Foundations of DL

Deep Learning

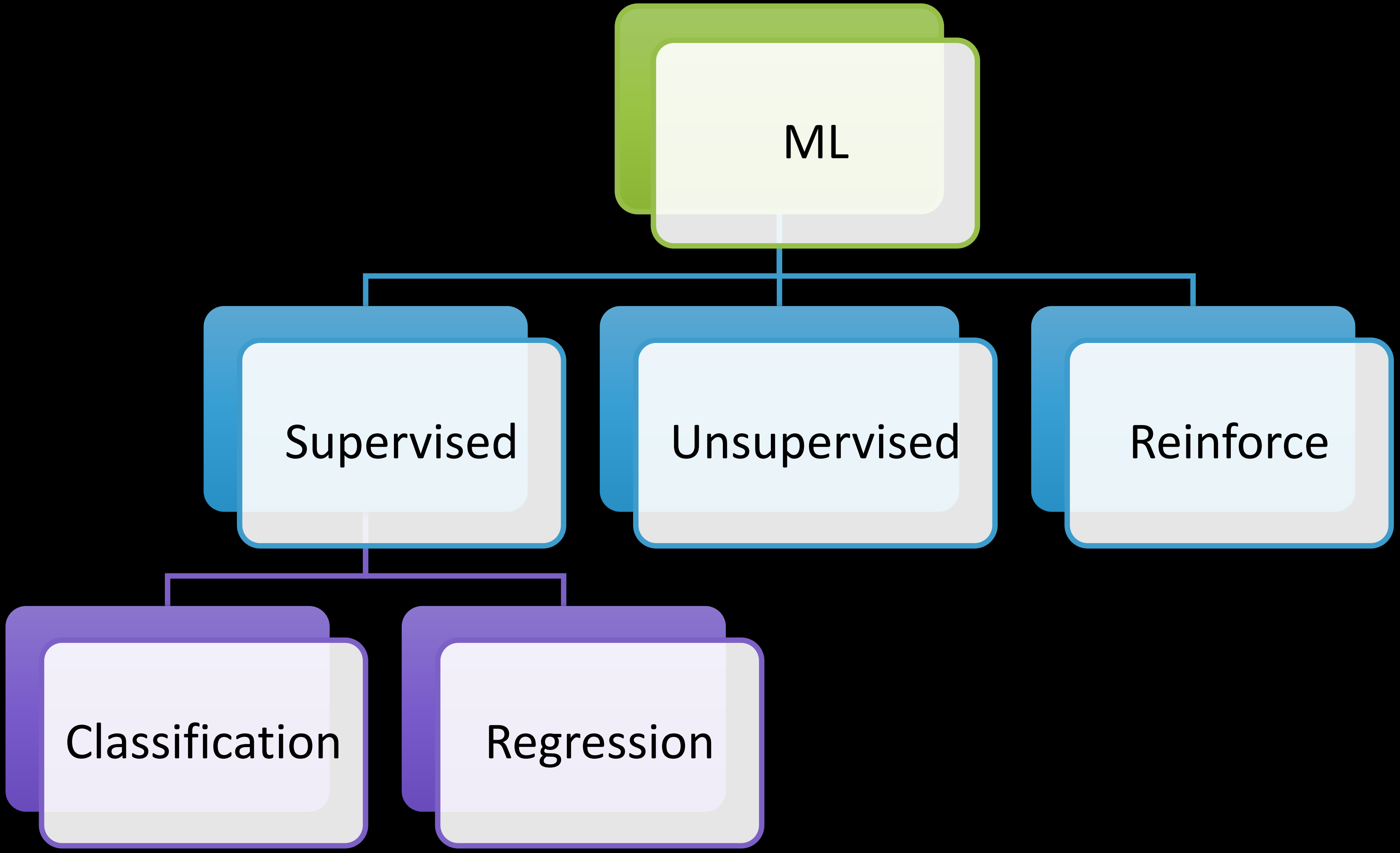
ALF

Alfredo Canziani, Ritchie Ng
@alfcnz, @RitchieNg

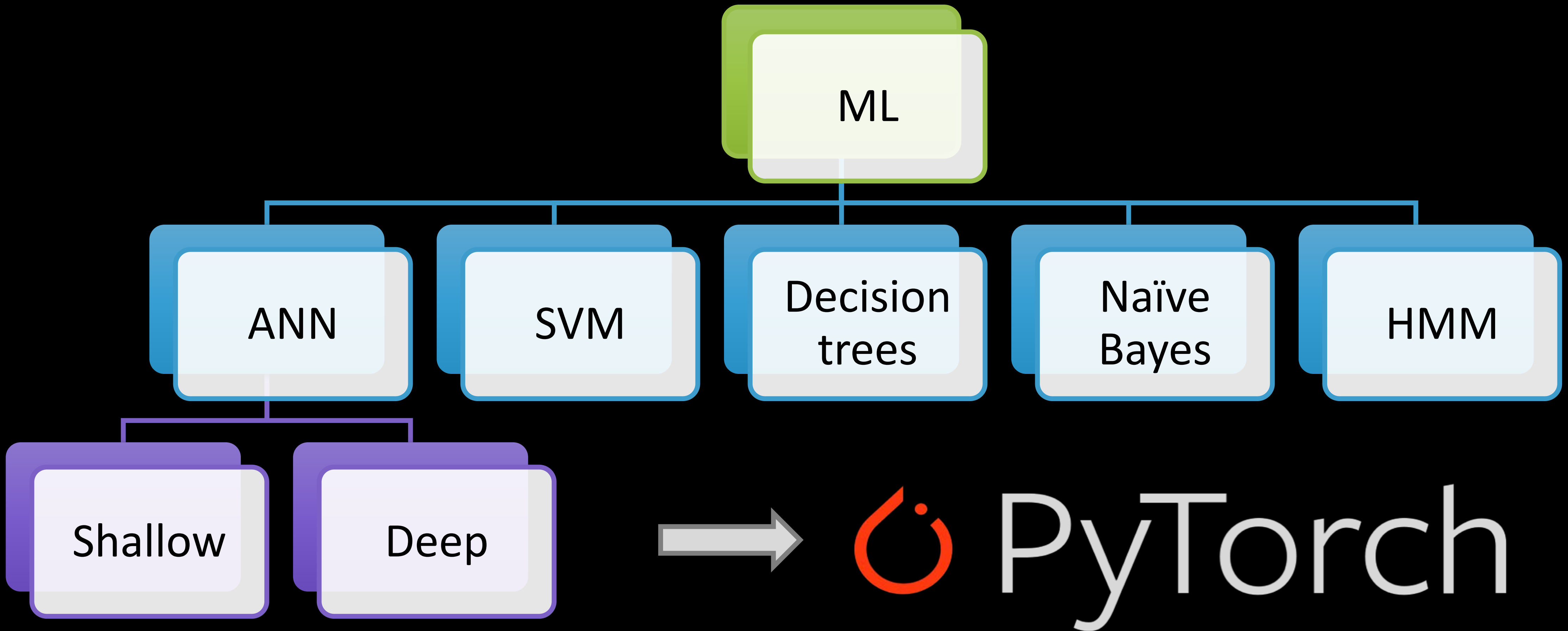
Taxonomy of ML

Learning paradigms and algorithms

Machine Learning (ML)

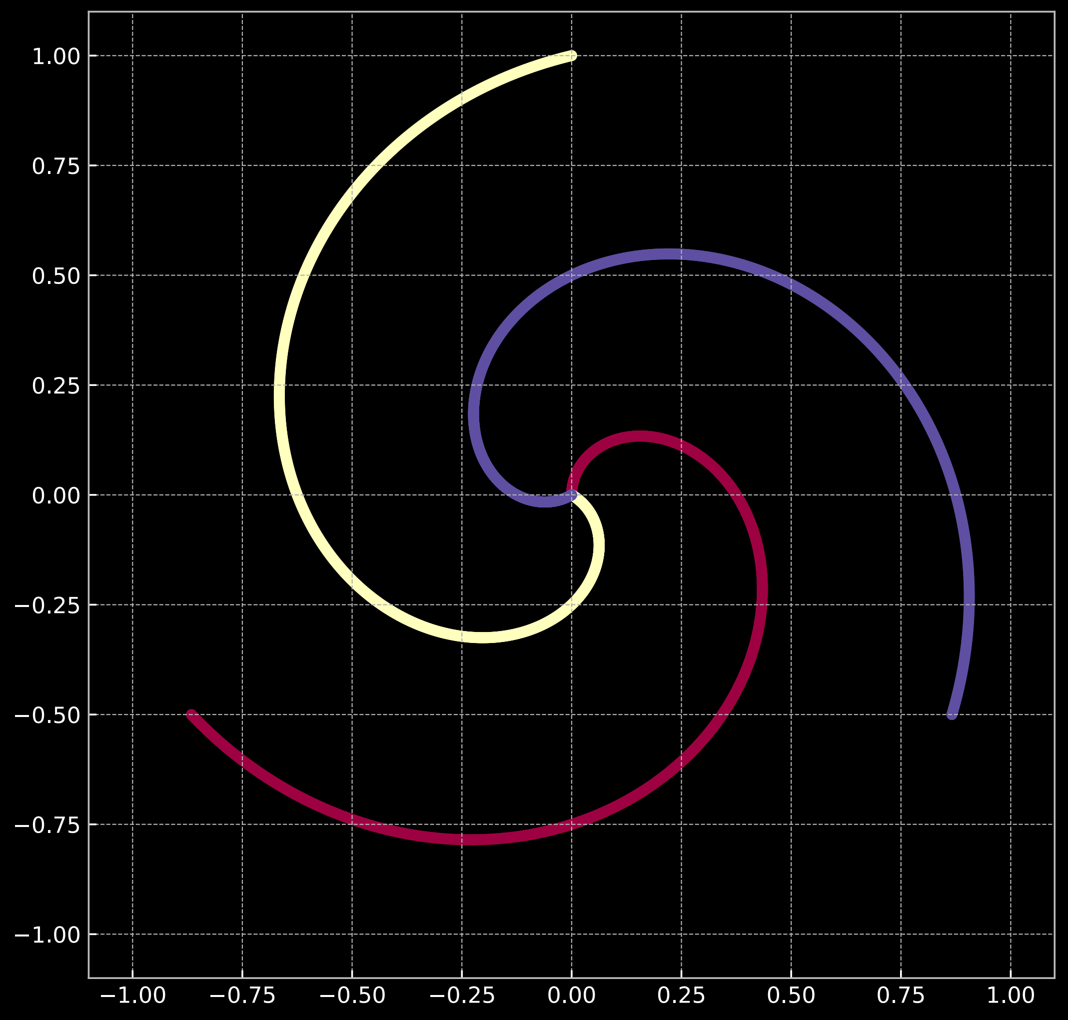


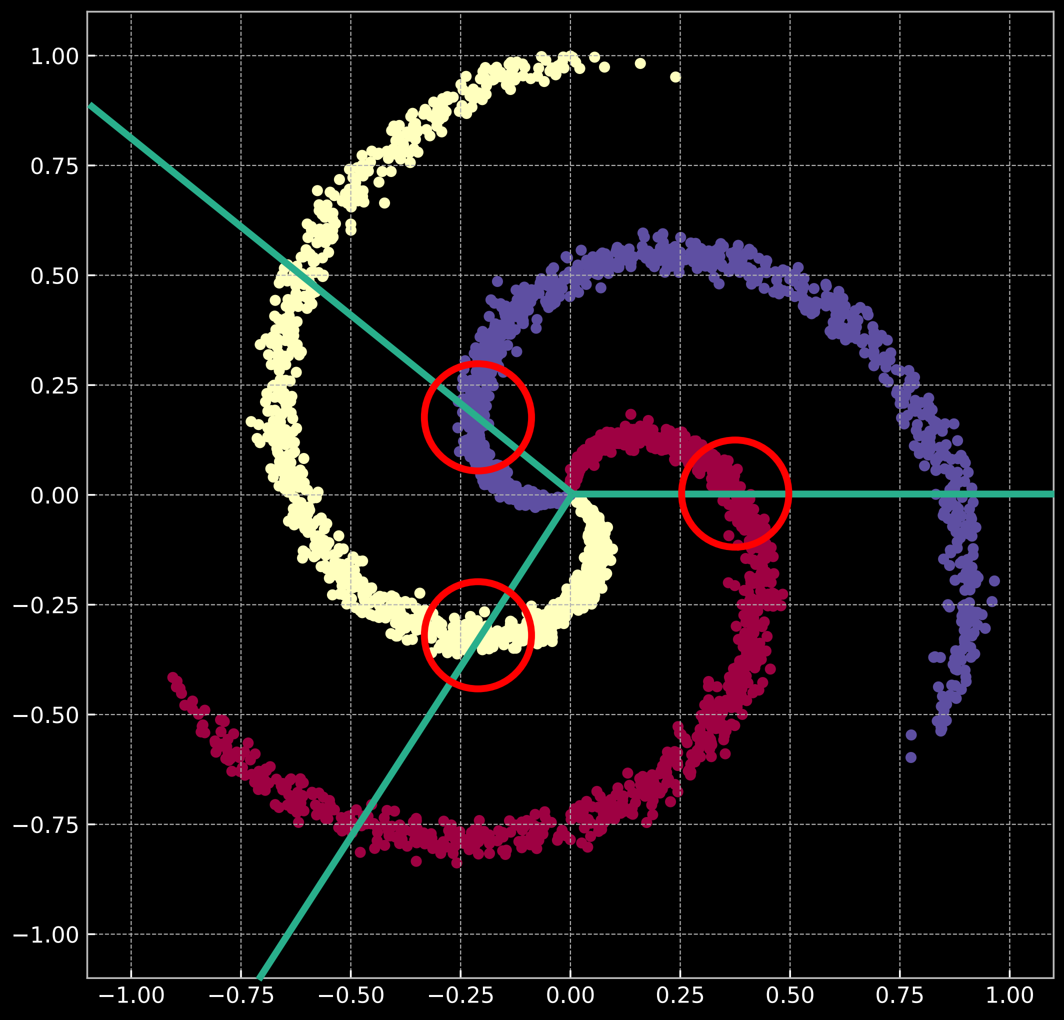
ML algorithms



ANN – supervised learning

Classification



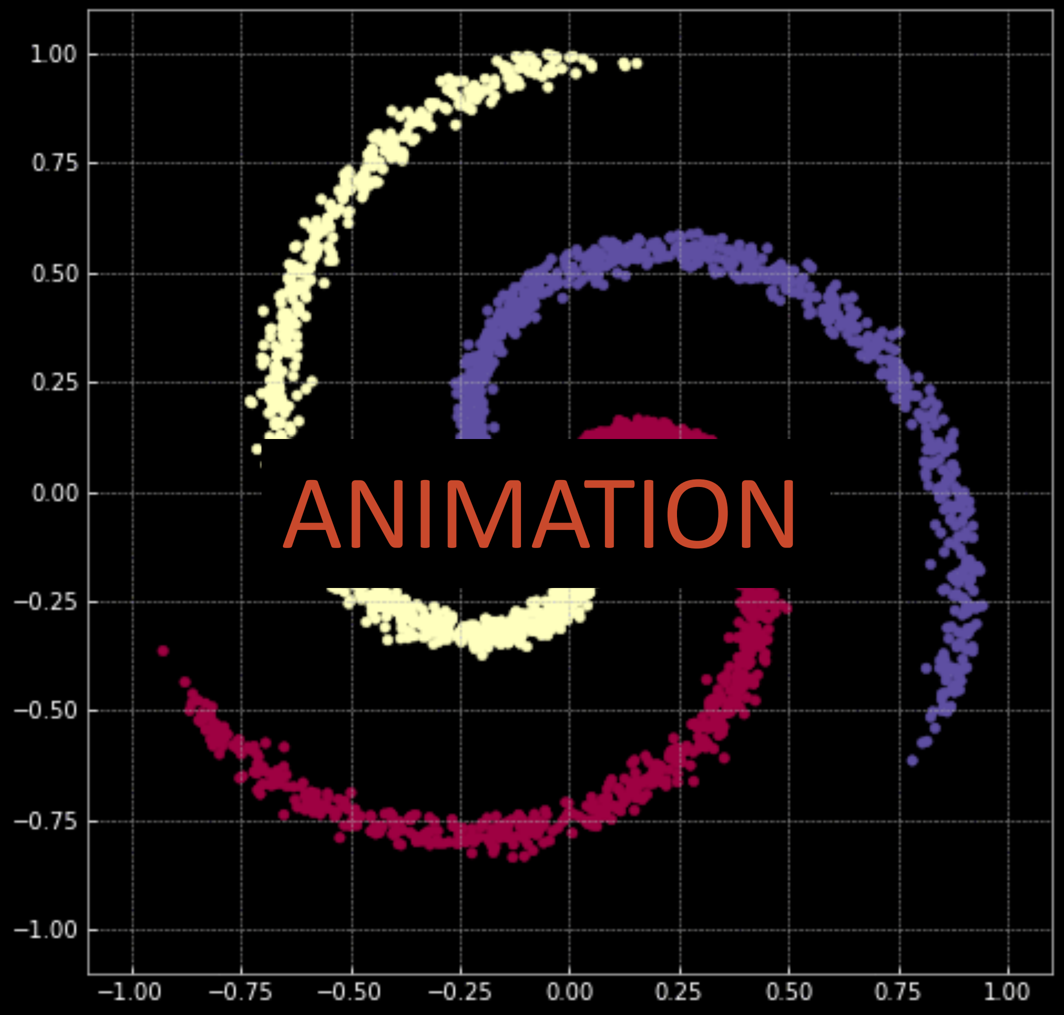


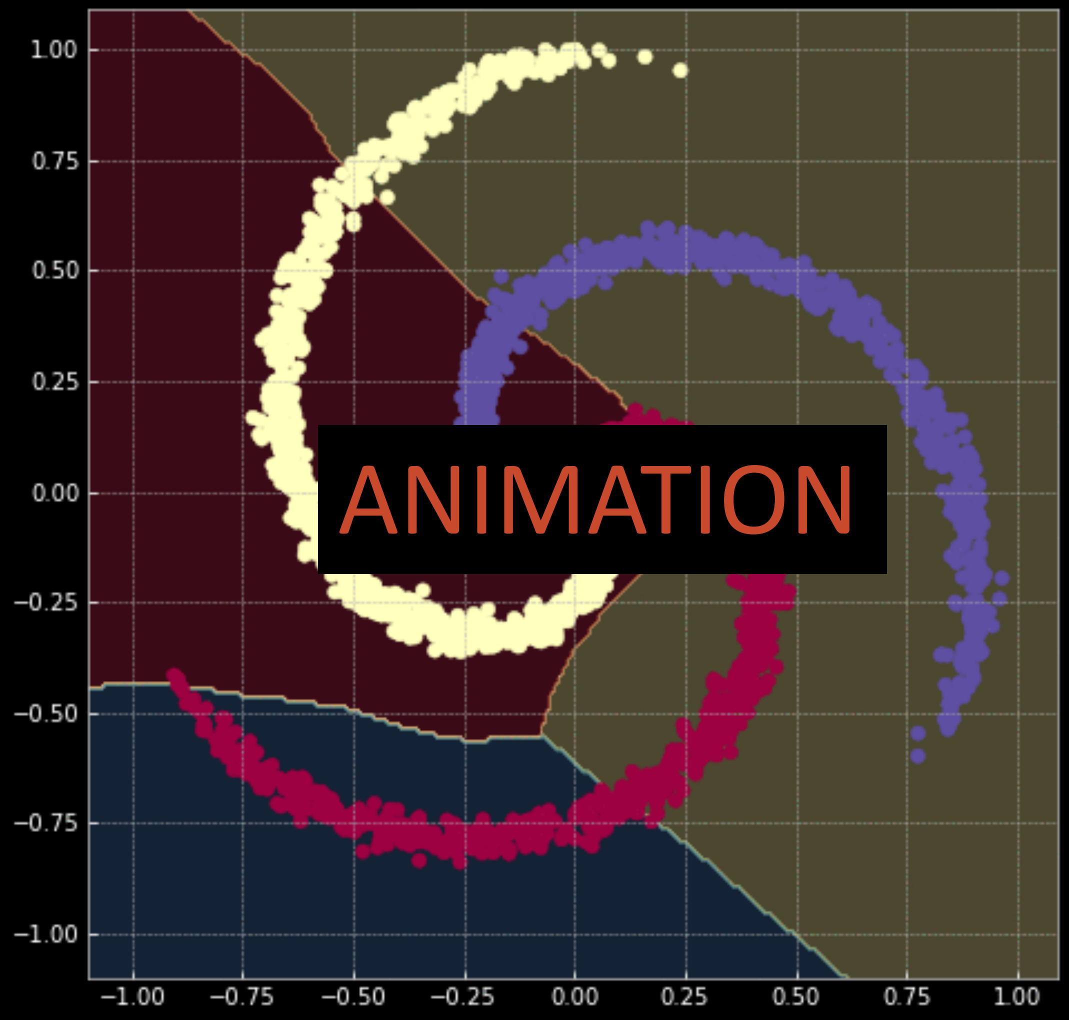
$$X_c(t) = t \begin{pmatrix} \sin\left(\frac{2\pi}{C}\left(2t + c - 1\right)\right) \\ \cos\left(\frac{2\pi}{C}\left(2t + c - 1\right)\right) \end{pmatrix}$$

$$0 \leq t \leq 1, \quad c = 1, \cdots, C$$

$$X_c(t) = t \begin{pmatrix} \sin\left(\frac{2\pi}{C}\left(2t + c - 1\right)\right) \\ \cos\left(\frac{2\pi}{C}\left(2t + c - 1\right)\right) \end{pmatrix}$$

$$\boxed{+\mathcal{N}(0, \sigma^2)}$$


ANIMATION
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


ANIMATION
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

Training data

$(0\ 0\ 1)$
$(0\ 1\ 0)$ 1-hot encoding
$(1\ 0\ 0)$

$$X = \begin{array}{c} n \\ \begin{bmatrix} \text{---} & \underline{x}^{(1)} & \text{---} \\ \text{---} & \underline{x}^{(2)} & \text{---} \\ & \vdots & \\ \text{---} & \underline{x}^{(m)} & \text{---} \end{bmatrix} \end{array} m \qquad Y = \begin{array}{c} G \\ \begin{bmatrix} \text{---} & \underline{y}^{(1)} & \text{---} \\ \text{---} & \underline{y}^{(2)} & \text{---} \\ & \vdots & \\ \text{---} & \underline{y}^{(m)} & \text{---} \end{bmatrix} \end{array} m \qquad \boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \in \mathbb{R}^m$$

$\boldsymbol{x}^{(i)} \in \mathbb{R}^n,\ n = 2$

m : # samples in data set

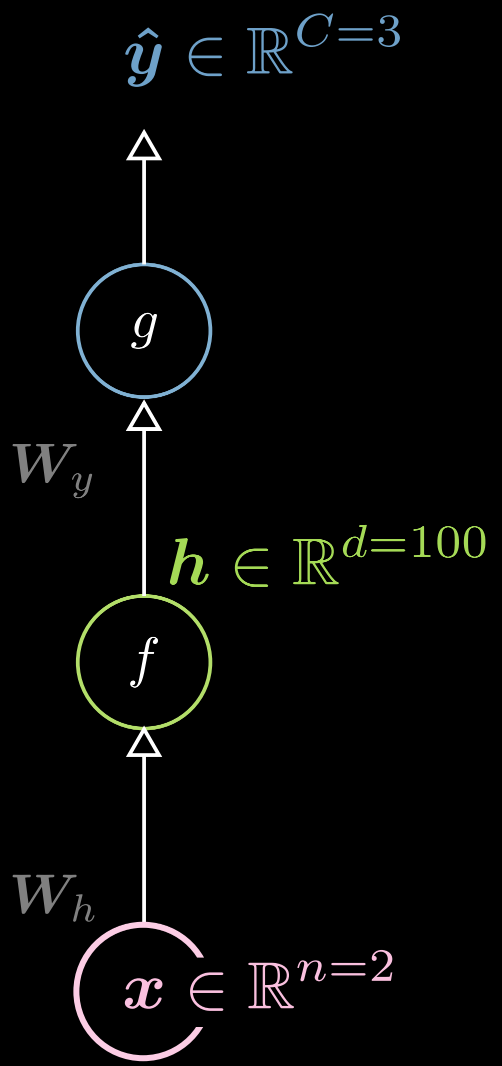
$\boldsymbol{y}^{(i)} \in \mathbb{R}^G,\ G = 3$

$c_i \in \{1, \dots, G\} \subseteq \mathbb{N}^+$

Useful functions

- `import torch`
- `torch.linspace()`
- Select *k*-th sample: `X[k-1] = blah`
- Select *k*-th sample *j*-th component: `X[k-1,j-1]` or `X[k-1][j-1]`

Neural network (inference)



$$\boldsymbol{h} = f(\boldsymbol{W_h} \boldsymbol{x} + \boldsymbol{b_h})$$
$$\hat{\boldsymbol{y}} = g(\boldsymbol{W_y} \boldsymbol{h} + \boldsymbol{b_y})$$

$$\boldsymbol{W_h} \in \mathbb{R}^{d \times n}$$
$$\boldsymbol{b_h} \in \mathbb{R}^{d}$$
$$\boldsymbol{W_y} \in \mathbb{R}^{C \times d}$$
$$\boldsymbol{b_y} \in \mathbb{R}^{C}$$

$$f(\cdot), g(\cdot) : \mathrm{ReLU}(\cdot) \doteq (\cdot)^+, \sigma(\cdot), \tanh(\cdot), \mathrm{softmax}(\cdot)$$

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}(\boldsymbol{x}), \quad \hat{\boldsymbol{y}} : \mathbb{R}^n \to \mathbb{R}^C, \quad \boldsymbol{x} \mapsto \hat{\boldsymbol{y}}$$

$$\hat{\boldsymbol{y}} : \mathbb{R}^n \to \mathbb{R}^d \to \mathbb{R}^C, \quad d \gg n, C$$

Neural network (training I)

$$\boldsymbol{h} = f(\boldsymbol{W}_h \boldsymbol{x} + \boldsymbol{b}_h)$$
$$\hat{\boldsymbol{y}} = g(\boldsymbol{W}_y \boldsymbol{h} + \boldsymbol{b}_y)$$

logits: output of final layer

$$\text{softmax}(\boldsymbol{l})[c] \doteq \frac{\exp(\boldsymbol{l}[c])}{\sum_{j=1}^{C} \exp(\boldsymbol{l}[j])} \in (0,1)$$

$$\mathcal{L}(\hat{\boldsymbol{Y}}, \boldsymbol{c}) \doteq \frac{1}{m} \sum_{i=1}^{m} \ell(\hat{\boldsymbol{y}}^{(i)}, c^{(i)}), \quad \ell(\hat{\boldsymbol{y}}, c) \doteq -\log(\hat{\boldsymbol{y}}[c])$$

cross entropy / negative log-likelihood

$$\underline{x} \quad c = 1 \implies \underline{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\underline{\hat{y}}(\underline{x}) = \begin{pmatrix} \sim 1 \\ \sim 0 \\ \sim 0 \end{pmatrix} \implies \mathcal{L}\left(\begin{pmatrix} \sim 1 \\ \sim 0 \\ \sim 0 \end{pmatrix}, 1\right) \to 0^{+}$$

$$\underline{\hat{y}}(\underline{x}) = \begin{pmatrix} \sim 0 \\ \sim 1 \\ \sim 0 \end{pmatrix} \implies \mathcal{L}\left(\begin{pmatrix} \sim 0 \\ \sim 1 \\ \sim 0 \end{pmatrix}, 1\right) \to +\infty$$

$$h = f(W_h x + b_h)$$
$$\hat{y} = g(W_y h + b_y)$$

Neural network (training II)

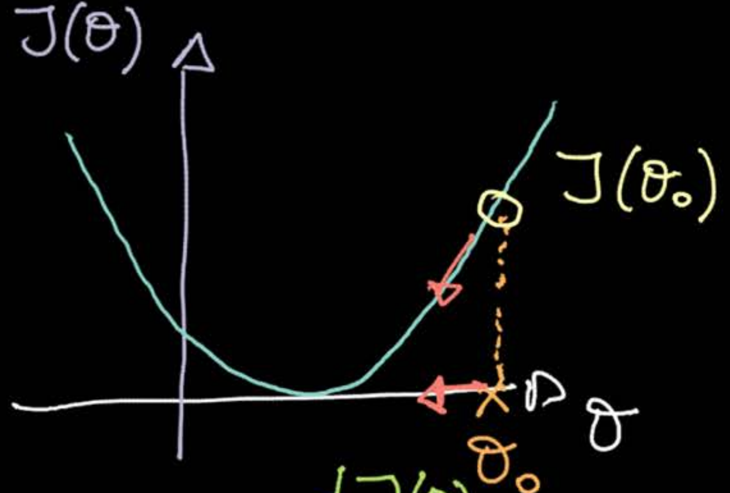
$$\Theta \equiv \{W_h, \boldsymbol{b}_h, W_y, \boldsymbol{b}_y\}$$

$$J(\Theta) \equiv \mathcal{L}(\hat{Y}(\Theta), \underline{c}) \in \mathbb{R}^+$$

$$\frac{\partial J(\Theta)}{\partial W_y} = \frac{\partial J(\Theta)}{\partial \underline{\hat{y}}} \frac{\partial \hat{y}}{\partial W_y}$$

$$\frac{\partial J(\Theta)}{\partial W_h} = \frac{\partial J(\Theta)}{\partial \underline{\hat{y}}} \frac{\partial \hat{y}}{\partial \underline{h}} \frac{\partial \underline{h}}{\partial W_h}$$

Back propagation



$$-\frac{dJ(\theta)}{d\theta}(\theta_0)$$

Gradient descent

Gradient computation example

```
x = torch.tensor([[1, 2], [3, 4]], requires_grad=True)
y = x - 2
z = y * y * 3
out = z.mean()
```

out: $o = \frac{1}{4} \sum_{i=1}^{4} z_i$

$z = 3y^2 = 3(x-2)^2$

$\frac{\partial o}{\partial x_i} = \frac{1}{4} 3 \cdot 2 (x-2) = \frac{3}{2}(x-2)$

$x = 1 \quad \frac{\partial o}{\partial x}\Big|_{x=1} = -1.5$

$\frac{\partial o}{\partial x}\Big|_{x=2} = 0 \quad \frac{\partial o}{\partial x}\Big|_{x=3} = 1.5 \quad \frac{\partial o}{\partial x}\Big|_{x=4} = 3$