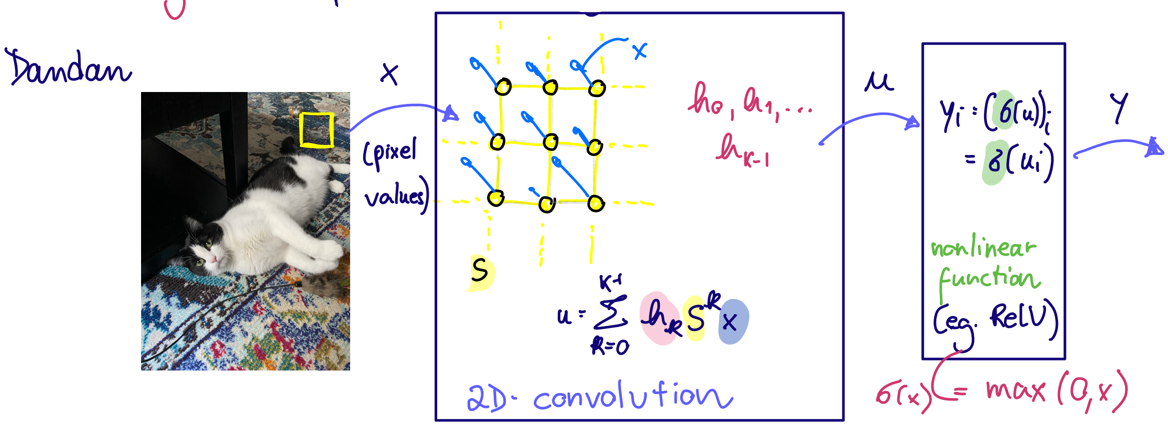
Today: - a graph perceptron
- multi-layer graph perceptron
- full-fledged GNN
- PyTorch

→ Graph filters work reasonably well for many problems, but they are limited to linear representations, which lack expressivity for complex tasks

→ This is not exclusive to graph problems; regular conv's suffer from the same limitation in, e.g., image processing

→ Inspired by the immense success of CNNs (see Nobel prize winner Geoffrey Hinton, Yan LeCunn, etc.) & GSP (circa 2013), GNNs extend CNNs to the graph domain

E.g.: a simple CNN



Graph perceptron

Let $\sigma : \mathbb{R} \to \mathbb{R}$. The graph perceptron is defined as

$$x \in \mathbb{R}^n \longrightarrow \boxed{u = \sum_{k=0}^{K-1} h_k S^k x \qquad \text{(graph conv.)}} \xrightarrow{u}$$

$$\xrightarrow{u} \boxed{y = \sigma(u) \qquad \text{(pointwise nonlinearity)}} \xrightarrow{y}$$

I.e., a graph conv. followed by point. nonlinear fcn

$[y]_i = \sigma([u]_i)$ (node-wise on the graph)

σ examples:

$\sigma(x) = \max(0, x)$ (ReLU)

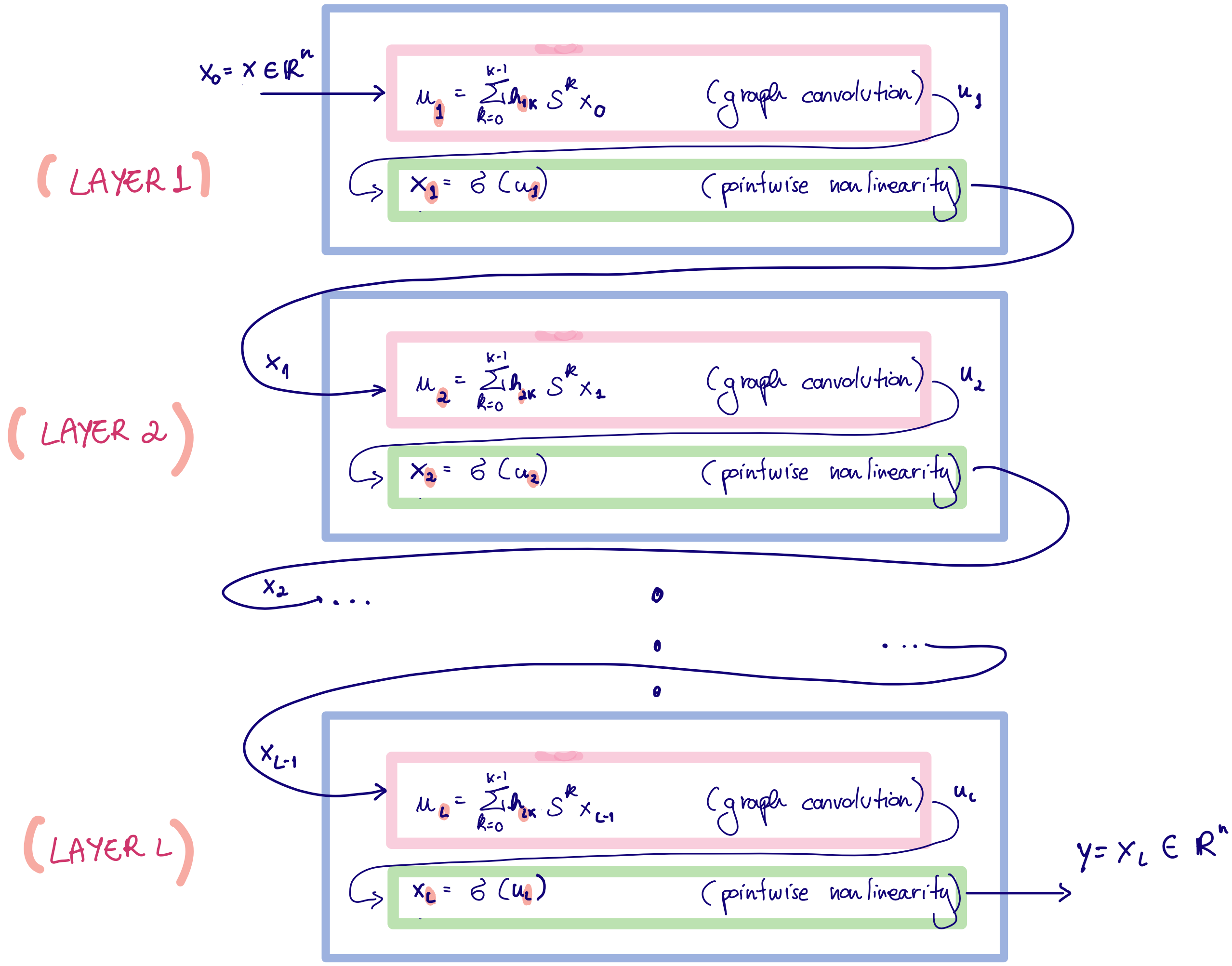
$\sigma(x) = \dfrac{1}{1+e^{-x}}$ (sigmoid)

$\sigma(x) = \tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x}+e^{-x}}$ (hyp. tangent)

y is also a graph signal

Deep (multi-layer) graph perceptron

An L-layer graph perceptron is given by:



$$x_\ell = \sigma(u_\ell) = \sigma\left(\sum_{k=0}^{K-1} h_{\ell,k} S^k x_{\ell-1}\right), \quad 1 \le \ell \le L$$

$x_0 = x$ and $y = x_L$ — x_ℓ the ℓ-layer embedding

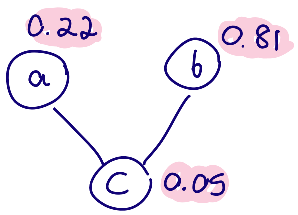
For conciseness, we group $\mathcal{H} = \{h_{k,\ell} \mid 0 \le k \le K-1,\ 1 \le \ell \le L\}$ and represent the ML graph perc. $y = \Phi_{\mathcal{H}}(S, \mathbf{x})$

▶ Full-fledged GNNs

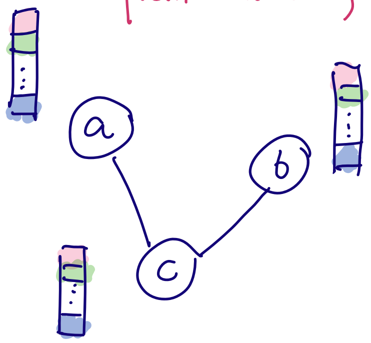
- Real-world graph data is often high dimensional:

  $X \in \mathbb{R}^{n \times d}$, where d is the data dimension or the nb of features

E.g.: so far, $d = 1$ (scalars)



from now on, $d > 1$



For instance, suppose a, b, c are drones communicating via wifi and x is their spatial coordinates in 3D : $x \in \mathbb{R}^{\overset{n}{3} \times \overset{d}{3}}$

Similarly, we might want our embeddings to be multi-dimensional for higher expressivity; think of each entry of a node embedding as encoding a relevant feature for the task multiple-input multiple-output

↳ convolutional filter bank or MIMO graph convolution

Let $X \in \mathbb{R}^{n \times d}$, $H_k \in \mathbb{R}^{d \times d'}$ for $k = 0, \dots, K-1$

$$Y = \sum_{k=0}^{K-1} \underbrace{S^k}_{n \times n} \underbrace{X}_{n \times d} \underbrace{H_k}_{d \times d'}$$

graph diffusion/ shift / message-passing

linear transform mapping d feats to d' feats

A "full-fledged" GNN layer is given by:

$$X_\ell = \sigma(U_\ell) = \sigma\left(\sum_{k=0}^{K-1} S^k X_{\ell-1} H_{\ell,k} \right)$$

$$H_{\ell,k} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$$

for $1 \leq \ell \leq L$, where $X_0 = X \in \mathbb{R}^{n \times d_0}$

$$Y = X_L \in \mathbb{R}^{n \times d_L}$$

X_ℓ is still called ℓ- layer embedding

For conciseness, we group $\mathcal{H} = \{ H_{k,\ell} \mid 0 \leq k \leq K-1, 1 \leq \ell \leq L \}$

and represent the GNN as $Y = \Phi_{\mathcal{H}}(S, X)$