Deep Learning Szeminárium
2. előadás

Adrián Csiszárik
(Rényi AI research group)

2022. 02. 18.

Outline

- Mesterséges neuron, mesterséges neutrális háló
- Előrecsatolt / rekurrens hálózat
- Aktivációs függvény
- Veszteségfüggvény (hibafüggvény)
- Neurális hálózatok tanítása
- (Stochastic) Gradient Descent (SGD)
- Minibatch méret
- Tanulási ráta (learning rate)
- Hibavisszaterjesztés (backpropagation)
- Tanító / validációs / teszt adatok
- Kiértékelés: tanulási és általánosítási hiba
- Regularizáció

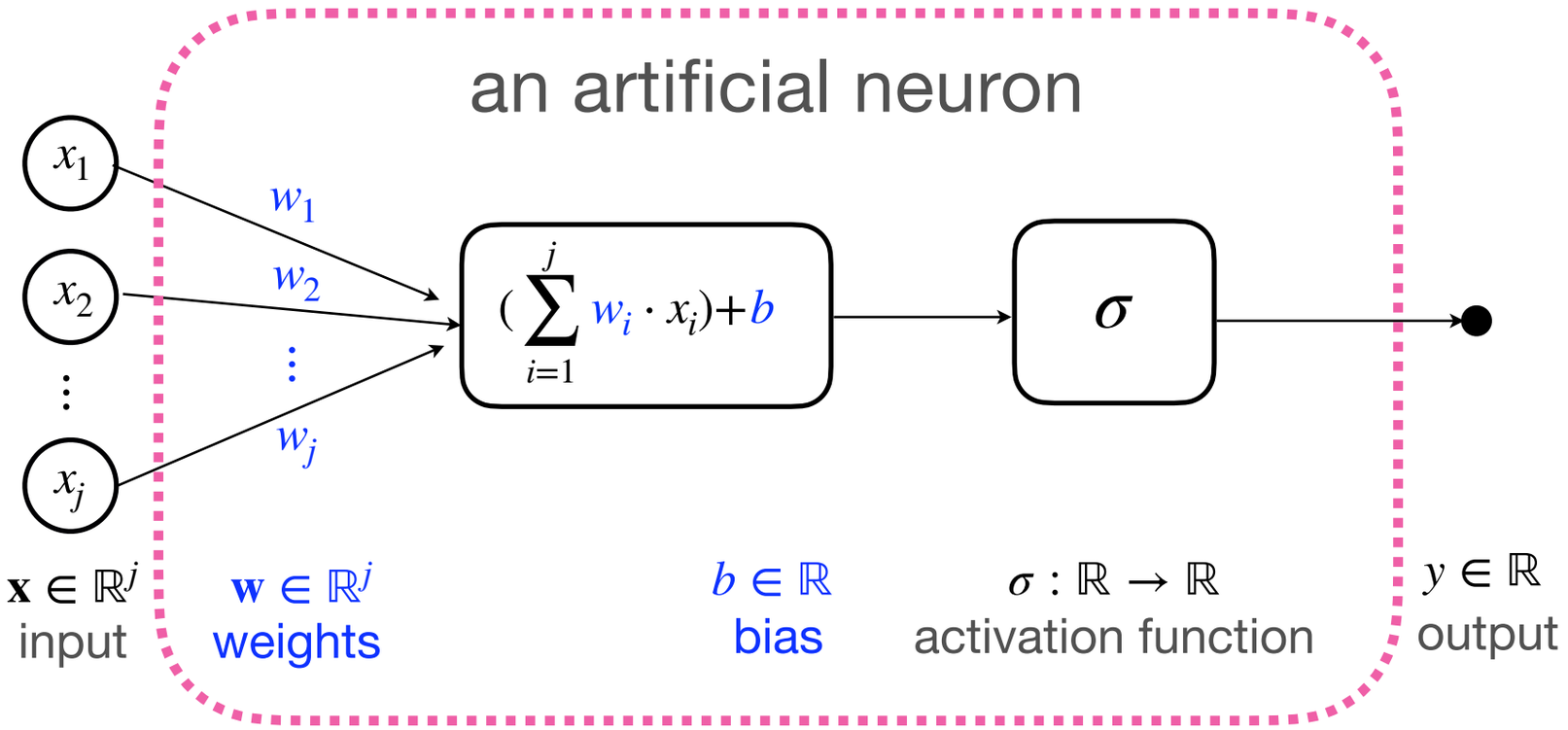
Neural networks

Artificial neural networks are parametrized function classes:

$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m$$

$\theta \in \mathbb{R}^M$ is a real parameter vector

with a specific structure, its building blocks are artificial neurons:



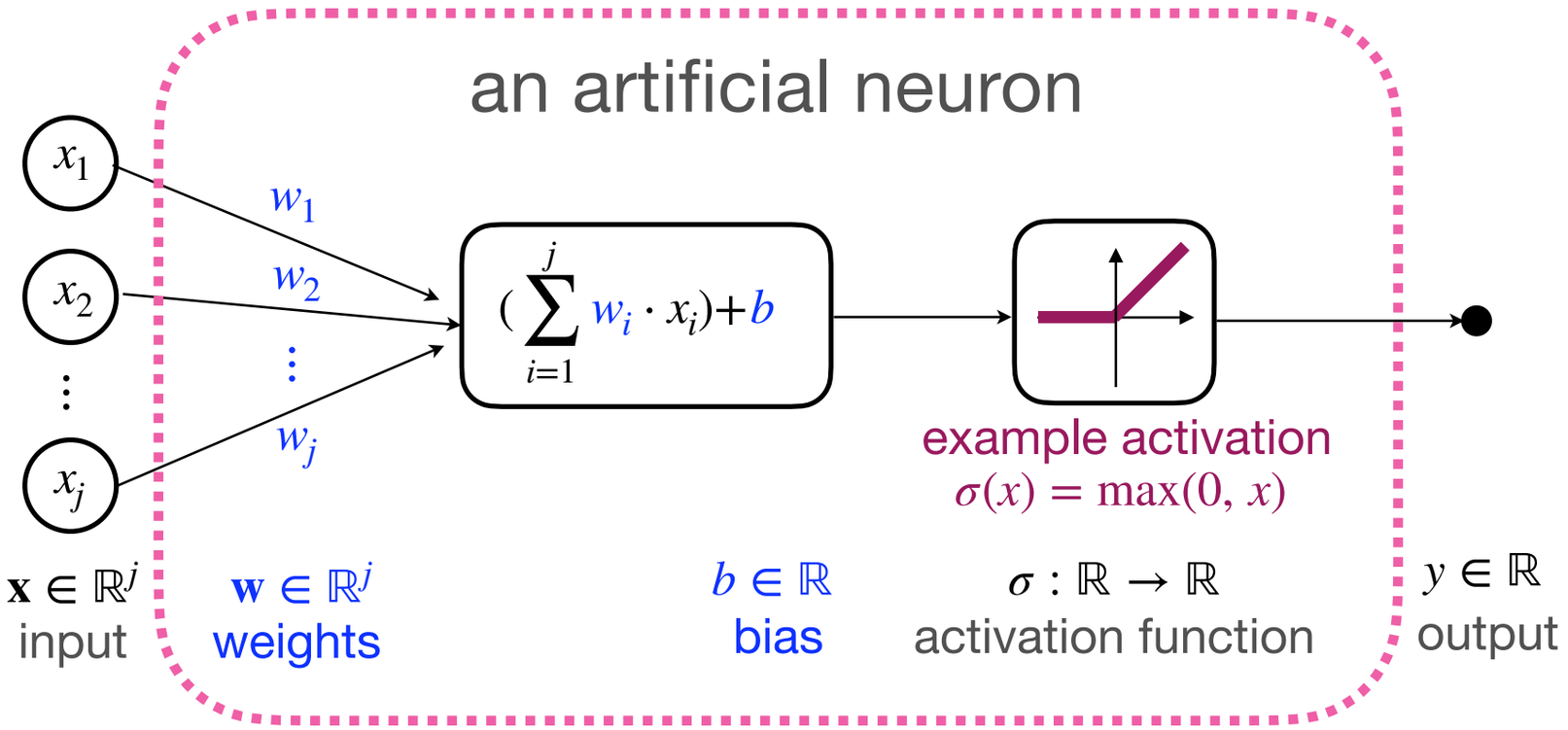
Neural networks

Artificial neural networks are parametrized function classes:

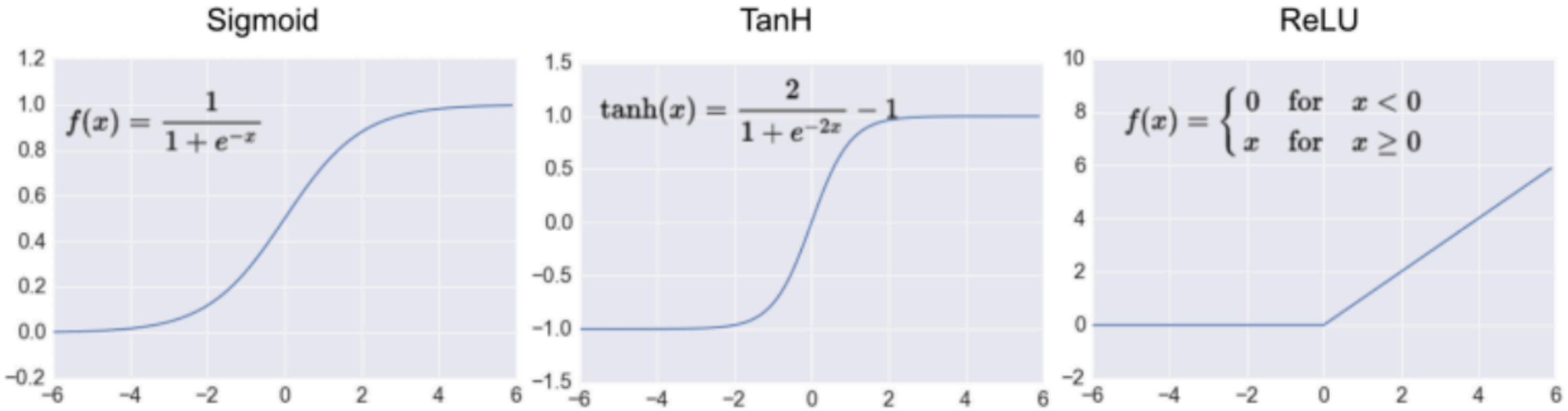
$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m$$

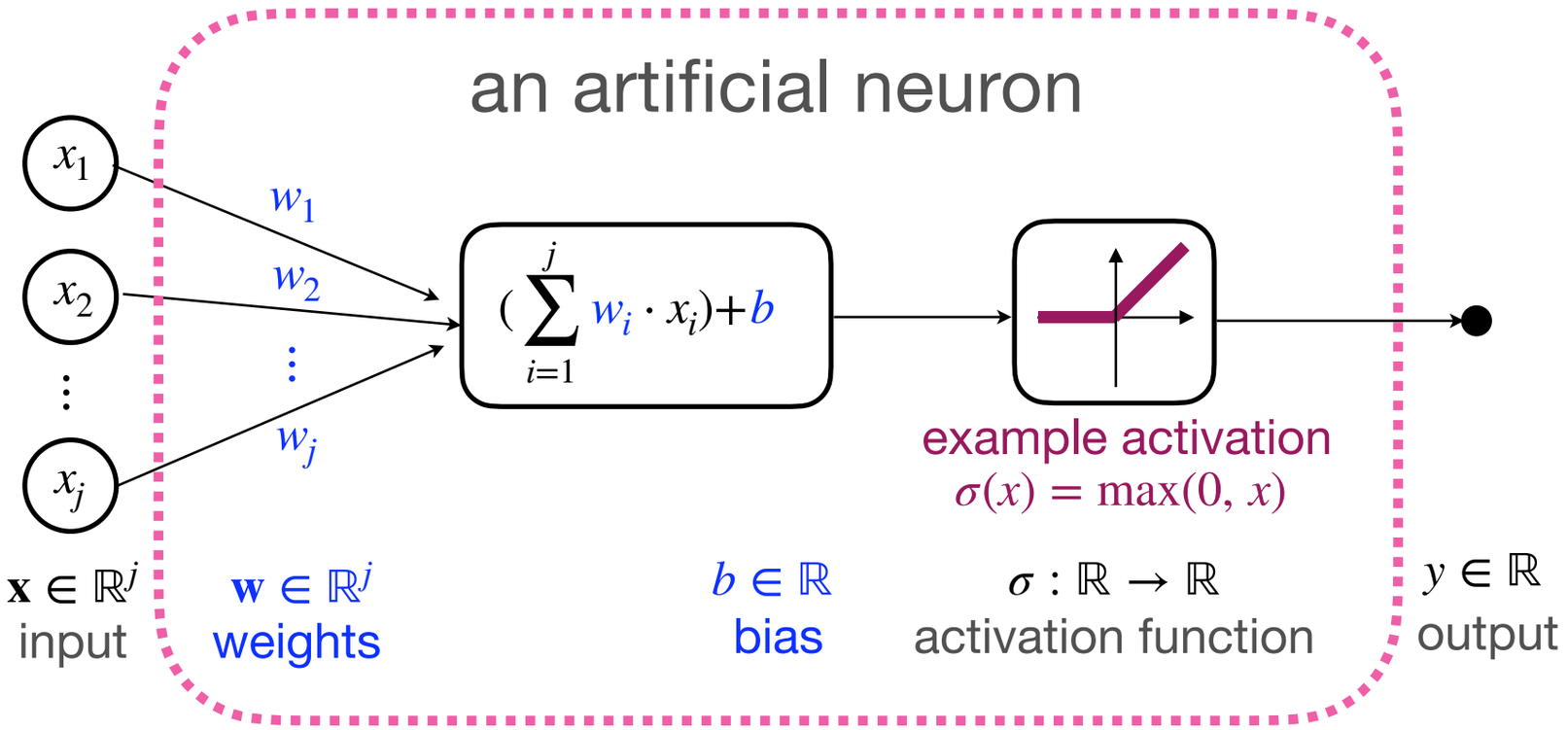
$\theta \in \mathbb{R}^M$ is a real parameter vector

with a specific structure, its building blocks are artificial neurons:



More examples of activation functions





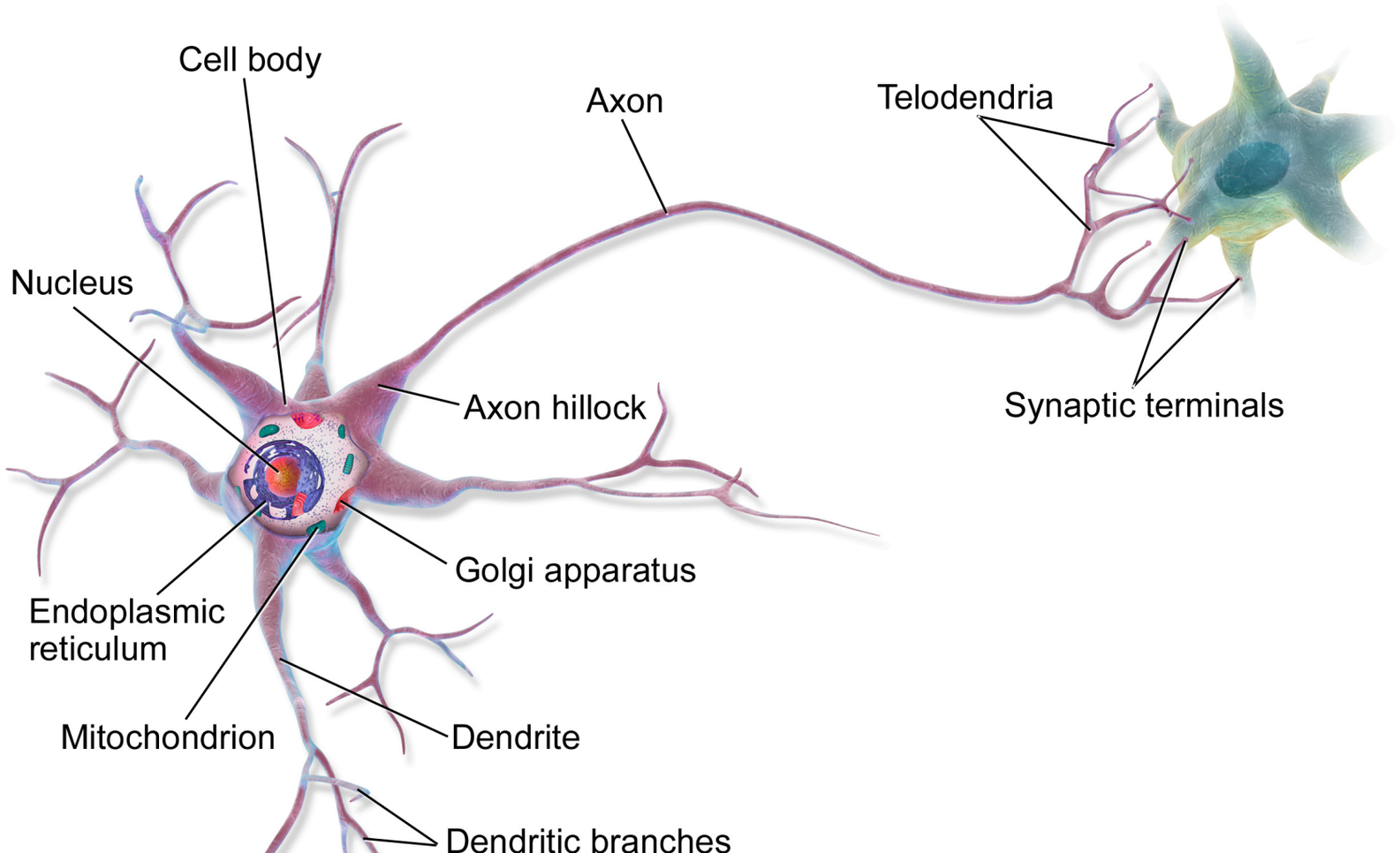
Neural networks

Artificial neural networks are parametrized function classes:

$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m$$

$\theta \in \mathbb{R}^M$ is a real parameter vector

with a specific structure, its building blocks are artificial neurons:



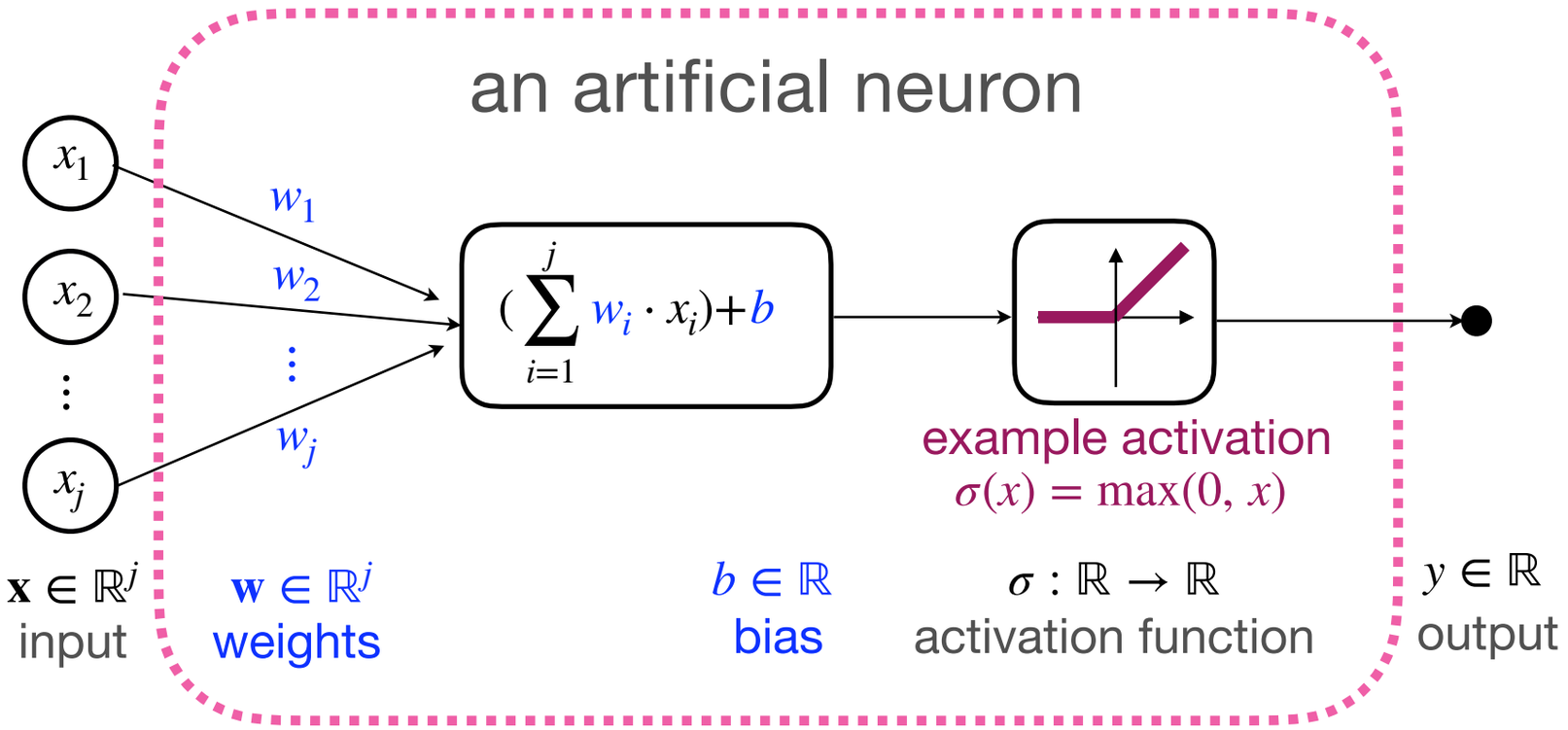
Neural networks

Artificial neural networks are parametrized function classes:

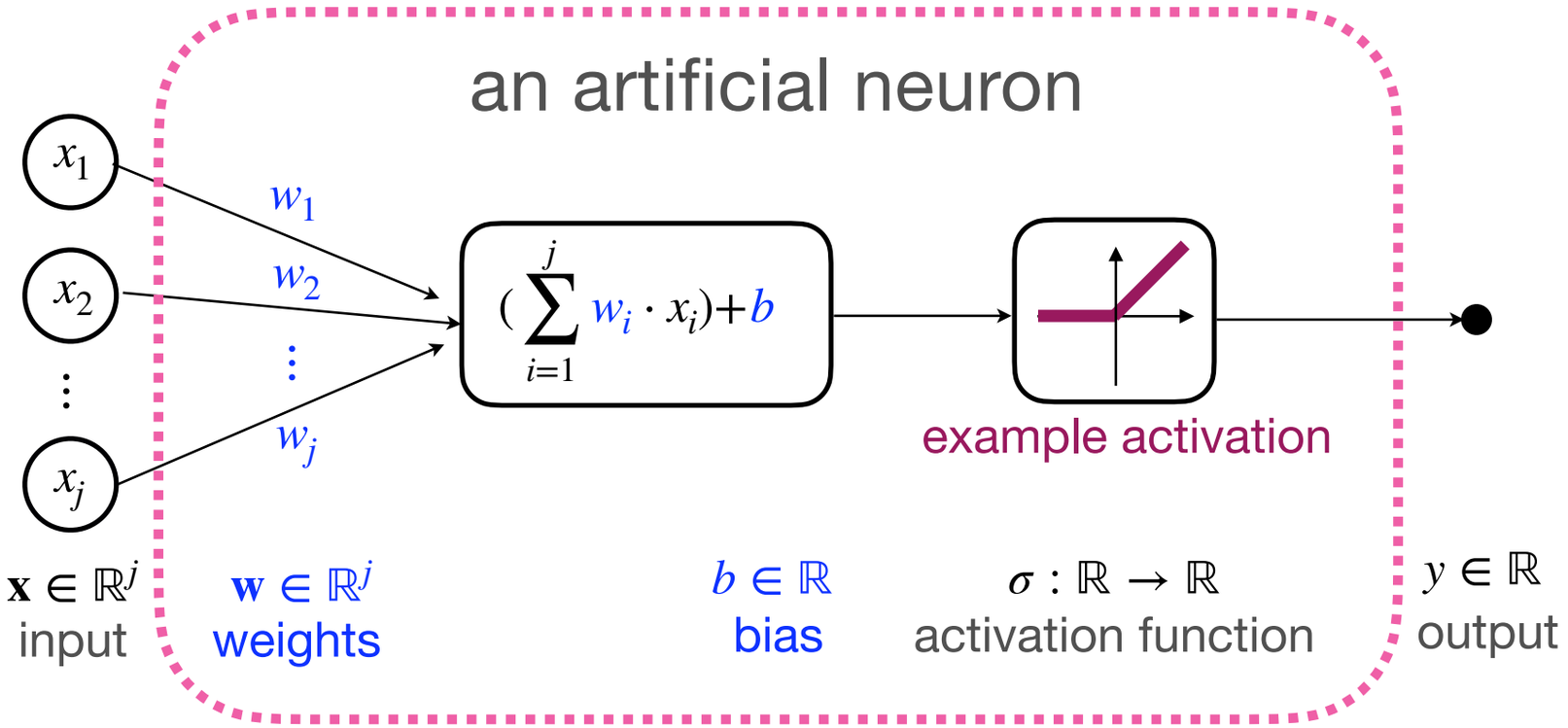
$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m$$

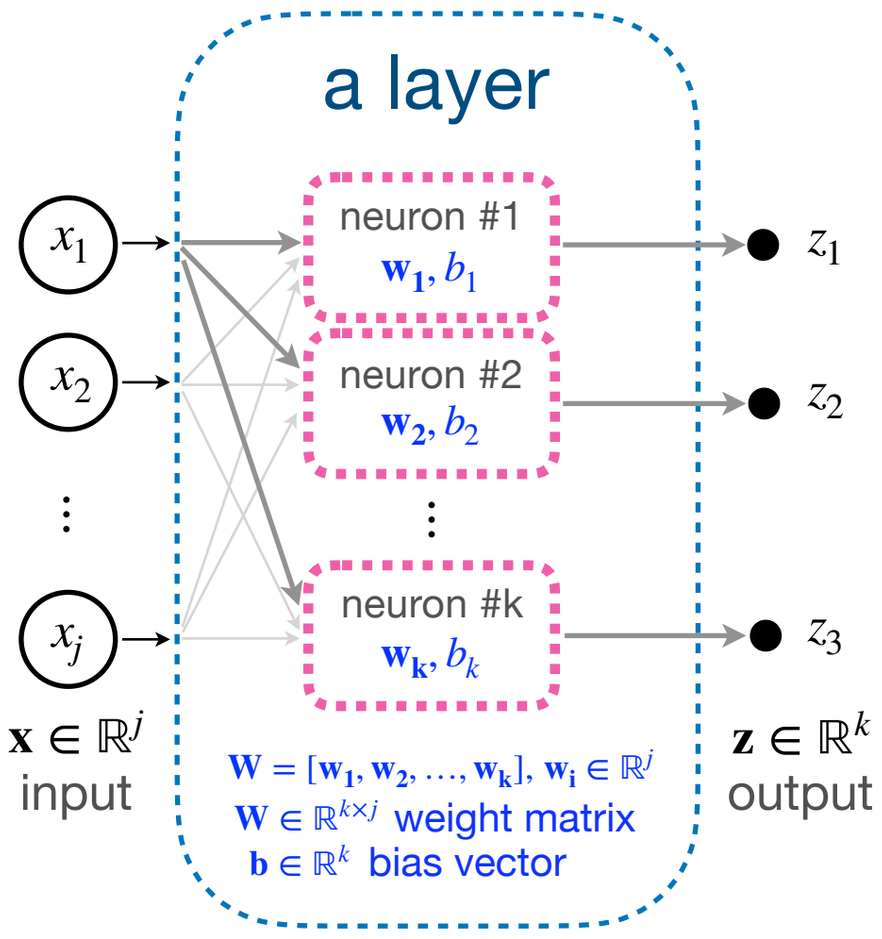
$\theta \in \mathbb{R}^M$ is a real parameter vector

with a specific structure, its building blocks are artificial neurons:



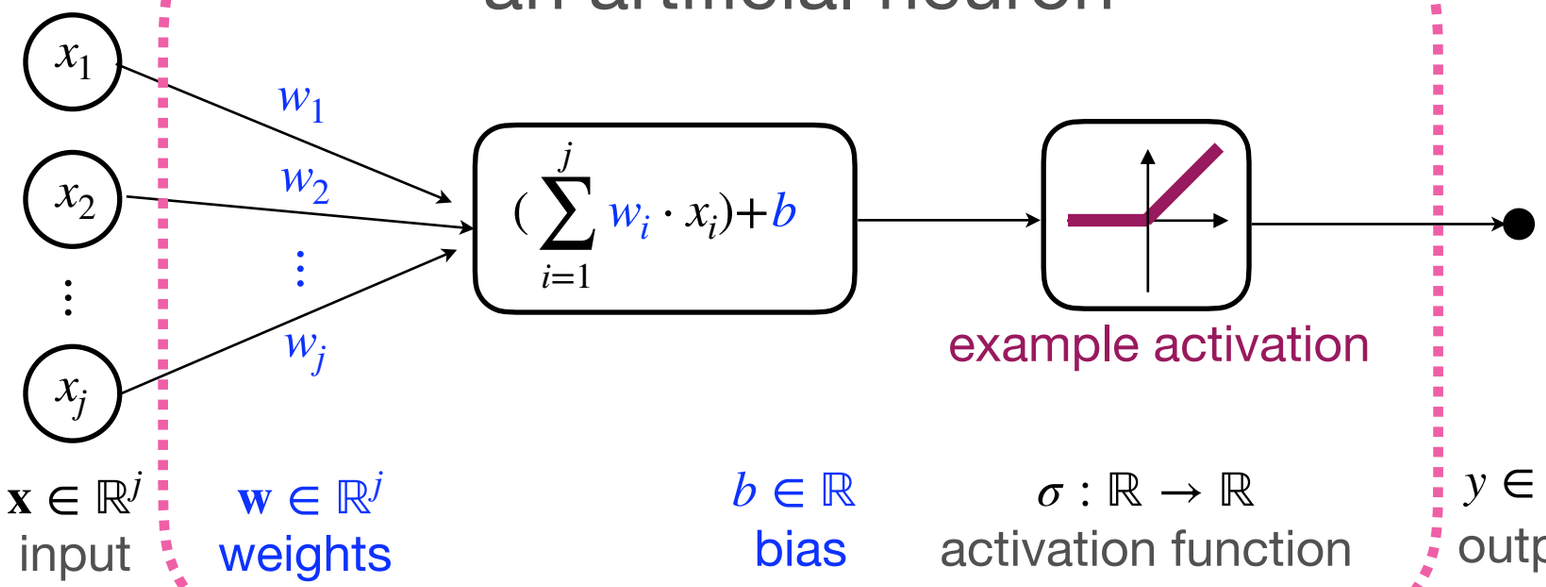
These are our elements of computation.



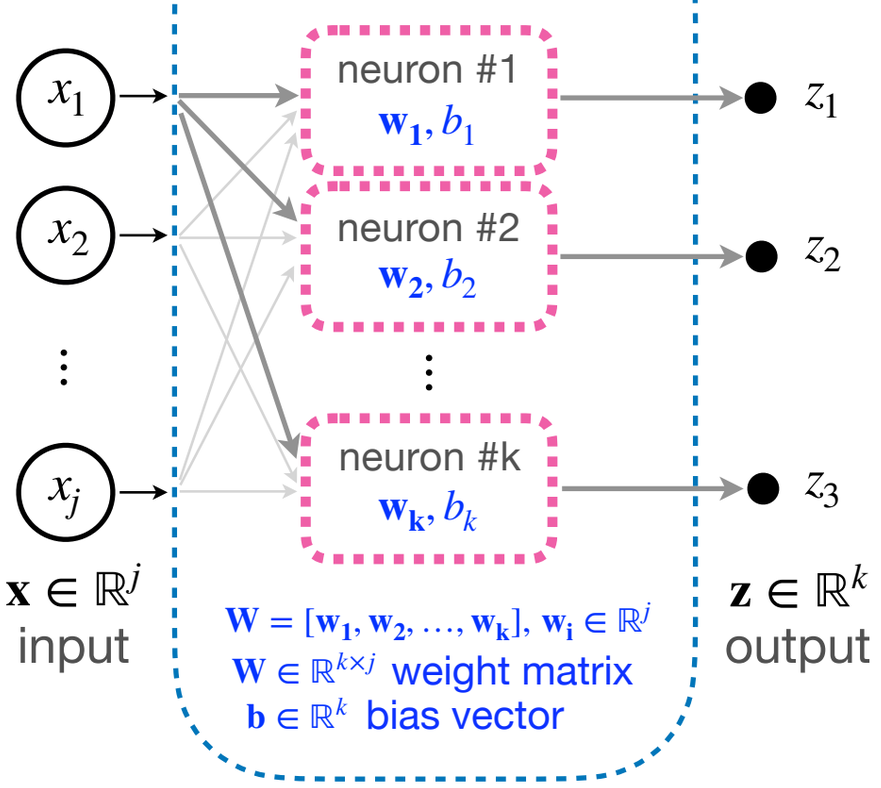


$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x}+\mathbf{b})$$

an artificial neuron

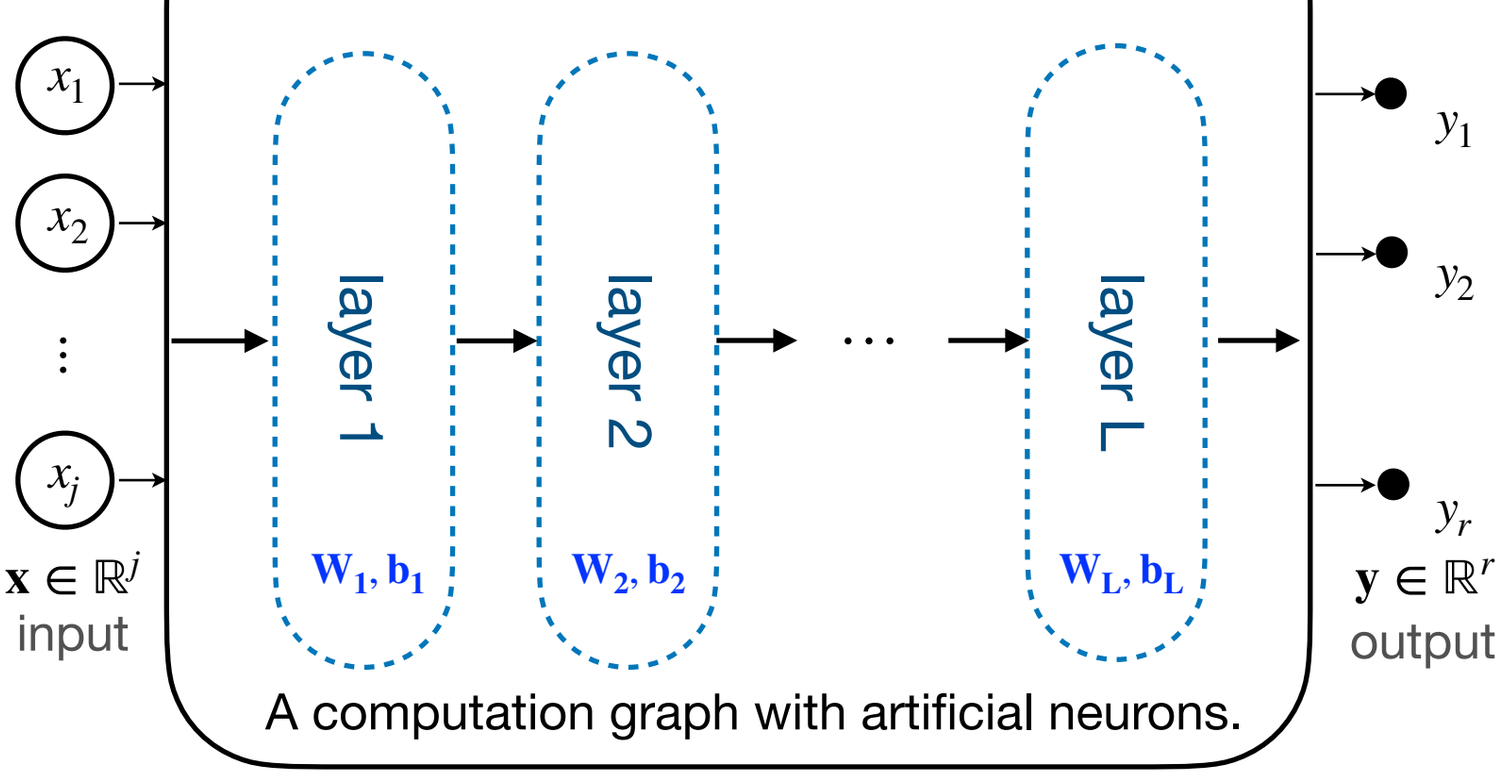


a layer

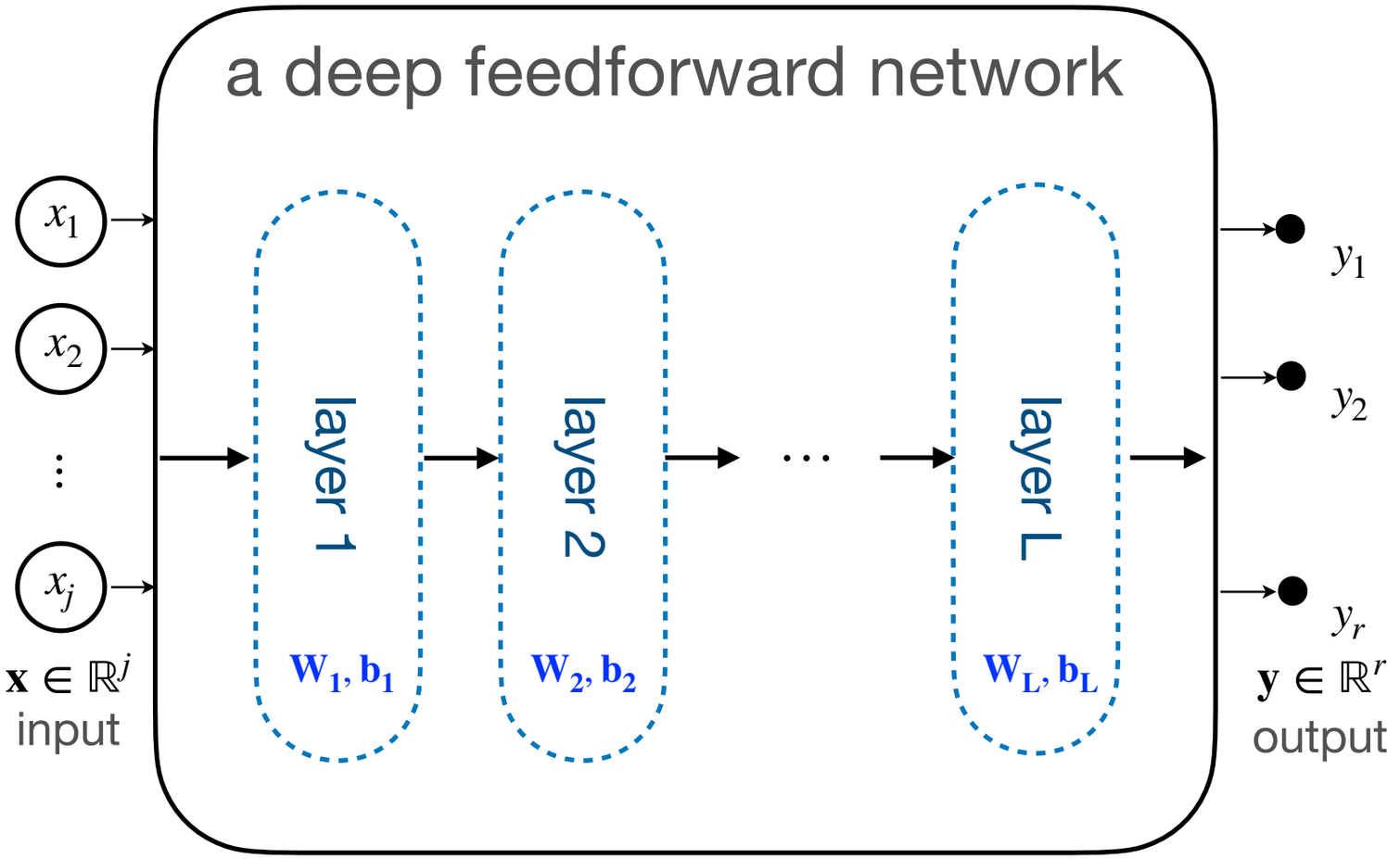


$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x}+\mathbf{b})$$

a deep feedforward network



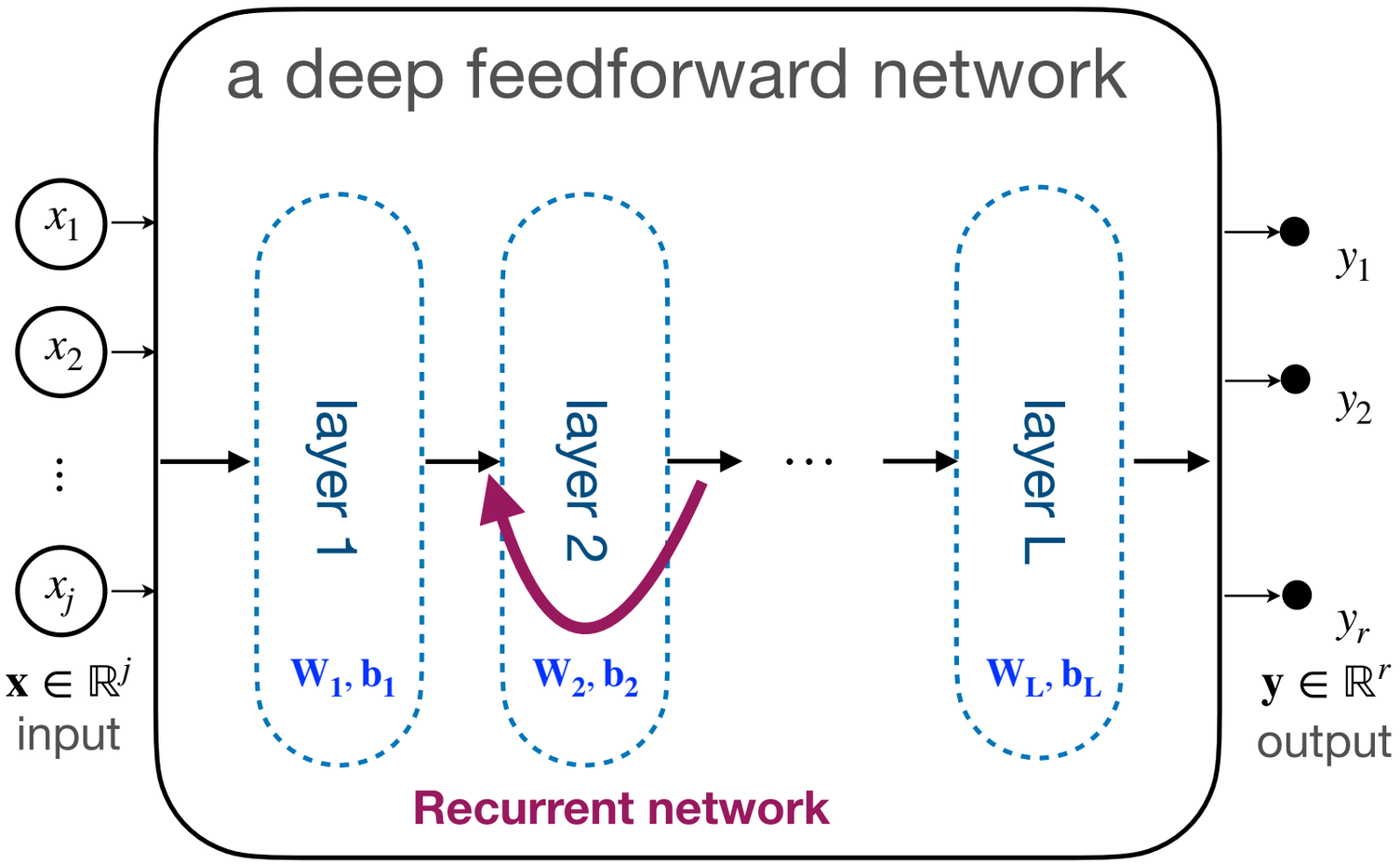
$$\mathbf{y} = \sigma(\mathbf{W_L}(\cdots\sigma(\mathbf{W_2}(\sigma(\mathbf{W_1}\mathbf{x}+\mathbf{b_1}))+\mathbf{b_2}))+\mathbf{b_L})$$



$$\mathbf{y} = \sigma(\mathbf{W_L}(\cdots\sigma(\mathbf{W_2}(\sigma(\mathbf{W_1}\mathbf{x}+\mathbf{b_1}))+\mathbf{b_2}))+\mathbf{b_L})$$

$$f_\theta(\mathbf{x}) = f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1(x)$$

$\theta = (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \cdots, \mathbf{W}_L, \mathbf{b}_L)$ are the parameters.



$$\mathbf{y} = \sigma(\mathbf{W_L}(\cdots\sigma(\mathbf{W_2}(\sigma(\mathbf{W_1}\mathbf{x}+\mathbf{b_1}))+\mathbf{b_2}))+\mathbf{b_L})$$

$$f_\theta(\mathbf{x}) = f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1(x)$$

$\theta = (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \cdots, \mathbf{W}_L, \mathbf{b}_L)$ are the parameters.

A general modelling tool.

Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

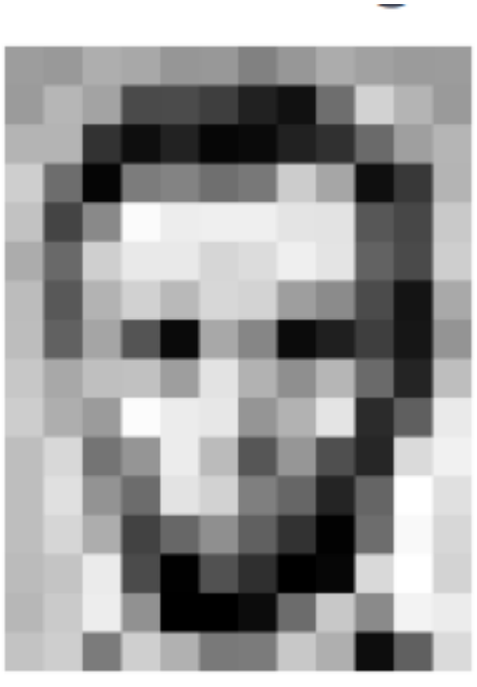
Key words. Neural networks, Approximation, Completeness.

And many more results regarding this line of research: Hornik ('91), Leshno et al. ('93), Pinkus ('99), Lu et al. ('17), Hanin and Shellke ('18), Kidger, Lyons ('20), and many more…

How do we make ‘AI’ from this?

Many things can be modelled by $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ functions.

e.g., model images by pixel intensities



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

How do we make ‘AI’ from this?

Many things can be modelled by $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^m$ functions.

e.g., model language by encoding words, letters

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L →	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U →	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
K →	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A →	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S →	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

How do artificial neural networks learn?

- We need a lots of examples of desired input-output pairs:

$$\{(x_i, y_i)\}_{i=1}^{N} \in \mathbb{R}^n \times \mathbb{R}^m$$

input desired output (supervisory signal)

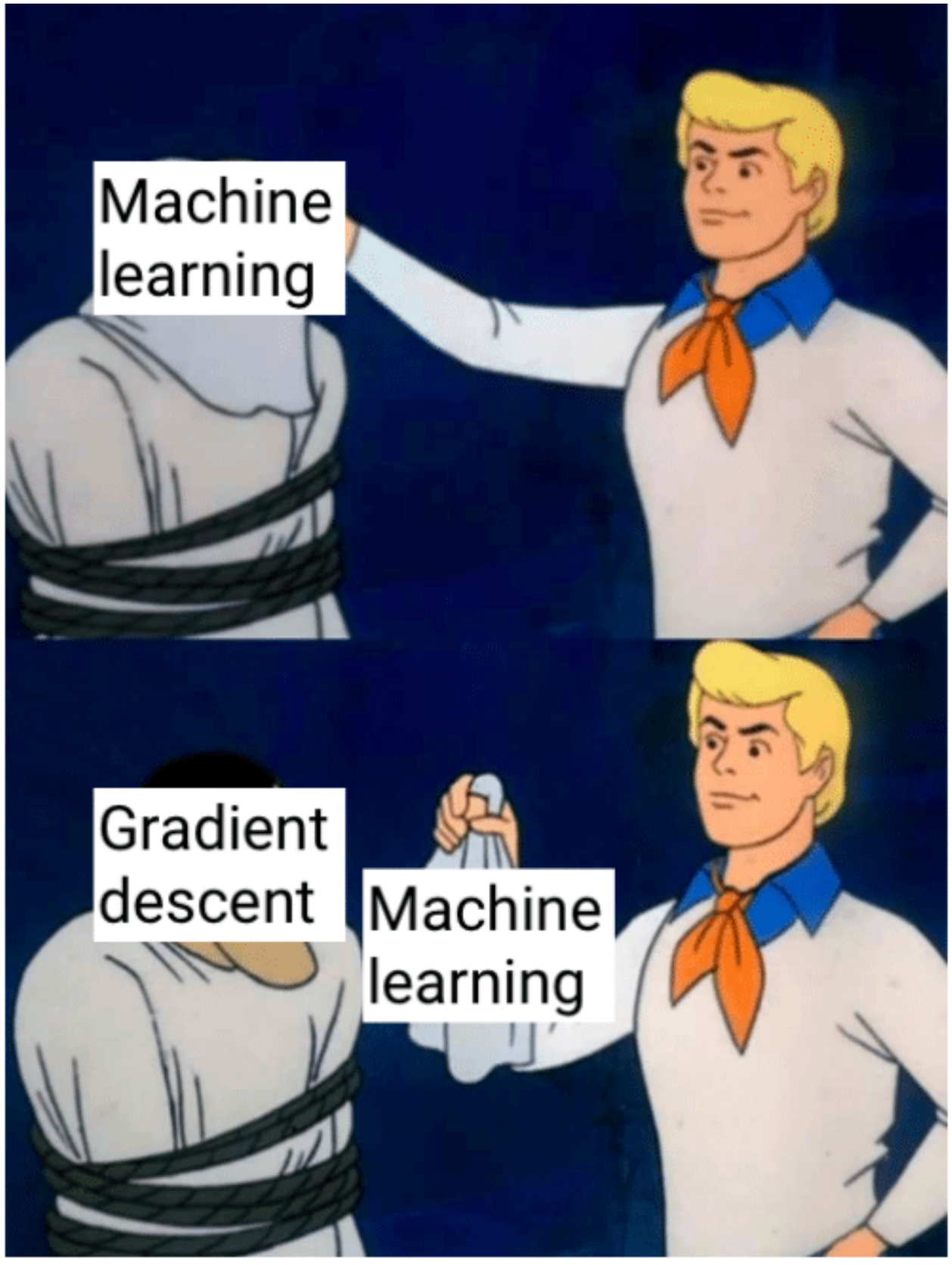
- We define a real-valued **loss function** which compares the output of the neural network with the desired output:

$$L : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$$

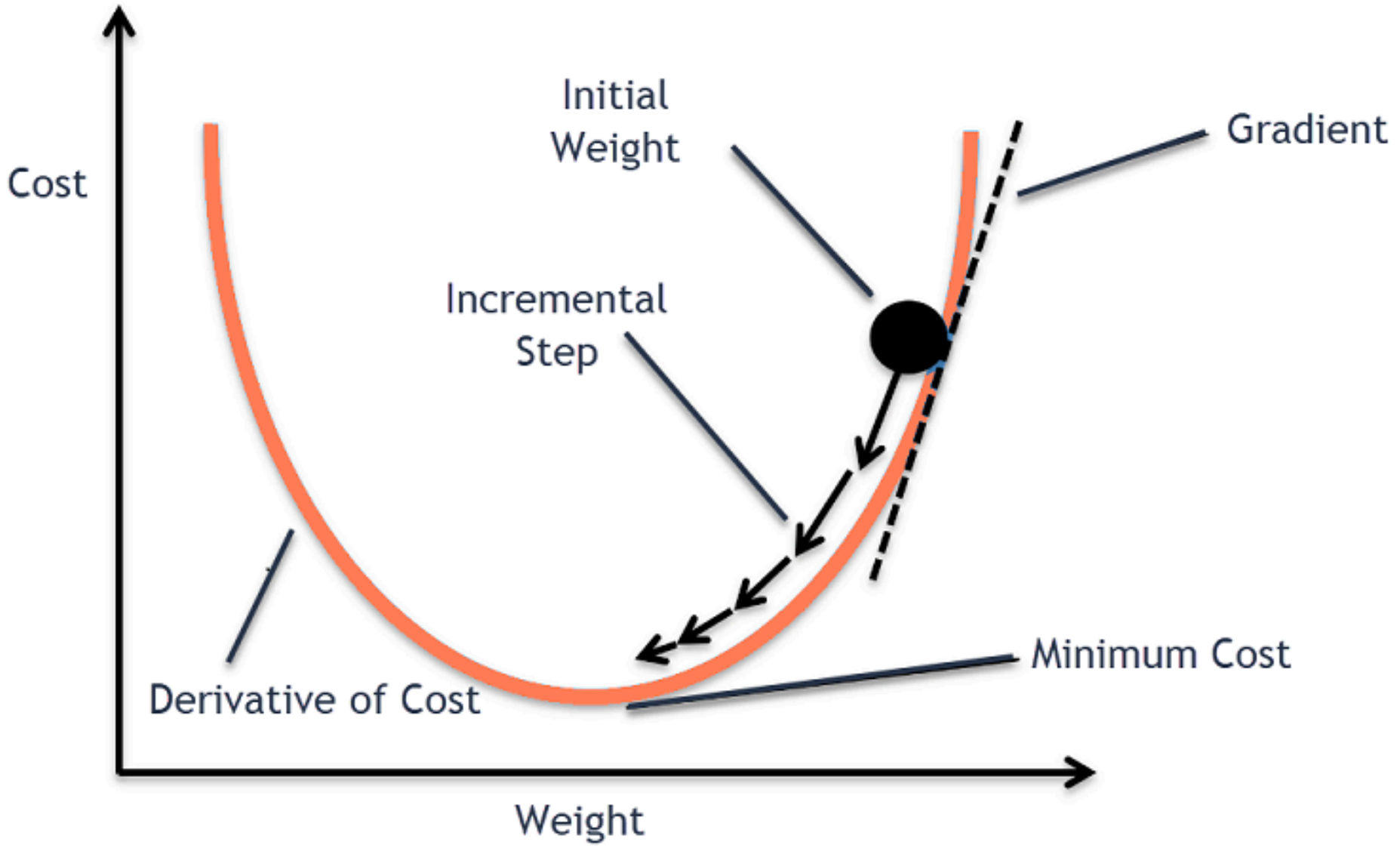
- We start from a random θ, and iteratively move it in the direction that decreases the loss, calculating $\frac{\partial L(f_\theta(x_i), y_i)}{\partial \theta}$ for a random x (or random subset of the data):

$$\theta \leftarrow \theta - \lambda \frac{\partial L(f_\theta(x_i), y_i)}{\partial \theta} \qquad (\lambda \in \mathbb{R})$$

Stochastic Gradient Descent (SGD)



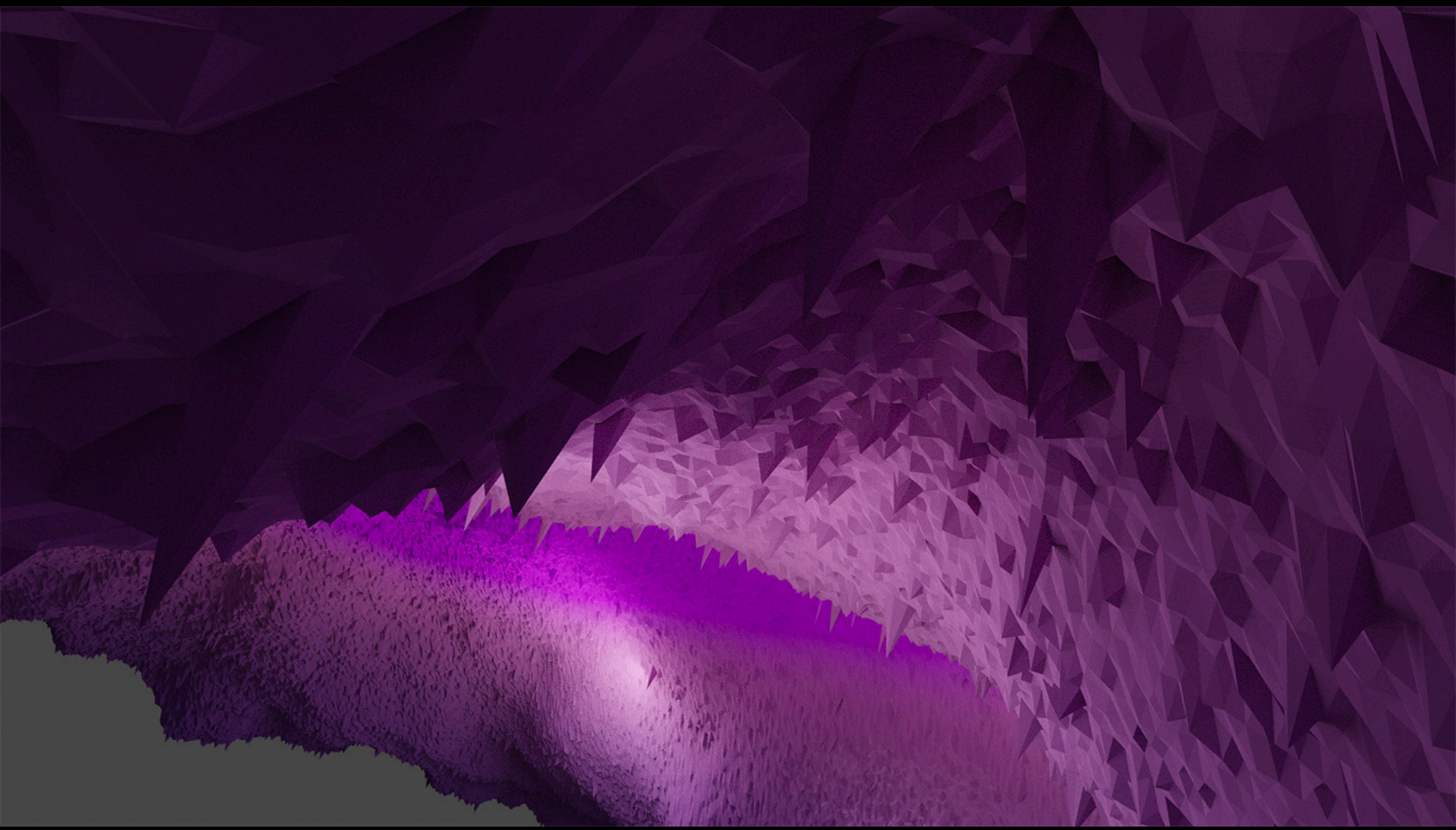
Machine learning behind the scenes


Cost
Initial
Weight
Gradient
Incremental
Step
Derivative of Cost
Minimum Cost
Weight

A gradient step in the weight space

- Based on the entire dataset: Batch Gradient Descent
- Based on one random data point: **Stochastic Gradient Descent**
- Based on a bunch of random points: **MiniBatch Gradient Descent**

Too small minibatch -> slow, noisy training
Too large minibatch -> overly aggregated training signal



MINIMA IN HIGH DIMENSIONAL SPACE

SHORTCUTS IN HIGH DIM. SPACE

RANGE IN WEIGHT SPACE

θ HIGH DIMENSIONAL SPACE

1

1 BILLION DIMENSIONS

LOSS LANDSCAPE VISUALIZATION

2

DIMENSIONALITY REDUCTION TECHNIQUE

−1,−1

α, β

$\alpha, \beta, f(\alpha, \beta)$

INITIALIZATION

SGD

GRADIENT DESCENT

LOSS: 2.54

LOSS: 1.01

LOSS: 0.02

SMOOTH MORPHOLOGY

MINIMA

MODE CONNECTIVITY arXiv: 1802.10026

MINIMA

MINIMA

LOSS: 0.01

MINIMA

LOSS: 0.02

ROUGH MORPHOLOGY

MINIMA

+1, −1

α, β

−1, 0

α, β

η

$\alpha, \beta, f(\alpha, \beta)$

3

θ 2D SLICE

RAND ORTHO DIRECTIONS arXiv:1712.09913

LOSS: 0.02

MINIMA

$\alpha, \beta, f(\alpha, \beta)$

$\alpha, \beta, f(\alpha, \beta)$

LOSS: 2.47

LOW DIMENSIONALITY INTERPRETATION

4

3 DIMENSIONS

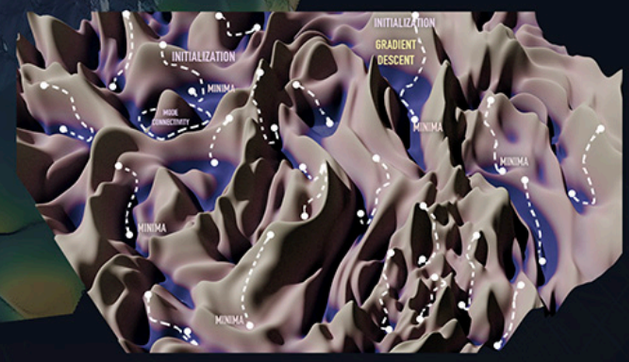
$$f(\alpha, \beta) = L(\theta + \alpha\delta + \beta\eta)$$

δ

MINIMA

5

THE BLESSING OF DIMENSIONALITY

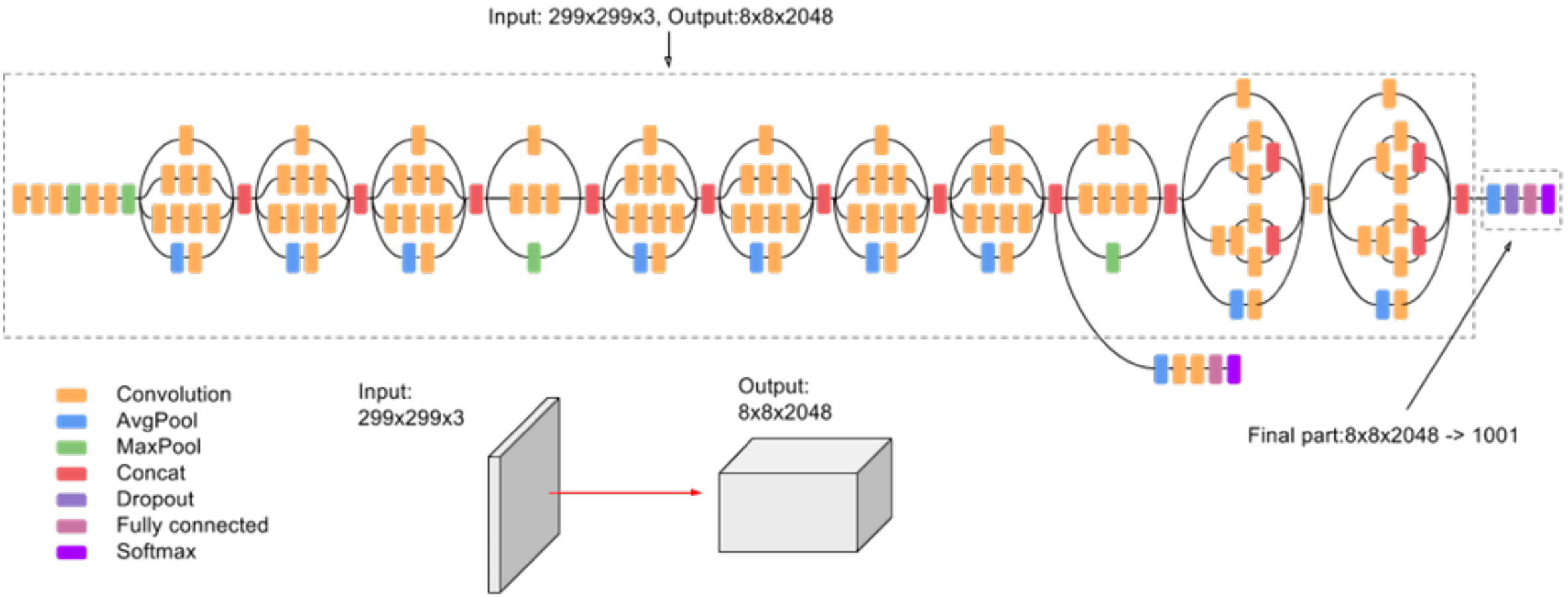


FINDING A MINIMA BECOMES A “LOCAL” CHALLENGE

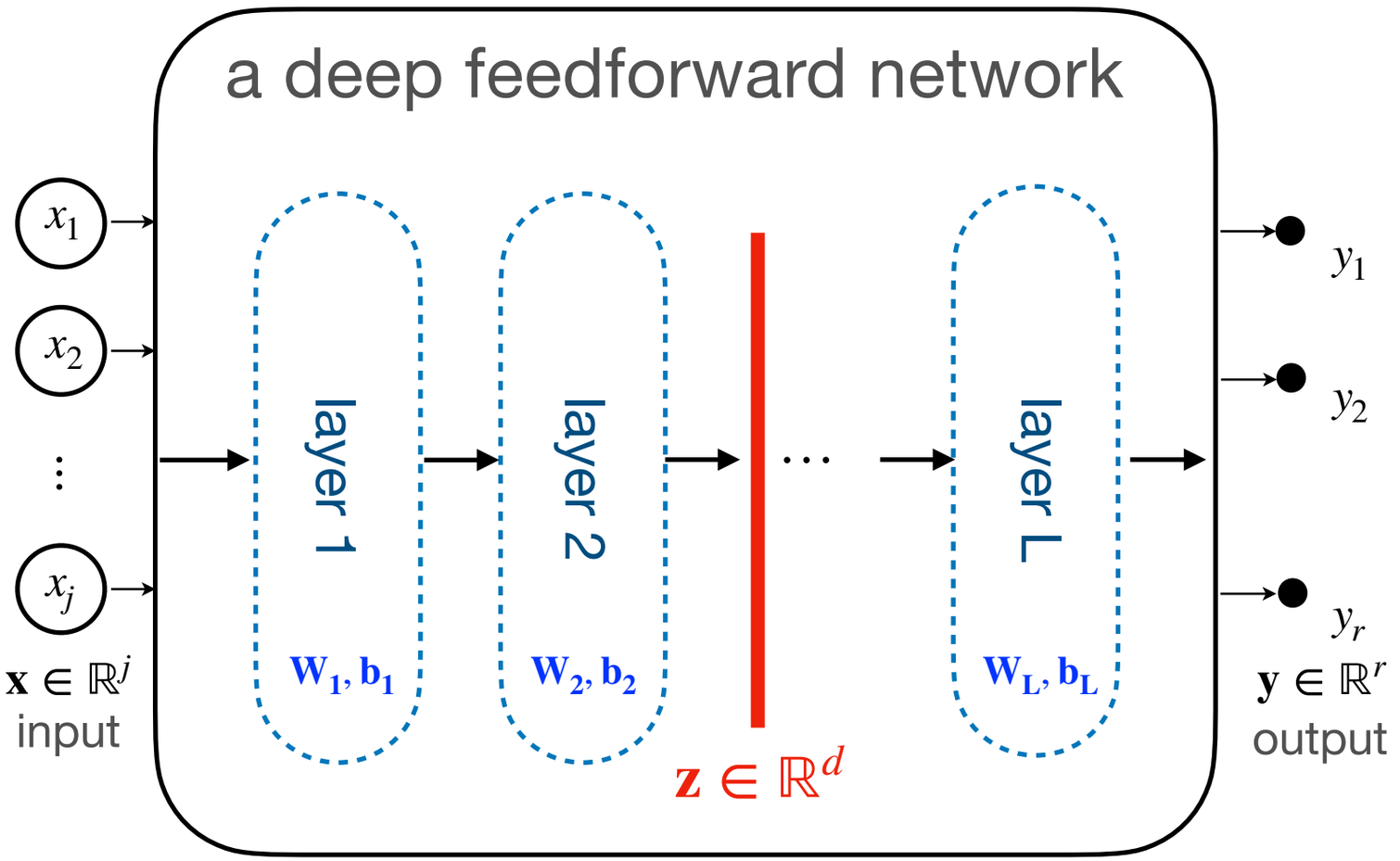
VISUALIZATION BY JAVIER IDEAMI / LOSSLANDSCAPE.COM

The InceptionV3 network

(Szegedy et al., 2016)



Parameter count: 24M



$$\mathbf{y} = \sigma(\mathbf{W_L}(\cdots\sigma(\mathbf{W_2}(\sigma(\mathbf{W_1}\mathbf{x}+\mathbf{b_1}))+\mathbf{b_2}))+\mathbf{b_L})$$

$$f_\theta(\mathbf{x}) = f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1(x)$$

$\theta = (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \cdots, \mathbf{W}_L, \mathbf{b}_L)$ are the parameters.

Inner representations: partial results of this function composition

e.g., $f_{inner2}(x) : \mathbb{R}^n \to \mathbb{R}^d$

$$f_{inner2}(x) = f_2 \circ f_1(x)$$

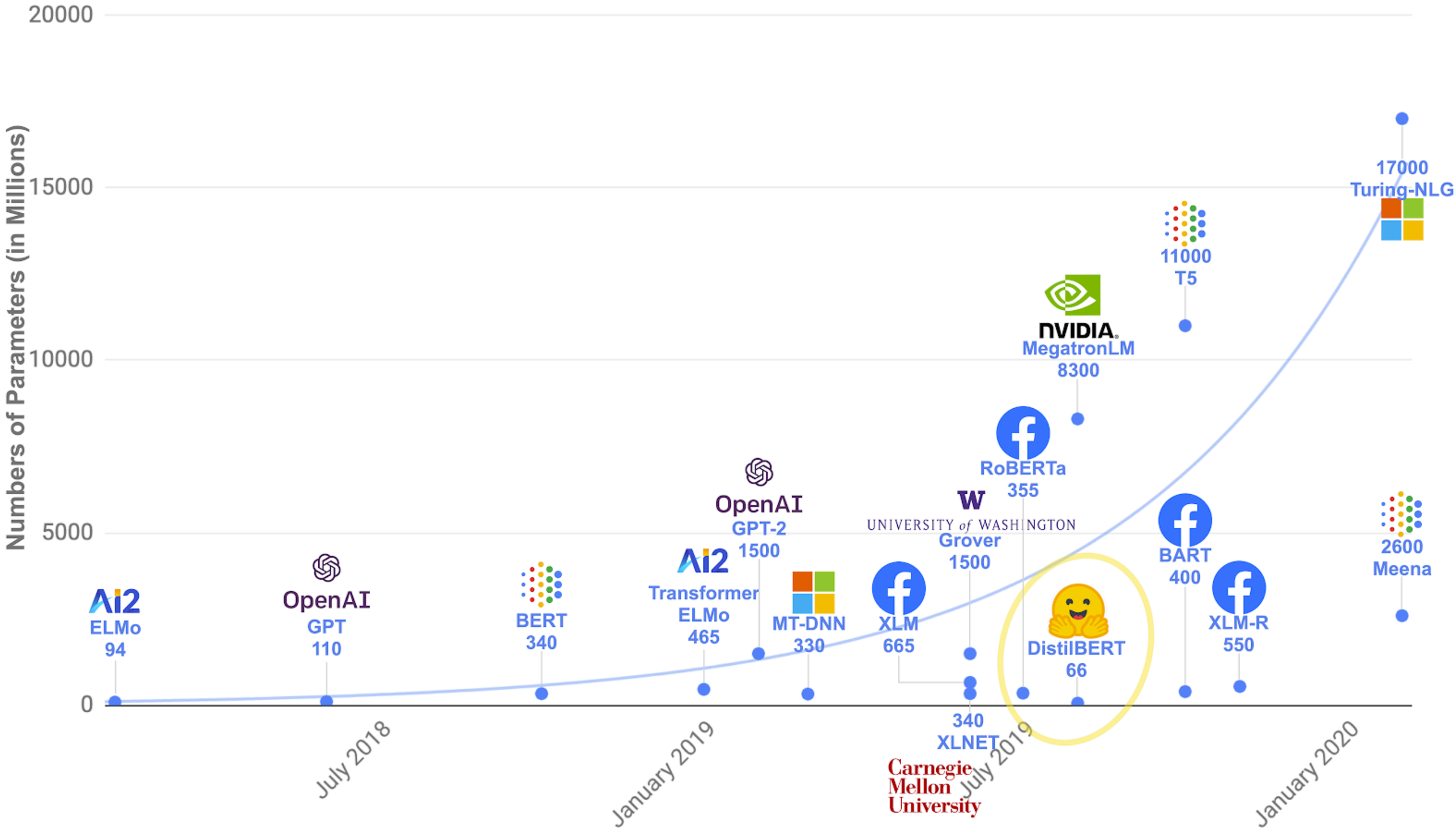
Lots of data,
lots of compute

- CLIP was trained on 400 million (image, text) pairs.

- 32 epochs. (Each of the 400M pairs was shown this many times to the network during training.)

- Training took 18 days on 592 NVIDIA V100 GPUs.

- That amount of computation would cost a few 100 thousand dollars on the market. (OpenAI can thank Microsoft, though.)


Numbers of Parameters (in Millions)
20000
15000
10000
5000
0
ELMo
94
GPT
110
BERT
340
Transformer
ELMo
465
GPT-2
1500
MT-DNN
330
XLM
665
Grover
1500
340
XLNET
RoBERTa
355
11000
T5
DistilBERT
66
BART
400
XLM-R
550
2600
Meena
17000
Turing-NLG
MegatronLM
8300
July 2018
January 2019
July 2019
January 2020

Summarising the magic of deep learning

- We have systems that are easy to define (matrix-vector products, vector addition, max operation).
- Scaled them to enormous sizes (millions or billions of parameters).
- Collected a lot of data (e.g., entire Wikipedia, and HUGE internet crawls with text and images).
- Give these systems these highly non-linear optimisation problems.
- Over a massively overparametrized class of functions.
- All to be solved with the dead simple algorithm of Stochastic Gradient Descent.
- **Still, the “Star Wars” neuron emerges.**

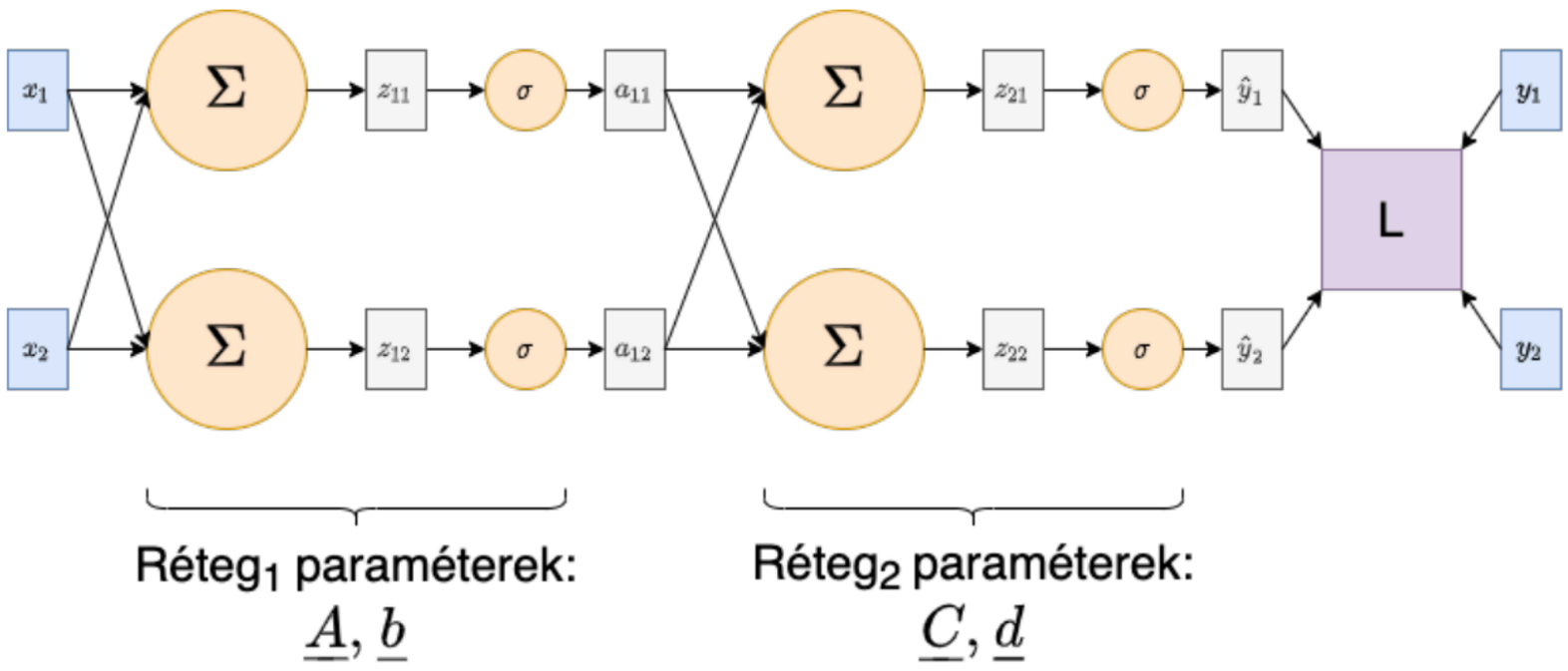
Backpropagation algorithm

Backpropagation algorithm

Chain rule done efficiently

$$(f \circ g)'(x) = f'(g(c)) \cdot g'(x)$$

$$(f \circ g \circ h)'(x) = f'(g(h(x))) \cdot g'(h(x)) \cdot h'(x)$$



Backpropagation algorithm

$$\underline{z_1} = \boldsymbol{A}\underline{x} + \underline{b}$$

$$\underline{a_1} = \sigma(\underline{z_1})$$

$$\underline{z_2} = \boldsymbol{C}\underline{a_1} + \underline{d}$$

$$\underline{a_2} = \sigma(\underline{z_2})$$

$$\underline{\hat{y}} \equiv \underline{a_2}$$

$$L = \frac{1}{2}\sum(\underline{\hat{y}} - \underline{y})^2$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = -\frac{1}{(1+e^{-x})^2} \cdot e^{-x} \cdot (-1)$$

$$= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}}$$

$$= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}}$$

$$= \sigma(x) \cdot (1-\sigma(x))$$

Keressük: $\frac{\partial L}{\partial \boldsymbol{A}}$, $\frac{\partial L}{\partial \underline{b}}$, $\frac{\partial L}{\partial \boldsymbol{C}}$, $\frac{\partial L}{\partial \underline{d}}$

Backpropagation algorithm

$$\underline{z_1} = \boldsymbol{A}\underline{x} + \underline{b}$$
$$\underline{a_1} = \sigma(\underline{z_1})$$
$$\underline{z_2} = \boldsymbol{C}\underline{a_1} + \underline{d}$$
$$\underline{a_2} = \sigma(\underline{z_2})$$
$$\underline{\hat{y}} \equiv \underline{a_2}$$
$$L = \frac{1}{2}\sum(\underline{\hat{y}} - \underline{y})^2$$

Keressük: $\frac{\partial L}{\partial \boldsymbol{A}}$, $\frac{\partial L}{\partial \underline{b}}$, $\frac{\partial L}{\partial \boldsymbol{C}}$, $\frac{\partial L}{\partial \underline{d}}$

A gradienseket a kimenettől visszafelé számítjuk, a láncszabállyal.

$$\frac{\partial L}{\partial \boldsymbol{C}} = \frac{\partial L}{\partial \underline{a_2}} \cdot \frac{\partial \underline{a_2}}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \boldsymbol{C}}$$
$$= (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_2})(\underline{1} - \sigma(\underline{z_2})) \cdot \underline{a_1}^T$$
$$\frac{\partial L}{\partial \underline{d}} = \frac{\partial L}{\partial \underline{a_2}} \cdot \frac{\partial \underline{a_2}}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \underline{d}}$$
$$= (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_2})(\underline{1} - \sigma(\underline{z_2})) \cdot \underline{1}$$

Visszaterjesztett hiba:

$$\underline{\delta_2} \equiv \frac{\partial L}{\partial \underline{z_2}} = (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_2})(\underline{1} - \sigma(\underline{z_2}))$$

Backpropagation algorithm

$$\underline{z_1} = \boldsymbol{A}\underline{x} + \underline{b}$$
$$\underline{a_1} = \sigma(\underline{z_1})$$
$$\underline{z_2} = \boldsymbol{C}\underline{a_1} + \underline{d}$$
$$\underline{a_2} = \sigma(\underline{z_2})$$
$$\underline{\hat{y}} \equiv \underline{a_2}$$
$$L = \frac{1}{2}\sum(\underline{\hat{y}} - \underline{y})^2$$

Keressük: $\frac{\partial L}{\partial \boldsymbol{A}}$, $\frac{\partial L}{\partial \underline{b}}$, $\frac{\partial L}{\partial \boldsymbol{C}}$, $\frac{\partial L}{\partial \underline{d}}$

Kiszámoljuk a rejtett réteghez tartozó gradienseket.

$$\frac{\partial L}{\partial \boldsymbol{A}} = \frac{\partial L}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \underline{a_1}} \cdot \frac{\partial \underline{a_1}}{\partial \underline{z_1}} \cdot \frac{\partial \underline{z_1}}{\partial \boldsymbol{A}}$$
$$= \boldsymbol{C}^T \cdot \underline{\delta_2} \odot \sigma(\underline{z_1})(\underline{1} - \sigma(\underline{z_1})) \cdot \underline{x}^T$$
$$\frac{\partial L}{\partial \underline{b}} = \frac{\partial L}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \underline{a_1}} \cdot \frac{\partial \underline{a_1}}{\partial \underline{z_1}} \cdot \frac{\partial \underline{z_1}}{\partial \underline{b}}$$
$$= \boldsymbol{C}^T \cdot \underline{\delta_2} \odot \sigma(\underline{z_1})(\underline{1} - \sigma(\underline{z_1})) \cdot \underline{1}$$

Visszaterjesztett hiba:

$$\underline{\delta_1} \equiv \frac{\partial L}{\partial \underline{z_1}} = \boldsymbol{C}^T \cdot \underline{\delta_2} \odot \sigma(\underline{z_1})(\underline{1} - \sigma(\underline{z_1}))$$

Backpropagation algorithm

1. A kimenettől visszafelé haladva kiszámoljuk a $\underline{\delta_i}$ visszaterjesztett hibatagokat.

$$\underline{\delta_n} \equiv \frac{\partial L}{\partial \underline{z_n}} = (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_n})(\underline{1} - \sigma(\underline{z_n}))$$

$$\underline{\delta_{k-1}} \equiv \frac{\partial L}{\partial \underline{z_{k-1}}} = \boldsymbol{W_k}^T \cdot \underline{\delta_k} \odot \sigma(\underline{z_{k-1}})(\underline{1} - \sigma(\underline{z_{k-1}}))$$

2. $\underline{\delta_k}$ alapján kiszámoljuk a paraméterek szerinti gradienst

$$\frac{\partial L}{\partial \boldsymbol{W_k}} = \underline{\delta_k} \odot \underline{a_{k-1}}^T$$

$$\frac{\partial L}{\partial \underline{b_k}} = \underline{\delta_k} \odot \underline{1}$$

Egyetlen tanulási lépés

- **Előreterjesztés**
  Rétegen belüli számítás párhuzamosítható
  Rétegek számában lineáris
- **Visszaterjesztés, súlymodosítás**
  Naívan négyzetes idő
  Visszaterjesztett hiba segítségével lineáris

Mára mindezt ajándékba kapjuk a modern tenzor könyvtáraktól.

A tanulás folyamatának finomhangolása

Rengeteg optimalizációs módszer a tanulás irányítására

- Tanulási ráta
- Paraméterenként külön-külön tanulási ráta.
- Momentum módszer: megjegyezzük, hogy a múltban merre mozdultak a paraméterek.
- Múltbeli gradiensek második momentumát is figyelembe vehetjük.

Manapság a leggyakrabban használt módszerek:

- Nesterov momentum
- Adaptive Gradient Algorithm (AdaGrad)
- Root Mean Square Propagation (RMSProp)
- Adam

A tanulás folyamatának finomhangolása

Rengeteg optimalizációs módszer a tanulás irányítására

- Tanulási ráta
- Paraméterenként külön-külön tanulási ráta.
- Momentum módszer: megjegyezzük, hogy a múltban merre mozdultak a paraméterek.
- Múltbeli gradiensek második momentumát is figyelembe vehetjük.

Manapság a leggyakrabban használt módszerek:

- Nesterov momentum
- Adaptive Gradient Algorithm (AdaGrad)
- Root Mean Square Propagation (RMSProp)
- Adam

Ezek részletesen a 4. előadáson

Az adathalmaz szétbontása

- Tanító adatok: a hálózat paramétereinek beállítása.
- Validációs adatok: a hálózat és a tanulási folyamat hiperparamétereinek beállítása.
- Teszt adatok: végső kiértékelés.

A tanulás kiértékelése

1. Tanulási hiba: mennyire illeszkedik a modell a tanító pontokra?
    - Egy tipikus neurális hálózat erősen túlparametrizált és nagyon sok féle módon tud jól illeszkedni a tanító pontokra.
2. Általánosítási hiba: mennyire jól általánosodik a modell tanítás során nem látott adatokra?
    - Regularizáció

Regularizáció

- Aktívan kutatott terület.
- Minden olyan módszer, mely megakadályozza a túltanulást.

1. Architekturális változtatások (Dropout . . .)
2. A tanulás folyamatának módosítása (Korai Leállás . . .)
3. Adatok módosítása (Augmentáció . . .)
4. Új tag hozzávétele a hibafüggvényhez (Weight Decay . . .)