Rosenblatt Perceptron

Neuroinformatics Tutorial 6

Duc Duy Pham[1]

[1]Intelligent Systems, Faculty of Engineering,
University of Duisburg-Essen, Germany

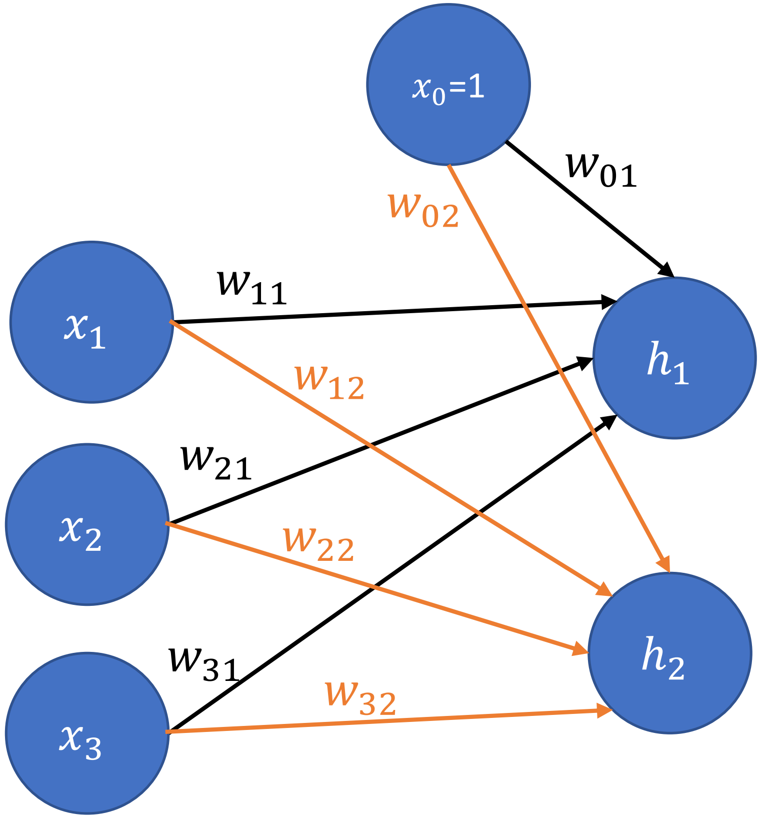
Content

- Revision: Practical Task
- Revision: Lecture
- New Practical Task

Content

- **Revision: Practical Task**
- Revision: Lecture
- New Practical Task

Calculation of propagated value



$$h_1 = \sum_{i=0}^{3} w_{i1} x_i$$

$$h_2 = \sum_{i=0}^{3} w_{i2} x_i$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad W = \begin{bmatrix} w_{01} & w_{02} \\ w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \qquad \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = W^T \cdot x$$

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \ldots x_n \in \{-1, 1\}$

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \ldots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$

  - Weights in between all neurons

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$

  - Weights in between all neurons

  - Learning rate α

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$

  - Weights in between all neurons

  - Learning rate α

- Learning Rule:
  - Update weights by comparing similarity of outputs

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$
  - Weights in between all neurons
  - Learning rate α
- Learning Rule:
  - Update weights by comparing similarity of outputs
  - $\Delta w_{i,j} := \alpha \cdot x_i \cdot h_j$

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$
  - Weights in between all neurons
  - Learning rate α
- Learning Rule:
  - Update weights by comparing similarity of outputs
  - $\Delta w_{i,j} \coloneqq \alpha \cdot x_i \cdot h_j$
  - $w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$
  - Weights in between all neurons
  - Learning rate α
- Learning Rule:
  - Update weights by comparing similarity of outputs
  - $\Delta w_{i,j} := \alpha \cdot x_i \cdot h_j$
  - $w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$

Hebbian Learning Rule

- Given:
  - Outputs of previous neurons (can also be input neurons)
  - In case of scheme : input neurons, i.e:

    $x_0, x_1, x_2, \dots x_n \in \{-1, 1\}$

  - Output of current neurons
  - In case of scheme: $h_1, h_2 \in \{-1, 1\}$

  - Weights in between all neurons

  - Learning rate α

- Learning Rule:
  - Update weights by cor
  - $\Delta w_{i,j} \coloneqq \alpha \cdot x_i \cdot h_j$
  - $w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$

$$\begin{pmatrix} x_0 \\ \vdots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} h_1 & h_2 \end{pmatrix}$$

Content

- Revision: Practical Task
- **Revision: Lecture**
- New Practical Task

Revision: Lecture

- Which statements regarding Rosenblatt Perceptron are true?

Revision: Lecture

- Which statements regarding Rosenblatt Perceptron are true?
  1. The weights and threshold of a RBP defines a hyperplane
  2. The extended version of the RBP lifts the input space into a higher dimension
  3. In the extended version of the RBP the hyperplane always goes through the origin
  4. The RBP can be viewed as a generalized McCulloch Pitts Neuron

Revision: Lecture

- Which statements regarding Rosenblatt Perceptron are true?
  1. The weights and threshold of a RBP defines a hyperplane
  2. The extended version of the RBP lifts the input space into a higher dimension
  3. In the extended version of the RBP the hyperplane always goes through the origin
  4. The RBP can be viewed as a generalized McCulloch Pitts Neuron

A: all

B: 1,2,4

C: 1

D: 1,3,4

Revision: Lecture

- Which statements regarding Rosenblatt Perceptron are true?
  1. The weights and threshold of a RBP defines a hyperplane
  2. The extended version of the RBP lifts the input space into a higher dimension
  3. In the extended version of the RBP the hyperplane always goes through the origin
  4. The RBP can be viewed as a generalized McCulloch Pitts Neuron

A: all

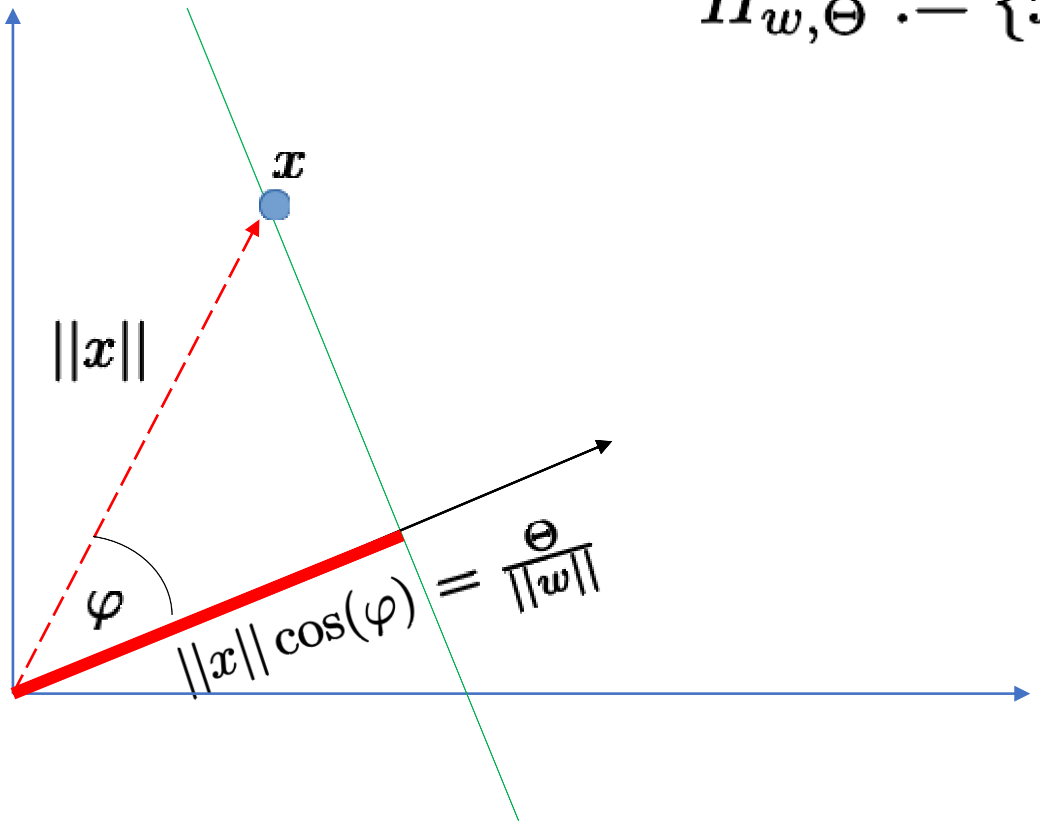
B: 1,2,4

C: 1

D: 1,3,4

Revision: Lecture

- Interpretation of formulas

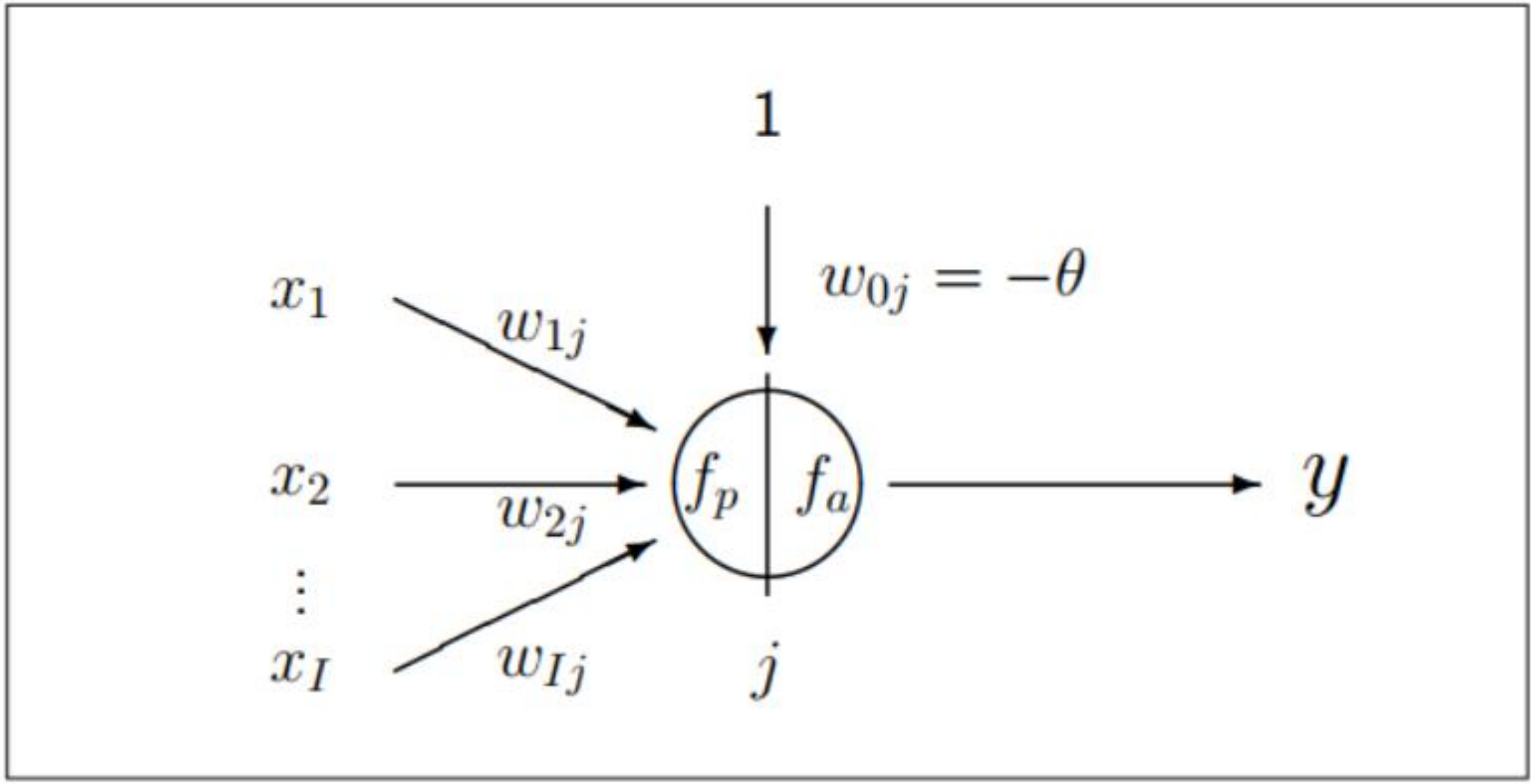


$$H_{w,\Theta} := \{x \in \mathbb{R}^n | w^T x = \Theta\}, w \in \mathbb{R}^n, \Theta \in \mathbb{R}$$

$$\Rightarrow \Theta = ||w|| \cdot ||x|| \cdot \cos(\varphi)$$

$$\Rightarrow ||x|| \cdot \cos(\varphi) = \frac{\Theta}{||w||}$$

Scheme of Artificial Neuron



$f_p|f_a$ wird oft weggelassen, wenn aus dem Zusammenhang klar.

Rosenblatt Learning Rule

- How does the Rosenblatt Learning Rule/Algorithm work? (must know!)

Rosenblatt Learning Rule

- How does the Rosenblatt Learning Rule/Algorithm work? (must know!)
  - Let $w := (-\Theta, w_1, \ldots, w_n)^T \in \mathbb{R}^{n+1}$
    denote the extended weight vector including the bias

Rosenblatt Learning Rule

- How does the Rosenblatt Learning Rule/Algorithm work? (must know!)
  - Let $w := (-\Theta, w_1, \dots, w_n)^T \in \mathbb{R}^{n+1}$ denote the extended weight vector including the bias
  - Let $w(i)$ denote the weight vector at iteration i

Rosenblatt Learning Rule

- How does the Rosenblatt Learning Rule/Algorithm work? (must know!)
  - Let $w := (-\Theta, w_1, \ldots, w_n)^T \in \mathbb{R}^{n+1}$ denote the extended weight vector including the bias
  - Let $w(i)$ denote the weight vector at iteration i
  - Let $x := (1, x_1, \ldots, x_n)^T \in \Omega := \mathcal{P} \cup \mathcal{N} \subset \mathbb{R}^{n+1}$ denote an arbitrary extended sample point from the trainind data set

Rosenblatt Learning Rule

- Let $\hat{y}(x) := \begin{cases} 1 & \text{if } x \in \mathcal{P} \\ -1 & \text{else} \end{cases}$ denote the desired target output

Rosenblatt Learning Rule

- Let $\hat{y}(x) := \begin{cases} 1 & \text{if } x \in \mathcal{P} \\ -1 & \text{else} \end{cases}$ denote the desired target output
- Let $\tilde{y}_{w(i)}(x) := f_a(f_p(x))$ denote the actual output of the perceptron with weight vector $w(i)$

Rosenblatt Learning Rule

- Idea:
  - Draw a sample point $\boldsymbol{x}$ randomly

Rosenblatt Learning Rule

- Idea:
  - Draw a sample point $\boldsymbol{x}$ randomly
  - Check if perceptron output is target output

Rosenblatt Learning Rule

- Idea:
  - Draw a sample point $\boldsymbol{x}$ randomly
  - Check if perceptron output is target output
  - If not:
    - If should have been positive: add $\boldsymbol{x}$ to weight vector

Rosenblatt Learning Rule

- Idea:
  - Draw a sample point x randomly
  - Check if perceptron output is target output
  - If not:
    - If should have been positive: add x to weight vector
    - If should have been negative: subtract x from weight vector

Rosenblatt Learning Rule

- If $\hat{y}(x) == \tilde{y}_{w(i)}(x)$

  Do nothing

Rosenblatt Learning Rule

- If $\hat{y}(x) == \tilde{y}_{w(i)}(x)$

  Do nothing

- If $\hat{y}(x) \neq \tilde{y}_{w(i)}(x)$

Rosenblatt Learning Rule

- If $\hat{y}(x) == \tilde{y}_{w(i)}(x)$

  Do nothing

- If $\hat{y}(x) \neq \tilde{y}_{w(i)}(x)$
  - If $\hat{y}(x) == 1$

    $w(i+1) \leftarrow w(i) + x$

Rosenblatt Learning Rule

- If $\hat{y}(x) == \tilde{y}_{w(i)}(x)$

  Do nothing

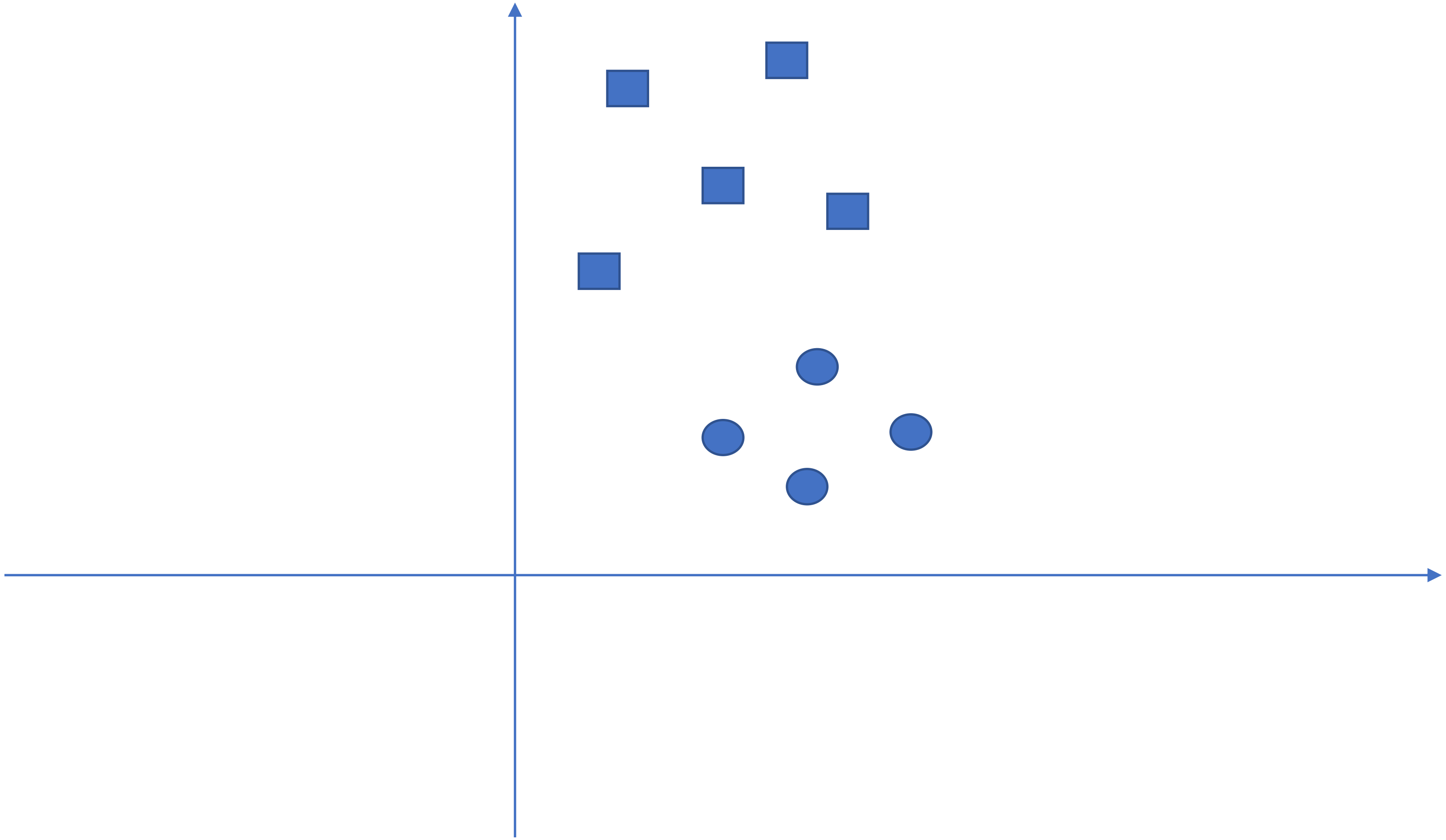
- If $\hat{y}(x) \neq \tilde{y}_{w(i)}(x)$
  - If $\hat{y}(x) == 1$

    $$w(i+1) \leftarrow w(i) + x$$

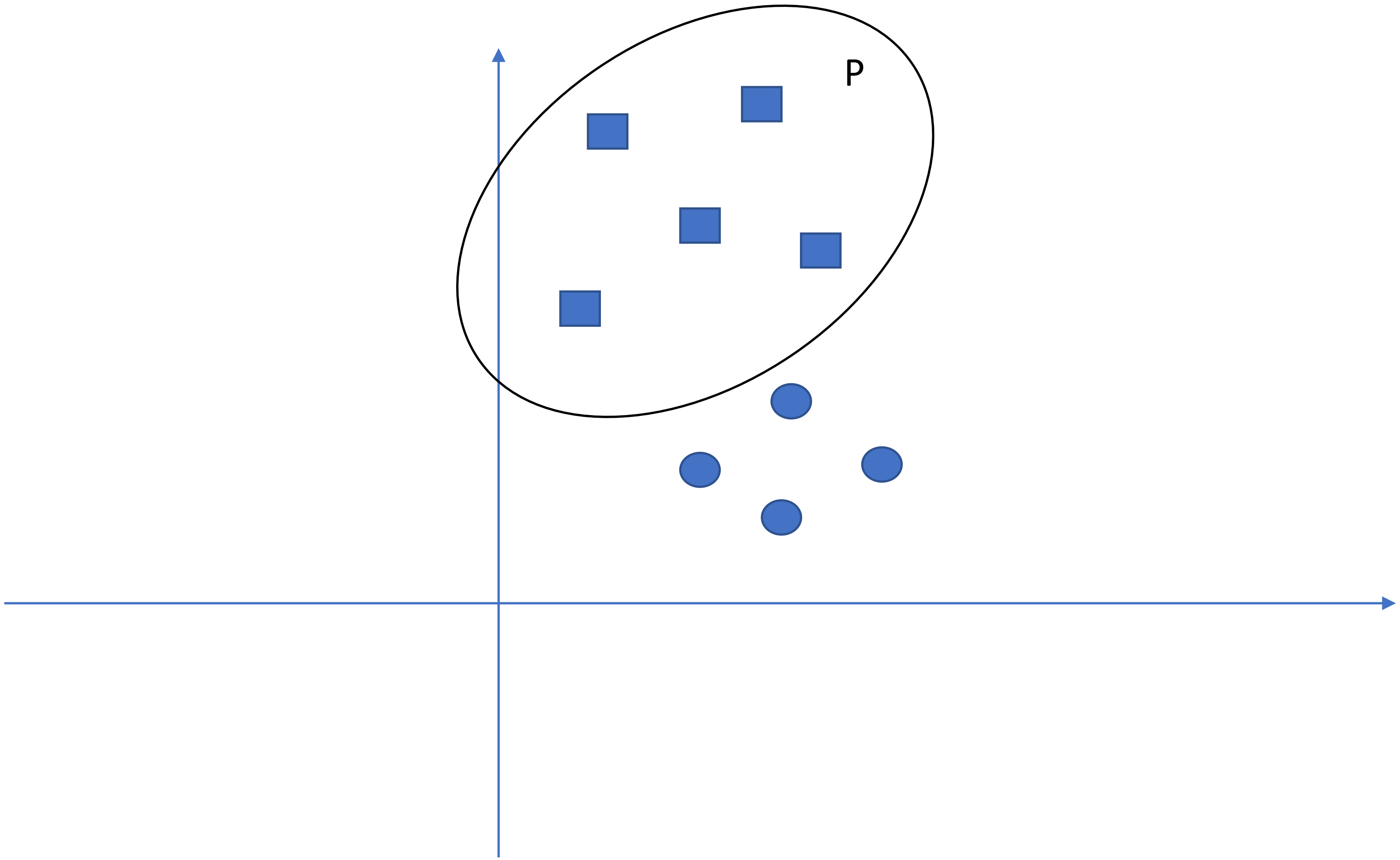
  - else

    $$w(i+1) \leftarrow w(i) - x$$

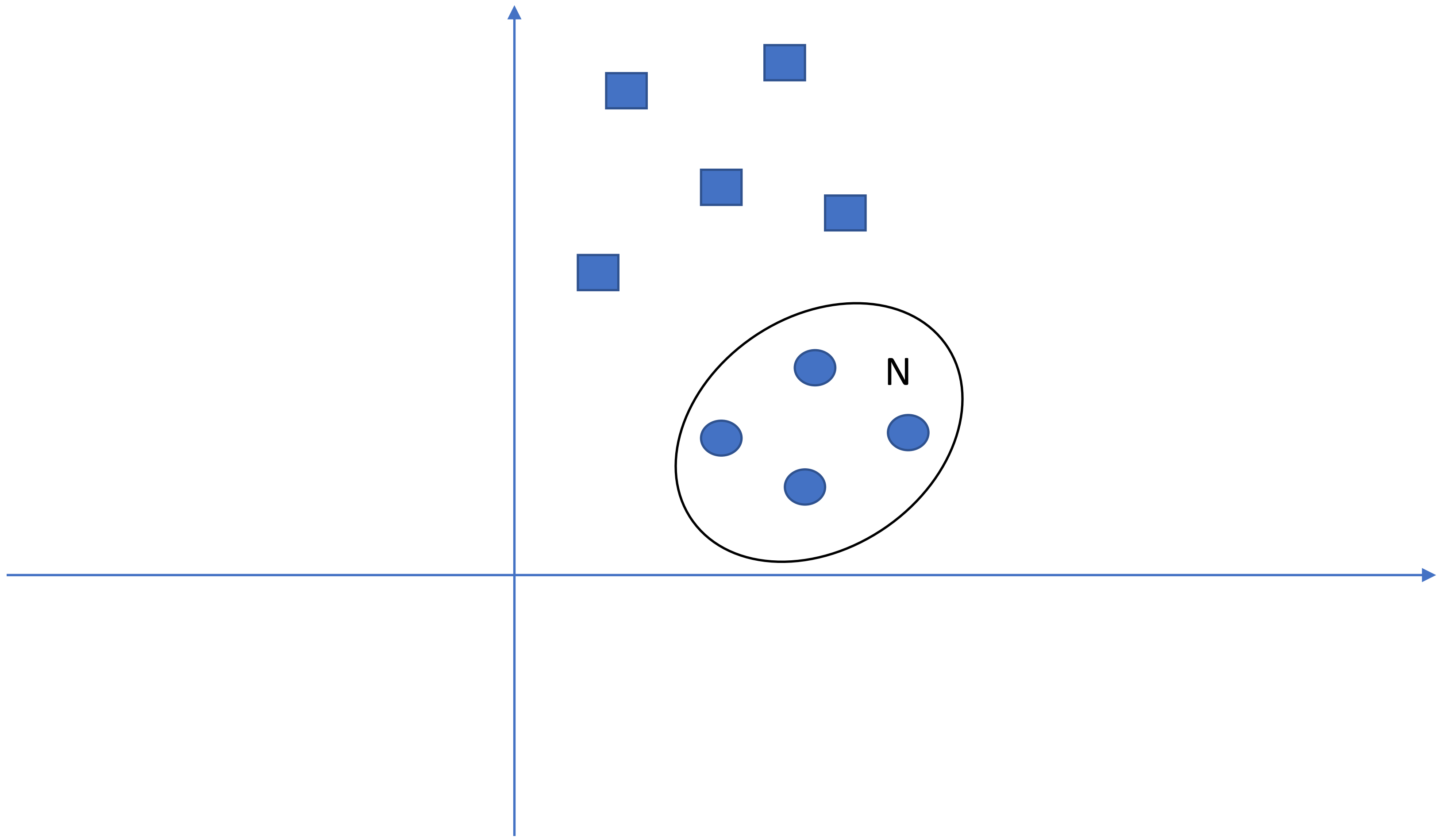
Rosenblatt Learning Rule



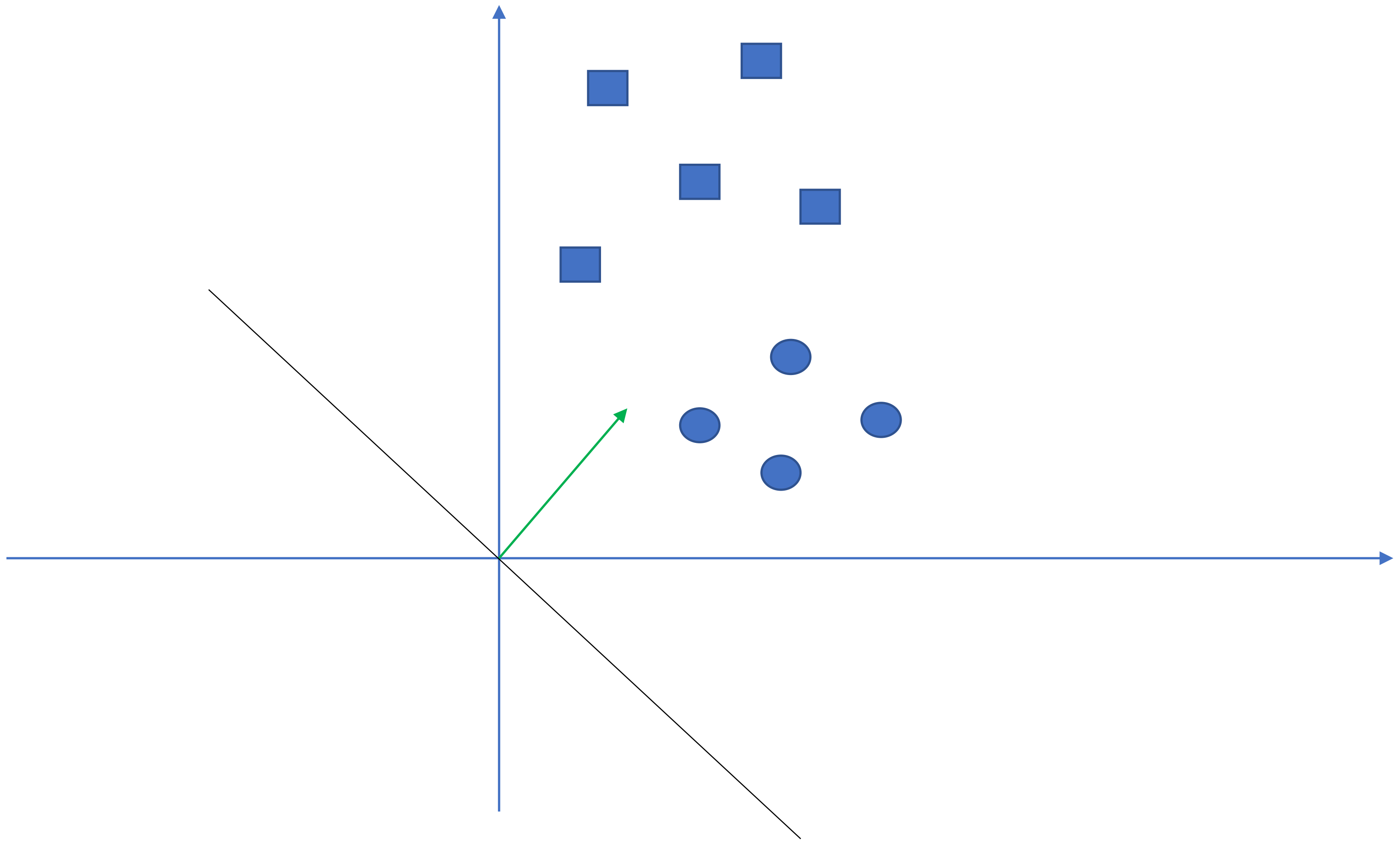
Rosenblatt Learning Rule



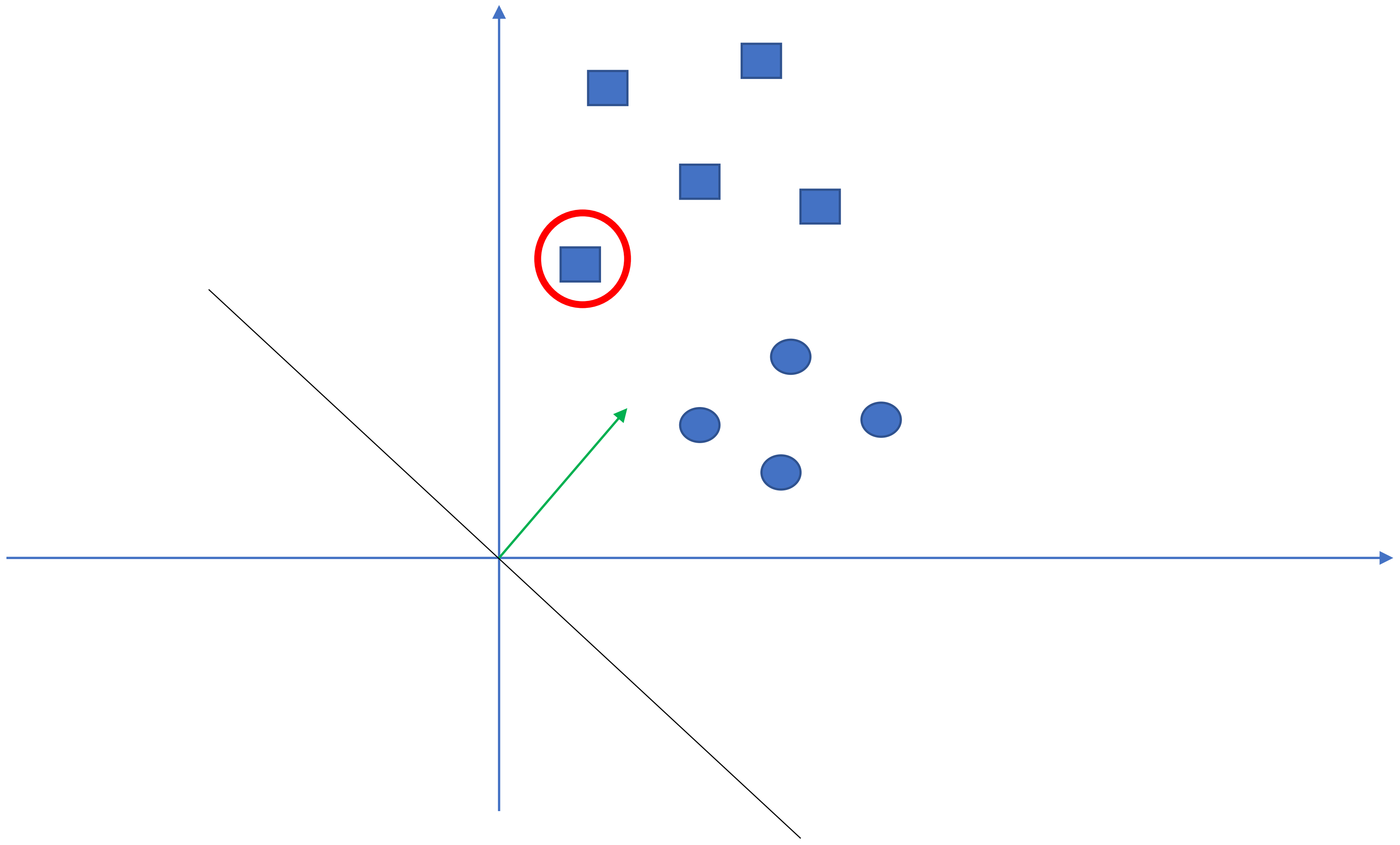
Rosenblatt Learning Rule



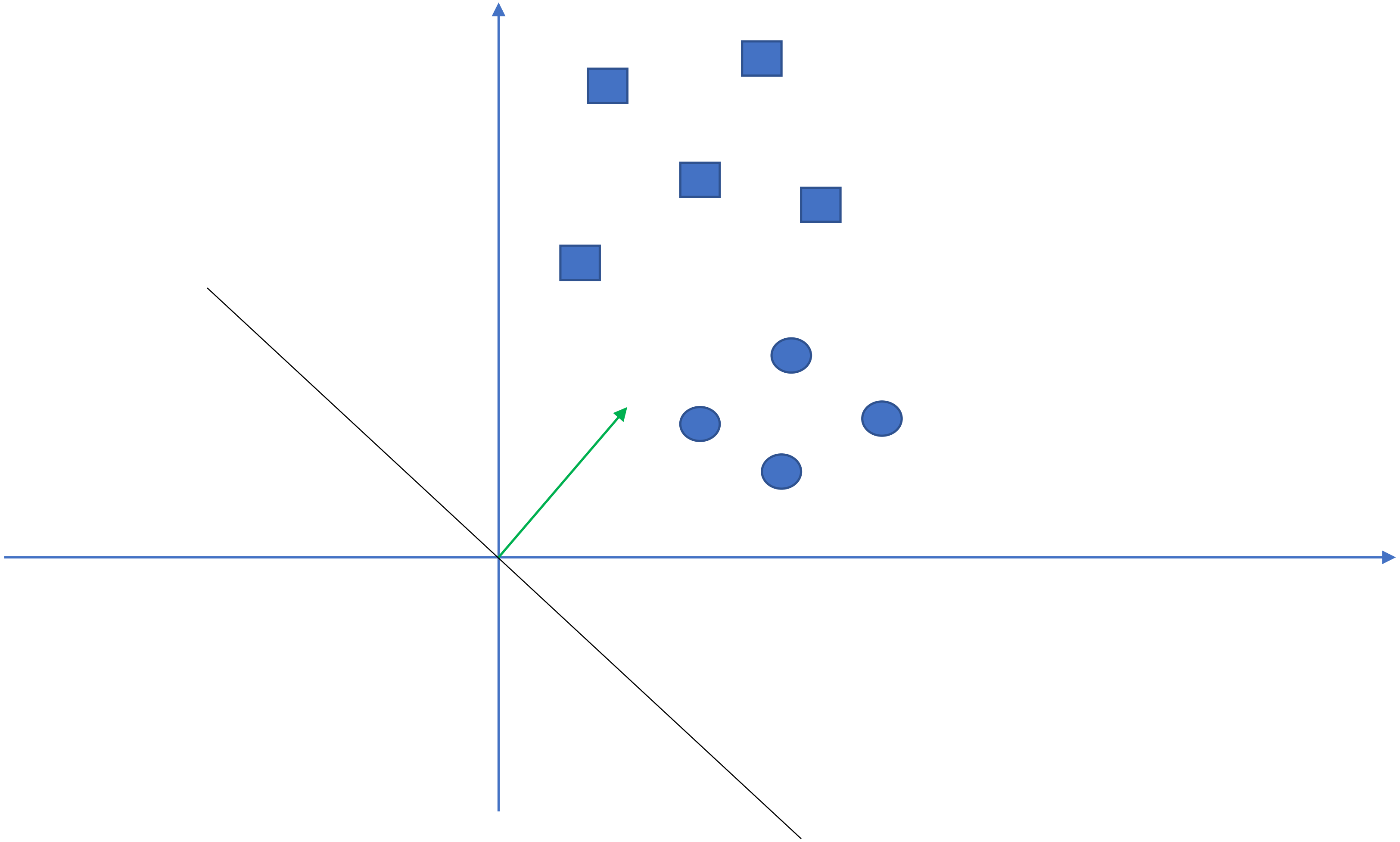
Rosenblatt Learning Rule



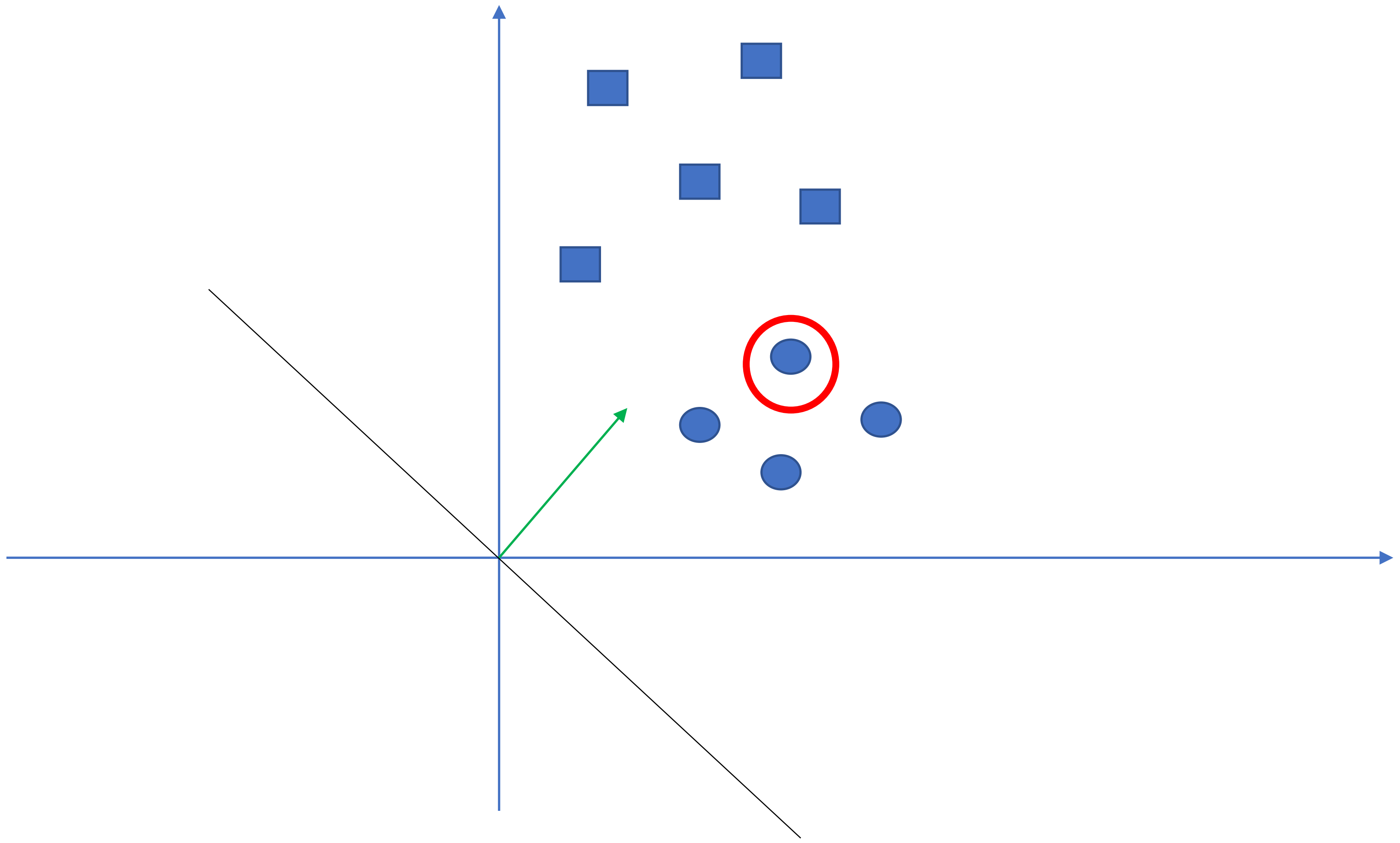
Rosenblatt Learning Rule



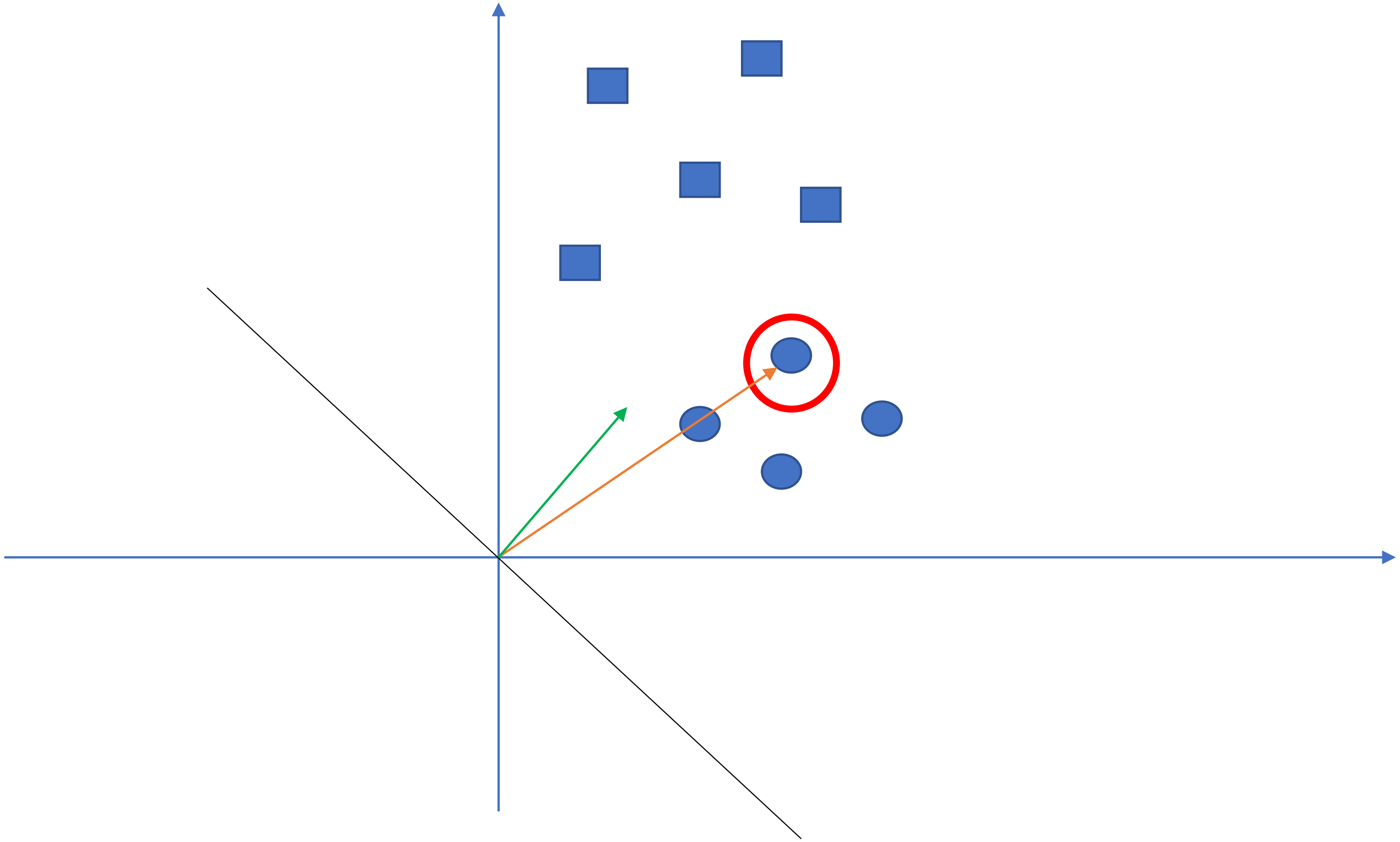
Rosenblatt Learning Rule



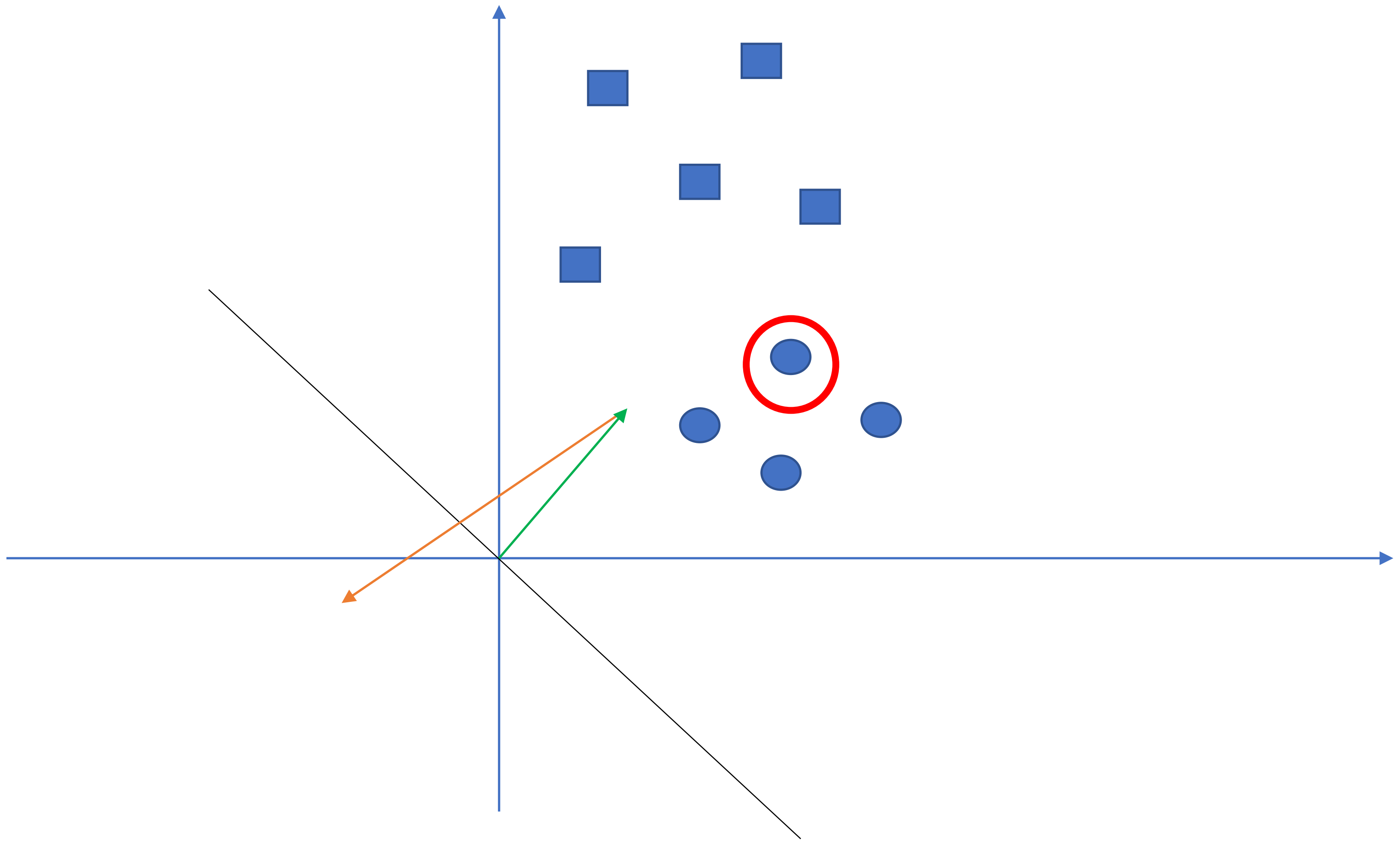
Rosenblatt Learning Rule



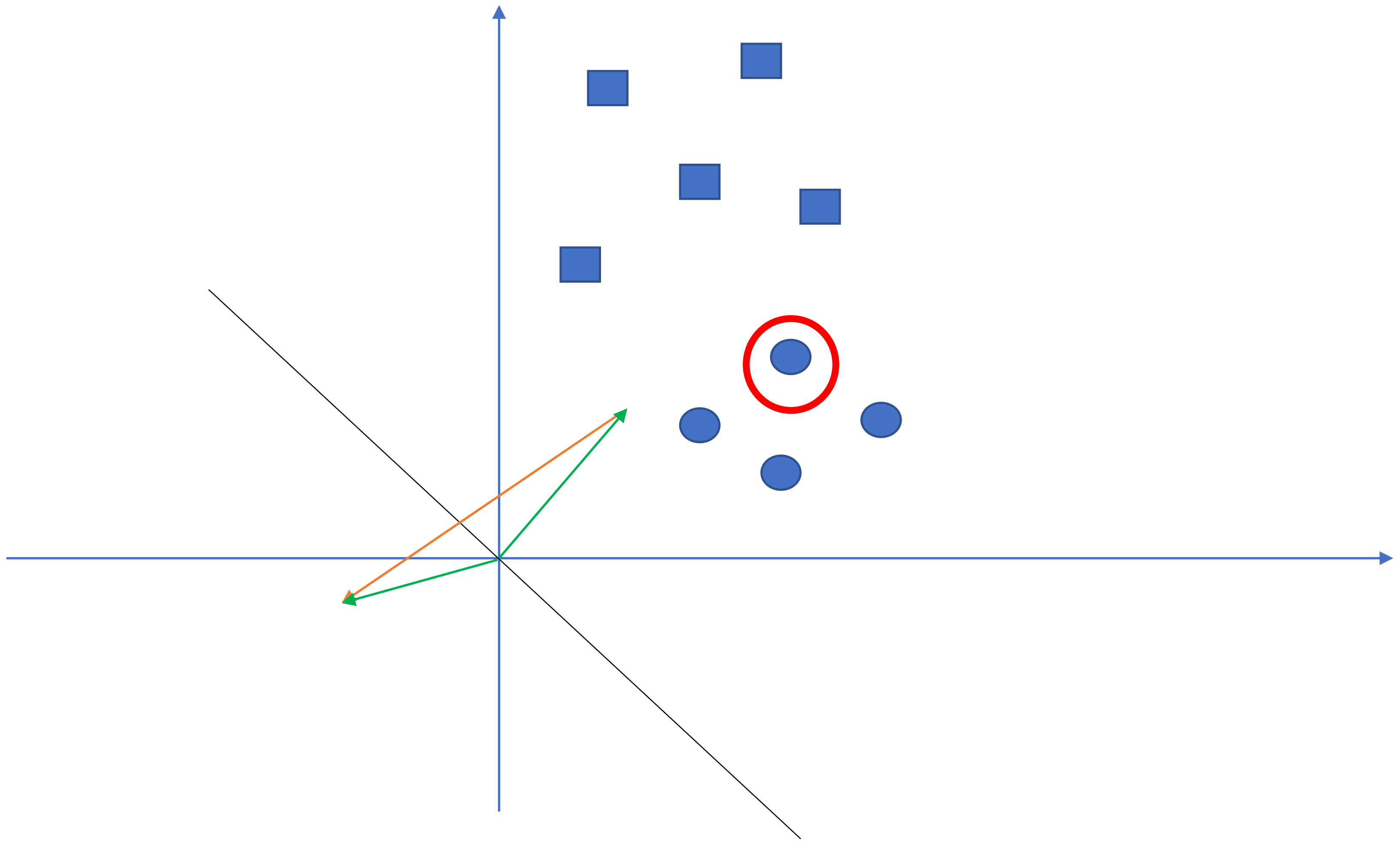
Rosenblatt Learning Rule



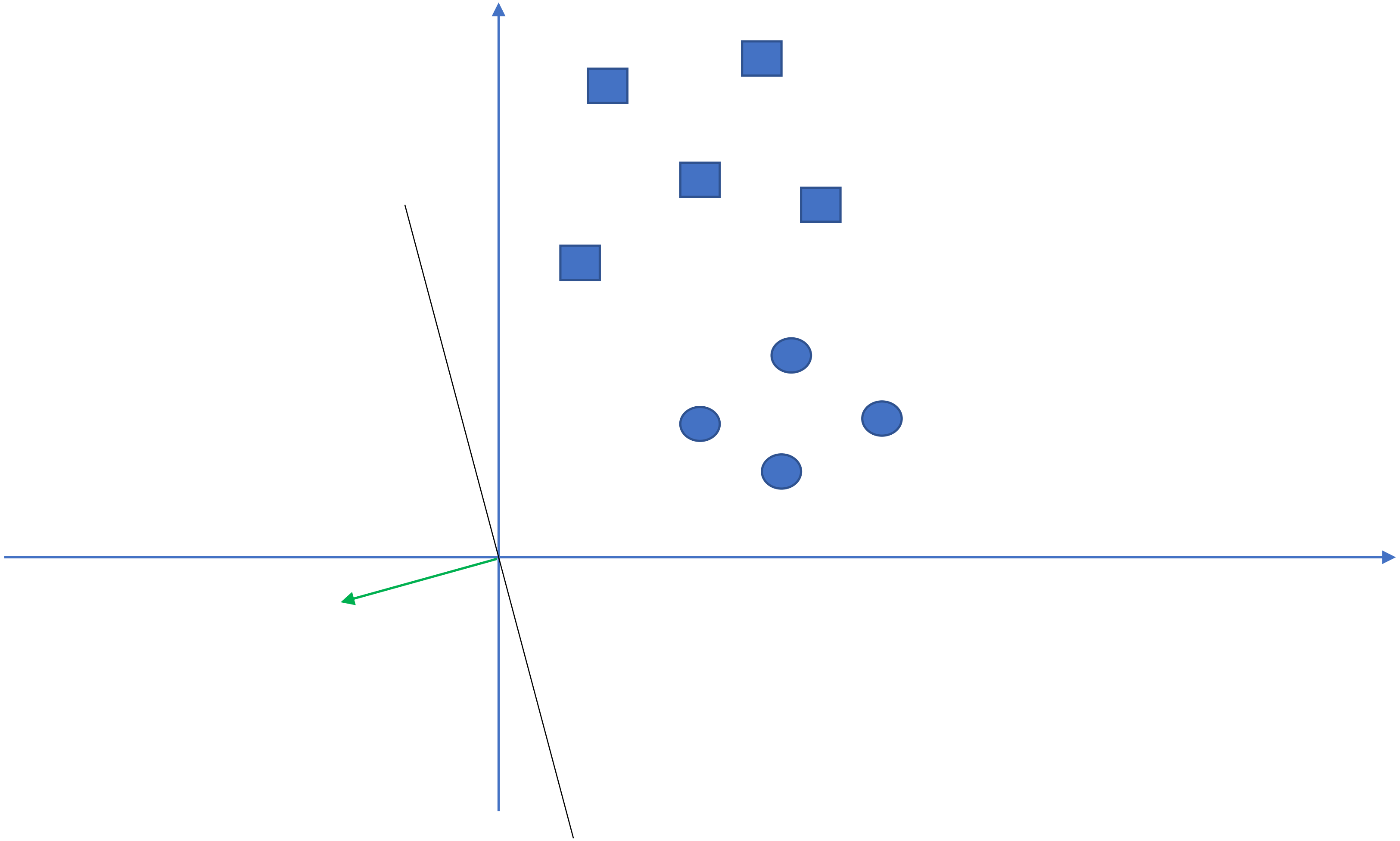
Rosenblatt Learning Rule



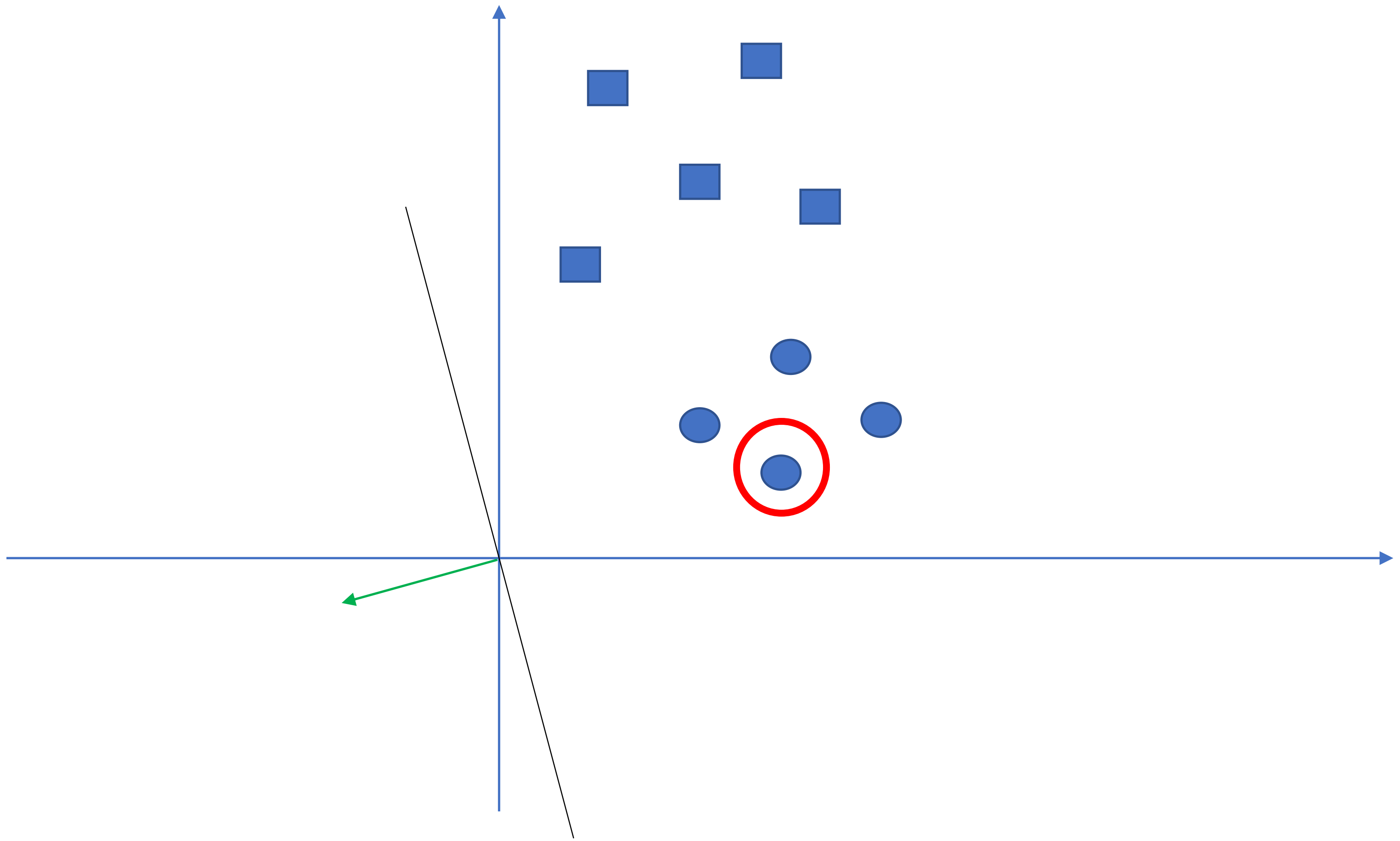
Rosenblatt Learning Rule



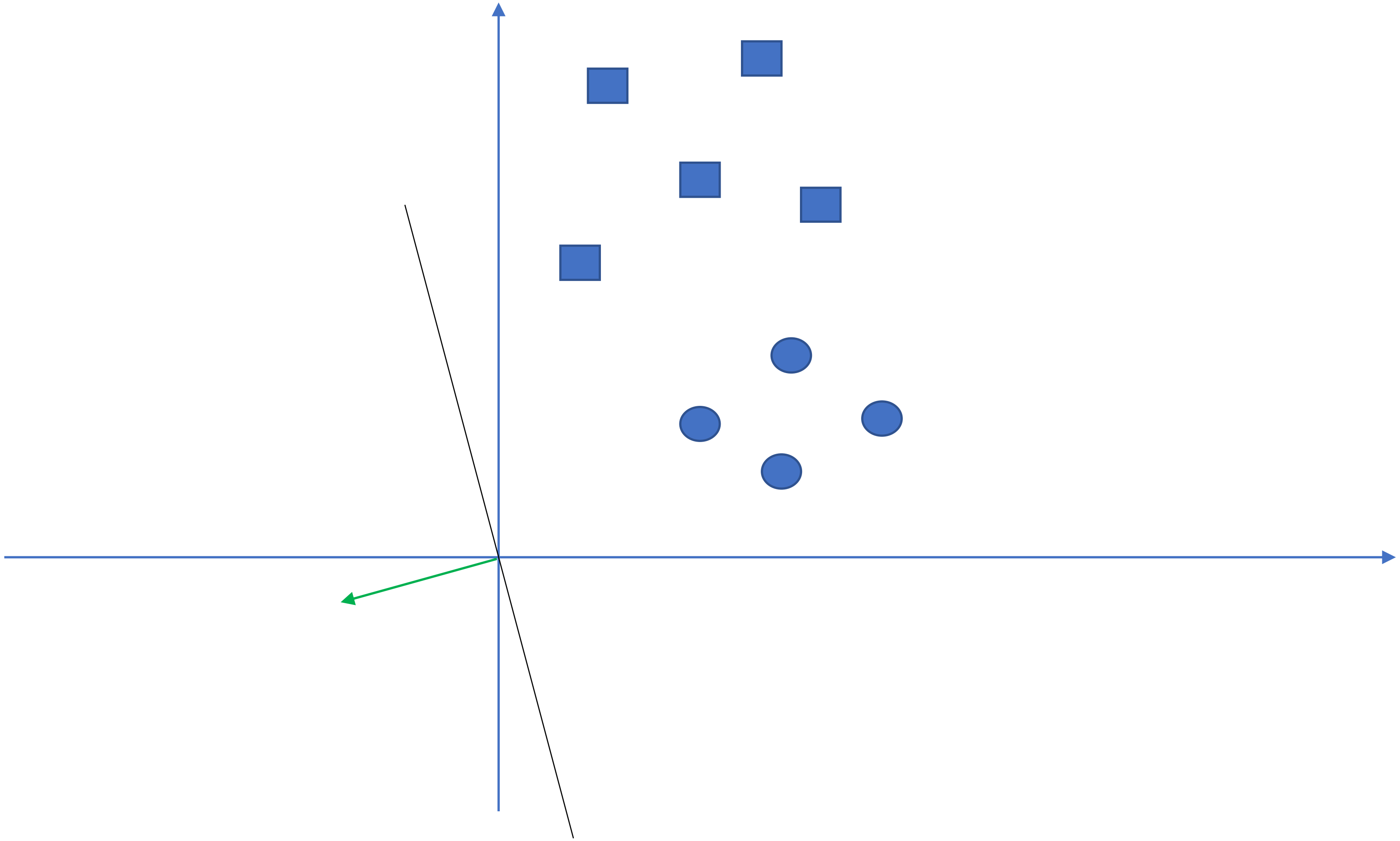
Rosenblatt Learning Rule



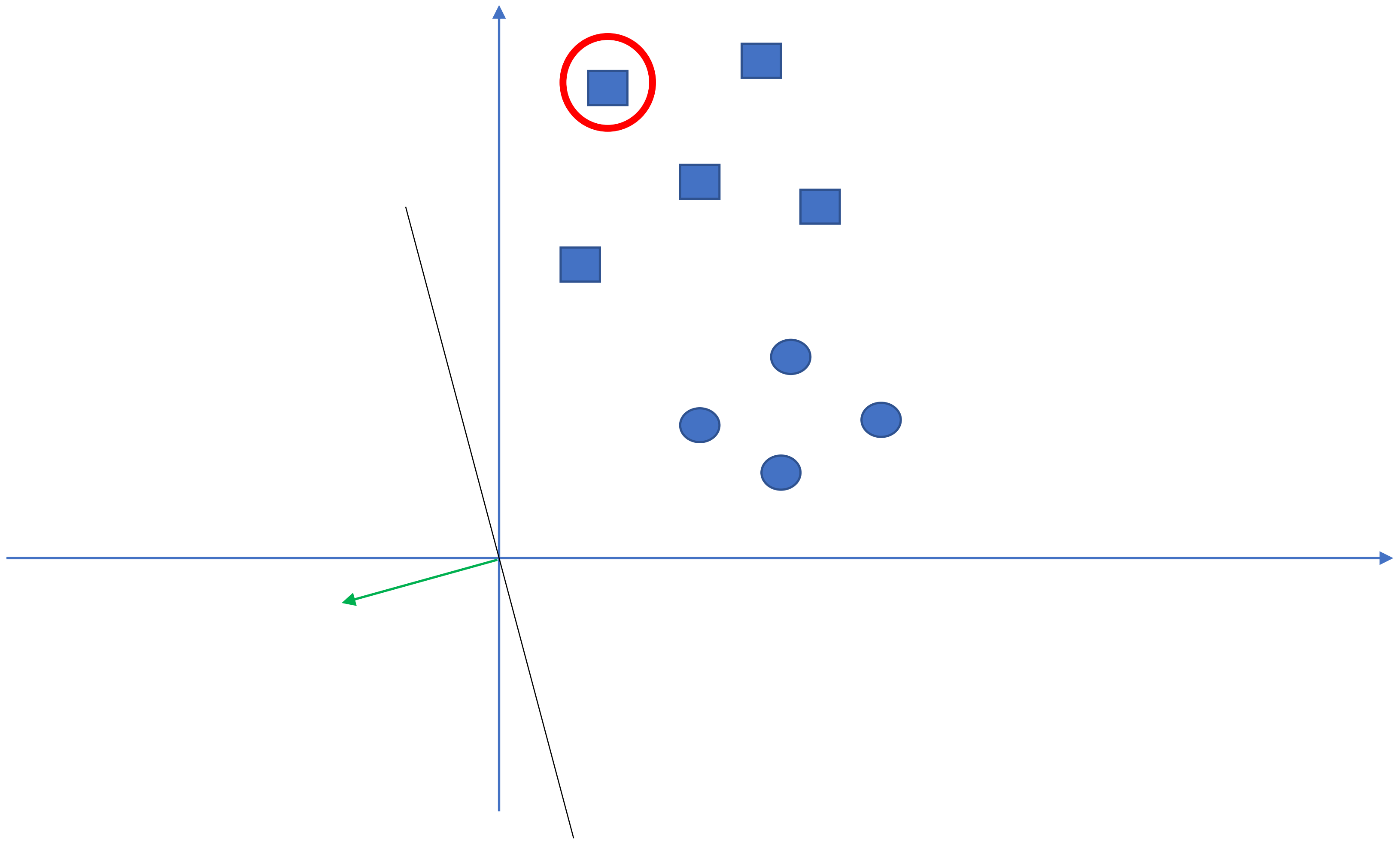
Rosenblatt Learning Rule



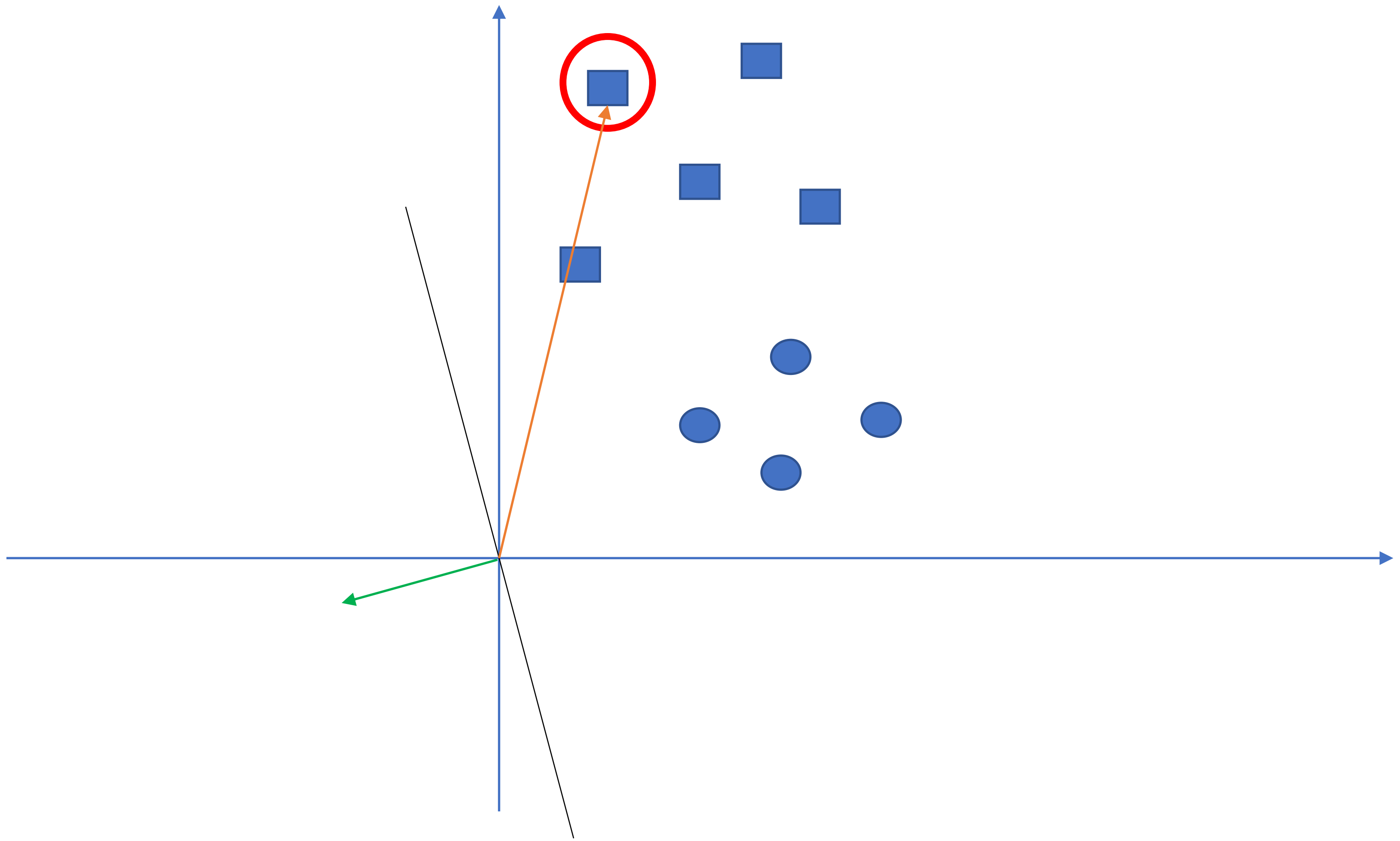
Rosenblatt Learning Rule



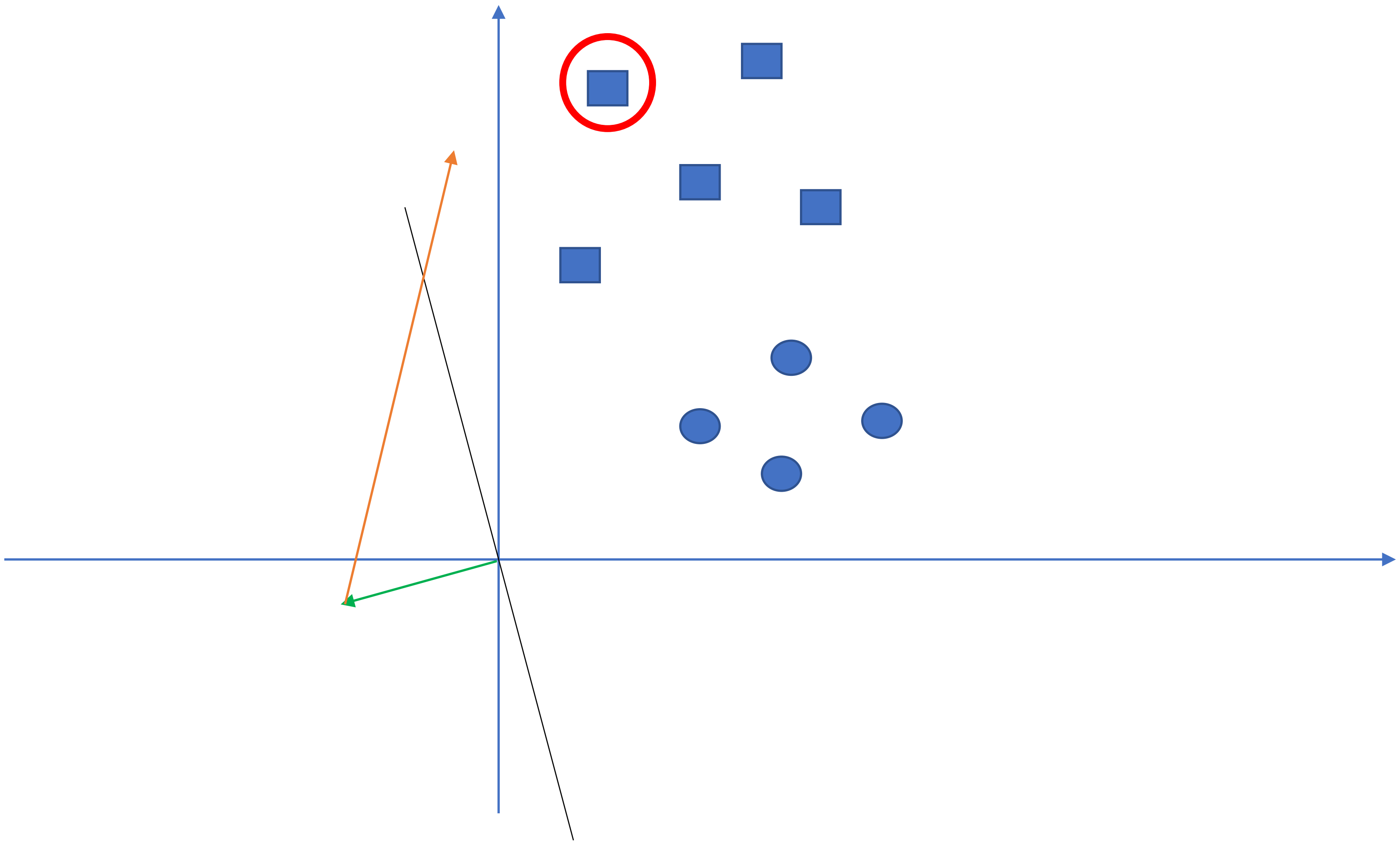
Rosenblatt Learning Rule



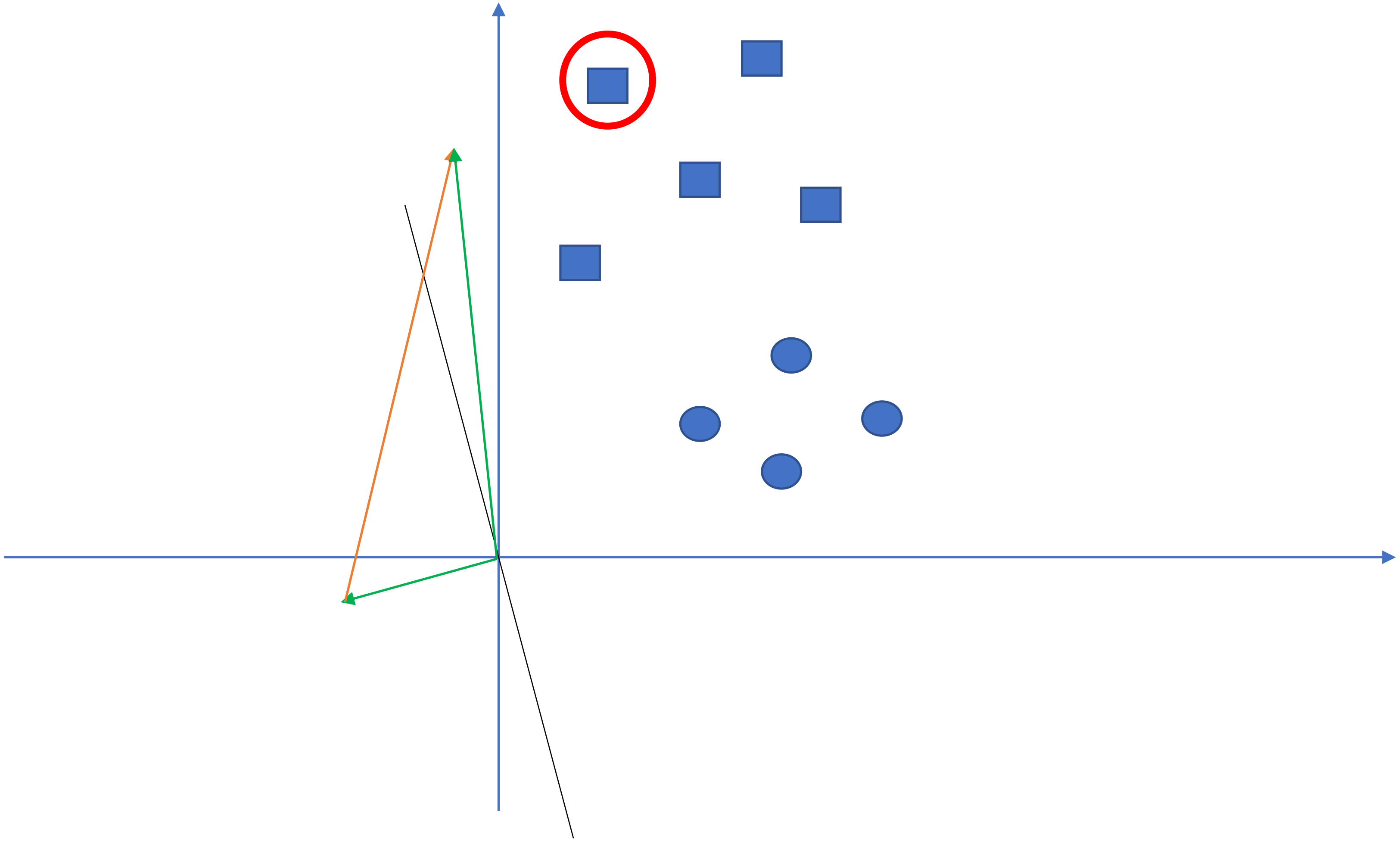
Rosenblatt Learning Rule



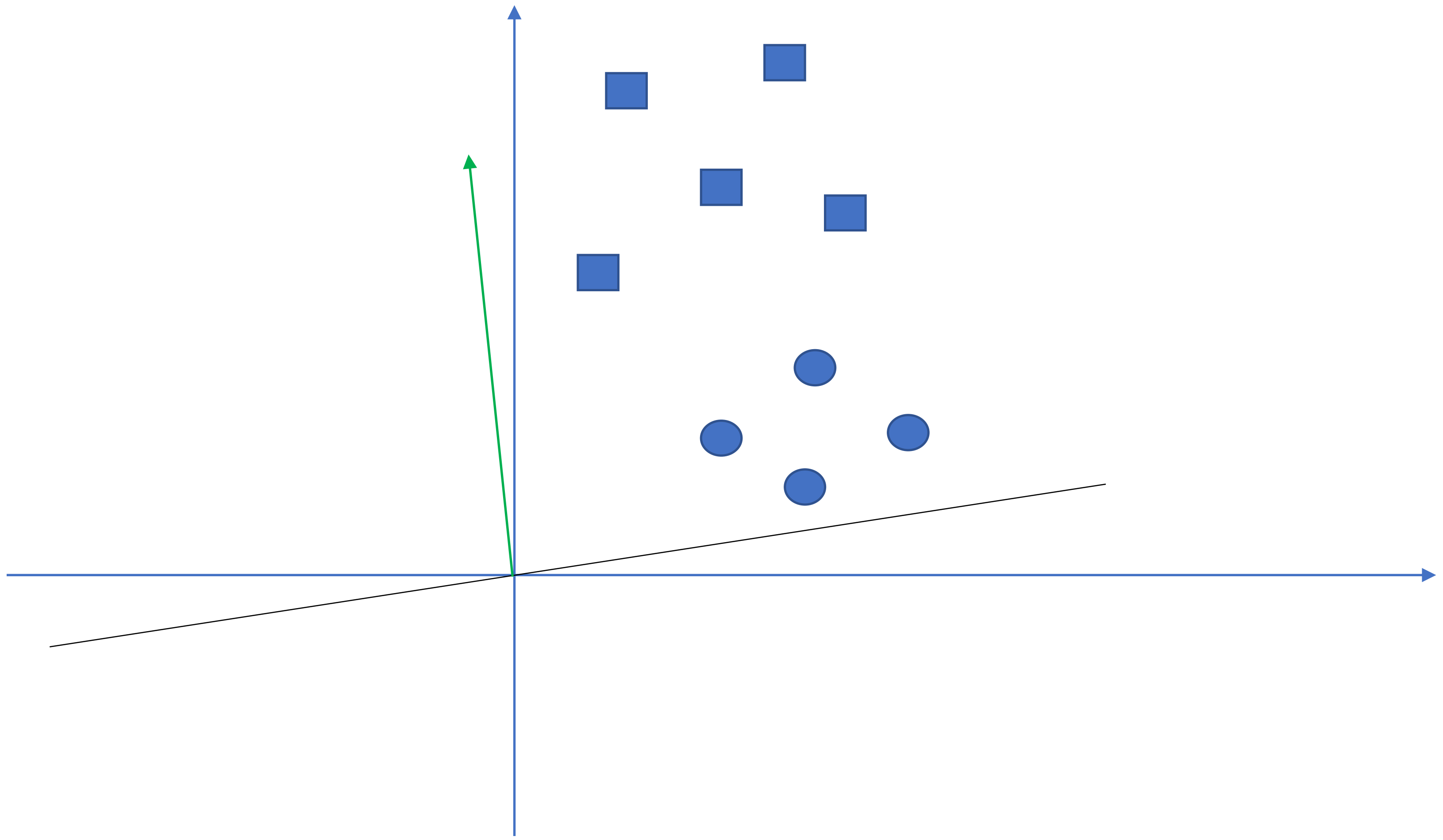
Rosenblatt Learning Rule



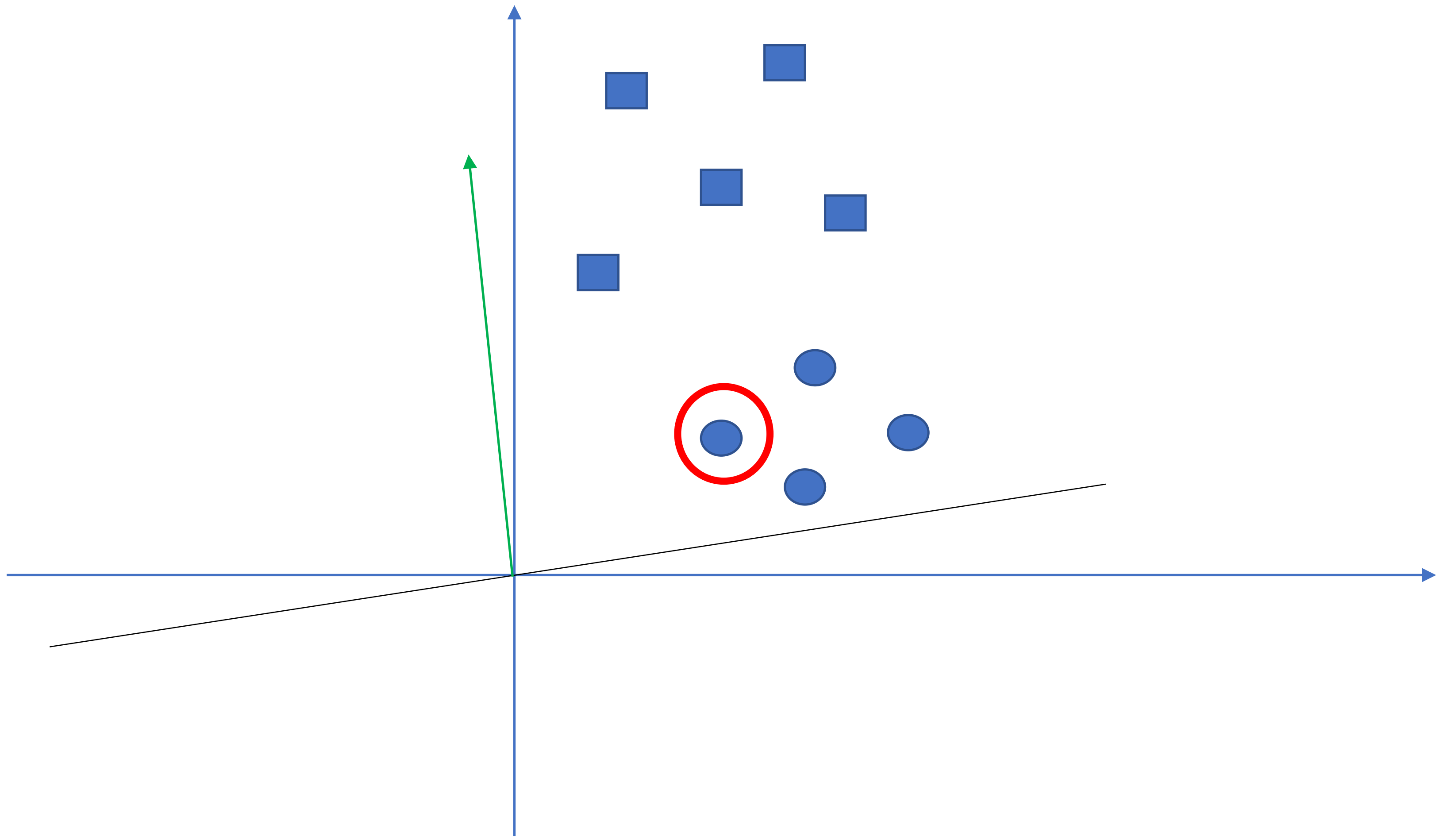
Rosenblatt Learning Rule



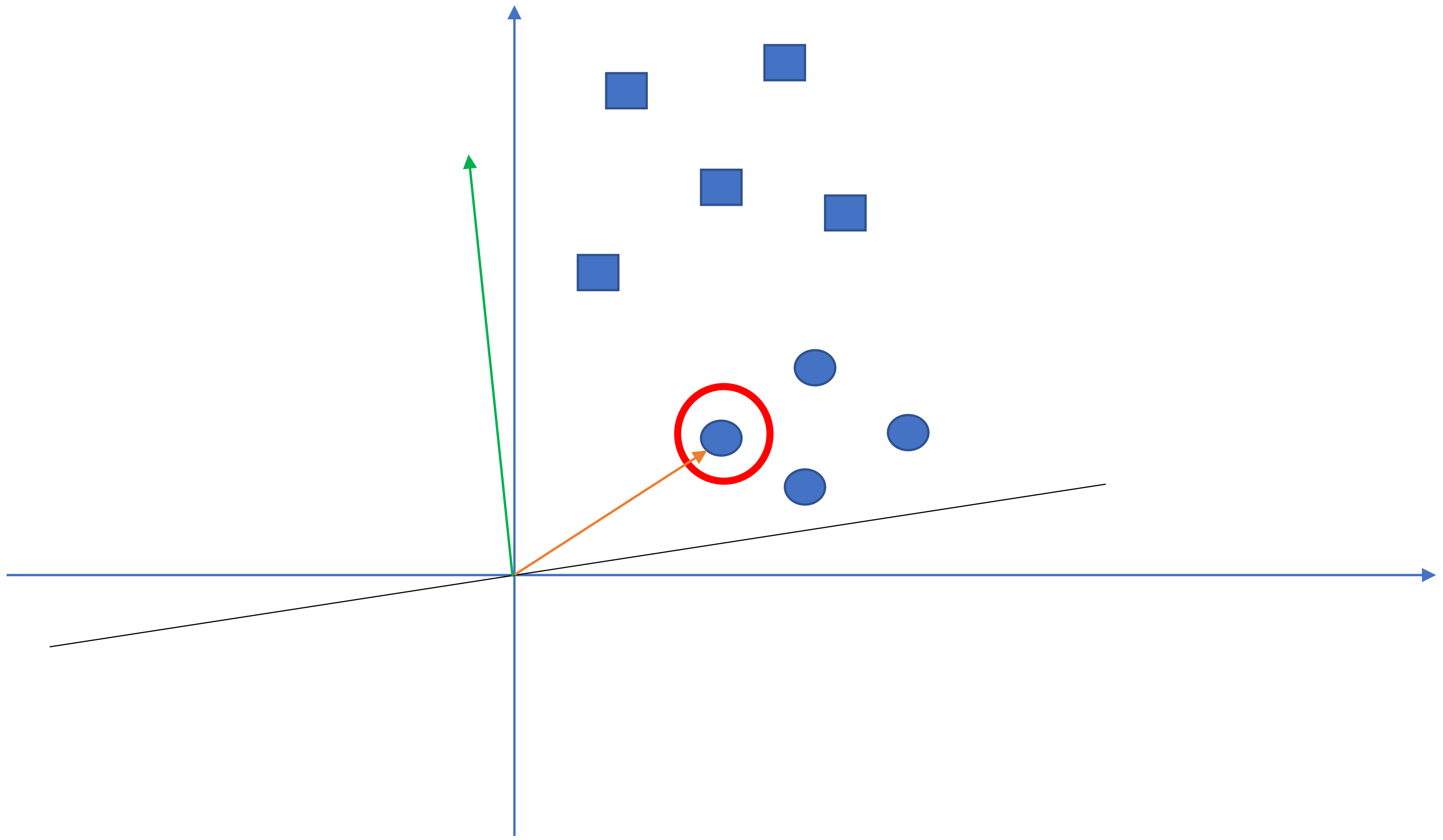
Rosenblatt Learning Rule



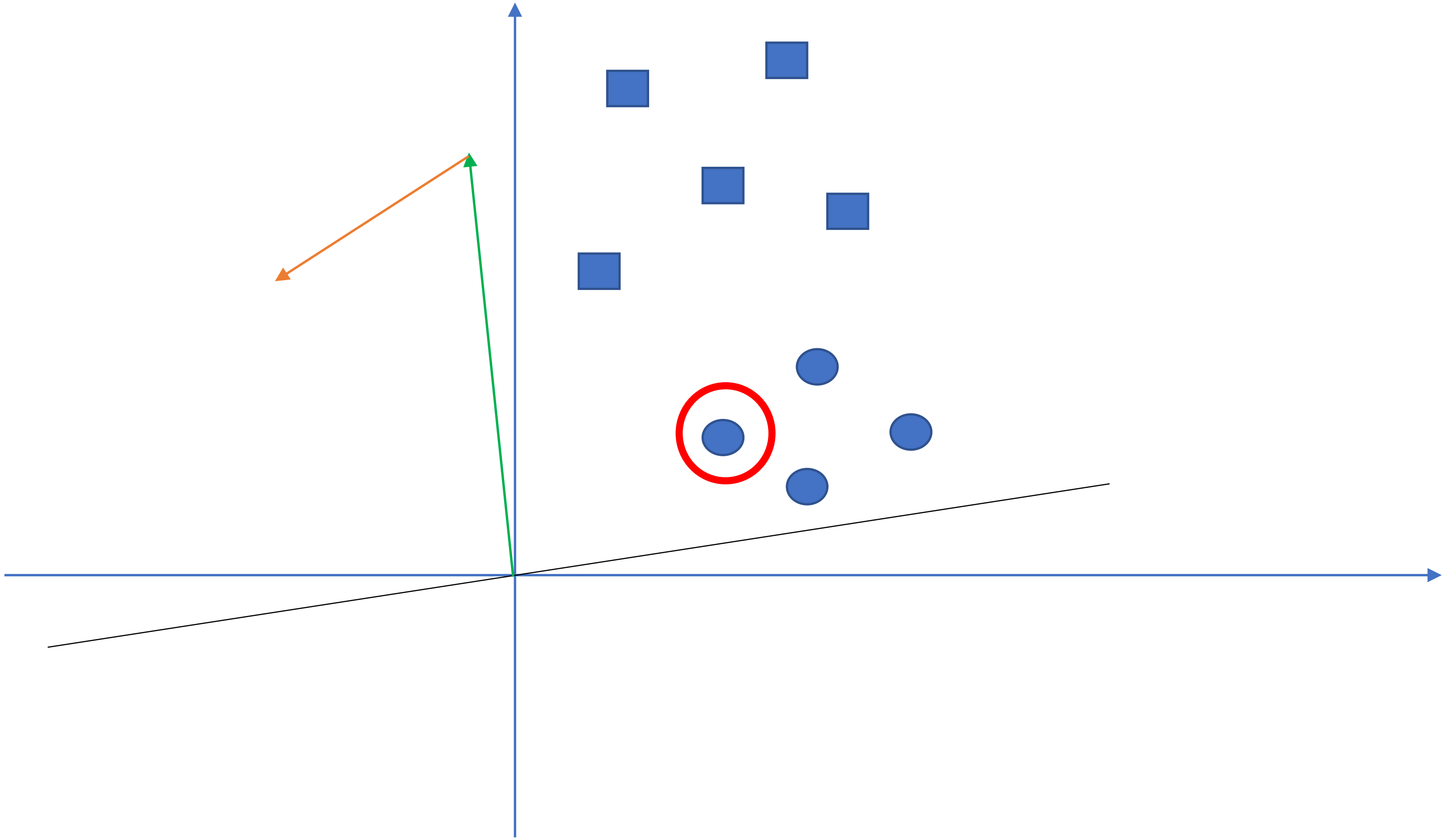
Rosenblatt Learning Rule



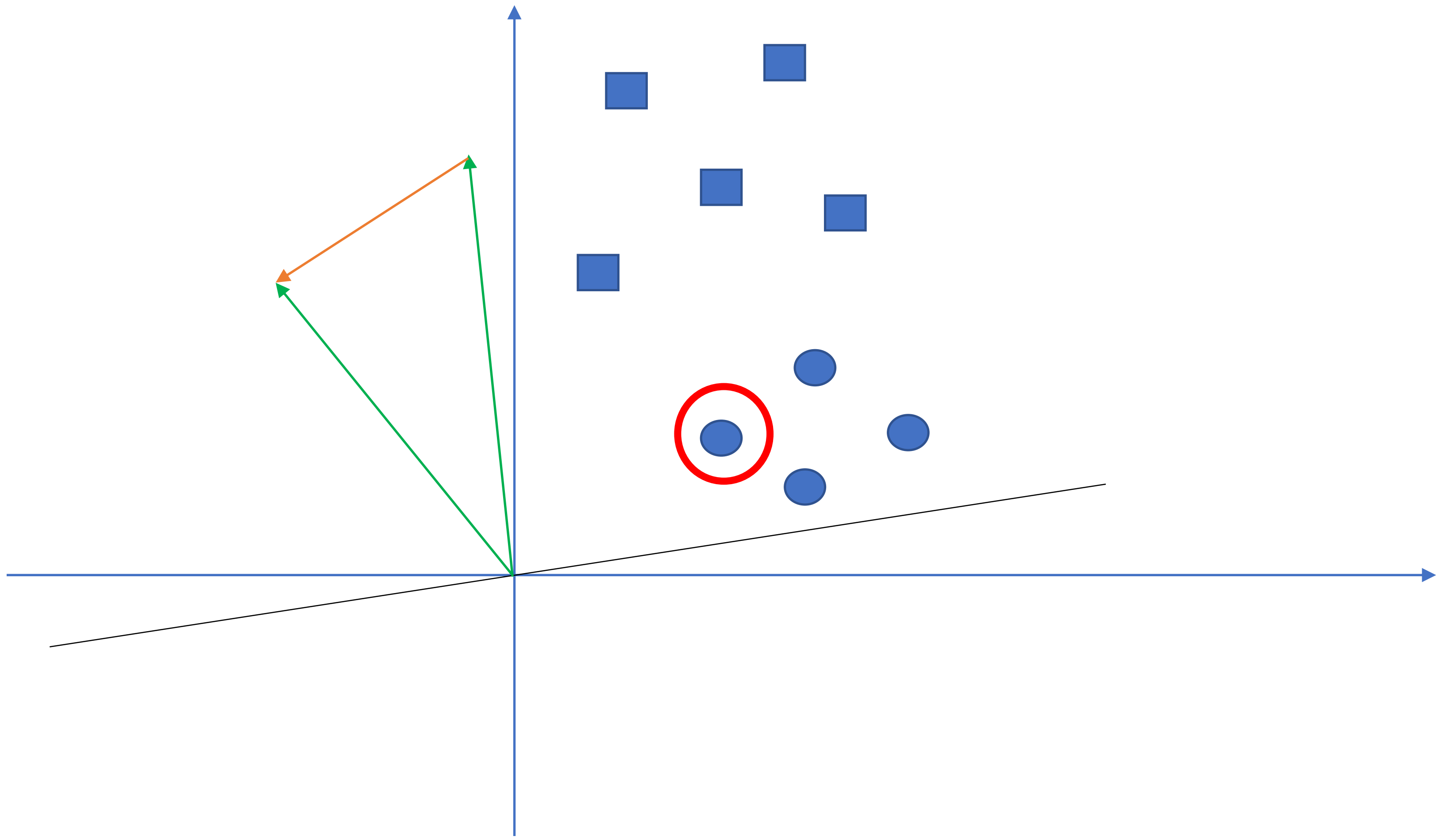
Rosenblatt Learning Rule



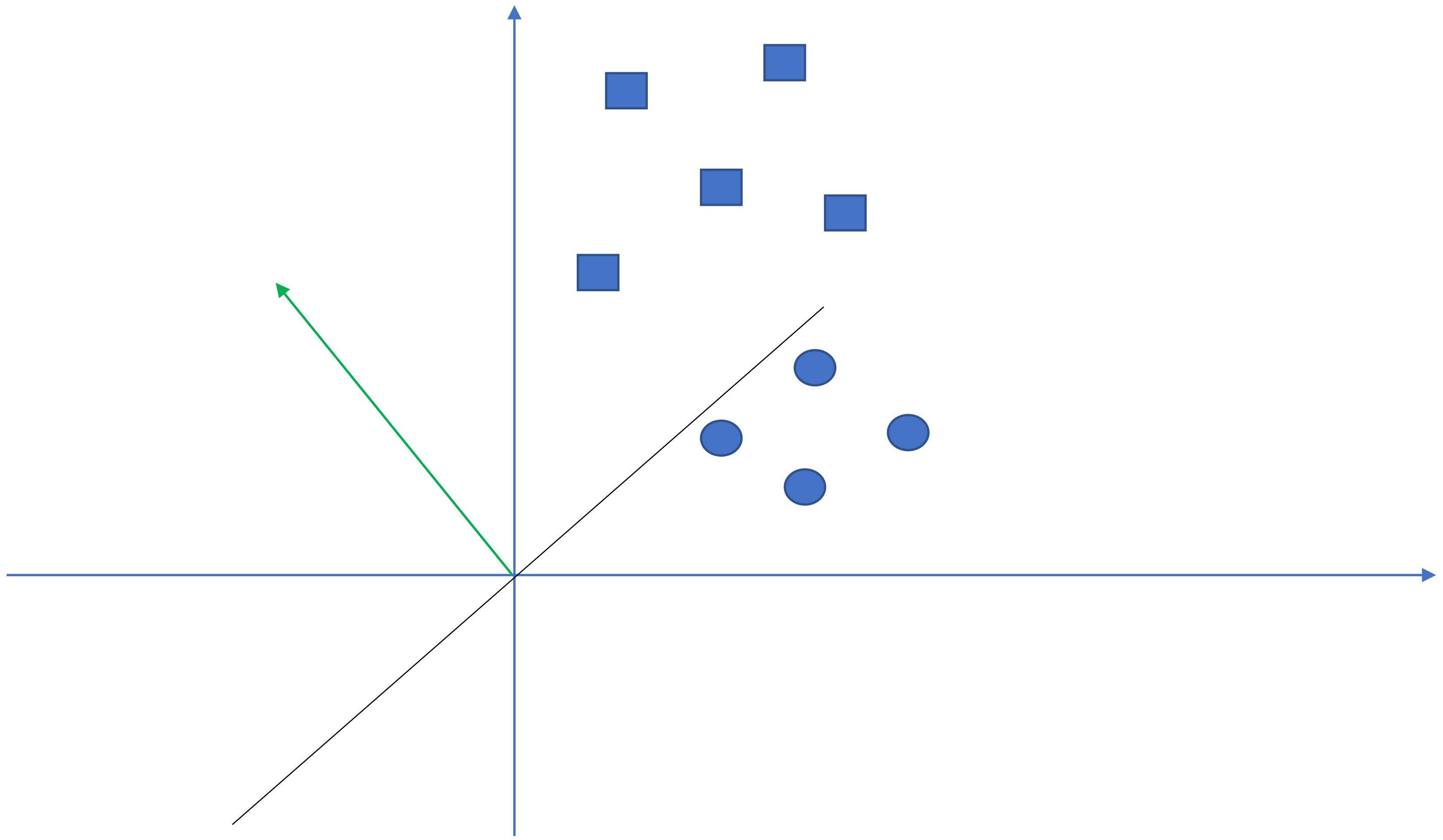
Rosenblatt Learning Rule



Rosenblatt Learning Rule



Rosenblatt Learning Rule



Revision: Lecture

- Under which condition(s) does the Perceptron Learning Rule terminate?

Revision: Lecture

- Under which condition(s) does the Perceptron Learning Rule terminate?
  1. The alg always terminates
  2. Both sets need to be absolutely linearly separable
  3. The alg never terminates
  4. Both sets need to be linearly separable

Revision: Lecture

- Under which condition(s) does the Perceptron Learning Rule terminate?
  1. The alg always terminates
  2. Both sets need to be absolutely linearly separable
  3. The alg never terminates
  4. Both sets need to be linearly separable

A: 1

B: 2

C: 3

D: 2,4

Revision: Lecture

- Under which condition(s) does the Perceptron Learning Rule terminate?
  1. The alg always terminates
  2. Both sets need to be absolutely linearly separable
  3. The alg never terminates
  4. Both sets need to be linearly separable

A: 1

B: 2

C: 3

D: 2,4

Revision: Lecture

- What is the *equivalent* Learning Problem?

Revision: Lecture

- What is the *equivalent* Learning Problem?
  1. Points from N are all negated
  2. Points from N are all point reflected though the origin
  3. Try to find hyperplane such that all (transformed) points are on same side
  4. Try to find hyperplane that separates transformed points from original points

Revision: Lecture

- What is the *equivalent* Learning Problem?
  1. Points from N are all negated
  2. Points from N are all point reflected though the origin
  3. Try to find hyperplane such that all (transformed) points are on same side
  4. Try to find hyperplane that separates transformed points from original points

A: 1,3

B: 1,2,3

C: 1,4

D: all

Revision: Lecture

- What is the *equivalent* Learning Problem?
    1. Points from N are all negated
    2. Points from N are all point reflected though the origin
    3. Try to find hyperplane such that all (transformed) points are on same side
    4. Try to find hyperplane that separates transformed points from original points

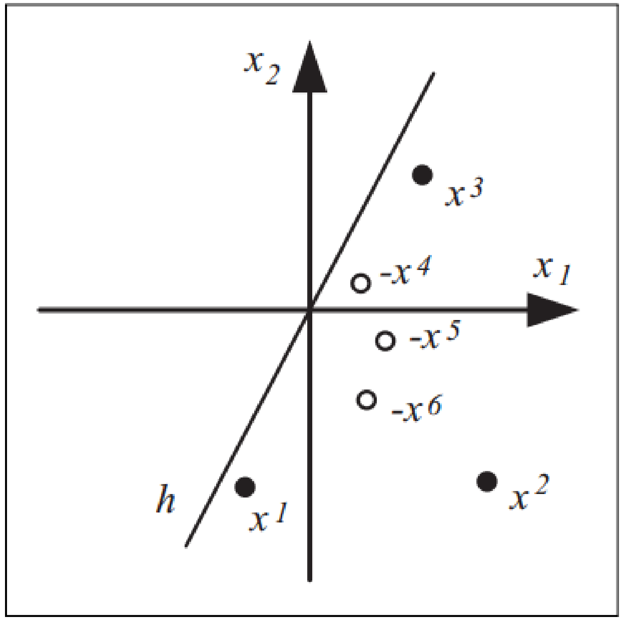
A: 1,3

B: 1,2,3

C: 1,4

D: all

Revision: Lecture



$\Omega' := \mathcal{P}' \cup \mathcal{N}'$ mit

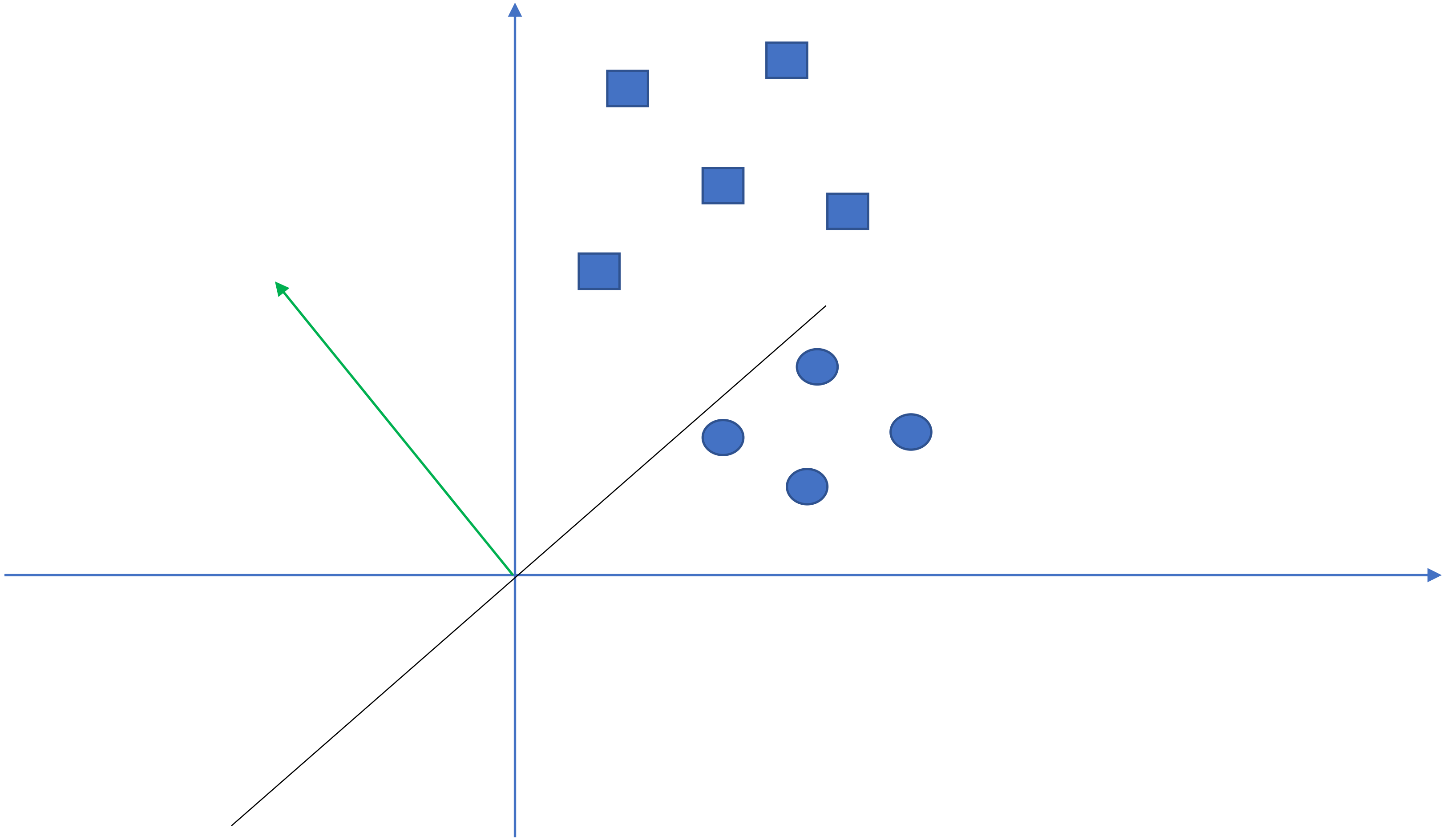
$\mathcal{P}' :=$
$\{\zeta^m = \boldsymbol{x}^m | \boldsymbol{x}^m \in \mathcal{P}\}$

und

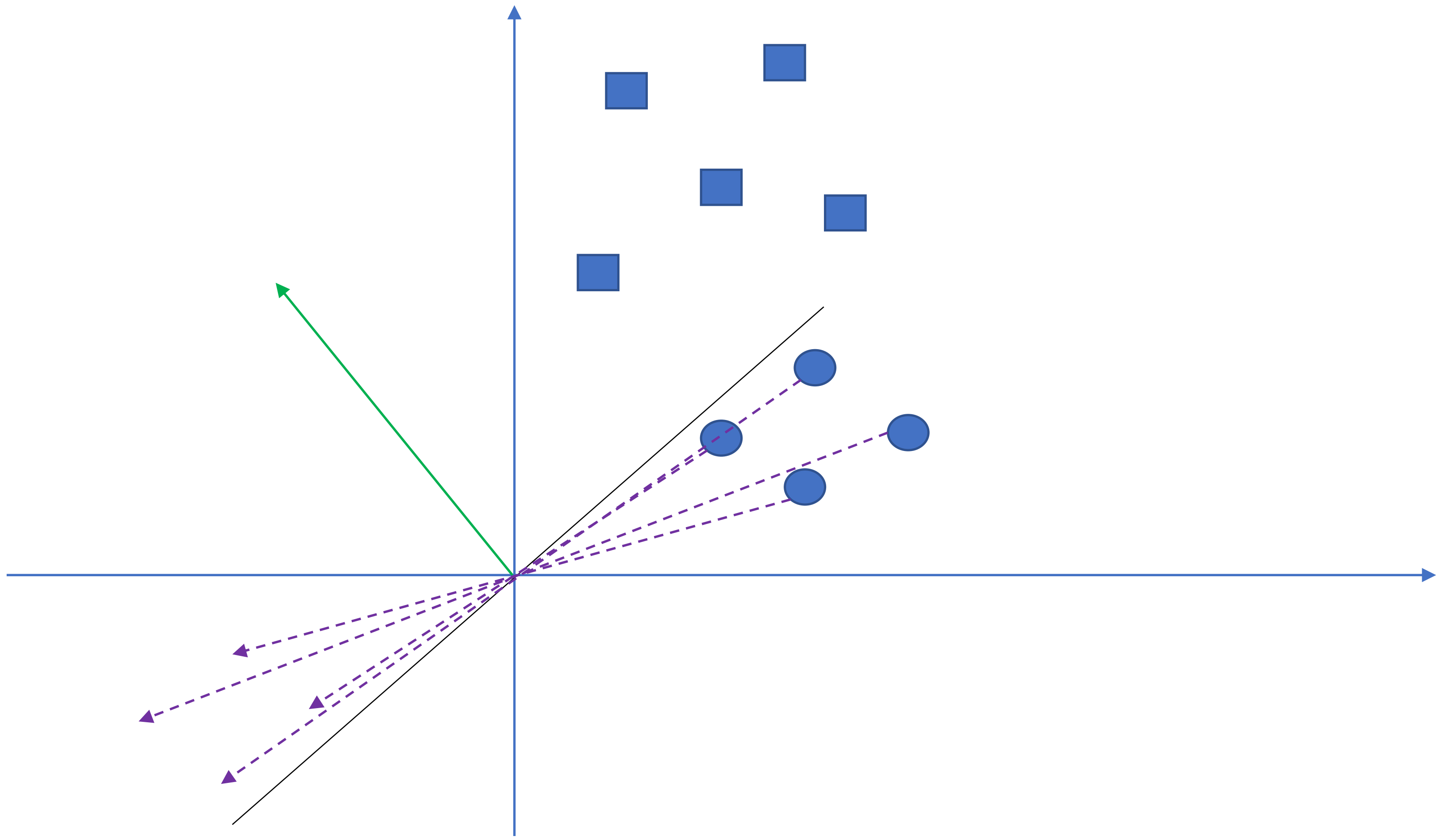
$\mathcal{N}' :=$
$\{\zeta^m = -\boldsymbol{x}^m | \boldsymbol{x}^m \in \mathcal{N}\}$

Äquivalentes Lernproblem: Finde Gewichtsvektor $\boldsymbol{w}$, so dass $\boldsymbol{w}^T\zeta > 0, \forall \zeta \in \Omega'$.

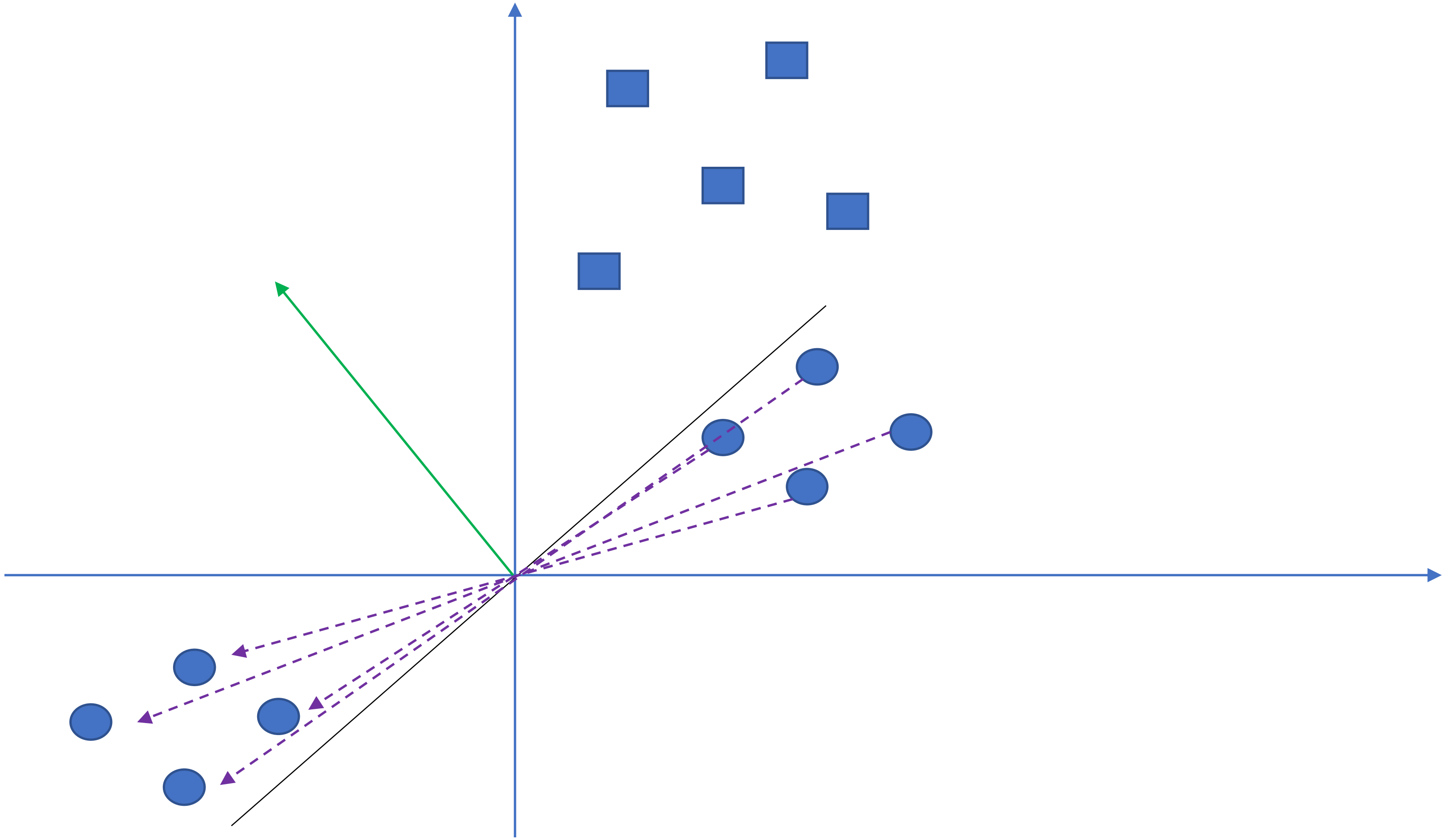
Revision: Lecture



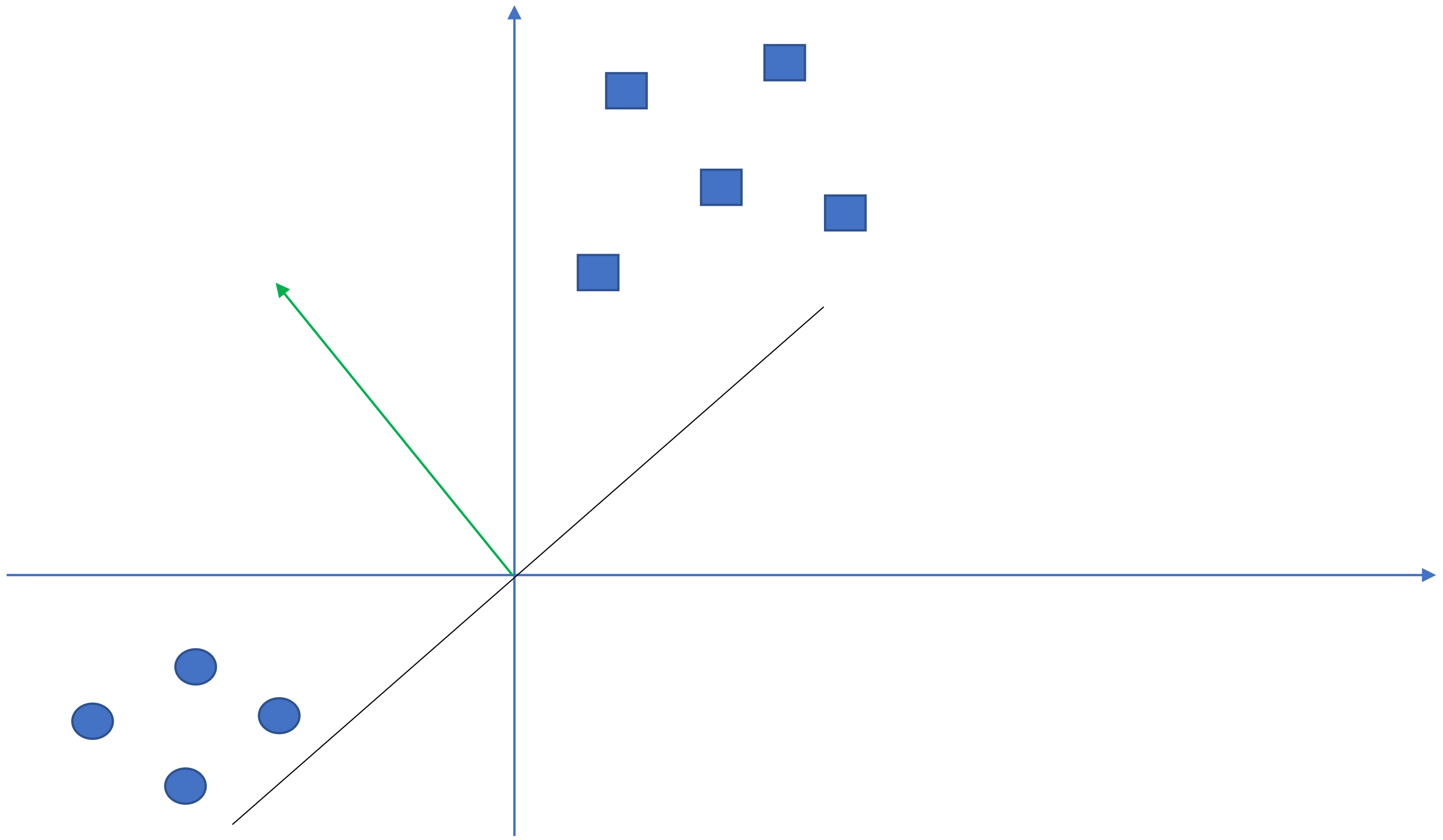
Revision: Lecture



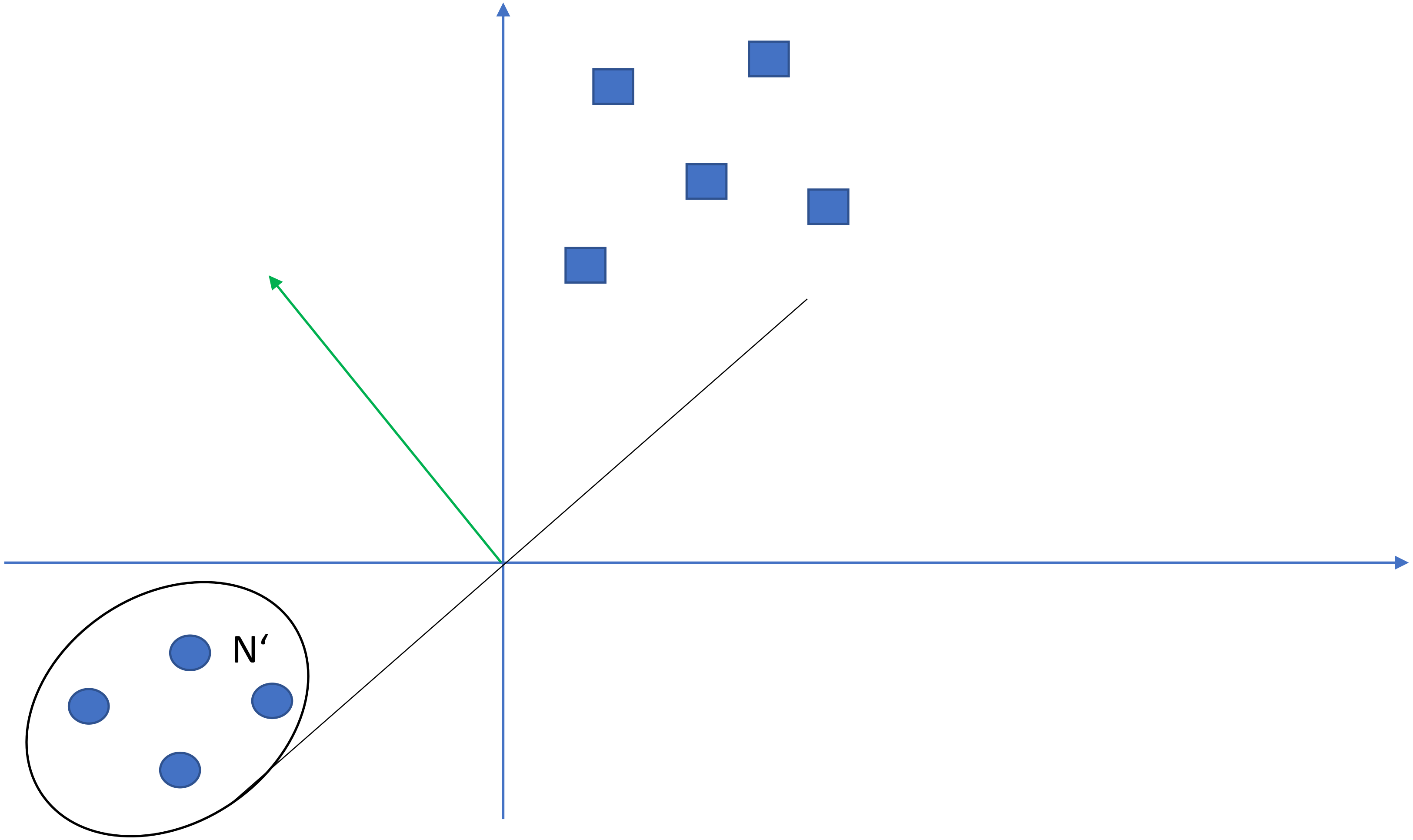
Revision: Lecture



Revision: Lecture



Revision: Lecture



Content

- Revision: Practical Task
- Revision: Lecture
- New Practical Task