X_1	X_2	X_3
1	2	3
4	5	6
5	6	6
5	7	2
5	8	3
6	6	2
6	6	1
.	.	.
.	.	.

batch size = 3

input ①

* Hadamard product = np.multiply(a, b) ⚠ pas dot product mais multiplication élément par élément

marche pour ① vecteur + matrice ② vecteur + vecteur

la multiplication matricielle conserve le nbr de lignes

m = batch size

* **forward** : $\hat{Y}_{m,k} = \sigma\left[\overbrace{X_{m,d}}^{z_{m,k}} \cdot W_{i,k}\right]$ → formule générale

↳ pour un perceptron : $Y_{m,1} = X_{m,i} \; W_{i,1}$ → 1 output

* **activation** : Linéaire : $\sigma(x) = x \quad \forall x$

m : batch size ($m \leq$ nbr de lignes dataset)

* **loss function** : (ici : MSE)

↳ formule générale : (régression) output = scalaire $\frac{1}{m} \sum_{i=1}^{m} \left(Y_{i,1} - \hat{Y}_{i,1}\right)^2$

$\frac{1}{m} \text{sum}\left(\begin{bmatrix} \Delta y_{1,1} \\ \Delta y_{2,1} \\ \Delta y_{3,1} \end{bmatrix}^{\top} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right)$ = un scalaire

* **update** :

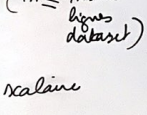
* weights = weights $- \eta \nabla_w$

* bias = bias $- \eta$ bias

* **backpropagation**

comment le cost function change (direction à prendre) en fonction des w

① chain rule sur perception : $\frac{\partial E}{\partial W_{i,1}} = \frac{\partial E}{\partial A} * \frac{\partial A}{\partial \sigma} * \frac{\partial \sigma}{\partial z}$



② équations perception :

pour notre perception

→ activation function : $f : x \longrightarrow x \Rightarrow f'(x) = 1$

→ $\frac{\partial E}{\partial A} = \frac{\partial E}{\partial A_1} = \frac{\partial MSE}{\partial A_1} = \frac{\partial \frac{1}{2m} \sum_{i=1}^{m} (Y_{i,1} - \hat{Y}_{i,1})^2}{\partial A} = -\frac{1}{2m} \sum_{i=1}^{m} \left(Y_{i,1} - \hat{Y}_{i,1}\right)$

→ $\frac{\partial A}{\partial z} = \sigma'(x) = 1$

→ $\frac{\partial z}{\partial W} = A_{(-1)} = X_{1,i}$

biais

$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial A} * \frac{\partial A}{\partial z} * \frac{\partial z}{\partial b}$ → 1

$z = AX + b \Rightarrow \frac{\partial z}{\partial b} = 1$