

Perceptron

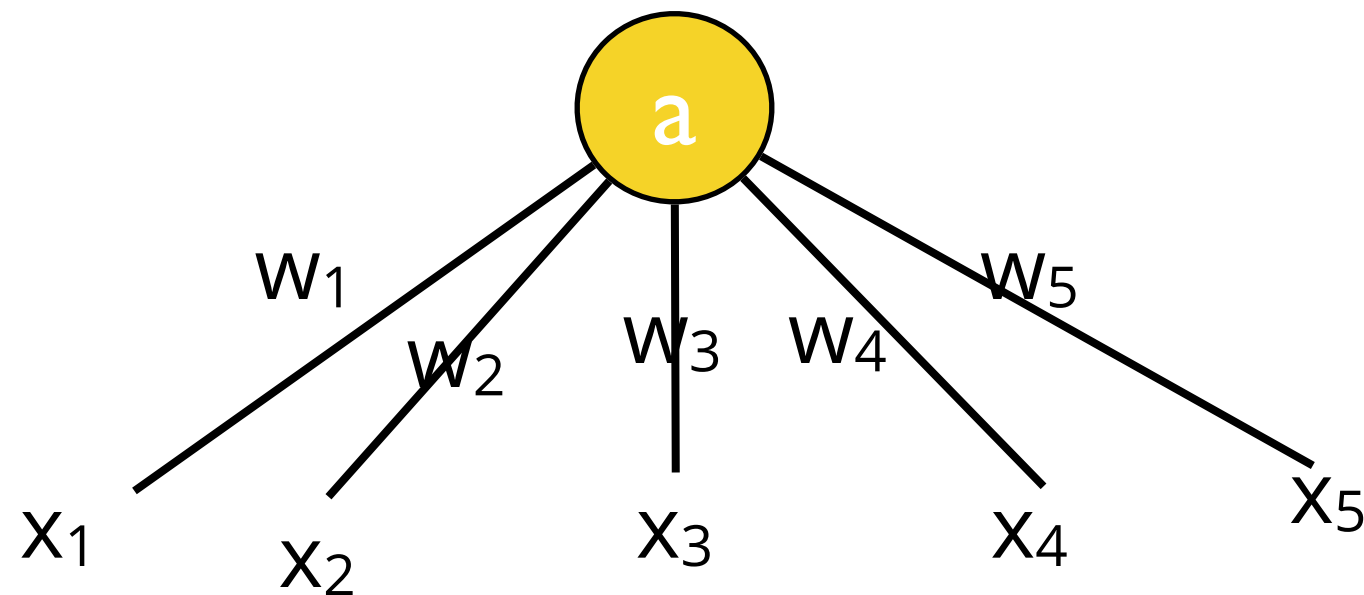
Danushka Bollegala

Bio-inspired model

- Perceptron is a bio-inspired algorithm that tries to mimic a single neuron
- We simply multiply each input (feature) by a weight and check whether this weighted sum (activation) is greater than a threshold.
- If so, then we “fire” the neuron (i.e. a decision is made based on the activation)

A single neuron

$$\text{activation (score)} = a = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$



if $a > \theta$ then
 output = 1
else
 output = 0

If the activation is greater than a predefined threshold, then the neuron fires.

Bias

- Often we need to adjust a fixed *shift* from zero, if the “interesting” region happens to be far from the origin.
- We adjust the previous model by including a bias term b as follows

$$a = b + \sum_{i=1}^D w_i x_i$$

Notational trick

- By introducing a feature that is always ON (i.e. $x_0 = 1$ for all instances), we can squeeze the bias term b into the weight vector by setting $w_0 = b$

$$a = \sum_{i=0}^D w_i x_i = \mathbf{w}^\top \mathbf{x}$$

This is more “elegant” as we can write the activation as the inner-product between the weight vector and the feature vector. However, we should keep in mind that bias term still appears in the model.

Perceptron

- Consider only one training instance at a time
 - online learning
 - k-NN considers ALL instances (batch learning)
- Learn only if we make a mistake when we classify using the current weight vector. Otherwise, we do not make adjustments to the weight vector
 - Error-driven learning

Algorithm 5 PERCEPTRONTRAIN(\mathbf{D} , $MaxIter$)

```
1:  $w_d \leftarrow 0$ , for all  $d = 1 \dots D$  // initialize weights
2:  $b \leftarrow 0$  // initialize bias
3: for  $iter = 1 \dots MaxIter$  do
4:   for all  $(x, y) \in \mathbf{D}$  do
5:      $a \leftarrow \sum_{d=1}^D w_d x_d + b$  // compute activation for this example
6:     if  $ya \leq 0$  then
7:        $w_d \leftarrow w_d + yx_d$ , for all  $d = 1 \dots D$  // update weights
8:        $b \leftarrow b + y$  // update bias
9:     end if
10:  end for
11: end for
12: return  $w_0, w_1, \dots, w_D, b$ 
```

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

```
1:  $a \leftarrow \sum_{d=1}^D w_d \hat{x}_d + b$  // compute activation for the test example
2: return SIGN( $a$ )
```

slide credit: CIML (Daume III)

Detecting errors

- In Line 6 of PerceptronTrain code we have
 - $ya \leq 0$
 - If the current instance is positive ($y = 1$), we should have a positive activation ($a > 0$) in order to have a correct prediction
 - If the current instance is negative ($y = -1$), we should have a negative activation ($a < 0$) in order to have a correct prediction
 - In both cases $ya > 0$.
 - Therefore, if $ya \leq 0$ then we have a misclassification

Update rule — Intuitive Explanation

- Perceptron update rule is
 - $\mathbf{w} = \mathbf{w} + y\mathbf{x}$
- If we incorrectly classify a positive instance as negative
 - We should have a higher (more positive) activation to avoid this
 - We should increase $\mathbf{w}^T\mathbf{x}$
 - Therefore, we should ADD the current instance to the weight vector
- If we incorrectly classify a negative instance as positive
 - We should have a lower (more negative) activation to avoid this
 - We should decrease $\mathbf{w}^T\mathbf{x}$
 - Therefore, we should DEDUCT the current instance from the weight vector

Update rule — Math Explanation

$$\begin{aligned} a' &= \sum_{d=1}^D w'_d x_d + b' \\ &= \sum_{d=1}^D (w_d + x_d) x_d + (b + 1) \\ &= \sum_{d=1}^D w_d x_d + b + \sum_{d=1}^D x_d x_d + 1 \\ &= a + \sum_{d=1}^D x_d^2 + 1 > a \end{aligned}$$

If the misclassified instance is a positive one, then after we update using $\mathbf{w} = \mathbf{w} + \mathbf{x}$, the new activation a' is greater than the old activation a .

Quiz 1

- Show that the analysis in the previous slide holds when $y = -1$ (i.e. we misclassified a negative instance)

Things to remember

- There is no guarantee that we will correctly classify a misclassified instance in the next round.
- We have simply increased/decreased the activation but this adjustment might not be sufficient. We might need to do more aggressive adjustments
- There are algorithms that enforce such requirements explicitly such as the Passive Aggressive Classifier (not discussed here)

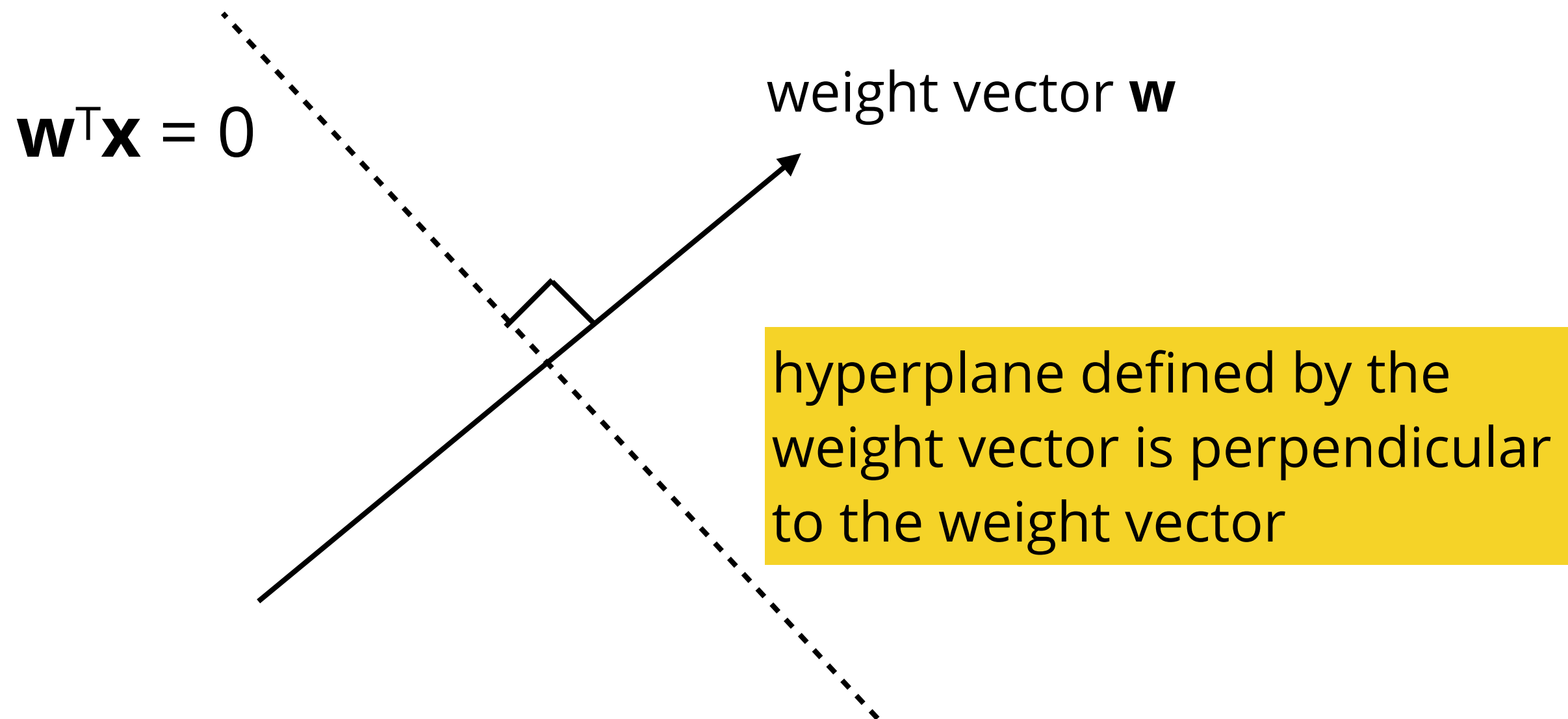
Ordering of instances

- Ordering training instances randomly within each iteration produces good results in practice
- Showing only all the positives first and all the negatives next is a bad idea

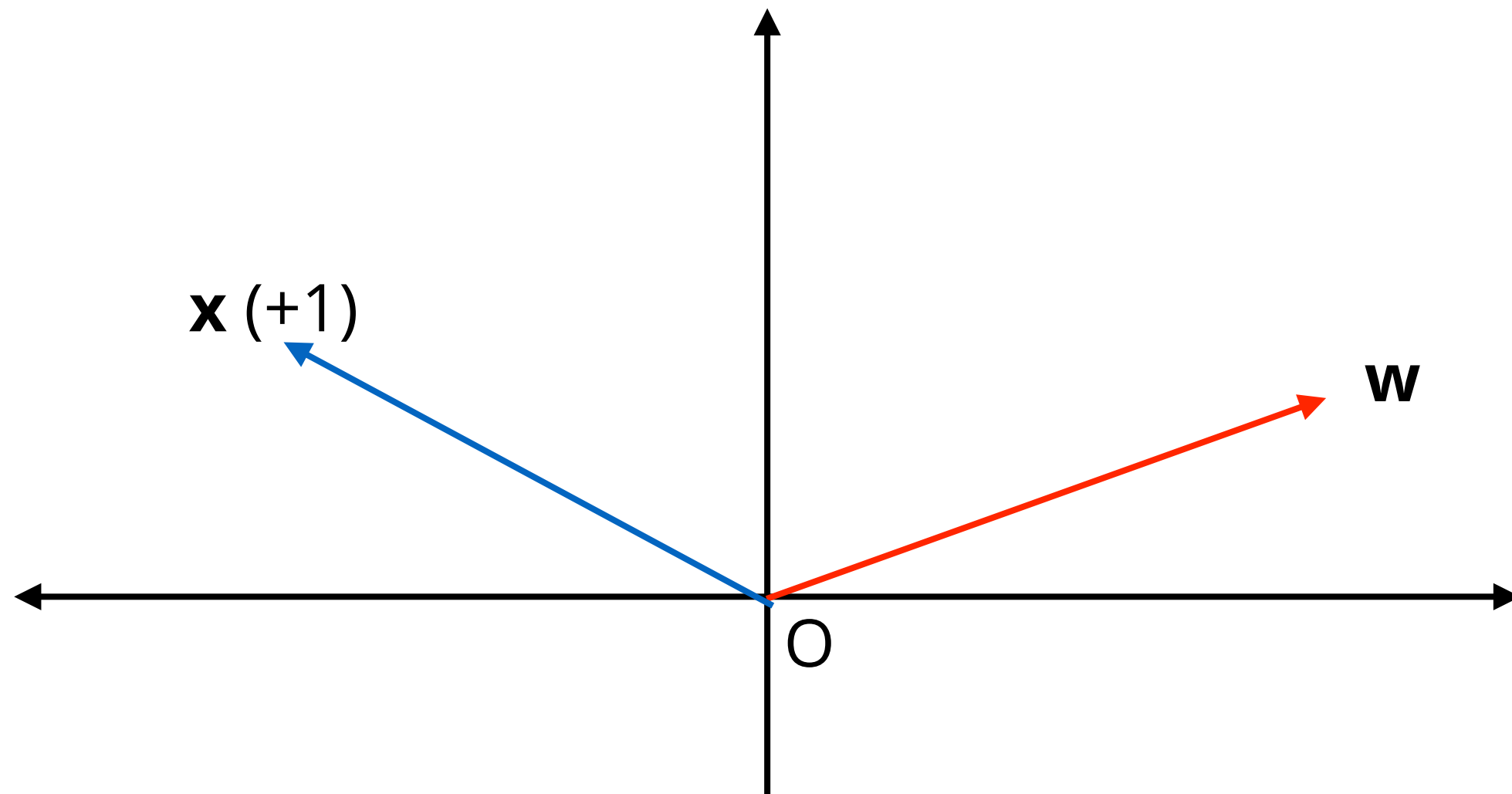
Hyperplane

- The decision in perceptron is made depending on $\mathbf{w}^T \mathbf{x} > 0$ or $\mathbf{w}^T \mathbf{x} \leq 0$
- Therefore, $\mathbf{w}^T \mathbf{x} = 0$ is the critical region (decision boundary)
- $\mathbf{w}^T \mathbf{x} = 0$ defines a hyperplane
- Example:
 - In 2D space we have $w_1 x_1 + w_2 x_2 = 0$ (ignoring the bias term), which is a straight line through the origin.
 - In N dimensional space this is an (N-1) dimensional hyperplane

Geometric Interpretation of Hyperplane

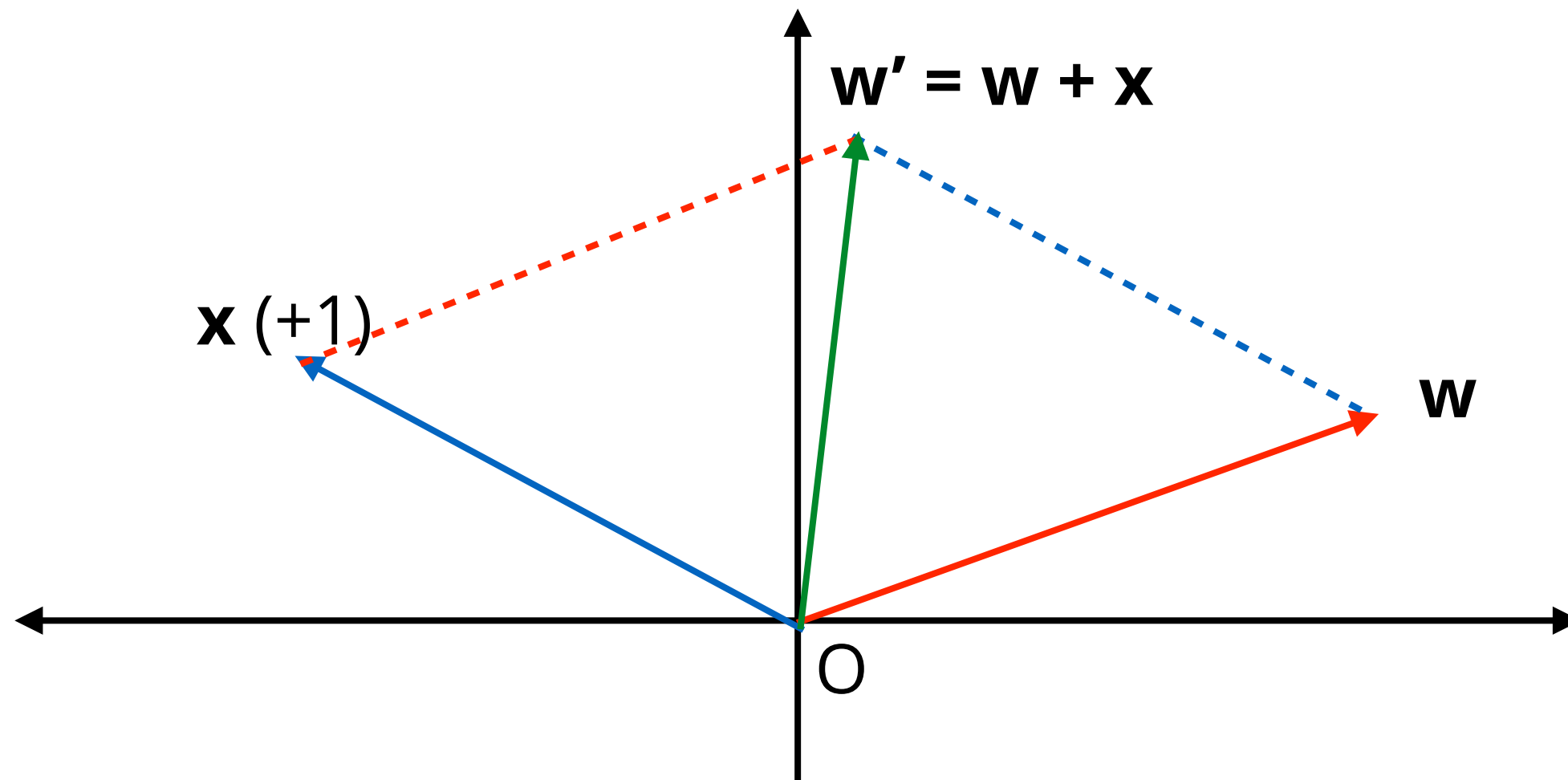


Geometric interpretation



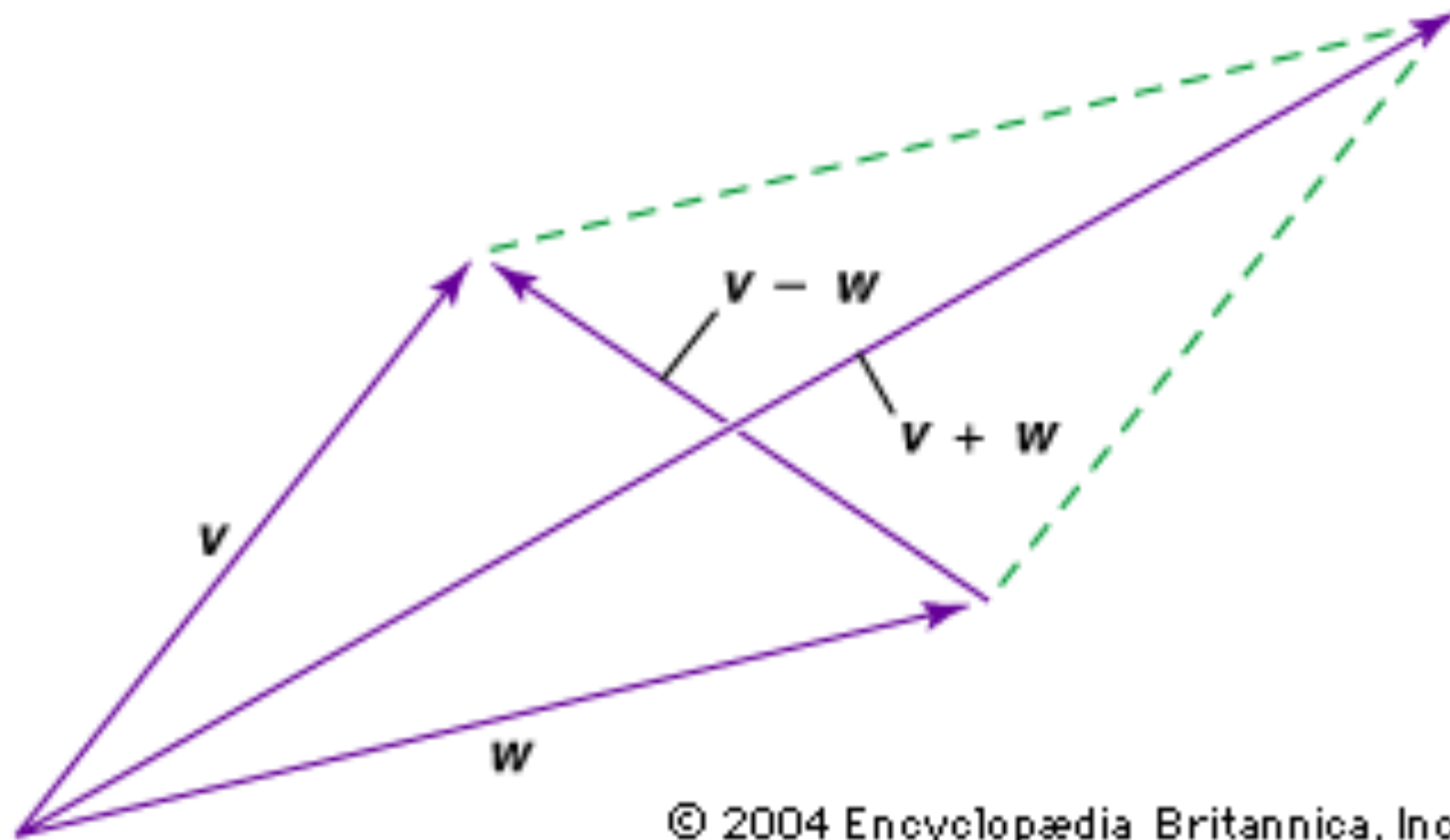
The angle between the current weight vector **w** and the positive instance **x** is greater than 90° . Therefore, $\mathbf{w}^\top \mathbf{x} < 0$, and this instance is going to get misclassified as negative.

Geometric interpretation



The new weight vector w' is the addition of $w + x$ according to the perceptron update rule. It lies in between x and w . Notice that the angle between w' and x is less than 90° . Therefore, x will be classified as positive by w' .

Vector algebra revision



Quiz 2

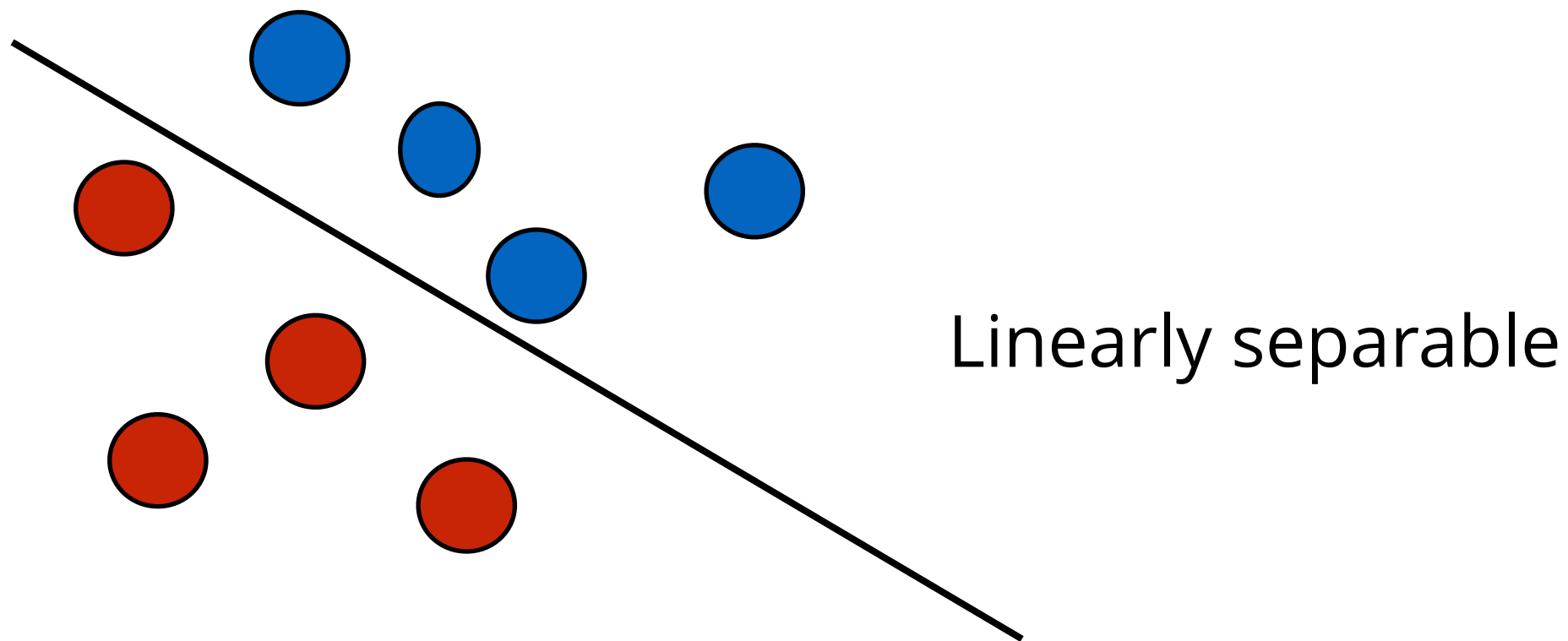
- Let $\mathbf{x} = (1, 0)^T$ and $\mathbf{y} = (1, 1)^T$. Compute $\mathbf{x} + \mathbf{y}$ and $\mathbf{x} - \mathbf{y}$ using the parallelogram approach described in the previous slide.

Quiz 3

- Provide a geometric interpretation for the update rule in Perceptron when a negative instance is mistaken to be positive.

Linear separability

- If a given set of positive and negative training instances can be separated into those two groups using a straight line (hyperplane), then we say that the dataset is *linearly separable*.

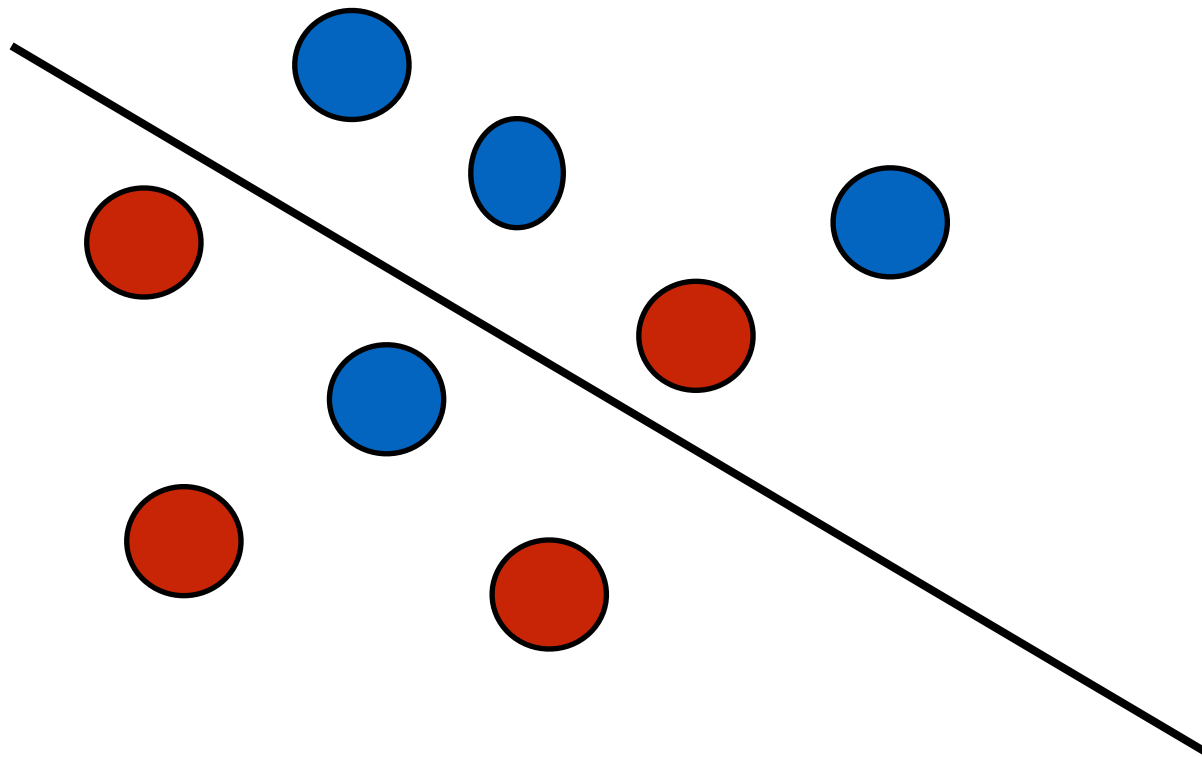


Remarks

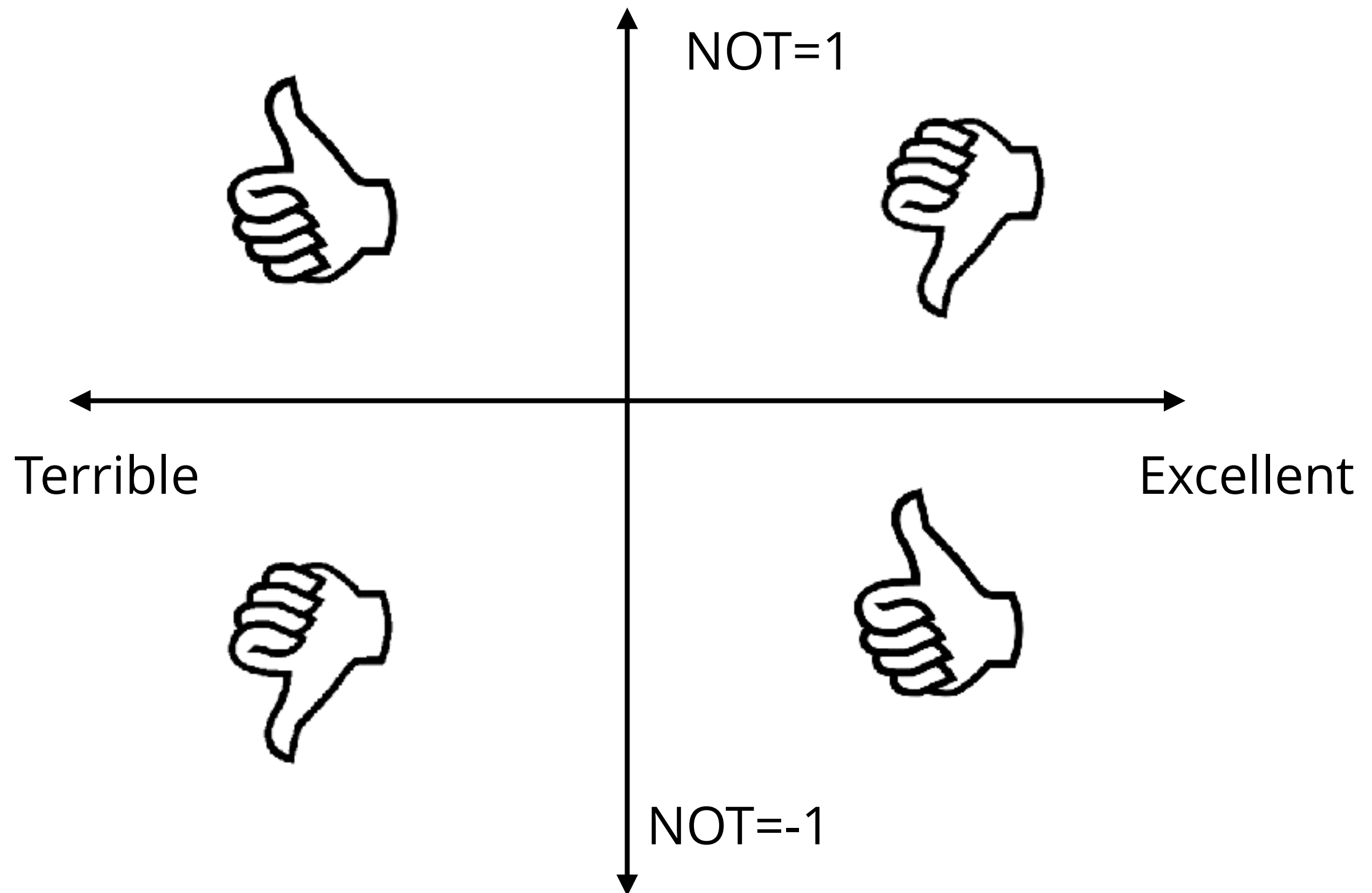
- When a dataset is linearly separable, there can exist more than one hyperplanes that separates the dataset into positive/negative groups.
- In other words, the hyperplane that linearly separates a linearly separable dataset might not be unique.
- However, (by definition) if a dataset is non-linearly separable, then there exist NO hyperplane that separates the dataset into positive/negative groups.

A non-linearly separable case

No matter how we draw straight lines, we cannot separate the red instances from the blue instances



Negation handling in Sentiment Classification



Mutually exclusive OR (XOR): $XOR(A,B) = 1$ only when one of the two inputs is 1.

Further Remarks

- When a dataset is linearly separable it can be proved that the perceptron will always find a separating hyperplane!
- The final weight vector returned by the Perceptron is more influenced by the final training instances it sees.
- Take the average over all weight vectors during the training (averaged perceptron algorithm)