EECS 442 Discussion 5

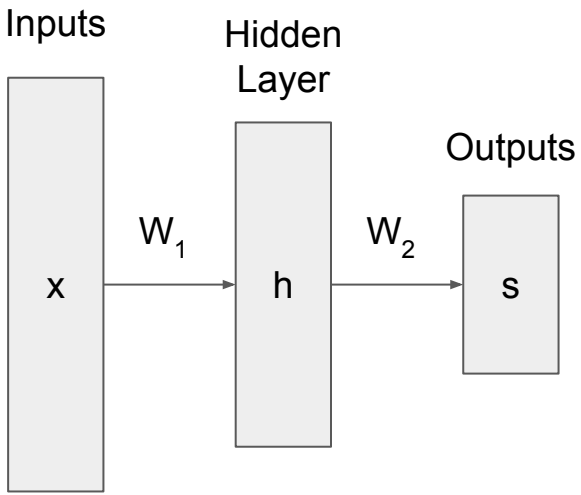
Backpropagation

Discussion Agenda

- Neural Network
- Layers
  - Fully Connected
  - ReLU
- Computational Graphs
  - Forward and Backward Passes
  - Chain Rule
- Backpropagation

Neural Networks

- Models that can learn varying features of data by approximating *almost any* nonlinear function

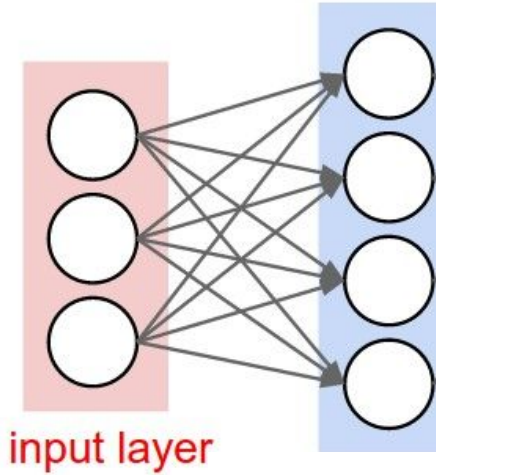


$$f(x) = W_2\, h(W_1 x + b_1) + b_2$$

Fully Connected Layer

For regular neural networks, the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections.

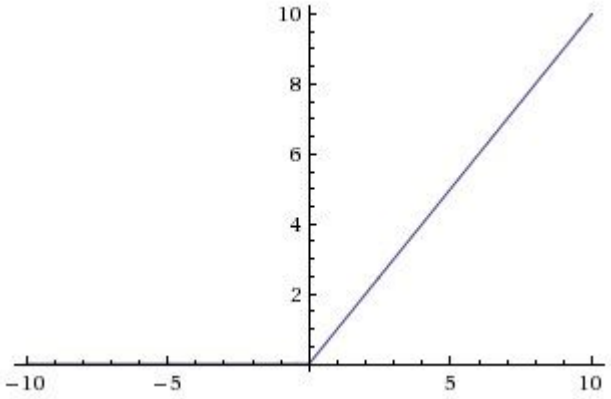
The weight dimension is (3, 4) in the right example.



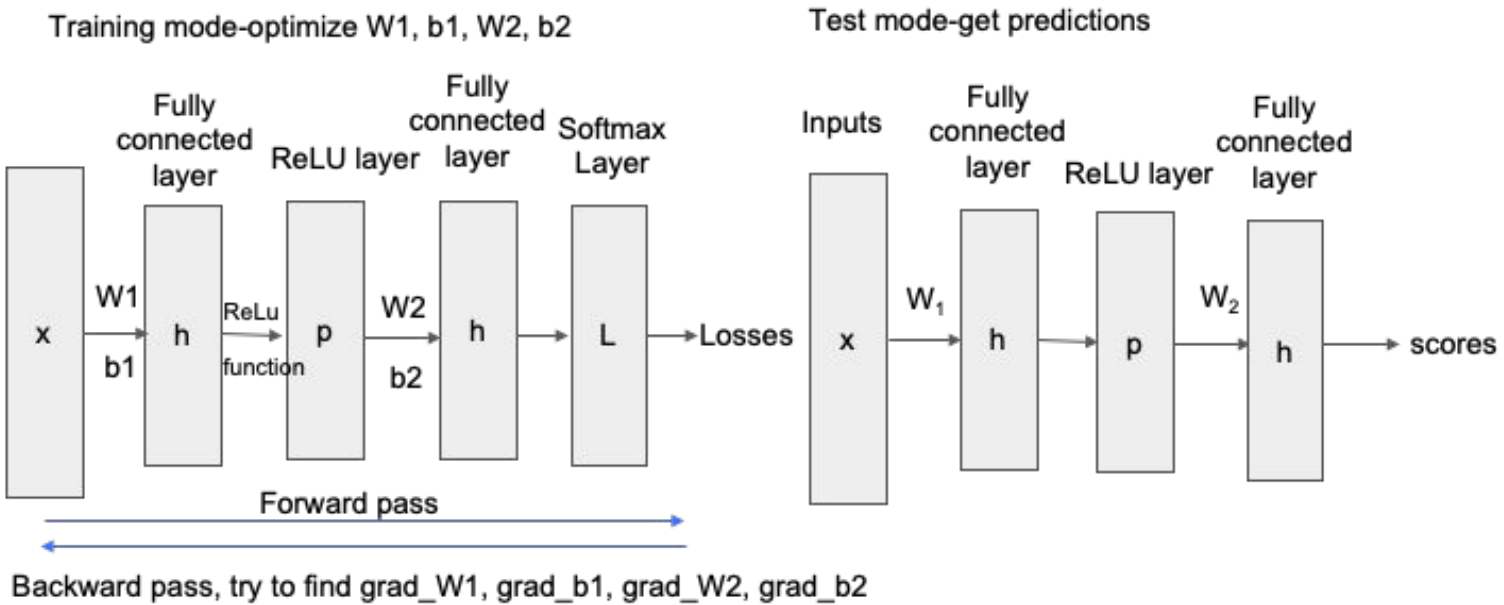
CS231n Convolutional Neural Networks for Visual Recognition

Rectified Linear Unit (ReLU)

- Activation function: introduces non-linearity
- Thresholded at zero
- f(x) = max(0, x)
- Accelerates the convergence of Stochastic Gradient Descent (SGD)
- Simple to implement and fast to compute

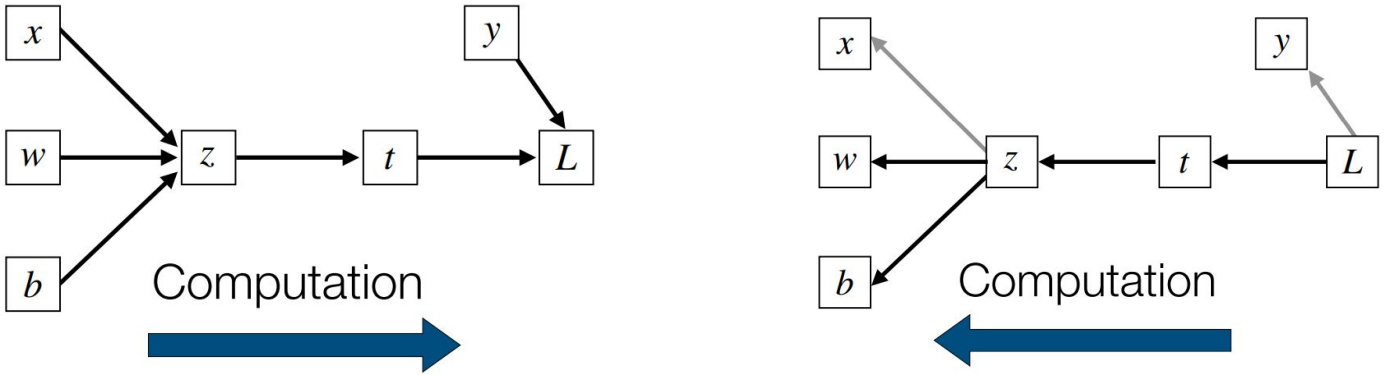


Network Structure for PS4



Computational Graphs

- Computing gradients is infeasible for complex models
  - Need to analytically derive all gradients
- Instead: modularize computation!



Forward and Backward Passes

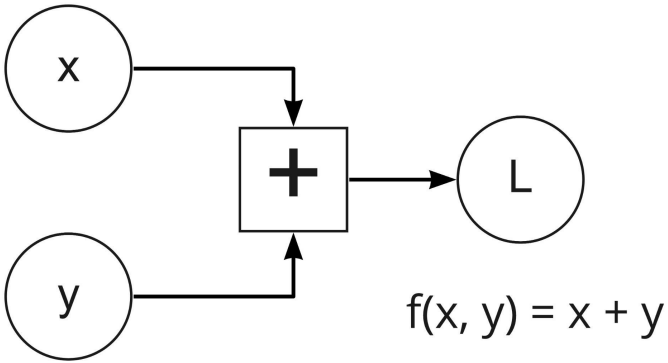
$$f(x, y) = x + y$$

1. **Forward pass:** Compute outputs

$$L = x + y$$

2. **Backward pass:** Compute derivatives

$$\frac{\partial f}{\partial y} = 1 \quad \frac{\partial f}{\partial x} = 1$$



Chain Rule

Break down composed functions into simple expressions

i.e. $f(x, y, z) = (x + y) * z$

Forward Pass: $q = x + y \qquad f = q * z$

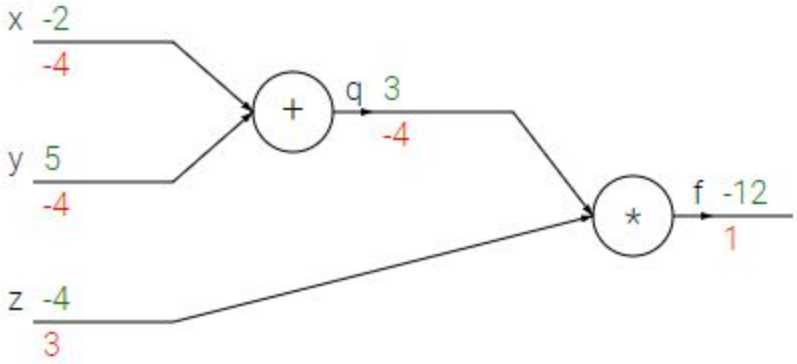
Backward Pass with Chain Rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial y}$$

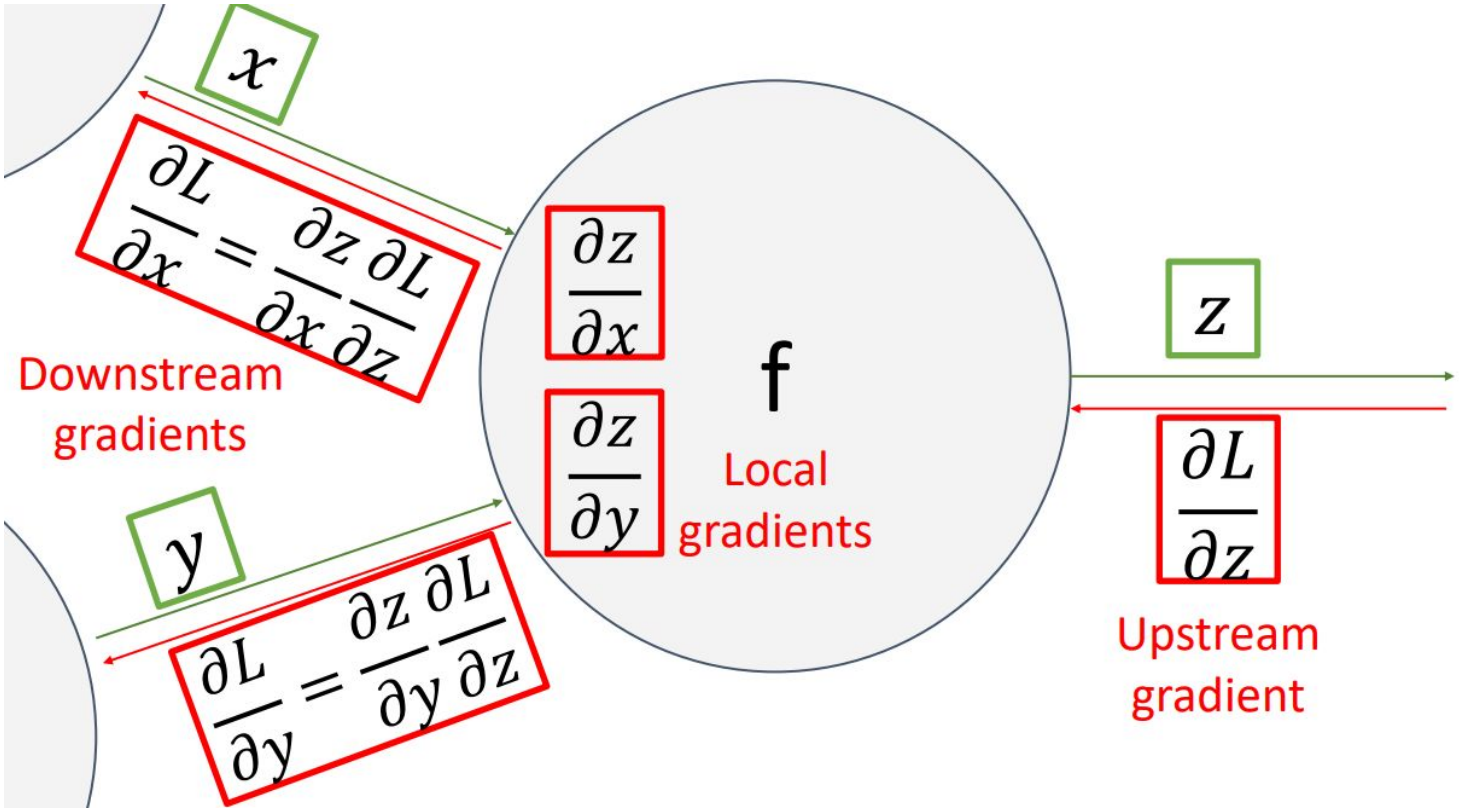
Need to compute gradient for each node

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial z}$$

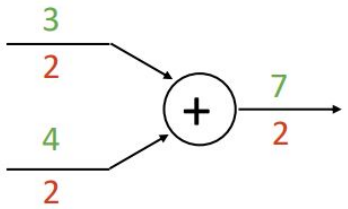


Gradient Computation at One Node

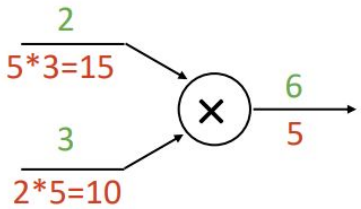


Backpropagation

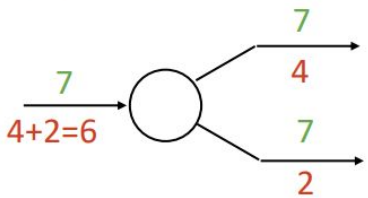
add gate: gradient distributor



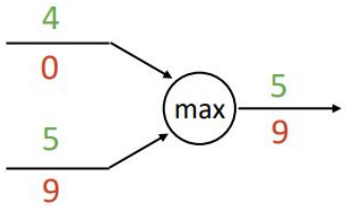
mul gate: “swap multiplier”



copy gate: gradient adder



max gate: gradient router

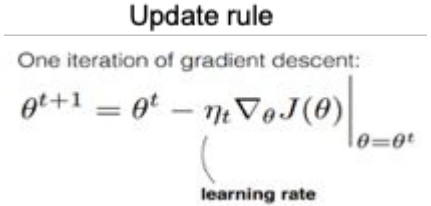


SGD + Momentum

Stochastic Gradient Descent

Select training samples instead of looping over all training examples

$$\nabla J(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \nabla L(x_i, y_i, \theta)$$



where B is a minibatch: a random subset of examples

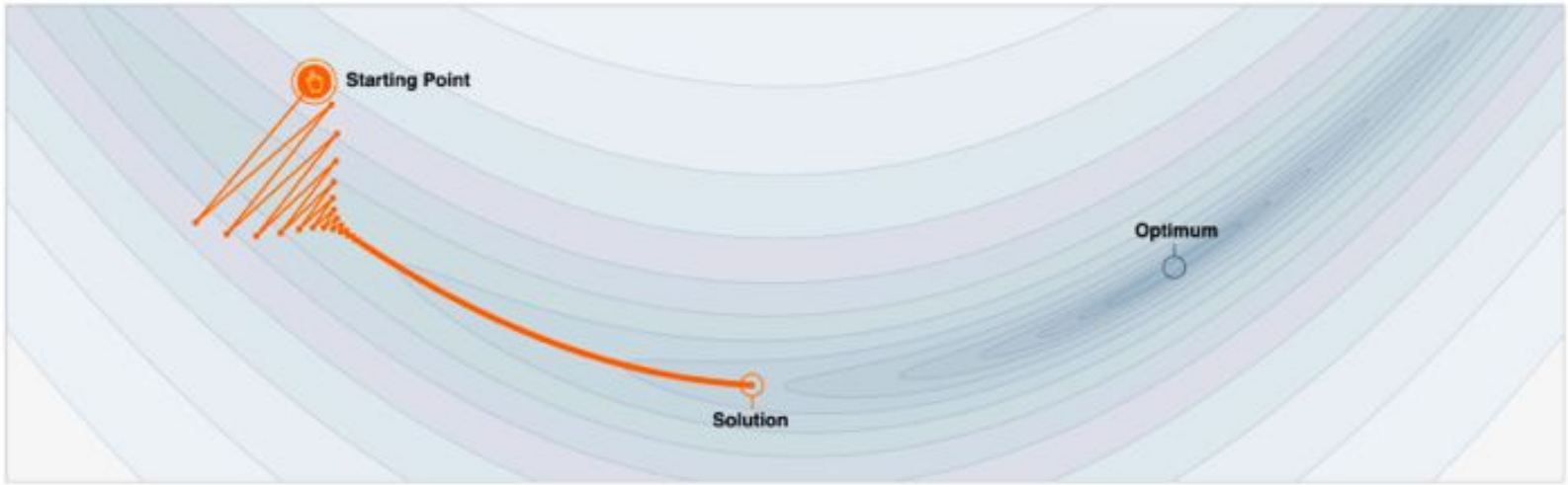
Momentum

SGD with Momentum has the following update rule

$$v \leftarrow dW + \beta * v$$
$$W \leftarrow W - \text{learning\_rate} * v$$

where beta is a scalar in range [0,1], dW is the gradient of the network parameter W, v is velocity initialized as all zeros.

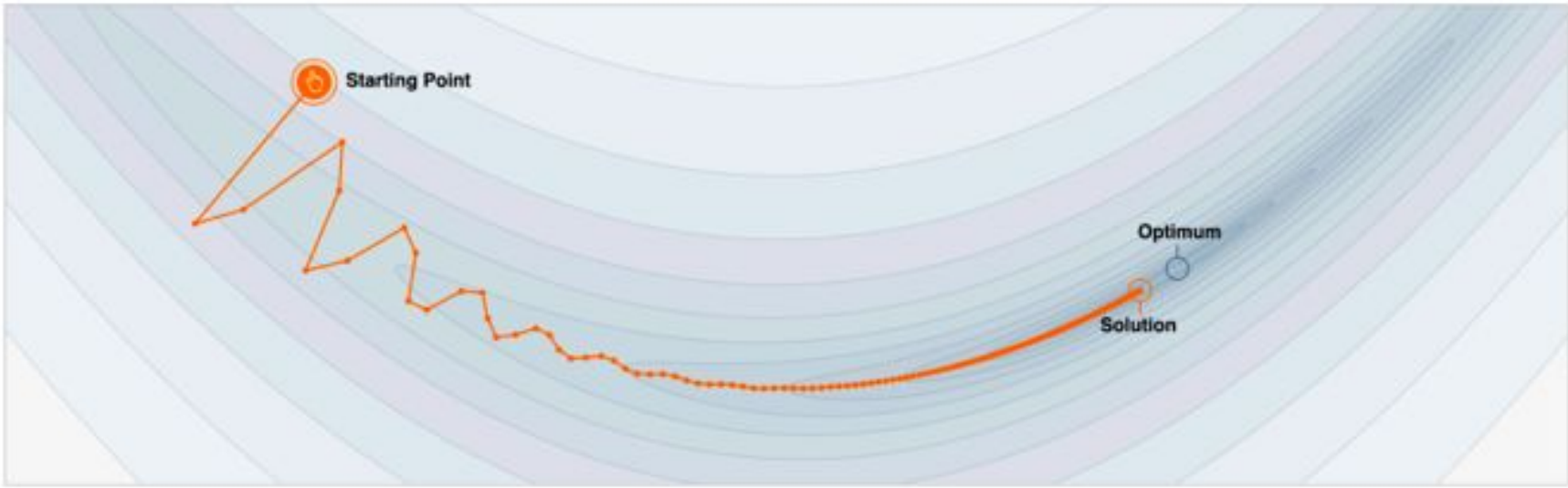
No Momentum



$\beta = 0$ i.e. disable momentum

Source: https://distill.pub/2017/momentum

With Momentum



$$\beta = 0.99$$

Source: https://distill.pub/2017/momentum