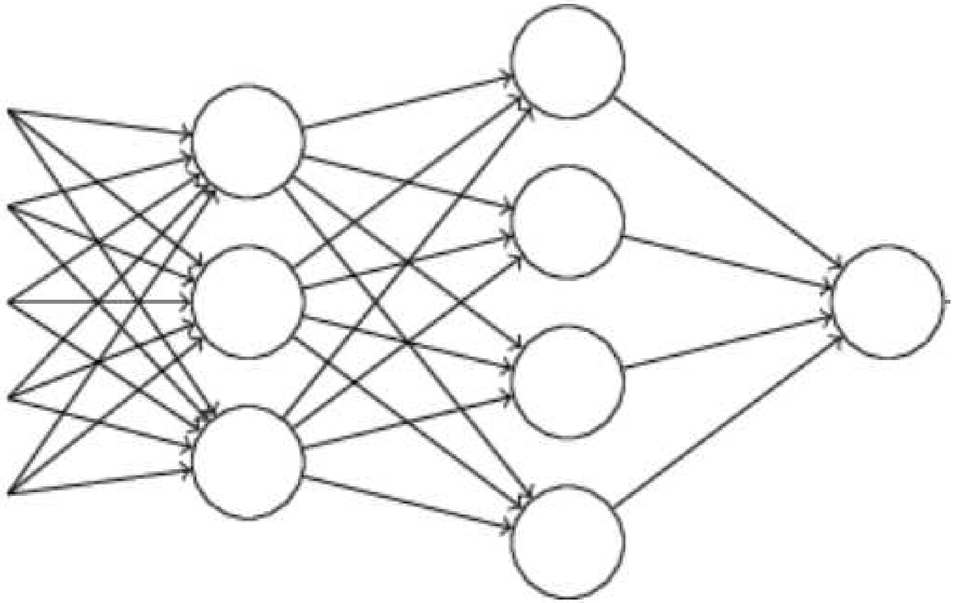
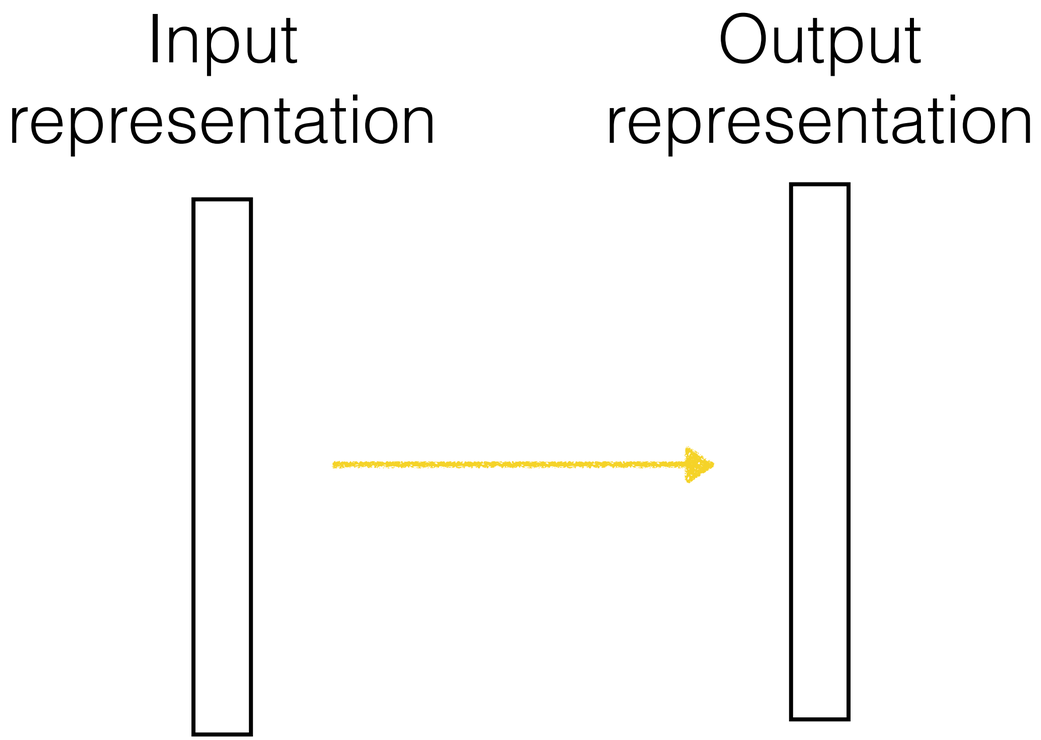
Introduction to Neural Networks: Structure



Computation in a neural net

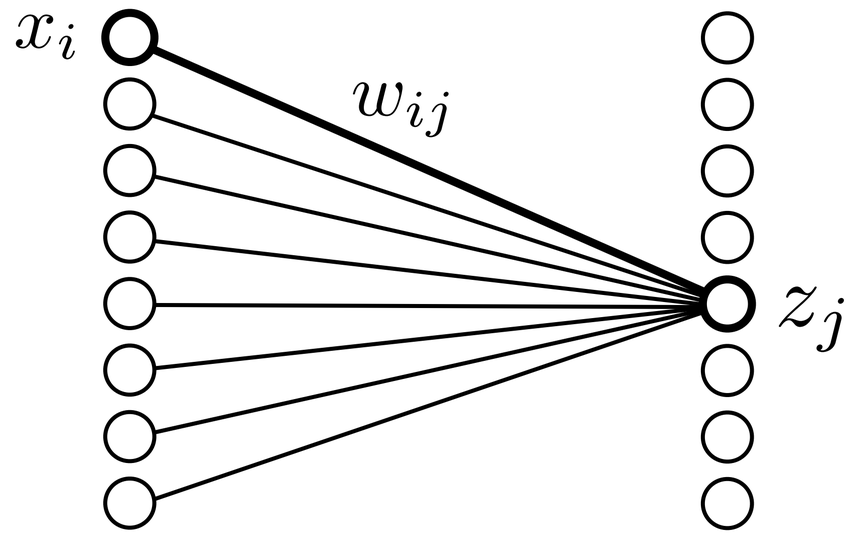


Computation in a neural net

Linear layer

Input representation

Output representation



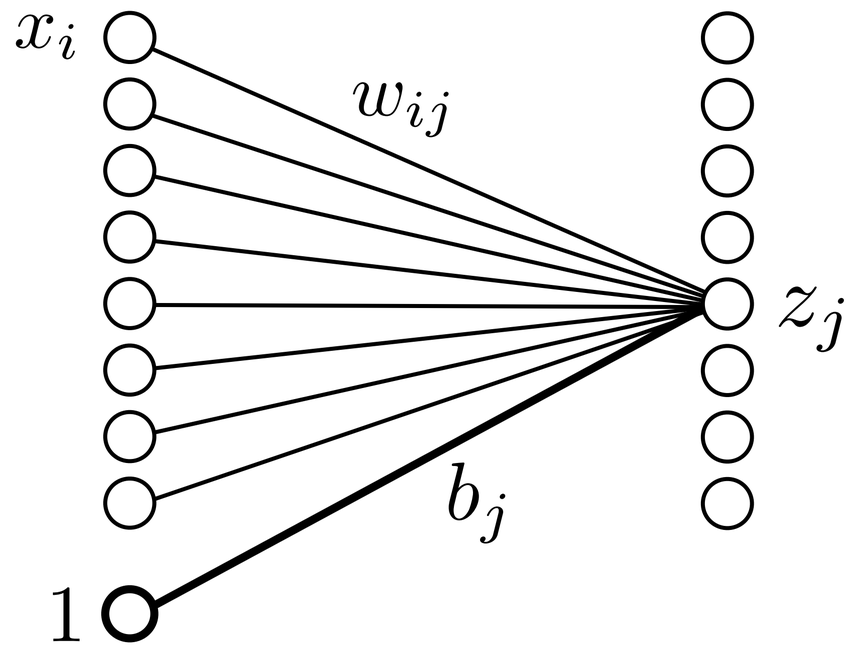
$$z_j = \sum_i w_{ij} x_i$$

Computation in a neural net

Linear layer

Input representation

Output representation



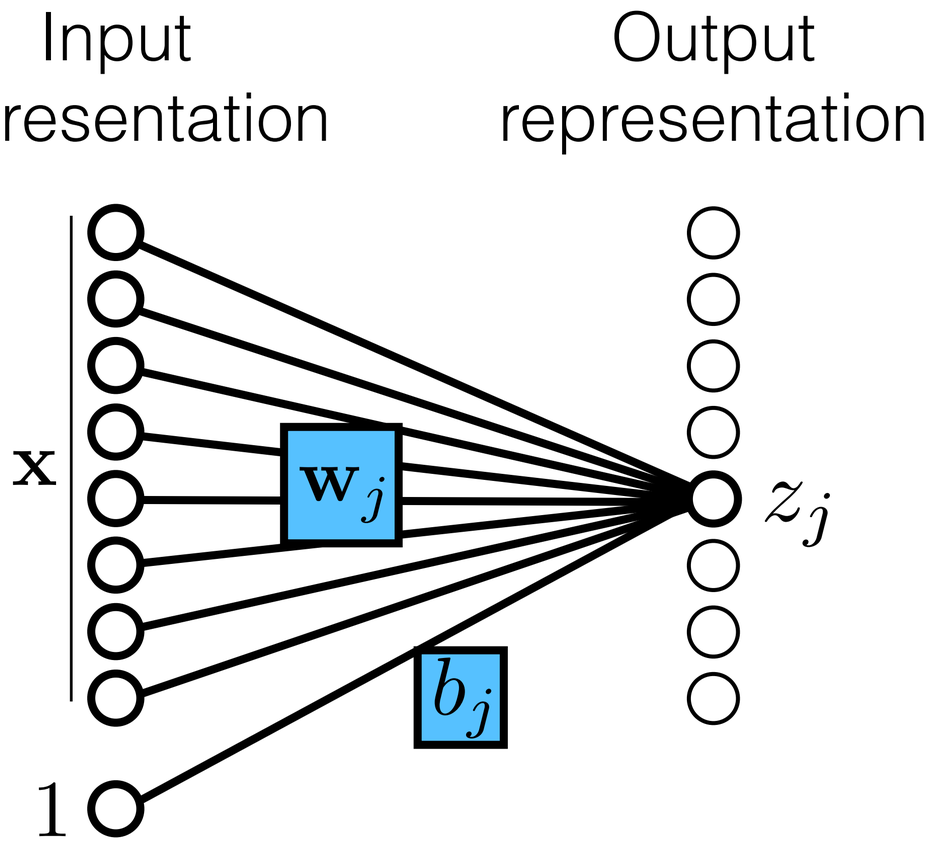
$$z_j = \sum_i w_{ij} x_i + b_j$$

weights

bias

Computation in a neural net

Linear layer



$$z_j = \mathbf{x}^T \mathbf{w}_j + b_j$$

weights

bias

$$\theta = \{\mathbf{W}, \mathbf{b}\}$$

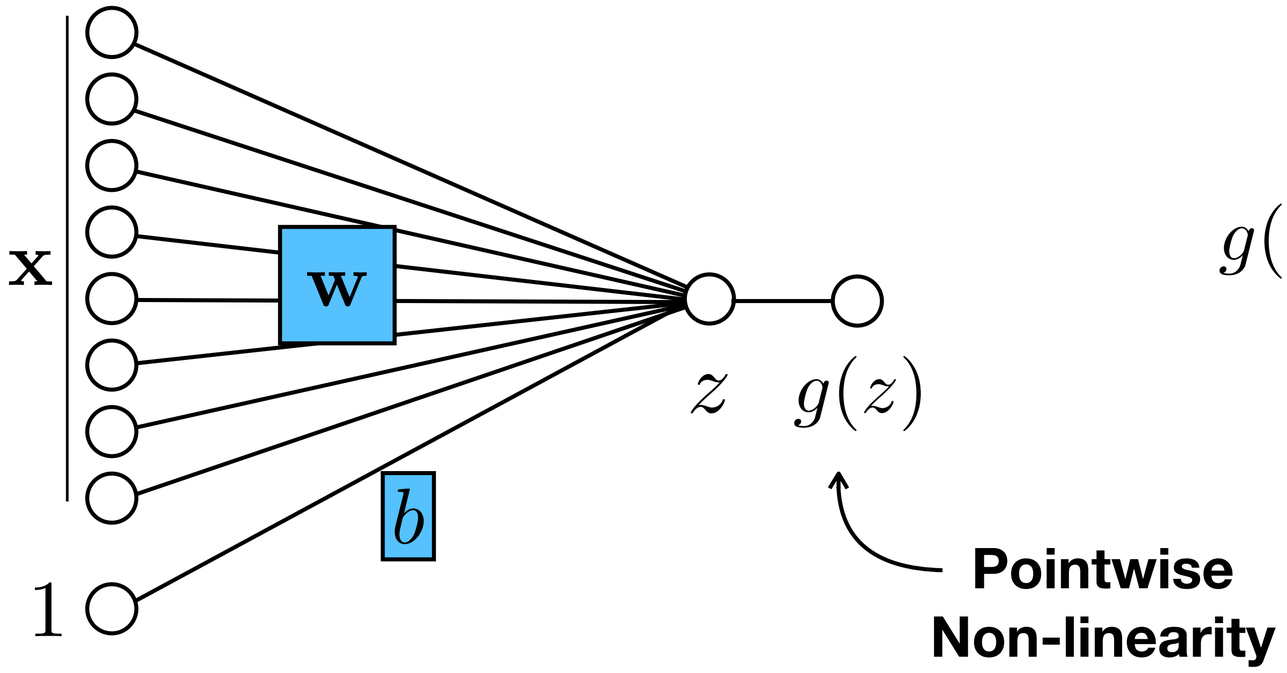
parameters of the model

Computation in a neural net

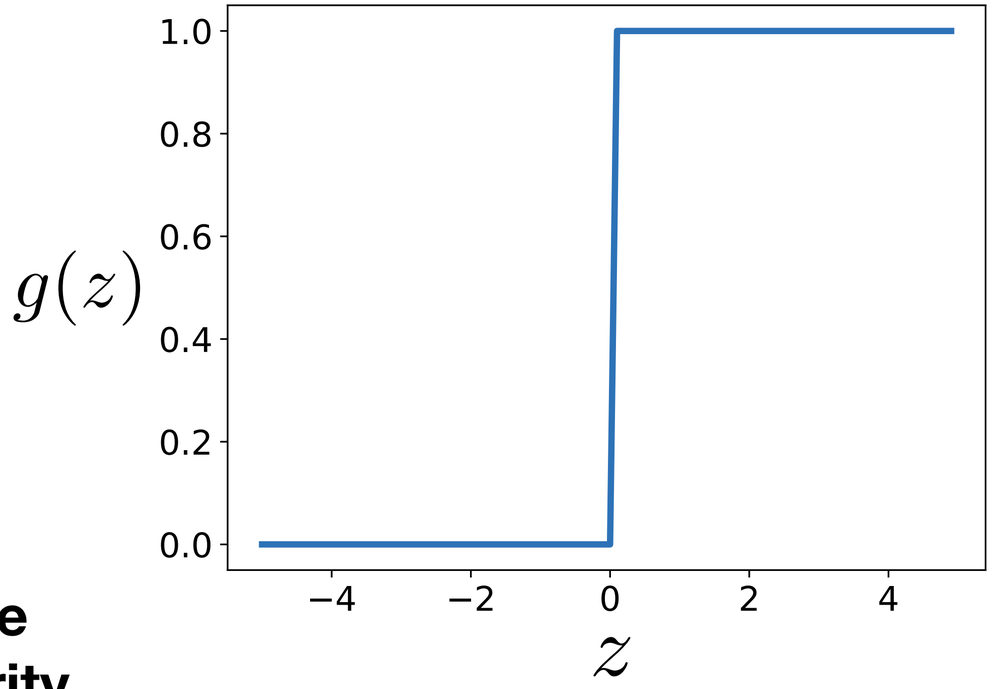
“Perceptron”

Input representation

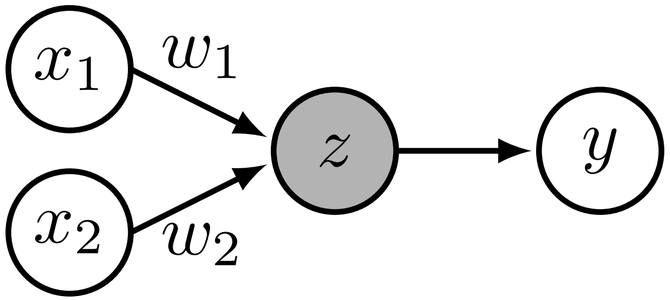
Output representation



$$g(z) = \begin{cases} 1, & \text{if} \quad z > 0 \\ 0, & \text{otherwise} \end{cases}$$

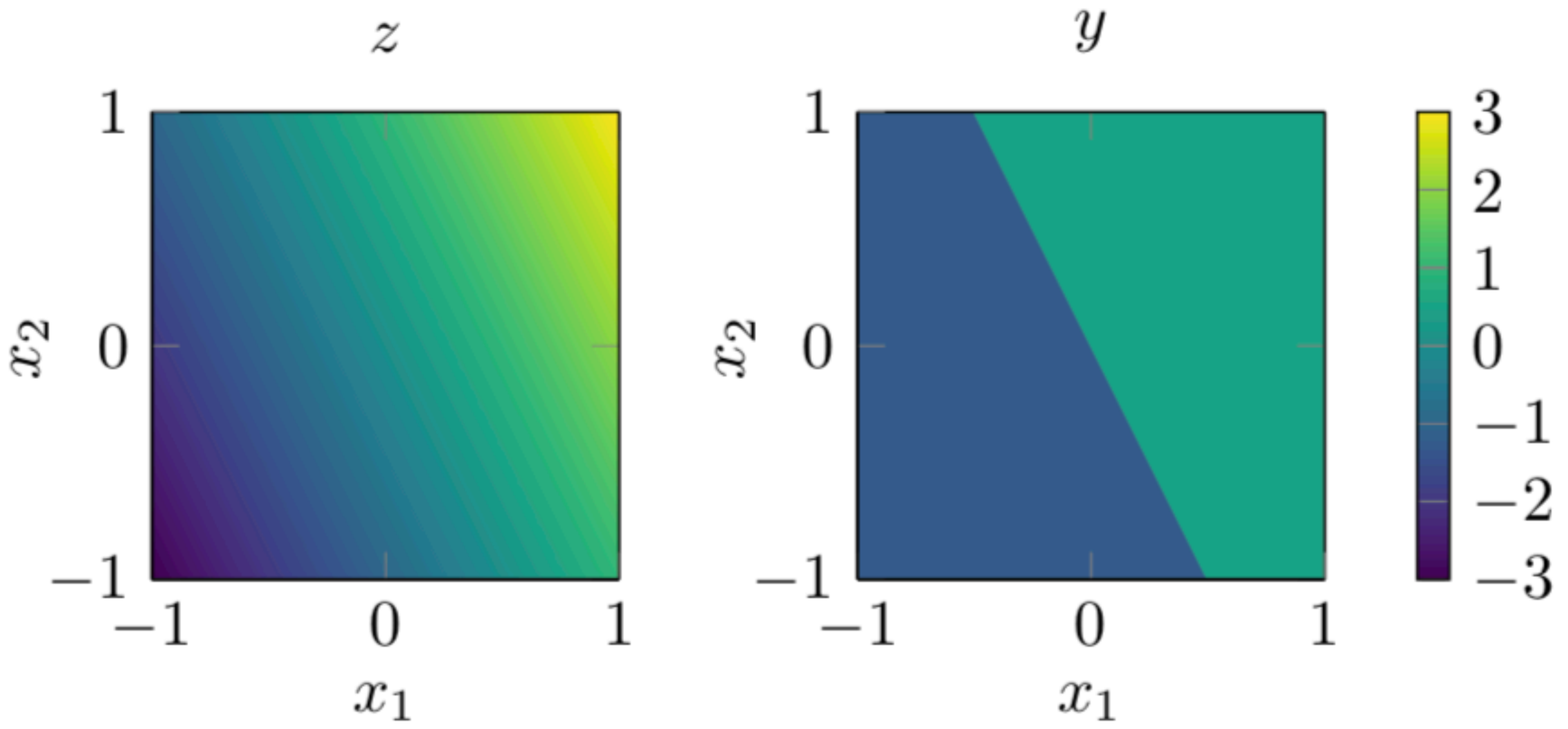


Example: linear classification with a perceptron



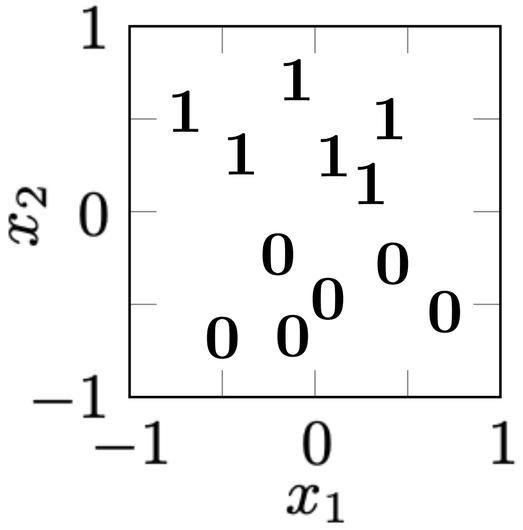
$$z = \mathbf{x}^T \mathbf{w} + b$$

$$y = g(z)$$



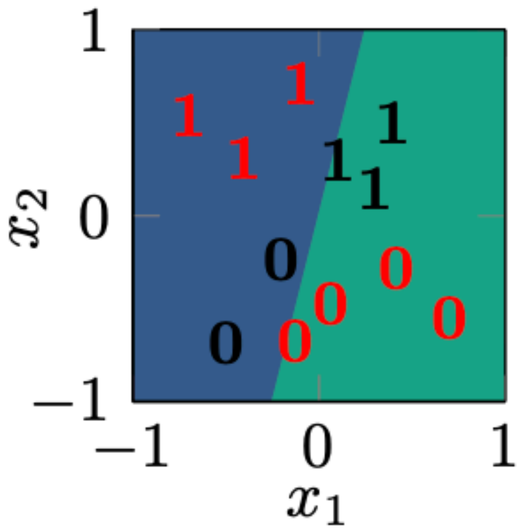
One layer neural net (perceptron) can perform linear classification!

Training data



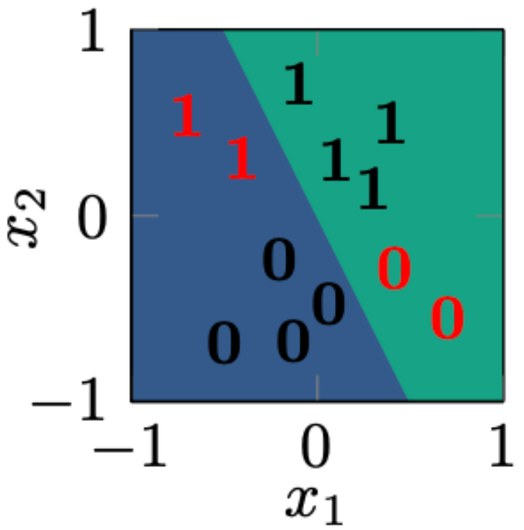
Bad fit

7 misclassifications



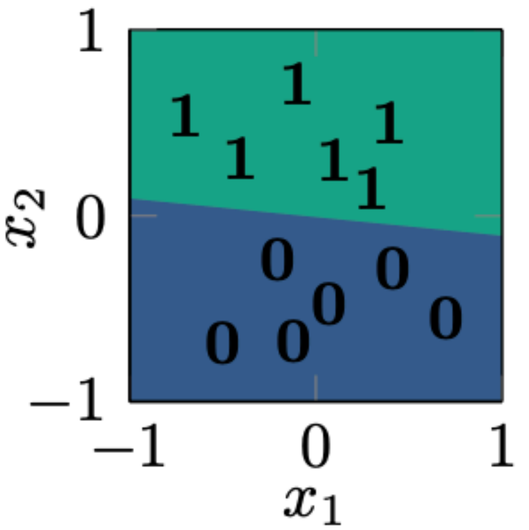
Okay fit

4 misclassifications



Good fit

0 misclassifications

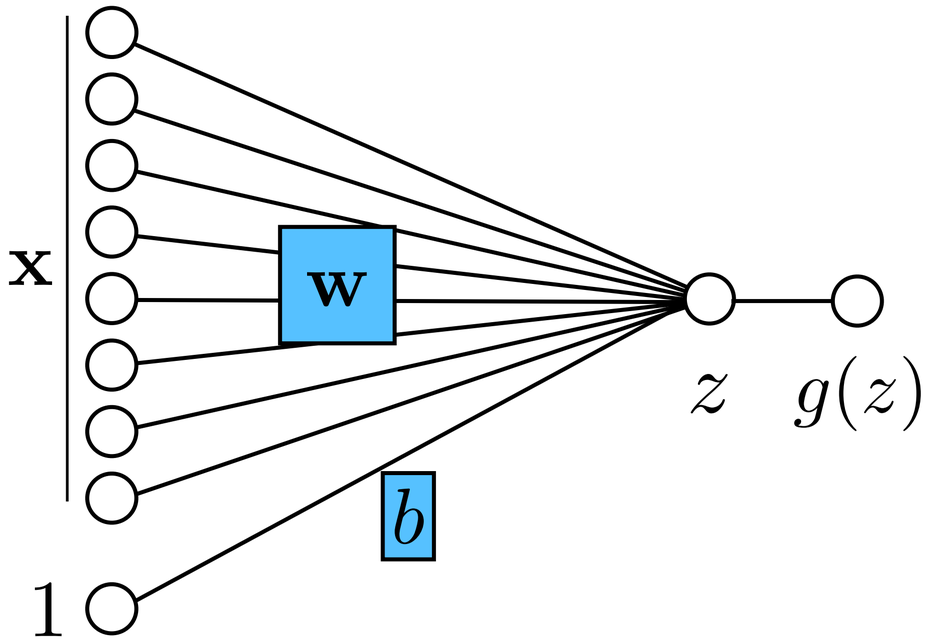


$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \sum_{i=1}^{N} \mathcal{L}(g(z^{(i)}), y^{(i)})$$

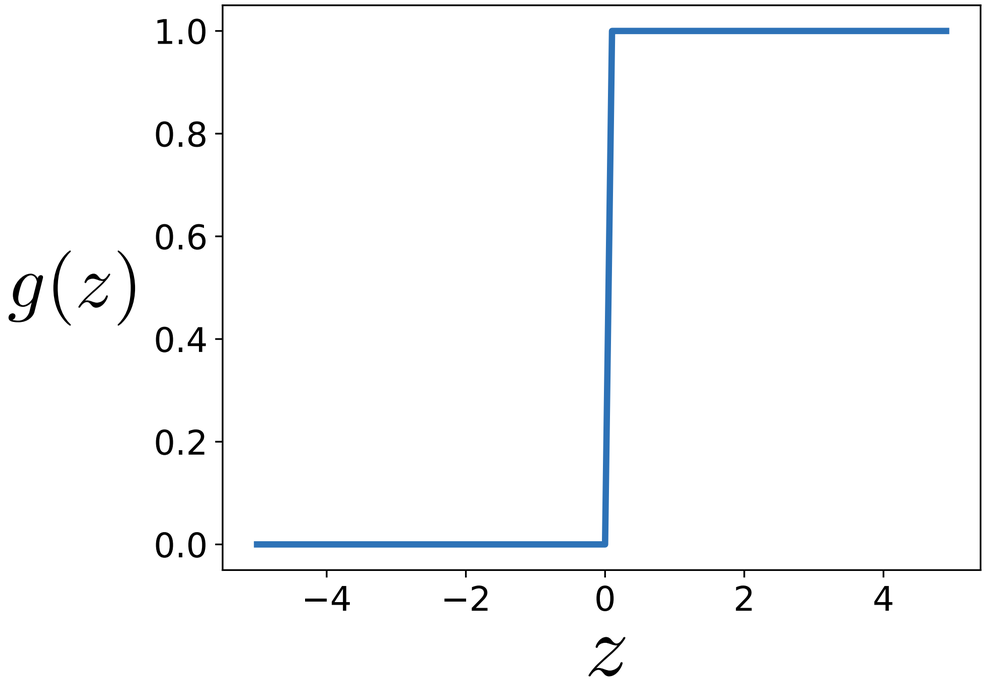
Computation in a neural net

Input representation

Output representation



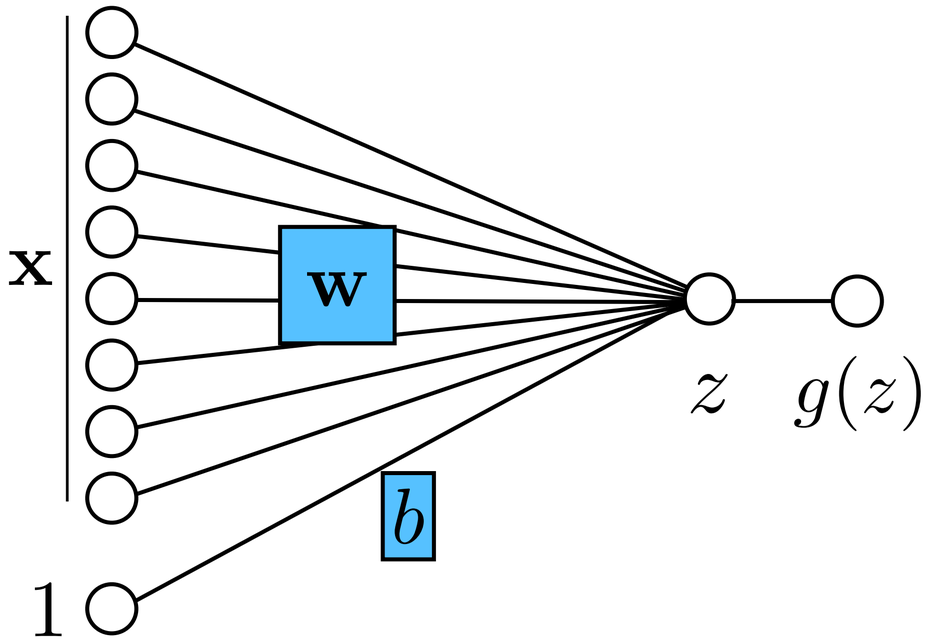
$$g(z) = \begin{cases} 1, & \text{if} \quad z > 0 \\ 0, & \text{otherwise} \end{cases}$$



Computation in a neural net — nonlinearity

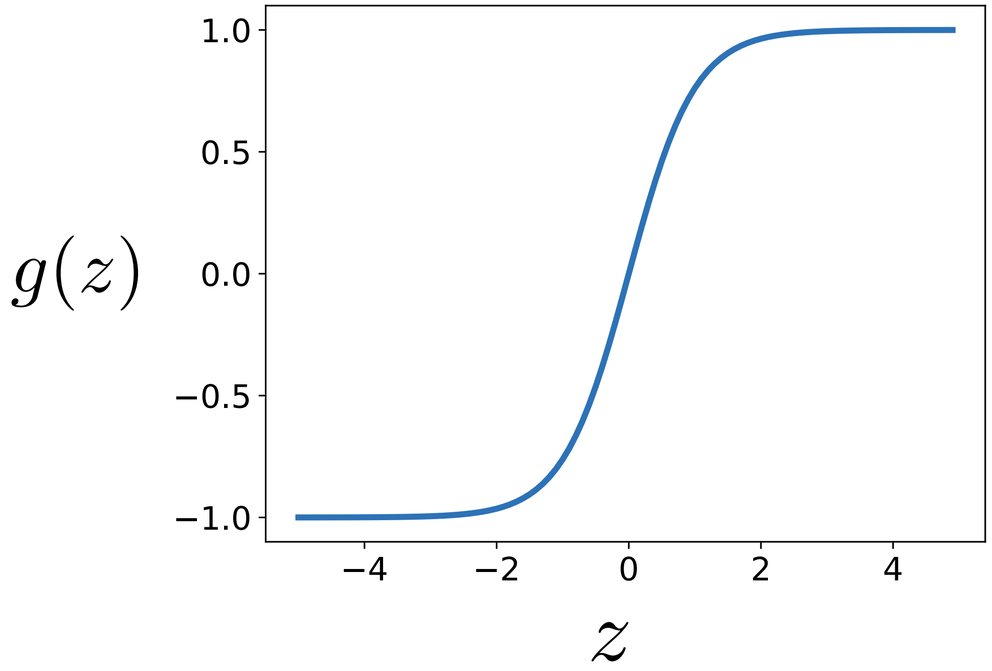
Input representation

Output representation



Tanh

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

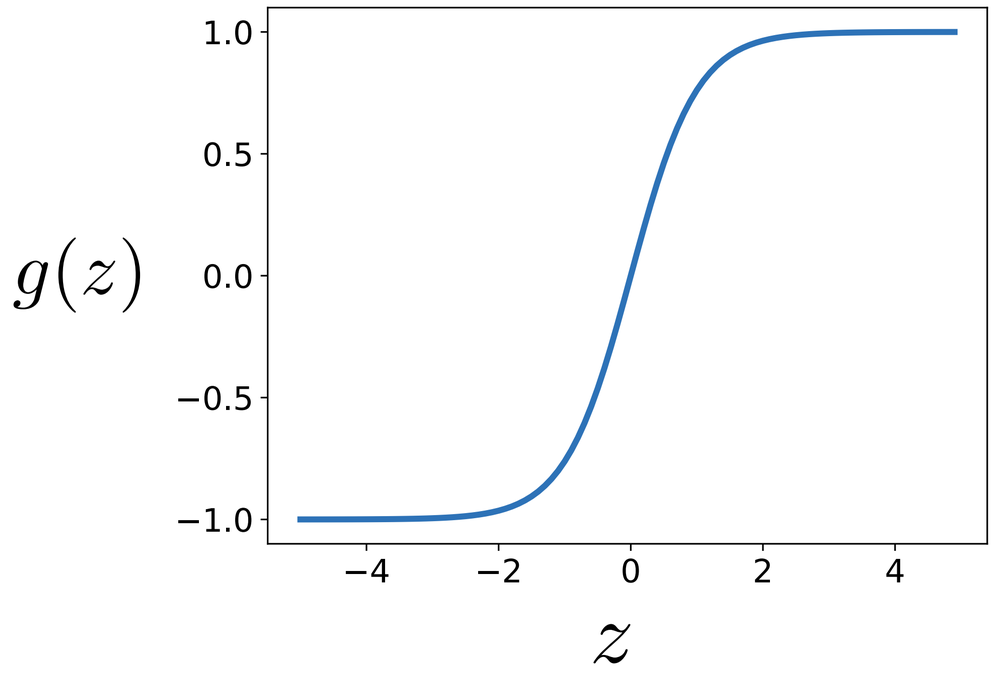


Computation in a neural net — nonlinearity

- Bounded between [-1,+1]
- Saturation for large +/- inputs
- Gradients go to zero
- Outputs centered at 0
- tanh(z) = 2 sigmoid(2z) –1

Tanh

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

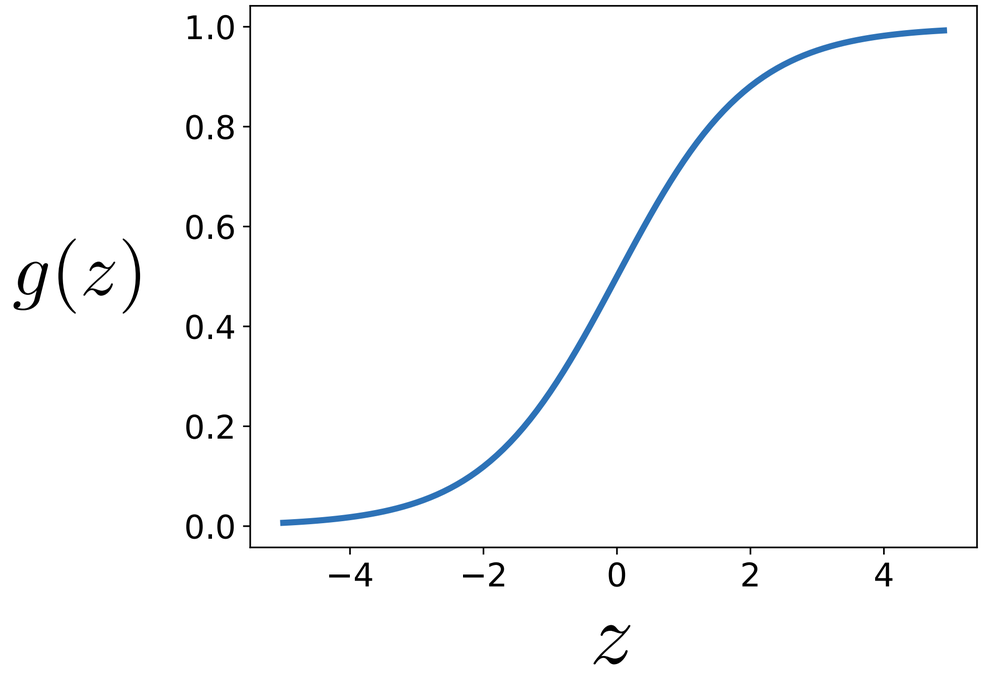


Computation in a neural net — nonlinearity

- Interpretation as firing rate of neuron
- Bounded between [0,1]
- Saturation for large +/- inputs
- Gradients go to zero
- Outputs centered at 0.5
  (poor conditioning)
- Not used in practice

Sigmoid

$$g(z) = \frac{1}{1 + e^{-h}}$$

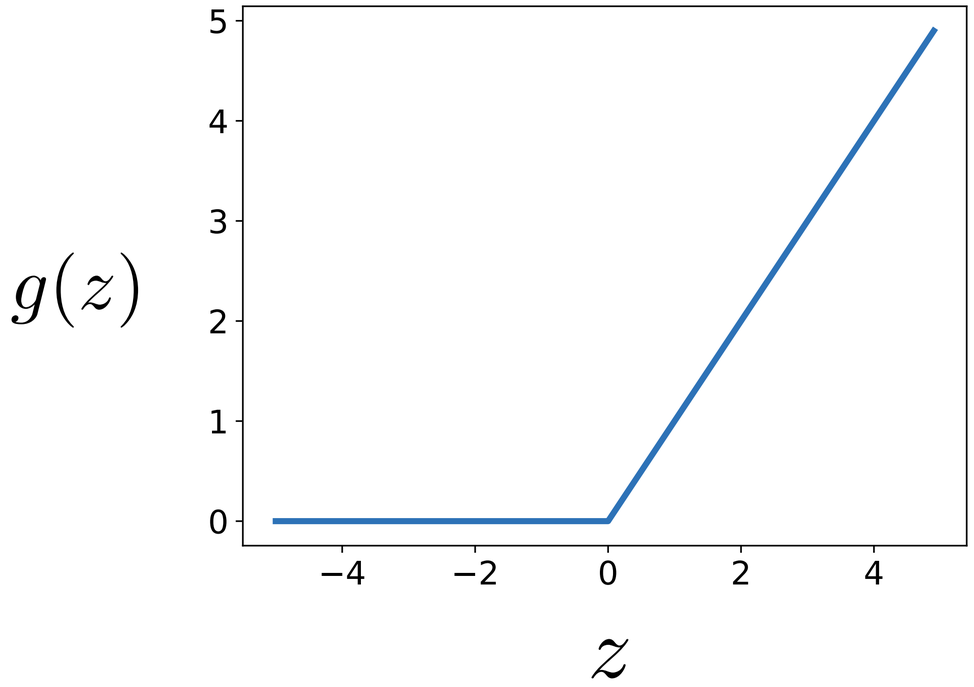


Computation in a neural net — nonlinearity

- Unbounded output (on positive side)
- Efficient to implement: $\frac{\partial g}{\partial z} = \begin{cases} 0, & \text{if} \quad z < 0 \\ 1, & \text{if} \quad z \geq 0 \end{cases}$
- Also seems to help convergence (see 6x speedup vs tanh in [Krizhevsky et al.])
- Drawback: if strongly in negative region, unit is dead forever (no gradient).
- Default choice: widely used in current models.

Rectified linear unit (ReLU)

$$g(z) = \max(0, z)$$

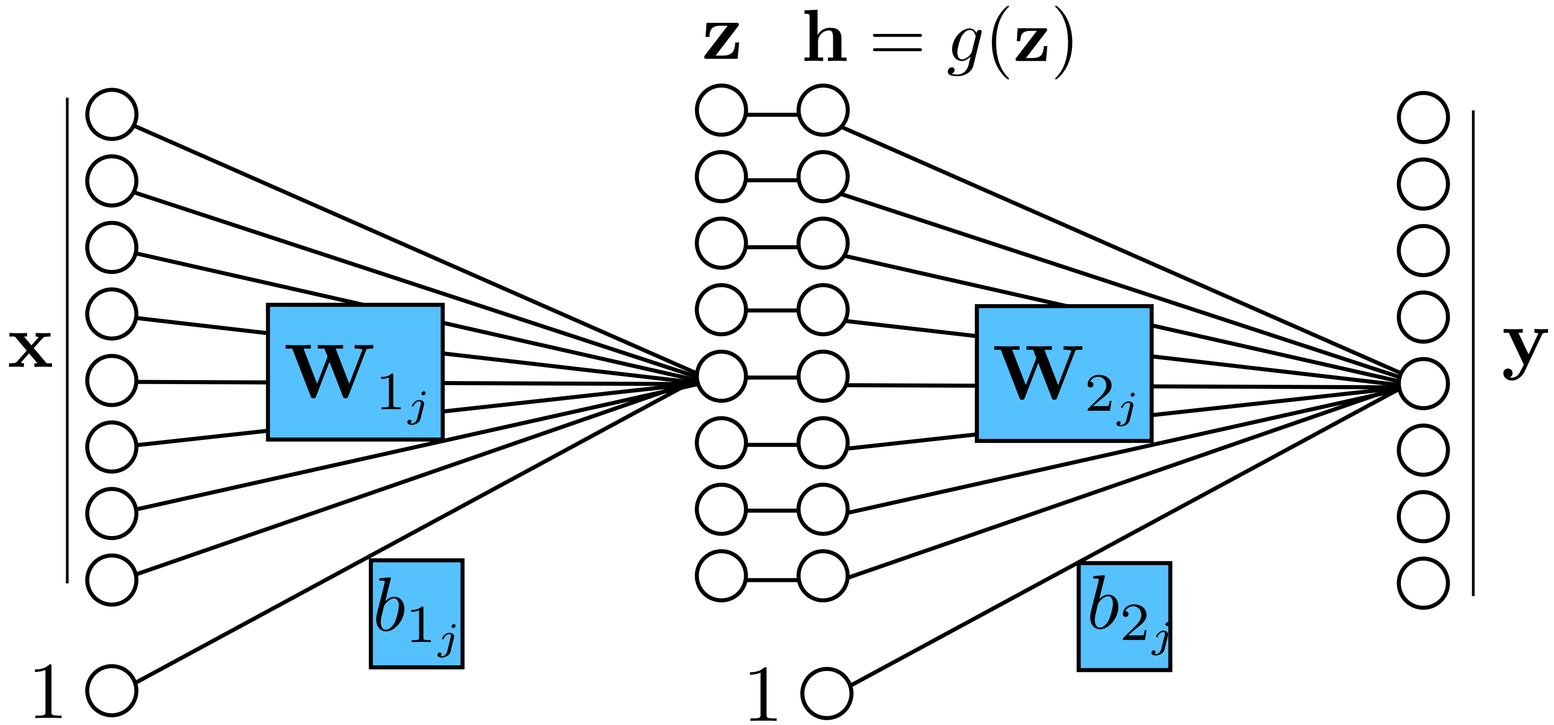


Stacking layers

Input representation

Intermediate representation

Output representation



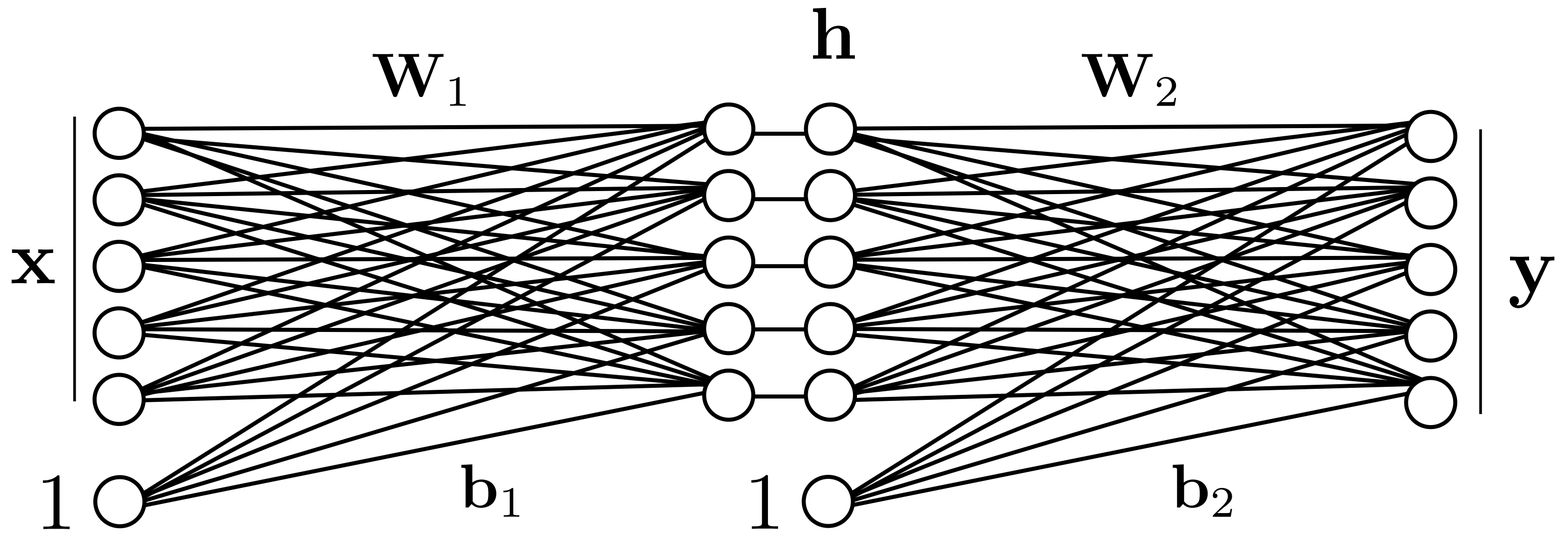
$\mathbf{z}, \mathbf{h}$ = **“hidden units”**

Stacking layers

Input representation

Intermediate representation

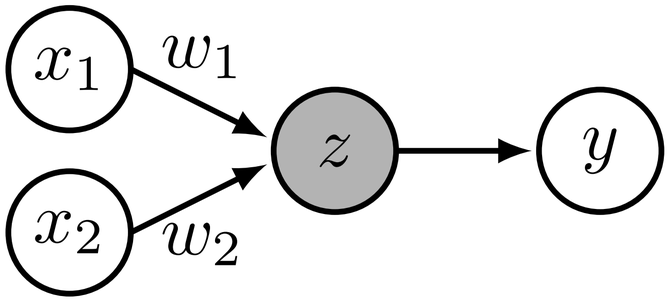
Output representation



$$\mathbf{h} = g(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \qquad \mathbf{y} = g(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2)$$

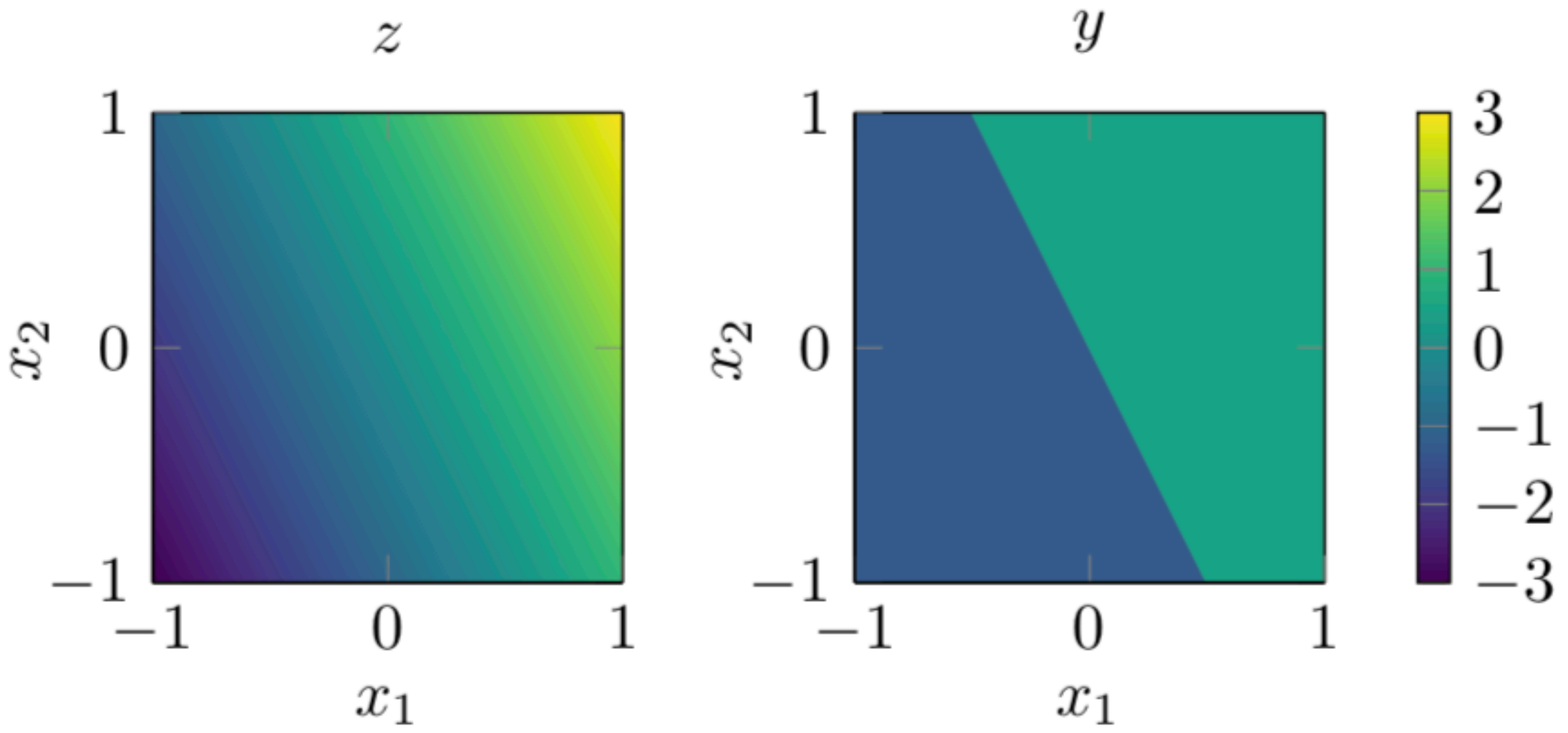
$$\theta = \{\mathbf{W}_1, \ldots, \mathbf{W}_L, \mathbf{b}_1, \ldots, \mathbf{b}_L\}$$

Example: linear classification with a perceptron



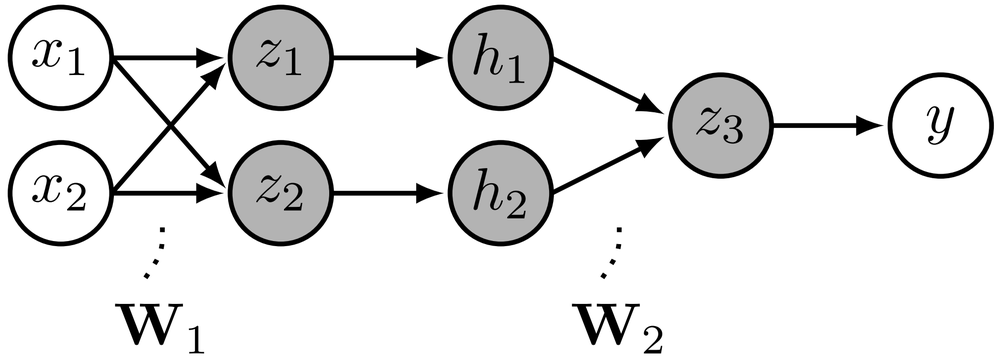
$$z = \mathbf{x}^T \mathbf{w} + b$$

$$y = g(z)$$



One layer neural net (perceptron) can perform linear classification!

Example: nonlinear classification with a deep net net



$$\mathbf{z} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$
$$\mathbf{h} = g(\mathbf{z})$$
$$z_3 = \mathbf{W}_2\mathbf{h} + b_2$$
$$y = 1(z_3 > 0)$$