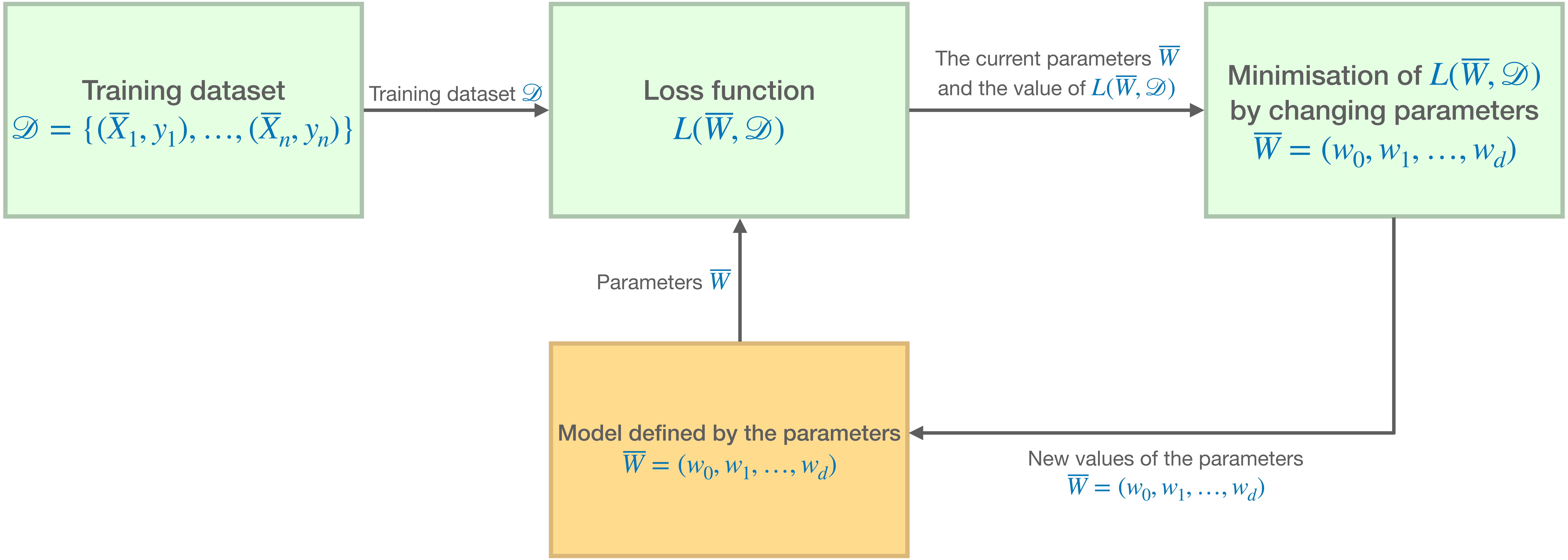
Perceptron

Loss Function Minimisation

Procheta Sen

UNIVERSITY OF LIVERPOOL

Loss function minimisation point of view



Perceptron: the training algorithm

`PerceptronTrain`(Training data: `D`, `MaxIter`)

$w_i = 0$ `for all` $i = 1, \ldots, d$;

$b = 0$

for `iter = 1 … MaxIter` **do**

    for `all` $(\overline{X}, y) \in D$ **do**

        $a = \overline{W}^T \overline{X} + b$

        if $y \cdot a \leq 0$ **then**

            $w_i = w_i + y \cdot x_i$, `for all` $i = 1, \ldots, d$

            $b = b + y$

return $b, w_1, w_2, \ldots, w_d$

Loss function: step function

Let $\mathscr{D} = \{(\overline{X}_1, y_1), \ldots, (\overline{X}_n, y_n)\}$ be the training dataset, where $\overline{X}_k = (x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(d)})^T$ for every $k = 1,\ldots,n$.

Let $a_k = b + \sum_{i=1}^{d} w_i x_k^{(i)}$.

Define **loss function** on a single training object $(\overline{X}_k, y_k)$ as

$L(b, \overline{W}, \overline{X}_k, y_k) = 1$ if $\overline{X}_k$ misclassified and $L(b, \overline{W}, \overline{X}_k, y_k) = 0$, otherwise.

Define **loss function** for the training dataset $\mathscr{D}$ as

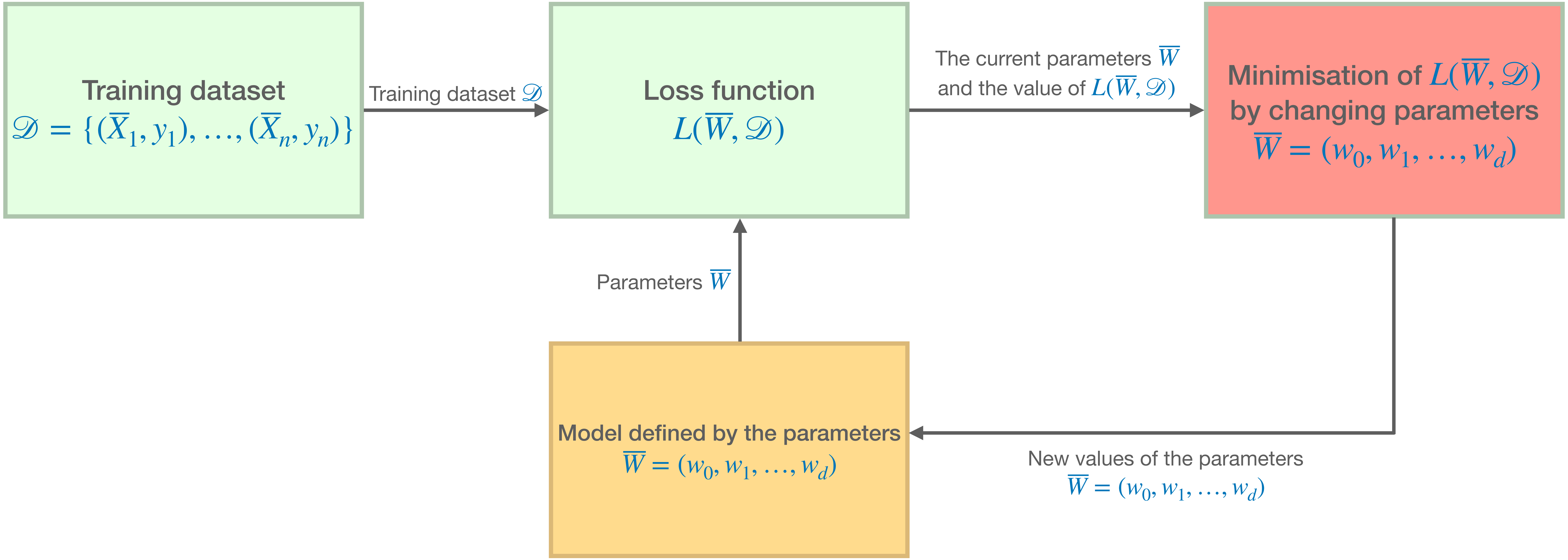
$$L(b, \overline{W}, \mathscr{D}) = \sum_{k=1}^{n} L(b, \overline{W}, \overline{X}_k, y_k) = \text{no. of misclassifications}$$

- this function is piecewise-constant with many discontinuities;
- the derivative of this function (when exists) is equal to 0;

Hence the gradient descent is not applicable!

Loss function: number of misclassifications

Loss function for the training dataset $\mathscr{D}$ as $L(b, \overline{W}, \mathscr{D}) = \sum_{k=1}^{n} L(b, \overline{W}, \overline{X}_k, y_k) =$ no. of misclassifications



Loss function: $h(t) = \max(0,t)$

Let $\mathscr{D} = \{(\overline{X}_1, y_1), \ldots, (\overline{X}_n, y_n)\}$ be the training dataset, where $\overline{X}_k = (x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(d)})^T$ for every $k = 1,\ldots,n$.

Let $a_k = b + \sum_{i=1}^{d} w_i x_k^{(i)}$.

Let $h(t) = \max(0,t)$.

Define **loss function** on a single training object $(\overline{X}_k, y_k)$ as $L(b, \overline{W}, \overline{X}_k, y_k) = h(-y_k \cdot a_k)$.

Loss function $h(t) = \max(0,t)$

Let $\mathscr{D} = \{(\overline{X}_1, y_1), \ldots, (\overline{X}_n, y_n)\}$ be the training dataset, where $\overline{X}_k = (x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(d)})^T$ for every $k = 1,..,n$.

Let $a_k = b + \sum_{i=1}^{d} w_i x_k^{(i)}$.

Let $h(t) = \max(0,t)$.

Define **loss function** for a single object $\overline{X}_k$ as $L(b, \overline{W}, \overline{X}_k, y_k) = h(-y_k \cdot a_k)$.

Define **loss function** for the training dataset $\mathscr{D}$ as $L(b, \overline{W}, \mathscr{D}) = \sum_{k=1}^{n} L(b, \overline{W}, \overline{X}_k, y_k) = \sum_{k=1}^{n} h(-y_k \cdot a_k)$.

Note that

$L(b, \overline{W}, \overline{X}_k, y_k) = 0$, if $\overline{X}_k$ is classified correctly

$L(b, \overline{W}, \overline{X}_k, y_k) = -y_k \cdot a_k \geq 0$, if $\overline{X}_k$ misclassified.

Hence the more misclassifications the model (with the parameters b and $\overline{W}$) does, the larger the **loss function** for the training dataset $L(b, \overline{W}, \mathscr{D})$ becomes.

We want to find the parameters b and $\overline{W}$ that minimise $L(b, \overline{W}, \mathscr{D})$!

Loss function minimisation

$$L(b, \overline{W}, \mathscr{D}) = \sum_{k=1}^{n} L(b, \overline{W}, \overline{X}_k, y_k) = \sum_{k=1}^{n} h(-y_k \cdot a_k)$$

Use the gradient descent method:

$$(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T - \mu \nabla_{b,w_1,\ldots,w_d} L(b, \overline{W}, \mathscr{D})$$

$$\nabla_{b,w_1,\ldots,w_d} L(b, \overline{W}, \mathscr{D}) = \sum_{k=1}^{n} \nabla_{b,w_1,\ldots,w_d} L(b, \overline{W}, \overline{X}_k, y_k) = \sum_{k=1}^{n} \nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k)$$

Computation of $\nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k)$

$h'(t) = 0$, when $t < 0$

$h'(t) = 1$, when $t \geq 0$ (h is not differentiable at $t = 0$, but we can extend h' by setting $h'(0) = 1$)

$$\frac{\partial h(-y_k \cdot a_k)}{\partial b} = h'(-y_k \cdot a_k) \cdot \frac{\partial}{\partial b}(-y_k \cdot a_k) = \begin{cases} -y_k, & \text{if } \overline{X}_k \text{ misclassified} \\ 0, & \text{otherwise} \end{cases}$$

Computation of $\nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k)$

$h'(t) = 0$, when $t < 0$

$h'(t) = 1$, when $t \geq 0$ (h is not differentiable at $t = 0$, but we can extend h' by setting $h'(0) = 1$)

$$\frac{\partial h(-y_k \cdot a_k)}{\partial w_i} = h'(-y_k \cdot a_k) \cdot \frac{\partial}{\partial w_i}(-y_k \cdot a_k) = \begin{cases} -y_k \cdot x_k^{(i)}, & \text{if } \overline{X}_k \text{ misclassified} \\ 0, & \text{otherwise} \end{cases}$$

Computation of $\nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k)$

$h'(t) = 0$, when $t < 0$

$h'(t) = 1$, when $t \geq 0$ (h is not differentiable at $t = 0$, but we can extend h' by setting $h'(0) = 1$)

$$\frac{\partial h(-y_k \cdot a_k)}{\partial b} = h'(-y_k \cdot a_k) \cdot \frac{\partial}{\partial b}(-y_k \cdot a_k) = -\, y_k$$

$$\frac{\partial h(-y_k \cdot a_k)}{\partial w_i} = h'(-y_k \cdot a_k) \cdot \frac{\partial}{\partial w_i}(-y_k \cdot a_k) = -\, y_k \cdot x_k^{(i)}$$

If $\overline{X}_k$ misclassified, then

$$\nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k) = \left(\frac{\partial h(-y_k \cdot a_k)}{\partial b}, \frac{\partial h(-y_k \cdot a_k)}{\partial w_1}, \ldots, \frac{\partial h(-y_k \cdot a_k)}{\partial w_d} \right)^T = -\, y_k \cdot \left(1, x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(d)}\right)^T$$

Otherwise $\nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k) = (0,0,0,\ldots,0)^T$

Loss function minimisation

$$\nabla_{b,w_1,\ldots,w_d} L(b, \overline{W}, \mathcal{D}) = \sum_{k=1}^{n} \nabla_{b,w_1,\ldots,w_d} L(b, \overline{W}, \overline{X}_k, y_k) = \sum_{k=1}^{n} \nabla_{b,w_1,\ldots,w_d} h(-y_k \cdot a_k)$$

Use the gradient descent method:

$$(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T + \mu \sum_{k:\ \overline{X}_k \text{ misc.}} y_k \cdot \left(1, x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(d)}\right)^T$$

Batch Gradient Descent
- to make a single update of the parameters we need to use whole training dataset
- extremely slow if we have a huge dataset

Online Gradient Descent

Idea: make parameter updates after each misclassification

Instead of $(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T - \mu \nabla_{b, w_1, \ldots, w_d} L(b, \overline{W}, \mathscr{D})$

For a misclassified object $(\overline{X}_k, y_k)$ the update becomes

$$(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T - \mu \nabla_{b, w_1, \ldots, w_d} L(b, \overline{W}, \overline{X}_k, y_k)$$

$$(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T - \mu \nabla_{b, w_1, \ldots, w_d} h(-y_k \cdot a_k)$$

Update rule for Perceptron

For a misclassified training object $(\overline{X}, y)$ with the activation score $a = b + \sum_{i=1}^{d} w_i x_i$ the weights are updated as follows

$$(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T - \mu \nabla_{b,w_1,\ldots,w_d} h(-y \cdot a)$$

$$\nabla_{b,w_1,\ldots,w_d} h(-y \cdot a) = -y \cdot \left(1, x_1, x_2, \ldots, x_d\right)^T$$

$$(b, w_1, \ldots, w_d)^T \leftarrow (b, w_1, \ldots, w_d)^T + \mu \cdot y \cdot \left(1, x_1, x_2, \ldots, x_d\right)^T$$

$b \leftarrow b + \mu \cdot y$

$w_i \leftarrow w_i + \mu \cdot y \cdot x_i$ for all $i = 1, \ldots, d$

By setting $\mu = 1$ we obtain exactly the update rule for Perceptron

The training algorithm

```
PerceptronTrain(Training data: D, MaxIter)
```

$w_i = 0$ `for all` $i = 1, \ldots, d$;

$b = 0$

for `iter = 1 … MaxIter` **do**

  for `all` $(\overline{X}, y) \in D$ **do**

    $a = \overline{W}^T \overline{X} + b$

    if $y \cdot a \leq 0$ **then**

      $w_i = w_i + y \cdot x_i$, `for all` $i = 1, \ldots, d$

      $b = b + y$

return $b, w_1, w_2, \ldots, w_d$

For $\mu = 1$

$w_i \leftarrow w_i + y \cdot x_i$

$b \leftarrow b + y$