Perceptron Update

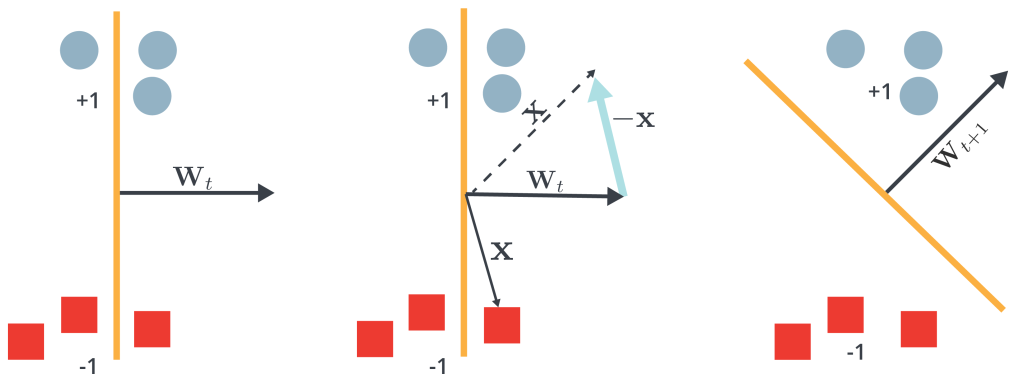
The perceptron algorithm aims to obtain a weight vector $\mathbf{w}$ that defines a hyperplane that separates the data of two classes based on their feature values. So long as there exists a hyperplane that can separate the data completely, the perceptron algorithm will find it in a finite number of steps using repeated perceptron updates.

☆ **Key Points**

The hyperplane is initially set to $\mathbf{w} = 0$.

The hyperplane is iteratively adjusted (or "tilted") using one misclassified point at a time.

The final hyperplane perfectly separates the two classes.



Above is an illustration of a perceptron update.

In the left diagram, the hyperplane defined by $\mathbf{w}_t$, which is the weight vector at iteration t, misclassified one red (-1) and one blue (+1) point. In the middle diagram, the rightmost red point $\mathbf{x}$ is chosen and used for an update because it lies on the wrong side of the hyperplane. Its label is -1, so we need to subtract $\mathbf{x}$ from $\mathbf{w}_t$ to obtain a new weight vector. On the right, the updated hyperplane $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{x}$ now separates the two classes and the perceptron algorithm has converged.

Perceptron's final $\mathbf{w}$ vector can vary based on the order it iterates and updates over the misclassified data points. For instance, the perceptron algorithm update in the example above could have used the positive data point that was misclassified rather than the negative data point. If perceptron algorithm terminated on this update, the $\mathbf{w}$ could have been different from the previous run where the algorithm update had used the negative data point.

To reiterate: the perceptron algorithm finds **a hyperplane** if the data points of different classes can be separated by a hyperplane. There can be infinitely many hyperplanes dividing the two classes; perceptron algorithm only guarantees to find one, not which one.

Notes:

We can “absorb” the bias b into $\mathbf{w}$ by appending a 1 to each data point, i.e.,

$$\mathbf{x} \to \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \text{ and } \mathbf{w} \to \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

This means that when calculating the dot product between the weight vector $\mathbf{w}$ and a data point $\mathbf{x}$ as $\mathbf{w}^T\mathbf{x}$, we implicitly add the bias term to the original weight-data dot product.