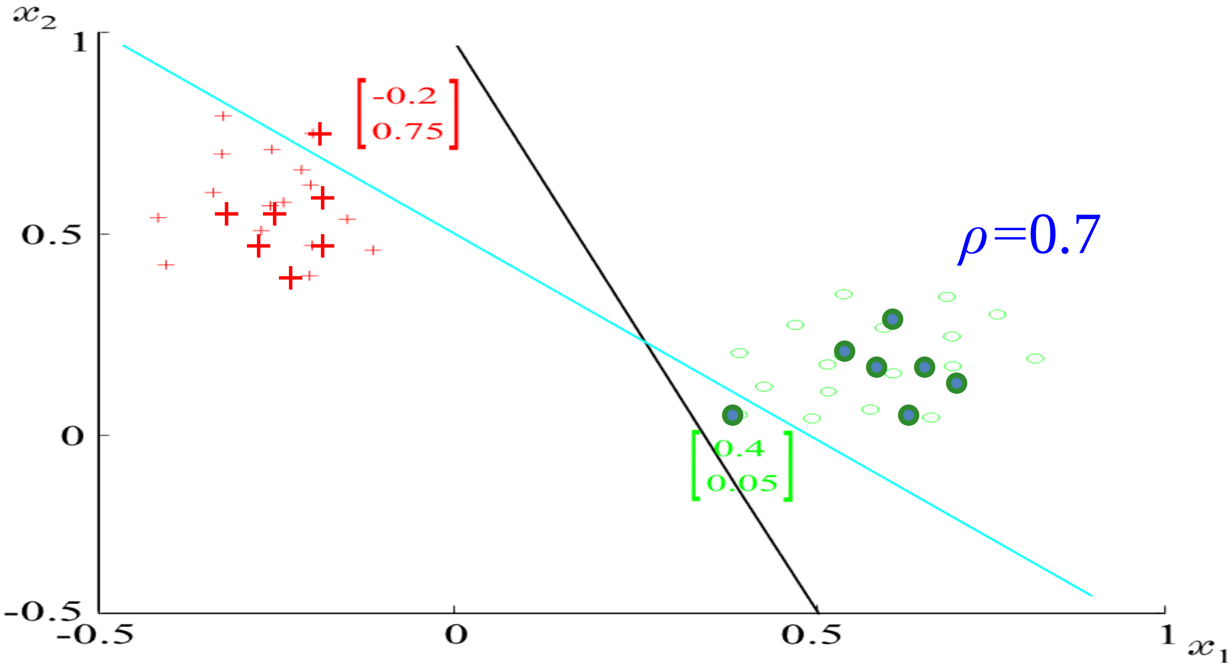
This is the celebrated Perceptron Algorithm

– Example: At some stage t the perceptron algorithm results in

$$w_1 = 1,\ w_2 = 1,\ w_0 = -0.5$$

$$x_1 + x_2 - 0.5 = 0$$

The corresponding hyperplane is



$$\underline{w}(t+1) = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7(-1)\begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} - 0.7(+1)\begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$

Variants of Perceptron Algorithm (1)

$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}_{(t)}\,, \quad \begin{matrix} \underline{w}^T(t)\underline{x}_{(t)} \le 0 \\ \underline{x}_{(t)} \in \omega_1 \end{matrix}$$

$$\underline{w}(t+1) = \underline{w}(t) - \rho \underline{x}_{(t)}\,, \quad \begin{matrix} \underline{w}^T(t)\underline{x}_{(t)} \ge 0 \\ \underline{x}_{(t)} \in \omega_2 \end{matrix}$$

$$\underline{w}(t+1) = \underline{w}(t) \quad \text{otherwise}$$

Variants of Perceptron Algorithm (1)

$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}_{(t)}\,, \quad \begin{matrix} \underline{w}^T(t)\underline{x}_{(t)} \le 0 \\ \underline{x}_{(t)} \in \omega_1 \end{matrix}$$

$$\underline{w}(t+1) = \underline{w}(t) - \rho \underline{x}_{(t)}\,, \quad \begin{matrix} \underline{w}^T(t)\underline{x}_{(t)} \ge 0 \\ \underline{x}_{(t)} \in \omega_2 \end{matrix}$$

update

$$\underline{w}(t+1) = \underline{w}(t) \quad \text{otherwise}$$

No Update

Variants of Perceptron Algorithm (1)

$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}_{(t)}, \quad \begin{array}{c} \underline{w}^T(t)\underline{x}_{(t)} \leq 0 \\ \underline{x}_{(t)} \in \omega_1 \end{array}$$

update

$$\underline{w}(t+1) = \underline{w}(t) - \rho \underline{x}_{(t)}, \quad \begin{array}{c} \underline{w}^T(t)\underline{x}_{(t)} \geq 0 \\ \underline{x}_{(t)} \in \omega_2 \end{array}$$

$$\underline{w}(t+1) = \underline{w}(t) \quad \text{otherwise}$$

No Update

– It is a reward and punishment type of algorithm

Variants of Perceptron Algorithm (2)

- initialize weight vector $\mathbf{w}(0)$
- define pocket $\mathbf{w}_s$.and history h_s
- generate next $\mathbf{w}(t+1)$. If it is better than $\mathbf{w}(t)$, store $\mathbf{w}(t+1)$ in $\mathbf{w}_s$ and change the h_s

Variants of Perceptron Algorithm (2)

- initialize weight vector $\mathbf{w}(0)$
- define pocket $\mathbf{w}_s$.and history h_s
- generate next $\mathbf{w}(t+1)$. If it is better than $\mathbf{w}(t)$, store $\mathbf{w}(t+1)$ in $\mathbf{w}_s$ and change the h_s

– It is pocket algorithm

Generalization of Perceptron Algorithm for M- Class case

Generalization of Perceptron Algorithm for *M*- Class case

- Let M classes $\omega_{1,}$ $\omega_{2,}$ $\omega_{3,}$ . . ., $\omega_{M,}$
- Let M linear discriminant functions, $\underline{w}_i$

Generalization of Perceptron Algorithm for M- Class case

- Let M classes $\omega_{1,}\ \omega_{2,}\ \omega_{3,}\ . . ., \omega_{M,}$
- Let M linear discriminant functions, $\underline{w}_i$
- The object $\underline{x}$ is classified to ω_i, if

$$\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}, \quad \forall j \neq i$$

Generalization of Perceptron Algorithm for M- Class case

- The object $\underline{x}$ is classified to ω_i, if

$$\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}, \quad \forall j \neq i$$

$$[w_i^T - w_j^T].x > 0$$

Generalization of Perceptron Algorithm for M- Class case

- The object $\underline{x}$ is classified to ω_i, if

$$\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}, \quad \forall j \neq i$$

$$[w_i^T - w_j^T].x > 0$$

can be written as

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, -w_j^T \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, x^T \cdots, 0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[0^T,\cdots,0^T,w_i^T,\cdots,0^T,-w_j^T\cdots,0^T].$$

$$[0^T,\cdots,0^T,x^T,\cdots,0^T,x^T\cdots,0^T]^T>0$$

Generalization of Perceptron Algorithm for M- Class case

$$[0^T,\cdots,0^T,w_i^T,\cdots,0^T,-w_j^T\cdots,0^T].$$

$$[0^T,\cdots,0^T,x^T,\cdots,0^T,x^T\cdots,0^T]^T>0$$

$$[0^T,\cdots,0^T,w_i^T,\cdots,0^T,w_j^T\cdots,0^T].$$

$$[0^T,\cdots,0^T,x^T,\cdots,0^T,-x^T\cdots,0^T]^T>0$$

Generalization of Perceptron Algorithm for M- Class case

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, \boxed{-w_j^T} \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, \boxed{x^T} \cdots, 0^T]^T > 0$$

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, \boxed{w_j^T} \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, \boxed{-x^T} \cdots, 0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[w_i^T - w_j^T].x > 0$$

$$[0^T,\cdots,0^T,w_i^T,\cdots,0^T,w_j^T\cdots,0^T].$$
$$[0^T,\cdots,0^T,x^T,\cdots,0^T,-x^T\cdots,0^T]^T > 0$$

$$[w_1^T,w_2^T\cdots,w_i^T,\cdots,w_j^T\cdots,w_M^T].$$
$$[0^T,\cdots,0^T,x^T,\cdots,0^T,-x^T\cdots,0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T > 0$$

Let, $w = [w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T]^T$

and $x_{i,j} = [0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T$

Then, the condition is $w^T x_{i,j} > 0$

Generalization of Perceptron Algorithm for M- Class case

- For each training vector of class ω_i, construct

$$x_{i,j} = [0^T, \cdots, 0^T, \underbrace{x^T}_{i\text{th location}}, \cdots, 0^T, \underbrace{-x^T}_{j\text{th location}} \cdots, 0^T]^T$$

$(l+1)M$ dimension

- Concatenate the weight vectors:

$$w = [w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T]^T$$

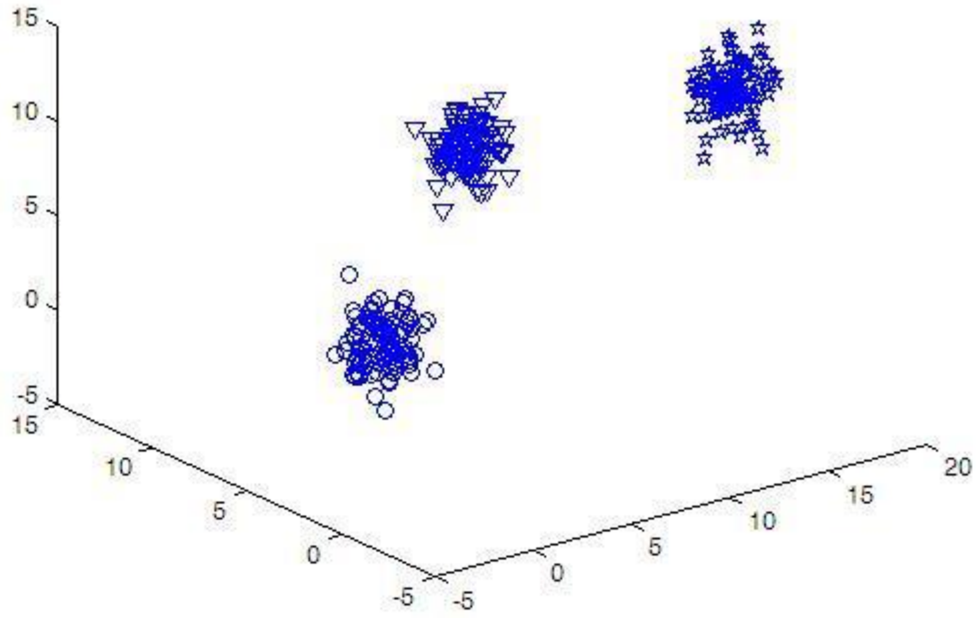
Generalization of Perceptron Algorithm for M- Class case

$$x_{i,j} = [0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T$$

$$w = [w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T]^T$$

- Use a single Perceptron to solve
- Parameters:
  - $(l+1)M$ feature dimension
  - All $N(M\text{-}1)$ training vectors to be on positive side
- This reorganization is known as Kesler's construction

Sample Data for Sessional on Perceptron Algorithms (Week 3 and 4)



Sample Data for Perceptron

Classes	*Features*	*Samples*
3	**3**	**300**

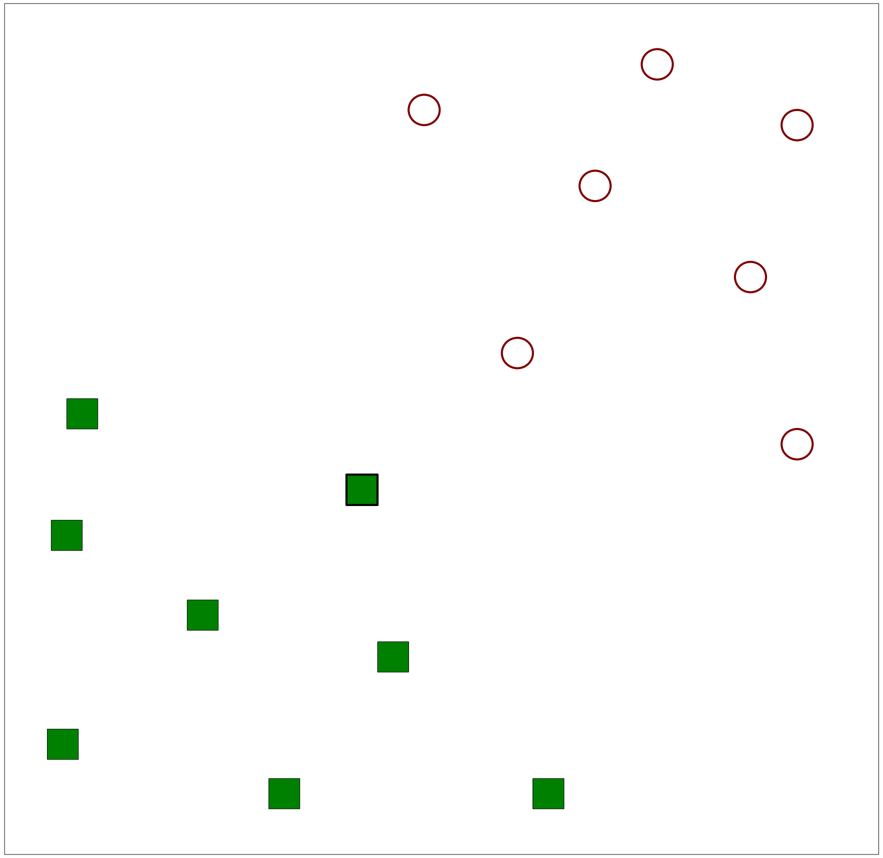
Feature1	*Feature2*	*Feature3*	*Class*
11.0306	9.0152	8.0199	1
11.4008	8.7768	6.7652	1
11.2489	9.5744	8.0812	1
9.3157	7.4360	5.6128	1
15.7777	1.5879	11.4440	2
15.8685	2.7902	11.2532	2
14.9448	0.7798	12.7481	2
15.9801	1.0142	14.2029	2
2.3979	5.6525	2.7566	3
2.5103	6.3484	1.4272	3
2.7527	4.6571	3.1138	3
-0.0195	4.5524	0.0118	3

What to do?

- No. of features and classes will be variable
- Use training file to train (1) *basic*, (2) *reward and punishment*, (3) *pocket* and (4) *kesler's reconsturction*
- Use test file to evaluate the performance and identify the misclassified samples
- Any programming language cannot be used
- Upload your program to moodle by 10 pm on next Sunday
- Use different data files during evaluation

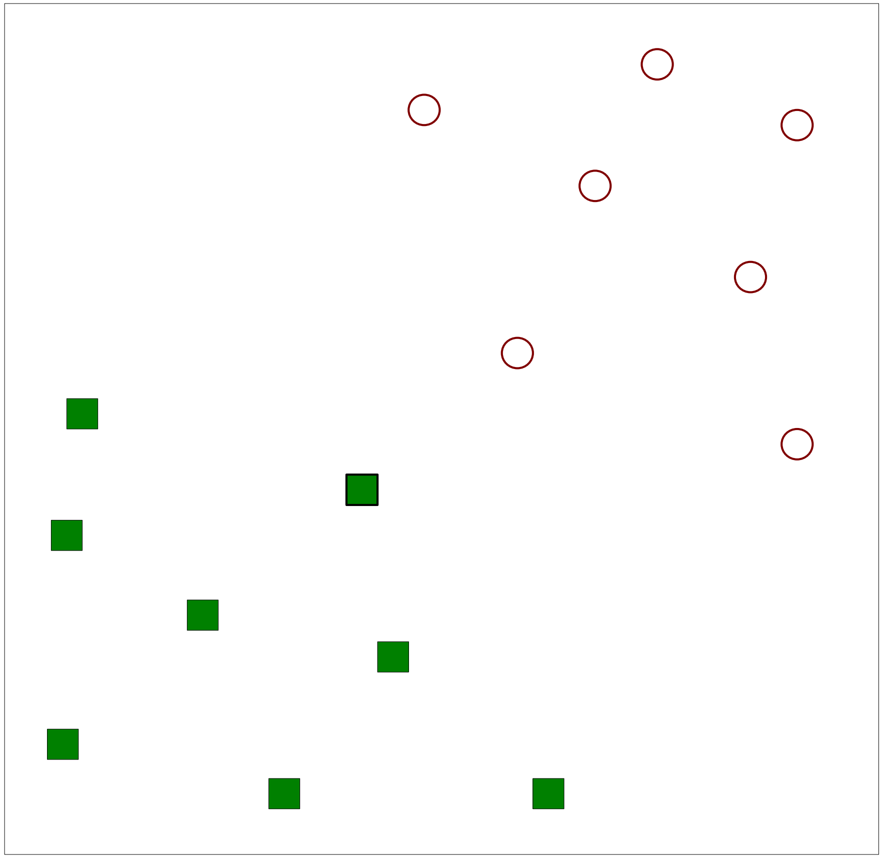
Two-Class case *again*

Let, we have N training samples



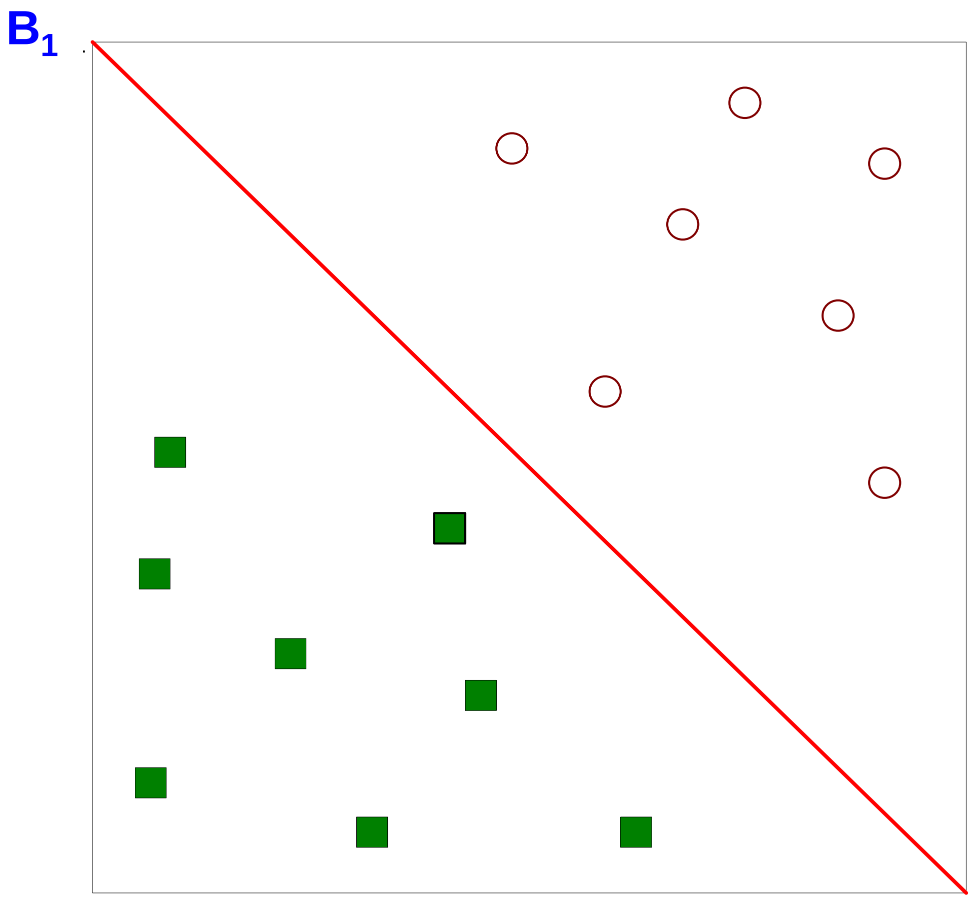
Two-Class case *again*

Let, we have N training samples



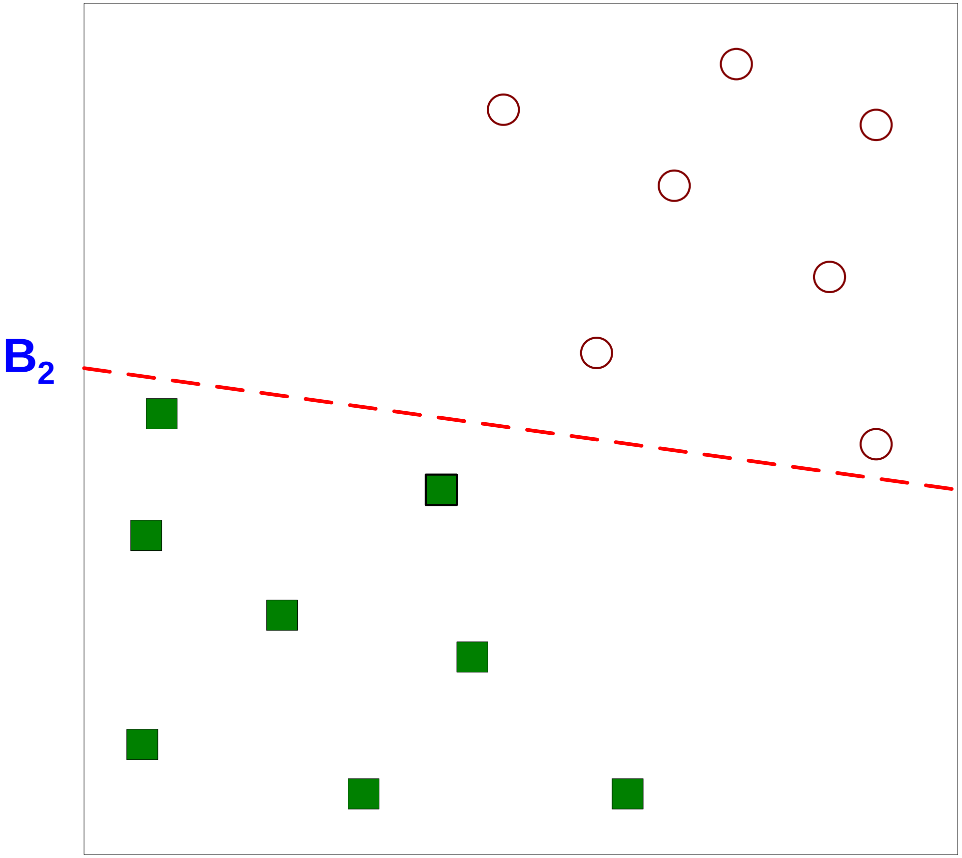
- Find a linear hyperplane (decision boundary) that will separate the data

Two-Class case again



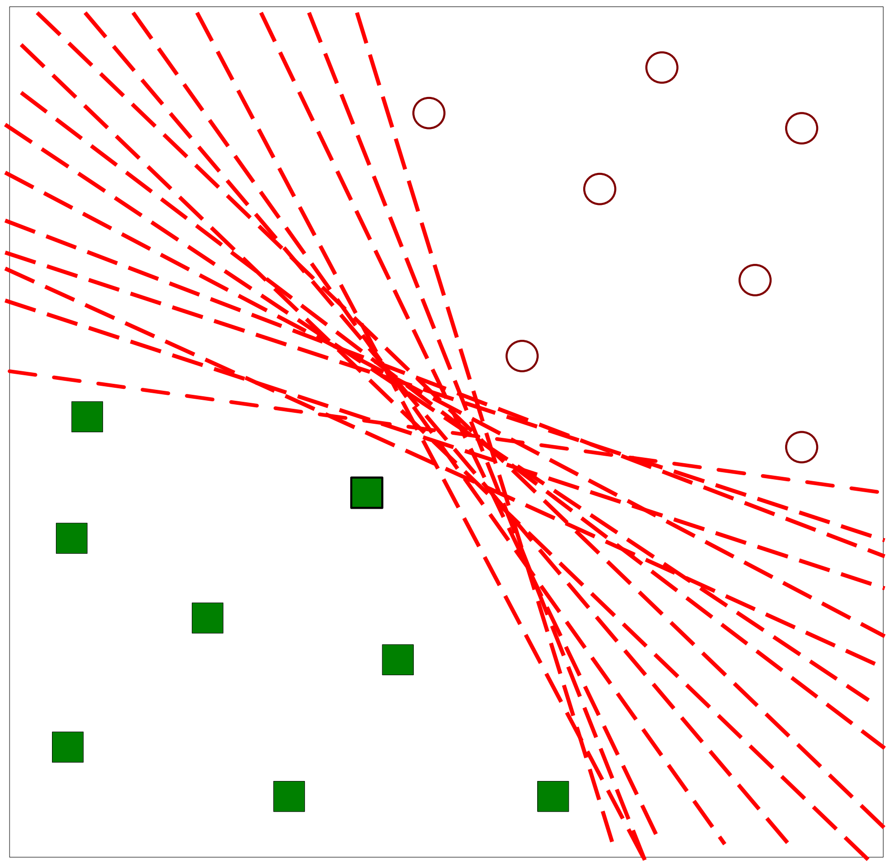
- One Possible Solution

Two-Class case again



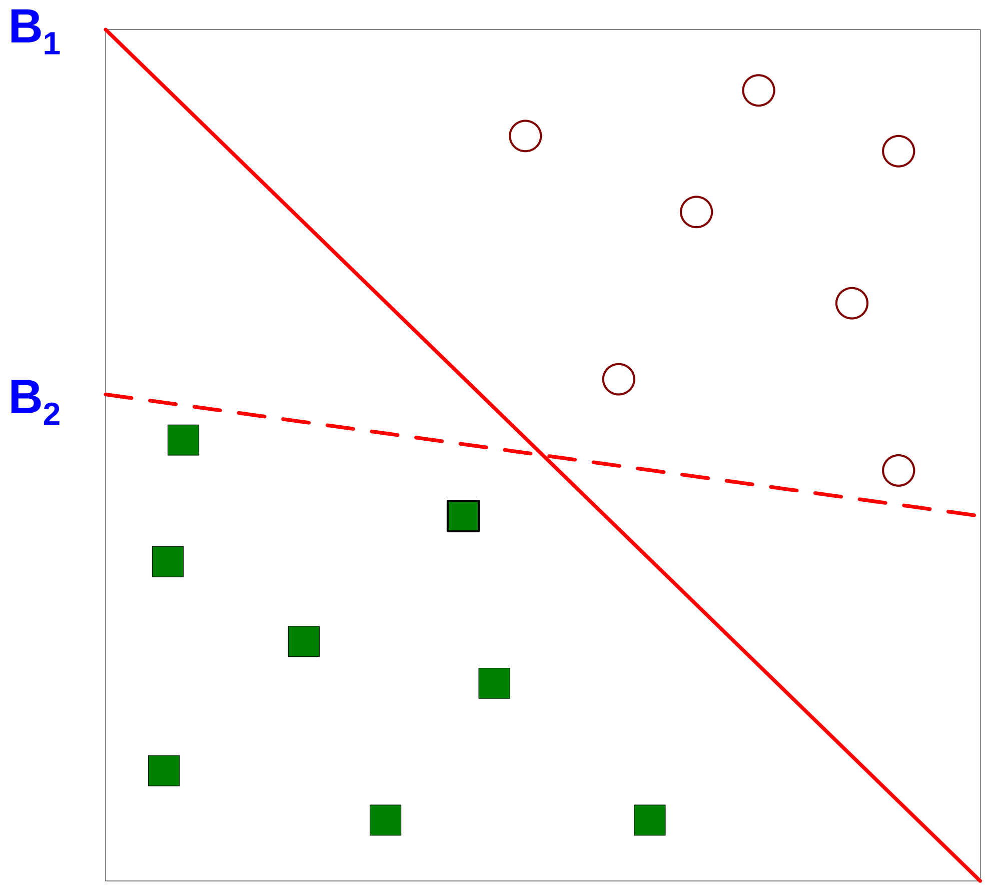
- Another possible solution

Two-Class case again



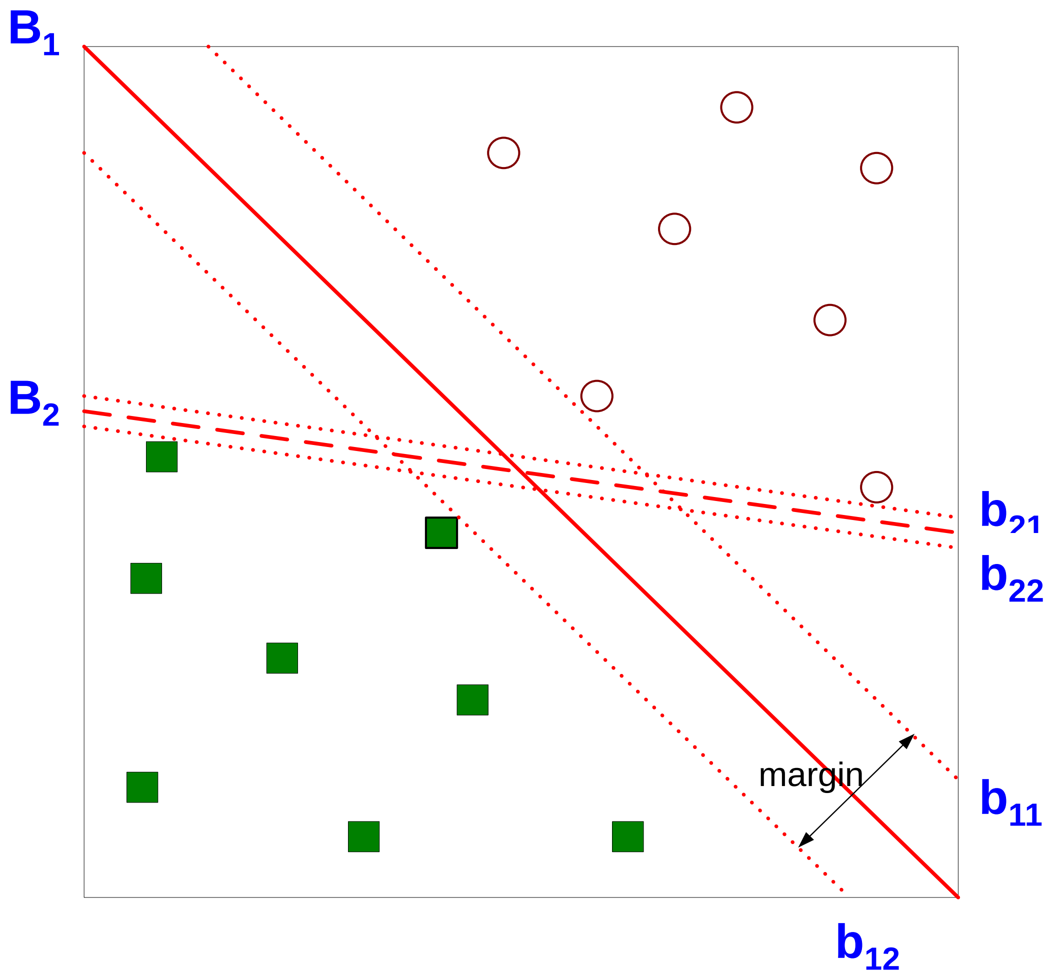
- Other possible solutions

Two-Class case again



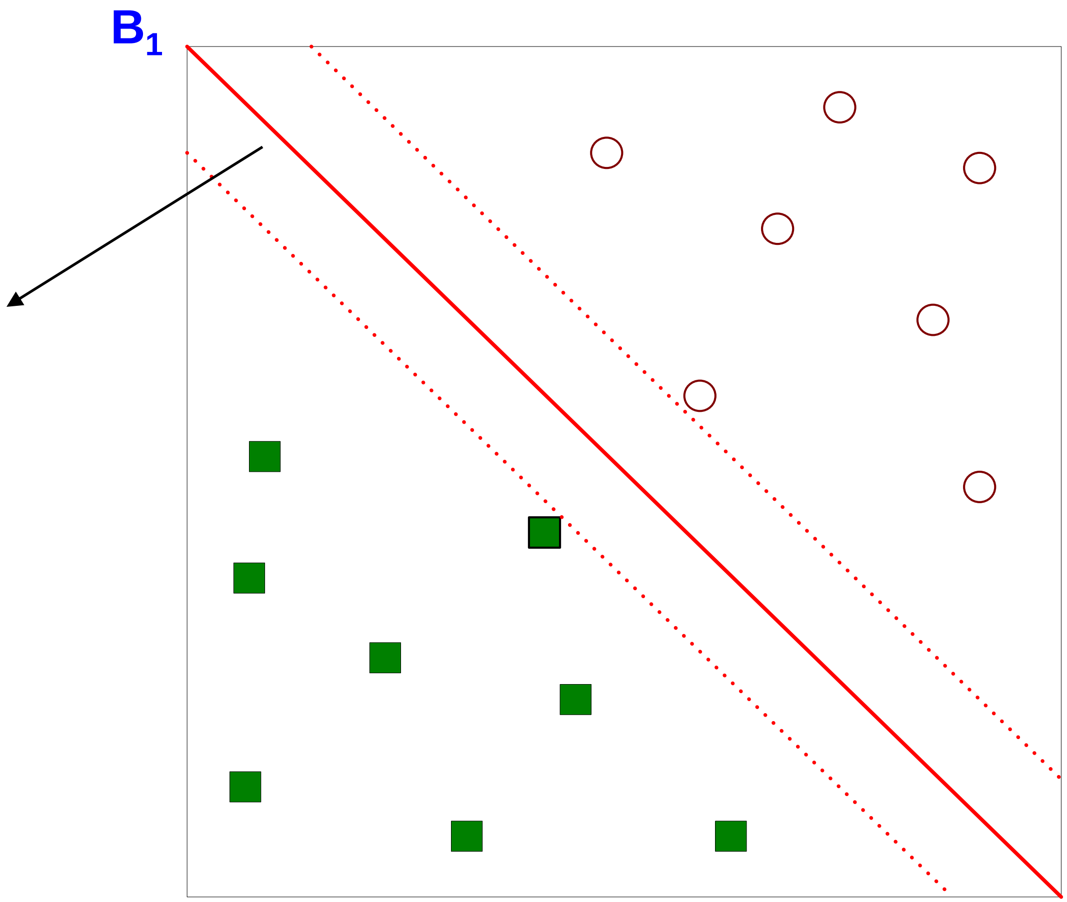
- Which one is better? B_1 or B_2?
- How do you define better?

Two-Class case again



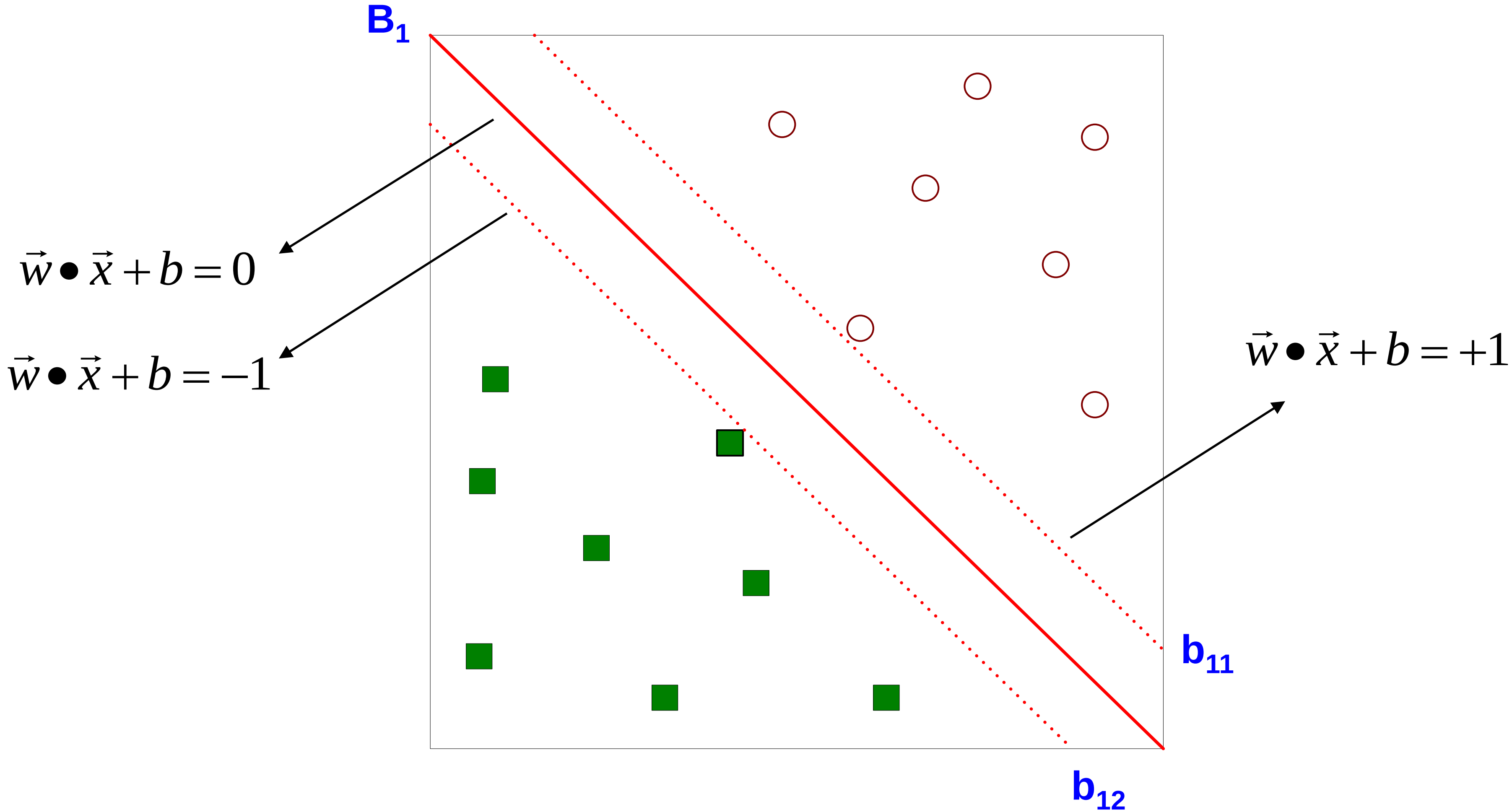
- Find hyperplane maximizes the margin => B1 is better than B2

Two-Class case again

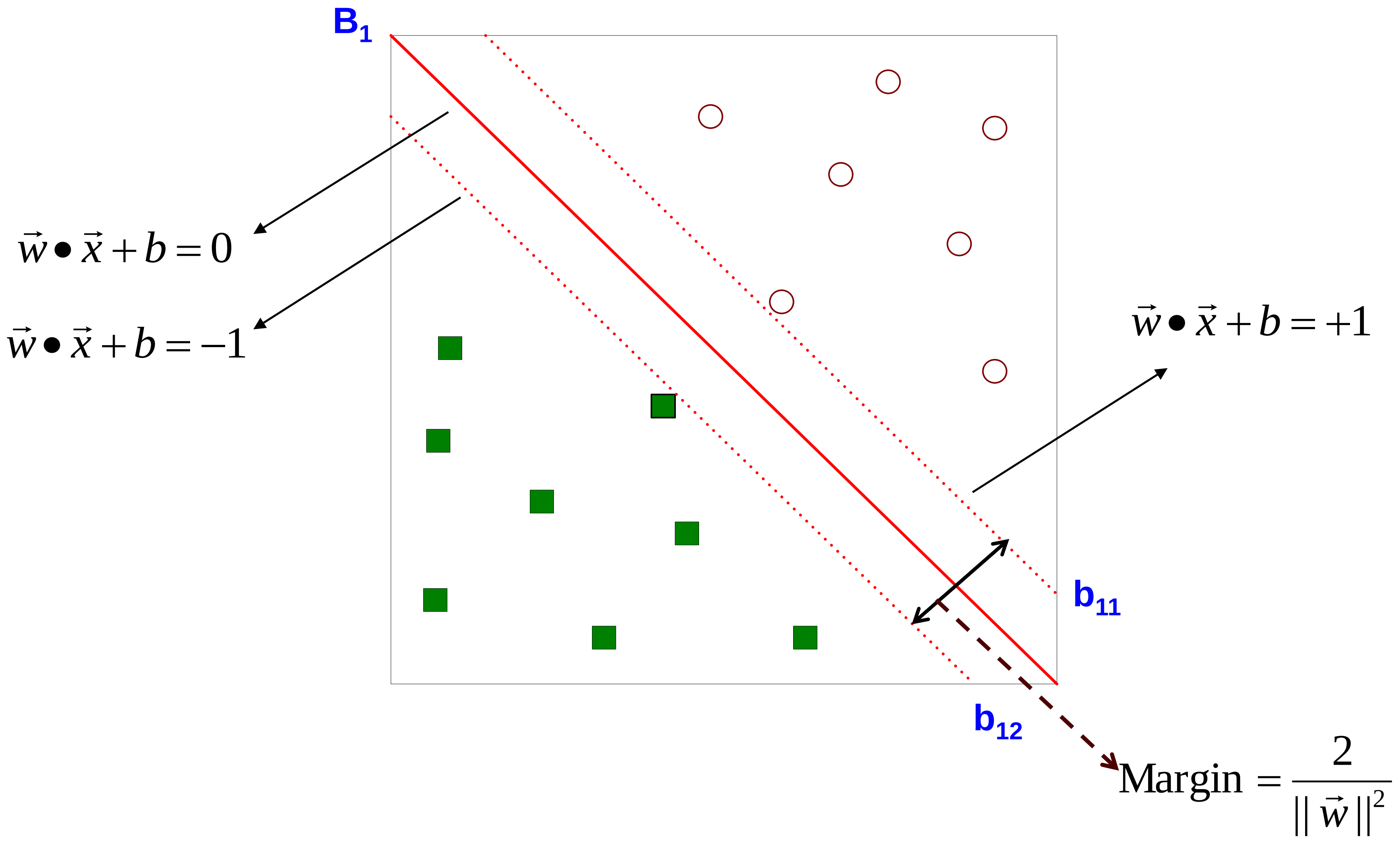


$$\vec{w} \bullet \vec{x} + b = 0$$

Two-Class case again



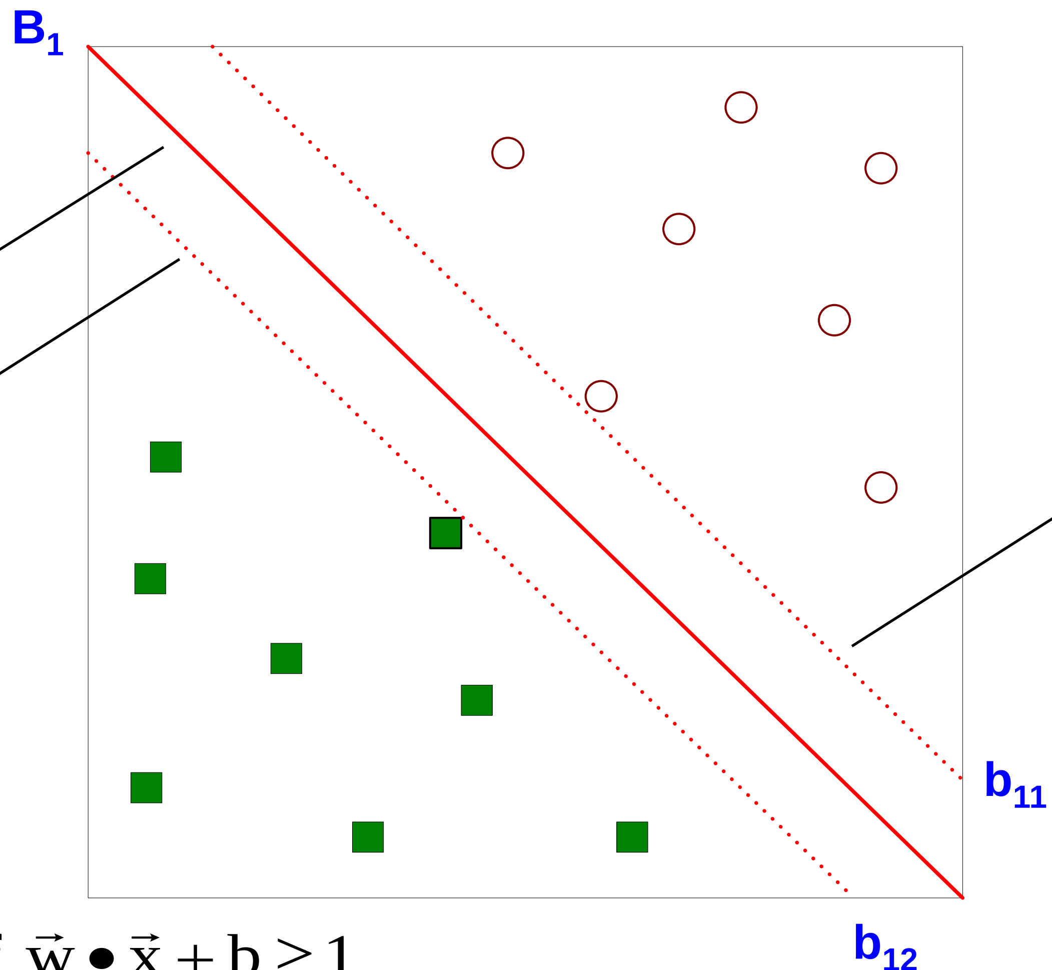
Two-Class case again



Two-Class case again

$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$



$$\vec{w} \bullet \vec{x} + b = +1$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

Support Vector Machine

- We want to maximize: $$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

  – subject to the following constraints:

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 \end{cases}$$

Support Vector Machine

- We want to maximize: $$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$
  - subject to the following constraints:

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 \end{cases} \quad \forall_i$$

Support Vector Machine

- We want to maximize: $$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$
  - Which is equivalent to minimizing: $$L(w) = \frac{\|\vec{w}\|^2}{2}$$
  - subject to the following constraints:
    - $$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 \end{cases}$$

Support Vector Machine

The Expression

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 \end{cases}$$

can be written as

$$y_i(\vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b}) \geq 1$$

Support Vector Machine

The Expression

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 \end{cases}$$

can be written as

$$y_i(\vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b}) \geq 1$$

- We can say :
  - minimize:

$$L(w) = \frac{\|\vec{w}\|^2}{2}$$

  - Subject to:

$$y_i(\vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b}) \geq 1$$

Support Vector Machine

- $L(w) = \dfrac{\|\vec{w}\|^2}{2}$ is a quadratic equation

- Solving for ***w*** and ***b*** is not easy

Support Vector Machine

- $L(w) = \frac{\|\vec{w}\|^2}{2}$ is a quadratic equation
- Solving for ***w*** and ***b*** is not easy
- What happens if ***w*** =0?

Support Vector Machine

- $L(w) = \dfrac{\|\vec{w}\|^2}{2}$ is a quadratic equation
- Solving for ***w*** and ***b*** is not easy
- What happens if ***w*** =0?

Some of $y_i(\vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b}) \geq 1$ may be infeasible

Support Vector Machines

- minimize: $$L(w) = \frac{\|\vec{w}\|^2}{2}$$
- Subject to: $$y_i(\vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b}) \geq 1 \qquad \forall_i$$

- Use Lagrange function:

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (w.x_i + b) - 1 \right)$$

Support Vector Machines

- Lagrange function:

$$L_p = \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

- New constraints are:

$$\frac{\partial L_p}{\partial \vec{w}} = 0$$

$$\frac{\partial L_p}{\partial b} = 0$$

Support Vector Machines

- Lagrange function:

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

- New constraints are:

$$\frac{\partial L_p}{\partial \vec{w}} = 0 \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i$$

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

Support Vector Machines

- Lagrange function:

$$L_p = \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

- constraints are:

$$\vec{w} = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

Still not solvable, many variables

Support Vector Machines

- Use Lagrange function:

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

- constraints are:

$$\vec{w} = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

From Karush-Kuhn_Tucker Transform,

$$\lambda_i \geq 0$$

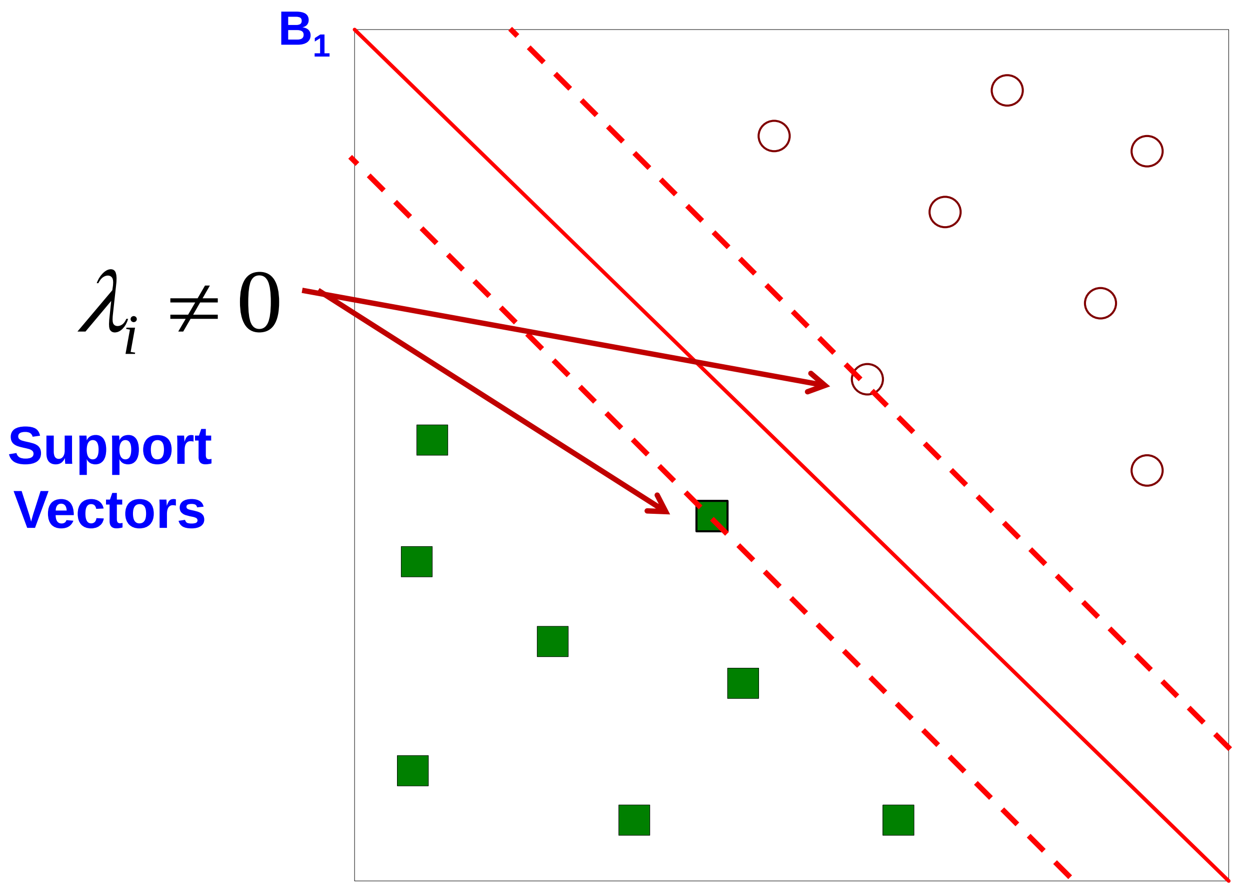
$$\lambda_i \left[y_i (\vec{w}.\vec{x}_i + b) - 1 \right] = 0$$

Support Vector Machines

$$\lambda_i \geq 0 \quad : \text{non-negative}$$

$$\lambda_i \left[y_i (\vec{w}.\vec{x}_i + b) - 1 \right] = 0$$

Support Vector Machines



$$\lambda_i \geq 0$$

$$\lambda_i \left[y_i (\vec{w}.\vec{x}_i + b) - 1 \right] = 0$$

Support Vector Machines

- Replace w with λ's in L_p :

put $\vec{w} = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i$ and $\sum_{i=1}^{N} \lambda_i y_i = 0$

in $L_p = \frac{||\vec{w}||^2}{2} - \sum_{i=1}^{N} \lambda_i \big(y_i (\vec{w}.\vec{x}_i + b) - 1 \big)$

$$L_p = \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w} . \vec{x}_i + b) - 1 \right)$$

$$L_p = \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

$$= \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

$$= \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i - b \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i$$

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

$$= \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i - b\sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\vec{w} - b \times 0 + \sum_{i=1}^{N} \lambda_i$$

$$\vec{w} = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

$$= \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i - b\sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\vec{w} - b \times 0 + \sum_{i=1}^{N} \lambda_i$$

$$= \sum_{i=1}^{N} \lambda_i + \frac{\vec{w}.\vec{w}}{2} - \vec{w}.\vec{w}$$

$$
\begin{aligned}
L_p &= \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \Big(y_i (\vec{w}.\vec{x}_i + b) - 1 \Big) \\
&= \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i \\
&= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i - b \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \\
&= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\vec{w} - b \times 0 + \sum_{i=1}^{N} \lambda_i \\
&= \sum_{i=1}^{N} \lambda_i + \frac{\vec{w}.\vec{w}}{2} - \vec{w}.\vec{w} \\
&= \sum_{i=1}^{N} \lambda_i - \frac{\vec{w}.\vec{w}}{2}
\end{aligned}
$$

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

$$= \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\|\vec{w}\|^2}{2} - \vec{w}.\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i - b\sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i$$

.

.

$$= \sum_{i=1}^{N} \lambda_i - \frac{\vec{w}.\vec{w}}{2}$$

$$= \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i \cdot \sum_{j=1}^{N} \lambda_j y_j \vec{x}_j$$

$$L_p = \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

$$= \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i y_i \vec{w}.\vec{x}_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{\| \vec{w} \|^2}{2} - \vec{w}.\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i - b\sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i$$

.

.

$$= \sum_{i=1}^{N} \lambda_i - \frac{\vec{w}.\vec{w}}{2}$$

$$= \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i \cdot \sum_{j=1}^{N} \lambda_j y_j \vec{x}_j$$

$$= \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$

Support Vector Machines

- Replace w with λ's in L_p :

$$L_p = \frac{\| \vec{w} \|^2}{2} - \sum_{i=1}^{N} \lambda_i \left(y_i (\vec{w}.\vec{x}_i + b) - 1 \right)$$

- The dual to be maximized:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}$$

Support Vector Machines

- After solving λ's :
  - Find ***w*** and b:

$$\frac{\partial L_p}{\partial w} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i}$$

$$\vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} = 1$$

Support Vector Machines

- Classify an unknown example ***z***:

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \mathbf{z} + b)$$