

# Neural Networks and Backpropagation

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

# Pattern Recognition and Machine Learning

## Lecture 15 Neural Networks and Backpropagation



**Chizhi Chris ZHANG**

zhangchizhi@ciomp.ac.cn

Advanced Computing and Digital Technology Research Center

University of Chinese Academy of Sciences

Autumn 2026

# Lecture plan and time allocation

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

| Topic                  | Minutes |
|------------------------|---------|
| Neural representations | 45      |
| Forward propagation    | 45      |
| Backpropagation        | 45      |

## One small network throughout the calculation

Build nonlinear features; compute a prediction; trace its gradients; update together.

The numerical steps are shown on the slides.



# A neural network extends models we already know

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A linear score alone cannot separate the XOR pattern.

The output model still has the same statistical meaning.

Linear regression:  $\hat{y} = w^\top x + b$ ,

Logistic regression:  $p(y = 1 \mid x) = \sigma(w^\top x + b)$ ,

Neural network:  $h = \phi(W_1 x + b_1)$ ,  $p = \sigma(w_2^\top h + b_2)$ .



# One unit: weighted sum, bias, then activation

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

## Key idea

Changing  $b$  shifts the boundary without changing its normal direction  $w$ .

$$z = w^T x + b, \quad a = \phi(z).$$

| Quantity | Example                      | Interpretation               |
|----------|------------------------------|------------------------------|
| $x$      | $(2, -1)^T$                  | observed input               |
| $w, b$   | $(0.5, 1)^T, 0.2$            | learned parameters           |
| $z$      | $0.5(2) + 1(-1) + 0.2 = 0.2$ | score before activation      |
| $a$      | $\text{ReLU}(0.2) = 0.2$     | value sent to the next layer |

$$x = (0, 1)^T \quad z = 1.2$$



# XOR cannot be separated by one affine threshold

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Assume  $w_1x_1 + w_2x_2 + b > 0$  predicts class 1 and all other scores predict 0. The four XOR labels impose inequalities that contradict each other.

| $x_1$ | $x_2$ | $y$ | Required inequality    |
|-------|-------|-----|------------------------|
| 0     | 0     | 0   | $b \leq 0$             |
| 0     | 1     | 1   | $w_2 + b > 0$          |
| 1     | 0     | 1   | $w_1 + b > 0$          |
| 1     | 1     | 0   | $w_1 + w_2 + b \leq 0$ |

$$w_1 + w_2 + 2b > 0 \Rightarrow w_1 + w_2 + b > -b \geq 0.$$

The last row requires the same quantity to be nonpositive, a contradiction. One affine boundary cannot separate XOR.



## Two hidden tests make XOR separable

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The output separates (1, 0) from (0, 0) and (1, 1) in hidden-feature space.

$$h_1 = H(x_1 + x_2 - 0.5), \quad h_2 = H(x_1 + x_2 - 1.5), \quad \hat{y} = H(h_1 - h_2 - 0.5).$$

| Input  | $h_1$ Logical or | $h_2$ Logical AND | $h_1 - h_2$ | $\hat{y}$ |
|--------|------------------|-------------------|-------------|-----------|
| (0, 0) | 0                | 0                 | 0           | 0         |
| (0, 1) | 1                | 0                 | 1           | 1         |
| (1, 0) | 1                | 0                 | 1           | 1         |
| (1, 1) | 1                | 1                 | 0           | 0         |



**Check.** A hard threshold illustrates representation; its derivative is unsuitable for ordinary backpropagation.

# A ReLU hidden layer can also represent XOR

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Each hidden unit is simple; combining their outputs bends the response.

$$r = x_1 + x_2, \quad h_1 = \text{ReLU}(r), \quad h_2 = \text{ReLU}(r - 1), \quad f(x) = h_1 - 2h_2.$$

| Sum of input $r$ | $h_1$ | $h_2$ | $f(x)$ | XOR target |
|------------------|-------|-------|--------|------------|
| 0                | 0     | 0     | 0      | 0          |
| 1                | 1     | 0     | 1      | 1          |
| 2                | 2     | 1     | 0      | 0          |



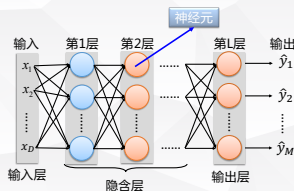
# Read a network from left to right

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A network learns combinations during training; domain knowledge still helps choose inputs and evaluate errors.  
Count learned transformations; an input node does not perform a learned transformation.

## 全连接前馈神经网络

■ 深度神经网络 ( Deep Neural Network , DNN ) : 多个隐含层



# Parameter count is independent of batch size

The inner dimensions must agree; the bias must match the output dimension.

A wider hidden layer increases representation capacity and memory use.

$$\mathbf{z}_1 = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1, \quad \mathbf{h} = \phi(\mathbf{z}_1), \quad \mathbf{z}_2 = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2.$$

| Object                       | Shape                    | Reason                       |
|------------------------------|--------------------------|------------------------------|
| $\mathbf{x}$                 | $4 \times 1$             | four input features          |
| $\mathbf{W}_1, \mathbf{b}_1$ | $3 \times 4, 3 \times 1$ | one row for each hidden unit |
| $\mathbf{z}_1, \mathbf{h}$   | $3 \times 1$             | three hidden values          |
| $\mathbf{W}_2, \mathbf{b}_2$ | $2 \times 3, 2 \times 1$ | two output scores            |
| $\mathbf{z}_2$               | $2 \times 1$             | one score per output         |

$$\text{\# parameters} = dm + m.$$

| Layer             | Weights | Biases | Sum |
|-------------------|---------|--------|-----|
| $4 \rightarrow 3$ | 12      | 3      | 15  |
| $3 \rightarrow 2$ | 6       | 2      | 8   |
| Whole network     | 18      | 5      | 23  |

Another network  $784 \rightarrow 128 \rightarrow 10$  Network:  
 $(784 + 1)128 + (128 + 1)10 = 101,770$

# Activations change both values and derivatives

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Its derivative determines how local changes influence earlier layers.

| Activation | Value $\phi(\mathbf{z})$ | Derivative away from kinks               | Typical use            |
|------------|--------------------------|------------------------------------------|------------------------|
| Sigmoid    | $1/(1 + e^{-z})$         | $\phi(\mathbf{z})(1 - \phi(\mathbf{z}))$ | binary output, gates   |
| Tanh       | $\tanh z$                | $1 - \tanh^2 z$                          | bounded hidden state   |
| ReLU       | $\max(0, z)$             | 0 or 1                                   | common hidden layer    |
| Leaky ReLU | $\max(\alpha z, z)$      | $\alpha$ or 1                            | negative-side gradient |

$$0 < \alpha < 1$$

**Check.** ReLU helps preserve derivatives on its positive side; it does not guarantee stable gradients in every deep network.

# Sigmoid saturates: the derivative explains why

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Far from zero, changing the score barely changes the activation.

$$a = \sigma(z), \quad \frac{da}{dz} = a(1 - a).$$

| $z$ | $a$   | $a(1 - a)$ | $z$ Response to minor changes |
|-----|-------|------------|-------------------------------|
| 0   | 0.500 | 0.250      | largest local response        |
| 2   | 0.881 | 0.105      | smaller response              |
| 5   | 0.993 | 0.00665    | nearly saturated              |



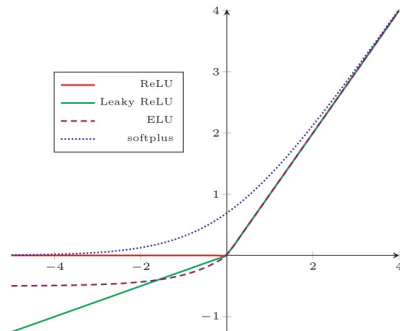
# ReLU variants keep different negative-side responses

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A unit inactive for every training example receives no ReLU gradient through that branch.

$$0 < \alpha < 1 \quad \mathbf{z} = -2$$

$$\alpha = 0.1$$



# Worked review: explain the representation

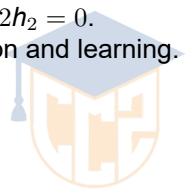
ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

At input  $(1, 1)$ , the ReLU construction gives  $r = 2$ ,  $h_1 = 2$ ,  $h_2 = 1$  and output  $h_1 - 2h_2 = 0$ . Representation specifies a possible function; the next section calculates prediction and learning.

For 5 inputs, 4 hidden units and 2 output scores, the two dense layers contain

$$(5 + 1) \times 4 + (4 + 1) \times 2 = 24 + 10 = 34.$$

In each bracket,  $+1$  is one bias per output unit.



# Without a nonlinear activation, depth collapses

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Consider two affine transformations with no hidden nonlinearity.

A final sigmoid alone still places the 0.5 boundary at one affine score equal to zero.

$$h = W_1x + b_1,$$

$$z = W_2h + b_2 = (W_2W_1)x + (W_2b_1 + b_2).$$

$$h = 2x + 1 \quad z = 3h - 4$$

$$z = 6x - 1$$



# Depth composes reusable features

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

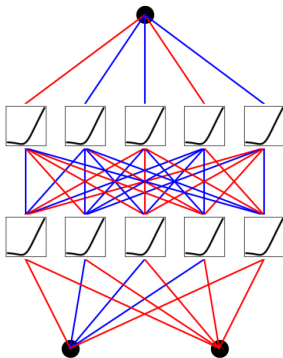
It does not promise a small model, successful training, or good generalization.

$$f(x) = f_L(f_{L-1}(\cdots f_1(x))).$$



# Compose intermediate features across layers

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER



Read this reference diagram from the inputs upward. Each layer combines previous outputs into new features. The following numerical example uses the separately displayed small network and its fixed parameters.



# Batch computation: keep one convention throughout

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The same bias vector is added to every row; it is not a separate parameter for each example.

$$Z_1 = XW_1^\top + \mathbf{1}b_1^\top, \quad H = \phi(Z_1), \quad Z_2 = HW_2^\top + \mathbf{1}b_2^\top.$$

---

| Object | $d = 4, m = 3, K = 2$ : resulting shape | Role |
|--------|-----------------------------------------|------|
|--------|-----------------------------------------|------|

---

|     |              |
|-----|--------------|
| $X$ | $B \times 4$ |
|-----|--------------|

|                     |
|---------------------|
| one row per example |
|---------------------|

|       |              |
|-------|--------------|
| $W_1$ | $3 \times 4$ |
|-------|--------------|

|                                            |
|--------------------------------------------|
| same weights as the single-example formula |
|--------------------------------------------|

|          |              |
|----------|--------------|
| $Z_1, H$ | $B \times 3$ |
|----------|--------------|

|                                      |
|--------------------------------------|
| one row of hidden values per example |
|--------------------------------------|

|       |              |
|-------|--------------|
| $W_2$ | $2 \times 3$ |
|-------|--------------|

|                                          |
|------------------------------------------|
| maps hidden coordinates to output scores |
|------------------------------------------|

|       |              |
|-------|--------------|
| $Z_2$ | $B \times 2$ |
|-------|--------------|

---

|                        |
|------------------------|
| one row per prediction |
|------------------------|

# Forward pass: compute every hidden value

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

For one example, use column vectors and ReLU hidden units:

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = 1, \quad W_1 = \begin{bmatrix} 0.1 & 0.2 \\ -0.3 & 0.4 \end{bmatrix},$$

$$b_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$z_1 = W_1 x + b_1, \quad h = \text{ReLU}(z_1), \quad v = \begin{bmatrix} 0.5 \\ -0.2 \end{bmatrix},$$

$$c = 0.1, \quad s = v^T h + c.$$

$$p = \sigma(s)$$

The model assigns about 56.2% probability to class 1 for this input.

## Worked calculation

$$z_{1,1} = 0.1(1) + 0.2(2) = 0.5,$$

$$z_{1,2} = -0.3(1) + 0.4(2) = 0.5,$$

$$h = (0.5, 0.5)^T,$$

$$s = 0.5(0.5) - 0.2(0.5) + 0.1 = 0.25,$$

$$p = \sigma(0.25) \approx 0.562177.$$

# Binary cross-entropy rewards the true class

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Minimizing this loss maximizes the Bernoulli likelihood of the observed label.

$$L = -y \log p - (1 - y) \log(1 - p).$$

$$y = 1 \quad p = 0.562177$$

$$L = -\log(0.562177) \approx 0.575939.$$

| $y = 1$ for $p$ | Loss  | Interpretation                     |
|-----------------|-------|------------------------------------|
| 0.9             | 0.105 | high probability on the true class |
| 0.5             | 0.693 | uncertain prediction               |
| 0.1             | 2.303 | confidently favors the wrong class |



# Regression needs a numeric output, not a class probability

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Changing the prediction target changes the output and the loss.

$$\hat{y} = v^T h + c, \quad L = \frac{1}{2}(\hat{y} - y)^2, \quad \frac{\partial L}{\partial \hat{y}} = \hat{y} - y.$$

| Target  | Predicted       | Loss  | Output derivative |
|---------|-----------------|-------|-------------------|
| $y = 2$ | $\hat{y} = 1.5$ | 0.125 | -0.5              |
| $y = 2$ | $\hat{y} = 3$   | 0.5   | 1                 |



# Compute classification losses from logits

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A  $K$ -class single-label task usually has  $K$  logits; the target specifies one class index.

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}, \quad L = -\log p_y.$$

$$\mathbf{z} = (2, 1, 0)$$

$$\mathbf{p} = \frac{(e^2, e, 1)}{e^2 + e + 1} \approx (0.6652, 0.2447, 0.0900).$$

Supply raw logits when the API expects logits; do not apply softmax twice.

$$L(\mathbf{s}, y) = \log(1 + e^s) - ys, \quad \frac{\partial L}{\partial \mathbf{s}} = \sigma(\mathbf{s}) - y.$$

$$L(\mathbf{z}, y) = -z_y + \text{logsumexp}(\mathbf{z}),$$

$$\text{logsumexp}(\mathbf{z}) = m + \log \sum_j e^{z_j - m}, \quad m = \max_j z_j.$$



# A batch loss combines example losses and regularization

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The data gradient is the mean of the individual example gradients.

$$J(\theta) = \frac{1}{B} \sum_{i=1}^B L_i(\theta) + \frac{\lambda}{2} \|\mathbf{W}\|_F^2.$$



**Check.** Switching from mean to sum changes gradient magnitudes and the effective learning-rate scale.

# Chain rule multiplies local effects

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The backward calculation combines local derivatives along the dependency paths.

$$u = wx, \quad z = u + b, \quad a = \phi(z),$$

$$\hat{y} = va + c, \quad L = \frac{1}{2}(\hat{y} - y)^2.$$

$$u = wx \quad \partial u / \partial w = x$$

$$z = u + b \quad \partial z / \partial u = 1$$

$$a = \phi(z) \quad \partial a / \partial z = \phi'(z)$$

$$\hat{y} = va + c \quad \partial \hat{y} / \partial a = v$$

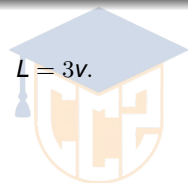
## Key idea

Increasing  $x$  by 0.001 changes  $L$  by about  $24(0.001) = 0.024$ . The exact change is 0.024012.

$$\begin{array}{lll} u = 2x, & v = u^2, & L = 3v. \\ x = 1 & u = 2 & \\ v = 4 & L = 12 & \end{array}$$

$$\frac{dL}{dx} = \frac{dL}{dv} \frac{dv}{du} \frac{du}{dx} = 3 \cdot (2u) \cdot 2 = 24.$$

$$24(0.001) = 0.024$$



# Forward values are also backward inputs

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A convolution reuses one filter at many positions; its weight gradient sums all those contributions.

$$a = x^2 \quad b = 3x$$

$$L = a + b$$

$$\frac{dL}{dx} = \frac{\partial L}{\partial a} \frac{da}{dx} + \frac{\partial L}{\partial b} \frac{db}{dx} = 2x + 3.$$

$$x = 2 \quad dL/dx = 7$$

**Check.** Using only one outgoing path produces an incomplete gradient.

Automatic differentiation records the operations needed to obtain derivatives.

$$x = (1, 2) \quad \mathbf{z}_1 = \mathbf{h} = (0.5, 0.5)$$

$$p = 0.562177$$



## Worked review: target, score, probability, loss

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The target matches the logit shape; do not let unintended broadcasting create extra examples.

```
import torch
import torch.nn.functional as F
X = torch.tensor([[1., 2.]])          # B x 2
W1 = torch.tensor([[.1, .2], [-.3, .4]])
b1 = torch.zeros(2)
v = torch.tensor([[.5, -.2]])
c = torch.tensor([.1])
h = F.relu(X @ W1.T + b1)             # B x 2
logits = h @ v.T + c                  # B x 1
target = torch.ones_like(logits)
loss = F.binary_cross_entropy_with_logits(logits, target)
```

The current target is still 1. We now trace its loss backward and update the same parameters.

$$p = \sigma(0.25) \approx 0.5622$$

# Start backward at the binary output score

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The forward pass produced  $h = (0.5, 0.5)$ ,  
 $s = 0.25$  and  $p \approx 0.562177$  for target 1.

$h = (0.5, 0.5)$        $s = 0.25$

The target is  $y = 1$ , the probability is  
 $p = .562177$  and  $L = -\log p$ .

$y = 1$        $p = .562177$

$L = -\log p$

$$\frac{\partial L}{\partial p} = -\frac{1}{p}, \quad \frac{\partial p}{\partial s} = p(1 - p),$$

$$\frac{\partial L}{\partial s} = -\frac{1}{p} p(1 - p) = p - 1 \approx -.437823.$$

The score is too low for target 1; decreasing  
the loss requires raising the score locally.

$$\delta_s = \frac{\partial L}{\partial s} = p - y = 0.562177 - 1 = -0.437823.$$

$$s = v^T h + c$$

$$\nabla_v L = \delta_s h = \begin{bmatrix} -0.218912 \\ -0.218912 \end{bmatrix},$$

$$\frac{\partial L}{\partial c} = \delta_s = -0.437823.$$

# Send the gradient to each hidden unit

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Increasing the first hidden value raises the score and helps this example.

$$\frac{\partial L}{\partial \mathbf{h}} = \mathbf{v} \delta_s = \begin{bmatrix} 0.5 \\ -0.2 \end{bmatrix} (-0.437823) = \begin{bmatrix} -0.218912 \\ 0.087565 \end{bmatrix}.$$

$$\mathbf{z}_{1,j} = 0.5 > 0$$

$$\delta_1 = \frac{\partial L}{\partial \mathbf{z}_1} = \frac{\partial L}{\partial \mathbf{h}} \odot \text{ReLU}'(\mathbf{z}_1) = \begin{bmatrix} -0.218912 \\ 0.087565 \end{bmatrix}.$$

$$\mathbf{v}_2 < 0$$



# Each first-layer weight gets input times error

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Unit 1 is active. Multiply its hidden error by input  $x_2 = 2$ .

## Worked calculation

$$x_2 = 2 \quad v_1 = .5$$

$$\frac{\partial L}{\partial W_{1,12}} = \underbrace{\frac{\partial L}{\partial s}}_{-.437823} \underbrace{\frac{\partial s}{\partial h_1}}_{.5} \underbrace{\frac{\partial h_1}{\partial z_{1,1}}}_1 \underbrace{\frac{\partial z_{1,1}}{\partial W_{1,12}}}_2 \approx -.437823.$$

A larger input can produce a larger weight gradient for the same hidden error.

$$z_1 = W_1 x + b_1$$

$$\nabla_{W_1} L = \delta_1 x^\top = \begin{bmatrix} -0.218912 \\ 0.087565 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} -0.218912 & -0.437823 \\ 0.087565 & 0.175129 \end{bmatrix}.$$

$$\nabla_{b_1} L = \delta_1 = (-0.218912, 0.087565)^\top.$$

$$(2 \times 1)(1 \times 2) = 2 \times 2$$

# One gradient step lowers the running-example loss

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Update every parameter using its own gradient and learning rate .1. Recompute the forward pass with all updated parameters. The loss on this same example decreases from .575939 to .515597.

$$\theta_{\text{new}} = \theta - \eta \nabla_{\theta} L, \quad 0.1 - 0.1(-0.21891) = 0.121891.$$

$$W_{1,\text{new}} \approx \begin{bmatrix} 0.121891 & 0.243782 \\ -0.308756 & 0.382487 \end{bmatrix}, \quad b_{1,\text{new}} \approx \begin{bmatrix} 0.021891 \\ -0.008756 \end{bmatrix},$$

$$v_{\text{new}} \approx (0.521891, -0.178109)^{\top}, \quad c_{\text{new}} \approx 0.143782.$$



| Forward result | Before     | After                |
|----------------|------------|----------------------|
| $h$            | (0.5, 0.5) | (0.631347, 0.447461) |
| $s$            | 0.25       | 0.393580             |
| $p$            | 0.562177   | 0.597144             |
| $L$            | 0.575939   | 0.515597             |

## Worked review: trace one changed example

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Return to the original parameters and change the example to  $x = (2, 1)^T$ ,  $y = 0$ . The second hidden preactivation is negative, so only the first hidden branch receives a gradient on this row.

$$z_1 = (.4, -.2)^T \quad h = (.4, 0)^T$$

$$s = 0.5(0.4) - 0.2(0) + 0.1 = 0.3.$$

$$p = \sigma(.3) \approx .574443,$$

$$L = -\log(1 - p) \approx .854355,$$

$$\delta_s = p - y = .574443,$$

$$\delta_{1,1} = \delta_s(.5)\mathbf{1}\{.4 > 0\} \approx .287221,$$

$$\delta_{1,2} = \delta_s(-.2)\mathbf{1}\{-.2 > 0\} = 0.$$



**Check.** The second ReLU is inactive for this row. Other branches still learn, and another input can activate this unit. A positive  $\delta_s$  makes gradient descent locally lower the score.

# Multiple output paths add at a hidden unit

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

If one hidden value has outgoing weights 0.4, -0.2 and the two score gradients are 0.3, -0.5, its incoming gradient is  $0.4(0.3) + (-0.2)(-0.5) = 0.22$ .

$$\frac{\partial L}{\partial h_j} = \sum_k w_{2,kj} \delta_{2,k}, \quad \delta_1 = (W_2^T \delta_2) \odot \phi'(z_1).$$

Given outgoing weights .4 and -0.2 , with score gradients .3 and -0.5 , and

$$\frac{\partial L}{\partial h_j} = 0.4(0.3) + (-0.2)(-0.5) = 0.12 + 0.10 = 0.22.$$

Multiply by this hidden unit's activation derivative to obtain its preactivation gradient.



# Batch backpropagation has predictable shapes

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

For row-wise examples,  $Z = AW^\top + \mathbf{1}b^\top$ , with  $A \in \mathbb{R}^{B \times d}$  and  $W \in \mathbb{R}^{m \times d}$ :

$$Z = AW^\top + \mathbf{1}b^\top \quad A \in \mathbb{R}^{B \times d}$$

$$W \in \mathbb{R}^{m \times d}$$

$$G = \frac{\partial J}{\partial Z} \in \mathbb{R}^{B \times m}, \quad \nabla_W J = G^\top A, \quad \nabla_b J = G^\top \mathbf{1}, \quad \frac{\partial J}{\partial A} = GW.$$

**Check.** For mean loss,  $G$  already includes  $1/B$ . Do not divide the weight gradient by  $B$  a second time.



## Two examples contribute to one weight gradient

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

This is a separate arithmetic check of the preceding batch formulas. The local derivative matrix  $G = \partial J / \partial Z$  is supplied and already includes any mean-loss factor.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} .1 & .2 \\ .3 & .4 \end{bmatrix}.$$

### Weight gradient

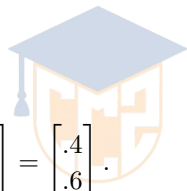
$$\nabla_w J = G^T A = \begin{bmatrix} .1 & .3 \\ .2 & .4 \end{bmatrix}.$$

Each entry adds the corresponding input-times-error contributions from both rows.

### Bias gradient

$$\nabla_b J = G^T \mathbf{1} = \begin{bmatrix} .1 + .3 \\ .2 + .4 \end{bmatrix} = \begin{bmatrix} .4 \\ .6 \end{bmatrix}.$$

Two examples still update one shared bias per output. Do not divide by the batch size a second time.



# A complete training step follows one order

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

```
optimizer.zero_grad()  
hidden = F.relu(X @ W1.T + b1)  
logits = hidden @ v.T + c  
loss = F.binary_cross_entropy_with_logits(logits, target)  
loss.backward()  
optimizer.step()
```

The architecture defines the function, the loss defines the objective, backpropagation computes derivatives, and the optimizer changes parameters.



# Check one derivative by a small perturbation

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Change only the selected parameter and hold all others fixed.

$$g_j^{\text{num}} = \frac{L(\theta + \epsilon \mathbf{e}_j) - L(\theta - \epsilon \mathbf{e}_j)}{2\epsilon}.$$

$$v_1 = 0.5 \quad \epsilon = 10^{-4}$$

$$L(v_1 + \epsilon) \approx 0.575917530, \quad L(v_1 - \epsilon) \approx 0.575961312, \quad g^{\text{num}} \approx -0.218912.$$

$$g^{\text{backprop}} = (p - y)h_1 \approx -0.218912.$$



# Initialization scale uses the number of inputs

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

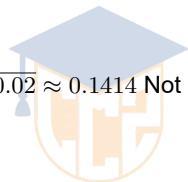
Use different random initial weights with a controlled scale. Biases can often start at zero.

He variance is 0.02 and standard deviation is  $\sqrt{0.02} \approx 0.1414$ .

$$\text{Var}(z) \approx d \text{Var}(w) \text{Var}(x).$$

$$d = 100$$

For example, ReLU initialization uses  $\text{Var}(w) = 2/100 = 0.02$  gives standard deviation  $\sqrt{0.02} \approx 0.1414$  Not 0.02.



# Products of local factors can shrink or grow

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The pre-activation scale grows to about two so the ReLU output second moment stays near one.

$$d = 100 \quad 2/d = 0.02$$

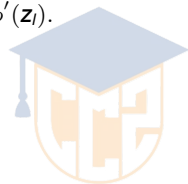
$$\mathbb{E}[\mathbf{z}^2] \approx 100(0.02)(1) = 2.$$

$$\mathbb{E}[\text{ReLU}(\mathbf{z})^2] \approx \frac{1}{2}\mathbb{E}[\mathbf{z}^2] \approx 1.$$

**Check.** This uses the second moment. ReLU outputs have a positive mean, so their variance is not generally equal to their second moment.

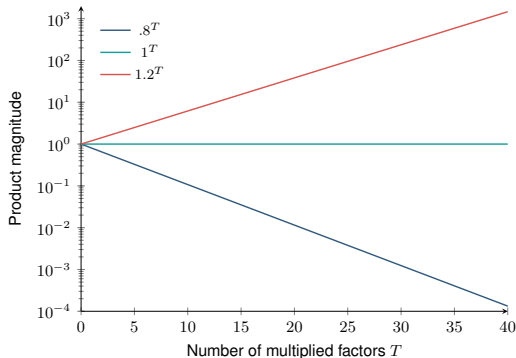
Initialization controls a starting scale; residual paths and normalization in later lectures provide additional ways to improve training.

$$\delta_l = (\mathbf{W}_{l+1}^\top \delta_{l+1}) \odot \phi'(\mathbf{z}_l).$$



# Repeated factors can shrink or enlarge a gradient

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER



The horizontal axis counts repeated multiplications; the vertical axis uses a logarithmic scale. Repeating  $.8$  shrinks a signal,  $1$  preserves it, and  $1.2$  enlarges it. Real gradients involve different local factors along different paths.



# What did one successful gradient step prove?

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Run the hand-calculated example and compare forward values and gradients.

Our complete parameter update lowered the same-example loss from .575939 to .515597. What conclusion is supported?

The displayed forward values, derivatives and update are internally checkable, and this step improves this example. Repeating the process can fit training data. Generalization still needs cases excluded from parameter fitting and model selection.

## Condition to check

A zero ReLU gradient for one row does not establish that the unit is permanently inactive.