



Réseau de neurone en C

- le MNIST
- choix du modèle de réseau
- objectifs du groupe
- méthodologie
- difficultés
- démonstration
- questions / réponses

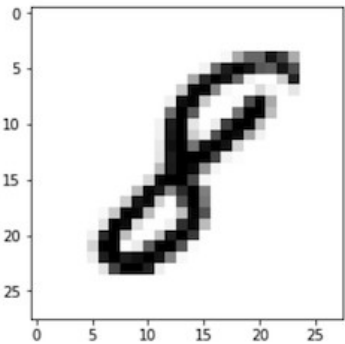
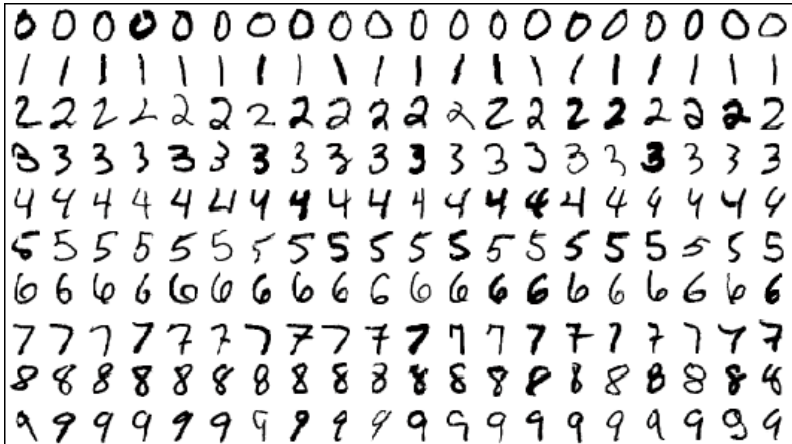


si vis pacem, para bellum





Le MINST



Fichier des bitmaps

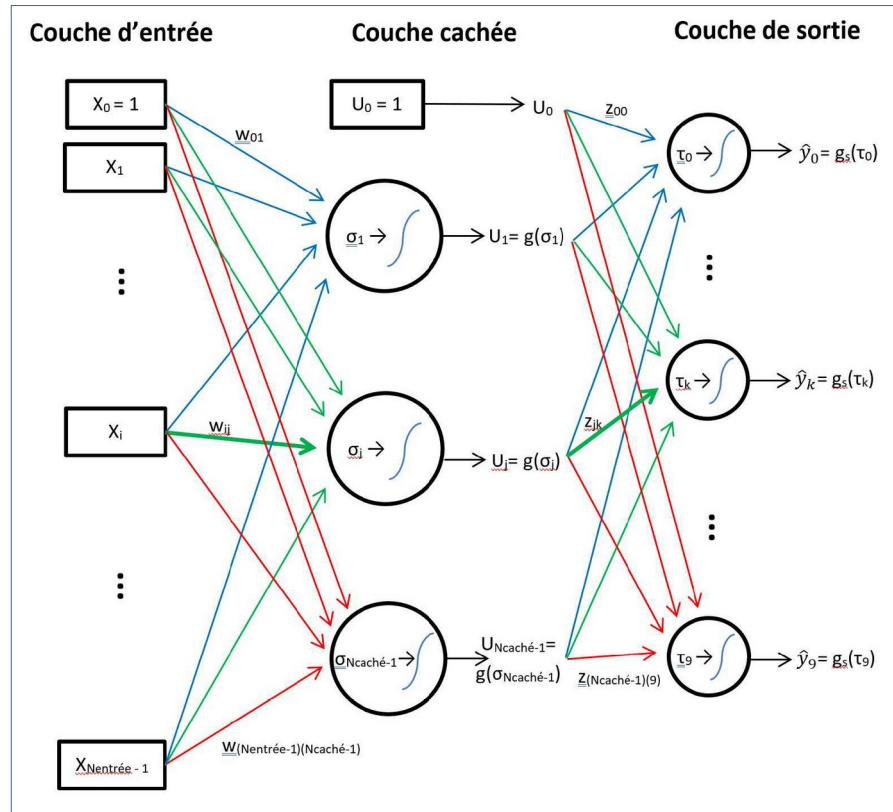
32 bits	32 bits	32 bits	32 bits	28x28	28x28	28x28	28x28	28x28	...
Magic	Nb pic	largeur	hauteur	1 btmp	1btmp	1btmp	1btmp	1btmp

Fichier des labels

32 bits	32 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	...
Magic	Nb pic	label	label	label	label	label	label	label

si vis pacem, para bellum

Choix du modèle de réseau



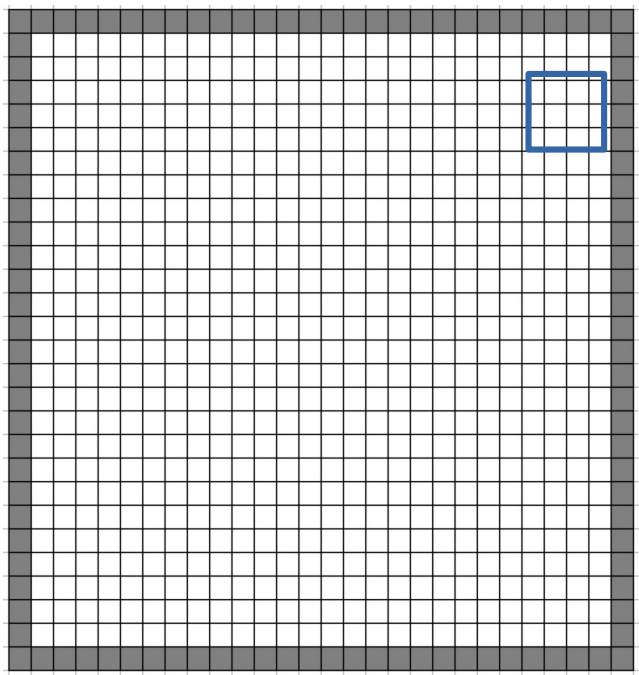
Sortie : 10 neurones

Cachée : $X + 1$ neurones

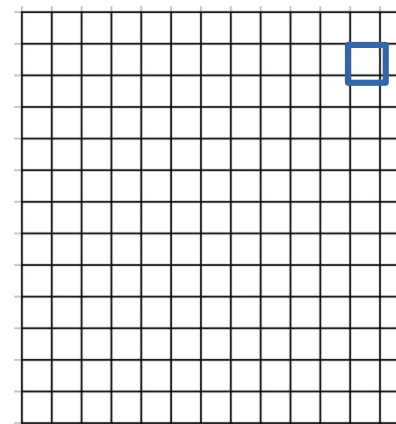


Choix du modèle de réseau

Le MaxPooling :



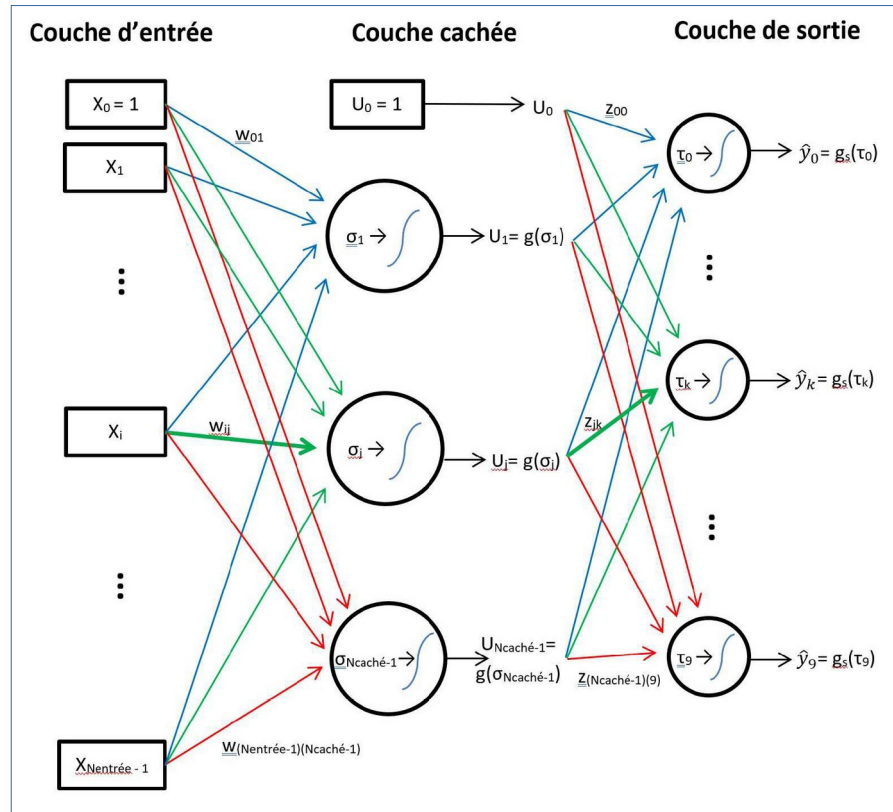
28 x 28



13 x 13

si vis pacem, para bellum

Choix du modèle de réseau



Entrée : 1 bitmap
13 x 13 = 169 pixels
+ 1 biais = 170

Sortie : 10
neurones

Cachée : 65 + 1 neurones

(neurone de biais)

si vis pacem, para bellum



Objectifs du groupe

Développement en 2 temps

1 – propagation simple :

- poids des neurones fournis
- passer les 60 000 images dans le réseau
- vérifier les prédictions

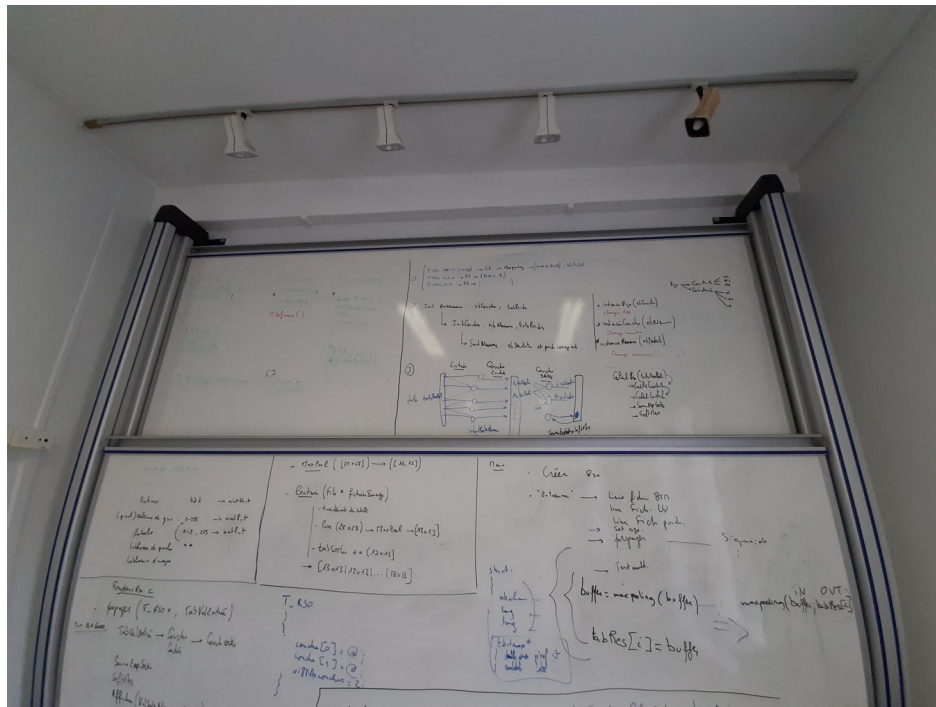
2 – rétro-propagation

- poids initiaux aléatoires
- calcul des coûts / erreurs
- retro correction des poids
- auto apprentissage du réseau

si vis pacem, para bellum



Méthodologie



CADRAGE :

Définition des principales fonctions du programme

Définition des structures à manipuler

Définition des types utiliser / normalisation

si vis pacem, para bellum



Méthodologie

```
→ src ls -la
total 248
drwxr-xr-x 15 maskott staff 480 24 jui 10:27 .
drwxr-xr-x 6 maskott staff 192 24 jui 08:43 ..
-rwxr-xr-x 1 maskott staff 537 23 jui 16:47 Makefile
-rw-r--r-- 1 maskott staff 4276 23 jui 18:30 Util.c
-rw-r--r-- 1 maskott staff 308 23 jui 18:30 Util.h
-rw-r--r--@ 1 maskott staff 6034 23 jui 13:02 calculs.c
-rw-r--r--@ 1 maskott staff 854 23 jui 13:02 calculs.h
-rw-r--r--@ 1 maskott staff 8250 24 jui 09:16 fonctionsRso.c
-rw-r--r-- 1 maskott staff 403 23 jui 18:30 fonctionsRso.h
-rw-r--r--@ 1 maskott staff 6748 24 jui 08:43 fonctionsIO.c
-rw-r--r--@ 1 maskott staff 832 23 jui 16:47 fonctionsIO.h
-rw-r--r--@ 1 maskott staff 1569 24 jui 08:33 main.c
-rwxr-xr-x 1 maskott staff 51840 24 jui 09:16 rso-mnist
-rw-r--r--@ 1 maskott staff 5873 23 jui 16:47 structures.c
-rw-r--r--@ 1 maskott staff 2012 23 jui 18:30 structures.h
→ src
```

STRUCTURE DU PROGRAMME:

Définition des principaux fichiers
Leurs objectifs



Répartition des fichiers au sein du groupe



Mise en place du dépôt git pour les fusions

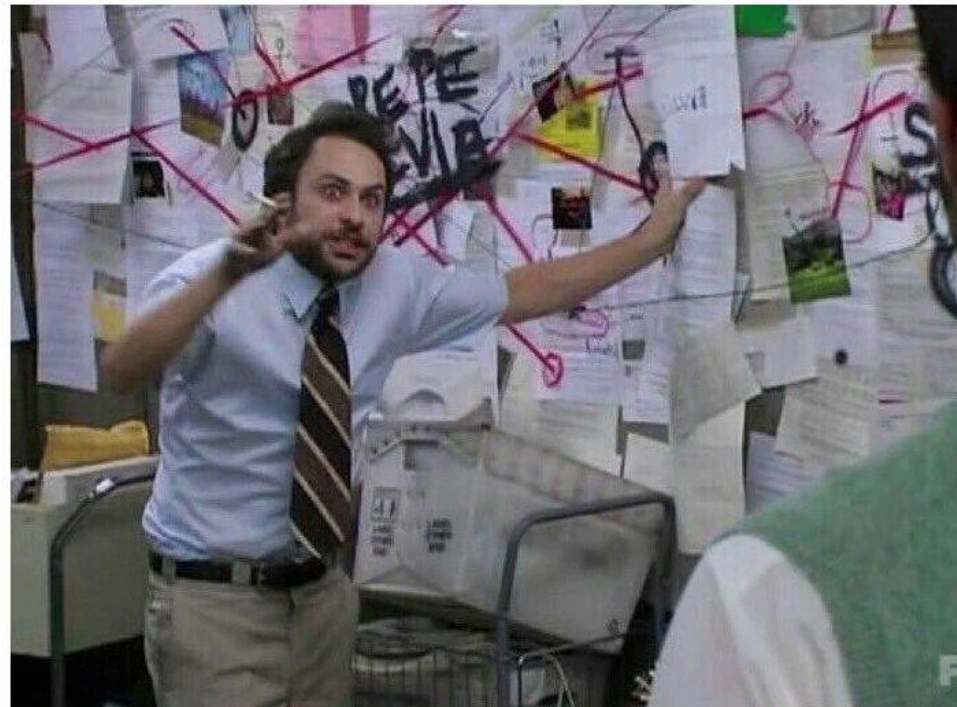
si vis pacem, para bellum



Méthodologie

LUNDI : réflexion

How it looks, when you explain your code to your non-programmer friend



@CodeDoesMeme

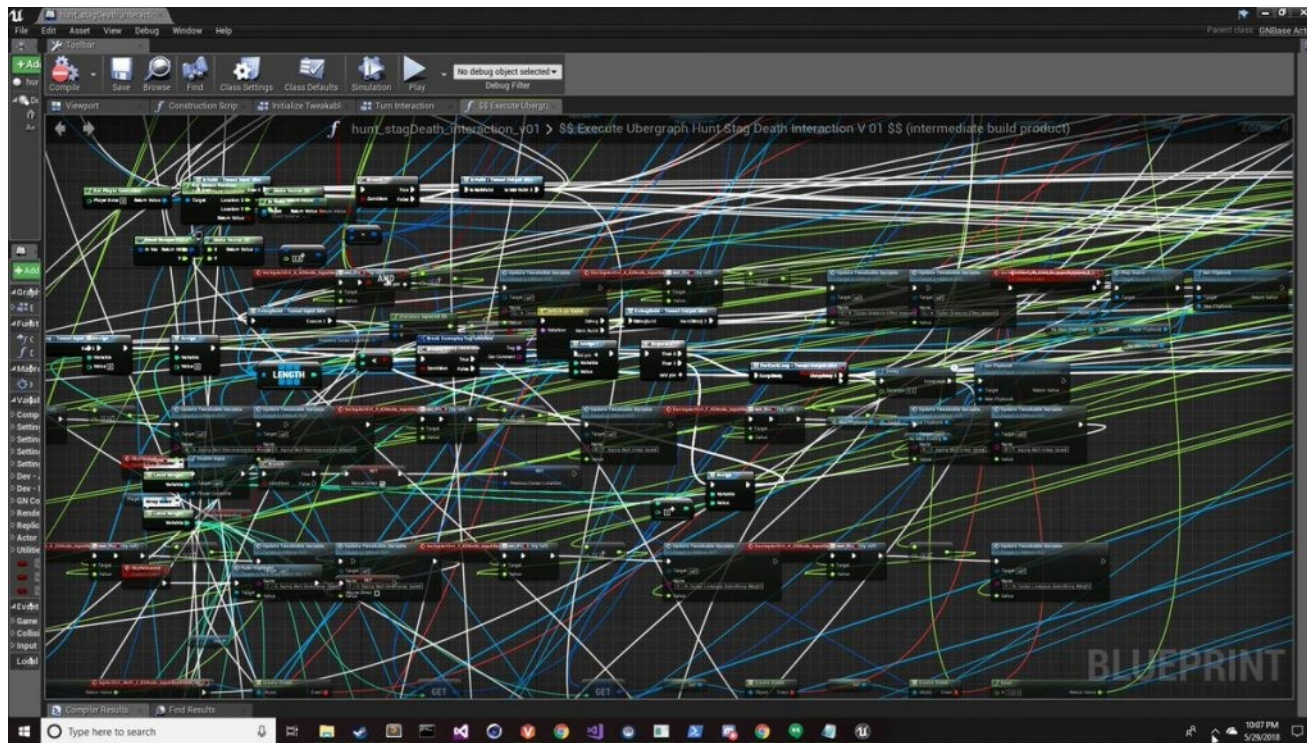
si vis pacem, para bellum



Méthodologie

LUNDI : réflexion

MARDI : codage séparé



si vis pacem, para bellum

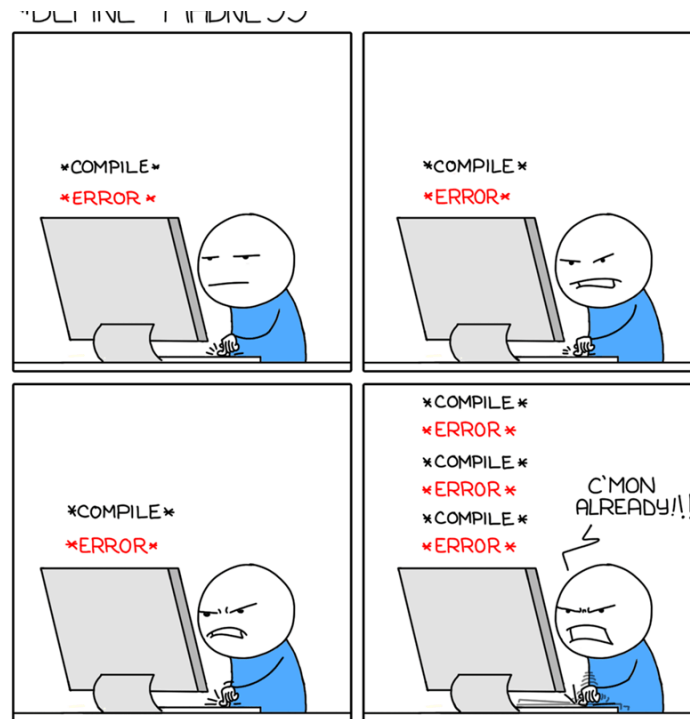


Méthodologie

LUNDI : réflexion

MARDI : codage séparé

MERCREDI : mise en commun, uniformisation des IO, **debugage**



si vis pacem, para bellum



Méthodologie

LUNDI : réflexion

MARDI : codage séparé

MERCREDI : mise en commun, uniformisation des IO, **debugage**

MERCREDI soir : objectif 1 atteint

```
+ src ./rso-mnist
paBitmap = 0x0
pReseau = 0x10d712080
paBitmap = 0x0
Nombre de bitmaps : 60000, p 0x7ffee24f58bc
Instanciation d'un tableau de 60000 T_BITMAP
Tout est instancié
Avant le fclose
cptValLues Final = 11710
Avant Propager
Pointeurs : 0x10d712080 0x7fecb2500000
Nombre de résultats corrects / tests totaux : 56391/60000 soit 93.985000/100
+ src
```

si vis pacem, para bellum



Méthodologie



Sauvegarde du code

Nouvelle branche du git pour l'objectif 2

si vis pacem, para bellum



Méthodologie

JEUDI : itération de la méthode en vue de l'objectif 2

Définition des fonctions / structures / fichier à ajouter ou modifier

Codage et **intégration au fur et à mesure**.



Effectué pour quelques fonctions :
Initialisation des poids aléatoires
Calcul du coût
Statistiques

si vis pacem, para bellum

Difficultés

Le langage C :

- uniformisation des données
(type de variables) → `<stdint.h>`
- manipulation des structures
`*p → *p → *p → valeur`
- gestion de la mémoire : `malloc()` / `free()`

Complexité lié à la structure des réseaux de neurones :

- toutes les pièces du codes sont interdépendantes
- difficulté à séparer les travaux entre individus

Complexité intrinsèque à l'exercice :

- fonctionnement de la retro propagation
- formules des gradients sur les différentes couches

1) rétropropagation sur les z_{00}, z_{10}, z_{20} (corrections à effectuer)

A) si $E^{(k)} = (\hat{y}^{(k)} - y^{(k)})^2$, un calcul identique à celui de page 1 (rétropropagation sur un neurone seul) en substituant U_i à x_i , donne :

$$\Delta z_{i0} = -\alpha \cdot \frac{2}{P} \sum_k g'(\sum_j U_j z_{j0}) \left[1 - g(\sum_j U_j z_{j0}) \right] U_i^{(k)} \cdot (\hat{y}^{(k)} - y^{(k)})$$

$$= -\frac{2\alpha}{P} \sum_k \hat{y}^{(k)} \cdot [1 - \hat{y}^{(k)}] \cdot U_i^{(k)} = -\frac{2\alpha}{P} \sum_k \hat{y}^{(k)} [1 - \hat{y}^{(k)}] \left\{ \frac{1}{U_i^{(k)}} \right\}$$

B) si $C^{(k)} = \text{logloss}$, un calcul identique à la page 1 donne :

$$\Delta z_{i0} = -\alpha \sum_k (\hat{y}^{(k)} - y^{(k)}) U_i^{(k)} = -\alpha \sum_k (\hat{y}^{(k)} - y^{(k)}) \left\{ \frac{1}{U_i^{(k)}} \right\}$$

2) rétropropagation de l'erreur sur U_1 et U_2 (si $C^{(k)} = \text{logloss}$)

$$\frac{\partial C^{(k)}}{\partial U_1} = \frac{\partial C^{(k)}}{\partial \hat{y}^{(k)}} \cdot \frac{\partial \hat{y}^{(k)}}{\partial U_1} \quad \text{avec:} \quad \frac{\partial \hat{y}^{(k)}}{\partial U_1} = g'(\sum_j U_j z_{j0}) \cdot z_{10} = \hat{y}^{(k)} \cdot (1 - \hat{y}^{(k)}) \cdot z_{10}$$

$$= \frac{[\hat{y}^{(k)} - y^{(k)}] \cdot \hat{y}^{(k)} (1 - \hat{y}^{(k)}) z_{10}}{\hat{y}^{(k)} (1 - \hat{y}^{(k)})} = [\hat{y}^{(k)} - y^{(k)}] z_{10}$$

$$\frac{\partial C^{(k)}}{\partial U_2} = [\hat{y}^{(k)} - y^{(k)}] z_{20}$$

De même :

$$\frac{\partial C^{(k)}}{\partial U_1} = [\hat{y}^{(k)} - y^{(k)}] z_{10} \cdot U_1 \cdot (1 - U_1) \quad \text{car } g = \text{sigmoïde donc } g'(a) = U_1 (1 - U_1)$$

$$\frac{\partial C^{(k)}}{\partial U_2} = [\hat{y}^{(k)} - y^{(k)}] z_{20} \cdot U_2 \cdot (1 - U_2) \quad \dots \quad g'(a) = U_2 (1 - U_2)$$



Point positifs

Projet bien dimensionné :

complexe mais permettant de se confronter au C en quelques jours

Utilisation d'outils de travail collaboratif (git)

Apprentissage pour tous devant les « bugs »

si vis pacem, para bellum



Démonstrations

- 1 – Initialisation du réseau avec un ensemble de poids aléatoires entre -2 et 2
- 2 – Initialisation du réseau avec des poids corrects (fournis)
- 3 – Affichage des bitmaps mal identifiés par le réseau

si vis pacem, para bellum



Questions / réponses

si vis pacem, para bellum