Introduction au Deep Learning
J. Rynkiewicz
Les modèles
Problèmes et solutions
Gradient vanishing and exploding
Gated recurrent units
Long Short Term Memory
Application

Introduction au Deep Learning

Réseaux récurrents

J. Rynkiewicz

Université Paris 1

2020

Réseaux de neurones récurrents

Les réseaux de neurones récurrents sont définis en fonction du temps (ou itération de l'algorithme) t :

- Notons :
  - X_t, l'observation au temps t.
  - h_t le vecteur d'état caché du réseau.
  - Y_t la sortie du réseau au temps t.
  - θ le vecteur paramètre du modèle.
- On initialise le vecteur d'état caché h_0.
- Le système dynamique implémenté par le réseau sera :

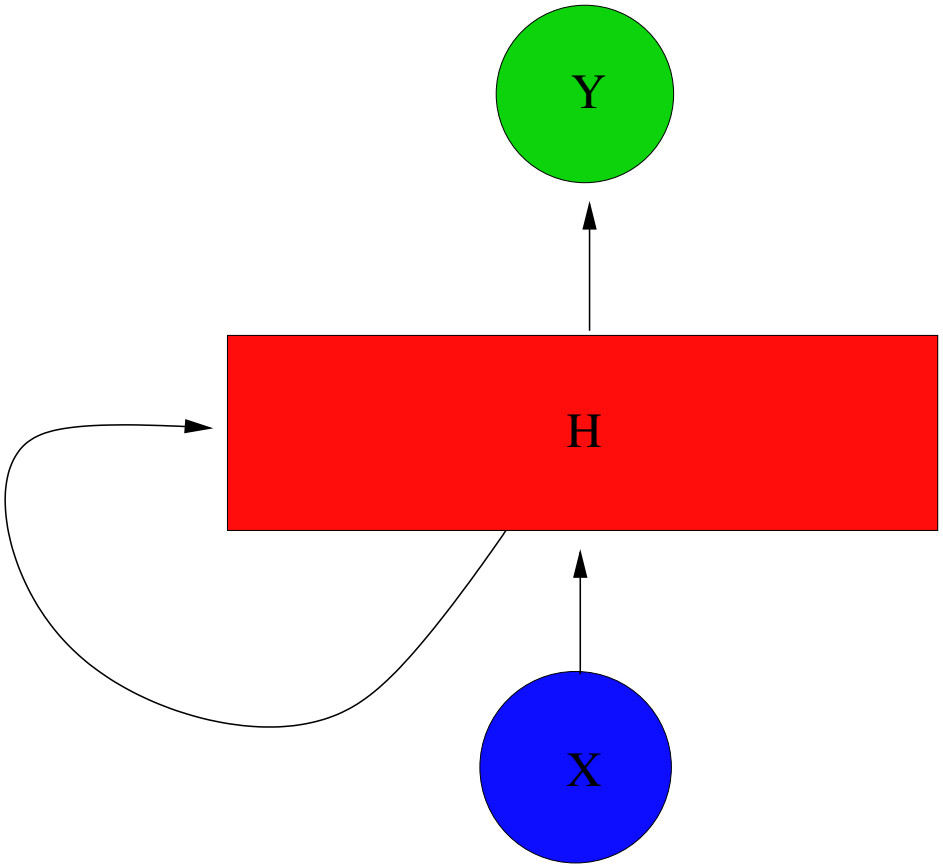
$$\begin{cases} h_t = g_\theta(h_{t-1}, X_t) \\ Y_t = f_\theta(h_t) \end{cases}$$

- On peut remarquer que h_{t-1} est une fonction de h_{t-2} et X_{t-1} et récursivement on peut remonter jusqu'à X_1. Ainsi, Y_t peut être une fonction de tous le passé des observations.

Principe des réseaux récurrents

Ces réseaux permettent de tenir compte de l'état de la couche caché au temps précédent pour influencer la couche cachée au temps présent :

$$\begin{cases} h_t = g_\theta(h_{t-1}, X_t) \\ Y_t = f_\theta(h_t) \end{cases}$$



Réseau récurrent simple

- Supposons que les observations $(X_1, \cdots, X_n)$ sont dans un alphabet $\mathcal{A}$ de cardinal $d = |\mathcal{A}|$.
- Le i-ème symbole de l'alphabet est codé par un vecteur de $\mathbb{R}^d$ avec que des zéros sauf un 1 en i-ème position.
- Pour $u = (u_1, \cdots, u_d) \in \mathbb{R}^d$, on note
$$softmax(u) = \left(\frac{\exp(u_1)}{\sum_{j=1}^d \exp(u_j)}, \cdots, \frac{\exp(u_d)}{\sum_{j=1}^d \exp(u_j)} \right).$$
- Sous une de ses formes la plus simple, l'équation d'un réseau récurrent avec un état caché de dimension H sera :

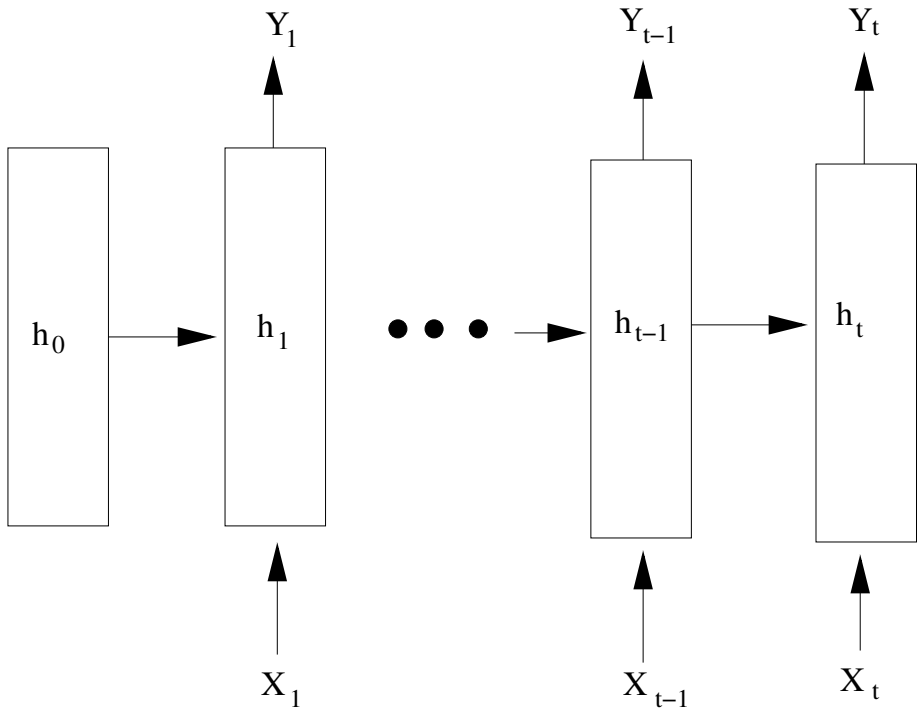
$$\begin{cases} h_0 = 0_{\mathbb{R}^H} \\ \text{si } t \geq 1 \ : \\ h_t = \tanh(W_{hh} h_{t-1} + W_{hx} X_t + b_h) \\ Y_t = softmax(W_{yh} h_t) \end{cases}$$

où $\theta = (b_h, W_{hx}, W_{hh}, W_{yh}) \in \mathbb{R} \times \mathbb{R}^{H \times d} \times \mathbb{R}^{H \times H} \times \mathbb{R}^{d \times H}$ est le paramètre du modèle.

Schéma d'un réseau récurrent simple

Introduction au Deep Learning
J. Rynkiewicz
Les modèles
Problèmes et solutions
Gradient vanishing and exploding
Gated recurrent units
Long Short Term Memory
Application

Un réseau récurrent est un réseau "feedforwfard" qui "s'allonge" avec la suite des observations et dont tous les poids, d'une couche à une autre, sont contraints à être égaux.



Sur ce schéma, les flèches représentent les poids du réseau.

Le problème de l'optimisation des réseaux récurrents

Le gradient se calcule par rétropopagation d'une quantité δ à travers les couches (toutes identiques), suivant une application linéaire du type :

$$\delta_{l+1} = W_l \delta_l.$$

Puisque le réseau "grandit" avec le nombre d'observations, le nombre de rétropropagations aussi, il peut se passer différent phénomènes :

- Si la matrice W_l est contractante : $\|X\| < \|W_l X\|$, la quantité δ_l va converger exponentiellement vite vers 0 et le gradient devient rapidement négligeable pour les couches éloignées de l'observation t. Le réseau oublie alors exponentiellement vite les observations éloignées dans le temps.
- Si, au contraire, $\|X\| > \|W_l X\|$, la quantité δ_l va exploser et le gradient aussi. Les observations du passé auront un poids démesuré.

La solution à ce problème est de munir le réseau de fonctions paramétriques, qui se rappellent ou bien oublient le passé en fonction des observations.

les réseaux GRU (Gated Recurrent Units)

On note $\sigma((u_1, \cdots, u_d)^T) = \left(\frac{exp(u_1)}{1+exp(u_1)}, \cdots, \frac{exp(u_d)}{1+exp(u_d)}\right)^T$ la fonction sigmodïde appliquée à un vecteur, $\circ$ le produit d'Hadamard (i.e. terme à terme) de deux matrices. Les équations vérifiées par un GRU sont les suivantes :

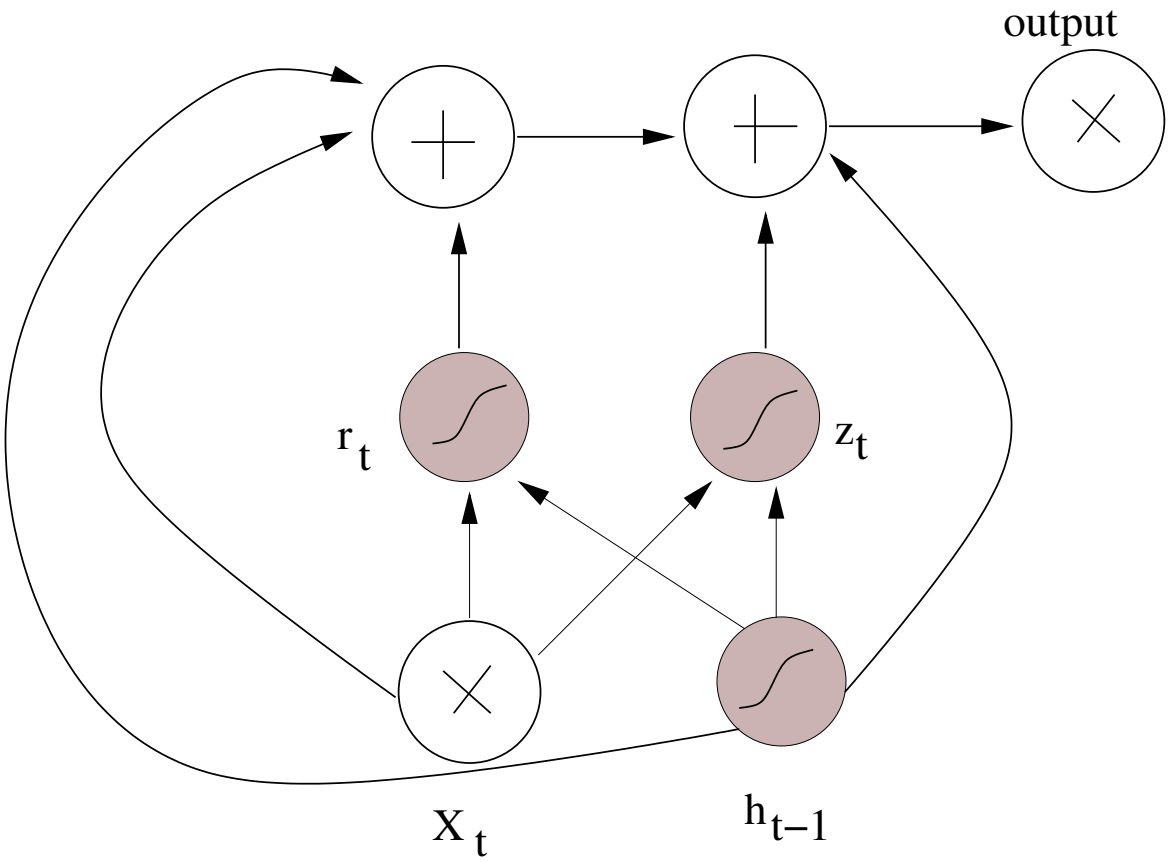
$$\begin{cases} h_0 = 0 \\ z_t = \sigma\left(W_{zh} h_{t-1} + W_{zx} X_t + b_z\right) \\ r_t = \sigma\left(W_{rh} h_{t-1} + W_{rx} X_t + b_r\right) \\ h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma\left(W_{hx} X_t + W_{hh}(r_t \circ h_{t-1}) + b_h\right) \\ Y_t = softmax(W_{yh} h_t) \end{cases}$$

On dit que :

- z_t est la "update gate", si elle est proche de 0, h_t sera proche de h_{t-1} (on se souvient pleinement du passé).
- r_t est la "reset gate", si elle est proche de 0 et que z_t est proche de 1, le passé sera oublié.

Illustration du GRU

On peut schématiser le GRU de la façon suivante :



les réseaux LSTM (Long Short Term Memory)

Avec les mêmes notations que précédemment on aura les équations du LSTM :

$$\begin{cases} h_0 = 0 \\ f_t = \sigma\left(W_{fh} h_{t-1} + W_{fx} X_t + b_f\right) \\ i_t = \sigma\left(W_{ih} h_{t-1} + W_{ix} X_t + b_i\right) \\ o_t = \sigma\left(W_{oh} h_{t-1} + W_{ox} X_t + b_o\right) \\ c_t = f_t \circ c_{t-1} + i_t \circ \sigma\left(W_{cx} X_t + W_{ch} h_{t-1} + b_c\right) \\ h_t = o_t \circ \sigma(c_t) \\ Y_t = softmax(W_{yh} h_t) \end{cases}$$

- f_t est la “forget gate”, si la composante j de f_t est proche de 1, la composante j de c_t sera proche de la composante j de c_{t-1} (on se souvient pleinement du passé).
- i_t est la “input gate”, si si la composante j de i_t est proche de 1, l’entrée courante X_t pourra avoir une grande influence sur la composante j de c_t.
- o_t est la “output gate”, si la composante j de o_t est proche de 1, la composante j de h_t sera proche de la composante j de c_t, sinon elle sera proche de 0.
- c_t est la cellule cachée (un peu un précurseur à l’état caché h_t).

Illustration des équations de la cellule cachée c_t

Introduction au Deep Learning

J. Rynkiewicz

Les modèles

Problèmes et solutions

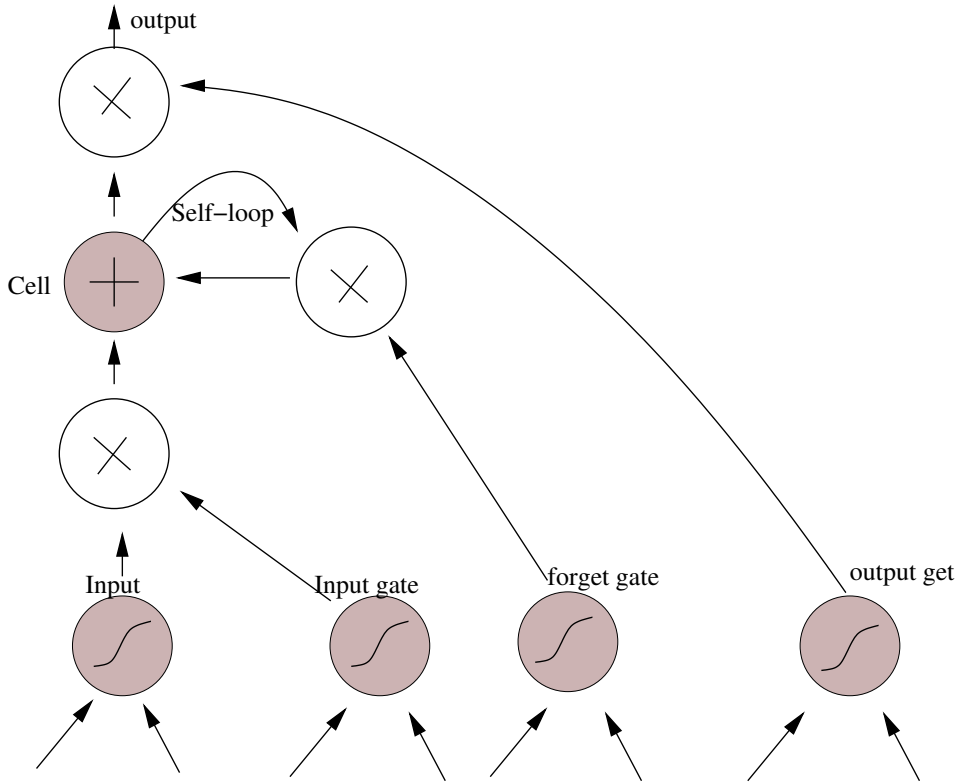
Gradient vanishing and exploding

Gated recurrent units

Long Short Term Memory

Application

On peut schématiser les équations de la cellule cachée c_t de la façon suivante :



Estimation des paramètres

- $\theta_{n+1} = \theta_n - \gamma \frac{1}{B} \sum_{i=1}^{B} \frac{\partial \ln(L_\theta(x_{t+i}, y_{t+i}))}{\partial \theta}$. Au cours de l'algorithme, γ décroît régulièrement.
- On utilise Adagrad ou Adam pour accélérer la descente du gradient. Ces algorithmes améliorent la méthode du gradient stochastique en déterminant automatiquement un taux d'apprentissage pour chaque paramètre.
- On peut utiliser les techniques de régularisation telle que weight decay et drop-out.
- On coupe l'ensemble d'apprentissage en deux :
  - *n* premières données pour l'apprentissage.
  - *m* suivantes pour la validation (hold out).
- la portion des données dans l'ensemble d'apprentissage et des données dans l'ensemble de validation dépend du nombre de données disponibles.

Un exemple : Modèles “caractères” pour des textes (1)

Introduction au Deep Learning
J. Rynkiewicz
Les modèles
Problèmes et solutions
Gradient vanishing and exploding
Gated recurrent units
Long Short Term Memory
Application

- On considère qu’un texte est une suite de caractères.
- Les caractères appartiennent à un alphabet $\mathcal{A} = \{a_1, \cdots, a_d\}$ (les lettres de l’alphabet, la ponctuation, les espaces, les sauts à la ligne).
- On va essayer de prévoir le prochain caractère en fonction des caractères passés.
- Les chaînes de Markov semblent être un modèle raisonnable : Pour $k \in \mathcal{A}$

$$P(X_t = k | X_{t-1}, \cdots, X_1) = P(X_t = k | X_{t-1}, \cdots, X_{t-p})$$

- Les réseaux récurrents, notamment avec les “gates” permettent d’implémenter de tels modèles en gardant un nombre de paramètres libres raisonnable et en permettant l’ordre de la chaîne de Markov p à être relativement grand.
- Un codage canonique des caractères $a_1, \cdots, a_d$ est sur le simplexe (one hot encoding) :

$$a_i = (0, \cdots, 0, 1, 0, \cdots 0) \text{ avec le 1 en ième position.}$$

Un exemple : Modèles “caractères” pour des textes (2)

Introduction au Deep Learning
J. Rynkiewicz
Les modèles
Problèmes et solutions
Gradient vanishing and exploding
Gated recurrent units
Long Short Term Memory
Application

- La sortie du réseau récurrent sera la probabilité du prochain caractère : $Y_t = (P(X_{t+1} = a_1), \cdots, P(X_{t+1} = a_d))^T$.
- Les poids sont estimés en minimisant l'opposée de la log-vraisemblance.
- Le réseau sélectionné sera celui qui minimisera l'erreur de classification sur l'ensemble de validation.
- Une fois que le réseau est entrainé, on va générer des textes en tirant au hasard le caractère suivant à l'aide des probabiblités calculées par le réseau.
- Les caractères générés seront utilisés par le réseau pour générer les caractères suivant.
- Pour générer des cactères très probables, on utilisera un vecteur de probabilités de la forme : $softmax(\frac{1}{T} W_{yh} h_t)$, où T est la température.
- Plus T est petite, plus les caractères les plus probables auront des chances d'être tirés.

Un exemple : Modèles “caractères” pour des textes (3)

Introduction au Deep Learning
J. Rynkiewicz
Les modèles
Problèmes et solutions
Gradient vanishing and exploding
Gated recurrent units
Long Short Term Memory
Application

- On a estimé un LSTM avec 3 couches cachées et 512 unités cachées sur chaque couche sur l’oeuvre intégrale de Shakespeare (1.4 millions de caractères).
- Le pas du gradient était $\gamma = 0.001$, on a repassé 10 fois sur les données d’apprentissage.
- Les 200000 dernières lettres on était utilisée pour l’ensemble de validation.
- On a généré ensuite un texte avec une température $T = 0.3$, en voici un extrait :
  - CASSANDRA : The world is strong and sick, but that I would The sun that strikes to come to be the season.
  - FERDINALUS : What would you think the bright and sound of men That is a second strength to strike the sun ?
  - CASSIUS : What is this the devil shall I see the state ?
  - CASSIUS : The duke and honest country with the state Which the most father shall be subject’d with him.
  - KING JOHN : A little sword and the most part of death, And then the sun to her that makes the fool The state of soldiers and the little gates And the conflict of this command and mouth The way to see the true and subjects forth.