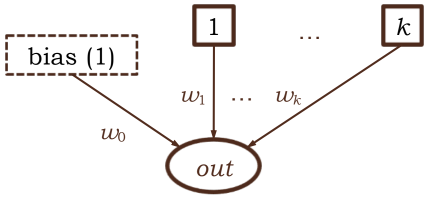
Gradient Descent: Simple Example

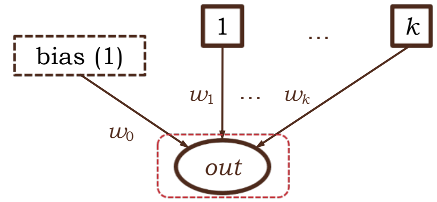


- Suppose we build a simple perceptron network
  - A single input layer, with k feature-values $x_1, \ldots, x_k$
  - An output using the Sigmoid/logistic function

$$out = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}}}$$

1

Measuring Error for Gradient Descent

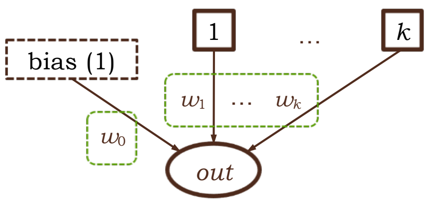


- We can measure loss by computing the mean squared error (MSE) at the output neuron, for n data-points:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - out_i)^2$$

2

Updating Weights



- We want to descend the gradient of the error
- We change each particular weight w according to the derivative of the error with respect to that weight:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial h_{\mathbf{w}}(\mathbf{x})}\,\frac{\partial h_{\mathbf{w}}(\mathbf{x})}{\partial \mathbf{w}\cdot\mathbf{x}}\,\frac{\partial \mathbf{w}\cdot\mathbf{x}}{\partial w}$$

3

Deriving the Final Result

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial h_{\mathbf{w}}(\mathbf{x})}\,\frac{\partial h_{\mathbf{w}}(\mathbf{x})}{\partial \mathbf{w}\cdot\mathbf{x}}\,\frac{\partial \mathbf{w}\cdot\mathbf{x}}{\partial w}$$

- We use the chain rule for derivatives to decompose what we want (LHS in the above equation) into 3 parts:

1. Derivative of MSE loss with respect to the output (logistic)

$$\frac{\partial \mathcal{L}}{\partial h_{\mathbf{w}}(\mathbf{x})} = -2(y - h_{\mathbf{w}}(\mathbf{x}))$$

2. Derivative of output (logistic) with respect to its linear-sum input

$$\frac{\partial h_{\mathbf{w}}(\mathbf{x})}{\partial \mathbf{w}\cdot\mathbf{x}} = h'_{\mathbf{w}}(\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))$$

3. Derivative of linear sum with respect to the weight

$$\frac{\partial \mathbf{w}\cdot\mathbf{x}}{\partial w} = \mathbf{x}$$

4

Deriving the Final Result

$$\frac{\partial \mathcal{L}}{\partial w} = -2(y - h_{\mathbf{w}}(\mathbf{x}))(h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))\mathbf{x}$$

- The final form of the update can be simplified:

1. We are going to *subtract* the gradient term from the weight to minimize loss, meaning we can just drop the negative sign and *add* instead
2. The constant 2, while giving an exact solution, is *unnecessary* for gradient descent (especially since we typically multiply the update by learning rate α)

- Our final update then is to update the weights by:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - h_{\mathbf{w}}(\mathbf{x}))(h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))\mathbf{x}$$