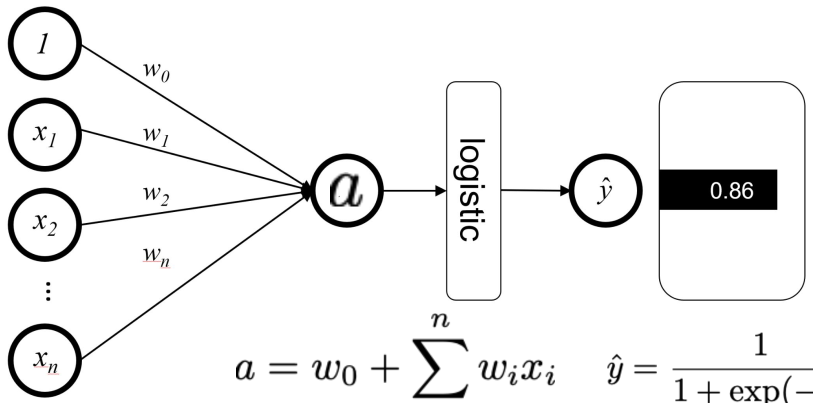
Georgia Tech
College of Computing

CX4240 Spring 2026
Neural Networks: Backpropagation

Bo Dai
School of CSE, Georgia Tech
bodai@cc.gatech.edu

Compact Nonlinear Parametrization: Neural Network

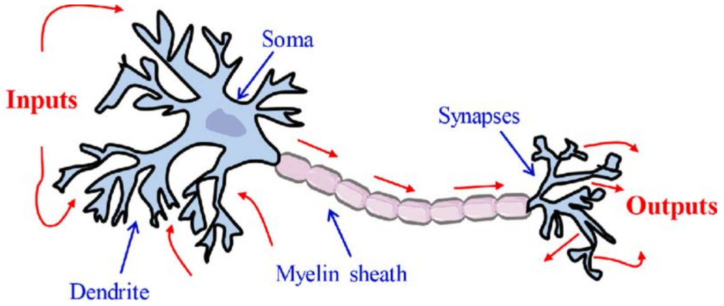
Logistic Regression Revisit

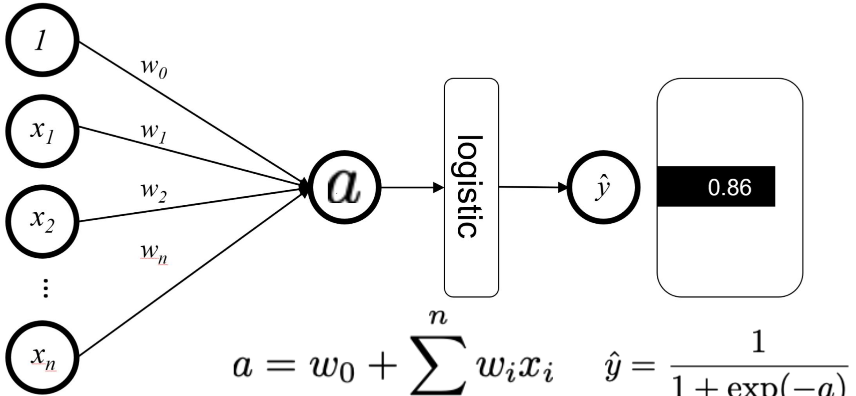


$$a = w_0 + \sum_{i=1}^{n} w_i x_i \qquad \hat{y} = \frac{1}{1+\exp(-a)}$$

Better Nonlinear Parametrization: Neural Network

Neuron $\Longleftrightarrow$ Logistic Regression

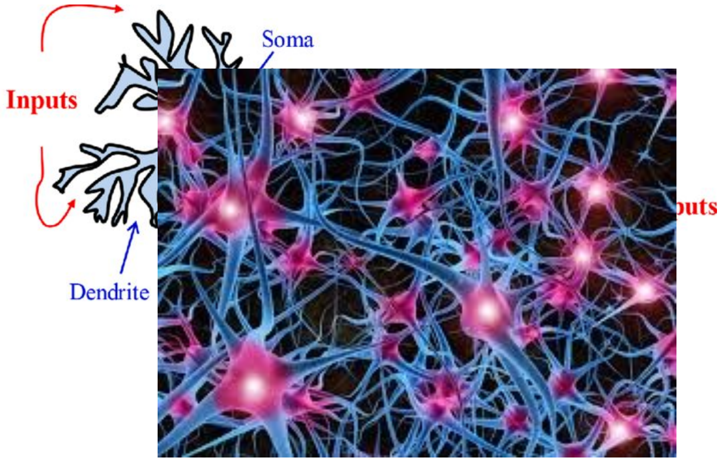


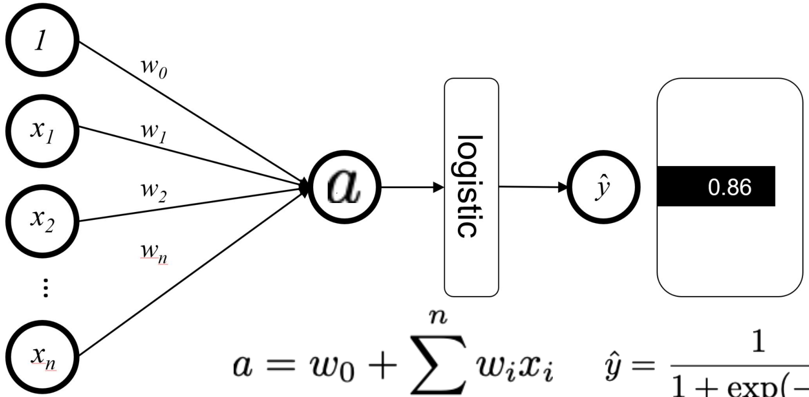


$$a = w_0 + \sum_{i=1}^{n} w_i x_i \qquad \hat{y} = \frac{1}{1 + \exp(-a)}$$

Better Nonlinear Parametrization: Neural Network

Neural Network $\Longleftrightarrow$ Composition of Neurons

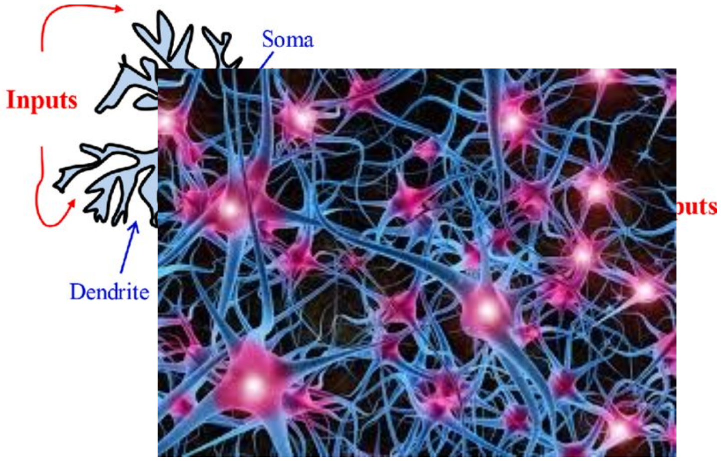


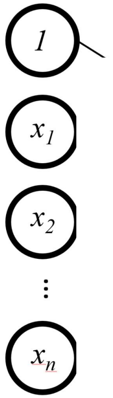


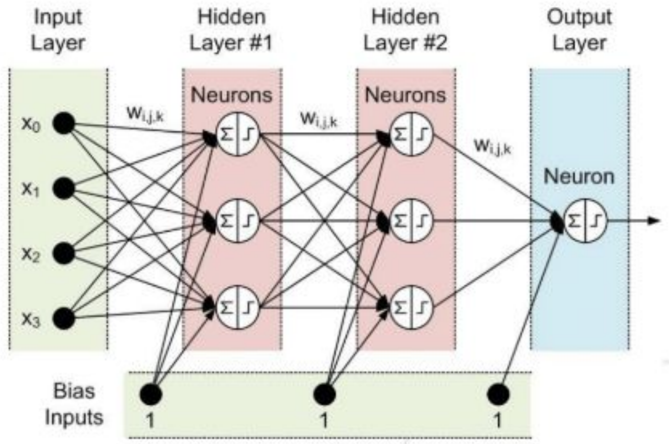
$$a = w_0 + \sum_{i=1}^{n} w_i x_i \qquad \hat{y} = \frac{1}{1 + \exp(-a)}$$

Better Nonlinear Parametrization: Neural Network

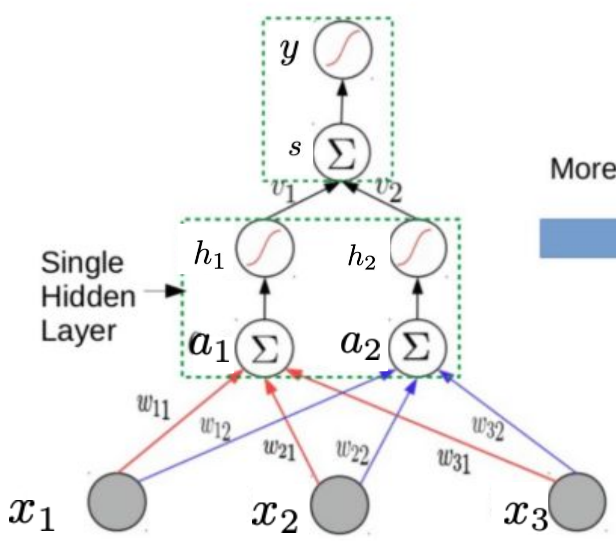
Neural Network $\Longleftrightarrow$ Composition of Neurons



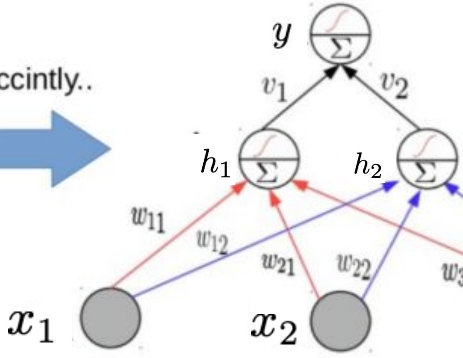




Neural Network Revisit



More succintly..



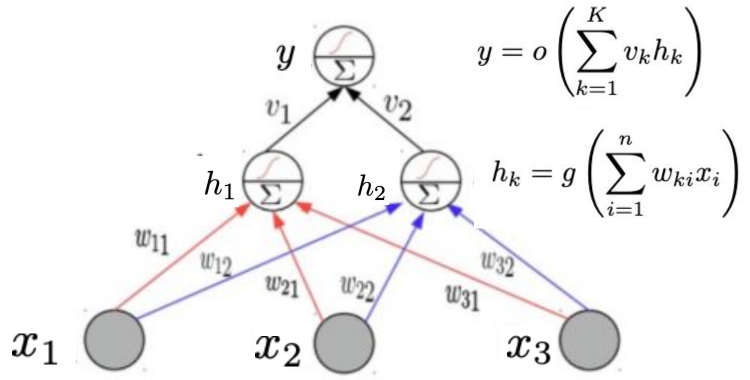
$$y = o\left(\sum_{k=1}^{K} v_k h_k\right)$$

$$h_k = g\left(\sum_{i=1}^{n} w_{ki} x_i\right)$$

Vector Formation

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$

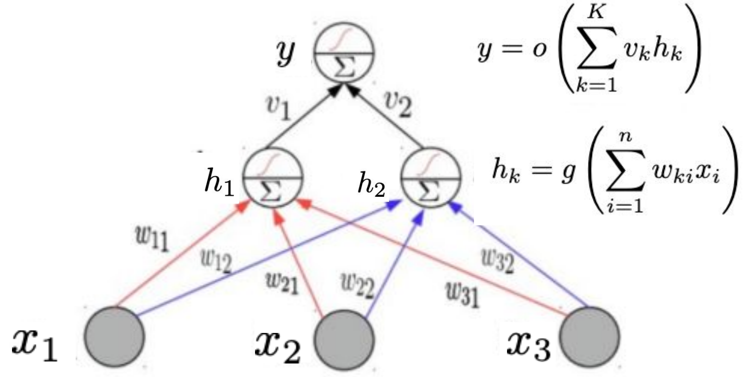

y
$y = o\left(\sum_{k=1}^{K} v_k h_k\right)$
v_1
v_2
h_1
h_2
$h_k = g\left(\sum_{i=1}^{n} w_{ki} x_i\right)$
w_{11}
w_{12}
w_{21}
w_{22}
w_{32}
w_{31}
x_1
x_2
x_3

Vector Formation

$$h = [h_1, h_2]^\top = g(Wx)$$

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$



y
v_1
v_2
h_1
h_2
w_{11}
w_{12}
w_{21}
w_{22}
w_{31}
w_{32}
x_1
x_2
x_3
$y = o\left(\sum_{k=1}^{K} v_k h_k\right)$
$h_k = g\left(\sum_{i=1}^{n} w_{ki} x_i\right)$

Vector Formation

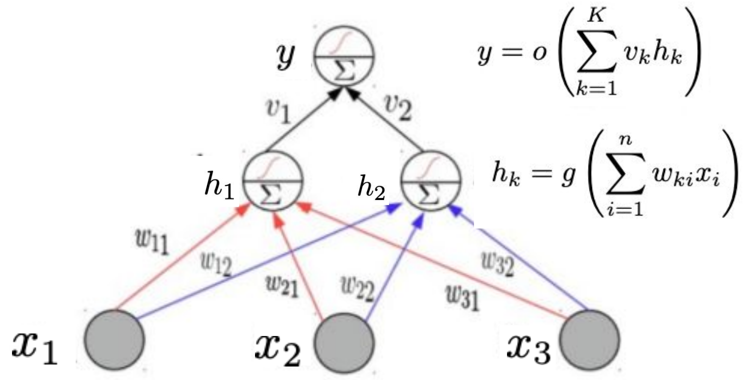
$$y = o(Vh)$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$


y
v_1
v_2
h_1
h_2
w_{11}
w_{12}
w_{21}
w_{22}
w_{31}
w_{32}
x_1
x_2
x_3
$y = o\left(\sum_{k=1}^{K} v_k h_k\right)$
$h_k = g\left(\sum_{i=1}^{n} w_{ki} x_i\right)$

Vector Formation

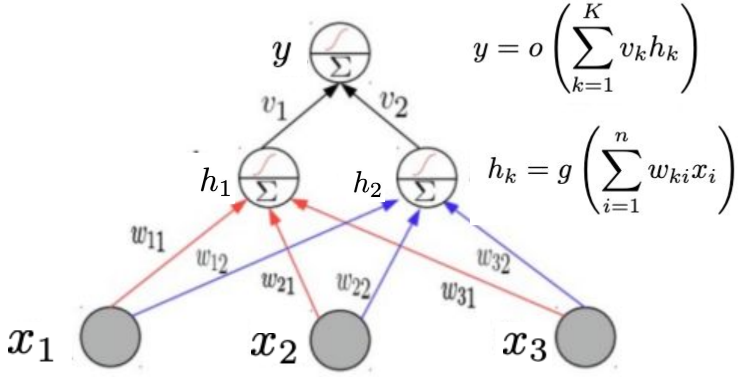
$$y = o(Vg(Wx))$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

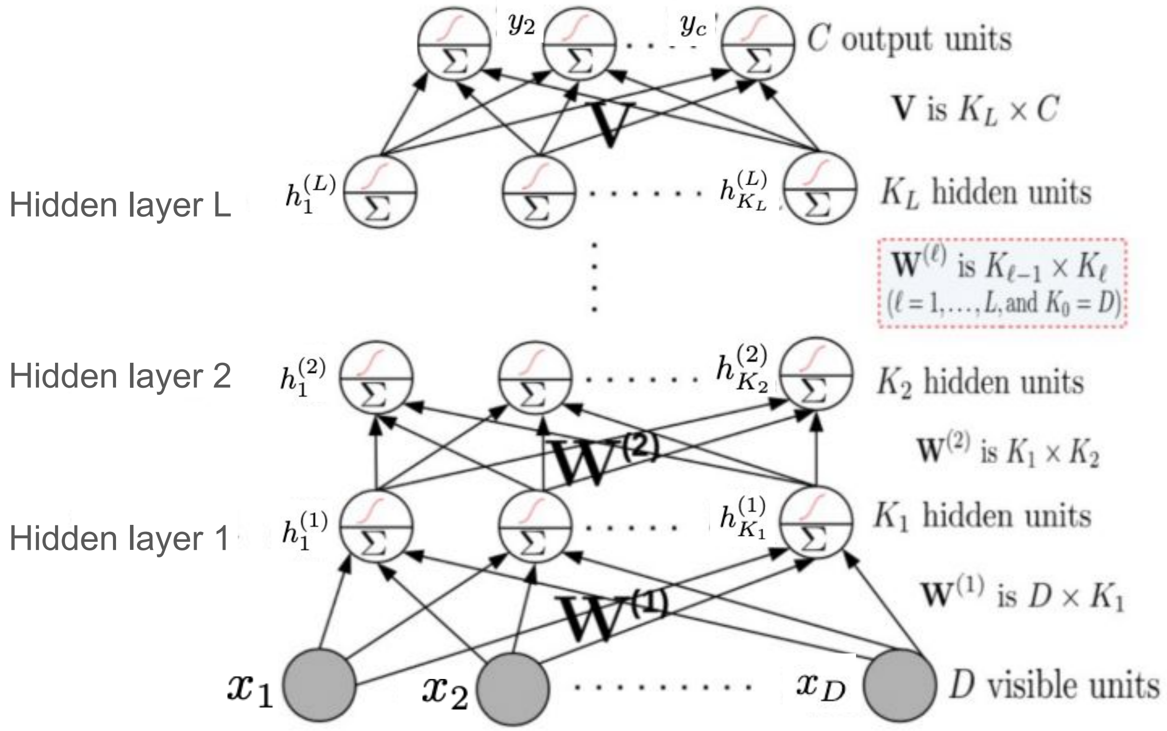
$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$

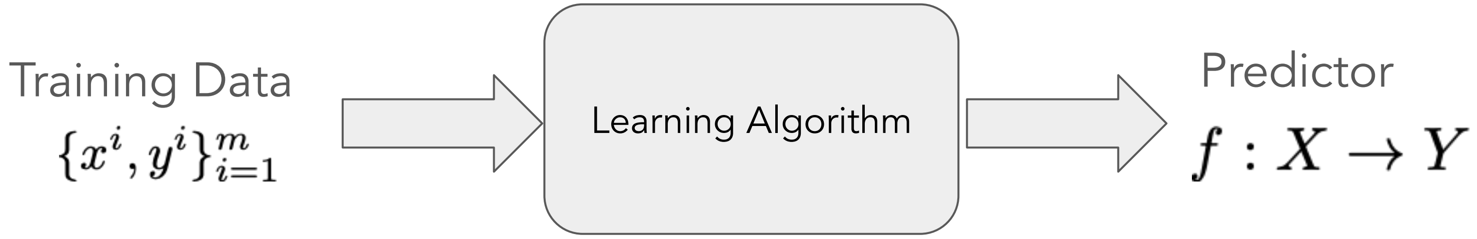

y
v_1
v_2
$y = o\left(\sum_{k=1}^{K} v_k h_k\right)$
h_1
h_2
$h_k = g\left(\sum_{i=1}^{n} w_{ki} x_i\right)$
w_{11}
w_{12}
w_{21}
w_{22}
w_{32}
w_{31}
x_1
x_2
x_3

Multi-Layer Perception

$$y = f_L(W_L f_{L-1}(W_{L-1} \dots f_1(W_1 x)))$$



Regression Algorithms



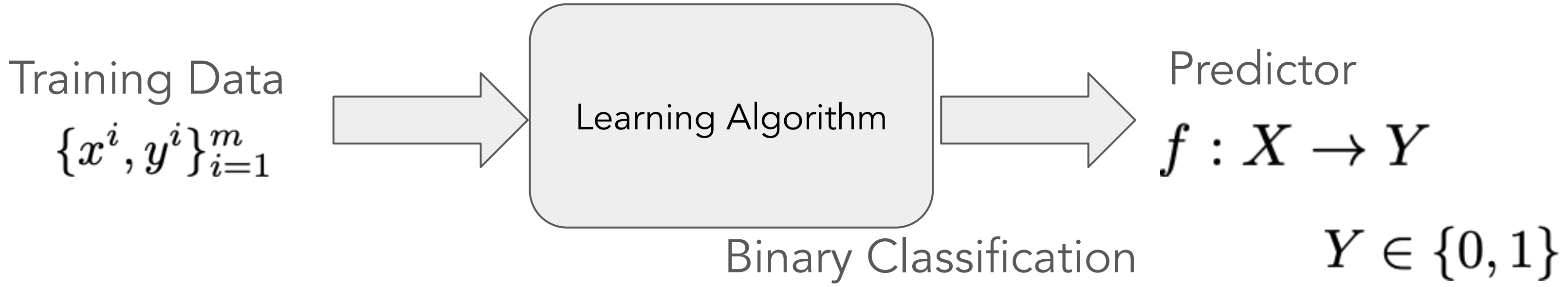
$$Y \in \mathbb{R}$$

Linear Regression Pipeline

1. Build probabilistic models:
   Gaussian Distribution + Neural Network
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) GD

$$y = o(V g(W x))$$

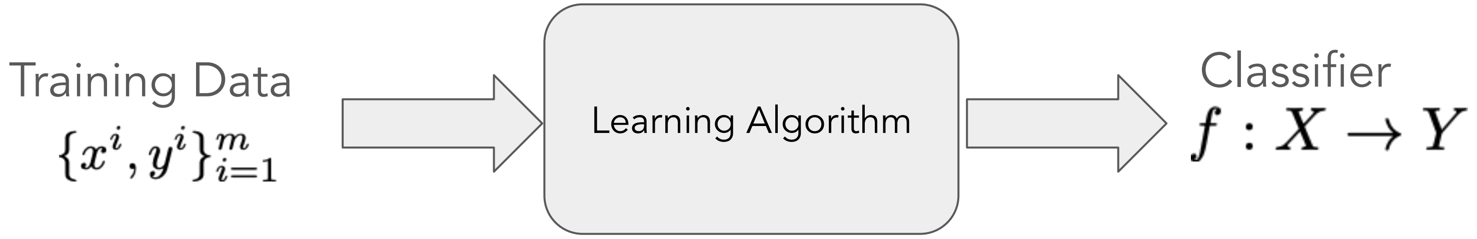
Binary Classification Algorithms



Binary Logistic Regression Pipeline

1. Build probabilistic models:
   Bernoulli Distribution + Neural Network $y = o(Vg(Wx))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Multiclass Logistic Regression Algorithms



Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Multiclass Logistic Regression Pipeline

1. Build probabilistic models:
   Categorical Distribution + Neural Network $y = o(V g(W x))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Select Optimizer

$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

$$L(\theta) = \sum_{i=1}^{m} \ell(x^i, y^i, \theta) + \lambda\Omega(\theta)$$

$$\theta = [V, W]$$

Select Optimizer

$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

$$L(\theta) = \sum_{i=1}^{m} \ell(x^i, y^i, \theta) + \lambda\Omega(\theta)$$

$$\theta = [V, W]$$

$$\ell(x^i, y^i, \theta) = -y^i \log \sigma(o(Vg(Wx^i))) \\ -(1 - y^i) \log(1 - \sigma(o(V_j g(Wx^i))))$$

Select Optimizer

$$L(\theta) = \sum_{i=1}^{m} \ell(x^i, y^i, \theta) + \lambda\Omega(\theta)$$

$$\theta = [V, W]$$

$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

$$\ell(x^i, y^i, \theta) = -y^i \log \sigma(o(Vg(Wx^i))) \\ -(1 - y^i) \log(1 - \sigma(o(V_j g(Wx^i))))$$

$$\ell(x^i, y^i, \theta) = -\sum_{j=1}^{k} y^i \log \frac{\exp(o(V_j g(Wx^i)))}{\sum_{c=1}^{k} \exp(o(V_c g(Wx^i)))}$$

Select Optimizer

$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

$$L(\theta) = \sum_{i=1}^{m} \ell(x^i, y^i, \theta) + \lambda\Omega(\theta)$$

$$\theta = [V, W]$$

$$\ell(x^i, y^i, \theta) = -y^i \log \sigma(o(Vg(Wx^i))) \\ -(1 - y^i) \log(1 - \sigma(o(V_j g(Wx^i))))$$

$$\ell(x^i, y^i, \theta) = -\sum_{j=1}^{k} y^i \log \frac{\exp(o(V_j g(Wx^i)))}{\sum_{c=1}^{k} \exp(o(V_c g(Wx^i)))}$$

- (Stochastic) Gradient Descent

(Stochastic) Gradient Descent

- Initialize parameter θ^0
- Sample $\{x^i, y^i\}_{i=1}^B$
- Do $$\theta^{t+1} \leftarrow \theta^t - \eta \sum_{i=1}^{B} \nabla_\theta \ell(x^i, y^i, \theta^t) - (\lambda \nabla \Omega(\theta^t))$$

Chain Rule

- A composite function is the combination of two functions: a function that takes as input the output of another function

$$h(\theta) = g(f(\theta))$$

$$\begin{tikzcd} \theta \arrow[r] & \boxed{f(\theta)} \arrow[r] & \boxed{g(\theta)} \arrow[r] & \text{output} \; h(\theta) \end{tikzcd}$$

$$\text{E.g, } f(\theta) = 2\theta + 1,\ g(\theta) = \theta^4,\ h(\theta) = (2\theta + 1)^4$$

Let's call $u = f(\theta)$ the output of the inner function $\rightarrow h(\theta) = g(u)$

$$h' = \frac{dh}{d\theta} = \frac{dh}{du}\frac{du}{d\theta}$$

$$\frac{dh}{du} = 4(2\theta + 1)^3 \qquad \frac{du}{d\theta} = 2 \qquad h' = 8(2\theta + 1)^3$$

Derivative of outer part of *h(θ)* Derivative of inner part of *h(θ)*

Chain Rule

$h(\theta) = g(f(\theta))$

$u = f(\theta)$

where $\theta \in \mathbb{R}^m, \mathbf{u} \in \mathbb{R}^n$

$h : \mathbb{R}^m \to \mathbb{R}$

$f : \mathbb{R}^m \to \mathbb{R}^n$ Vector function

$g : \mathbb{R}^n \to \mathbb{R}$

The inner vector function f maps m inputs to n outputs, while the outer function g receives n inputs to produce one output, h.

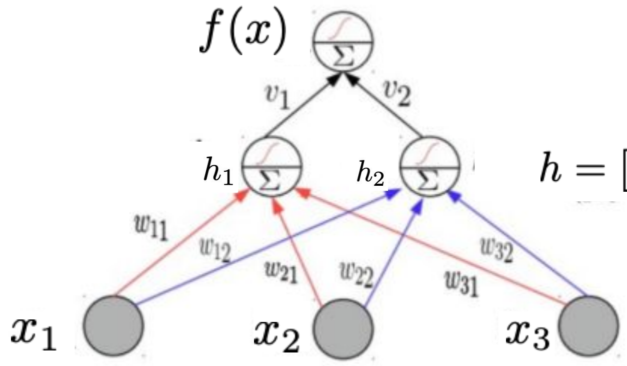
The chain rule allows to compute the variation (i.e., the partial derivative) of the function w.r.t. each component of the multivariate input → Gradient vector of $h(\theta)$

$$\boxed{\frac{\partial h}{\partial \theta_i} = \frac{\partial h}{\partial u_1}\frac{\partial u_1}{\partial \theta_i} + \frac{\partial h}{\partial u_2}\frac{\partial u_2}{\partial \theta_i} + \cdots + \frac{\partial h}{\partial u_n}\frac{\partial u_n}{\partial \theta_i} = \sum_{j=1}^{n} \frac{\partial h}{\partial u_j}\frac{\partial u_j}{\partial \theta_i}} \qquad i = 1, \ldots, m$$

$$\boxed{\nabla h(\theta) = \left(\frac{\partial h}{\partial \theta_1}, \frac{\partial h}{\partial \theta_2}, \ldots, \frac{\partial h}{\partial \theta_m}\right)^T}$$

Backpropagation: Chain Rule on Neural Network

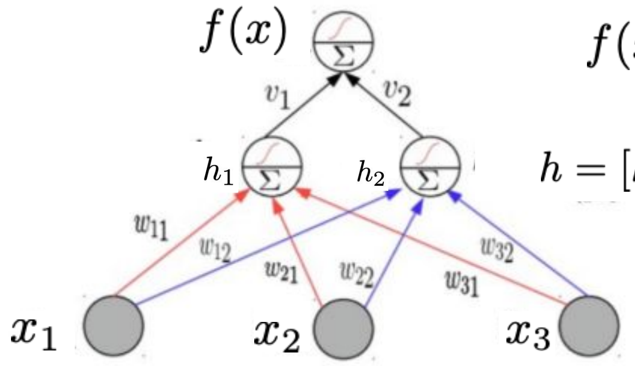
$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$



$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

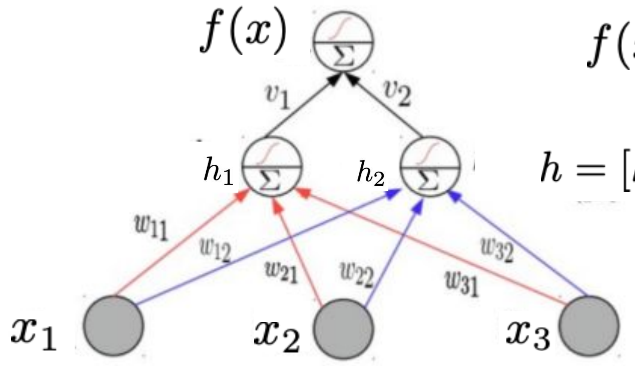


$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$



$f(x) = o(V$

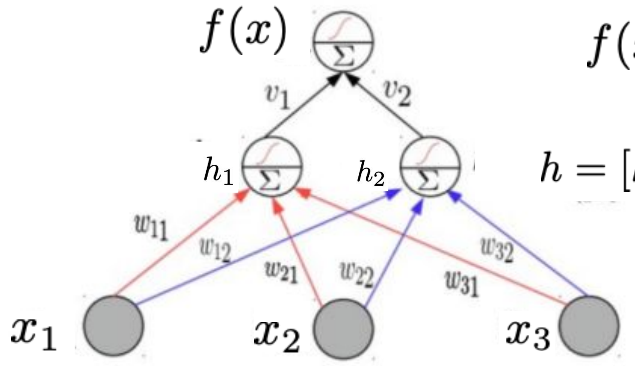
$h = [h_1, h_2]^\top =$ $Wx)$

Forward Pass

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\nabla_\theta \ell(x^i, y^i, \theta)$$



$$f(x) = o(Vh)$$

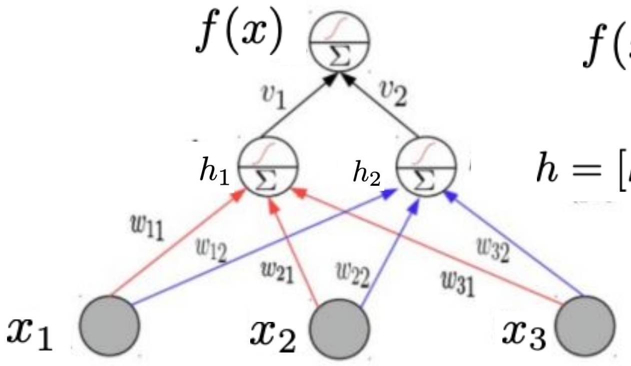
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\nabla_\theta \ell(x^i, y^i, \theta) = \left[\frac{\partial \ell(x^i, y^i, \theta)}{\partial V}, \frac{\partial \ell(x^i, y^i, \theta)}{\partial W}\right]$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial f}{\partial V}$$



$$f(x) = o(Vh)$$

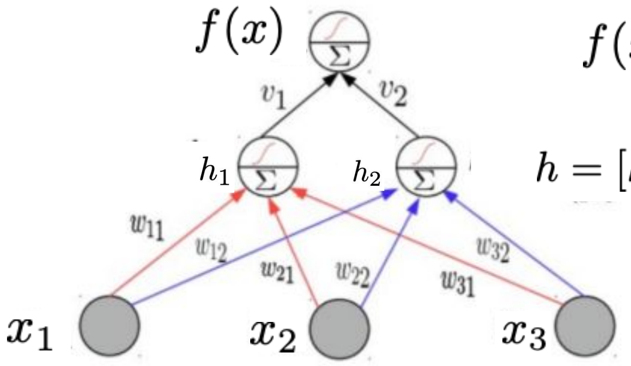
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial f}{\partial V}$$

$$\frac{\partial f(x)}{\partial V} = \frac{\partial o(Vh)}{\partial V}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

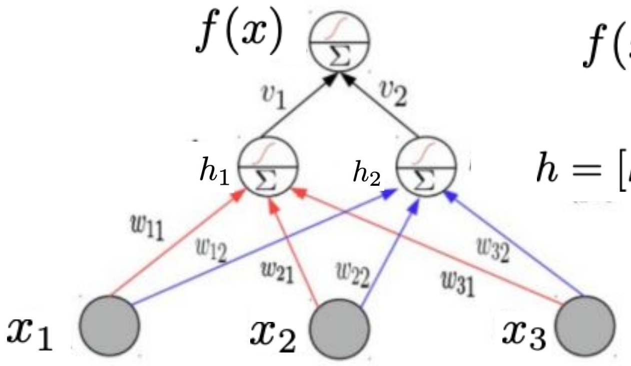
Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial f}{\partial V}$$

$$\frac{\partial f(x)}{\partial V} = \frac{\partial o(Vh)}{\partial V}$$

$$= \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial V}$$



$$f(x) = o(Vh)$$

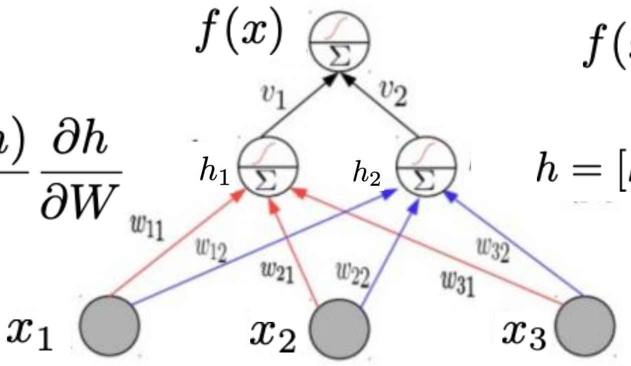
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial f}{\partial W}$$

$$\frac{\partial f}{\partial W} = \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

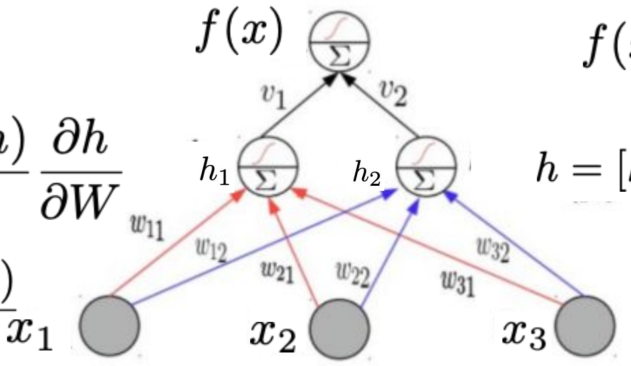
Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial f}{\partial W}$$

$$\frac{\partial f}{\partial W} = \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$

$$\frac{\partial h}{\partial W} = \frac{\partial g(Wx)}{\partial W}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

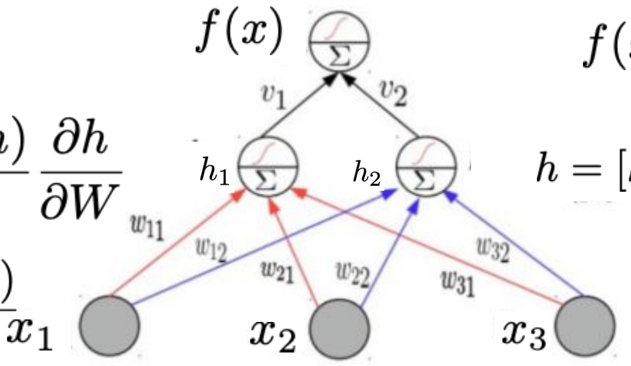
$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial f}{\partial W}$$

$$\frac{\partial f}{\partial W} = \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$

$$\frac{\partial h}{\partial W} = \frac{\partial g(Wx)}{\partial W}$$

$$= \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$



$$f(x) = o(Vh)$$

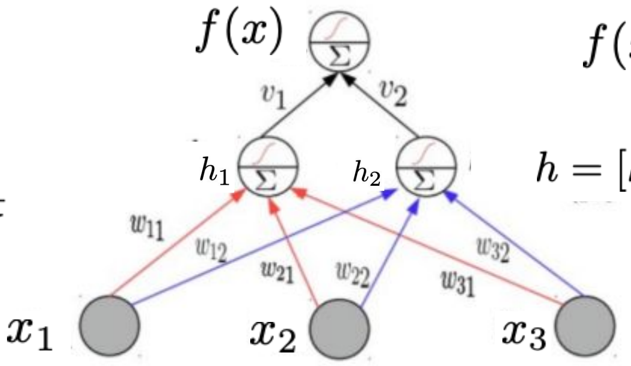
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial V}$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$



$$f(x) = o(Vh)$$

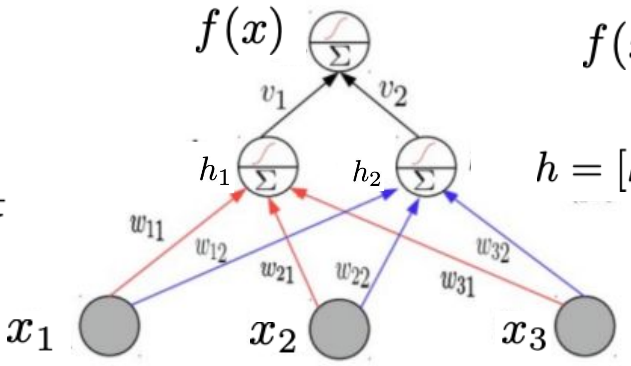
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \boxed{\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}} \frac{\partial o(Vh)}{\partial V}$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \boxed{\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}} \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$



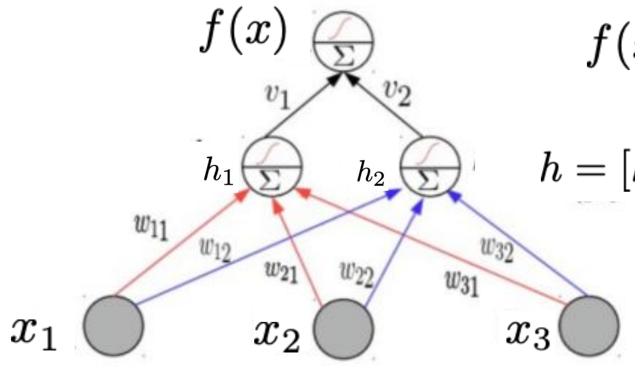
$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$



$$f(x) = o(Vh)$$

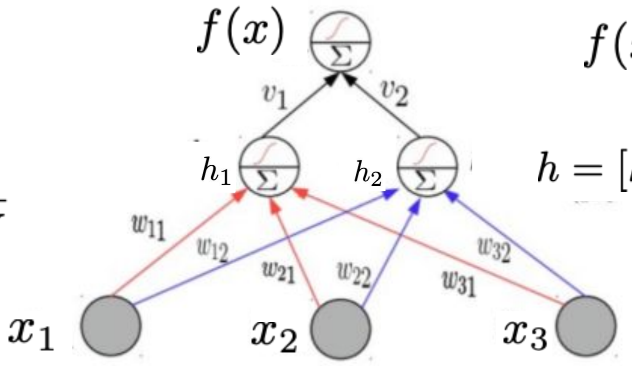
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial V}$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial h} \frac{\partial h}{\partial W}$$



$$f(x) = o(Vh)$$

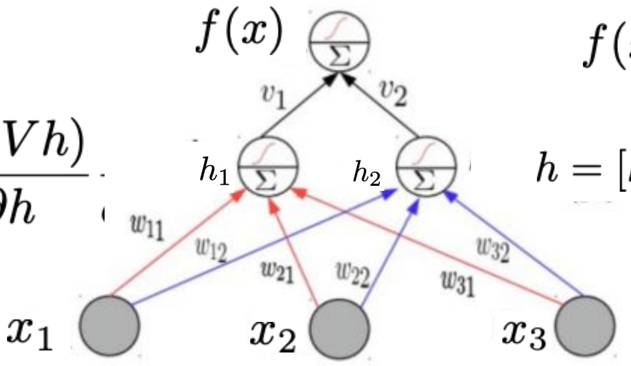
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$

$$\frac{\partial o(Vh)}{\partial V} \quad \frac{\partial o(Vh)}{\partial h}$$



$$f(x) = o(Vh)$$

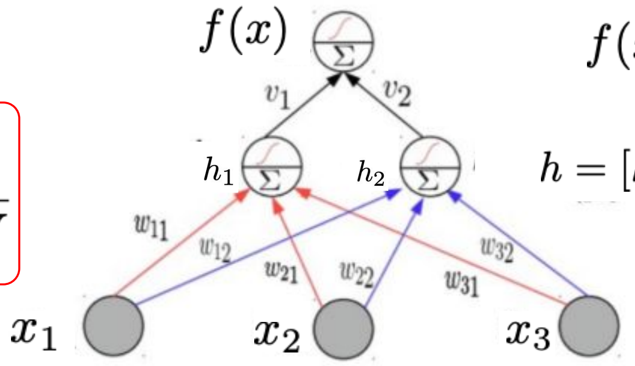
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial V} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial V}$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial W} = \frac{\partial \ell(x^i, y^i, \theta)}{\partial f} \frac{\partial o(Vh)}{\partial h} \boxed{\frac{\partial h}{\partial W}}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

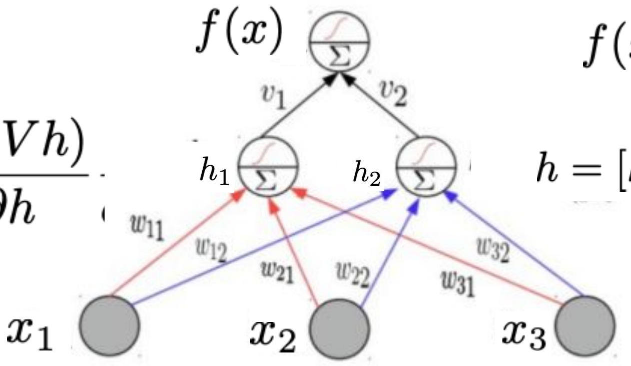
Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$

$$\frac{\partial o(Vh)}{\partial V} \quad \frac{\partial o(Vh)}{\partial h}$$

$$\frac{\partial h}{\partial W}$$



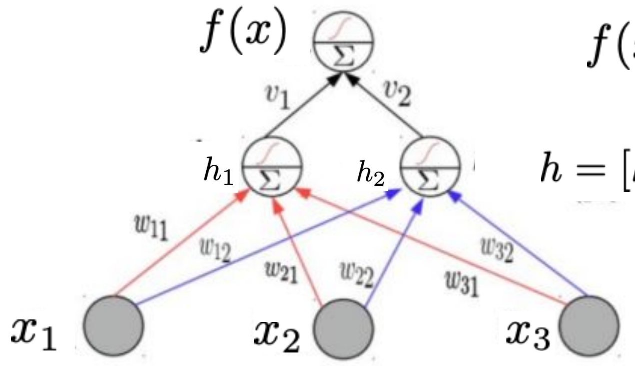
$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$



$$f(x) = o(Vh)$$

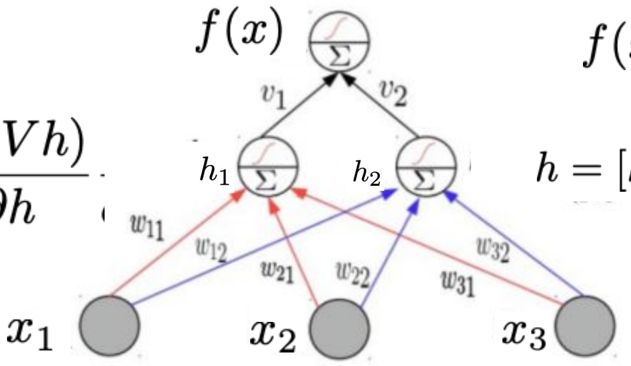
$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$

$$\frac{\partial o(Vh)}{\partial V} \quad \frac{\partial o(Vh)}{\partial h}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

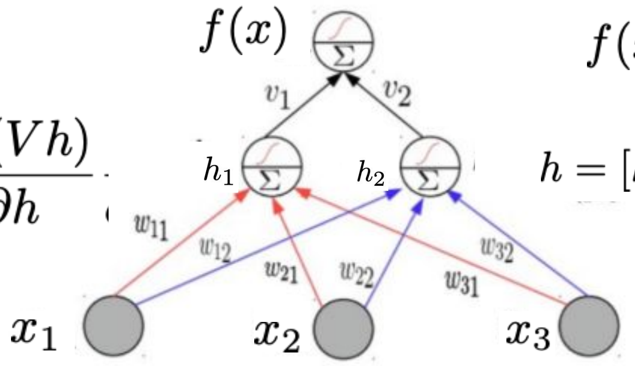
Backpropagation: Chain Rule on Neural Network

$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$

$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$

$$\frac{\partial o(Vh)}{\partial V} \quad \frac{\partial o(Vh)}{\partial h}$$

$$\frac{\partial h}{\partial W}$$

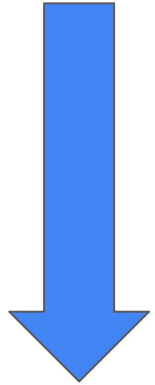


$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backpropagation: Chain Rule on Neural Network

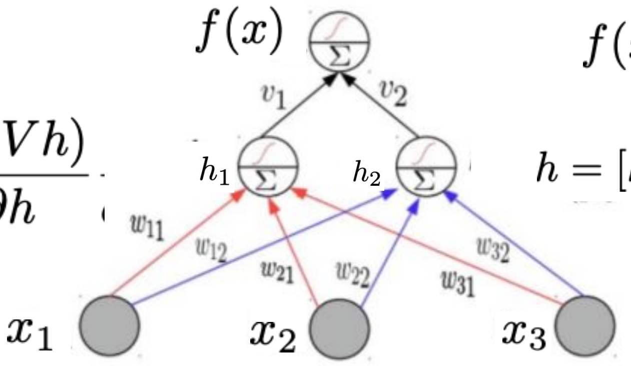
$$\ell(x^i, y^i, \theta) = (f(x^i, {\color{red}V}, {\color{red}W}) - y^i)^2$$



$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$

$$\frac{\partial o(Vh)}{\partial V} \quad \frac{\partial o(Vh)}{\partial h}$$

$$\frac{\partial h}{\partial W}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backward Pass

(Stochastic) Gradient Descent

- Initialize parameter θ^0
- Sample $\{x^i, y^i\}_{i=1}^B$
- Do $$\theta^{t+1} \leftarrow \theta^t - \eta \sum_{i=1}^{B} \nabla_\theta \ell(x^i, y^i, \theta^t) - (\lambda \nabla \Omega(\theta^t))$$

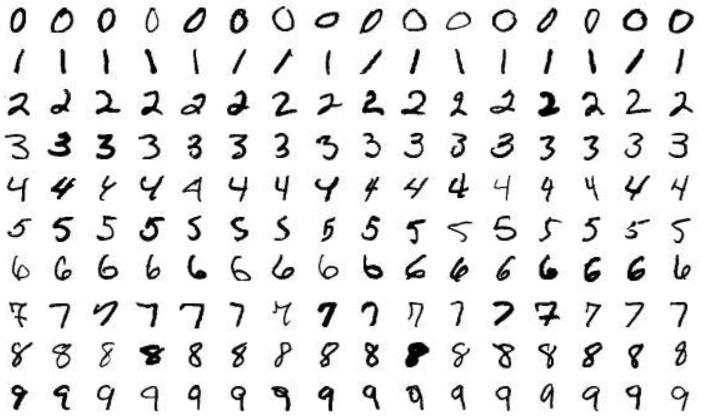
Auto-differentiation Packages

PyTorch

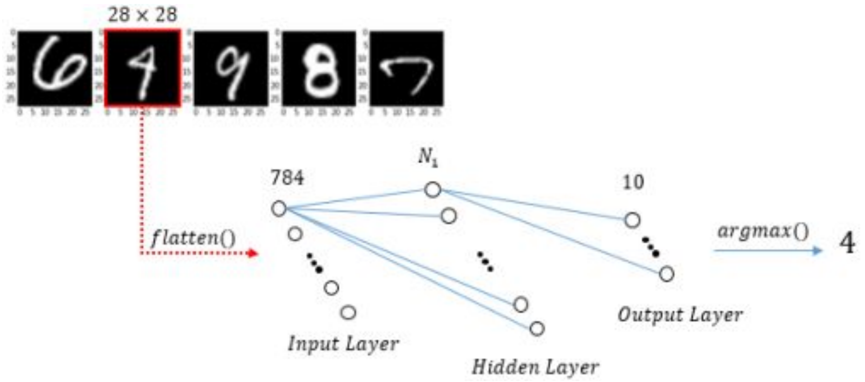
JAX

Tensorflow

MLP example: MNIST

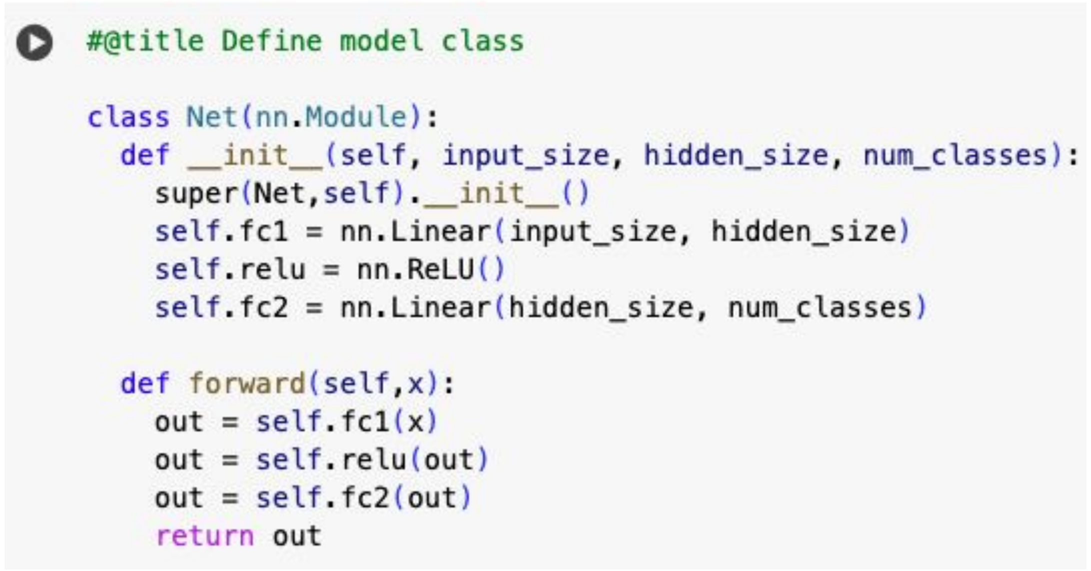


MNIST hand-written character recognition

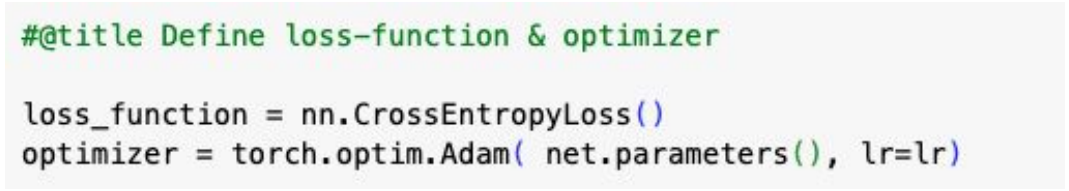


- 60,000 images
- 28x28 pixels = 784
- Grayscale, from 0 to 255 → Converted to [0,1]

PyTorch

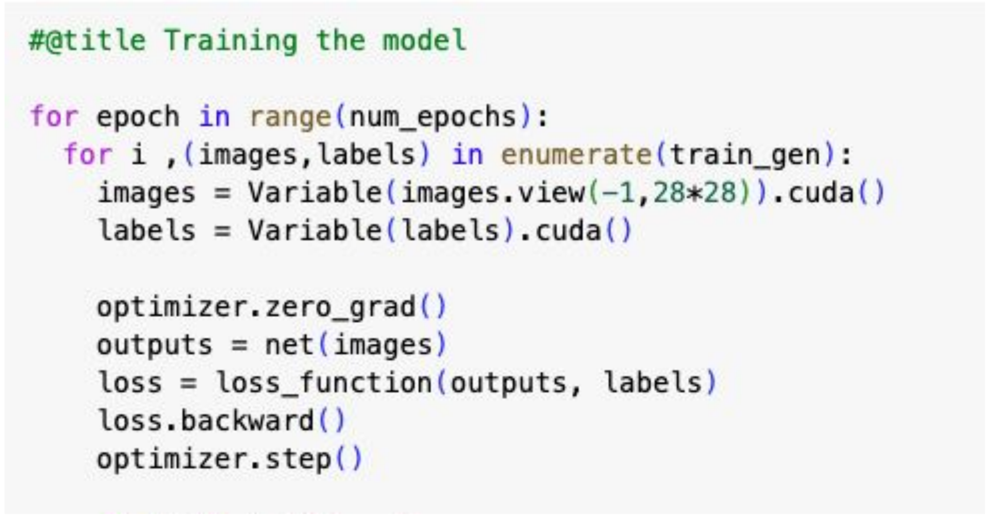


```
#@title Define model class

class Net(nn.Module):
  def __init__(self, input_size, hidden_size, num_classes):
    super(Net,self).__init__()
    self.fc1 = nn.Linear(input_size, hidden_size)
    self.relu = nn.ReLU()
    self.fc2 = nn.Linear(hidden_size, num_classes)

  def forward(self,x):
    out = self.fc1(x)
    out = self.relu(out)
    out = self.fc2(out)
    return out
```



```
#@title Define loss-function & optimizer

loss_function = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam( net.parameters(), lr=lr)
```

PyTorch



```
#@title Training the model

for epoch in range(num_epochs):
  for i ,(images,labels) in enumerate(train_gen):
    images = Variable(images.view(-1,28*28)).cuda()
    labels = Variable(labels).cuda()

    optimizer.zero_grad()
    outputs = net(images)
    loss = loss_function(outputs, labels)
    loss.backward()
    optimizer.step()
```

Q&A