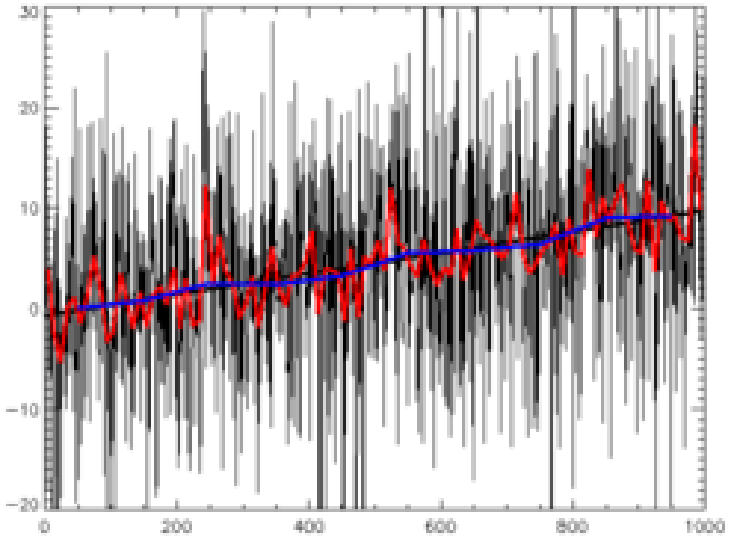
Troude Jean-Benoit
Prevotat Lucas

Mercredi 7 Avril 2021

L’arrivée de l’attention dans la prédiction des séries temporelles



Sommaire

1) Présentation des séries temporelles et de leur prédiction

A / Introduction

Une série temporelle est une suite de valeurs numériques (x1, · · · , xn) représentant l'évolution d'une quantité spécifique au cours du temps. Le nombre n est appelé la longueur de la série. Il est la plupart du temps bien utile de représenter la série temporelle sur un graphe construit de la manière suivante : en abscisse le temps, en ordonnée la valeur de l'observation à chaque instant. Pour des questions de lisibilité, les points ainsi obtenus sont reliés par des segments de droite.

Pourquoi vouloir prévoir ?

Très souvent pour des raisons économiques (prévoir l'évolution des ventes d'un certain produit, prévoir la consommation d'électricité pour ajuster au mieux la production).

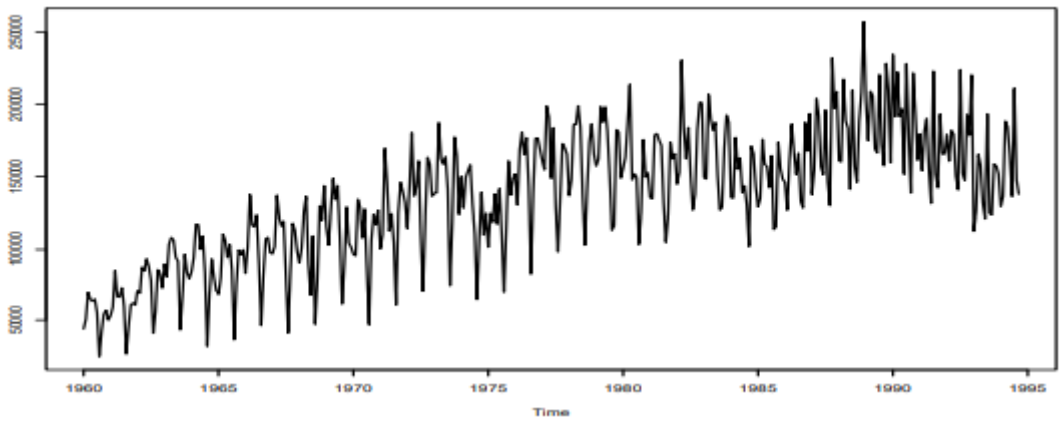
Comment prévoir ?

Toujours en s'appuyant sur ce qui s'est passé avant l'instant à partir duquel démarre la prévision. Et parfois en utilisant des renseignements annexes.

Peut on prévoir parfaitement bien ?

Jamais. Il y a toujours une erreur de prévision, et les bonnes méthodes fournissent non pas une prévision, mais un intervalle de prévision. Il arrive souvent qu'une amélioration minime de la qualité de la prévision ait une grosse incidence sur les coûts.

En finance, de nombreux modèles avancés de prévision des séries de temps sont utilisés pour prédire le cours des actions, car dans les séquences historiques il y a beaucoup de bruit et une grande incertitude dans l'information, qui peut dépendre de plusieurs facteurs pas toujours étroitement liés au marché boursier.



Titre : *Série temporelle de la vente de voitures de 1960 à 1995*

Les données courent sur 35 ans, et il y en a une par mois. Soit 420 points. On y voit une croissance non linéaire (peut être de type logarithmique). Les deux saisonnalités d'un an et de 6 mois se voient moins que dans la série précédente à cause du plus grand resserrement des abscisses. On voit aussi une rupture très nette aux environs de l'année 1983. Cette rupture fait que la série est extrêmement difficile à modéliser et donne de très mauvaises prévisions quelle que soit la méthode utilisée.

B / Méthode

La méthode générale pour étudier une série temporelle est la suivante :

1. Dans un premier temps, on trace la série des données et on repère ses principales caractéristiques. On regarde en particulier si on décèle :
    - une tendance
    - une composante saisonnière
    - une ou des ruptures dans le comportement de la série
    - une ou des observations aberrantes.

2. Dans un deuxième temps, on s'attache à estimer ou enlever la tendance et la composante saisonnière (mt et st) pour obtenir une série Xt de résidus stationnaires. Pour cela on peut utiliser plusieurs techniques : transformation des données, estimer les tendances et composantes saisonnières puis les enlever des données, différencier la série.

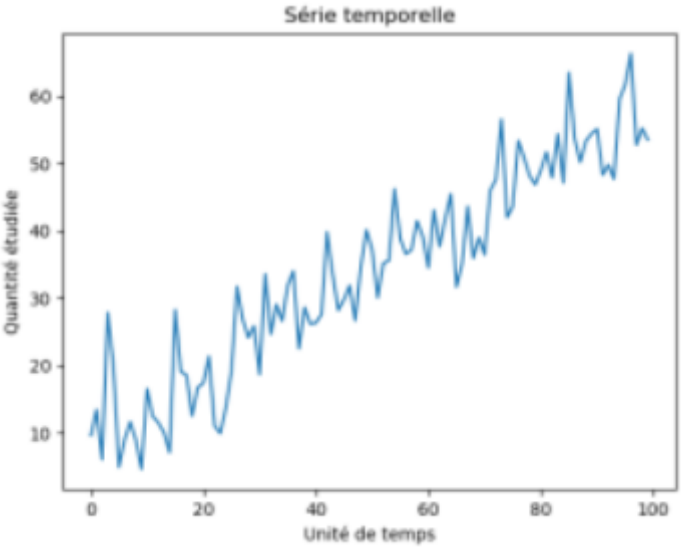
3. Choisir un modèle de série stationnaire pour les résidus

4. Prévoir les valeurs futures de la série en prévoyant d'abord celles des résidus puis "remonter" jusqu'à la série initiale en les transformations inverses.

C / Traduction mathématique

Les séries temporelles stationnaires sont rares. Dans la pratique, on tente de s'y ramener en effectuant des transformations qui éliminent par exemple une composante structurelle déterministe. Le modèle additif le plus simple est de la forme
Xt = mt + st | {z } (partie déterministe) + Zt |{z} (partie aléatoire) où m t , s t , et Zt jouent les rôles suivants :

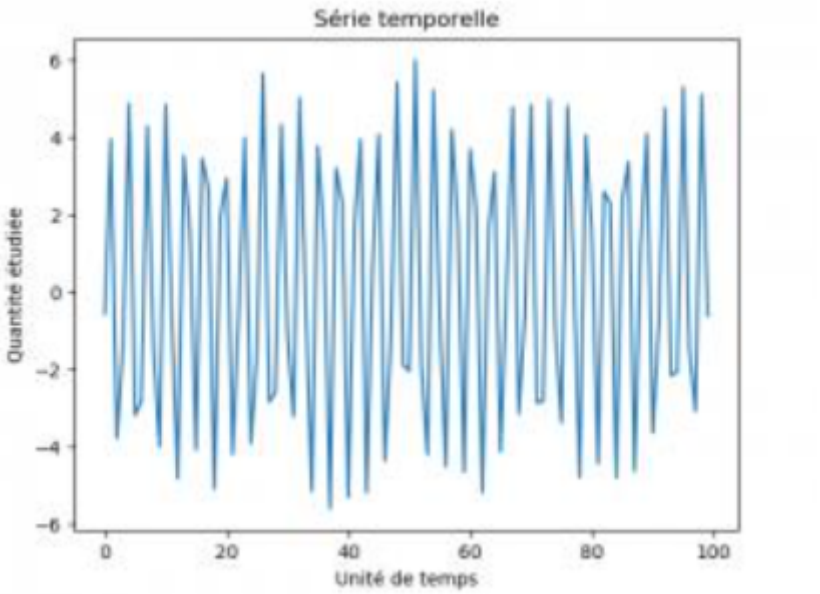
A. Tendance.
   La fonction t → m t est une fonction qui varie lentement, appelée tendance
   La tendance représente la direction générale des données. Pour ce faire on observe la série uniquement sur une longue durée, on ignore les variations à court terme ou les variations liées au bruit.



B. Saisonnalité.

La fonction t → s t est une fonction périodique, appelée composante saisonnière

La saisonnalité fait référence aux fluctuations périodiques qui se répètent tout au long de la période de la série chronologique

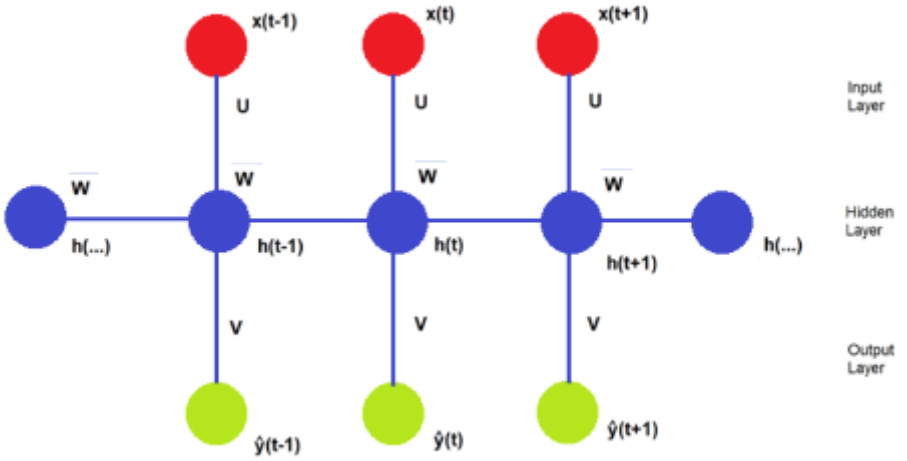


C. Bruit.

Le processus (Zt) t ∈ Z est un bruit qu'on espère stationnaire .

L'utilisation du Deep Learning pour les séries temporelles surmonte les inconvénients traditionnels de l'apprentissage automatique avec de nombreuses approches différentes. Dans l'article mentionné tout en haut 5 différentes architecture d'apprentissage profond pour la prévision de séries time sont présentés:

-Réseaux neuronaux récurrents (RRN), qui sont l'architecture la plus classique et la plus utilisée pour les problèmes de prévision des séries de temps;



-Mémoire à long terme (LSTM), qui sont une évolution des ARN développés afin de surmonter le problème de gradient de fuite;

-Unité récurrente fermée (GRU), qui sont une autre évolution des ARN, semblable à LSTM;

-Modèle Encoder-Décoder, qui est un modèle pour les ARN introduits afin de résoudre les problèmes où les séquences d'entrée diffèrent en longueur des séquences de sortie;

-Mécanisme d'attention, c'est-à-dire une évolution du modèle Encoder-Décoder, développé afin d'éviter d'oublier les parties antérieures de la séquence.

Nous n'allons pas expliquer toutes ces méthodes car elles sont très bien détaillées dans l'article et il serait inutile ici d'en faire un copier-coller.

En conclusion de cette première partie, Les réseaux neuronaux récurrents sont la technique d'apprentissage profond la plus populaire pour la prévision des séries de temps puisqu'ils permettent de faire des prédictions fiables sur les séries de temps dans de nombreux problèmes différents.Le principal problème avec les ARN est qu'ils souffrent du problème de gradient de disparition lorsqu'ils sont appliqués à de longues séquences.

2) Présentation de la prédiction par LSTM

A / Méthode

LSTM et GRU ont été créés afin d'atténuer le problème de gradient de disparition des ARN avec l'utilisation de barrières, qui régulent le flux d'informations à travers la chaîne de séquence. L'utilisation de LSTM et GRU donne des résultats remarquables dans des applications comme la reconnaissance vocale, la synthèse vocale, la compréhension naturelle du langage, etc.

Le modèle Encoder-Decoder pour les réseaux neuronaux récurrents est la technique la plus courante ou les problèmes de cartographie séquence à séquence où les séquences d'entrée diffèrent en longueur des séquences de sortie.

Le mécanisme Attention est une évolution du modèle Encoder-Decoder, qui est né pour résoudre la diminution des performances du modèle Encoder-Decoder en présence de longues séquences, en utilisant un vecteur de contexte différent pour chaque étape. Il donne des résultats remarquables par exemple dans de nombreux domaines comme la NLP et la classification des documents.

Comme expliqué dans l'article, bien qu'ils soient construits spécifiquement pour gérer des séquences, les RNN simples ont certaines limites.
Plusieurs variantes aux RNN standards ont vu le jour pour remédier aux problèmes évoqués. Nous allons ici décrire les LSTM, pour Long Short-Term Memory. Ce type de RNN est très utilisé en traitement du langage naturel.

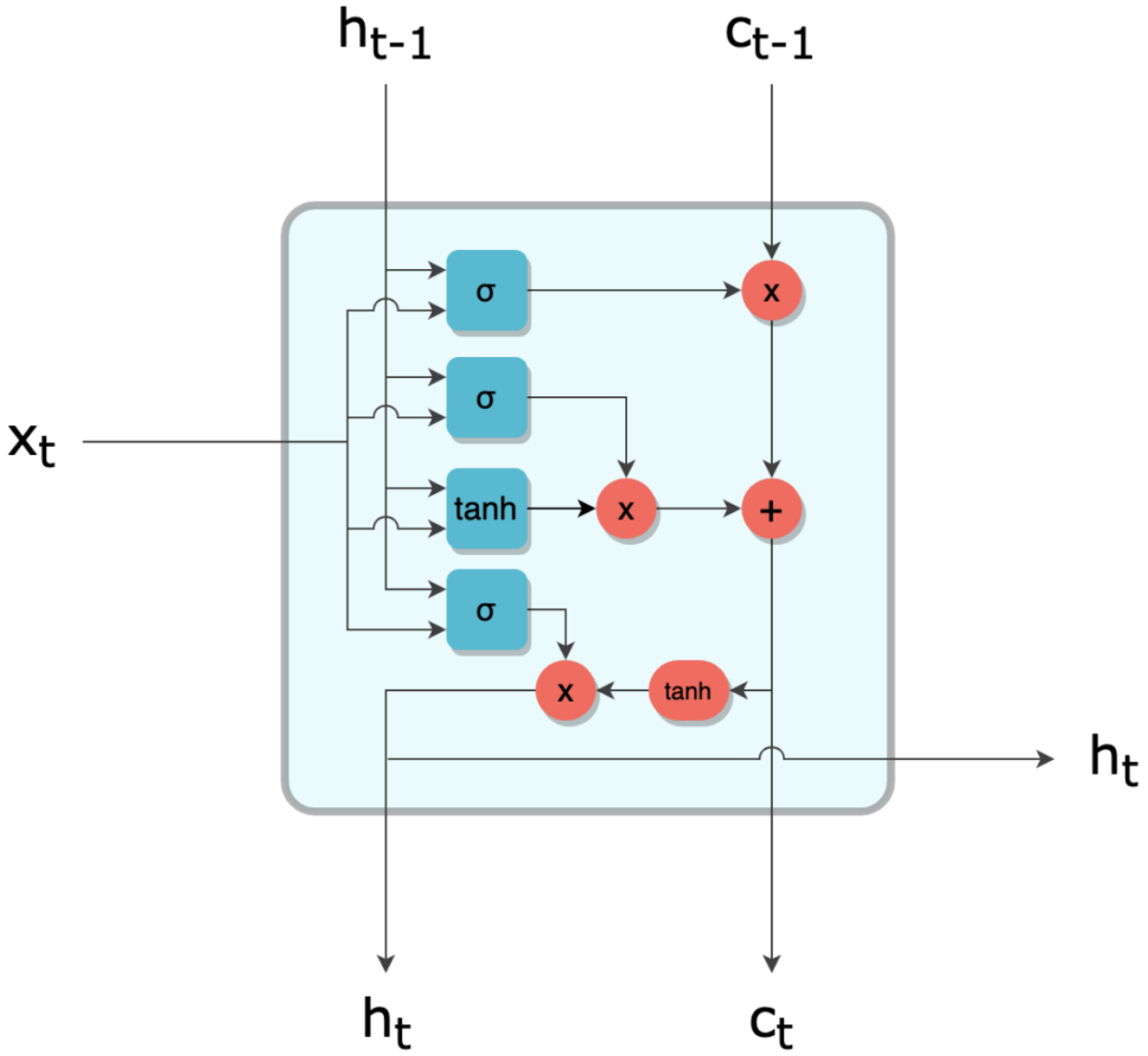
L’idée derrière ce choix d’architecture de réseaux de neurones est de diviser le signal entre ce qui est important à court terme à travers le hidden state (analogue à la sortie d’une cellule de RNN simple), et ce qui l’est à long terme, à travers le cell state. Ainsi, le fonctionnement global d’un LSTM peut se résumer en 3 étapes :

1. Détecter les informations pertinentes venant du passé, piochées dans le cell state à travers la forget gate
2. Choisir, à partir de l’entrée courante, celles qui seront pertinentes *à long terme*, via l’input gate. Celles-ci seront ajoutées au cell state qui fait office de mémoire longue ;
3. Piocher dans le nouveau cell state les informations importantes *à court terme* pour générer le hidden state suivant à travers l’output gate.

B / Représentation

Comme le RNN, le LSTM définit donc une relation de récurrence, mais utilise une variable supplémentaire qui est le cell state c. L'information transite d'une cellule à la suivante par deux canaux, h et c. À l'instant t, ces deux canaux se mettent à jour par l'interaction entre leurs valeurs précédentes ht-1 et ct-1 ainsi que l'élément courant de la séquence xt.

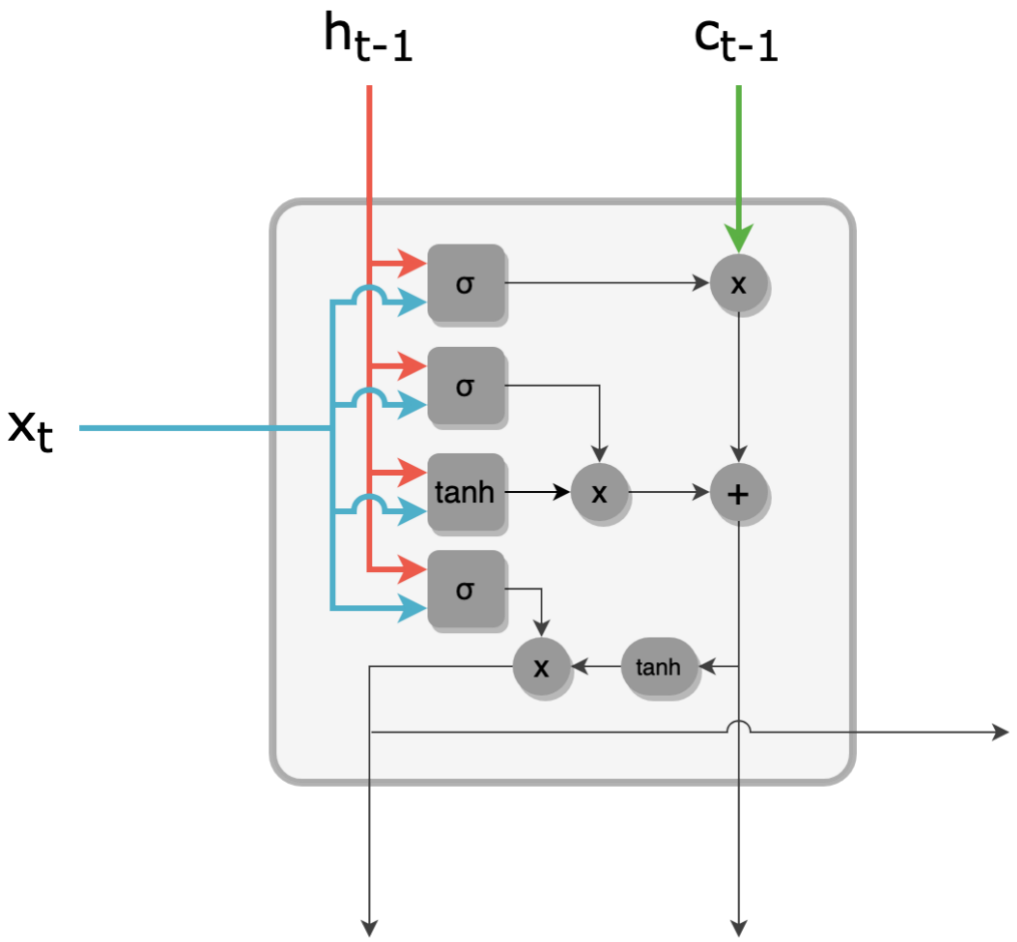
Dans le paragraphe suivant, nous allons essayer de comprendre pas à pas les opérations réalisées par une cellule LSTM.



1 – La tème cellule LSTM prend 3 vecteurs en entrée :

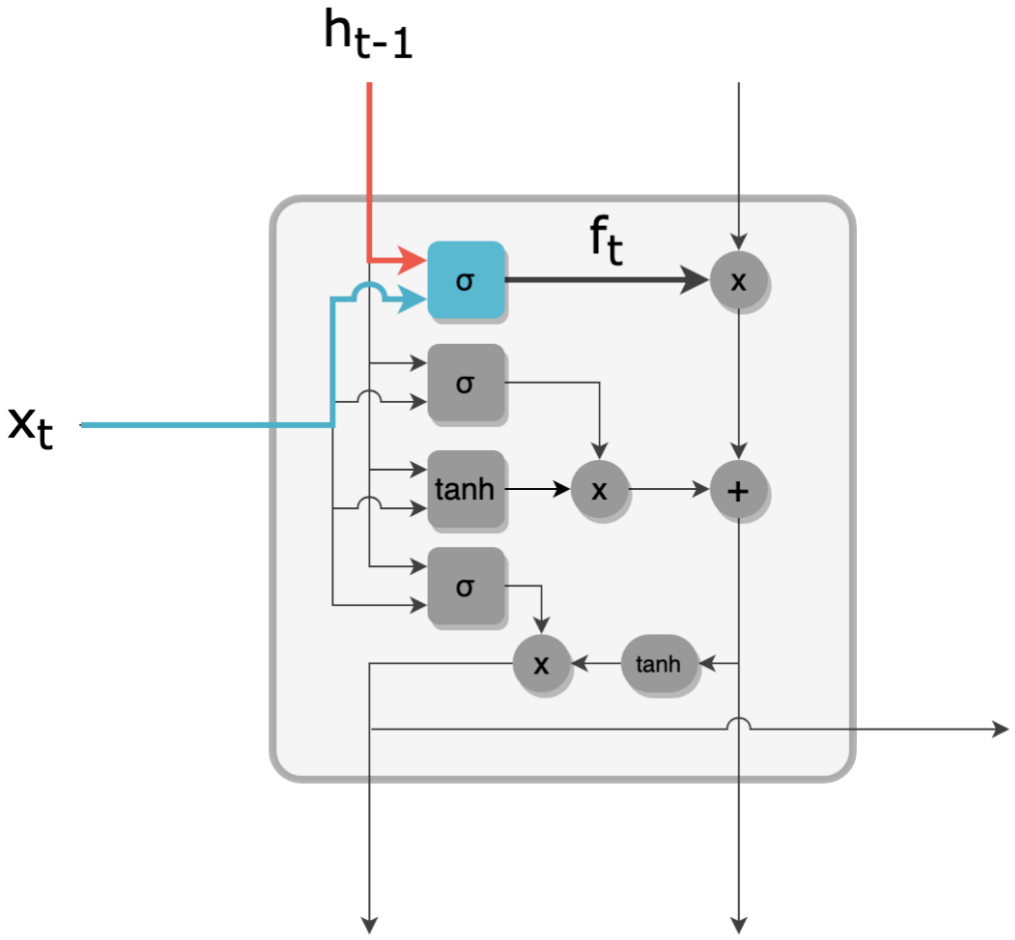
- L'élément courant de la séquence **xt** : représentation vectorielle du tème caractère (vecteur de taille **M**)
- Le hidden state de la cellule précédente **ht-1** (vecteur de taille **R**)
- Le cell state de la cellule précédente **ct-1** (vecteur de taille **R**).

C'est ce dernier vecteur que nous allons suivre particulièrement. Il s'agit d'une route privilégiée de transmission d'information sur la séquence. Pour éviter le problème du vanishing gradient, le cell state est en effet mis à jour de façon **additive** à chaque étape, sans passer par une activation.



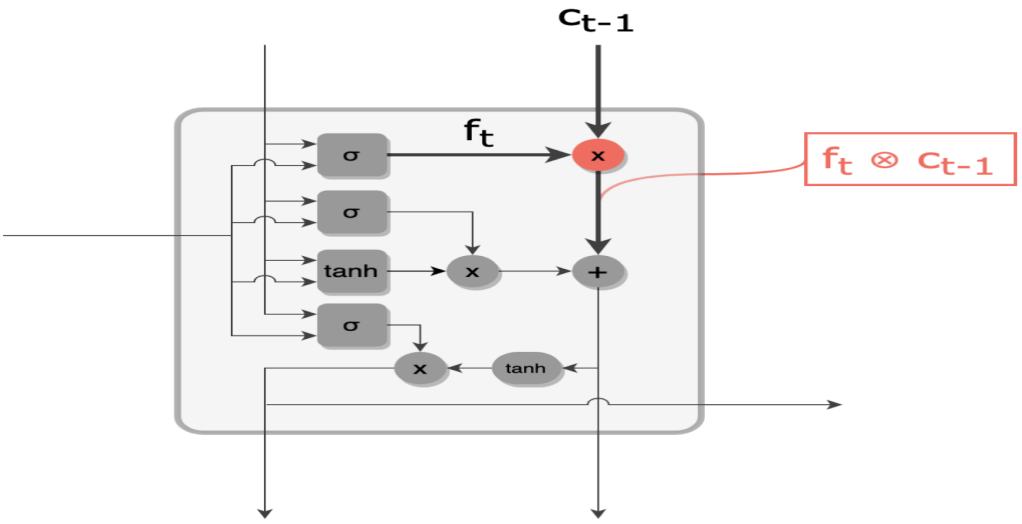
Titre : *Entrées d'une cellule LSTM : input, hidden state, cell state*

2 – La forget gate est une couche dense de taille R avec une activation sigmoïde. À partir de ht-1 et xt, cette forget gate produit un vecteur ft de taille R, dont les valeurs sont comprises entre 0 et 1.



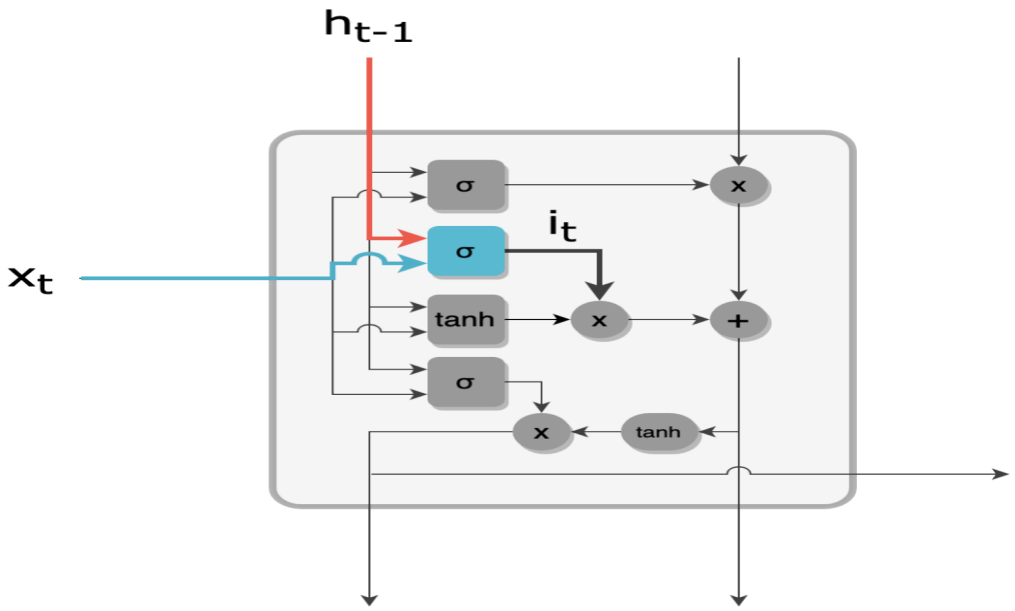
Titre : *Filtre de la forget gate*

3 – La forget gate agit comme un filtre pour « oublier » certaines informations du cell state. En effet, on effectue une multiplication terme à terme entre ft et ct-1, ce qui a tendance à annuler les composantes de ct-1 dont les homologues côté ft sont proches de 0. On obtient alors un cell state filtré, toujours de taille R.



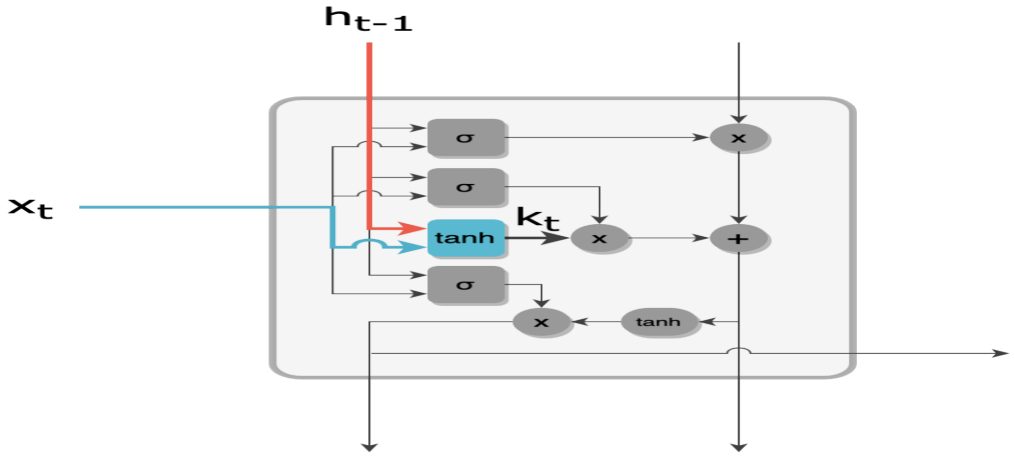
Titre : *Cell state filtré par la forget gate*

4 – L'input gate produit un filtre it de valeurs comprises entre 0 et 1 et de taille R, à partir de ht-1 et xt, de façon similaire à la forget gate.



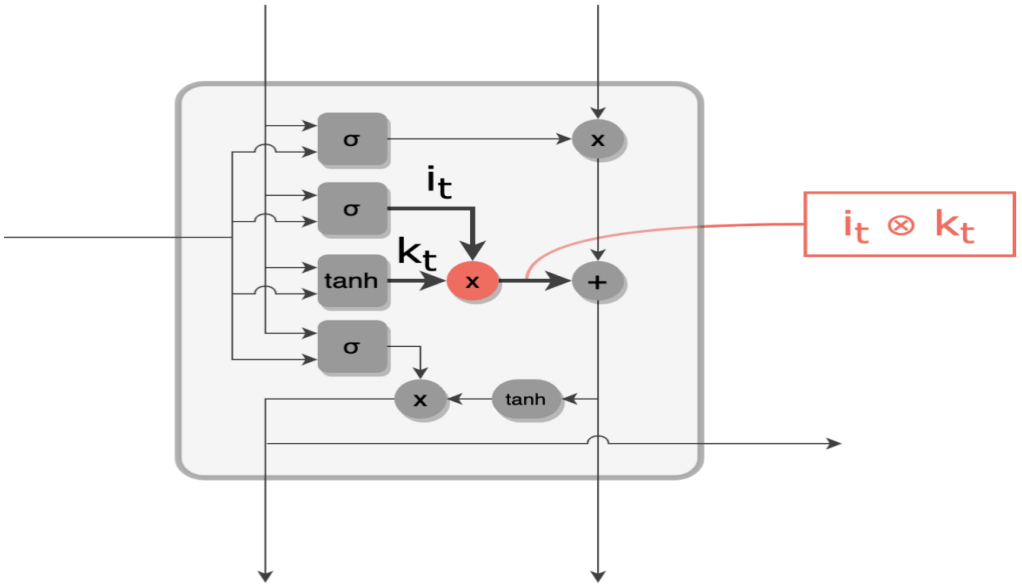
Titre : *Filtre de l'input gate*

5 – En parallèle, un vecteur kt de taille R est créé par une couche *tanh*. kt est le vecteur candidat pour mettre à jour le cell state.



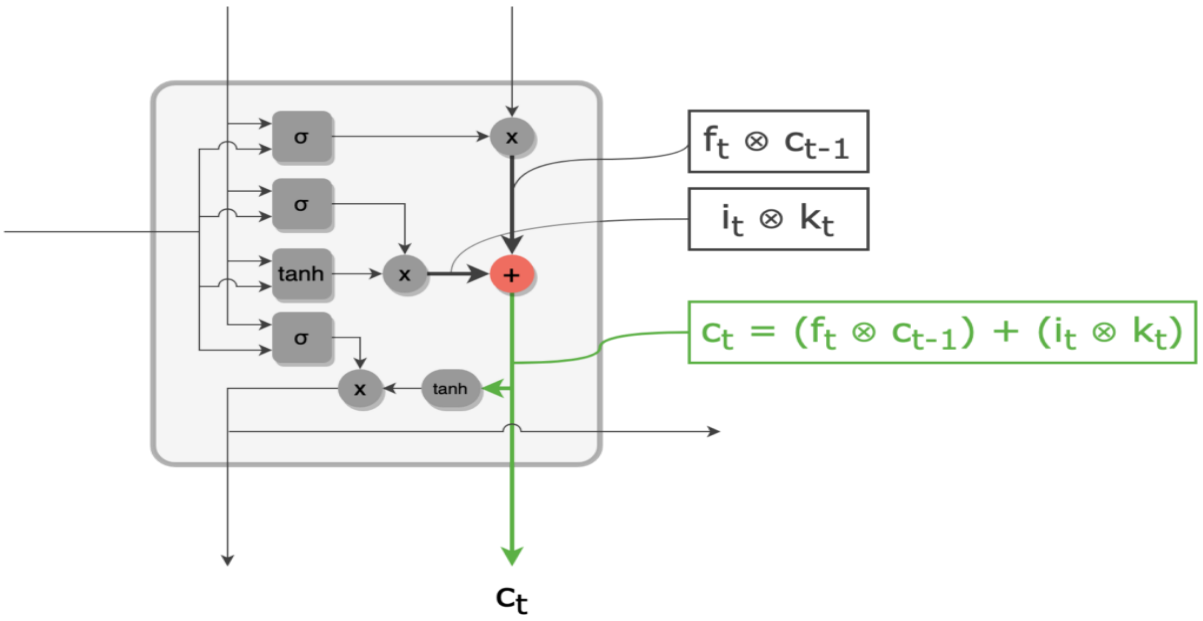
Titre : *Vecteur candidat à la mise à jour du cell state*

6 – Le candidat kt est filtré par l'input gate it via une multiplication terme à terme. On obtient le vecteur de mise à jour du cell state.



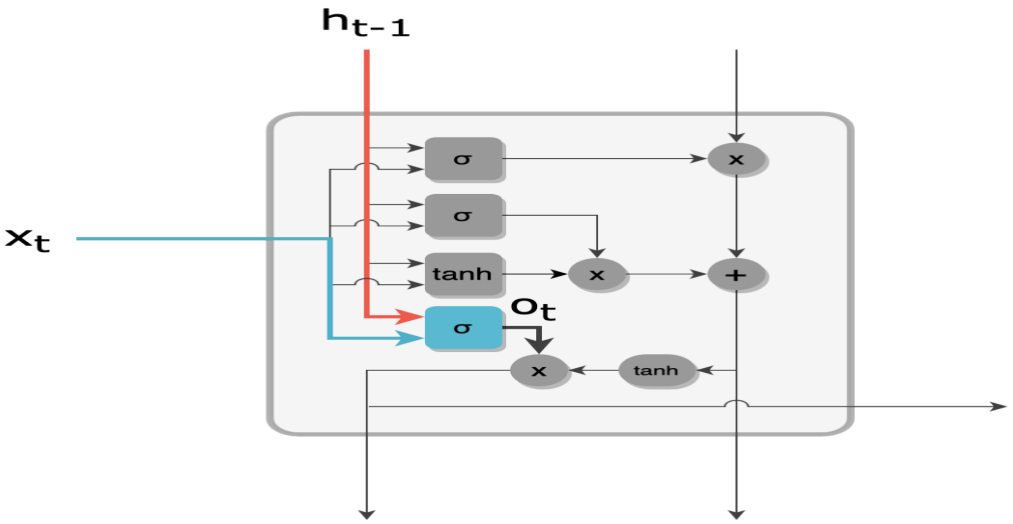
Titre : *Vecteur candidat filtré par l'input gate*

7 – Le cell state filtré (obtenu à l'étape 3) est mis à jour grâce au vecteur candidat filtré (obtenu à l'étape 6). La mise à jour est une simple addition terme à terme de ces deux vecteurs. On obtient alors le nouveau cell state ct.



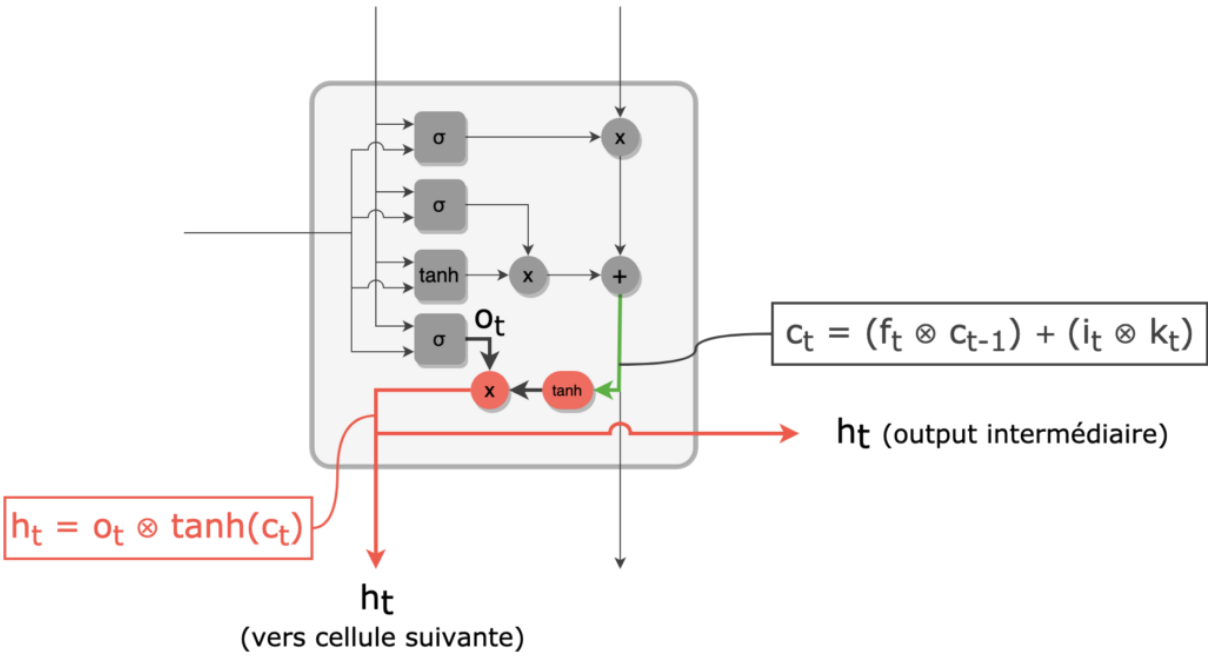
Titre : *Mise à jour du cell state par addition du vecteur candidat filtré*

8 – De façon analogue à ft et it, l'output gate produit un filtre ot de valeurs entre 0 et 1, et de taille R.



Titre : *Filtre de l'output gate*

9 – Les valeurs du nouveau cell state ct sont ramenées à l'intervalle]-1, 1[par une activation tanh. Un filtrage par l'output gate ot est ensuite effectué pour enfin obtenir la sortie ht.



Titre : *Nouveau hidden state*

Bien qu'efficace, cette approche a quand même quelques inconvénients. L'un des défauts des LSTM est qu'il est nécessaire de lire entièrement une séquence pour produire une prédiction. En traduction par exemple, cette démarche reviendrait à lire entièrement une phrase en mémorisant tous ses mots, pour ensuite produire une phrase traduite d'un seul coup. On imagine mal un humain procéder de la même façon.

Les architectures utilisant le mécanisme d'attention répondent à cette limitation des LSTM. Elles suivent l'intuition que tous les mots n'ont pas le même poids sémantique dans la production d'une phrase traduite, et qu'on ne traduit jamais mot à mot. En se focalisant sur des parties de la phrase, et en identifiant un contexte « utile » pour chaque mot, ces mécanismes permettent d'améliorer encore plus les performances des modèles.

3) Deep Learning par CapsulsNets

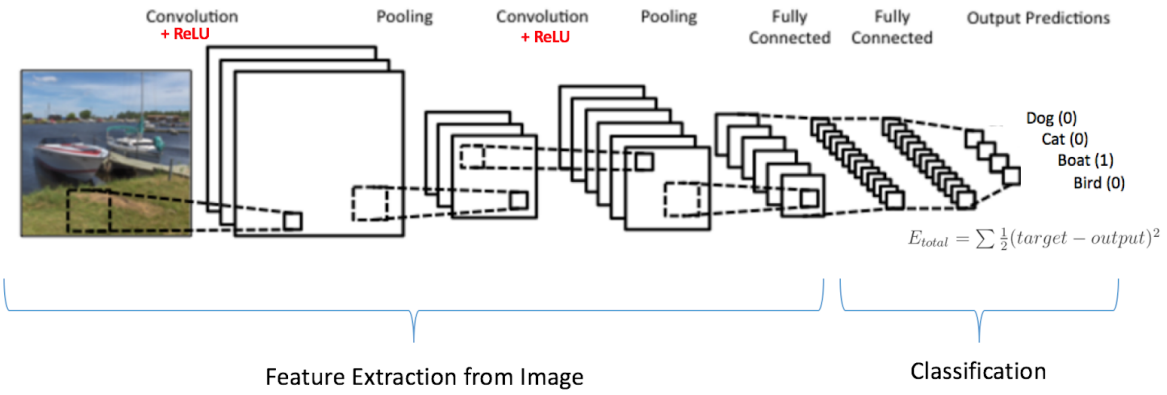
A / Introduction

Yann LeCun a proposé le CNN pour la première fois en 1998. Il a ensuite été en mesure de détecter des chiffres manuscrits avec un simple réseau neuronal convolutionnel à cinq couches qui a été formé sur l'ensemble de données MNIST, contenant plus de 60.000 exemples. L'idée est simple : former le réseau, identifier les fonctionnalités des images et les classer. Puis, en 2019 EfficientNet-B7 a réalisé la performance de pointe dans la classification des images sur l'ensemble de données ImageNet. Le réseau peut identifier l'étiquette d'une image particulière à partir de plus de 1,2 million d'images avec 84,4% de précision.

En examinant ces résultats et ces progrès, on peut en déduire que les approches convolutionnelles permettent d'apprendre de nombreuses fonctionnalités sophistiquées avec des calculs simples.

Pour comprendre la mise en commun, il faut savoir comment fonctionne un CNN. La pierre angulaire des NN est la couche convolutionnelle. Ces couches sont responsables de l'identification des caractéristiques d'une image donnée, comme les courbes, les bords, la netteté et la couleur. En fin de compte, les couches entièrement connectées du réseau combinent des fonctionnalités de très haut niveau et produiront des prédictions de classification.

Vous pouvez voir ci-dessous à quoi ressemble un réseau convolutionnel de base.



Contrairement aux neurones normaux, les capsules effectuent leurs calculs sur leurs entrées, puis « encapsulent » les résultats en un petit vecteur de sorties très instructives. Une capsule pourrait être considérée comme un remplacement ou un substitut à un neurone artificiel moyen; tandis qu'un neurone artificiel traite des scalaires, une capsule traite des vecteurs.

B / Etapes de fonctionnement

1) La multiplication des vecteurs d'entrée par les matrices de poids

   Les vecteurs d'entrée représentent soit l'entrée initiale, soit une entrée fournie par une couche antérieure du réseau. Ces vecteurs sont d'abord multipliés par les matrices de poids. La matrice de poids, comme décrit précédemment, capture les relations spatiales. Disons qu'un objet est centré autour d'un autre, et qu'il est également proportionné en taille. Le produit du vecteur d'entrée et la matrice de poids signifieront la fonction de haut niveau.

2) La multiplication des résultats par le poids

   Un réseau capsule ajuste les poids de telle sorte qu'une capsule de bas niveau est fortement associée à des capsules de haut niveau qui sont à proximité.La capsule de haut niveau qui a la distance minimale entre l'amas de prédictions déjà faites et la capsule nouvellement prédite aura un poids plus élevé, et les capsules restantes se voir attribuer des poids inférieurs, en fonction de la mesure de la distance.

3) Calculer la somme pondérée des vecteurs d'entrée

   ->Cela résume toutes les sorties obtenues à partir de l'étape précédente.

4) Appliquer une fonction d'activation (vectorielle non linéarité) pour obtenir la sortie

Dans un réseau capsule, la non-linéarité vectorielle est obtenue en « écraser » (c'est-à-dire via une fonction d'activation) le vecteur de sortie, pour qu'il soit d'une longueur de 1 et d'une direction constante. La fonction de non-linéarité est donnée par :

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|}$$

où sJ est la sortie obtenue à partir de l'étape précédente, et vJ est la sortie obtenue après l'application de la non-linéarité. Le côté gauche de l'équation effectue des écrasements supplémentaires, tandis que le côté droit de l'équation effectue l'échelle unitaire du vecteur de sortie.

L'architecture CapsNet se compose d'un encoder et d'un décodeur, où chacun dispose d'un ensemble de trois couches. Un encoder a une couche convolutionnelle, une couche PrimaryCaps et une couche DigitCaps; le décodeur a 3 couches entièrement connectées.

Tout ceci est détaillé dans l'article donc nous ne le ferons pas ici.

C / Conclusion

Pour conclure cette partie, un réseau capsule pourrait être considéré comme une « véritable imitation » du cerveau humain. Contrairement aux réseaux neuronaux convolutionnels, qui n'évaluent pas les relations spatiales dans les données données données, les réseaux de capsules considèrent l'orientation des parties d'une image comme un élément clé de l'analyse des données. Ils examinent ces relations hiérarchiques pour mieux identifier les images. Le mécanisme inverse graphique dont notre cerveau fait usage est imité ici pour construire une représentation hiérarchique d'une image, et la faire correspondre à ce que le réseau a appris.

Bien qu'il ne soit pas encore efficace sur le plan informatique, il semble y avoir un coup de pouce de précision bénéfique dans la lutte contre les scénarios du monde réel. Le routage dynamique des capsules est ce qui rend tout cela possible. Il utilise une stratégie inhabituelle de mise à jour des poids dans un réseau, évitant ainsi l'opération de mise en commun. Au fil du temps, les réseaux de capsules pénétreront sûrement dans divers autres domaines, rendant les machines encore plus humaines.