Base MNIST

- 60 000 caractères « train »

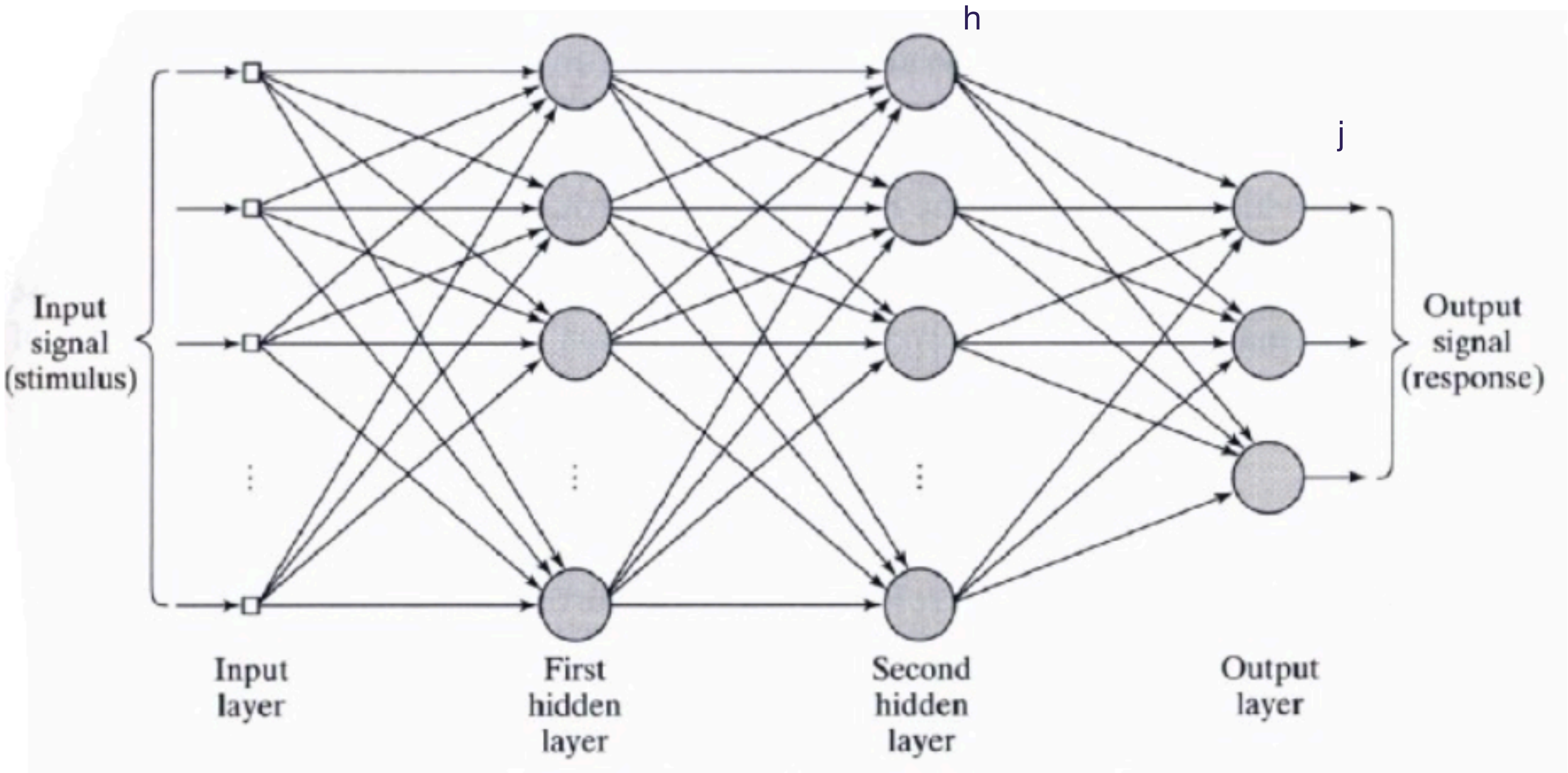
- 10 000 caractères « test »

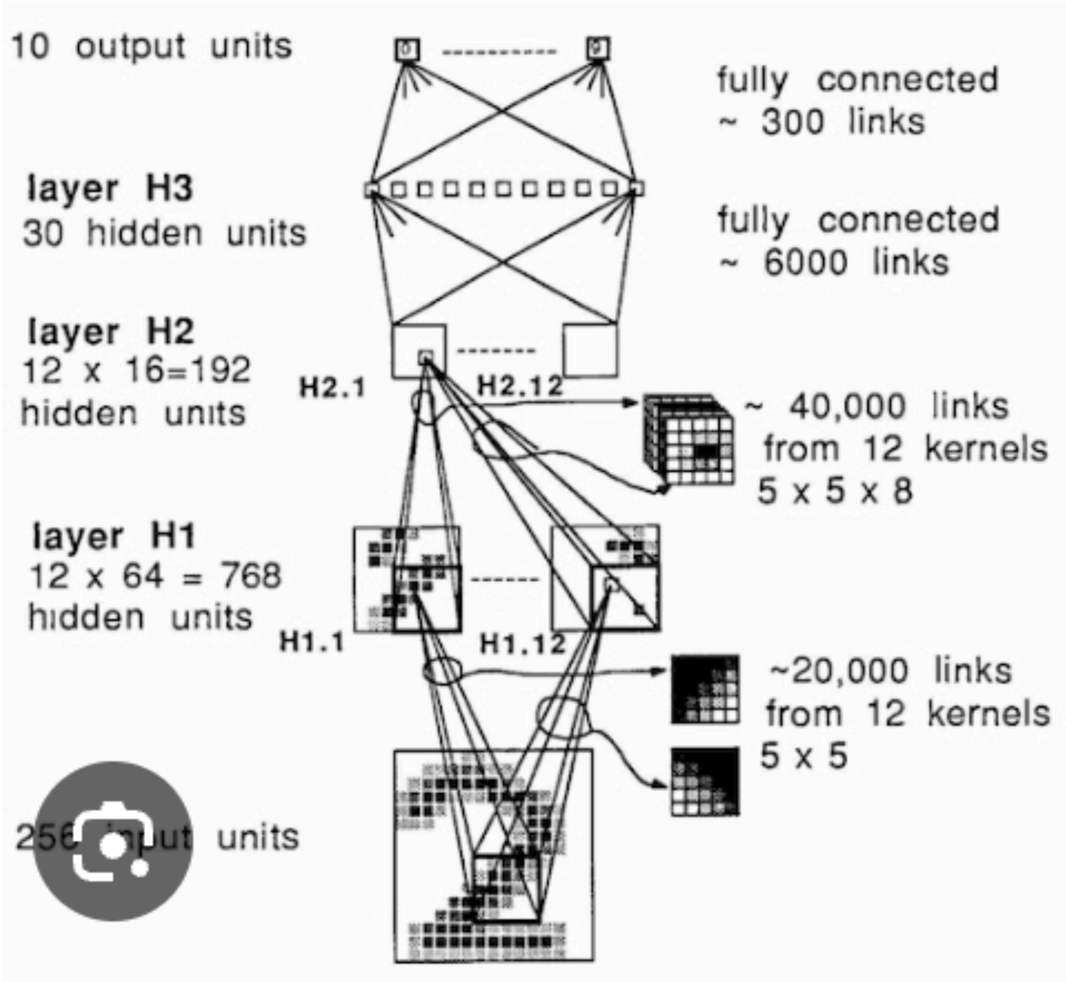
back - propagation

In

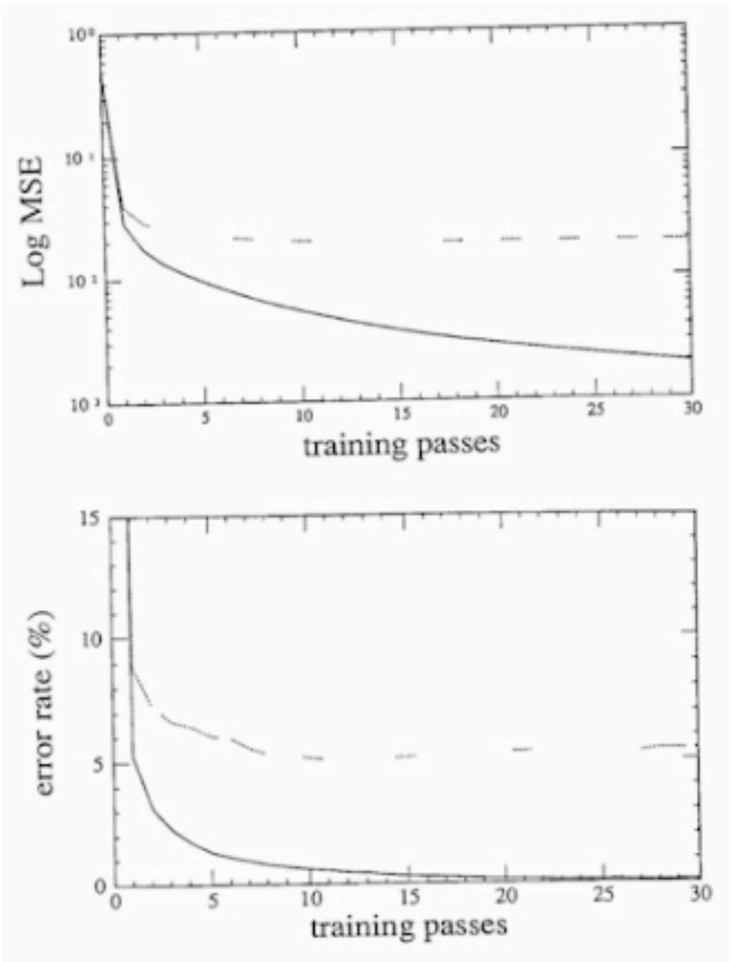
update
hidden

update
output

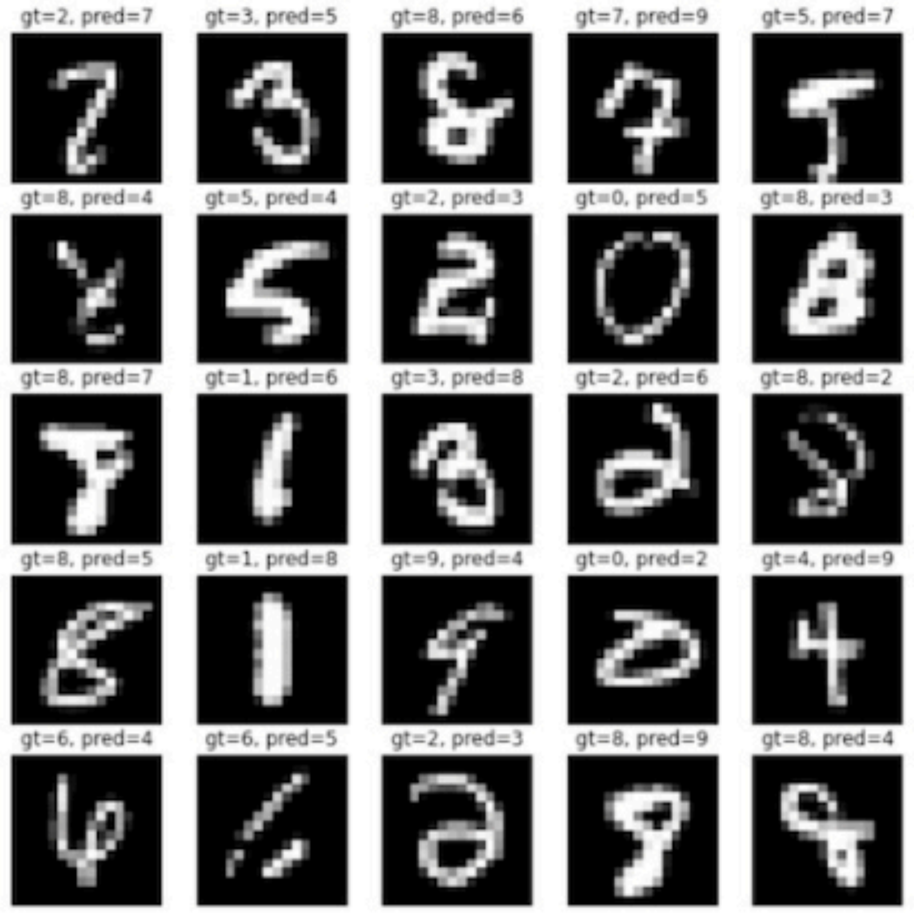




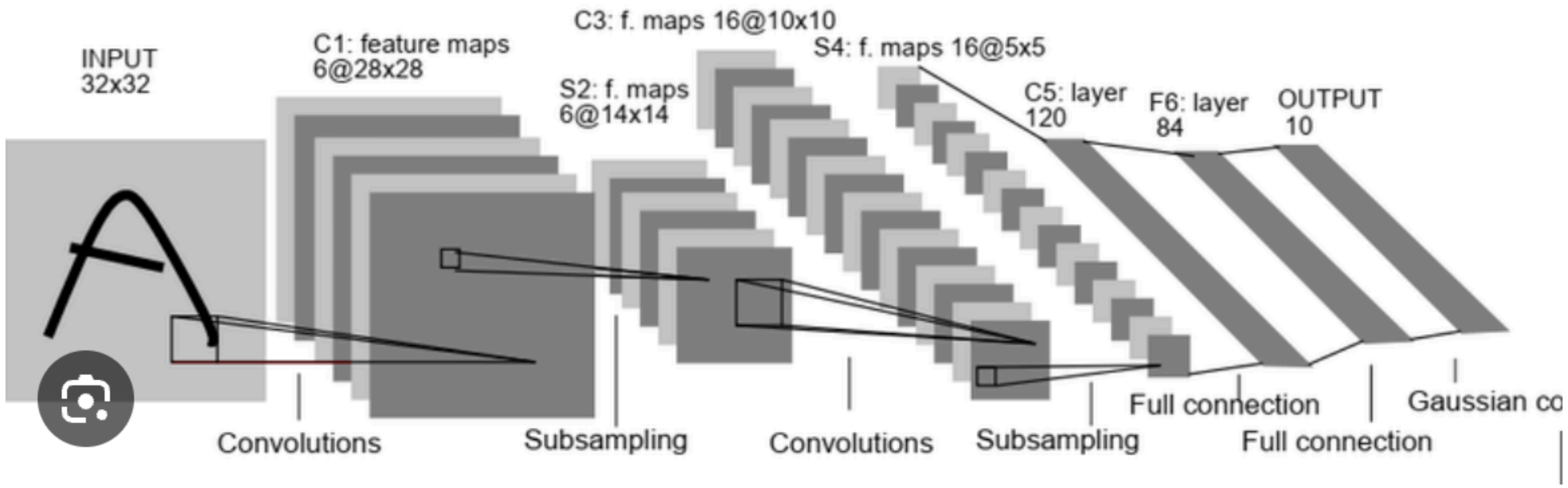
10 output units
fully connected ~ 300 links
layer H3
30 hidden units
fully connected ~ 6000 links
layer H2
12 x 16=192 hidden units
H2.1
H2.12
~ 40,000 links from 12 kernels 5 x 5 x 8
layer H1
12 x 64 = 768 hidden units
H1.1
H1.12
~20,000 links from 12 kernels 5 x 5
units



Log MSE
training passes
error rate (%)
training passes

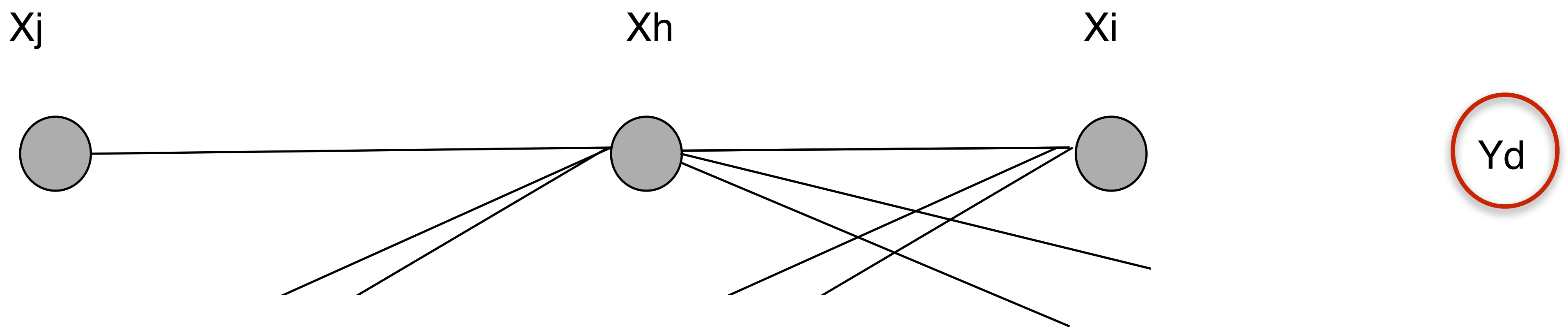


gt=2, pred=7
gt=3, pred=5
gt=8, pred=6
gt=7, pred=9
gt=5, pred=7
gt=8, pred=4
gt=5, pred=4
gt=2, pred=3
gt=0, pred=5
gt=8, pred=3
gt=8, pred=7
gt=1, pred=6
gt=3, pred=8
gt=2, pred=6
gt=8, pred=2
gt=8, pred=5
gt=1, pred=8
gt=9, pred=4
gt=0, pred=2
gt=4, pred=9
gt=6, pred=4
gt=6, pred=5
gt=2, pred=3
gt=8, pred=9
gt=8, pred=4



INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian co

back - propagation



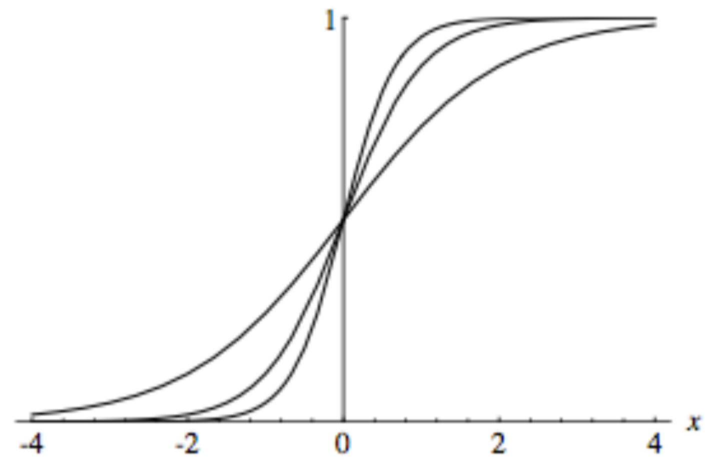


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).

2. Propager sur la rétine : $X_j = \text{Pix}_j/255$.

3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
$pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
$pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

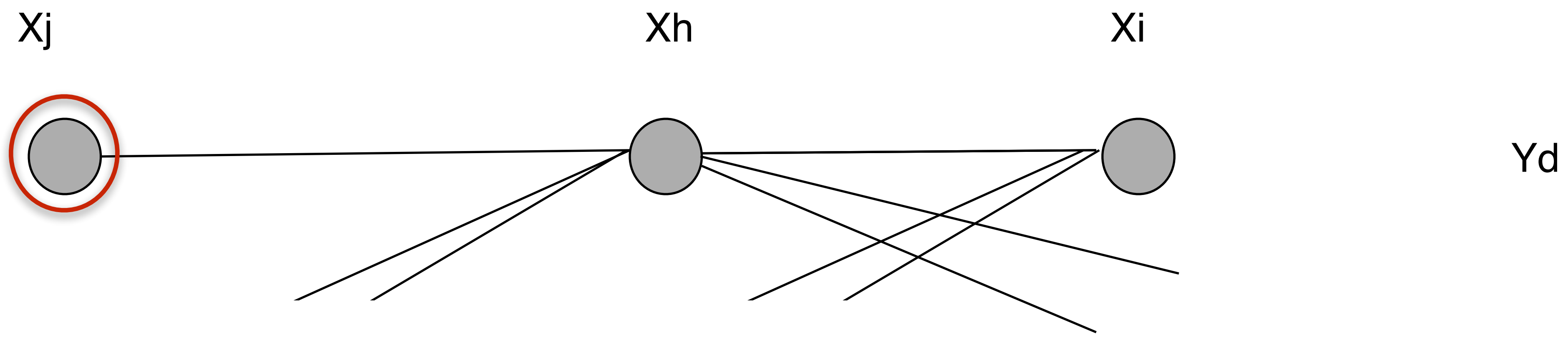
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$

8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



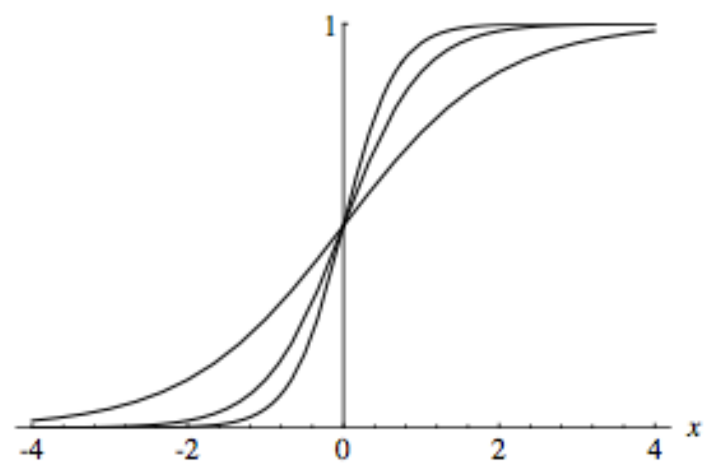


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).

2. Propager sur la rétine : $X_j = \mathrm{Pix}_j/255$.

3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
$pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
$pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

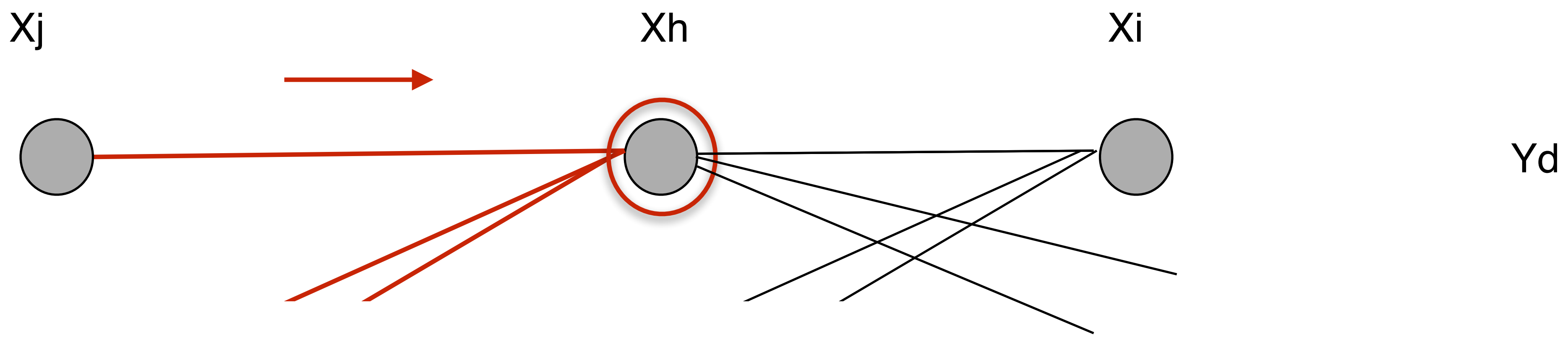
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$

8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



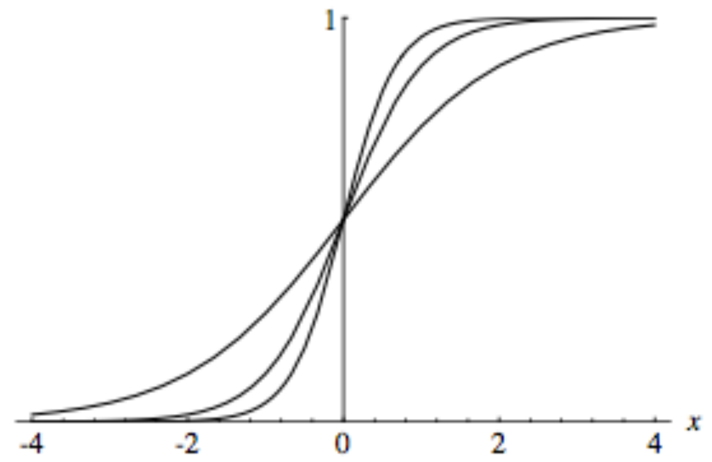


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).

2. Propager sur la rétine : $X_j = \mathrm{Pix}_j/255$.

3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
$pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
$pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

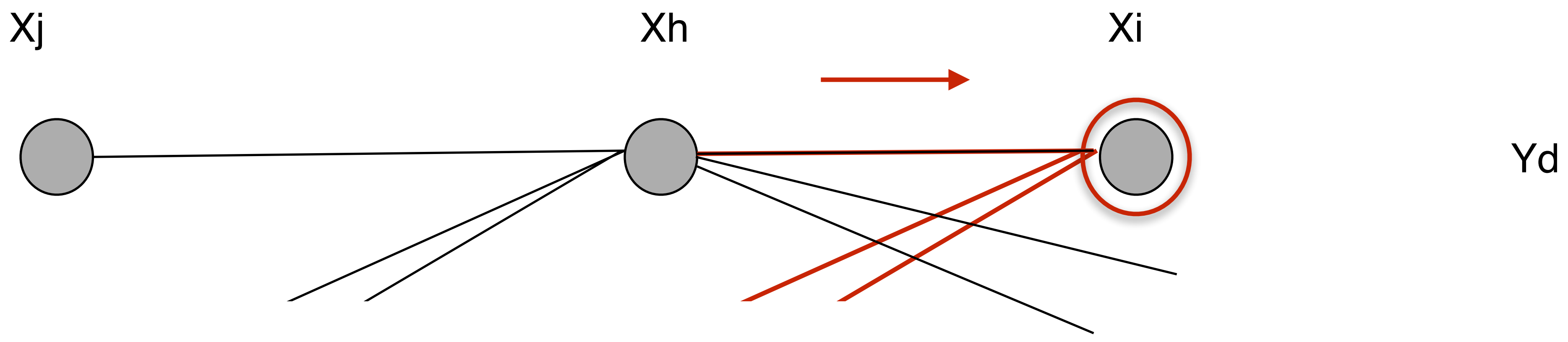
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$

8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



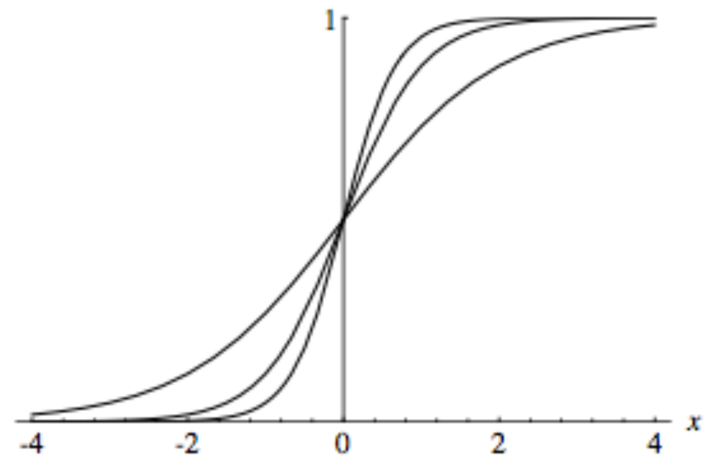


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).

2. Propager sur la rétine : $X_j = \mathrm{Pix}_j/255$.

3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
$pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
$pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

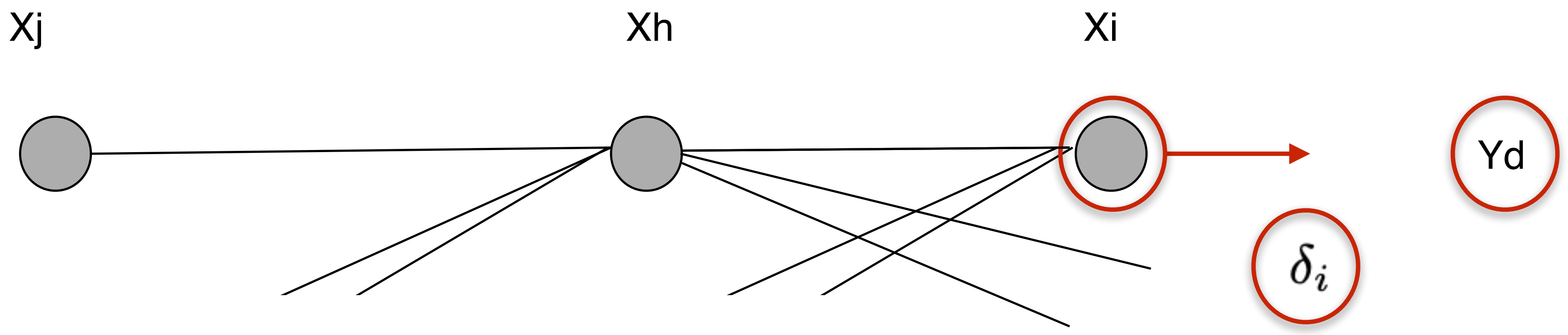
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$

8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



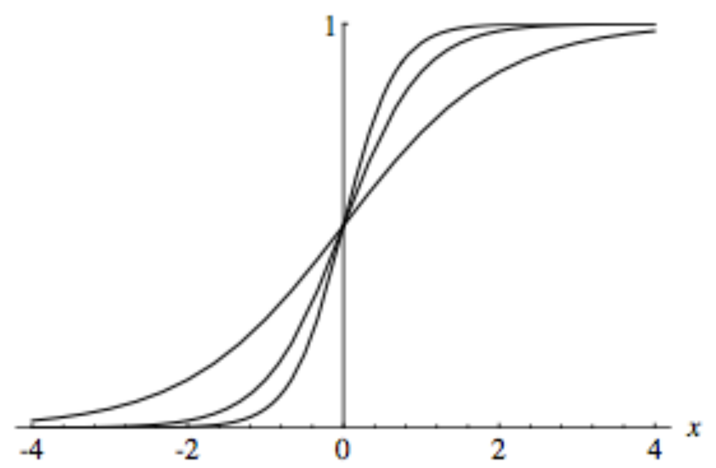
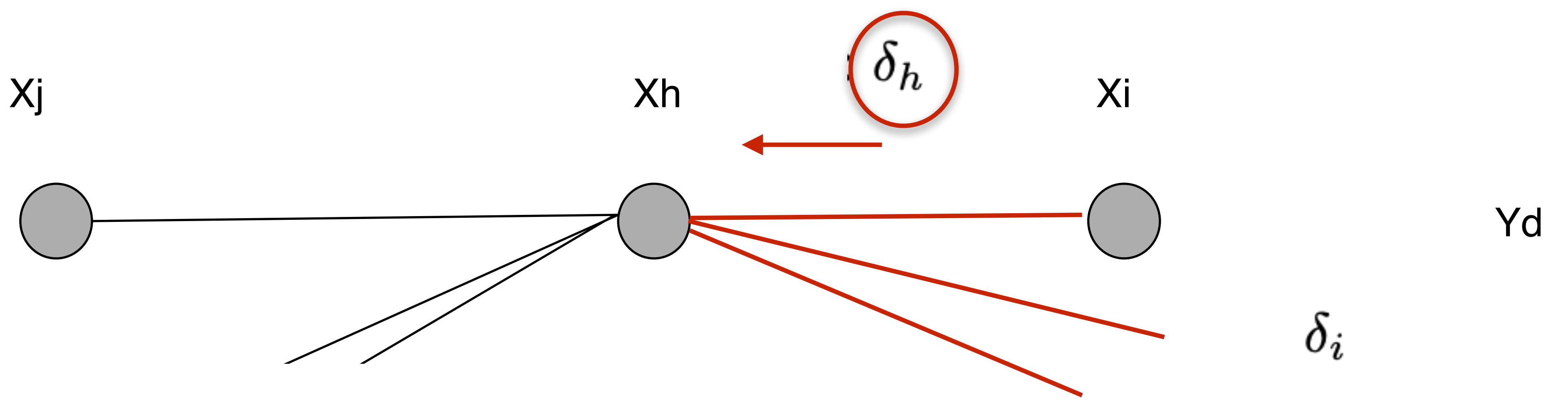


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).
2. Propager sur la rétine : $X_j = \text{Pix}_j/255$.
3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$ $pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.
4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$ $pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$
6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$
7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$
8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



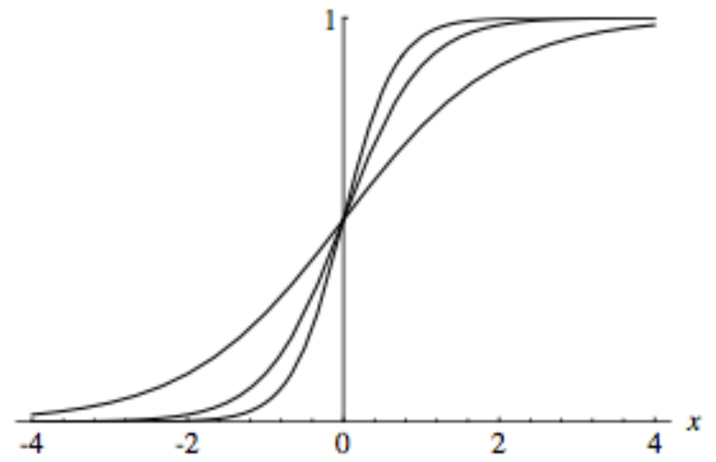


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).

2. Propager sur la rétine : $X_j = \text{Pix}_j/255$.

3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
$pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
$pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

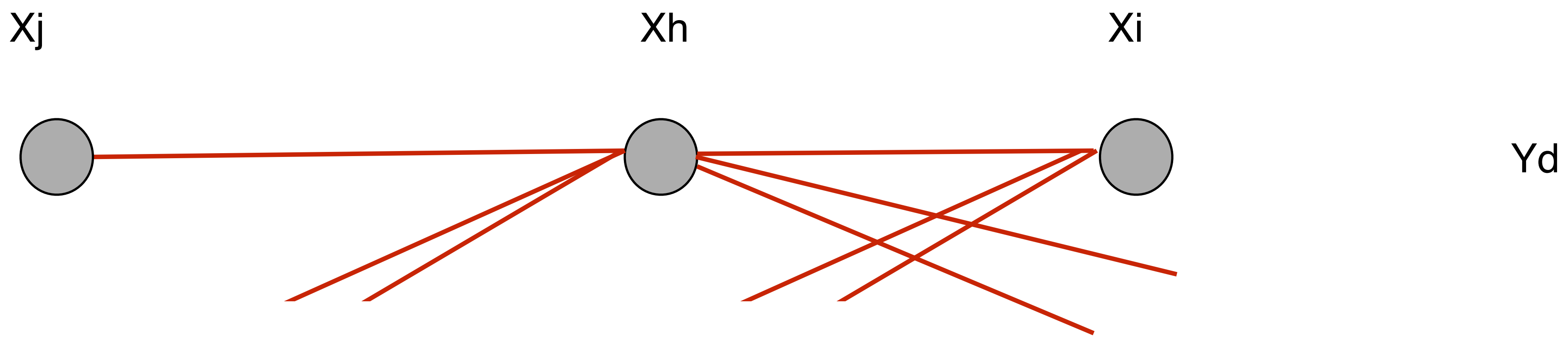
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$

8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



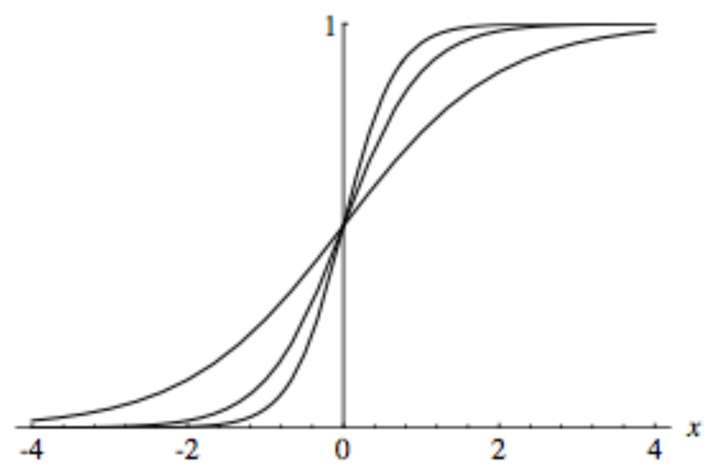


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).

2. Propager sur la rétine : $X_j = \mathrm{Pix}_j/255$.

3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$
$pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$
$pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.

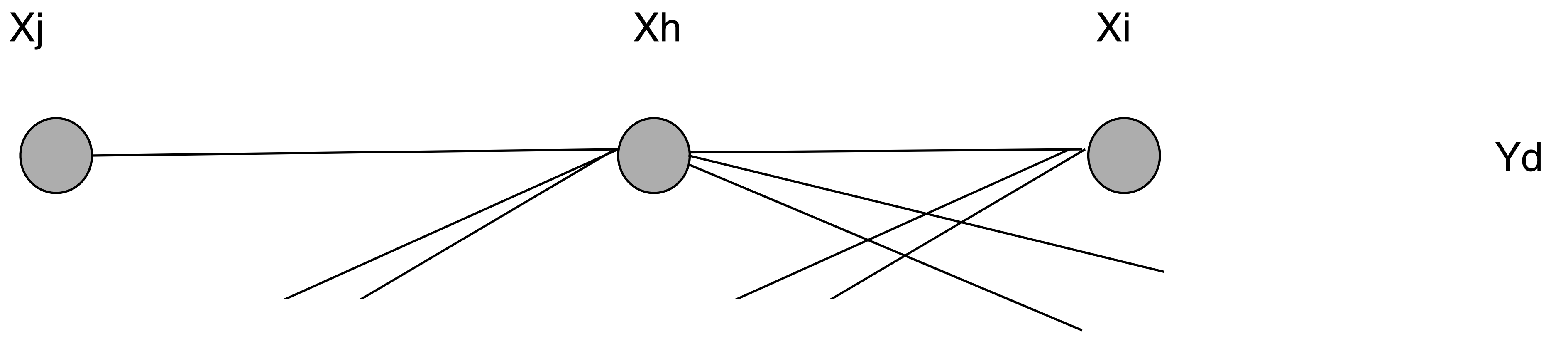
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$
remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$

7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$

8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



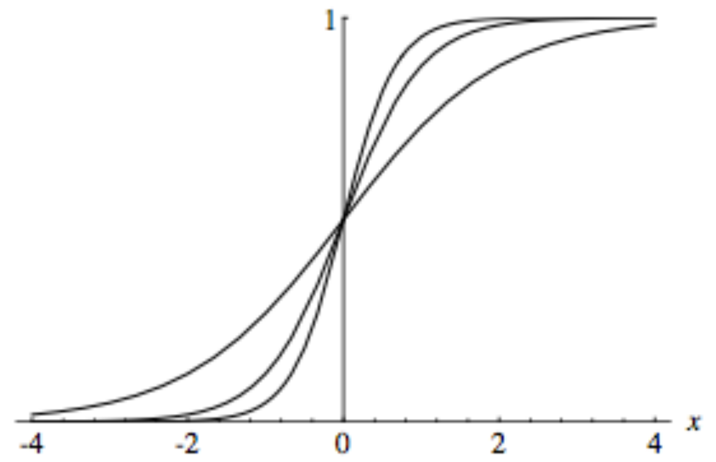
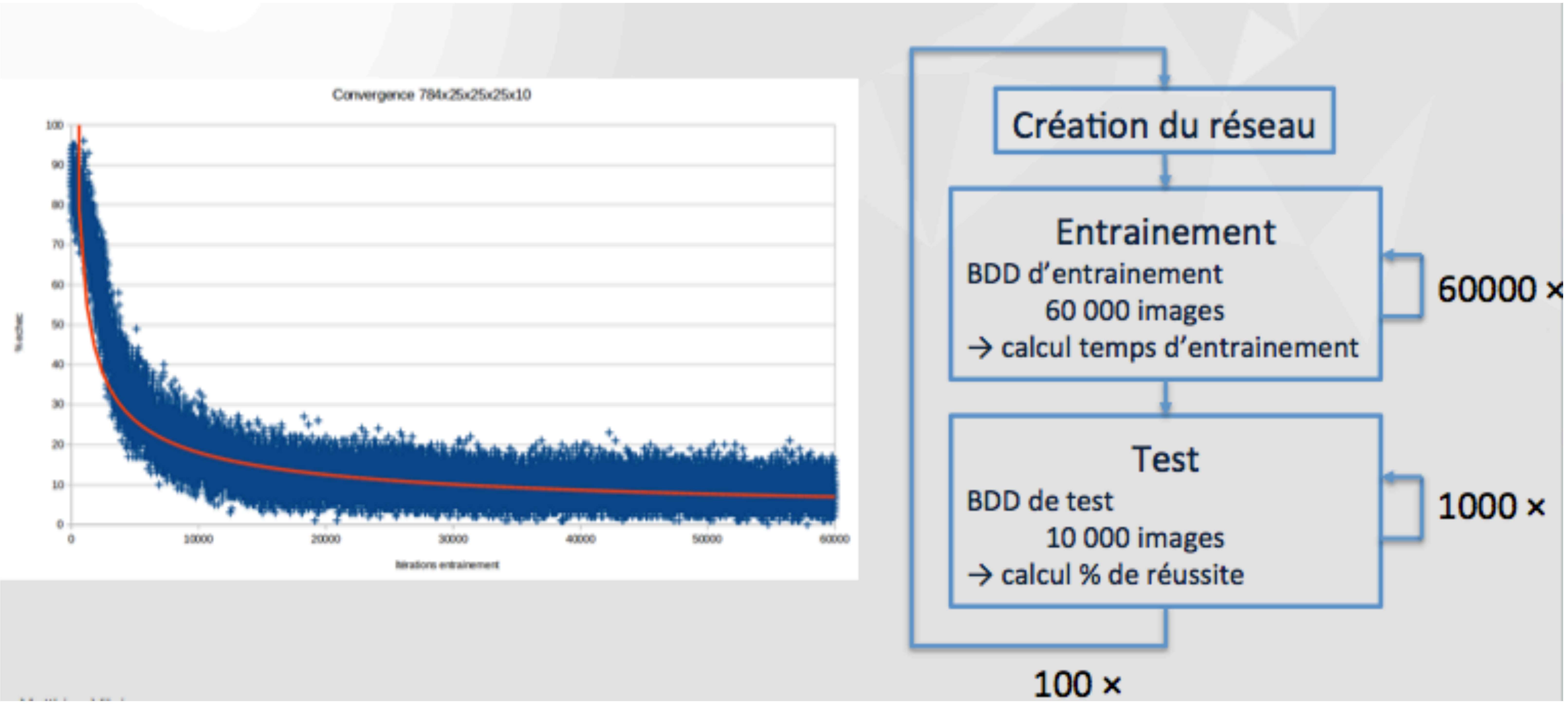


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

Algorithme

1. Choisir une classe au hasard, c'est à dire un motif parmis les 60000 de la base d'entrainement. Charger le motif et mémoriser le Y_d^k correspondant au label (label de la base d'entrainement).
2. Propager sur la rétine : $X_j = \text{Pix}_j/255$.
3. Calculer la sortie des neurones de la couche cachée: $X_h = f(pot_h)$ $pot_h = \sum_j W_{hj} X_j$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.
4. Calculer la sortie des neurones de la couche de sortie: $X_i = f(pot_i)$ $pot_i = \sum_h W_{ih} X_h$ avec $f(x) = \frac{1}{1+e^{-x}}$, dérivable et continue.
5. Pour chaque neurone de la couche de sortie : $\delta_i = f'(pot_i) \cdot Yd_i^k - X_i$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$
6. Pour chaque neurone de la couche cachée : $\delta_h = f'(pot_h) \cdot \sum_i \delta_i \cdot W_{hi}$ remarque : $f(x)$ est dérivable et continue, on a $f'(x) = f(x) \cdot (1 - f(x))$
7. Apprendre : $W_{ij}(t+1) = W_{ij}(t) + \epsilon \cdot \delta_i \cdot X_j$
8. Calculer le pourcentage d'erreur sur p (par exemple $p = 100$) motifs pris au hasard dans la base d'entrainement présentés $Err = \frac{\sum_p \sum_i |\delta_i|}{p}$. si $Err > seuil$ aller à l'étape 1, sinon fin (on doit normalement pouvoir atteindre un seuil de 96% de bonnes réponses sur la base de test).

back - propagation



Applications

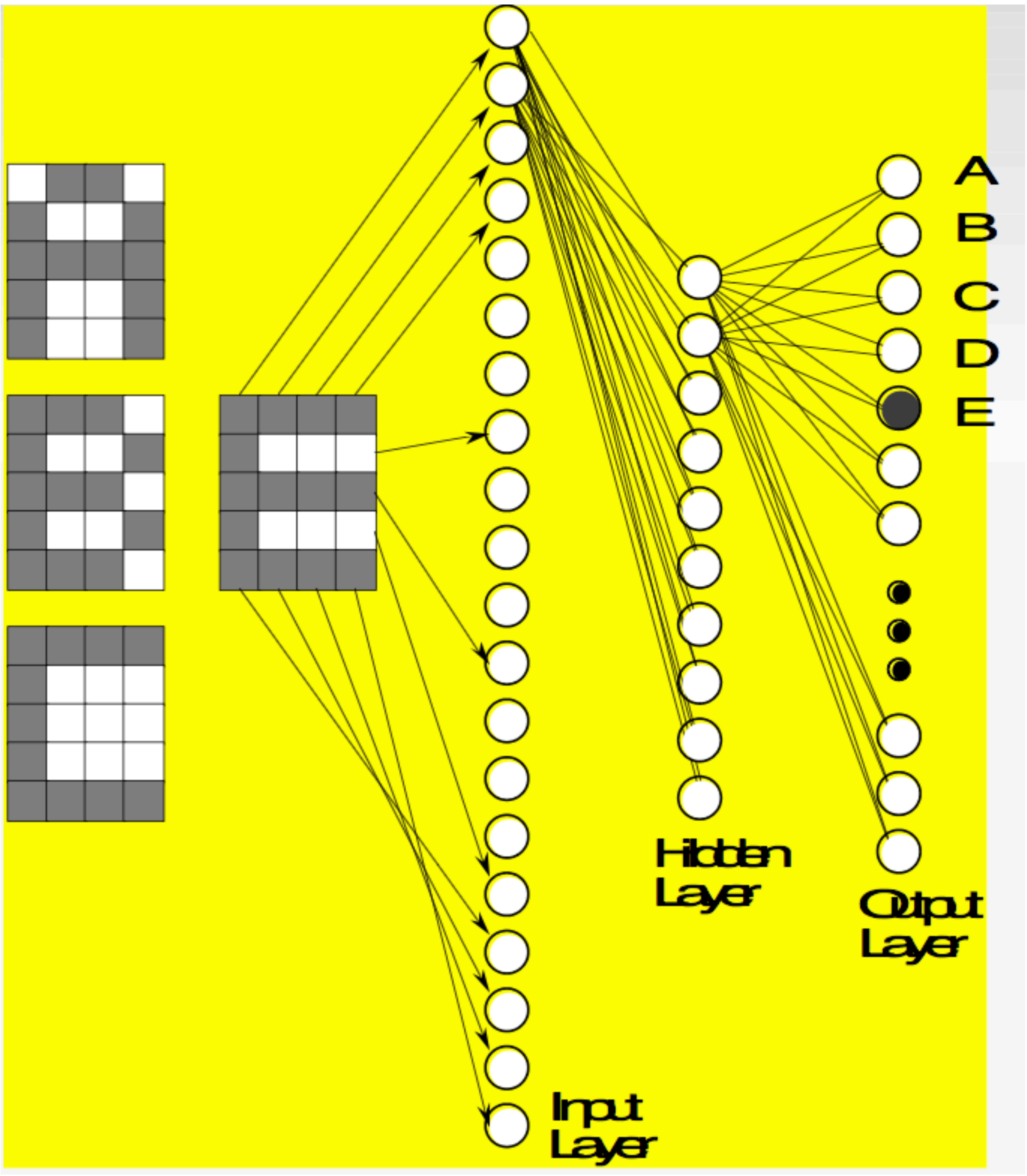
- The properties of neural networks define where they are useful.
  - Can learn complex mappings from inputs to outputs, based solely on samples

  •Difficult to analyse: firm predictions about neural network behaviour difficult:

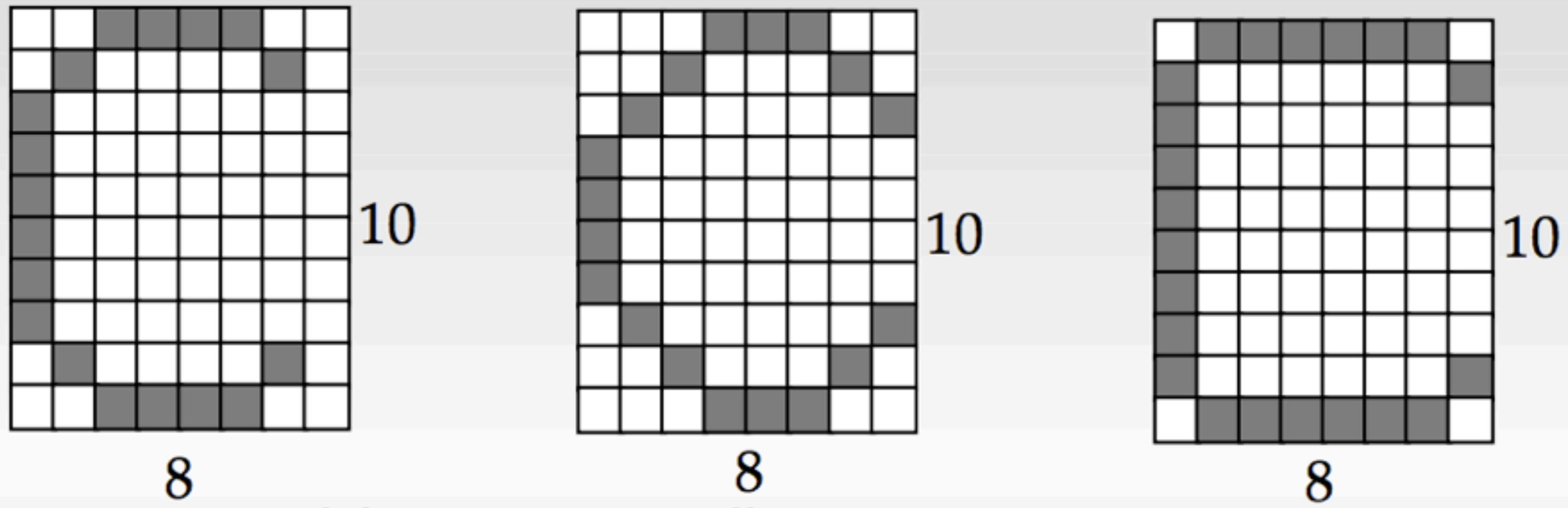
    •Unsuitable for safety-critical applications.

  - Require limited understanding from trainer, who can be guided by heuristics.

Applications



Applications



- NN are able to generalise
- learning involves generating a partitioning of the input space
- for single layer network input space must be linearly separable
- what is the dimension of this input space?
- how many points in the input space?
- this network is binary(uses binary values)
- networks may also be continuous

Applications

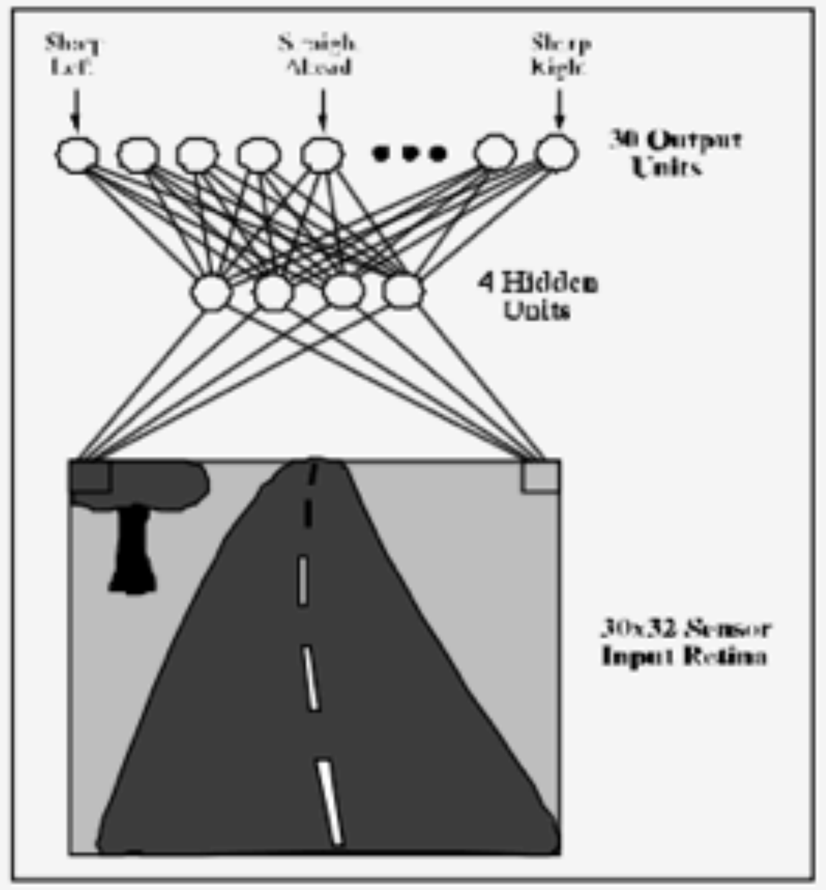
ALVINN

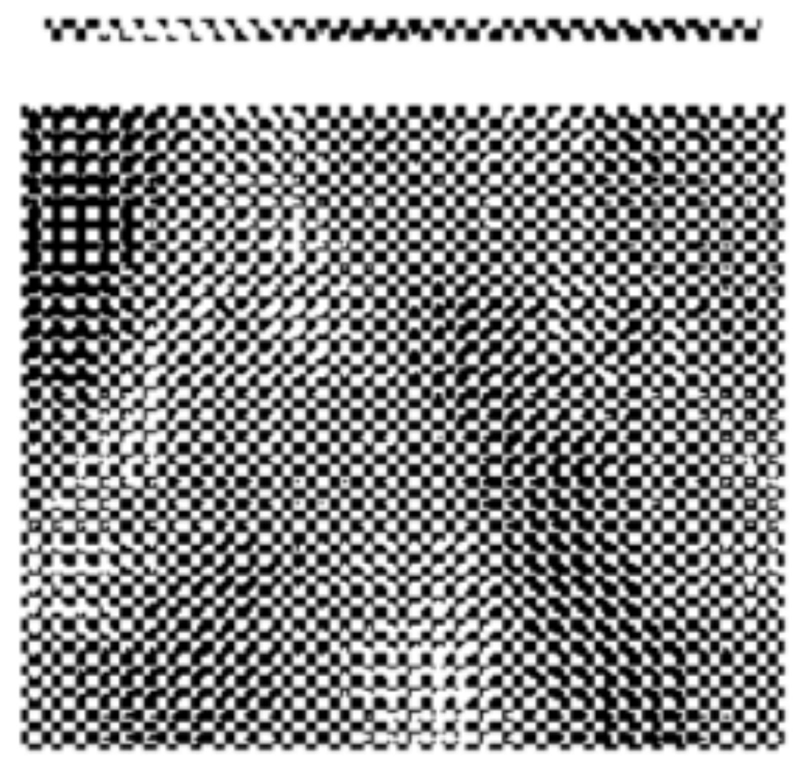
Drives 70 mph on a public highway

30 outputs for steering

4 hidden units

30x32 pixels as inputs





30x32 weights into one out of four hidden unit