Réseaux de neurones

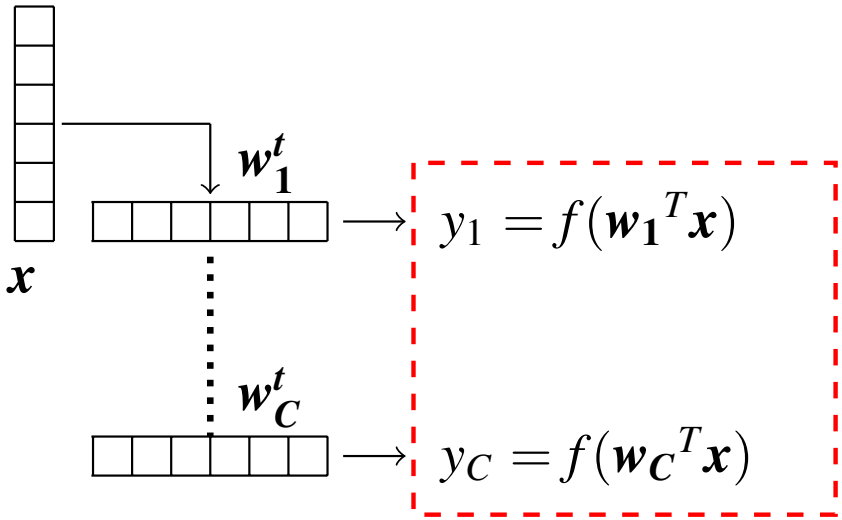
Alexandre Allauzen

Université Paris-Sud / LIMSI-CNRS

Plan

Classification multi-classe

Classification binaire *vs* C classes (Maxent)



$$f(a_j = w_j{}^T x) = \frac{e^{a_j}}{\sum_{j'=1}^{K} e^{a_{j'}}} = \frac{e^{a_j}}{Z(x)}$$

Réseau de neurones associé

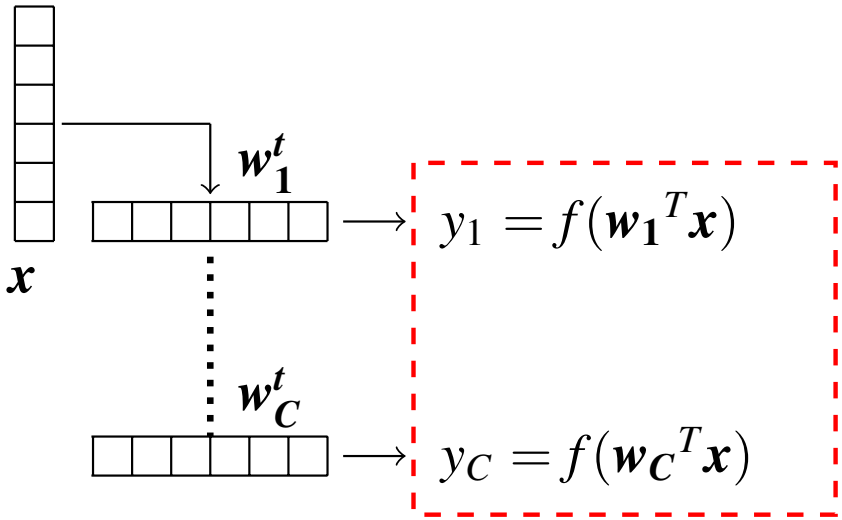
$y_1 = f(w_1^t x)$

$y_C = f(w_C^t x)$

- x: *input layer*
- y: *output layer*
- chaque y_j a ses paramètres $(w_j) \rightarrow W$
- f : softmax

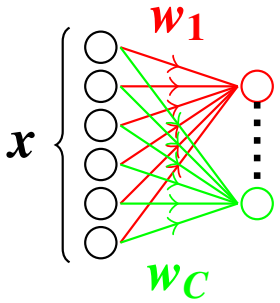
Classification multi-classe

Classification binaire *vs* C classes (Maxent)



$$f(a_j = w_j{}^T x) = \frac{e^{a_j}}{\sum_{j'=1}^{K} e^{a_{j'}}} = \frac{e^{a_j}}{Z(x)}$$

Réseau de neurones associé

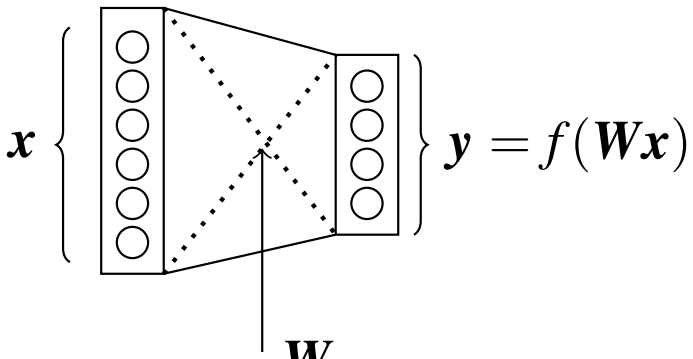


$$y_1 = f(w_1^t x)$$

$$y_C = f(w_C^t x)$$

- x: *input layer*
- y: *output layer*
- chaque y_j a ses paramètres $(w_j) \rightarrow W$
- f : softmax

Deux couches neuronales

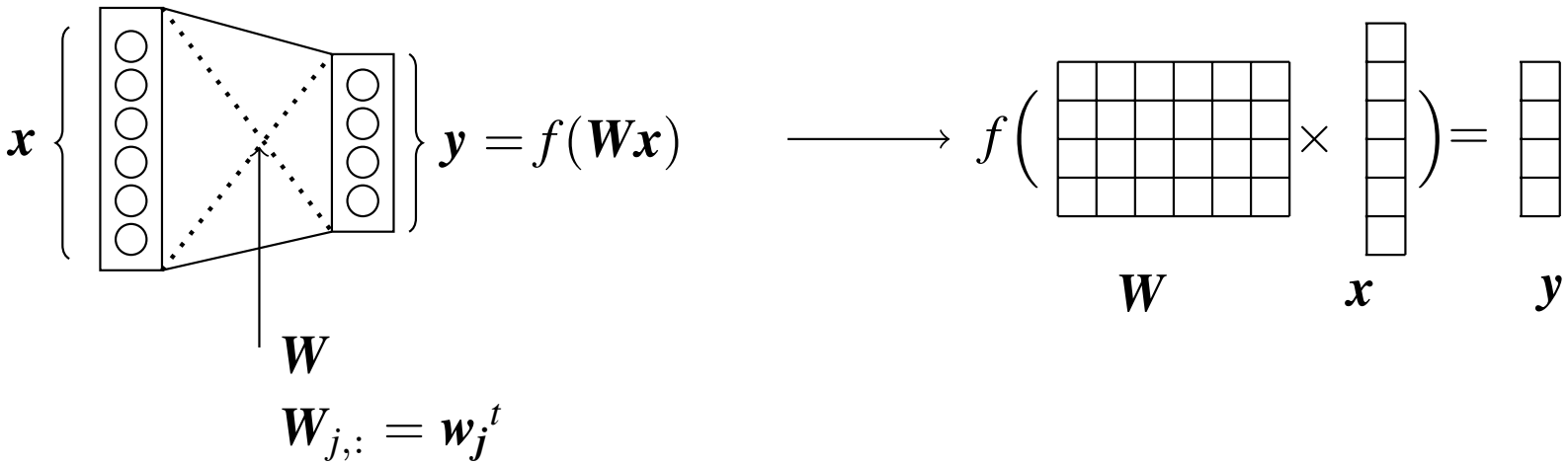


- f est souvent non-linéaire
- f s'applique composante par composante
- *e.g* le softmax:

$$y_j = P(c = j|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_j{}^T \boldsymbol{x}}}{\sum_{j'} e^{\boldsymbol{w}_{j'}{}^T \boldsymbol{x}}} = \frac{e^{\boldsymbol{W}_{j,:}\boldsymbol{x}}}{\sum_{j'} e^{\boldsymbol{W}_{j',:}\boldsymbol{x}}}$$

- tanh, sigmoid, relu, ...

Deux couches neuronales

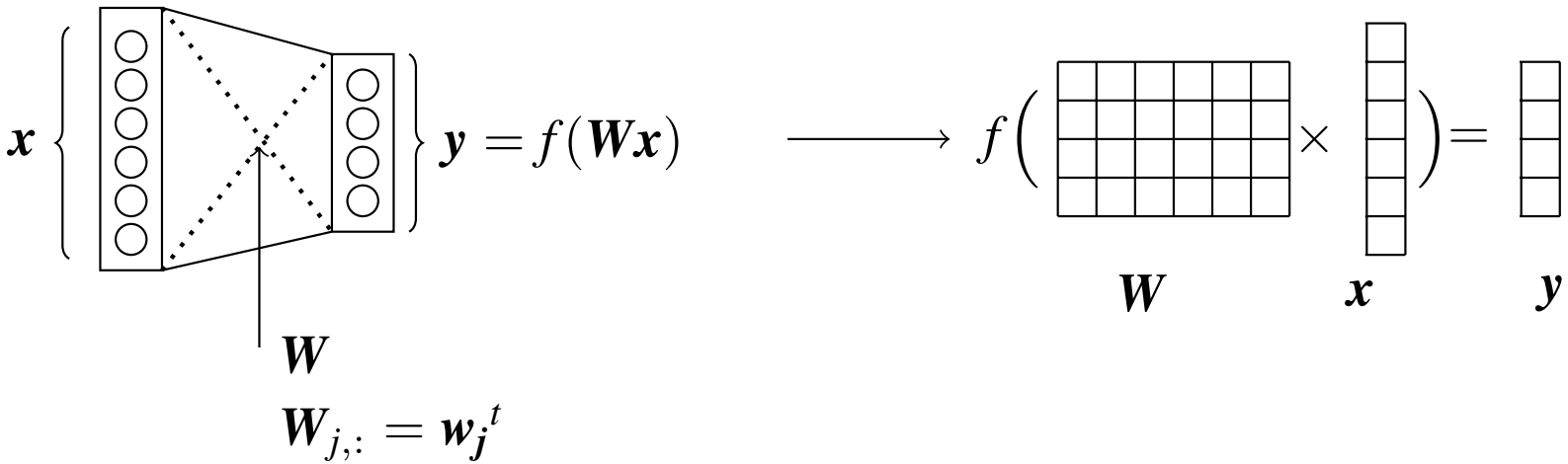


- f est souvent non-linéaire
- f s'applique composante par composante
- *e.g* le softmax:

$$y_j = P(c = j|\boldsymbol{x}) = \frac{e^{\boldsymbol{w_j}^T \boldsymbol{x}}}{\sum_{j'} e^{\boldsymbol{w_{j'}}^T \boldsymbol{x}}} = \frac{e^{\boldsymbol{W}_{j,:}\boldsymbol{x}}}{\sum_{j'} e^{\boldsymbol{W}_{j',:}\boldsymbol{x}}}$$

- tanh, sigmoid, relu, ...

Deux couches neuronales

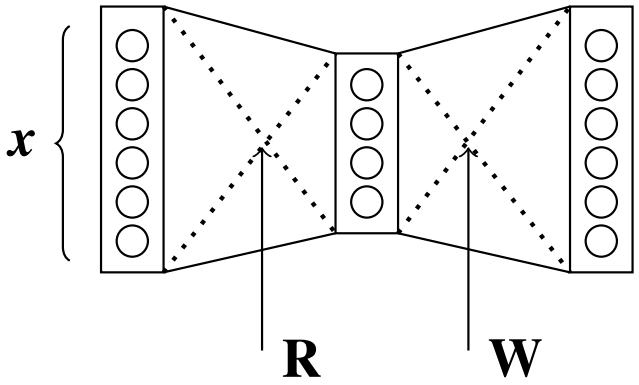


- f est souvent non-linéaire
- f s'applique composante par composante
- *e.g* le softmax:

$$y_j = P(c = j|\boldsymbol{x}) = \frac{e^{\boldsymbol{w_j}^T \boldsymbol{x}}}{\sum_{j'} e^{\boldsymbol{w_{j'}}^T \boldsymbol{x}}} = \frac{e^{\boldsymbol{W}_{j,:}\boldsymbol{x}}}{\sum_{j'} e^{\boldsymbol{W}_{j',:}\boldsymbol{x}}}$$

- tanh, sigmoid, relu, ...

Classification de document



$$P(y = j|\boldsymbol{x};\boldsymbol{\theta})$$

- $\boldsymbol{x}$: $(|\mathcal{V}|, 1)$
- $\mathbf{R}$: $(K, |\mathcal{V}|)$
- $\mathbf{d}$: $(K, 1)$
- $\mathbf{W}$: (C, K)
- $\boldsymbol{y}$: $(C, 1)$

$$\mathbf{d} = \mathbf{R} \times \boldsymbol{x}$$

$$\boldsymbol{y} = softmax(\mathbf{W} \times \mathbf{d})$$

Fonction objectif et Complexité

La fonction softmax est “coûteuse” à calculer:

- La multiplication matricielle de sortie est en $(C \times K)$,
- puis la normalisation (somme et division sur C éléments)

Log-loss (conditional log-likelihood, cross-entropy)

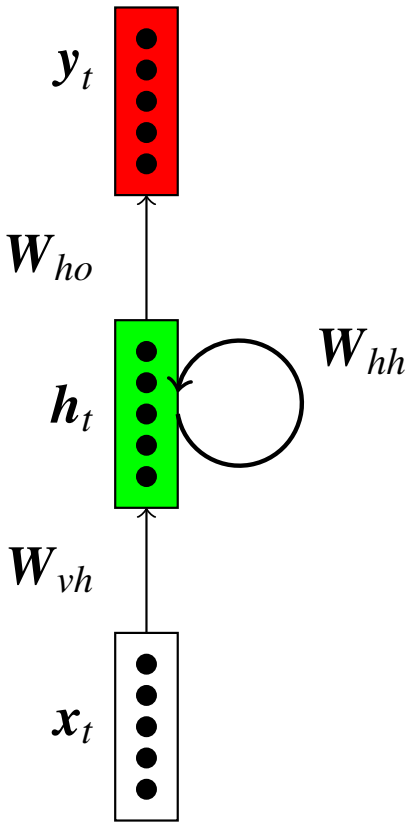
Les données : $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^{N}$, $c_{(i)} \in \{1, 2, \ldots, C\}$

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) = \sum_{i=1}^{N} \Big(- \sum_{c=1}^{C} \mathbb{I}\{c = c_{(i)}\} \log(P(c|\boldsymbol{x}_{(i)})) \Big) \quad (1)$$

$$l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) = - \sum_{k=1}^{C} \mathbb{I}\{k = c_{(i)}\} \log(y_k) \quad (2)$$

Plan

Réseau récurrent

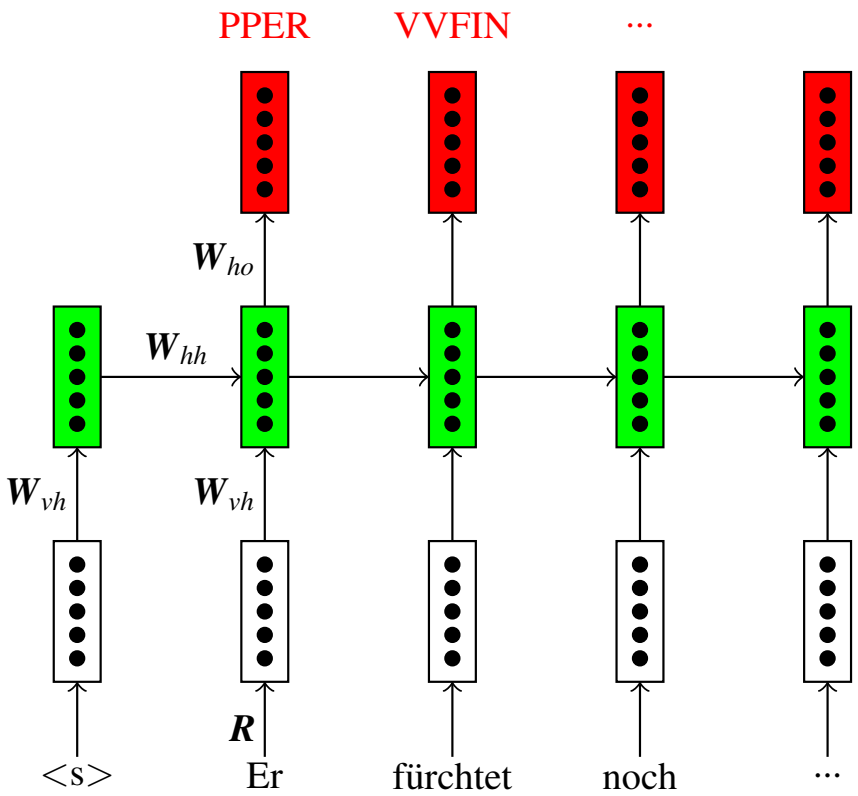


Un modèle de séquence.
Pour chaque instant t:

- maintient une représentation interne (état caché) de l'historique: $\boldsymbol{h}_t$
- Mise à jour avec une observation $\boldsymbol{x}_t$ et l'état précédent $\boldsymbol{h}_{t-1}$
- La prédiction $\boldsymbol{y}_t$ dépend de l'état caché ($\boldsymbol{h}_t$)
- $\boldsymbol{x}_t$ vient des embeddings.

Les mêmes paramètres sont partagés au cours du temps

Un modèle déplié dans le temps



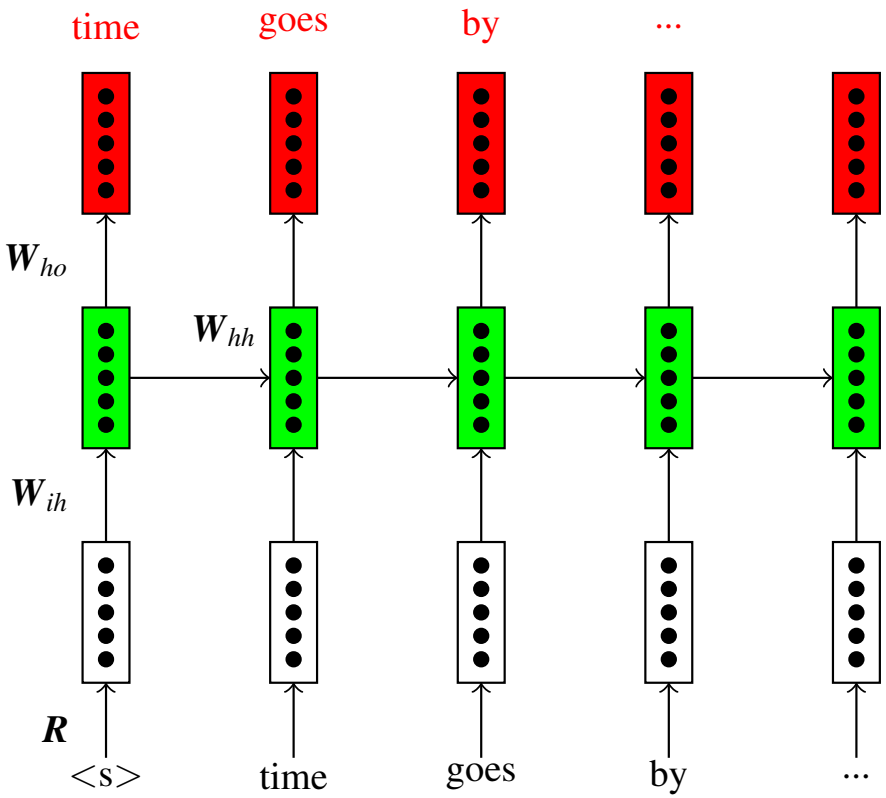
à chaque instant t

- Lire le mot $w_t \rightarrow \boldsymbol{x}_t$ de $\boldsymbol{R}$
- mettre à jour l'état interne
  $\boldsymbol{h}_t = f(\boldsymbol{W}_{vh}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1})$
- (prédire à t une sortie à partir de $\boldsymbol{h}_t$:

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ho}\boldsymbol{h}_t)$$

- g est la fonction softmax

Variante : Prédiction de séquence



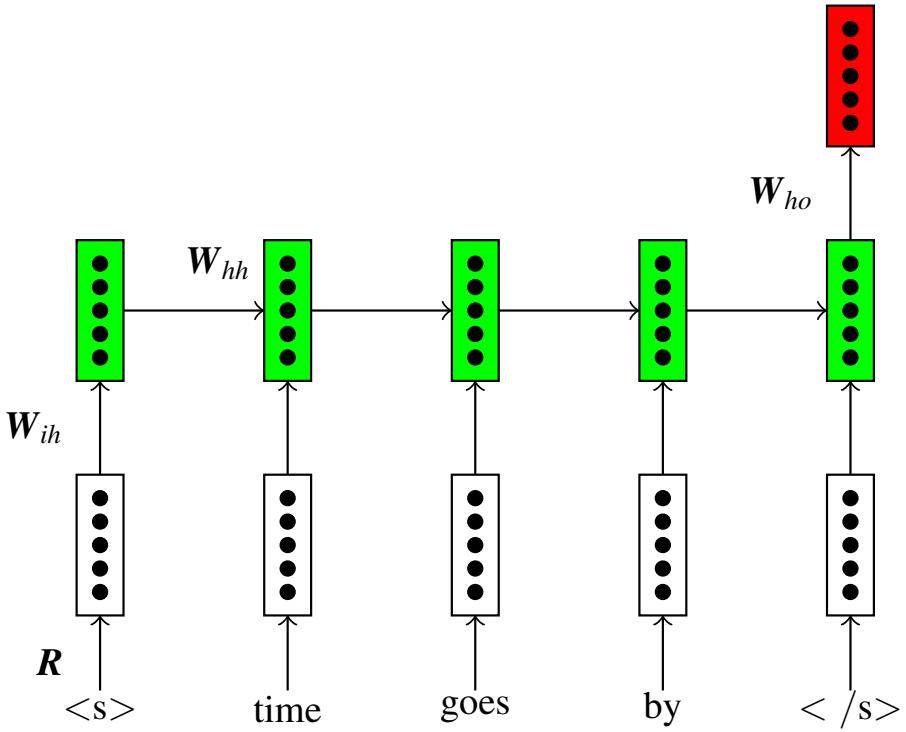
à chaque instant t

- lire $w_t \rightarrow \boldsymbol{x}_t$ de $\boldsymbol{R}$
- $\boldsymbol{h}_t = f(\boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1})$
- prédire le mot/symbôle à $t+1$ à partir $\boldsymbol{h}_t$:

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ho}\boldsymbol{h}_t)$$

- g : softmax

Variante : Classification de séquence



à chaque instant t

- lire $w_t \rightarrow \boldsymbol{x}_t$ de $\boldsymbol{R}$
- $\boldsymbol{h}_t = f(\boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1})$

prédire la classe à partir de $\boldsymbol{h}_L$

Apprentissage

Algorithme

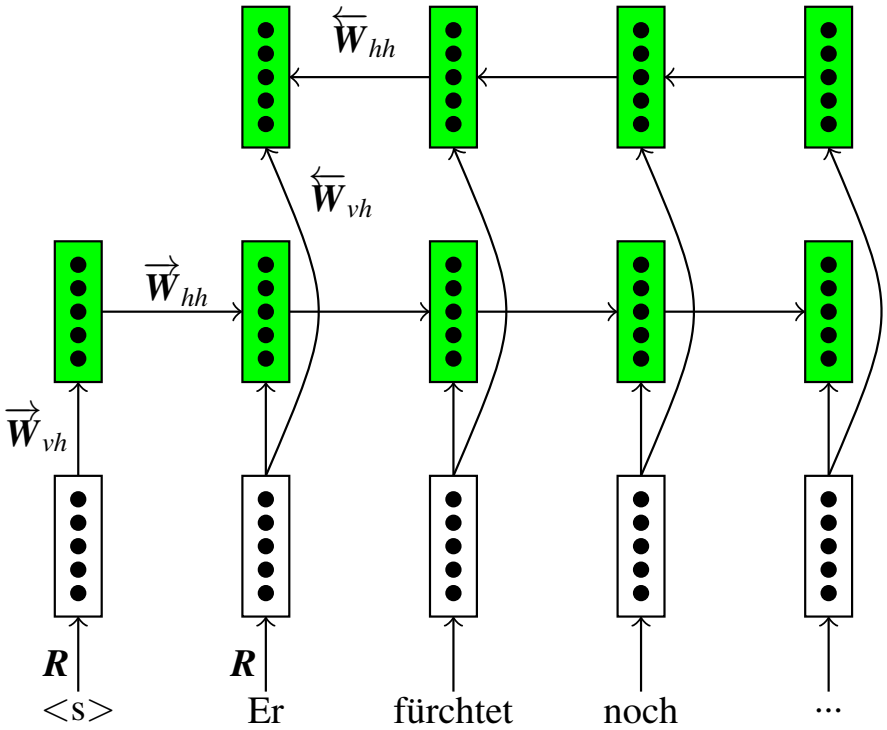
Back-Propagation through time

- à chaque instant t
  - calcul de la fonction objectif et de son gradient
  - Back-Propagation au travers du réseau “déplié”

Problèmes

- Vanishing/exploding gradient [**?**] $\rightarrow$ LSTM, Gradient clipping
- Mémoire à long-terme $\rightarrow$ Bi-recurrent network

Bi-recurrent network



à chaque instant t, de gauche à droite

- $w_t \rightarrow \boldsymbol{x}_t$
- $\overrightarrow{\boldsymbol{h}}_t = f(\overrightarrow{\boldsymbol{W}}_{vh}\boldsymbol{x}_t + \overrightarrow{\boldsymbol{W}}_{hh}\overrightarrow{\boldsymbol{h}}_{t-1})$

à chaque instant t, de droite à gauche

- $w_t \rightarrow \boldsymbol{x}_t$
- $\overleftarrow{\boldsymbol{h}}_t = f(\overleftarrow{\boldsymbol{W}}_{vh}\boldsymbol{x}_t + \overleftarrow{\boldsymbol{W}}_{hh}\overleftarrow{\boldsymbol{h}}_{t-1})$

prédiction:

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ho}[\overrightarrow{\boldsymbol{h}}_t; \overrightarrow{\boldsymbol{h}}_t])$$

$[\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t]$: représentation contextuelle de w_t