Guided Tour of Machine Learning in Finance

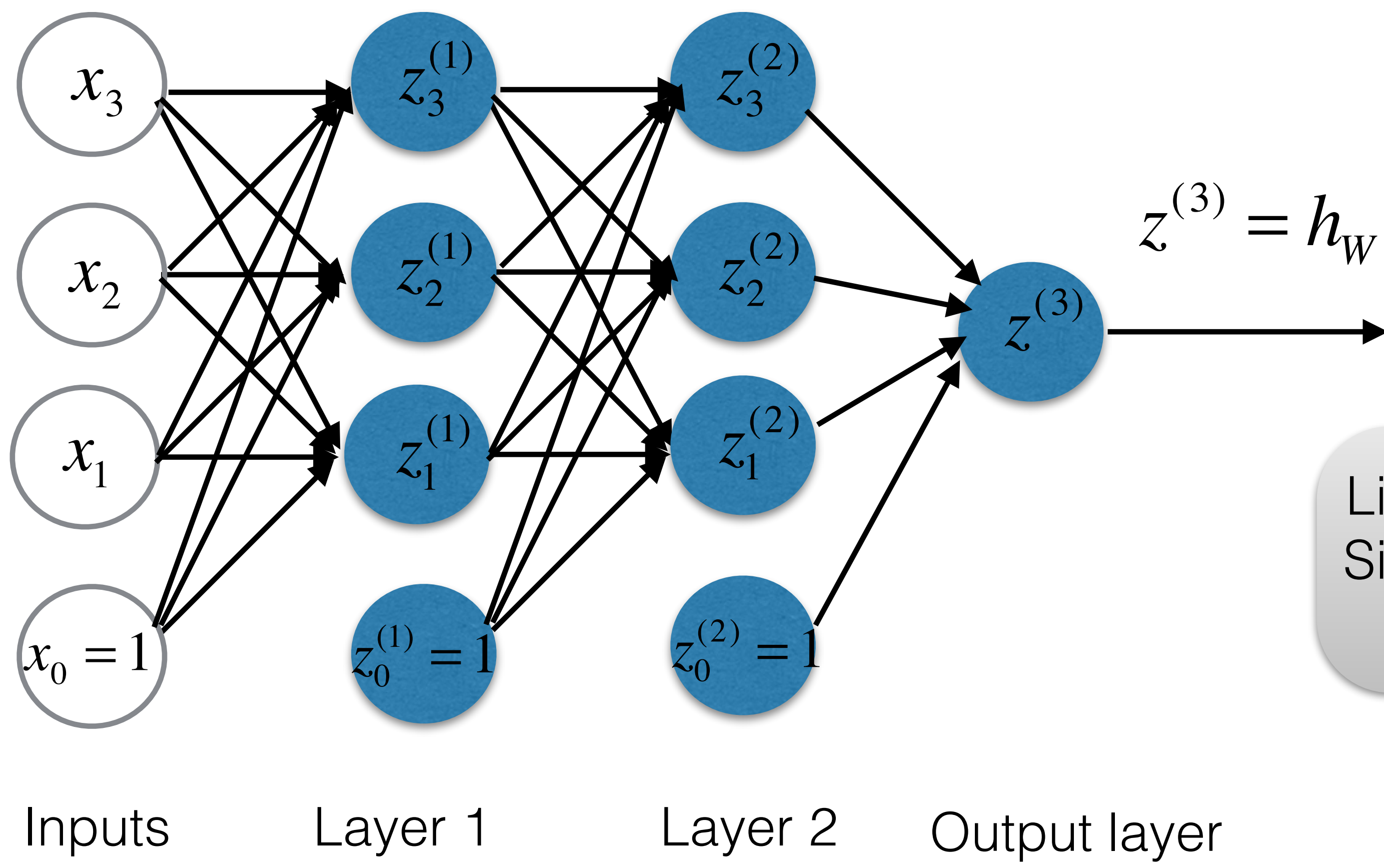
Week 2 - Lesson 1 - part 6:
Gradient Descent for Neural Networks

Igor Halperin

NYU Tandon School of Engineering, 2017

Gradient Descent for Neural Networks

- Gradient descent can be applied not only to one neuron, but to a neural network too:



$$z^{(3)} = h_W(z^{(2)}) = f\left(\sum_{i=0}^{N} W_i z_i^{(2)}\right)$$

Linear unit for regression
Sigmoid for classification

- Trained by **backpropagation** - a groundbreaking algorithm for neural networks proposed by Rumelhart, Hinton and Williams in 1986
- Backpropagation = Gradient Descent with a reverse-mode autodiff

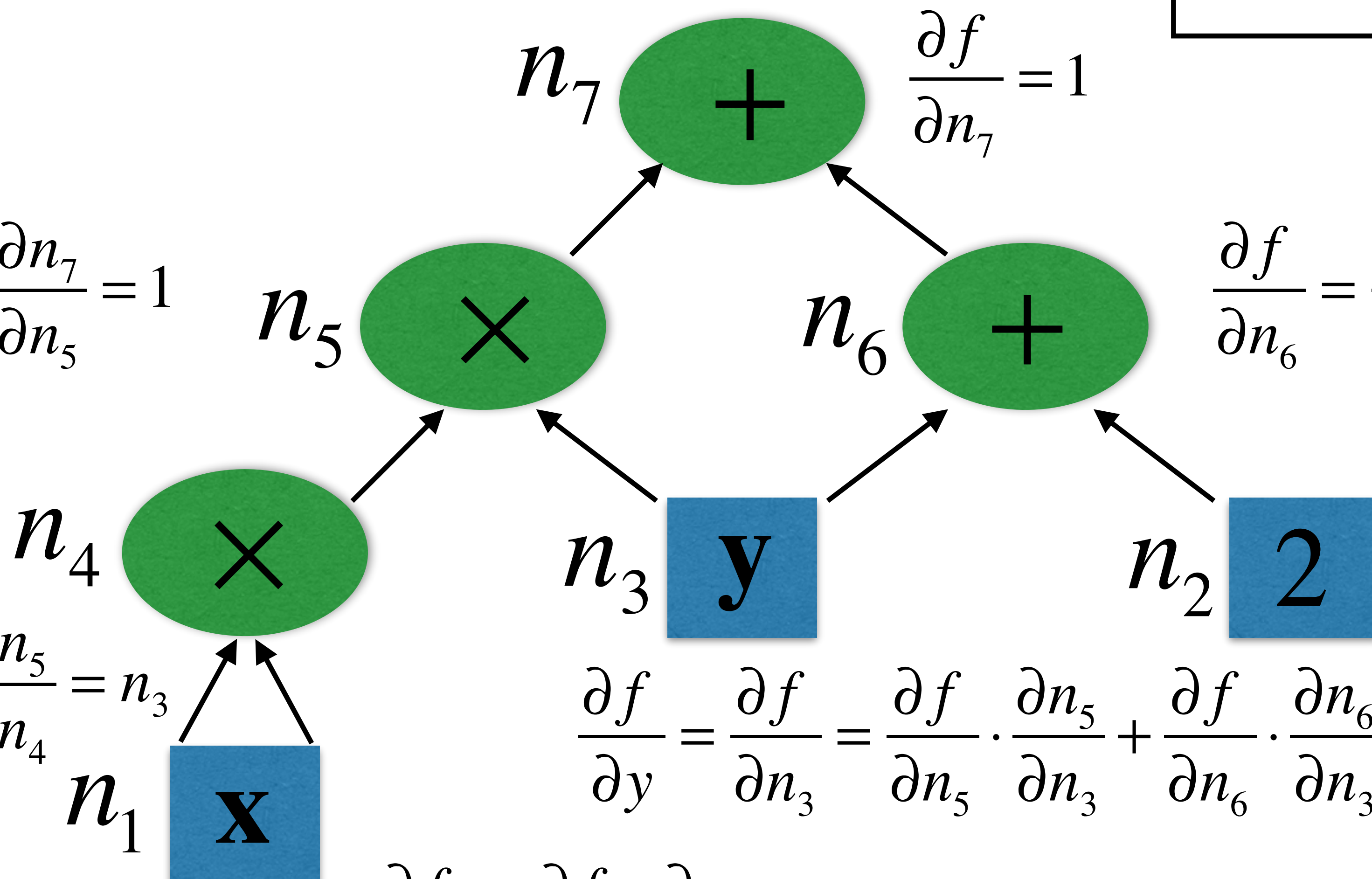
Reverse-Mode Autodiff in TF

Reverse-mode autodiff implements automatic derivatives of any functions:

- **Forward pass** (from inputs to outputs)
- **Backward pass** (from outputs to inputs)
- Use the chain rule for a composite function $f = f\left(n_i(x)\right)$

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_i} \cdot \frac{\partial n_i}{\partial x}}$$

$$\boxed{f(x,y) = x^2 y + y + 2}$$



$$\frac{\partial f}{\partial n_7} = 1$$

$$\frac{\partial f}{\partial n_5} = \frac{\partial f}{\partial n_7} \cdot \frac{\partial n_7}{\partial n_5} = \frac{\partial n_7}{\partial n_5} = 1$$

$$\frac{\partial f}{\partial n_6} = \frac{\partial f}{\partial n_7} \cdot \frac{\partial n_7}{\partial n_6} = \frac{\partial n_7}{\partial n_6} = 1$$

$$\frac{\partial f}{\partial n_4} = \frac{\partial f}{\partial n_5} \cdot \frac{\partial n_5}{\partial n_4} = \frac{\partial n_5}{\partial n_4} = n_3$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial n_3} = \frac{\partial f}{\partial n_5} \cdot \frac{\partial n_5}{\partial n_3} + \frac{\partial f}{\partial n_6} \cdot \frac{\partial n_6}{\partial n_3} = n_4 + 1 = x^2 + 1$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_4} \cdot \frac{\partial n_4}{\partial x} = n_3 \cdot 2x = 2xy$$

Gradient Descent with backpropagation

Gradient Descent with Backpropagation is similar to the reverse-mode autodiff**:**

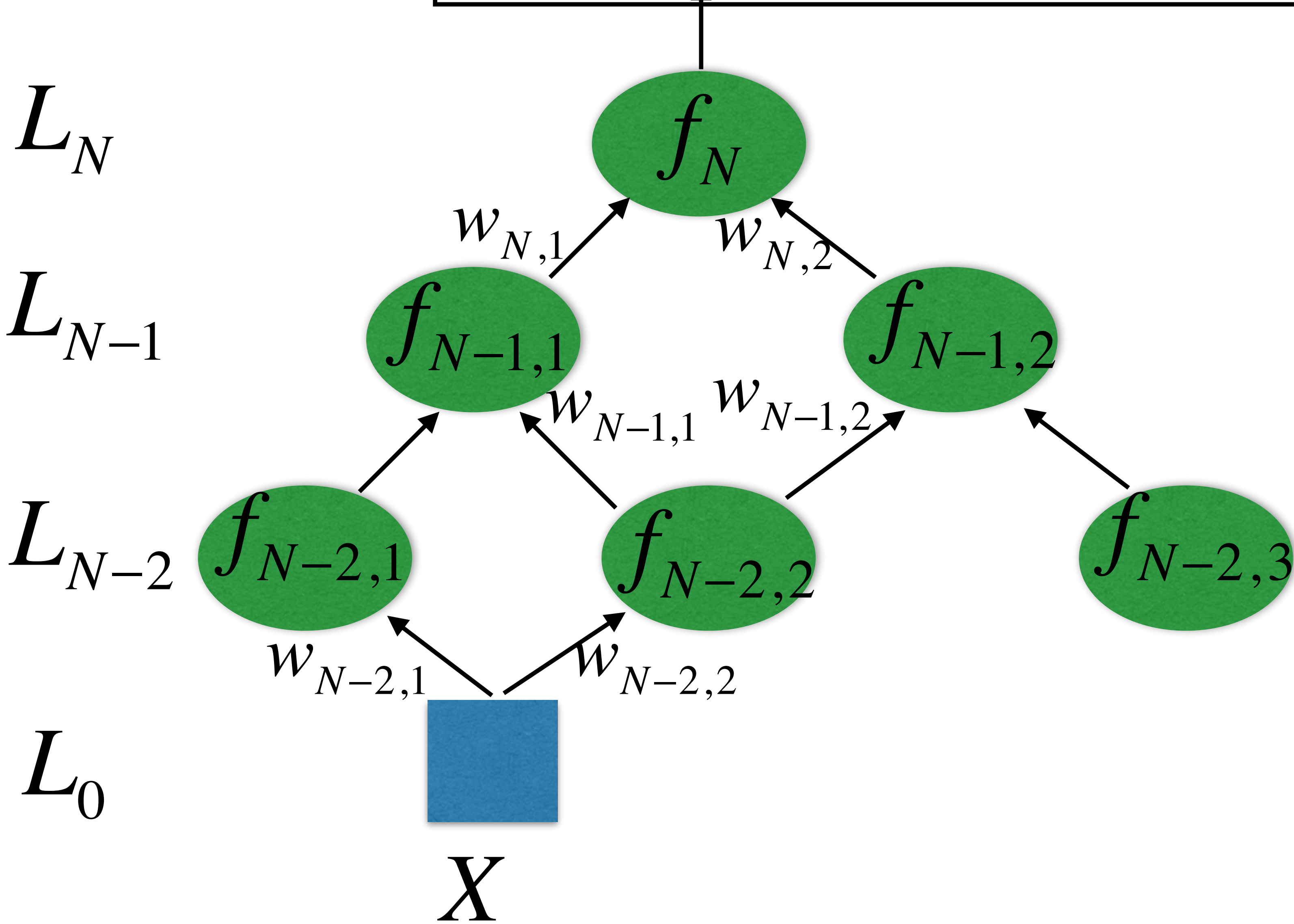
- **Backward pass** (from outputs to inputs)
- Use the chain rule for a composite function $f = \mathrm{MSE}_{train} = \frac{1}{N_{train}} \left\| \hat{\mathbf{Y}}^{train}(w) - \mathbf{Y}^{train} \right\|_2^2$

Gradient Descent with backpropagation

Gradient Descent with Backpropagation is similar to the reverse-mode autodiff**:**

- **Backward pass** (from outputs to inputs)
- Use the chain rule for a composite function $f = \mathrm{MSE}_{train} = \frac{1}{N_{train}} \left\| \hat{\mathbf{Y}}^{train}(w) - \mathbf{Y}^{train} \right\|_2^2$

$$\boxed{\mathrm{f} = \mathrm{f}\Big(\mathrm{f}_{\mathrm{N}}\Big(\mathrm{f}_{\mathrm{N-1}}\big(\mathrm{W}_{\mathrm{N-1}}, \mathrm{f}_{\mathrm{N-1}}(\mathrm{W}_{\mathrm{N-2}}, \mathrm{x}\big), \mathrm{W}_{\mathrm{N}}\Big)\Big)}$$

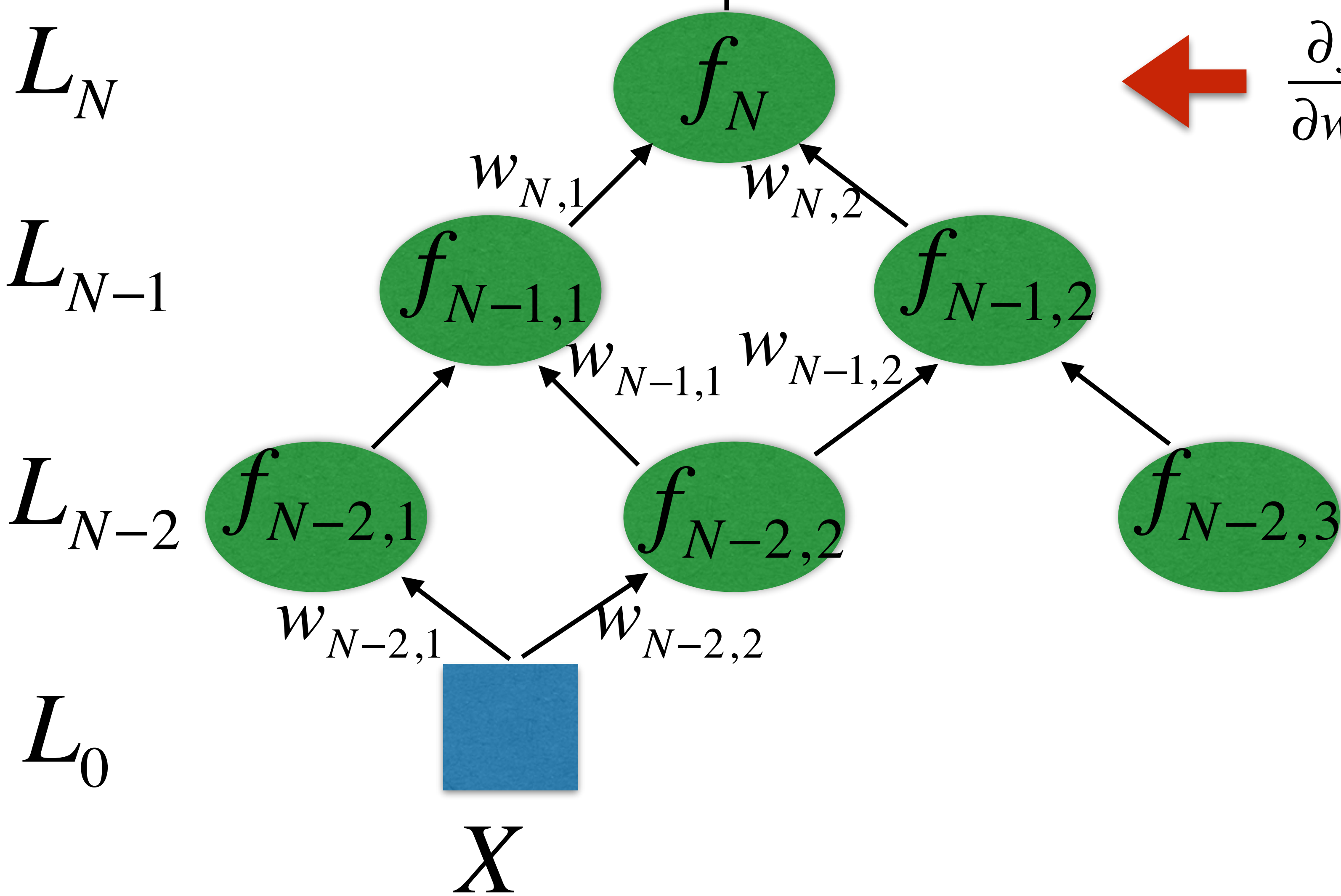


Gradient Descent with backpropagation

Gradient Descent with Backpropagation is similar to the reverse-mode autodiff**:**

- **Backward pass** (from outputs to inputs)
- Use the chain rule for a composite function $f = \mathrm{MSE}_{train} = \frac{1}{N_{train}} \left\| \hat{\mathbf{Y}}^{train}(w) - \mathbf{Y}^{train} \right\|_2^2$

$$\mathrm{f} = \mathrm{f}\Big(\mathrm{f}_{\mathrm{N}}\Big(\mathrm{f}_{\mathrm{N-1}}\Big(\mathrm{W}_{\mathrm{N-1}}, \mathrm{f}_{\mathrm{N-1}}(\mathrm{W}_{\mathrm{N-2}}, \mathrm{x}\Big), \mathrm{W}_{\mathrm{N}}\Big)\Big)$$



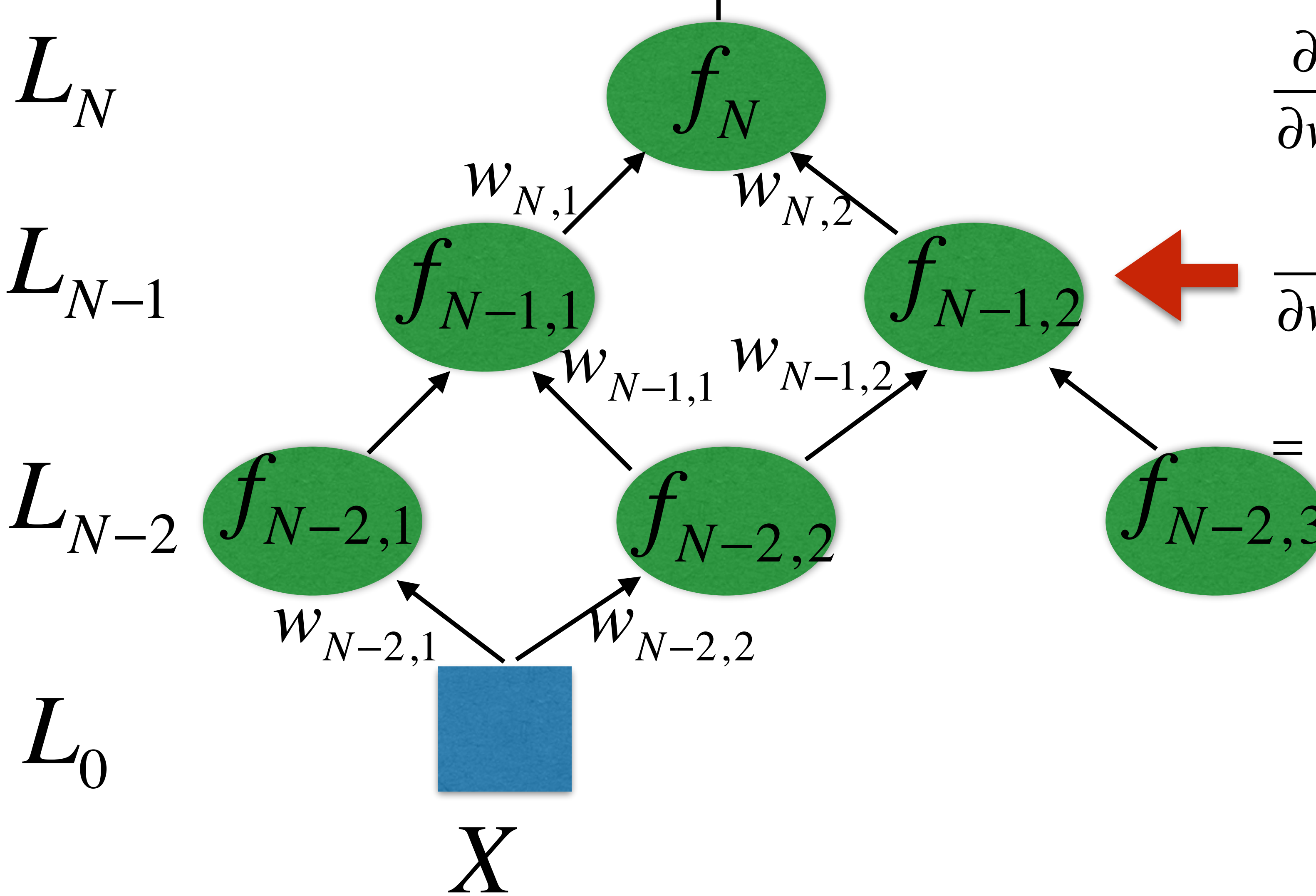
$$\frac{\partial f}{\partial w_{Ni}} = \frac{\partial f}{\partial f_N} \cdot \frac{\partial f_N}{\partial w_{Ni}} \equiv \delta_N \frac{\partial f_N}{\partial w_{Ni}}$$

Gradient Descent with backpropagation

Gradient Descent with Backpropagation is similar to the reverse-mode autodiff**:**

- **Backward pass** (from outputs to inputs)
- Use the chain rule for a composite function $f = \mathrm{MSE}_{train} = \frac{1}{N_{train}} \left\| \hat{\mathbf{Y}}^{train}(w) - \mathbf{Y}^{train} \right\|_2^2$

$$\boxed{\mathrm{f} = \mathrm{f}\left(\mathrm{f}_{\mathrm{N}}\left(\mathrm{f}_{\mathrm{N-1}}\left(\mathrm{W}_{\mathrm{N-1}}, \mathrm{f}_{\mathrm{N-1}}(\mathrm{W}_{\mathrm{N-2}}, \mathrm{x}), \mathrm{W}_{\mathrm{N}}\right)\right)\right)}$$



$$\frac{\partial f}{\partial w_{Ni}} = \frac{\partial f}{\partial f_N} \cdot \frac{\partial f_N}{\partial w_{Ni}} \equiv \delta_N \frac{\partial f_N}{\partial w_{Ni}}$$

$$\frac{\partial f}{\partial w_{N-1,i}} = \frac{\partial f}{\partial f_N} \frac{\partial f_N}{\partial f_{N-1,i}} \cdot \frac{\partial f_{N-1,i}}{\partial w_{N-1,i}}$$

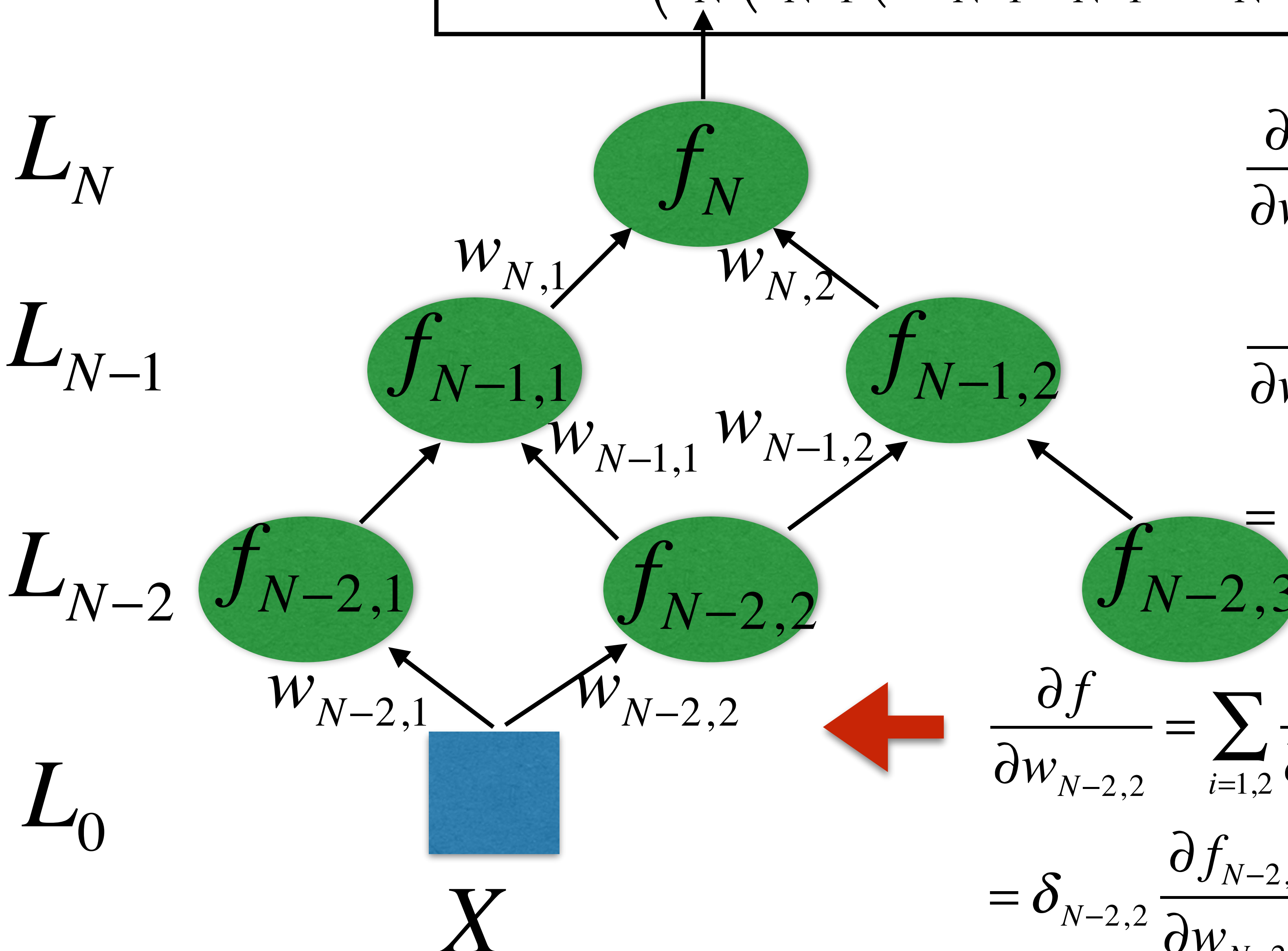
$$= \delta_{N-1,i} \frac{\partial f_{N-1,i}}{\partial w_{N-1,i}}, \quad \delta_{N-1,i} = \delta_N \frac{\partial f_N}{\partial f_{N-1,i}}$$

Gradient Descent with backpropagation

Gradient Descent with Backpropagation is similar to the reverse-mode autodiff**:**

- **Backward pass** (from outputs to inputs)
- Use the chain rule for a composite function $f = \mathrm{MSE}_{train} = \frac{1}{N_{train}} \left\| \hat{\mathbf{Y}}^{train}(w) - \mathbf{Y}^{train} \right\|_2^2$

$$\boxed{\mathrm{f} = \mathrm{f}\Big(\mathrm{f}_{\mathrm{N}}\Big(\mathrm{f}_{\mathrm{N-1}}\Big(\mathrm{W}_{\mathrm{N-1}}, \mathrm{f}_{\mathrm{N-1}}(\mathrm{W}_{\mathrm{N-2}}, \mathrm{x}\Big), \mathrm{W}_{\mathrm{N}}\Big)\Big)}$$



$$\frac{\partial f}{\partial w_{Ni}} = \frac{\partial f}{\partial f_N} \cdot \frac{\partial f_N}{\partial w_{Ni}} \equiv \delta_N \frac{\partial f_N}{\partial w_{Ni}}$$

$$\frac{\partial f}{\partial w_{N-1,i}} = \frac{\partial f}{\partial f_N} \frac{\partial f_N}{\partial f_{N-1,i}} \cdot \frac{\partial f_{N-1,i}}{\partial w_{N-1,i}}$$

$$= \delta_{N-1,i} \frac{\partial f_{N-1,i}}{\partial w_{N-1,i}}, \quad \delta_{N-1,i} = \delta_N \frac{\partial f_N}{\partial f_{N-1,i}}$$

$$\frac{\partial f}{\partial w_{N-2,2}} = \sum_{i=1,2} \frac{\partial f}{\partial f_N} \frac{\partial f_N}{\partial f_{N-1,i}} \frac{\partial f_{N-1,i}}{\partial f_{N-2,2}} \cdot \frac{\partial f_{N-2,2}}{\partial w_{N-2,2}}$$

$$= \delta_{N-2,2} \frac{\partial f_{N-2,2}}{\partial w_{N-2,2}}, \quad \delta_{N-2,2} = \sum_{i=1,2} \delta_{N-1,i} \frac{\partial f_{N-1,i}}{\partial f_{N-2,i}}$$

Stochastic Gradient Descent

Gradient descent minimizes a train error - an approximation to the generalization error

$$E[f] = \int L(f(x), y)\, d\, P(x, y) \Rightarrow E_n[f] = \frac{1}{N}\sum_{i=1}^{N} L\big(f(x_i, w), y_i\big)$$

:

Stochastic Gradient Descent

Gradient descent minimizes a train error - an approximation to the generalization error

$$E[f]=\int L(f(x),y)\,d\,P(x,y)\Rightarrow E_n[f]=\frac{1}{N}\sum_{i=1}^{N}L\big(f(x_i,w),y_i\big)$$

Gradient Descent:

$$w^{(k+1)}=w^{(k)}-\eta\frac{1}{N}\sum_{i=1}^{N}\nabla_w L\big(f(x_i,w^{(k)}),y_i\big)$$

:

Stochastic Gradient Descent

Gradient descent minimizes a train error - an approximation to the generalization error

$$E[f] = \int L(f(x), y)\, d\,P(x,y) \Rightarrow E_n[f] = \frac{1}{N}\sum_{i=1}^{N} L\big(f(x_i, w), y_i\big)$$

Gradient Descent:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i=1}^{N} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

Stochastic Gradient Descent (SGD) with randomly selected **mini-batches** M_k of some fixed size $N_{MB} \ll N$:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i \in M_k} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

Stochastic Gradient Descent

Gradient descent minimizes a train error - an approximation to the generalization error

$$E[f] = \int L(f(x), y)\, d\,P(x,y) \Rightarrow E_n[f] = \frac{1}{N}\sum_{i=1}^{N} L\big(f(x_i, w), y_i\big)$$

Gradient Descent:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i=1}^{N} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

Stochastic Gradient Descent (SGD) with randomly selected mini-batches M_k of some fixed size $N_{MB} \ll N$:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i \in M_k} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

As mini-batches are randomly drawn from a data-generating distribution, **SGD minimizes the generalization error plus noise:**

$$w^{(k+1)} = w^{(k)} - \eta \nabla_w E[f] + \varepsilon^{(k)}$$

Stochastic Gradient Descent

Gradient descent minimizes a train error - an approximation to the generalization error

$$E[f] = \int L(f(x), y)\, d\,P(x,y) \Rightarrow E_n[f] = \frac{1}{N}\sum_{i=1}^{N} L\big(f(x_i, w), y_i\big)$$

Gradient Descent:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i=1}^{N} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

Stochastic Gradient Descent (SGD) with randomly selected mini-batches M_k of some fixed size $N_{MB} \ll N$:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i \in M_k} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

As mini-batches are randomly drawn from a data-generating distribution, SGD minimizes the generalization error plus noise:

$$w^{(k+1)} = w^{(k)} - \eta \nabla_w E[f] + \varepsilon^{(k)}$$

On-line SGD ($N_{MB} = 1$): convergence is slow, but guaranteed under certain conditions on η

Stochastic Gradient Descent

Gradient descent minimizes a train error - an approximation to the generalization error

$$E[f] = \int L(f(x), y)\, d\, P(x,y) \Rightarrow E_n[f] = \frac{1}{N}\sum_{i=1}^{N} L\big(f(x_i, w), y_i\big)$$

Gradient Descent:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i=1}^{N} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

Stochastic Gradient Descent (SGD) with randomly selected mini-batches M_k of some fixed size $N_{MB} \ll N$:

$$w^{(k+1)} = w^{(k)} - \eta \frac{1}{N}\sum_{i \in M_k} \nabla_w L\big(f(x_i, w^{(k)}), y_i\big)$$

As mini-batches are randomly drawn from a data-generating distribution, SGD minimizes the generalization error plus noise:

$$w^{(k+1)} = w^{(k)} - \eta \nabla_w E[f] + \varepsilon^{(k)}$$

On-line SGD ($N_{MB} = 1$): convergence is slow, but guaranteed under certain conditions on η

SGD is one of the most important algorithms for neural networks, and for ML in general!

Control question

Q: Select all correct statements:

1. The Backpropagation algorithm for Neural Networks amounts to Gradient Descent applied to the train error, with a reverse-mode autodiff for a recursive calculation of all derivatives.
2. Stochastic Gradient Descent is a practical version of Gradient Descent, named so in recognition of the fact that numerical algorithms often have some numerical noise due to round-up errors etc., so that outputs of Gradient Descent would always be somewhat random.
3. Stochastic Gradient Descent attempts at a direct minimization of the generalization error, by producing samples from a data generating distribution in the form of mini-batches.
4. The on-line SGD typically converges much faster than the mini-batch SGD, because in this case there is only one term to evaluate in the loss function.

Correct answers: 1, 3.