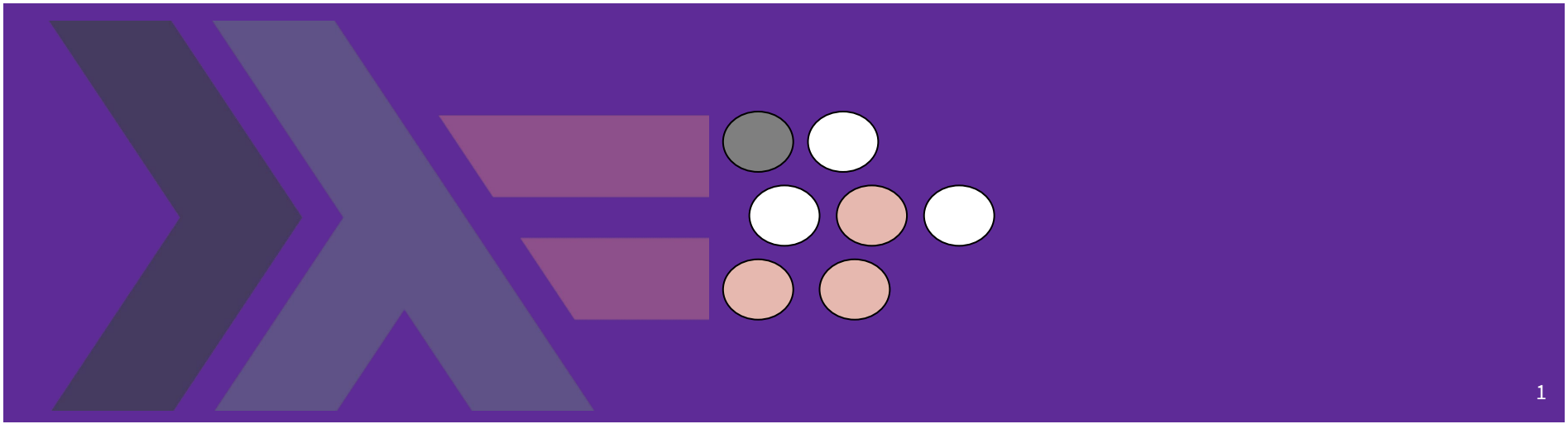
ensisa
École nationale supérieure
d'ingénieurs sud Alsace

Neural Networks in Haskell

By : Omar CHICHAOUI
Poras VEDI
Direct by : DR. L.Thiry(HDR)



CONTENT :

1- The Project

2- Why Haskell (Models/equations)

3- Results/demo

4- Perspective

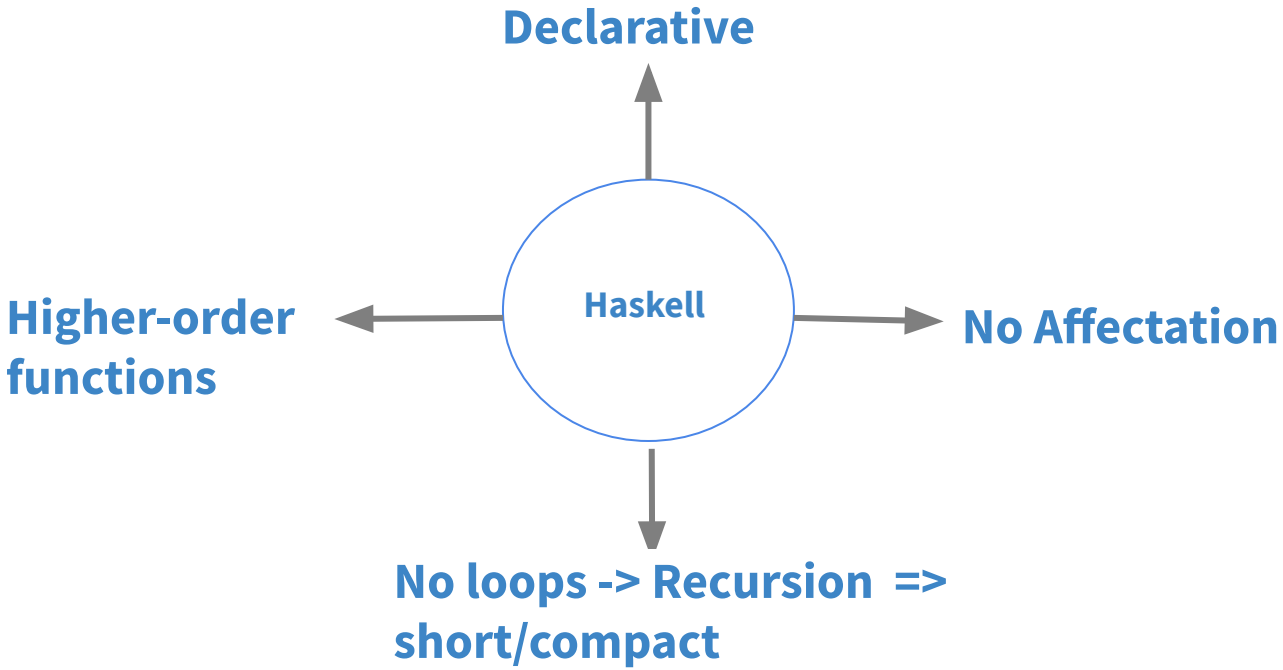
WHAT IS THE PROJECT ABOUT

The Implementation of Neural Networks using Haskell

Basic examples
Advanced examples ANN
Performances

- Steps Needed :
  - Clean Data
  - Neural Network
  - Data analysis

WHY HASKELL ?


Declarative
Higher-order functions
Haskell
No Affectation
No loops -> Recursion => short/compact

WHY HASKELL ?

Examples : only 3 operators needed !

- **zipWith (*) [x1,x2,...,xn] [y1,y2,...,yn] = [x1*y1,x2*y2,...,xn*yn]**
- **map f [x1,x2,...,xn] = [f(x1), f(x2),...,f(xn)]**
- **foldr1 + [x1,x2,...,xn] = [x1+x2+x3....+xn]**

WHY HASKELL ?

Examples : Basic Function

Mult X Y = zipWith (*) [x1,x2,...,xn] [y1,y2,...,yn] = [x1*y1,x2*y2,...,xn*yn]

Add X Y = zipWith (+)[x1,x2,...,xn] [y1,y2,...,yn] = [x1+y1,x2+y2,...,xn+yn]

Minus X Y= zipWith (-)[x1,x2,...,xn] [y1,y2,...,yn] = [x1-y1,x2-y2,...,xn-yn]

Sum X = foldr1 (+) X

WHY HASKELL ?

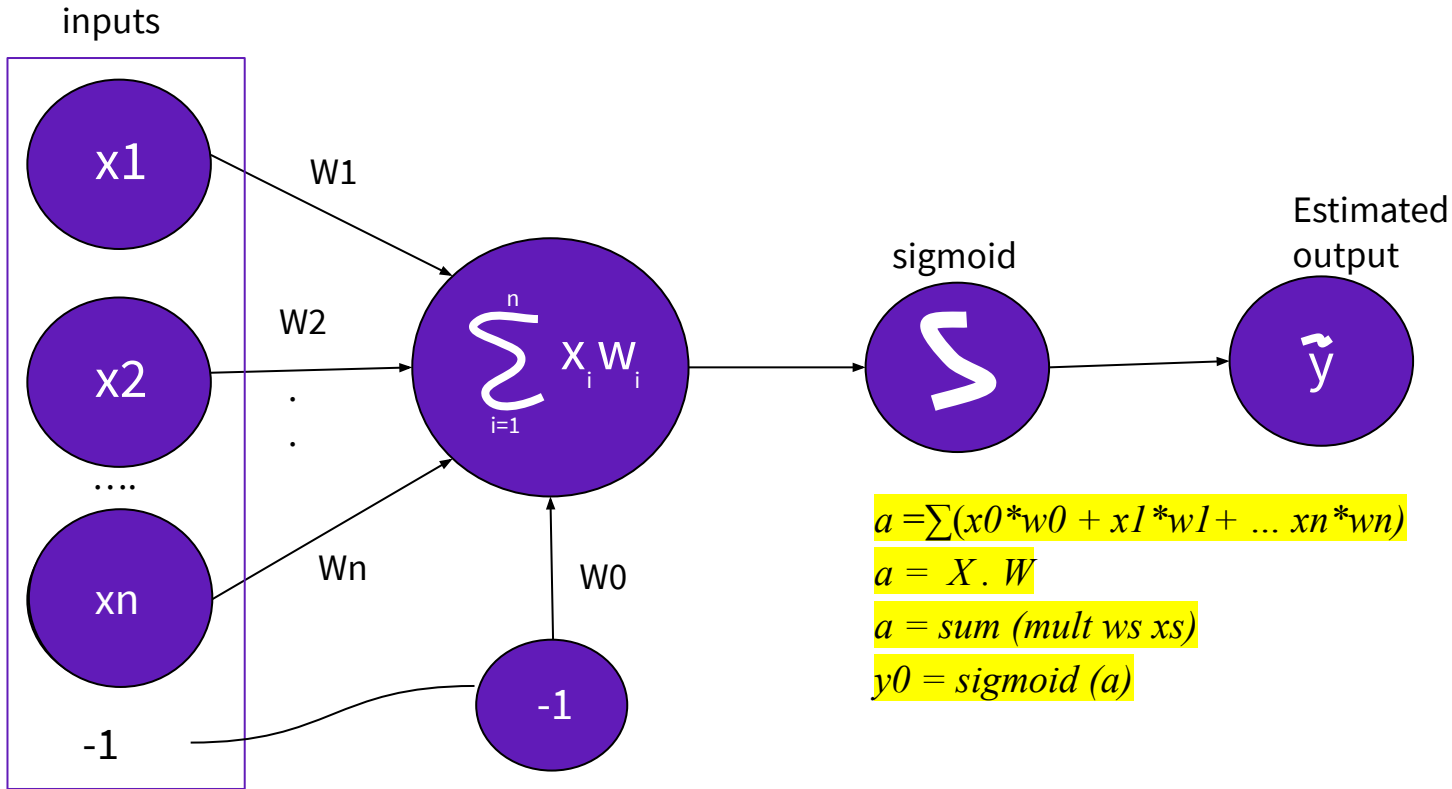
Examples : Function

AddMatrix X Y = zipWith (add) [X1,X2,...,Xn] [Y1,Y2,...,Yn] =
[add X1 Y1,add X2 Y2,...,add Xn Yn]

Sigmoid

WHY HASKELL ?

Models/equations for Neural Networks => Perceptron



$a = \sum(x0*w0 + x1*w1 + \ldots xn*wn)$

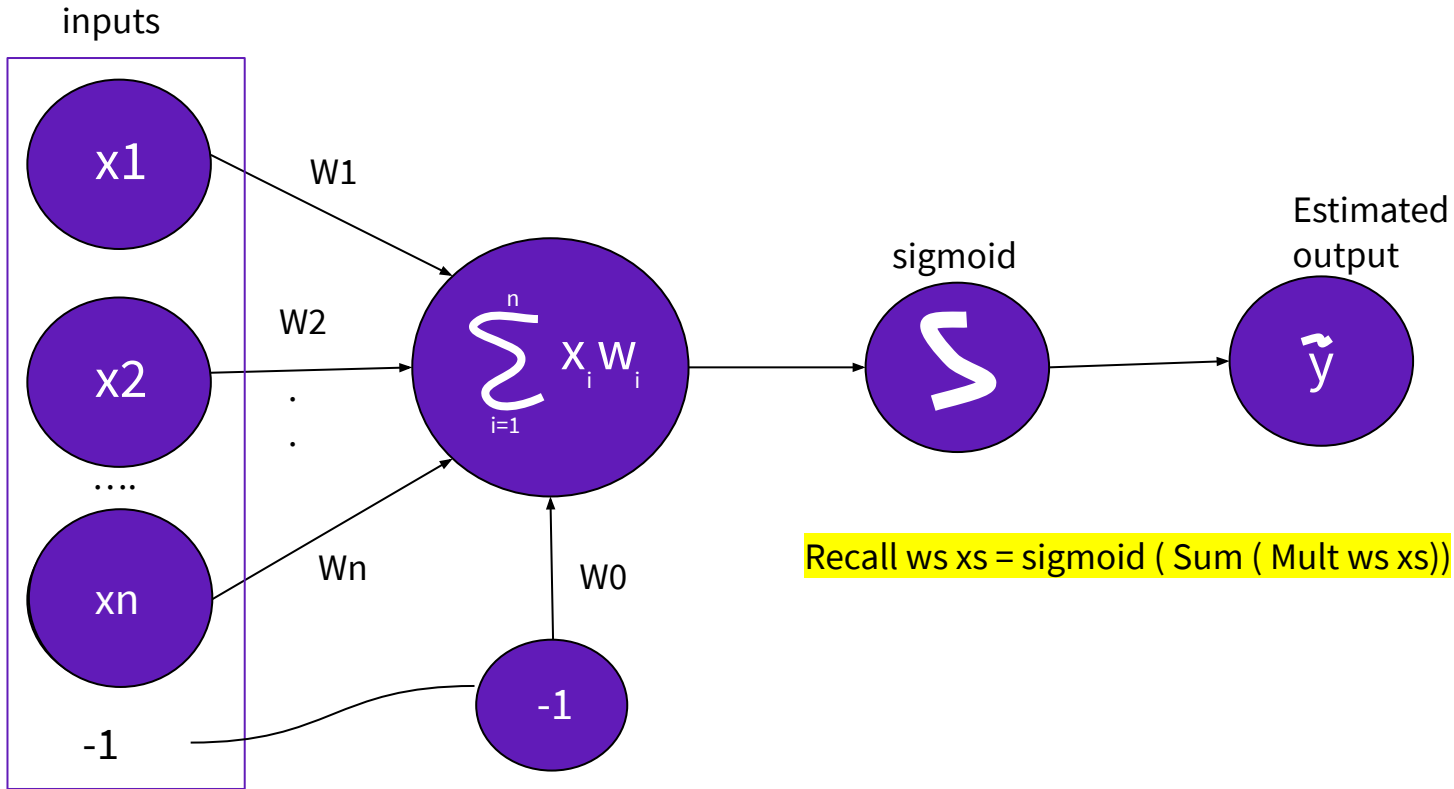
$a = X \,.\, W$

$a = sum\ (mult\ ws\ xs)$

$y0 = sigmoid\ (a)$

WHY HASKELL ?

Models/equations for Neural Networks => Perceptron



Recall ws xs = sigmoid (Sum (Mult ws xs))

WHY HASKELL ?

Models/equations for Neural Networks => Learning / Adaptation

$$error\ e = \frac{(\tilde{y} - y)^2}{2}$$

$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial a} \cdot \frac{\partial a}{\partial w}$$

$$\frac{\partial e}{\partial \tilde{y}} = \tilde{y} - y \qquad \frac{\partial \tilde{y}}{\partial a} = sig'(a) \qquad \frac{\partial a}{\partial w} = xs$$

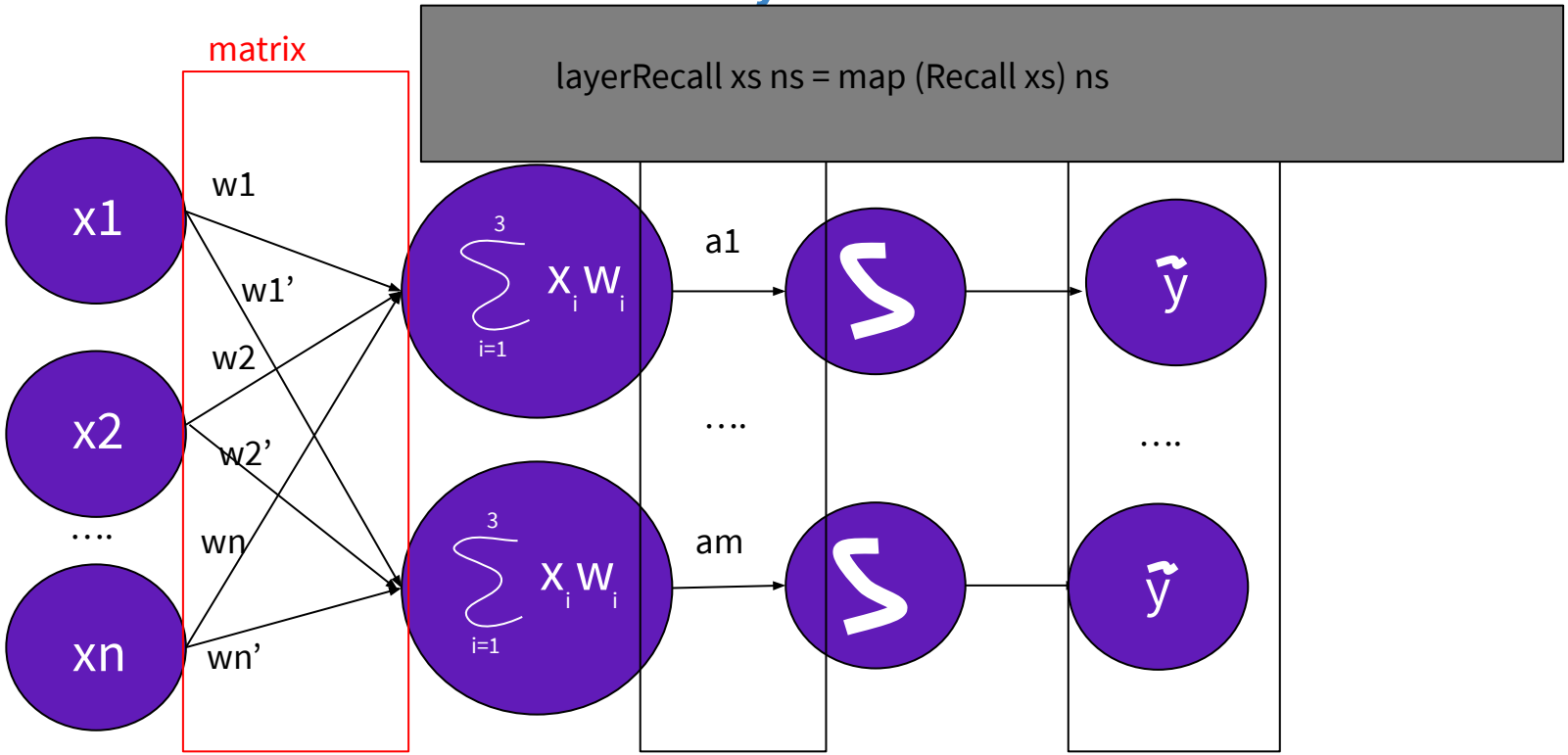
$$\partial e = \tilde{y} - y \qquad \partial \tilde{y} = sig'(a) \qquad \partial a = shift\ (xs) \qquad \partial w = mult\ (\partial e \cdot \partial \tilde{y})\ xs$$

$$sig'(x) = sig(x) * (1 - sig(x))$$

Update Neuron = zipWith (+) ws (map (0.01*err) (-1:xs))

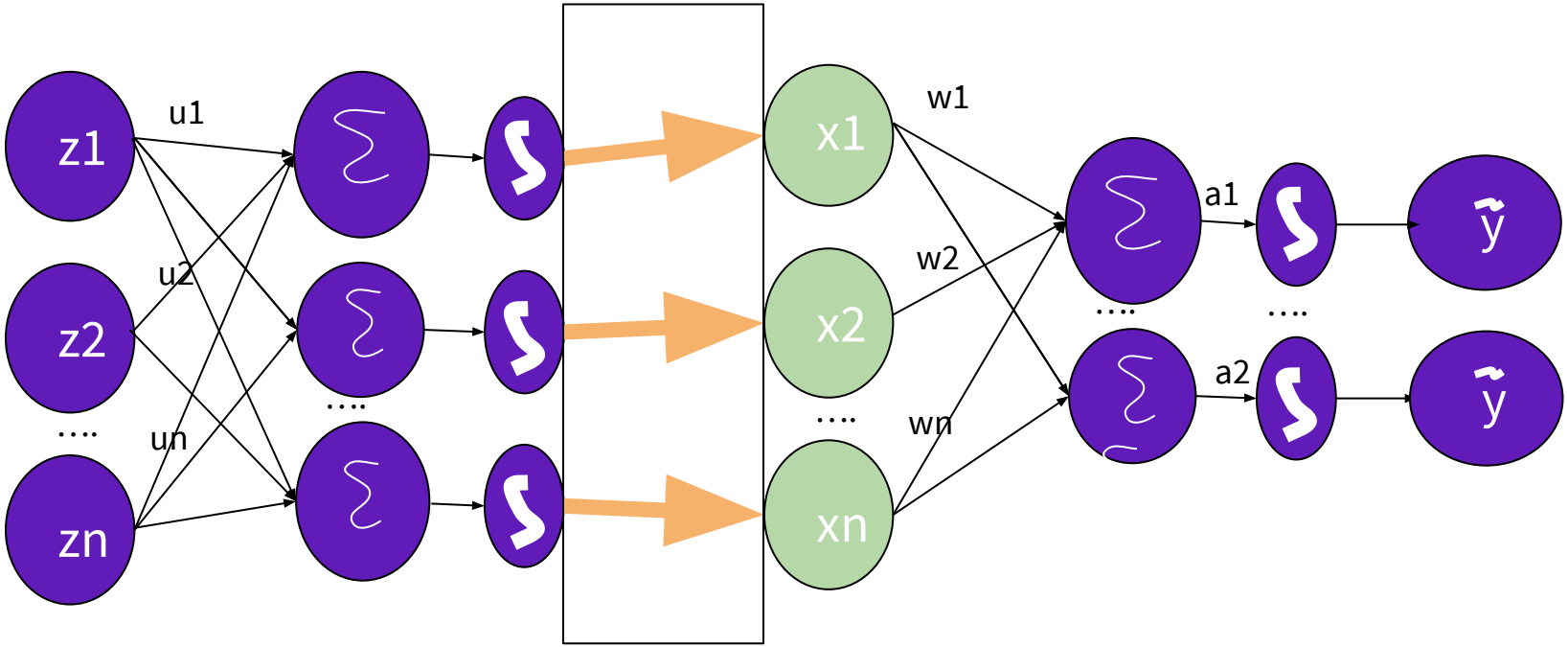
WHY HASKELL ?

Models/equations for Neural Networks => Layer



WHY HASKELL ?

Models/equations for Neural Networks => Net



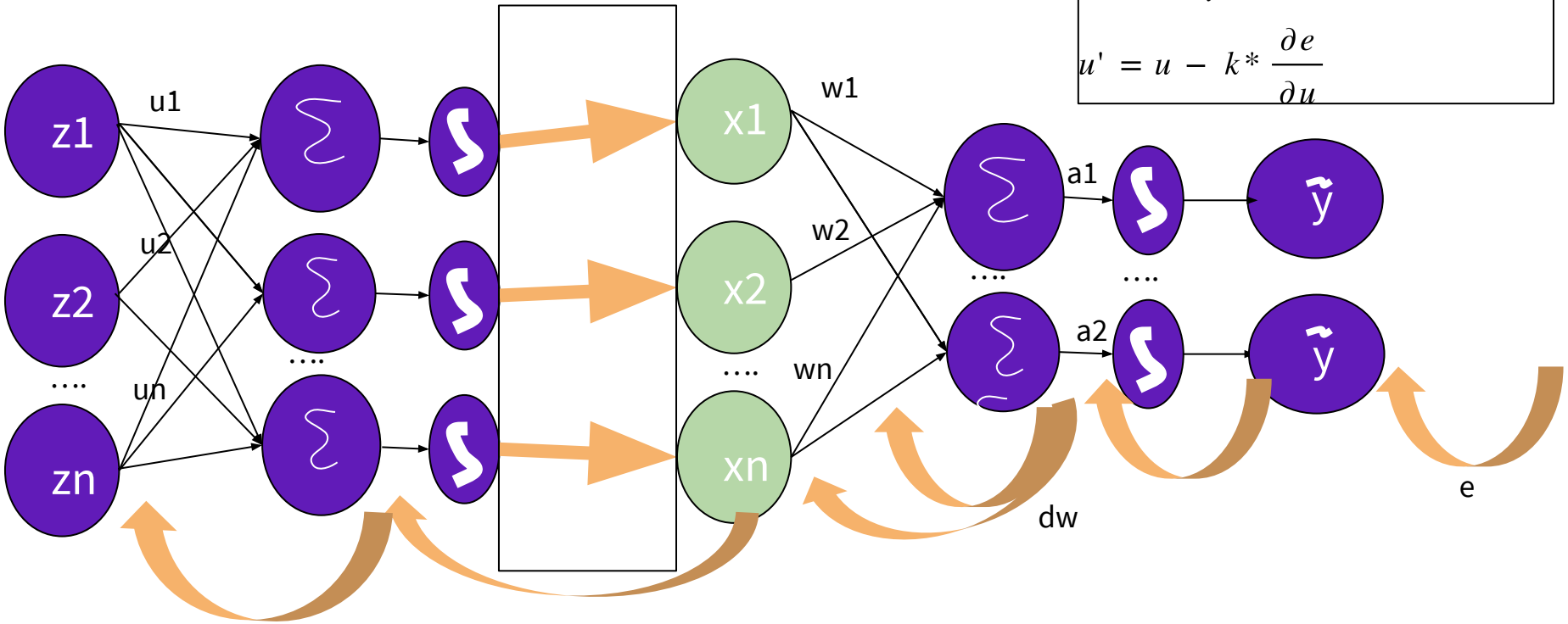
WHY HASKELL ?

Models/equations for Neural Networks

$$error\ e = \frac{(\tilde{y} - y)^2}{2}$$

$$\frac{\partial e}{\partial u} = \frac{\partial e}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial a} \cdot \frac{\partial a}{\partial x} \cdot \frac{\partial x}{\partial b} \cdot \frac{\partial b}{\partial u}$$

$$u' = u - k * \frac{\partial e}{\partial u}$$

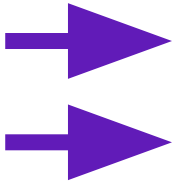


RESULTS/DEMO

Demo XOR Net : organising data

0.0	0.0
0.0	1.0
1.0	0.0
1.0	1.0

0.0
1.0
1.0
0.0



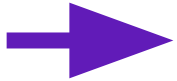
xor.dat

```
[([0.0, 0.0], [0.0])
,([0.0, 1.0], [1.0])
,([1.0, 0.0], [1.0])
,([1.0, 1.0], [0.0])]
```

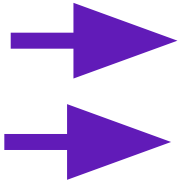
RESULTS/DEMO

Demo AutoEncoder Net : data

0.0	0.0	0.0
1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0



0.0	0.0	0.0
1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

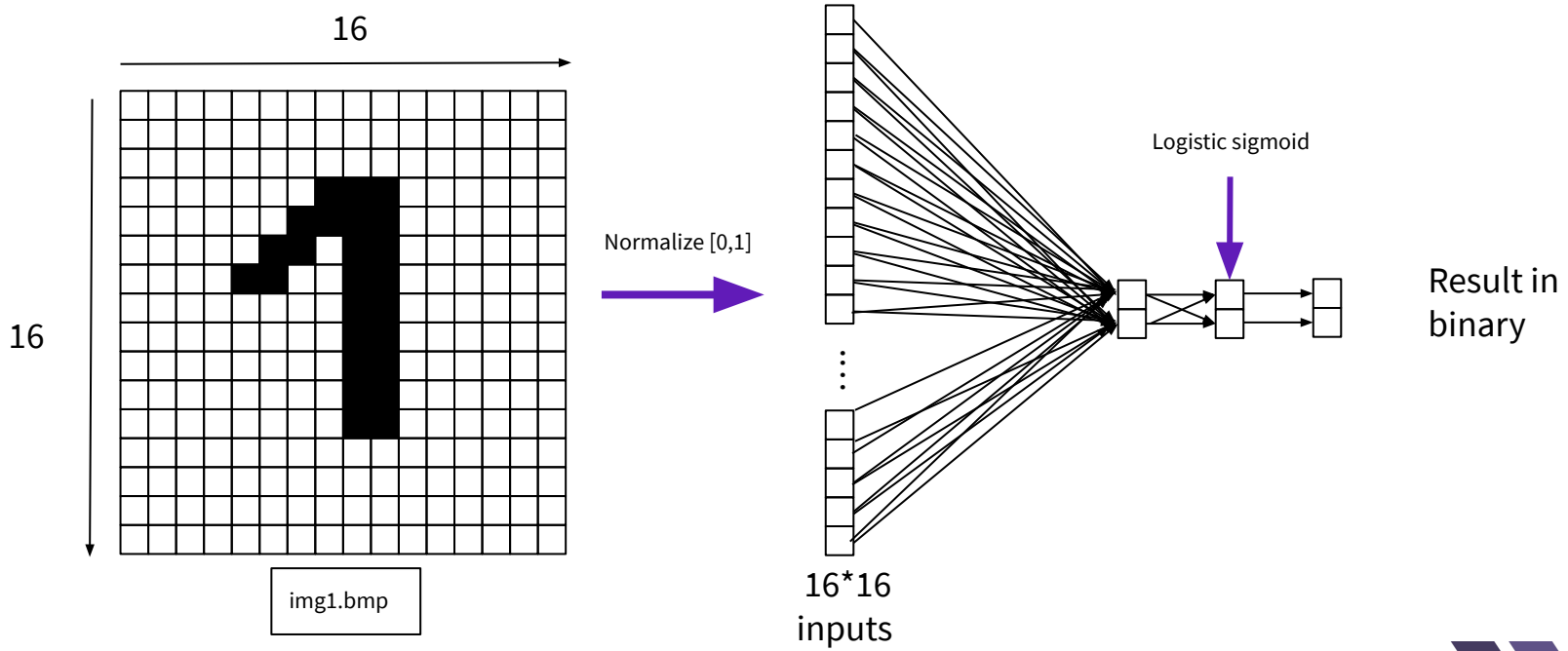


autoencoder.dat

```
[([0.0,0.0,0.0], [0.0,0.0,0.0])
,([1.0,0.0,0.0], [1.0,0.0,0.0])
,([0.0,1.0,0.0], [0.0,1.0,0.0])
,([0.0,0.0,1.0], [0.0,0.0,1.0])]
```

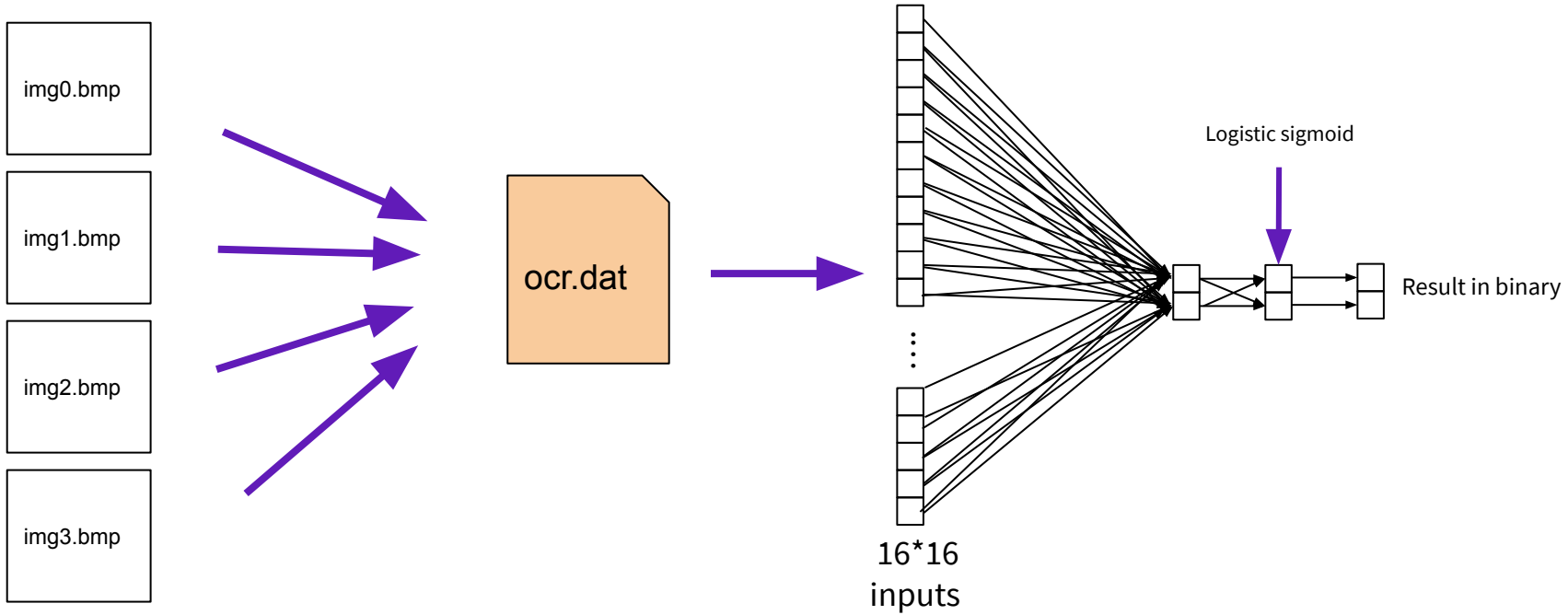
RESULTS/DEMO

Demo Digit GraySCale Recognition Net : how it works



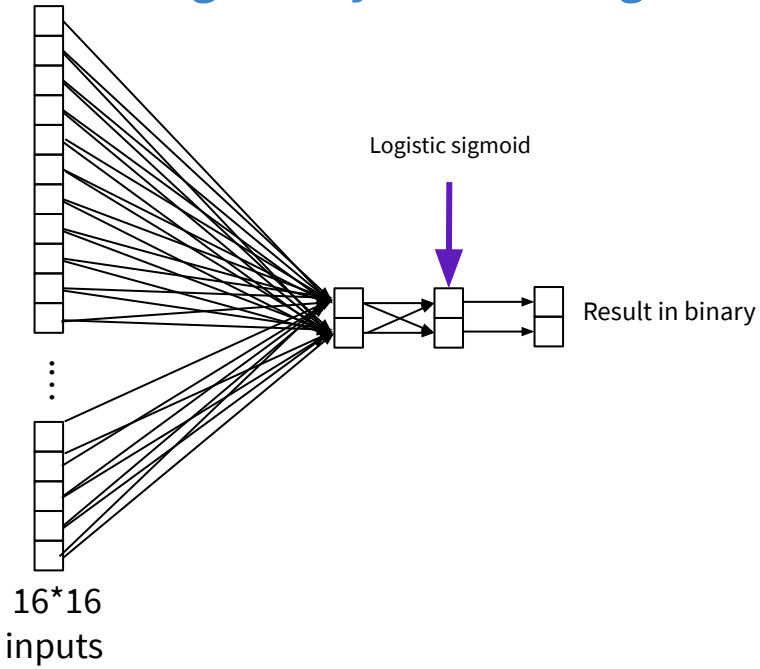
RESULTS/DEMO

Demo Digit GraySCale Recognition Net : training data



RESULTS/DEMO

Demo Digit GraySCale Recognition Net : performance



20 510 bytes (49 KB on disk)

RESULTS/DEMO

K-means Algorithm : Parsing CSV dataset

77.73.191.136	2328926	Nigeria
80.78.120.66	99237	Iraq
41.242.140.0	49518	Rwanda
195.189.46.127	146669	Cyprus
149.11.26.14	390903	Greece
217.76.128.224	2264397	Portugal

Reprise :

By other data science enthusiast

More Exciting Example to implement

To help : documentation available

ensisa
École nationale supérieure
d'ingénieurs sud Alsace

THANK YOU

ANY QUESTIONS ?