PUBLIC LECTURE 4 CHEATSHEET

# From a weighted sum to a neuron

A neural network becomes expressive when affine maps are separated by nonlinear activations.

**The loss says what counts as a good prediction. The network says which predictions are possible.**

## 1 · The linear starting point

One scalar output begins with

$$z = \boldsymbol{w}^\top \boldsymbol{x} + b.$$

Each edge contributes "weight × incoming value"; the node adds the contributions and the bias.

INPUT $\boldsymbol{x} \in \mathbb{R}^d$
WEIGHTS $\boldsymbol{w} \in \mathbb{R}^d$
BIAS $b \in \mathbb{R}$
OUTPUT $z \in \mathbb{R}$

### Concrete weighted sum

For $\boldsymbol{x} = (2, -1)$, $\boldsymbol{w} = (1.5, 0.5)$, and $b = -0.2$:

$$z = 1.5(2) + 0.5(-1) - 0.2 = 2.3.$$

The bias is equivalent to an extra constant feature equal to $1$.

## 2 · Batch form and shape audit

Stack $n$ examples as rows of $\boldsymbol{X} \in \mathbb{R}^{n \times d}$:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w} + b\mathbf{1}.$$

Or append a ones column $\tilde{\boldsymbol{X}}$ and stack the bias into $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{y}} = \tilde{\boldsymbol{X}}\boldsymbol{\theta}.$$

Shape check: $(n \times (d+1))((d+1) \times 1) \to (n \times 1)$.

## 3 · A neuron adds a nonlinear stage

$$z = \boldsymbol{w}^\top \boldsymbol{x} + b, \quad a = \varphi(z).$$

The affine stage chooses an input direction and offset. The activation gates or reshapes the resulting score.

For $\boldsymbol{x} = (2, -1)$, $\boldsymbol{w} = (1, -1)$, $b = 0$:

$$z = 3, \quad a = \mathrm{ReLU}(3) = 3.$$

## 4 · Activation map

| Name | Rule / range | Use and warning |
|---|---|---|
| sigmoid | $\sigma(z) = \frac{1}{1+e^{-z}}$ | probability-like switch; saturates |
| tanh | $\tanh(z) \in (-1, 1)$ | signed switch; saturates |
| ReLU | $\max(0, z)$ | simple ramp; dead for $z < 0$ |
| GELU | $\approx z\sigma(1.702z)$ | smooth, input-dependent gate |

### Saturation is a learning issue

A flat activation has a tiny derivative. For sigmoid,

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)).$$

Near $z = 0$, $\sigma'(0) = 0.25$; near $z = 6$, $\sigma'(6) \approx 0.0025$. The same input nudge produces about $100\times$ less output change in the saturated regime.

## 5 · Why nonlinearity is essential

Two affine layers collapse:

$$\boldsymbol{h} = \boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1, \quad \boldsymbol{y} = \boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2$$

$$\boldsymbol{y} = \underbrace{\boldsymbol{W}_2\boldsymbol{W}_1}_{\boldsymbol{W}_{\text{eq}}}\boldsymbol{x} + \underbrace{\boldsymbol{W}_2\boldsymbol{b}_1 + \boldsymbol{b}_2}_{\boldsymbol{b}_{\text{eq}}}.$$

Without $\varphi$, extra layers add parameters but no new bends or learned feature shapes.

**Affine map = select and shift a direction. Activation = introduce a bend. Composition = build a representation.**

PUBLIC LECTURE 4 CHEATSHEET

# Read an MLP by its shapes and features

Hidden units learn coordinates in which the final prediction can be simpler.

## 1 · One hidden layer

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

With input width $d$, hidden width $m$, and $K$ outputs:

$\boldsymbol{W}_1 \in \mathbb{R}^{m\times d}$ $\quad$ $\boldsymbol{b}_1 \in \mathbb{R}^m$
$\boldsymbol{W}_2 \in \mathbb{R}^{K\times m}$ $\quad$ $\boldsymbol{b}_2 \in \mathbb{R}^K$

$$(m \times d)(d \times 1) \to m, \quad (K \times m)m \to K.$$

Trainable scalars $= m(d+1) + K(m+1)$.

## 2 · XOR: features can solve what one line cannot

A single binary linear score has one straight decision boundary. For the four XOR corners, create

$$h_1 = \mathrm{ReLU}(x_1 + x_2), \quad h_2 = \mathrm{ReLU}(x_1 + x_2 - 1),$$

then use

$$z = 2h_1 - 4h_2 - 1, \quad \hat{y} = \mathbb{1}[z \geq 0].$$

$(0,0) \to 0$ $\quad$ $(1,0) \to 1$ $\quad$ $(0,1) \to 1$ $\quad$ $(1,1) \to 0$

Exact on the four Boolean points does not imply the same rule labels every point inside filled XOR regions. State the input domain.

## 3 · Width and depth answer different needs

**Width:** more units in one layer; more features at the same stage.
**Depth:** more successive transformations; later features reuse earlier features.

Choose width for variety and depth for composition. A layer receives the previous representation, not the original input unless an explicit skip connection supplies it.

## 4 · ReLU networks assemble hinges

One shifted ReLU is a hinge:

$$h_{j(x)} = \mathrm{ReLU}(w_j x + b_j).$$

A linear output combines many such features:

$$g(x) = c + \textstyle\sum_j v_j \, \mathrm{ReLU}(w_j x + b_j).$$

The input weights and biases place the hinges; the output weights choose their slope changes. In higher dimensions, ReLU units fold space along hyperplanes.

## 5 · Universal approximation: read the claim carefully

For a continuous target $f$ on a closed, bounded region $D$ and any $\varepsilon > 0$, a sufficiently wide finite network with a suitable nonlinear activation can satisfy

$$\sup_{x\in D} |f(x) - g(x)| < \varepsilon.$$

This is an **existence and capacity** statement. It does not promise that gradient-based training will find the network, that the network will be small, or that it will generalize beyond $D$.

## 6 · Output-layer contract

| Task | Raw output | Final map |
|---|---|---|
| regression | one or more scores | identity / task constraint |
| binary | one logit $z$ | sigmoid $\sigma(z)$ |
| $K$ classes | $K$ logits | softmax over $K$ |

The hidden representation may stay the same; the number and interpretation of output nodes follows the target.

### Fast architecture audit

- ✓ Write the shape of every value and parameter.
- ✓ Point to the nonlinearity between affine layers.
- ✓ Explain each hidden layer as a learned feature map.
- ✓ Match the output width and transform to the task.
- ✓ Separate representability from trainability.