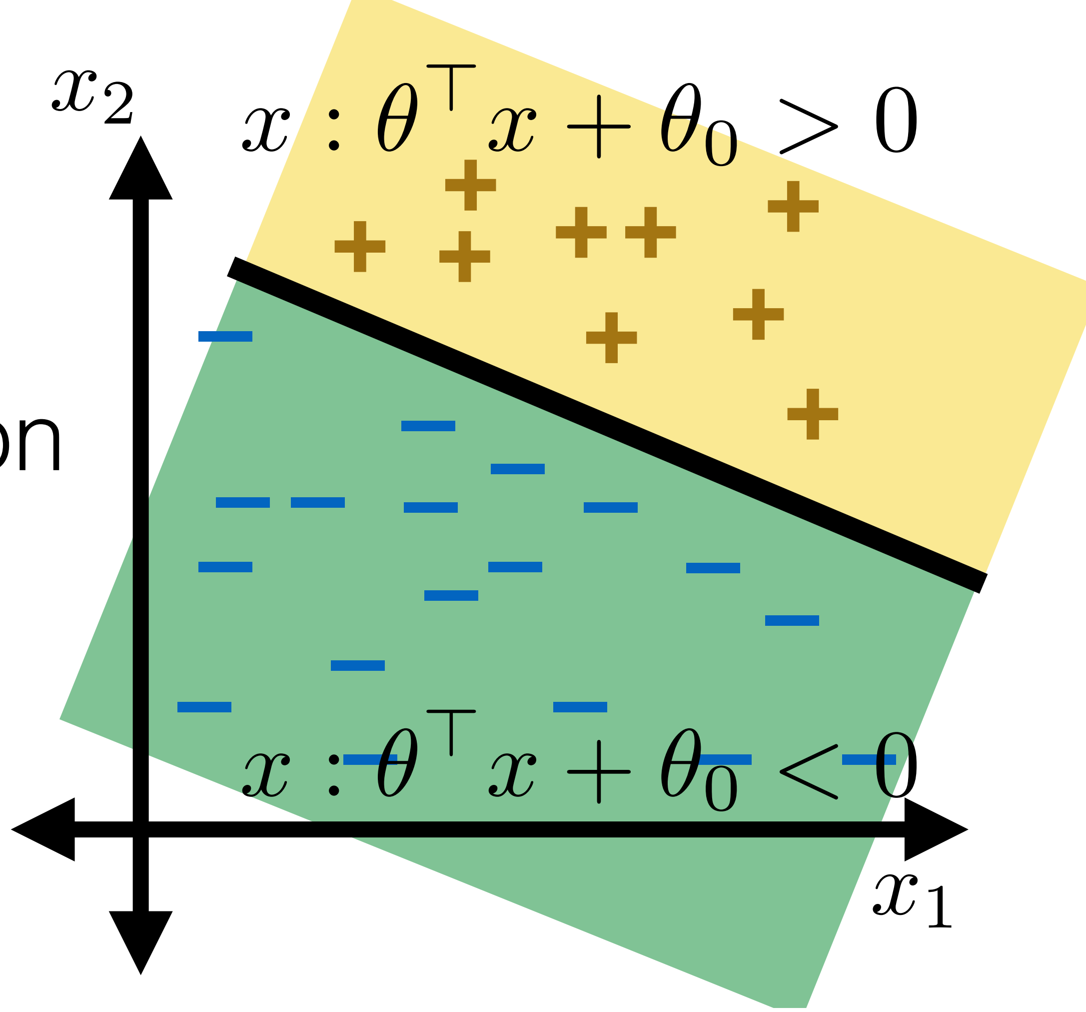
Neural Networks

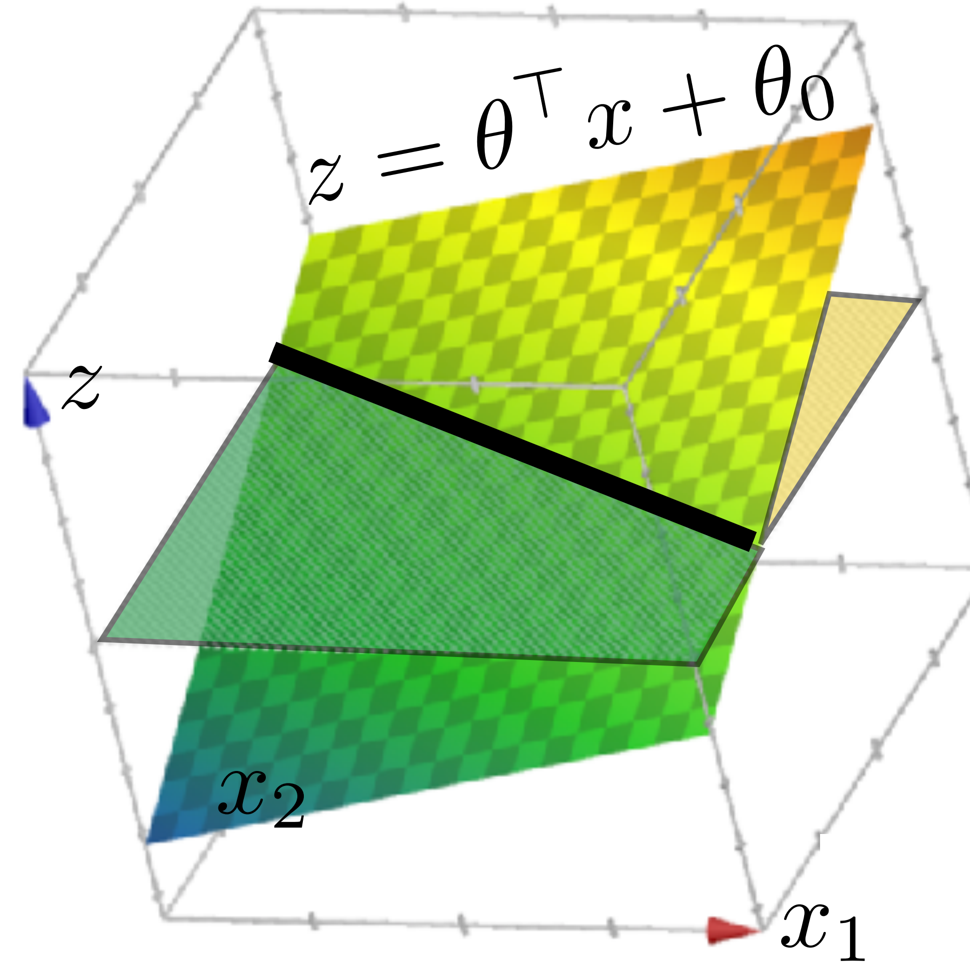
Prof. Tamara Broderick

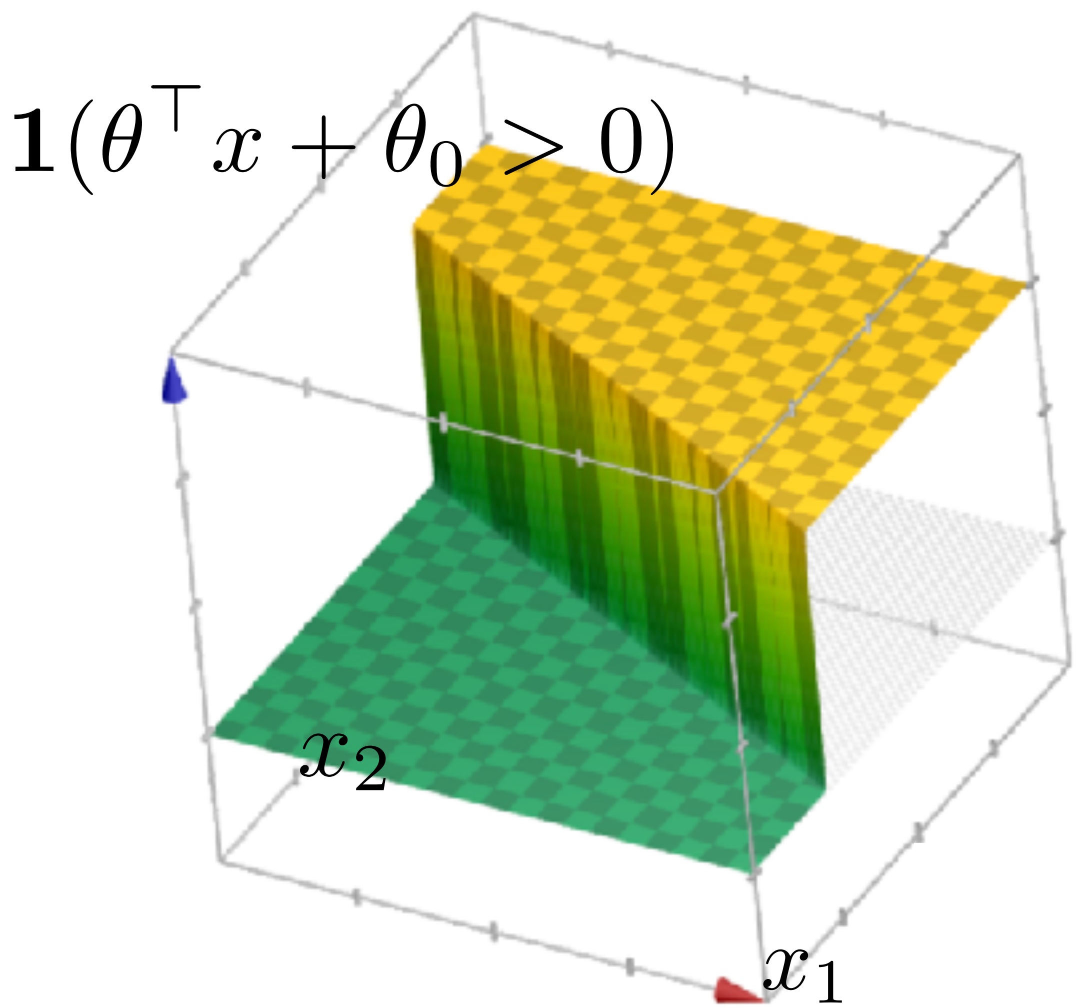
Edited From 6.036 Fall21 Offering

Recall

- Linear classification with default features:





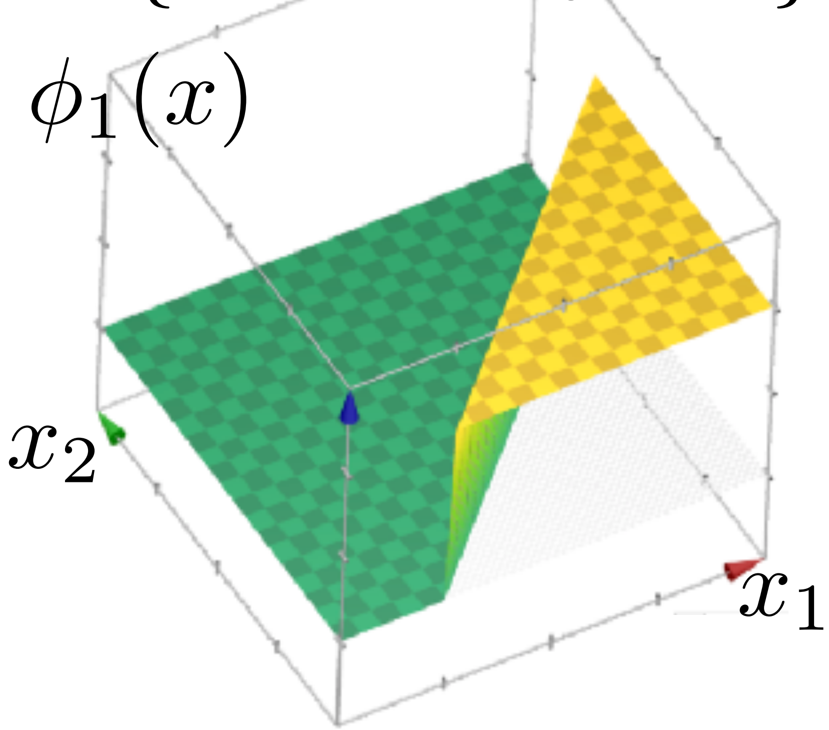


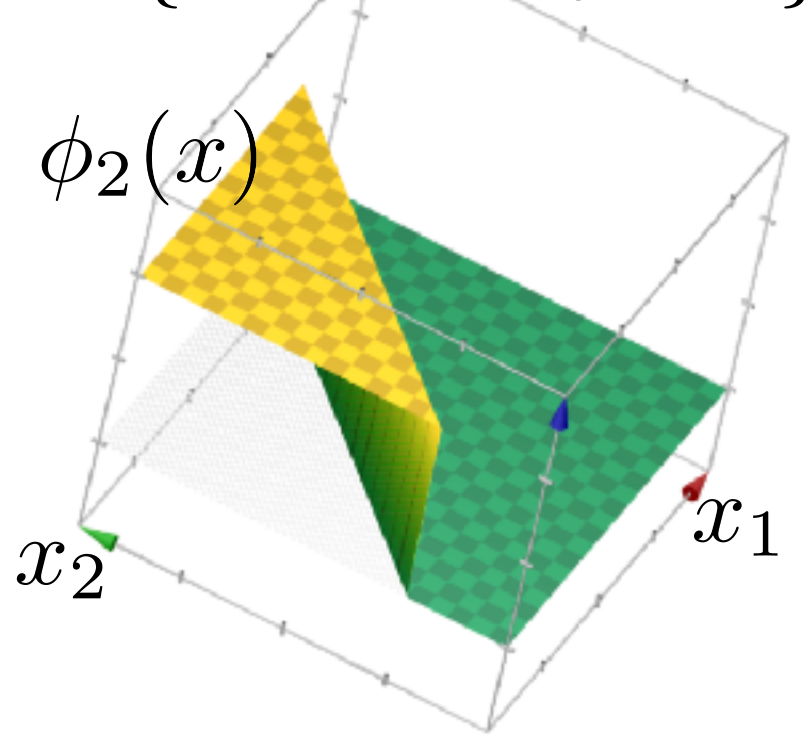
- We're used to using step functions to classify
- New idea today: we'll use step functions as *features*, with their own parameters

New features: step functions!

$\phi_1(x) = \mathbf{1}\{w^\top x + w_0 \geq 0\}$

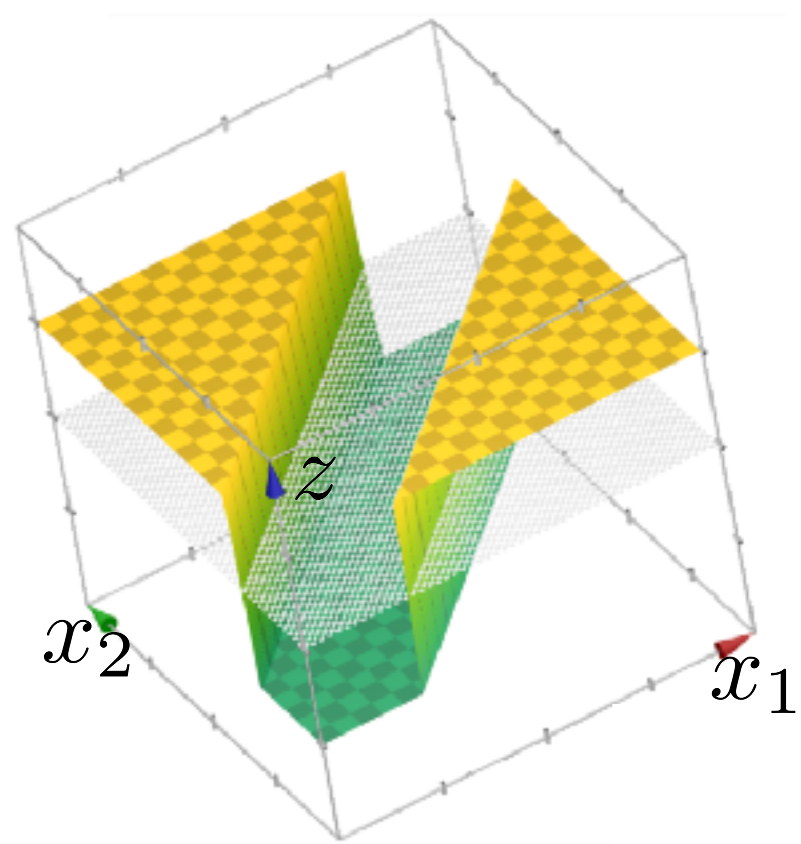
$\phi_2(x) = \mathbf{1}\{\tilde{w}^\top x + \tilde{w}_0 \geq 0\}$

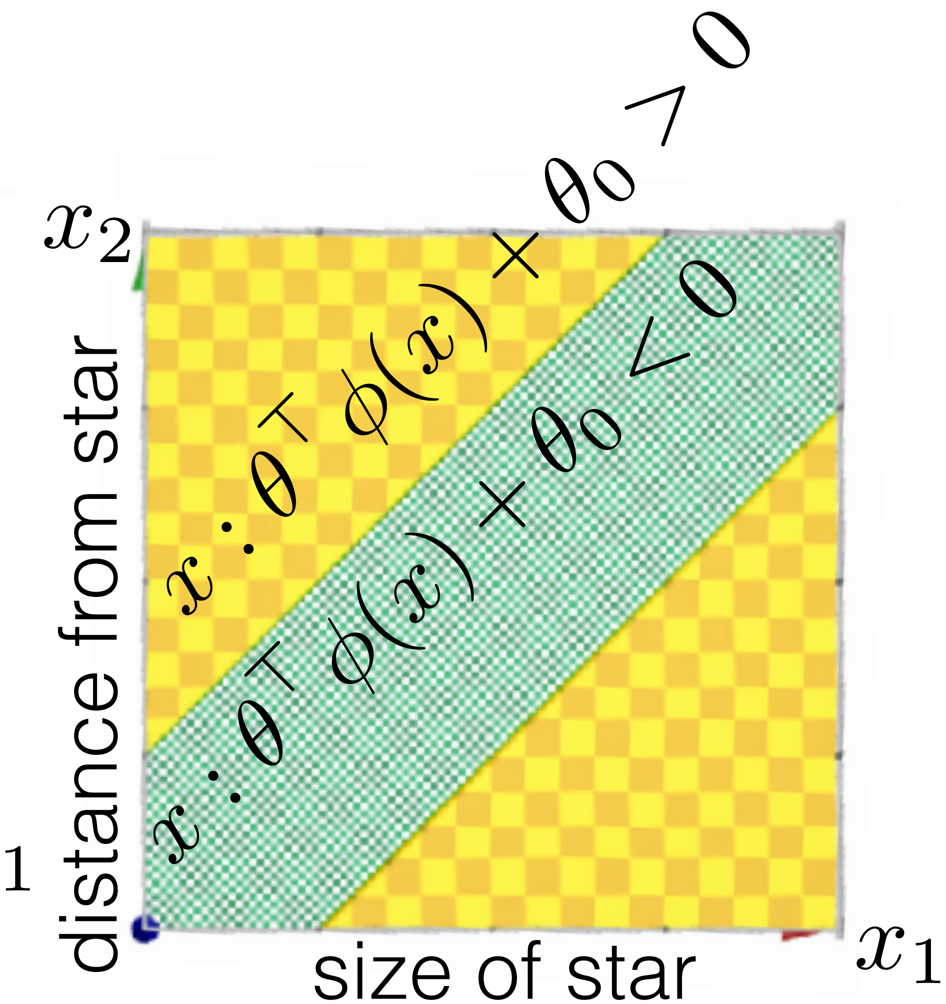




Is an exoplanet outside the habitable zone?

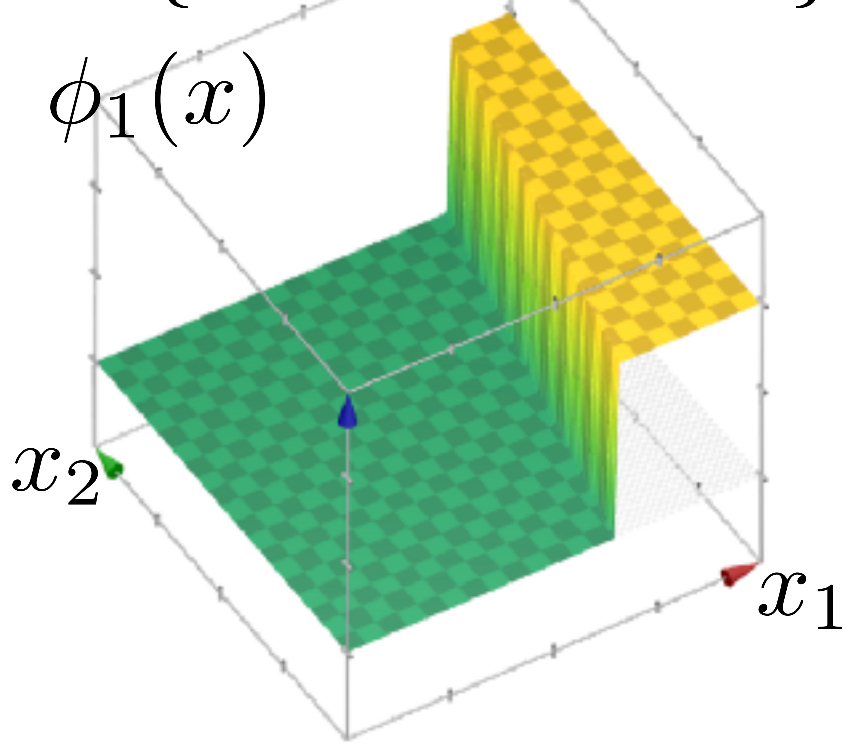
$$z = \theta^\top \phi(x) + \theta_0$$
$$= \theta_1\phi_1(x) + \theta_2\phi_2(x) + \theta_0$$
$$= 1 \cdot \phi_1(x) + 1 \cdot \phi_2(x) + (-0.5)$$



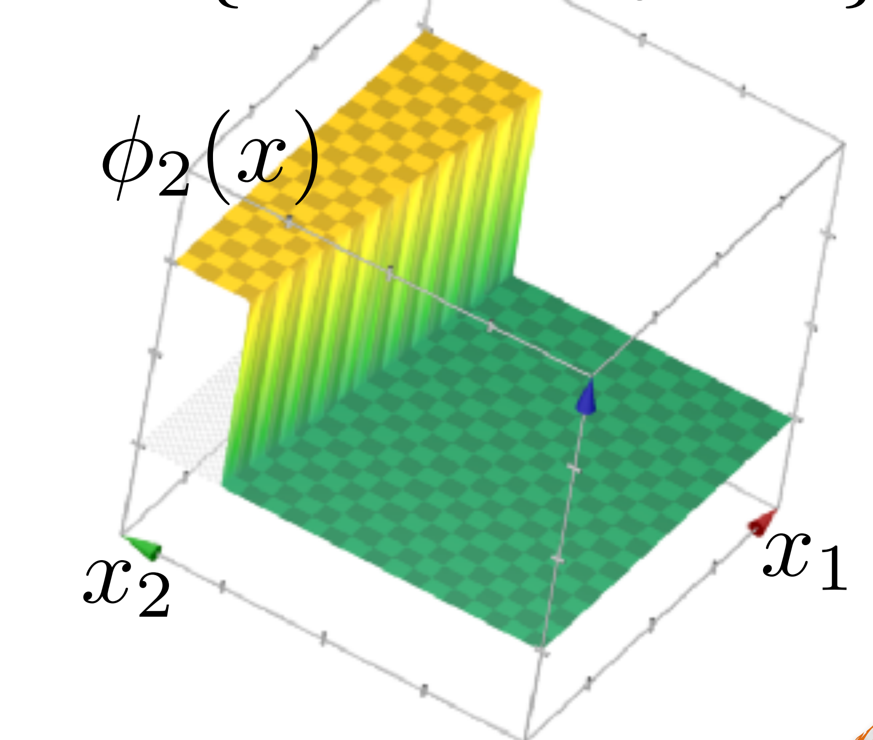


New features: step functions!

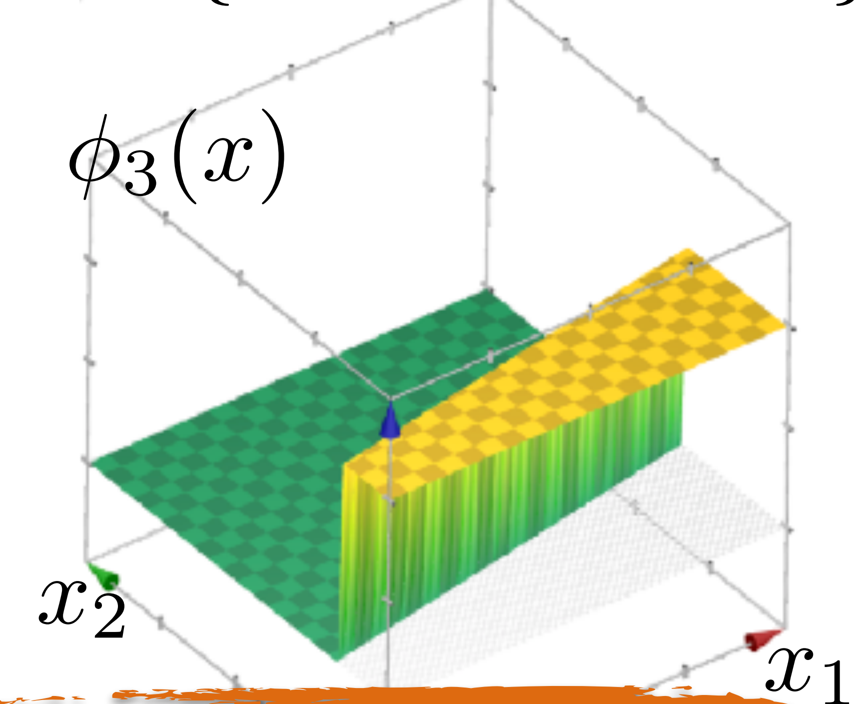
$\phi_1(x) = \mathbf{1}\{w^\top x + w_0 \geq 0\}$



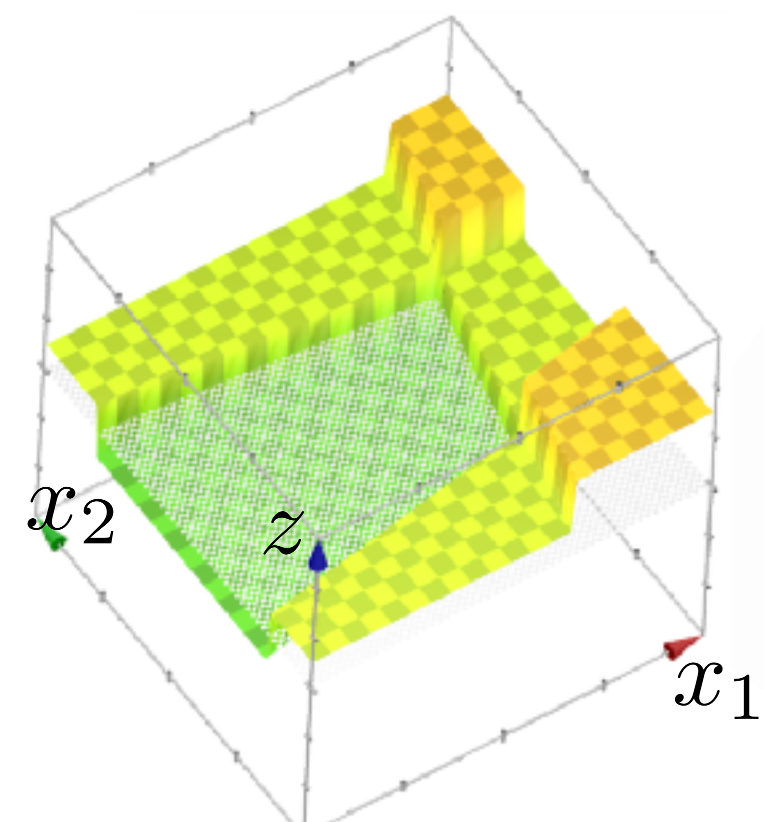
$\phi_2(x) = \mathbf{1}\{\tilde{w}^\top x + \tilde{w}_0 \geq 0\}$

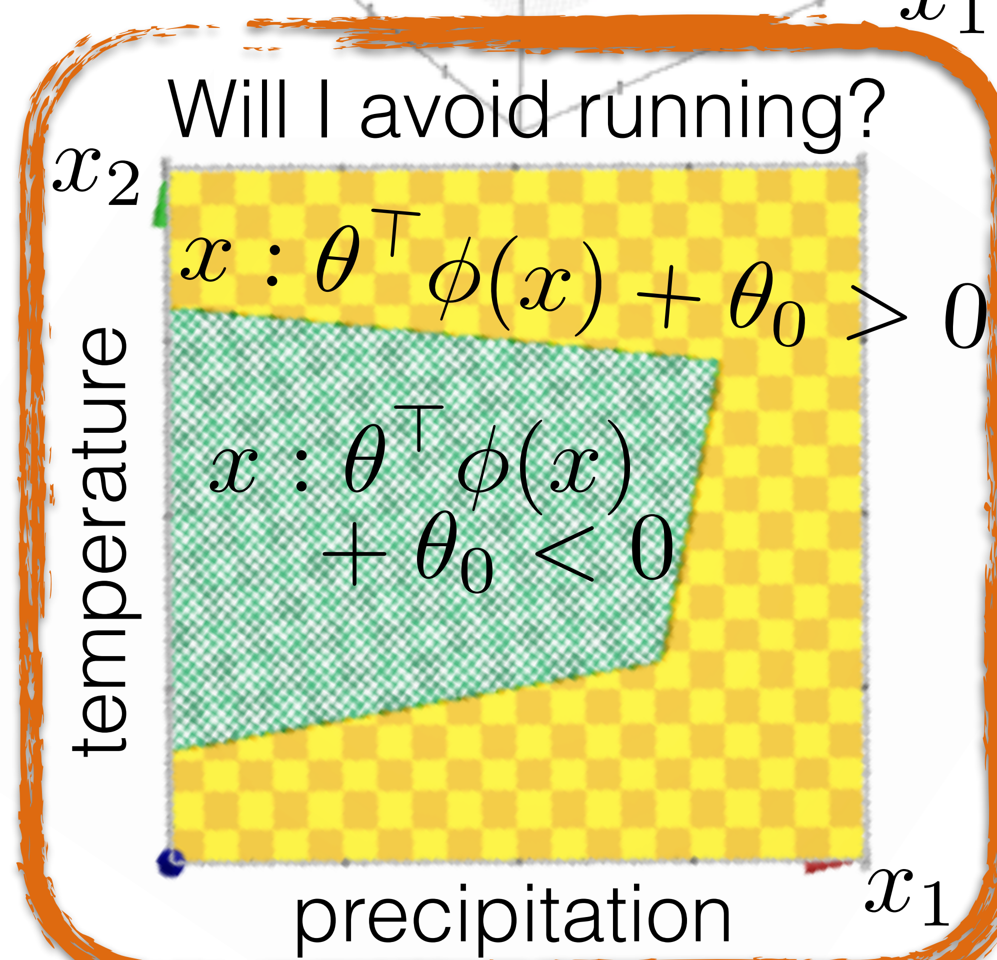


$\phi_3(x) = \mathbf{1}\{\tilde{\tilde{w}}^\top x + \tilde{\tilde{w}}_0 \geq 0\}$



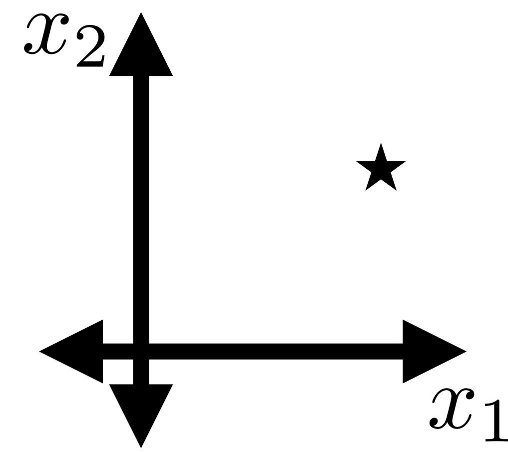
$$
\begin{aligned}
z &= \theta^\top \phi(x) + \theta_0 \\
&= \theta_1 \phi_1(x) + \theta_2 \phi_2(x) \\
&\quad + \theta_3 \phi_3(x) + \theta_0 \\
&= 1 \cdot \phi_1(x) + 1 \cdot \phi_2(x) \\
&\quad + 1 \cdot \phi_3(x) + (-0.5)
\end{aligned}
$$

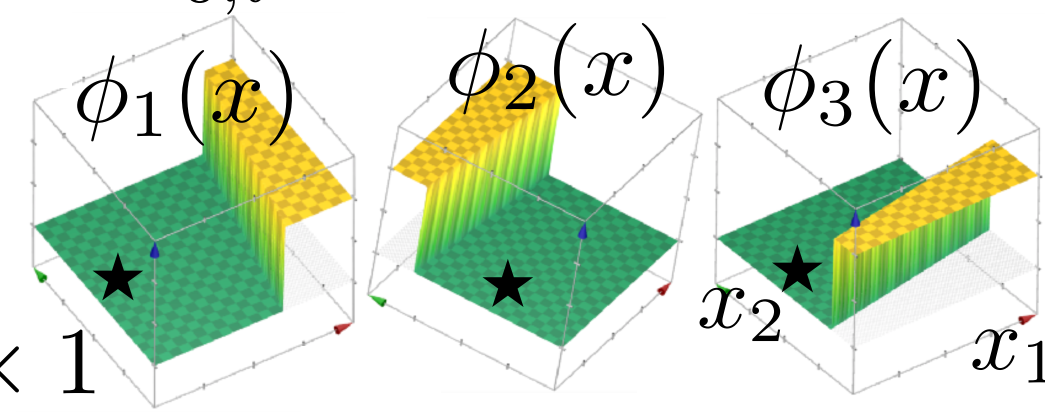




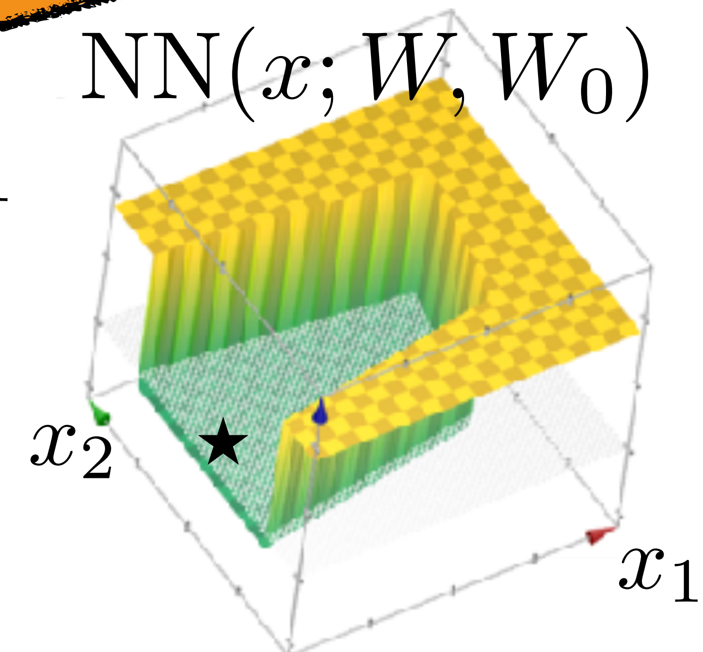
Let's get some new notation

- 1st layer, constructing the features:
  - Input x (a data point): size $m^{(1)} \times 1$ $\boxed{m^{(1)} = d}$
  - Output $A^{(1)}$ (vector of features): size $n^{(1)} \times 1$
  - The ith feature: $A_i^{(1)} = f^{(1)}(w_i^{(1)\top} x + w_{0,i}^{(1)})$
    - All the features at once:
      - $A^{(1)} = f^{(1)}(W^{(1)\top} x + W_0^{(1)})$
      - $W^{(1)} : m^{(1)} \times n^{(1)}; W_0^{(1)} : n^{(1)} \times 1$



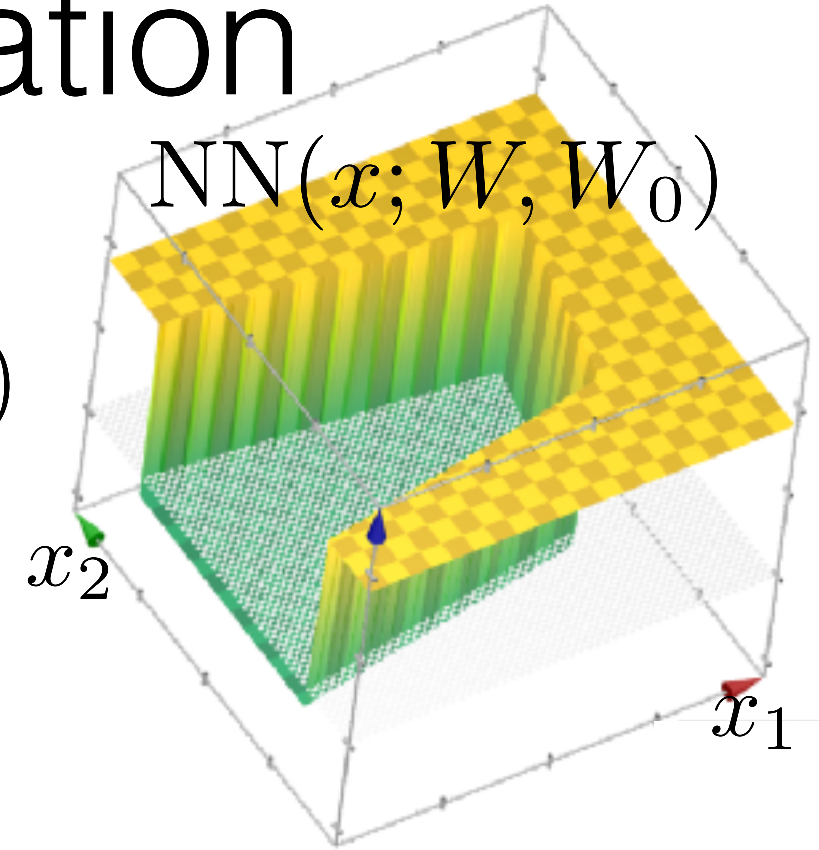


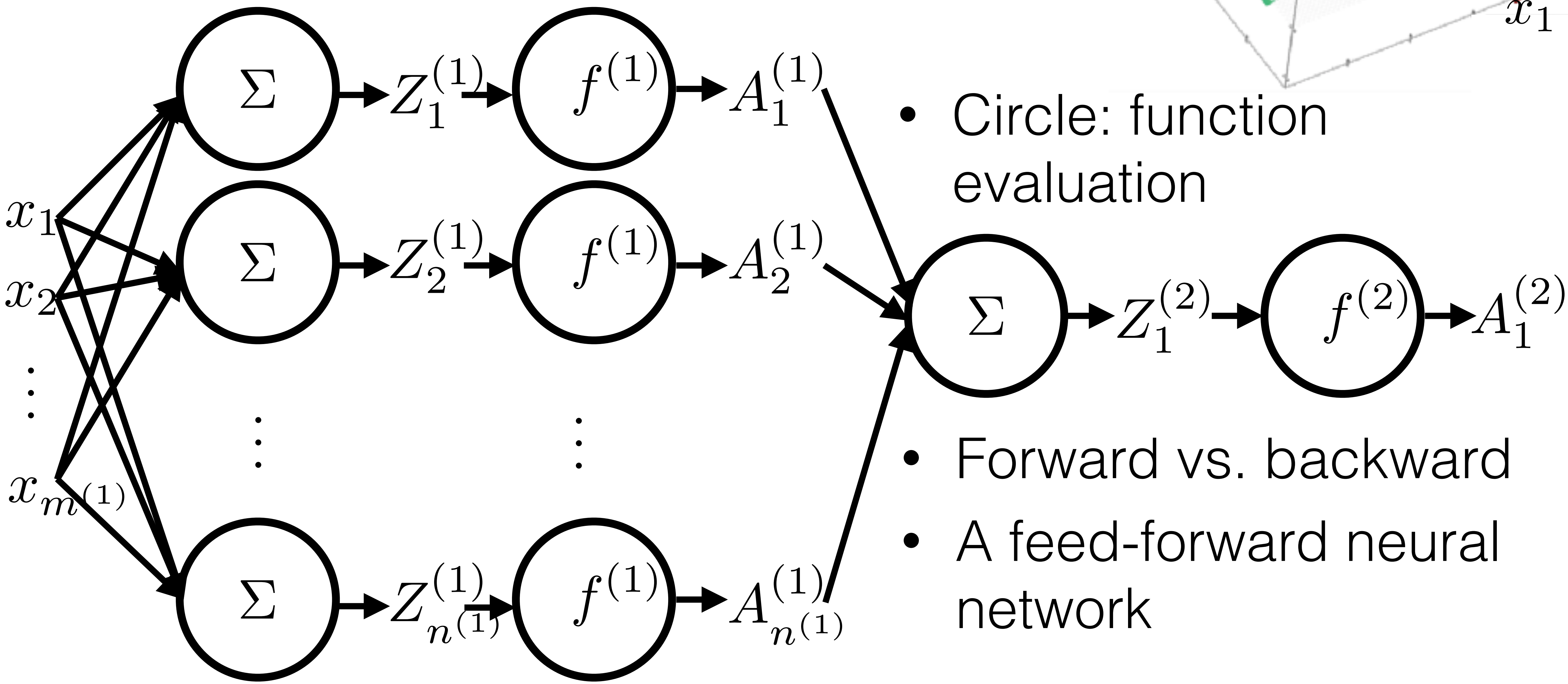
- 2nd layer, assigning a label (or labels):
  - Input (the features): size $m^{(2)} \times 1$ $\boxed{m^{(2)} = n^{(1)}}$
  - Output $A^{(2)}$(vector of labels): size $n^{(2)} \times 1$
  - The ith label: $A_i^{(2)} = f^{(2)}(w_i^{(2)\top} A^{(1)} + w_{0,i}^{(2)})$
    - All: $A^{(2)} = f^{(2)}(W^{(2)\top} A^{(1)} + W_0^{(2)})$ $\boxed{\text{Like } y}$
    - $W^{(2)} : m^{(2)} \times n^{(2)}; W_0^{(2)} : n^{(2)} \times 1$



Function graph representation

- 1st layer: $A^{(1)} = f^{(1)}(W^{(1)\top}x + W_0^{(1)})$
- 2nd layer: $A^{(2)} = f^{(2)}(W^{(2)\top}A^{(1)} + W_0^{(2)})$
- Whole thing: $A^{(2)} = \mathrm{NN}(x; W, W_0)$

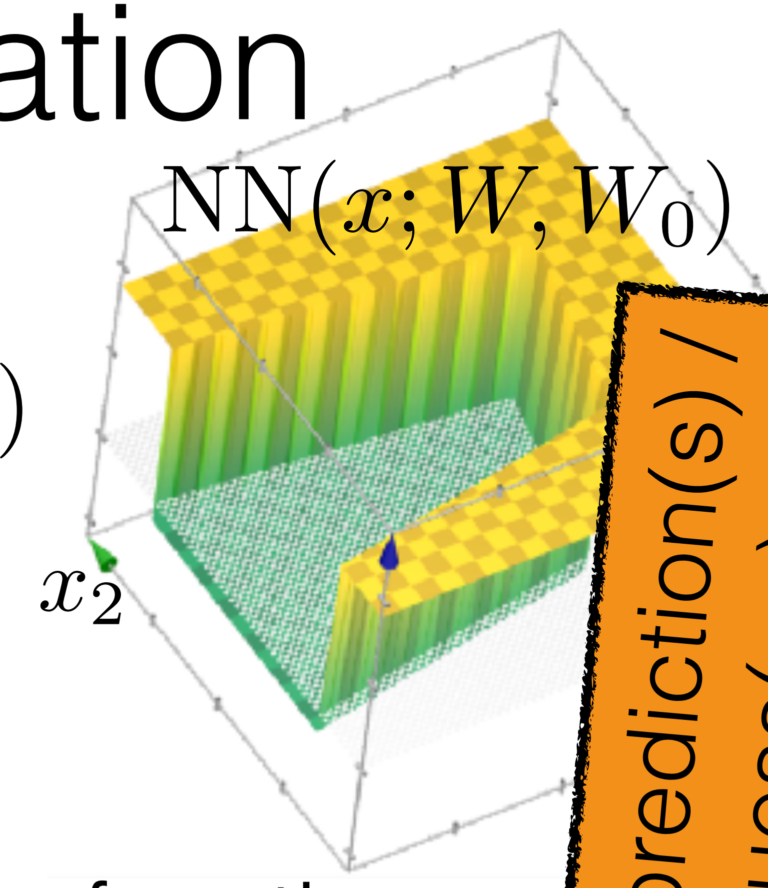


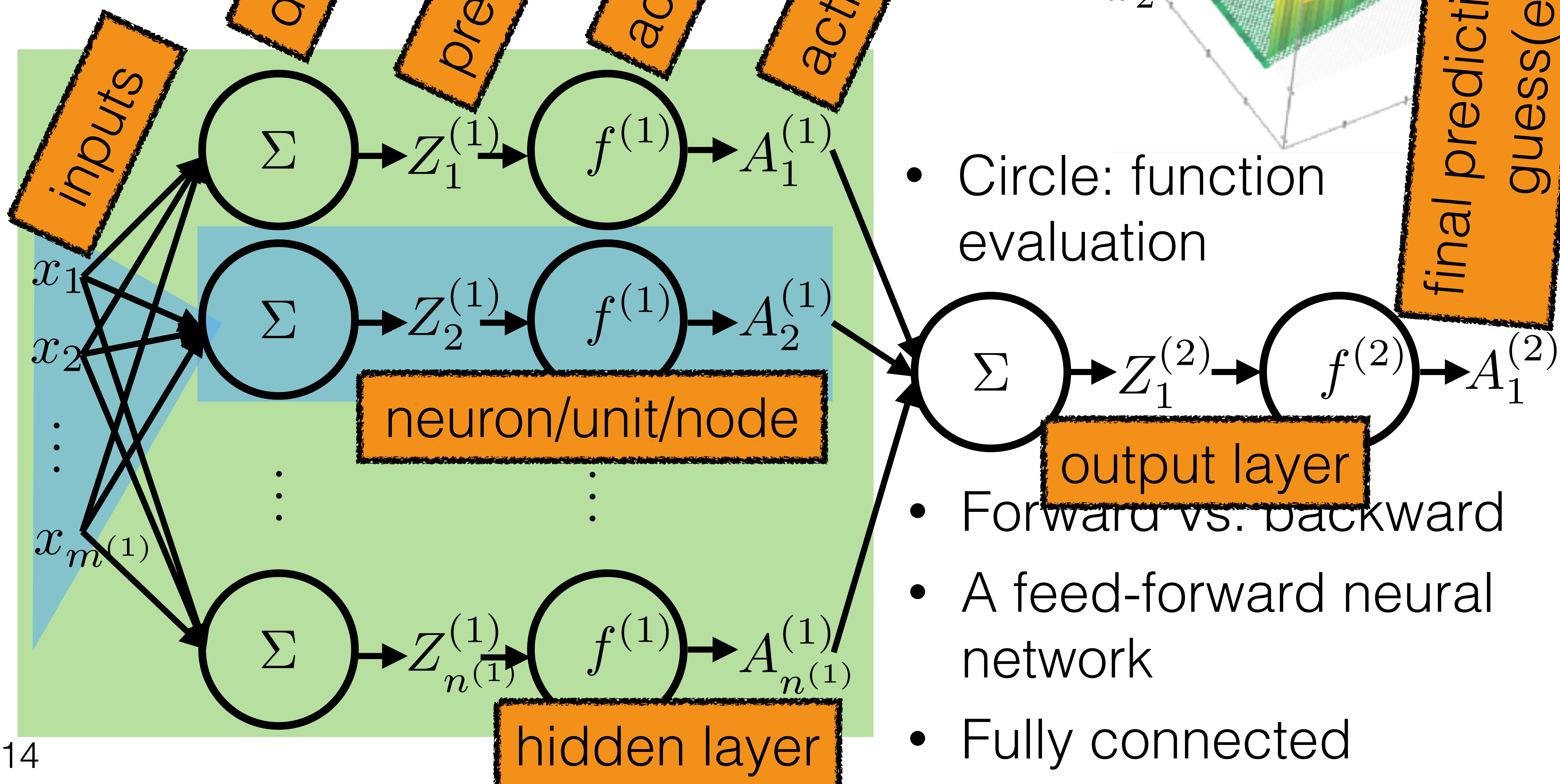


- Circle: function evaluation
- Forward vs. backward
- A feed-forward neural network

Function graph representation

- 1st layer: $A^{(1)} = f^{(1)}(W^{(1)\top}x + W_0^{(1)})$
- 2nd layer: $A^{(2)} = f^{(2)}(W^{(2)\top}A^{(1)} + W_0^{(2)})$
- Whole thing: $A^{(2)} = \mathrm{NN}(x; W, W_0)$





- Circle: function evaluation
- Forward vs. backward
- A feed-forward neural network
- Fully connected

Problem setup

- # layer ℓ inputs $m^{(\ell)}$
- # layer ℓ outputs $n^{(\ell)}$

$n^{(\ell)}$ is NOT the number of data points

1. Choose a hypothesis class. E.g.,

   $$h(x; W, W_0) = \mathrm{NN}(x; W, W_0)$$

   - 1st layer: $A^{(1)} = f^{(1)}(W^{(1)\top} x +$
   - 2nd layer: $A^{(2)} = f^{(2)}(W^{(2)\top} A^{(1)} + W_0^{(2)})$

2. Choose a loss. (E.g. for classification: 0-1 loss, asymmetric, NLL.) Set up an objective.
3. Learn the parameters. E.g. gradient descent or SGD
4. Predict on new data using these parameters

Problem setup

- # layer ℓ inputs $m^{(\ell)}$
- # layer ℓ outputs $n^{(\ell)}$

1. Choose a hypothesis class. E.g.,
   $$h(x; W, W_0) = \mathrm{NN}(x; W, W_0)$$
   - 1st layer: $A^{(1)} = f^{(1)}(W^{(1)\top} x + W_0^{(1)})$
   - 2nd layer: $A^{(2)} = f^{(2)}(W^{(2)\top} A^{(1)} + W_0^{(2)})$
2. Choose a loss. (E.g. for classification: 0-1 loss, asymmetric, NLL.) Set up an objective.
3. Learn the parameters. E.g. gradient descent or SGD
4. Predict on new data using these parameters

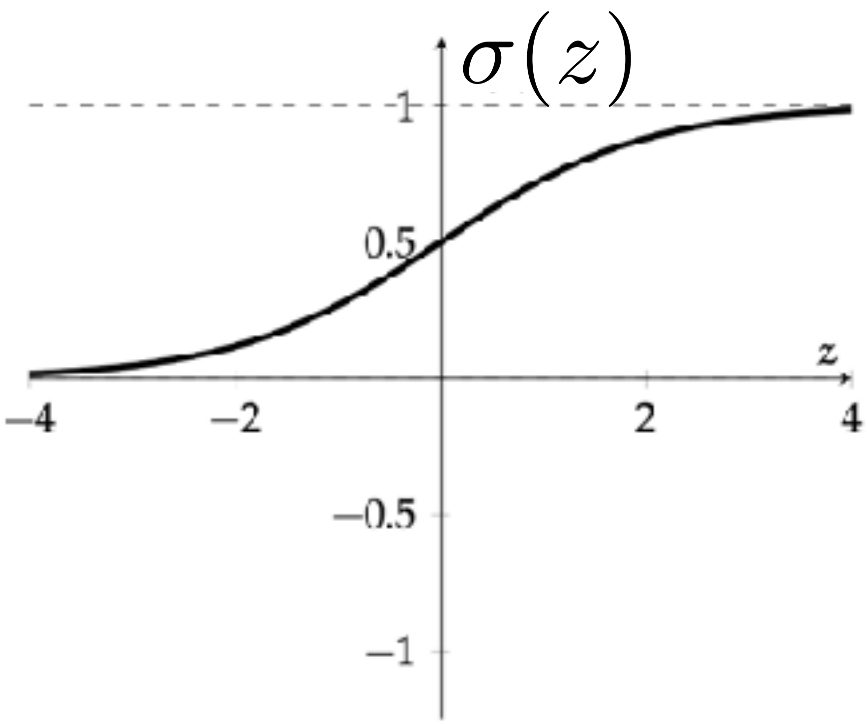
Issues:

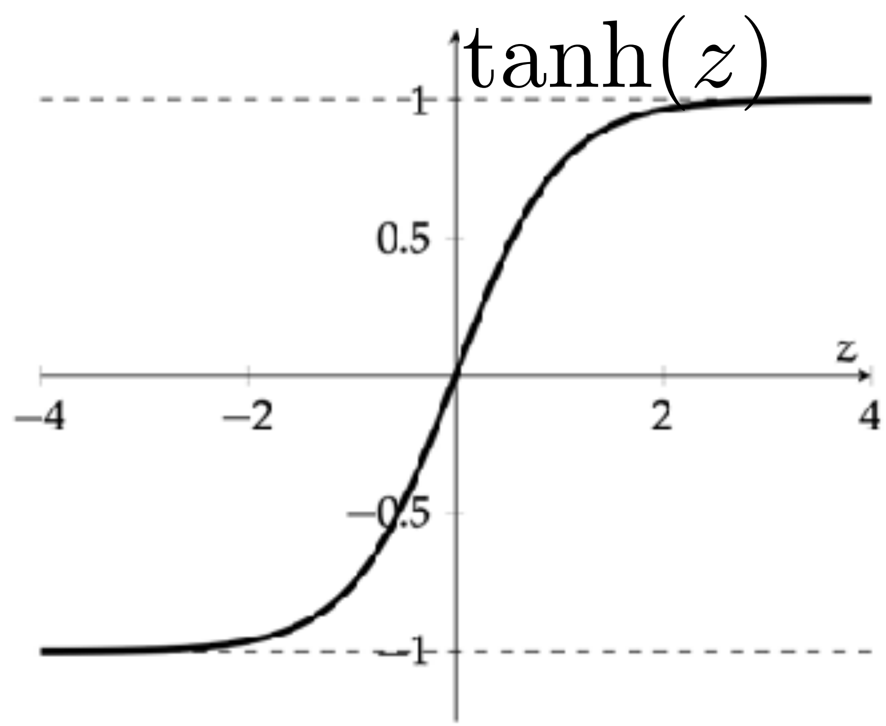
- What if I want to do regression or use NLL loss?
- Derivatives (of loss with respect to parameters) are zero (or undefined) if we use step function activation
  - So (S)GD won't do what we want
- How to compute the derivatives in (S)GD?

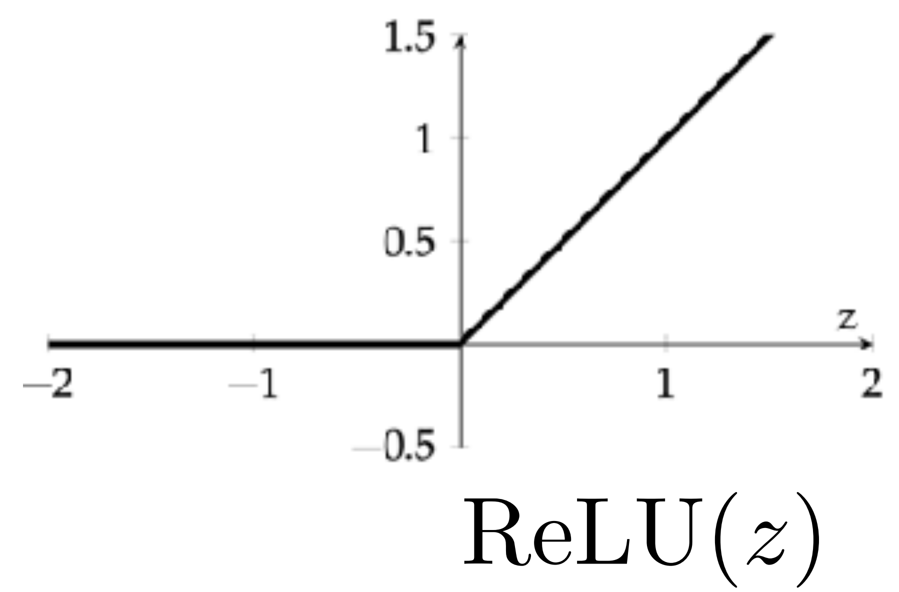
Different activation functions

1. Hypotheses $h(x; W, W_0) = \mathrm{NN}(x; W, W_0)$
    - 1st layer: $A^{(1)} = f^{(1)}(W^{(1)\top} x + W_0^{(1)})$
    - 2nd layer: $A^{(2)} = f^{(2)}(W^{(2)\top} A^{(1)} + W_0^{(2)})$

- What if I want to do regression? $f^{(2)}(z) = z$
- What if I want to use NLL loss? $f^{(2)}(z) = \sigma(z)$
- Need non-zero derivatives for (S)GD: Above & $f^{(1)}(z) =$

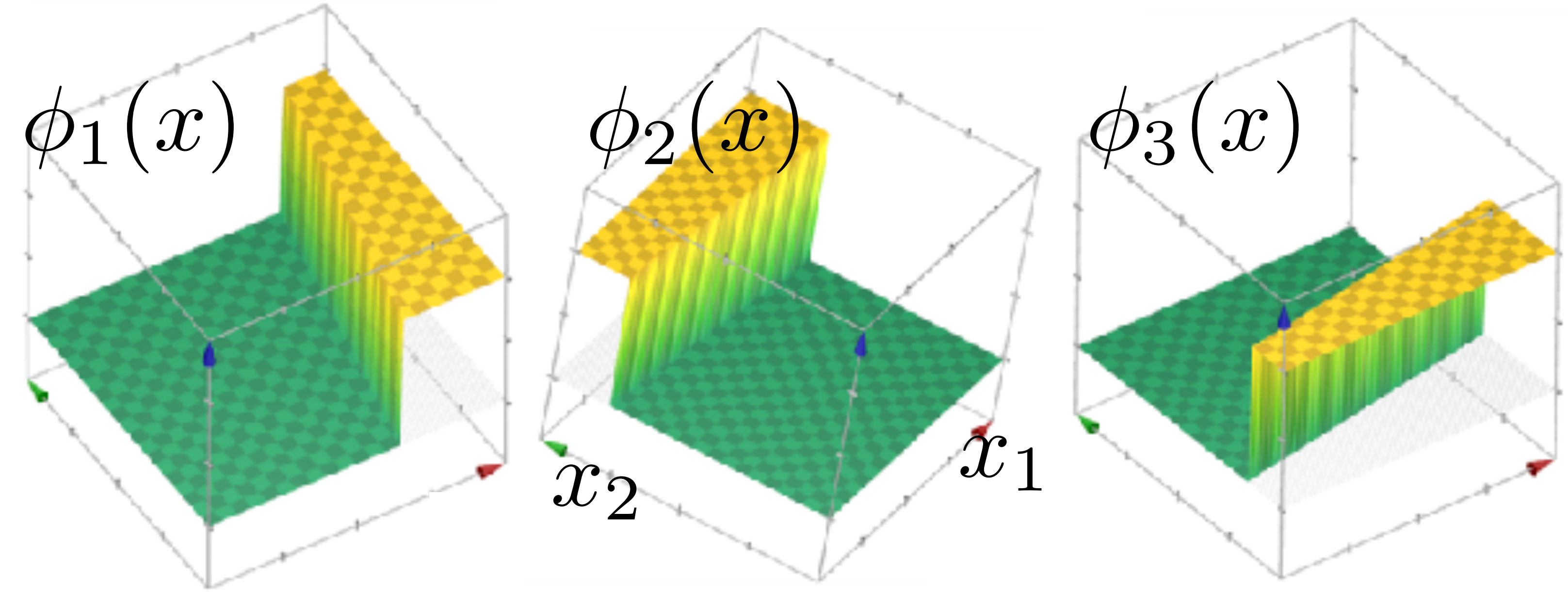




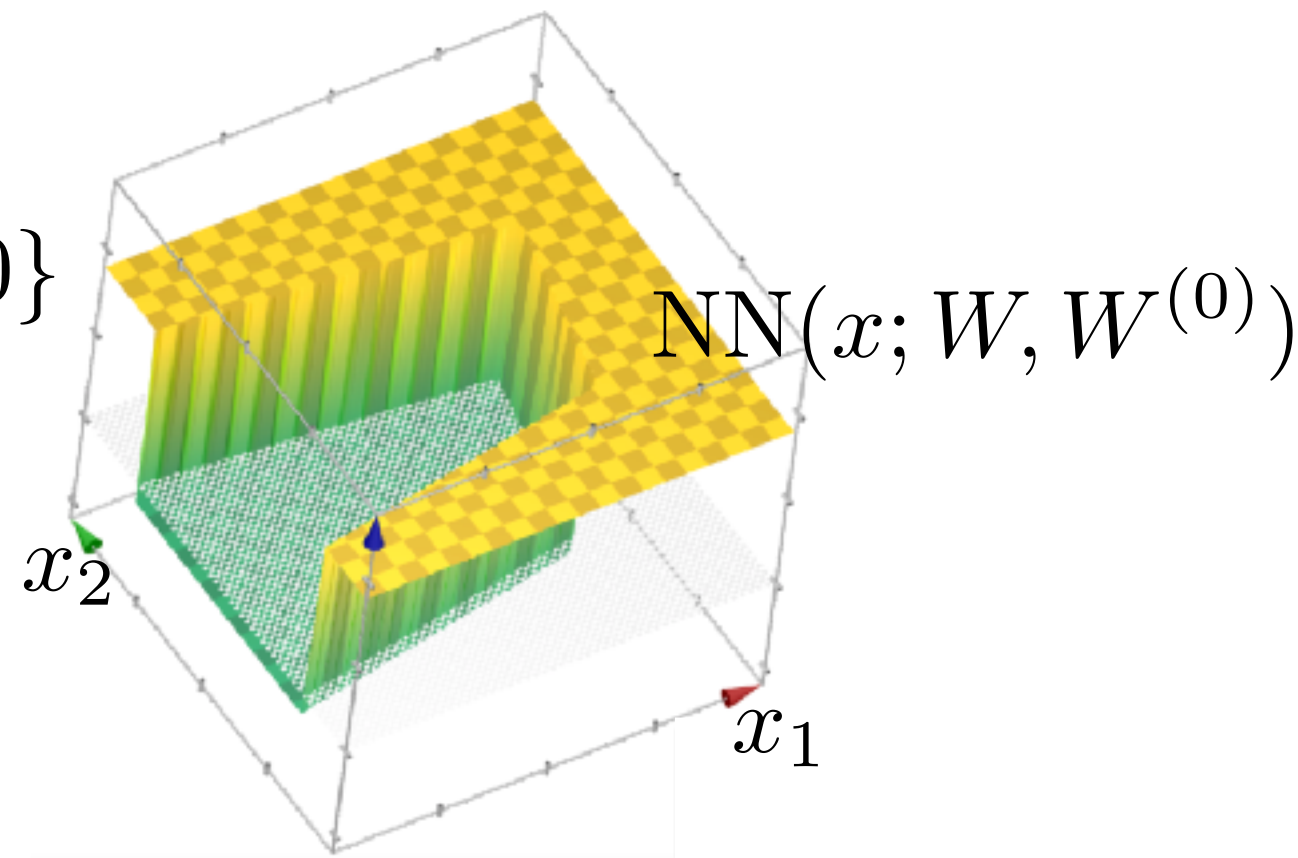


Choices of activation function

- 1st layer: $A_i^{(1)} = f^{(1)}(w_i^{(1)\top} x + w_0^{(1)})$
  - Choose
    $f^{(1)}(z) = \mathbf{1}\{z \geq 0\}$

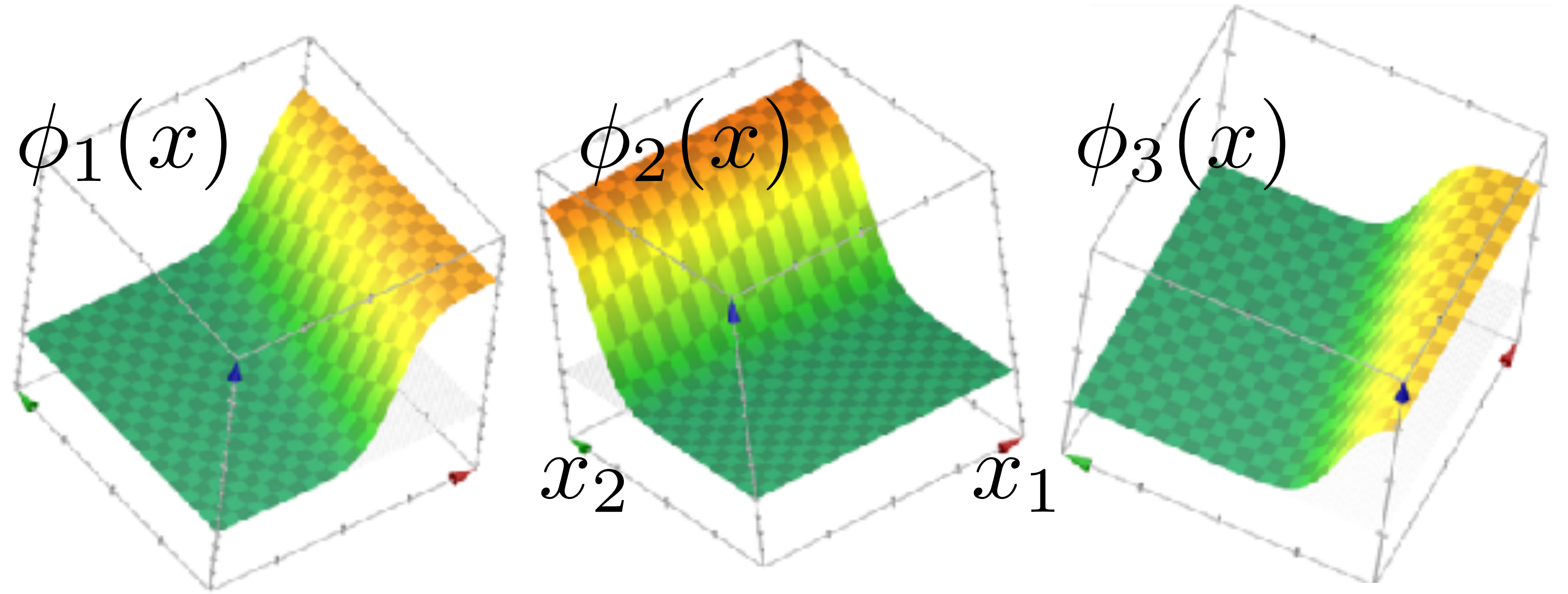


- 2nd layer: $A_i^{(2)} = f^{(2)}(w_i^{(2)\top} A^{(1)} + w_0^{(2)})$
  - Choose
    $f^{(2)}(z) = \mathbf{1}\{z \geq 0\}$



Choices of activation function

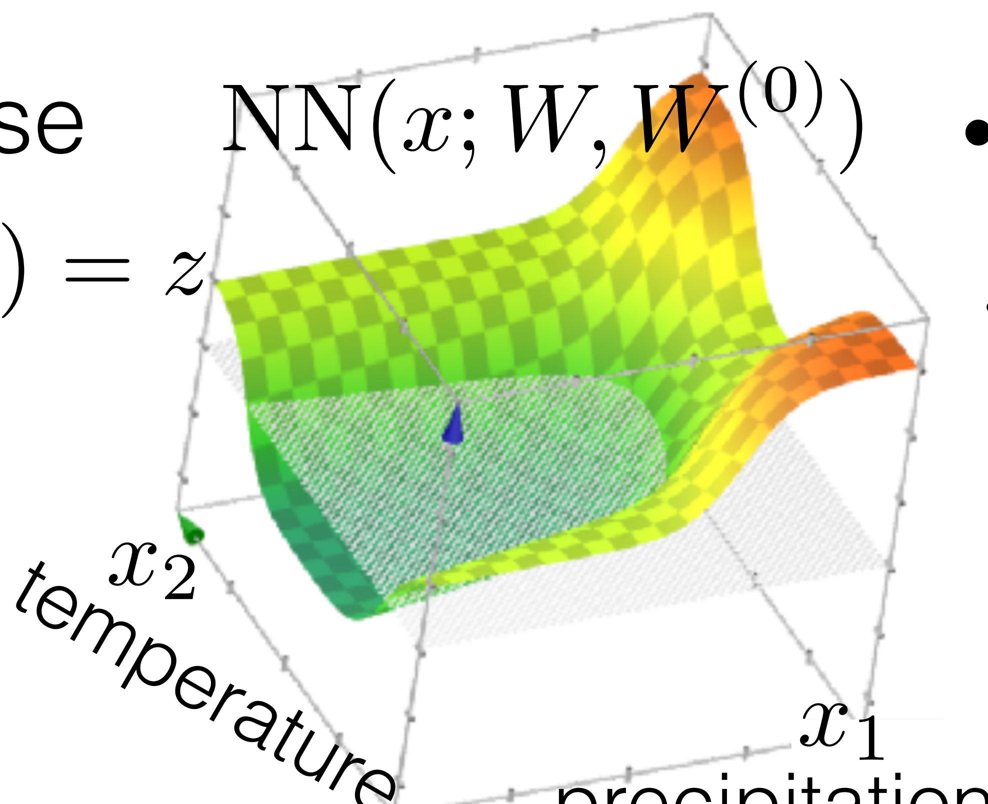
- 1st layer: $A_i^{(1)} = f^{(1)}(w_i^{(1)\top} x + w_0^{(1)})$
  - Choose $f^{(1)}(z) = \sigma(z)$



- 2nd layer: $A_i^{(2)} = f^{(2)}(w_i^{(2)\top} A^{(1)} + w_0^{(2)})$
  - Choose $f^{(2)}(z) = z$
  - Choose $f^{(2)}(z) = \sigma(z)$

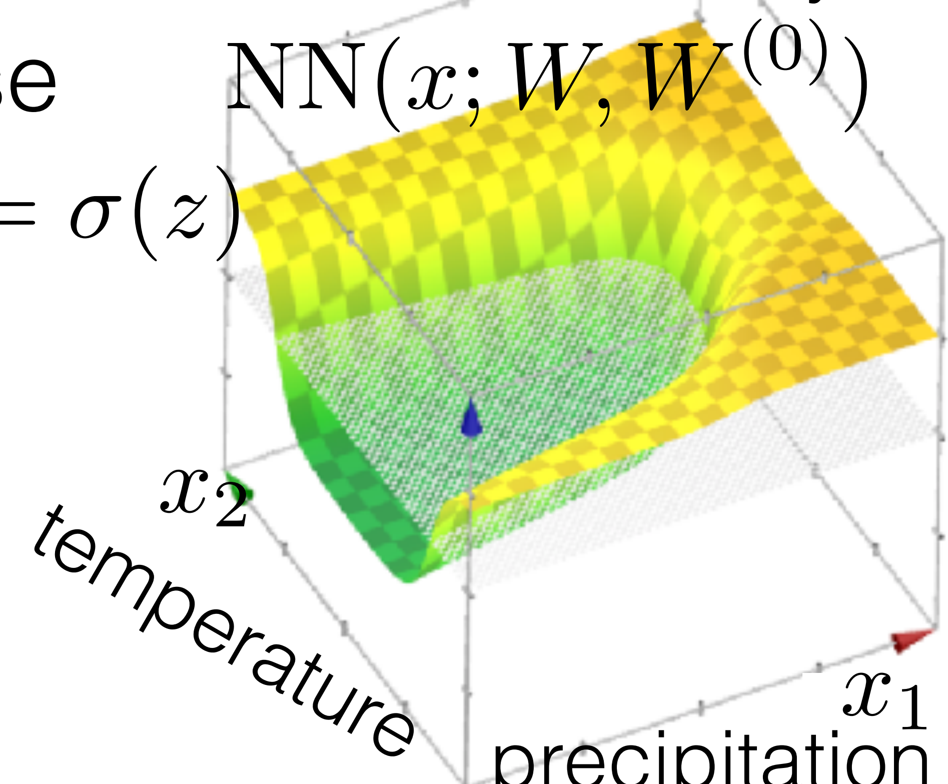
$\mathrm{NN}(x; W, W^{(0)})$

How much will I dislike running today?



$\mathrm{NN}(x; W, W^{(0)})$

Probability that I won't run today



Derivatives: Notation

- Objective: $J(W, W_0) = \frac{1}{n_{\text{data}}} \sum_{i=1}^{n_{\text{data}}} \overbrace{L(\text{NN}(x^{(i)}; W, W_0), y^{(i)})}^{\text{loss}_i}$
- For SGD, we'll want derivatives of loss$_i$ w.r.t. parameters
- Convention: for v of size 1×1 and u of size 1×1

  $\partial v / \partial u$ is the (scalar) partial derivative of v w.r.t. u
  - Example: $\underset{1\text{x}1}{\partial \text{loss}_i} / \underset{1\text{x}1}{\partial W^{(1)}_{1,1}}$ 1x1
- Convention: for v of size 1×1 and u of size $b \times 1$

  $\partial v / \partial u$ is a vector of size $b \times 1$ with j th entry $\partial v / \partial u_j$
  - Example: $\underset{1\text{x}1}{\partial \text{loss}_i} / \underset{n^1\text{x}1}{\partial W^{(1)}_0}$ n^1x1 $= \nabla_{W_0^{(1)}} \text{loss}_i$
- Convention: for v of size $c \times 1$ and u of size $b \times 1$

  $\partial v / \partial u$ is a matrix of size $b \times c$ with (j, k) entry $\partial v_k / \partial u_j$
  - Example: $\underset{n^1\text{x}1}{\partial A^{(1)}} / \underset{n^1\text{x}1}{\partial Z^{(1)}}$ n^1xn^1
- Convention: for v of size 1×1 and u of size $b \times c$

  $\partial v / \partial u$ is a matrix of size $b \times c$ with (j, k) entry $\partial v / \partial u_{j,k}$
  - Example: $\underset{1\text{x}1}{\partial \text{loss}_i} / \underset{m^1\text{x}n^1}{\partial W^{(1)}}$ m^1xn^1

Taking derivatives for SGD

- Example 1-layer NN
  - data dimension $m^{(1)} = 4$; # outputs $n^{(1)} = 3$

$$\underset{3\text{x}1}{A^{(1)}} = \underbrace{W^{(1)}}_{4\text{x}3}{}^{\top} \underset{4\text{x}1}{x} + \underset{3\text{x}1}{W_0^{(1)}}$$

$$\underset{1\text{x}1}{\text{loss}} = L(A^{(1)}, y) = \textstyle\sum_{k=1}^{3} (A_k^{(1)} - y_k)^2$$

- Part of SGD step *t*: randomly choose (*x*,*y*) and update:

value at end of step *t*

$$\underset{1\text{x}1}{W_{0,j}^{(1)}} \leftarrow \underset{1\text{x}1}{W_{0,j}^{(1)}} - \eta(t) \frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}$$

notice: we're dropping *i*

all values on this side are from step *t*-1

- Use the scalar chain rule from calculus:

$$\underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}} = \sum_{k=1}^{3} \underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial A_k^{(1)}}} \underset{1\text{x}1}{\frac{\partial A_k^{(1)}}{\partial W_{0,j}^{(1)}}} = \sum_{k=1}^{3} \frac{\partial A_k^{(1)}}{\partial W_{0,j}^{(1)}} \frac{\partial \text{loss}}{\partial A_k^{(1)}}$$

$$\underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}} = \underset{1\text{x}3}{\frac{\partial A^{(1)}}{\partial W_{0,j}^{(1)}}} \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(1)}}} \qquad \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W_0^{(1)}}} = \underset{3\text{x}3}{\frac{\partial A^{(1)}}{\partial W_0^{(1)}}} \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(1)}}}$$

Taking derivatives for SGD

- Example 1-layer NN
  - data dimension $m^{(1)} = 4$; # outputs $n^{(1)} = 3$

$$\underset{3\text{x}1}{A^{(1)}} = \underbrace{W^{(1)}}_{4\text{x}3}{}^{\top} \underset{4\text{x}1}{x} + \underset{3\text{x}1}{W_0^{(1)}}$$

1x1 $\text{loss} = L(A^{(1)}, y) = \sum_{k=1}^{3} (A_k^{(1)} - y_k)^2$

- Part of SGD step *t*: randomly choose (*x*,*y*) and update:

value at end of step *t*

$$\underset{1\text{x}1}{W_{0,j}^{(1)}} \leftarrow \underset{1\text{x}1}{W_{0,j}^{(1)}} - \eta(t) \frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}$$

notice: we're dropping *i*

all values on this side are from step *t*-1

- Use the scalar chain rule from calculus:

$$\underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}} = \sum_{k=1}^{3} \underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial A_k^{(1)}}} \underset{1\text{x}1}{\frac{\partial A_k^{(1)}}{\partial W_{0,j}^{(1)}}} = \sum_{k=1}^{3} \frac{\partial A_k^{(1)}}{\partial W_{0,j}^{(1)}} \frac{\partial \text{loss}}{\partial A_k^{(1)}}$$

Exercise: compute this matrix

$$\underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}} = \underset{1\text{x}3}{\frac{\partial A^{(1)}}{\partial W_{0,j}^{(1)}}} \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(1)}}} \qquad \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W_0^{(1)}}} = \underset{3\text{x}3}{\boxed{\frac{\partial A^{(1)}}{\partial W_0^{(1)}}}} \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(1)}}}$$

Taking derivatives for SGD

- Example 1-layer NN
  - data dimension $m^{(1)} = 4$; # outputs $n^{(1)} = 3$

$$\underset{3\text{x}1}{A^{(1)}} = \underbrace{W^{(1)\top}}_{4\text{x}3} \underset{4\text{x}1}{x} + \underset{3\text{x}1}{W_0^{(1)}}$$

$$1\text{x}1 \quad \text{loss} = L(A^{(1)}, y) = \textstyle\sum_{k=1}^{3} (A_k^{(1)} - y_k)^2$$

- Part of SGD step *t*: randomly choose (*x*,*y*) and update:

value at end of step *t*

$$\underset{1\text{x}1}{W_{0,j}^{(1)}} \leftarrow \underset{1\text{x}1}{W_{0,j}^{(1)}} - \eta(t) \frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}}$$

notice: we're dropping *i*

all values on this side are from step *t*-1

- Use the scalar chain rule from calculus:

$$\frac{\partial \text{loss}}{\partial W_{0,j}^{(1)}} = \sum_{k=1}^{3} \frac{\partial \text{loss}}{\partial A_k^{(1)}} \frac{\partial A_k^{(1)}}{\partial W_{0,j}^{(1)}} = \sum_{k=1}^{3} \frac{\partial A_k^{(1)}}{\partial W_{0,j}^{(1)}} \frac{\partial \text{loss}}{\partial A_k^{(1)}}$$

Exercise: compute this matrix

Notice: We can't just substitute the weight **matrix** in the final formula. (Why not?)

$$\underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W_0^{(1)}}} = \underset{3\text{x}3}{\frac{\partial A^{(1)}}{\partial W_0^{(1)}}} \underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(1)}}}$$

Taking derivatives for SGD

- Example 1-layer NN
  - data dimension $m^{(1)} = 4$; # outputs $n^{(1)} = 3$

$$\underset{3\text{x}1}{A^{(1)}} = \underbrace{W^{(1)}}_{4\text{x}3}{}^{\top} \underset{4\text{x}1}{x} + \underset{3\text{x}1}{W_0^{(1)}}$$

1x1 $\text{loss} = L(A^{(1)}, y) = \sum_{k=1}^{3} (A_k^{(1)} - y_k)^2$

- Our last derivation doesn't work for a full weight matrix. (Exercise: why not?) So what about the full matrix case?

1x1 $$\frac{\partial \text{loss}}{\partial W_{j,k}^{(1)}} = \sum_{p=1}^{3} \frac{\partial A_p^{(1)}}{\partial W_{j,k}^{(1)}} \frac{\partial \text{loss}}{\partial A_p^{(1)}} = \frac{\partial A_k^{(1)}}{\partial W_{j,k}^{(1)}} \frac{\partial \text{loss}}{\partial A_k^{(1)}} = x_j \frac{\partial \text{loss}}{\partial A_k^{(1)}}$$

- Observe: if u is a $c \times 1$ vector and v is a $b \times 1$ vector, $vu^\top$ is a matrix of size $b \times c$ with (j, k) entry $v_j u_k$
- By our convention, $\partial \text{loss} / \partial W^{(1)}$ is a matrix with (j, k) entry $\partial \text{loss} / \partial W_{j,k}^{(1)}$
  - So: $$\underset{4\text{x}3}{\frac{\partial \text{loss}}{\partial W^{(1)}}} = \underset{4\text{x}1}{x} \underbrace{\left(\frac{\partial \text{loss}}{\partial A^{(1)}} \right)}_{3\text{x}1}{}^{\top}$$

Taking derivatives for SGD

- General case: NN with L layers
  - Layer ℓ has $m^{(l)}$ inputs and $n^{(l)} = m^{(l+1)}$ outputs

$$A^{(\ell)} = f^{\ell}(Z^{(\ell)}), Z^{(\ell)} = W^{(\ell)\top} A^{(\ell-1)} + W_0^{(\ell)} \quad \text{with} \quad A^{(0)} = x$$

- To find $\partial \text{loss}/\partial W^{(\ell)}$, break off the “final” weight derivative

$$\underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial W_{j,k}^{(\ell)}}} = \underset{1\text{x}1}{\frac{\partial Z_k^{(\ell)}}{\partial W_{j,k}^{(\ell)}}} \underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial Z_k^{(\ell)}}} = \underset{1\text{x}1}{A_j^{(\ell-1)}} \underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial Z_k^{(\ell)}}} \Rightarrow \underset{m^{\ell}\text{x}n^{\ell}}{\frac{\partial \text{loss}}{\partial W^{(\ell)}}} = \underset{m^{\ell}\text{x}1}{A^{(\ell-1)}} \underbrace{\left(\frac{\partial \text{loss}}{\partial Z^{(\ell)}}\right)^{\top}}_{1\text{x}n^{\ell}}$$

  - It remains to find $\partial \text{loss}/\partial Z^{(\ell)}$

$$\underset{n^{\ell}\text{x}1}{\frac{\partial \text{loss}}{\partial Z^{(\ell)}}} = \underset{n^{\ell}\text{x}n^{\ell}}{\frac{\partial A^{(\ell)}}{\partial Z^{(\ell)}}} \underset{n^{\ell}\text{x}n^{\ell+1}}{\frac{\partial Z^{(\ell+1)}}{\partial A^{(\ell)}}} \underset{n^{\ell+1}\text{x}n^{\ell+1}}{\frac{\partial A^{(\ell+1)}}{\partial Z^{(\ell+1)}}} \cdots \underset{n^{L-1}\text{x}n^{L-1}}{\frac{\partial A^{(L-1)}}{\partial Z^{(L-1)}}} \underset{n^{L-1}\text{x}n^{L}}{\frac{\partial Z^{(L)}}{\partial A^{(L-1)}}} \underset{n^{L}\text{x}n^{L}}{\frac{\partial A^{(L)}}{\partial Z^{(L)}}} \underset{n^{L}\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(L)}}}$$

    - Lots of this computation will be shared across layers
    - More efficient to compute the shared parts only once

- Can similarly show: $\partial \text{loss}/\partial W_0^{(\ell)} = \partial \text{loss}/\partial Z^{(\ell)}$

Taking derivatives for SGD

Exercise: use these formulas to show that the final derivatives w.r.t. W^ℓ are 0 if any layer above ℓ has step function activations

Exercise: Check that you agree with these formulas!

- … ers
  … nd … $A^{(\ell-1)} + W_0^{(\ell)}$ with $A^{(0)} = x$
- … ff the "final" weight derivative

$$\underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial W^{(\ell)}_{j,k}}} = \underset{1\text{x}1}{\frac{\partial Z^{(\ell)}_k}{\partial W^{(\ell)}_{j,k}}} \underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial Z^{(\ell)}_k}} = \underset{1\text{x}1}{A^{(\ell-1)}_j} \underset{1\text{x}1}{\frac{\partial \text{loss}}{\partial Z^{(\ell)}_k}} \quad \Rightarrow \quad \underset{m^\ell\text{x}n^\ell}{\frac{\partial \text{loss}}{\partial W^{(\ell)}}} = \underset{m^\ell\text{x}1}{A^{(\ell-1)}} \underbrace{\left(\frac{\partial \text{loss}}{\partial Z^{(\ell)}}\right)^\top}_{1\text{x}n^\ell}$$

- It remains to find $\partial \text{loss}/\partial Z^{(\ell)}$

$$\underset{n^\ell\text{x}1}{\frac{\partial \text{loss}}{\partial Z^{(\ell)}}} = \underset{n^\ell\text{x}n^\ell}{\frac{\partial A^{(\ell)}}{\partial Z^{(\ell)}}} \underset{n^\ell\text{x}n^{\ell+1}}{\frac{\partial Z^{(\ell+1)}}{\partial A^{(\ell)}}} \underset{n^{\ell+1}\text{x}n^{\ell+1}}{\frac{\partial A^{(\ell+1)}}{\partial Z^{(\ell+1)}}} \cdots \underset{n^{L-1}\text{x}n^{L-1}}{\frac{\partial A^{(L-1)}}{\partial Z^{(L-1)}}} \underset{n^{L-1}\text{x}n^L}{\frac{\partial Z^{(L)}}{\partial A^{(L-1)}}} \underset{n^L\text{x}n^L}{\frac{\partial A^{(L)}}{\partial Z^{(L)}}} \underset{n^L\text{x}1}{\frac{\partial \text{loss}}{\partial A^{(L)}}}$$

  - Lots of this computation will be shared across layers
  - More efficient to compute the shared parts only once

- Can similarly show: $\partial \text{loss}/\partial W_0^{(\ell)} = \partial \text{loss}/\partial Z^{(\ell)}$