Lecture 1.3: Backpropagation

Module 1: Foundations of Neural Networks
Deep and Reinforcement Learning

Srikanth Pai

Madras School of Economics

Preliminaries

Differentials Are Perturbations

A differential records the first-order effect of a small change.

For a vector input, choose any direction $d\mathbf{x} \in \mathbb{R}^n$:

$$f(\mathbf{x} + d\mathbf{x}) = f(\mathbf{x}) + df + \text{higher-order terms.}$$

For a matrix input, choose any direction dW of the same shape:

$$L(W + dW) = L(W) + dL + \text{higher-order terms.}$$

The symbols $d\mathbf{x}$, $d\mathbf{h}$, $d\mathbf{z}$ are vectors; dW is a matrix; dL is a scalar.

Vector Gradient Read-Off

For a scalar function $f : \mathbb{R}^n \to \mathbb{R}$:

$$df = \sum_i \frac{\partial f}{\partial x_i}\, dx_i = (\nabla_{\mathbf{x}} f)^\top d\mathbf{x}.$$

If we rewrite the same scalar as

$$df = \mathbf{a}^\top d\mathbf{x}$$

for every perturbation $d\mathbf{x}$, then

$$\boxed{\nabla_{\mathbf{x}} f = \mathbf{a}}.$$

Example. If $df = \begin{pmatrix} 2 \\ 3 \end{pmatrix}^\top d\mathbf{x}$, then $\nabla_{\mathbf{x}} f = \left[\begin{smallmatrix} 2 \\ 3 \end{smallmatrix}\right]$.

Matrix Gradient Read-Off

For matrices of the same shape,

$$\langle A, B \rangle = \mathrm{tr}(A^\top B) = \sum_{i,j} A_{ij} B_{ij}.$$

For a scalar loss $L(W)$, define the matrix gradient by

$$dL = \langle \nabla_W L,\, dW \rangle.$$

If we rewrite the same scalar as

$$dL = \langle A, dW \rangle$$

for every perturbation dW, then

$$\boxed{\nabla_W L = A}.$$

Weight Term Read-Off

Let

$$\mathbf{h} \in \mathbb{R}^m, \quad \boldsymbol{\delta} \in \mathbb{R}^n, \quad dW \in \mathbb{R}^{m \times n}.$$

Start with the scalar

$$\mathbf{h}^\top dW\, \boldsymbol{\delta}.$$

Componentwise,

$$\mathbf{h}^\top dW\, \boldsymbol{\delta} = \sum_{i=1}^{m} \sum_{j=1}^{n} h_i\, dW_{ij}\, \delta_j = \sum_{i=1}^{m} \sum_{j=1}^{n} (h_i \delta_j) dW_{ij}.$$

Since

$$\left(\mathbf{h}\boldsymbol{\delta}^\top\right)_{ij} = h_i \delta_j,$$

$$\boxed{\mathbf{h}^\top dW\, \boldsymbol{\delta} = \left\langle \mathbf{h}\boldsymbol{\delta}^\top, dW \right\rangle}.$$

Vector Term Read-Offs

Let

$$W \in \mathbb{R}^{m \times n}, \quad \boldsymbol{\delta} \in \mathbb{R}^n, \quad d\mathbf{h} \in \mathbb{R}^m, \quad d\mathbf{b} \in \mathbb{R}^n.$$

Previous-activation term:

$$\boldsymbol{\delta}^\top W^\top d\mathbf{h} = (W\boldsymbol{\delta})^\top d\mathbf{h}.$$

So the coefficient of $d\mathbf{h}$ is

$$\boxed{\nabla_{\mathbf{h}} L = W\boldsymbol{\delta}}.$$

Bias term:

$$\boldsymbol{\delta}^\top d\mathbf{b}$$

is already in the read-off form $\mathbf{c}^\top d\mathbf{b}$, so

$$\boxed{\nabla_{\mathbf{b}} L = \boldsymbol{\delta}}.$$

Affine Layer Differential

Let

$$\mathbf{z} = W^\top \mathbf{h} + \mathbf{b}, \qquad W \in \mathbb{R}^{m \times n}, \quad \mathbf{h} \in \mathbb{R}^m, \quad \mathbf{z}, \mathbf{b} \in \mathbb{R}^n.$$

Then

$$d\mathbf{z} = dW^\top \mathbf{h} + W^\top d\mathbf{h} + d\mathbf{b}.$$

Component check:

$$dz_j = \sum_i dW_{ij} h_i + \sum_i W_{ij} dh_i + db_j.$$

Pointwise Activation Differential

Let $\mathbf{h} = g(\mathbf{z})$, where g is applied coordinate by coordinate:

$$h_i = g(z_i).$$

Then

$$dh_i = g'(z_i)\, dz_i, \qquad d\mathbf{h} = g'(\mathbf{z}) \odot d\mathbf{z}.$$

If $dL = \boldsymbol{\delta}_{\mathbf{h}}^{\top} d\mathbf{h}$, then

$$dL = \big(\boldsymbol{\delta}_{\mathbf{h}} \odot g'(\mathbf{z})\big)^{\top} d\mathbf{z}.$$

Read off:

$$\boxed{\boldsymbol{\delta}_{\mathbf{z}} = \boldsymbol{\delta}_{\mathbf{h}} \odot g'(\mathbf{z})}.$$

Backpropagation

Gradient Descent

We train by minimizing a scalar loss $\mathcal{L}(\boldsymbol{\theta})$.

For a scalar function, we use column-vector gradients:

$$d\mathcal{L} = \sum_j \frac{\partial \mathcal{L}}{\partial \theta_j}\, d\theta_j = (\nabla_{\boldsymbol{\theta}} \mathcal{L})^\top d\boldsymbol{\theta}.$$

Gradient descent moves parameters in the negative-gradient direction:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}.$$

Problem. A neural network has many parameters, so we need all gradients efficiently.

The Gradient Challenge

We need $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ with respect to *every* weight and bias.

Naive approach: finite differences

$$\frac{\partial \mathcal{L}}{\partial \theta_j} \approx \frac{\mathcal{L}(\boldsymbol{\theta} + \varepsilon \mathbf{e}_j) - \mathcal{L}(\boldsymbol{\theta})}{\varepsilon}$$

Requires **one forward pass per parameter**. For $P = 10^8$ parameters, this is not feasible.

Backpropagation:

- one forward pass: compute and **store** activations;
- one backward pass: propagate errors and accumulate gradients;
- total cost: $O(P)$.

One Layer and Its Perturbations

At layer $l = 1, \ldots, L$:

$$\mathbf{z}^{(l)} = W^{(l)\top}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \qquad \mathbf{h}^{(l)} = g_l(\mathbf{z}^{(l)}).$$

Objects and perturbations have matching shapes:

$$\mathbf{h}^{(l-1)}, d\mathbf{h}^{(l-1)} \in \mathbb{R}^{n_{l-1}}, \qquad W^{(l)}, dW^{(l)} \in \mathbb{R}^{n_{l-1} \times n_l},$$

$$\mathbf{z}^{(l)}, d\mathbf{z}^{(l)}, \mathbf{b}^{(l)}, d\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}.$$

One Layer: Differential

From

$$\mathbf{z}^{(l)} = W^{(l)\top}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)},$$

the first-order change is

$$d\mathbf{z}^{(l)} = dW^{(l)\top}\mathbf{h}^{(l-1)} + W^{(l)\top}d\mathbf{h}^{(l-1)} + d\mathbf{b}^{(l)}.$$

Componentwise:

$$dz_j^{(l)} = \sum_i dW_{ij}^{(l)} h_i^{(l-1)} + \sum_i W_{ij}^{(l)} dh_i^{(l-1)} + db_j^{(l)}.$$

Delta Means Coefficient of $d\mathbf{z}$

Define

$$\boldsymbol{\delta}^{(l)} := \nabla_{\mathbf{z}^{(l)}} \mathcal{L} \in \mathbb{R}^{n_l}.$$

This means

$$d\mathcal{L} = \left(\boldsymbol{\delta}^{(l)}\right)^\top d\mathbf{z}^{(l)} = \sum_j \delta_j^{(l)}\, dz_j^{(l)}.$$

Since $d\mathbf{z}^{(l)}$ is a vector, its coefficient $\boldsymbol{\delta}^{(l)}$ is also a vector in $\mathbb{R}^{n_l}$.

One Layer: Expand $d\mathcal{L}$

Substitute the layer differential into $d\mathcal{L} = (\boldsymbol{\delta}^{(l)})^\top d\mathbf{z}^{(l)}$:

$$d\mathcal{L} = \left(\boldsymbol{\delta}^{(l)}\right)^\top \left(dW^{(l)\top}\mathbf{h}^{(l-1)} + W^{(l)\top} d\mathbf{h}^{(l-1)} + d\mathbf{b}^{(l)} \right).$$

Distribute:

$$d\mathcal{L} = \left(\mathbf{h}^{(l-1)}\right)^\top dW^{(l)}\boldsymbol{\delta}^{(l)} + \left(W^{(l)}\boldsymbol{\delta}^{(l)}\right)^\top d\mathbf{h}^{(l-1)} + \left(\boldsymbol{\delta}^{(l)}\right)^\top d\mathbf{b}^{(l)}.$$

One Layer: Read Off Gradients

Use the matrix read-off on the first term:

$$\left(\mathbf{h}^{(l-1)}\right)^\top dW^{(l)}\boldsymbol{\delta}^{(l)} = \left\langle \mathbf{h}^{(l-1)}\left(\boldsymbol{\delta}^{(l)}\right)^\top, dW^{(l)} \right\rangle.$$

So the three coefficients are:

$$\boxed{\nabla_{W^{(l)}}\mathcal{L} = \mathbf{h}^{(l-1)}\left(\boldsymbol{\delta}^{(l)}\right)^\top}$$

$$\boxed{\nabla_{\mathbf{b}^{(l)}}\mathcal{L} = \boldsymbol{\delta}^{(l)}} \qquad \boxed{\nabla_{\mathbf{h}^{(l-1)}}\mathcal{L} = W^{(l)}\boldsymbol{\delta}^{(l)}}.$$

Through the Activation

Since

$$\mathbf{h}^{(l-1)} = g_{l-1}(\mathbf{z}^{(l-1)}), \qquad d\mathbf{h}^{(l-1)} = g'_{l-1}(\mathbf{z}^{(l-1)}) \odot d\mathbf{z}^{(l-1)}.$$

With $\nabla_{\mathbf{h}^{(l-1)}} \mathcal{L} = W^{(l)} \boldsymbol{\delta}^{(l)}$,

$$d\mathcal{L} = \left(W^{(l)} \boldsymbol{\delta}^{(l)}\right)^\top d\mathbf{h}^{(l-1)}$$

$$= \left(W^{(l)} \boldsymbol{\delta}^{(l)} \odot g'_{l-1}(\mathbf{z}^{(l-1)})\right)^\top d\mathbf{z}^{(l-1)}.$$

Hence

$$\boxed{\boldsymbol{\delta}^{(l-1)} = W^{(l)} \boldsymbol{\delta}^{(l)} \odot g'_{l-1}(\mathbf{z}^{(l-1)})}.$$

Backpropagation Formulas

For $l = L, L-1, \ldots, 1$:

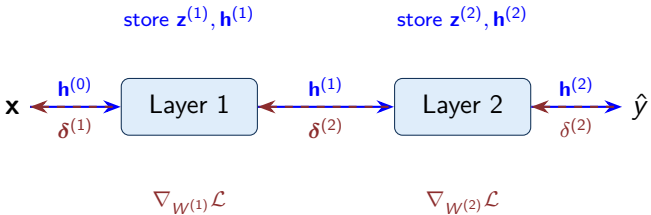
$$\nabla_{W^{(l)}} \mathcal{L} = \mathbf{h}^{(l-1)} \left(\boldsymbol{\delta}^{(l)}\right)^\top, \qquad \nabla_{\mathbf{b}^{(l)}} \mathcal{L} = \boldsymbol{\delta}^{(l)}.$$

Then pass the vector coefficient backward:

$$\nabla_{\mathbf{h}^{(l-1)}} \mathcal{L} = W^{(l)} \boldsymbol{\delta}^{(l)}.$$

$$\boldsymbol{\delta}^{(l-1)} = \nabla_{\mathbf{h}^{(l-1)}} \mathcal{L} \odot g'_{l-1}(\mathbf{z}^{(l-1)}).$$

Forward and Backward Pass



Forward (left to right):
compute and store activations layer by layer

Backward (right to left):
propagate deltas, accumulate gradients

Forward Pass Algorithm

Input: $\mathbf{x}$; parameters $\{W^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$
Output: $\hat{y}$; stored $\{\mathbf{z}^{(l)}, \mathbf{h}^{(l)}\}$

$\mathbf{h}^{(0)} \leftarrow \mathbf{x}$;
for $l = 1, 2, \ldots, L$ **do**
  $\mathbf{z}^{(l)} \leftarrow W^{(l)\top}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$;
  $\mathbf{h}^{(l)} \leftarrow g_l(\mathbf{z}^{(l)})$;
end
return $\mathbf{h}^{(L)}$;

Critical: store every $\mathbf{z}^{(l)}$ and $\mathbf{h}^{(l)}$. They are read by the backward pass.

Backward Pass Algorithm

Input: target y; stored $\{\mathbf{z}^{(l)}, \mathbf{h}^{(l)}\}$; $\{W^{(l)}\}$
Output: $\{\nabla_{W^{(l)}}\mathcal{L},\ \nabla_{\mathbf{b}^{(l)}}\mathcal{L}\}_{l=1}^{L}$

$\boldsymbol{\delta}^{(L)} \leftarrow -(y - \hat{y})$; // MSE: output gradient
for $l = L, L-1, \ldots, 1$ **do**
  $\nabla_{W^{(l)}}\mathcal{L} \leftarrow \mathbf{h}^{(l-1)}(\boldsymbol{\delta}^{(l)})^{\top}$;
  $\nabla_{\mathbf{b}^{(l)}}\mathcal{L} \leftarrow \boldsymbol{\delta}^{(l)}$;
  if $l > 1$ **then**
    $\boldsymbol{\delta}^{(l-1)} \leftarrow W^{(l)}\boldsymbol{\delta}^{(l)} \odot g'_{l-1}(\mathbf{z}^{(l-1)})$;
  end
end
return gradients;

Classwork: Trace the Chain Rule

Try it (5 minutes)

Consider a scalar 2-layer network:

$$z_1 = w_1 x + b_1, \quad h_1 = \max(0, z_1), \quad \hat{y} = w_2 h_1 + b_2, \quad \mathcal{L} = \tfrac{1}{2}(y - \hat{y})^2.$$

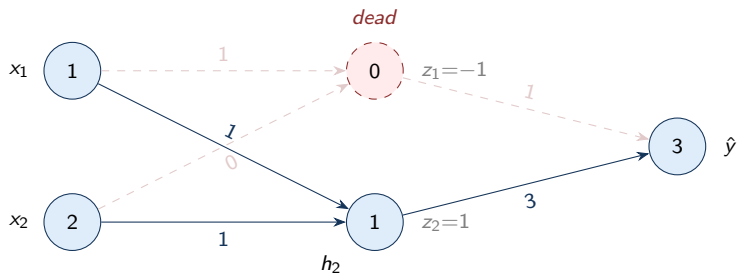
With $x = 2$, $w_1 = 1$, $b_1 = -1$, $w_2 = 3$, $b_2 = 0$, $y = 0$:

1. Compute z_1, h_1, $\hat{y}$, $\mathcal{L}$ (forward pass).
2. Compute $\partial \mathcal{L}/\partial w_2$ and $\partial \mathcal{L}/\partial b_2$.
3. Compute $\delta_1 = \nabla_{z_1} \mathcal{L}$ using the chain rule. Then find $\partial \mathcal{L}/\partial w_1$.

[Hint: $\delta^{(L)} = -(y - \hat{y})$; then $\delta^{(1)} = w_2 \delta^{(L)} \cdot g'(z_1)$.]

Worked Example: The 2-2-1 ReLU Network

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad y = 1, \qquad W^{(1)} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{b}^{(1)} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \quad \mathbf{w}^{(2)} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad b^{(2)} = 0$$



Worked Example: Forward Pass

Step	Quantity	Value
1	$\mathbf{z}^{(1)} = W^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}$	$\begin{bmatrix} 1\cdot 1+0\cdot 2 \\ 1\cdot 1+1\cdot 2 \end{bmatrix} + \begin{bmatrix} -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
2	$\mathbf{h}^{(1)} = \max(0, \mathbf{z}^{(1)})$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ **unit 1 is dead ($z_1 < 0$)**
3	$\hat{y} = \mathbf{w}^{(2)\top}\mathbf{h}^{(1)} + b^{(2)}$	$1 \cdot 0 + 3 \cdot 1 + 0 = 3$
4	$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$	$\frac{1}{2}(1-3)^2 = 2$

Worked Example: Backward Pass

Step	Quantity	Value
1	$\boldsymbol{\delta}^{(2)} = -(y - \hat{y})$	$-(1-3) = 2$
2	$\nabla_{\mathbf{w}^{(2)}} \mathcal{L} = \mathbf{h}^{(1)} \cdot \boldsymbol{\delta}^{(2)}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot 2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$
3	$\nabla_{\mathbf{h}^{(1)}} \mathcal{L} = \mathbf{w}^{(2)} \cdot \boldsymbol{\delta}^{(2)}$	$\begin{bmatrix} 1 \\ 3 \end{bmatrix} \cdot 2 = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$
4	$g'(\mathbf{z}^{(1)})$ (ReLU deriv.)	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ (unit 1 dead)
5	$\boldsymbol{\delta}^{(1)} = \nabla_{\mathbf{h}^{(1)}} \mathcal{L} \odot g'(\mathbf{z}^{(1)})$	$\begin{bmatrix} 2 \\ 6 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \end{bmatrix}$
6	$\nabla_{W^{(1)}} \mathcal{L} = \mathbf{x}(\boldsymbol{\delta}^{(1)})^\top$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 0 & 6 \end{bmatrix} = \begin{bmatrix} 0 & 6 \\ 0 & 12 \end{bmatrix}$

The Dead ReLU

Unit 1 is dead: $z_1 = -1 < 0 \Rightarrow g'(z_1) = 0 \Rightarrow \boldsymbol{\delta}_1^{(1)} = 0.$

Consequence for parameters feeding unit 1:

$$\nabla_{W^{(1)}_{:,1}} \mathcal{L} = \mathbf{x} \cdot \delta_1^{(1)} = \mathbf{0}, \qquad \nabla_{b_1^{(1)}} \mathcal{L} = 0.$$

No gradient reaches the parameters of a dead unit. If a unit is dead on *every* training example, its parameters **never update**: the unit is permanently frozen.

Fixes: careful initialisation, small learning rate, leaky ReLU ($g(z) = \max(\alpha z, z)$ for small $\alpha > 0$).

Classwork: One More Backward Pass

Try it (5 minutes)

Use the same 2-2-1 network (same parameters $W^{(1)}, \mathbf{b}^{(1)}, \mathbf{w}^{(2)}, b^{(2)}$) but now with input $\mathbf{x} = [0, 1]^\top$, $y = 0$.

1. Run the forward pass: compute $\mathbf{z}^{(1)}$, $\mathbf{h}^{(1)}$, $\hat{y}$, and $\mathcal{L}$.
2. Are both hidden units active for this input?
3. Compute $\boldsymbol{\delta}^{(2)}$ and $\boldsymbol{\delta}^{(1)}$.

Key Takeaways

- **Backpropagation** computes all parameter gradients in $O(P)$: one forward pass (store activations) plus one backward pass (propagate deltas).
- **The delta recurrence**: $\boldsymbol{\delta}^{(l)} = W^{(l+1)}\boldsymbol{\delta}^{(l+1)} \odot g_l'(\mathbf{z}^{(l)})$ propagates errors back through the weight matrices, gated by the activation derivative.
- **Parameter gradients** are outer products: $\nabla_{W^{(l)}}\mathcal{L} = \mathbf{h}^{(l-1)}(\boldsymbol{\delta}^{(l)})^\top$.
- **Dead ReLU units** produce zero delta and zero parameter gradient; their parameters cannot be updated.
- **Optimisation**: gradients are used by SGD / minibatch SGD / Adam to update parameters (see Appendix of notes).