Alexandre Martin

Etude d'article :

Bidirectional Recurrent Neural Networks

Mike Schuster et Kuldip K. Paliwal, 1997

Sommaire

1. Introduction

L'article de Schuster et Paliwal construit une structure de réseau de neurones récurrent bidirectionnel, à partir de la structure connue du réseau de neurones récurrent unidirectionnel. Les auteurs comparent les performances de cette nouvelle structure à d'autres structures connues vis-à-vis de deux problèmes :

- le problème de régression linéaire, avec comme fonction à minimiser l'erreur quadratique moyenne
- le problème de catégorisation, avec comme fonction à maximiser le taux de reconnaissance

Les auteurs montrent qu'il est toutefois possible d'obtenir de meilleurs résultats pour certains problèmes particuliers en modifiant légèrement ces réseaux bidirectionnels.

2. Modèles et résultats

2.1. Réseaux de neurones récurrents (RNN)

2.1.1. Réseau simple

Les réseaux de neurones récurrents permettent de rendre compte efficacement des corrélations entre les données proches temporellement dans la séquence. Il existe deux types de RNN unidirectionnels :

- l'architecture collecte toutes les données passées (x(1), x(2), …, x(t)) pour prédire y(t) : c'est un RNN dit Forward.
- l'architecture collecte toutes les données futures (x(t), x(t+1), ..., x(T)) pour prédire y(t) : c'est un RNN dit Backward.

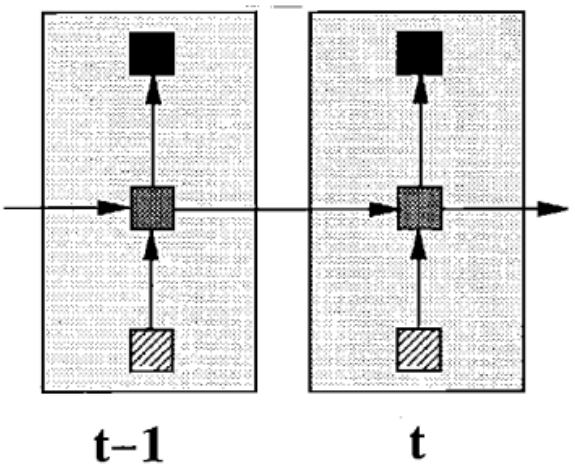

Figure 1 : Structure simple d'un RNN Forward pour 1 période de temps

2.1.2. Réseau avec délai

Il est possible d'intégrer un délai à la structure des RNN. Dans le cas d'un RNN Forward, on utilisera alors également quelques données futures pour prédire $y(t_c)$. Le tableau suivant dépicte ainsi la quantité d'information utilisée pour prédire la sortie $y(t_c)$ selon la structure utilisée (MLP est le perceptron multicouche, à propagation directe) :

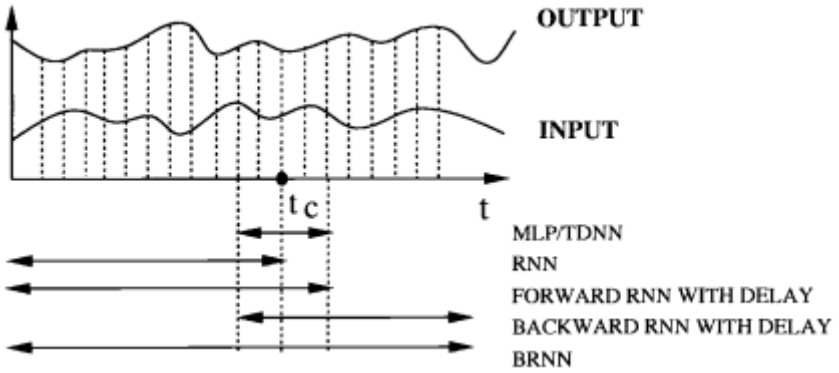

Figure 2 : Quantité d'information utilisée selon le type de structure

Bien que le délai puisse être choisi aussi grand que possible pour utiliser également toute l'information future, un délai choisi trop élevé diminue la qualité de la prédiction. Une explication possible à ce phénomène est que la puissance du modèle est alors accordée majoritairement à la mémorisation des données plutôt qu'à la prédiction elle-même.

Il est également possible d'effectuer une procédure de fusion des résultats en utilisant un RNN Forward et un RNN Backward, puis en moyennant arithmétiquement (pour la régression) ou géométriquement (pour la classification) les résultats. La structure virtuelle ainsi obtenue sera nommée Merge. Cependant, cette procédure présente des limites lorsque les données d'entrée ne sont pas indépendantes.

2.2. Réseaux de neurones récurrents bidirectionnels (BRNN)

2.2.1. Construction

L'idée de départ est de séparer une couche de neurones en deux parties :

- la première partie est responsable de la propagation dans le sens du temps (RNN Forward)
- la seconde partie est responsable de la propagation dans le sens inverse des temps (RNN Backward)

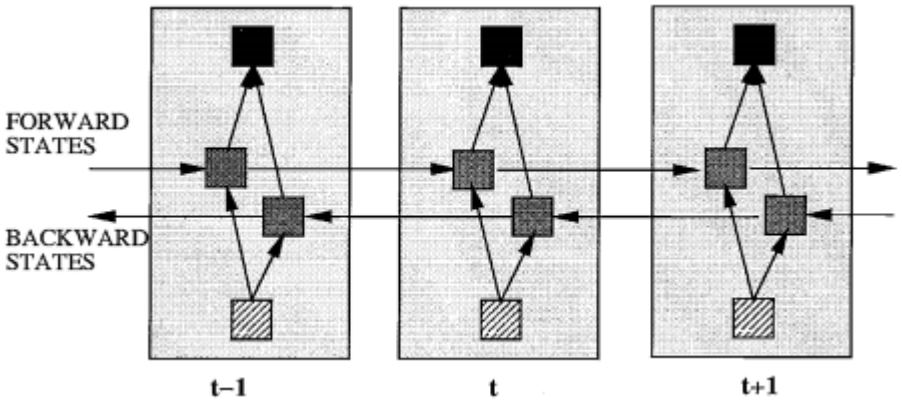

Figure 3 : Structure d'un BRNN pour 2 périodes de temps

La fonction objectif sera ainsi optimisée conjointement vis-à-vis du passé et du futur, ce qui diffère de la structure Merge où on conserve la moyenne de deux fonctions objectifs optimisées séparément.

2.2.2. Entraînement

La procédure d'entraînement résumée est la suivante :

1) Sens direct : prédiction du BRNN à partir des données
2) Sens indirect : prédiction du BRNN à partir des données
3) Optimisation de la fonction objectif conjointe et mise à jour des poids

2.3. Résultats

2.3.1. Avec des données artificielles

La structure BRNN est ici mise en compétition avec les structures RNN Forward, RNN Backward et Merge pour les problèmes de régression et de classification. Les données utilisées sont d'abord artificielles : on génère un vecteur de 10 000 nombres tirés aléatoirement dans l'intervalle [0, 1] qu'on utilise comme entrée. La sortie souhaitée est la moyenne pondérée des entrées entre t - 10 et t + 20, où les poids décroissent linéairement avec l'éloignement vis-à-vis de t :

$$y(t) = \frac{1}{10} \sum_{\Delta t=-10}^{-1} x(t+\Delta t) \cdot \left(1 - \frac{|\Delta t|}{10}\right) + \frac{1}{20} \sum_{\Delta t=0}^{19} x(t+\Delta t) \cdot \left(1 - \frac{|\Delta t|}{20}\right).$$

Pour le problème de classification, les classes sont les sorties supérieures à 0,5 d'une part, celles inférieures d'autre part.

Les structures RNN Forward, RNN Backward et Merge sont testées pour plusieurs valeurs du délai.

- Problème de régression :

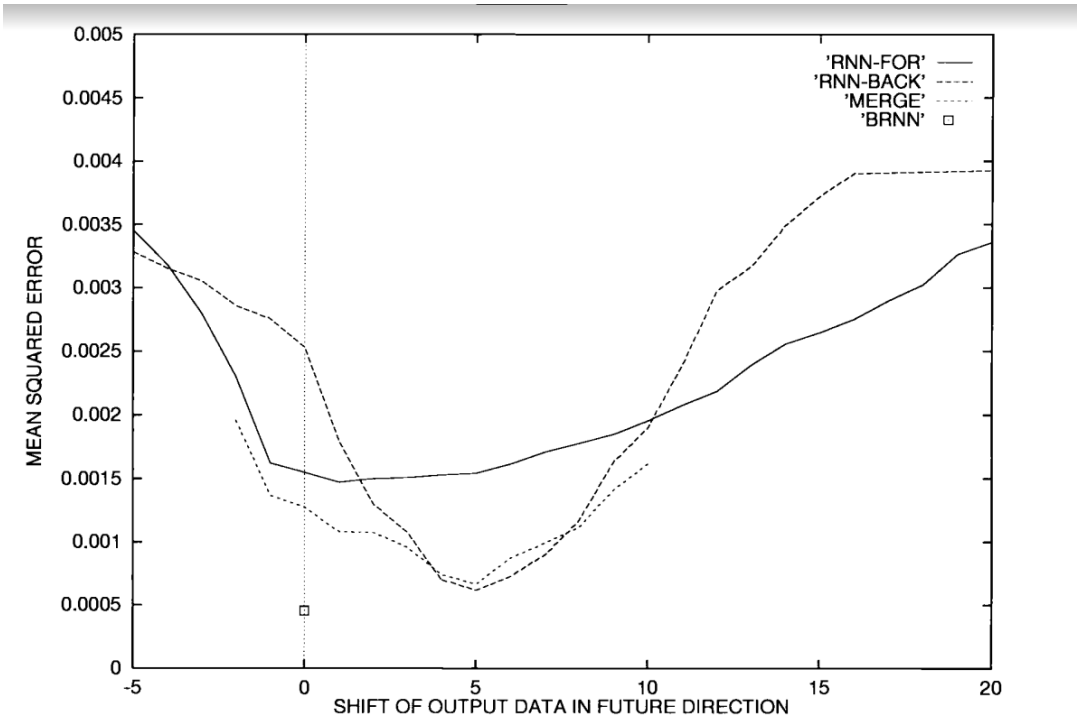

Figure 4 : Résultats obtenus pour le problème de régression

- Problème de classification :

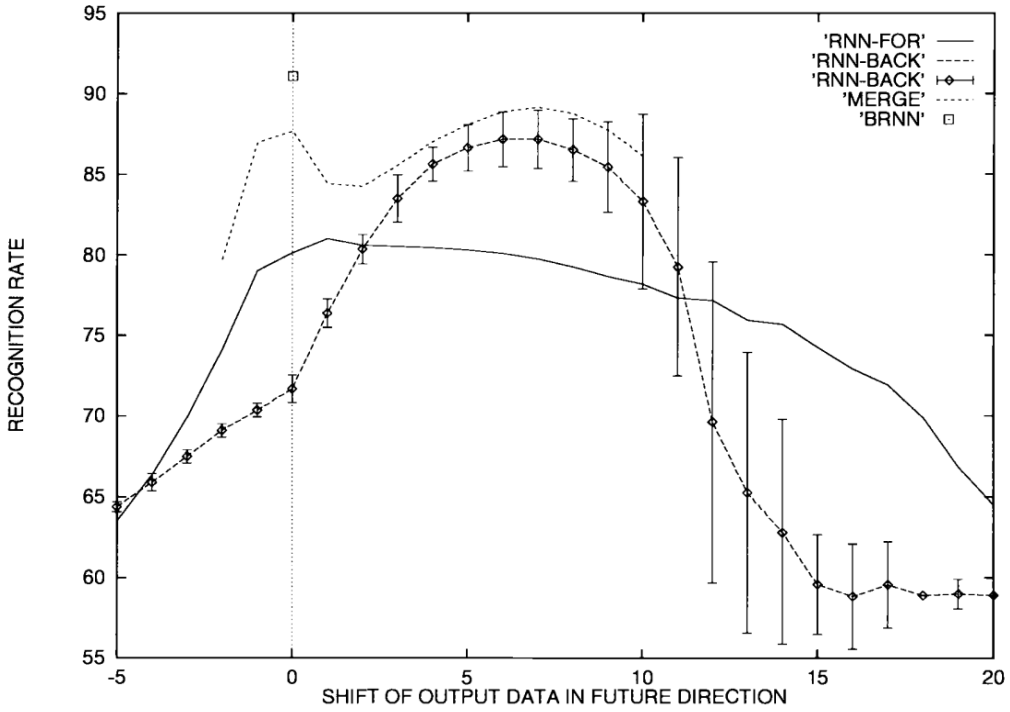

Figure 5 : Résultats obtenus pour le problème de classification

La structure BRNN est ainsi strictement plus efficace que les autres structures pour les deux problèmes, et ce quel que soit la valeur du délai.

2.3.2. Avec des données réelles

Cette fois-ci, les données utilisées sont 6300 phrases prononcées par 630 personnes. Près de 4000 phrases sont allouées à l'entraînement, le reste sert au test.
Il est tout d'abord nécessaires de convertir ces fichiers audios en suites de nombres. Il a été choisi ici de simuler de telle manière une distribution gaussienne, car celles-ci donnent les meilleurs résultats. On n'étudie ici que le problème de classification : le BRNN obtient une nouvelle fois les meilleurs résultats.

TIMIT Phoneme Classification Results for Full Training and Test Data Sets with ≈ 13 000 Parameters

Structure	Rec-Rate % TRAIN 61 (39)	Rec-Rate % TEST 61 (39)
MLP-1 (1 segment)	61.32 (70.20)	59.67 (68.95)
MLP-3 (3 segments)	68.37 (75.74)	65.69 (73.48)
MLP-5 (5 segments)	66.97 (74.60)	64.32 (72.35)
FOR-RNN	65.42 (74.27)	63.20 (72.51)
BACK-RNN	64.57 (72.83)	61.91 (70.94)
FOR-RNN (1 delay)	68.45 (75.37)	65.83 (73.00)
FOR-RNN (2 delay)	65.97 (73.03)	63.27 (70.77)
MERGE (FOR+BACK)	66.94 (75.01)	65.28 (73.73)
BRNN	70.73 (77.33)	68.53 (75.48)

Figure 6 : Résultats pour des données réelles

2.3.3. Utilisation d'un modèle bidirectionnel modifié

Lorsqu'on s'intéresse au problème de la reconnaissance vocale, il est important de noter que le problème de classification est légèrement différent : plutôt que de chercher à estimer la probabilité d'une classe k à l'instant t, on va rechercher la probabilité d'occurrence d'une séquence de classes interdépendantes (les mots).

Une structure efficiente vis-à-vis de ce problème est la suivante : un BRNN légèrement modifié, où plutôt que d'envoyer les données au neurone de couche suivante associé au temps t, les neurones de la chaîne Forward l'enverront au neurone de la couche suivante de temps t+1, et ceux de la chaîne Backward l'enverront à celui de temps t-1. De plus, on introduit un décalage L qui coupe les L premières (dernières) données : on obtient un BRNN modifié Forward (Backward).

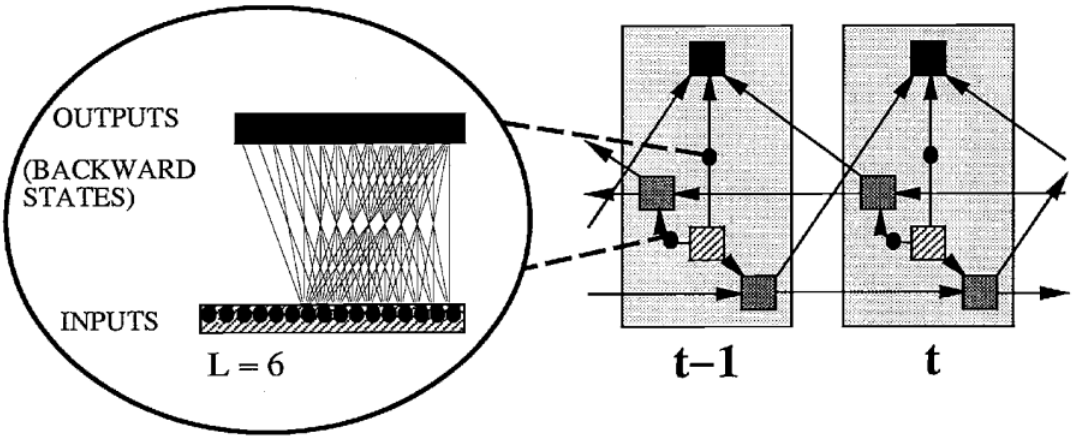

Figure 7 : BRNN modifié Forward

En utilisant cette structure pour la base de données audio, on remarque que les résultats sont encore meilleurs, et encore plus si l'on conserve la moyenne arithmétique ou géométrique associée à deux BRNN modifiés de même décalage dans les sens différents (structure virtuelle BRNN mod. merged).

CLASSIFICATION RESULTS FOR FULL TIMIT
TRAINING AND TEST DATA WITH 61 (39) SYMBOLS

Structure	Rec-Rate % TRAIN 61 (39)	Rec-Rate % TEST 61 (39)
forward. mod. BRNN	79.11 (84.42)	72.70 (79.08)
backward. mod. BRNN	79.38 (83.27)	72.74 (77.44)
both merged, lin.	83.57 (87.17)	77.53 (82.11)
both merged, log.	83.89 (87.45)	77.75 (82.38)

Figure 8 : Résultats pour des données réelles

3. Lien avec l'article Deep Speech

Le modèle utilisé dans l'article Deep Speech est un réseau de neurones à 5 couches, dont :

- les trois premières ne sont pas récurrentes
- la quatrième est récurrente bidirectionnelle, c'est-à-dire divisé en deux parties, l'une où le transfert d'informations est réalisé dans le sens du temps, l'autre où il l'est dans le sens inverse (BRNN)
- la cinquième n'est pas récurrente, mais prend en entrée les résultats des deux parties de la couche 4

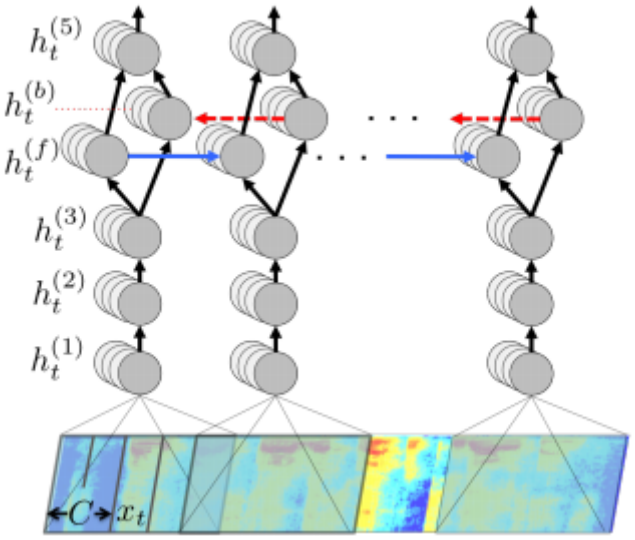

Figure : Structure du RNN de Deep Speech

Cette structure est en effet plus efficace qu'une structure unidirectionnelle comme montré précédemment : elle ajoute de la cohérence globale à l'entraînement. Cependant, celle-ci n'a été appliquée que sur une couche. Elle crée en effet certaines difficultés de parallélisation du calcul.

4. Deux articles d'ouverture

4.1. Neural Networks: A Pattern Recognition Perspective (Bishop, 1996)

Dans les applications pratiques des réseaux de neurones, le prétraitement des données, c'est-à-dire la manière dont on transforme les données (un signal audio dans le cas de Deep Speech) en série de nombres, est l'un des facteurs de performance les plus importants. Ce travail permet en effet d'atténuer l'effet de la "malédiction de la dimension" en réduisant le nombre d'entrées du système. Celles-ci peuvent en effet être combinées, de manière linéaire ou non. Même si une partie de l'information est perdue, cela est plus que compensé par les bénéfices apportés par un dimensionnement plus faible de l'entrée.

Un autre avantage du prétraitement des données est qu'il permet d'apporter une connaissance a priori du système (approche bayésienne), s'ajoutant ainsi aux données pures et améliorant les résultats. Par exemple, pour la reconnaissance d'un chiffre, la localisation du chiffre dans l'image n'apporte pas d'information pertinente : il ne faut retenir que les données indépendantes de la position, ce qui permet d'obtenir des résultats plus performants.

Enfin, le prétraitement des données permet d'éliminer les lacunes des fichiers de données (comme un mot labélisé NONE pour Deep Speech) qui peuvent causer des problèmes lors de l'entraînement et de la prédiction.

4.2. Several Improvements to a Recurrent Error Propagation Network Phone Recognition System (Robinson, 1991)

Les réseaux de neurones donnent de bonnes performances pour la tâche de reconnaissance vocale, et ce indépendamment du parleur. Il est cependant possible d'apporter plusieurs améliorations à ces structures :

- complexification du prétraitement des données (comme vu dans l'article précédent)
- utilisation d'une structure proche d'une gaussienne pour l'information d'entrée, après prétraitement
- normalisation de l'information de sortie
- pondération équivalente pour chaque son cible
- modification de représentation de la sortie pour éliminer les erreurs de quantification
- recyclage de données lors de l'entraînement

Ces changements permettent de diminuer le nombre d'inconnues du système, et ainsi de diminuer l'effet de la "malédiction de la dimension". La prise en compte de ces amélioration permet de diminuer le taux d'erreur de 16% (de 36,5 à 30,7%).