Réseaux de neurones (Partie 1a) « Backprop » illustré sur le XOR

Bruno Bouzy

bruno.bouzy@u-paris.fr

Janvier 2022

Cours apprentissage machine

Le problème du XOR avec 2 neurones (1/2)

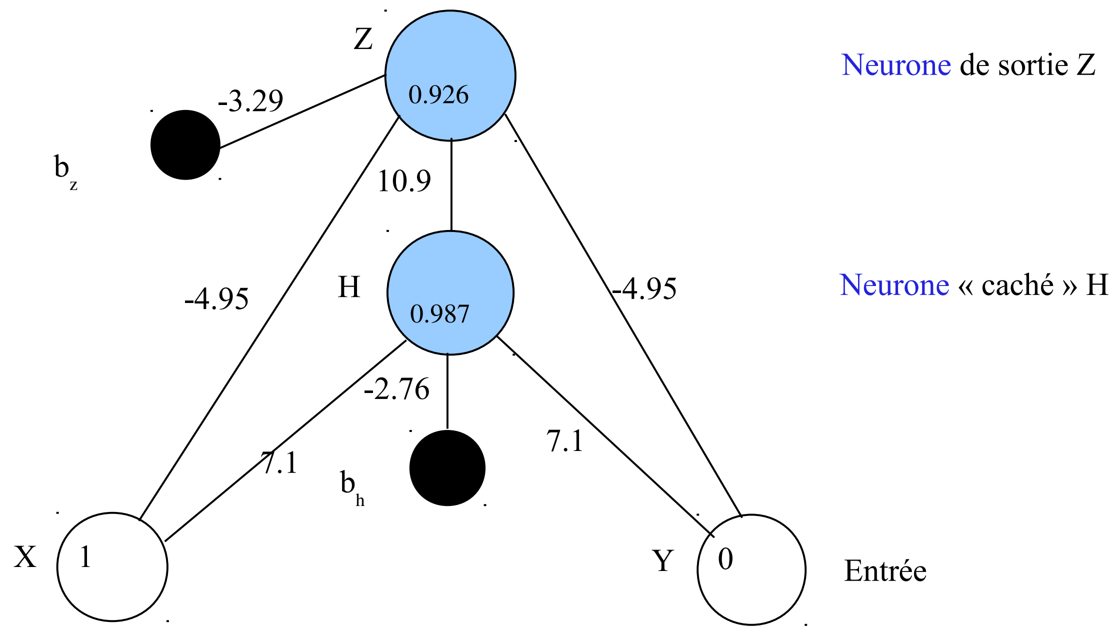
- La fonction z = XOR(x, y)

x	0	0	1	1
y	0	1	0	1
XOR	0	1	1	0

- On souhaite approximer cette fonction avec un (mini-)réseau de neurones

Une solution avec 2 neurones

- mini-réseau de neurones approximant la fonction XOR



Vocabulaire

- Neurone
  - Possède des entrées, une sortie avec une valeur d’activation.
  - Possède des connexions avec des neurones entrants, un biais.
  - Possède une fonction d’activation
  - Il appartient à une couche.
  - Synonyme : « unité »
- Connexion
  - Relie deux neurones entre eux
  - Elle a un poids.
- Biais, poids
  - Valeur numérique d’une connexion
- Valeur d’activation
  - Valeur de sortie du neurone
- Entrée, sortie
  - Les entrées d’un neurone sont soit les entrées du réseau, soit connectées à la sortie de neurones entrants
  - La sortie d’un neurone est connectée à un neurone, elle contient la valeur d’activation du neurone.
- Couche
  - Ensemble de neurones lorsque le réseau est structurés en couches
- Réseau
  - Ensemble de couches si le réseau est structurés en couches, ou de neurones sinon

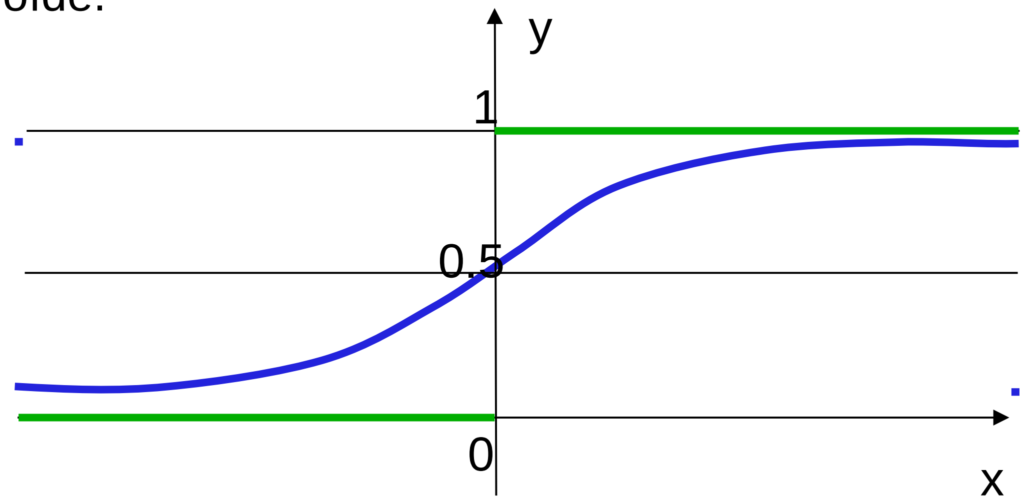
Calcul de la valeur d'activation ou « sortie » d'un neurone

- Pour calculer sa sortie, le neurone :
  - Effectue une combinaison linéaire de ses entrées avec les poids des connexions et le biais :
    - Pour le neurone H :

      somme = 1 x 7.1 + 1 x (– 2.76) + 0 x 7.1 = 4.34
  - Applique une fonction d'activation f:
    - Pour le neurone H :
    - sortie = f(somme)
    - sortie = f(4.34) = 0.987

Fonction d'activation sigmoïde

- La fonction d'activation permet de calibrer les sorties.
- Sur l'exemple, c'est la fonction sigmoïde.
  - $f(x) = 1 / (1+e^{-x})$



- Propriétés
  - Valeurs entre 0 et 1
  - $f(0) = 0.5$ $f(-\infty) = 0$ $f(+\infty) = 1$
  - $f'(x) = f(x)\,(1 - f(x))$ (sauriez vous le montrer ?)Valeurs ent
  - Approximation continue et dérivable de la fonction à seuil
    - seuil(x) = 1 si x>0
    - seuil(x) = 0 si x<0
    - seuil(0) = 0.5

Calcul en avant, sortie du réseau

- Calcul en avant :
  - Les sorties des neurones sont calculées en cascade des neurones proches des entrées vers la sortie.
  - Pour (x, y) = (1, 0)
    - Sortie de H = 0.987
    - Pour Z :
      - somme = 1x(-4.95) + 0x(-4.95) + 0.987x 10.9 + 1x(-3.29) = 2.52
      - sortie = f(somme) = 0.926
- Sur les 4 entrées possibles, le calcul en avant du réseau donne :

x	0	0	1	1
y	0	1	0	1
XOR	0	1	1	0
réseau	0.067	0.926	0.926	0.092
erreur	0.067	0.074	0.074	0.092

- Le résultat est satisfaisant :
  - les sorties sont « correctes »
  - l’erreur de sortie est inférieure à 0.1

Formalisation

- Pour un neurone j ayant les entrées i:

  $o_j = f(net_j)$ avec $net_j = b_j + \sum_i w_{ij}\, o_i$ (1b)

  o_j : sortie (output)

  w_{ij} : poids (weight) de la connexion i j

  o_i : entrée i du neurone j (égale sortie du neurone i)

Backprop

- Rumelhart & al 1986
- But : trouver W un vecteur des poids des connexions de manière automatique
  - pour que les sorties du réseau correspondent aux sorties attendues.
- Idée : considérer la fonction d'erreur E du réseau.
  - Soit W le vecteur des poids w_{ij} du réseau
  - Soit E(W, X) l'erreur commise par le réseau sur un exemple X en utilisant W
    - erreur = différence entre la sortie attendue du réseau et la sortie effective du réseau.
    - (La sortie attendue est fournie par l'expertise du domaine, un « oracle »)
  - Soit E(W) l'erreur commise par le réseau sur un ensemble d'exemples connus en utilisant W
    - Par exemple : erreur = moyenne des erreurs sur tous les exemples
  - E(W) est une fonction continue dérivable de W : son gradient est calculable analytiquement
  - On peut appliquer l'algorithme de la descente de gradient.
  - Cela donne des formules de mises à jour de W
  - A la fin de la descente de gradient E(W) W est proche du minimum W*.
  - Le réseau à appris W pour que l'erreur soit la plus faible possible.

Formules de mises à jour de W (1/2)

- Pour un neurone k en sortie du réseau, on considère son « signal d'erreur » d_k
  - $d_k = (t_k – o_k) f'(net_k)$ (2)
  - t_k est la valeur attendue (cible, Target en anglais)
- F est la fonction sigmoïde
  - $f'(net_k) = o_k(1 - o_k)$ (3)
- Formule issue de la descente de gradient (cf algo descente de gradient)
  - $\Delta w_{jk} = \nu d_k o_j$ (4)
  - ν est le « pas » d'apprentissage
  - Δw_{jk} signifie que l'on modifie w_{jk} d'une valeur égale à celle située à droite du signe = de l'équation.

Formules de mises à jour de W (2/2)

- Pour un neurone caché j, on considère aussi son « signal d'erreur » d_j
  - $d_j = f'(net_j) \sum_k d_k \, w_{jk}$ (5)
  - Les indices k désignent les neurones dont une entrée correspond à la sortie du neurone j
  - Le signal d'erreur est une moyenne pondérée des signaux d'erreurs des neurones « au dessus » du neurone j
- Formule issue de la descente de gradient (cf algo descente de gradient)
  - $\Delta w_{ij} = \nu \, d_j \, o_i$ (6)

Sur neurone Z du problème du XOR (1/2)

- Supposons que :

  W = 0

  x=1, y=0, t=1 (valeur attendue du XOR(1, 0))

  ν = 0.1

- On effectue un calcul en avant pour connaître h et z
  - h=0.5
  - z=0.5

x	1
y	0
XOR	1
réseau	0.5
erreur	0.5

- Sur le neurone Z qui possède 3 entrées X, Y, H, les formules (2) (3) (4) donnent :
  - $d_z = (1 - 0.5)\ 0.5\ (1 – 0.5) = 0.125$
  - $w_{zx} = 0 + 0.1x0.125x1 = 0.0125$
  - $w_{zy} = 0 + 0.1x0.125x0 = 0$
  - $w_{zh} = 0 + 0.1x0.125x0.5 = 0.00625$
  - $w_{zbz} = 0 + 0.1x0.125x1 = 0.0125$

Sur le neurone caché du problème du XOR (2/2)

- Supposons que :

  W est issu de la mise à jour sur le neurone Z

  x=1, y=0, t=1 (valeur attendue du XOR(1, 0))

  ν = 0.1
- Sur le neurone H qui possède 2 entrées X, Y les formules (5) (6) donnent :
  - d_h = 0.5 (1 – 0.5) 0.125 0.00625 = 0.000195
  - W_{hx} = 0 + 0.1 0.000195 1 = 0.0000195
  - W_{hy} = 0 + 0.1 0.000195 0 = 0
  - W_{zh} = 0 + 0.1 0.125 0.5 = 0.00625
  - W_{hbh} = 0 + 0.1 0.000195 1 = 0.0000195
- W a encore été modifié très légèrement.
  - Sur l'exemple (1, 0), Z = 0.507
  - L'erreur à légèrement diminué
- On a effectué un calcul en arrière.
- D'où le nom backprop de l'algorithme.

x	1
y	0
XOR	1
réseau	0.507
erreur	0.493

Sur le problème du XOR

- On effectue le calcul en arrière sur les 4 exemples. Après 1 itération :

x	0	0	1	1
y	0	1	0	1
XOR	0	1	1	0
réseau	0.4999	0.4998	0.4998	0.4997
erreur	0.4999	0.5002	0.5002	0.4997

Les valeurs ont très légèrement changé (pas nécessairement dans le bon sens).

- Backprop itère plusieurs fois (c'est une descente de gradient) sur les 4 exemples.
- Backprop s'arrête lorsque l'erreur est descendue en dessous d'un seuil acceptable.
- A la fin, le réseau donne les sorties attendues à l'erreur près.

Sur le problème du XOR

- Avec ν = 0.1 et après suffisamment d'itérations du calcul en arrière sur les 4 exemples:

x	0	0	1	1
y	0	1	0	1
XOR	0	1	1	0
réseau	0.07	0.92	0.92	0.09
erreur	0.07	0.08	0.08	0.09

Les valeurs sont correctes à 0.09 près.

- Backprop possède les mêmes qualités/défaut d'une descente de gradient.

Qualités et défauts de Backprop (1/2)

- Dépendance aux valeurs initiales
  - W = 0
  - W initialisé aléatoirement
- Minimum local ou global
  - Dépend de l'initialisation
  - Du pas d'apprentissage
- Convergence dépendante du pas d'apprentissage ν
  - Petites valeurs :
    - convergence lente : nombre d'itérations grand.
  - Valeurs mises au point :
    - convergence aussi rapide que possible : nombre d'itérations « minimal »
  - Grandes valeurs :
    - convergence vers un point incorrect
    - non convergence

Qualités et défauts de Backprop (2/2)

- Deux initialisations différentes produisent des réseaux corrects mais complètement différents

	Whx	Why	Whb	Wzx	Wzy	Wzh	Wzb
Init.	0	0	0	0	0	0	0
fin	8,9	8,9	-3,88	-9,46	-9,46	19,56	-5,23
Init.	0,34	-0,1	0,28	0,29	0,41	-0,3	-0,16
fin	9,88	-9,09	4,62	9,68	-9,37	-19,5	14,42

Conclusion de la partie 1a

- Illustration de backprop sur le problème du XOR
  - 2 neurones
  - Backprop est une descente de gradient
  - Rumelhart, Hinton, Williams : Learning internal representations by error propagation, (1986)
- A suivre :
  - Propriété théorique d'un unique neurone (partie 1b)
    - Perceptron
  - Réseaux à plusieurs couches (partie 2)
    - Multi-Layer Perceptron (MLP)
    - Deep learning