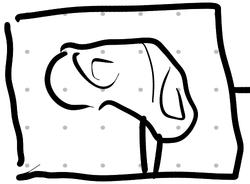
Statistics and Data Science for Engineers
E178 / ME276DS

Neural networks

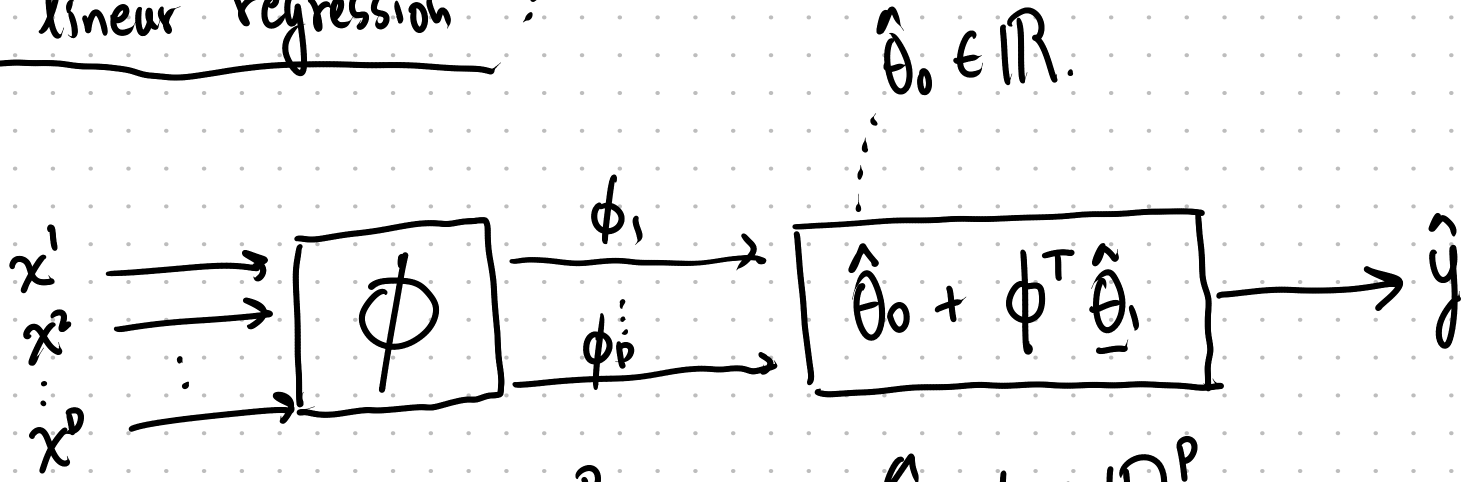
$x \rightarrow$  $\rightarrow y$

$x \rightarrow \boxed{h(x)} \rightarrow \hat{y}$

similar

- Linear Regression (regression)
- Logistic regression (classification).
- ⋮
- Support Vector machine
- Decision trees
- Neural networks.

Recall linear regression:

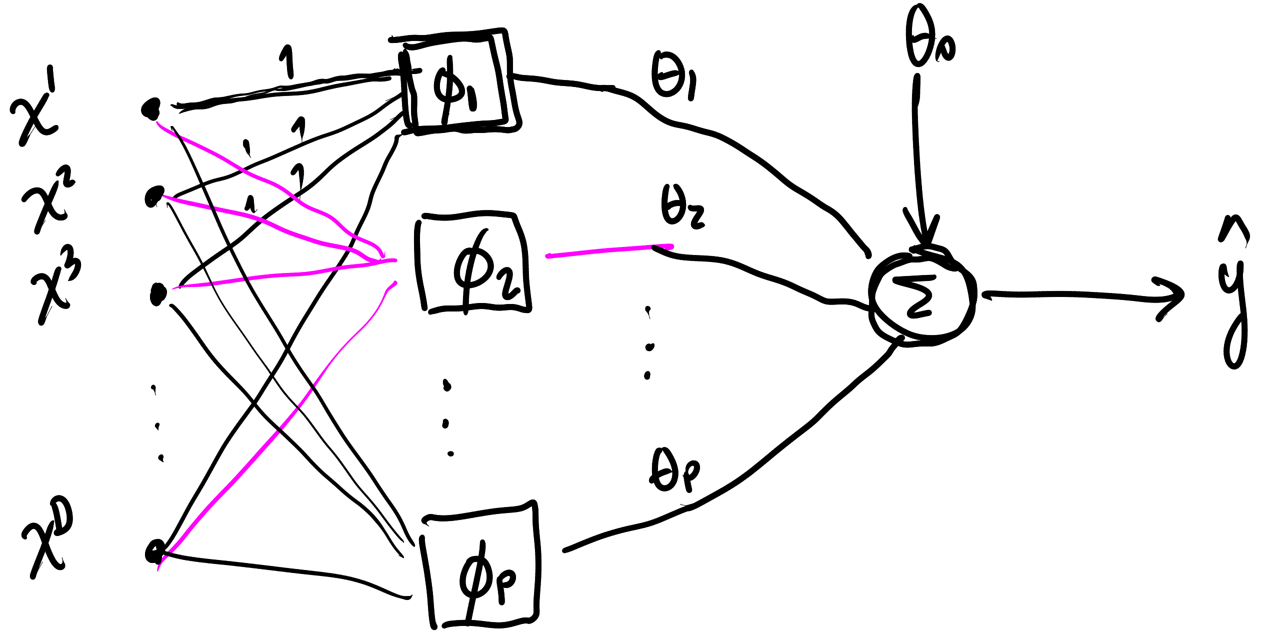


$$\phi : \underset{\text{inputs}}{\mathbb{R}^D} \rightarrow \underset{\text{features}}{\mathbb{R}^P} \qquad \hat{\theta}_1, \phi \in \mathbb{R}^P$$

- I have to manually design the features ϕ.
- Neural networks can be understood as a way to design the features automatically.

Pictorial representation of linear regression

$$\hat{y} = \theta_0 + \phi^T(x)\underline{\theta}_1 = \theta_0 + \sum_{j=1}^{P} \phi_j(x)\theta_j$$

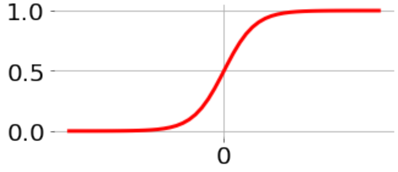


<u>3 steps to NN:</u>

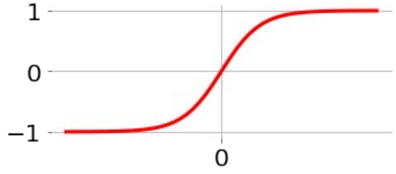
1. Replace the ϕ with "activation functions"
2. Put weights on <u>all</u> edges.
3. Replicate layers.

1) Generic nonlinearities, a.k.a. *activation functions*

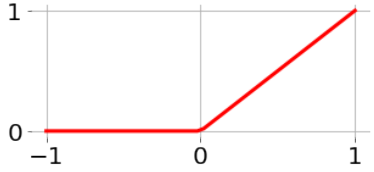
sigmoid: $$\phi(\xi) = \frac{1}{1 + e^{-\xi}}$$



tanh: $$\phi(\xi) = \frac{e^{2\xi} - 1}{e^{2\xi} + 1}$$

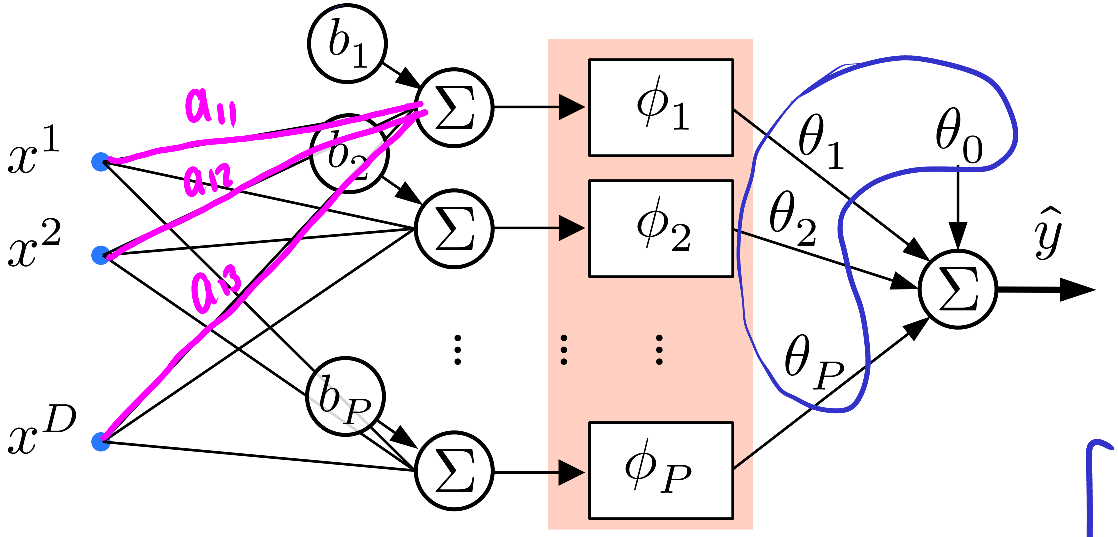


ReLU: $$\phi(\xi) = \text{max}(0, \xi)$$



2) Weights on the inputs

$$\hat{y} \;=\; \theta_0 + \phi^T(b + Ax)\underline{\theta}_1 \;=\; \theta_0 + \sum_{j=1}^{P} \phi_j(b_j + A_j x)\theta_j$$



$P \times D$ "a" coeff.

P "b" coeff.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{P1} & \vdots & a_{P3} \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ \vdots \\ b_P \end{bmatrix}$$

$$\phi_1 (a_{11} x^1 + a_{12} x^2 + a_{13} x^3 + b_1) \; \theta_1$$

$$\phi_1 (A_1 x + b_1)$$

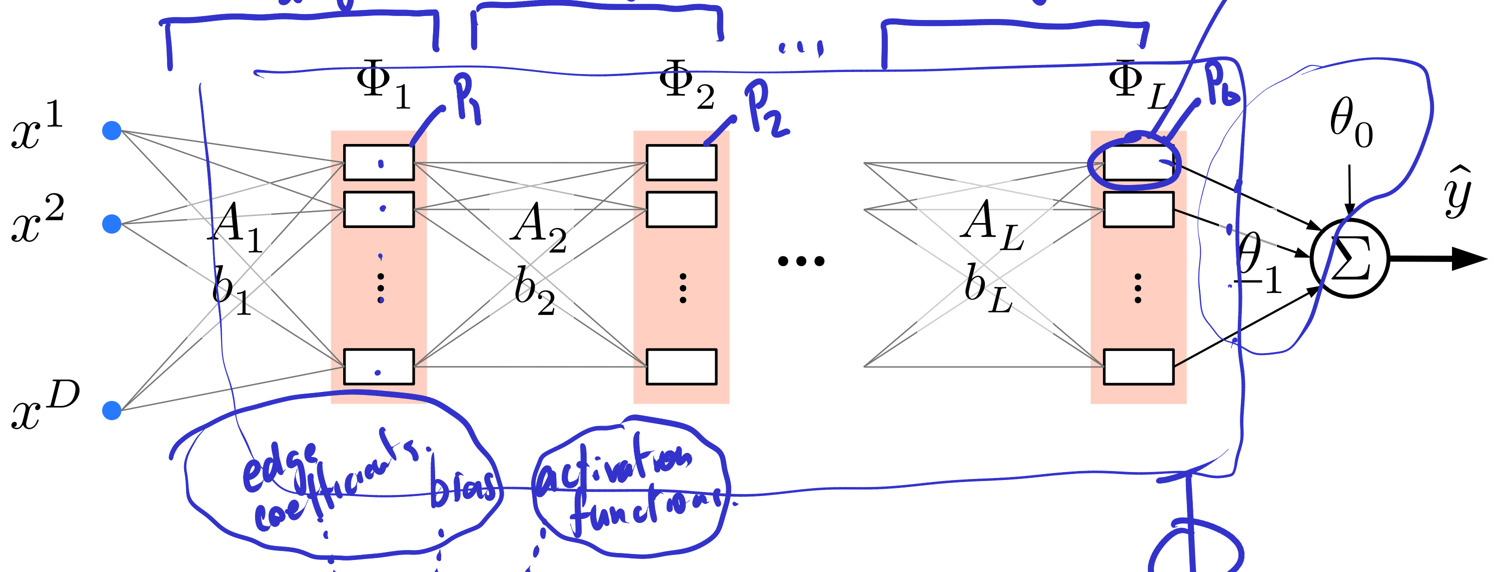
$$A_1 = [a_{11} , a_{12} , a_{13}]$$

3) Multiple layers

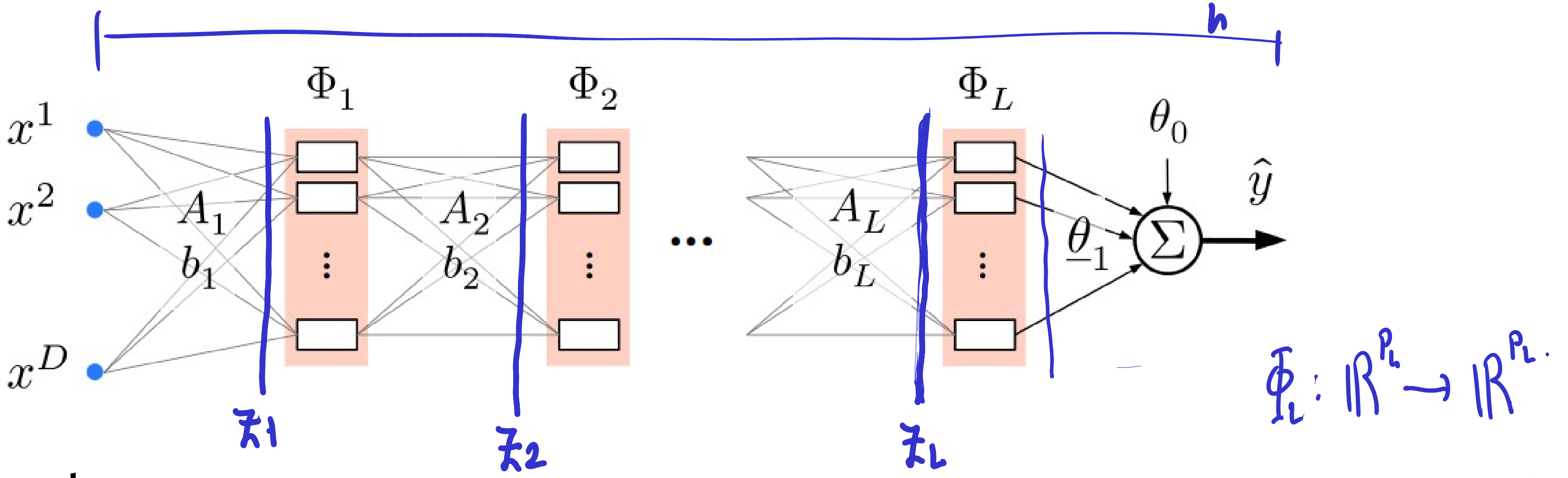
1st layer

2nd layer

Lth layer.



- Hidden layers: $(A_\ell, b_\ell, \Phi_\ell, P_\ell)$ $\ell = 1 \dots L$
- Output layer: $(\theta_0, \underline{\theta}_1)$ ... final linear regression. (regression problem)



Put it all together :

$$\hat{y} = h(x) = \theta_0 + \underline{\theta}_1^T \Phi_L\Big(b_L + A_L \cdot \Phi_{L-1}\big(\underbrace{b_{L-1} + A_{L-1}\Phi_{L-2}(\cdots \Phi_1(b_1 + A_1 x)\ldots)}_{z_{L-1}}\big)\Big).$$

... complicated nested. function.

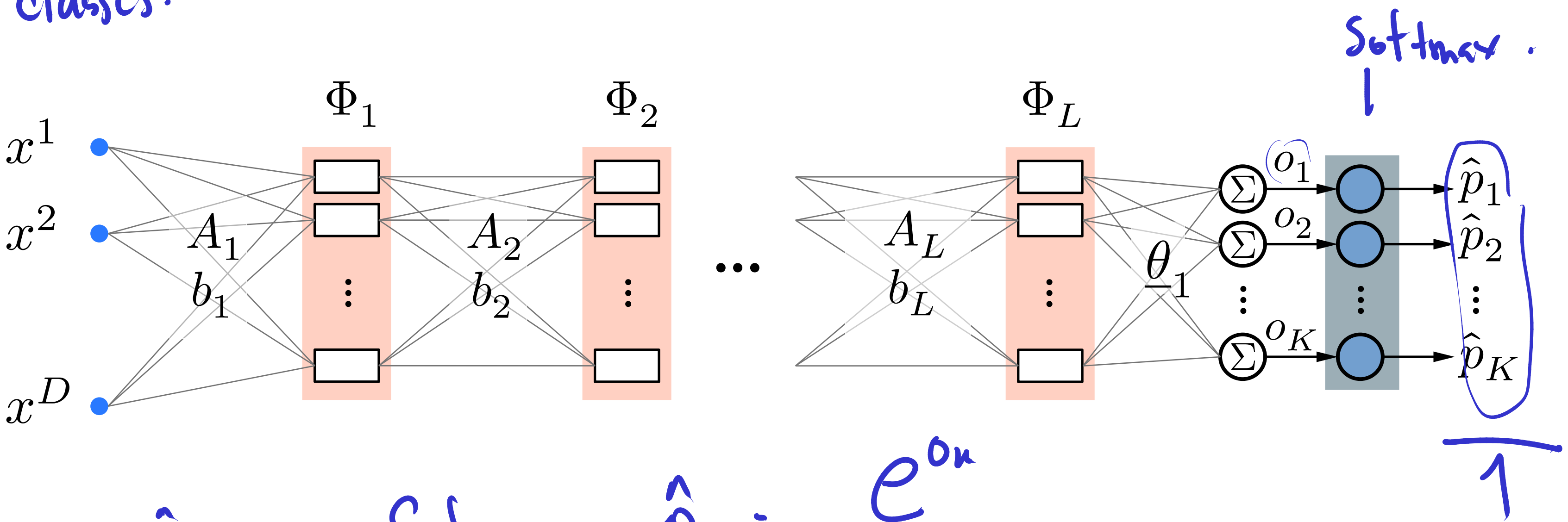
... perceptron.

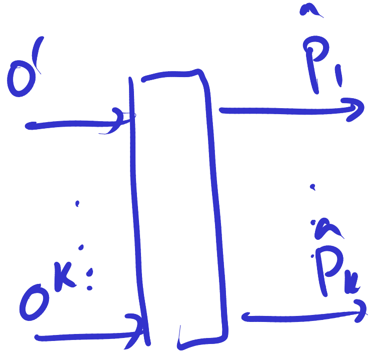
Write this in a recursive form.

$$\begin{cases} \hat{y} = \theta_0 + \underline{\theta}_1^T \Phi_L(z_L) \\ z_L = b_L + A_L \cdot \Phi_{L-1}(z_{L-1}) \\ z_{L-1} = b_{L-1} + A_{L-1} \Phi_{L-2}(z_{L-2}) \\ \quad \vdots \\ z_\ell = b_\ell + A_\ell \Phi_{\ell-1}(z_{\ell-1}) \qquad \text{(general form)} \\ \quad \vdots \\ z_1 = b_1 + A_1 x \end{cases}$$

Classification networks

K ... classes.





Softmax: $\hat{p}_k = \dfrac{e^{o_k}}{\sum e^{o_k}}$

Properties:

- $\hat{p}_k > 0$.
- $\sum_k \hat{p}_k = 1$
- preserves order: $o_i > o_j \implies \hat{p}_i > \hat{p}_j$

K entries.

One-hot encoding (OHE)

$$c_1 \longrightarrow \begin{bmatrix}1\\0\\\vdots\\0\\0\end{bmatrix} \qquad c_2 \longrightarrow \begin{bmatrix}0\\1\\\vdots\\0\\0\end{bmatrix} \qquad \cdots \qquad c_{K-1} \longrightarrow \begin{bmatrix}0\\0\\\vdots\\1\\0\end{bmatrix} \qquad c_K \longrightarrow \begin{bmatrix}0\\0\\\vdots\\0\\1\end{bmatrix}$$

Example:
Binary output

	0-1 encoding	OHE
y_i	$y_i = 0$ for c_1 $y_i = 1$ for c_2	$y_i = \begin{bmatrix}y_i^1\\y_i^2\end{bmatrix}$ $\nearrow = \begin{bmatrix}1\\0\end{bmatrix}$ for c_1 $\searrow = \begin{bmatrix}0\\1\end{bmatrix}$ for c_2
$\hat{p}_i$	$\hat{p}_i \in [0,1]$.	$\hat{p}_i = \begin{bmatrix}\hat{p}_i^1\\\hat{p}_i^2\end{bmatrix} = \begin{bmatrix}1-\hat{p}_i\\\hat{p}_i\end{bmatrix}$ .....

0.3 ... c1

$\begin{bmatrix}0.7\\0.3\end{bmatrix}$ ... c1.

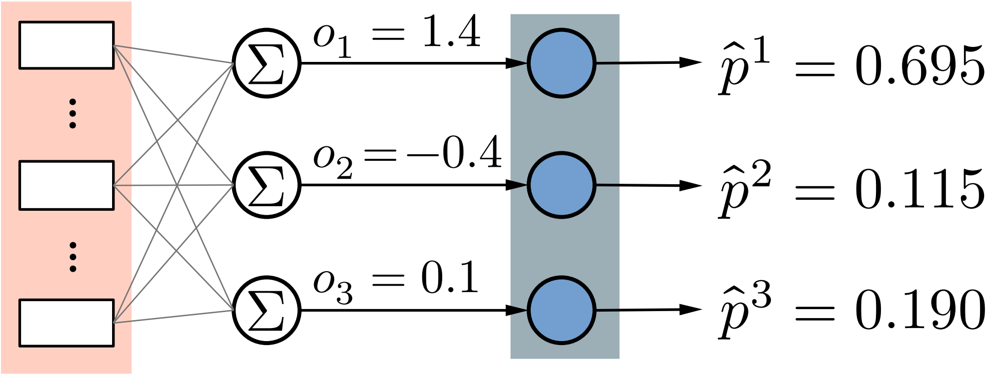
Loss function: Multi-class cross entropy

$$\mathsf{CE}(y_i, \hat{p}_i) = -\sum_{k=1}^{K} y_i^k \log \hat{p}_i^k$$

Recall: Binary cross entropy under 0-1 encoding

$$\mathsf{CE}(y_i, \hat{p}_i) = -y_i \log(\hat{p}_i) - (1 - y_i) \log(1 - \hat{p}_i)$$

Example



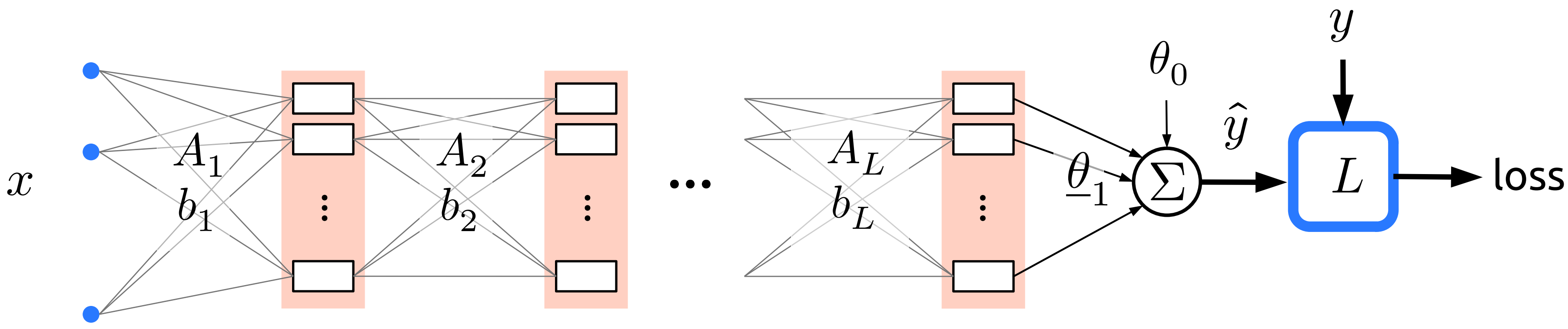
$$\begin{array}{cc} y^k & -y^k \log \hat{p}^k \\ \left(\begin{array}{c} 0 \\ 0 \\ 1 \end{array}\right) & \begin{array}{c} 0 \\ 0 \\ -\log 0.19 \end{array} \end{array}$$

$$\mathsf{CE}(y, \hat{p}) = -\log 0.19$$

Training the neural network

- Hyper-parameters
  - # of layers L
  - # of “neurons” in each layer p_ℓ
  - activation function for each layer

- Tunable parameters
  - All edge weights

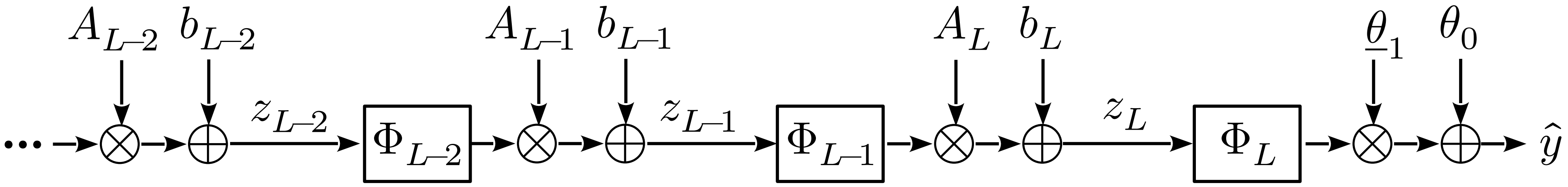


$$\theta = (A_1, b_1, A_2, b_2, \ldots, A_L, b_L, \theta_0, \underline{\theta}_1)$$

Gradient of the loss

Computing $\nabla_\theta h$ with back-propagation

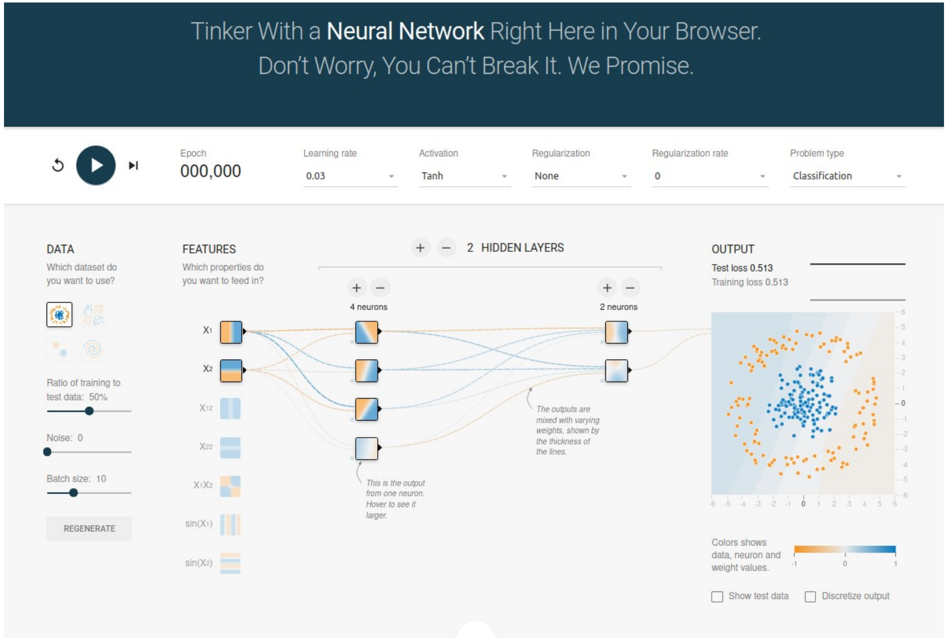
$$
\begin{aligned}
\hat{y} &= \theta_0 + \Phi_L^T(z_L)\underline{\theta}_1) \\
z_L &= b_L + A_L \Phi_{L-1}(z_{L-1}) \\
z_{L-1} &= b_{L-1} + A_{L-1}\Phi_{L-2}(z_{L-2}) \\
&\vdots
\end{aligned}
$$

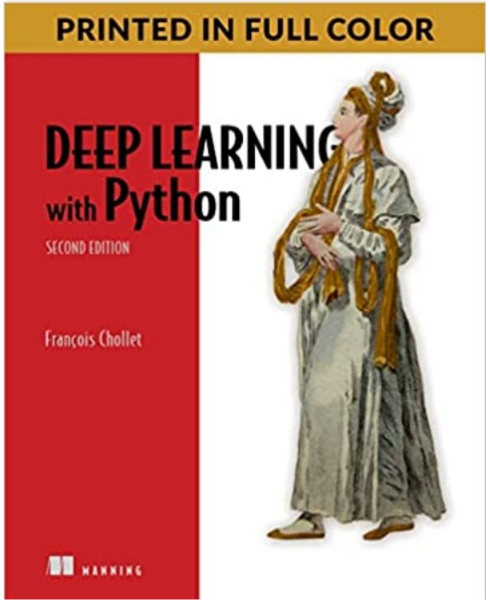


https://playground.tensorflow.org

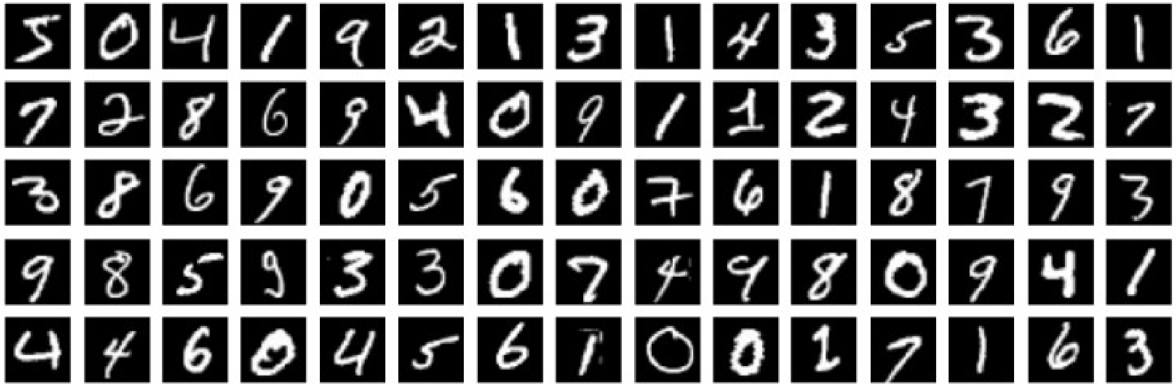
Keras

TensorFlow

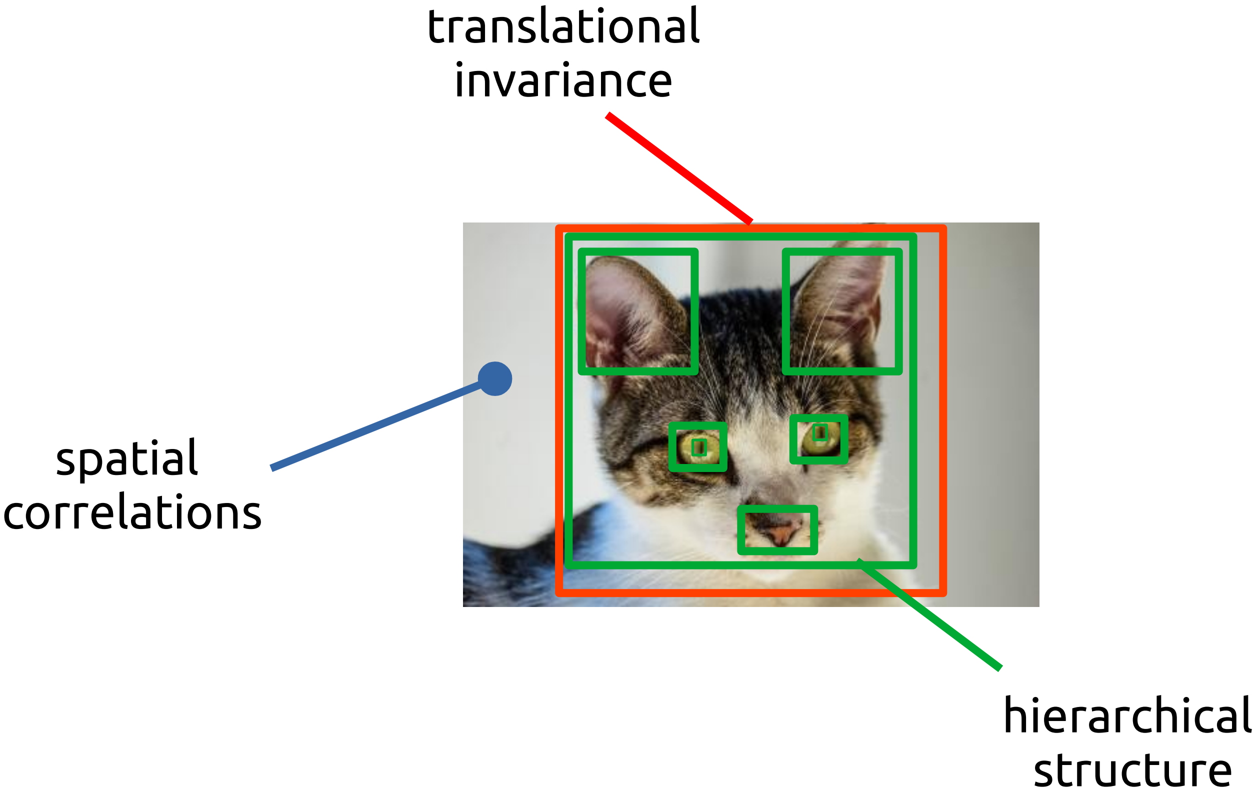




MNIST demo



Neural networks for images



Convolution operator

$$\overset{\text{pixels}}{\begin{array}{|c|c|c|}\hline p_1 & p_2 & p_3 \\ \hline p_4 & p_5 & p_6 \\ \hline p_7 & p_8 & p_9 \\ \hline \end{array}} \quad \overset{\substack{\text{convolution}\\ \text{operator}}}{*} \quad \left\{ \overbrace{\overset{\text{kernel}}{\begin{array}{|c|c|c|}\hline f_1 & f_2 & f_3 \\ \hline f_4 & f_5 & f_6 \\ \hline f_7 & f_8 & f_9 \\ \hline \end{array}} \;,\; \overset{\text{bias}}{b} \;,\; \overset{\text{activation}}{\phi(\cdot)}}^{\text{filter}} \right\} = \phi\left(b + \sum_{i=1}^{9} p_i\, f_i\right)$$

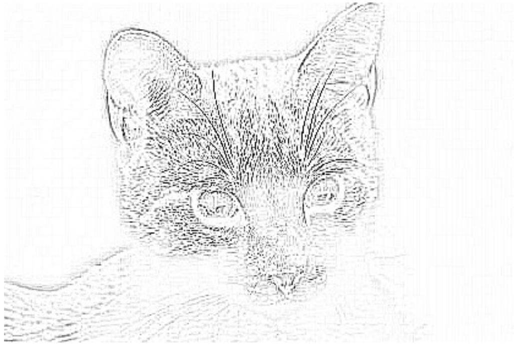
Example

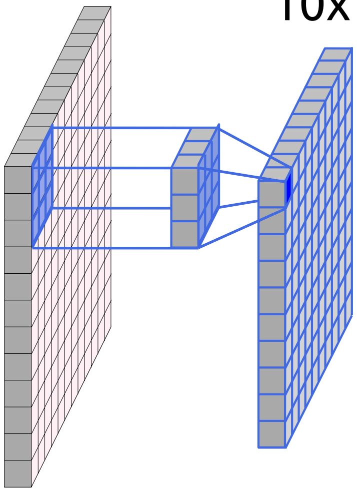
$$
\begin{array}{|c|c|c|}\hline 1.0 & 0.0 & 0.8 \\ \hline 0.1 & 0.3 & 0.5 \\ \hline 0.1 & 0.1 & 0.0 \\ \hline \end{array} * \left\{ \begin{array}{|c|c|c|}\hline -0.2 & 1.2 & 0.2 \\ \hline -0.3 & 0.5 & 0.5 \\ \hline 0.0 & 0.0 & 0.1 \\ \hline \end{array} \;,\; b=1 \;,\; \phi=\text{ReLU} \right\}
$$

$$
\begin{aligned}
&= \phi\left(b + \sum_i p_i\, f_i\right) \\
&= \text{ReLU}(1 + (1.0)(-0.2) + (0.0)(1.2) + \ldots) \\
&= \text{ReLU}(1.33) \\
&= 1.33
\end{aligned}
$$

Convolution of full images

$* \{$  $, \; b \; , \; \phi(\cdot) \} =$





Padding $\{$  $, \ b \ , \ \phi(\cdot) \ , \ p=\text{True} \}$

Stride $\{$  $, \ b \ , \ \phi(\cdot) \ , \ s=2 \}$

 $* \{$  $, \ b \ , \ \phi(\cdot) \ , \ s=2 \}$ $=$ 

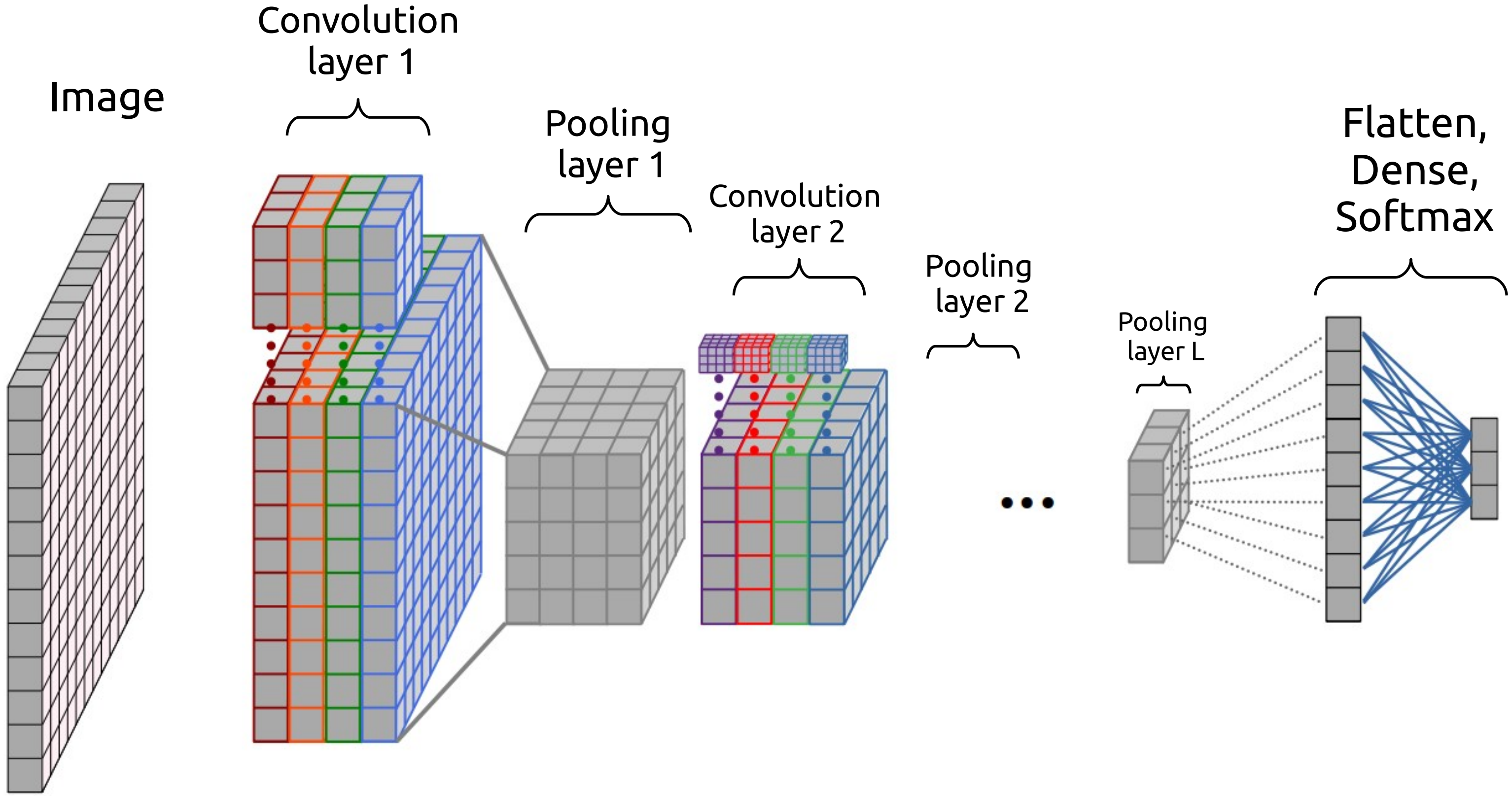


Image
Convolution layer 1
Pooling layer 1
Convolution layer 2
Pooling layer 2
Pooling layer L
Flatten, Dense, Softmax

Pooling