Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

A review of popular spatiotemporal pattern recognition techniques

F. Rhéaume
DRDC Valcartier

Defence R&D Canada – Valcartier
Technical Memorandum
DRDC Valcartier TM 2008-315
March 2009

Canada

A review of popular spatiotemporal pattern recognition techniques

F. Rhéaume
DRDC Valcartier

Defence R&D Canada – Valcartier

Technical Memorandum

DRDC Valcartier TM 2008-315

March 2009

Principal Author

F. Rhéaume

Approved by

Éloi Bossé
Head/A Section

Approved for release by

C. Carrier
Chief Scientist

Abstract

Recognition is an important process of situation analysis in military command and control systems. This memorandum focuses on recognition tasks that must rely on data that evolve in time. This type of problem is known as spatiotemporal pattern recognition. The document provides an objective review of supervised learning approaches for spatiotemporal data. The list is non-exhaustive but provides a variety of popular methods presented in the literature. The spatiotemporal pattern recognition methods presented in this review are sliding windows, hidden Markov models and conventional recurrent neural networks, including different training methods such as backpropagation through time and real-time recurrent learning. It also covers Long-Short Term Memories and the reservoir computing approach, where Echo states networks and Liquid state machines are presented as the main reservoir methods. The review explains and discusses the concepts and algorithms behind the different techniques.

Résumé

La reconnaissance est une fonction importante du processus d'analyse de la situation pour les systèmes militaires de commandement et contrôle. Ce memorandum traite du cas où les données d'intérêt évoluent dans le temps, c'est-à-dire la reconnaissance de structures spatiotemporelles. Ce document fournit une revue objective des approches d'apprentissage supervisées pour des données qui évoluent dans le temps. La liste est non exhaustive mais inclut une série de méthodes populaires présentées dans la littérature. Les méthodes de reconnaissance de structures spatiotemporelles présentées dans cette revue sont les fenêtres glissantes, les modèles cachés de Markov et les réseaux de neurones récurrents conventionnels, avec différentes méthodes d'entraînement telles que la rétropropagation à travers le temps et l'apprentissage récurrent en temps réel. La revue couvre également les mémoires à long-court terme ainsi que l'approche par réservoir, où les réseaux d'états avec écho et les machines d'état liquide sont présentés en tant que méthodes par réservoir principales. Les concepts et les algorithmes derrière les différentes techniques sont expliqués et discutés.

This page intentionally left blank.

Executive summary

A review of popular spatiotemporal pattern recognition techniques

F. Rhéaume; DRDC Valcartier TM 2008-315; Defence R&D Canada – Valcartier; March 2009.

Military command and control systems operate in some time changing environnements. In order to make the good decisions and to take adequate actions, military command and control systems rely on a situation analysis process. The situation analysis process evaluates the evolving scene with respect to the military objectives. Very often, this evaluation must consider data collected over a given amount of time and output some results based on the temporal evolution of the collected data. This is different from relying only on static data of a specific time.

One of the principal areas of situation analysis is recognition, that is about identifying something as having been previously observed and learned. When a recognition task must rely on the evolution in time of some given quantities, it is said to be in the domain of spatiotemporal pattern recognition. The problem of spatiotemporal pattern recognition is the theme of this memorandum. The memorandum aims at providing an objective review of supervised learning approaches for data that variate in time. The list is non-exhaustive but provides a variety of popular methods presented in the literature, from the simplest ones to the more complex and recent ones.

The spatiotemporal pattern recognition methods presented in this review are sliding windows, hidden Markov models and conventional recurrent neural networks, including different training methods such as backpropagation through time and real-time recurrent learning. It also covers long-short term memories and the reservoir computing approach, where Echo states networks and Liquid state machines are presented as the main reservoir methods. The review explains and discusses the concepts and algorithms behind the different techniques.

This page intentionally left blank.

Sommaire

A review of popular spatiotemporal pattern recognition techniques

F. Rhéaume ; DRDC Valcartier TM 2008-315 ; Recherche et développement pour la défense Canada - Valcartier ; mars 2009.

Les systèmes militaires de commandement et contrôle fonctionnent dans des environnements qui évoluent dans le temps. Afin de prendre les bonnes décisions et les actions appropriées, les systèmes militaires de commandement et contrôle se fondent sur un processus d'analyse de la situation. Le processus d'analyse de la situation évalue la scène en évolution en fonction des objectifs militaires. Très souvent, cette évaluation doit tenir compte des données obtenues sur une période de temps donnée et fournir une réponse basée sur l'évolution temporelle des données observées. Ceci contraste avec l'évaluation de données statiques pour des instants spécifiques.

Une des principales fonctions de l'analyse de la situation est la reconnaissance, qui consiste à identifier quelque chose à partir d'observations antérieures et en se basant sur un apprentissage préalable. Lorsque qu'une tâche d'identification doit tenir compte de l'évolution temporelle de quantités particulières, celle-ci est dite du domaine de reconnaissance spatiotemporel. Le problème de la reconnaissance des structures spatiotemporelles est le thème de ce memorandum. Celui-ci vise à fournir une revue objective des approches d'apprentissage supervisées pour des données qui évoluent dans le temps. La liste est non exhaustive mais inclut une série de méthodes populaires présentées dans la littérature, allant des plus simples aux plus complexes et plus récentes.

Les méthodes de reconnaissance des structures spatiotemporelles présentées dans cette revue sont les fenêtres glissantes, les modèles cachés de Markov et les réseaux de neurones récurrents conventionnels, avec différentes méthodes d'entraînement telles que la rétropropagation à travers le temps et l'apprentissage récurrent en temps réel. La revue couvre également les mémoires à long-court terme ainsi que l'approche par réservoir, où les réseaux d'états avec écho et les machines d'état liquide sont présentés en tant que méthodes par réservoir principales. Les concepts et les algorithmes derrière les différentes techniques sont expliqués et discutés.

This page intentionally left blank.

Table of contents

List of figures

This page intentionally left blank.

1 Introduction

This memorandum covers the problem of spatiotemporal pattern recognition. It aims at reviewing the principal approaches for supervised learning on data that variate in time. The list is non-exhaustive but provides a variety of popular methods presented in the literature, from the simplest ones to the more complex and recent ones.

Spatiotemporal pattern recognition may be exploited in different tasks of Situation Analysis (SA) as part of military Command and Control (C2) systems. Surveillance, for example, usually necessitates different recognition functions. Object recognition, behavior recognition or situation recognition, for example, are among those functions that most probably should deal with temporal data.

In Chapter 2, the classical spatial supervised learning is defined. Chapter 3 then defines the spatiotemporal case. Chapter 4 presents one of the most intuitive spatiotemporal pattern recognition method, the sliding window. In Chapter 5, Hidden Markov Models (HMMs) are discussed as a probabilistic approach. Thereafter, recurrent neural network (RNN) methods are presented. Chapter 6 presents some conventional recurrent neural networks along with training algorithms. It also introduce Long-Short Term Memory (LSTM), a more complex recurrent network that improves learning for long time lags. In Chapter 7, reservoir computing, a recent approach that avoids the problems of training RNNs, is presented. The document ends with some closing remarks in Chapter 8.

This page intentionally left blank.

2 Classical (spatial) supervised learning problem

The classical supervised learning problem deals with non-temporal data. It is about the prediction of classes of new observations given a set of training observations[1]. The work of [1] states the optical character recognition example where an observation corresponds to an image of a hand-written character and a class corresponds to a letter of the alphabet. Let $\boldsymbol{x}$ denote an observation and y its class. A training example is a pair $(\boldsymbol{x}, y)$ consisting of an object and its associated class label. The training examples are preferably assumed independently and identically distributed from the joint distribution $P(\boldsymbol{x}, y)$. A classifier is a function h that maps objects to classes based on a training set of N examples. The learning process tries to find the function h that best predicts the class $y = h(\boldsymbol{x})$ of each test object $\boldsymbol{x}$. A test object is a new object that is not drawn from the training set.

[1] The problem could also include regression, but the work in [1] concentrates on classification.

This page intentionally left blank.

3 Spatiotemporal supervised learning problem

The following sections present different representations of spatiotemporal supervised learning. These representations mainly depend on the learning task, that is what kinds of outputs the learning system must produce based on the inputs. Sequential learning, time-series prediction, sequence classification and a more general spatiotemporal pattern recognition representation are discussed.

3.1 Sequential learning

Suppose that, instead of being independently and identically distributed as in Chaopter 2, the observations of some states of an environnement are ordered into sequences of pairs $(\boldsymbol{x}, y)$ and that the nearby pairs are likely to be related to each other, *i.e* correlated. Such sequences of pairs $(\boldsymbol{x}, y)$ are called sequential patterns. Learning the correlation among observations can improve the accuracy of a classifier. For example, in optical character recognition of English texts, if a classifier outputs the class label $y = Q$ for an observed letter, then the class label of the next observation is very likely to be U. In [1], the sequential supervised learning problem is defined as follows. Let $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ be a set of N training examples. Each example is a pair of sequence $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ of length T_i, where $\boldsymbol{x}_i = (\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, ..., \boldsymbol{x}_{i,T_i})$ and $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, ..., y_{i,T_i})$. The goal is to construct a classifier h that can output a class label sequence $\boldsymbol{y} = h(\boldsymbol{x})$ given an observation sequence $\boldsymbol{x}$.

3.2 Time-series prediction

The goal is to predict the next element y_{t+1} of a sequence $(y_1, y_2, ..., y_t)$. As a first case, a classifier can only rely on the sequence $(y_1, y_2, ..., y_t)$ to predict y_{t+1}. Note that y can also be a vector $\boldsymbol{y}$. The task is then to predict the next elements $\boldsymbol{y}_{t+1}$ of a collection of parallel time series $(\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_t)$. A second case is when the prediction of $\boldsymbol{y}_{t+1}$ can rely on a set of other variables $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t)$ as more to the sequence $(y_1, y_2, ..., y_t)$.

3.2.1 Difference between time-series prediction and sequential supervised learning

In sequential supervised learning, the prediction of the y value uses the whole observation sequences $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t)$. Time-series prediction may only relate on a segment $(\boldsymbol{y}_{t-d}, ...\boldsymbol{y}_{t-1}, \boldsymbol{y}_t)$ of the whole sequence $(\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_t)$ to predict the next values $\boldsymbol{y}_{t+1}$. Also, in time-series analysis, the true y values are known up to time t, while in sequential supervised learning the y values are not given and must all be predicted.

3.3 Sequence classification

Sequence classification associates a single class label y to a whole observation sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t)$. This contrasts to sequential supervised learning where a class label is output for each observation of the input sequence.

As mentioned in [1], sequential supervised, time-series prediction and sequence classification are closely related problems. Depending on the application, these problems can sometimes be mixed. For example, the sequence classification problem of recognizing the class of a sequence of observations ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_t$) could first be decomposed into many sequential supervised problems that assign a class to each observation. Then, the resulting class sequence could be used to infer the unique class for the whole sequence. Also, the time-series analysis problem could be solved taking a sequential learning approach, where only a segment of the sequence ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_t$) is considered instead of the whole sequence.

3.4 Spatiotemporal pattern recognition problem

The problem described in [2] is about recognizing spatiotemporal patterns from a signal x and among classes of patterns. A spatiotemporal pattern α is defined as a tuple $(\alpha_s, \alpha_e, \{x(t)\}|_{\alpha_s}^{\alpha_e})$ where α_s is the start time of the pattern, α_e is the end time of the pattern and $\{x(t)\}|_{\alpha_s}^{\alpha_e}$ is the signal x between times α_s and α_e. A recognizer is a function g that returns a set (t_s, t_e, c), where c is the class of the pattern and where t_s and t_e are the start and end times of the pattern, respectively. According to this, the recognition of a spatiotemporal pattern α in a signal x is represented by $(\alpha_x, \alpha_e, c) \in g(x)$.

4 Sliding window method

The sliding window method given in [1] assumes the pair of sequence $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ of length T_i, where $\boldsymbol{x}_i = (\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, ..., \boldsymbol{x}_{i,T_i})$ and $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, ..., y_{i,T_i})$. The method simply uses the sequential data as an input to classical supervised learning algorithms. This is done by windowing the sequential input data into segment of width w. A sliding window classifier h_w outputs an individual value y for each segment $\boldsymbol{x}$. Suppose $d = (w-1)/2$ is the half width of the window, h_w predicts $y_{i,t}$ using the window $\boldsymbol{x}_i = (x_{i,t-d}, x_{i,t-d+1}, ..., x_{i,t}, ..., x_{i,t+d-1}, x_{i,t+d})$. This is shown in Figure 1. A data set i consisting of the pair of sequential data $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ can be used to train a classifier h_w, where the window converts the sequential data into input vectors onto which standard supervised learning algorithms are applied for training.

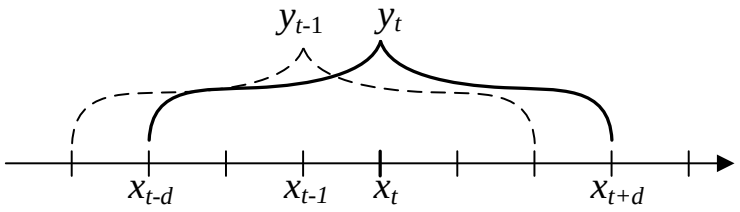


Figure 1: The sliding window method described in [1]

The sliding window method is also known as "tapped delay line memory". The idea behind tapped delay line memory is the same: to make the temporal domain spatial. Figure 2 shows a tapped delay line example, where inputs are put in a delayed buffer and discretely shifted as time passes.

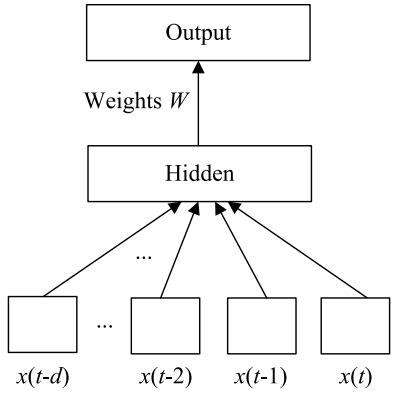


Figure 2: Tapped delay line feedforward network presented in [3]

Although the sliding window approach allows the use of any classical supervised learning algorithm, such as nearest neighbor methods or feedforward neural networks for example, its performance are limited in many applications, this approach has some drawbacks. One

of them is buffering of the input, where the method must have a way to read the buffer at some appropriate times. Also, the spatial vector as a fixed and limited length w, which provides the learning algorithm with inputs of constant time length whatever the nature of the signal. Furthermore, this approach does not make the difference between the temporal positions of the signal, since the signal is shifted into a spatial vector. For example, suppose the two following spatial vectors

$$[0, 1, 1, 1, 0, 0, 0, 0, 0] \tag{1}$$

and

$$[0, 0, 0, 1, 1, 1, 0, 0, 0]. \tag{2}$$

From the temporal dimension, these two sequences seem very similar, where one sequence is just shifted in time compared to the other. But from the spatial dimensions, these two vectors are distant one from another. Therefore, this would make it harder for a spatial learning algorithm to make any correspondance between these vectors.

Also, in some applications the method could take advantage of the correlations between nearby y_t values, in the case where there is significant dependance between them. Recurrent sliding windows are a way to manage this correlation. The difference between a conventional sliding window method and a recurrent one is that the later feeds back the predicted $\bar{y}_t$ as an input to predict the next value $\bar{y}_{t+1}$. Then, the input vector a mix of x and y values. The recurrent sliding window method presented in [1] uses the most recent d predictions, ($\bar{y}_{t-d}$, $\bar{y}_{t-d+1}$, ..., $\bar{y}_t$, ..., $\bar{y}_{t-1}$), along with the sliding window ($x_{i,t-d}$, $x_{i,t-d+1}$, ..., $x_{i,t}$, ..., $x_{i,t+d-1}$, $x_{i,t+d}$) to predict y_t. To train such a method, two choices are possible about the $\bar{y}_t$ inputs. The first choice is to train a non-recurrent classifier, using the conventional sliding window described above, and use resulting $\bar{y}_t$ values as inputs. The second choice of inputs is to use the real y_t values of the training data set. In that case, each training sample ($x_{i,t-d}$, $x_{i,t-d+1}$, ..., $x_{i,t}$, ..., $x_{i,t+d-1}$, $x_{i,t+d}$, y_{t-d}, y_{t-d+1}, ..., y_t, ..., y_{t-1}) can be constructed independently from each other.

An example of recurrent sliding window is presented in [4, 5], where a feedforward network has as inputs a window of past inputs and a window of its own past outputs. Such a network is illustrated in Figure 3.

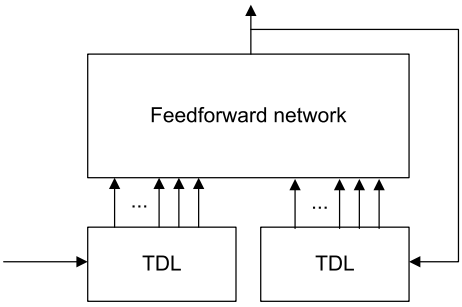


Figure 3: Tapped delay line recurrent network

This page intentionally left blank.

5 Hidden Markov Model

The Hidden Markov Model (HMM) is a statistical tool for modeling signals produced by an underlying process. The signals are known as the observable outputs of the process. As stated in [6], a process can be stationary or non-stationary. The statistical properties of a stationary process are constant in time while they are variable in time for non-stationary process.

The main interest behind HMM is signal modeling. Signal models can be divided into two types: deterministic and statistical. Deterministic models are those that exploit some known specific properties of the signal and for which specification of the model is generally straightforward. For example, the model may be described by a few parameters such as amplitude, frequency or phase of a sine wave for examples. However, statistical models characterize only the statistical properties of the signal. They assume that the signal can be characterized as a parametric random process. Gaussian processes, Poisson processes, Markov processes, and hidden Markov processes are examples of statistical models. In a Markov process, the state is directly observable, and therefore the state transition probabilities are the only parameters. In a hidden Markov process, the state is not directly observable, only variables influenced by the state are observable.

5.1 Discrete (observable) Markov processes

Suppose a system of N distinct states S_1, S_2, ..., S_N. The state of the system changes at regular discrete times and according to state transition probabilities. In a Markov process, the state probability is only described according to the current state and the previous state of the system. Thus the probability of the system being at state S_j at time t if it was at state S_i at time $t-1$ is described as:

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i],\ i \leq N,\ j \leq N \tag{3}$$

where q_t denotes the state of the system at time t and where the a_{ij} values are called the state transition probabilities or coefficients, that obeys

$$a_{ij} \geq 0 \tag{4}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{5}$$

Because the output of the process is the state of the system, the process is called an observable Markov model. In [6], a simple 3-state Markov model is presented. The process consist of the weather of the day and has the following states:

$$S_1 : \text{rain} \tag{6}$$

$$S_2 : \text{cloudy} \tag{7}$$

$$S_3 : \text{sunny} \tag{8}$$

The transition matrix for the three above states is

$$\boldsymbol{A} = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \tag{9}$$

Because the states are directly observable, calculating the probability $P(O|\text{Model})$ of having a specific observation sequence O is easy. Suppose the following 7-day weather observation sequence "sun-sun-rain-rain-sun-cloudy-sun", that can be represented as $O = \{S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3\}$. The probability of O is

$$\begin{aligned} P(O|Model) =& P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|\text{Model}] \\ =& P[S_3] \cdot P[S_3|S_3] \cdot P[S_3|S_3] \cdot P[S_1|S_3] \cdot P[S_1|S_1] \cdot P[S_3|S_1] \\ & \cdot P[S_2|S_3] \cdot P[S_3|S_2] \\ =& \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ =& 1.536 \times 10^{-4} \end{aligned} \tag{10}$$

with

$$\pi_i = P[q_1 = S_i], \;\; 1 \leq i \leq N \tag{11}$$

5.2 Hidden Markov models

A hidden Markov model is a model where the observations are probabilistic functions of the states. As opposed to conventional Markov models, the states of the underlying process are not observable. However, observations are produced from a different stochastic process that is conditioned on the states.

Formally an HMM is made of

1. a set of N states $S = S_1, S_2, ..., S_N$ where any state can be reached from any other state.

2. a set of M observation symbols $V = \{v_1, v_2, ..., v_3\}$. They correspond to the physical output of the system being modeled.

3. a state transition probability distribution $\boldsymbol{A} = \{a_{ij}\}$ where

$$a_{ij} = P[q_t = S_j|q_{t-1} = S_i], \; i \leq N, \; j \leq N \tag{12}$$

4. an observation symbol probability distribution when the system is in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \text{ at } t|q_t = Sj], \;\; 1 \leq j \leq N, \; 1 \leq k \leq M \tag{13}$$

5. an initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P[q_1 = S_i], \;\; 1 \leq i \leq N \tag{14}$$

To extent the previous weather example, suppose the following observation set $V = \{\text{warm, humid, windy}\} = \{v_1, v_2, v_3\}$ and the following symbol probability distribution

$$\boldsymbol{B} = \begin{bmatrix} 0.1 & 0.1 & 0.6 \\ 0.8 & 0.5 & 0.2 \\ 0.1 & 0.4 & 0.2 \end{bmatrix} \tag{15}$$

where $B(k, j)$ gives the probability of observing v_k if the state is S_j. For instance, the probability of observing $v_1 =$ "warm" when the state is $S_3 =$ "sunny" is 0.6.

To denote a specific HMM, the symbol λ is used to represent the model with the three probability measures A, B and π:

$$\lambda = (A, B, \pi). \tag{16}$$

5.3 Making use of hidden Markov models

Given a model λ, real-world applications necessitate different manipulations with the model λ. In [6], these manipulations are divided into three problems, as follows:

> **Problem 1:** Given the observation sequence $O = O_1, O_2, ..., O_T$ and a model $\lambda = (A, B, \pi)$, compute the probability $P(O|\lambda)$ of the observation sequence given the model.
>
> **Problem 2:** Given the observation sequence $O = O_1, O_2, ..., O_T$ and a model $\lambda = (A, B, \pi)$, find the corresponding state sequence $Q = q_1, q_2, ..., q_T$ that is optimal. Optimality may have different meanings that will be discussed in the next paragraphs.
>
> **Problem 3:** Adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$.

The three problems and some of their solutions are presented in the next subsections.

5.3.1 Computing the probability of an observation sequence (problem 1)

Problem 1 has relevance for the evaluation of a model's suitability to an observation sequence. It is useful for the case where the task is to choose the most appropriate model among a set of many different models.

Although there exists more efficient methods for calculating the probability of an observation sequence, such as the Forward-Backward Procedure [6], the simplest solution is presented below to better understand the problem.

Recall the problem of calculating the probability $P(O|\lambda)$ of an observation sequence $O = O_1, O_2, ..., O_T$ given a model $\lambda = (A, B, \pi)$. Considering a fixed state sequence $Q =$

q_1, q_2, ..., q_T and assuming independent observations, the probability of the observation sequence O for the state sequence Q is given by

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda), \tag{17}$$

$$= b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T). \tag{18}$$

Moreover, the probability of the state sequence Q is

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \tag{19}$$

The probability of having both the observation sequence O and the state sequence Q is

$$P(O,Q|\lambda) = P(O|Q,\lambda)\, P(Q,\lambda) \tag{20}$$

To obtain $P(O|\lambda)$, the joint probability of (20) must be summed over all possible state sequences:

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda)\, P(Q,\lambda) \tag{21}$$

$$= \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T). \tag{22}$$

This solution necessitates $2T \cdot N^T$ calculations, which grows rapidly with the number of observations and is impracticable unless there is a very small numbers of states and observations. Hence, it is only shown for an academic usage.

5.3.1.1 Examples

Using the weather example and the matrix A and B defined in (9) and (15), respectively, some different observation probabilities will be computed in Matlab as an exercise. Also the following initial state distribution will be used:

$$[\pi(\text{rain}),\, \pi(\text{cloudy}),\, \pi(\text{sunny})] = [\pi(S_1),\, \pi(S_2),\, \pi(S_3)] = [1/3,\, 1/3,\, 1/3] \tag{23}$$

Suppose one wants to know the probability of each possible observation, warm, humid or windy, for the next day (independently of the current day). This gives

$$P([\text{warm}]|\lambda) = 0.27 \tag{24}$$

$$P([\text{humid}]|\lambda) = 0.5 \tag{25}$$

$$P([\text{windy}]|\lambda) = 0.23 \tag{26}$$

using (22). Thus, there are more chances to observe a humid day. This explains from the fact that the second row of the observation distribution matrix B, which corresponds to the humid observation for each state, has the highest values.

Next, suppose that the question is to find the probabilities of the different observation scenarios for two consecutive days. According to (22), here are a few possibilities and their corresponding probability

$$P([\text{warm, warm}]|\lambda) = 0.12 \tag{27}$$
$$P([\text{humid, humid}]|\lambda) = 0.24 \tag{28}$$
$$P([\text{windy, windy}]|\lambda) = 0.06 \tag{29}$$
$$P([\text{warm, humid}]|\lambda) = 0.09 \tag{30}$$
$$P([\text{humid, warm}]|\lambda) = 0.13 \tag{31}$$
$$P([\text{warm, windy}]|\lambda) = 0.06 \tag{32}$$

Note here that the results depend on the state transition matrix A, which was not the case when only one day is considered.

5.3.2 Finding the optimal state sequence (problem 2)

Problem 2 is about trying to find the most relevant state sequence for an observation sequence and in terms of an optimality criterion. This raises another problem of choosing an appropriate optimality criterion. As stated in [6], the most widely used criterion is the one that maximizes $P(Q|O, \lambda)$, that is to find the single best state sequence. A popular algorithm that solves this problem is the Viterbi algorithm, a backtracking algorithm that finds the best state sequence, $Q = q_1, q_2, ..., q_T$, given an observations sequence, $O = O_1, O_2, ..., O_T$. The algorithm needs the calculation of the highest probability along a single sequence up to time t, $\delta_t(i)$, which accounts for the first t observations and ends in state S_i

$$\delta_t(i) = \max_{q_1, a_2, \cdots, q_{T-1}} P[q_1\, q_2\, \cdots\, q_t = i, O_1, O_2, ..., O_t|\lambda] \tag{33}$$

Iteration in time for this probability is given by

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}). \tag{34}$$

To be able to backtrack the state sequence, the argument of $\delta_t(j)$ needs to be saved for each t and j in an array represented by $\psi_t(j)$.

The algorithm has the four following steps

1. Initialization of $\delta_1(i)$ and $\psi_1(i)$:

$$\delta_1(i) = \pi_i b_i(O_1), \ 1 \leq i \leq N \tag{35}$$
$$\psi_1(i) = 0, \qquad 1 \leq i \leq N \tag{36}$$

2. Recursion:

$$\delta_t(j) = [\max_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T, \ 1 \leq j \leq N \tag{37}$$
$$\psi_t(j) = [\text{argmax}_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij}], \qquad 2 \leq t \leq T, \ 1 \leq j \leq N. \tag{38}$$

3. Termination:

$$P^* = \max_{1 \leq i \leq N} \delta_T(i), \tag{39}$$

$$q_T^* = \text{argmax}_{1 \leq i \leq N} \, \delta_T(i). \tag{40}$$

4. State sequence backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, \, T-2, \cdots, \, 1. \tag{41}$$

5.3.2.1 Examples

Using the weather data of Subsection 5.3.1.1, the optimal state sequence for some observations sequences will be computed using the Viterbi algorithm.

Consider the single observation case, where the goal is to find the optimal state of the weather. The warm, humid and windy observations yields, respectively:

$$\text{warm} = \text{sunny} \tag{42}$$

$$\text{humid} = \text{rain} \tag{43}$$

$$\text{windy} = \text{cloudy} \tag{44}$$

Those results are straightforward since they only depend on the maximum probability of the observations over all possible states, found in matrix B.

Suppose the more complicated task of finding the optimal state sequences for many consecutive observations. For example, what is the optimal weather state sequence if the observation sequence is "humid-humid-humid-windy". Here is the resulting optimal sequence as well as other examples:

$$\text{humid-humid-humid-windy} = \text{rain-rain-rain-rain} \tag{45}$$

$$\text{windy-humid} = \text{cloudy-cloudy} \tag{46}$$

$$\text{warm-humid-humid} = \text{sunny-rain-rain} \tag{47}$$

$$\text{windy-humid-warm} = \text{sunny-sunny-sunny} \tag{48}$$

$$\text{humid-humid-windy-windy-warm} = \text{rain-rain-sunny-sunny-sunny} \tag{49}$$

$$\text{warm-humid-windy-windy-warm} = \text{sunny-sunny-sunny-sunny-sunny} \tag{50}$$

The results show a tendency over sunny days when a warm observation is present, even though it is not the dominating observation of the sequence. The main reason for this is that the "sunny" state has the highest transition values to itself. Summing the columns of the state transition matrix gives:

$$[0.7, 1, \, 1.3], \tag{51}$$

which means that the state which is more transited to is the sunny state, with the highest score of 1.3.

5.3.3 Training the model (problem 3)

Problem 3 is the training problem. It is the most difficult to solve, where the model must be adjusted to best describe the observations. The training of the model necessitate real observation sequences from which the model can be adjusted.

Although there is no way to solve the problem analytically, an iterative procedure for solving it is presented in [6], among other solutions. Broadly, the method calculates estimates of the model parameters $\bar{A}$, $\bar{B}$ and $\bar{\pi}$ based on some initial model parameters A, B and π. The estimated model $\bar{\lambda}$ is then compared to the initial one and if we have $P(O|\bar{\lambda}) > P(O|\lambda)$, the estimated model $\bar{\lambda}$ is the winning the model over λ. This is done iteratively, obtaining each time a new estimated model from the latest winning model and comparing them until some limiting point is reached. Estimation of the parameters is given by:

$$\bar{\pi}_i = \text{Expected frequency in state } S_i \text{ at time } t = 1 \tag{52}$$

$$\bar{a}_{ij} = \frac{\text{Expected number of transition from state } S_i \text{ to state } S_j}{\text{Expected number of transitions from state } S_i} \tag{53}$$

$$\bar{b}_j(k) = \frac{\text{Expected number of times in state } S_j \text{ and observing symbol } v_k}{\text{Expected number of times in state } S_j} \tag{54}$$

Without detailing the full equations for calculating the expected quantities defined above, they mainly depend on the probability $\xi(i, j)$ of being in state S_i at time t and state S_j at time $t + 1$, given the model and the observation sequence, as well as $\gamma_t(i)$, the probability of being in state S_i at time t.

5.3.4 Types of hidden Markov models

The type of a HMM is define according to its state transition matrix A, where specific properties of the a_{ij} values may exist. Depending on the application, different types may be required. An example of this is the left-right type of HMM, whose property is that $a_{ij} = 0$ for all $j < i$. This type of HMM is suitable for modeling signals whose properties change over time, such as speech for example.

5.3.5 Continuous observation densities

In many applications, such as speech recognition for example, observations are continuous signals instead of discrete symbols. A simple way to deal with continuous observations densities is to transform them into discrete symbols using quantization methods. This has the drawback of the degradation of the information contained in the observations. The best solution is to use continuous densities in the observation model B. In [6], a mixture of simple parametric probability density function is used to represent the observation probabilities $b_j(O)$, such as mixture of Gaussian densities for example.

5.4 Speech recognition application

In this section, HMM theory is applied to the problem of speech recognition. The presented overall recognition system is made of five consecutive steps that are shown in Figure 4.

The first one, feature analysis, converts the speech signals into observation vectors O. Continuous and discrete observation models are presented in [6] that both rely on a spectral analysis that will not be discussed in this work. The following four steps, unit matching, lexical decoding, syntactic analysis and semantic analysis, act roughly the same in that they all output the most likely speech structure based on their input. The difference between each of them is in the complexity of the speech structure that is processed, which is increasing as the steps are advanced. The unit is the most simple speech structure that needs to be recognized. Different choices of units are possible, including sub-word units such as phones and syllables, among others. Units can also represent whole words and even group of words. The task of the unit matching system is to output the most likely units based on the observation vectors it receives. Lexical decoding does the same, except that instead of outputting units, it outputs a higher level speech structure. For example, if a unit is a sub-word structure, the lexical decoding could output words based on the sub-words it receives and according to a word dictionary. Syntactic analysis is still at an higher level. Its task is to find the proper word sequence based on the individual it receives from lexical decoding and as defined by a word grammar. Similarly, semantic analysis adds further constraints on the word sequence, depending on the task at hand. Based on a task model, the most likely word path is searched.

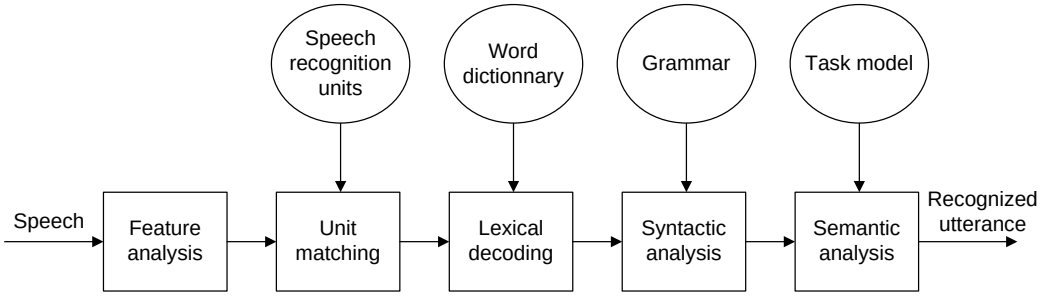


Figure 4: Speech recognition system presented in [6]

Each steps that follow feature analysis can be seen as a process that takes in an observation sequence and that outputs a most probable model for explaining the observation sequence. Therefore, these steps of unit matching, lexical decoding, syntactic analysis and semantic analysis all consist of the comparison of different HMMs, with a decision on the model whose probability of the observation sequence is highest is chosen. Formally, suppose that for a particular step there are V models ,λ^1, λ^2, ..., λ^V, that represent V different elements of the speech structure. The task is to choose the model λ^{u*} whose probability $P(O|\lambda^{u*})$ of the observation sequence given the model is the highest among all V models.

A simple example of a word recognition system is shown in Figure 5. The system is only made of a unit matching system, where a unit represents a word. For each word u, an HMM λ^u must be trained to maximize $P(O_u|\lambda^u)$, the probability related to model λ^u when an observation sequence O_u of the word u is presented to the model. Then, when a new observation sequence O is presented to the system, the chosen word for this sequence is the

one for which its HMM has the highest probability:

$$u^* = \text{argmax}_{1 \leq u \leq V} [P(O|\lambda^u)]. \tag{55}$$

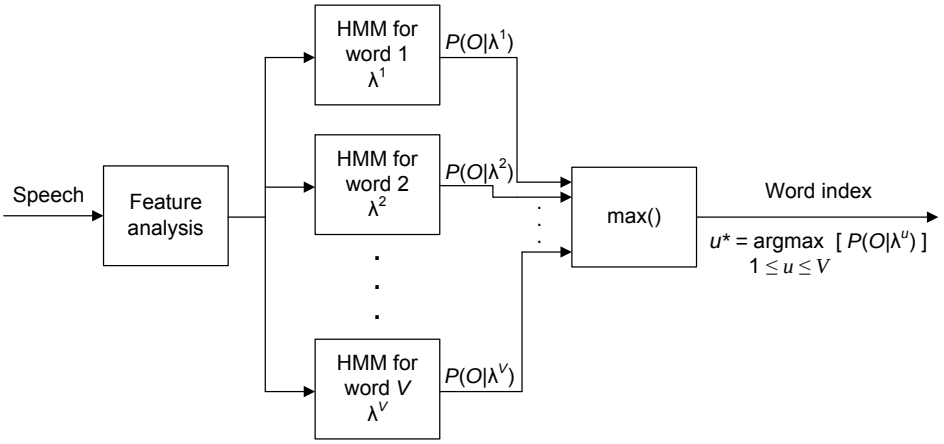


Figure 5: Isolated word recognition system presented in [6]

This page intentionally left blank.

6 Recurrent neural networks

The concept of recurrent network is about representing time by the effect it has on the processing of inputs. The idea is that the processing at a time t be dependent on previous processing results in time. This is where recurrence comes into play, connecting the output of a processing unit A to the input of a unit B whose output is directly or indirectly connected to A. A can also be connected to itself. Units with recurrent connections must therefore process data generated at different times, so that the state of the units represent some kind of memory.

Although much more complex recurrent network have been developed, their mechanisms are based on the principles of memory representation of the simple recurrent network presented in [7, 3] and shown in Figure 6(b). The simple recurrent network is made of a single hidden layer that is placed between the output and the input layers. The difference with feedforward networks is that the state of the hidden layer depends concurrently on its previous state and on the input layer. Because the hidden units have their own previous state as part of their input, their internal representations depend on the time dependencies of the input. In [7], this behavior is called a "memory for sequences" since the internal states are sensitive to temporal context.

Formally, a recurrent neural network operates on an input space and also, as opposed to feedforward networks, on an internal state space that enables the representation of the temporal dependencies of the input. Suppose a two-layered feedforward network whose output $\boldsymbol{y}_o(t)$ for an input $\boldsymbol{u}(t)$ is given by:

$$\boldsymbol{y}_o(t) = G(F(\boldsymbol{u}(t))) \tag{56}$$

where F and G represent to two layer of weights of the network.

According to this feedforward network representation, a recurrent network can be represented similarly, except for an internal state $\boldsymbol{x}(t)$ that is defined recurrently over itself and the input space:

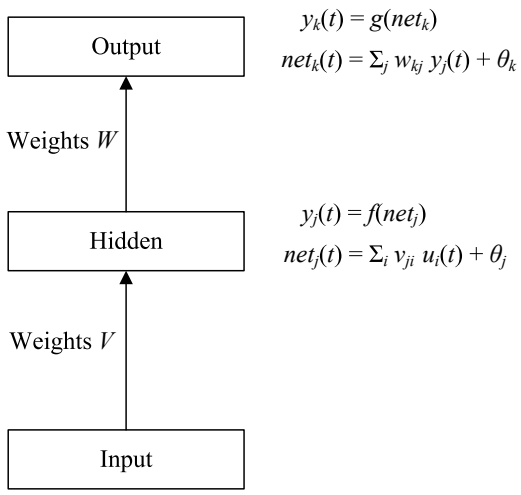
$$\boldsymbol{y}_o(t) = G(\boldsymbol{x}(t)) \tag{57}$$

$$\boldsymbol{x}(t) = F(\boldsymbol{x}(t-1), \boldsymbol{u}(t))) \tag{58}$$

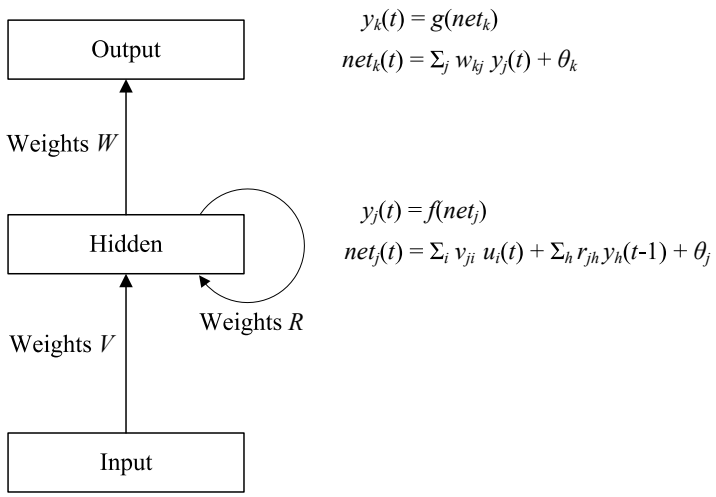
A feedforward network and a recurrent network of both two layers are shown in Figures 6(a) and 6(b), respectively.

Suppose $y_j(t)$ is the output of unit j on only hidden layer of the networks presented in Figure 6. The feedforward network would have

$$y_j(t) = f(\sum_i^n u_i(t) v_{ji} + \theta_j), \tag{59}$$



(a) Two-layered feedforward network



(b) Two-layered recurrent network

Figure 6: Comparison of a feedforward network and a recurrent network presented in [3]

while the recurrent network would have

$$y_j(t) = f(\sum_i^n u_i(t)v_{ji} + \sum_h^m y_h(t-1)r_{jh}\theta_j), \tag{60}$$

where f is any differentiable activation function.

6.1 Training methods

An open problem of recurrent neural network is the one of training. The use of recurrent connections leave the classical error backpropagation methods unfeasible. Nevertheless, some approach have been proposed that are based on the principle of error backpropagation. Two of the most popular are the Backpropagation through time (BPTT) and real-time recurrent learning (RTRL) methods, that are discussed in the next subsections.

6.1.1 Backpropagation through time

Backpropagation consists in adjusting the weights of a network such that an error function is minimized [8, 3]. The weights are adjusted according to the gradient of the error function with respect to the weights. The most frequently used error function, also called a cost function, is the summed squared error, that compares desired outputs to the actual ones in the network. It is defined as

$$C = \frac{1}{2}\sum_p^n \sum_k^m (d_{pk} - y_{pk})^2 \tag{61}$$

where d is the desired output at unit k and for a training pattern p. n training samples and m output nodes are assumed.

In the case where gradient descent is used, weight change is represented by

$$\Delta w = -\eta \frac{\partial C}{\partial w} \tag{62}$$

where η is a learning rate.

The weight change can be defined according to an error component $\delta = -\partial C/\partial net$, such that

$$\Delta w = \delta \frac{\partial net}{\partial w} \tag{63}$$

Assuming m output units, the error of an output unit $k \in \{1, 2 \ldots m\}$ at time t is

$$\delta_k(t) = -\frac{\partial y_k}{net_k}\frac{\partial C}{\partial y_k} = f'_k(net_k(t))(d_k(t) - y_k(t)), \tag{64}$$

where

$$y_i(t) = f_i(net_i(t)) \tag{65}$$

is the activation of a non-input unit i with differentiable activation function f_i. $net_i(t)$ represents unit i's input at time t:

$$net_i(t) = \sum_j w_{ij} y_j(t-1) \tag{66}$$

where w_{ij} is the weight on the connection from unit j to i. The backpropagation through time (BPTT) of the error to non-output units is given by

$$\delta_j(t) = f'_j(net_j(t)) \sum_k^m w_{kj} \delta_k(t+1), \tag{67}$$

where $\delta_k(t+1)$ is the error of an output unit $k \in \{1, 2 \dots m\}$ at time $t+1$ and $\delta_j(t)$ is the error of a non-output j at time t. The backpropagation of the error through time can be understood by the unfolding principle of a recurrent network shown in Figure 7.

If n signal samples are used from a training set, the weight change is then

$$\Delta w_{jl} = \eta \sum_{p=1}^{n} \delta_{pj}(t) y_{pl}(t-1), \tag{68}$$

where δ_{pj} represents the error at node j for the training sample p and $y_{pl}(t-1)$ is the output ar node l when p is presented to the network.

6.1.2 Real-time recurrent learning

In [8], an evolution of the backpropagation through time method (BPTT) is presented, called real-time recurrent learning (RTRL). The difference with backpropagation through time is in the memory requirement. As opposed to the conventionnal backpropagation through time approach, real-time recurrent learning does not have the growing memory problem for long training sequences. According to this method, the weights w_{ij} are incremented at each time step by the amount $\Delta w_{ij}(t)$, instead of accumulating the $\Delta w_{ij}(t)$ values over time and making the weight change at the end of the training sequence.

6.2 Long-Short Term Memory

In [9], a critiscism of classical recurrent backpropagation methods is made, including BPTT and RTRL, that are defined in subsections 6.1.1 and 6.1.2, respectively. More precisely, it treats the problem of insufficient and decaying error backpropagation, especially when long time dependencies must be learned. It is shown that for q time steps, the error backpropagated from a unit u to a unit v will either increase or decrease exponentionnally with q, depending on the type of activation function and on the weight values. This explains

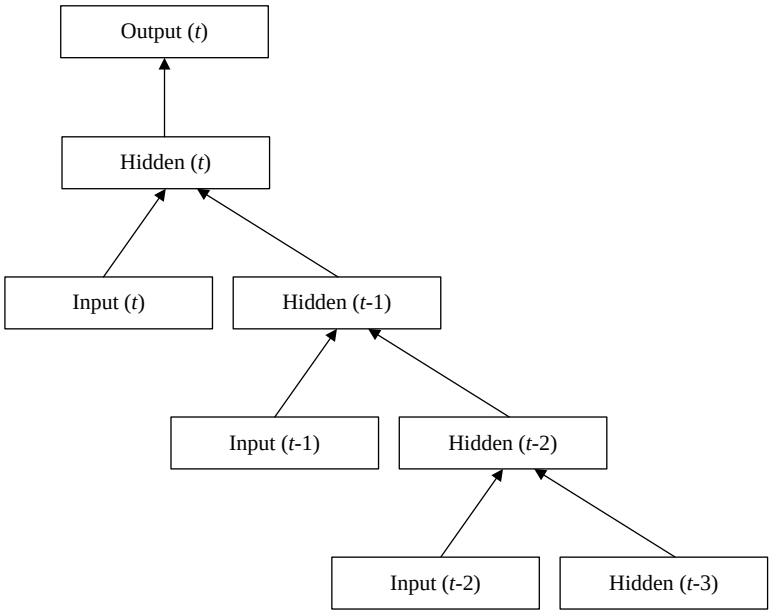


Figure 7: Unfolding principle of backpropagation through time (BPTT)

why backpropagation methods such as BPTT and RTRL have poor long term memory capabilities.

Long-Short Term Memory is a method that was created to correct the vanishing error problem. It seeks for a compromise between short term learning capabilities and long term learning capabilities. This is achieved through the use of memory cells that embodies "constant error flow" instead of the exponential variations found with conventionnal BPTT and RTRL. A memory cell is shown in Figure 8. The idea is to propagate a constant error through the recursive connexion of a unit, known as "constant error carrousel" (CEC), such that a unit j's local error be

$$\delta_j(t) = f'_j(net_j(t))w_{jj}\delta_j(t+1), \tag{69}$$

requiring

$$f'_j(net_j(t))w_{jj} = 1. \tag{70}$$

Integrating (70), we get $f_j(net_j(t)) = \frac{net_j(t)}{w_{jj}}$. A value of 1 is chosen for w_{jj} to have a constant and linear activation for unit j. This yields a unit capable of long term error propagation, or state propagation when the network is in running mode. To control when an error should enter memory and when the memory contents should be sent to other units, an "input gate unit" (in_j) and an "output gate unit" (out_j) are used. Both units are connected to the network and must be trained in a manner that the propagated errors are appropriately carried in and read out of the CEC.

It has been demonstrated in [9] that, for some specific problems involving signals with long time dependencies, LSTM does better than conventionnal recurrent network methods that

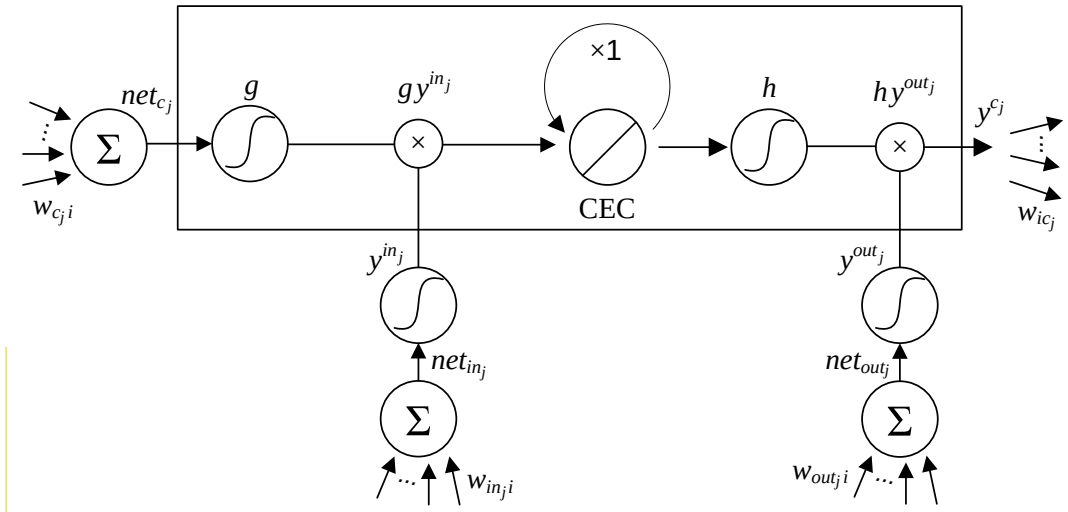


Figure 8: Memory cell in a Long-Short Term Memory [9]

use BPTT or RTRL. The examples in [9] sum up to predict a class given a signal containing a restricted number of relevant elements to the class, with long time lags between each relevant elements of the signal. Note that, although some examples include noisy signals for the non-relevant elements, the relevant elements are always free of noise.

7 Reservoir computing

Reservoir methods were first introduced as a way to avoid the problem of training recurrent neural networks. They rely on two connected parts: a reservoir and a readout function. The readout function is also called the learning function. The concept of reservoir computing is illustrated in Figure 9. It is based on the assumption that, by feeding an input stream into a recurrent network of a given complexity, the latter should contain meaningful information about the recent input signals.

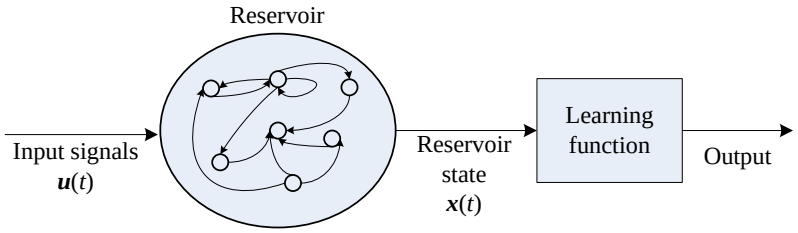


Figure 9: The concept of reservoir computing. Input signals stimulate the reservoir in a complex, non-linear and recurrent fashion. The learning function extracts and compute the relevant information from the reservoir to infer an answer about the inputs.

Generally, the reservoir consists of an untrained and randomly generated RNN. As stated in [10], the reservoir acts as a nonlinear temporal expansion function. As shown in Figure 10, the state $\boldsymbol{x}$ that is produced out of this function represents the expansion of the input signals $\boldsymbol{u}$ in time into a higher dimensional space $\Re^{N_x}$, with $N_x > N_u$. The reservoir state $\boldsymbol{x}(n)$ at a time step n is the result of the stimulation of the reservoir by the inputs $\boldsymbol{u}(1)$, ..., $\boldsymbol{u}(n-1)$, $\boldsymbol{u}(n)$.

The readout function performs the recognition of the patterns. It uses the reservoir state $\boldsymbol{x}(n)$ as an input to produce a final output $\boldsymbol{y}(n)$. Therefore, the recognition function does not have to deal with time since $\boldsymbol{x}(n)$ corresponds to the expansion into space of the time dimension of the input signal. Simple non-recurrent recognition methods can then be used.

In [11], a general reservoir computing model is presented. The model is not so general since it involves a linear readout function, but this simplifies notation. The model is shown in Figure 11.

The state equation of the reservoir is

$$\boldsymbol{x}(t+1) = f(\boldsymbol{W}_{res}^{res}\boldsymbol{x}(t) + \boldsymbol{W}_{in}^{res}\boldsymbol{u}(t) + \boldsymbol{W}_{out}^{res}\boldsymbol{y}(t) + \boldsymbol{W}_{bias}^{res}), \tag{71}$$

where f is an activation function, $\boldsymbol{x}(t)$ is the reservoir state, $\boldsymbol{u}(t)$ is the input signal and $\boldsymbol{y}(t)$ is the output of the readout function at time t. $\boldsymbol{W}_{res}^{res}$, $\boldsymbol{W}_{in}^{res}$ and W_{out}^{res} are the corresponding weight matrices, respectively, and $\boldsymbol{W}_{bias}^{res}$ is a bias weight matrice.

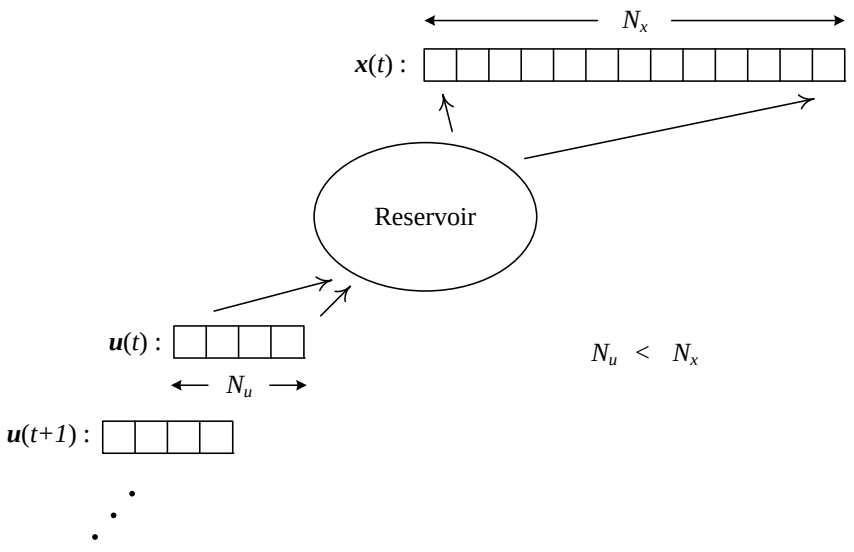


Figure 10: The state expansion concept as explained in [10, 11]

The linear readout model, supposing teacher forced training, is

$$\hat{\boldsymbol{y}}(t+1) = \boldsymbol{W}_{res}^{out}\boldsymbol{x}(t+1) + \boldsymbol{W}_{in}^{out}u(t) + \boldsymbol{W}_{out}^{out}\boldsymbol{y}(t) + \boldsymbol{W}_{bias}^{out}, \tag{72}$$

where the weight matrices $\boldsymbol{W}_{res}^{out}$, $\boldsymbol{W}_{in}^{out}$, $\boldsymbol{W}_{out}^{out}$ and $\boldsymbol{W}_{bias}^{out}$ are trained. Note that this model assumes a delay of one time step between the readout output $\hat{\boldsymbol{y}}(t+1)$ and the signal input $\boldsymbol{u}(t)$. Therefore, according to this model, the readout output at time t does not depend on the actual input value at time t, but on previous input values. A different model could be used where the readout output depends on the actual input value:

$$\hat{\boldsymbol{y}}(t+1) = \boldsymbol{W}_{res}^{out}\boldsymbol{x}(t+1) + \boldsymbol{W}_{in}^{out}\boldsymbol{u}(t+1) + \boldsymbol{W}_{out}^{out}\boldsymbol{y}(t) + \boldsymbol{W}_{bias}^{out}. \tag{73}$$

7.1 Reservoir methods

Among the few popular reservoir methods developed in the recent years, the work of [10] cites the Echo States Networks (ESNs) and the Liquid State Machines (LSMs) as the pioneering methods, as well as the Evolino and Backpropagation-Decorrelation approaches. Evolino and Backpropagation-Decorrelation extend the ESN approach by way of reservoir training strategies. The former uses genetic training of the internal weights, while the latter uses an alteration of the classical error backpropagation approach discussed in Subsection 6.1.1. Echo State Networks and Liquid State Machines are presented in more details in the following paragraphs.

7.1.1 Echo State Networks

Echo State Networks have reservoirs that are made of recurrent neural networks with simple neuron models. Those models consists of the addition of weighted inputs and application

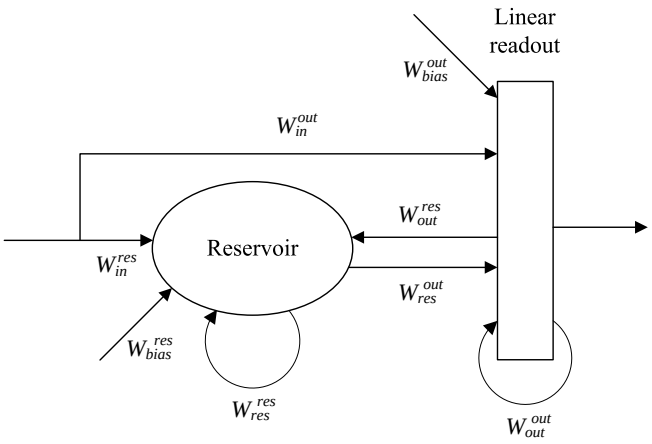


Figure 11: General reservoir computing model presented in [11]

of an activation function. The term "echo states" has to do with the activation state $\boldsymbol{x}(n)$ of the recurrent neural network [12], since it is assumed that $\boldsymbol{x}(n)$ depends on past inputs $\boldsymbol{u}(n-1)$, $\boldsymbol{u}(n-2), ...$ as more to the current inputs $\boldsymbol{u}(n)$. A network is said to have echo states if there exits echo functions $\boldsymbol{E} = (e_1, ..., e_N)$ such that

$$\boldsymbol{x}(n) = \boldsymbol{E}(\boldsymbol{u}(n_0), ..., \boldsymbol{u}(n-1), \boldsymbol{u}(n)) \tag{74}$$

where n_0 can be any time such that $n_0 < n$.

Suppose the echo states model presented in [12], made of K inputs units with their activations $\boldsymbol{u}(n) = (u_1(n), ..., u_K(n))$ at time n, N internal units with their activations $\boldsymbol{x}(n) = (x_1(n), ..., x_N(n))$ and L output units $\boldsymbol{y}(n) = (y_1(n), ..., y_L(n))$. The corresponding weight matrices are $\boldsymbol{W}_{in}$, a $N \times K$ input weight matrix, $\boldsymbol{W}$, a $N \times N$ internal weight matrix , $\boldsymbol{W}_{back}$, a $N \times L$ output to internal units weight matrix, and $\boldsymbol{W}_{out}$, a $L \times (K + N + L)$ output weight matrix. The internal activation state $\boldsymbol{x}(n+1)$ at time $n+1$ is given by

$$\boldsymbol{x}(n+1) = \boldsymbol{f}(\boldsymbol{W}_{in}\boldsymbol{u}(n+1) + \boldsymbol{W}\boldsymbol{x}(n) + \boldsymbol{W}_{back}\boldsymbol{y}(n)). \tag{75}$$

Typical ESNs use sigmoid function for the output functions of the internal units, $\boldsymbol{f} = (f_1, ..., f_N)$. Note that output feedback $\boldsymbol{W}_{back}$ may not be necessary depending on the learning task.

A network must respect certain conditions in order to have the echo state property defined in (74). The most basic condition to have echo state relates to the weight matrix $\boldsymbol{W}$. Assuming sigmoid activation functions, a network has echo states if the single largest value of $\boldsymbol{W}$, σ_{max}, is lower than 1, that is $\sigma_{max} < 1$. ESNs are also sensitive to inputs containing zeros because of the sigmoid functions. Precisely, an ESN will have an unstable and non-echo state if the largest absolute eigenvalue of $\boldsymbol{W}$, $|\lambda_{max}|$, is higher than 1.

7.1.2 Liquid State Machines

Liquid State Machines uses the same reservoir concept as ESNs. They differentiate from ESNs mostly in the composition of the recurrent neural network. As opposed to ESNs, the recurrent neural network of a LSM has biological foundations. Indeed, the neurons and their connexions follow biologically inspired models.

The neuron models used in LSMs are in the family of spiking models. Spiking neurons communicate by sending pulses, or spikes, through time. This is called pulse coding, as opposed to rate coding found in conventional non-spiking neural networks, such as those used in ESNs. Using timed pulses gives much more computational power than conventional rate coding found in non-spiking networks. To prove this statement, consider the example where numbers that can take up to M different values need to be encoded using a binary representation. The task is to encode the number using only zeros (0s) and ones (1s), for example. With rate coding, this task necessitates $N = \log_2 M$ different inputs, as shown in Figure 12. With pulse coding, the timing of the pulse, or its delay according to a reference time, expresses a particular information. Furthermore, suppose that a time interval Δt is available between each input number and that the minimum time resolution is Δd. According to this, a single input can convey $\Delta t/\Delta d$ different pulses, according to their delays. Suppose the case where $(\Delta t/\Delta d) > M$, then a single input can easily encode the M different numbers (N=1). This case, which is illustrated in Figure 13, only considered a single pulse per time interval Δt. Suppose the case where it is possible to have two pulses during the interval Δt. A single input can then encode $M \times (M-1)/2$ different numbers. For the general case where it is possible to have p pulses during the interval Δt, a single input can encode $M!/(M-p)!p!$ different numbers, which corresponds to the number of combinations $\binom{M}{p}$. Hence, according to this example, pulse coding has more computational power than rate coding since it can encode more numbers while using fewer inputs.

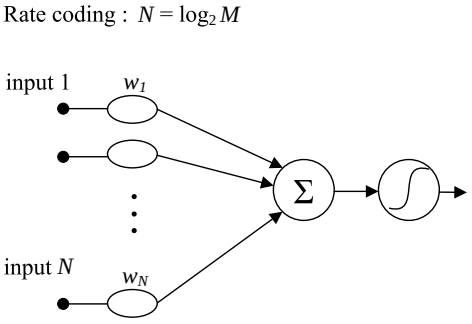


Figure 12: Binary representation of M input values using rate coding

The most widely used spiking neuron model in LSMs is the Integrate-and-fire model, which makes a simple representation of the key properties of real biological neurons. The model, illustrated in Figure 14, consist of an electronic circuit that is made of a capacitor C and a resistor R placed in parallel and in which a current $I(t)$ flows. When the voltage V_m over

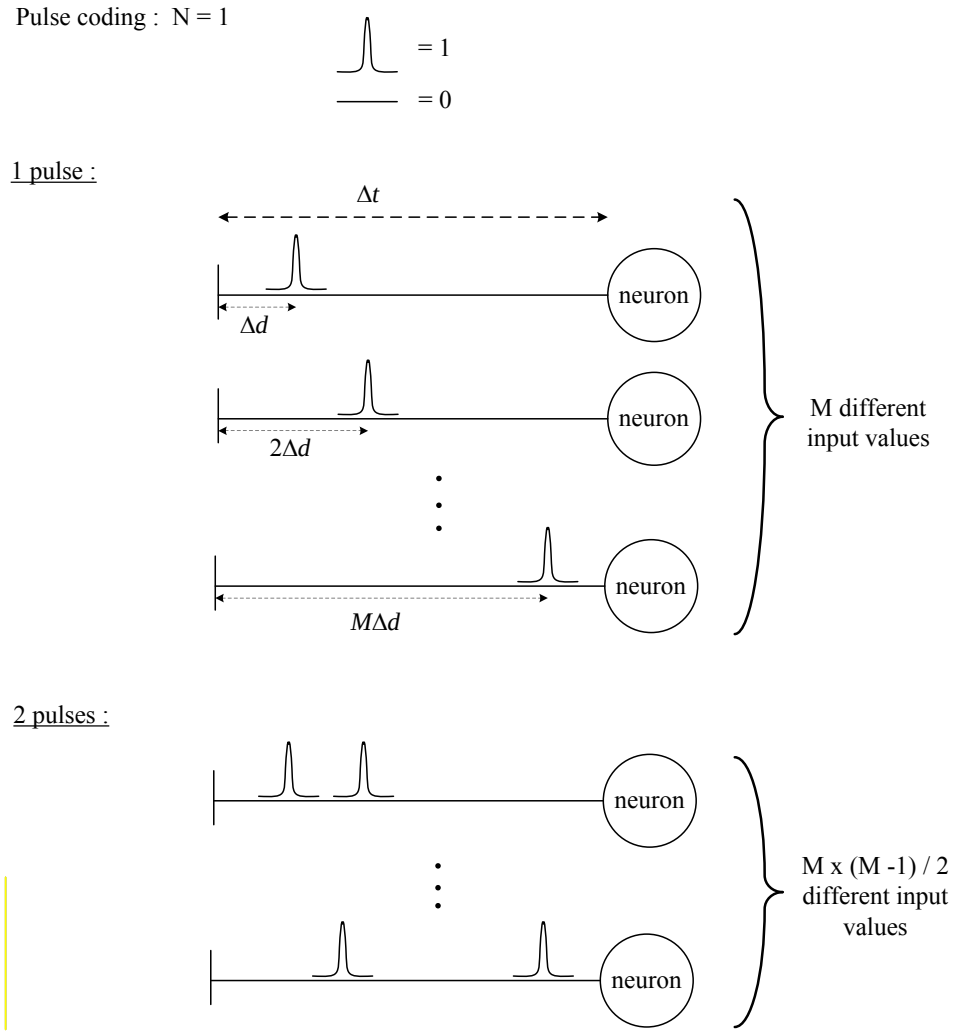


Figure 13: Binary representation of M input values using pulse coding

the capacitor reaches a threshold Θ, a pulse is emitted.

According to [2], the spiking recurrent neural network that forms the "liquid" can be defined as

1. a set of V spiking neurons,
2. a set $E \subseteq V \times V$ of synapses,
3. a weight $w_{u,v}$, a delay $d_{u,v} \geq 0$ and a response function $\gamma_{u,v}$ for each synapse $(u, v) \in E$, where (u, v) is the connection between neurons u and v,
4. and a threshold function Θ_v for each neuron $v \in V$.

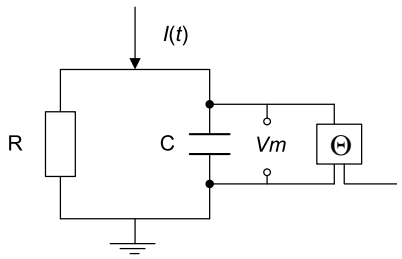


Figure 14: Integrate-and-fire neuron model. The neuron is modeled as an electronic circuit where a current $I(t)$ splits into a capacitor C and a resistor R.

As inputs are entered into the liquid, different waves of spikes are produced at the different input times and propagate through the network, mingling together. The state of the liquid at some time is then determined by the spiking activity of the network. There are many possible ways to read the state of the liquid. One possibility is to read the voltage of the neurons and set the state vector with the different voltage values. Another possibility is to consider the spikes. In that case, the spikes trains need to be transformed into analog values. Filtering techniques may be used, for example. The current state of the liquid should then hold information about recent inputs, and its high-dimensionality should allow for easy pattern classification using simple spatial classifiers.

7.2 Reservoir adaptation methods to improve their performance

Although the principle of untrained and randomly generated recurrent networks attached to basic learning algorithms has proven to be a promising spatiotemporal learning approach, the random nature of the reservoirs limits the learning capacity for specific tasks or applications [10]. This is why it may be useful to reduce randomness in the generation of reservoirs. Three kinds of adaptation methods for obtaining better reservoirs are defined in [10]:

1. **Universal methods:** These are general guidelines that do not depend on the application and that do not depend on the input signals $\boldsymbol{u}$ and the desired outputs $\boldsymbol{y}_{target}$. The guidelines may be seen as rules of thumbs that cover, among others, the reservoir size, topology and connectivity.

2. **Unsupervised methods:** Reservoirs are pre-trained with respect to the input signals $\boldsymbol{u}$ only. Among the different techniques are those that try to optimize the reservoir according to some measures of performance. For example, the review of [10] points out the pairwise correlation of the reservoir states $\boldsymbol{x}(n)$ and the entropy of the distribution of $\boldsymbol{x}(n)$, which should be minimum. These are mostly used with ESNs. Another more detailed measure is the computational power [13, 10], that has more to do with LSMs. It consists in taking a number k of different segment of inputs $\boldsymbol{u}^i(n)$, $i = 1, ..., k$ and collecting the corresponding reservoir state vector $\boldsymbol{x}^{u^i}(n_0)$ after some time n_0 into a matrix $M \in \Re^{k \times N_x}$. The quality of the reservoir is then given

by the rank of matrix M, where a high rank means a good quality. This measure is also known as the "separation property". Finally, there exists other unsupervised training methods that are more specific to the reservoir nature, *i.e.* its type of neurons. For example, LSMs can make use of synaptic plasticity, a property found in biological neurons where the weights of the connections adjust automatically based on past responses to some inputs [13, 14, 10].

3. **Supervised methods:** Reservoirs are pre-trained with respect to the input signals $\boldsymbol{u}$ and the desired outputs $\boldsymbol{y}_{target}$. This approach consists in generating a reservoir and checking its suitability for a particular task, given a particular readout function. The simplest supervised reservoir adaptation method is the evolutive method that generates k reservoirs and picks the best.

This page intentionally left blank.

8 Conclusion

This memorandum has discussed different tasks in the domain of spatiotemporal supervised learning, like time-series prediction or sequence classification, for example. It has also presented various approaches on this subject. Sliding windows seem limited in terms of the range of problems it can handle, but are still of interest since they permit any classical supervised algorithm to be used. Also, they have shown to perform well in some specific applications, such as the word prononciation task cited in [1]. Hidden Markov Models have proven to work well in some applications such as speech recognition, but they are vulnerable to noisy inputs, as stated in [6]. Recurrent neural networks have looked to be a promising approach. Nonetheless, there are still many open issues to be solved or improved, such as training for example. Reservoir computing is an extension of recurrent neural network that is supposed to resolve the training problem. However, the random nature of the reservoirs leaves many open challenges to improve their performance. It comes out that each approach has some advantages and drawbacks. The choice of an approach should be carefully studied depending on the nature of the task, its complexity and the desired performance level.

This page intentionally left blank.

References

[1] Dietterich, T. G. (2002), Machine Learning for Sequential Data: A Review, *Structural, Syntactic, and Statistical Pattern Recognition; Lecture Notes in Computer Science*, 2396, 15–30.

[2] Goodman, E. and Ventura, D. (2006), Spatiotemporal Pattern Recognition via Liquid State Machines, In *Proceedings of the International Joint Conference on Neural Networks*, pp. 7579–7584, Vancouver, British Columbia.

[3] Boden, M. (2002), A guide to recurrent neural networks and backpropagation., In *Technical report, In The DALLAS project, Report from the NUTEKSupported Project AIS-8: Application of Data Analysis with Learning Systems*, Sweden.

[4] Narendra, K.S. and Parthasarathy, K. (1990), Identification and control of dynamical systems using neural networks, *IEEE Transaction on Neural Networks*, 1, 4–27.

[5] Horne, Bill G. and Giles, C. Lee (1995), An experimental comparison of recurrent neural networks, In Tesauro, G., Touretzky, D., and Leen, T., (Eds.), *Advances in Neural Information Processing Systems*, Vol. 7, pp. 697–704, The MIT Press.

[6] Rabiner, L. R. (1989), A tutorial on hidden Markov models and selected applications in speech recognition, In *Proceedings of the IEEE*, Vol. 77, pp. 257–286.

[7] Elman, J. L. (1990), Finding Structure in Time, *Cognitive Science*, 14, 179–211.

[8] Williams, R. J. (1989), A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, *Neural Computation*, 1, 270–280.

[9] Hochreiter, S. and Schmidhuber, J. (1997), Long short term memory, *Neural Computation*, 9(8), 1735–1790.

[10] Lukosevicius, M. and Jaeger, H. (2007), Overview of Reservoir Recipes, (Technical Report 11) School of Engineering and Science, Jacobs University, Germany.

[11] Schrauwen, B., Verstraeten, D., and Campenhout, J. Van (2007), An overview of reservoir computing: theory, applications and implementations, In *Proceedings of the 15th European Symposium on Artificial Neural Networks*, pp. 471–482.

[12] Jaeger, H. (2001), The "echo state" approach to analysing and training recurrent neural networks, (Technical Report Technical Report GMD Report 148) German National Research Center for Information Technology.

[13] Vreeken, J. (2003), Liquid State Machines, a review, Technical Report Institute for Information and Computing Sciences, Utrecht University.

[14] Maass, W. and Zador, A. (1998), Computing and learning with dynamic synapses, In Maass, W. and Bishop, C., (Eds.), *Pulsed Neural Networks*, pp. 321–336, MIT-Press (Cambridge).

This page intentionally left blank.

Distribution list

DRDC Valcartier TM 2008-315

Internal distribution

1 Director General

1 Document Library

1 Head/Decision Support Systems

1 Dr P. Valin

1 Head/Information and Knowledge Management

1 Head/System of Systems

1 Dr A. Benaskeur

1 J. Berger

1 E. Dorion

1 Dr A.-L. Jousselme

1 P. Maupin

1 F. Rhéaume (author)

Total internal copies: 12

External distribution

1 Library and Archives Canada

1 DRD KIM (PDF file)

1 Director Science & Technology Maritime (DSTM)
Constitution Bldg., 305 Rideau St., Ottawa, ON K1N 9E5.

1 Director Science & Technology Land (DSTL)
Constitution Bldg., 305 Rideau St., Ottawa, ON K1N 9E5.

1 Director Science & Technology Air (DSTA)
Constitution Bldg., 305 Rideau St., Ottawa, ON K1N 9E5.

1 Director Science & Technology C4ISR (DSTC4ISR)
Constitution Bldg., 305 Rideau St., Ottawa, ON K1N 9E5.

Total external copies: 6

Total copies: 18

FICHE DE CONTRÔLE DU DOCUMENT

1. PROVENANCE (le nom et l'adresse)
François Rhéaume
RDDC - Valcartier
2495, boul. Pie-Xi Nord
Québec Qc, G3J 1X5 Canada

2. COTE DE SÉCURITÉ
(y compris les notices d'avertissement, s'il y a lieu)
UNCLASSIFIED

3. TITRE (Indiquer la cote de sécurité au moyen de l'abréviation (S, C, R ou U) mise entre parenthèses, immédiatement après le titre.)
A review of popular spatiotemporal pattern recognition techniques (U)

4. AUTEURS (Nom de famille, prénom et initiales. Indiquer les grades militaires, ex.: Bleau, Maj. Louis E.)
François Rhéaume

5. DATE DE PUBLICATION DU DOCUMENT (mois et année)	6a. NOMBRE DE PAGES	6b. NOMBRE DE REFERENCES
march 2009	**54**	**14**

7. DESCRIPTION DU DOCUMENT (La catégorie du document, par exemple rapport, note technique ou mémorandum. Indiquer les dates lorsque le rapport couvre une période définie.)
Mémorandum

8. PARRAIN (le nom et l'adresse)

9a. NUMÉRO DU PROJET OU DE LA SUBVENTION (Spécifier si c'est un projet ou une subvention)	9b. NUMÉRO DE CONTRAT

10a. NUMÉRO DU DOCUMENT DE L'ORGANISME EXPÉDITEUR	10b. AUTRES NUMÉROS DU DOCUMENT
DRDC Valcartier TM 2008-315	N/A

11. ACCÈS AU DOCUMENT (Toutes les restrictions concernant une diffusion plus ample du document, autres que celles inhérentes à la cote de sécurité.)

- ☒ Diffusion illimitée
- ☐ Diffusion limitée aux entrepreneurs des pays suivants (spécifier)
- ☐ Diffusion limitée aux entrepreneurs canadiens (avec une justification)
- ☐ Diffusion limitée aux organismes gouvernementaux (avec une justification)
- ☐ Diffusion limitée aux ministères de la Défense
- ☐ Autres

12. ANNONCE DU DOCUMENT (Toutes les restrictions à l'annonce bibliographique de ce document. Cela correspond, en principe, aux données d'accès au document (11). Lorsqu'une diffusion supplémentaire (à d'autres organismes que ceux précisés à la case 11) est possible, on pourra élargir le cercle de diffusion de l'annonce.)

UNLIMITED

13. SOMMAIRE (Un résumé clair et concis du document. Les renseignements peuvent aussi figurer ailleurs dans le document. Il est souhaitable que le sommaire des documents classifiés soit non classifié. Il faut inscrire au commencement de chaque paragraphe du sommaire la cote de sécurité applicable aux renseignements qui s'y trouvent, à moins que le document lui-même soit non classifié. Se servir des lettres suivantes: (S), (C), (R) ou (U). Il n'est pas nécessaire de fournir ici des sommaires dans les deux langues officielles à moins que le document soit bilingue.)

Recognition is an important process of situation analysis in military command and control systems. This report focuses on recognition tasks that must rely on data that evolve in time. This type of problem is known as spatiotemporal pattern recognition. The report provides an objective review of supervised learning approaches for spatiotemporal data. The list is non-exhaustive but provides a variety of popular methods presented in the literature. The spatiotemporal pattern recognition methods presented in this review are sliding windows, hidden Markov models and conventional recurrent neural networks, including different training methods such as backpropagation through time and real-time recurrent learning. It also covers long-short term memories and the reservoir computing approach, where Echo states networks and Liquid state machines are presented as the main reservoir methods. The review explains and discusses the concepts and algorithms behind the different techniques.

14. MOTS-CLÉS, DESCRIPTEURS OU RENSEIGNEMENTS SPÉCIAUX (Expressions ou mots significatifs du point de vue technique, qui caractérisent un document et peuvent aider à le cataloguer. Il faut choisir des termes qui n'exigent pas de cote de sécurité. Des renseignements tels que le modèle de l'équipement, la marque de fabrique, le nom de code du projet militaire, la situation géographique, peuvent servir de mots-clés. Si possible, on doit choisir des mots-clés d'un thésaurus, par exemple le "Thesaurus of Engineering and Scientific Terms (TESTS)". Nommer ce thésaurus. Si l'on ne peut pas trouver de termes non classifiés, il faut indiquer la classification de chaque terme comme on le fait avec le titre.)

Spatiotemporal pattern recognition, supervised learning, sliding windows, recurrent neural networks, hidden markov models

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE R&D DÉFENSE

www.drdc-rddc.gc.ca