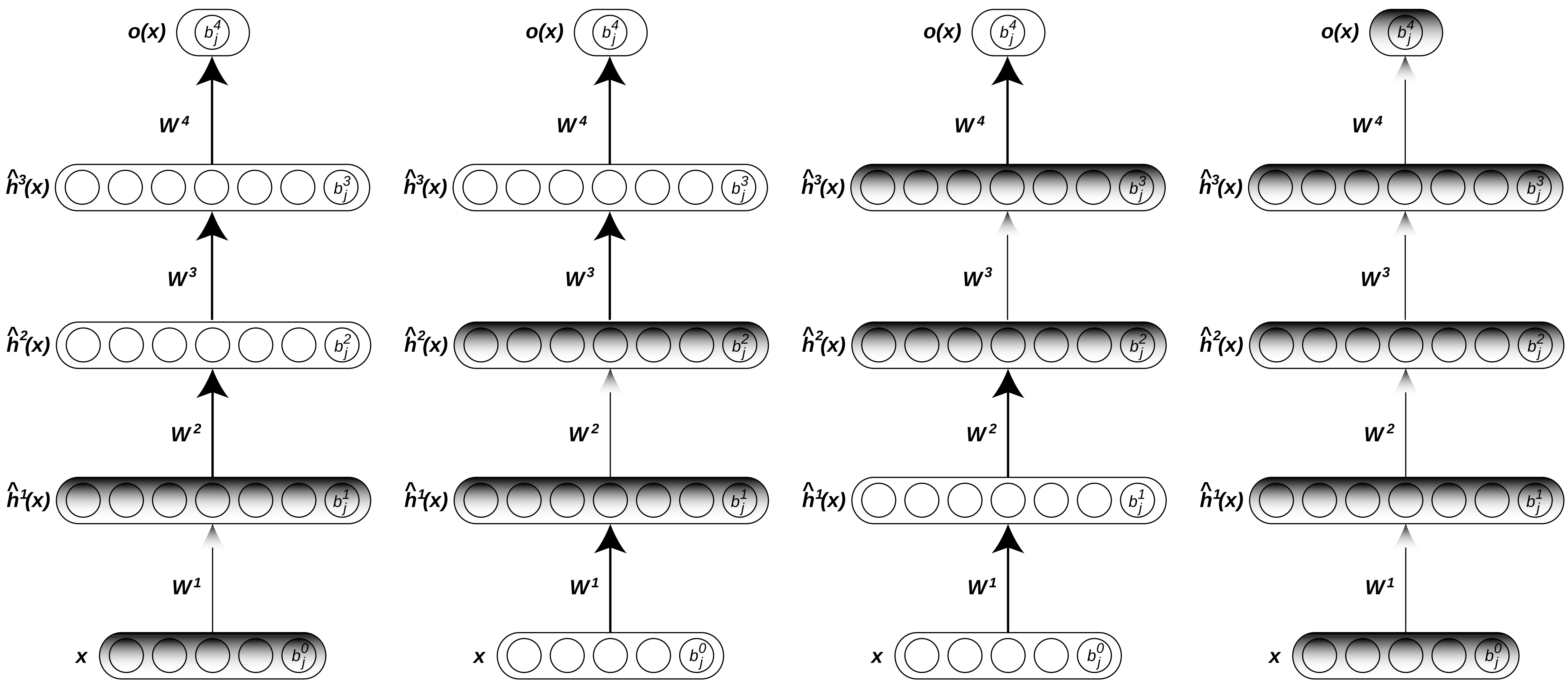
Vue d'ensemble sur l'apprentissage de réseaux profonds

Hugo Larochelle
University of Toronto

Apprentissage de réseaux profonds: la procédure



(a) First hidden layer pre-training

(b) Second hidden layer pre-training

(c) Third hidden layer pre-training

(d) Fine-tuning of whole network

Pseudocode: pré-entraînement

Initialize weights $\mathbf{W}^i_{jk} \sim U(-a^{-0.5}, a^{-0.5})$ with $a = \max(|\widehat{\mathbf{h}}^{i-1}|, |\widehat{\mathbf{h}}^i|)$ and set biases $\mathbf{b}^i$ to 0

% Pre-training phase
for $i \in \{1, \ldots, l\}$ **do**
  while Pre-training stopping criterion is not met **do**
    Pick input example $\mathbf{x}_t$ from training set
    $\widehat{\mathbf{h}}^0(\mathbf{x}_t) \leftarrow \mathbf{x}_t$
    for $j \in \{1, \ldots, i-1\}$ **do**
      $\mathbf{a}^j(\mathbf{x}_t) = \mathbf{b}^j + \mathbf{W}^j \widehat{\mathbf{h}}^{j-1}(\mathbf{x}_t)$
      $\widehat{\mathbf{h}}^j(\mathbf{x}_t) = \mathrm{sigm}\left(\mathbf{a}^j(\mathbf{x}_t)\right)$
    end for
    Using $\widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)$ as input example, update weights $\mathbf{W}^i$ and biases $\mathbf{b}^{i-1}$, $\mathbf{b}^i$ with learning rate $\varepsilon_{\text{pre-train}}$ according to a layer-wise unsupervised criterion (see pseudocodes in appendices B and C)
  end while
end for

Pseudocode: raffinement supervisé

% Fine-tuning phase
while Fine-tuning stopping criterion is not met **do**
  Pick input example $(\mathbf{x}_t, y_t)$ from training set

  % Forward propagation
  $\widehat{\mathbf{h}}^0(\mathbf{x}_t) \leftarrow \mathbf{x}_t$
  for $i \in \{1, \ldots, l\}$ **do**
    $\mathbf{a}^i(\mathbf{x}_t) = \mathbf{b}^i + \mathbf{W}^i \widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)$
    $\widehat{\mathbf{h}}^i(\mathbf{x}_t) = \mathrm{sigm}\left(\mathbf{a}^i(\mathbf{x}_t)\right)$
  end for
  $\mathbf{a}^{l+1}(\mathbf{x}_t) = \mathbf{b}^{l+1} + \mathbf{W}^{l+1} \widehat{\mathbf{h}}^l(\mathbf{x}_t)$
  $\mathbf{o}(\mathbf{x}_t) = \widehat{\mathbf{h}}^{l+1}(\mathbf{x}_t) = \mathrm{softmax}\left(\mathbf{a}^{l+1}(\mathbf{x}_t)\right)$

⋮

Pseudocode: raffinement supervisé

⋮

% Backward gradient propagation and parameter update

$\frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial a_j^{l+1}(\mathbf{x}_t)} \leftarrow \mathbf{1}_{y_t=j} - o_j(\mathbf{x}_t)$ **for** $j \in \{1,\ldots,K\}$

$\mathbf{b}^{l+1} \leftarrow \mathbf{b}^{l+1} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{l+1}(\mathbf{x}_t)}$

$\mathbf{W}^{l+1} \leftarrow \mathbf{W}^{l+1} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{l+1}(\mathbf{x}_t)} \widehat{\mathbf{h}}^{l}(\mathbf{x}_t)^{\mathsf{T}}$

for $i \in \{1,\ldots,l\}$, in decreasing order **do**

$\quad \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \widehat{\mathbf{h}}^{i}(\mathbf{x}_t)} \leftarrow \left(\mathbf{W}^{i+1}\right)^{\mathsf{T}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{i+1}(\mathbf{x}_t)}$

$\quad \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial a_j^{i}(\mathbf{x}_t)} \leftarrow \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \widehat{h}_j^{i}(\mathbf{x}_t)} \widehat{h}_j^{i}(\mathbf{x}_t)\left(1-\widehat{h}_j^{i}(\mathbf{x}_t)\right)$ **for** $j \in \{1,\ldots,|\widehat{\mathbf{h}}^{i}|\}$

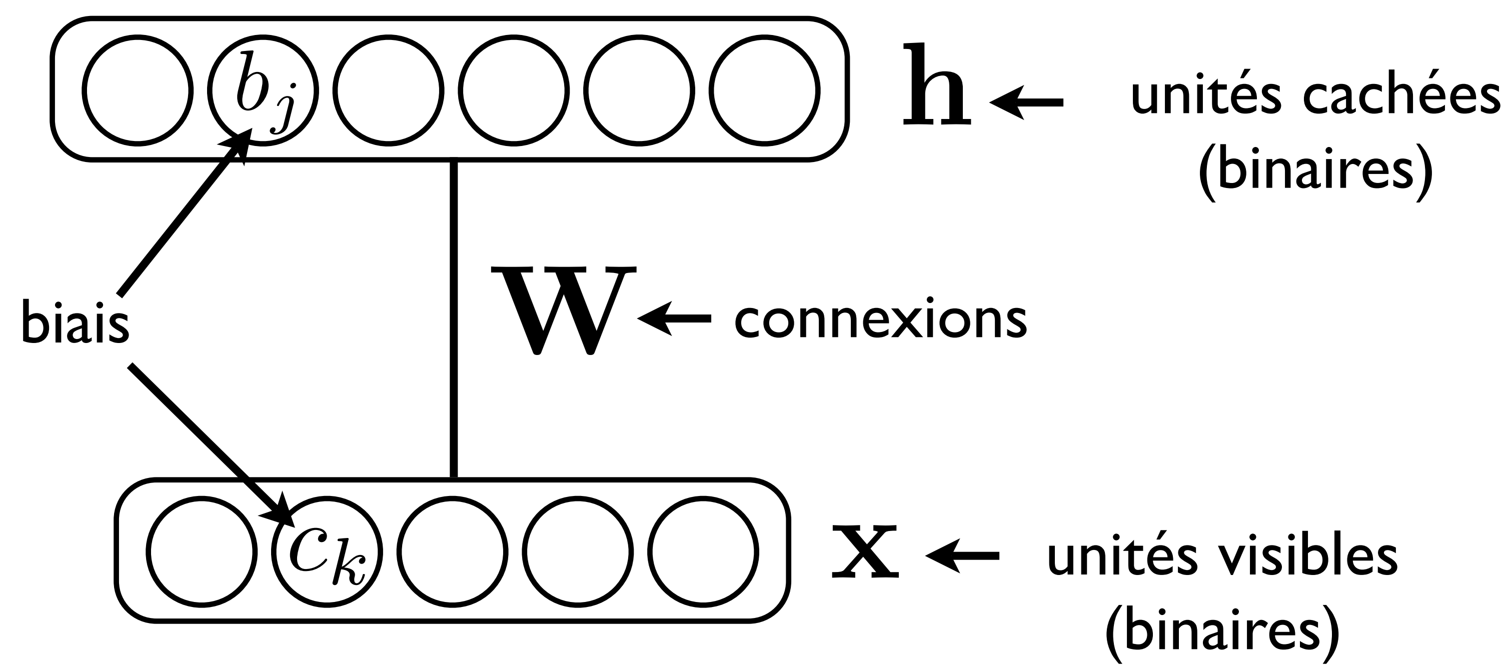
$\quad \mathbf{b}^{i} \leftarrow \mathbf{b}^{i} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{i}}$

$\quad \mathbf{W}^{i} \leftarrow \mathbf{W}^{i} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{i}} \widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)^{\mathsf{T}}$

end for

end while

Restricted Boltzmann Machine

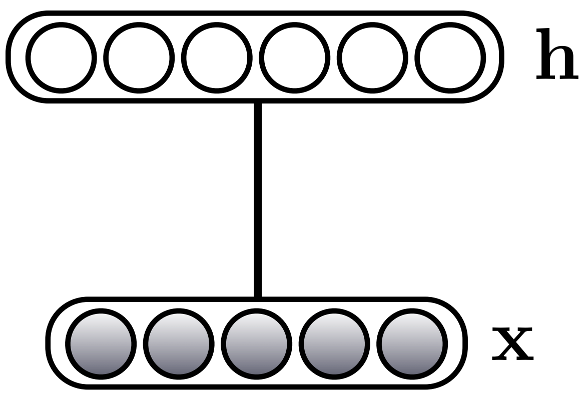


Fonction d'énergie :

$$E(\mathbf{x},\mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$
$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution: $p(\mathbf{x},\mathbf{h}) = \exp(-E(\mathbf{x},\mathbf{h}))/Z$

Inférence

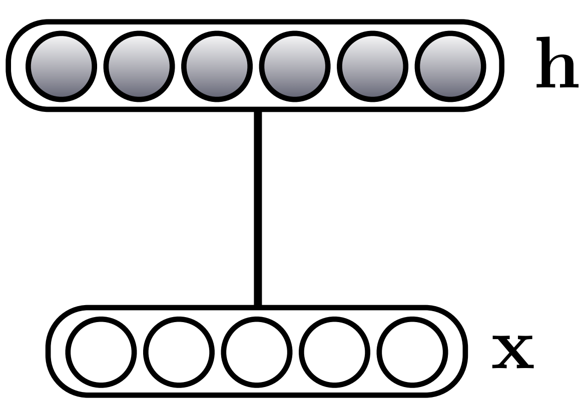


$$p(\mathbf{h}|\mathbf{x}) = \prod_j p(h_j|\mathbf{x})$$

$$p(h_j = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(b_j + \mathbf{W}_{j\cdot}\mathbf{x}))}$$

$$= \mathrm{sigm}(b_j + \mathbf{W}_{j\cdot}\mathbf{x})$$

(montrer preuve au tableau)

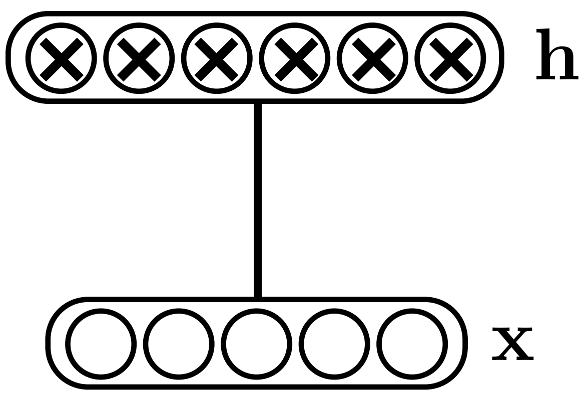


$$p(\mathbf{x}|\mathbf{h}) = \prod_k p(x_k|\mathbf{h})$$

$$p(x_k = 1|\mathbf{h}) = \frac{1}{1 + \exp(-(c_k + \mathbf{h}^\top \mathbf{W}_{\cdot k}))}$$

$$= \mathrm{sigm}(c_k + \mathbf{h}^\top \mathbf{W}_{\cdot k})$$

Inférence



$$p(\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^H} p(\mathbf{x},\mathbf{h}) = \sum_{\mathbf{h}\in\{0,1\}^H} \exp(-E(\mathbf{x},\mathbf{h}))/Z$$

$$\sum_{\mathbf{h}\in\{0,1\}^H} \exp(-E(\mathbf{x},\mathbf{h}))/Z = \sum_{h_1\in\{0,1\}} \cdots \sum_{h_H\in\{0,1\}} \exp(-E(\mathbf{x},\mathbf{h}))/Z$$

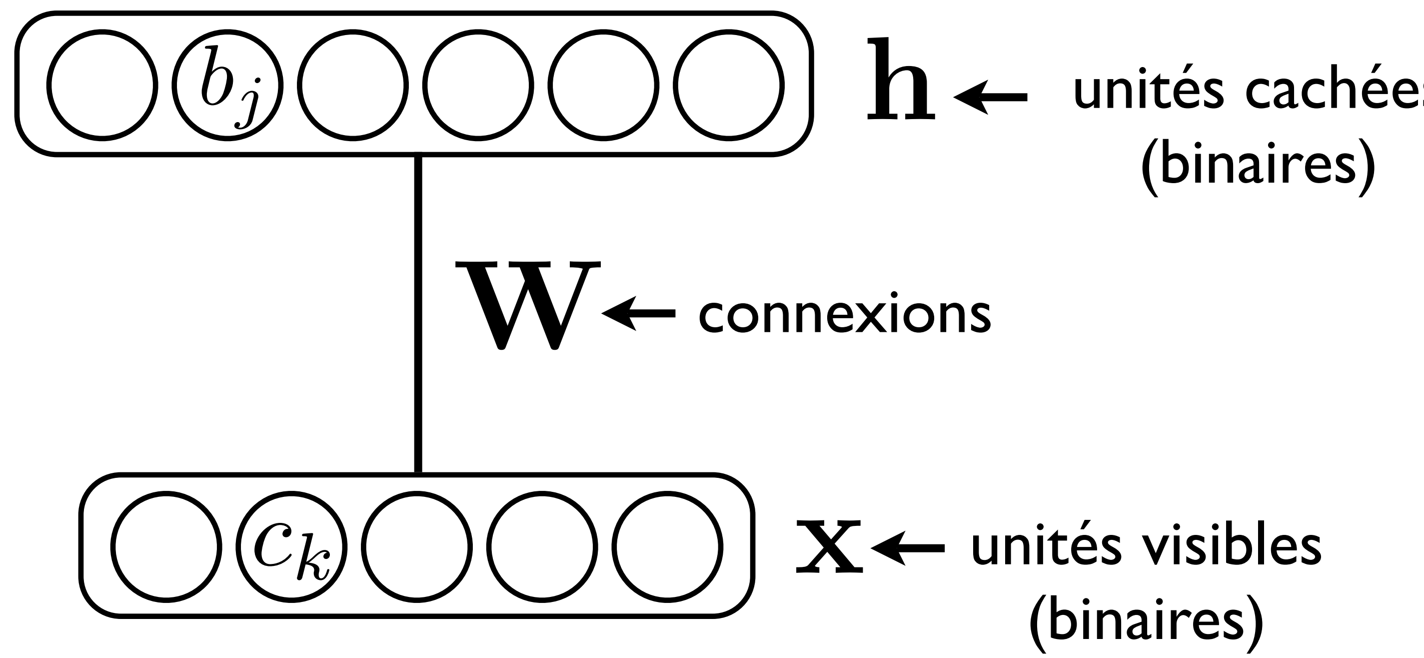
“free energy”

$$= \exp\left(\sum_{j=1}^{H} \log(1+\exp(b_j + \mathbf{W}_{j}.\mathbf{x}))\right)/Z_F$$

$$= \exp(-F(\mathbf{x}))/Z_F$$

(montrer preuve au tableau)

Classification Restricted Boltzmann Machine

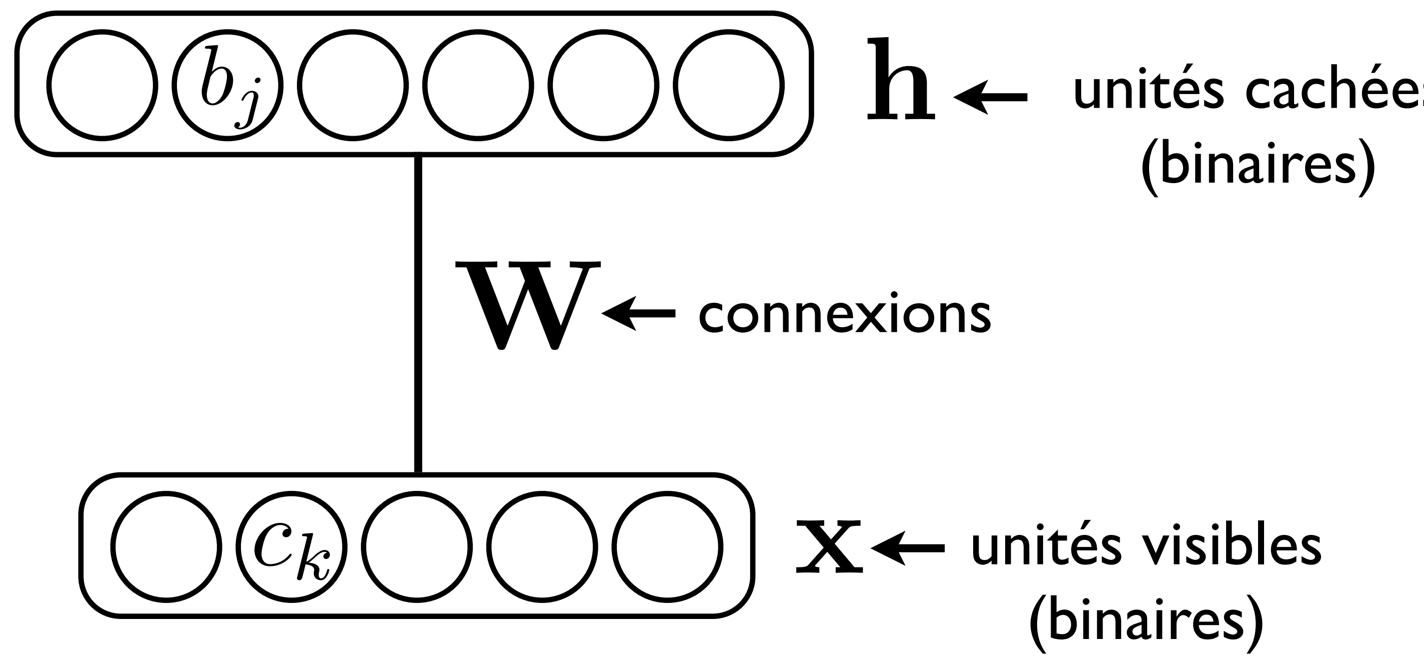


Fonction d'énergie :

$$E(\mathbf{x},\mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$
$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution: $p(\mathbf{x},\mathbf{h}) = \exp(-E(\mathbf{x},\mathbf{h}))/Z$

Classification Restricted Boltzmann Machine



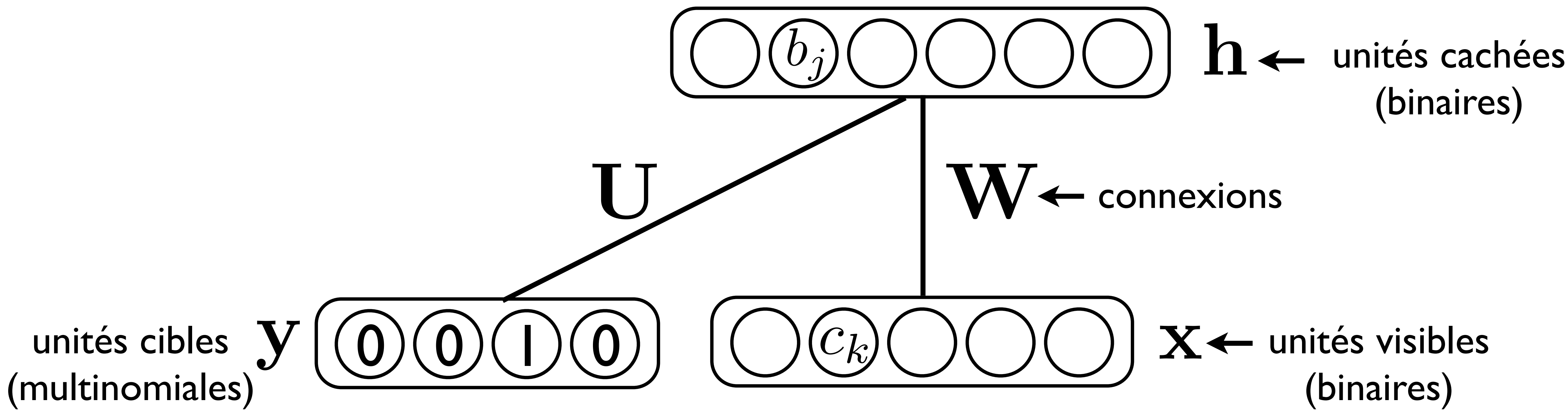
Fonction d'énergie :

$$= -\mathbf{h}^\top \mathbf{W} \mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$

$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution:

Classification Restricted Boltzmann Machine



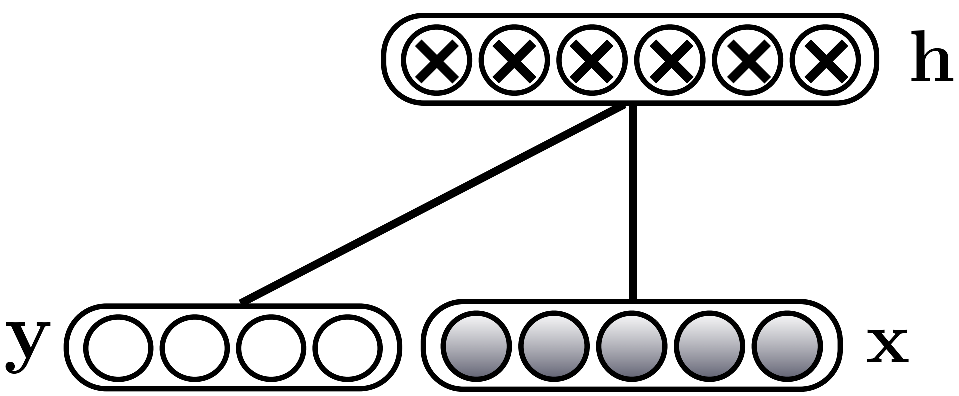
Fonction d'énergie :

$$E(\mathbf{x},\mathbf{h},\mathbf{y}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{U}\mathbf{y} - \mathbf{a}^\top \mathbf{y}$$

$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j - \sum_{jl} U_{jl} h_j y_l - \sum_l a_l y_l$$

Distribution: $p(\mathbf{x},\mathbf{h},\mathbf{y}) = \exp(-E(\mathbf{x},\mathbf{h},\mathbf{y}))/Z$

Inférence



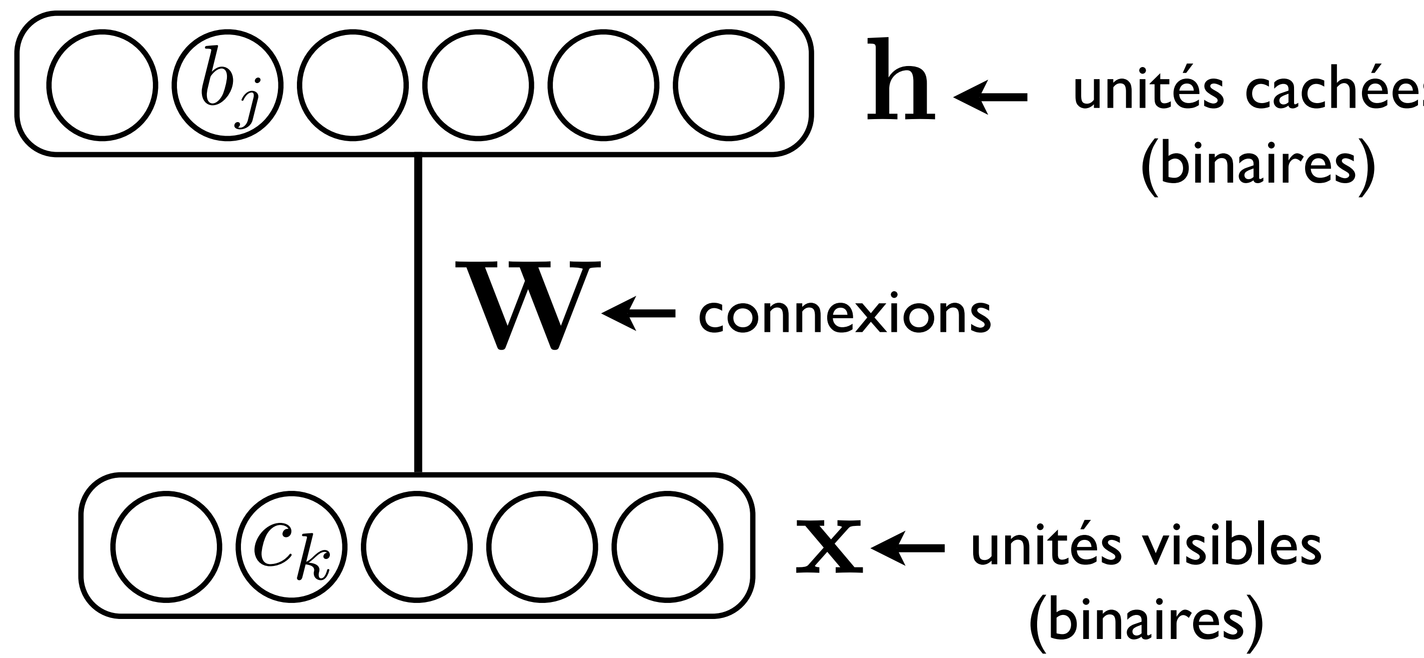
$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^H} p(\mathbf{y},\mathbf{h}|\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^H} \frac{p(\mathbf{y},\mathbf{h},\mathbf{x})}{p(\mathbf{x})}$$

$$= \frac{\sum_{\mathbf{h}\in\{0,1\}^H} p(\mathbf{y},\mathbf{h},\mathbf{x})}{\sum_{\mathbf{y}^*\in\mathcal{Y}} \sum_{\mathbf{h}^*\in\{0,1\}^H} p(\mathbf{x},\mathbf{h}^*,\mathbf{y}^*)}$$

$$= \frac{\exp(-F(\mathbf{x},\mathbf{y}))}{\sum_{\mathbf{y}^*\in\mathcal{Y}} \exp(-F(\mathbf{x},\mathbf{y}^*))}$$

(montrer preuve au tableau)

Conditional Restricted Boltzmann Machine

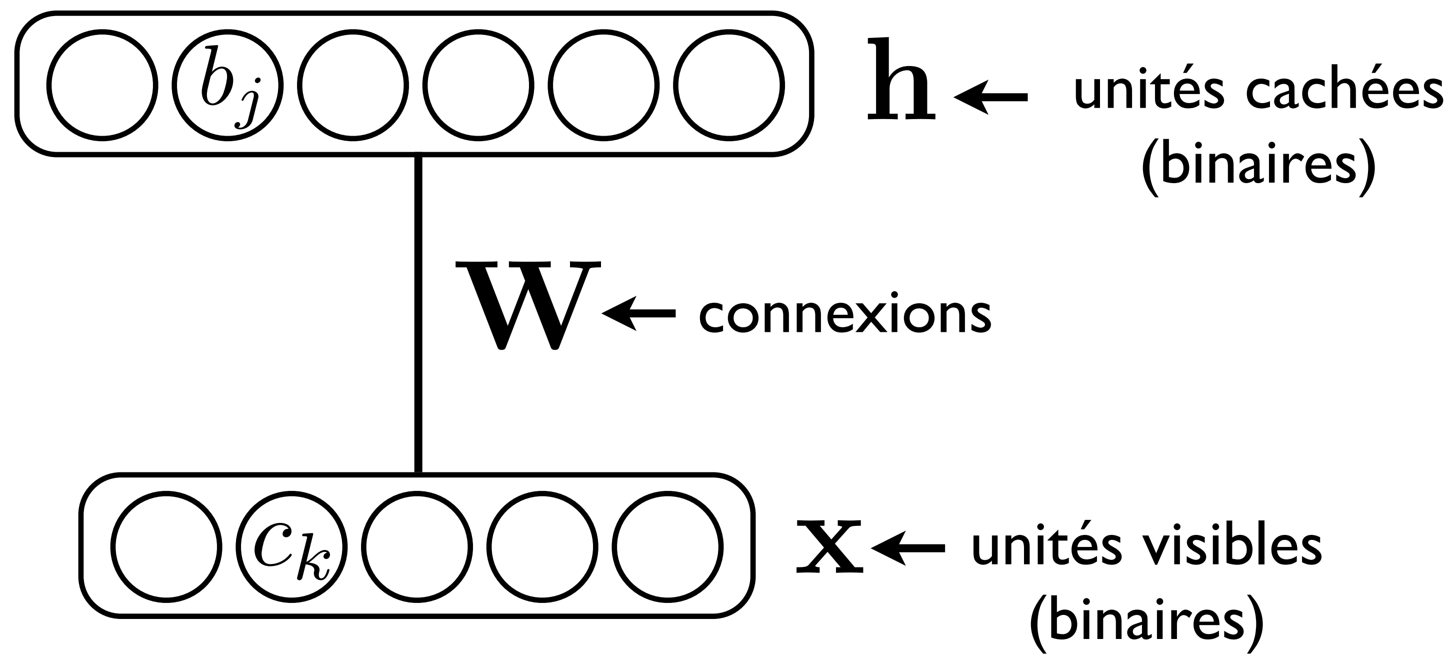


Fonction d'énergie :

$$
\begin{aligned}
E(\mathbf{x},\mathbf{h}) &= -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h} \\
&= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j
\end{aligned}
$$

Distribution: $p(\mathbf{x},\mathbf{h}) = \exp(-E(\mathbf{x},\mathbf{h}))/Z$

Conditional Restricted Boltzmann Machine



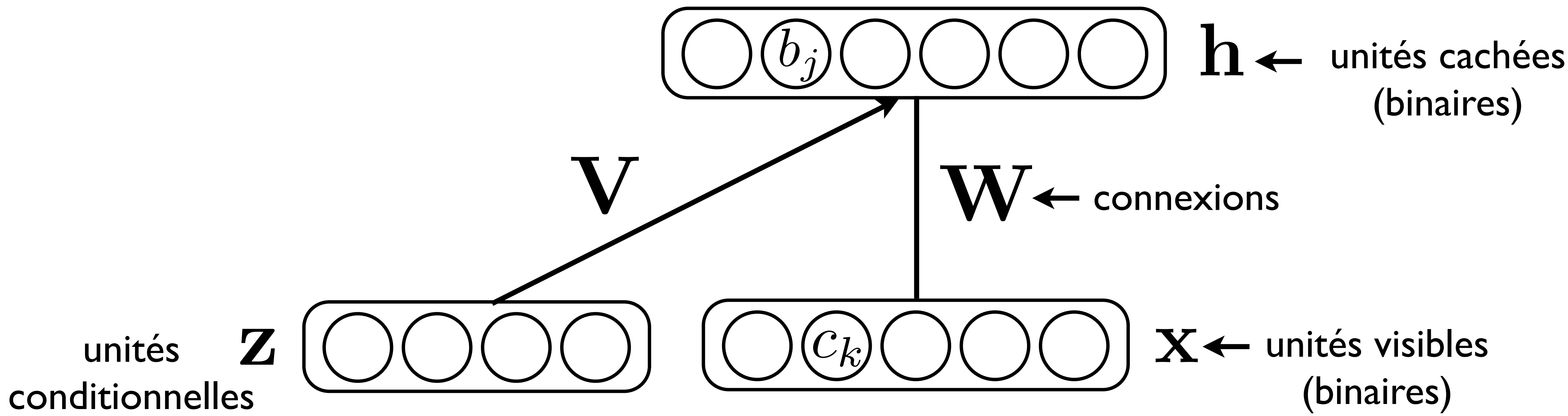
Fonction d'énergie :

$$= -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$

$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution:

Conditional Restricted Boltzmann Machine

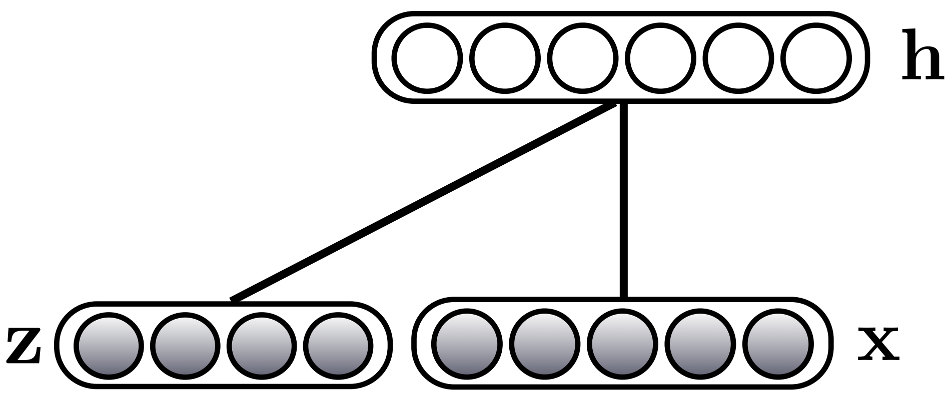


Fonction d'énergie :

$$\begin{aligned} E(\mathbf{x}, \mathbf{h}|\mathbf{z}) &= -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{V}\mathbf{z} \\ &= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j \\ &\quad - \sum_{jl} V_{jl} h_j z_l \end{aligned}$$

Distribution: $p(\mathbf{x}, \mathbf{h}|\mathbf{z}) = \exp(-E(\mathbf{x}, \mathbf{h}|\mathbf{z}))/Z(\mathbf{z})$

Inférence

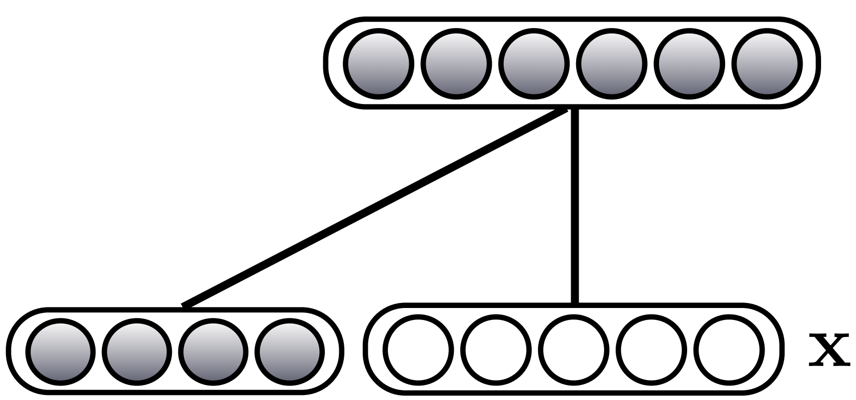


$$p(\mathbf{h}|\mathbf{x},\mathbf{z}) = \prod_j p(h_j|\mathbf{x},\mathbf{z})$$

$$p(h_j = 1|\mathbf{x},\mathbf{z}) = \frac{1}{1+\exp(-(b_j + \mathbf{W}_{j\cdot}\mathbf{x} + \mathbf{V}_{j\cdot}\mathbf{z}))}$$

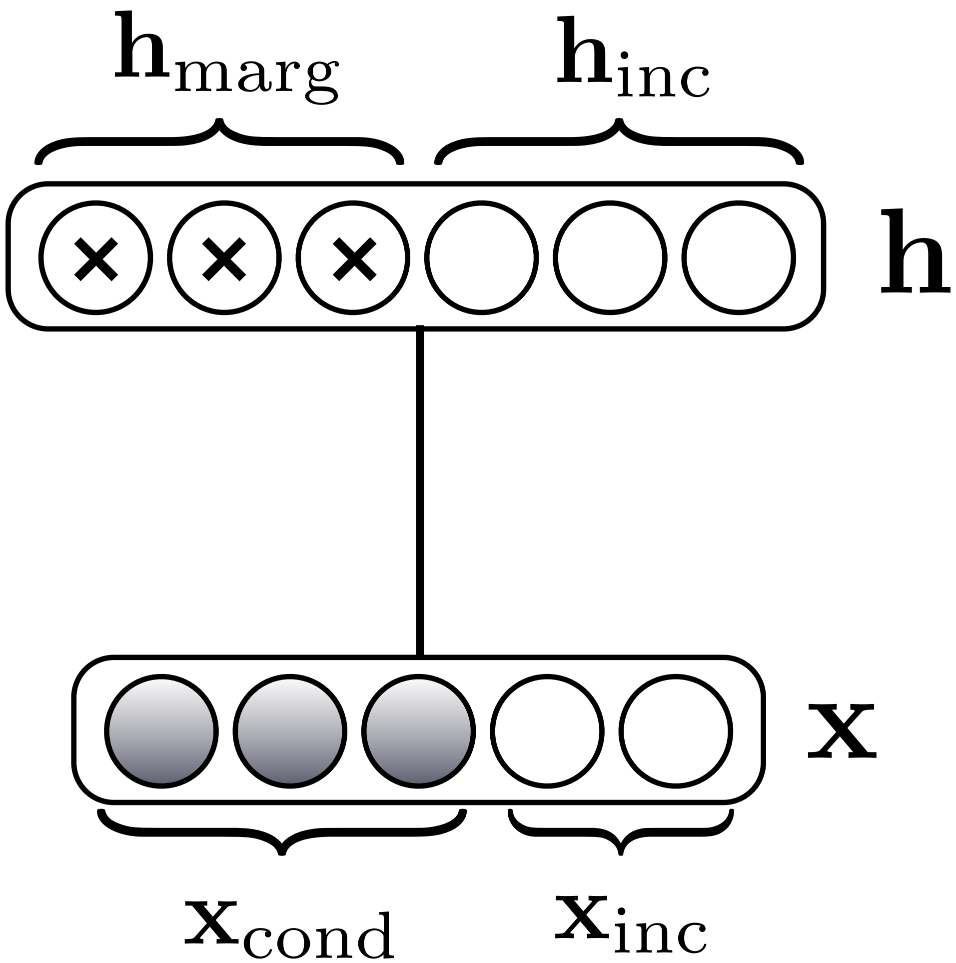
$$= \mathrm{sigm}(b_j + \mathbf{W}_{j\cdot}\mathbf{x} + \mathbf{V}_{j\cdot}\mathbf{z}))$$

(montrer preuve au tableau)



$$p(\mathbf{x}|\mathbf{h},\mathbf{z}) = p(\mathbf{x}|\mathbf{h}) = \prod_k p(x_k|\mathbf{h})$$

Inférence: règle générale



$\mathbf{x}_{\text{inc}}, \mathbf{h}_{\text{inc}}$: variables inconnues, dont la distribution nous intéresse

$\mathbf{x}_{\text{cond}}$: variables connues, qui conditionne la distribution de

$\mathbf{h}_{\text{marg}}$: variables inconnues, non-intéressante et que l'on marginalise

$$p(\mathbf{x}_{\text{inc}}, \mathbf{h}_{\text{inc}}) = \frac{\sum_{\mathbf{h}_{\text{marg}}} \exp(E(\mathbf{x}_{\text{inc}}, \mathbf{h}_{\text{inc}}, \mathbf{h}_{\text{marg}} | \mathbf{x}_{\text{cond}}))}{\sum_{\mathbf{x}_{\text{inc}}} \sum_{\mathbf{h}_{\text{inc}}} \sum_{\mathbf{h}_{\text{marg}}} \exp(E(\mathbf{x}_{\text{inc}}, \mathbf{h}_{\text{inc}}, \mathbf{h}_{\text{marg}} | \mathbf{x}_{\text{cond}}))}$$

Apprentissage

- Afin d'entraîner une RBM, on minimise la log-vraisemblance négative (NLL)

$$\mathcal{L}_{\text{gen}}(\mathcal{D}_{\text{train}}) = \frac{1}{n} \sum_{\mathbf{x}_t \in \mathcal{D}_{\text{train}}} - \log p(\mathbf{x}_t)$$

- On procède par descente de gradient (stochastique)

$$\frac{\partial - \log p(\mathbf{x}_t)}{\partial \theta} = \underbrace{\mathbb{E}_{\mathbf{h}} \left[\frac{\partial E(\mathbf{x}_t, \mathbf{h})}{\partial \theta} \middle| \mathbf{x}_t \right]}_{\text{phase positive}} - \underbrace{\mathbb{E}_{\mathbf{x},\mathbf{h}} \left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta} \right]}_{\text{phase négative}}$$

Apprentissage

- En général, la phase positive est plutôt simple à calculer et ne présente pas de difficultés

- Tous les algorithmes d'apprentissage pour RBMs suivent le même principe général, et diffèrent seulement dans la façon d'estimer la phase négative

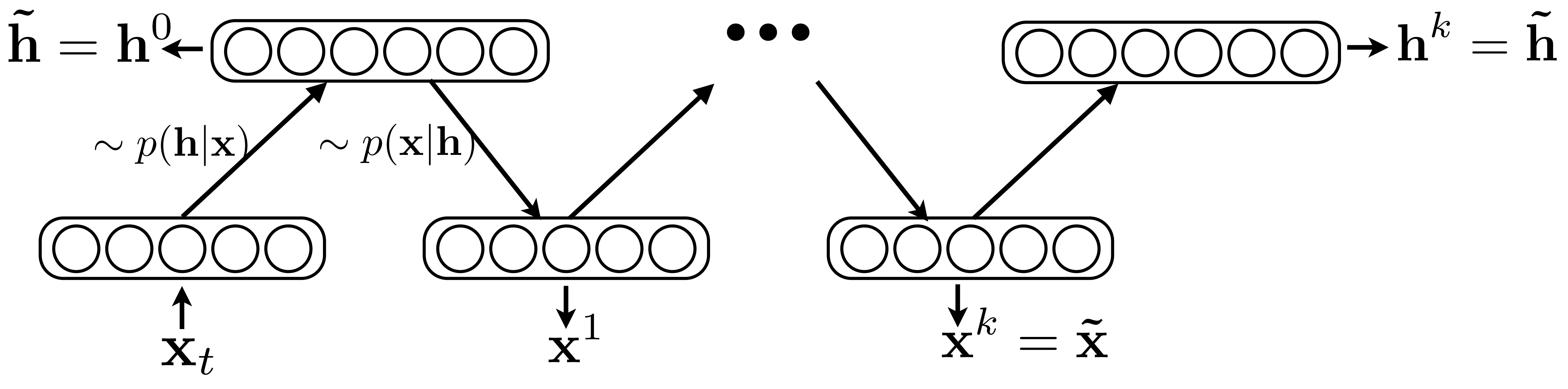
$$\mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \theta}\right]$$

Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

- Idée:

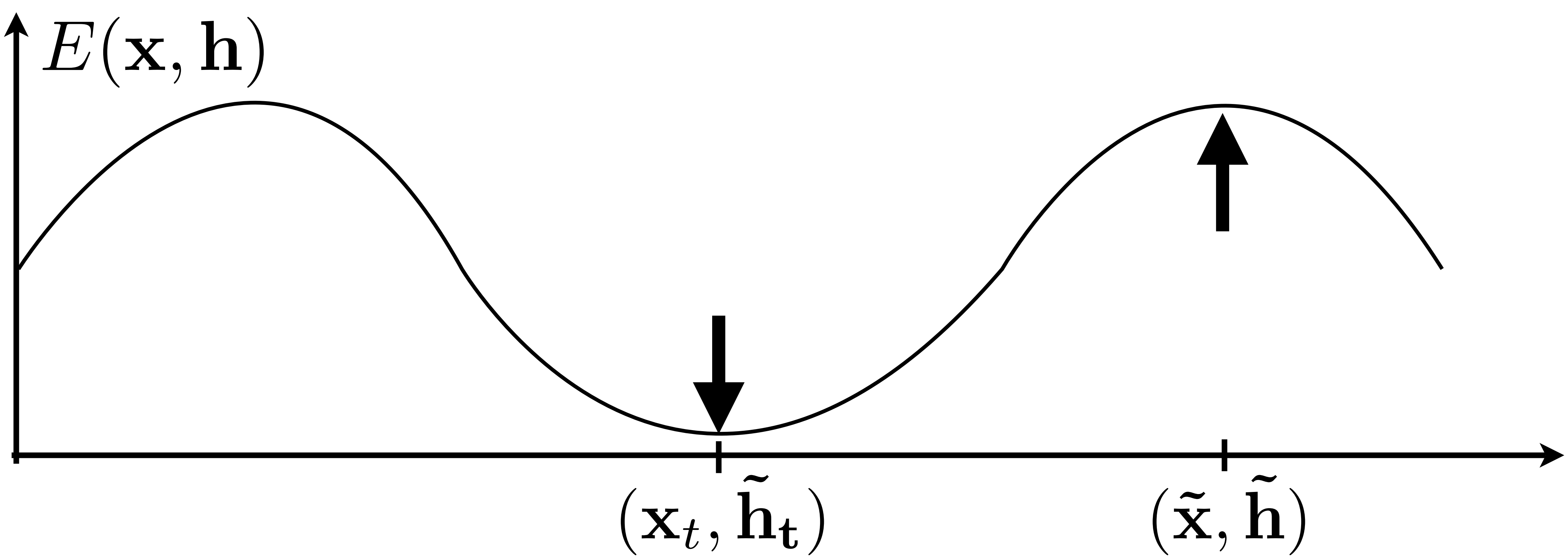
  1. remplacer l'espérance par l'estimation en un seul point $\tilde{\mathbf{x}}, \tilde{\mathbf{h}}$
  2. obtenir $\tilde{\mathbf{x}}, \tilde{\mathbf{h}}$ en échantillonnant
  3. commencer la chaîne à $\mathbf{x}_t$



Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

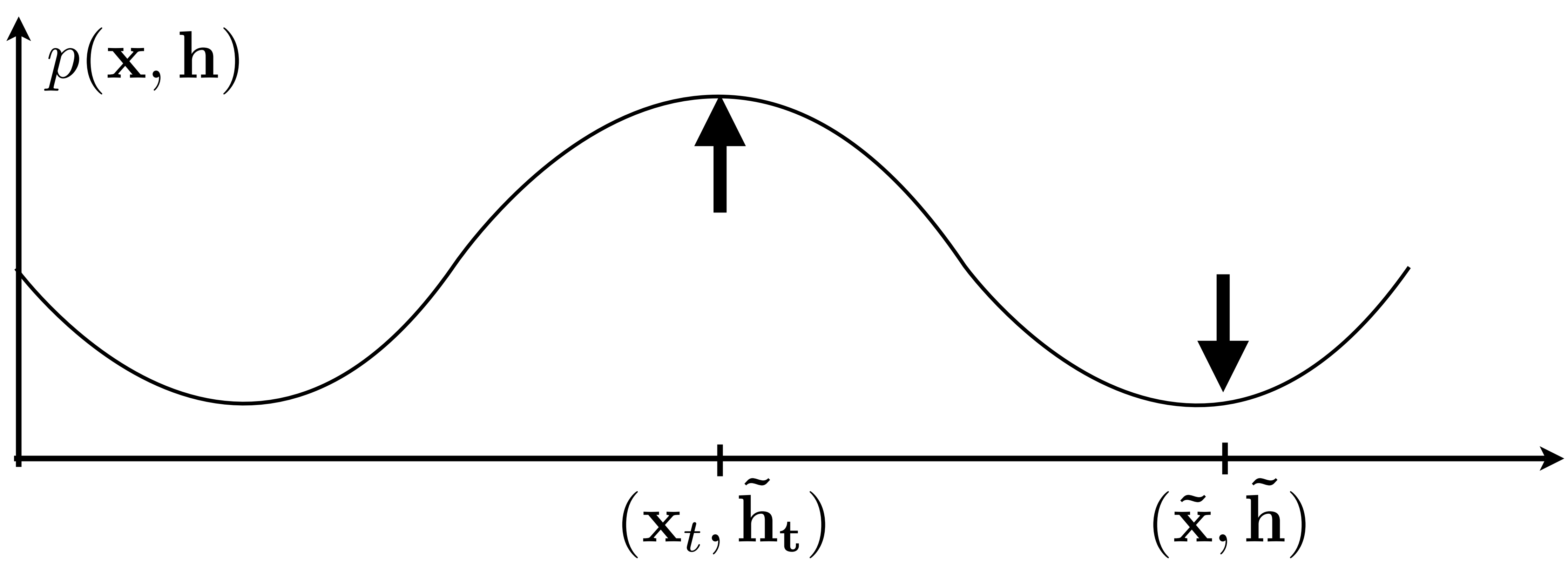
$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}_t,\mathbf{h})}{\partial\theta}\Big|\mathbf{x}_t\right]\approx\frac{\partial E(\mathbf{x}_t,\tilde{\mathbf{h}}_{\mathbf{t}})}{\partial\theta}\qquad\mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right]\approx\frac{\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}$$



Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}_t,\mathbf{h})}{\partial\theta}\bigg|\mathbf{x}_t\right] \approx \frac{\partial E(\mathbf{x}_t,\tilde{\mathbf{h}}_{\mathbf{t}})}{\partial\theta} \qquad \mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right] \approx \frac{\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}$$



Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

- CD-k: contrastive divergence avec k itérations d'échantillonnage de Gibbs
- En général, plus k est grand, moins **biaisé** est l'estimation du gradient
- En pratique, k=1 marche assez bien
- Dans le cas d'une RBM, les mises à jour de paramètres ne dépendront pas de $\tilde{\mathrm{h}}$

Dérivation de la règle d'apprentissage

- Calcul de $\dfrac{\partial E(\mathbf{x},\mathbf{h})}{\partial \theta}$ pour $\theta = W_{jk}$

$$\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial W_{jk}} = \frac{\partial}{\partial W_{jk}} \left(-\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j \right)$$

$$= -\frac{\partial}{\partial W_{jk}} \sum_{jk} W_{jk} h_j x_k$$

$$= -h_j x_k$$

$$\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \mathbf{W}} = -\mathbf{h}\ \mathbf{x}^\top$$

Dérivation de la règle d'apprentissage

- Calcul de $\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \theta}\Big|\mathbf{x}\right]$ pour $\theta = W_{jk}$

$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial W_{jk}}\Big|\mathbf{x}\right] = \mathbb{E}_{\mathbf{h}}\left[-h_j x_k \Big|\mathbf{x}\right] = \sum_{h_j\in\{0,1\}} -h_j x_k p(h_j|\mathbf{x})$$

$$= -x_k p(h_j = 1|\mathbf{x})$$

$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \mathbf{W}}\Big|\mathbf{x}\right] = -\mathbf{h}(\mathbf{x})\ \mathbf{x}^\top$$

$$\boxed{\begin{aligned}\mathbf{h}(\mathbf{x}) &\overset{\text{def}}{=} \begin{pmatrix} p(h_1=1|\mathbf{x}) \\ \dots \\ p(h_H=1|\mathbf{x})\end{pmatrix} \\ &= \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}}$$

Dérivation de la règle d'apprentissage

- Ainsi, étant donné $\mathbf{x}_t$ et $\tilde{\mathbf{x}}$, la règle d'apprentissage pour $\theta = \mathbf{W}$ devient

$$\mathbf{W} \leftarrow \mathbf{W} - \epsilon \left(- \frac{\partial \log p(\mathbf{x}_t)}{\partial \mathbf{W}} \right)$$

$$\leftarrow \mathbf{W} - \epsilon \left(\mathbb{E}_{\mathbf{h}} \left[\frac{\partial E(\mathbf{x}_t, \mathbf{h})}{\partial \mathbf{W}} \middle| \mathbf{x}_t \right] - \mathbb{E}_{\mathbf{x},\mathbf{h}} \left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \mathbf{W}} \right] \right)$$

$$\leftarrow \mathbf{W} - \epsilon \left(\mathbb{E}_{\mathbf{h}} \left[\frac{\partial E(\mathbf{x}_t, \mathbf{h})}{\partial \mathbf{W}} \middle| \mathbf{x}_t \right] - \mathbb{E}_{\mathbf{h}} \left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \mathbf{W}} \middle| \tilde{\mathbf{x}} \right] \right)$$

$$\leftarrow \mathbf{W} + \epsilon \left(\mathbf{h}(\mathbf{x}_t)\ \mathbf{x}_t^\top - \mathbf{h}(\tilde{\mathbf{x}})\ \tilde{\mathbf{x}}^\top \right)$$

Pseudocode CD-k

1. Pour chaque exemple d'entraînement $\mathbf{x}_t$

   i. générer un échantillon négatif $\tilde{\mathbf{x}}$
   à l'aide de k itérations d'échantillonnage
   de Gibbs

   ii. mettre à jour les paramètres

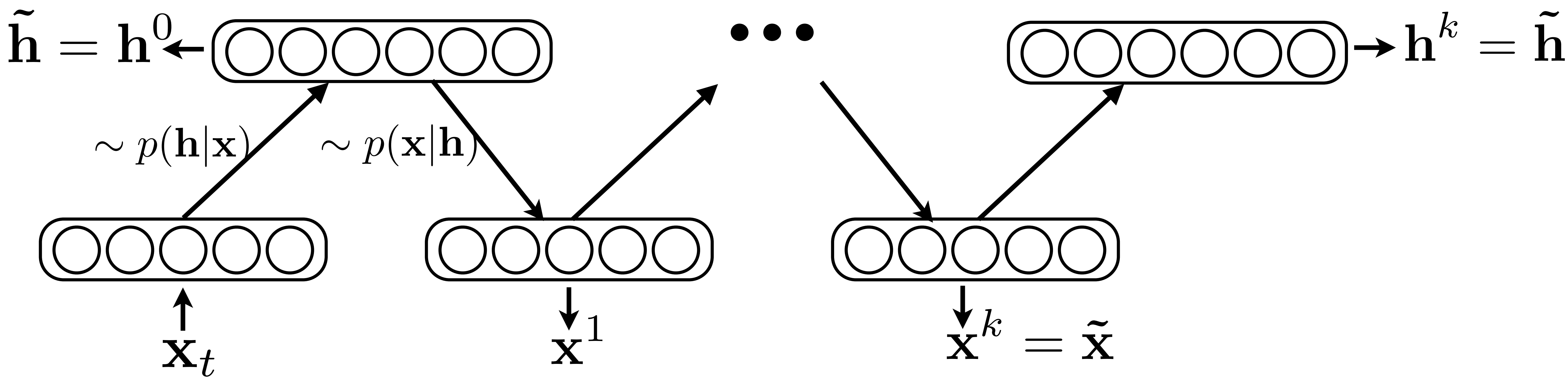
$$\mathbf{W} \Leftarrow \mathbf{W} + \epsilon \left(\mathbf{h}(\mathbf{x}_t)\ \mathbf{x}_t^\top - \mathbf{h}(\tilde{\mathbf{x}})\ \tilde{\mathbf{x}}^\top \right)$$
$$\mathbf{b} \Leftarrow \mathbf{b} + \epsilon \left(\mathbf{h}(\mathbf{x}_t) - \mathbf{h}(\tilde{\mathbf{x}}) \right)$$
$$\mathbf{c} \Leftarrow \mathbf{c} + \epsilon \left(\mathbf{x}_t - \tilde{\mathbf{x}} \right)$$

2. Si n'a pas convergé, revenir à 1

Autres algorithmes d'apprenttisage: Mean-Field CD (MF-CD)

(Welling and Hinton, ICANN2002)

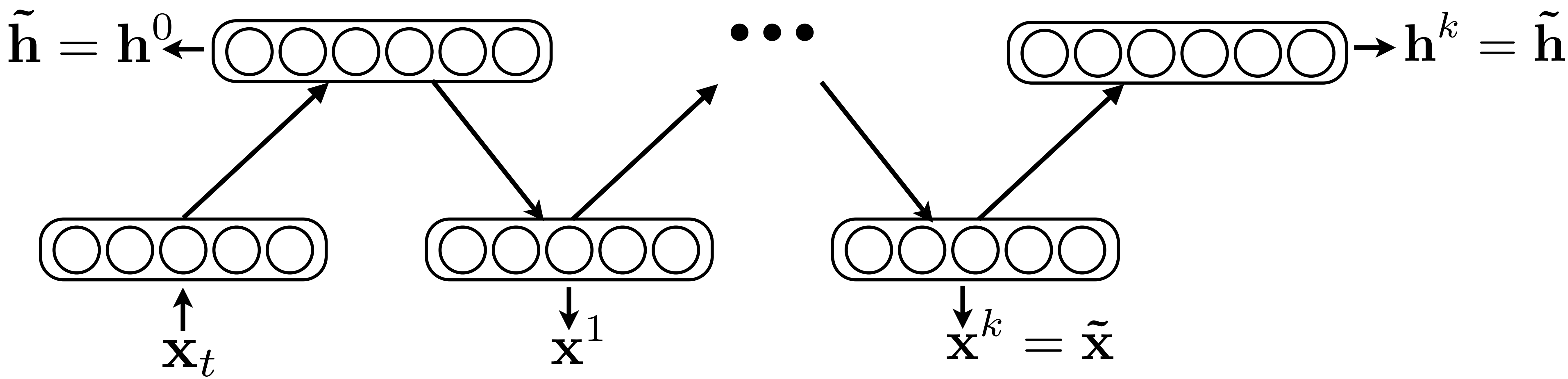
- Idée: plutôt qu'échantillonner, utiliser la moyenne des éléments d'une couche étant donnée l'entre couche



Autres algorithmes d'apprenttisage: Mean-Field CD (MF-CD)

(Welling and Hinton, ICANN2002)

- Idée: plutôt qu'échantillonner, utiliser la moyenne des éléments d'une couche étant donnée l'entre couche

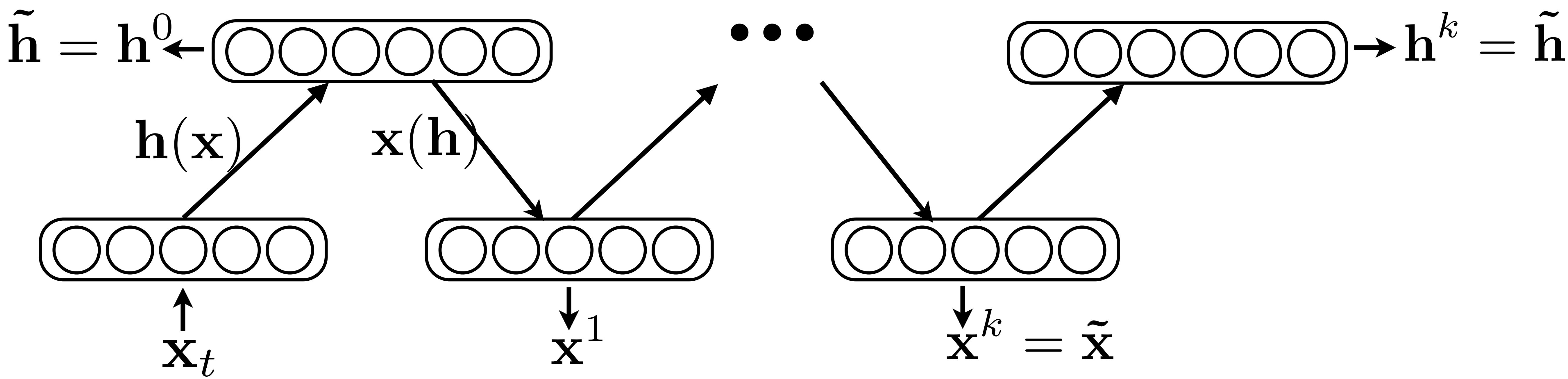


Autres algorithmes d'apprenttisage: Mean-Field CD (MF-CD)

(Welling and Hinton, ICANN2002)

- Idée: plutôt qu'échantillonner, utiliser la moyenne des éléments d'une couche étant donnée l'entre couche

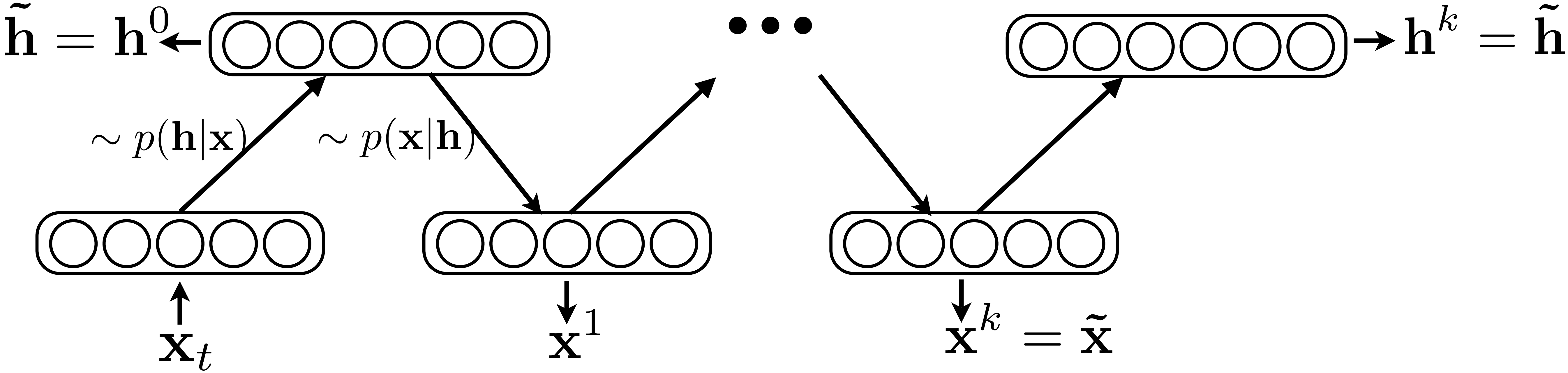
$$\boxed{\mathbf{h}(\mathbf{x}) \stackrel{\text{def}}{=} \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})} \quad \boxed{\mathbf{x}(\mathbf{h}) \stackrel{\text{def}}{=} \text{sigm}(\mathbf{c} + \mathbf{W}^\top \mathbf{h})}$$



Autres algorithmes d'apprenttisage: Persistent CD (PCD)

(Tieleman, ICML2008)

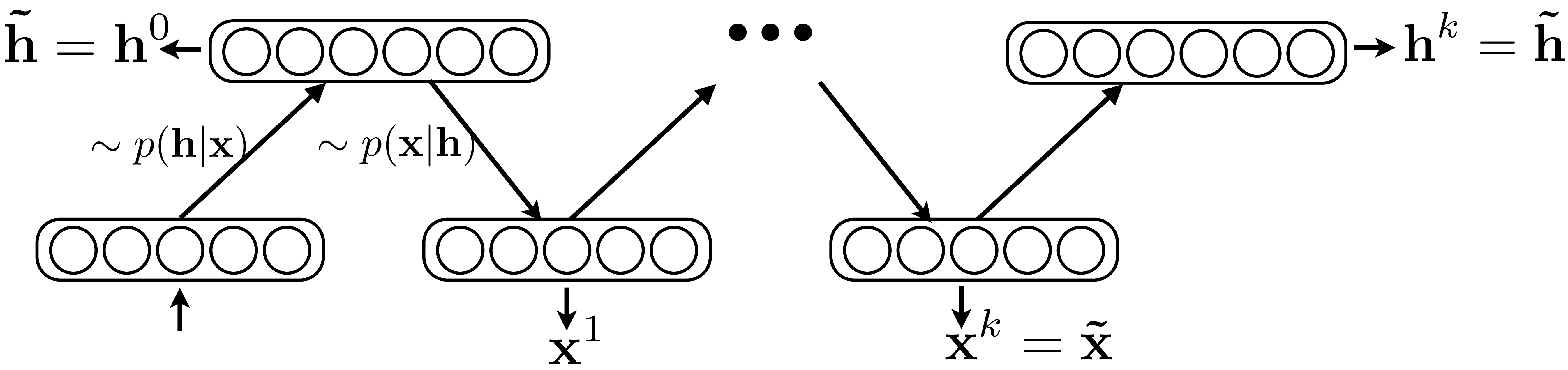
- Idée: plutôt que d'initialiser la chaîne de Gibbs à $\mathbf{x}_t$, initialiser à $\tilde{\mathbf{x}}$ de l'itération précédente



Autres algorithmes d'apprenttisage: Persistent CD (PCD)

(Tieleman, ICML2008)

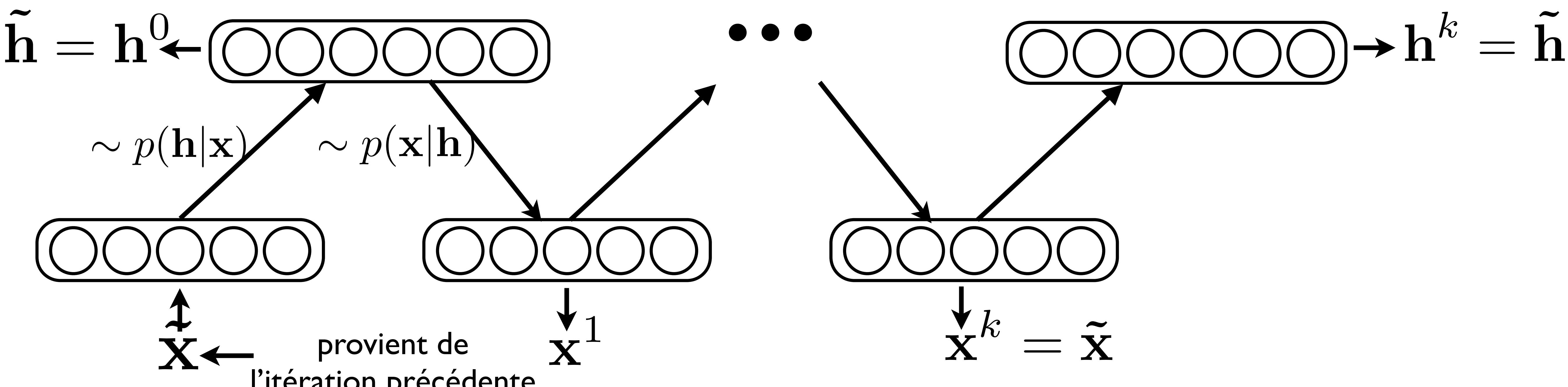
- Idée: plutôt que d'initialiser la chaîne de Gibbs à $\mathbf{x}_t$, initialiser à $\tilde{\mathbf{x}}$ de l'itération précédente



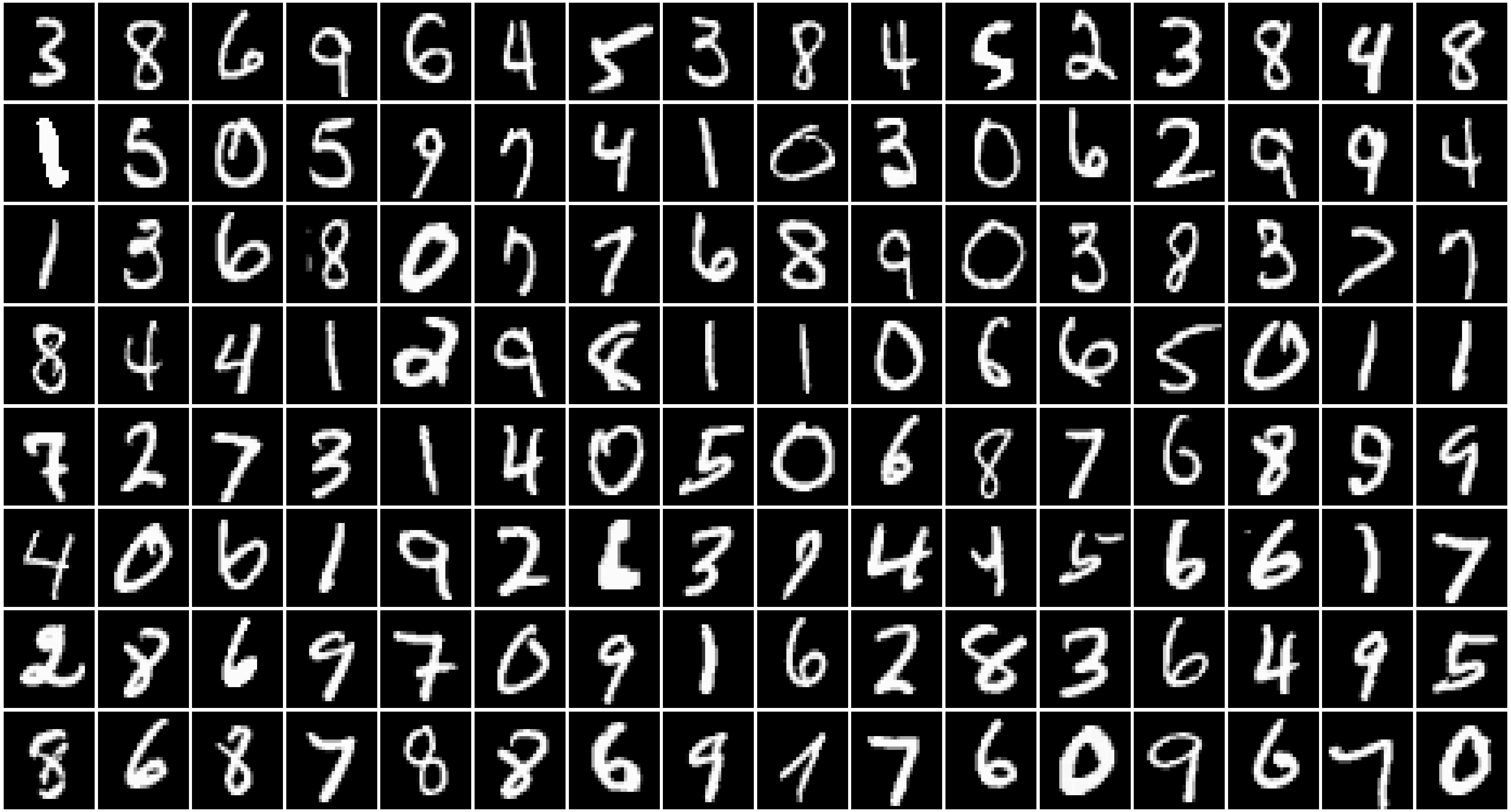
Autres algorithmes d'apprenttisage: Persistent CD (PCD)

(Tieleman, ICML2008)

- Idée: plutôt que d'initialiser la chaîne de Gibbs à $\mathbf{x}_t$, initialiser à $\tilde{\mathbf{x}}$ de l'itération précédente



Exemple de données: MNIST



Comparaison d'algorithmes

(Tieleman, ICML2008)

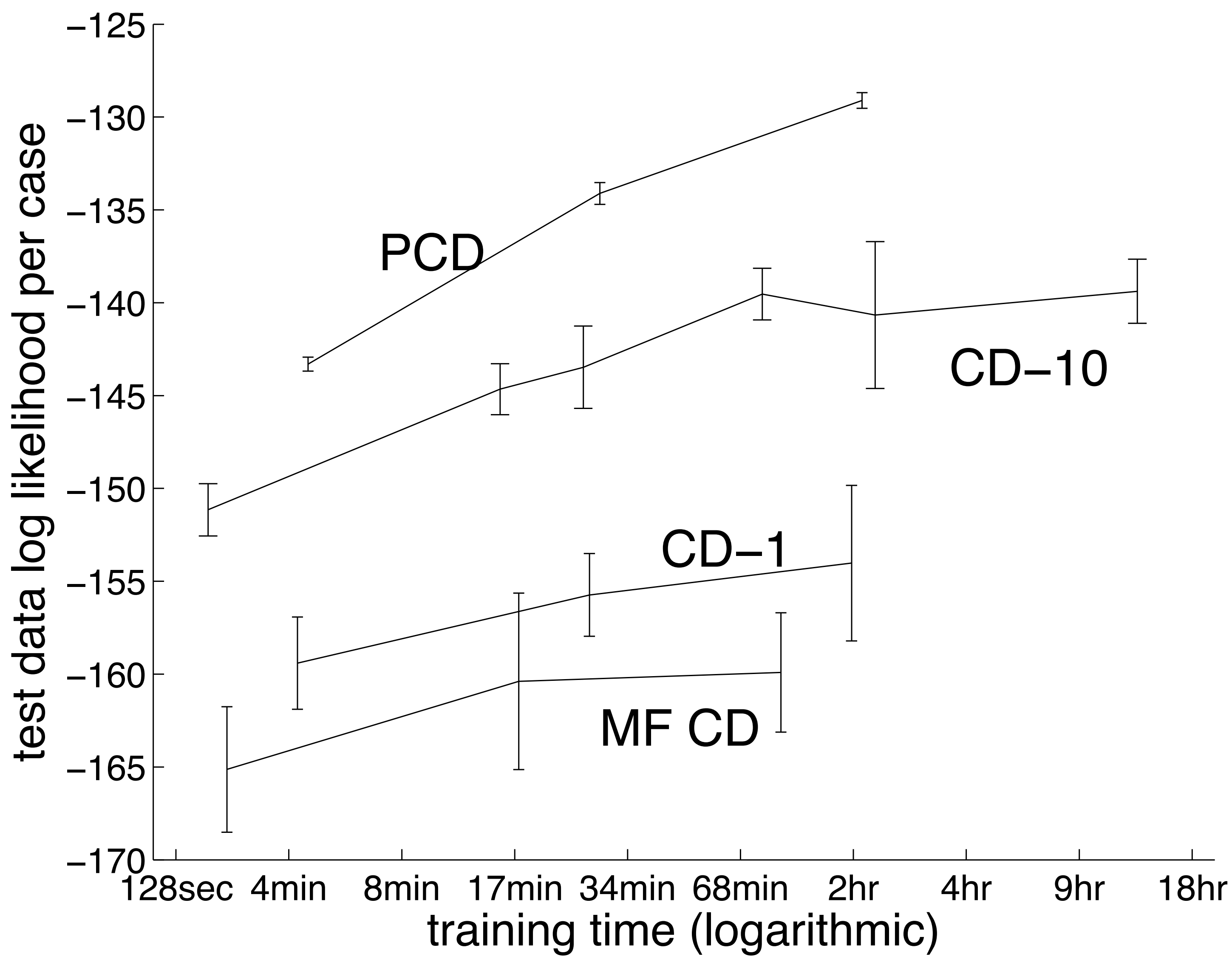


Figure 1. Modeling MNIST data with 25 hidden units (exact log likelihood)

Échantillons générés (RBMs)

(Tieleman, ICML2008)

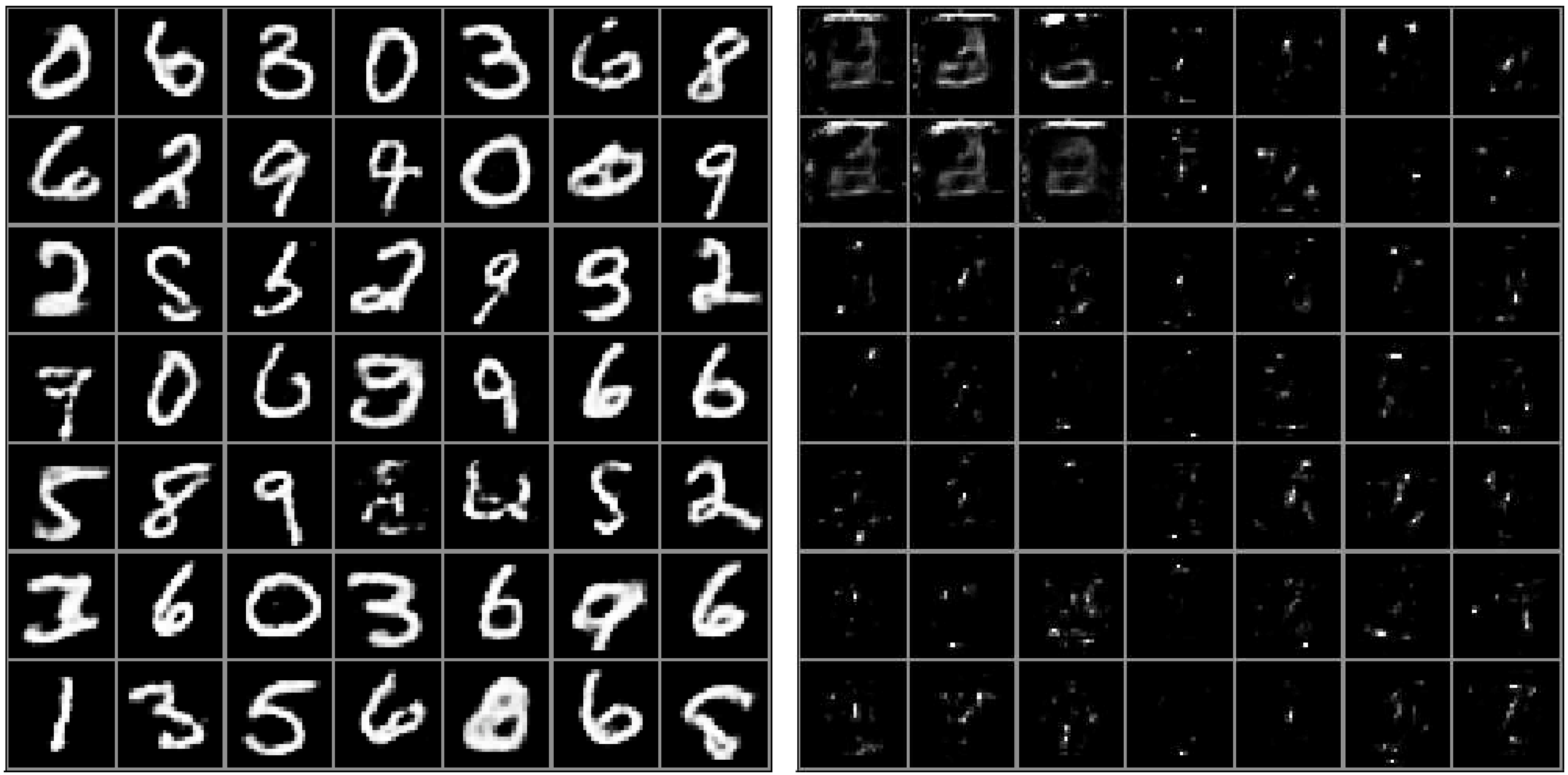
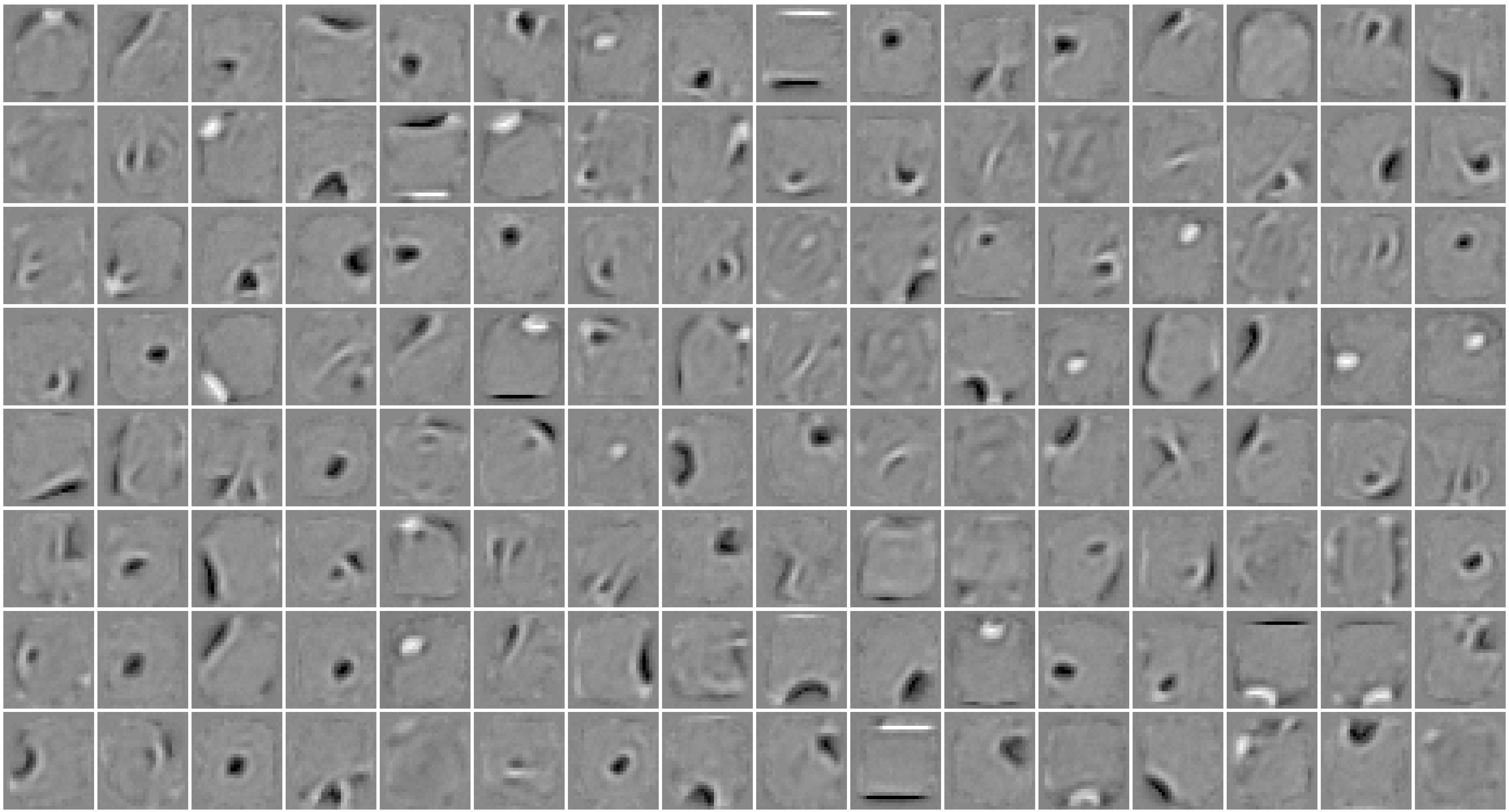


Figure 4. Samples from an RBM that was trained using PCD (left) and an RBM that was trained using CD-1 (right). Clearly, CD-1 did not produce an accurate model of the MNIST digits. Notice, however, that some of the CD-1 samples vaguely resemble a three.

Filtres

(Larochelle et al., JMLR2009)



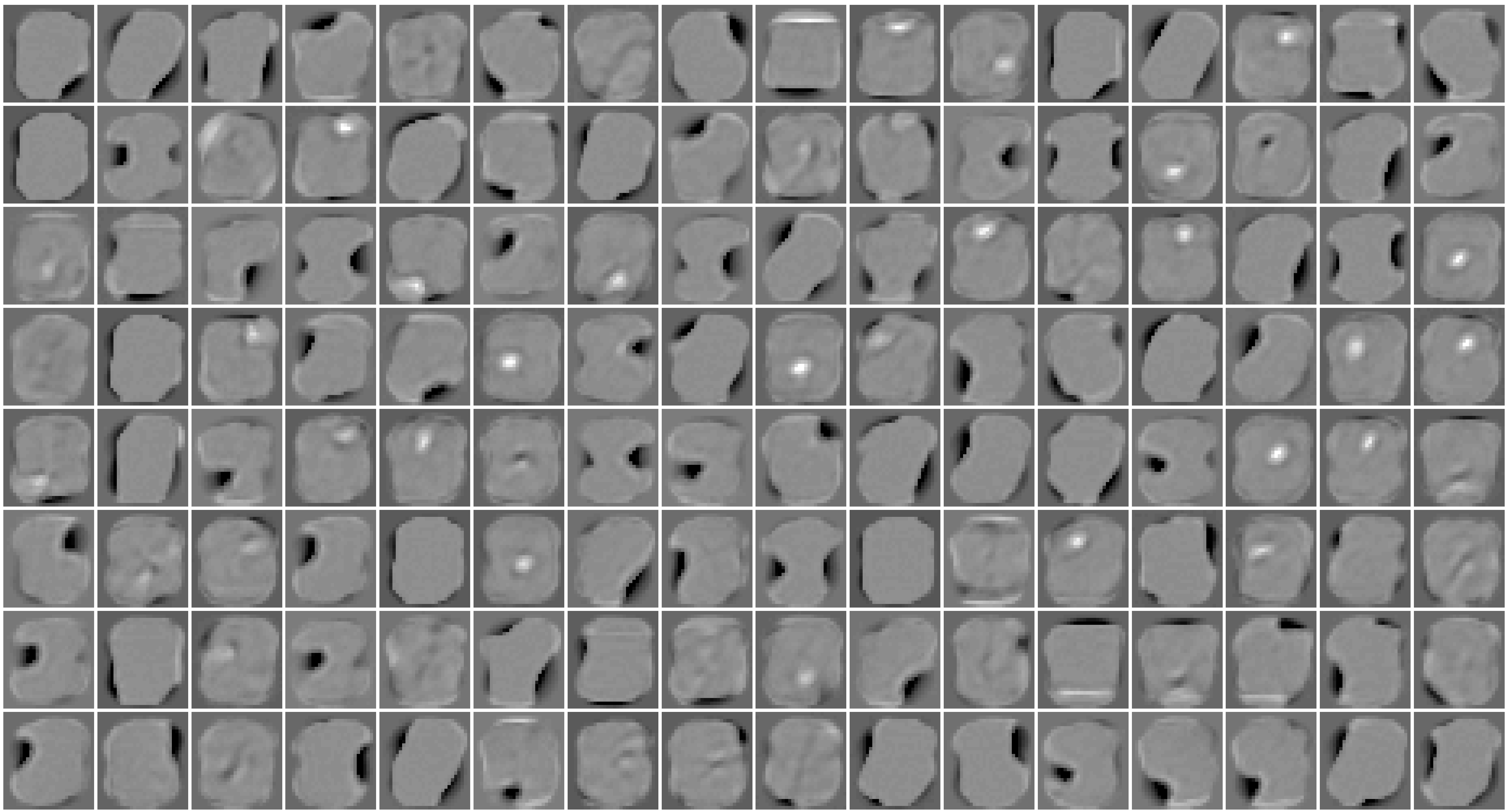
Extension à des unités non-binaires

(Bengio et al., NIPS2007)

- Unités $\mathbf{x}$ réelles entre 0 et 1
  - ★ Peut utiliser la même fonction d'énergie
  - ★ Obtient des intégrales plutôt que des sommes sur les valeurs de $\mathbf{x}$
- Unités $\mathbf{x}$ réelles Gaussiennes
  - ★ Ajout d'un terme quadratique à la fonction d'énergie
- Plusieurs autres possibilités

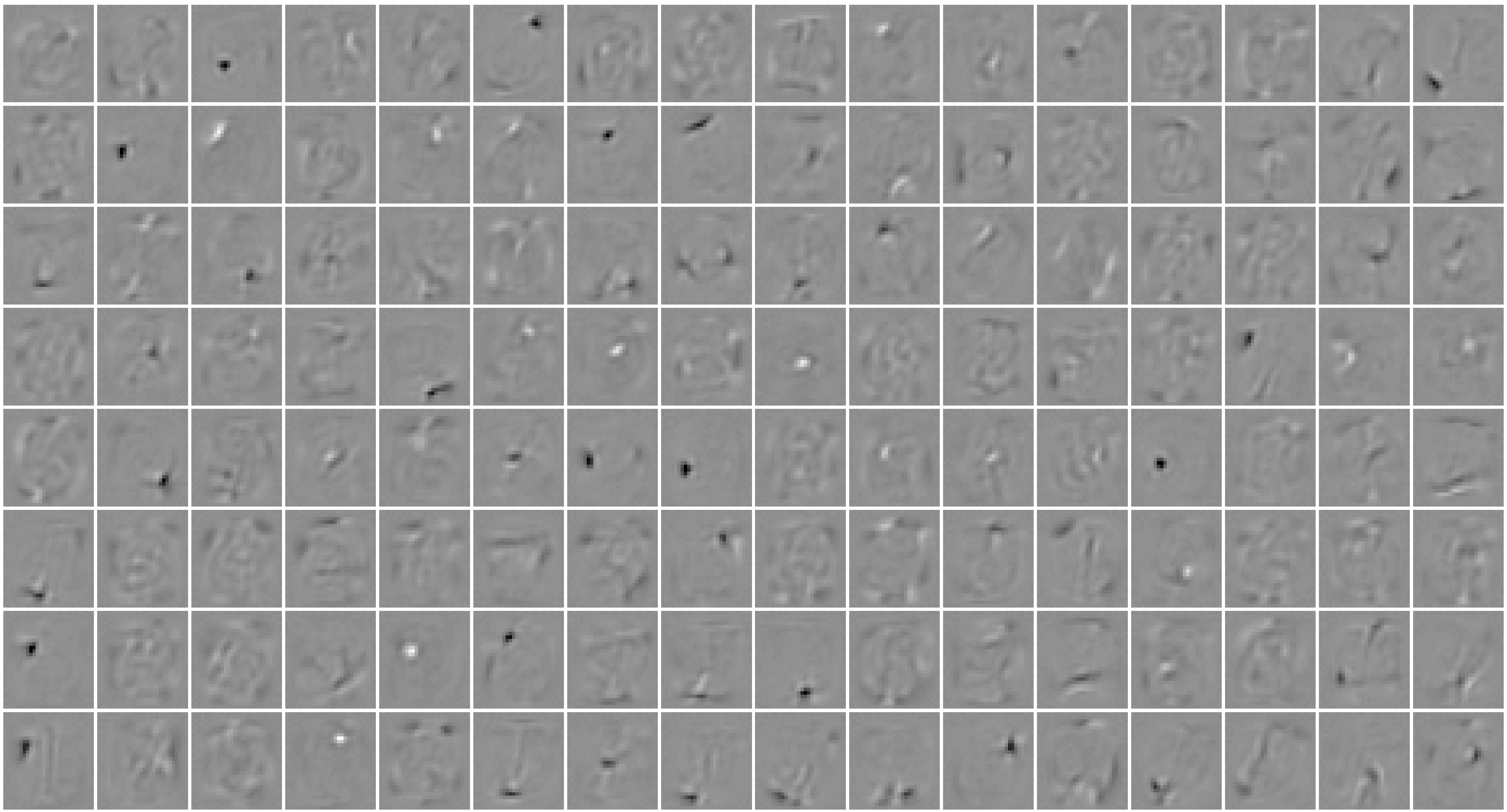
Filtres (trunc. exp.)

(Larochelle et al., JMLR2009)



Filtres (Gauss.)

(Larochelle et al., JMLR2009)

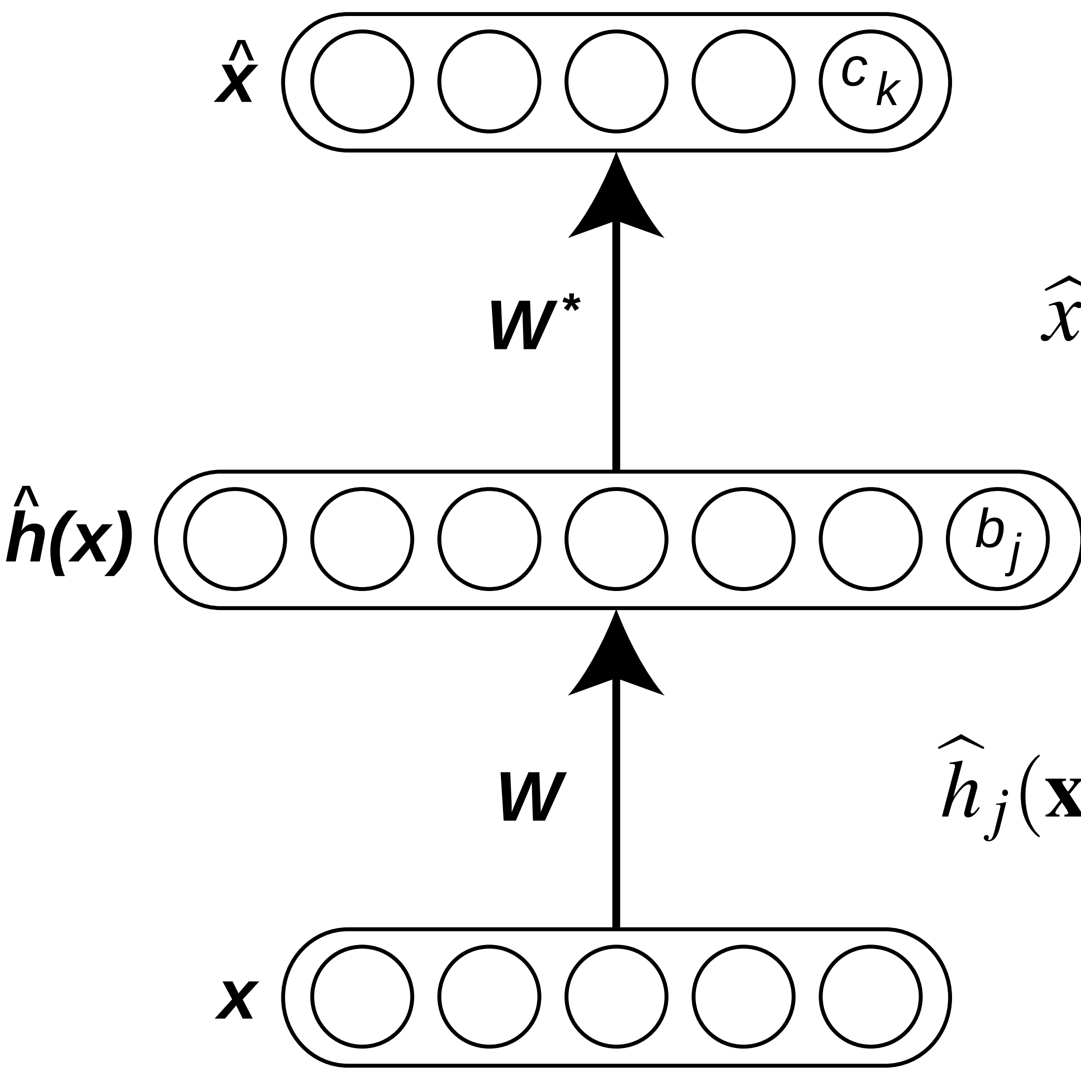


Comparaison

SRBM input type	Train.	Valid.	Test
Bernoulli	10.50%	18.10%	20.29%
Gaussian	0%	20.50%	21.36%
Truncated exponential	0%	14.30%	14.34%

Et si on ne veut pas utiliser des RBMs?

Autoencodeurs



$$\widehat{x}_k = g\left(\widehat{a}_k\right) \text{ where } \widehat{a}_k = c_k + \sum_j W^*_{jk} \widehat{h}_j(\mathbf{x})$$

$$\widehat{h}_j(\mathbf{x}) = f\left(a_j\right) \text{ where } a_j(\mathbf{x}) = b_j + \sum_k W_{jk} x_k$$

Autoencodeurs

- Coût à optimiser
  - ★ Somme des erreurs au carré

$$C(\widehat{\mathbf{x}}, \mathbf{x}) = \sum_k (\widehat{x}_k - x_k)^2$$

Même chose que faire de la PCA!!

  - ★ "Cross-entropy"

$$C(\widehat{\mathbf{x}}, \mathbf{x}) = -\sum_k \left(x_k \log(\widehat{x}_k) + (1 - x_k) \log(1 - \widehat{x}_k)\right)$$

Pseudocode: autoencodeur

% Forward propagation

$\mathbf{a}(\mathbf{x}) \leftarrow \mathbf{b} + \mathbf{W}\mathbf{x}$

$\mathbf{h}(\mathbf{x}) \leftarrow \mathrm{sigm}(\mathbf{a}(\mathbf{x}))$

$\widehat{\mathbf{a}}(\mathbf{x}) \leftarrow \mathbf{c} + \mathbf{W}^{\top}\mathbf{h}(\mathbf{x})$

$\widehat{\mathbf{x}} \leftarrow \mathrm{sigm}(\widehat{\mathbf{a}}(\mathbf{x}))$

% Backward gradient propagation

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})} \leftarrow \widehat{\mathbf{x}} - \mathbf{x}$

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}(\mathbf{x})} \leftarrow \mathbf{W}\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial a_j(\mathbf{x})} \leftarrow \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}_j(\mathbf{x})}\widehat{h}_j(\mathbf{x})\left(1 - \widehat{h}_j(\mathbf{x})\right)$

for $j \in \{1, \ldots, |\widehat{\mathbf{h}}(\mathbf{x})|\}$

⋮

⋮

% Update

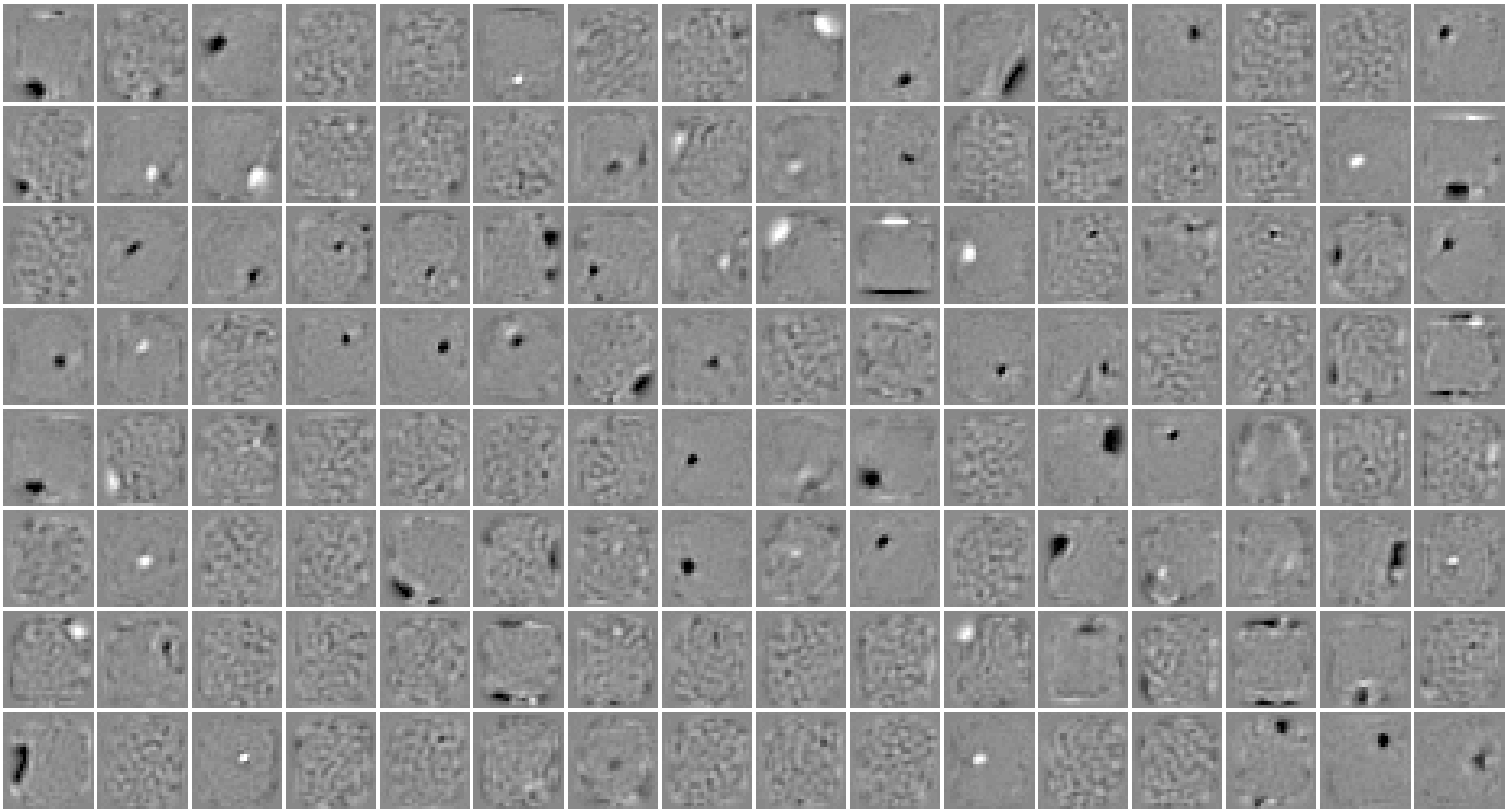
$\mathbf{W}^i \leftarrow \mathbf{W}^i - \varepsilon\left(\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}\mathbf{x}^{\top} + \widehat{\mathbf{h}}(\mathbf{x})\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}^{\top}\right)$

$\mathbf{b}^i \leftarrow \mathbf{b}^i - \varepsilon\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}$

$\mathbf{b}^{i-1} \leftarrow \mathbf{b}^{i-1} - \varepsilon\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$

Filtres (autoencodeur)

(Larochelle et al., JMLR2009)



Encoder vs. Générer

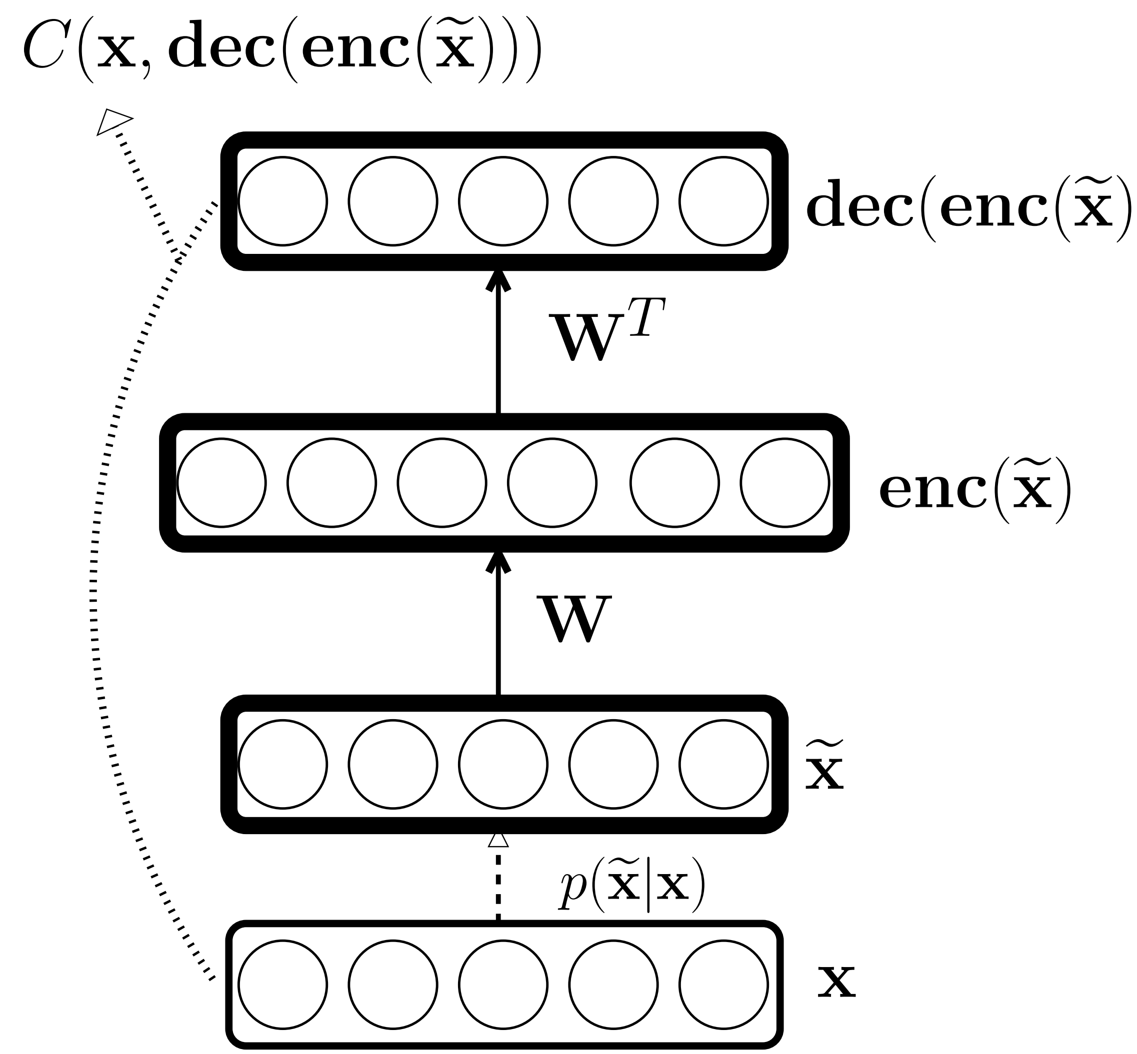
- En empilant des RBMs, il est possible que l'information se perde
- Pourrait combiner les critères de la RBM et de l'autoencodeur
- Sur MNIST, erreur tombe à 1.1%

“Denoising Autoencoder”

(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

- Idée: représentation devrait être robuste à l’absence de certaines entrées

- Introduit un processus de corruption $p(\widetilde{\mathbf{x}}|\mathbf{x})$ fixant à 0 certaines entrées avec probabilité ν



$$\mathbf{enc}(\widetilde{\mathbf{x}}) = \mathrm{sigm}(\mathbf{b} + \mathbf{W}\widetilde{\mathbf{x}})$$

$$\mathbf{dec}\left(\mathbf{enc}(\widetilde{\mathbf{x}})\right) = \mathrm{sigm}\left(\mathbf{c} + \mathbf{W}^T\mathbf{enc}(\widetilde{\mathbf{x}})\right)$$

Pseudocode: "denoising autoencoder"

% Forward propagation

$\mathbf{a}(\mathbf{x}) \leftarrow \mathbf{b} + \mathbf{W}\mathbf{x}$

$\mathbf{h}(\mathbf{x}) \leftarrow \mathrm{sigm}(\mathbf{a}(\mathbf{x}))$

$\widehat{\mathbf{a}}(\mathbf{x}) \leftarrow \mathbf{c} + \mathbf{W}^{\mathsf{T}}\mathbf{h}(\mathbf{x})$

$\widehat{\mathbf{x}} \leftarrow \mathrm{sigm}(\widehat{\mathbf{a}}(\mathbf{x}))$

% Backward gradient propagation

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})} \leftarrow \widehat{\mathbf{x}} - \mathbf{x}$

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}(\mathbf{x})} \leftarrow \mathbf{W}\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial a_j(\mathbf{x})} \leftarrow \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}_j(\mathbf{x})}\widehat{h}_j(\mathbf{x})\left(1 - \widehat{h}_j(\mathbf{x})\right)$

$\quad \textbf{for } j \in \{1, \ldots, |\widehat{\mathbf{h}}(\mathbf{x})|\}$

$\vdots$

$\vdots$

% Update

$\mathbf{W}^i \leftarrow \mathbf{W}^i - \varepsilon \left(\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}\mathbf{x}^{\mathsf{T}} + \widehat{\mathbf{h}}(\mathbf{x})\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}^{\mathsf{T}} \right)$

$\mathbf{b}^i \leftarrow \mathbf{b}^i - \varepsilon \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}$

$\mathbf{b}^{i-1} \leftarrow \mathbf{b}^{i-1} - \varepsilon \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$

Pseudocode: "denoising autoencoder"

% Forward propagation

$\mathbf{a}(\mathbf{x}) \leftarrow \mathbf{b} + \mathbf{W}\widetilde{\mathbf{x}}$

$\mathbf{h}(\mathbf{x}) \leftarrow \mathrm{sigm}(\mathbf{a}(\mathbf{x}))$

$\widehat{\mathbf{a}}(\mathbf{x}) \leftarrow \mathbf{c} + \mathbf{W}^\mathsf{T}\mathbf{h}(\mathbf{x})$

$\widehat{\mathbf{x}} \leftarrow \mathrm{sigm}(\widehat{\mathbf{a}}(\mathbf{x}))$

% Backward gradient propagation

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})} \leftarrow \widehat{\mathbf{x}} - \mathbf{x}$

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}(\mathbf{x})} \leftarrow \mathbf{W} \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$

$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial a_j(\mathbf{x})} \leftarrow \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}_j(\mathbf{x})} \widehat{h}_j(\mathbf{x}) \left(1 - \widehat{h}_j(\mathbf{x})\right)$

$\quad \textbf{for } j \in \{1, \ldots, |\widehat{\mathbf{h}}(\mathbf{x})|\}$

$\vdots$

$\vdots$

% Update

$\mathbf{W}^i \leftarrow \mathbf{W}^i - \varepsilon \left(\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})} \mathbf{x}^\mathsf{T} + \widehat{\mathbf{h}}(\mathbf{x}) \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}^\mathsf{T} \right)$

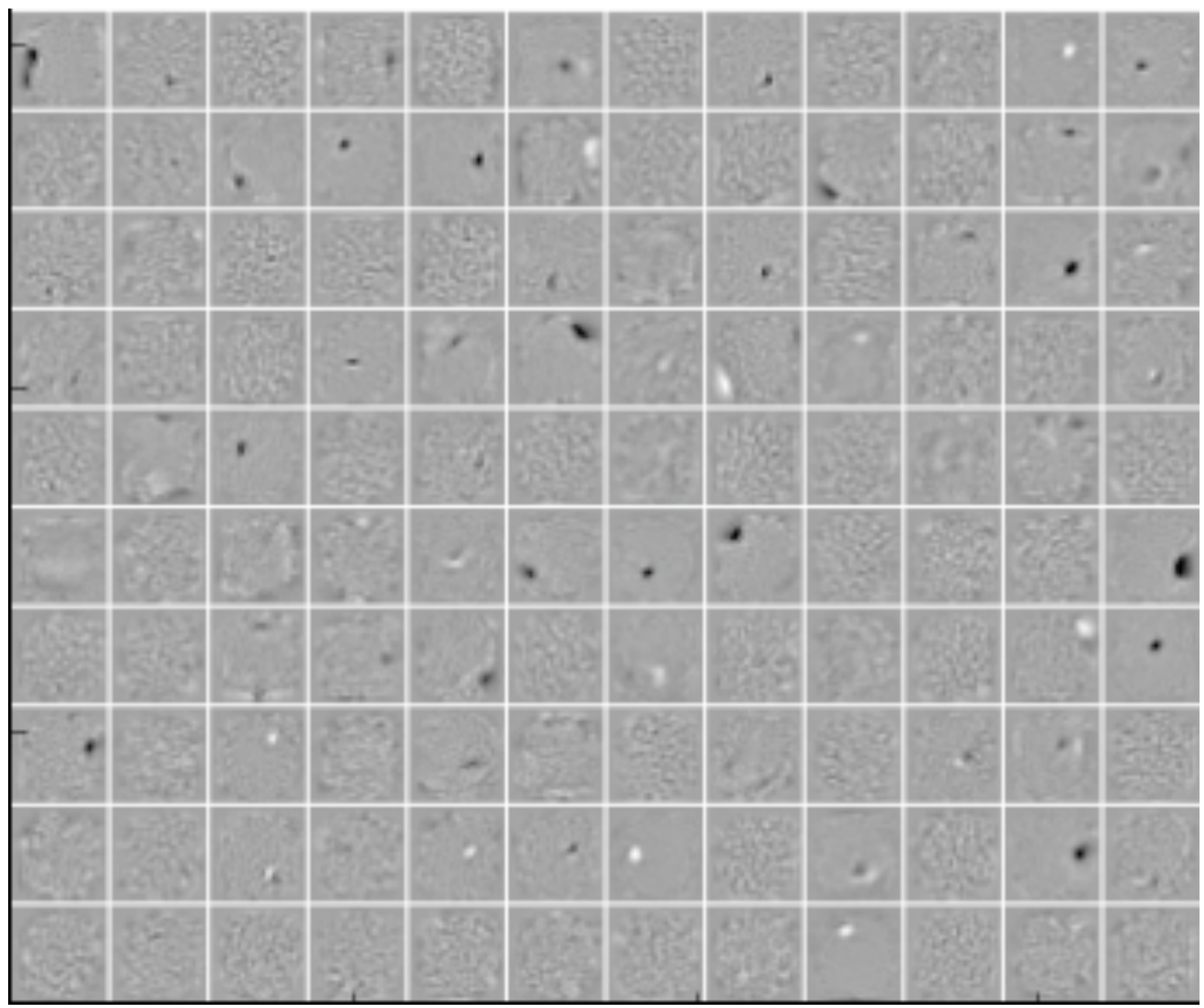
$\mathbf{b}^i \leftarrow \mathbf{b}^i - \varepsilon \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}$

$\mathbf{b}^{i-1} \leftarrow \mathbf{b}^{i-1} - \varepsilon \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$

Filtres ("denoising autoencoder")

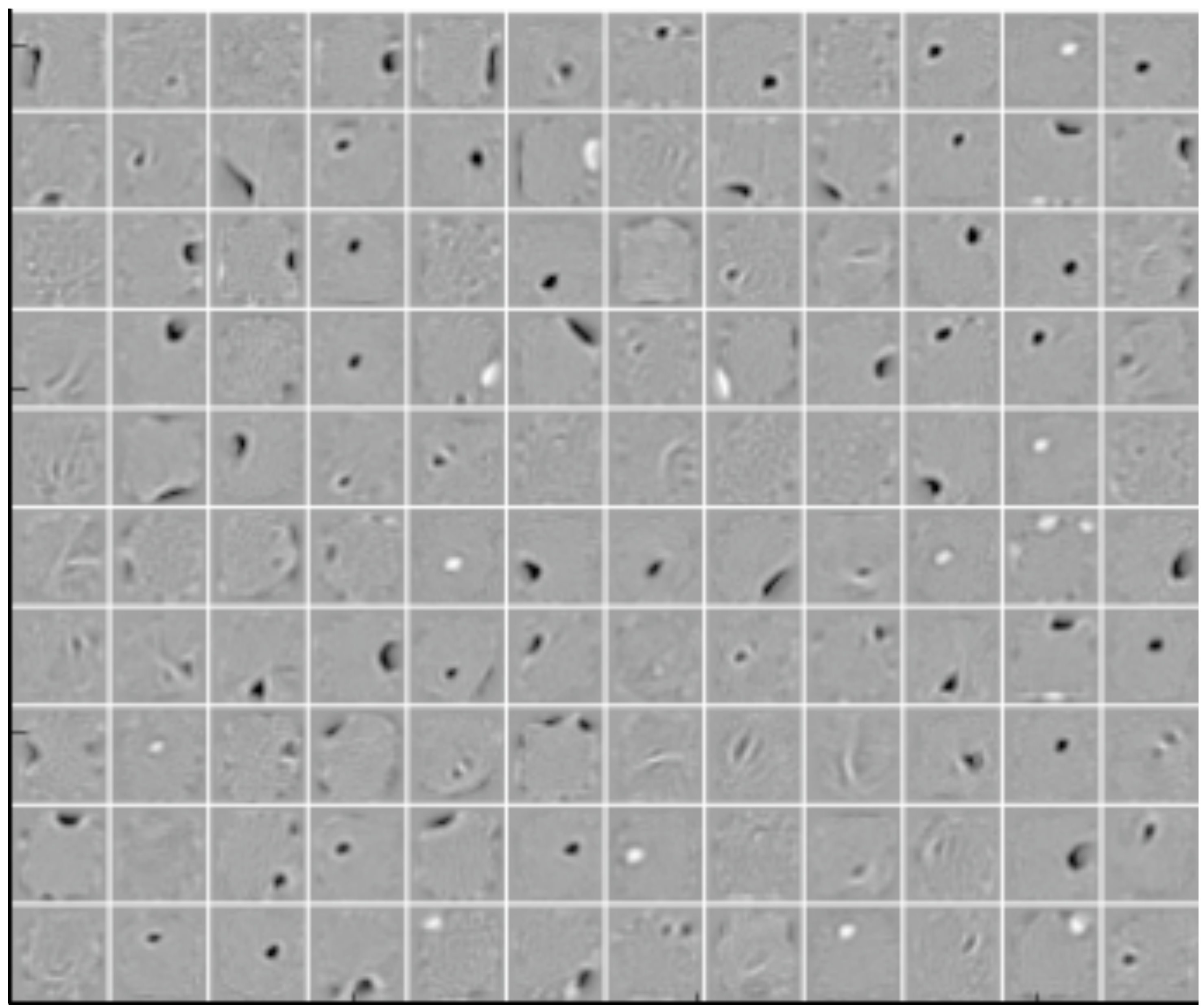
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

- Sans entrée manquante



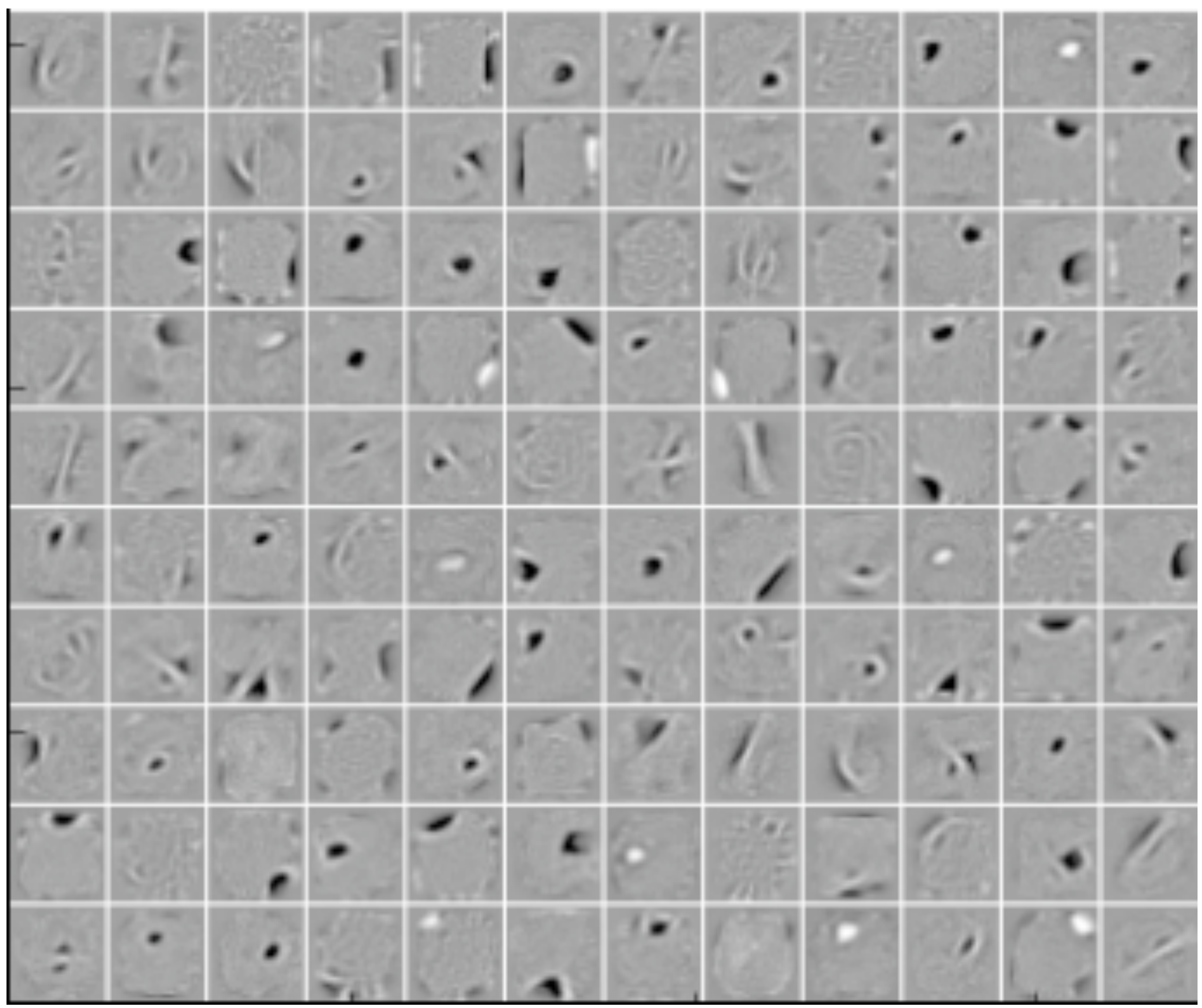
Filtres ("denoising autoencoder")
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

- 25% d'entrées manquantes:



Filtres ("denoising autoencoder")
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

- 50% d'entrée manquantes:

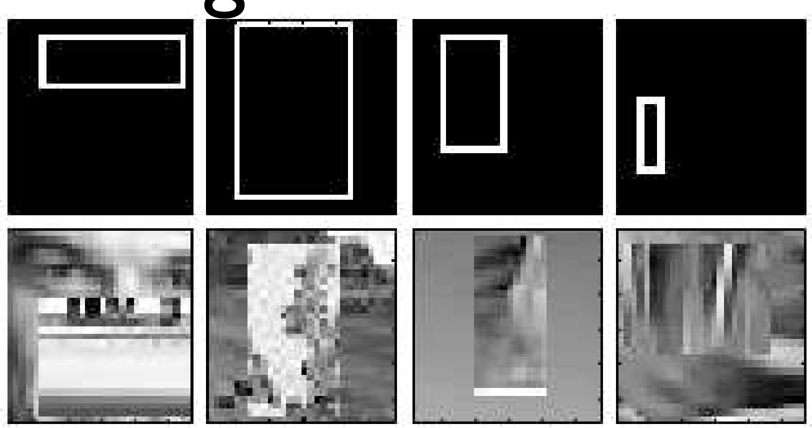


Comparaisons expérimentales

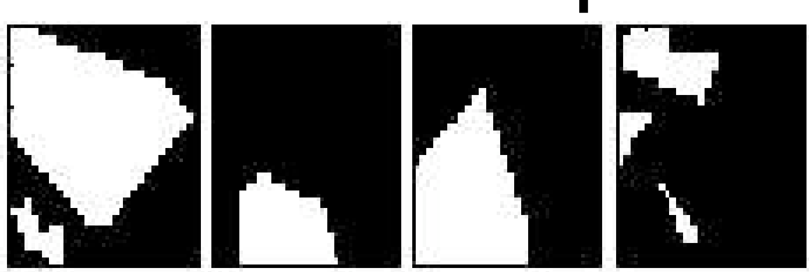
(Larochelle, et al. ICML 2007)

- Problèmes générés

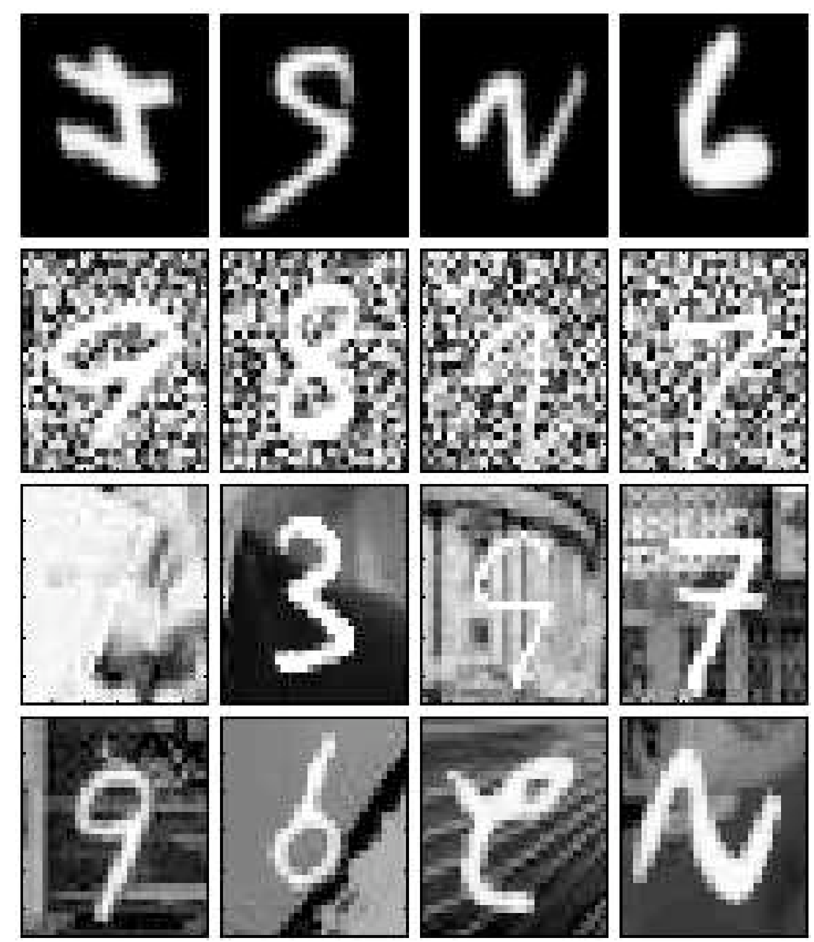
Large ou mince?



Convexe ou pas?



Variations sur MNIST



Résultats

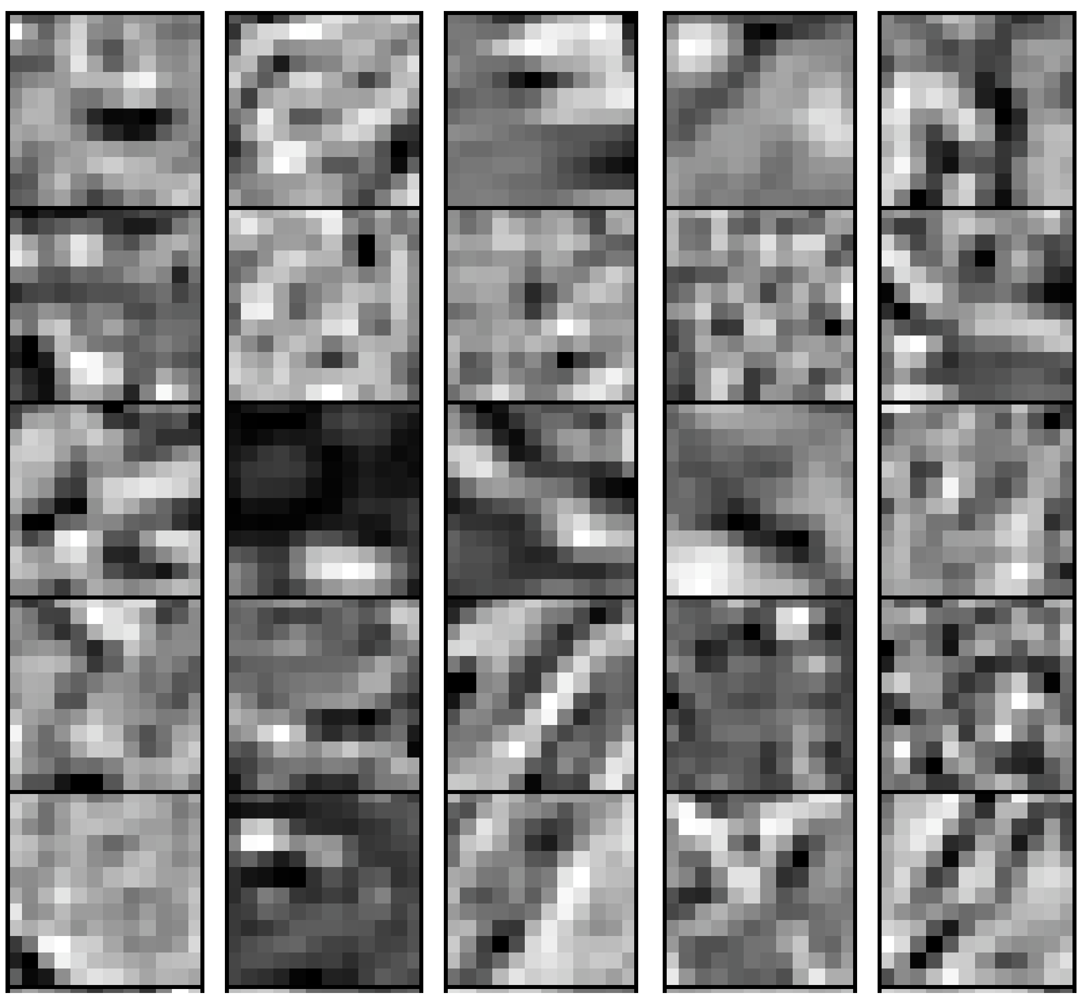
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

- Résultats

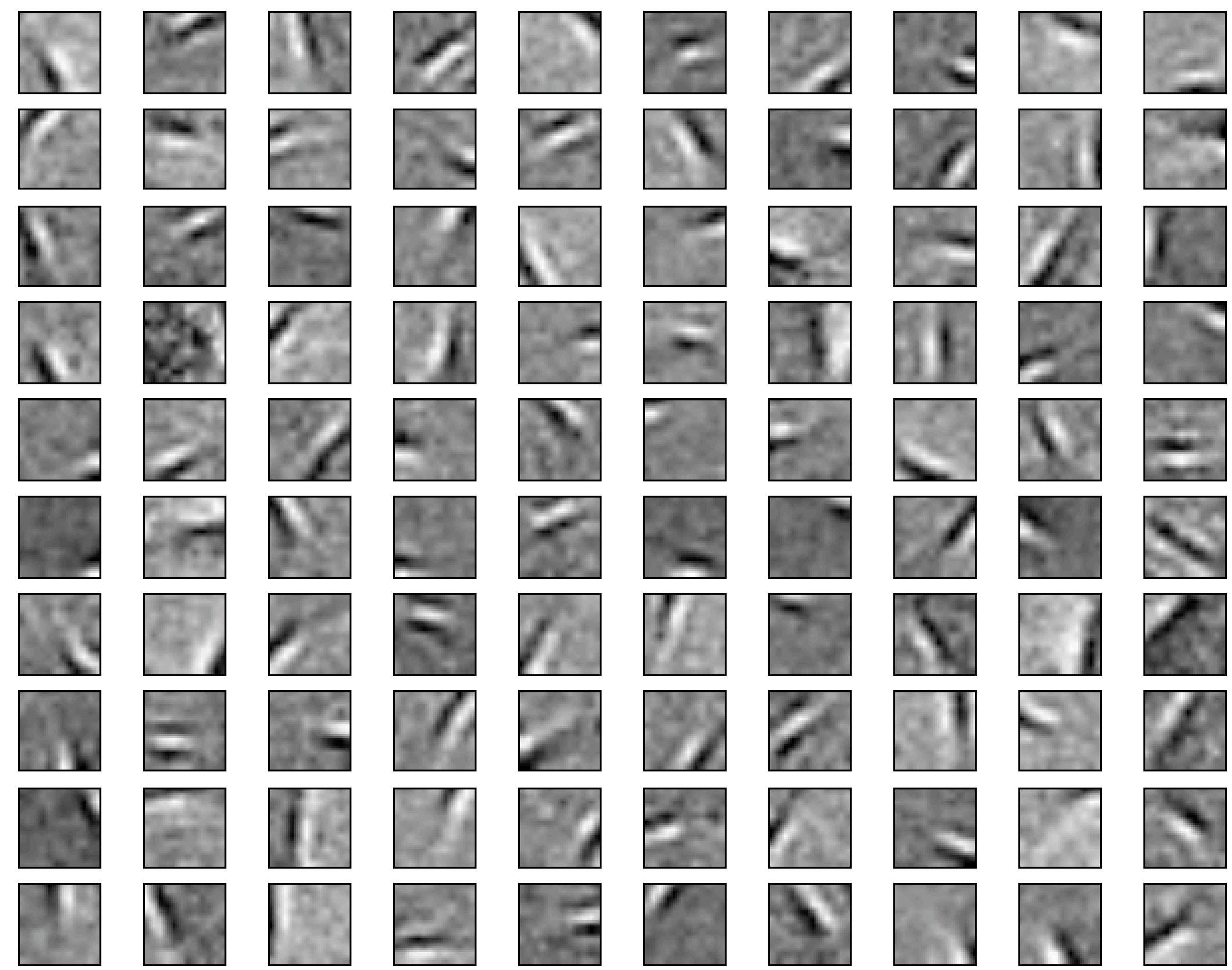
Dataset	**SVM**$_{rbf}$	**SAA-3**	**DBN-3**	**SdA-3** (ν)
basic	**3.03**$\pm$**0.15**	3.46$\pm$0.16	3.11$\pm$0.15	**2.80**$\pm$**0.14** (10%)
rot	11.11$\pm$0.28	**10.30**$\pm$**0.27**	**10.30**$\pm$**0.27**	**10.29**$\pm$**0.27** (10%)
bg-rand	14.58$\pm$0.31	11.28$\pm$0.28	**6.73**$\pm$**0.22**	10.38$\pm$0.27 (40%)
bg-img	22.61$\pm$0.37	23.00$\pm$0.37	**16.31**$\pm$**0.32**	**16.68**$\pm$**0.33** (25%)
rot-bg-img	55.18$\pm$0.44	51.93$\pm$0.44	47.39$\pm$0.44	**44.49**$\pm$**0.44** (25%)
rect	**2.15**$\pm$**0.13**	2.41$\pm$0.13	2.60$\pm$0.14	**1.99**$\pm$**0.12** (10%)
rect-img	24.04$\pm$0.37	24.05$\pm$0.37	22.50$\pm$0.37	**21.59**$\pm$**0.36** (25%)
convex	19.13$\pm$0.34	**18.41**$\pm$**0.34**	**18.63**$\pm$**0.34**	**19.06**$\pm$**0.34** (10%)

Coût de l'erreur au carré

- La PCA n'est plus la meilleure solution



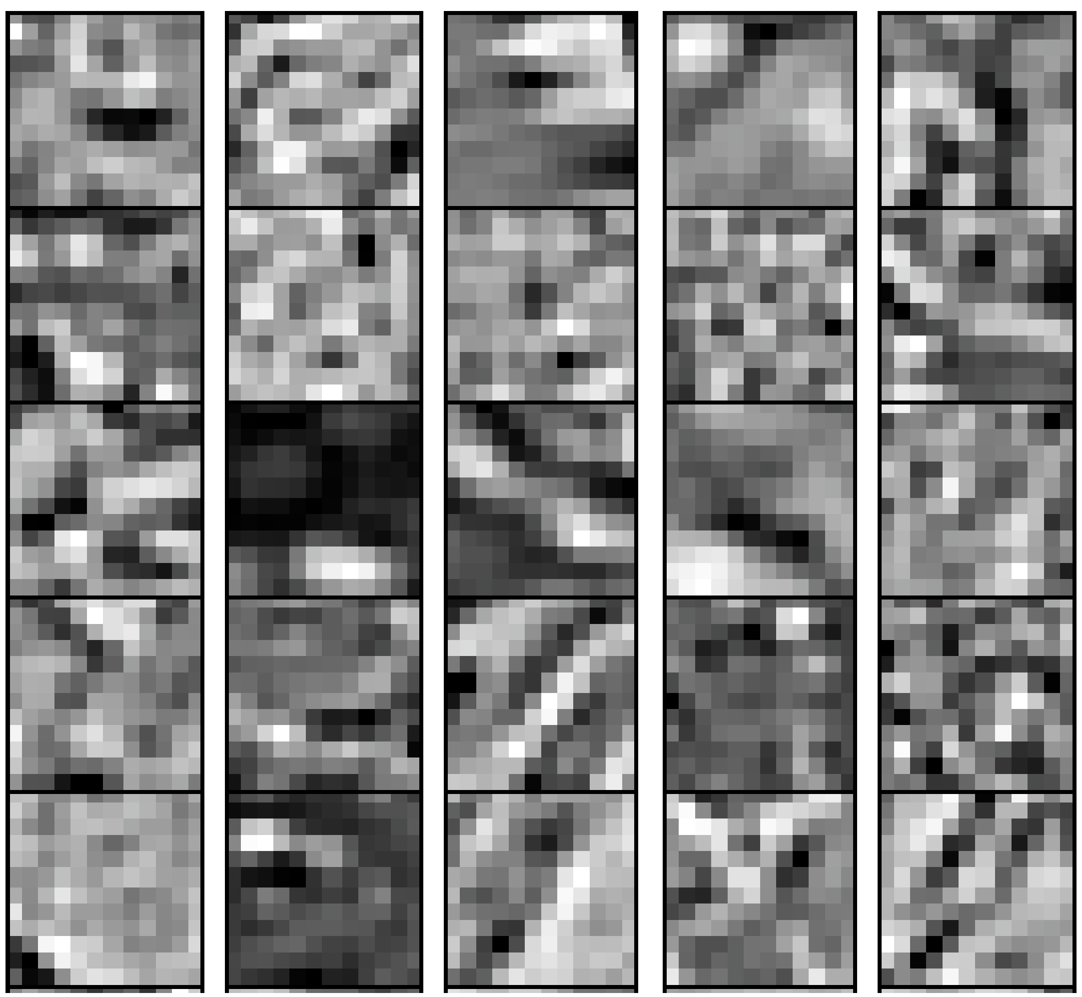
Données



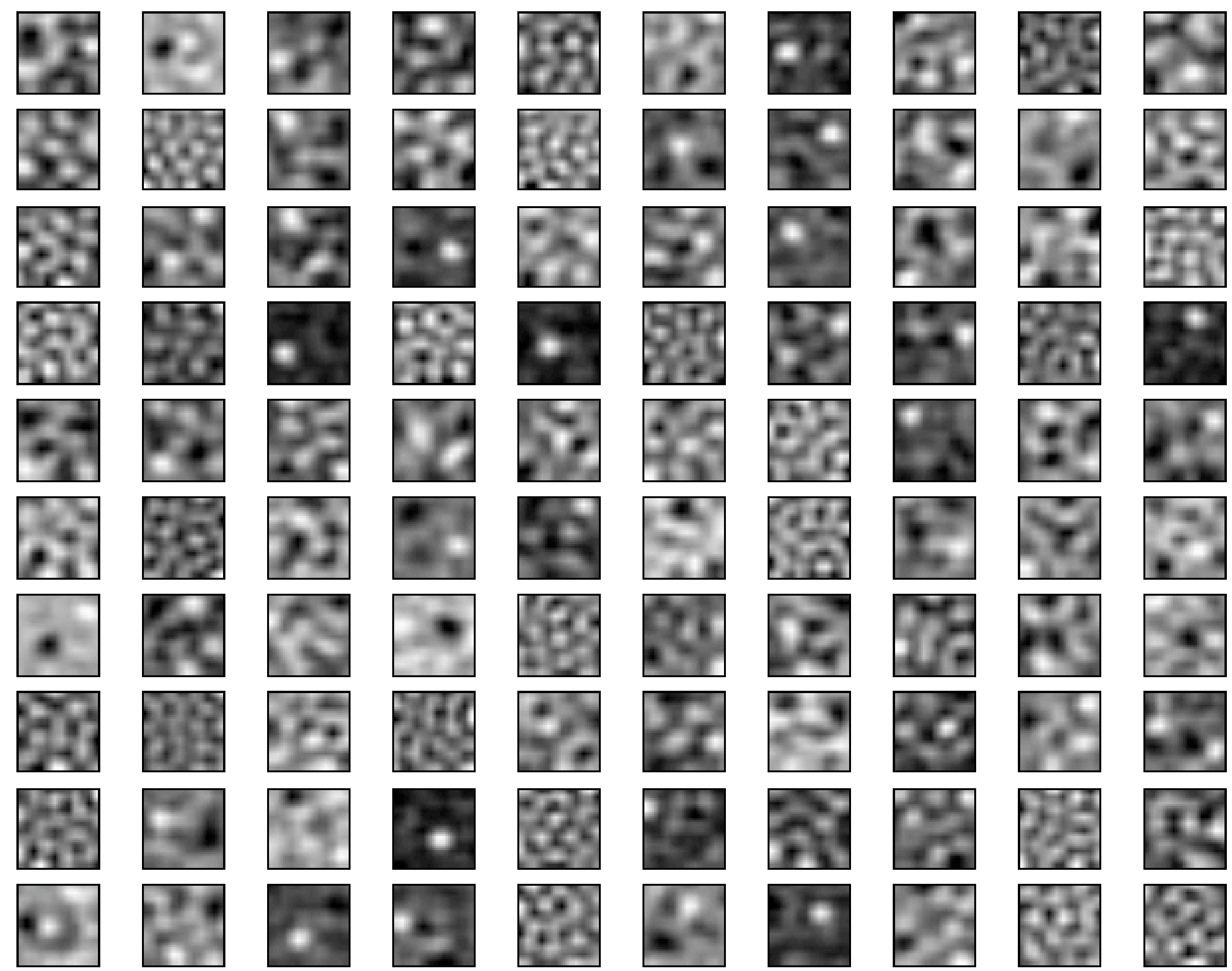
Filtres

Coût de l'erreur au carré

- N'est pas équivalent à du "weight decay"



Données



Filtres

Quelle sorte de bruit utiliser?

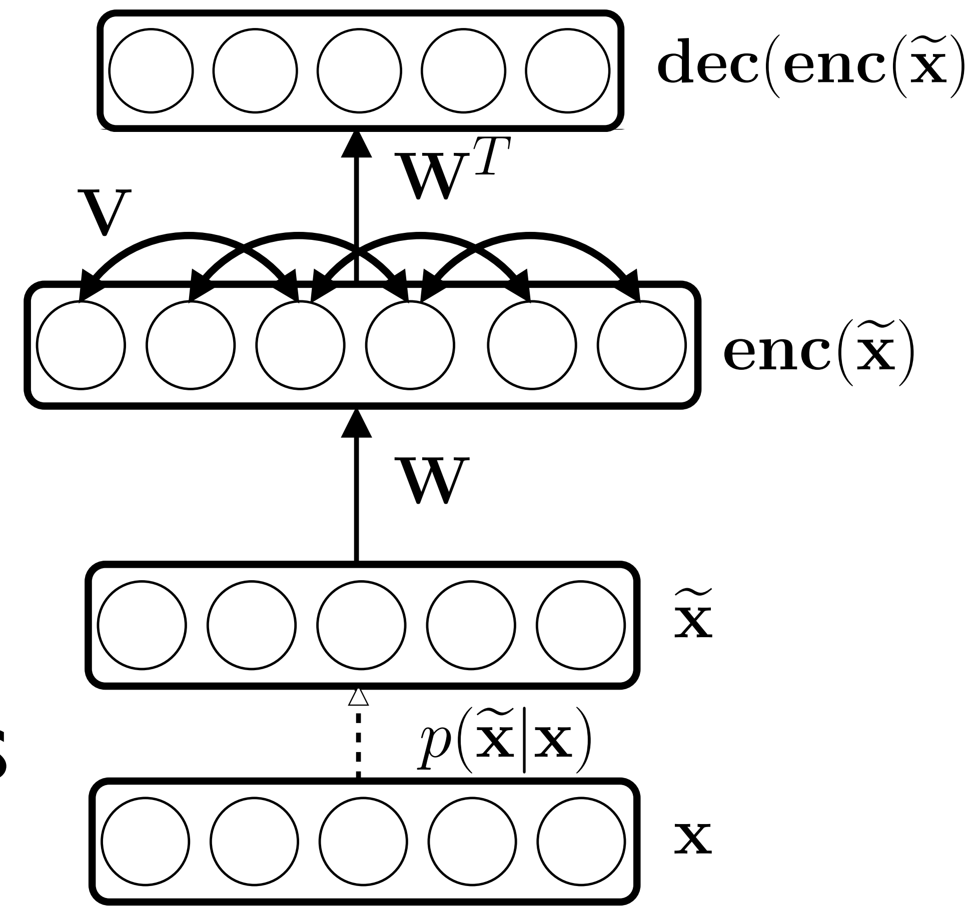
- “Enlever” un pourcentage des unités d’entrées marche bien, mais n’est peut-être pas optimal
- Est-ce que le bruit Gaussien est le mieux qu’on puisse faire pour des données continues?

! Piste de recherche !

Deep Learning using Robust Interdependent Codes

(Larochelle, Erhan and Vincent, 2009)

- On introduit des connexions latérales V parmi les éléments de la couche cachée
- Typiquement, ces connexions impliquent des interactions récurrentes entre les neurones (Shriki, Sompolinsky and Lee, 2001)



$$\mathbf{enc}(\widetilde{\mathbf{x}})_j = \mathrm{sigm}\left(b_j + \sum_k W_{jk}\tilde{x}_k + \sum_{k \neq j} V_{jk}\mathbf{enc}(\widetilde{\mathbf{x}})_k\right)$$

Deep Learning using Robust Interdependent Codes

(Larochelle, Erhan and Vincent, 2009)

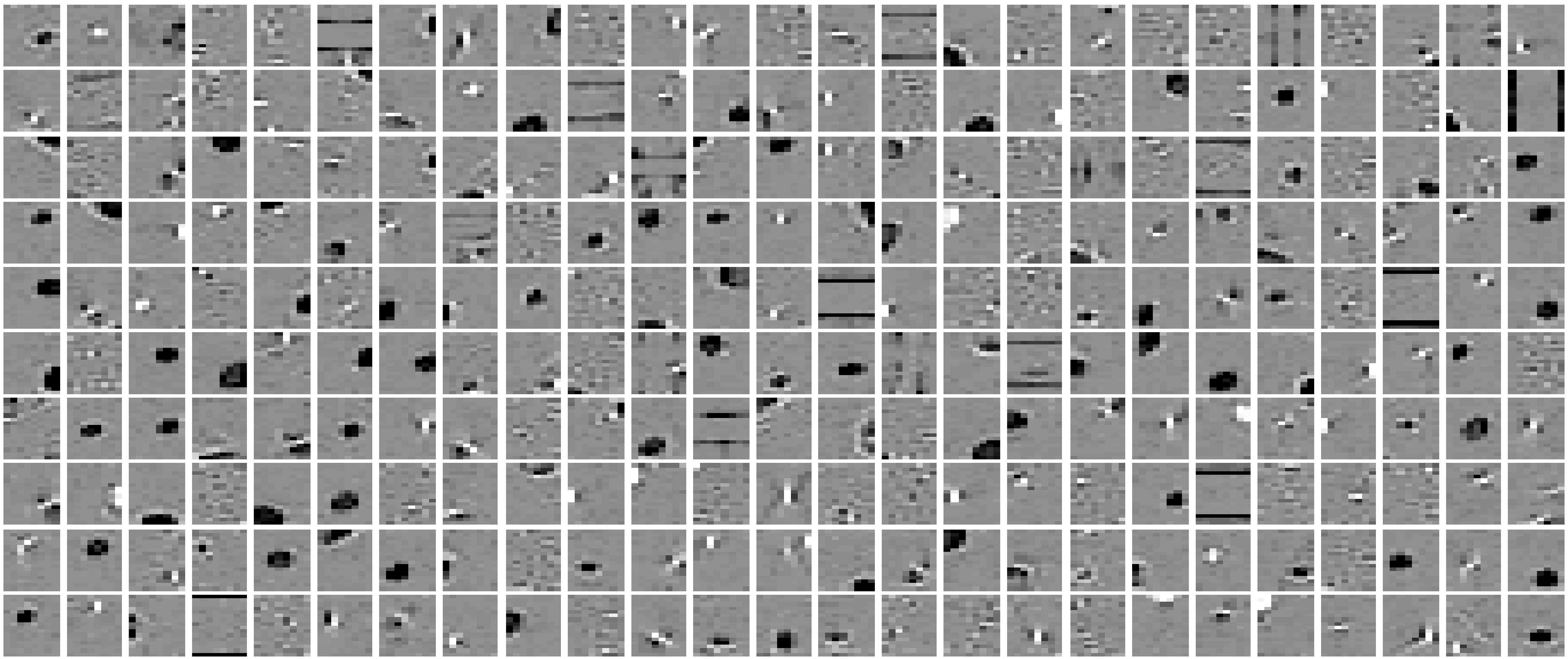
- Malheureusement, un tel encodeur nécessite plusieurs itérations pour converger
- À la place, on insère des connexions non-récurrentes

1) $\widehat{\mathbf{enc}(\widetilde{\mathbf{x}})} = \mathrm{sigm}(\mathbf{b} + \mathbf{W}\tilde{\mathbf{x}})$

2) $\mathbf{enc}(\widetilde{\mathbf{x}})_j = \mathrm{sigm}\left(d_j + V_{jj}\widehat{\mathbf{enc}(\widetilde{\mathbf{x}})}_j + \sum_{k \neq j} V_{jk}\widehat{\mathbf{enc}(\widetilde{\mathbf{x}})}_k\right), \text{ où } V_{jj} > 0$

Deep Learning using Robust Interdependent Codes

(Larochelle, Erhan and Vincent, 2009)

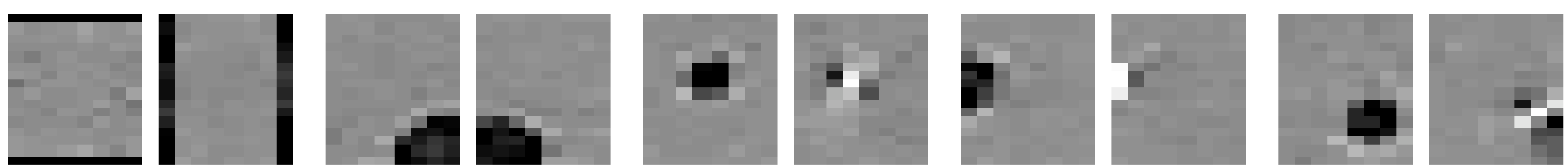


Deep Learning using Robust Interdependent Codes

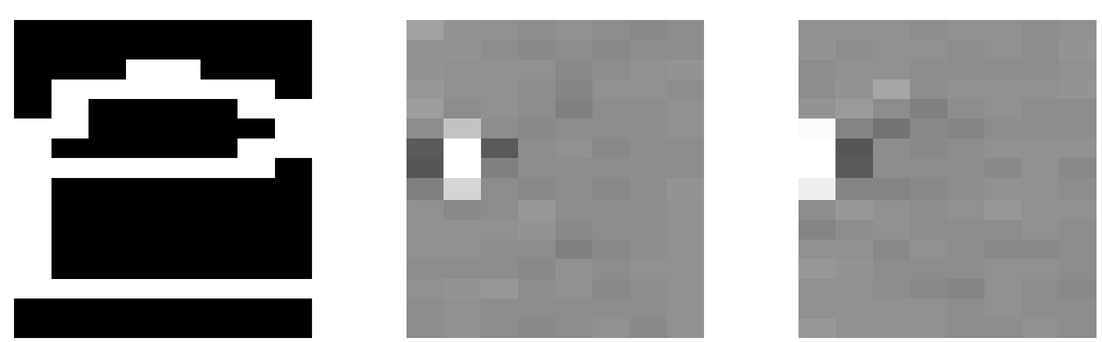
(Larochelle, Erhan and Vincent, 2009)

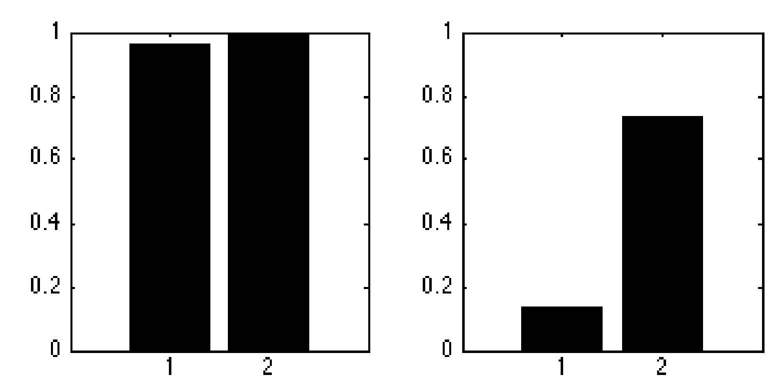
Neurones connectés positivement par $\mathbf{V}$:

Neurones connectés négativement par $\mathbf{V}$:



Exemple d'interaction d'inhibition →





Deep Learning using Robust Interdependent Codes

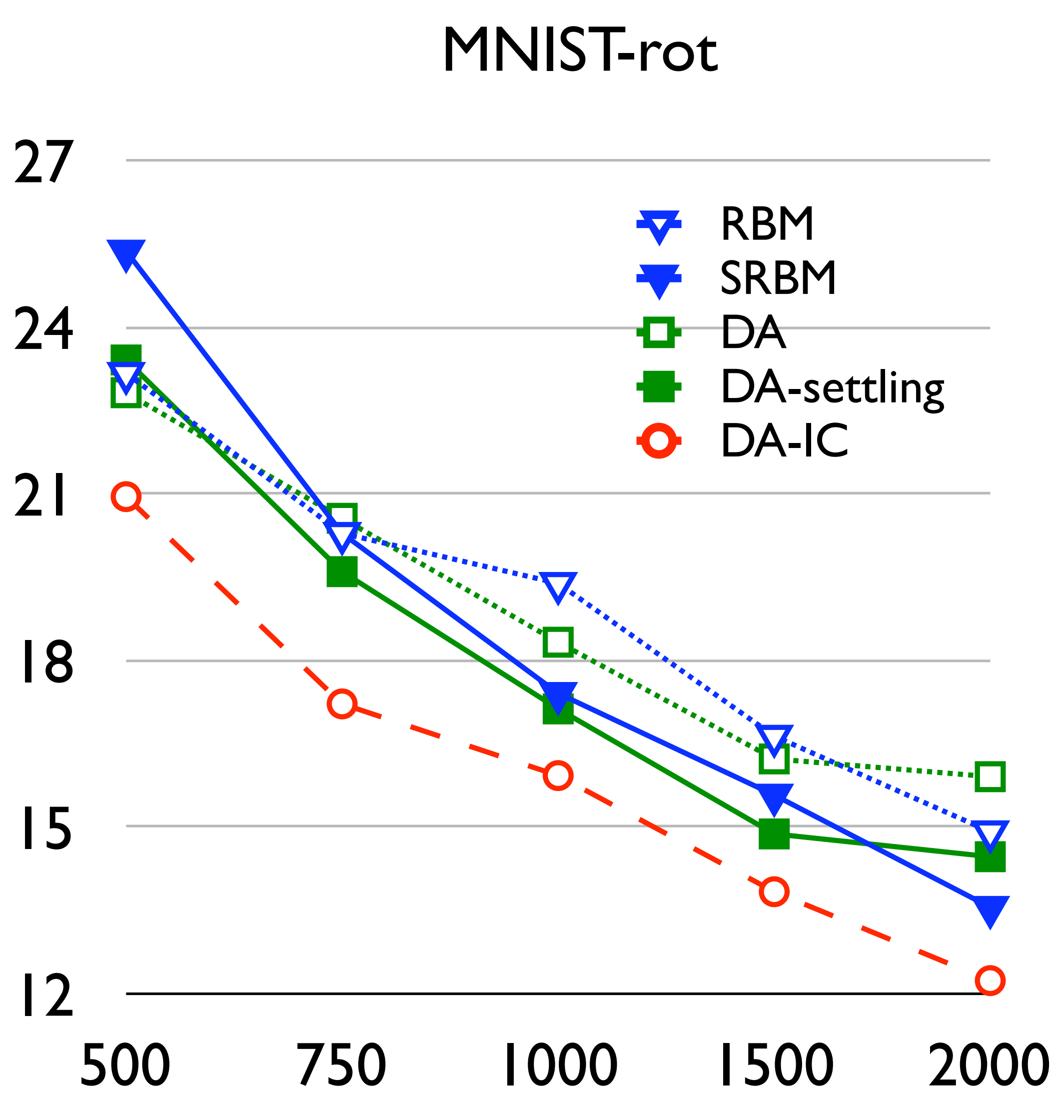
(Larochelle, Erhan and Vincent, 2009)

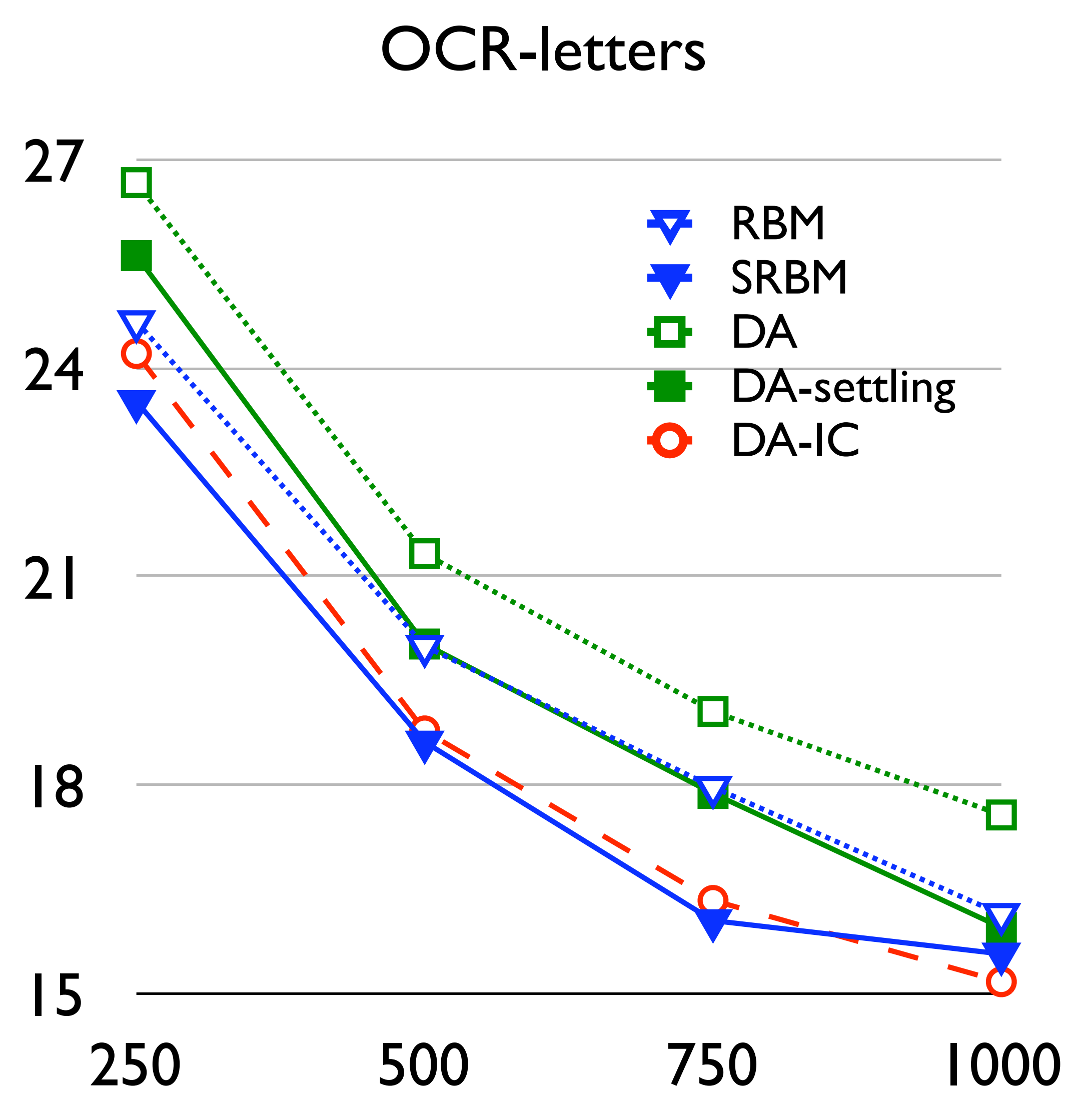
Résultats de classification

Dataset	**SVM**$_{rbf}$	**DBN-3**	**SDA-3**	**SDA-6**	**SDAIC-3**
MNIST-rot	11.11	9.01	9.53	9.68	**8.07**
OCR-letters (fold 1)	9.70	9.68	9.69	10.15	**9.60**
OCR-letters (fold 2)	9.36	9.68	9.92	9.92	**9.31**
OCR-letters (fold 3)	9.94	10.07	10.29	10.32	**9.46**
OCR-letters (fold 4)	10.32	10.46	10.42	10.51	**9.92**
OCR-letters (fold 5)	10.19	10.58	9.93	10.58	**9.50**
OCR-letters (all)	9.90	10.09	10.05	10.30	**9.56**

Deep Learning using Robust Interdependent Codes

(Larochelle, Erhan and Vincent, 2009)

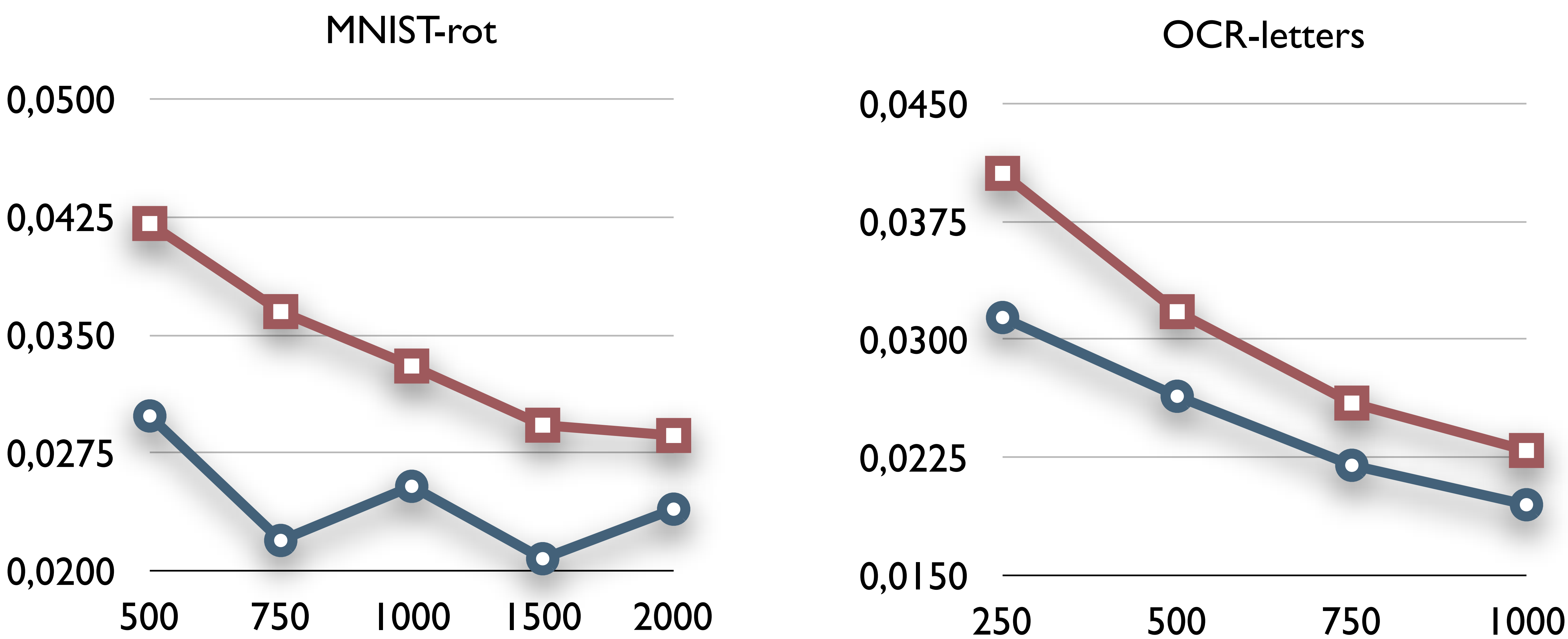




Résultats extraction de caractéristiques (features)

Deep Learning using Robust Interdependent Codes

(Larochelle, Erhan and Vincent, 2009)



Résultats extraction de caractéristiques (features)

Quelle sorte d’encodeur utiliser?

- Plusieurs autres choix d’encodeur pourraient sont possibles
- L’encodeur “optimal” dépend peut-être de la tâche (image vs. signal sonore vs. texte)

! Piste de recherche !

Deep Learning via Semi-Supervised Embedding
(Weston, Ratle and Collobert, ICML 2008)

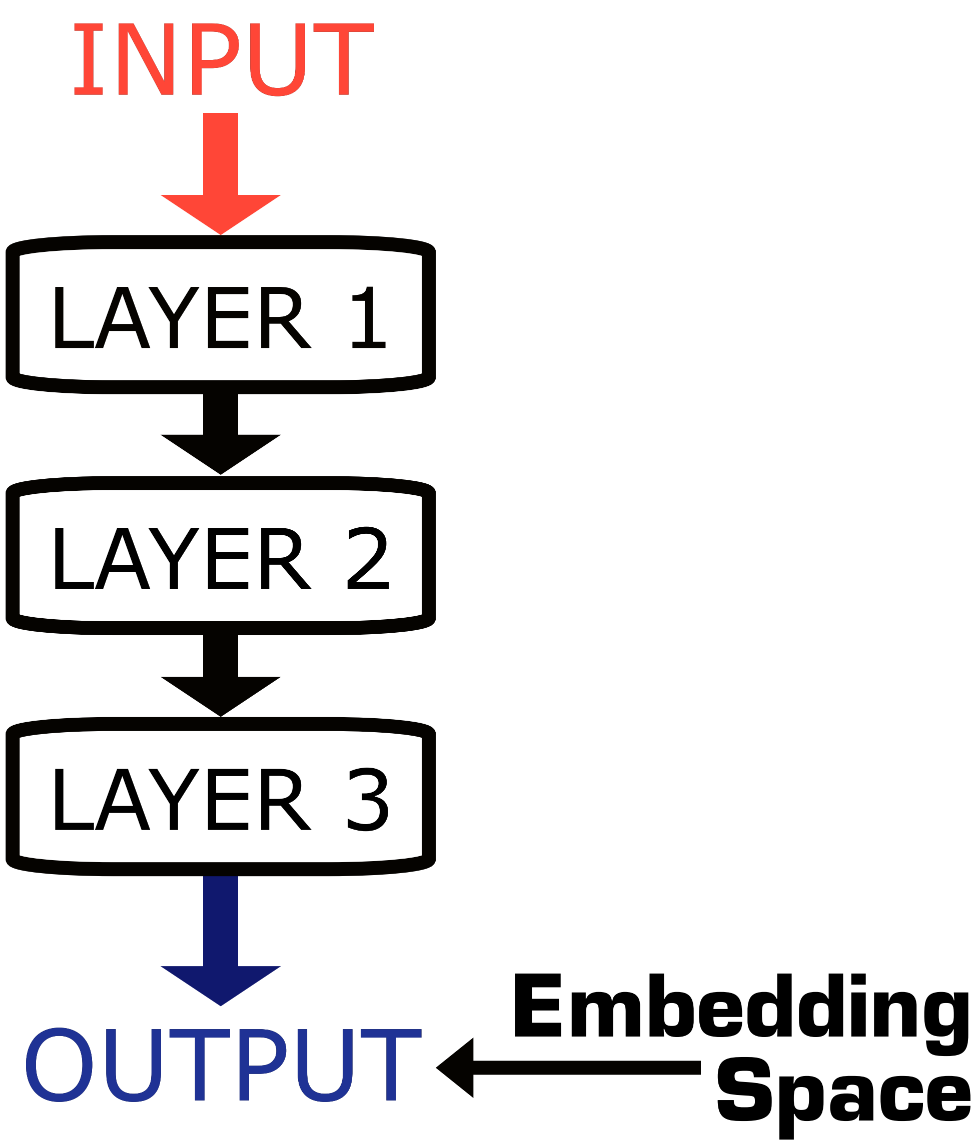
- Pourquoi ne pas s'inspirer de la littérature sur l'apprentissage semi-supervisé

$$L(f_i, f_j, W_{ij}) = \begin{cases} ||f_i - f_j||^2 & \text{if } W_{ij} = 1, \\ \max(0, m - ||f_i - f_j||^2) & \text{if } W_{ij} = 0 \end{cases}$$

Des entrées x_i, x_j devraient avoir des sorties f_i, f_j similaires **si elles sont voisines**

Deep Learning via Semi-Supervised Embedding

(Weston, Ratle and Collobert, ICML 2008)



coût supervisé

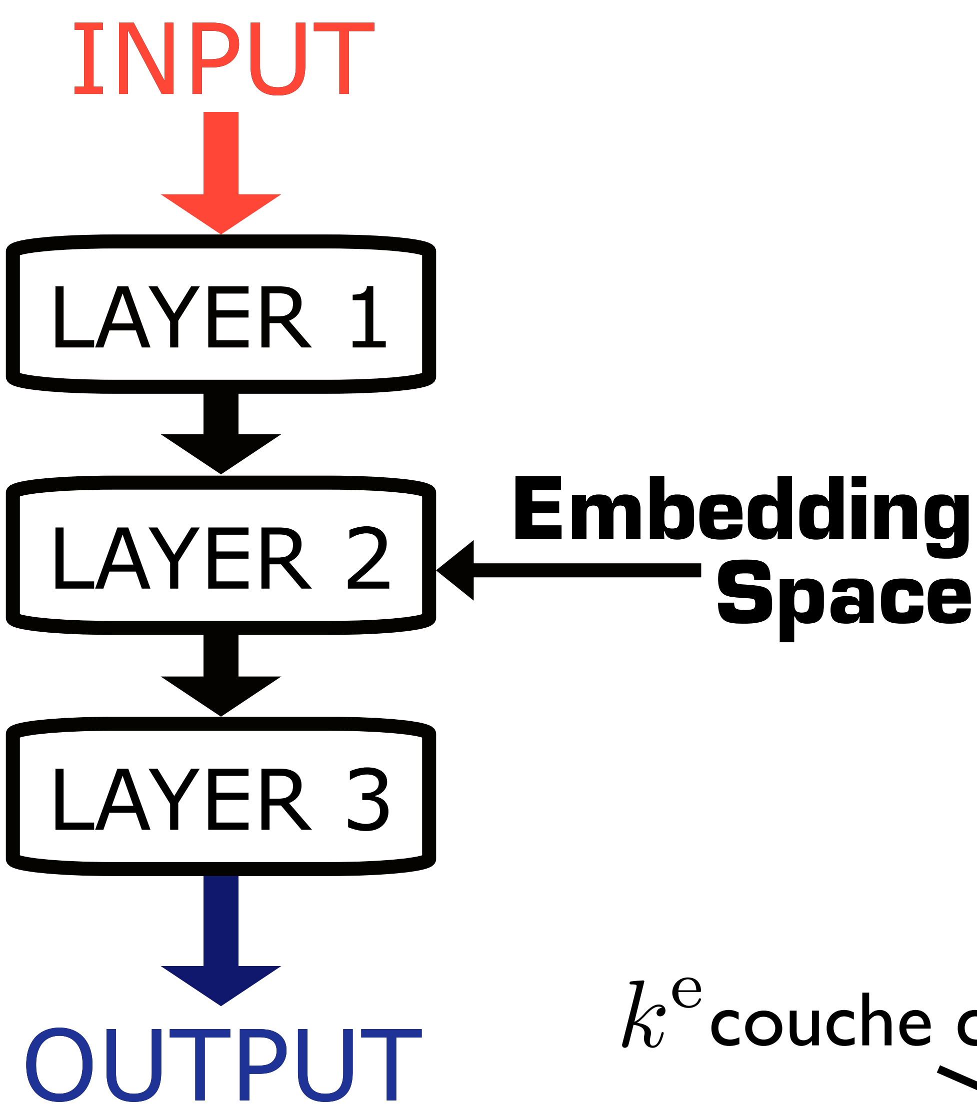
$$\sum_{i=1}^{L} \ell(f(x_i), y_i)$$

coût non-supervisé

$$+ \lambda \sum_{i,j=1}^{L+U} L(f(x_i), f(x_j), W_{ij})$$

Deep Learning via Semi-Supervised Embedding

(Weston, Ratle and Collobert, ICML 2008)



coût supervisé

$$\sum_{i=1}^{L} \ell(f(x_i), y_i)$$

coût non-supervisé

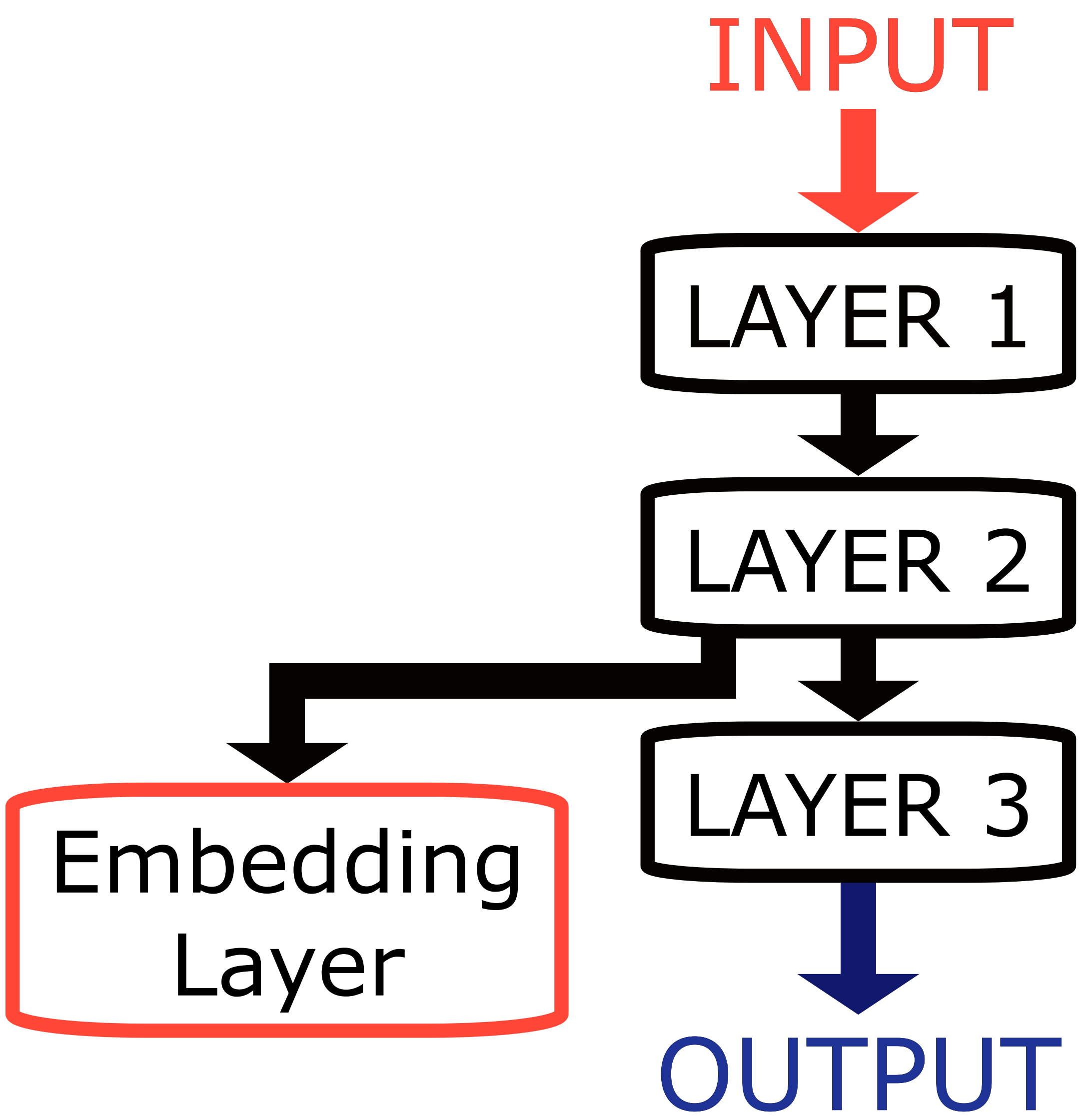
$$+ \lambda \sum_{i,j=1}^{L+U} L(f^k(x_i), f^k(x_j), W_{ij})$$

k^{e} couche cachée

$$f^k(x) = (h_1^k(x), \ldots, h_{N_k}^k(x))$$

Deep Learning via Semi-Supervised Embedding

(Weston, Ratle and Collobert, ICML 2008)



coût supervisé

$$\sum_{i=1}^{L} \ell(f(x_i), y_i)$$

coût non-supervisé

$$+ \lambda \sum_{i,j=1}^{L+U} L(\ g(x_i),\ \ g(x_j), W_{ij})$$

$$g_i(x) = \sum_j w_j^{AUX,i}\ h_j^k(x) + b^{AUX,i}$$

Deep Learning via Semi-Supervised Embedding

(Weston, Ratle and Collobert, ICML 2008)

Algorithm 1 *Embed*NN

Input: labeled data (x_i, y_i), $i = 1, \ldots, L$, unlabeled data x_i, $i = L+1, \ldots, U$, set of functions $f(\cdot)$, and embedding functions $g^k(\cdot)$, see Figure 1 and equations (9), (10) and (11).

repeat

  Pick a random *labeled* example (x_i, y_i)

  Make a gradient step to optimize $\ell(f(x_i), y_i)$

  for each embedding function $g^k(\cdot)$ **do**

    Pick a random pair of neighbors x_i, x_j.

    Make a gradient step for $\lambda L(g^k(x_i), g^k(x_j), 1)$

    Pick a random unlabeled example x_n.

    Make a gradient step for $\lambda L(g^k(x_i), g^k(x_n), 0)$

  end for

until stopping criteria is met.

Deep Learning via Semi-Supervised Embedding
(Weston, Ratle and Collobert, ICML 2008)

- Résultats

	Mnst1h	Mnst6h	Mnst1k	Mnst3k
SVM	23.44	8.85	7.77	4.21
TSVM	16.81	6.16	5.38	3.45
RBM$^{(*)}$	21.5	-	8.8	-
SESM$^{(*)}$	20.6	-	9.6	-
DBN-NCA$^{(*)}$	-	10.0	-	3.8
DBN-rNCA$^{(*)}$	-	8.7	-	3.3
NN	25.81	11.44	10.70	6.04
$Embed^{O}$NN	17.05	5.97	5.73	3.59
$Embed^{I1}$NN	16.86	9.44	8.52	6.02
$Embed^{A1}$NN	17.17	7.56	7.89	4.93
CNN	22.98	7.68	6.45	3.35
$Embed^{O}$CNN	11.73	3.42	3.34	2.28
$Embed^{I5}$CNN	7.75	3.82	2.73	1.83
$Embed^{A5}$CNN	7.87	3.82	2.76	2.07

Deep Learning via Semi-Supervised Embedding
(Weston, Ratle and Collobert, ICML 2008)

- Résultats

Table 4. Mnist1h dataset with deep networks of 2, 6, 8, 10 and 15 layers; each hidden layer has 50 hidden units. We compare classical NN training with *Embed*NN where we either learn an embedding at the output layer (^{O}) or an auxiliary embedding on all layers at the same time (^{ALL}).
.

	2	4	6	8	10	15
NN	26.0	26.1	27.2	28.3	34.2	47.7
$Embed^{O}$NN	19.7	15.1	15.1	15.0	13.7	11.8
$Embed^{ALL}$NN	18.2	12.6	7.9	8.5	6.3	9.3

Deep Learning via Semi-Supervised Embedding

(Weston, Ratle and Collobert, ICML 2008)

- Résultats

Table 5. Full Mnist60k dataset with deep networks of 2, 6, 8, 10 and 15 layers, using either 50 or 100 hidden units. We compare classical NN training with $Embed^{ALL}$NN where we learn an auxiliary embedding on all layers at the same time.

	2	4	6	8	10	15
NN (HUs=50)	2.9	2.6	2.8	3.1	3.1	4.2
$Embed^{ALL}$NN	2.8	1.9	2.0	2.2	2.4	2.6
NN (HUs=100)	2.0	1.9	2.0	2.2	2.3	3.0
$Embed^{ALL}$NN	1.9	1.5	1.6	1.7	1.8	2.4

Deep Learning via Semi-Supervised Embedding
(Weston, Ratle and Collobert, ICML 2008)

- Avantages
  - ★ n’a pas de décodeur, pratique en haute dimension
- Désavantages
  - ★ doit définir quelles paires d’entrées sont voisines

! Piste de recherche !

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Est-il possible d'avoir une machine à noyau profond?
- Est-il possible d'avoir une couche cachée infinie?

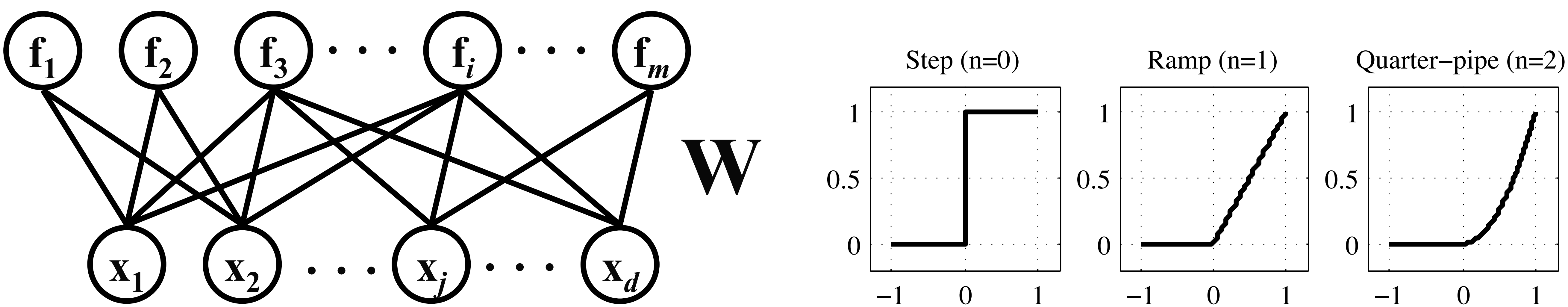


Figure 1: Single layer network and activation functions

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

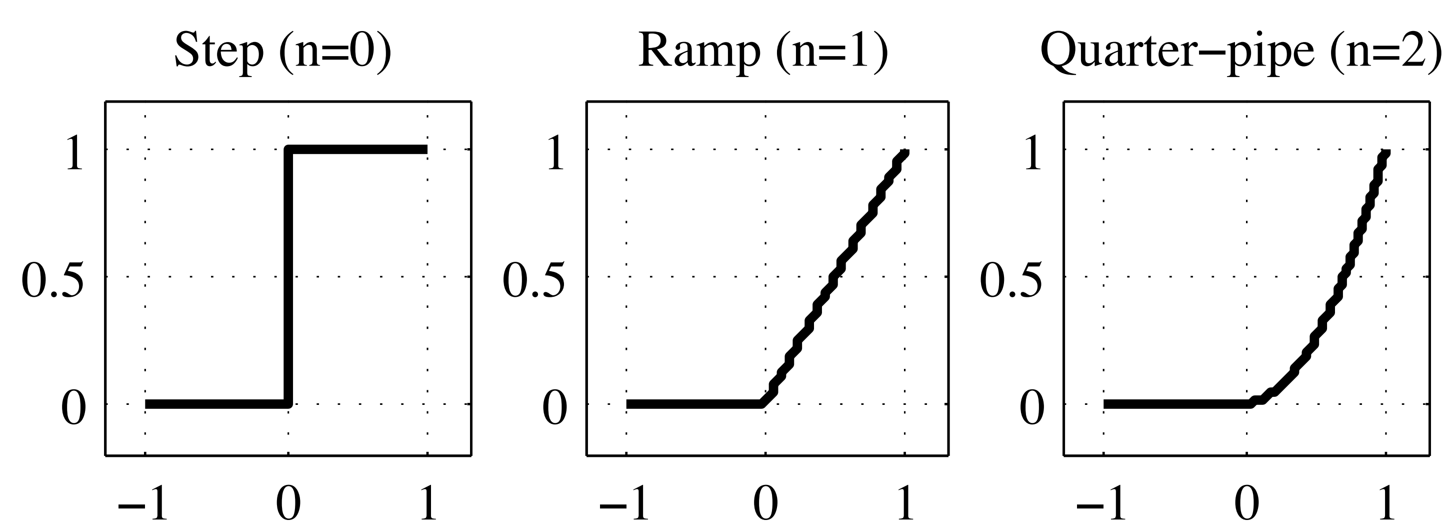
- Dans une machine à noyau, on s'intéresse aux produits scalaires entre vecteurs

$$\mathbf{f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{y}) \;=\; \sum_{i=1}^{m} \Theta(\mathbf{w}_i \cdot \mathbf{x})\Theta(\mathbf{w}_i \cdot \mathbf{y})(\mathbf{w}_i \cdot \mathbf{x})^n(\mathbf{w}_i \cdot \mathbf{y})^n,$$

$$\Theta(z) = \tfrac{1}{2}(1 + \mathrm{sign}(z))$$

$$g_n(z) = \Theta(z) z^n$$

fonction d'activation



Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Nb. d'unités infini: somme devient intégrale

$$\mathbf{f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{y}) \;=\; \sum_{i=1}^{m} \Theta(\mathbf{w}_i \cdot \mathbf{x})\Theta(\mathbf{w}_i \cdot \mathbf{y})(\mathbf{w}_i \cdot \mathbf{x})^n(\mathbf{w}_i \cdot \mathbf{y})^n,$$

$$k_n(\mathbf{x}, \mathbf{y}) \;=\; 2\int d\mathbf{w}\; \frac{e^{-\frac{\|\mathbf{w}\|^2}{2}}}{(2\pi)^{d/2}}\; \Theta(\mathbf{w} \cdot \mathbf{x})\; \Theta(\mathbf{w} \cdot \mathbf{y})\; (\mathbf{w} \cdot \mathbf{x})^n\; (\mathbf{w} \cdot \mathbf{y})^n$$

on suppose que les poids sont générés
selon des Gaussiennes N(0,1)

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Ce noyau peut être calculé analytiquement

$$\theta = \cos^{-1}\left(\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\right) \qquad J_n(\theta) = (-1)^n(\sin\theta)^{2n+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^n\left(\frac{\pi-\theta}{\sin\theta}\right)$$

$$k_n(\mathbf{x},\mathbf{y}) = \frac{1}{\pi}\|\mathbf{x}\|^n\|\mathbf{y}\|^n J_n(\theta)$$

$$\updownarrow$$

$$k_n(\mathbf{x},\mathbf{y}) = 2\int \mathbf{dw}\, \frac{e^{-\frac{\|\mathbf{w}\|^2}{2}}}{(2\pi)^{d/2}}\, \Theta(\mathbf{w}\cdot\mathbf{x})\,\Theta(\mathbf{w}\cdot\mathbf{y})\,(\mathbf{w}\cdot\mathbf{x})^n\,(\mathbf{w}\cdot\mathbf{y})^n$$

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Ce noyau peut être calculé analytiquement

$$\theta = \cos^{-1}\left(\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\right) \qquad J_n(\theta) = (-1)^n(\sin\theta)^{2n+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^n\left(\frac{\pi-\theta}{\sin\theta}\right)$$

$$k_n(\mathbf{x},\mathbf{y}) = \frac{1}{\pi}\|\mathbf{x}\|^n\|\mathbf{y}\|^n J_n(\theta)$$

Pour $n \in \{0, 1, 2\}$

$$\begin{aligned}
J_0(\theta) &= \pi - \theta \\
J_1(\theta) &= \sin\theta + (\pi-\theta)\cos\theta \\
J_2(\theta) &= 3\sin\theta\cos\theta + (\pi-\theta)(1+2\cos^2\theta)
\end{aligned}$$

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Il est même possible d'empiler des couches!

$$\theta_n^{(\ell)} = \cos^{-1}\left(k_n^{(\ell)}(\mathbf{x},\mathbf{y})\left[k_n^{(\ell)}(\mathbf{x},\mathbf{x})\,k_n^{(\ell)}(\mathbf{y},\mathbf{y})\right]^{-1/2}\right)$$

$$k_n^{(l+1)}(\mathbf{x},\mathbf{y}) = \frac{1}{\pi}\left[\,k_n^{(l)}(\mathbf{x},\mathbf{x})\;k_n^{(l)}(\mathbf{y},\mathbf{y})\,\right]^{n/2} J_n\!\left(\theta_n^{(\ell)}\right)$$

$$k^{(\ell)}(\mathbf{x},\mathbf{y}) = \underbrace{\boldsymbol{\Phi}(\boldsymbol{\Phi}(\ldots\boldsymbol{\Phi}}_{\ell\text{ times}}(\mathbf{x}))) \cdot \underbrace{\boldsymbol{\Phi}(\boldsymbol{\Phi}(\ldots\boldsymbol{\Phi}}_{\ell\text{ times}}(\mathbf{y})))$$

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

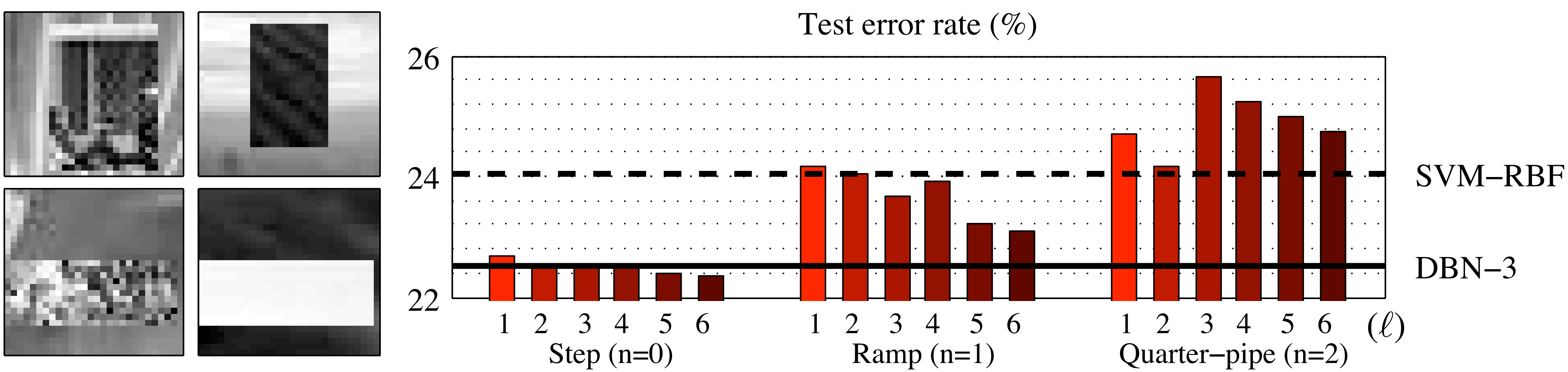


Figure 2: *Left*: examples from the *rectangles-image* data set. *Right*: classification error rates on the test set. SVMs with arc-cosine kernels have error rates from 22.36–25.64%. Results are shown for kernels of varying degree (n) and levels of recursion (ℓ). The best previous results are 24.04% for SVMs with RBF kernels and 22.50% for deep belief nets [11]. See text for details.

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

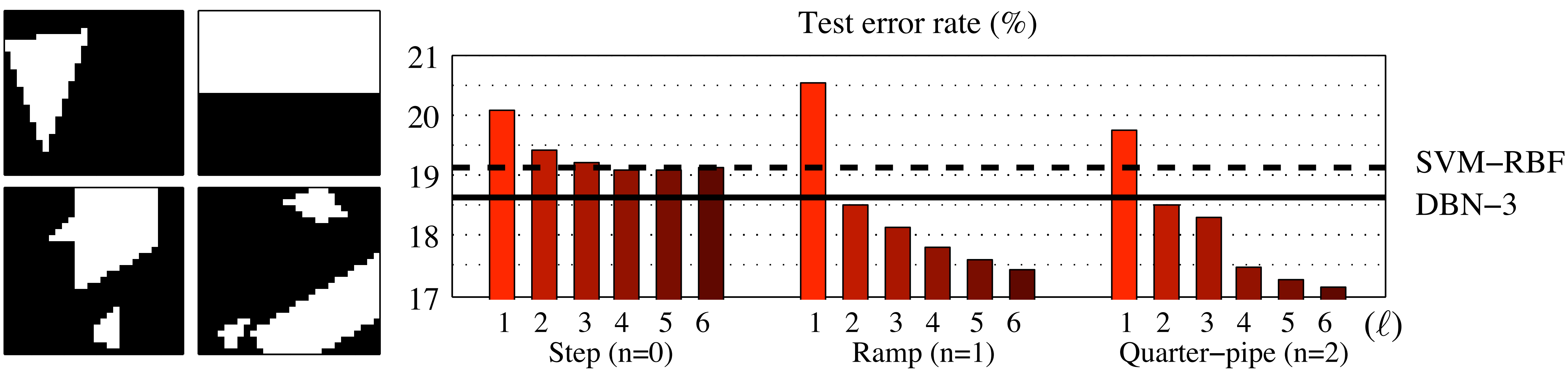


Figure 3: *Left*: examples from the *convex* data set. *Right*: classification error rates on the test set. SVMs with arc-cosine kernels have error rates from 17.15–20.51%. Results are shown for kernels of varying degree (n) and levels of recursion (ℓ). The best previous results are 19.13% for SVMs with RBF kernels and 18.63% for deep belief nets [11]. See text for details.

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Comment ajouter de l’apprentissage non-supervisé

1. Prune uninformative features from the input space.
2. Repeat ℓ times:
  (a) Compute principal components in the feature space induced by a nonlinear kernel.
  (b) Prune uninformative components from the feature space.
3. Learn a Mahalanobis distance metric for nearest neighbor classification.

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

- Comment ajouter de l'apprentissage non-supervisé

"kernel PCA"

1. Prune uninformative features from the input space.
2. Repeat ℓ times:
  (a) Compute principal components in the feature space induced by a nonlinear kernel.
  (b) Prune uninformative components from the feature space.
3. Learn a Mahalanobis distance metric for nearest neighbor classification.

"feature selection"

"large margin nearest neighbor"

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

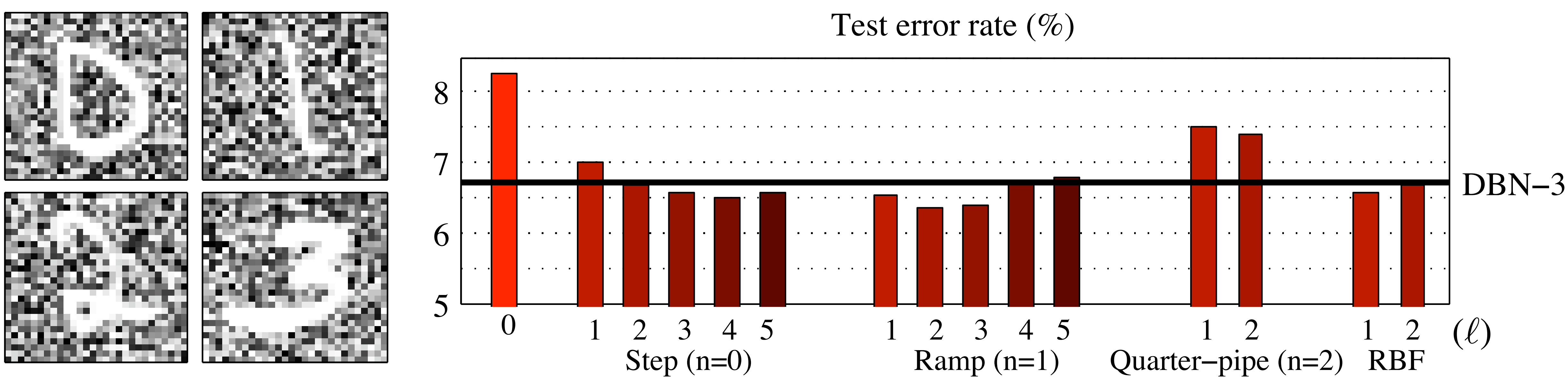


Figure 4: *Left*: examples from the *mnist-back-rand* data set. *Right*: classification error rates on the test set for MKMs with different kernels and numbers of layers ℓ. MKMs with arc-cosine kernel have error rates from 6.36–7.52%. The best previous results are 14.58% for SVMs with RBF kernels and 6.73% for deep belief nets [11].

Kernel Methods for Deep learning

(Cho and Saul, NIPS 2009)

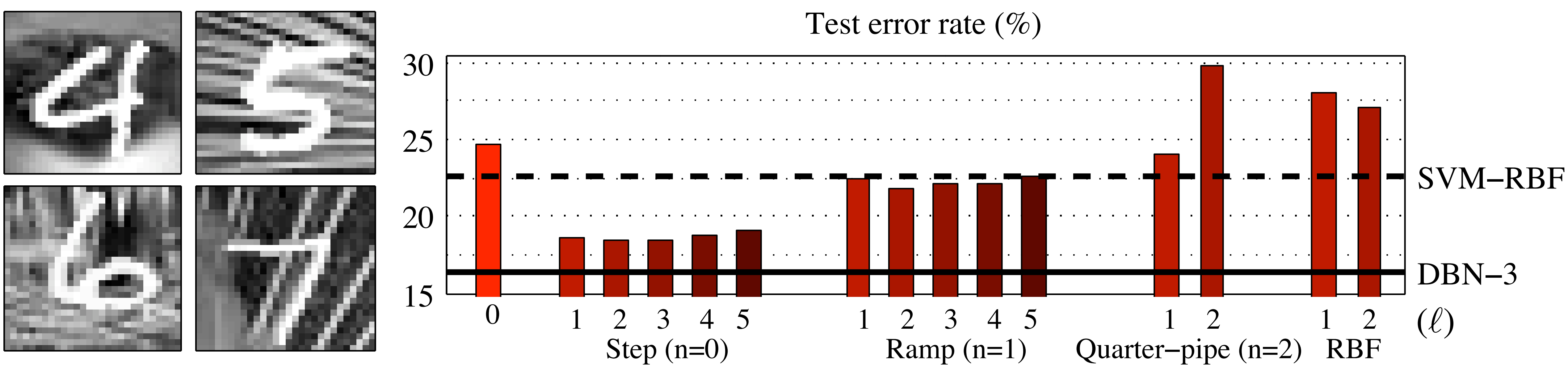


Figure 5: *Left*: examples from the *mnist-back-image* data set. *Right*: classification error rates on the test set for MKMs with different kernels and numbers of layers ℓ. MKMs with arc-cosine kernel have error rates from 18.43–29.79%. The best previous results are 22.61% for SVMs with RBF kernels and 16.31% for deep belief nets [11].

Kernel Methods for Deep learning
(Cho and Saul, NIPS 2009)

- Désavantages
  - ★ pas possible de faire du raffinement supervisé
  - ★ coûteux en temps et mémoire (kPCA)
- Avantages
  - ★ suite d'opérations convexes
  - ★ possible d'utiliser autre chose que la kPCA

! Piste de recherche !

Slow, Decorrelated Features for Pretraining Complex Cell-like Networks
(Bergstra and Bengio, NIPS 2009)

- Un autre exemple d'application du principe du pré-entraînement
  - ★ avec un critère non-supervisé différent
  - ★ avec un "encodeur" différent

Slow, Decorrelated Features for Pretraining Complex Cell-like Networks

(Bergstra and Bengio, NIPS 2009)

- Les neurones sont entraînés sur des séquences d'images
- Le critère:

$$L_{K2004} = \alpha \underbrace{\sum_{i!=j} \frac{\mathrm{Cov}_t(h_i, h_j)^2}{\mathrm{Var}(h_i)\mathrm{Var}(h_j)}}_{\text{neurones doivent être non-corrélés}} + \underbrace{\sum_t \sum_i \frac{(h_{i,t} - h_{i,t-1})^2}{\mathrm{Var}(h_i)}}_{\text{neurones doivent être ``lents''}}$$

Slow, Decorrelated Features for Pretraining Complex Cell-like Networks
(Bergstra and Bengio, NIPS 2009)

- Les neurones sont plus complexes

filtre du neurone — filtres excitateurs — filtres inhibiteurs

$$\frac{\sqrt{\log(1+e^{wx+b})^2 + \sum_{i=1}^{I}(u^{(i)}x)^2} - \sqrt{\sum_{j=1}^{J}(v^{(j)}x)^2}}{1.0 + \sqrt{\log(1+e^{wx+b})^2 + \sum_{i=1}^{I}(u^{(i)}x)^2} + \sqrt{\sum_{j=1}^{J}(v^{(j)}x)^2}}$$

- Inspirés des “complex cells” du cerveau

Slow, Decorrelated Features for Pretraining Complex Cell-like Networks

(Bergstra and Bengio, NIPS 2009)

- Résultats:
  - ⋆ MNIST: sans pré-entraînement = 1.56%
    avec pré-entraînement = 1.34%
  - ⋆ Avec seulement 100 exemples étiquetés

Pre-training Dataset	Number of pretraining iterations ($\times 10^4$)					
	0	1	2	3	4	5
MIXED-movies	23.1	21.2	20.8	20.8	20.6	20.6
MNIST-movies	23.1	19.0	18.7	18.8	18.4	18.6

Quel est le meilleur module non-supervisé?

- Y a-t-il d'autres critères d'apprentissage non-supervisé intéressants?
- Qu'est-ce qui est plus important: le critère non-supervisé ou le type d'encodeur?
- Est-ce utile d'introduire aussi un critère supervisé en pré-entraînement?

! Piste de recherche !

Applications

Automatic Identification of Instrument Classes in Polyphonic and Poly-Instrument Audio

(Philippe Hamel, Sean Wood and Douglas Eck, ISMIR 2009)

- Exemple d’application autre que vision

	SVM	MLP	DBN
Spectral Features (16)	0.51	0.74	0.81
12 MFCCs (72)	0.75	0.85	0.85
20 MFCCs (120)	0.81	0.86	0.87
All Features (136)	0.84	0.84	0.88

Table 1. Global F-score for different features subsets (features vector length in parenthesis)

Automatic Identification of Instrument Classes in Polyphonic and Poly-Instrument Audio

(Philippe Hamel, Sean Wood and Douglas Eck, ISMIR 2009)

	SVM	MLP	DBN	%
Bass	0.88	0.88	0.88	13.85%
Brass	0.87	0.88	0.91	22.37%
Guitar	0.0	0.0	**0.21**	2.13%
Organ	0.96	0.89	0.96	7.46%
Piano	0.45	0.43	**0.57**	6.39%
Strings	0.94	0.95	0.97	9.59%
Woodwind	0.82	0.85	0.89	29.83%
Global	0.84	0.84	0.88	

Table 2. F-score for solo instrument audio. The results that clearly outperforms the other models are highlighted in bold. The percentage of positive examples in the training set for each instrument is shown in the rightmost column

	SVM	MLP	DBN	%
Bass	0.86	0.83	0.85	50.00%
Brass	0.38	0.45	**0.63**	25.90%
Guitar	0.05	0.15	**0.28**	11.94%
Organ	0.84	0.84	0.85	62.99%
Piano	0.83	0.80	0.83	64.44%
Strings	0.37	0.37	0.36	18.82%
Woodwind	0.31	0.41	**0.52**	31.81%
Global	0.72	0.72	0.74	

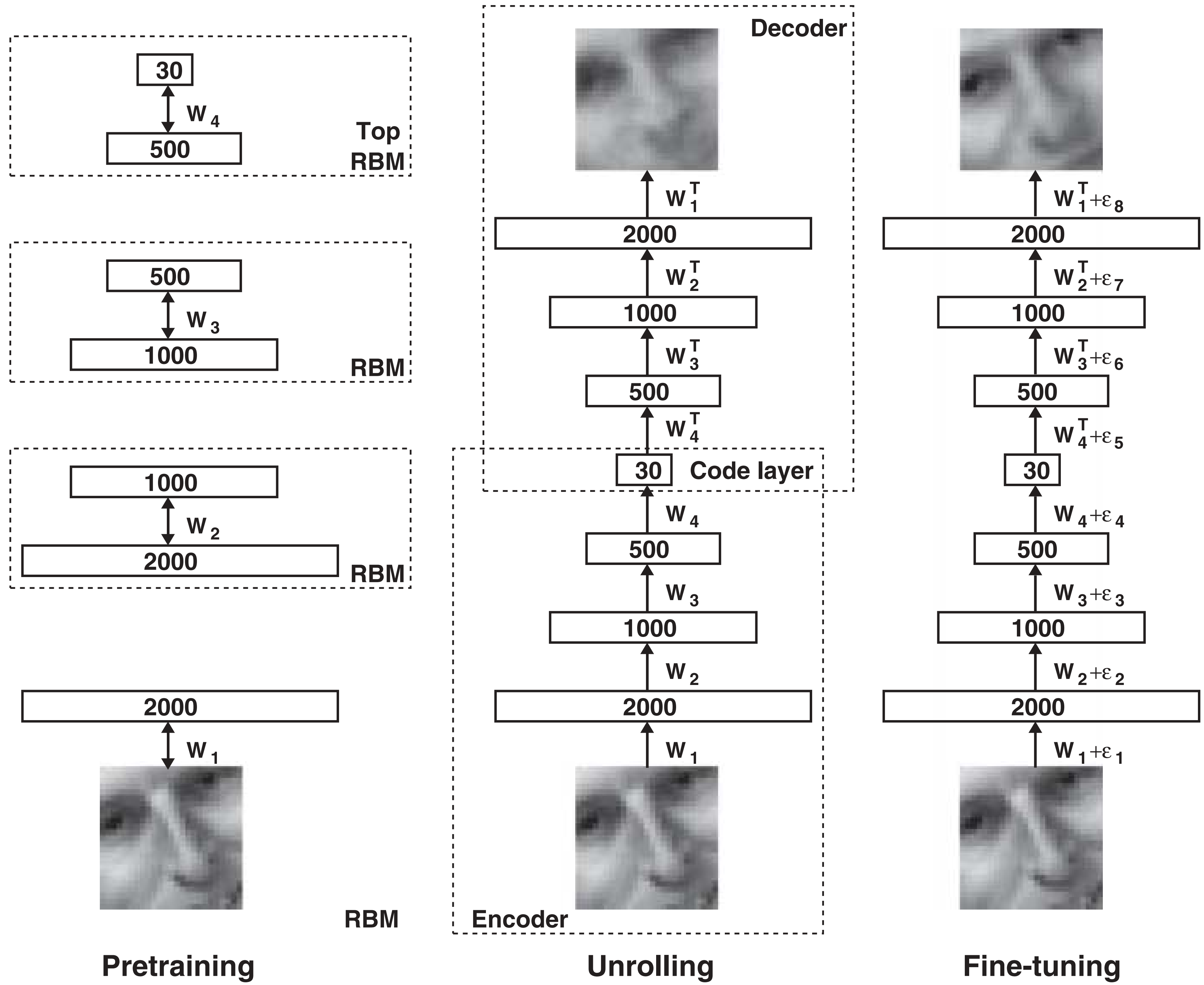
Table 3. F-score for poly-instrument audio. The results that clearly outperforms the other models are highlighted in bold. The percentage of positive examples in the training set for each instrument is shown in the rightmost column

Reducing the Dimensionality of Data with Neural Networks

(Hinton and Salakhutdinov, Science 2006)

- Application:

  réduction de dimensionalité



Reducing the Dimensionality of Data with Neural Networks

(Hinton and Salakhutdinov, Science 2006)

Résultats: reconstruction

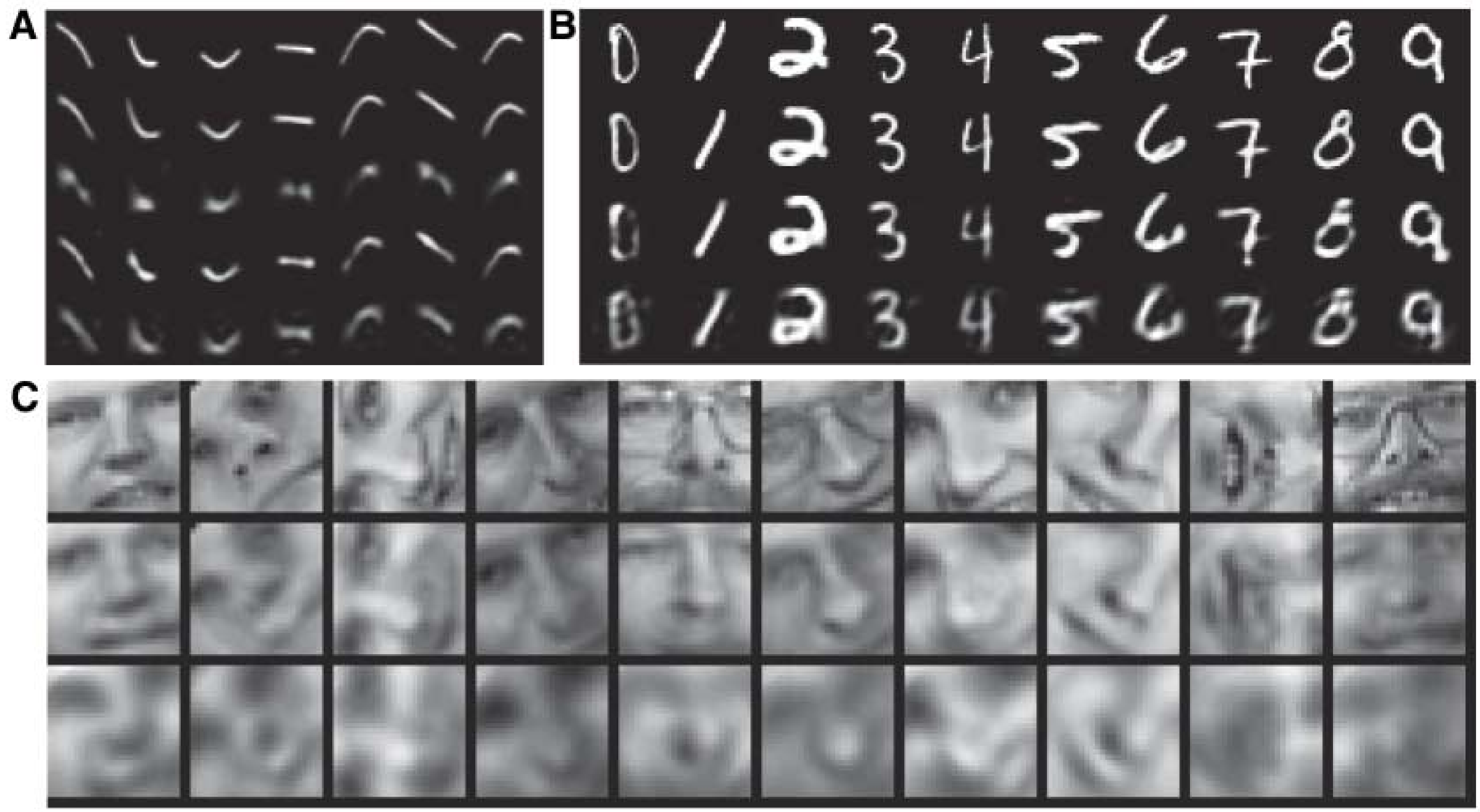


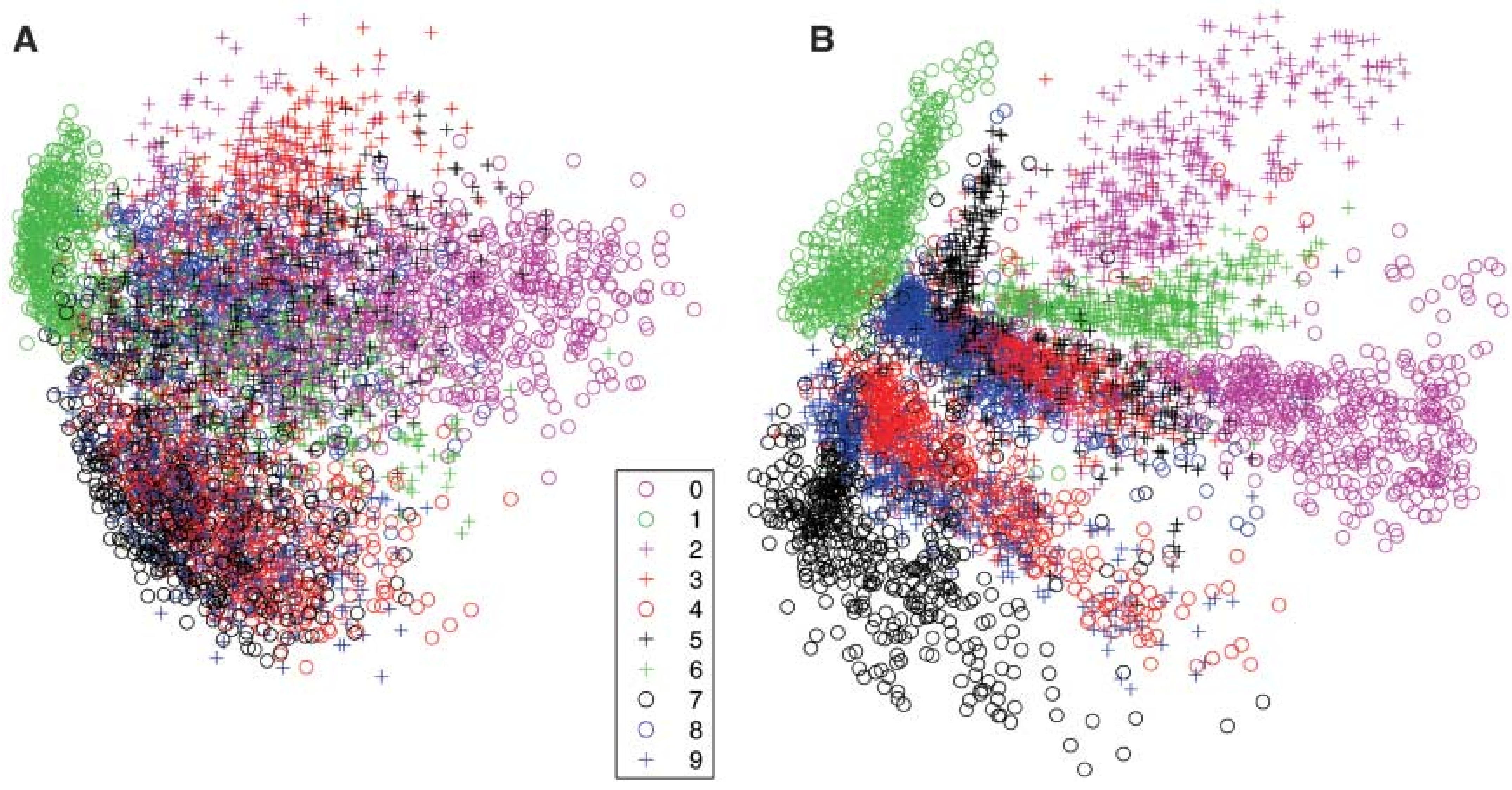
Fig. 2. (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error per image for the last four rows is 1.44, 7.64, 2.45, 5.90. (**B**) Top to bottom: A random test image from each class; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional logistic PCA and standard PCA. The average squared errors for the last three rows are 3.00, 8.01, and 13.87. (**C**) Top to bottom: Random samples from the test data set; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional PCA. The average squared errors are 126 and 135.

Reducing the Dimensionality of Data with Neural Networks

(Hinton and Salakhutdinov, Science 2006)

Résultats: visualisation

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

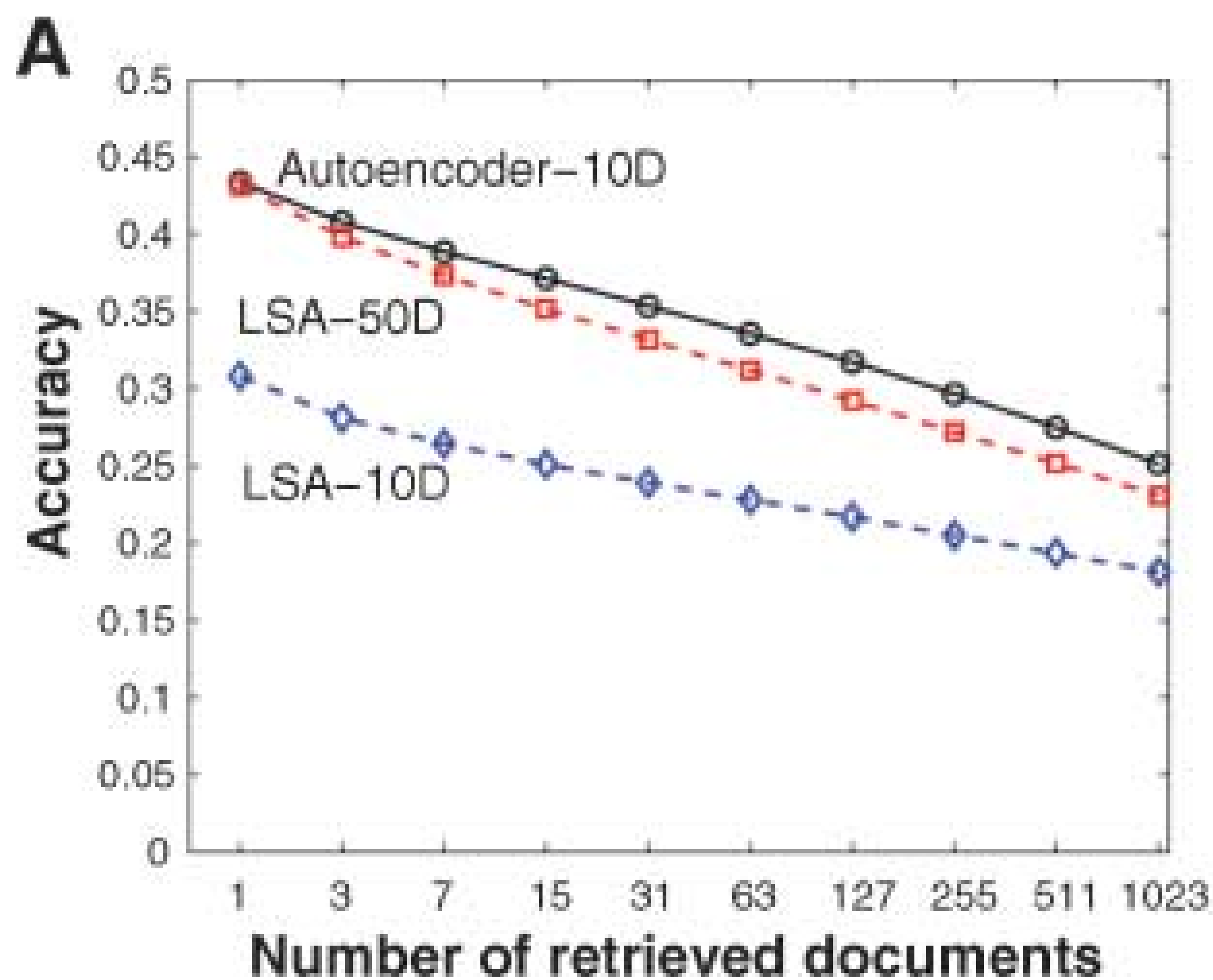


Reducing the Dimensionality of Data with Neural Networks

(Hinton and Salakhutdinov, Science 2006)

Résultats: recherche d'information

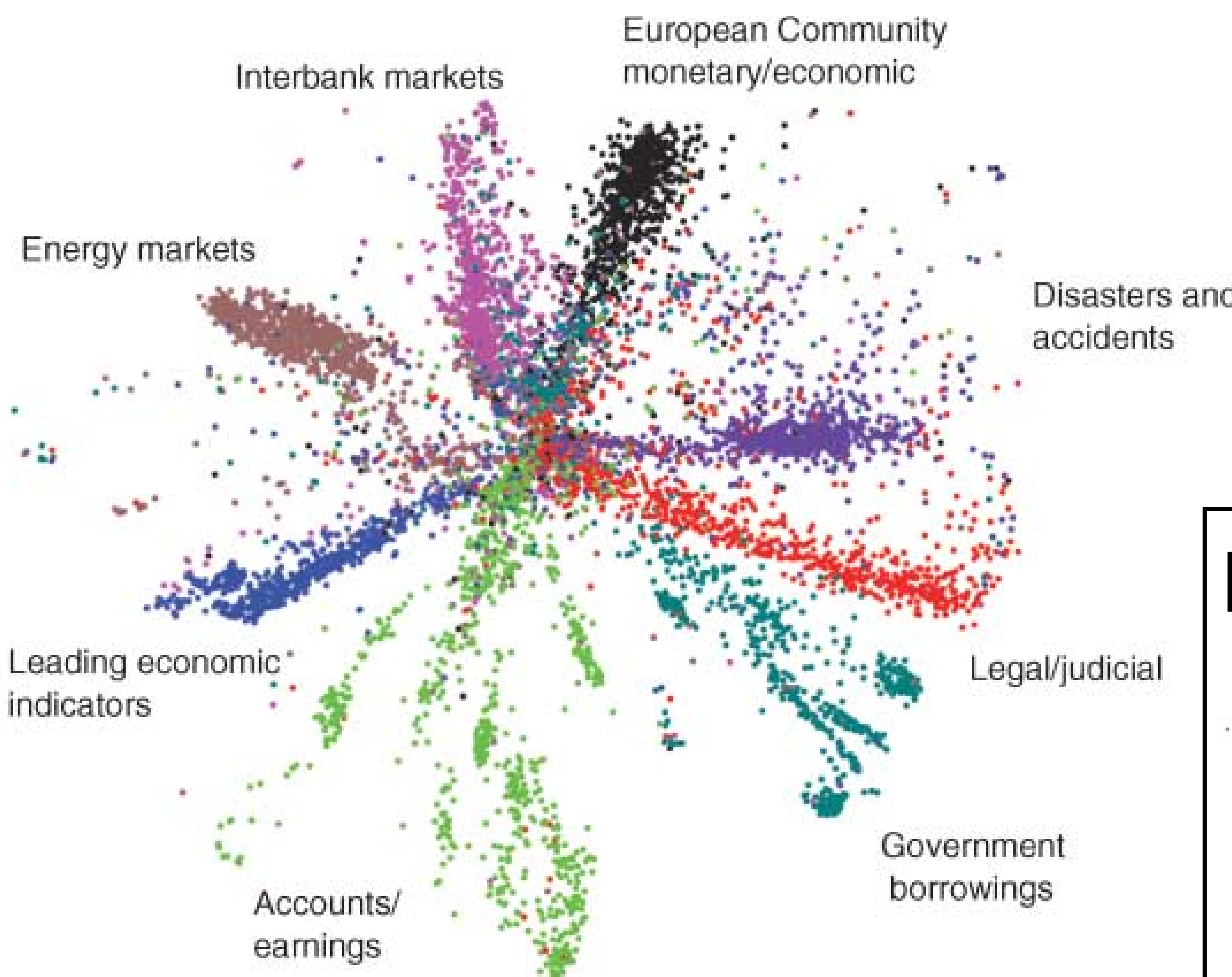
Fig. 4. (**A**) The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries.

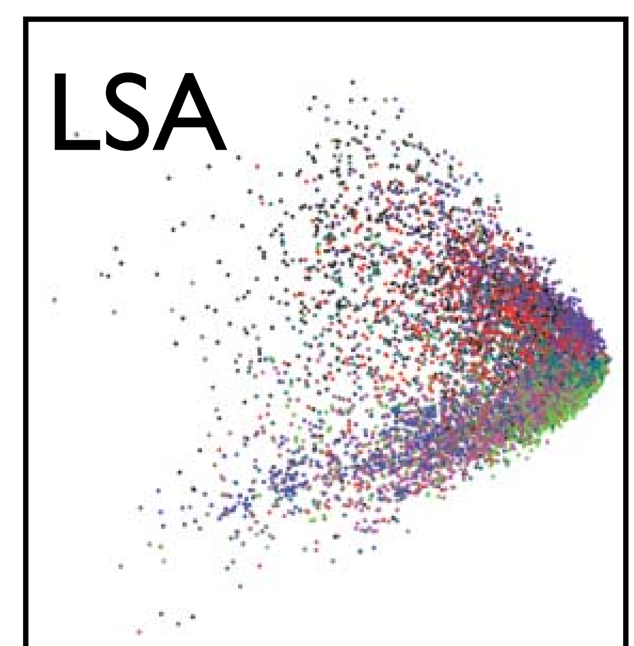


Reducing the Dimensionality of Data with Neural Networks

(Hinton and Salakhutdinov, Science 2006)

Résultats: visualisation





Semantic Hashing
(Salakhutdinov and Hinton, IJAR 2009)

- Application: recherche d'information

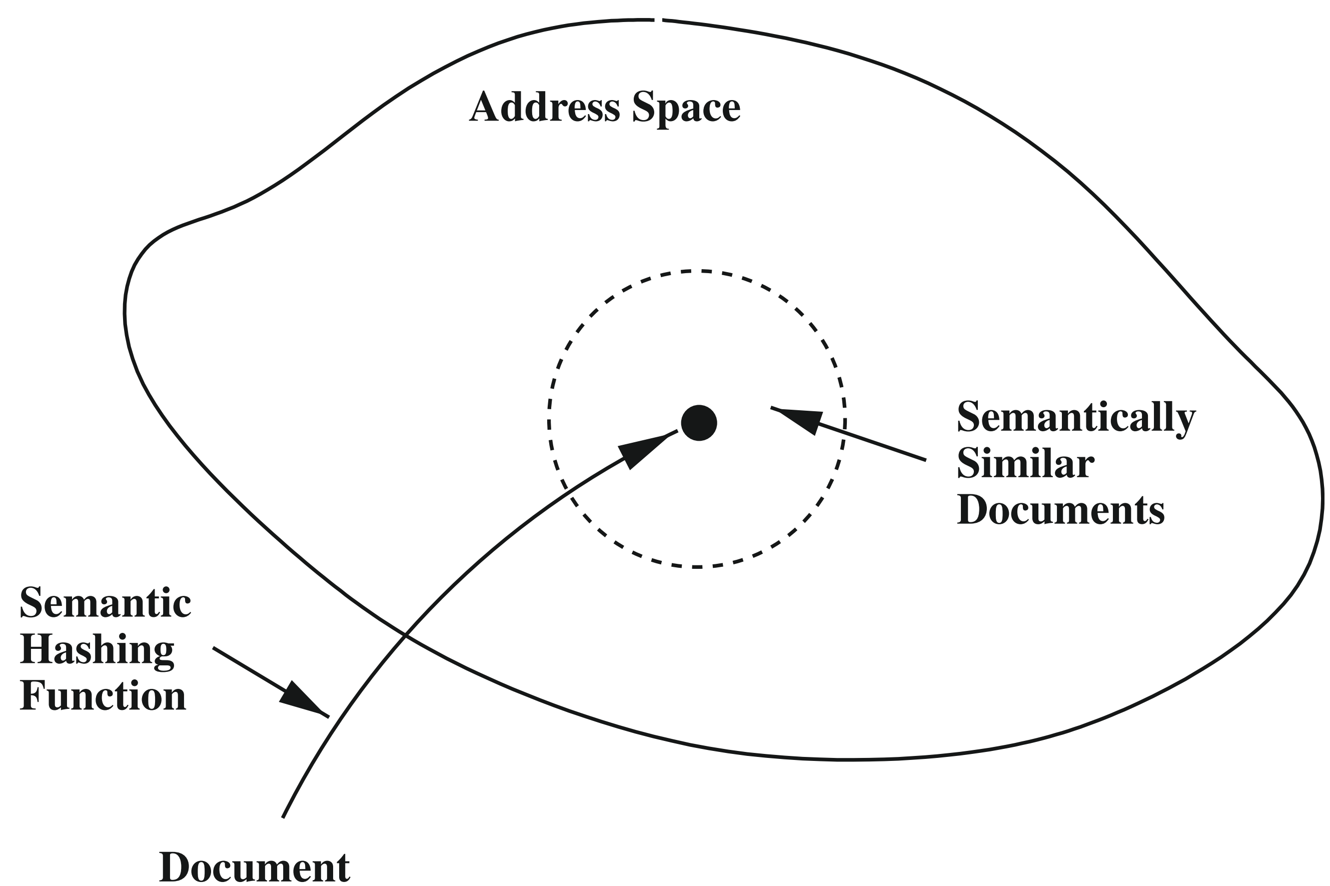
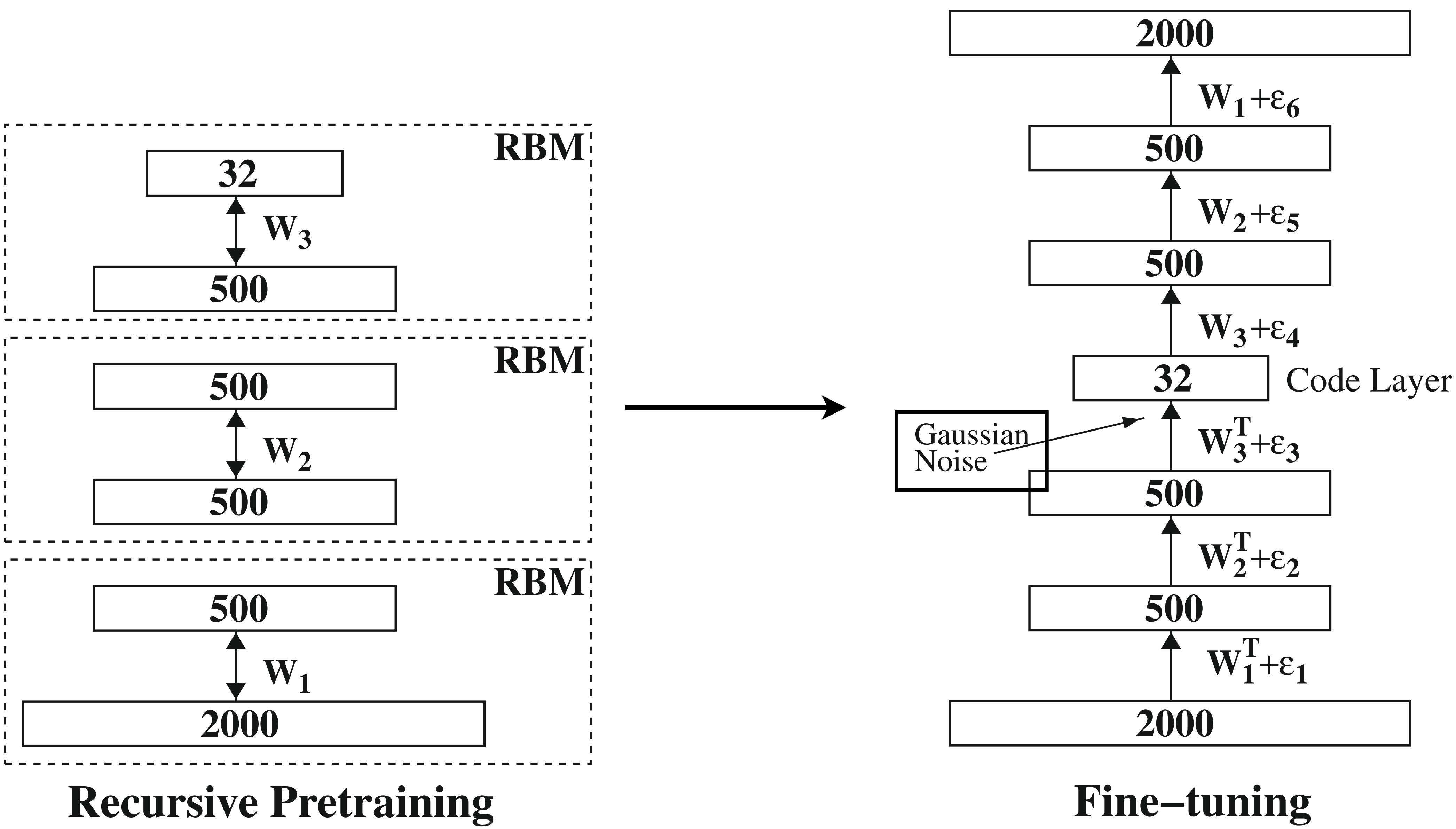


Fig. 1. A schematic representation of semantic hashing.

Semantic Hashing

(Salakhutdinov and Hinton, IJAR 2009)

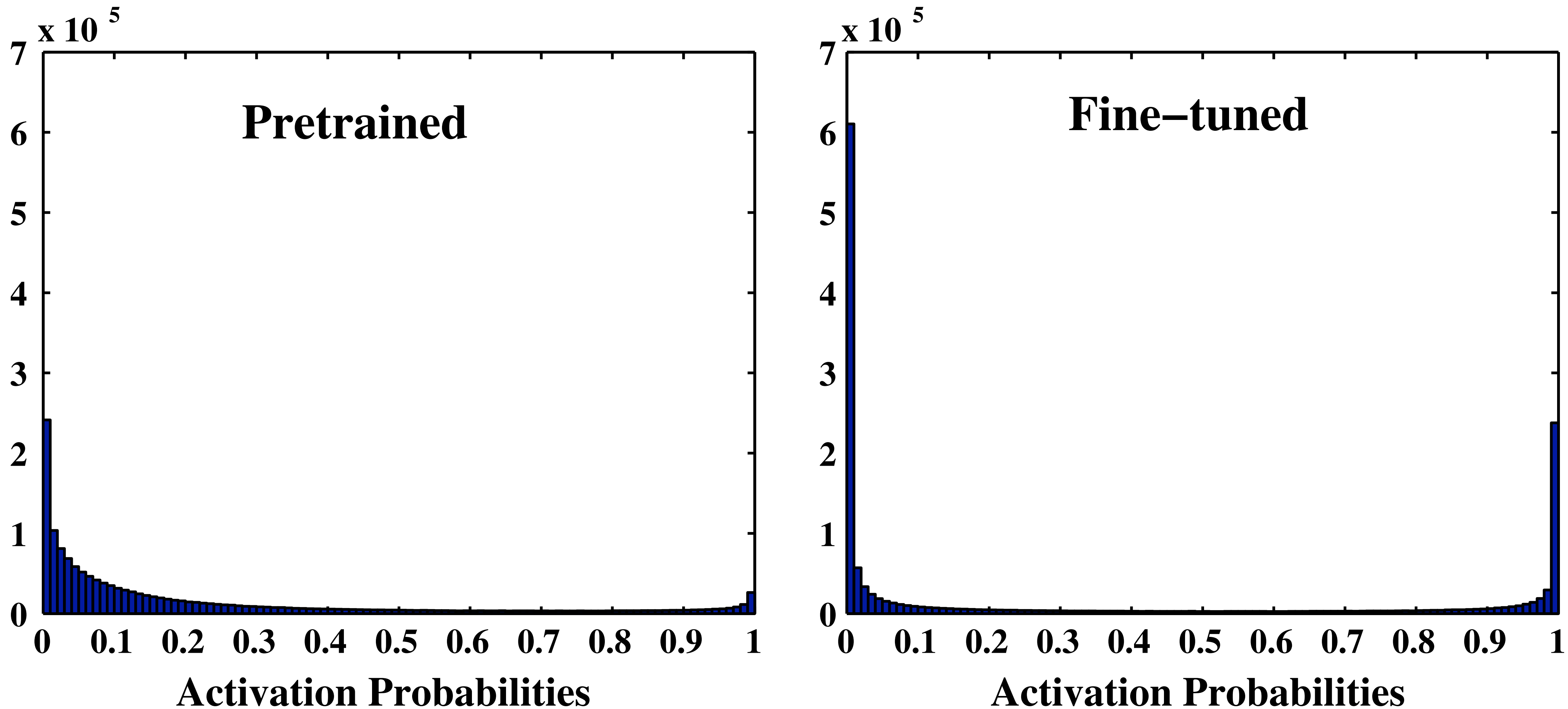
- Application: recherche d'information



Semantic Hashing

(Salakhutdinov and Hinton, IJAR 2009)

- Application: recherche d'information



Semantic Hashing

(Salakhutdinov and Hinton, IJAR 2009)

- "Constrained Poisson Model"

$$p(v_i = n|\mathbf{h}) = \mathrm{Ps}\left(n, \frac{\exp(\lambda_i + \sum_j h_j w_{ij})}{\sum_k \exp\left(\lambda_k + \sum_j h_j w_{kj}\right)} N\right)$$

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i w_{ij} v_i\right),$$

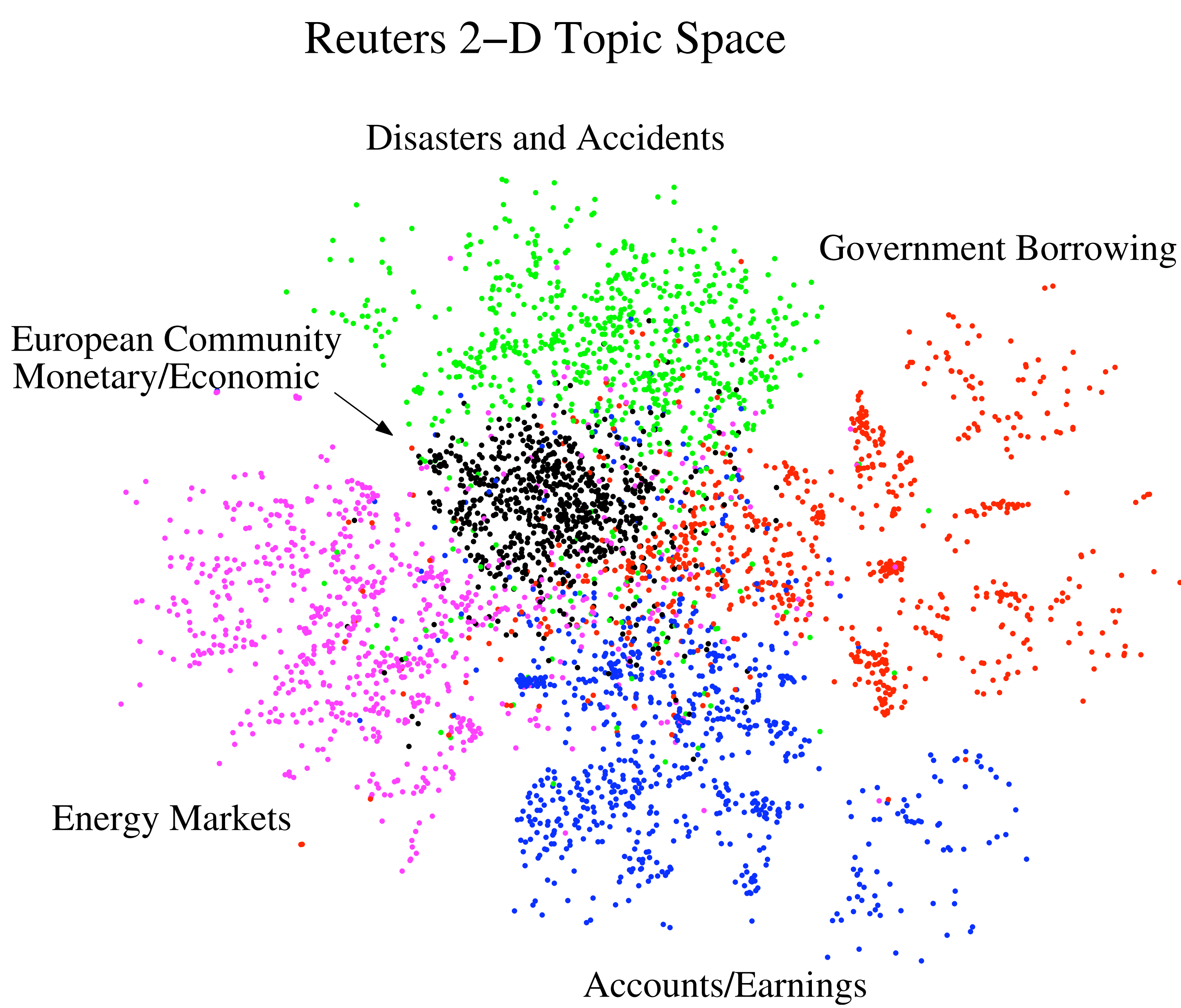
- Pas très élégant... Voir plutôt:

Replicated Softmax: an Undirected Topic Model, NIPS 2009

Semantic Hashing

(Salakhutdinov and Hinton, IJAR 2009)

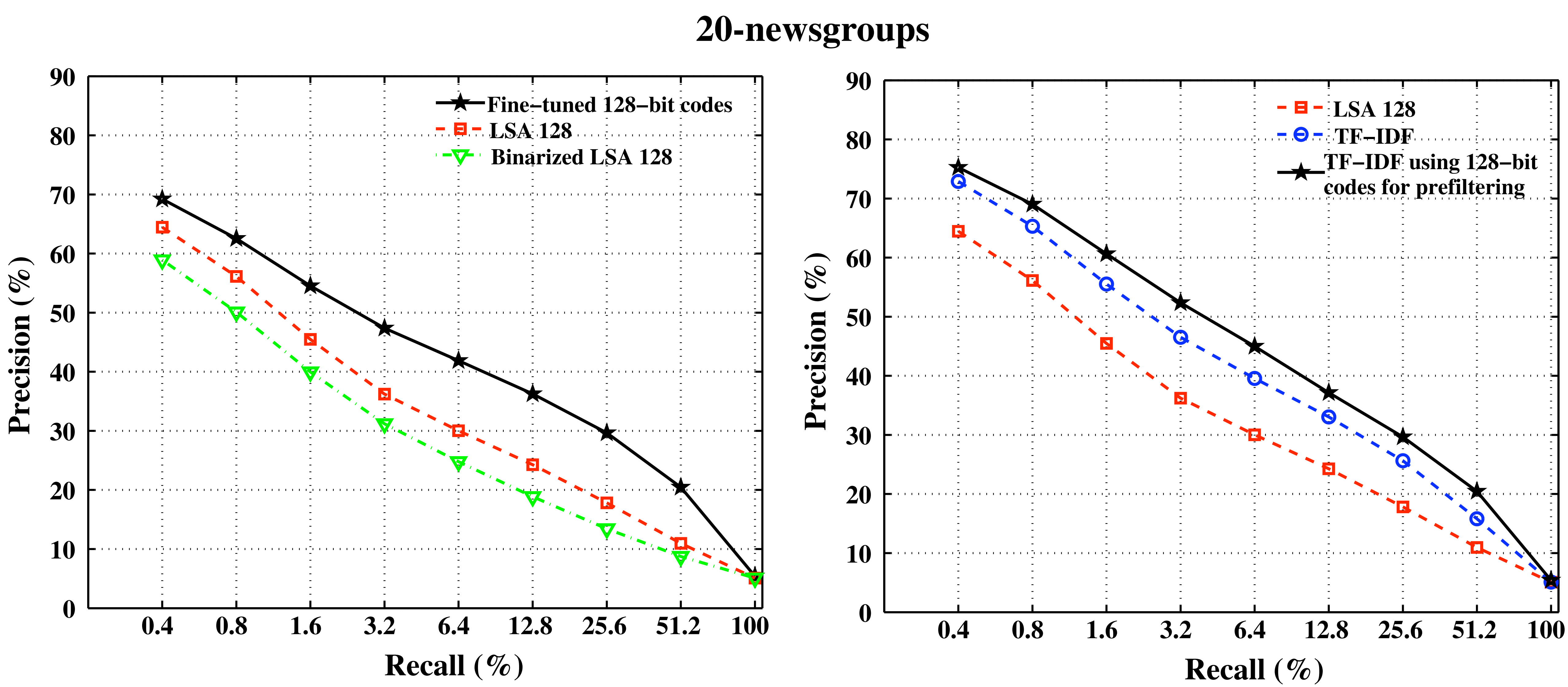
Résultats: visualisation



Semantic Hashing

(Salakhutdinov and Hinton, IJAR 2009)

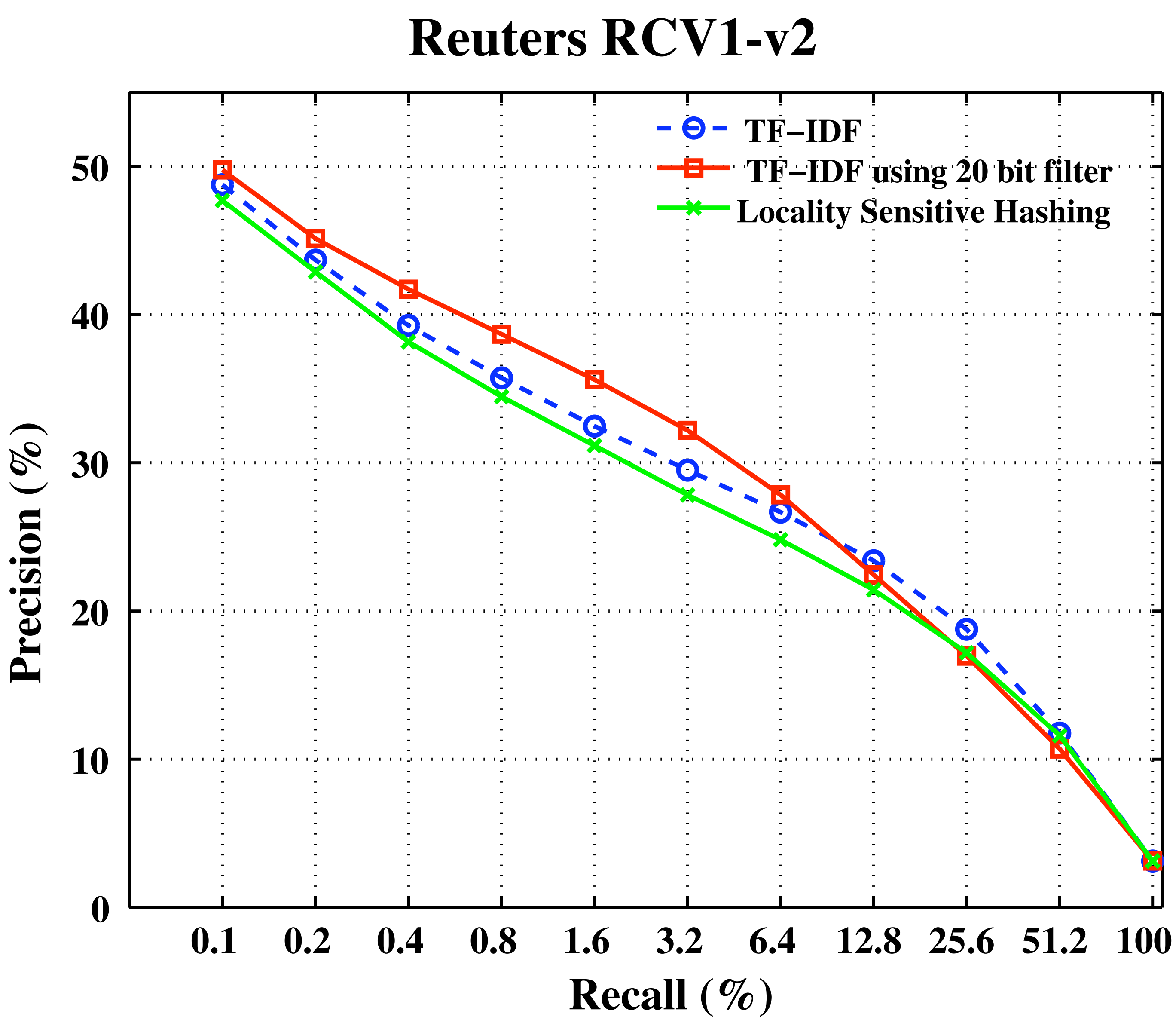
Résultats: recherche d'information



Semantic Hashing

(Salakhutdinov and Hinton, IJAR 2009)

Résultats: recherche d'information



Avec 20 bits +
"Hamming ball" de rayon 4

- 0.5 ms pour filtrer documents (~2500 / 402 207)

- 10 ms pour traiter ces documents avec tf-idf

Semantic Hashing

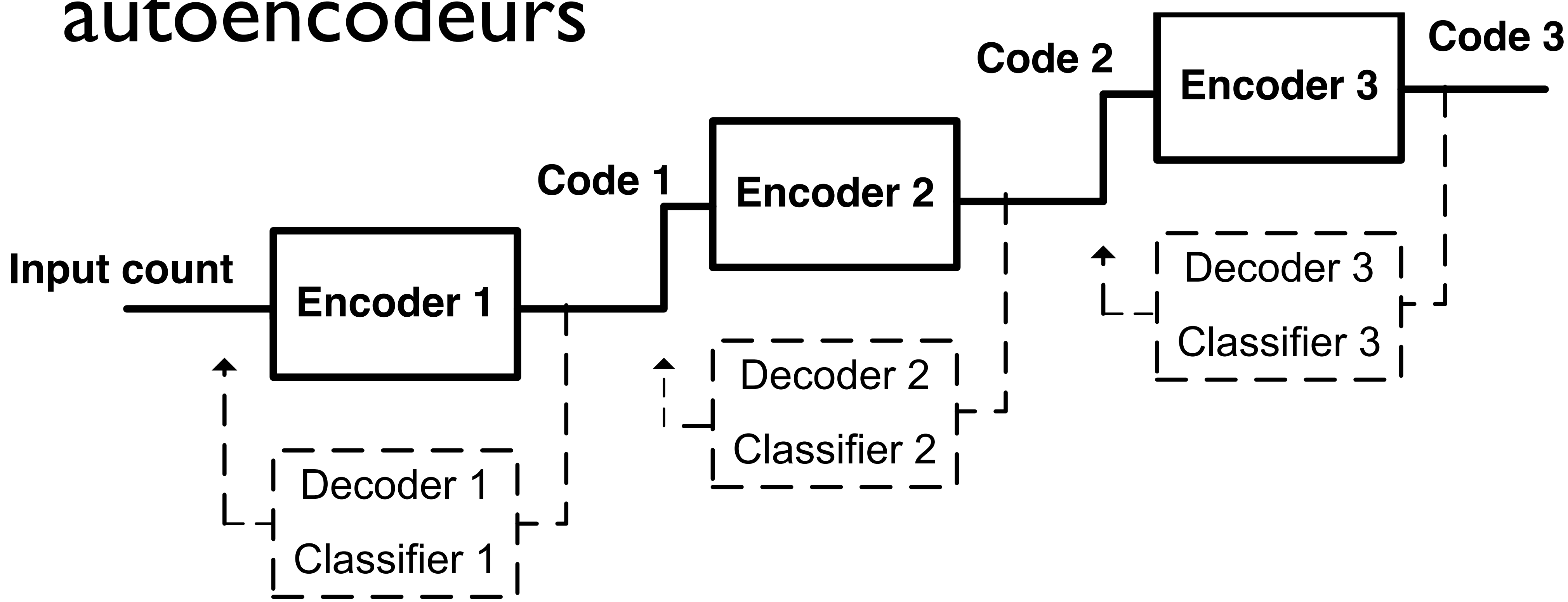
(Salakhutdinov and Hinton, IJAR 2009)

- Trucs pour la modélisation de texte
  - ★ enlever les mots communs
  - ★ enlever les informations de genre et de nombre
  - ★ porter attention à l'impact du nombre de mot de chaque document

Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

- Même application, mais en utilisant des autoencodeurs

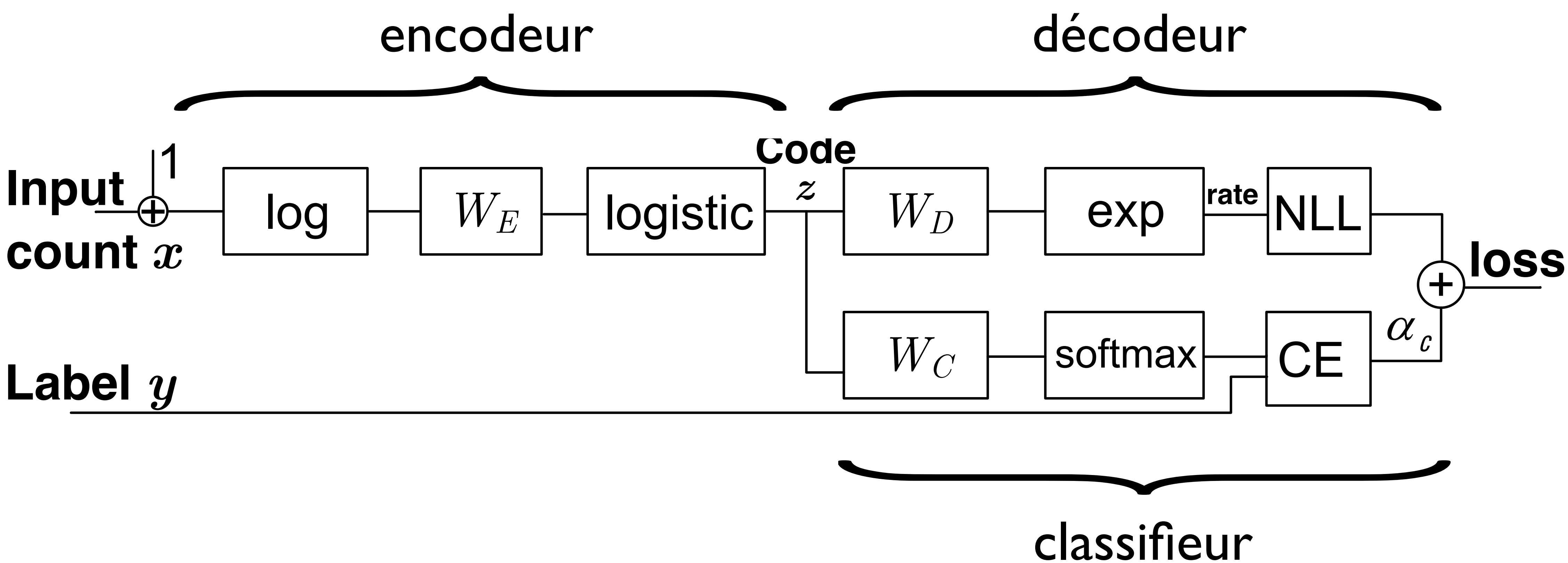


- Innovations:
  - ★ entraînement partiellement supervisé
  - ★ autoencodeur adapté aux documents

Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

- Autoencodeur (en images)



- Coût d'entraînement:

$$L = E_R + \alpha_c E_C$$

Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

- Autoencodeur (en équations)

log() à cause du exp() en sortie

encodeur $$\boldsymbol{z} = \boldsymbol{\sigma}(W_E \log(\boldsymbol{x}) + \boldsymbol{b}_E)$$

décodeur $$\lambda = \beta \mathrm{e}^{W_D \boldsymbol{z} + \boldsymbol{b}_D}$$

distribution Poisson $$P(\boldsymbol{x}) = \prod_i P(x_i) = \prod_i \mathrm{e}^{-\lambda_i} \frac{\lambda_i^{x_i}}{x_i!}$$

NLL Poisson $$E_R = \sum_i (\beta \mathrm{e}^{((W_D)_i \cdot \boldsymbol{z} + b_{Di})} - x_i (W_D)_i \cdot \boldsymbol{z} - x_i b_{Di} + \log x_i!)$$

Semi-supervised Learning of Compact Document Representations with Deep Networks

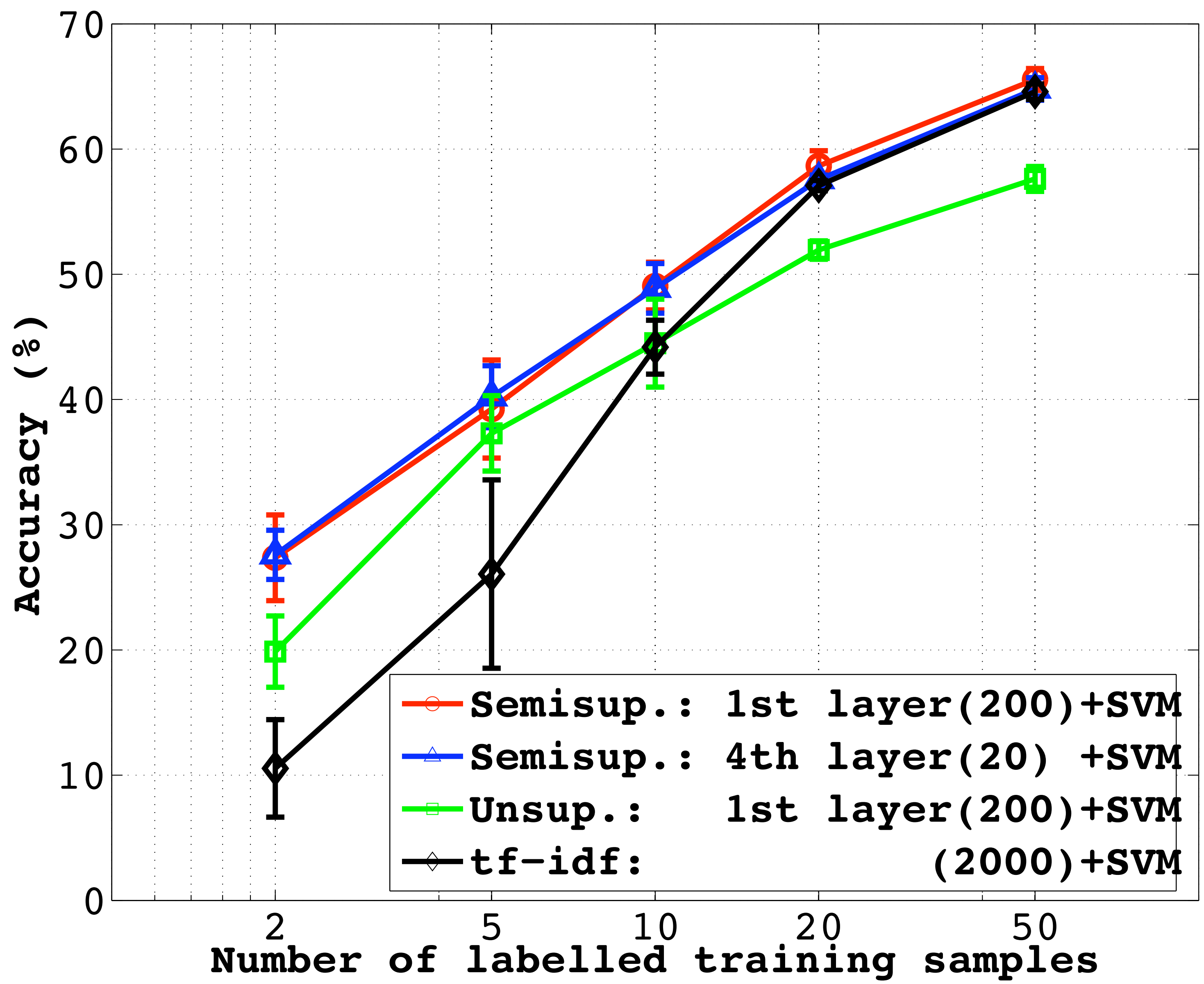
(Ranzato and Szummer, ICML 2008)

- Pour éviter log(0), on ajoute 1 à la fréquence de tous les mots d'un document
- Les autoencodeurs des couches subséquentes utilisent le coût de l'erreur au carré pour la reconstruction (sans log(x))

Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

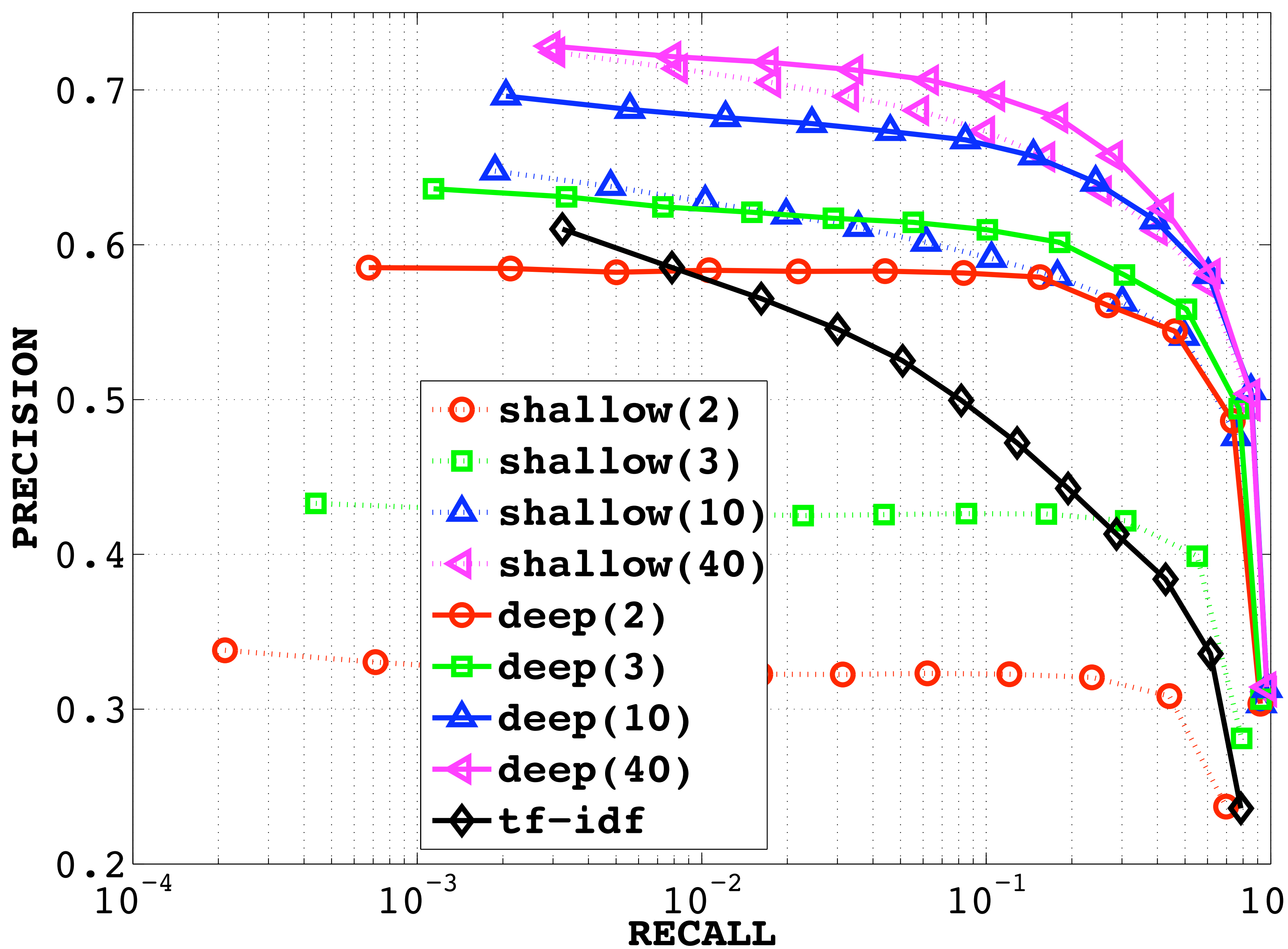
Résultats: classification (20 newsgroup)



Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

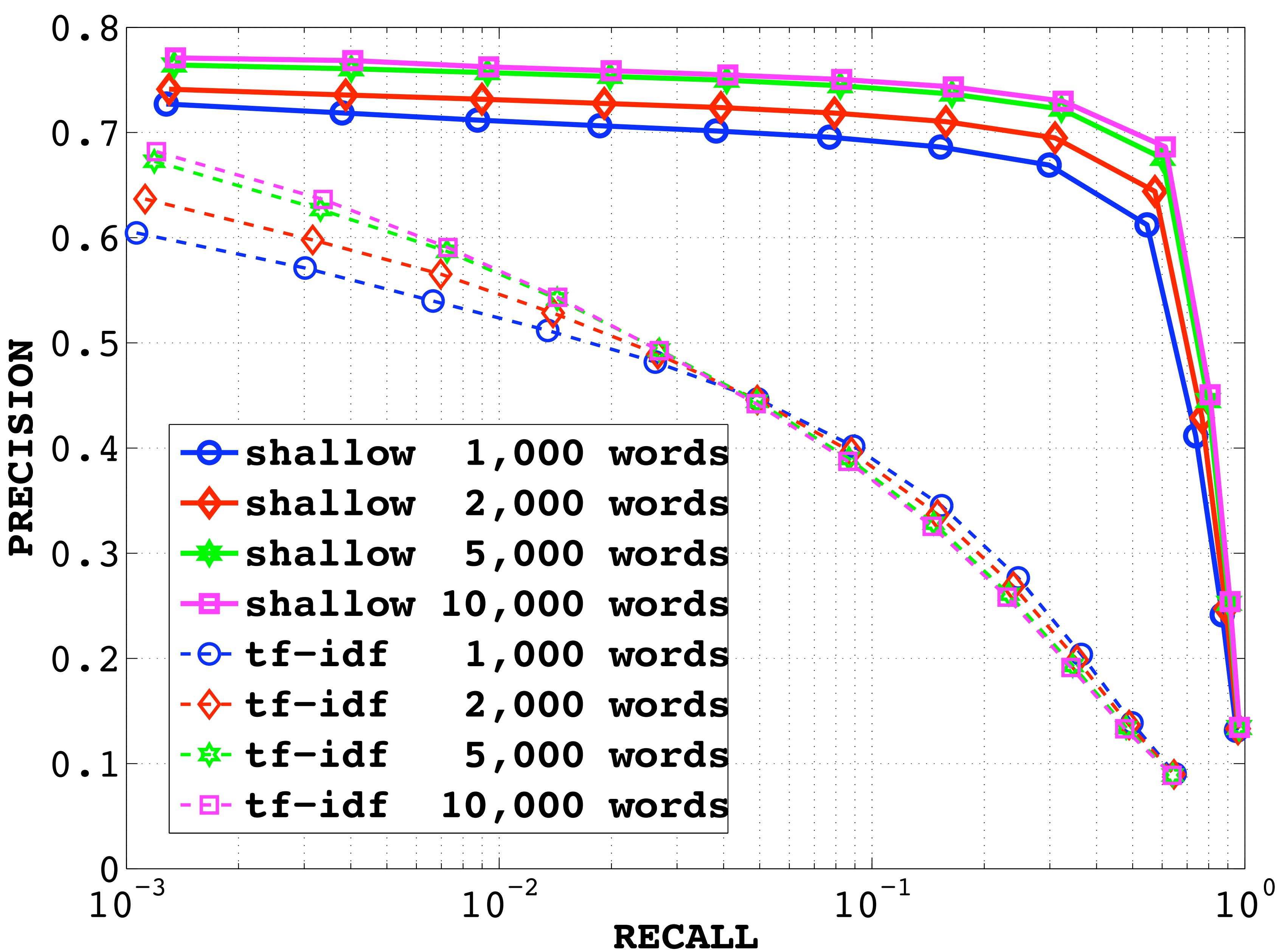
Résultats: recherche d'information (profond ou pas)



Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

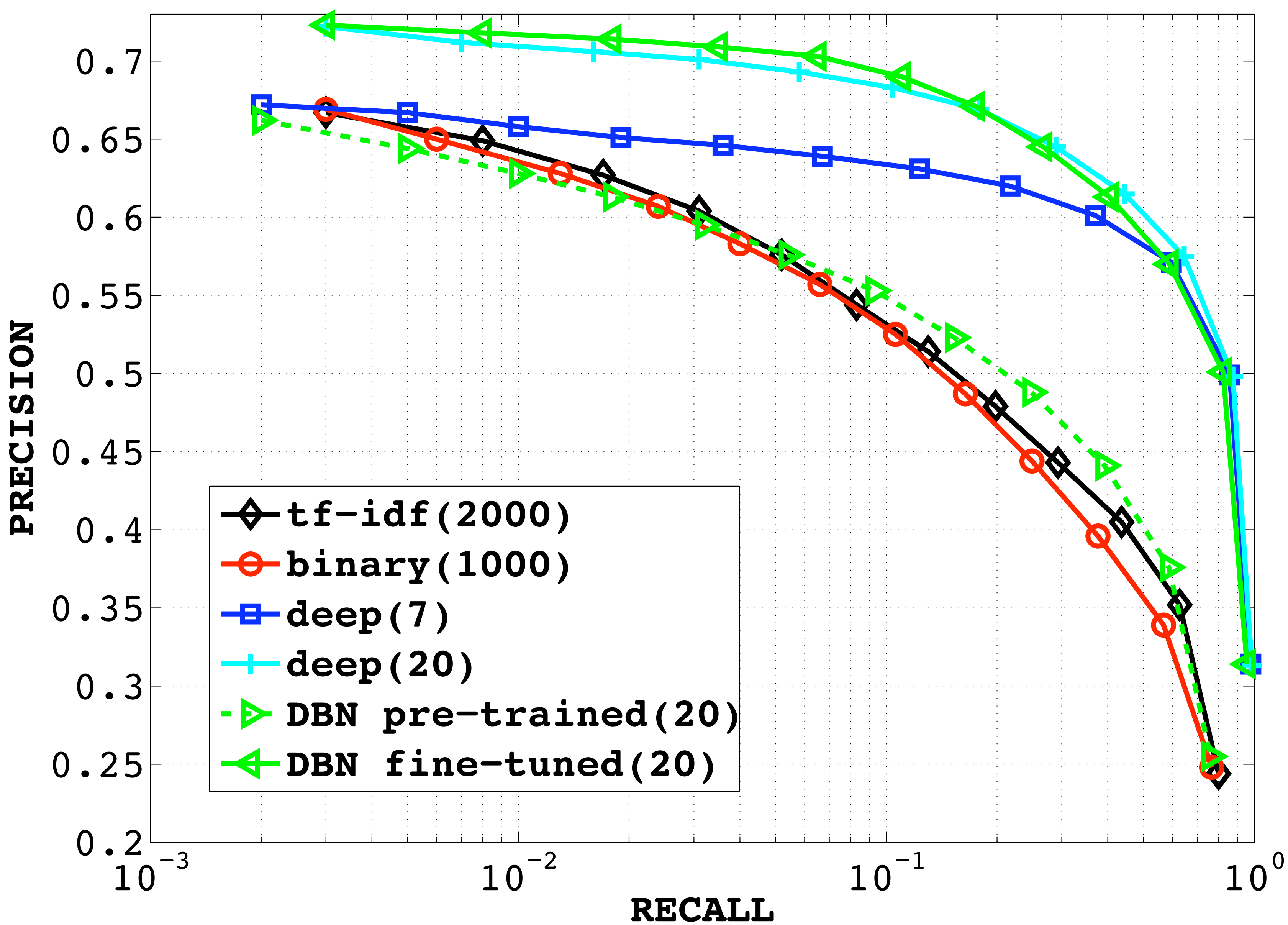
Résultats: recherche d'information (taille vocab.)



Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

Résultats: recherche d'information (autoencoder vs RBM)



Semi-supervised Learning of Compact Document Representations with Deep Networks

(Ranzato and Szummer, ICML 2008)

Résultats: similarité entre les mots

Word stem	**Neighboring word stems**
livestock	beef, meat, pork, cattle
lend	rate, debt, bond, downgrad
acquisit	merger, stake, takeov
port	ship, port, vessel, freight
branch	stake, merger, takeov, acquisit
plantat	coffe, cocoa, rubber, palm
barrel	oil, crude, opec, refineri
subcommitte	bill, trade, bond, committe
coconut	soybean, wheat, corn, grain
meat	beef, pork, cattl, hog
ghana	cocoa, buffer, coffe, icco
varieti	wheat, grain, agricultur, crop
warship	ship, freight, vessel, tanker
edibl	beef, pork, meat, poultri

“to be continued...”