Réseaux de neurones et apprentissage profond

Une brève introduction, partie 1

E. Viennet

L2TI
Université Paris 13

2018-2019

Plan du cours

1. Introduction au Deep Learning
   - Quelques applications au traitement d'images ou de vidéos
   - Principe général
   - Histoire du connexionnisme
2. Perceptrons multi-couches
   - Exemples en 2 dimensions
3. Apprentissage par descente du gradient

Plan

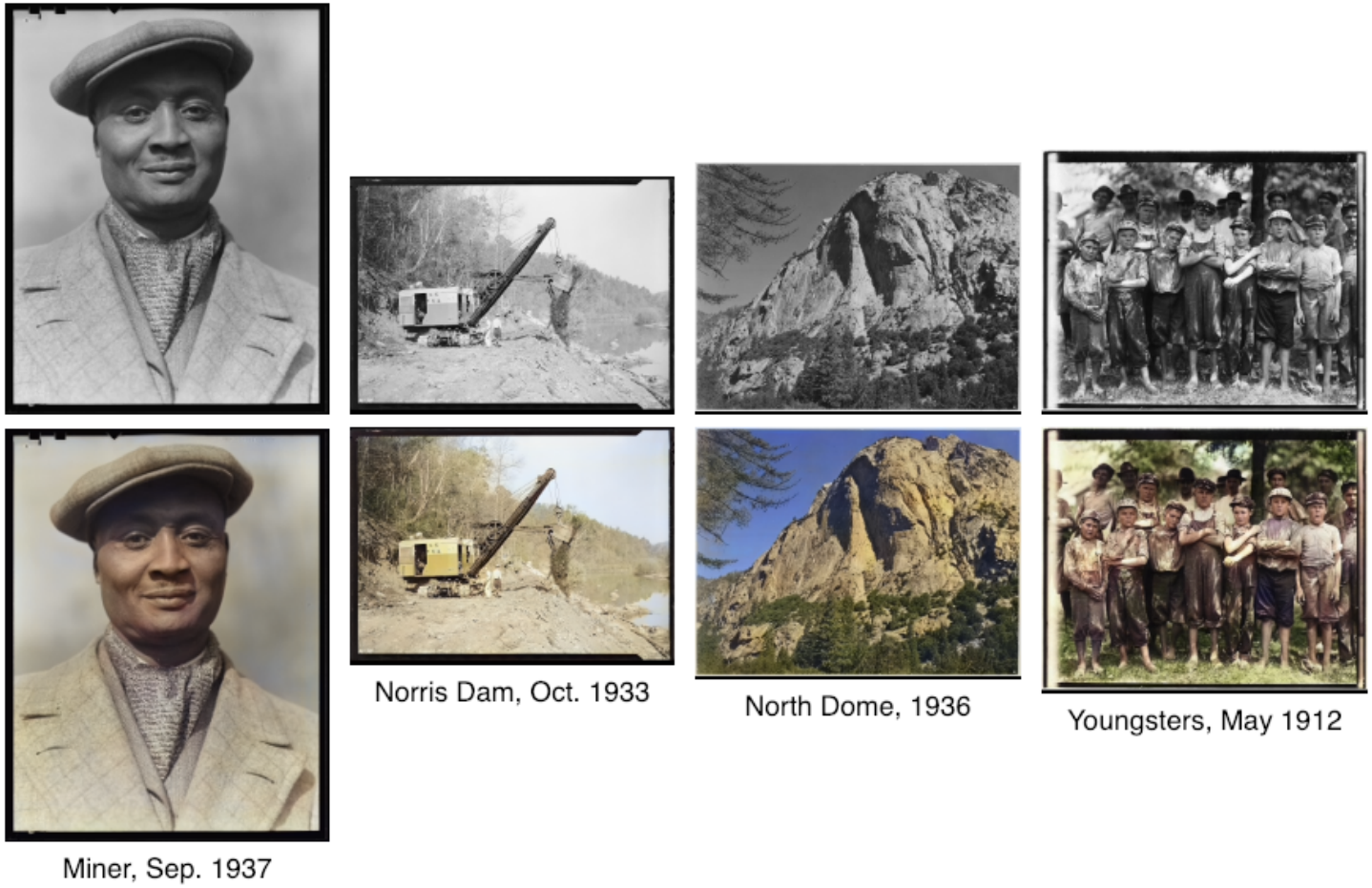
1. Introduction au Deep Learning
   - Quelques applications au traitement d'images ou de vidéos
   - Principe général
   - Histoire du connexionnisme
2. Perceptrons multi-couches
   - Exemples en 2 dimensions
3. Apprentissage par descente du gradient

Introduction au Deep Learning

- Les méthodes d'apprentissage de réseaux de neurones artificiels, *deep learning* connaissent depuis quelques années un succès fulgurant, à tel point qu'elles sont souvent confondues avec l'Intelligence Artificielle.
- Ces méthodes sont particulièrement bien adaptées au traitement des signaux et images, lorsqu'on dispose de beaucoup de données.

Coloriage

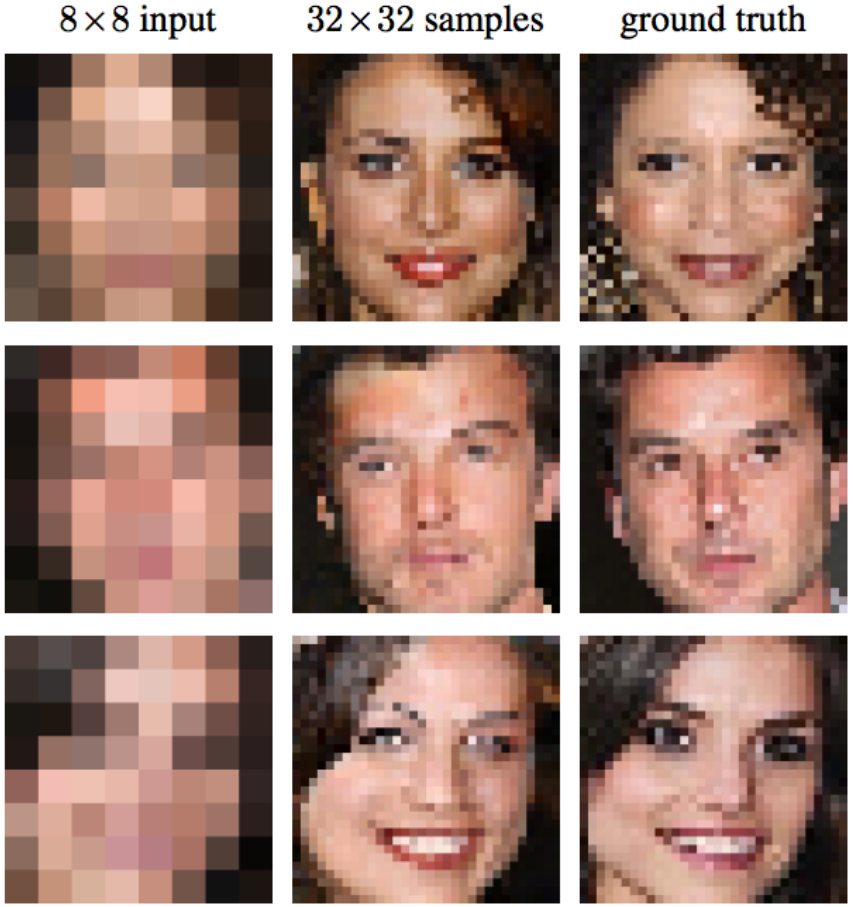
image en niveaux de gris $\rightsquigarrow$ image couleur



Source: S. Lizuka et al., 2017 https://arxiv.org/abs/1702.00783

Super-résolution

image basse résolution (ici 8x8 pixels)
$\rightsquigarrow$ image haute résolution (ici 32x32)



Source: R. Dahl et al., SIGGRAPH 2016 http://iizuka.cs.tsukuba.ac.jp/projects/colorization/en/

Estimation de la pose

image de scène $\rightsquigarrow$ squelette numérique

https://www.youtube.com/watch?v=pW6nZXeWlGM

Source: Z. Cao et al., Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields (2016)
https://arxiv.org/abs/1611.08050

Everybody Dance Now

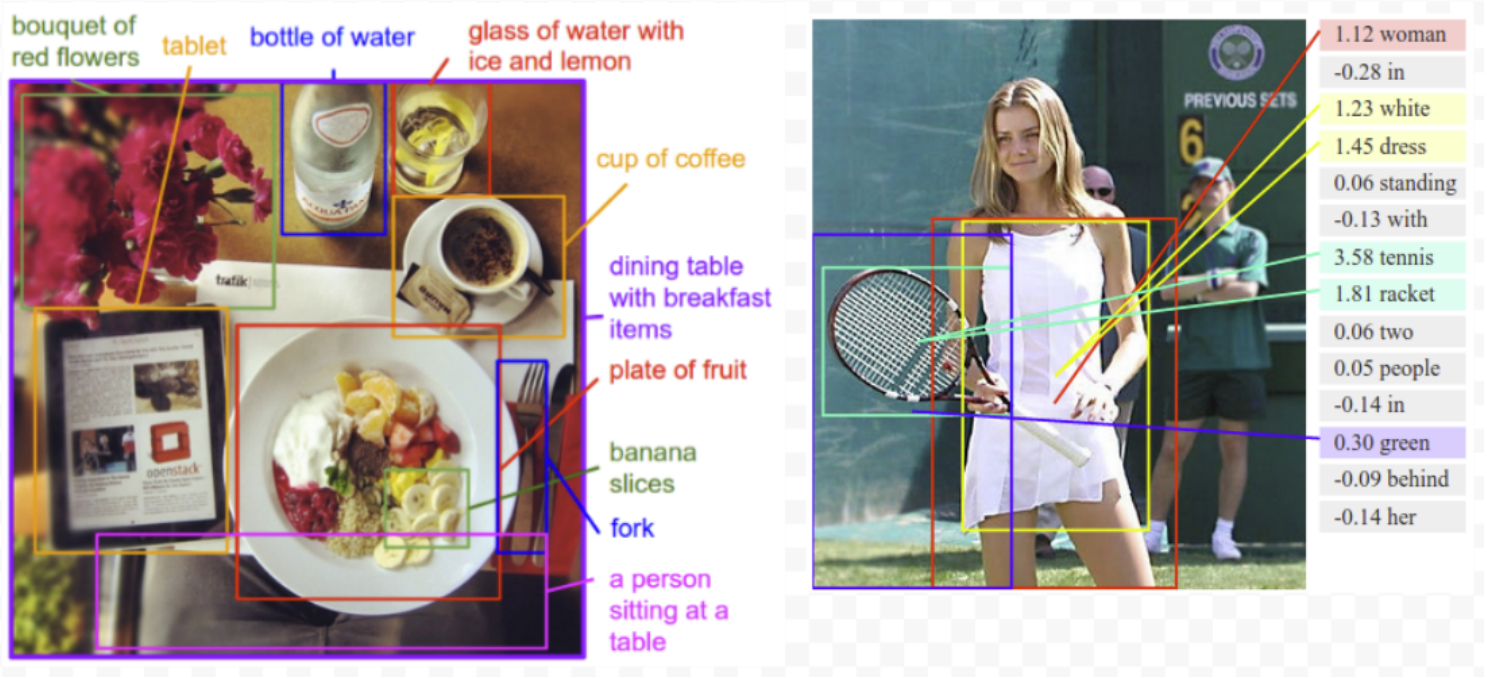
Vidéo de synthèse : faire danser une autre personne

https://www.youtube.com/watch?v=PCBTZh41Ris

Source: C. Chan (2018) https://arxiv.org/pdf/1808.07371.pdf

Décrire des photos

Décrire le contenu d'une image avec des mots ou des phrases

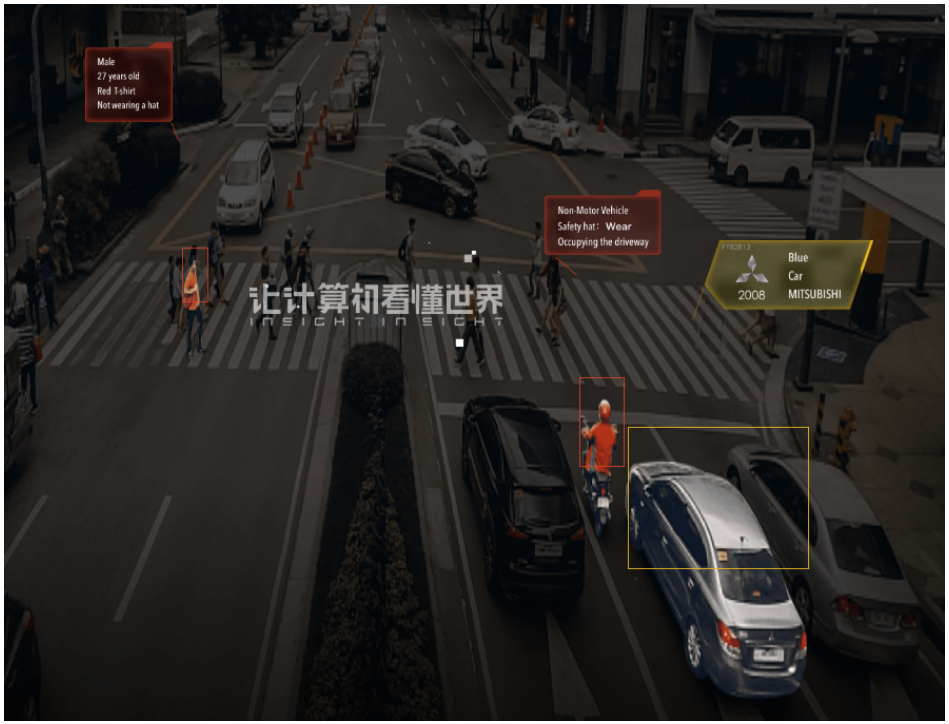


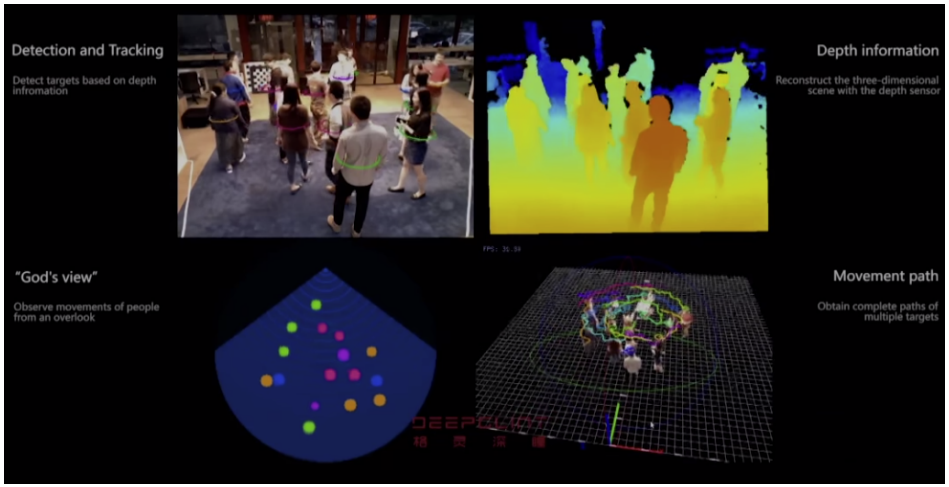
https://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/

Source: Karpathy et al. (2015) http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Karpathy_Deep_Visual-Semantic_Alignments_2015_CVPR_paper.pdf

Analyse de comportement, surveillance

Analyser des comportements en temps réel



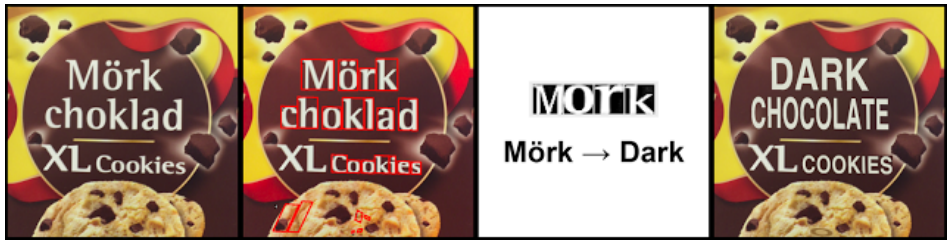


https://www.youtube.com/watch?v=xhp47v5OBXQ

Source: http://www.deepglint.com, CVPR 2016

Traduire (sur des images)

Détecter, traduire et remplacer du texte dans des images

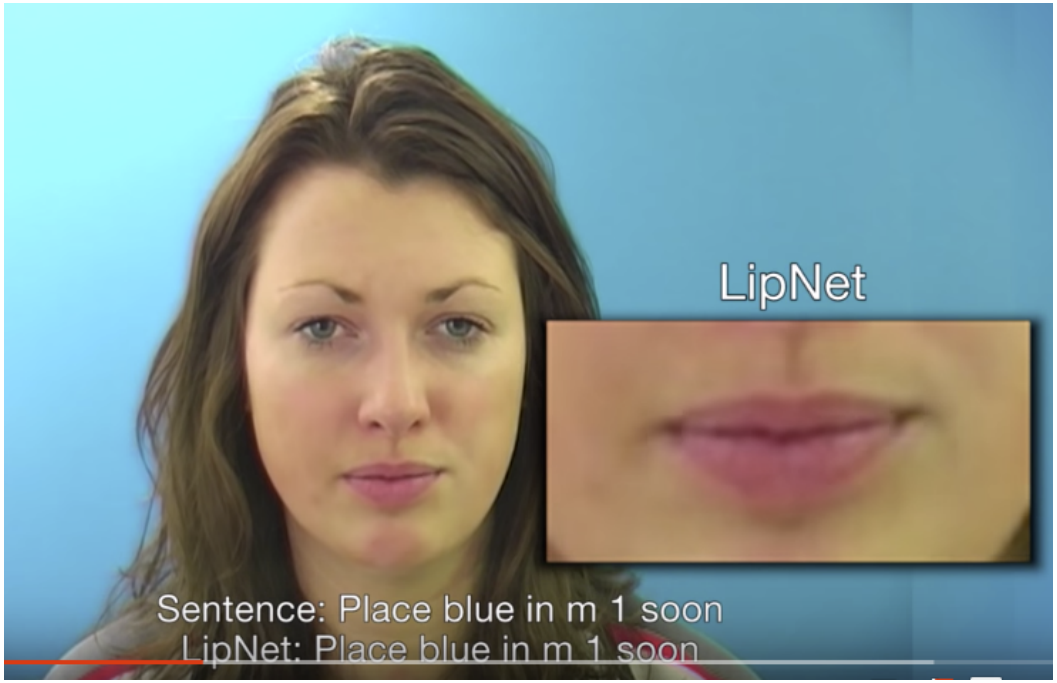


Sur la traduction automatique de texte, voir `https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html`

Source: https://ai.googleblog.com/2015/07/how-google-translate-squeezes-deep.html

Lire sur les lèvres

LipNet arrive à lire 93% des phrases,
que des humains entrainés ne lisent qu'à 52%



https://www.youtube.com/watch?v=fa5QGremQf8

Source: Assael et al. (2016) https://arxiv.org/pdf/1611.01599.pdf

Peinture artistiques

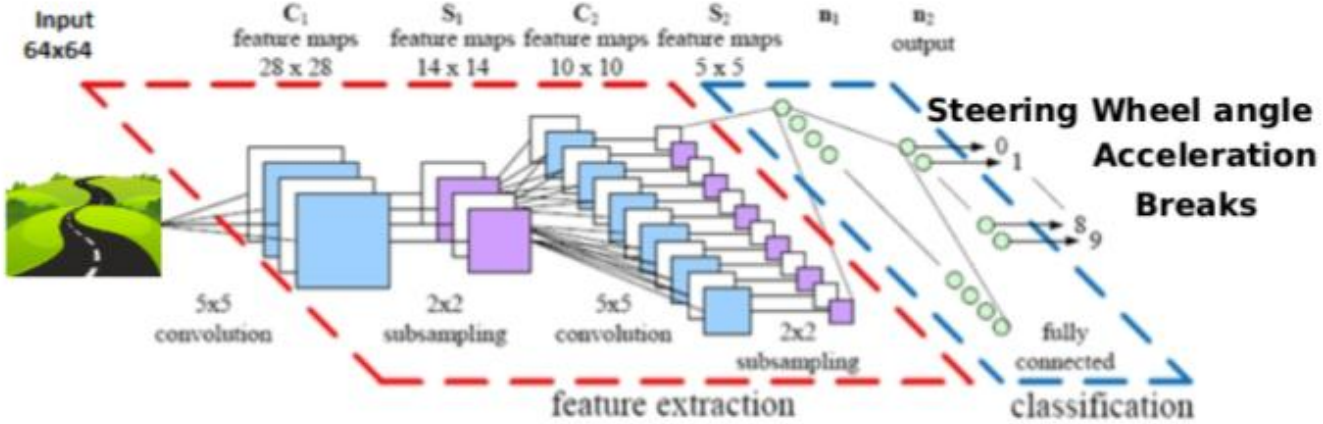
A partir d'une photo, générer des peintures selon un « style »

https://deepart.io/

http://genekogan.com/works/style-transfer/

Source: Gatys et al. (2016) http://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

Véhicules autonomes

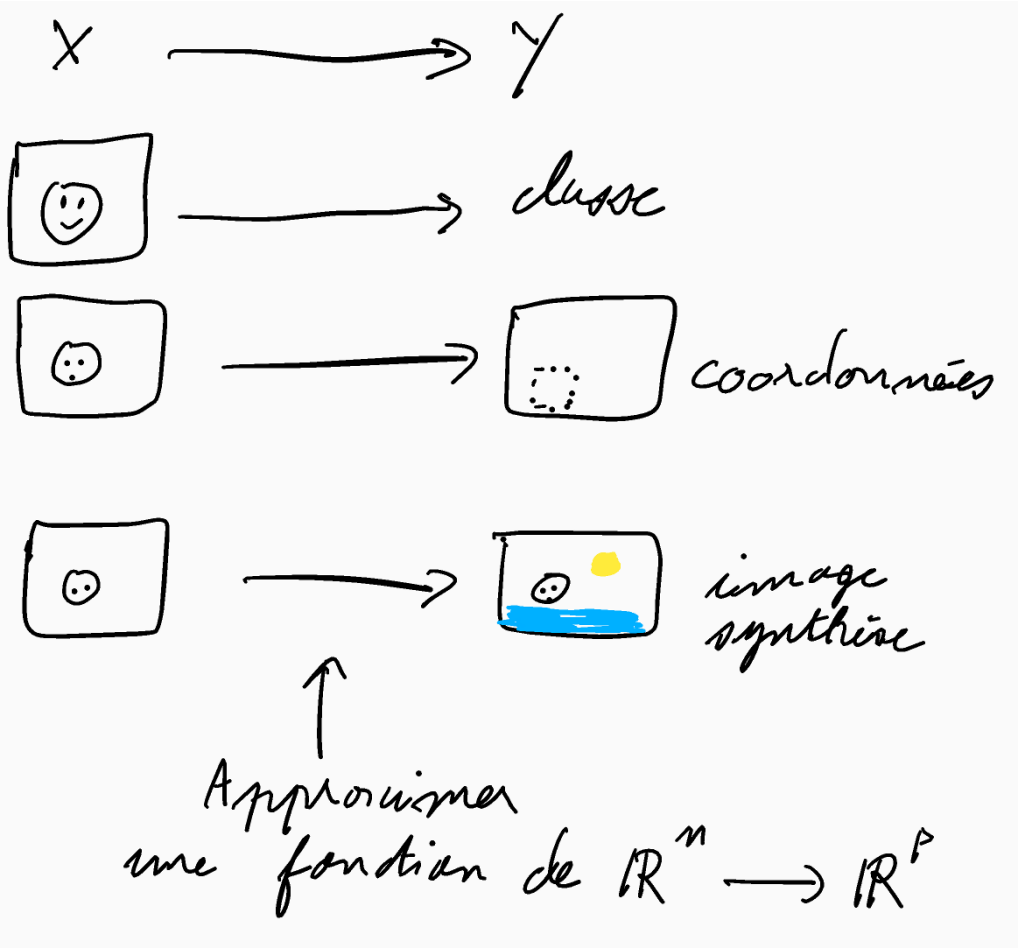


Des progrès rapides et des performances impressionnantes

- problèmes difficiles, dont on a longtemps pensé la solution lointaine (vision, jeux de Go, etc.)
- les images :
  - problèmes en grande dimension ;
  - conditions très variables : pose, éclairage,déformations, occultations.
- dans de nombreux cas, les systèmes artificiels ont déjà de meilleures performances que les humains.

Autres démonstrations : `https://ml-showcase.com`

Approximer des fonctions à partir d'exemples



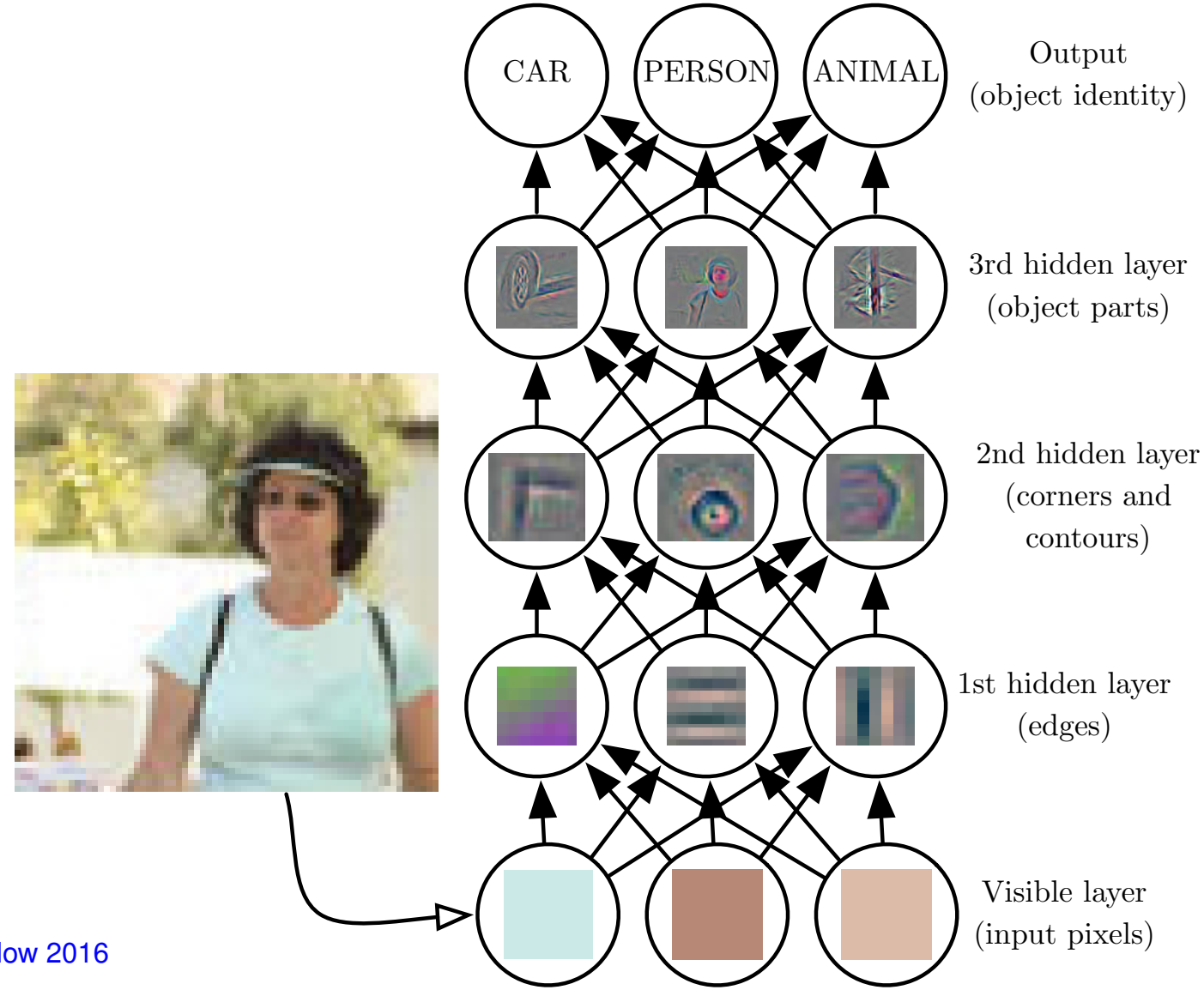
Approximer des fonctions à partir d'exemples

$$X \longrightarrow Y = f(x)$$

$$Y = f_n(X_n) = f_n(f_{n-1}(X_{n-1}))$$
$$= f_n(f_{n-1}(f_{n-2}(X_{n-2})))$$

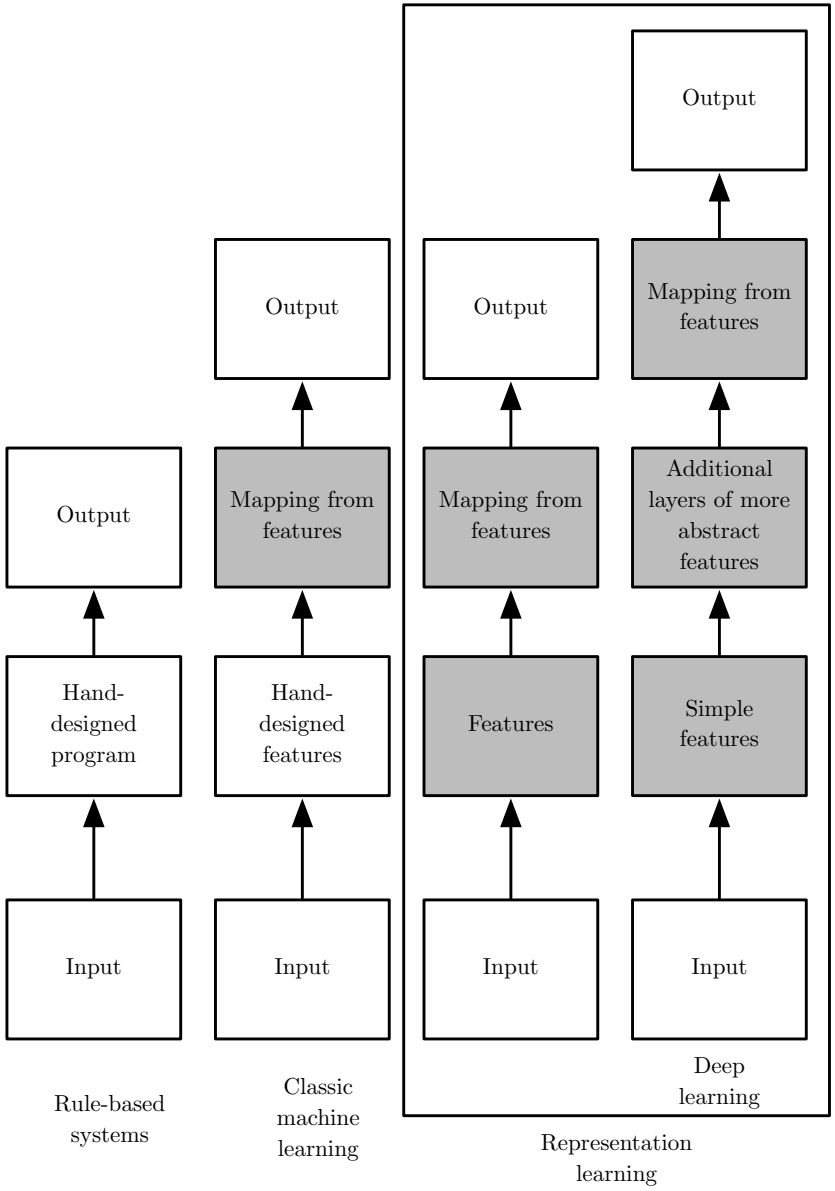
Composition de fonctions simples
comme des produits de matrices

Composer des fonctions : réseau multicouche profond



Source: Goodfellow 2016

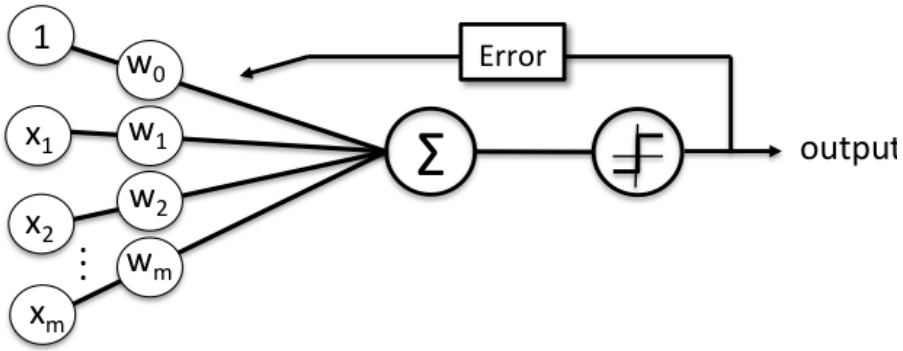
Approches de l'apprentissage



Source: Goodfellow 2016

1957 : le perceptron

Rosenblatt propose le perceptron pour la classification

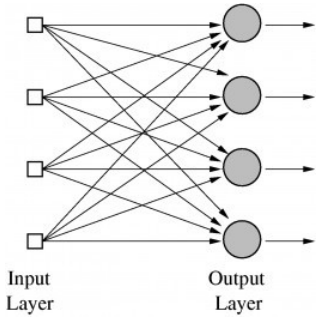


Schematic of a perceptron classifier.

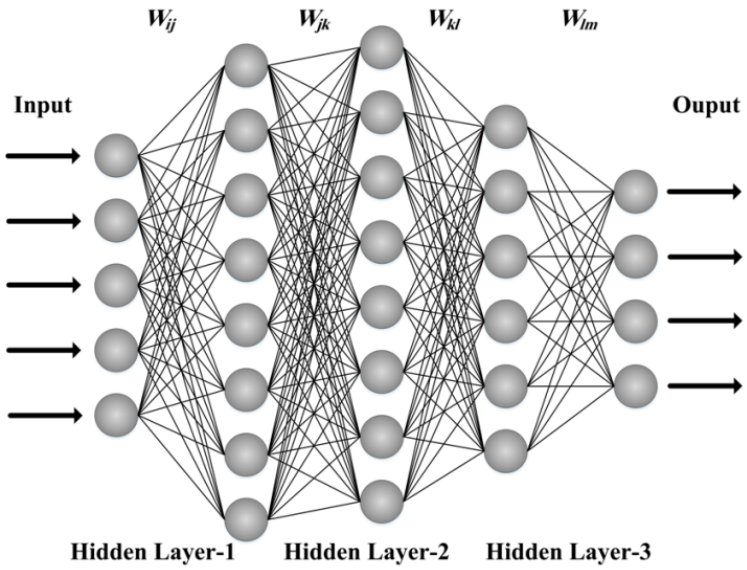
- un poids w_i par entrée x_i
- $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{sinon} \end{cases}$
- $\rightsquigarrow$ séparateur **linéaire** dans l'espace d'entrée

Années 80 : perceptrons multi-couches

- 1 perceptron : 1 décision
- si plusieurs classes (ex : reconnaissance de chiffres) : 1 cellule de sortie par classe



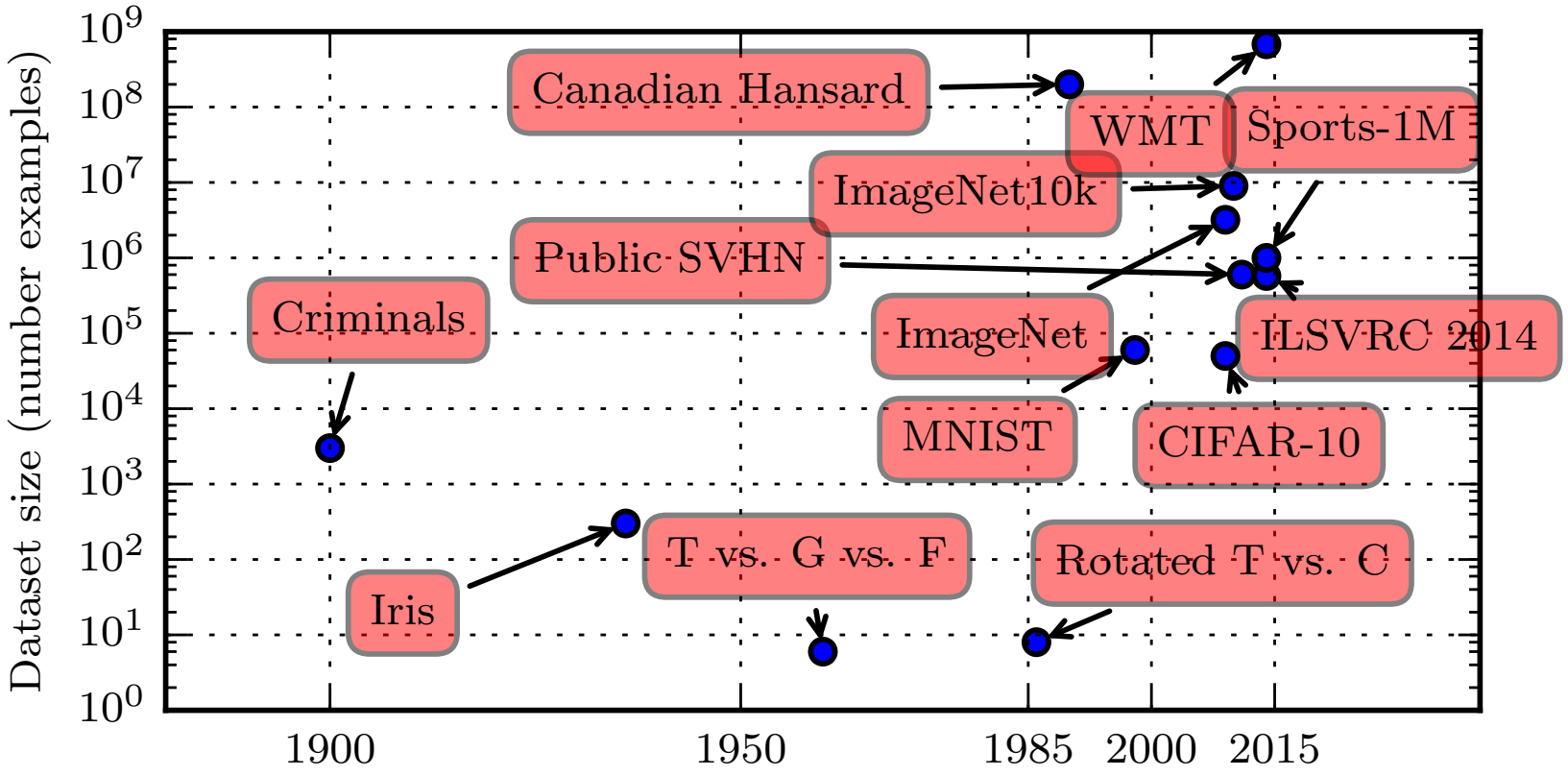
- enchaînement de plusieurs couches : perceptron multi-couche (MLP)
- on introduit des non linéarités entre les couches



Années 90 :

- Avancées en théorie et applications de l'apprentissage statistique (SVM, approches bayesiennes, ...) ;
- Premières applications industrielles des réseaux de neurones (LeNet, codes postaux).

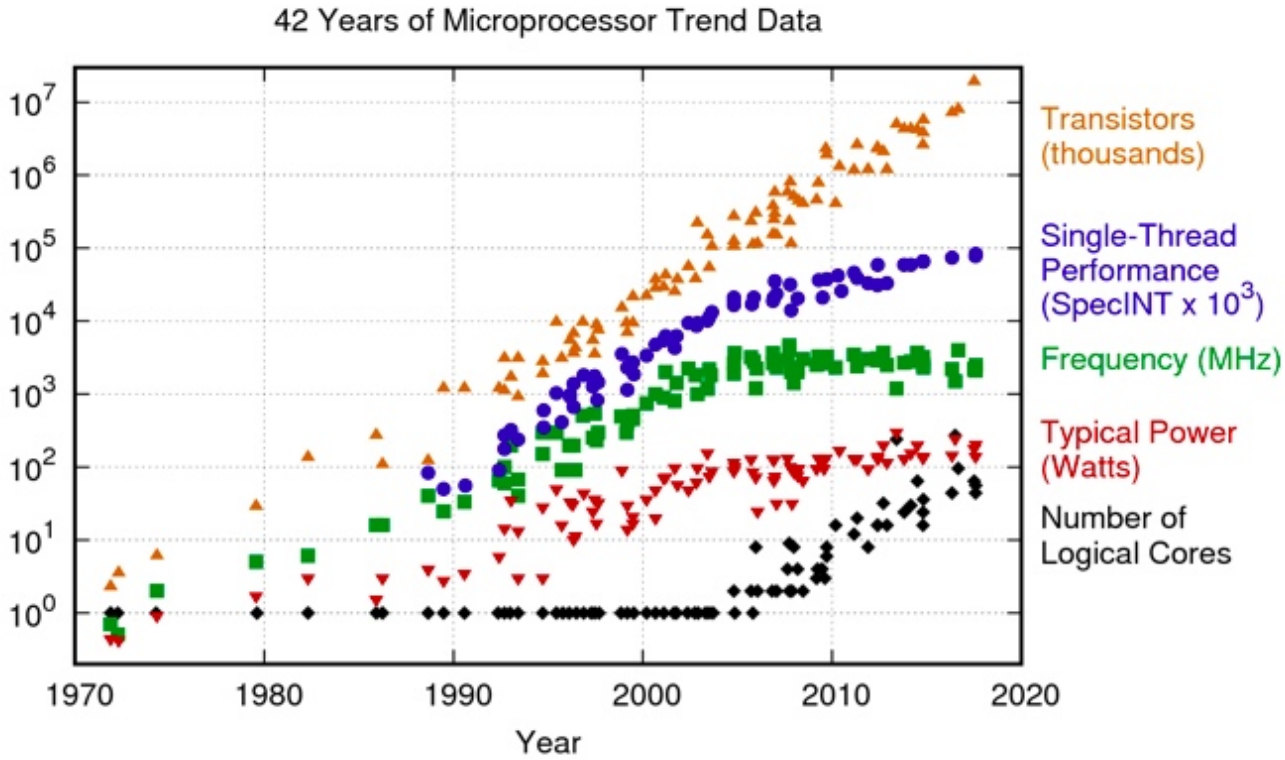
Années 2000-2019 : croissance de la taille des jeux de données



Source: Goodfellow 2016

Années 2000-2019 : croissance de la puissance de calcul

Loi de Moore depuis 50 ans, apparition des GPUs, des ASICs



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

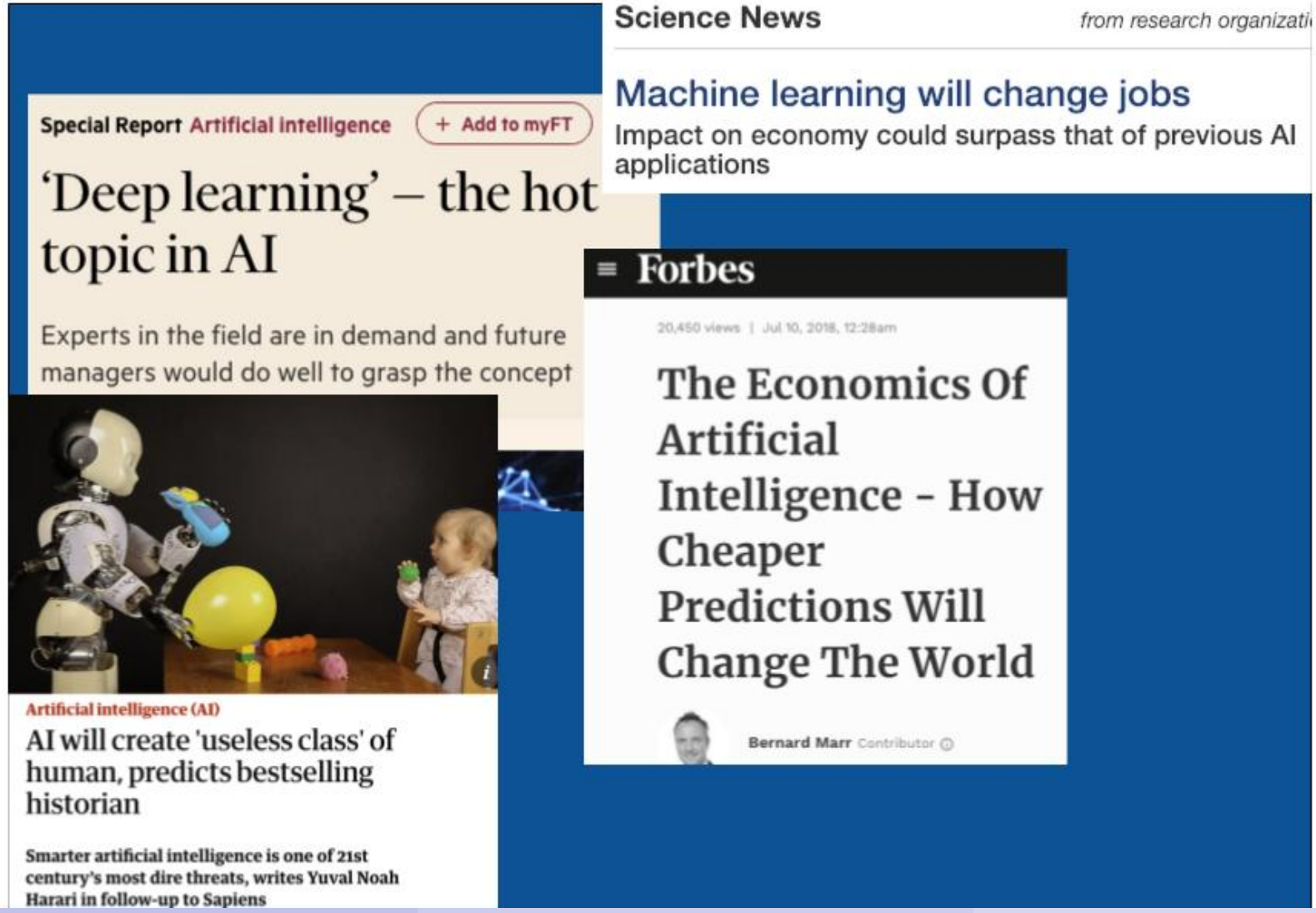
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

2010 : Deep learning

Hinton, Le Cun, Bengio, ...
ImageNet

Années 2010-2019 : nombreuses applications

forts enjeux économiques et sociétaux

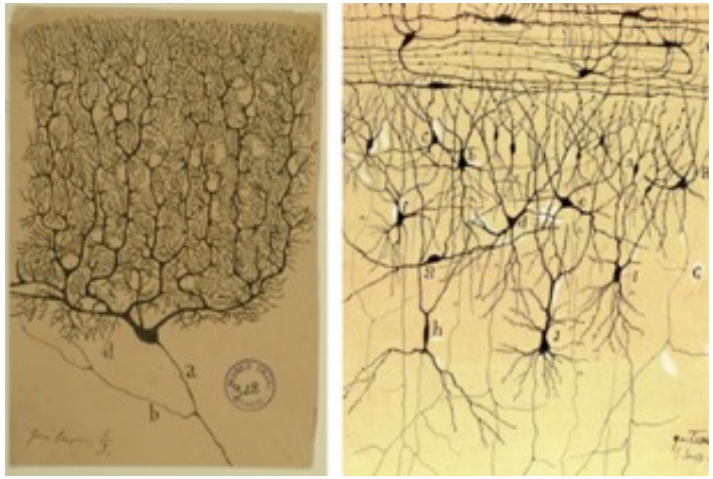


Plan

1. Introduction au Deep Learning
   - Quelques applications au traitement d'images ou de vidéos
   - Principe général
   - Histoire du connexionnisme
2. Perceptrons multi-couches
   - Exemples en 2 dimensions
3. Apprentissage par descente du gradient

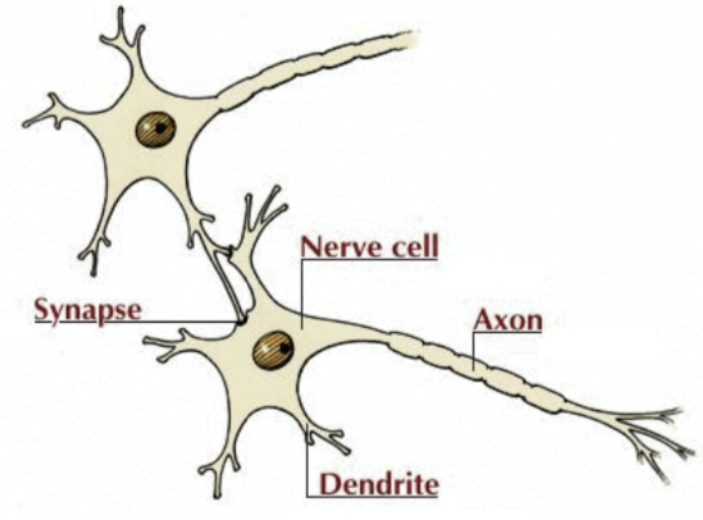
Inspiration biologique

- Si on veut construire une machine « intelligente », on peut s'inspirer du cerveau animal
- Le cerveau humain :
  - apprend continument
  - s'adapte à de nombreuses tâches
  - est très rapide
  - est massivement parallèle

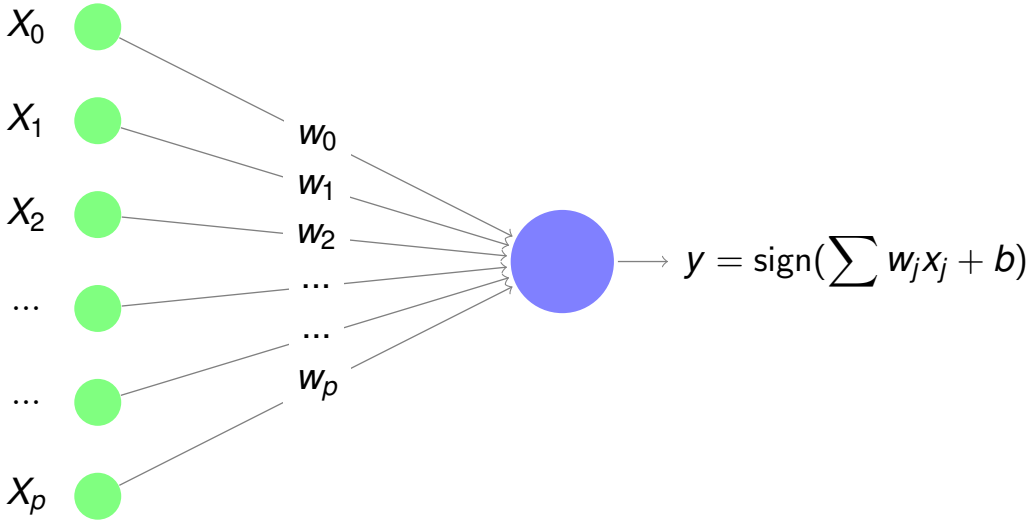


Source: http ://www.robotspacebrain.com/wp-content/uploads/2013/11/Santiago-Ramon-y-Cajal-Drawings.jpg

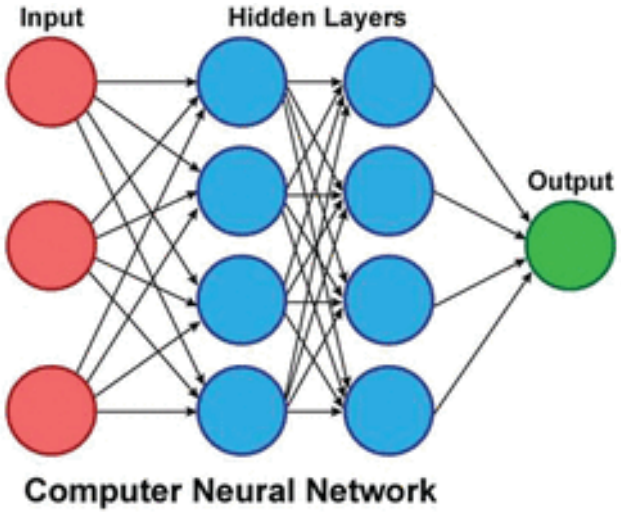
Qu'est-ce qu'un neurone ?

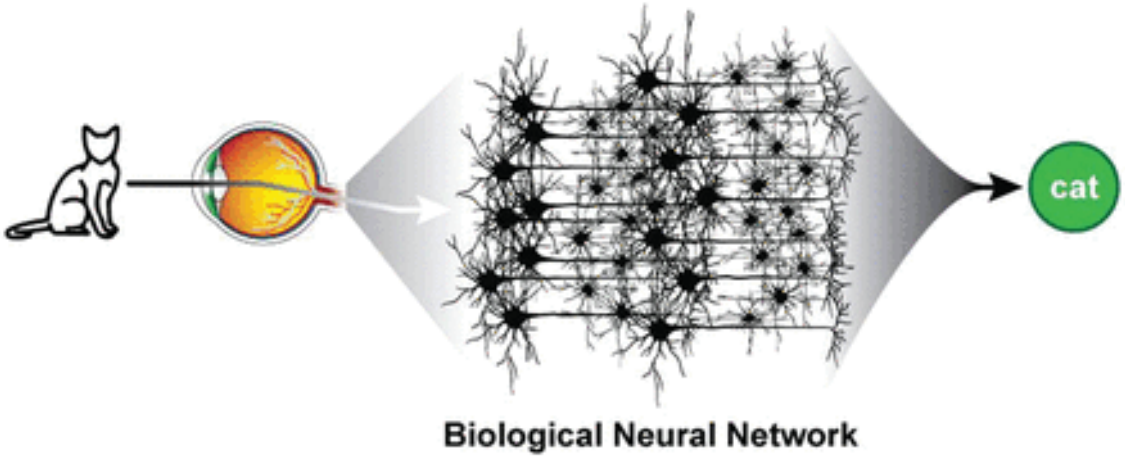


Modèle simplifié :



Anatomie d'un réseau multicouche (MLP)





Source: Deep Learning in Neuroradiology http://www.ajnr.org/content/early/2018/02/01/ajnr.A5543

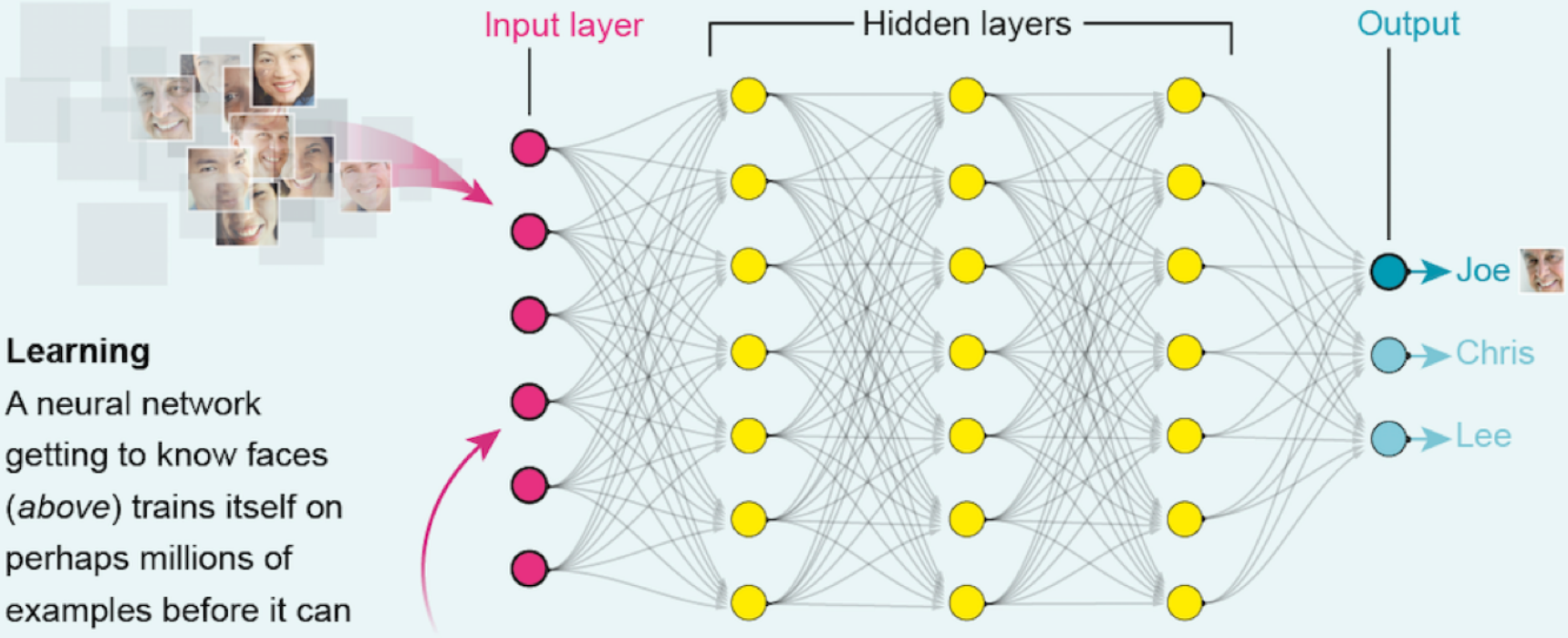
Perceptron multi-couches (MLP)

Les MLP sont des modèles puissants : on peut montrer qu'un MLP est un *approximateur universel* (Cybenko 1989) :

- toute fonction (d'un compact vers un compact) peut être approchée aussi bien que l'on veut par un MLP à trois couches, avec suffisamment de neurones sur la couche cachée et une sortie linéaire.

On peut donc résoudre tout problème de ML avec un MLP, *si on arrive à apprendre les poids...*

Anatomie d'un réseau multicouche (MLP)



Learning

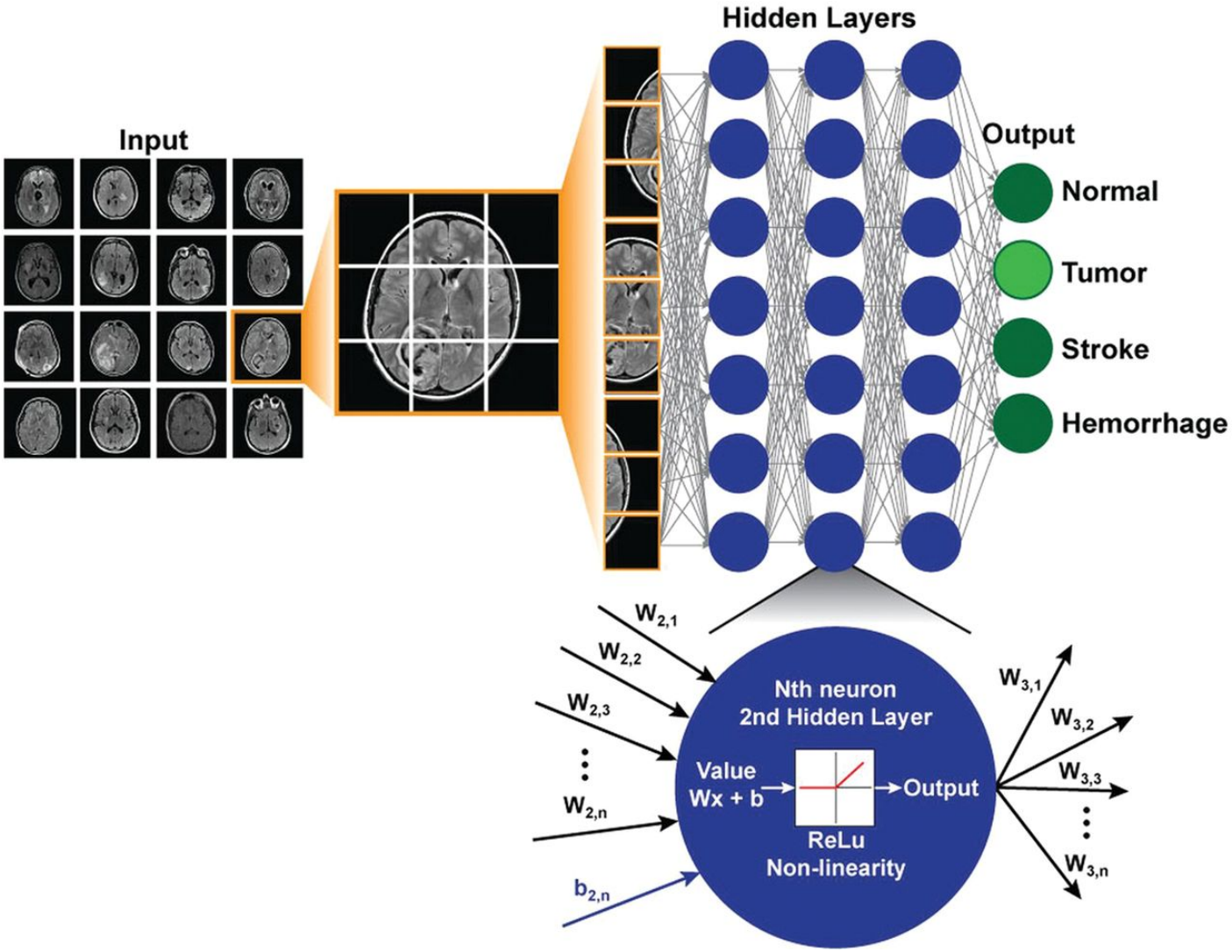
A neural network getting to know faces (*above*) trains itself on perhaps millions of examples before it can pick out an individual face from a crowd or a cluttered landscape.

Recognition

Input of a face into the network is analyzed at each layer before the network guesses correctly about its identity.

Each layer identifies progressively more complex features

Anatomie d'un réseau multicouche (MLP)



Source: Deep Learning in Neuroradiology http://www.ajnr.org/content/early/2018/02/01/ajnr.A5543

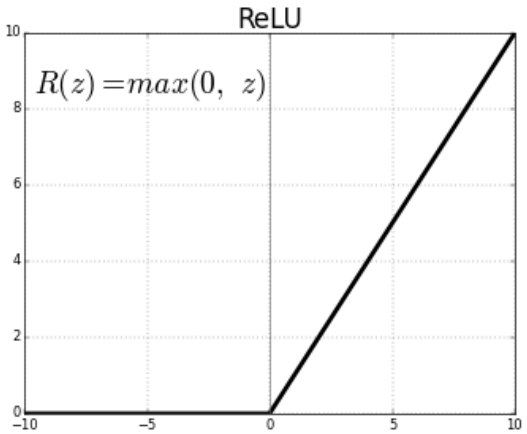
Neurone et fonction de transfert ReLu

Chaque neurone calcule son état :

$$y = f(W.X) = f(\sum_i w_i x_i)$$

où f est une fonction linéaire rectifiée (ReLu) :

$$f(s) = \begin{cases} s & \text{si } s > 0 \\ 0 & \text{sinon} \end{cases}$$



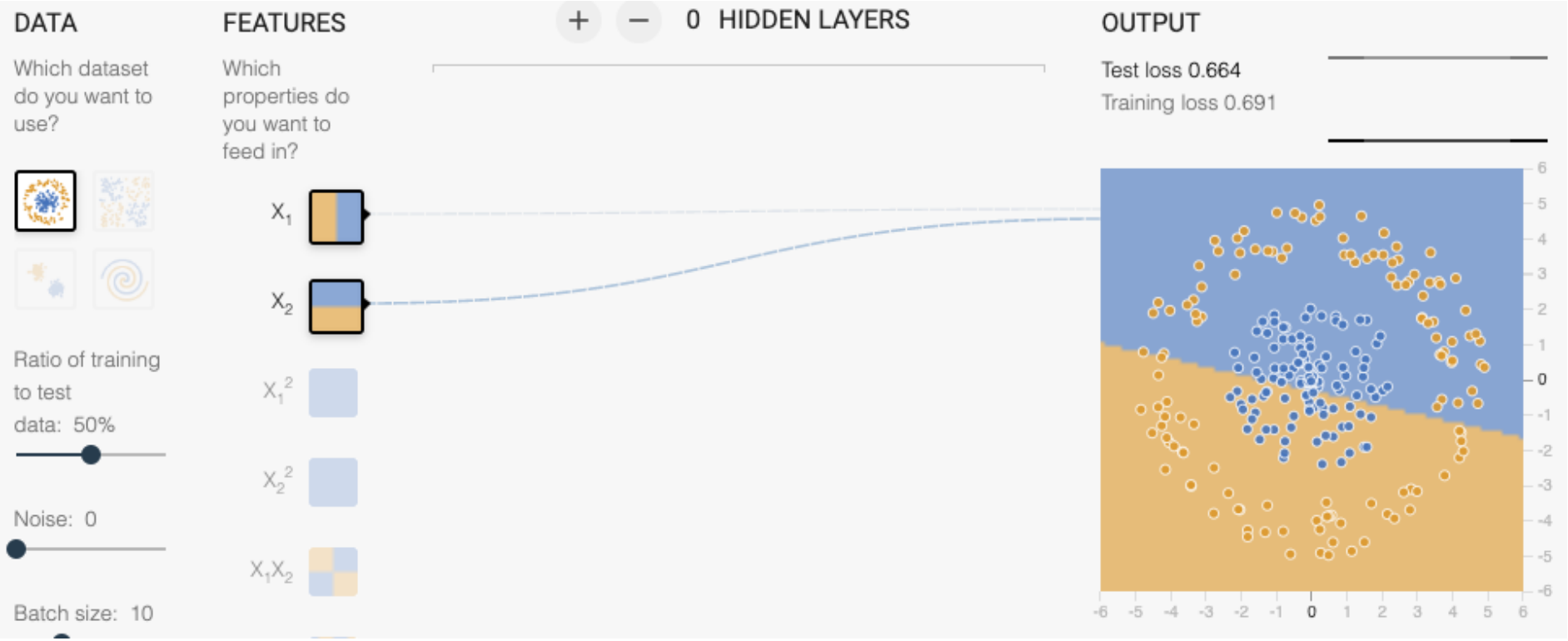
Exemples en 2 dimensions

Voir la demo de TensorFlow en JavaScript :

`http://playground.tensorflow.org`

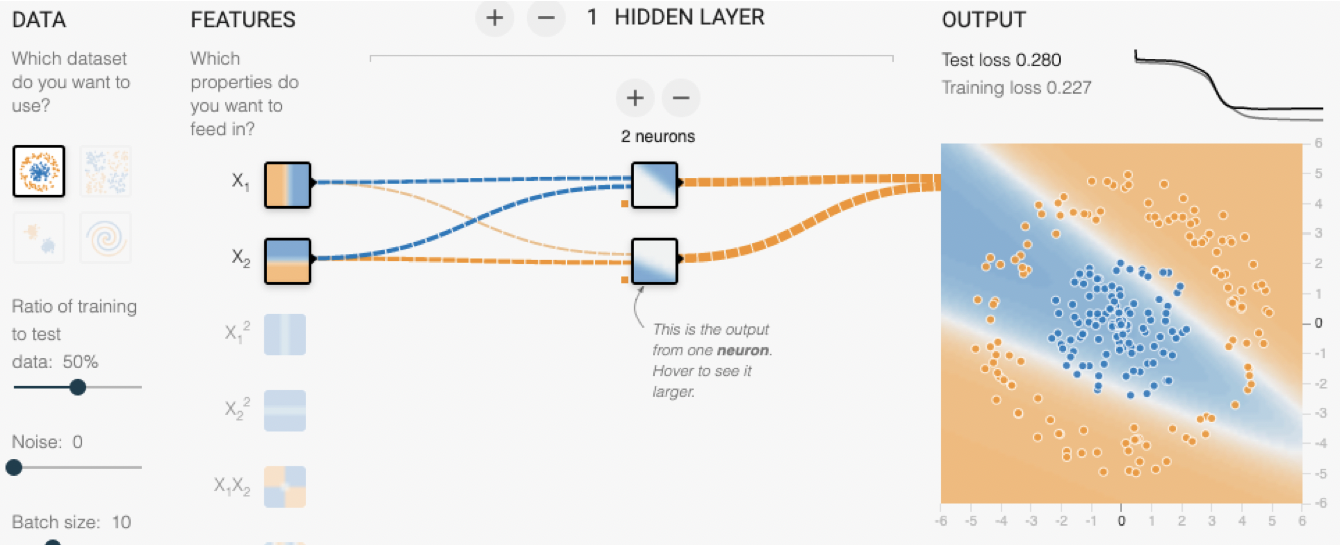
Exemple 2d : séparation linéaire dans le plan

Pas de couche cachée

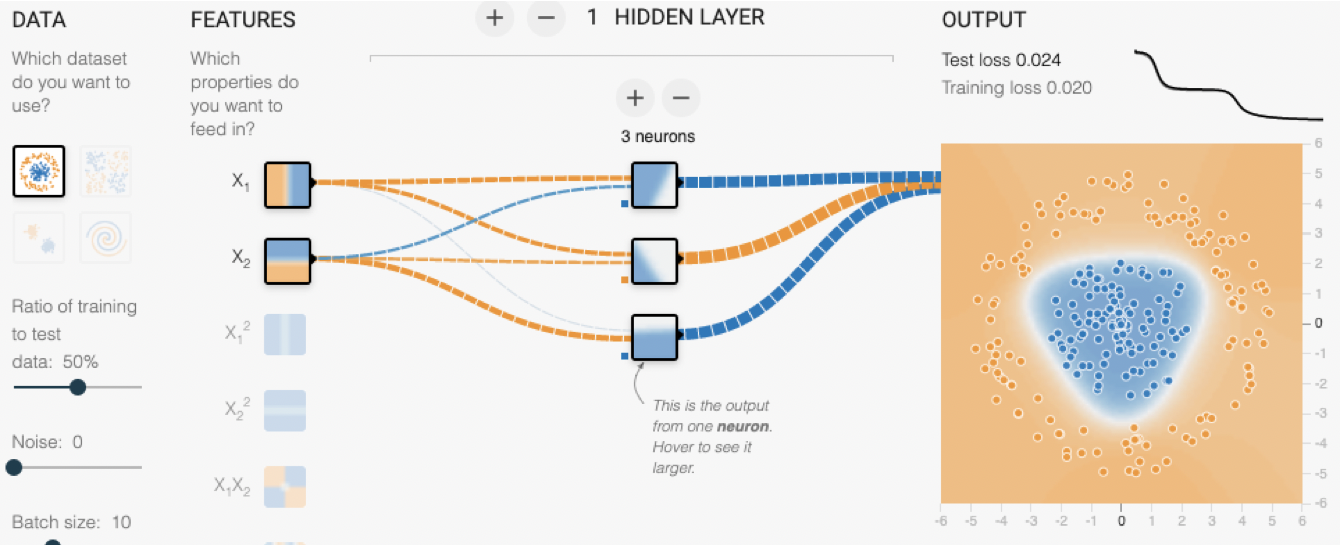


Exemple 2d : MLP à une couche cachée

One hidden layer
2 neurons

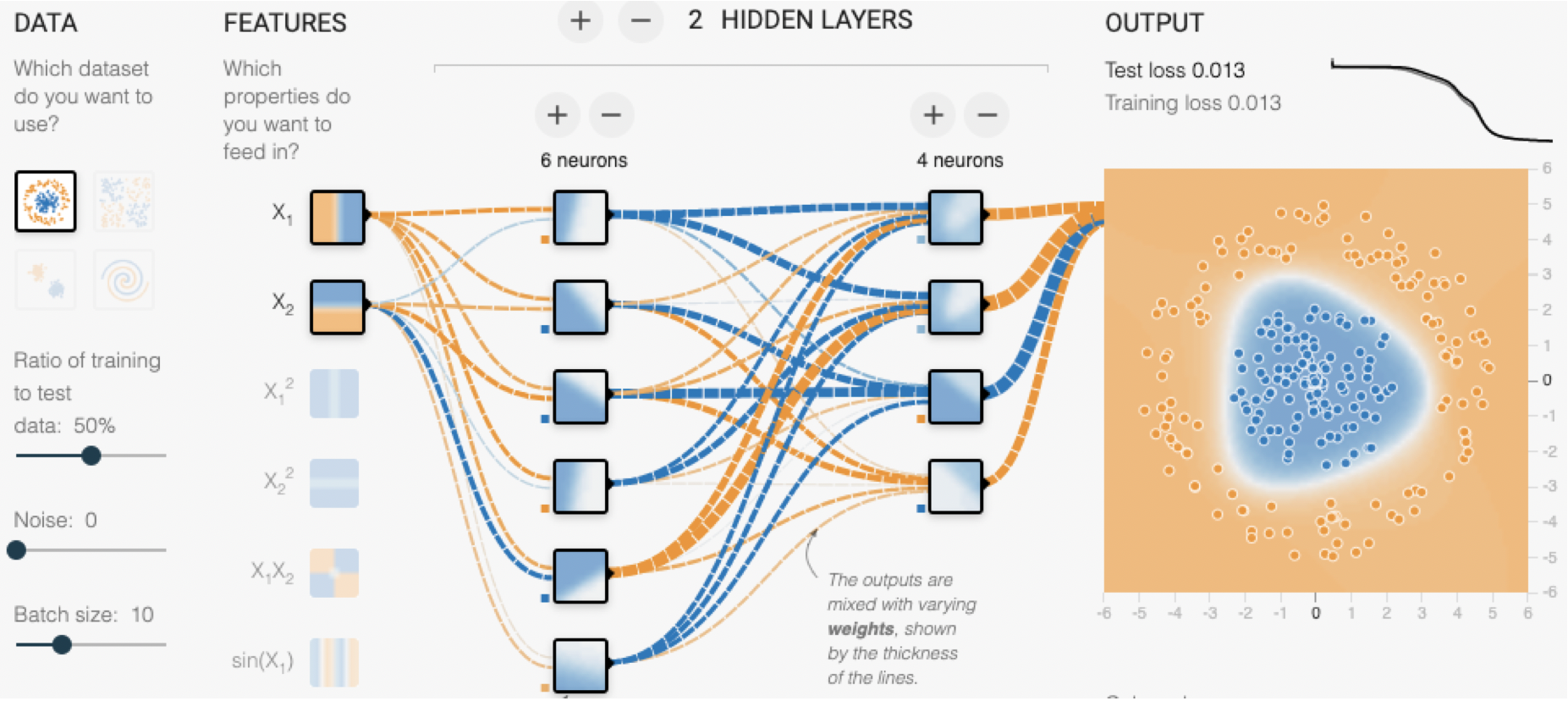


One hidden layer
3 neurons



Exemple 2d : MLP à deux couches cachées

Going deeper: 6 neurons on layer 1, 4 neurons on layer 2



Plan

1. Introduction au Deep Learning
   - Quelques applications au traitement d'images ou de vidéos
   - Principe général
   - Histoire du connexionnisme
2. Perceptrons multi-couches
   - Exemples en 2 dimensions
3. Apprentissage par descente du gradient

Apprentissage et optimisation

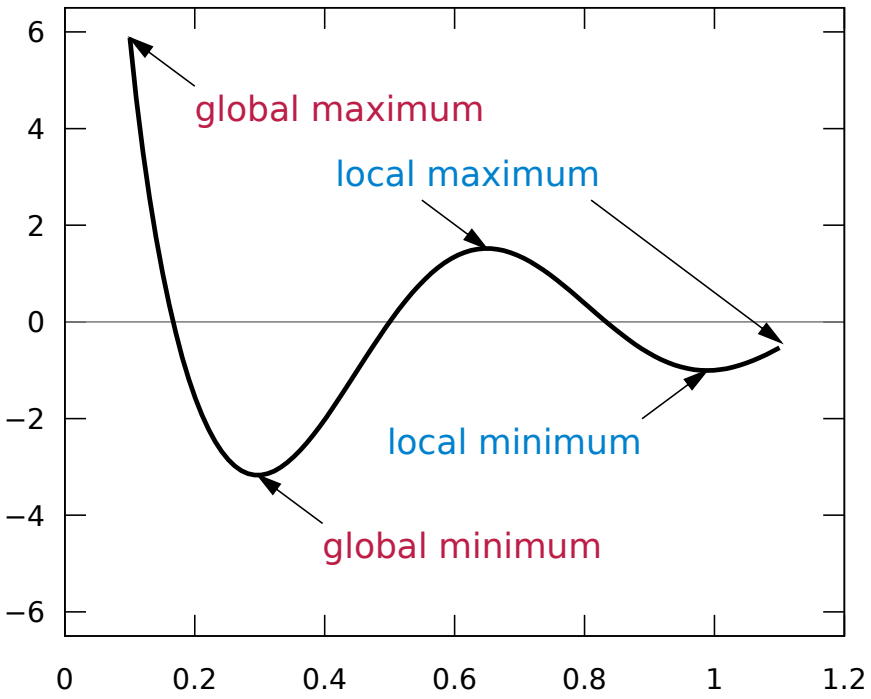
L'apprentissage consiste à trouver les bons paramètres w_i de toutes les connexions.
On cherche à minimer le *coût* (loss) L sur les exemples.
Exemples de coûts :

- Quadratique (MSE) :

$$L = \text{MSE} = \frac{1}{m} \sum_i (\hat{y}_i - y_i)^2$$

- Maximum de vraisemblance

Optimisation 1d



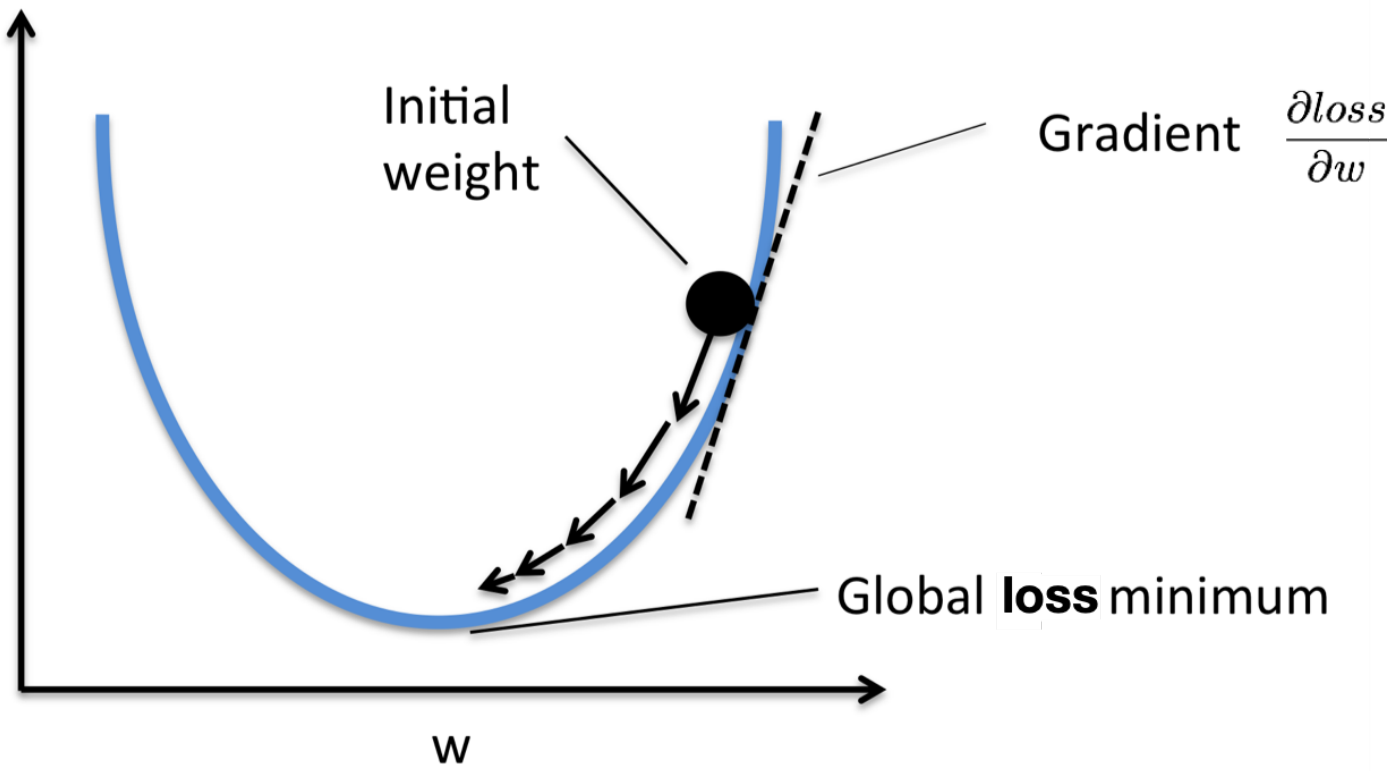
Source: figure Wikipedia

Exemple : optimisation MSE modèle linéaire

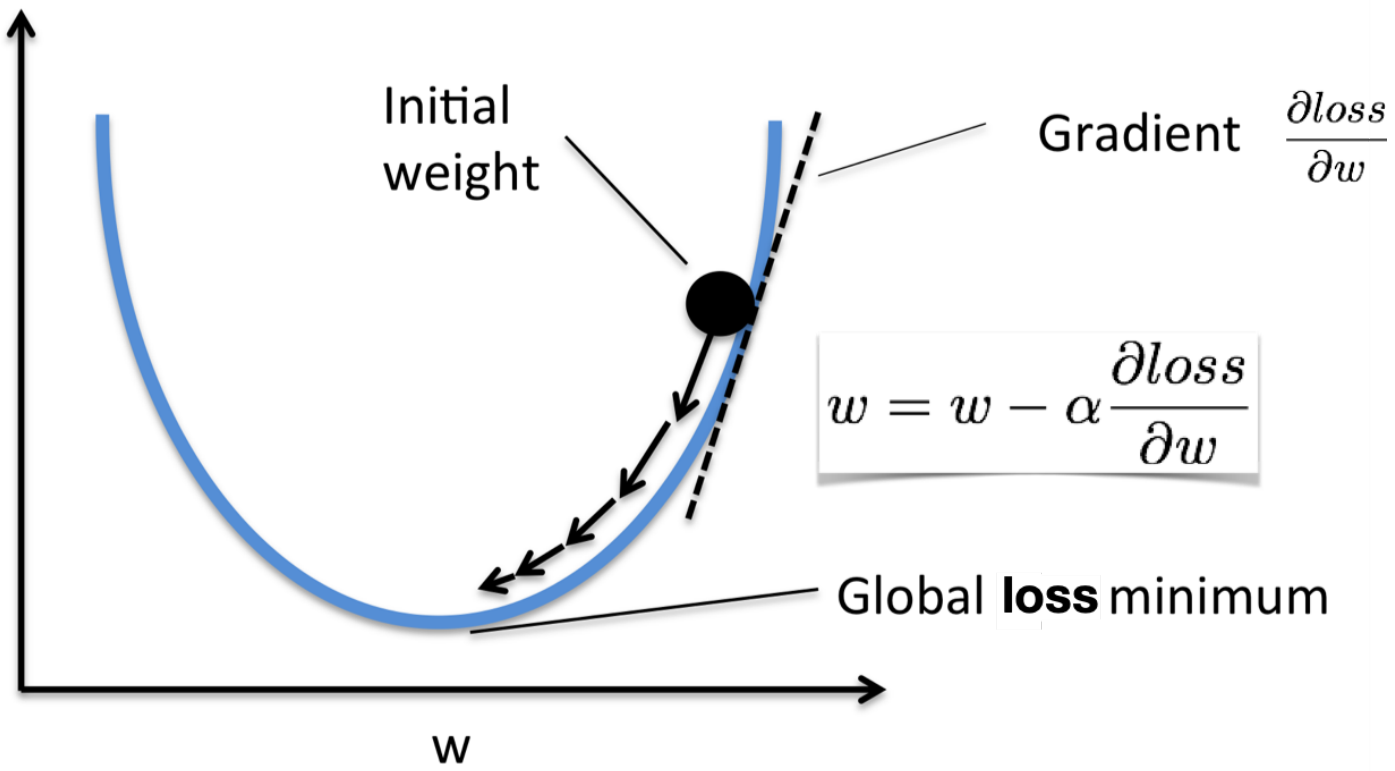
$$\hat{y} = w.x$$

$$\text{coût} = (\hat{y} - y)^2$$

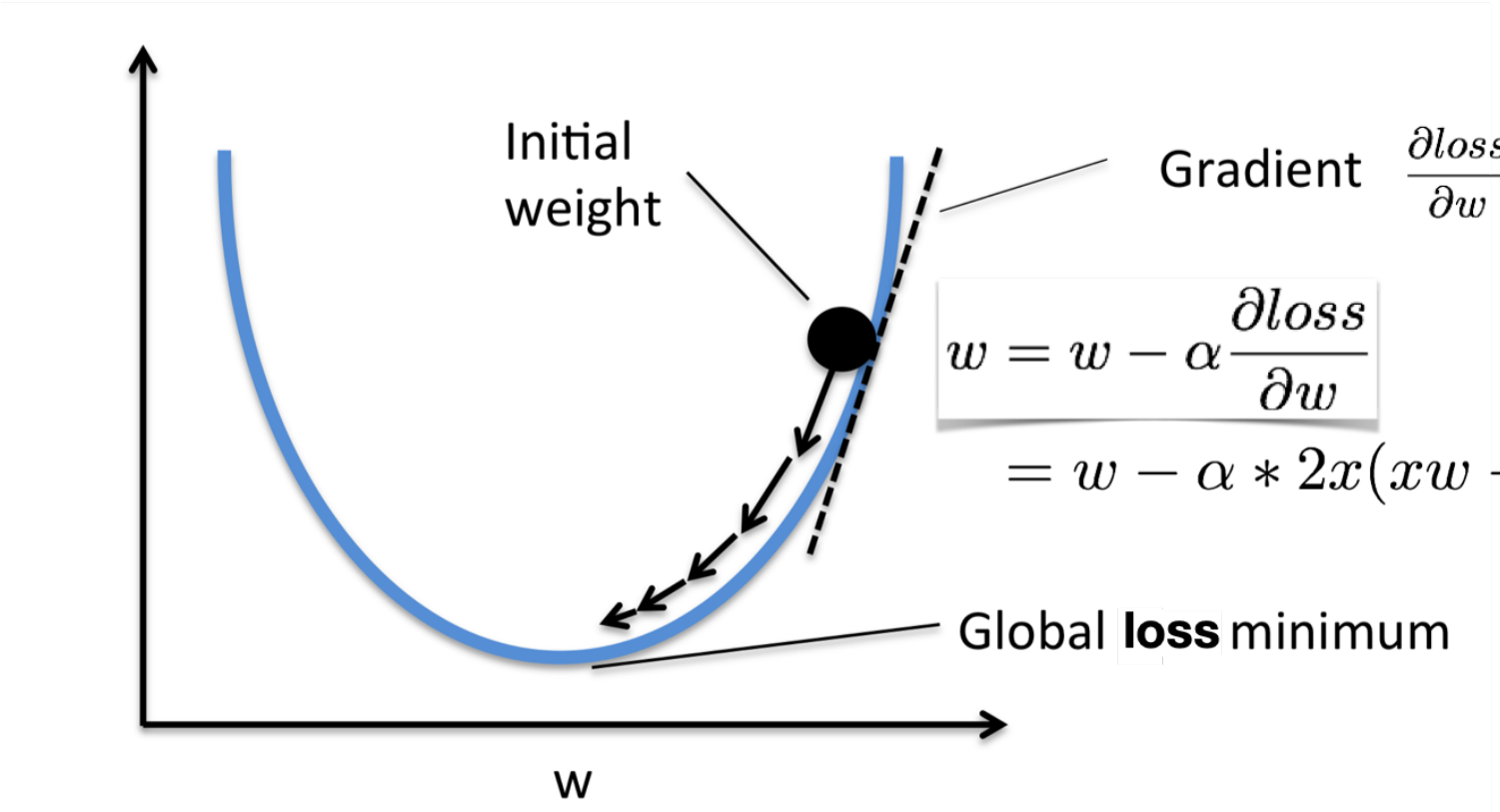
Exemple : optimisation MSE modèle linéaire



Exemple : optimisation MSE modèle linéaire



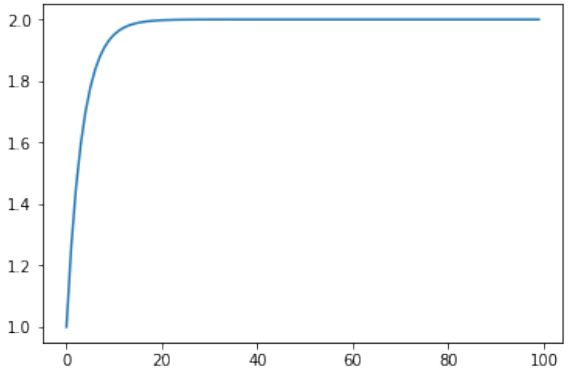
Exemple : optimisation MSE modèle linéaire

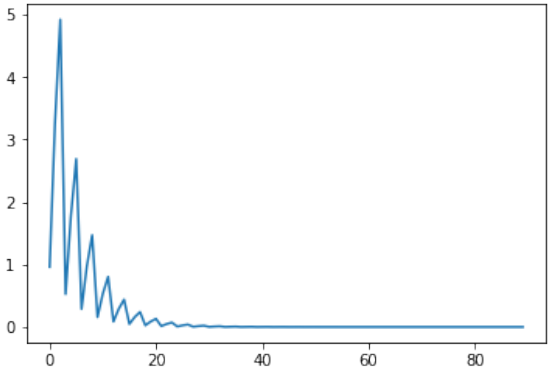


Exemple : optimisation MSE modèle linéaire

```python
# Données: entrées x (valeurs scalaires), sorties désirées y
x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

w = 1.0  # poids (paramètre) initial, au hasard

# modèle: calcul de la sortie ("forward pass")
def forward(x):
    return x * w


# Fonction de coût
def cout(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)


# calcul du gradient du cout par rapport au poids
def gradient(x, y):  # d_loss/d_w
    return 2 * x * (x * w - y)
```

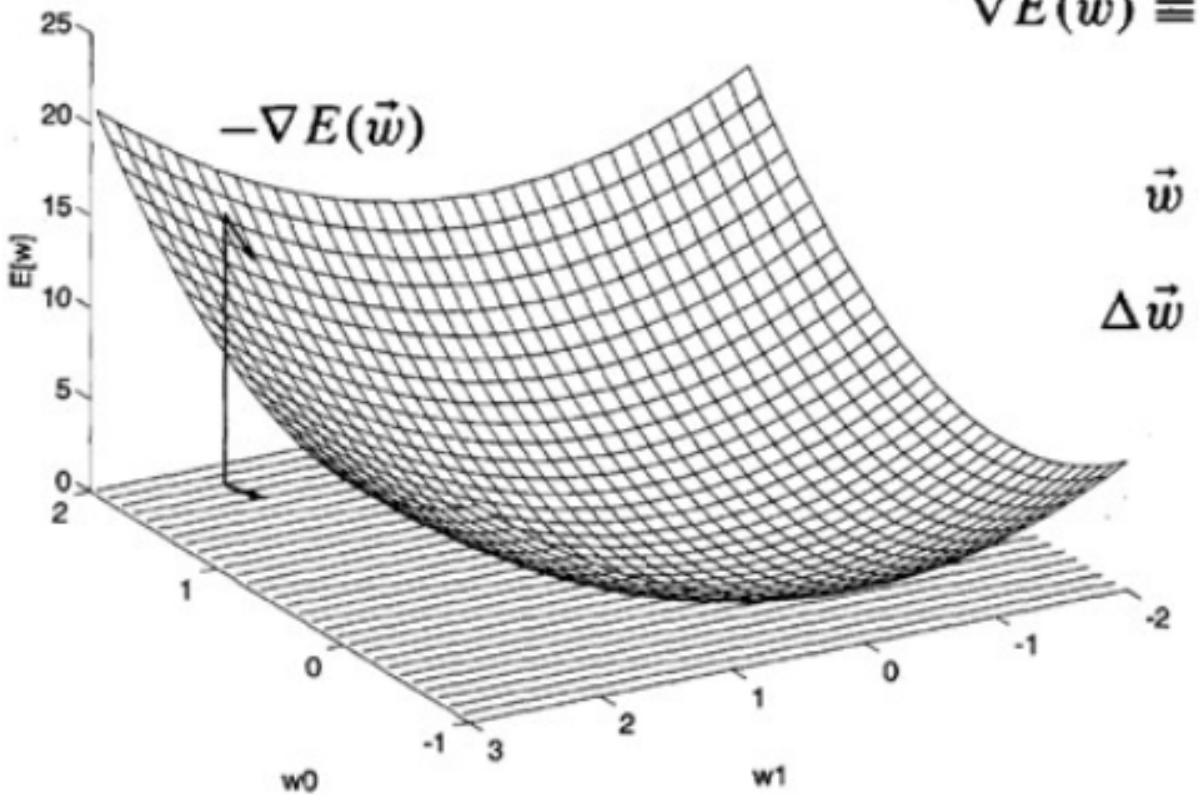
```python
# Before training
print("prévision avant apprentissage:",  4, forward(4))

# Boucle d'apprentissage
for epoch in range(100):
    for x_val, y_val in zip(x_data, y_data):
        grad = gradient(x_val, y_val)
        w = w - 0.01 * grad
        print("\tgrad: ", x_val, y_val, grad)
        l = cout(x_val, y_val)

    print("progress:", epoch, "w=", w, "loss=", l)

# After training
print("predict (after training)",  "4 hours", forward(4))
```





voir `01-exemple-1d.ipynb`

Apprentissage et optimisation

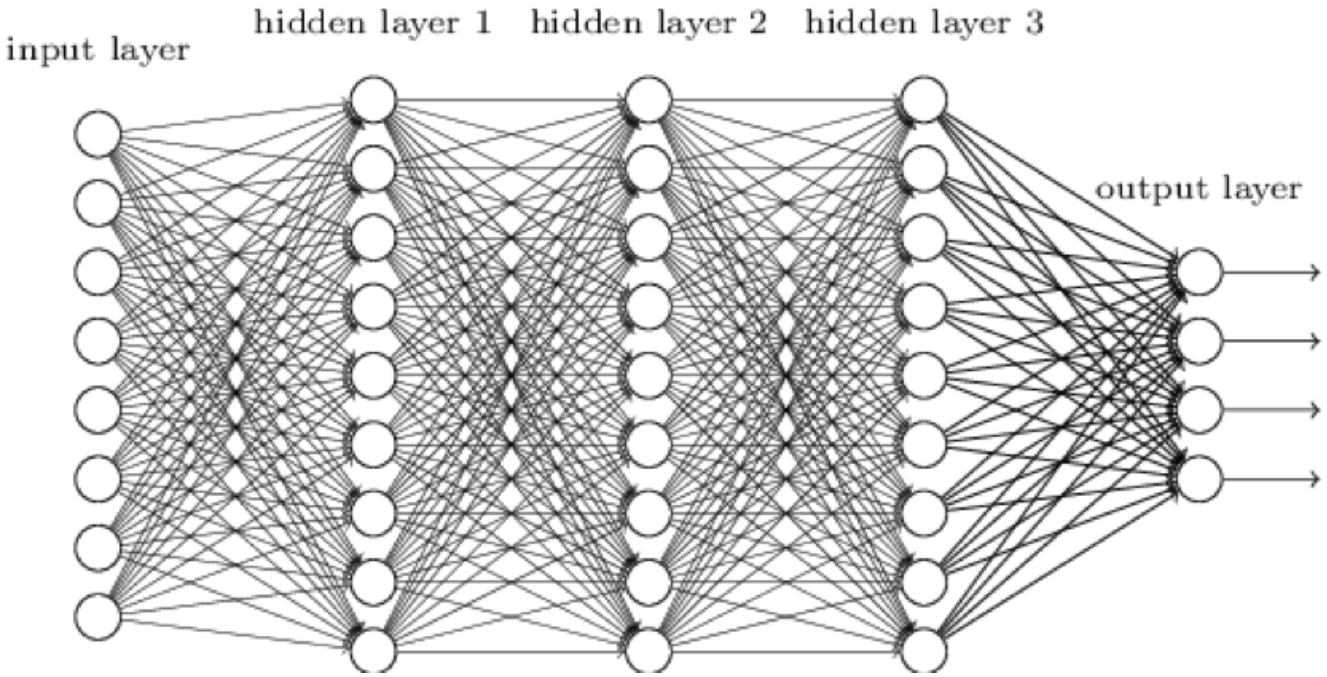


$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n}\right]$$

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

$$\Delta\vec{w} = -\eta \nabla E(\vec{w})$$

Apprentissage d'un réseau multicouche



Gradient du coût par rapport à $\boldsymbol{W}$: $\frac{\partial L}{\partial w} = ?$

Apprentissage d'un réseau multicouche

$$\underset{\text{entrée}}{X} \longrightarrow \underset{f}{\boxed{\text{COUCHE 1}}} \xrightarrow{X_1} \underset{g}{\boxed{\text{COUCHE 2}}} \longrightarrow \hat{Y}$$

$$L = \|\hat{Y} - Y\|^2$$

$$X_1 = f(X, w)$$

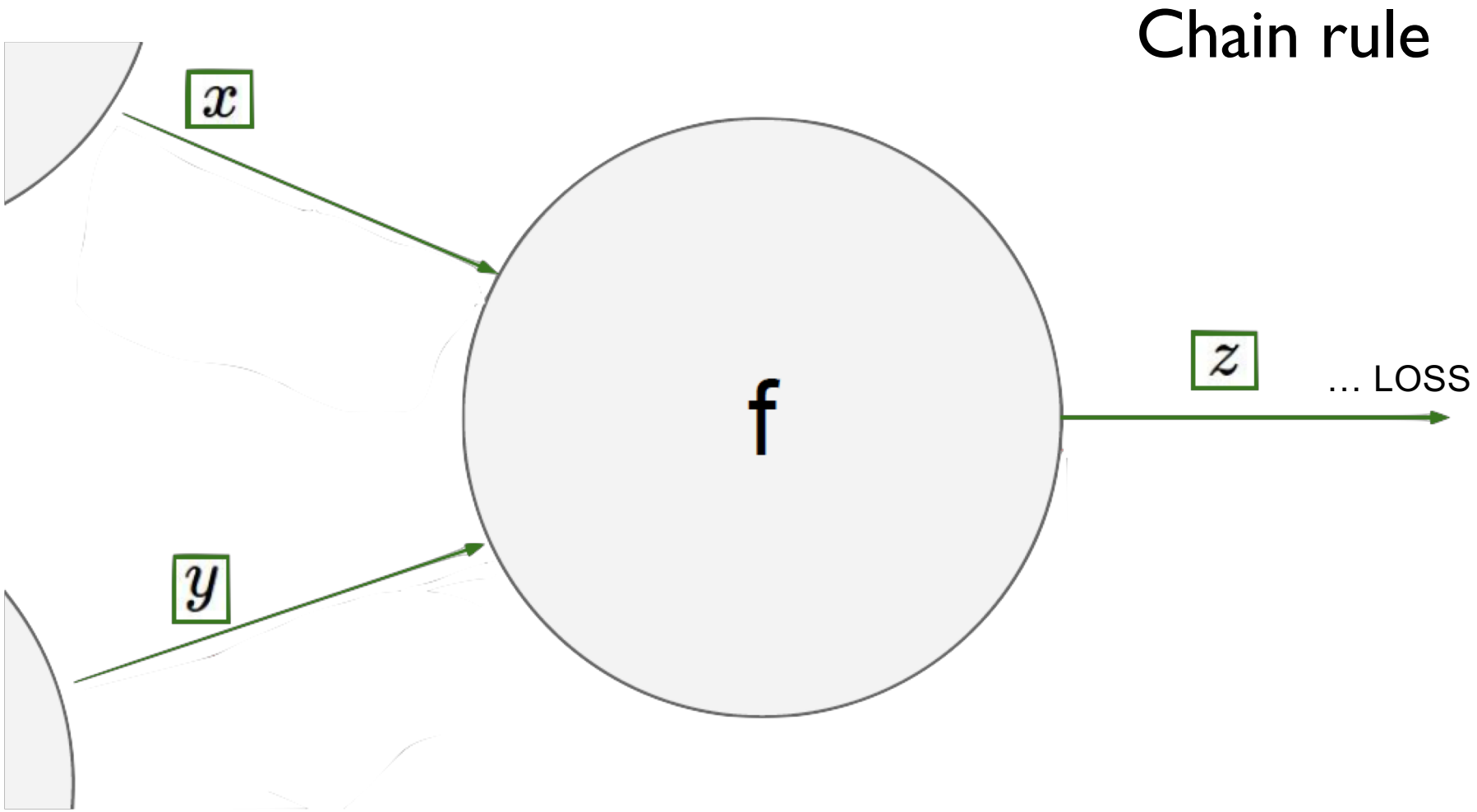
$$\hat{Y} = g(X_1) = g(f(X, w))$$

Si f dépend du paramètre w,

on veut $$\frac{\partial L}{\partial w} = \underbrace{\frac{\partial L}{\partial \hat{Y}} \times \frac{\partial \hat{Y}}{\partial X_1} \times \frac{\partial X_1}{\partial w}}_{= \text{"CHAIN RULE"}}$$

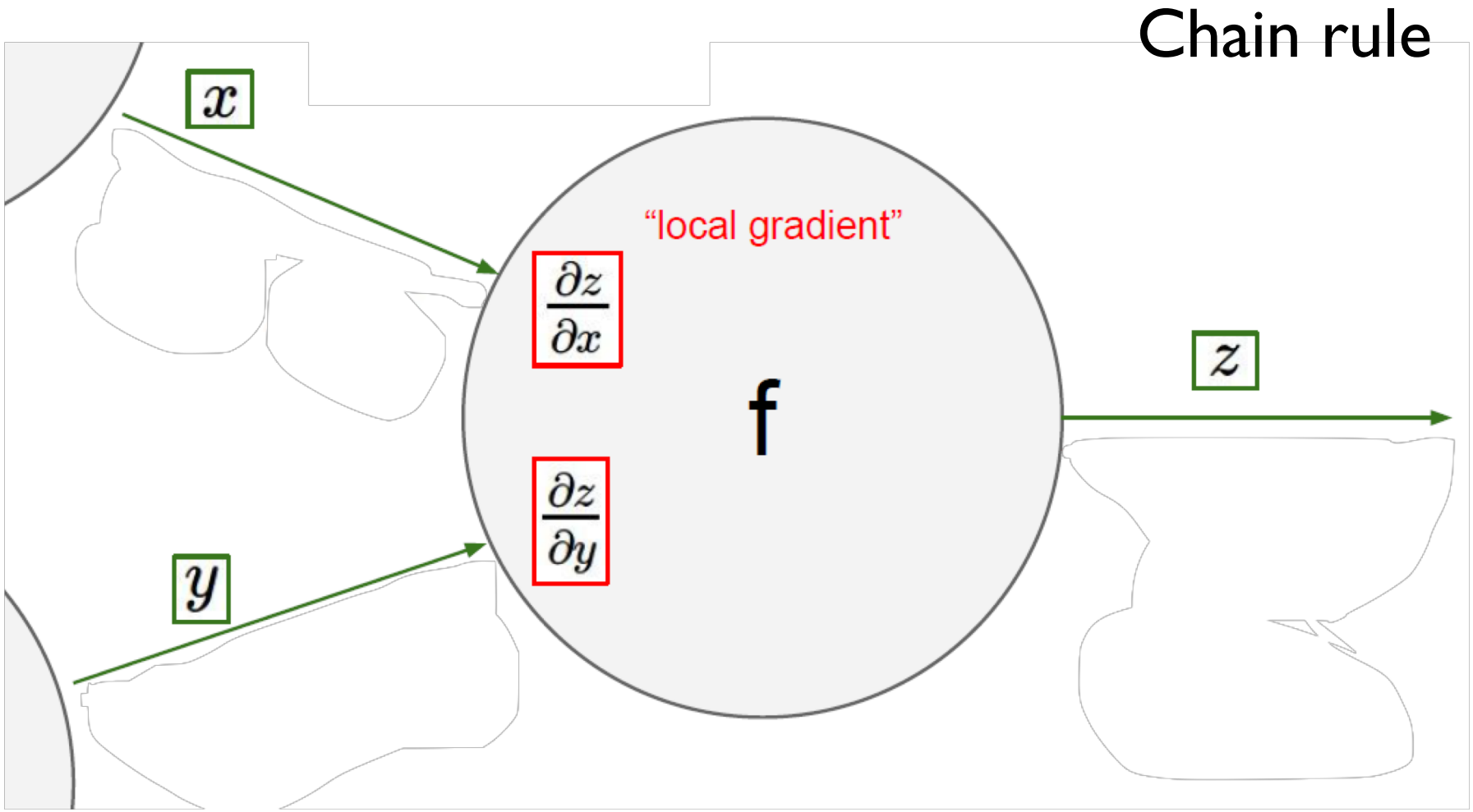
Dérivation des fonctions composées

Rétropropagation du gradient



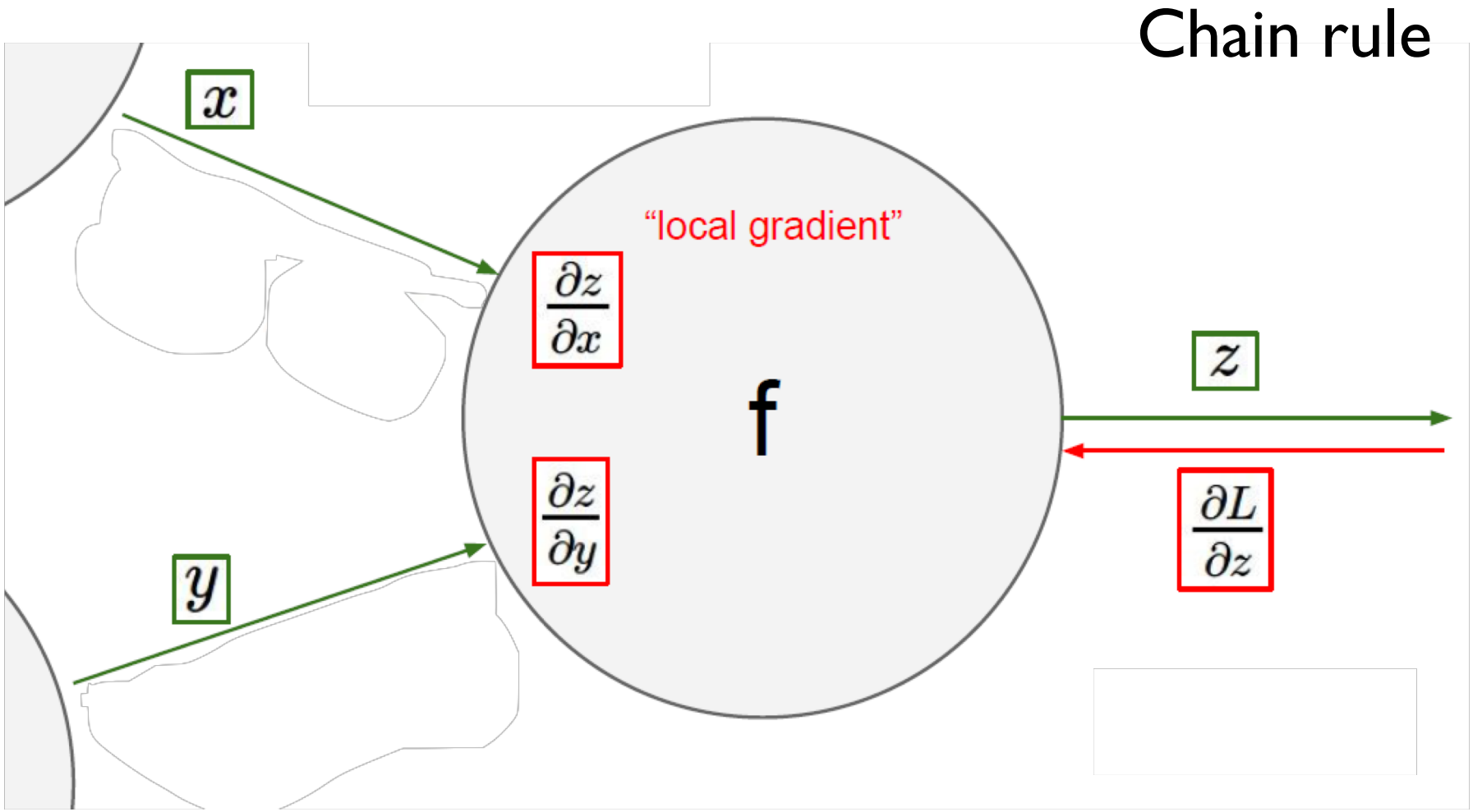
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



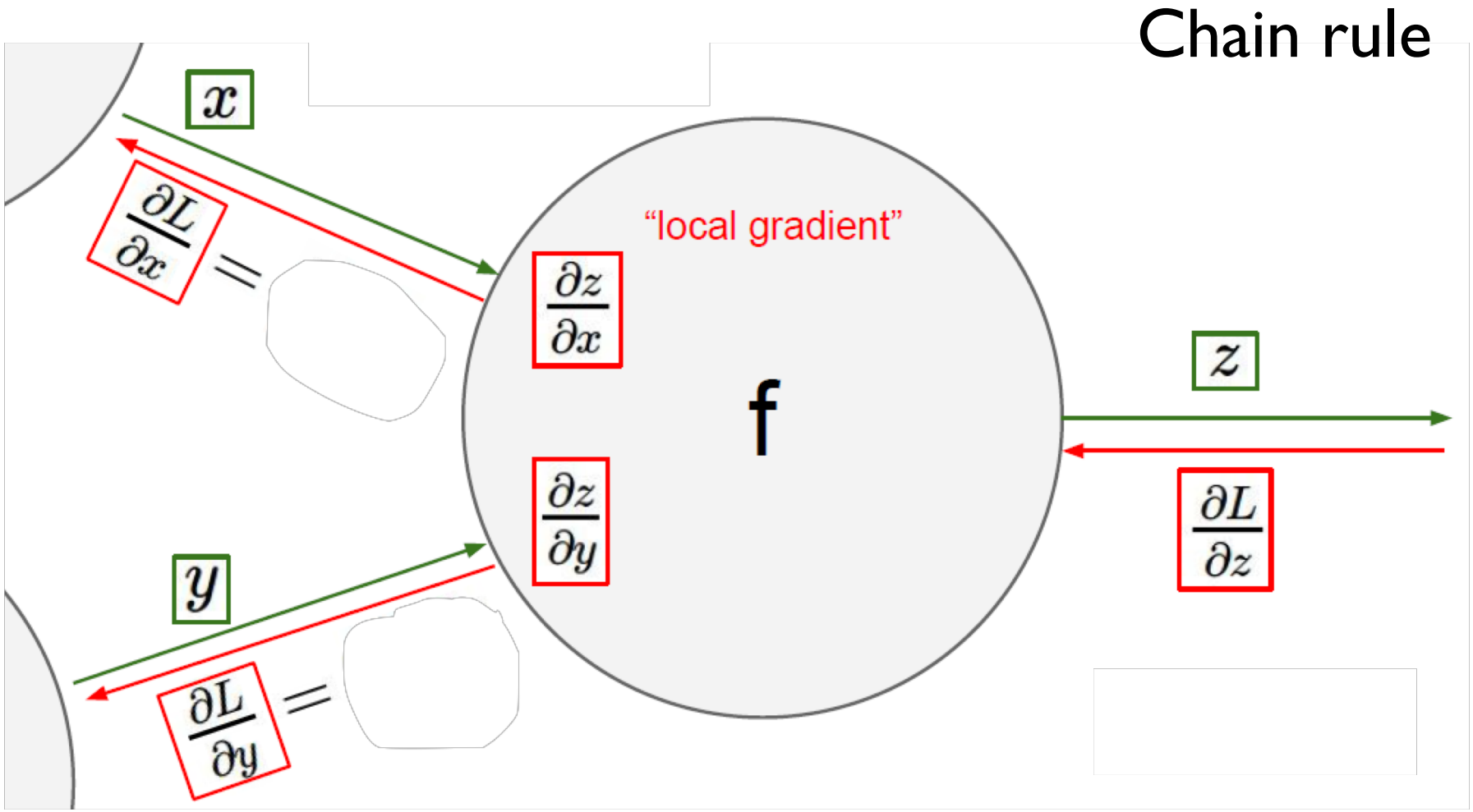
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



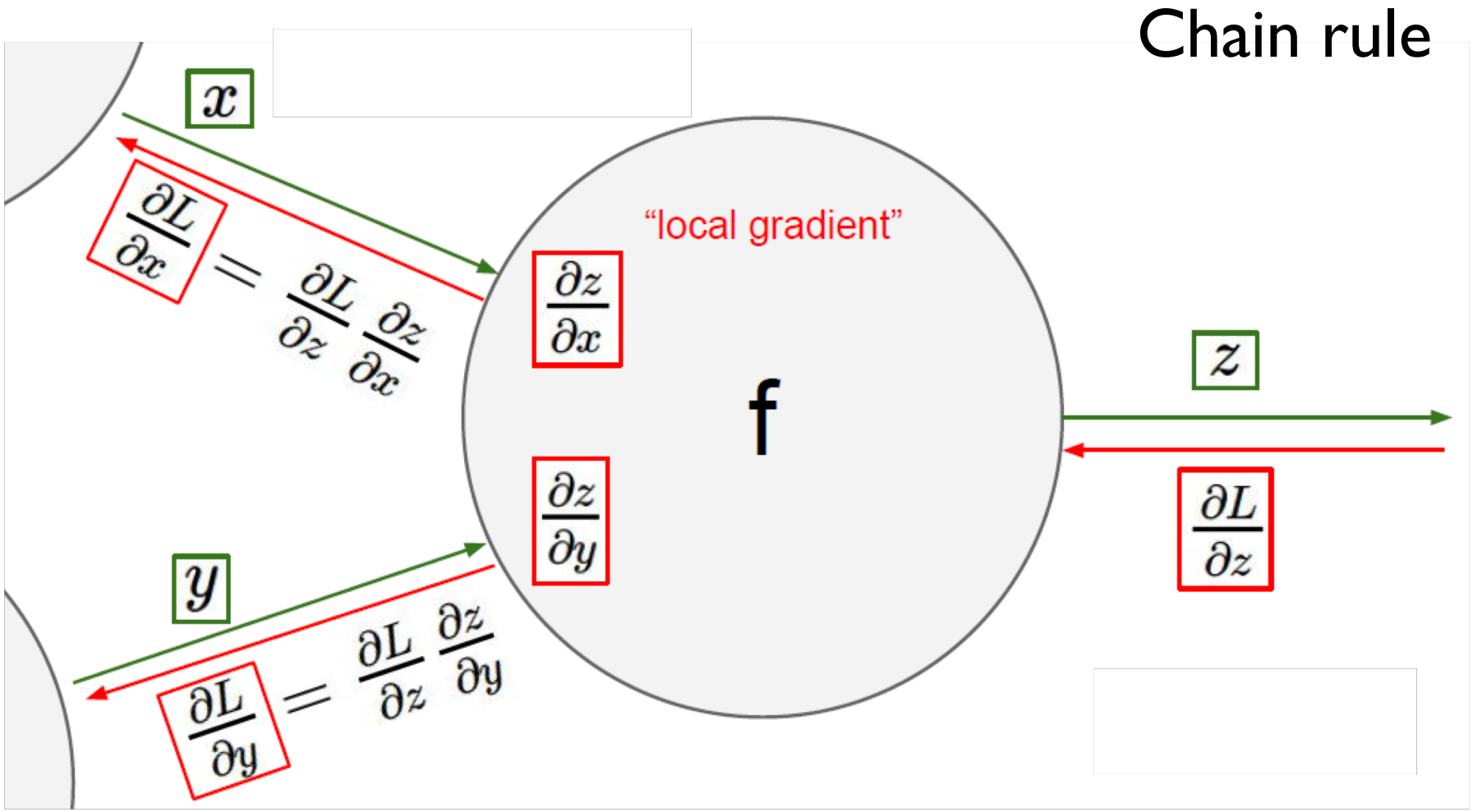
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



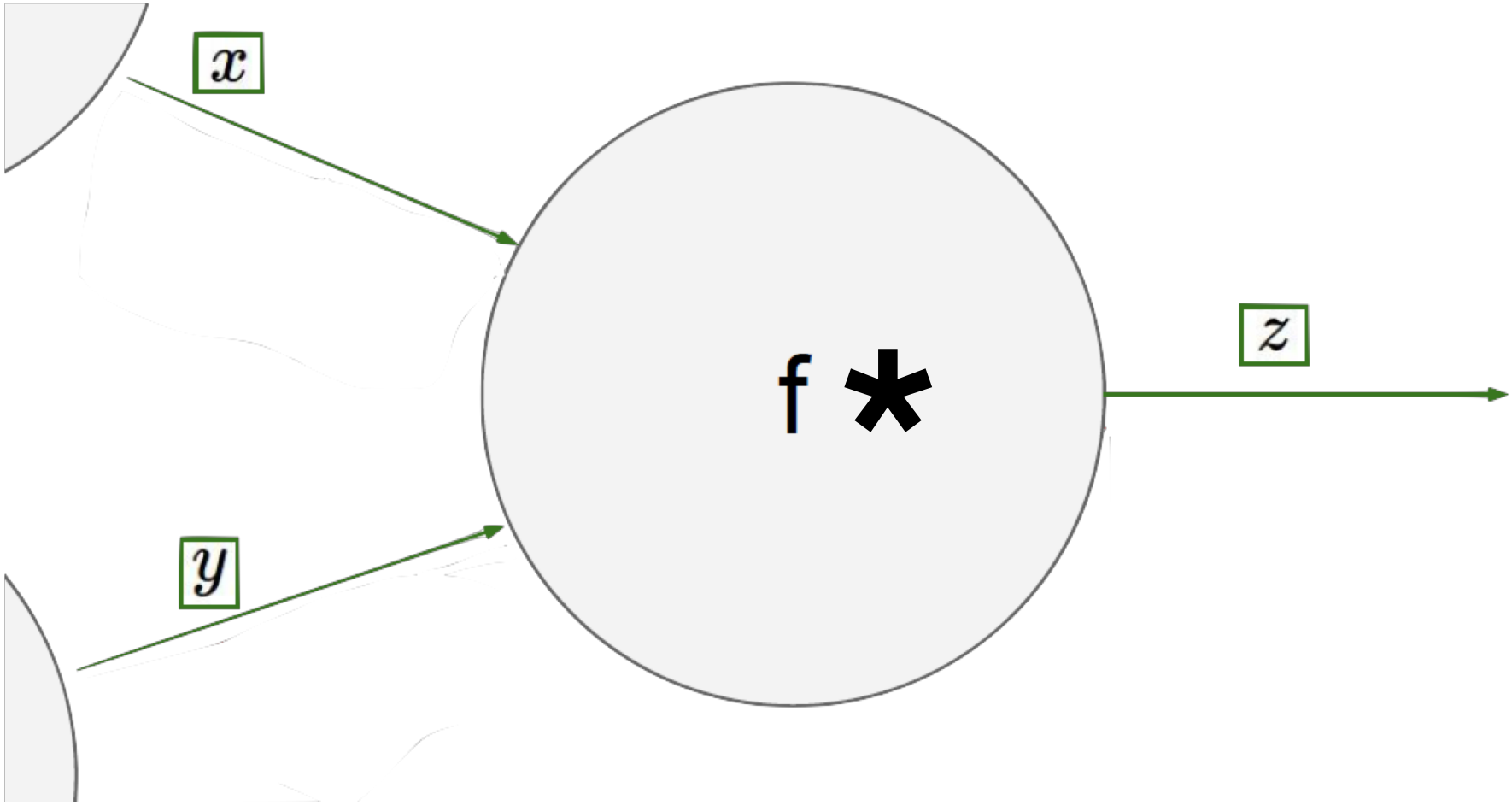
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



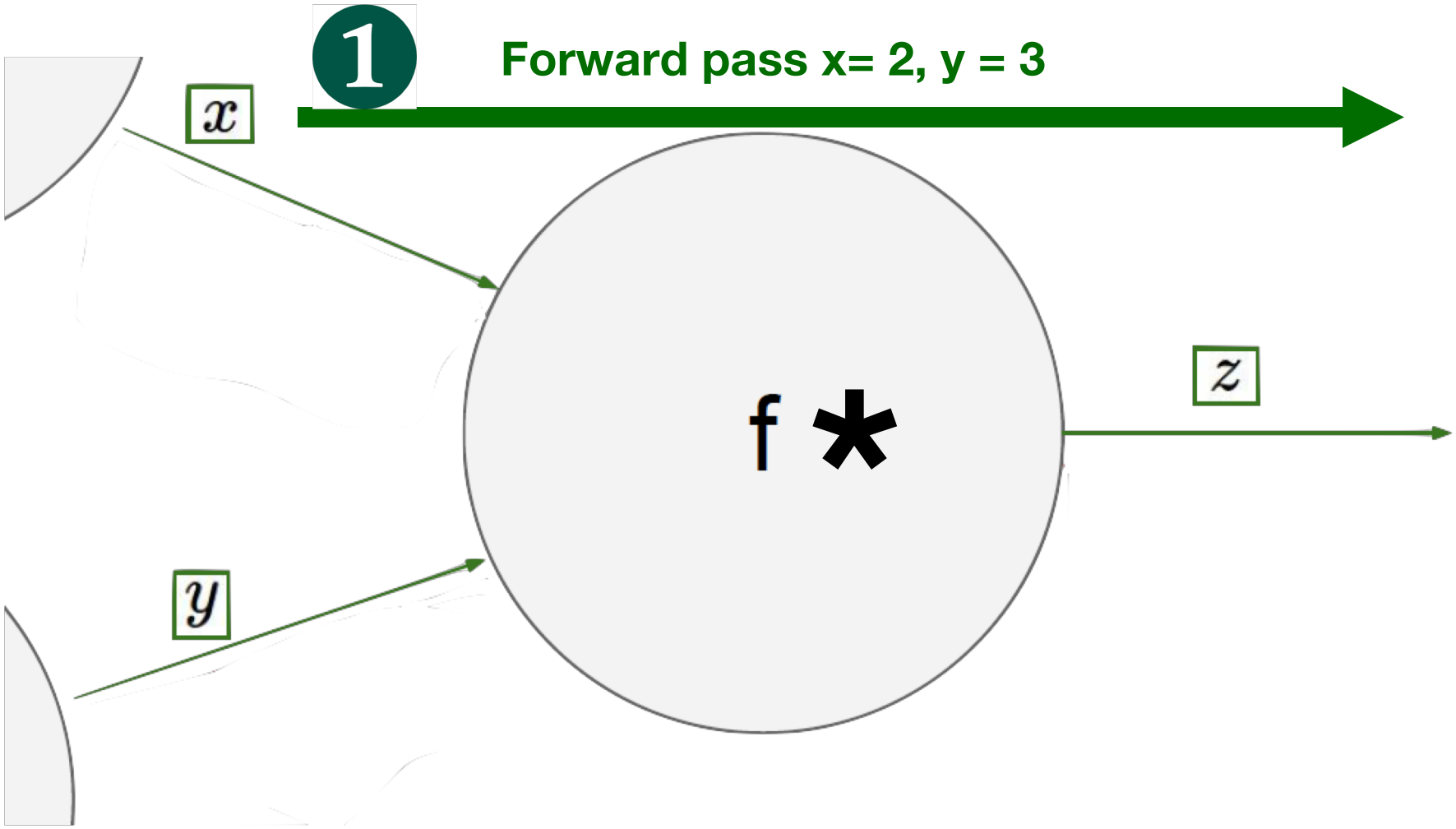
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



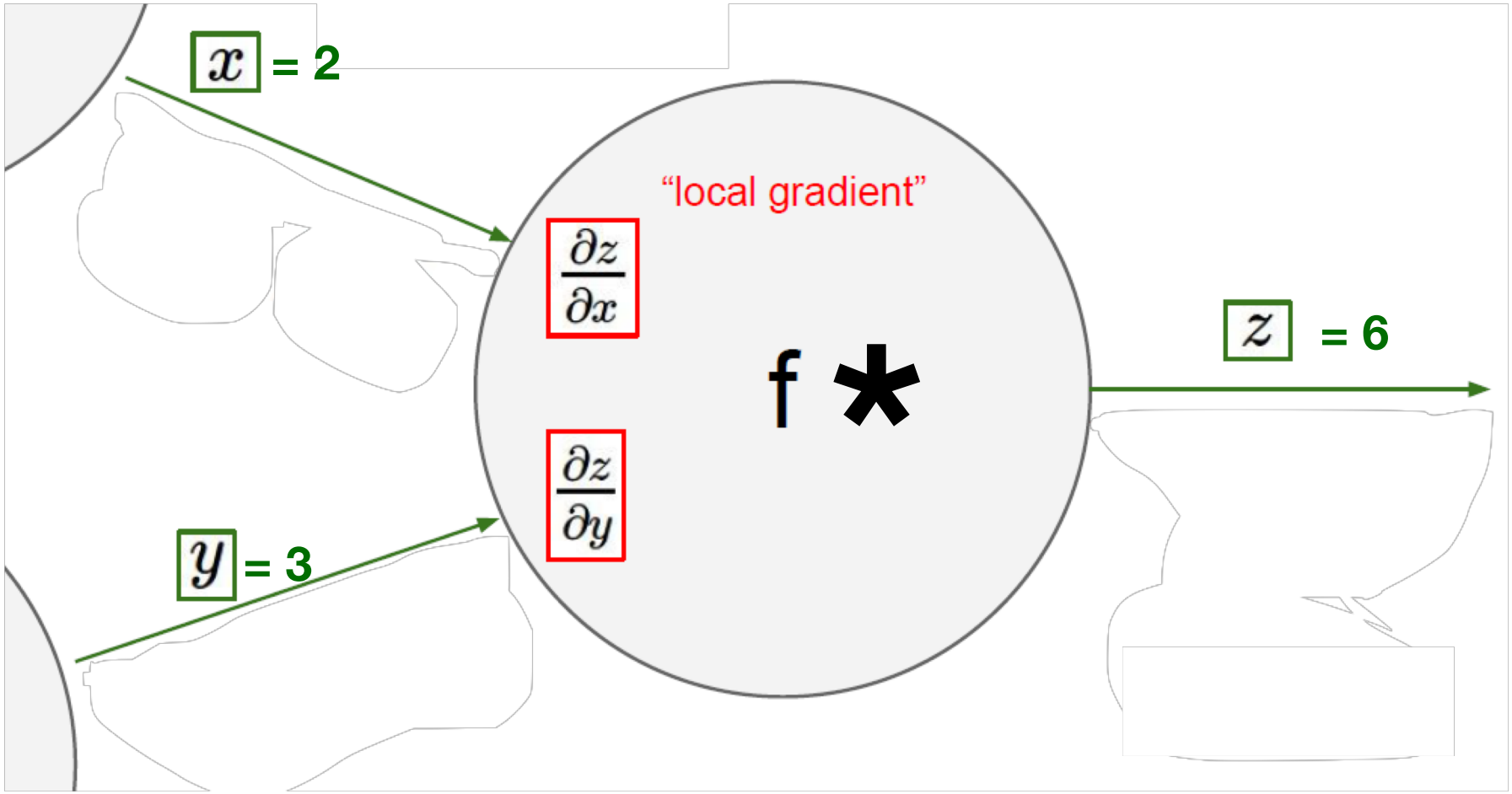
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



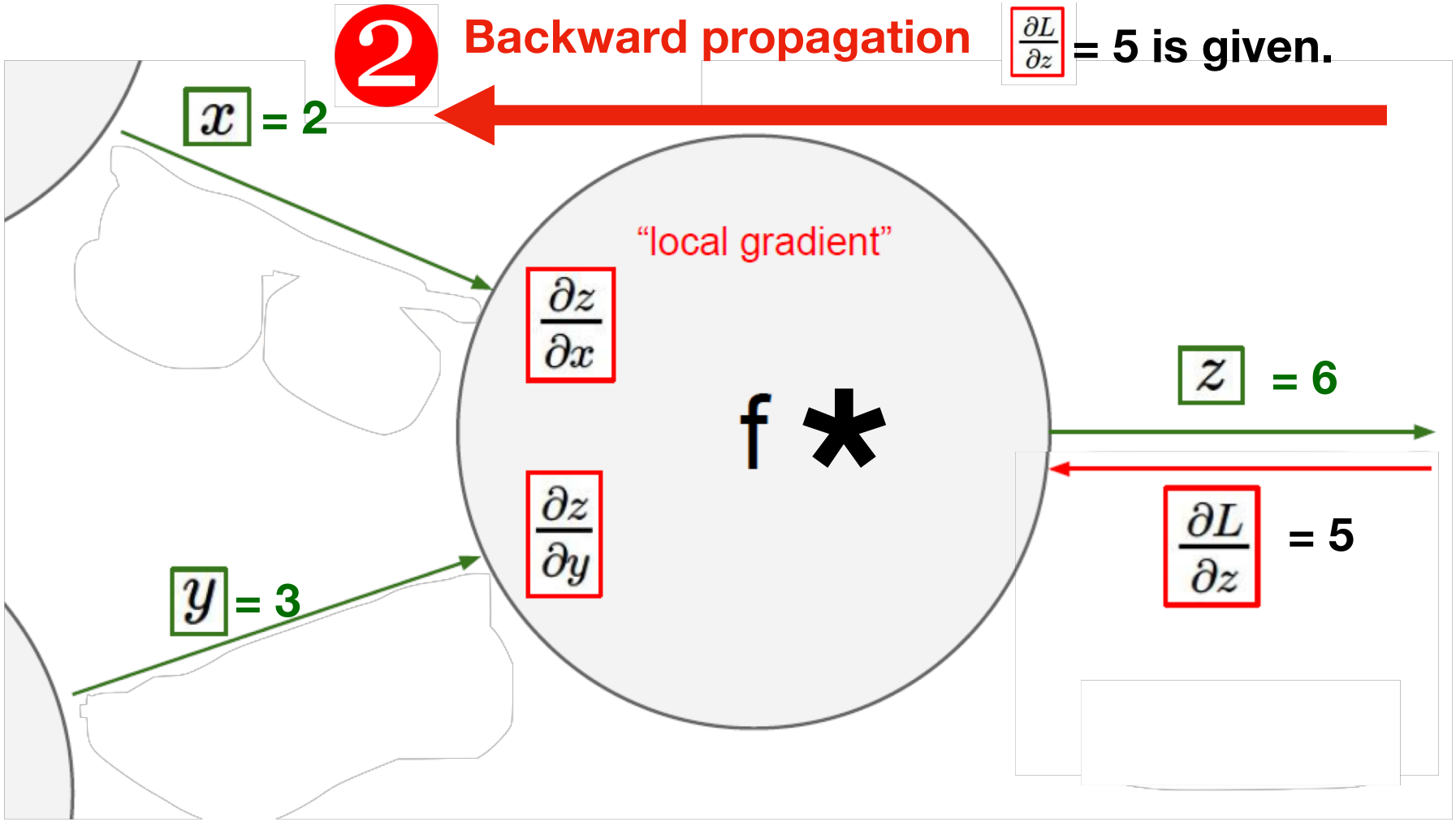
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



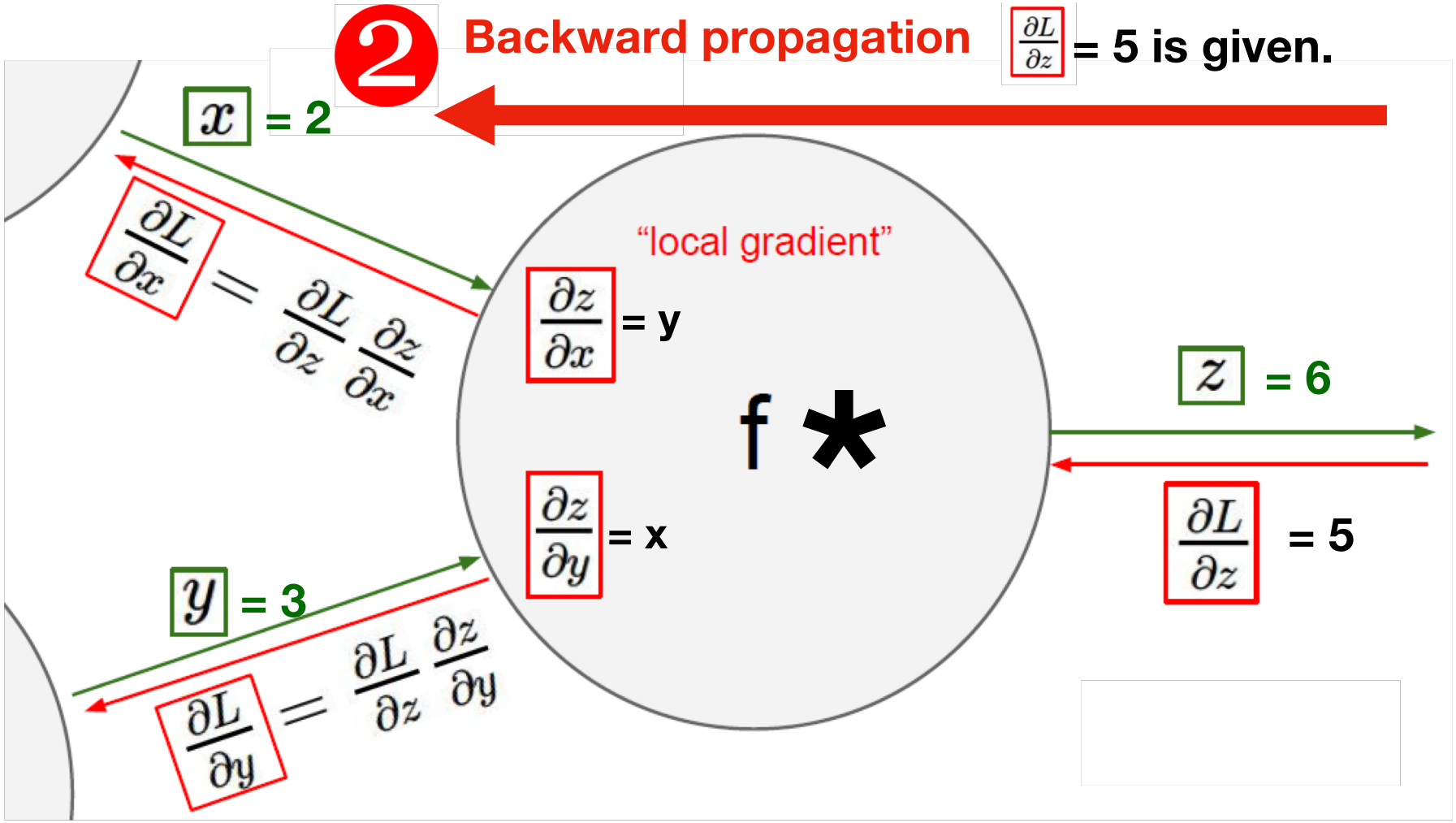
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



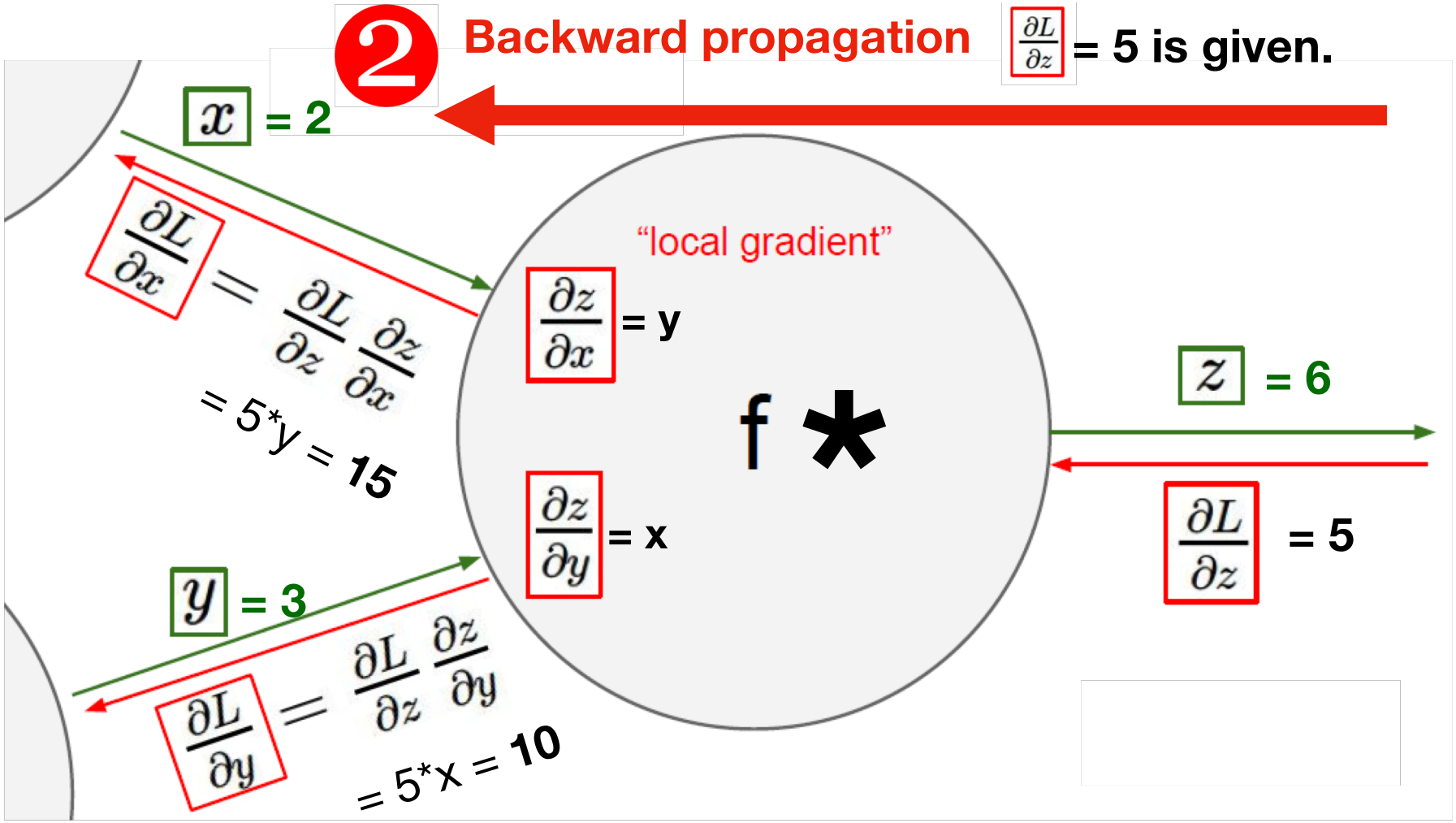
Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient



Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient

Computational graph

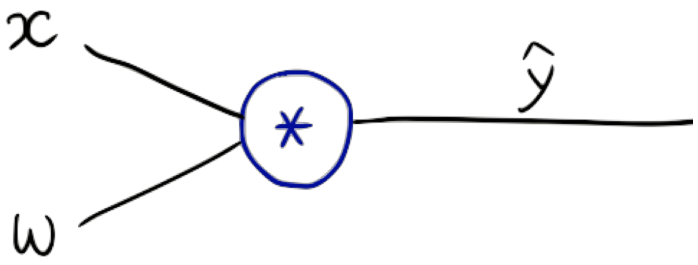
$$\hat{y} = x * w$$

Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient

Computational graph

$$\hat{y} = x * w$$

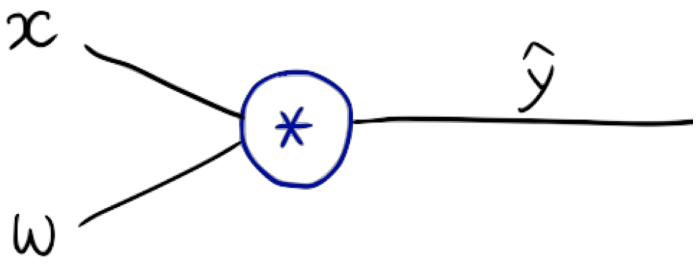


Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient

Computational graph

$$\hat{y} = x * w \qquad loss = (\hat{y} - y)^2 = (x * w - y)^2$$

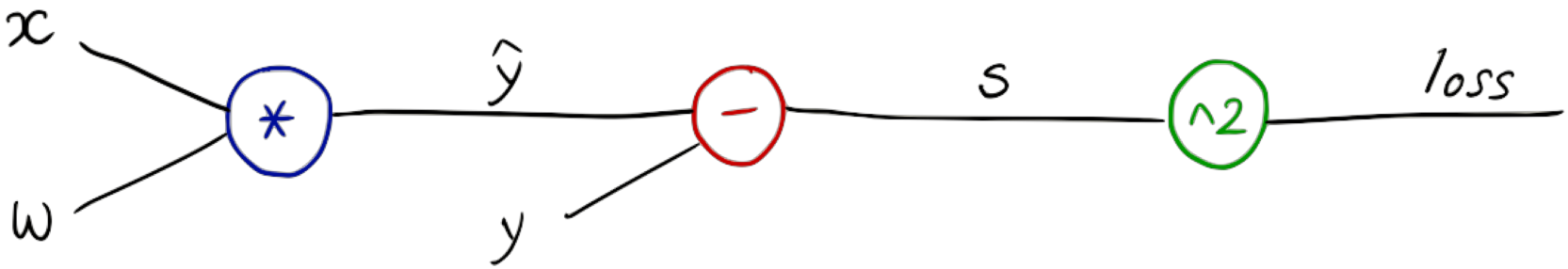


Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient

Computational graph

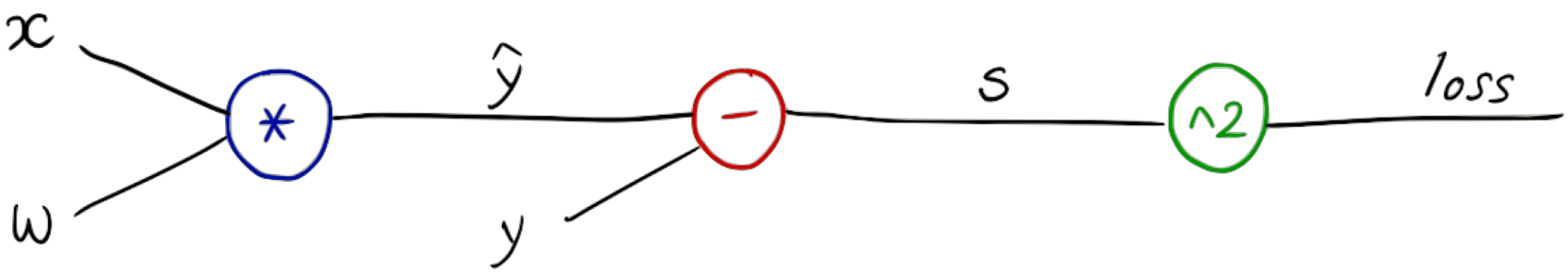
$$\hat{y} = x * w \qquad loss = (\hat{y} - y)^2 = (x * w - y)^2$$



Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

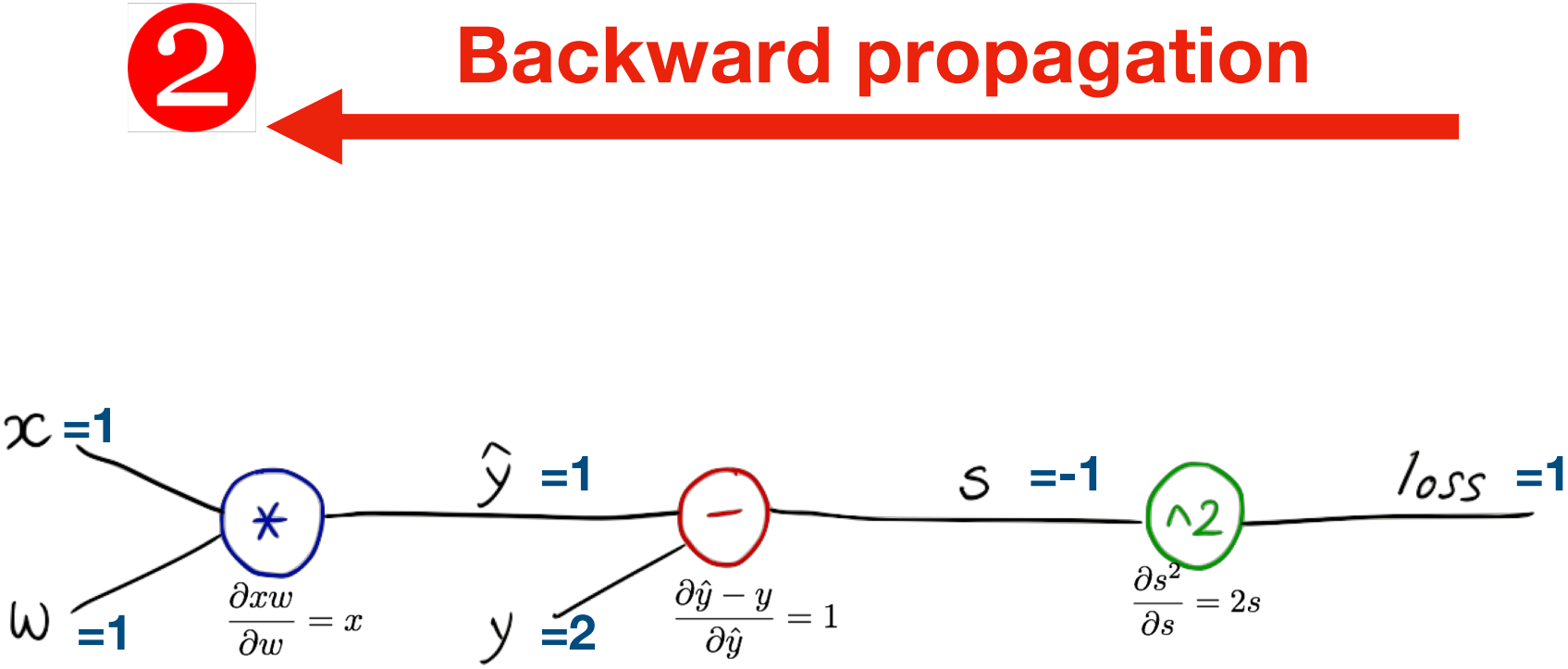
Rétropropagation du gradient

1 Forward pass x=1, y = 2 where w=1



Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

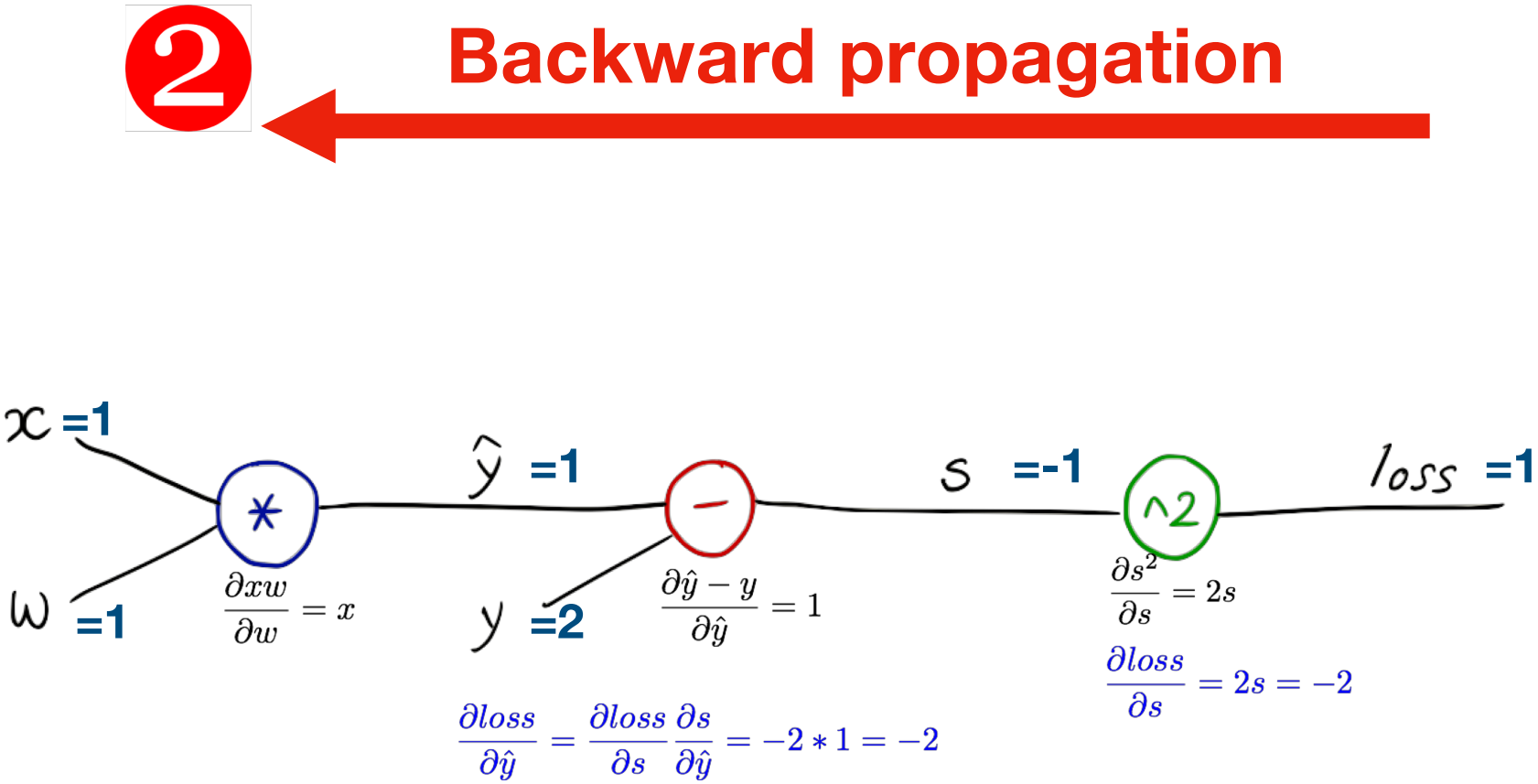
Rétropropagation du gradient



$$\frac{\partial loss}{\partial w} =$$

Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient

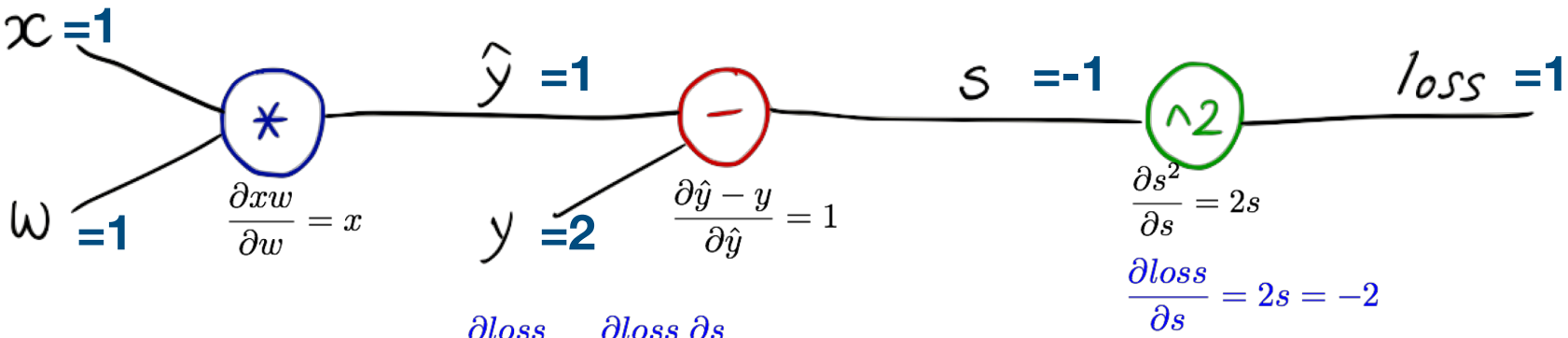


$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w} = -2 * x = -2 * 1 = -2$$

Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Rétropropagation du gradient

2 **Backward propagation**



$$\frac{\partial loss}{\partial s} = 2s = -2$$

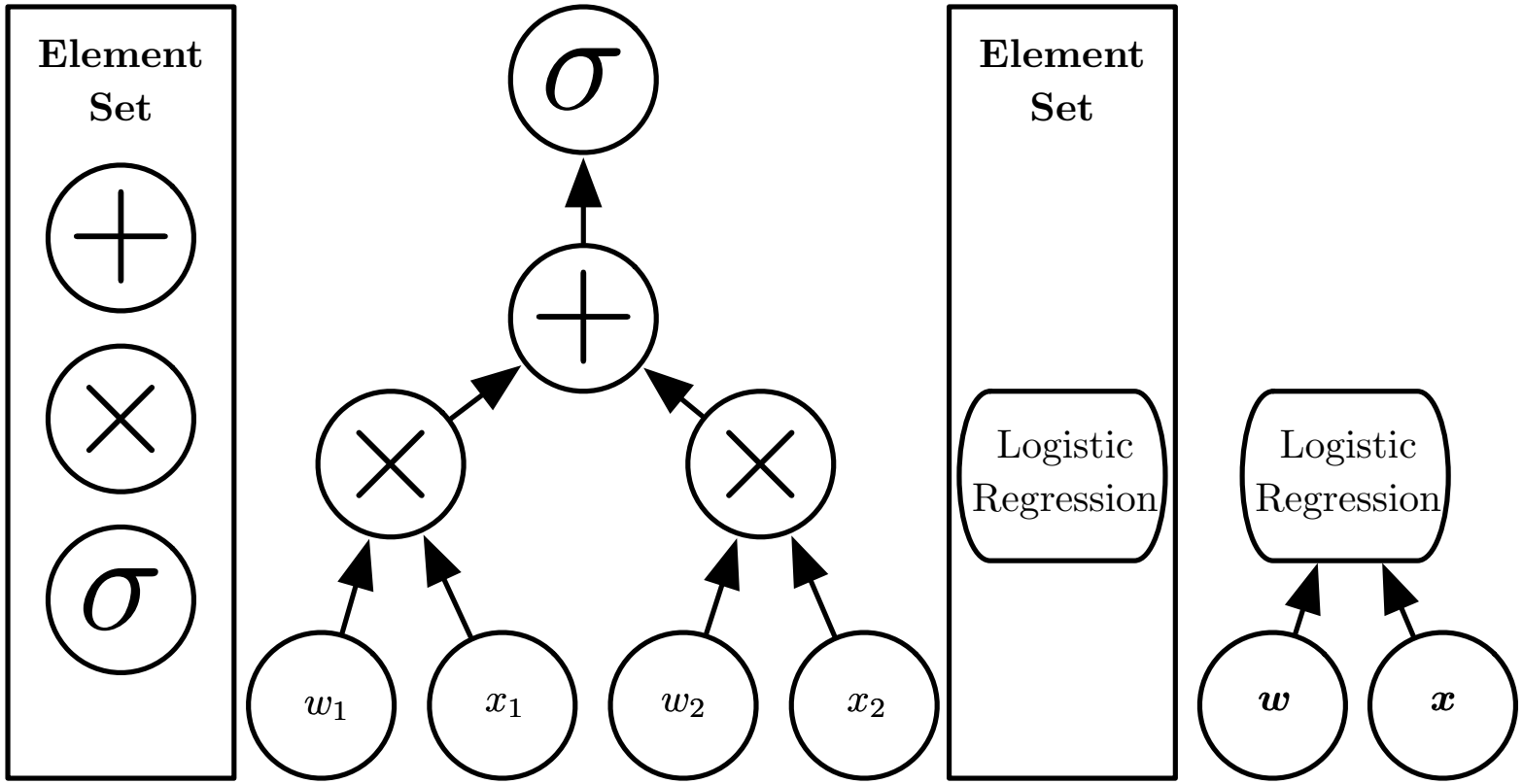
$$\frac{\partial loss}{\partial \hat{y}} = \frac{\partial loss}{\partial s}\frac{\partial s}{\partial \hat{y}} = -2 * 1 = -2$$

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w} = -2 * x = -2 * 1 = -2$$

```
    grad:  1.0 2.0 -2.0
    grad:  2.0 4.0 -7.84
    grad:  3.0 6.0 -16.2288
progress: 0 4.919240100095999
```

Source: Sung Kim, HKUST, http ://bit.ly/PyTorchZeroAll

Représentation graphique du calcul



Source: Goodfellow 2016

Exercice

Soit le modèle

$$\hat{y} = x^2.w_2 + x.w_1 + b$$

$$\text{loss} = (\hat{y} - y)^2$$

1. Calculer (manuellement) :

$$\frac{\partial loss}{w_1} = ?$$

$$\frac{\partial loss}{w_2} = ?$$

2. Coder ce calcul en Python.
3. Faire le même calcul avec PyTorch.

Résumé

Nous avons vu :

- l'importance du Deep Learning dans de nombreuses applications en vision ;
- un bref historique de l'apprentissage des réseaux connexionnistes ;
- les principes de l'apprentissage par descente du gradient ;
- le principe de l'algorithme de rétropropagation du gradient.

Installation de PyTorch

- `http://pytorch.org`
- `conda install pytorch torchvision -c pytorch`

Exemples : notebooks

- `02-pytorch-exemple-1d`
- `Toy-Classif-2classes-simplifie`

A suivre :

- réseaux à convolutions
- exemples