UNIVERSITY OF LEEDS

Class: Machine Learning

Neural Networks: Perceptron

Instructor: Matteo Leonetti

Learning outcomes

- Define an appropriate error function for the perceptron.
- Derive the corresponding update algorithm.
- Describe the difference between gradient descent and stochastic gradient descent.

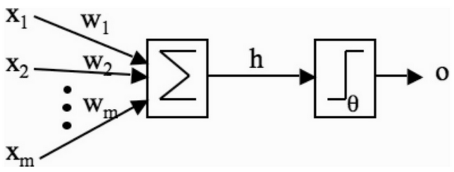
Recap

UNIVERSITY OF LEEDS

We want to apply gradient descent:

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

To the parameters of a perceptron:



So as to minimise an error (or loss) function, such as:

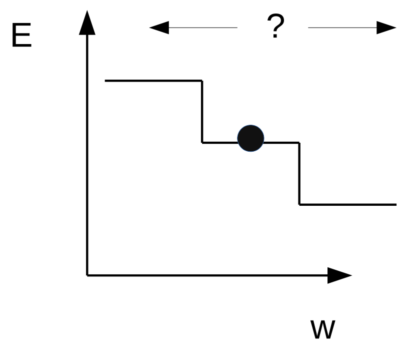
$$E(\boldsymbol{X}) = \sum_{\vec{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

Number of mistakes

UNIVERSITY OF LEEDS

$$E(\boldsymbol{X}) = \sum_{\vec{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

Number of mistakes on the dataset. Piecewise constant → no gradient.

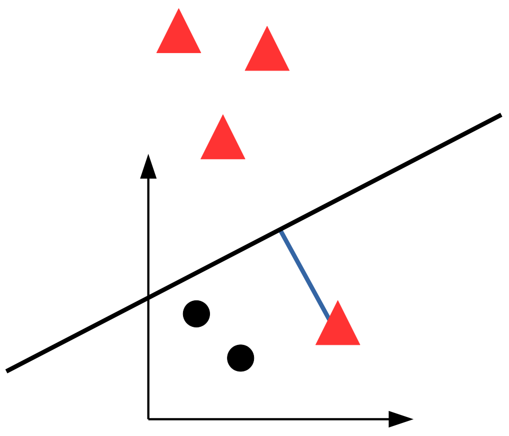


There is no local information on the direction of improvement

Rather than this function, which is piecewise constant, we would like an error function in which the error decreases as the current solution gets closer to a misclassified point.

Towards a better error function

UNIVERSITY OF LEEDS



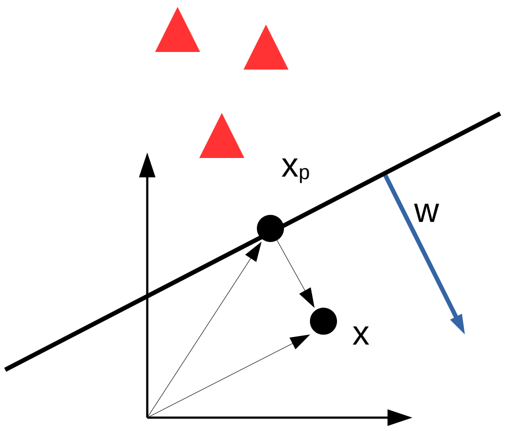
For each misclassified point, we would like to know not only that they are on the wrong side, but also **by how much**.

If we knew the distance of the misclassified point, we could tell whether by moving the decision boundary a little, we would be doing better or worse, that is, we would be decreasing or increasing the distance of the misclassified point from the boundary.

Towards a better error function

UNIVERSITY OF LEEDS

$$h_w(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = 0$$



Distance to the hyperplane

$$\mathbf{x} = \mathbf{x}_p + d\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$h_w(\mathbf{x}) = \mathbf{w}(\mathbf{x}_p + d\frac{\mathbf{w}}{\|\mathbf{w}\|}) + w_0$$

$$= \cancel{\mathbf{w}\mathbf{x}_p + w_0} + d\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} = d\|\mathbf{w}\|$$

Recall that:

$$\mathbf{w}^T\mathbf{w} = w_1^2 + w_2^2 + \cdots + w_n^2 = \|\mathbf{w}\|^2$$

In order to design such a function, we need a little geometry.

We can represent any point as the sum of its projection (x_p) on the neuron's hyperplane, and a vector orthogonal to the hyperplane (we know that this vector is the vector of weights **w**!).

If we evaluate the function h() on any point x, we get a number that is proportional to the distance of x from the hyperplane represented by h, since d is such a distance.

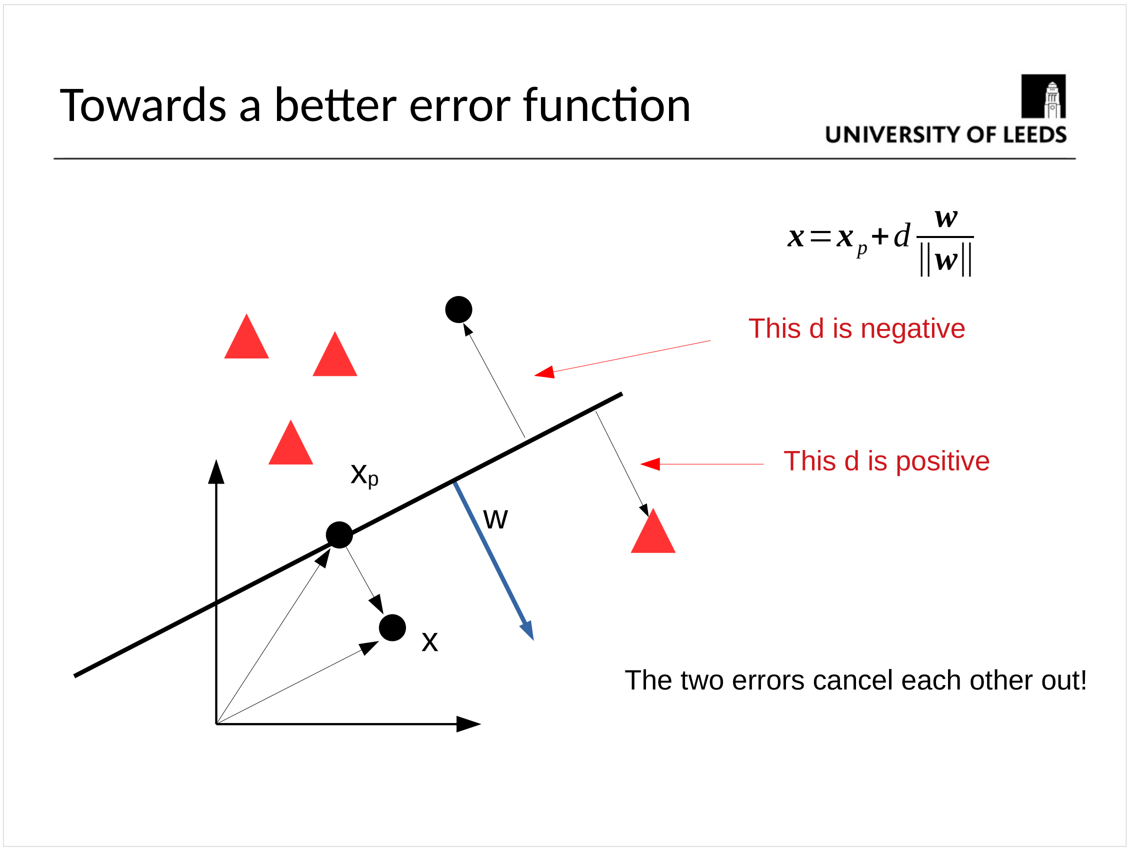
Towards a better error function

UNIVERSITY OF LEEDS

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = d\,\|\boldsymbol{w}\|$$

$$E(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} (\boldsymbol{w}^T \boldsymbol{x}_n + w_0)$$

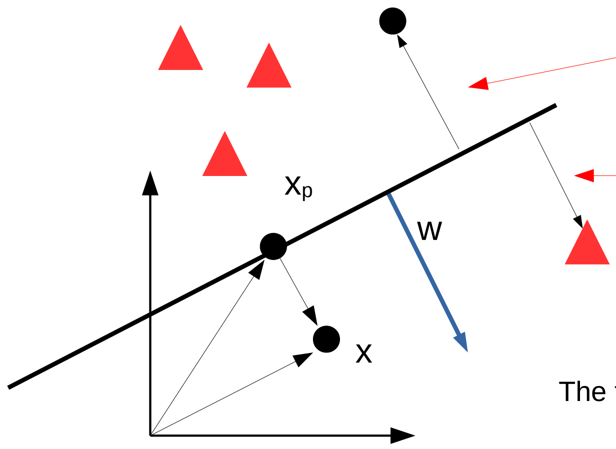
Is this a good error?



Towards a better error function

UNIVERSITY OF LEEDS

$$x = x_p + d\frac{w}{\|w\|}$$

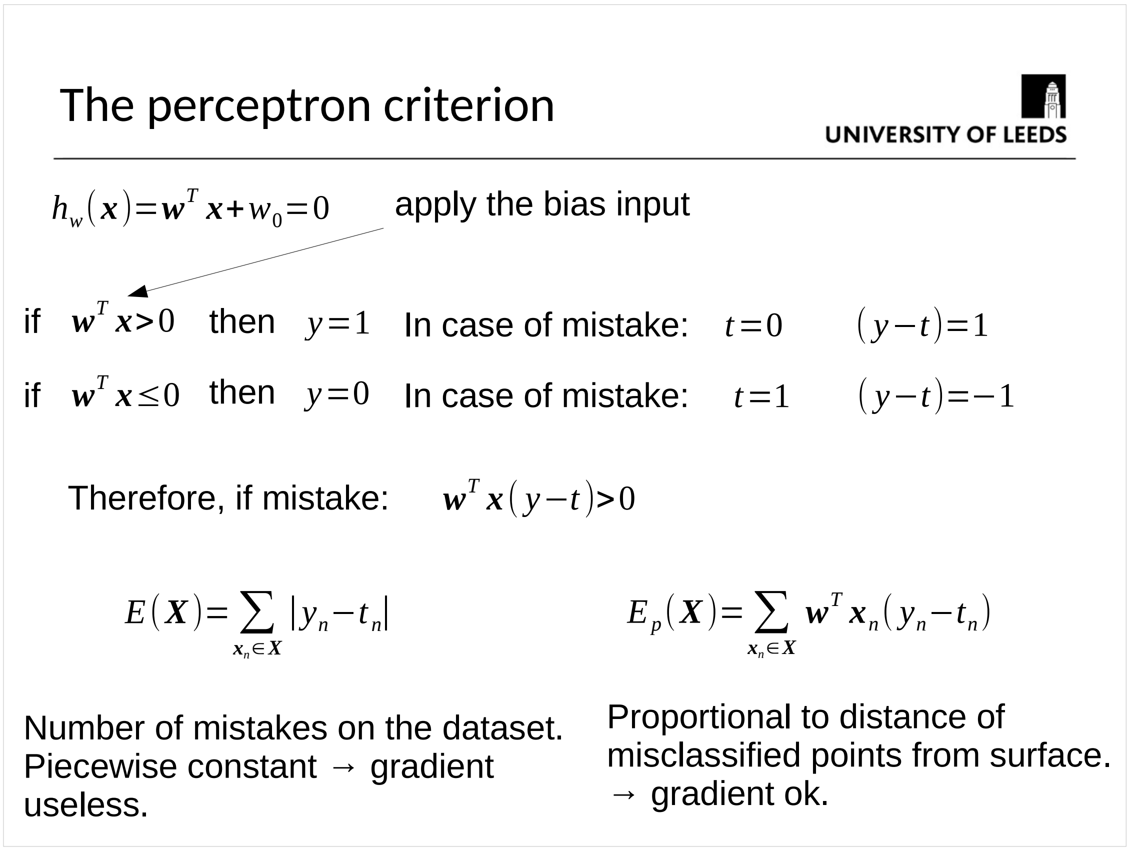


The two errors cancel each other out!

Each of the two misclassified point can be represented as $x_p + d\frac{w}{\|w\|}$ where x_p is the corresponding projection on the hyperplane (in 2D, straight line).

|| w || is always positive, so to give the two vectors opposite directions, one must have a positive d (the one in the direction of w) and the other a negative d (the one in the direction opposite to w's).

The two errors therefore cancel each other out, which must be avoided.



The perceptron criterion

UNIVERSITY OF LEEDS

$h_w(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$ apply the bias input

if $\boldsymbol{w}^T \boldsymbol{x} > 0$ then $y = 1$ In case of mistake: $t = 0$ $(y - t) = 1$

if $\boldsymbol{w}^T \boldsymbol{x} \leq 0$ then $y = 0$ In case of mistake: $t = 1$ $(y - t) = -1$

Therefore, if mistake: $\boldsymbol{w}^T \boldsymbol{x}(y - t) > 0$

$$E(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

$$E_p(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n)$$

Number of mistakes on the dataset. Piecewise constant → gradient useless.

Proportional to distance of misclassified points from surface. → gradient ok.

If we multiply $w^T x$ by (y-t) the corresponding value is always positive! (or zero if there is no error)

The new error function we just defined has several desirable properties:

It is proportional to the distance between the misclassified points and the hyperplane, therefore more informative than the number of mistakes.

It definitely has at least a minimum, because it is positive, therefore we cannot be trapped in an infinite loop going towards minus infinity.

It is easy to differentiate (as we will see shortly!).

Derivatives

What is the derivative of

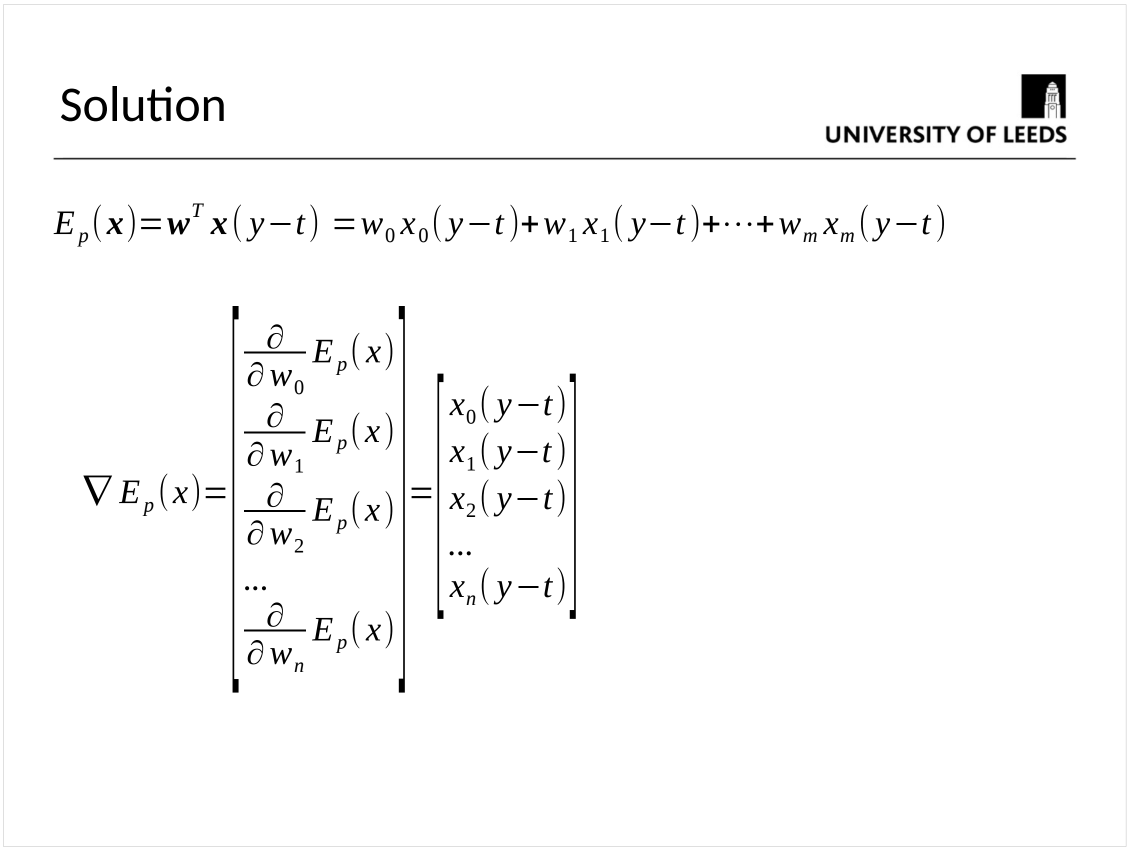
$y=2x$?

Question

UNIVERSITY OF LEEDS

Given the perceptron error (below), what is the gradient with respect to **w**?

$$E_p(\boldsymbol{X}) = \boldsymbol{w}^T \boldsymbol{x}(y-t) = (w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n)(y-t)$$



Solution

UNIVERSITY OF LEEDS

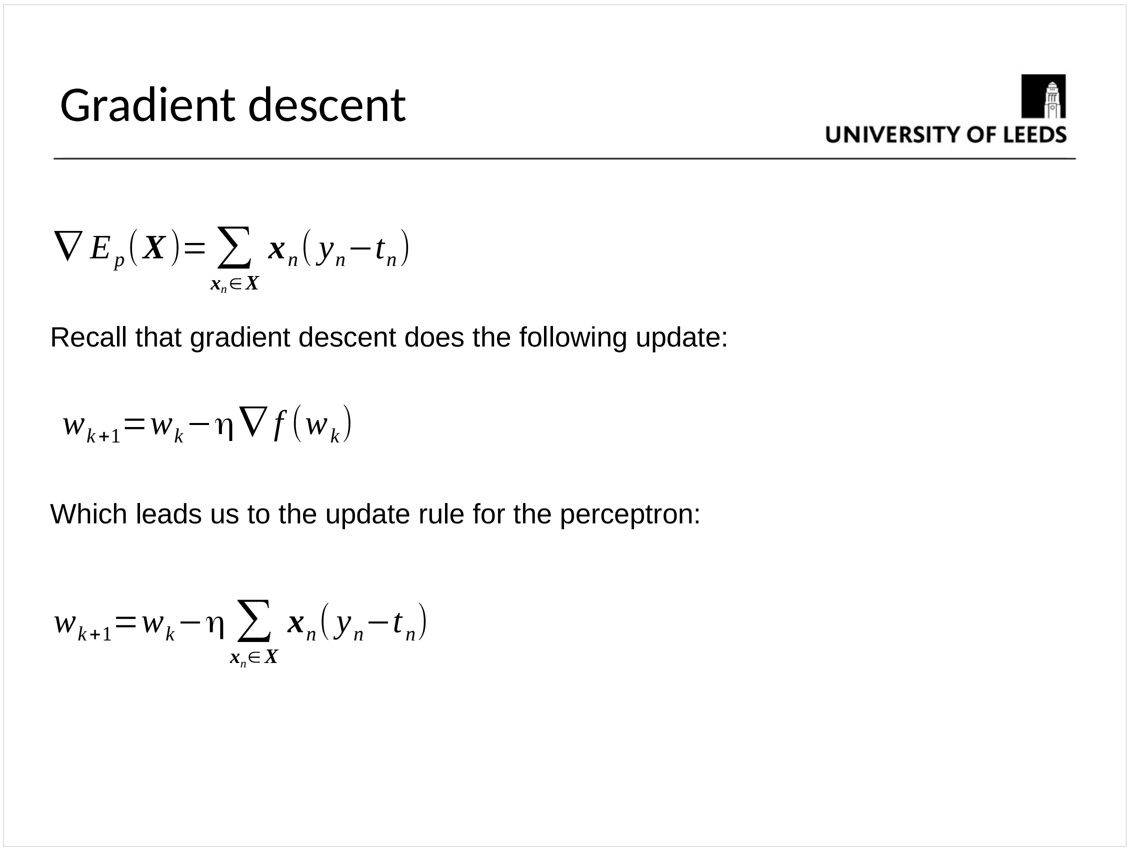
$$E_p(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}(y-t) = w_0 x_0 (y-t) + w_1 x_1 (y-t) + \cdots + w_m x_m (y-t)$$

$$\nabla E_p(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} E_p(x) \\ \frac{\partial}{\partial w_1} E_p(x) \\ \frac{\partial}{\partial w_2} E_p(x) \\ \ldots \\ \frac{\partial}{\partial w_n} E_p(x) \end{bmatrix} = \begin{bmatrix} x_0 (y-t) \\ x_1 (y-t) \\ x_2 (y-t) \\ \ldots \\ x_n (y-t) \end{bmatrix}$$

First, expand w into its components.

Then, consider the derivative with respect to each element of the vector w. All the other components of the sum disappear, because they are constant with respect to the derived element.

Therefore, the gradient is simply the vector of what is multiplied by each element, exactly as 2 is the derivative of 2x!



Gradient descent

UNIVERSITY OF LEEDS

$$\nabla E_p(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{x}_n (y_n - t_n)$$

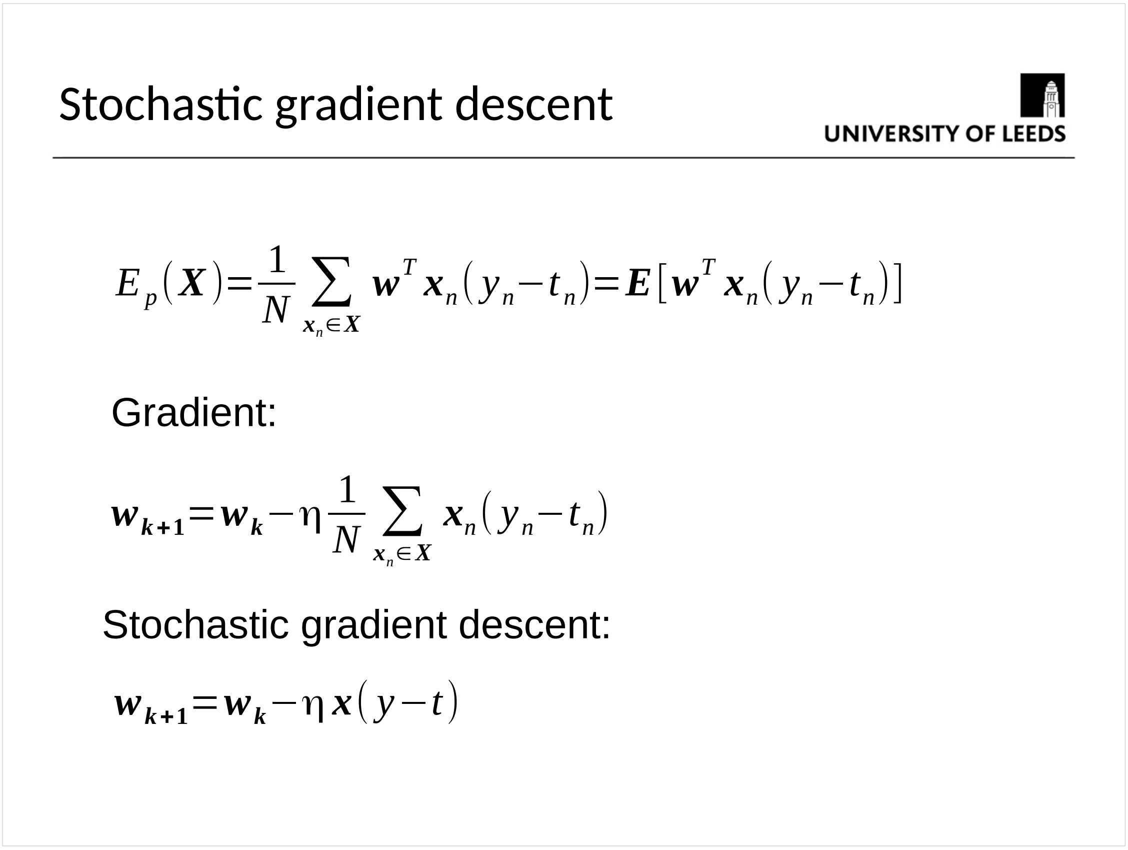
Recall that gradient descent does the following update:

$$w_{k+1} = w_k - \eta \nabla f(w_k)$$

Which leads us to the update rule for the perceptron:

$$w_{k+1} = w_k - \eta \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{x}_n (y_n - t_n)$$

The error here is computed over the whole dataset, hence the gradient is a sum of all the components due to each single point.



Stochastic gradient descent

UNIVERSITY OF LEEDS

$$E_p(\boldsymbol{X}) = \frac{1}{N} \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n) = \boldsymbol{E}[\boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n)]$$

Gradient:

$$\boldsymbol{w}_{\boldsymbol{k+1}} = \boldsymbol{w}_{\boldsymbol{k}} - \eta \frac{1}{N} \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{x}_n (y_n - t_n)$$

Stochastic gradient descent:

$$\boldsymbol{w}_{\boldsymbol{k+1}} = \boldsymbol{w}_{\boldsymbol{k}} - \eta\, \boldsymbol{x}(y - t)$$

Looking at the perceptron error, we can note that it is the sum of the error components due to each misclassified point.

Minimising the total error is the same as minmising the average error, since multiplying the objective function by a constant does not change the minima of the function.

The average error can be seen as an EXPECTED error, with respect to a distribution that extracts each point in the dataset with equal probability 1/N, where N is the number of points in the dataset.

There exists a version of gradient descent called *stochastic* gradient descent, in which the gradient is computed with respect to a single point extracted from a distribution. Therefore, under the assumption that all points have the same probability to be chosen (that is, 1/N), stochastic gradient descent converges **in expectation**, to the same solution as gradient descent on the average error.

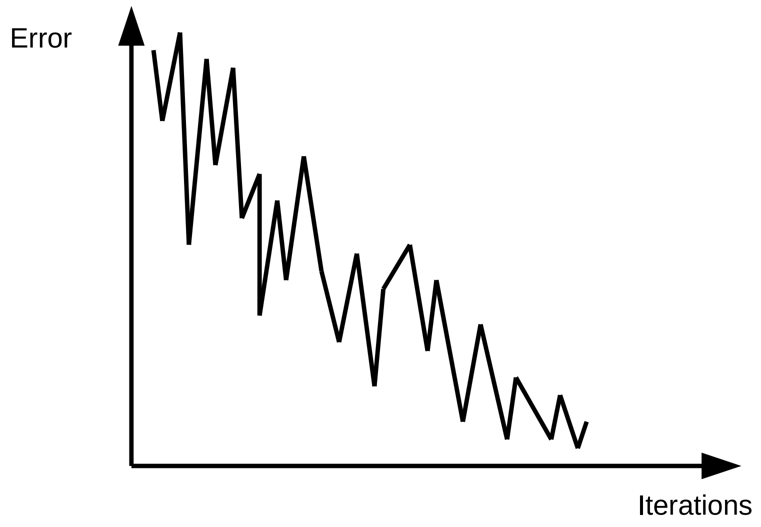
This gives us a much faster algorithm per iteration, since each update does not require to compute the sum over all the misclassified points. However, this algorithm will require, in general, more iterations.

So, each iteration is faster, but we will need more of them.

In practice, people use something in between, creating so called **mini-batches**. They use a certain number of mis-classified points (the size of the mini-batch), but not all the points in the dataset. This gives a better estimate of the gradient than a single point, but it is still faster to compute than the sum of all misclassified points.

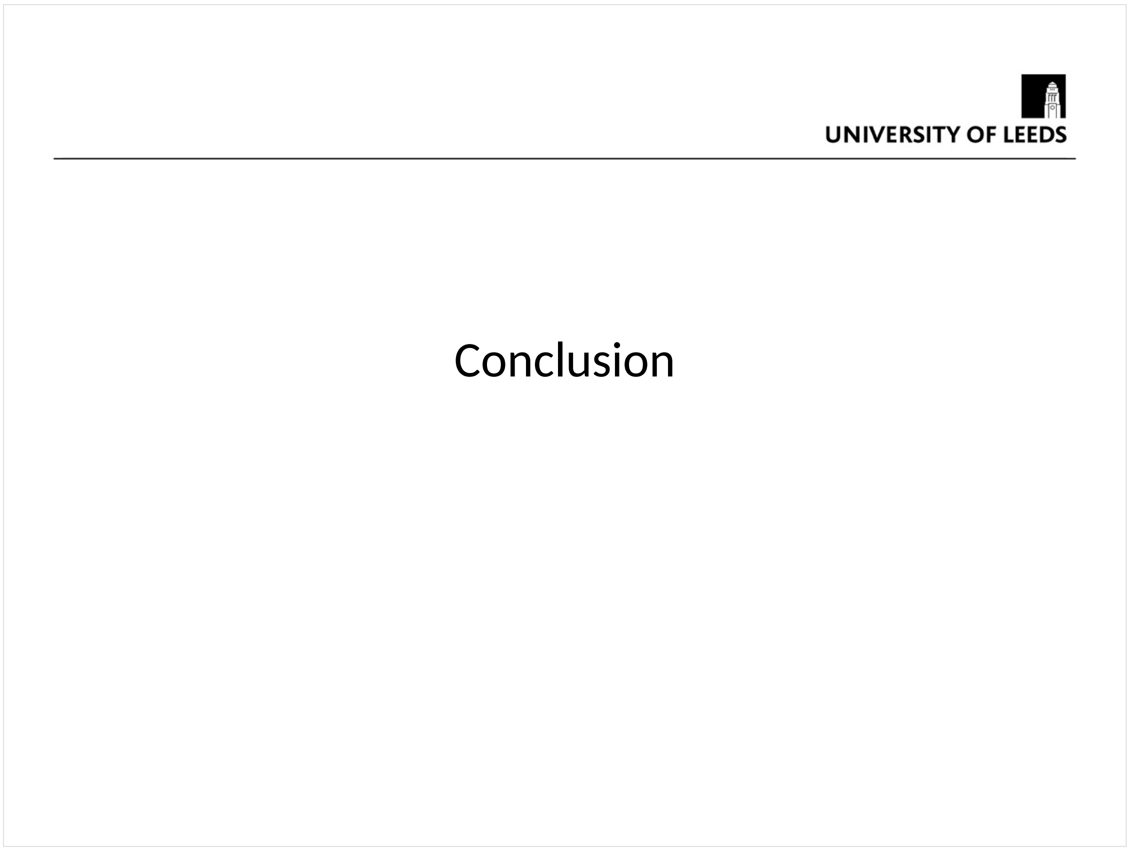
Stochastic gradient descent

UNIVERSITY OF LEEDS



While true gradient descent improves the function at every single step, stochastic gradient descent improves the function **in expectation**, that is **on average**.

The single steps will improve the error with respect to the point taken into account, but may increase the overall error over the whole dataset. However, the error will decrease on average.

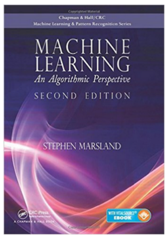


Conclusion

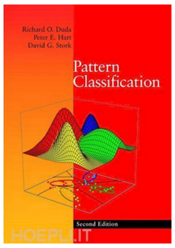
Learning outcomes

UNIVERSITY OF LEEDS

- Define an appropriate error function for the perceptron.
- Derive the corresponding update algorithm.
- Describe the difference between gradient descent and stochastic gradient descent.



Section 3.4



Book in Minerva
in “ Online Course Readings Folder”

Section 5.2.1, 5.4. and 5.5
(without convergence proof)