

ENM 5310: Data-driven Modeling and Probabilistic Scientific Computing

Lecture #12: Multi-layer perceptrons

Paris Perdikaris
March 14, 2023



Feed-forward neural networks

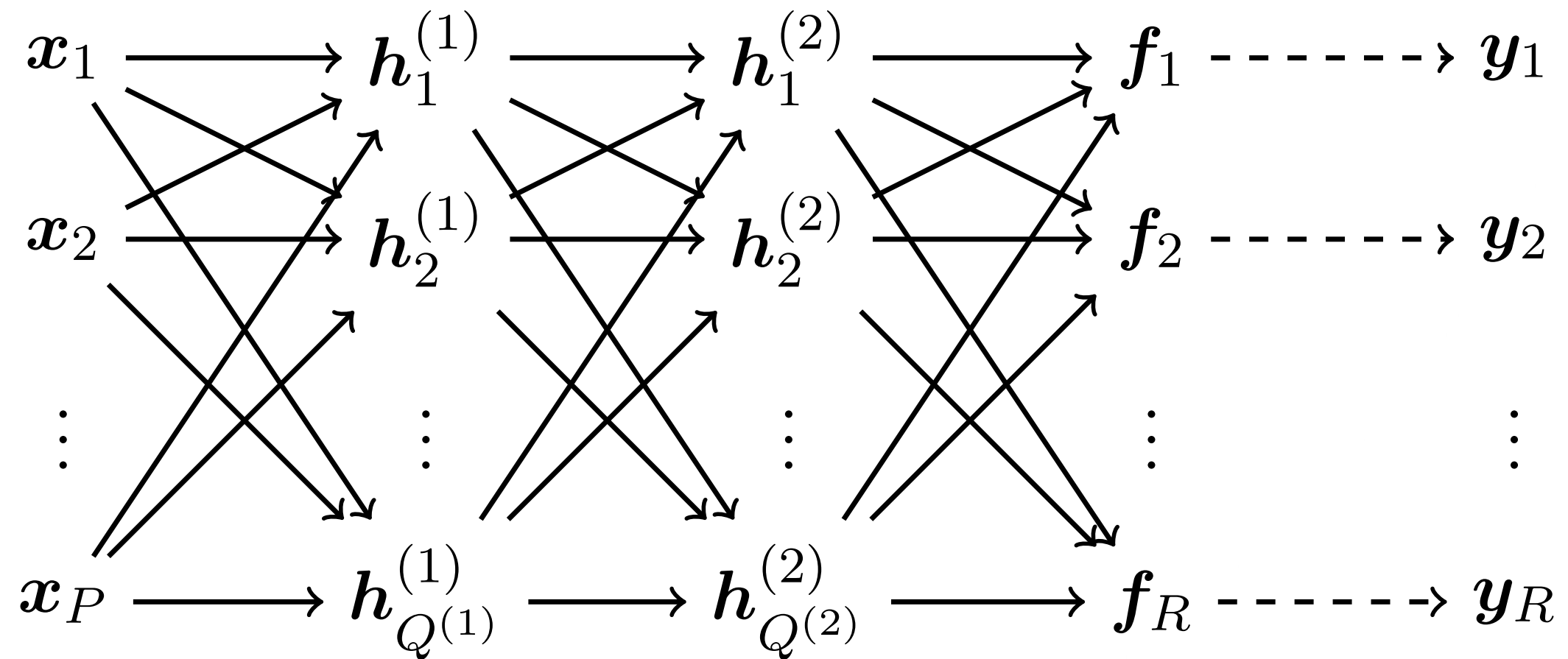
Pros:

- Adaptive features/basis functions (parametric)
- Flexible non-linear regression models that can approximate any function.
- Scalability to high dimensions.

Cons:

- The likelihood function is no longer a convex function of the model parameters.
- Over-fitting in data-scarce scenarios.
- Results are hard to interpret.

Feed-forward neural networks



Universal approximation theorem

Theorem 1. *Let σ be any continuous discriminatory function. Then finite sums of the form*

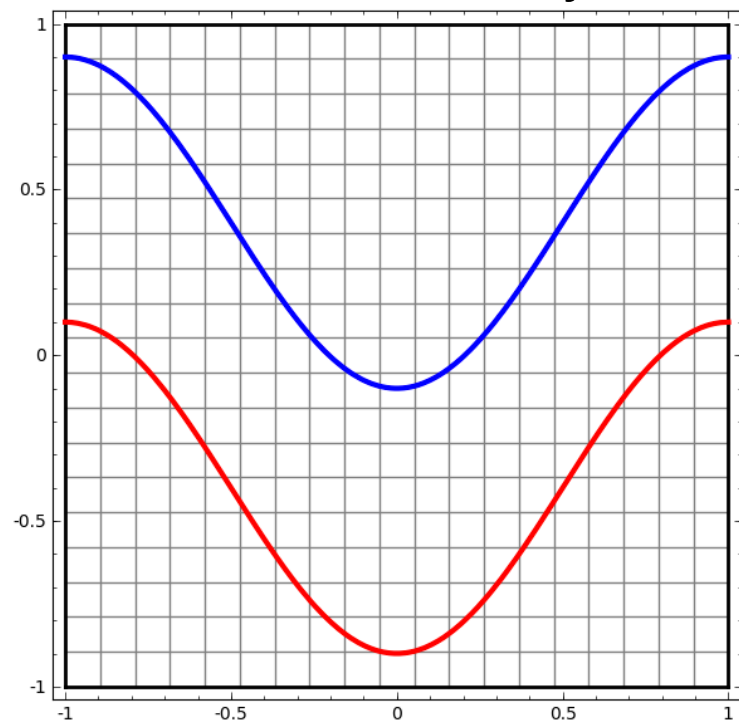
$$G(x) = \sum_{j=1}^N \alpha_j \sigma(y_j^T x + \theta_j) \quad (2)$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(x)$, of the above form, for which

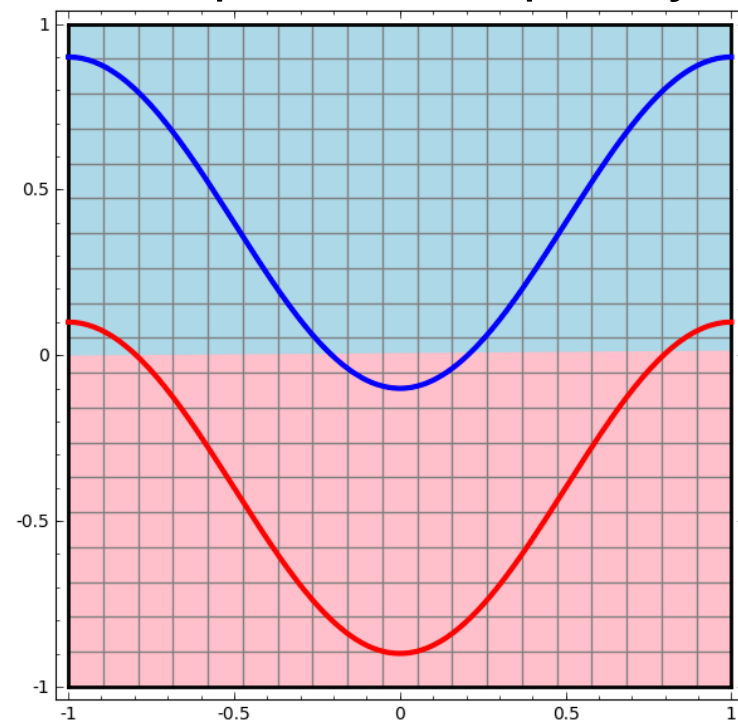
$$|G(x) - f(x)| < \varepsilon \quad \text{for all } x \in I_n.$$

Some intuition

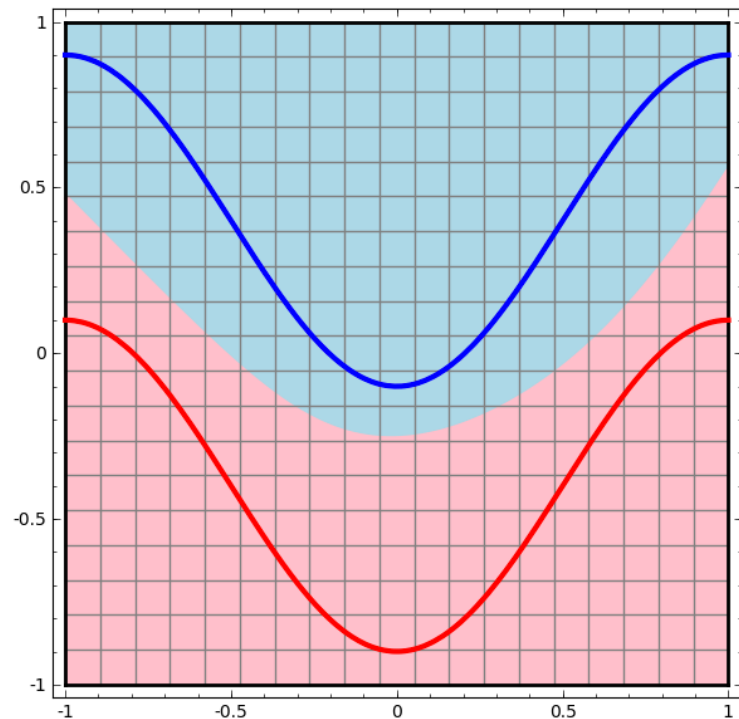
Data to classify



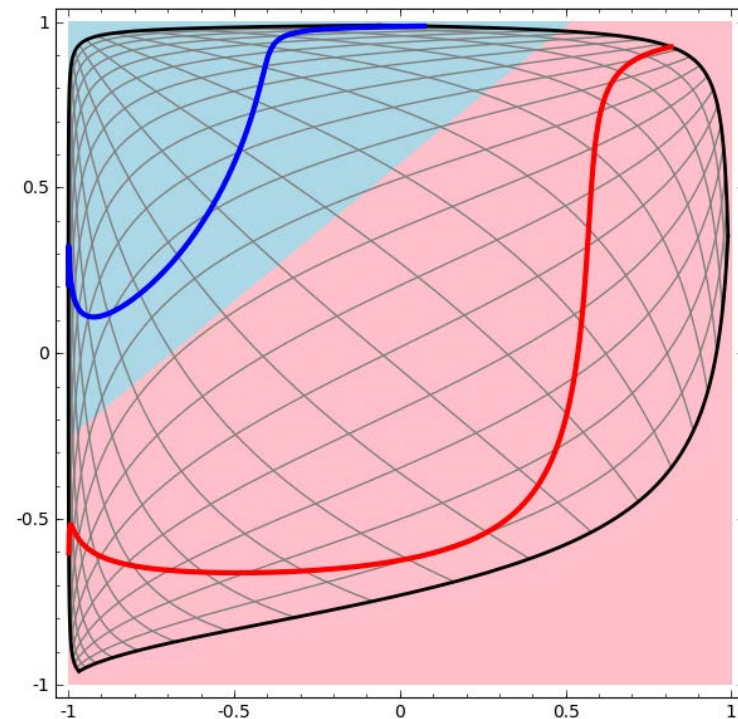
One input, one output layer



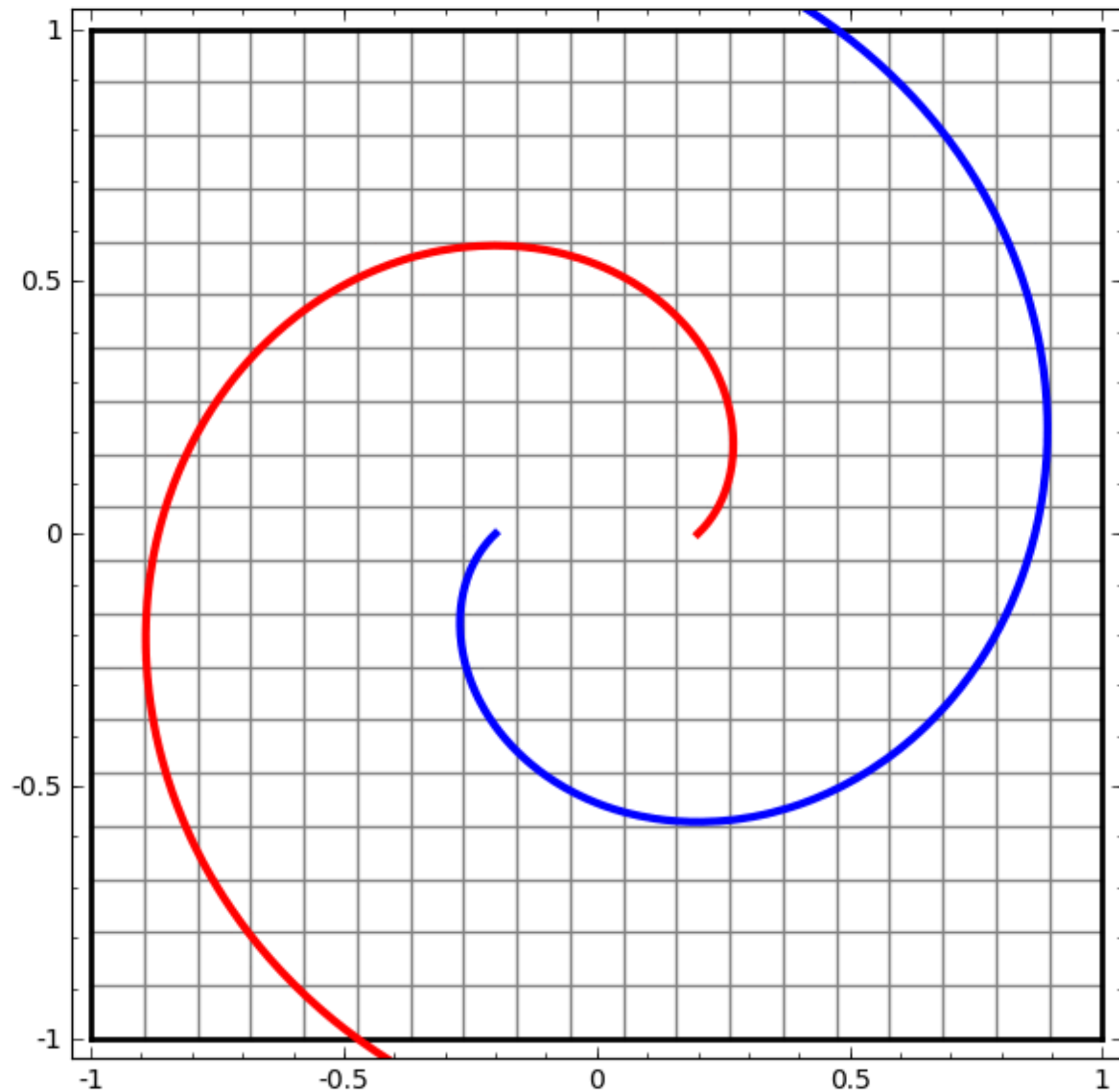
One input, one hidden, one output layer



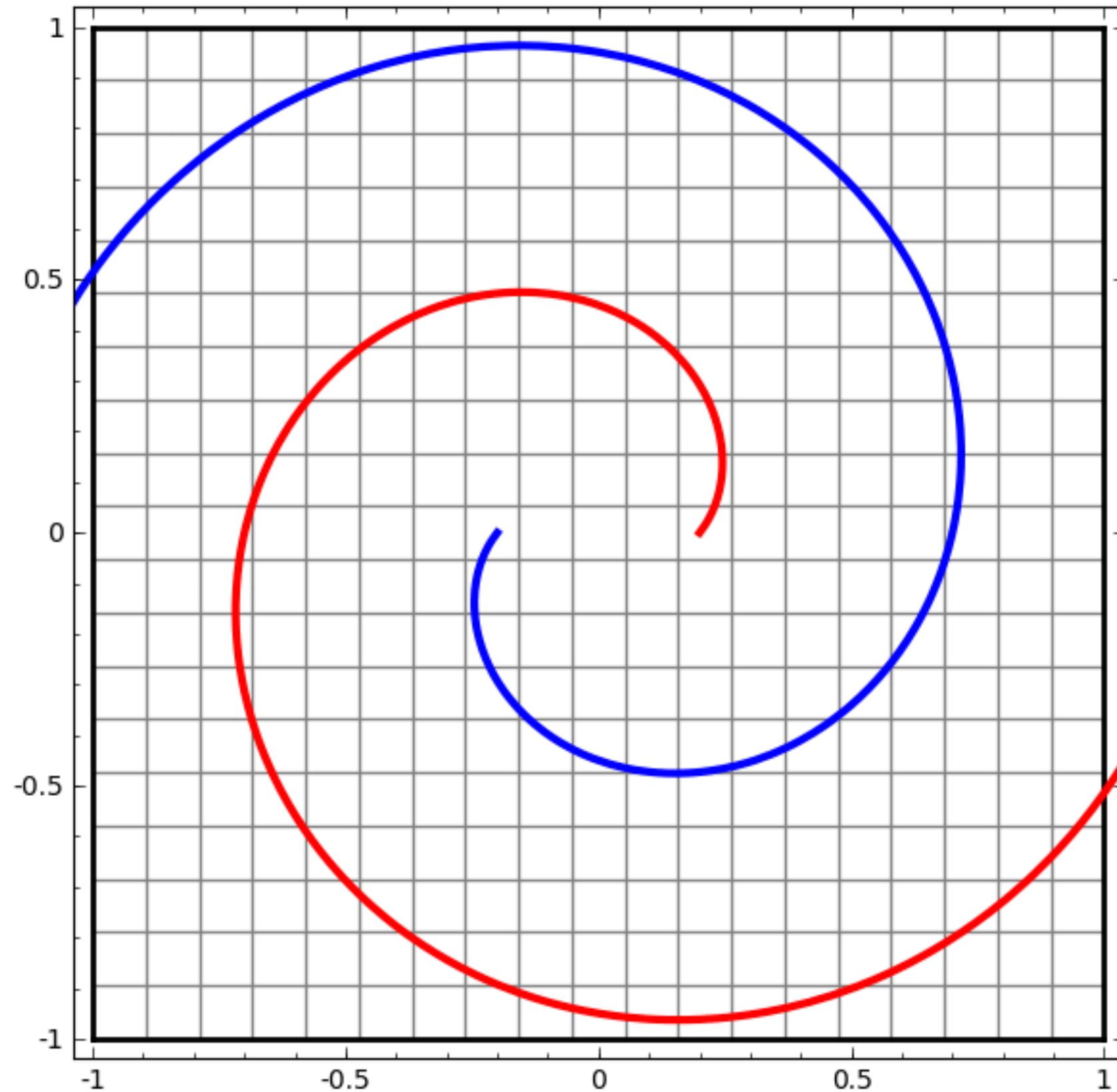
Visualizing the hidden layer



Some intuition



Some intuition



Some intuition

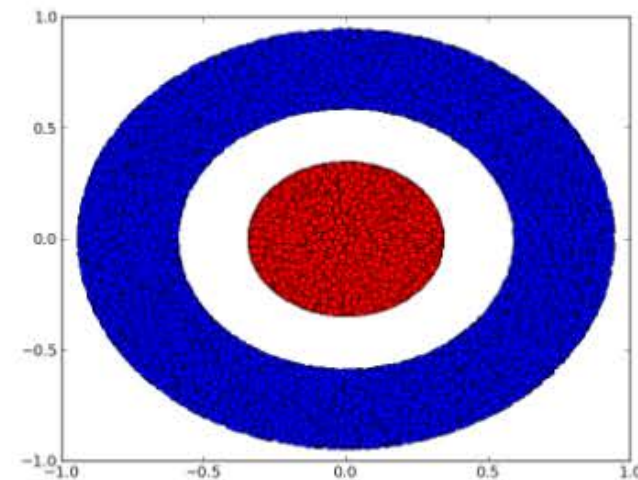
Topology and Classification

Consider a two dimensional dataset with two classes $A, B \subset \mathbb{R}^2$:

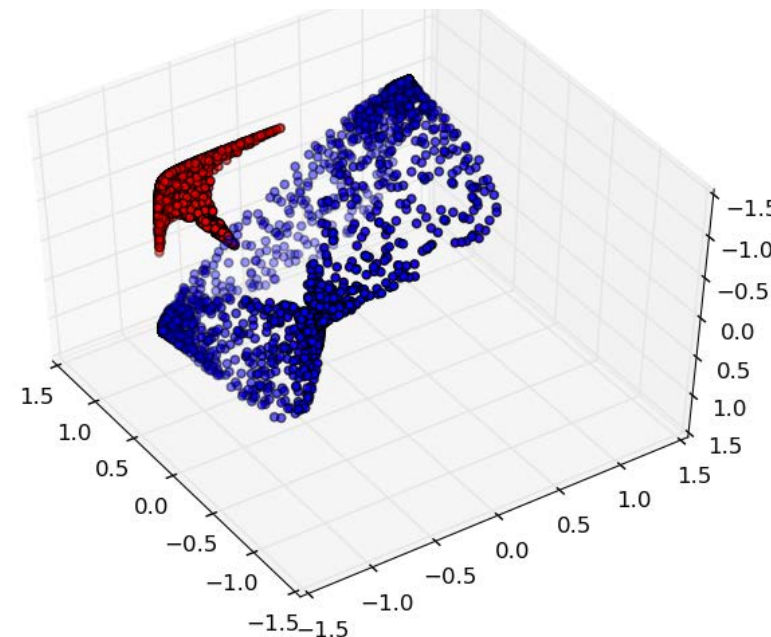
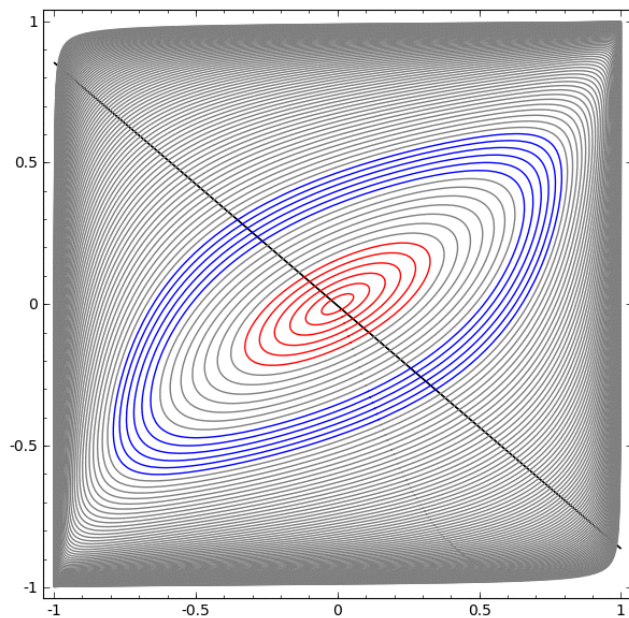
$$A = \{x | d(x, 0) < 1/3\}$$

$$B = \{x | 2/3 < d(x, 0) < 1\}$$





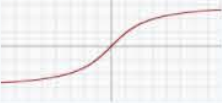



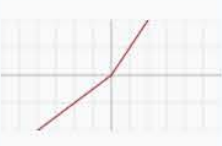

Claim: It is impossible for a neural network to classify this dataset without having a layer that has 3 or more hidden units, regardless of depth.



A is red, B is blue

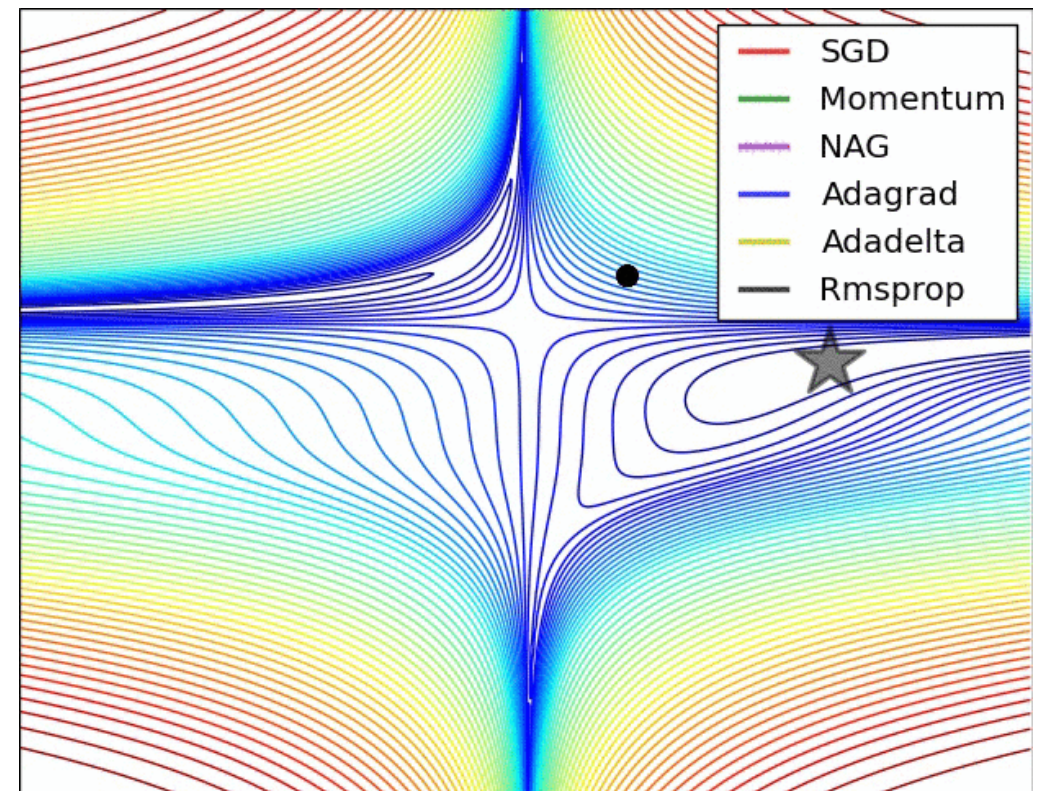
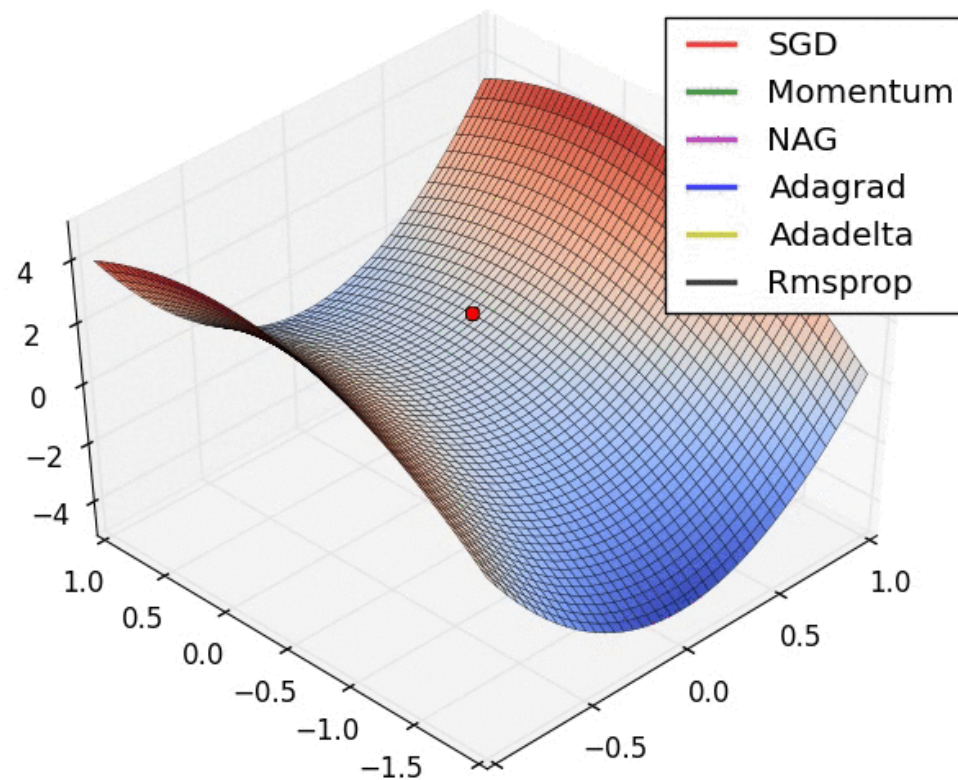


Activation functions

Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity	Monotonic	Derivative Monotonic	Approximates identity near the origin
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}	Yes	No	No
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞	Yes	No	No
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞	Yes	No	Yes
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞	Yes	No	Yes
Softsign ^{[7][8]}		$f(x) = \frac{x}{1 + x }$	$f'(x) = \frac{1}{(1 + x)^2}$	$(-1, 1)$	C^1	Yes	No	Yes
Inverse square root unit (ISRU) ^[9]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1 + \alpha x^2}} \right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$	C^∞	Yes	No	Yes
Rectified linear unit (ReLU) ^[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0	Yes	Yes	No
Leaky rectified linear unit (Leaky ReLU) ^[11]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes	No
Parametric rectified linear unit (PReLU) ^[12]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes iff $\alpha \geq 0$	Yes	Yes iff $\alpha = 1$



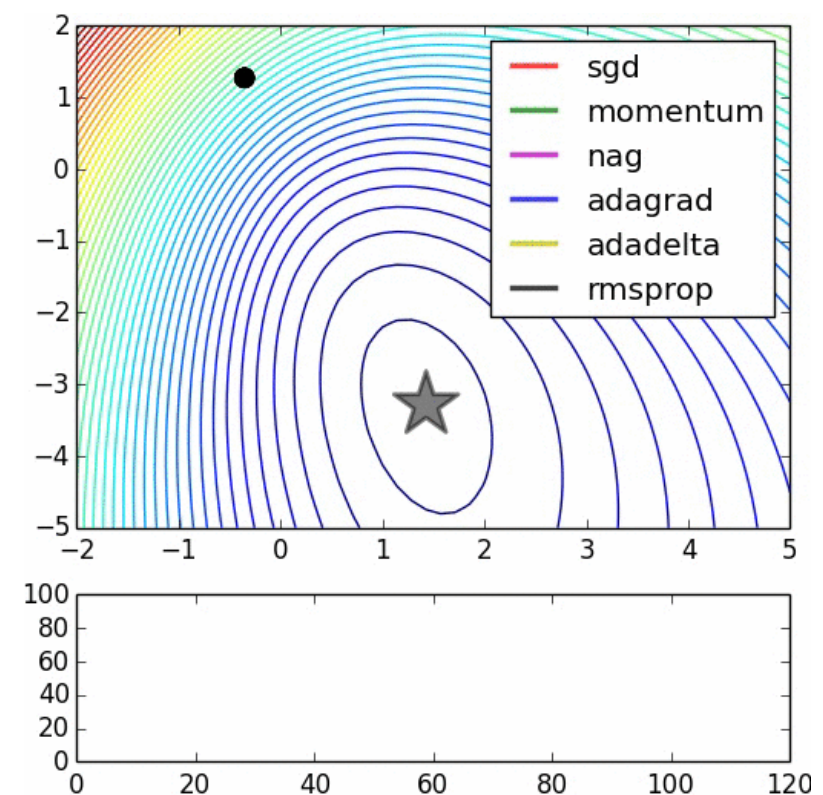
Modern SGD variants



<http://runder.io/optimizing-gradient-descent/>

<https://distill.pub/2017/momentum/>

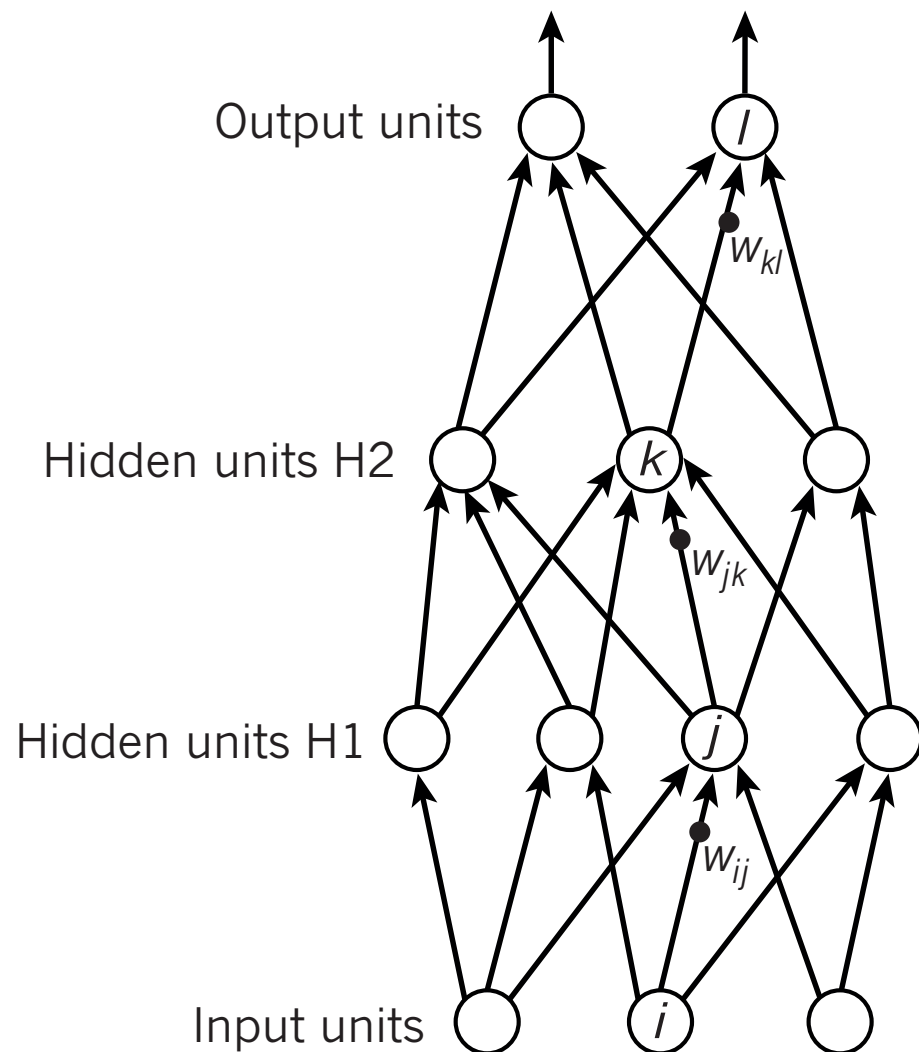
<http://louistiao.me/notes/visualizing-and-animating-optimization-algorithms-with-matplotlib/>



*animation credit: Alec Redford

Back-propagation

Forward pass



$$y_l = f(z_l)$$

$$z_l = \sum_{k \in H2} w_{kl} y_k$$

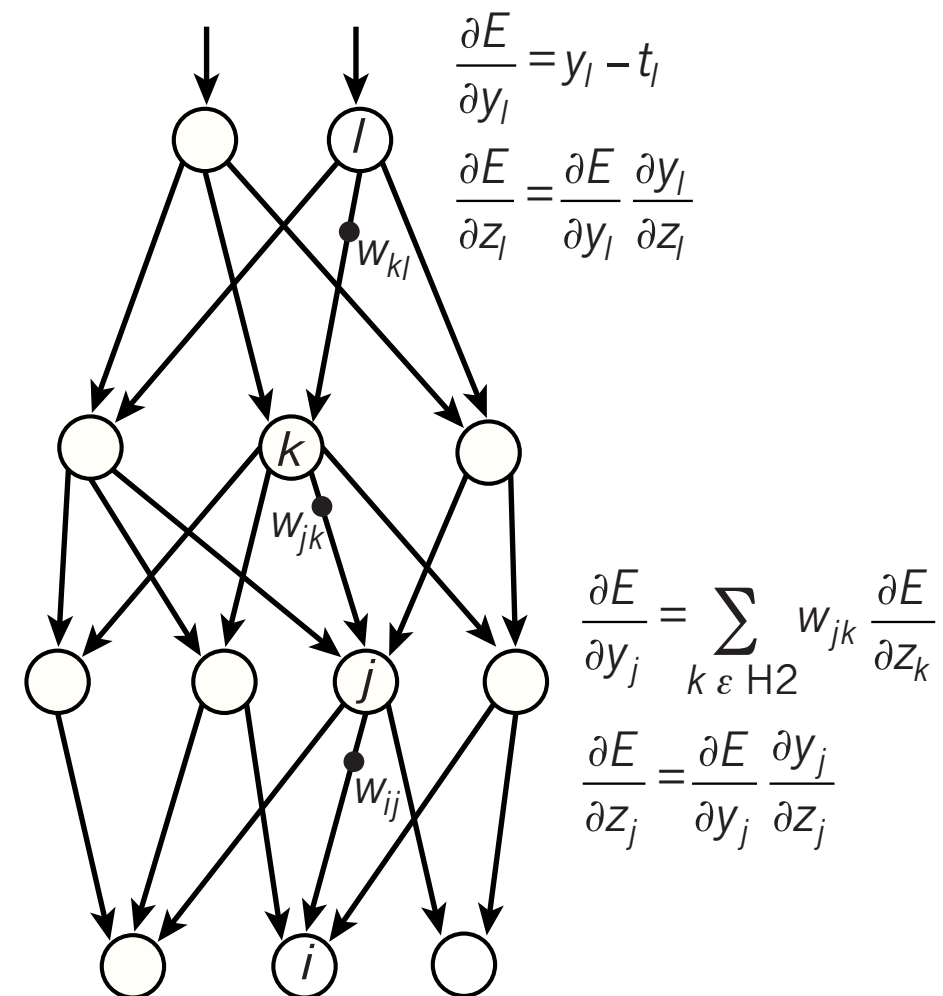
$$y_k = f(z_k)$$

$$z_k = \sum_{j \in H1} w_{jk} y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \in \text{Input}} w_{ij} x_i$$

Backward pass




The forward pass



$F = A_\ell W_\ell + b_\ell,$	$F \in \mathbb{R}^{n \times p_{\ell+1}},$	$A_\ell \in \mathbb{R}^{n \times p_\ell},$	$W_\ell \in \mathbb{R}^{p_\ell \times p_{\ell+1}},$	$b_\ell \in \mathbb{R}^{1 \times p_{\ell+1}},$
$A_\ell = \tanh(H_\ell),$	$A_\ell \in \mathbb{R}^{n \times p_\ell},$	$H_\ell \in \mathbb{R}^{n \times p_\ell},$		
$H_\ell = A_{\ell-1} W_{\ell-1} + b_{\ell-1},$	$H_\ell \in \mathbb{R}^{n \times p_\ell},$	$A_{\ell-1} \in \mathbb{R}^{n \times p_{\ell-1}},$	$W_{\ell-1} \in \mathbb{R}^{p_{\ell-1} \times p_\ell},$	$b_{\ell-1} \in \mathbb{R}^{1 \times p_\ell},$
$A_{\ell-1} = \tanh(H_{\ell-1}),$	$A_{\ell-1} \in \mathbb{R}^{n \times p_{\ell-1}},$	$H_{\ell-1} \in \mathbb{R}^{n \times p_{\ell-1}},$		
$H_{\ell-1} = A_{\ell-2} W_{\ell-2} + b_{\ell-2},$	$H_{\ell-1} \in \mathbb{R}^{n \times p_{\ell-1}},$	$A_{\ell-2} \in \mathbb{R}^{n \times p_{\ell-2}},$	$W_{\ell-2} \in \mathbb{R}^{p_{\ell-2} \times p_{\ell-1}},$	$b_{\ell-2} \in \mathbb{R}^{1 \times p_{\ell-1}},$
$A_{\ell-2} = \tanh(H_{\ell-2}),$	$A_{\ell-2} \in \mathbb{R}^{n \times p_{\ell-2}},$	$H_{\ell-2} \in \mathbb{R}^{n \times p_{\ell-2}},$		
\vdots	\vdots	\vdots	\vdots	\vdots
$H_2 = A_1 W_1 + b_1,$	$H_2 \in \mathbb{R}^{n \times p_2},$	$A_1 \in \mathbb{R}^{n \times p_1},$	$W_1 \in \mathbb{R}^{p_1 \times p_2},$	$b_1 \in \mathbb{R}^{1 \times p_2},$
$A_1 = \tanh(H_1),$	$A_1 \in \mathbb{R}^{n \times p_1},$	$H_1 \in \mathbb{R}^{n \times p_1},$		
$H_1 = X W_0 + b_0,$	$H_1 \in \mathbb{R}^{n \times p_1},$	$X \in \mathbb{R}^{n \times p_0},$	$W_0 \in \mathbb{R}^{p_0 \times p_1},$	$b_0 \in \mathbb{R}^{1 \times p_1},$

The backward pass

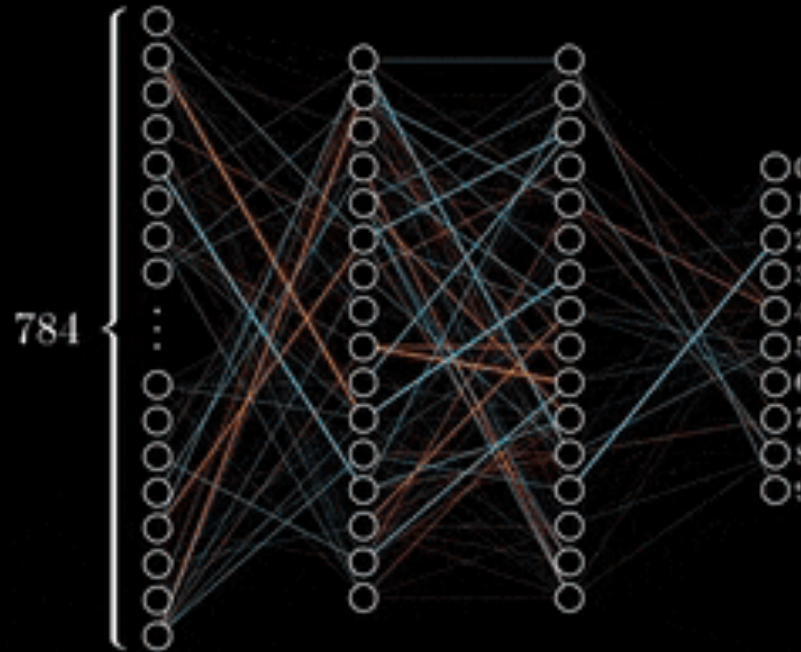
$$\mathcal{L} := \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^{p_{\ell+1}} (F_{i,j} - Y_{i,j})^2 \longrightarrow \nabla_{\theta} \mathcal{L}(\theta)$$



$G_{\ell} = F - Y \in \mathbb{R}^{n \times p_{\ell+1}},$	$\frac{\partial \mathcal{L}}{\partial W_{\ell}} = A_{\ell}^T G_{\ell} \in \mathbb{R}^{p_{\ell} \times p_{\ell+1}},$	$\frac{\partial \mathcal{L}}{\partial b_{\ell}} = \mathbf{1}^T G_{\ell} \in \mathbb{R}^{1 \times p_{\ell+1}},$
$G_{\ell-1} = (1 - A_{\ell} \odot A_{\ell}) \odot (G_{\ell} W_{\ell}^T) \in \mathbb{R}^{n \times p_{\ell}},$	$\frac{\partial \mathcal{L}}{\partial W_{\ell-1}} = A_{\ell-1}^T G_{\ell-1} \in \mathbb{R}^{p_{\ell-1} \times p_{\ell}},$	$\frac{\partial \mathcal{L}}{\partial b_{\ell-1}} = \mathbf{1}^T G_{\ell-1} \in \mathbb{R}^{1 \times p_{\ell}},$
$G_{\ell-2} = (1 - A_{\ell-1} \odot A_{\ell-1}) \odot (G_{\ell-1} W_{\ell-1}^T) \in \mathbb{R}^{n \times p_{\ell-1}},$	$\frac{\partial \mathcal{L}}{\partial W_{\ell-2}} = A_{\ell-2}^T G_{\ell-2} \in \mathbb{R}^{p_{\ell-2} \times p_{\ell-1}},$	$\frac{\partial \mathcal{L}}{\partial b_{\ell-2}} = \mathbf{1}^T G_{\ell-2} \in \mathbb{R}^{1 \times p_{\ell-1}},$
\vdots	\vdots	\vdots
$G_1 = (1 - A_2 \odot A_2) \odot (G_2 W_2^T) \in \mathbb{R}^{n \times p_2},$	$\frac{\partial \mathcal{L}}{\partial W_1} = A_1^T G_1 \in \mathbb{R}^{p_1 \times p_2},$	$\frac{\partial \mathcal{L}}{\partial b_1} = \mathbf{1}^T G_1 \in \mathbb{R}^{1 \times p_2},$
$G_0 = (1 - A_1 \odot A_1) \odot (G_1 W_1^T) \in \mathbb{R}^{n \times p_1},$	$\frac{\partial \mathcal{L}}{\partial W_0} = X^T G_0 \in \mathbb{R}^{p_0 \times p_1},$	$\frac{\partial \mathcal{L}}{\partial b_0} = \mathbf{1}^T G_0 \in \mathbb{R}^{1 \times p_1}.$

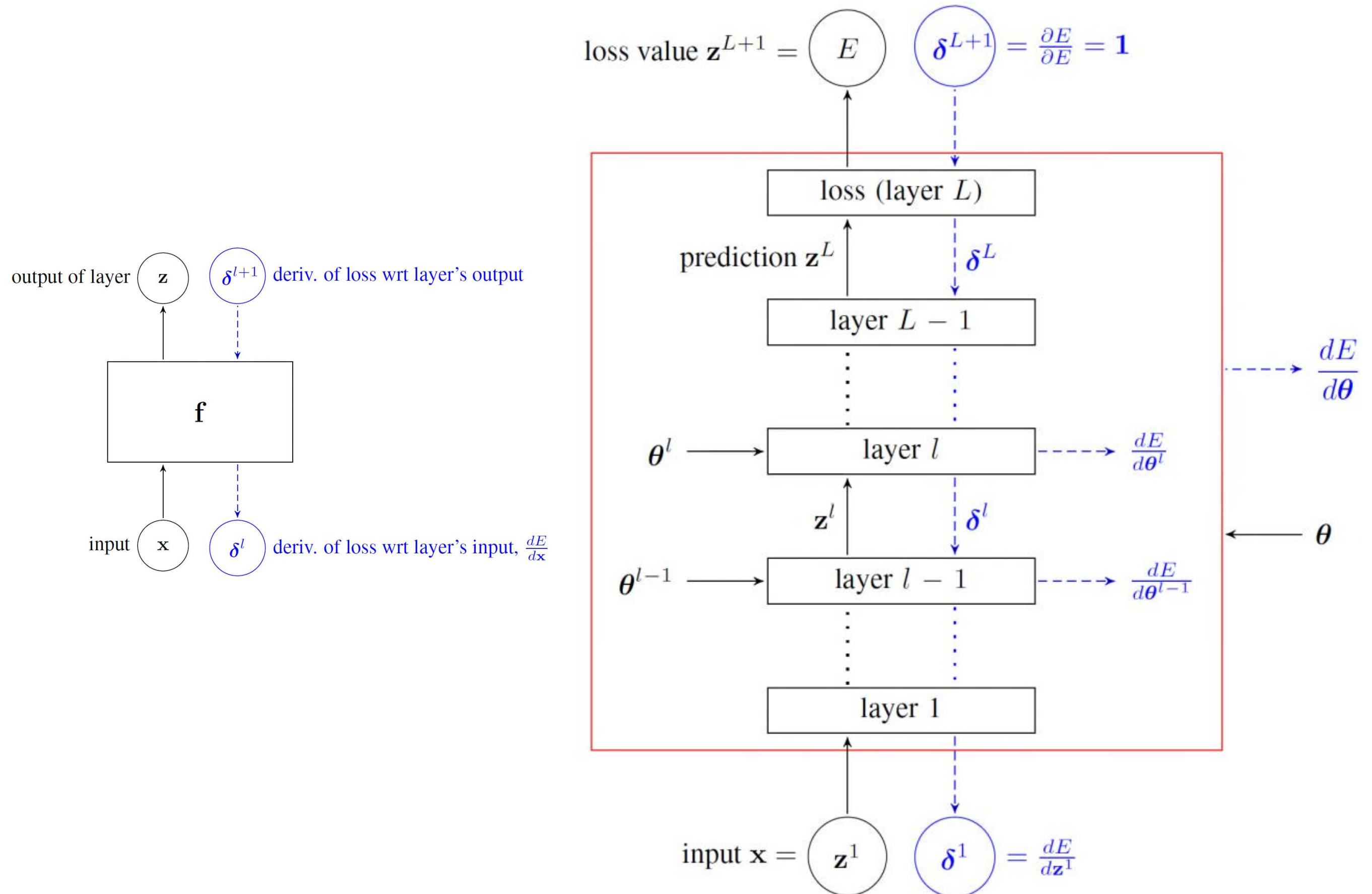
Backpropagation

Training in progress...



$$\begin{aligned}
 G_\ell &= F - Y \in \mathbb{R}^{n \times p_{\ell+1}}, & \frac{\partial \mathcal{L}}{\partial W_\ell} &= A_\ell^T G_\ell \in \mathbb{R}^{p_\ell \times p_{\ell+1}}, & \frac{\partial \mathcal{L}}{\partial b_\ell} &= \mathbf{1}^T G_\ell \in \mathbb{R}^{1 \times p_{\ell+1}}, \\
 G_{\ell-1} &= (1 - A_\ell \odot A_\ell) \odot (G_\ell W_\ell^T) \in \mathbb{R}^{n \times p_\ell}, & \frac{\partial \mathcal{L}}{\partial W_{\ell-1}} &= A_{\ell-1}^T G_{\ell-1} \in \mathbb{R}^{p_{\ell-1} \times p_\ell}, & \frac{\partial \mathcal{L}}{\partial b_{\ell-1}} &= \mathbf{1}^T G_{\ell-1} \in \mathbb{R}^{1 \times p_\ell}, \\
 G_{\ell-2} &= (1 - A_{\ell-1} \odot A_{\ell-1}) \odot (G_{\ell-1} W_{\ell-1}^T) \in \mathbb{R}^{n \times p_{\ell-1}}, & \frac{\partial \mathcal{L}}{\partial W_{\ell-2}} &= A_{\ell-2}^T G_{\ell-2} \in \mathbb{R}^{p_{\ell-2} \times p_{\ell-1}}, & \frac{\partial \mathcal{L}}{\partial b_{\ell-2}} &= \mathbf{1}^T G_{\ell-2} \in \mathbb{R}^{1 \times p_{\ell-1}}, \\
 &\vdots & &\vdots & &\vdots \\
 G_1 &= (1 - A_2 \odot A_2) \odot (G_2 W_2^T) \in \mathbb{R}^{n \times p_2}, & \frac{\partial \mathcal{L}}{\partial W_1} &= A_1^T G_1 \in \mathbb{R}^{p_1 \times p_2}, & \frac{\partial \mathcal{L}}{\partial b_1} &= \mathbf{1}^T G_1 \in \mathbb{R}^{1 \times p_2}, \\
 G_0 &= (1 - A_1 \odot A_1) \odot (G_1 W_1^T) \in \mathbb{R}^{n \times p_1}, & \frac{\partial \mathcal{L}}{\partial W_0} &= X^T G_0 \in \mathbb{R}^{p_0 \times p_1}, & \frac{\partial \mathcal{L}}{\partial b_0} &= \mathbf{1}^T G_0 \in \mathbb{R}^{1 \times p_1}.
 \end{aligned}$$

Modular implementation



Automatic differentiation

Many contemporary algorithms require the evaluation of a derivative of a given differentiable function, f , at a given input value, (x_1, \dots, x_N) , for example a gradient,

$$\left(\frac{\partial f}{\partial x_1} (x_1, \dots, x_N), \dots, \frac{\partial f}{\partial x_N} (x_1, \dots, x_N) \right),$$

or a directional derivative,¹

$$\vec{v}(f) (x_1, \dots, x_N) = \sum_{n=1}^N v_n \frac{\partial f}{\partial x_n} (x_1, \dots, x_N).$$

In its most basic description, automatic differentiation relies on the fact that all numerical computations are ultimately compositions of a finite set of elementary operations for which derivatives are known. Combining the derivatives of the constituent operations through the chain rule gives the derivative of the overall composition. This allows accurate evaluation of derivatives at machine precision with ideal asymptotic efficiency and only a small constant factor of overhead.

Automatic differentiation

The chain rule, forward and reverse accumulation [\[edit\]](#)

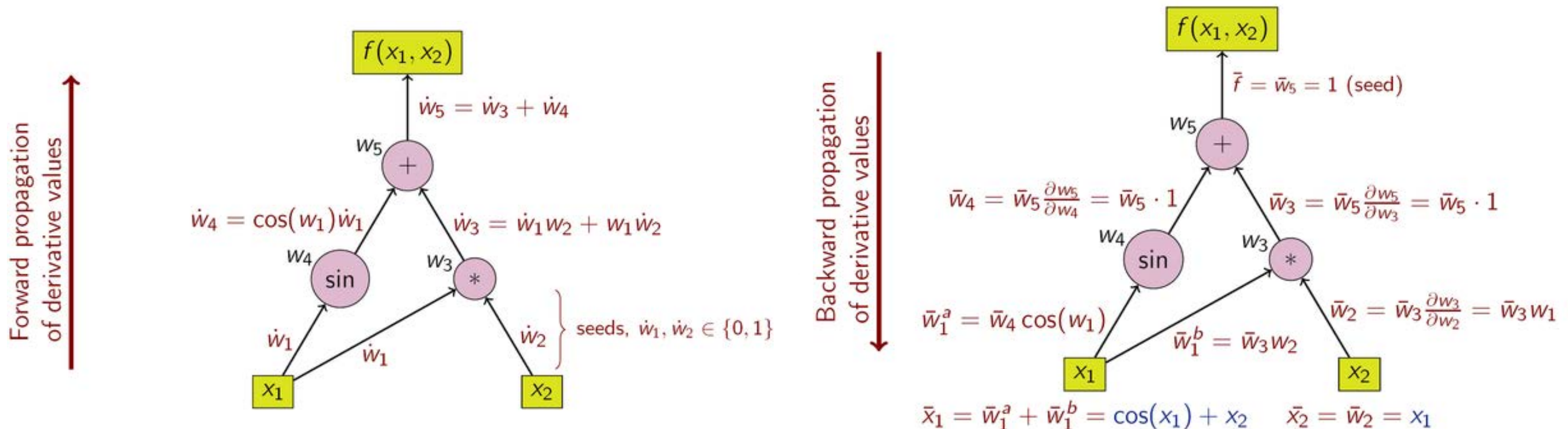
Fundamental to AD is the decomposition of differentials provided by the [chain rule](#). For the simple composition $y = f(g(h(x))) = f(g(h(w_0))) = f(g(w_1)) = f(w_2) = w_3$ the chain rule gives

$$\frac{dy}{dx} = \frac{dy}{dw_2} \frac{dw_2}{dw_1} \frac{dw_1}{dx}$$

Usually, two distinct modes of AD are presented, **forward accumulation** (or **forward mode**) and **reverse accumulation** (or **reverse mode**). Forward accumulation specifies that one traverses the chain rule from inside to outside (that is, first compute dw_1/dx and then dw_2/dx and at last dy/dx), while reverse accumulation has the traversal from outside to inside (first compute dy/dw_2 and then dy/dw_1 and at last dy/dx). More succinctly,

1. **forward accumulation** computes the recursive relation: $\frac{dw_i}{dx} = \frac{dw_i}{dw_{i-1}} \frac{dw_{i-1}}{dx}$ with $w_3 = y$, and,
2. **reverse accumulation** computes the recursive relation: $\frac{dy}{dw_i} = \frac{dy}{dw_{i+1}} \frac{dw_{i+1}}{dw_i}$ with $w_0 = x$.

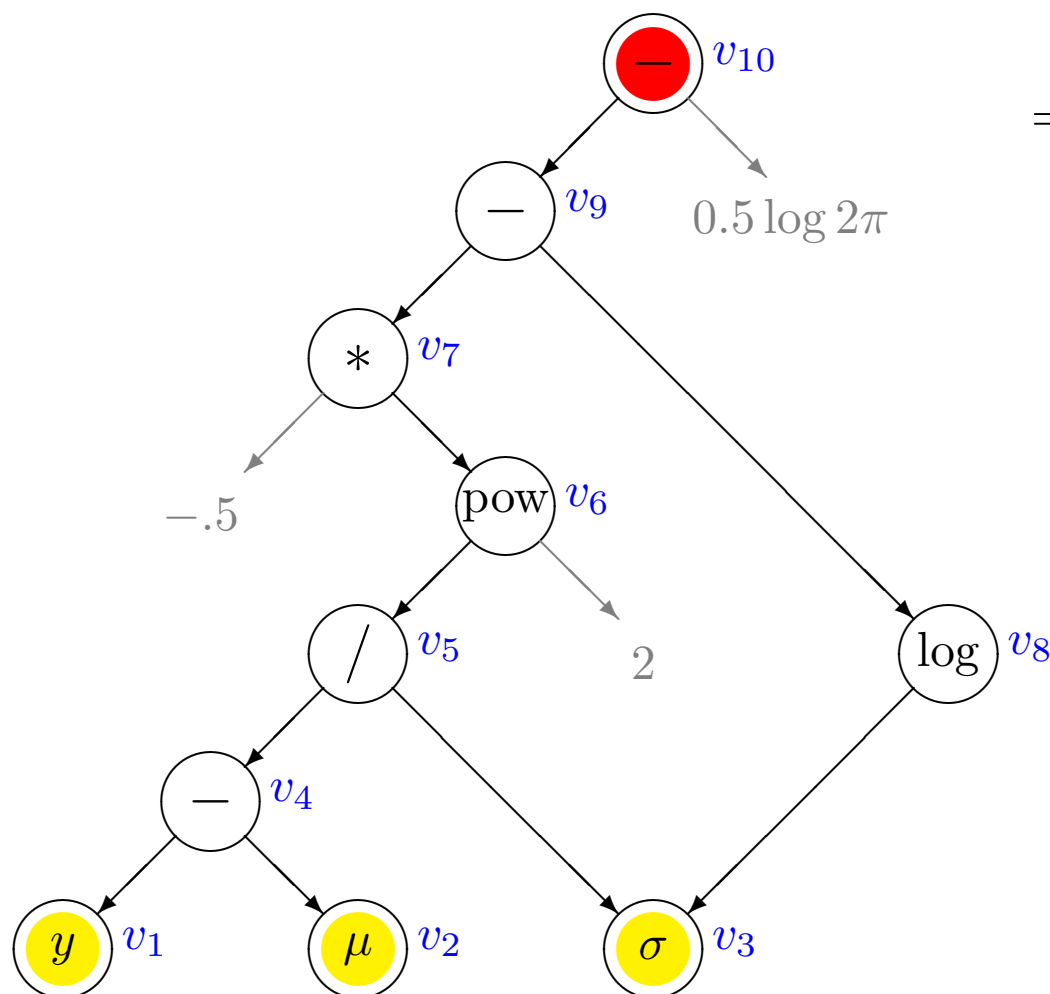
Example $z = f(x_1, x_2) = x_1 x_2 + \sin x_1$



Automatic differentiation

As an example, consider the log of the normal probability density function for a variable y with a normal distribution with mean μ and standard deviation σ ,

$$f(y, \mu, \sigma) = \log(\text{Normal}(y|\mu, \sigma)) = -\frac{1}{2} \left(\frac{y - \mu}{\sigma} \right)^2 - \log \sigma - \frac{1}{2} \log(2\pi) \quad (1)$$



<i>var</i>	<i>value</i>	<i>partials</i>
v_1	y	
v_2	μ	
v_3	σ	
v_4	$v_1 - v_2$	$\partial v_4 / \partial v_1 = 1 \quad \partial v_4 / \partial v_2 = -1$
v_5	v_4 / v_3	$\partial v_5 / \partial v_4 = 1/v_3 \quad \partial v_5 / \partial v_3 = -v_4 v_3^{-2}$
v_6	$(v_5)^2$	$\partial v_6 / \partial v_5 = 2v_5$
v_7	$(-0.5)v_6$	$\partial v_7 / \partial v_6 = -0.5$
v_8	$\log v_3$	$\partial v_8 / \partial v_3 = 1/v_3$
v_9	$v_7 - v_8$	$\partial v_9 / \partial v_7 = 1 \quad \partial v_9 / \partial v_8 = -1$
v_{10}	$v_9 - (0.5 \log 2\pi)$	$\partial v_{10} / \partial v_9 = 1$

Automatic differentiation

It is one of the most useful - and perhaps underused - tools in modern scientific computing!

Applications:

- real-parameter optimization (many good methods are gradient-based)
- sensitivity analysis (local sensitivity = $\partial(\text{result})/\partial(\text{input})$)
- physical modeling (forces are derivatives of potentials; equations of motion are derivatives of Lagrangians and Hamiltonians; etc.)
- probabilistic inference (e.g., Hamiltonian Monte Carlo)
- machine learning
- and who knows how many other scientific computing applications.