Numerical Analysis for Artificial Intelligence, Week 5

UCSD Summer session II 2018

CSE 190

Jacek Cyranka

High dimensional regression data

Let X be such that for all i $X_i = random([-1,1])^{10}$, and

$$\begin{aligned} y_1 &= P_1(X_1, X_2, \ldots, X_t), \\ y_2 &= P_2(X_1, X_2, \ldots, X_t), \\ \ldots, \\ y_n &= P_n(X_1, X_2, \ldots, X_t), \end{aligned}$$

where P_i's are some polynomials, i.e.

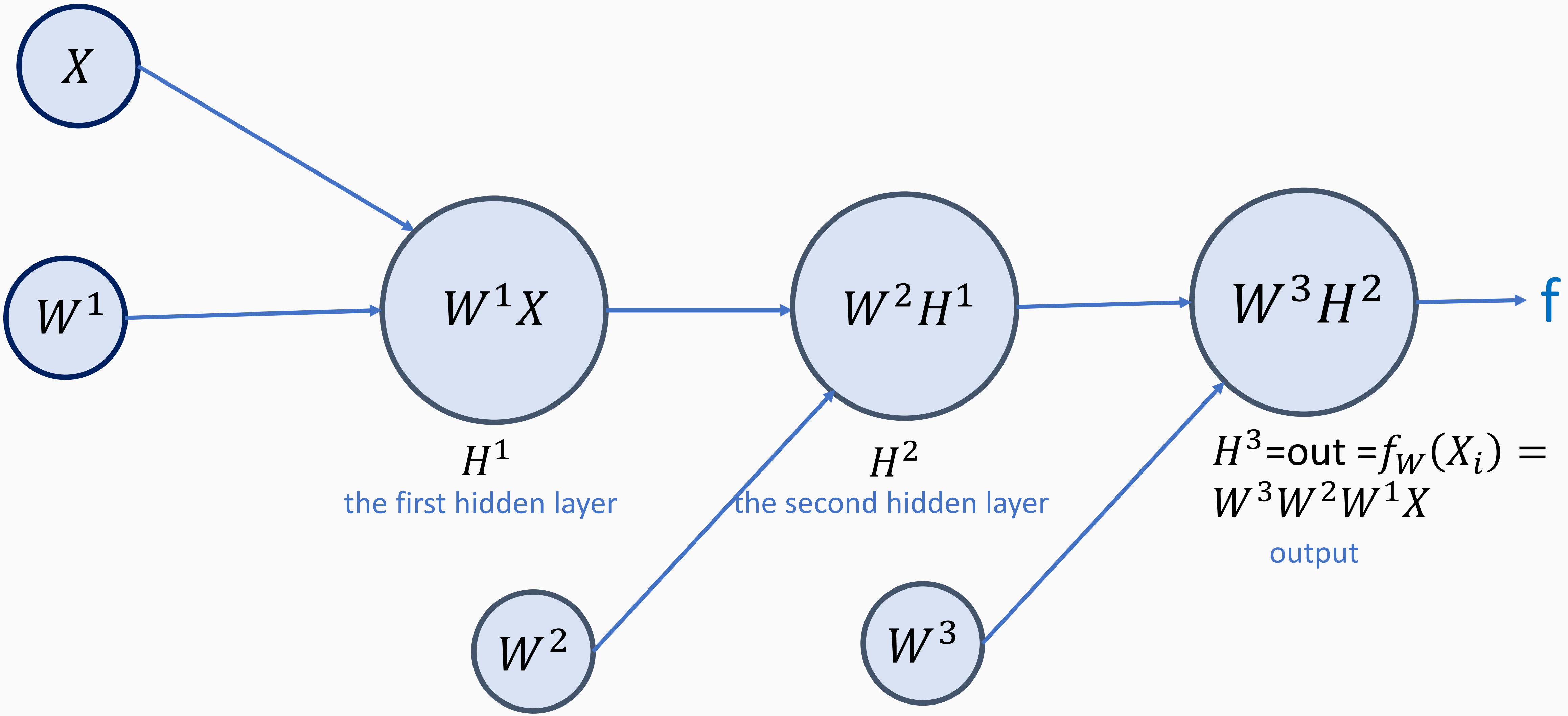
$$P_i \colon \mathbb{R}^{10\times 10} \to \mathbb{R}^{10}.$$

TOPIC:
DEEPER NEURAL NETWORKS FOR SUPERVISED LEARNING

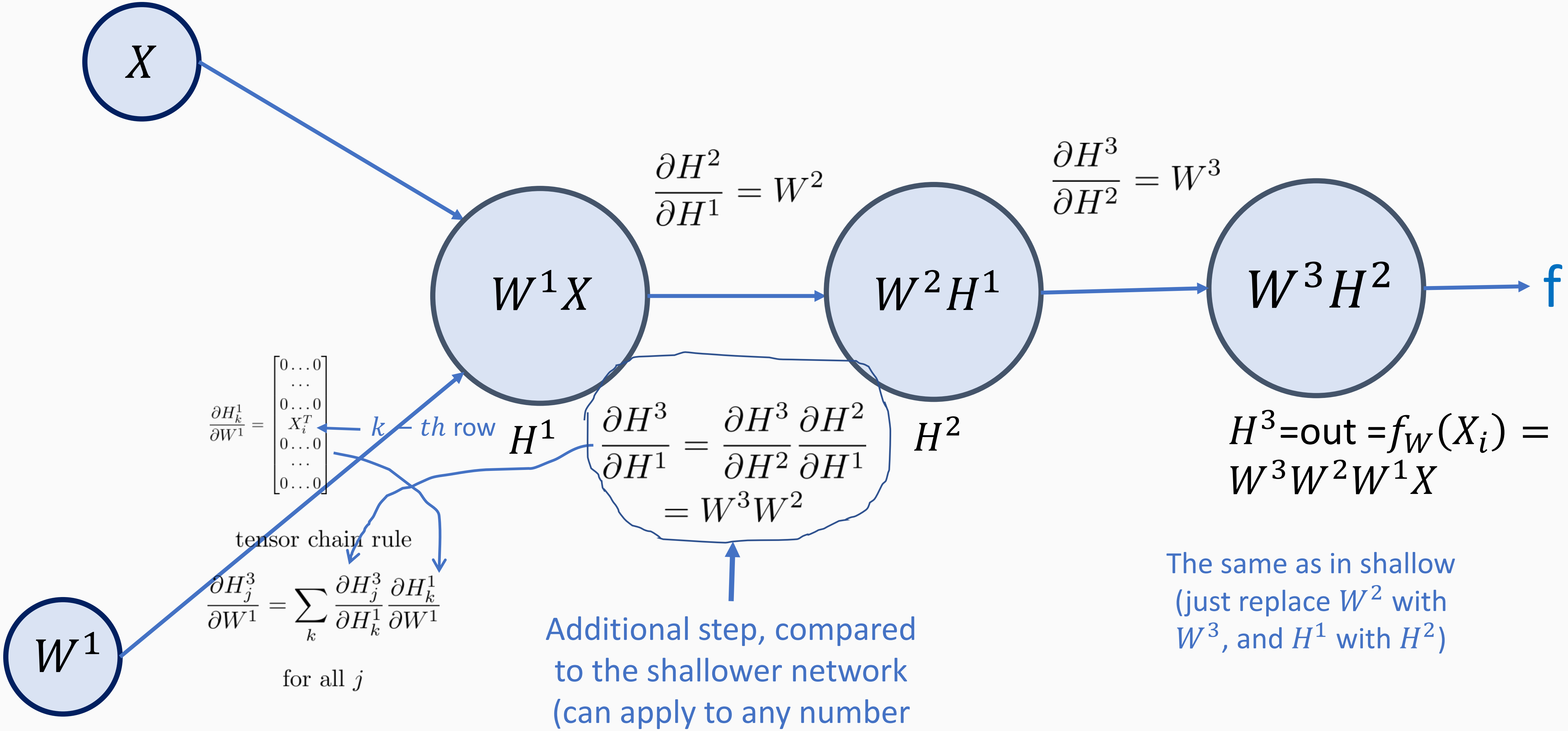
Deeper nets

The meme which is everywhere, so I include it too here lol

Deeper nets


X
W^1
W^1X
W^2H^1
W^3H^2
f
H^1
the first hidden layer
H^2
the second hidden layer
H^3=out =$f_W(X_i) =$
$W^3W^2W^1X$
output
W^2
W^3

Backprop for $NN(W, X) = W^3 W^2 W^1 X$



$$\frac{\partial H^2}{\partial H^1} = W^2$$

$$\frac{\partial H^3}{\partial H^2} = W^3$$

$$\frac{\partial H^1_k}{\partial W^1} = \begin{bmatrix} 0 \dots 0 \\ \dots \\ 0 \dots 0 \\ X_i^T \\ 0 \dots 0 \\ \dots \\ 0 \dots 0 \end{bmatrix}$$

$k-th$ row

tensor chain rule

$$\frac{\partial H^3_j}{\partial W^1} = \sum_k \frac{\partial H^3_j}{\partial H^1_k} \frac{\partial H^1_k}{\partial W^1}$$

for all j

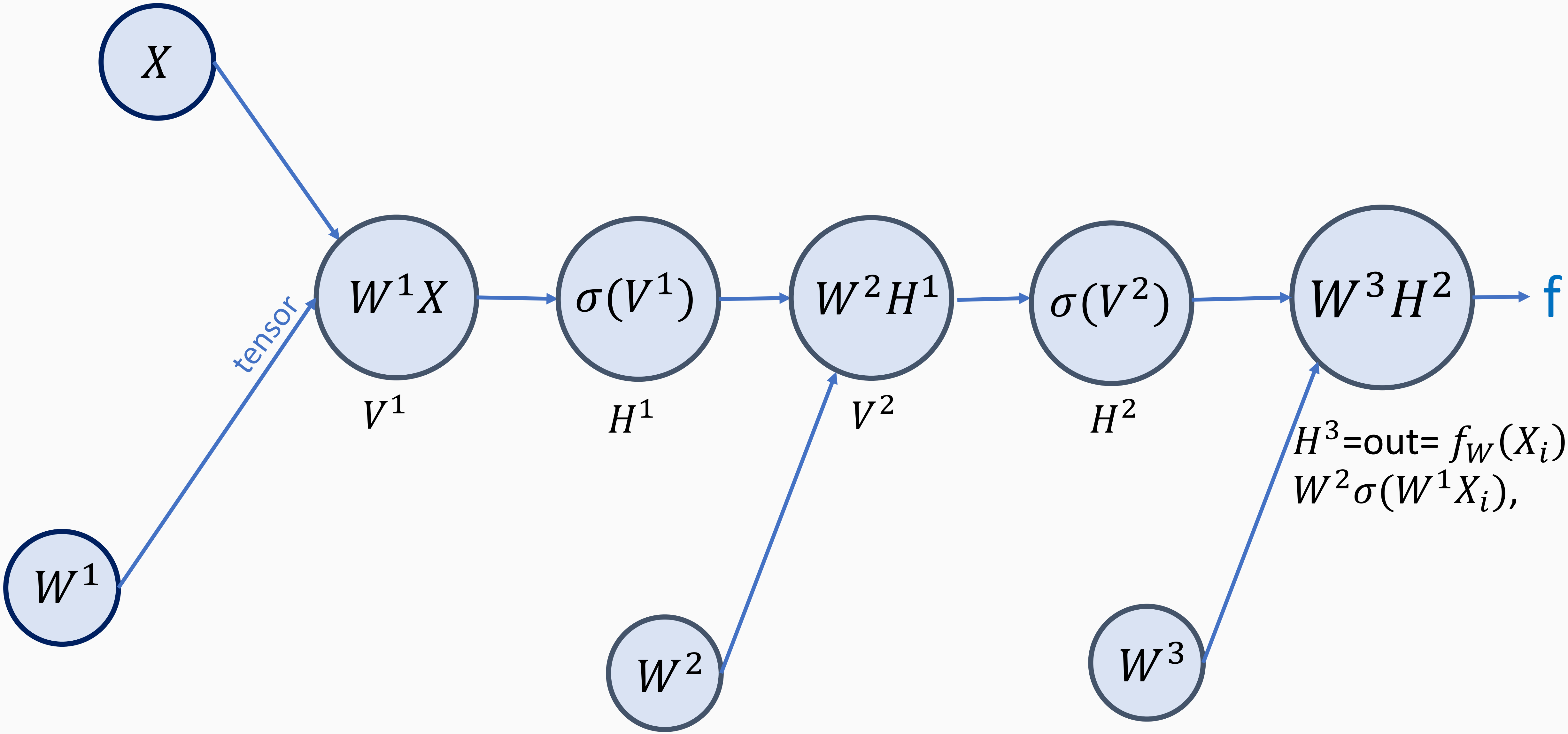
$$\frac{\partial H^3}{\partial H^1} = \frac{\partial H^3}{\partial H^2} \frac{\partial H^2}{\partial H^1} = W^3 W^2$$

Additional step, compared to the shallower network (can apply to any number of layers)

H^3=out $= f_W(X_i) = W^3 W^2 W^1 X$

The same as in shallow (just replace W^2 with W^3, and H^1 with H^2)

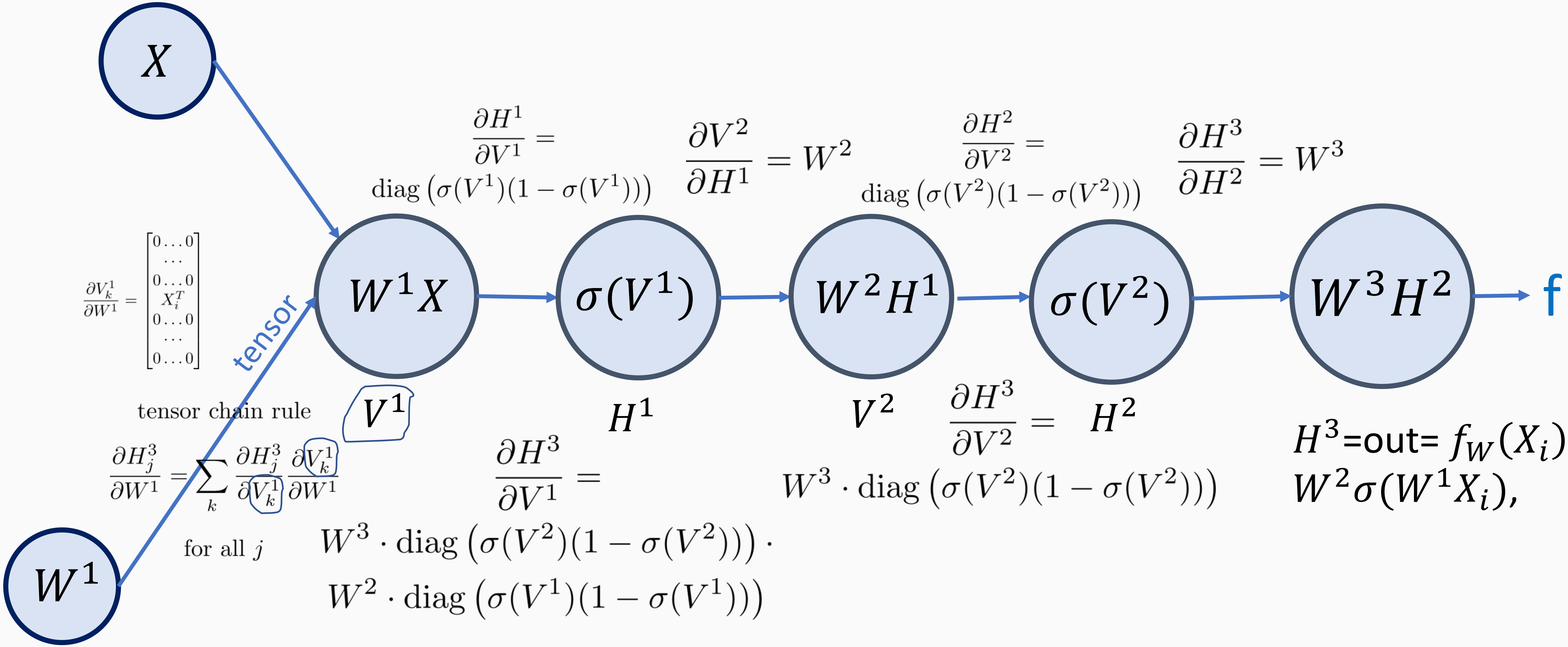
Deeper Sigmoidal network



H^3=out= $f_W(X_i) =$
$W^2\sigma(W^1X_i)$,

Backprop for deeper sigmoidal network

$NN(W,X) = W^3\sigma(W^2\sigma(W^1X))$



$$\frac{\partial H^1}{\partial V^1} = \operatorname{diag}\left(\sigma(V^1)(1-\sigma(V^1))\right)$$

$$\frac{\partial V^2}{\partial H^1} = W^2$$

$$\frac{\partial H^2}{\partial V^2} = \operatorname{diag}\left(\sigma(V^2)(1-\sigma(V^2))\right)$$

$$\frac{\partial H^3}{\partial H^2} = W^3$$

$$\frac{\partial V_k^1}{\partial W^1} = \begin{bmatrix} 0\dots 0 \\ \cdots \\ 0\dots 0 \\ X_i^T \\ 0\dots 0 \\ \cdots \\ 0\dots 0 \end{bmatrix}$$

tensor chain rule

$$\frac{\partial H_j^3}{\partial W^1} = \sum_k \frac{\partial H_j^3}{\partial V_k^1}\frac{\partial V_k^1}{\partial W^1}$$

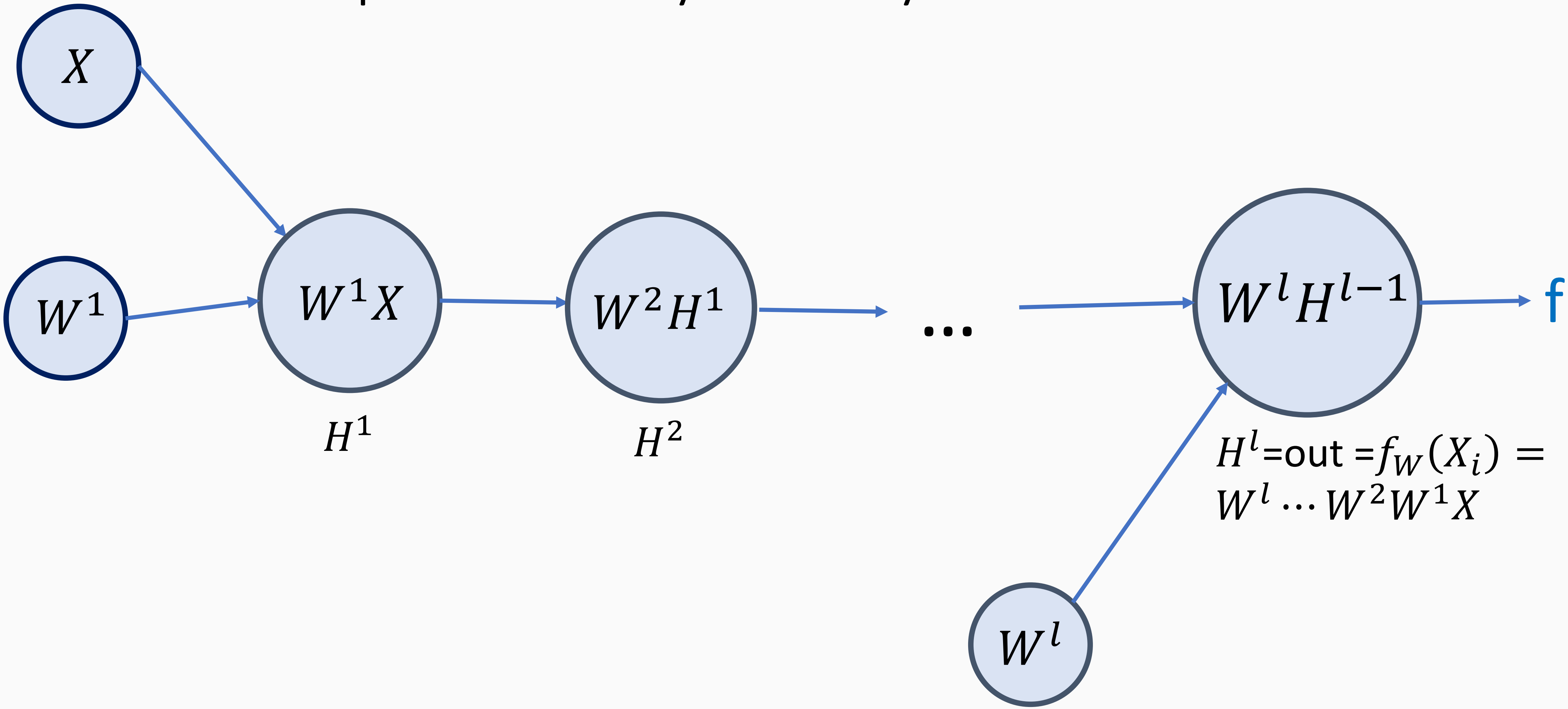
for all j

$$\frac{\partial H^3}{\partial V^1} = W^3 \cdot \operatorname{diag}\left(\sigma(V^2)(1-\sigma(V^2))\right) \cdot W^2 \cdot \operatorname{diag}\left(\sigma(V^1)(1-\sigma(V^1))\right)$$

$$\frac{\partial H^3}{\partial V^2} = W^3 \cdot \operatorname{diag}\left(\sigma(V^2)(1-\sigma(V^2))\right)$$

H^3=out= $f_W(X_i) = W^2\sigma(W^1X_i)$,

Next step : Arbitrary # of layers

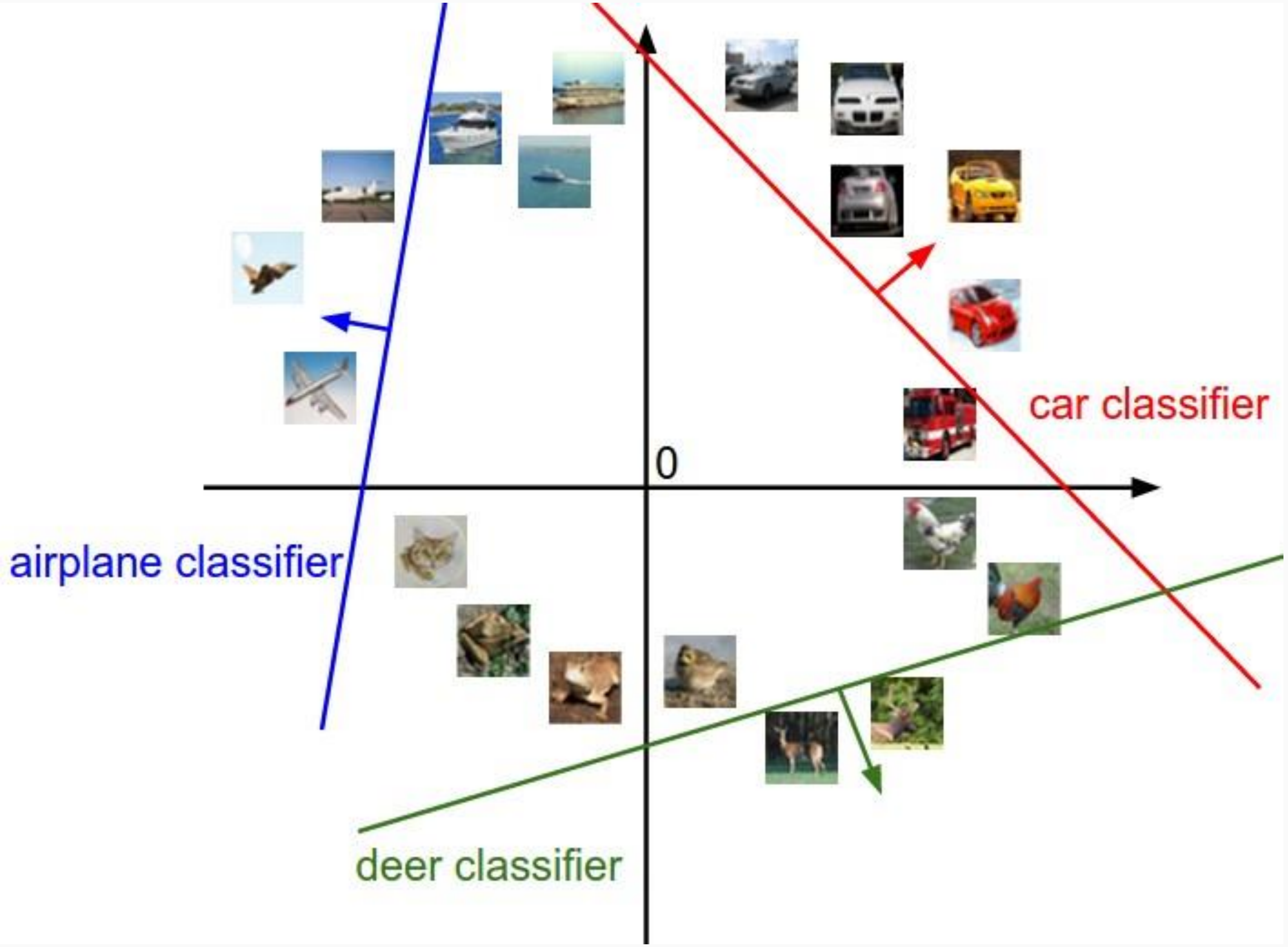


H^l=out =$f_W(X_i) =$
$W^l \cdots W^2W^1X$

Topic:
Classification using neural networks

Classification using NN

1. Linear classifier



SVM classifier

Given t training examples $\{x_i\}_{i=1}^{t}$, assign a class y_i to each of them, assuming the classifier has k outputs each corresponding to a class, y_i is the index of the output corresponding to the desired class for i-th example.

Let $f(x_i, W)$ be the output of given classifier for the input x_i and (trainable) weights W.
Hence, each class receives computed score $s_j = f(x_i, W)_j$ for $i = 1, \ldots, k$

Then, SVM loss to be minimized is given by

j-th class, $j \neq y_i$, 'error' in classifying

$$L = \sum_{i=1}^{t} L_i,$$

$$L_i = \sum_{j \neq y_j} \max\{0, s_j - s_{y_i} + \Delta\}$$

fixed margin

sum over all wrong classes

I. Linear SVM

The simplest SVM classifier is given by $f(x_i, W) = W \cdot x_i$, where $W \in k * n$

Number of classes (for the assignment $W \in 10 * 10$)

Then, SVM loss to be minimized is given by

j-th row of W, $W = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$

$$L = \sum_{i=1}^{t} L_i,$$

$$L_i = \sum_{j \neq y_i} \max\{0, w_j \cdot x_i - w_{y_i} \cdot x_i + \Delta\},$$

dot product

$$\nabla_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} 1\left(w_j \cdot x_i - w_{y_i} \cdot x_i + \Delta > 0\right) \right) x_i \quad \text{for all } i = 1, \ldots, t.$$

Gradient with respect to y_i'th row of the whole weight matrix W (10d vector)

Minimization of linear SVM loss

As previously use gradient descent

$$W := W - \alpha \sum_{i=1}^{t} \nabla_W L_i,$$

This is $10 * 10$ matrix

where each $\nabla_W L_i$ is given by the matrix

$$\nabla_W L_i = \begin{pmatrix} 0 \ldots 0 \\ \vdots \\ \nabla_{w_{y_i}}(L_i) \\ \vdots \\ 0 \ldots 0 \end{pmatrix}$$

$y_i - th$ row
And other elements are 0's

Recap

1. Linear algebra review (vectors/matrices/linear regression),
2. Calculus review (critical points minima/maxima/saddles) partial derivatives, second derivative test, chain rule,
3. Convex / Nonconvex optimization, naïve optimization,
4. Convex optimization using gradient descent, backpropagation, gradient checking, learning rate (step-size adjustment),
5. Problem of finding minimum of quadratic functions,
6. Solving linear regression using GD,
7. Supervised learning for NN –
    a) partial derivatives of a loss function,
    b) Tensor calculus,
    c) Backprop,
    d) Gradient descent,

THE END