Lecture 12 –
Introduction to Neural Networks

Prof. Maggie Makar

Announcements

- Sample midterm released today
- Midterm review on Monday 3/4 at 6:30 in DOW 1014
- Monday 3/4 class: *not* a review
- Wednesday 3/6: no class
- Quiz due this Friday (*not* Sunday)

Outline

- Recap AdaBoost
- AdaBoost example
- Neural Networks
  - Motivation
  - From a single neuron to a 3 layer NN
    - Building up notation
    - Building up graphical representation
    - Matrix notation
  - Describing a NN
  - Neural networks as universal approximators

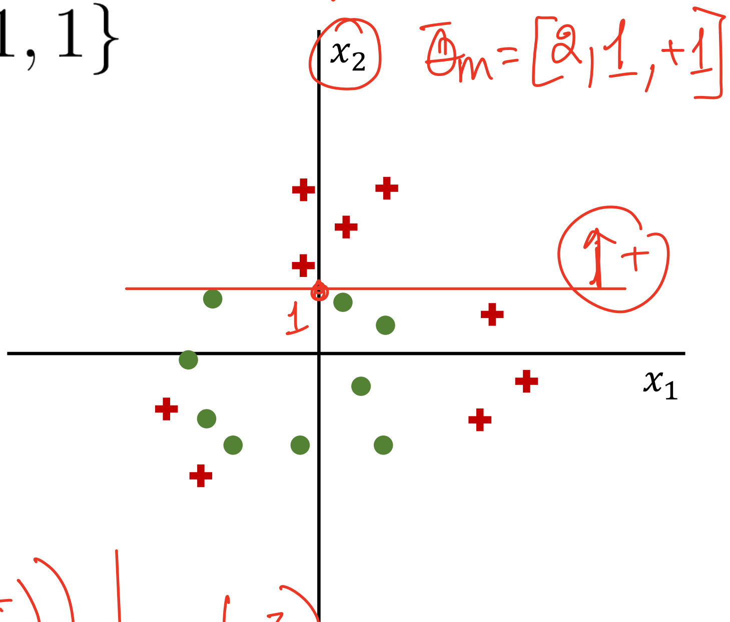
Setup

$\bar{\theta}_m = [d_m, \theta_{0,m}, \theta_{1,m}]$

which feat   split value   +/-

- Training data:

$$S_n = \{\bar{x}^{(i)}, y^{(i)}\}_{i=1}^{n}, \bar{x} \in \mathbb{R}^d, y \in \{-1, 1\}$$



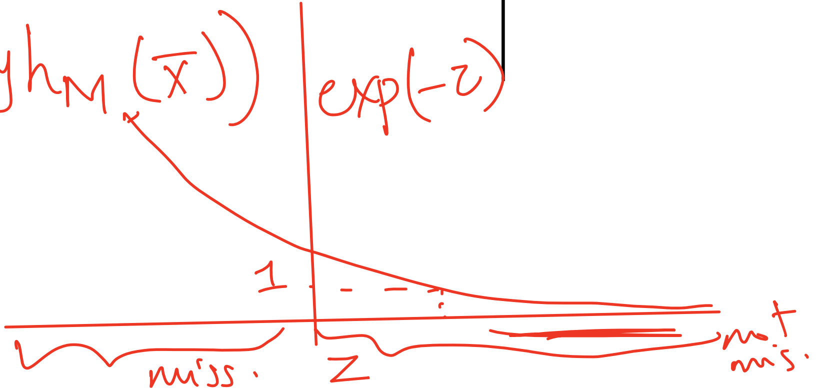
- A set of weak classifiers
  - Stumps with $\leq$ 50% misclassification rate

$$h(\bar{x}; \bar{\theta}_m) = \text{sign}(\theta_{1,m}(x_{d_m} - \theta_{0,m}))$$

- AdaBoost minimizes the exponential loss at each iteration

$$\text{Loss}_{\exp}(z) = \exp(-z) \quad \exp(-y h_M(\bar{x}))$$

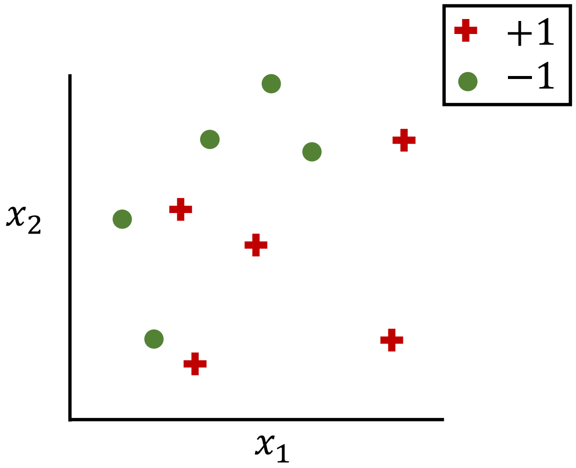
$\exp(-z)$

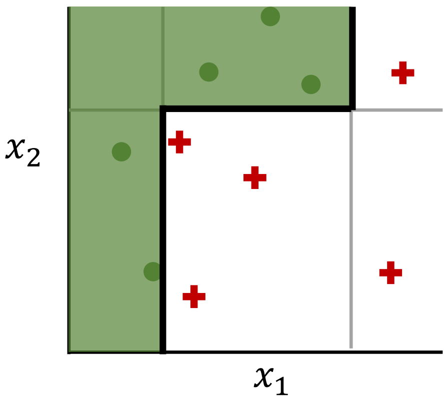


- Final classifier

$$h_M(\bar{x}) = \sum_{m=1}^{M} \alpha_m h(\bar{x}; \bar{\theta}_m)$$

AdaBoost





AdaBoost: the algorithm

weights are normalized to sum to 1

AdaBoost

1. Initialize the observation weights $\widetilde{w}_0^{(i)} = \frac{1}{n}$, for all $i \in [1 \ldots n]$
2. For $m = 1$ to M: XV
   (a) Find: $\bar{\theta}_m = \arg\min_{\bar{\theta}} \sum_{i=1}^{n} \widetilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta})]\!]$
   (b) Given $\bar{\theta}_m$, compute: $\hat{\epsilon}_m = \sum_{i=1}^{n} \widetilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta}_m)]\!]$
   (c) Compute $\alpha_m = \frac{1}{2} \ln \left(\frac{1-\hat{\epsilon}_m}{\hat{\epsilon}_m} \right)$. ≥ 0

   correctly classified rate >0.5

   missclass rate ≤ 0.5

   (d) Update un-normalized weights for all $i \in [1 \ldots n]$:

   $$w_m^{(i)} \leftarrow \widetilde{w}_{m-1}^{(i)} \cdot \exp\left[-y^{(i)} \alpha_m h(\bar{x}^{(i)}; \bar{\theta}_m) \right],$$

   (e) Normalize weights to sum to 1:

   $$\widetilde{w}_m^{(i)} \leftarrow \frac{w_m^{(i)}}{\sum_i w_m^{(i)}} := Z_m$$

3. Output the final classifier: $h_M(\bar{\theta}) = \sum_{m=1}^{M} \alpha_m h(\bar{x}; \bar{\theta}_m)$

At initialization: the cost (aka weight) of misclassifying any point is the same

For every possible stump: evaluate the sum the weights of misclassified points

Find the stump that minimizes the total weights of misclassified points

Compute the resulting weighted misclassification rate

α_m which controls how much we "value" $h(\bar{x}, \bar{\theta}_m)$ is inversely related to $h(\bar{x}, \bar{\theta}_m)$'s error

If i is correctly classified:

$w^{(i)} \rightarrow \widetilde{w}_{m-1}^{(i)} \exp(-\alpha_m)$ <1

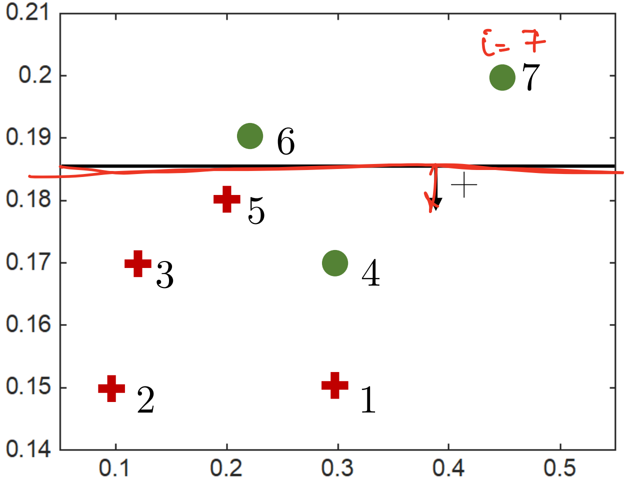
If i is incorrectly classified:

$w^{(i)} \rightarrow \widetilde{w}_{m-1}^{(i)} \exp(\alpha_m)$ >1

AdaBoost: example

+ are positive, ● are negative

- Find $\boxed{\hat{\epsilon}_1}$ and $\boxed{\alpha_1}$
- Find $\tilde{w}_1^{(i)}$ for $i = 1, \ldots, n$



AdaBoost

1. Initialize the observation weights $\tilde{w}_0^{(i)} = \frac{1}{n}$, for all $i \in [1 \ldots n]$
2. For $m = 1$ to M:
   (a) Find: $\bar{\theta}_m = \arg\min_{\bar{\theta}} \sum_{i=1}^n \tilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta})]\!]$
   (b) Given $\bar{\theta}_m$, compute: $\hat{\epsilon}_m = \sum_{i=1}^n \tilde{w}_{m-1}^{(i)} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta}_m)]\!]$.
   (c) Compute $\alpha_m = \frac{1}{2} \ln \left(\frac{1-\hat{\epsilon}_m}{\hat{\epsilon}_m} \right)$.
   (d) Update weights on all training examples, for all $i \in [1 \ldots n]$:
   $$\tilde{w}_m^{(i)} \leftarrow \frac{\tilde{w}_{m-1}^{(i)} \cdot \exp\left[-y^{(i)} \alpha_m h(\bar{x}^{(i)}; \bar{\theta}_m) \right]}{Z_m},$$
3. Output the final classifier: $h_M(\bar{\theta}) = \sum_{m=1}^M \alpha_m h(\bar{x}; \bar{\theta}_m)$

AdaBoost example solution

$\hat{\epsilon}_1 = \tilde{w}_0^{(1)} [\![y^{(1)} \neq h(\bar{x}^{(1)}; \bar{\theta}_1)]\!] + \cdots \tilde{w}_0^{(7)} [\![y^{(7)} \neq h(\bar{x}^{(7)}; \bar{\theta}_1)]\!]$

$= \frac{1}{7}(0) + \frac{1}{7}(0) + \cdots \frac{1}{7}(1) + \cdots = \frac{1}{7}$

$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-1/7}{1/7}\right) = \frac{1}{2}\ln(6) = \ln\sqrt{6}$

✚ are positive, ● are negative

- Find $\hat{\epsilon}_1$ and $\boxed{\alpha_1}$
- Find $\tilde{w}_1^{(i)}$ for $i = 1, \dots, n$

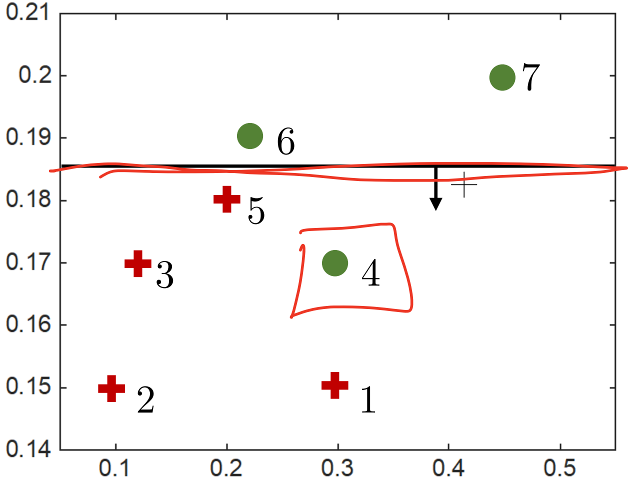
Unnormalized weights:

$i \neq 4$

$w_1^{(i)} = \tilde{w}_0^{(i)} \exp(-\alpha_1) = \frac{1}{7}\exp(-\ln\sqrt{6}) = \frac{1}{7\sqrt{6}}$

$i = 4$

$w_1^{(4)} = \tilde{w}_0^{(i)} \exp(\alpha_1) = \frac{1}{7}\exp(\ln\sqrt{6}) = \frac{\sqrt{6}}{7}$



Z_m

$Z_m = 6\left(\frac{1}{7\sqrt{6}}\right) + 1\left(\frac{\sqrt{6}}{7}\right) = \frac{2\sqrt{6}}{7}$

Normalized weights

for $i \neq 4$: $\tilde{w}_1^{(i)} = \frac{1}{7\sqrt{6}}\left(\frac{7}{2\sqrt{6}}\right) = 1/12$ | for $i = 4$ $\tilde{w}_1^{(4)} = \sqrt{6}/7\,(7/2\sqrt{6}) = \boxed{1/2}$

$\alpha = 0$

Neural Networks

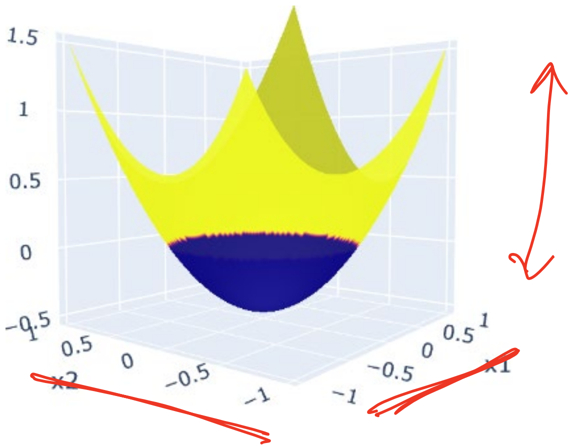
Not covered in midterm

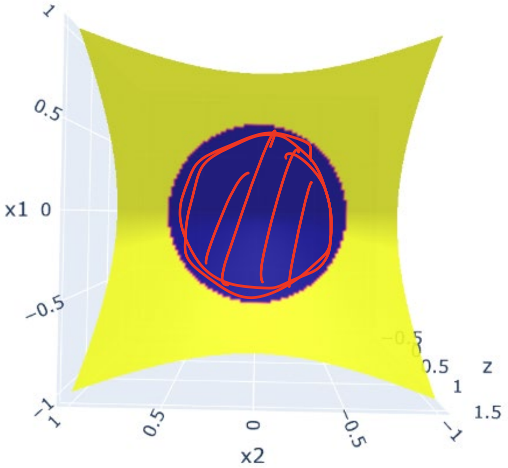
Recall feature mappings

1. Explicit feature mappings
   a) Polynomial kernels with feature mappings:

$$\phi(\bar{x}) = [x^0, x^1, ..., x^p]^\top$$

$$f(\bar{x};\theta) = \theta_0 x^0 + \theta_1 x^1 + \ldots + \theta_p x^p$$





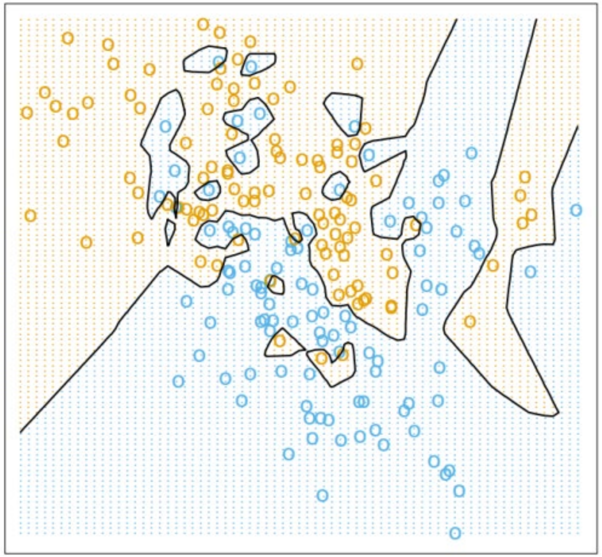
Recall feature mappings

1. Explicit feature mappings
   a) Polynomial kernels with feature mappings:

   b) Radial Basis function:

don't need to know details

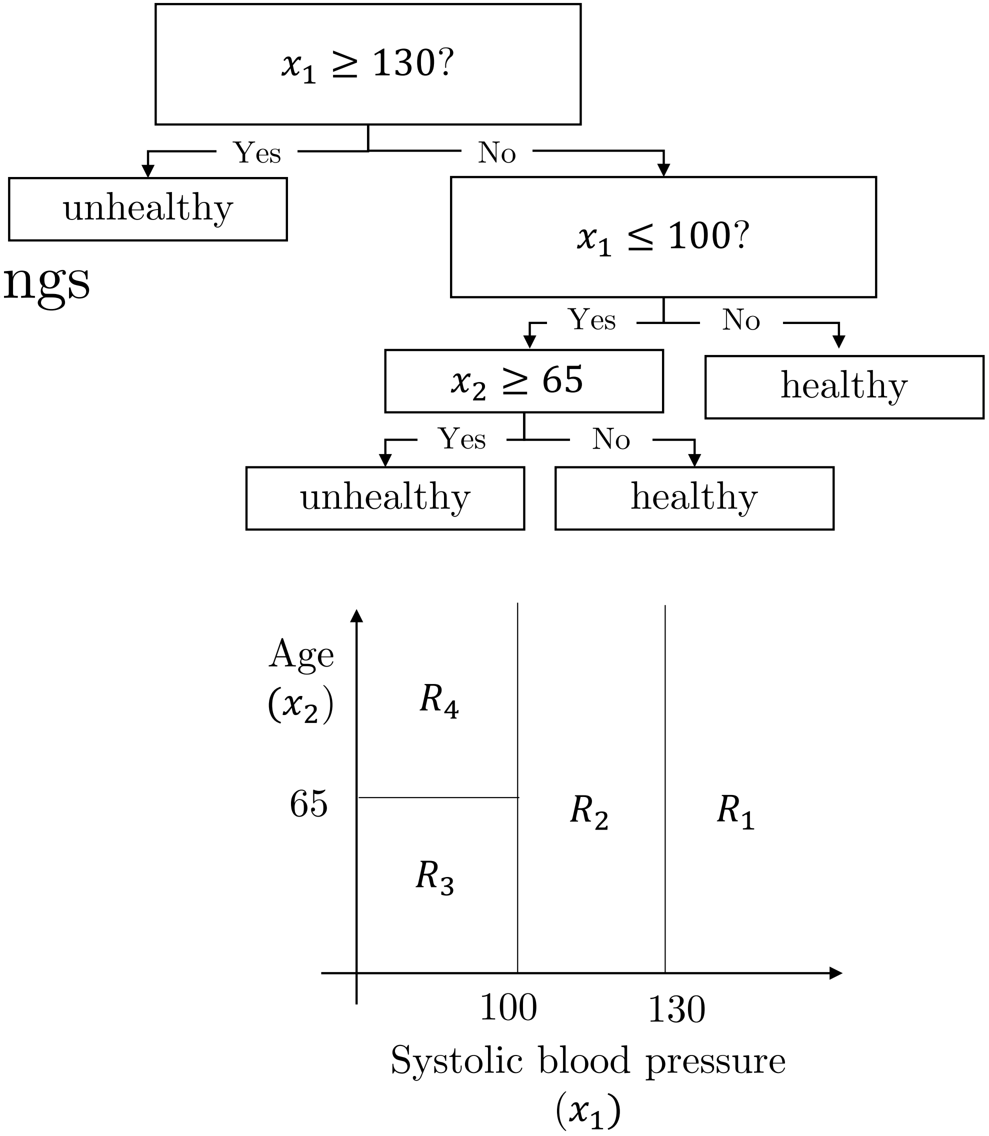
$$K(\bar{x}^{(i)}, \bar{x}^{(j)}) = \exp(-\gamma||\bar{x}^{(i)} - \bar{x}^{(j)}||^2)$$
$$= \sum_{\ell=1}^{\infty} \phi_\ell(\bar{x}^{(i)}) \cdot \phi_\ell(\bar{x}^{(j)})$$
$$= \sum_{\ell=1}^{\infty} (\sqrt{(}\lambda_\ell)e_\ell(\bar{x}^{(i)})) \cdot (\sqrt{(}\lambda_\ell)e_\ell(\bar{x}^{(j)}))$$



Recall feature mappings

1. Explicit feature mappings
2. Implicit (learned) feature mappings
   a) Decision trees

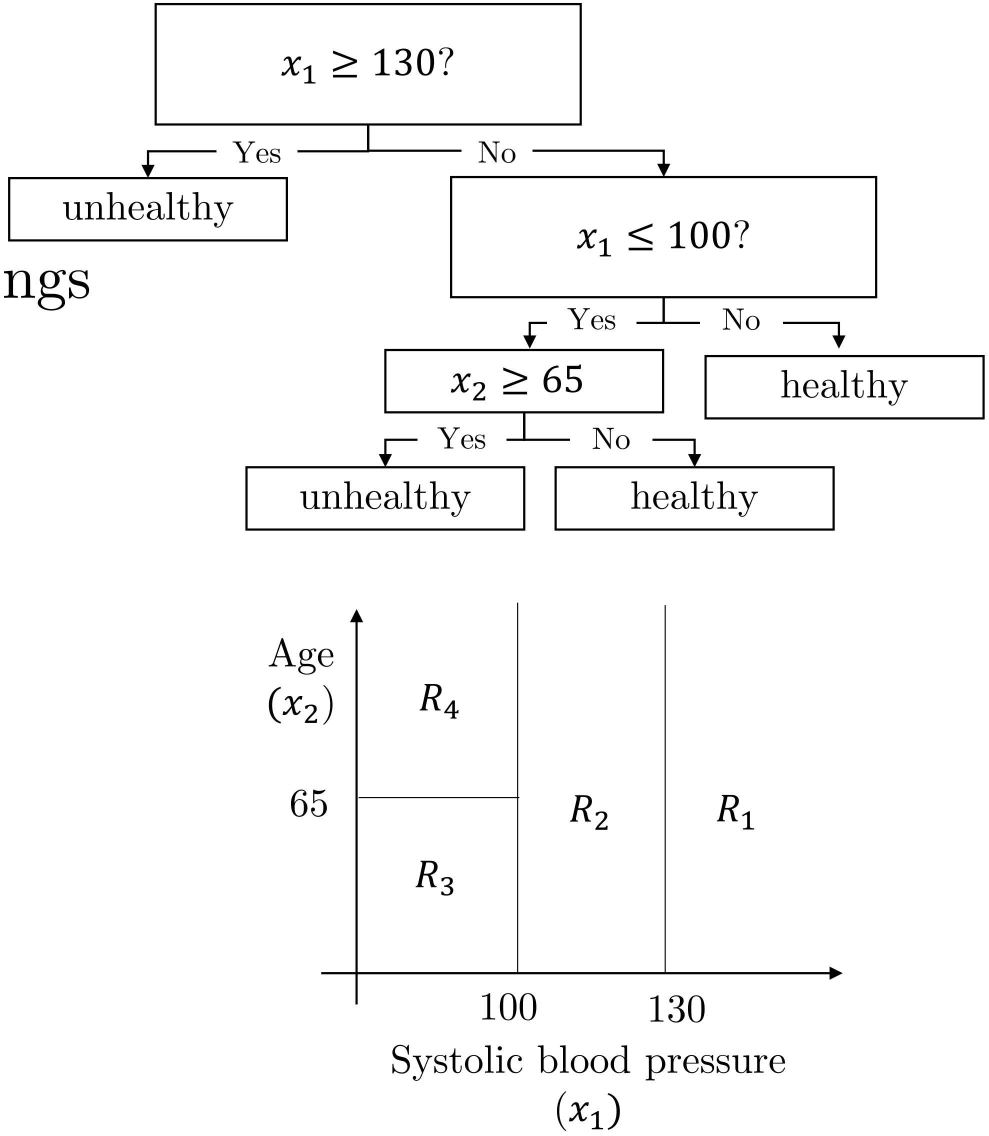
$$f(\bar{x}) = \sum_{m=1}^{M} \mu_m [\![\bar{x} \in R_m]\!]$$



Recall feature mappings

1. Explicit feature mappings
2. Implicit (learned) feature mappings
   a) Decision trees

$$f(\bar{x}) = \sum_{m=1}^{M} \mu_m [\![\bar{x} \in R_m]\!]$$
$$= \sum_{m=1}^{M} \mu_m \phi_m(\bar{x})$$



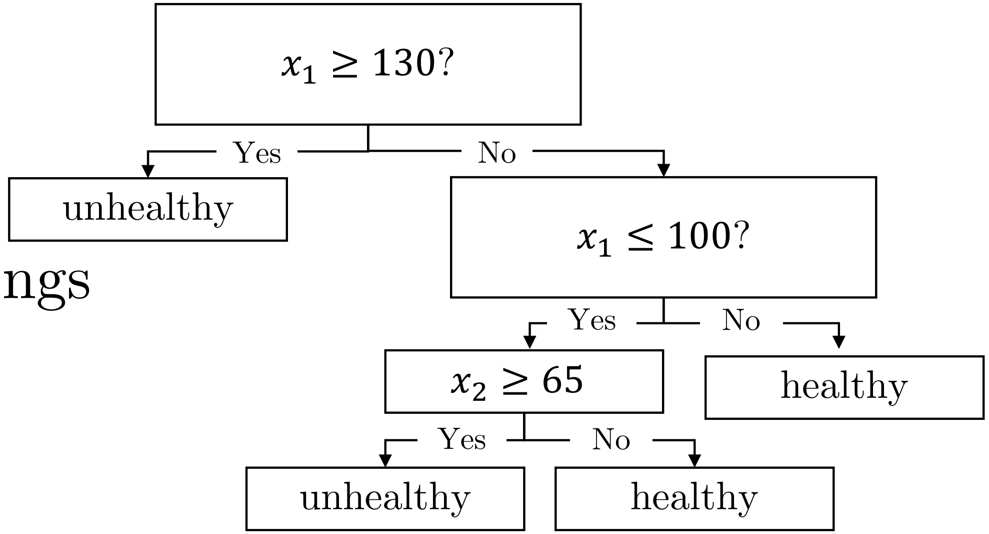
Recall feature mappings

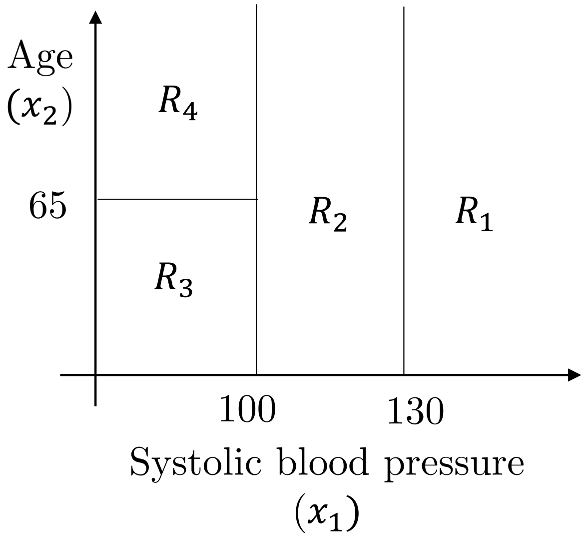
1. Explicit feature mappings
2. Implicit (learned) feature mappings
   a) Decision trees

$$f(\bar{x}) = \sum_{m=1}^{M} \mu_m [\![\bar{x} \in R_m]\!]$$

$$= \sum_{m=1}^{M} \mu_m \phi_m(\bar{x})$$

$$= \sum_{m=1}^{M} \theta_m \phi_m(\bar{x})$$





Recall feature mappings

1. Explicit feature mappings
2. Implicit (learned) feature mappings
   a) Decision trees

   b) Boosting

   $$h_M(\bar{x}) = \alpha_1 h(\bar{x};\bar{\theta}_1) + \alpha_2 h(\bar{x};\bar{\theta}_2) + \ldots + \alpha_M h(\bar{x};\bar{\theta}_M)$$

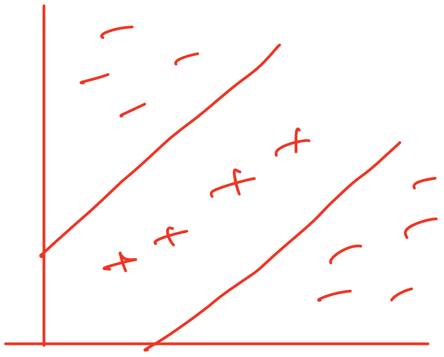
Recall feature mappings

1. Explicit feature mappings
2. Implicit (learned) feature mappings
   a) Decision trees

   b) Boosting

$$\begin{aligned} h_M(\bar{x}) &= \alpha_1 h(\bar{x};\bar{\theta}_1) + \alpha_2 h(\bar{x};\bar{\theta}_2) + \ldots + \alpha_M h(\bar{x};\bar{\theta}_M) \\ &= \alpha_1 \phi_1(\bar{x}) \quad + \alpha_2 \phi_2(\bar{x}) \quad + \ldots + \alpha_M \phi_M(\bar{x}) \end{aligned}$$

Recall feature mappings



1. Explicit feature mappings
2. Implicit (learned) feature mappings
   a) Decision trees

   b) Boosting

$$
\begin{aligned}
h_M(\bar{x}) &= \alpha_1 h(\bar{x};\bar{\theta}_1) + \alpha_2 h(\bar{x};\bar{\theta}_2) + \ldots + \alpha_M h(\bar{x};\bar{\theta}_M \\
&= \alpha_1 \phi_1(\bar{x}) \quad + \alpha_2 \phi_2(\bar{x}) \quad + \ldots + \alpha_M \phi_M(\bar{x}) \\
&= \theta_1 \phi_1(\bar{x}) \quad + \theta_2 \phi_2(\bar{x}) \quad + \ldots + \theta_M \phi_M(\bar{x})
\end{aligned}
$$

Towards very flexible feature mappings: Feedforward neural networks

hidden neurons.

- Instead of ϕ, we will use h
- How do we make our learned h flexible?
  - Allow them to be non-linear
  - Give them learnable parameters
  - Combining many learned h's

TL;DPA:

We can view everything we've done so far as learning feature mappings/representations that allow us to get better predictions

Neural networks can give us *very* flexible feature representations compared to what we've studied before.

Learning a single hidden neuron

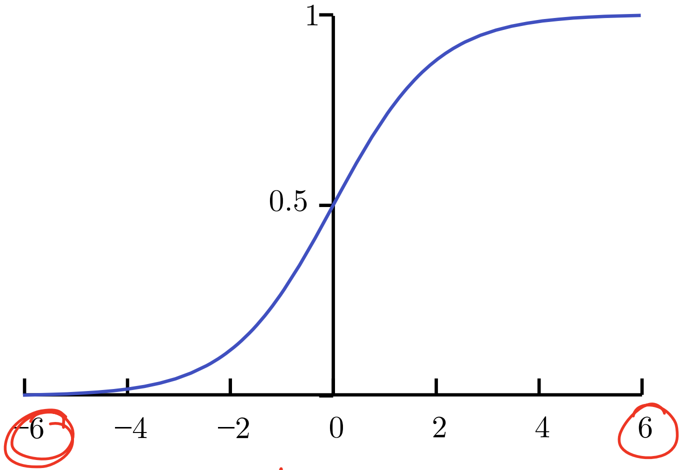
- $\bar{x} = [x_1\ x_2]^\mathsf{T}$ $= [x_1\ x_2\ 1]$

  $y \in \{0, 1\}$
- $z(\bar{x}; \bar{w}) = w_1 x_1 + w_2 x_2 + w_0$ $\to$ bias
- $h(z) = \mathrm{NonLinear}(z)$

Learning a single hidden neuron – activation function

- $\bar{x} = [x_1\ x_2]^\top$

- $z(\bar{x}; \overline{w}) = w_1 x_1 + w_2 x_2 + w_0$
- $h(z) = \frac{1}{1+e^{-z}} = \sigma(z)$

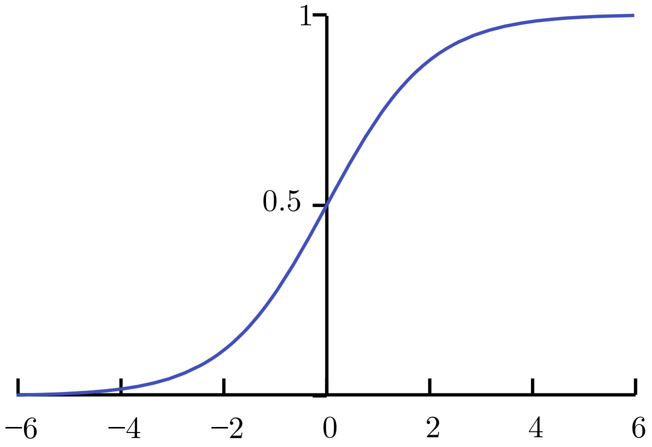


sigmoid

Learning a single hidden neuron – activation function

- $\bar{x} = [x_1\ x_2]^\top$

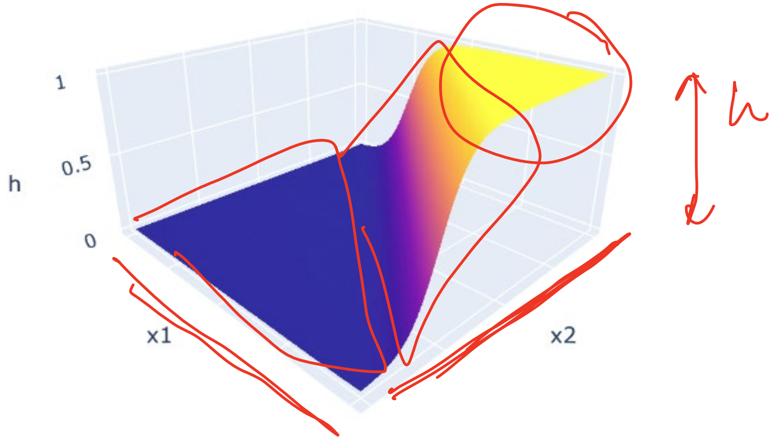
- $z = w_1 x_1 + w_2 x_2 + w_0$
- $h = \frac{1}{1+e^{-z}} = \sigma(z)$



Learning a single hidden neuron

- $\bar{x} = [x_1\ x_2]^\top$

- $z = w_1 x_1 + w_2 x_2 + w_0$
- $h = \sigma(z)$



A graphical representation: one hidden neuron

- A neuron does two things: weighted sum of input and non-linear activation

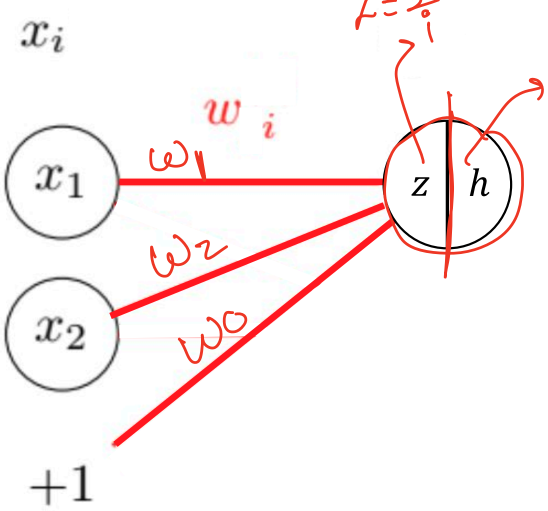
preactivation

z

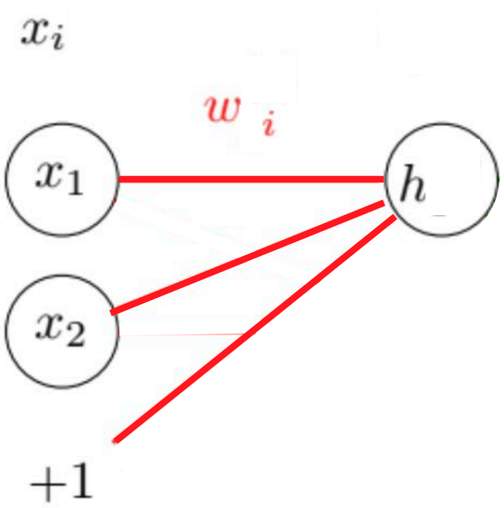
h

$z = \sum_i w_i x_i$

$h = \sigma(z)$



A graphical representation: one hidden neuron



Learning 2 hidden neurons – new notation

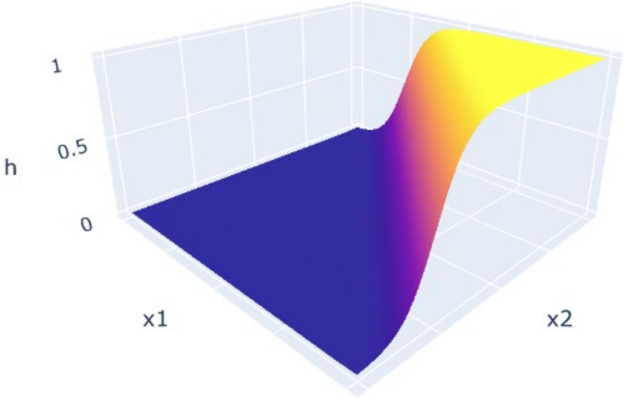
- $\bar{x} = [x_1\ x_2]^\top$

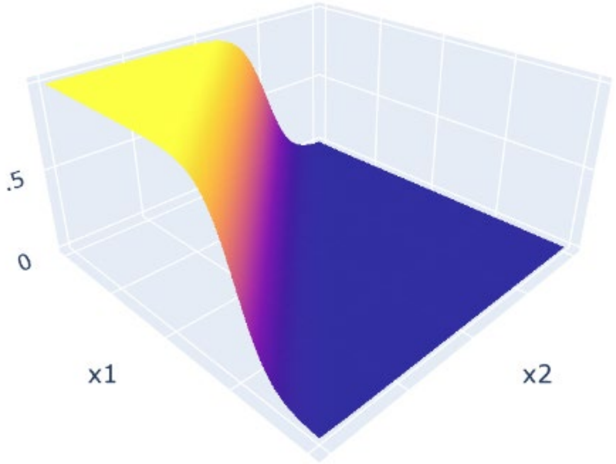
- $z_1 = w_{11}\ x_1 + \boxed{w_{12}}\ x_2 + w_{10}$
- $h_1 = \sigma(z_1)$

first hidden neuron.

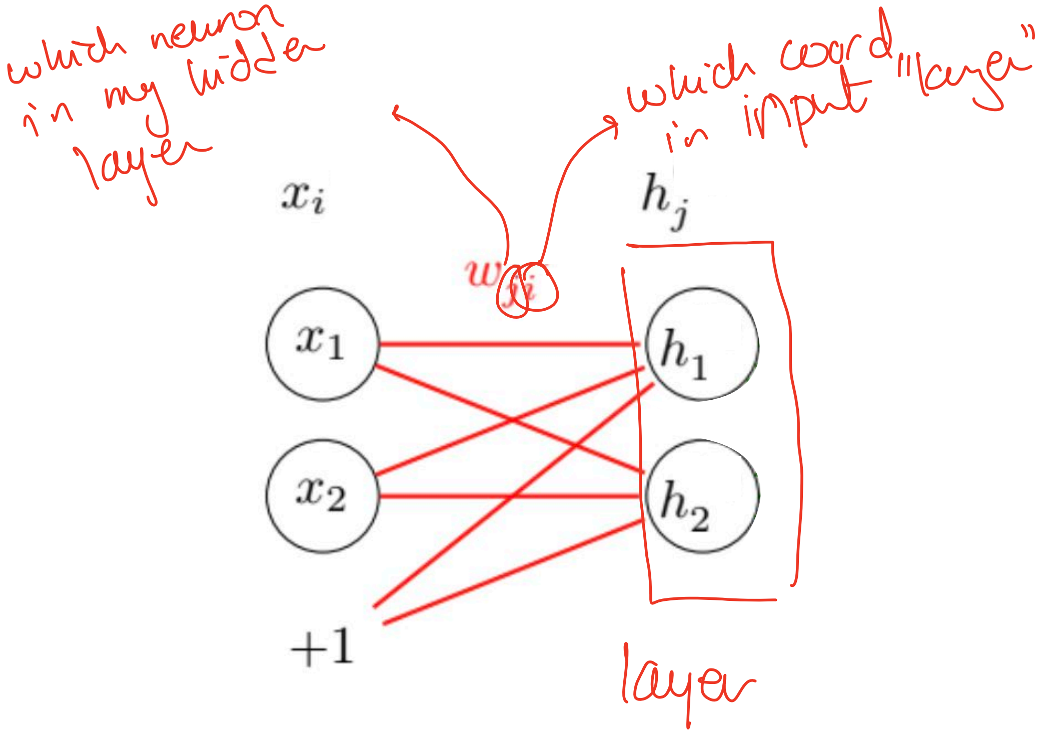
second dim in input

- $z_2 = w_{21} x_1 + w_{22}\ x_2 + w_{20}$
- $h_2 = \sigma(z_2)$





Updating the graphical representation



Prediction with 2 hidden neurons

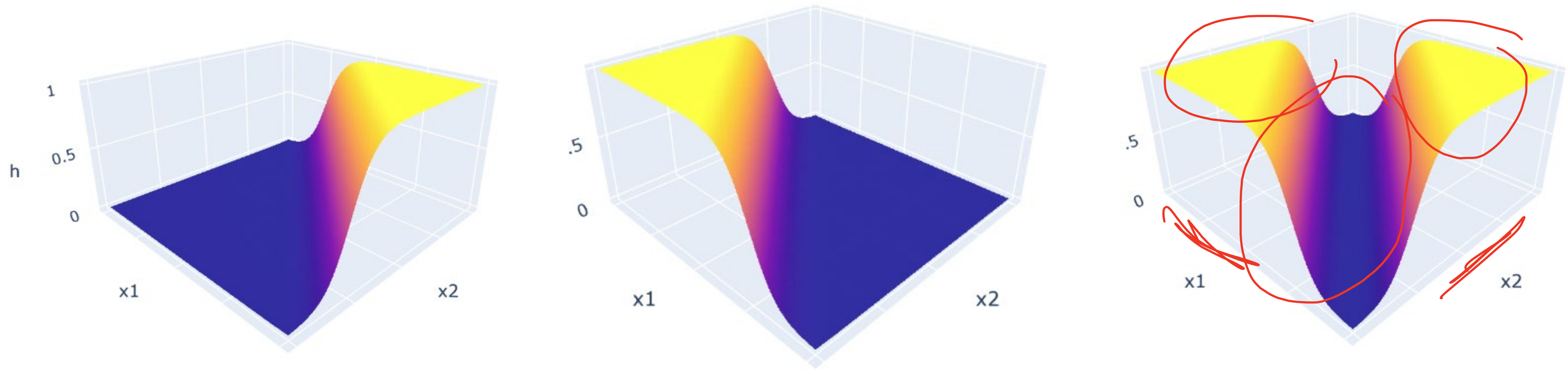
- $\bar{x} = [x_1\ x_2]^\top$

Final prediction:

$$\hat{y} = h_1 + h_2$$

- $z_1 = w_{11}\ x_1 + w_{12}\ x_2 + w_{10}$
- $h_1 = \sigma(z_1)$

- $z_2 = w_{21} x_1 + w_{22}\ x_2 + w_{20}$
- $h_2 = \sigma(z_2)$



Prediction with 2 hidden neurons

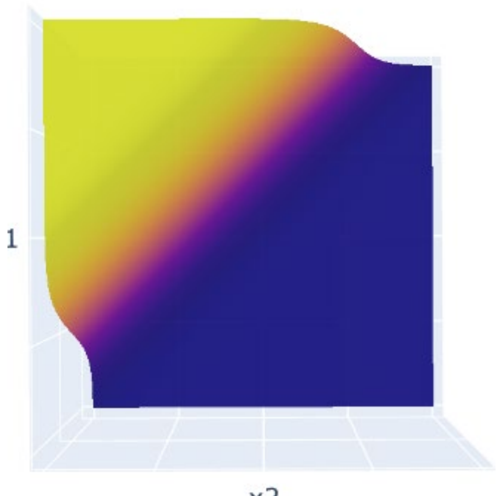
- $\bar{x} = [x_1\ x_2]^\top$

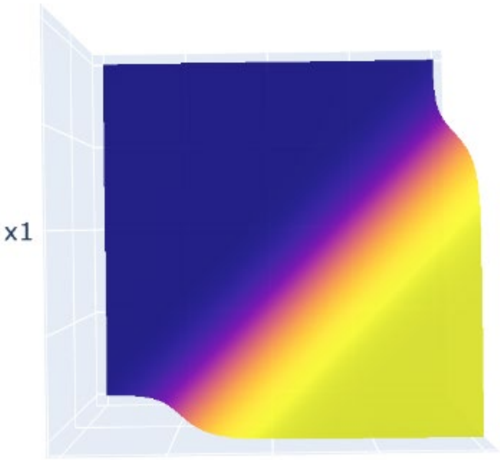
- $z_1 = w_{11}\, x_1 + w_{12}\, x_2 + w_{10}$
- $h_1 = \sigma(z_1)$

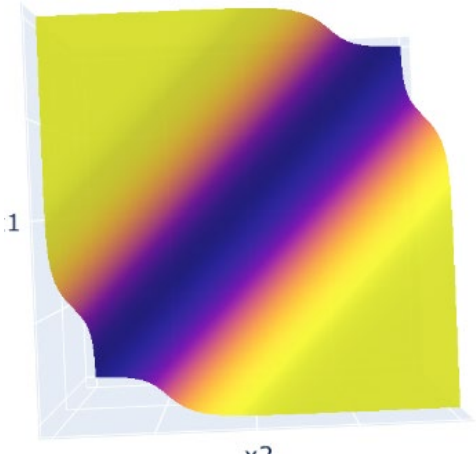
- $z_2 = w_{21} x_1 + w_{22}\, x_2 + w_{20}$
- $h_2 = \sigma(z_2)$

Final prediction:

$$\hat{y} = h_1 + h_2$$







Prediction with 2 hidden neurons – different weights

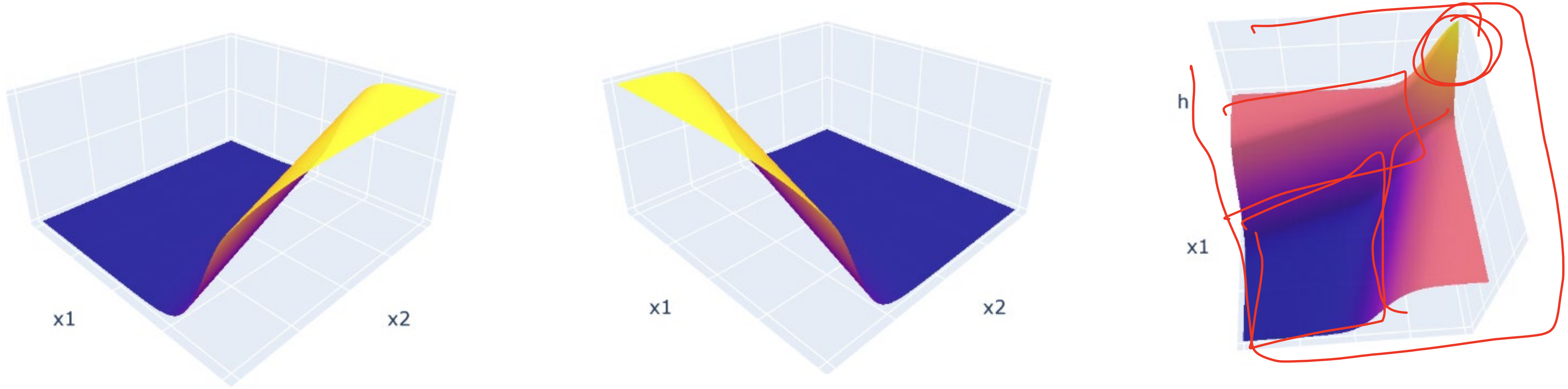
- $\bar{x} = [x_1\ x_2]^\top$

- $z_1 = w_{11}\ x_1 + w_{12}\ x_2 + w_{10}$
- $h_1 = \sigma(z_1)$

- $z_2 = w_{21} x_1 + w_{22}\, x_2 + w_{20}$
- $h_2 = \sigma(z_2)$

Final prediction:

$$\hat{y} = h_1 + h_2$$



Prediction with 2 hidden neurons – different weights

- $\bar{x} = [x_1\ x_2]^\top$

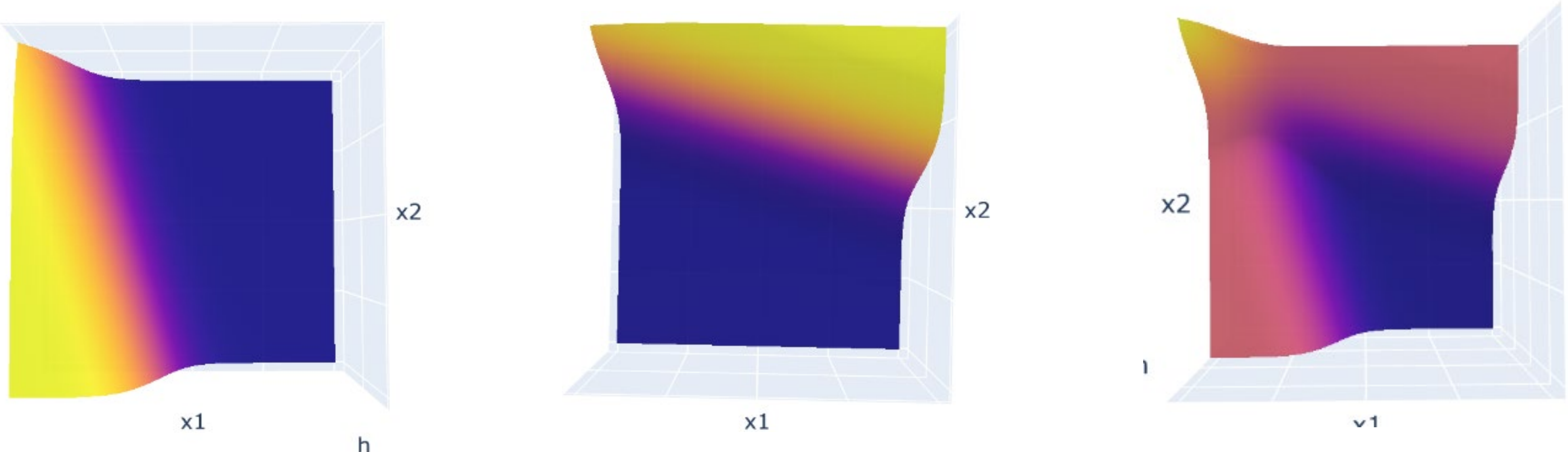
- $z_1 = w_{11}\ x_1 + w_{12}\ x_2 + w_{10}$
- $h_1 = \sigma(z_1)$

- $z_2 = w_{21}x_1 + w_{22}\,x_2 + w_{20}$
- $h_2 = \sigma(z_2)$

Final prediction:

$$\hat{y} = h_1 + h_2$$

why sum?



Prediction with 2 hidden neurons – weights for prediction

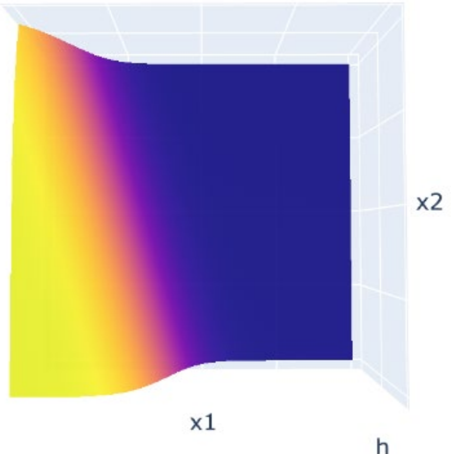
- $\bar{x} = [x_1\ x_2]^\top$

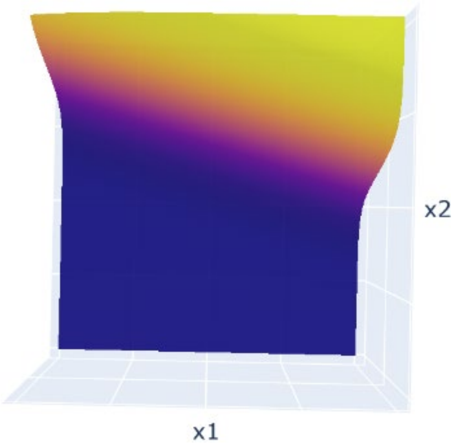
- $z_1 = w_{11}\ x_1 + w_{12}\ x_2 + w_{10}$
- $h_1 = \sigma(z_1)$

- $z_2 = w_{21} x_1 + w_{22}\ x_2 + w_{20}$
- $h_2 = \sigma(z_2)$

Final prediction:

$$\hat{y} = w_1 h_1 + w_2 h_2 + w_0$$



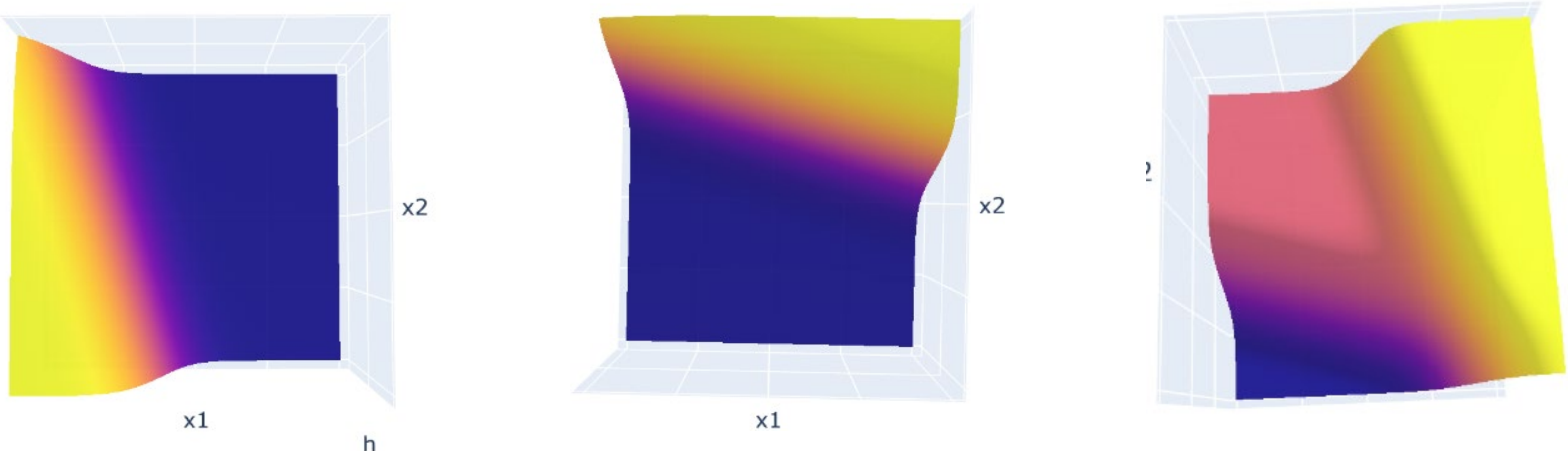


Prediction with 2 hidden neurons – weights for prediction

- $\bar{x} = [x_1 \; x_2]^\top$
- $z_1 = w_{11} \; x_1 + w_{12} \; x_2 + w_{10}$
- $h_1 = \sigma(z_1)$
- $z_2 = w_{21} x_1 + w_{22} \, x_2 + w_{20}$
- $h_2 = \sigma(z_2)$

Final prediction:

$\hat{y} = \sigma(w_1 h_1 + w_2 h_2 + \mathrm{w}_0)$



..More layers!

$\bar{x} = [x_1\ x_2]^\top$

$z_1^{(2)} = w_{11}^{(1)}\ x_1 + w_{12}^{(1)} x_2 + w_{10}^{(1)}$

$h_1^{(2)} = \sigma(z_1^{(2)})$

$z_2^{(2)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{20}^{(1)}$

$h_2^{(2)} = \sigma(z_2^{(2)})$

$\hat{y} = \sigma(w_1 h_1 + w_2 h_2 + \mathrm{w}_0)$

$\bar{h} = [h_1 \quad h_2]^\top$

$z_1^{(3)} = w_{11}^{(2)} h_1^{(2)} + w_{12}^{(2)} h_2^{(2)} + \mathrm{w}_{10}^{(2)}$

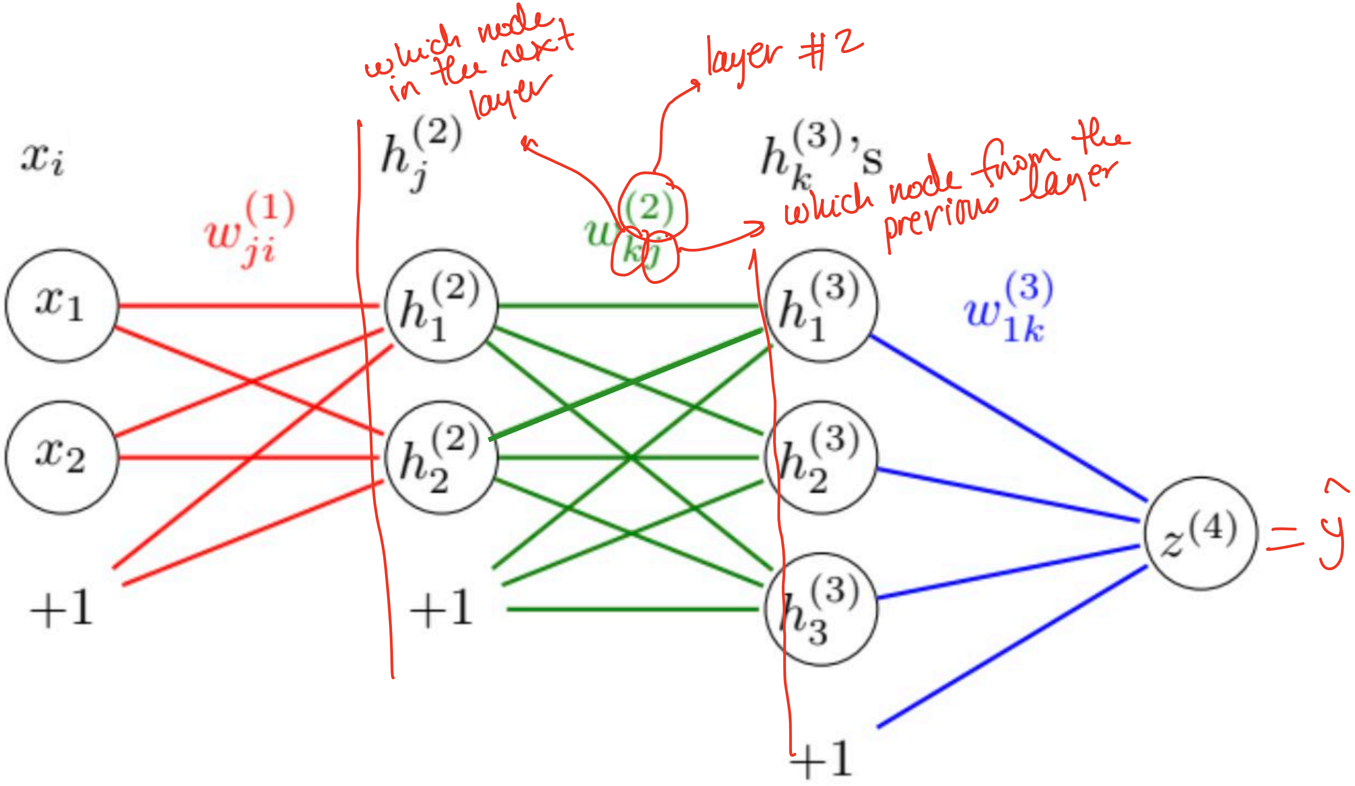
$h_1^{(3)} = \sigma(\mathrm{z}_1^{(3)})$

$z_2^{(3)} = w_{21}^{(2)} h_1^{(2)} + w_{22}^{(2)} h_2^{(2)} + w_{20}^{(2)}$

$h_2^{(3)} = \sigma(\mathrm{z}_2^{(3)})$

$\hat{y} = z^{(4)} = \sigma(w_{11}^{(3)} h_1^{(3)} + w_{12}^{(3)} h_2^{(3)} + \mathrm{w}_{10}^{(3)})$

Updating the graphical representation



Reading the notation

which layer # 2

layer # 1

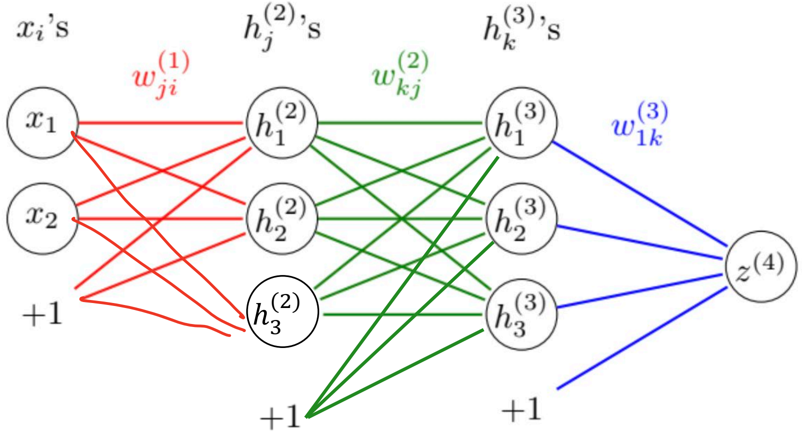
$$z_1^{(2)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{10}^{(1)}$$

first hidden neuron

get multiplied first coordinate

first hidden neuron in the next layer

Matrix notation



$$z_1^{(2)} = w_{11}^{(1)}\, x_1 + w_{12}^{(1)} x_2 + w_{10}^{(1)}$$

$$z_2^{(2)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{20}^{(1)}$$

$$z_3^{(2)} = w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{30}^{(1)}$$

$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} w_{10}^{(1)} \\ w_{20}^{(1)} \\ w_{30}^{(1)} \end{bmatrix}$$

$$\bar{z}^{(2)} = W^{(1)}\, \bar{x} + \bar{b}^{(1)}$$

$$\bar{h}^{(2)} = g(\bar{z}^{(2)})$$

$$g = \sigma$$

With this matrix notation, we can write:

$\bar{h}^{(1)} = \bar{x}$

$\bar{z}^{(2)} = W^{(1)}\, \bar{h}^{(1)} + \bar{b}^{(1)}$

$\bar{h}^{(2)} = g\left(\bar{z}^{(2)}\right)$

$\bar{z}^{(3)} = W^{(2)}\, \bar{h}^{(2)} + \bar{b}^{(2)}$

...

$\bar{z}^{(j+1)} = W^{(j)}\, \bar{h}^{(j)} + \bar{b}^{(j)}$

$\bar{h}^{(j+1)} = g\left(\bar{z}^{(j+1)}\right)$

...

$$g(\bar{z}^{(2)}) = \begin{bmatrix} g(z_1^{(2)}) \\ g(z_2^{(2)}) \\ g(z_3^{(2)}) \end{bmatrix}$$

TL;DPA:

We introduced notation for describing the learned features. Superscripts refer to the layer number and subscripts refer to nodes neurons within layers (generally speaking)

We introduced a graphical representation of NNs, where each node is either an input or a neuron. Edges denote learnable weights