UNIVERSITÉ DE MONTRÉAL

NOUVELLE APPROCHE ANALYTIQUE POUR L'APPRENTISSAGE DU QUANTRON

SIMON DE MONTIGNY
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

NOUVELLE APPROCHE ANALYTIQUE POUR L'APPRENTISSAGE DU QUANTRON

présentée par : DE MONTIGNY Simon
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. ANJOS Miguel F., Ph.D., président
M. LABIB Richard, Ph.D., membre et directeur de recherche
M. GAGNON Michel, Ph.D., membre
Mme JAPKOWICZ Nathalie, Ph.D., membre

DÉDICACE

À la mémoire des docteurs Gérard de Montigny (1909-2009)
et Claude de Montigny (1945-2012)

REMERCIEMENTS

Je veux tout d'abord remercier le FQRNT qui m'a accordé une bourse doctorale. Ce support financier a grandement facilité mes premières années de recherche. Je tiens aussi à remercier le département de mathématiques et de génie industriel, où j'ai eu la chance de dispenser des cours de probabilités et statistique. Merci aux professeurs avec qui j'ai collaboré : Antoine Saucier, André Turgeon, Bernard Clément, Mario Lefebvre et Sébastien Le Digabel. Merci également à Toufik Hammouche et Wissem Maazoun, collègues étudiants et chargés de cours.

Mon passage au doctorat a été parsemé de plusieurs rencontres avec des membres de la communauté de Polytechnique (actuels et anciens), entre autres : Jean Guérin, Michel Gagnon, Marc Laforest, Jean Conan, Guy Desaulniers, Guillaume Nepveu, Khadija Douib, Jonathan Daniel-Rivest, Aimé Kuiteing, Christian Desrosiers, Shadi Sharif Azadeh, Hind Rakkay, Zoumana Coulibaly et Cong-Kien Dang. Merci au personnel du département. Merci à Paul Brunelle, Benoît Forest et Hamza Cheriti pour leur assistance avec mes problèmes informatiques.

Merci à mes parents et grands-parents pour leur amour indéfectible. Merci à ma conjointe, qui a tant bien que mal toléré la présence de mon ordinateur et de ma paperasse dans notre vie. Merci à mes amis ; je vous ai vus trop peu souvent au cours des dernières années.

Enfin, je tiens à exprimer ma gratitude envers le docteur Richard Labib, mon directeur de recherche, sans qui cette thèse n'existerait pas. Sous sa supervision, j'ai appris les rudiments du travail de chercheur et d'enseignant. Toujours disponible et de bon conseil, il m'a offert un projet de recherche stimulant et une liberté académique exceptionnelle. En particulier, son aide a été précieuse pour les demandes de bourse, l'élaboration de la proposition de recherche et la soumission d'articles. De plus, grâce à lui, j'ai eu l'opportunité de participer à la conférence IJCNN 2011 à San José, en Californie.

RÉSUMÉ

Le quantron est un neurone artificiel inspiré d'un modèle stochastique de la diffusion synaptique. Ce type de neurone biologiquement réaliste a le potentiel d'améliorer les capacités de classification des réseaux de neurones utilisés en reconnaissance de formes. Cependant, le quantron présente des difficultés pour l'implémentation d'un algorithme d'apprentissage efficace. Ceci est dû à la présence de discontinuités dans la fonction de réponse qui caractérise l'émission ou l'absence d'émission de neurotransmetteurs en réaction à la stimulation des synapses d'entrée. Ces discontinuités nuisent à l'apprentissage par modification itérative des paramètres du neurone. Ainsi, nous adoptons une approche analytique pour contourner ces difficultés et développer de nouveaux algorithmes d'apprentissage pour entraîner un quantron ou un réseau de quantrons.

D'abord, nous nous intéressons au maximum de la fonction représentant le potentiel électrique du quantron, appelée fonction d'activation. Par comparaison à un seuil d'excitabilité, ce maximum détermine l'état d'activité du quantron, qui est alors utilisé comme classificateur. En utilisant des potentiels post-synaptiques ayant un profil rectangulaire, nous obtenons une approximation du maximum en substituant des fonctions quadratiques aux signaux provenant des synapses d'entrée. Avec cette approximation analytique, nous démontrons expérimentalement la possibilité d'entraîner le quantron en minimisant une surface d'erreur par descente du gradient. De plus, pour certains problèmes, nous observons une amélioration des résultats d'un algorithme de recherche directe.

Ensuite, en utilisant une configuration particulière du quantron, nous trouvons une forme analytique simple pour la fonction d'activation dans le cas où les potentiels post-synaptiques possèdent un profil rectangulaire ou en rampe. Cette expression permet de lier les paramètres du quantron aux caractéristiques géométriques de sa frontière de décision. En se basant sur ces résultats, nous développons deux algorithmes d'apprentissage distincts, l'un procédant par l'analyse des configurations de la frontière de décision, et l'autre par l'inversion directe d'un système d'équations. Ces algorithmes permettent une résolution efficace de problèmes de classification pour lesquels le quantron admet une représentation sans erreur.

Enfin, nous portons attention au problème de l'apprentissage d'un réseau de quantrons. Dans le cas de potentiels post-synaptique avec un potentiel triangulaire, nous proposons une approximation analytique du temps où s'active le quantron, qui est déterminé par le premier instant où la fonction d'activation atteint le seuil d'excitabilité. L'expression mathématique résultante, utilisée comme valeur de réponse du neurone, permet d'adapter l'algorithme de rétropropagation de l'erreur au réseau. Nous montrons qu'il devient alors possible d'entraîner

des neurones qui autrement resteraient inactifs lors de l'apprentissage. De plus, nous illustrons la capacité des réseaux de quantrons à résoudre certains problèmes de classification en nécessitant moins de paramètres que des réseaux de neurones impulsionnels ou des réseaux de perceptrons.

Les trois aspects du quantron étudiés dans cette thèse mènent à des algorithmes qui se distinguent des approches antérieures utilisées pour l'apprentissage des réseaux de neurones impulsionnels. En effet, notre approche analytique permet d'éviter les discontinuités qui perturbent le processus d'apprentissage grâce au lissage résultant de l'approximation analytique du maximum de la fonction d'activation et du temps d'activation. De plus, l'analyse géométrique de la frontière de décision est rendue possible par l'expression analytique de la fonction d'activation. Le résultat le plus probant est la tentative fructueuse de résolution du problème associé à l'entraînement des neurones inactifs, appelé problème des neurones silencieux. Par notre approche analytique de l'apprentissage du quantron, nous proposons donc des algorithmes originaux et innovateurs qui contribuent à une meilleure compréhension de l'apprentissage dans les réseaux de neurones biologiquement réalistes.

ABSTRACT

The quantron is an artificial neuron inspired by a stochastic model of synaptic diffusion. This type of biologically realistic neuron can improve the classification capacity of neural networks used in pattern recognition. However, the implementation of an efficient learning algorithm for the quantron proves to be challenging. This is due to the presence of discontinuities in the output function which characterizes the emission of neurotransmitters, or lack thereof, as a reaction to the stimulus applied to synaptic inputs. These discontinuities disrupt the iterative training of the neuron's parameters. Thus, in this work, we follow an analytical approach to avoid these difficulties and develop new learning algorithms adapted to the quantron and to networks of quantrons.

First, we study the maximum of the function representing the electrical potential of the quantron, called the activation function. By comparing this function to an excitability threshold, this maximum determines the activity state of the neuron, which can be used as a classifier. Using post-synaptic potentials with a rectangular profile, we obtain an analytical approximation of the maximum by substituting quadratic functions for the signals stemming from the synaptic inputs. With this analytical approximation, we provide an experimental demonstration of the quantron being trained by minimizing an error surface via gradient search. Also, for certain problems, we observe an improvement of the results obtained by using a direct search algorithm.

Second, using a specific configuration of the quantron, we find a simple analytical form for the activation function when the post-synaptic potentials have a rectangular or ramp profile. This expression links the parameters of the quantron to the geometrical characteristics of its decision boundary. Building upon these results, we obtain two distinct learning algorithms, one proceeding by analyzing the configurations of the decision boundary, and the other by solving directly a system of equations. These algorithms are able to solve efficiently classification problems for which the quantron admits an errorless representation.

Third, we focus on the problem of training a network of quantrons. For post-synaptic potentials having a triangular profile, we propose an analytical approximation of the time when the quantron's activation function reaches the excitability threshold. The resulting mathematical expression, used as the neuron's output, enables the adaptation of the error backpropagation algorithm to the network. We show that it is then possible to modify the parameters of neurons which would otherwise remain inactive during training. Furthermore, we show that networks of quantrons can solve particular classification problems using fewer parameters than networks of spiking neurons or networks of perceptrons.

The three aspects of the quantron studied in this thesis yield algorithms which differ from previous attempts to train spiking neural networks. Indeed, we avoid the discontinuities that disturb the training process due to the smoothing effect of the analytical approximation of the activation function's maximum and of the activation time. Also, the geometrical analysis of the decision boundary is made possible by the analytical expression of the activation function. The most important result is the successful attempt to solve the problem of training inactive neurons, called the silent neuron problem. By following an analytical approach in the study of the quantron, we propose original and innovative algorithms which contribute to a better understanding of the learning process in networks of biologically realistic neurons.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

STDP	Spike Timing Dependent Plasticity
MuSpiNN	Multi-Spiking Neural Network
DS	Direct Search
GS	Gradient Search
UCI	University of California, Irvine
Misc. err.	Misclassification error
RMSE	Root Mean Square Error
MAE	Mean Absolute Error

LISTE DES ANNEXES

INTRODUCTION

En intelligence artificielle, le neurone artificiel (ou neurone formel) est un modèle mathématique qui reproduit certaines caractéristiques des neurones biologiques. Un ensemble interconnecté de neurones, appelé réseau de neurones artificiels, peut être utilisé de manière analogue au cerveau humain pour mémoriser des formes ou des symboles à l'aide d'un algorithme d'apprentissage qui ajuste les paramètres du réseau. De plus, lorsqu'on lui présente une nouvelle forme, le réseau bien entraîné peut l'interpréter à la lumière des formes déjà connues. Ce type d'apprentissage, appelé apprentissage supervisé, trouve application dans divers domaines de l'ingénierie où l'on effectue de la classification automatique, de la reconnaissance de formes ou de la prévision (Blue *et al.*, 1993; Rowley *et al.*, 1996; Santos *et al.*, 2000; Birikundavyi *et al.*, 2002; Clark *et al.*, 2003; Brault *et al.*, 2011).

Outre l'algorithme d'apprentissage du réseau, son architecture ainsi que le type de neurone utilisé sont des facteurs ayant une influence importante sur ses capacités. En effet, l'analogie avec le cerveau est imparfaite si l'architecture du réseau est trop simple ou si le modèle du neurone biologique est rudimentaire. Le principal algorithme d'apprentissage des réseaux de neurones, appelé rétropropagation de l'erreur (Rumelhart *et al.*, 1986), s'adapte à plusieurs architectures. Par contre, la rétropropagation s'avère délicate dans son utilisation pour entraîner les réseaux de neurones artificiels biologiquement réalistes.

Définitions et concepts de base

Le sujet de cette thèse est le développement d'algorithmes d'apprentissage supervisé pour un modèle neuronal particulier appelé le quantron (Labib, 1999; Connolly & Labib, 2009). Ce modèle, issu de la représentation de la diffusion synaptique par des processus stochastiques, possède des caractéristiques rappelant à la fois le perceptron (McCulloch & Pitts, 1943) et les neurones impulsionnels (Gerstner, 1995). Pour faciliter la compréhension de la nature de notre recherche, nous décrivons d'abord le quantron de manière détaillée. Par la suite, nous expliquons la motivation de notre étude du quantron en le comparant au perceptron.

La figure 1 présente une vue schématique d'un quantron qui possède deux entrées notées x_1 et x_2, qui peuvent prendre des valeurs strictement positives (on généralise aisément à un nombre plus élevé d'entrées). Ces entrées, ainsi que les paramètres $w_1, s_1, \theta_1, w_2, s_2, \theta_2$, déterminent la forme de la fonction d'activation du quantron, qui représente l'évolution temporelle du potentiel électrique dans le corps du neurone. Chaque valeur d'entrée représente l'intervalle de temps entre les arrivées successives de vagues de neurotransmetteurs, des mo-

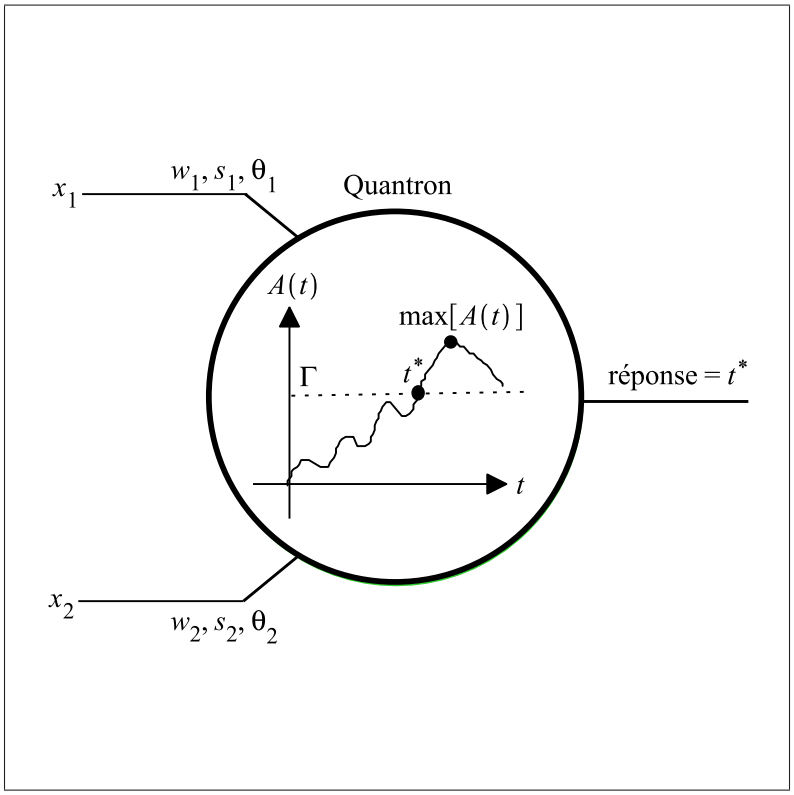

Figure 1 Fonction d'activation, maximum et temps d'activation d'un quantron. La forme de la fonction d'activation dépend des entrées x_1, x_2 et des paramètres $w_1, s_1, \theta_1, w_2, s_2, \theta_2$. Le maximum de la fonction d'activation $\max[A(t)]$ et le temps d'activation t^* sont illustrés par des points. Le seuil Γ est illustré par une ligne pointillée.

lécules responsables de la transmission de signaux d'un neurone à un autre à travers la fente synaptique. L'arrivée des neurotransmetteurs à la synapse induit un signal appelé potentiel post-synaptique, et la sommation de ces potentiels donne naissance à la fonction d'activation. Cette dernière, notée $A(t)$, est donnée par

$$A(t) = \sum_{j=1}^{2} \sum_{i=0}^{N_j - 1} w_j p_j \left(t - \theta_j - i x_j\right),$$

où N_j est le nombre de potentiels post-synaptiques issus de la $j^{\text{ième}}$ synapse du neurone $(j = 1, 2)$, et où la fonction $p_j(\cdot)$ représente la forme de ces potentiels. Les paramètres w_j et θ_j représentent respectivement la hauteur des potentiels et le délai initial d'un train de potentiels. Labib (1999) a modélisé la diffusion des neurotransmetteurs par un mouvement

brownien géométrique et a obtenu

$$p_j(t) = \begin{cases} 2Q\left(\frac{\ln a}{\sqrt{t}}\right) & \text{si } 0 \leq t < s_j, \\ 2Q\left(\frac{\ln a}{\sqrt{s_j}}\right) - 2Q\left(\frac{\ln a}{\sqrt{t-s_j}}\right) & \text{si } s_j \leq t < 2s_j, \\ 0 & \text{sinon,} \end{cases}$$

où $Q(z)$ est la probabilité qu'une variable aléatoire suivant une loi normale centrée et réduite prenne une valeur supérieure à z, et où la constante a représente la largeur de la fente synaptique. Le paramètre s_j représente la demi-largeur d'un potentiel.

Lorsque la fonction $A(t)$ atteint le seuil Γ au temps t^*, appelé temps d'activation, le quantron transmet comme réponse la valeur de t^*. Si le seuil n'est jamais atteint, le quantron ne transmet rien. Cette réponse permet de construire un réseau où des potentiels post-synaptiques sont propagés à travers des couches successives de quantrons.

Si l'on s'intéresse uniquement à l'état du quantron (actif ou inactif), la réponse devient binaire et est déterminée par le maximum de la fonction d'activation : si $\max[A(t)] \geq \Gamma$, le quantron est actif, sinon il est inactif.

Notons que t^* est la plus petite solution positive de l'équation $A(t) = \Gamma$. Puisque cette équation ne possède pas de solution analytique, on doit utiliser une méthode de calcul numérique pour déterminer approximativement la réponse du quantron. Il en va de même pour identifier la valeur du maximum. On peut également simplifier le modèle en utilisant une forme plus simple pour les potentiels post-synaptiques. Par exemple, on peut utiliser des potentiels de forme rectangulaire (de largeur s_j). Dans ce cas, on a

$$p_j(t) = \begin{cases} 1 & \text{si } 0 \leq t < s_j, \\ 0 & \text{sinon.} \end{cases}$$

Pour justifier l'intéret du quantron, nous le comparons maintenant au perceptron pour la résolution du problème du OU-exclusif (XOR). La figure 2 présente une version de ce problème de classification avec des valeurs strictement positives pour les entrées x_1 et x_2.

On trouve qu'il est possible de résoudre le problème du OU-exclusif à l'aide du quantron, et ce avec différentes assignations de valeurs aux paramètres (Labib, 1999). Par exemple, on peut utiliser les valeurs $w_1 = 1, w_2 = -1, s_1 = s_2 = 1.5, \theta_1 = \theta_2 = 0$ associées au seuil $\Gamma = 0.75$ pour un quantron possédant N potentiels rectangulaires pour chaque entrée ($N_1 = N_2 = N$). Ainsi, il faut six paramètres pour séparer correctement les exemplaires d'entrée (sept en comptant le seuil).

En comparaison, un seul perceptron à deux entrées ne peut pas résoudre le problème du

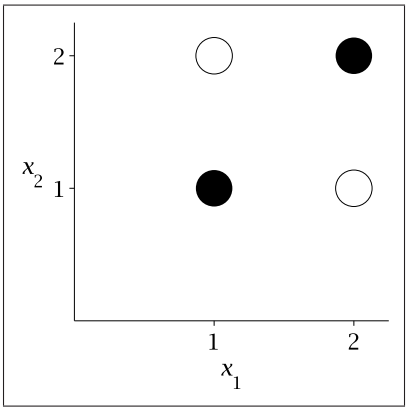

Figure 2 Problème du OU-exclusif. Pour résoudre le problème, le quantron doit s'activer lorsque $(x_1, x_2) = (1, 2)$ ou $(2, 1)$, et il doit rester inactif lorsque $(x_1, x_2) = (1, 1)$ ou $(2, 2)$. Les exemplaires d'entrée pour lesquels le quantron doit s'activer sont représentées par des ronds blancs, et ceux pour lesquels il doit rester inactif sont représentées par des ronds noirs.

OU-exclusif (Haykin, 1999). Un réseau de perceptrons avec une couche cachée de deux neurones ainsi qu'un neurone de sortie a besoin de neuf paramètres pour séparer les exemplaires d'entrée, alors qu'il n'en faut que sept pour le quantron. D'un point de vue théorique, la résolution d'un problème avec moins de paramètres et une structure de réseau plus simple (un seul neurone au lieu de trois) laisse croire que le quantron pourrait avoir une meilleure capacité de généralisation que le perceptron. Pour affirmer cela, nous nous basons sur une règle empirique qui stipule que le nombre d'exemplaires d'entrée nécessaires à une bonne généralisation est proportionnel au nombre de paramètres du réseau (Haykin, 1999). Plus précisément, on a $N = O\left(\frac{W}{\epsilon}\right)$, où :

- ϵ est le taux d'erreur maximal toléré sur de nouvelles valeurs d'entrée ;
- W est le nombre de paramètres du réseau ;
- N est le nombre minimal d'exemplaires de valeurs d'entrée à utiliser pour entrainer le réseau avec un taux d'erreur ϵ fixé ;
- $O(\cdot)$ désigne une relation d'ordre asymptotique.

Il est important de noter que ce résultat dépend de l'existence d'un algorithme d'apprentissage efficace, tel la rétropropagation de l'erreur pour les réseaux de perceptrons avec fonction de transfert lisse. De plus, rien ne garantit que le résultat s'applique directement aux réseaux de quantrons. Par conséquent, avant de pouvoir effectuer une comparaison exhaustive de la capacité de classification des réseaux de perceptrons et de quantrons, il est primordial de proposer un algorithme d'apprentissage adapté au fonctionnement du quantron.

Éléments de la problématique

Nous venons de souligner l'intérêt du quantron comme élément de base biologiquement réaliste des réseaux de neurones, avec une possibilité d'amélioration des performances en classification par rapport au perceptron. Toutefois, il est difficile d'entraîner un réseau de quantrons par rétropropagation de l'erreur. En effet, la rétropropagation fonctionne en modifiant itérativement la valeur des paramètres à rebours dans le réseau. Ce processus est guidé par la descente du gradient d'une surface qui représente l'erreur entre les réponses actuelles du réseau et les réponses désirées au problème de classification. L'utilisation de t^* et de $\max[A(t)]$ introduit des points de discontinuité dans la surface d'erreur et dans ses dérivées, ce qui perturbe la convergence de la descente du gradient. De plus, il n'est pas évident d'exprimer les dérivées de t^* et de $\max[A(t)]$ par rapport aux paramètres, étape pourtant nécessaire à l'implémentation efficace de l'algorithme de rétropropagation. Notons que ce problème se pose même pour un seul quantron.

La présence de discontinuités dans la surface d'erreur a déjà été rencontrée dans les travaux sur le perceptron à cause de la réponse binaire des neurones. Ne se prêtant pas à la rétropropagation, cette dernière a été remplacée par une réponse de type sigmoïde qui adoucit la transition brusque de l'état inactif à l'état actif, générant ainsi une surface d'erreur lisse. Cette solution simple ne s'applique pas directement à t^* et à $\max[A(t)]$. Cependant, elle fournit un point de départ à notre recherche : en substituant ces termes calculés numériquement par des expressions de nature analytique, nous pourrions éliminer les discontinuités de la surface d'erreur et ainsi faciliter l'apprentissage du quantron et d'un réseau de quantrons.

Objectifs de recherche

Après avoir présenté le quantron et les difficultés inhérentes à son apprentissage, nous pouvons maintenant énoncer les objectifs de notre recherche. Notre but est d'adapter la descente du gradient à un quantron ainsi que la rétropropagation de l'erreur à un réseau de quantrons. Pour ce faire, nous préconisons une approche analytique : simplifier ou approximer des parties du modèle pour éviter les problèmes d'expression des dérivées et de calcul numérique. En particulier, l'étude du maximum $\max[A(t)]$ est reliée à l'apprentissage d'un seul quantron, et l'étude du temps d'activation t^* vise l'apprentissage d'un réseau de quantrons. Une autre avenue possible est de travailler directement sur la fonction d'activation. En effet, l'utilisation d'une fonction moins complexe peut faciliter l'analyse du quantron.

Bien que nous étudions un neurone artificiel biologiquement réaliste, l'algorithme envisagé pour entraîner un réseau de quantron n'est pas, quant à lui, biologiquement réaliste. Notre intérêt pour la rétropropagation de l'erreur est motivé par des considérations pratiques

propres à l'application en reconnaissance de formes : c'est un algorithme qui a fait ses preuves et dont le comportement est bien compris. Nous cherchons donc à utiliser les avantages de la rétropropagation sans égard à son manque de réalisme.

Revue de littérature

Notre recherche nous a mené à la rédaction de trois articles portant sur différents aspects de l'apprentissage du quantron par une approche analytique. Nous présentons ici une revue de littérature générale qui est inspirée des introductions de nos trois articles. L'accent y est mis sur les résultats de travaux antérieurs pertinents à la compréhension de notre thèse. Par conséquent, nous ne reprenons pas intégralement les citations bibliographiques de chacun des articles.

Des modèles biologiquement réalistes, provenant de la recherche en neuroscience computationnelle, ont déjà été proposés comme neurone artificiel : il s'agit des neurones impulsionnels. En fait, ce terme regroupe divers modèles de complexité variable. Parmi eux, on retrouve les modèles intègre-et-décharge basés sur des équations différentielles (Lapicque, 1907; Hodgkin & Huxley, 1952) et le modèle à réponse impulsionnelle (Gerstner, 1995), qui ont été appliqués, entre autres, en robotique biomimétique (Martinez *et al.*, 2006; Liu *et al.*, 2009; Häusler *et al.*, 2011; Gamez *et al.*, 2012). Par contre, le temps nécessaire pour le calcul de la réponse de ces réseaux rend difficile l'utilisation d'algorithmes d'apprentissage supervisé (Ghosh-Dastidar & Adeli, 2009b). Pour alléger les calculs, une version simplifiée du modèle à réponse impulsionnelle a été intégrée à l'algorithme de rétropropagation de l'erreur, donnant naissance à l'algorithme *Spikeprop* (Bohte *et al.*, 2000, 2002). Cependant, les créateurs de Spikeprop ont reconnu que l'algorithme était sujet à des problèmes de convergence.

D'autres chercheurs ont proposé des modifications à Spikeprop, en relevant aussi la présence de problèmes de convergence (Xin & Embrechts, 2001; Schrauwen & Van Campenhout, 2004; Booij & tat Nguyen, 2005; Wu *et al.*, 2006; McKennoch *et al.*, 2006; Ghosh-Dastidar & Adeli, 2007, 2009a; Delshad *et al.*, 2010; Wakamatsu *et al.*, 2011; Thiruvarudchelvan *et al.*, 2013; Xu *et al.*, 2013). Certains ont spécifiquement étudié les discontinuités de la surface d'erreur et les effets du calcul numérique (Fujita *et al.*, 2008; Takase *et al.*, 2009). À notre connaissance, personne n'a réussi à éviter complètement ces problèmes.

À la lumière de ces travaux, nous relevons une limitation majeure de Spikeprop : la présence de neurones qui n'émettent jamais d'impulsions (appelés neurones silencieux). Des règles heuristiques ont été proposées pour résoudre le problème des neurones silencieux en modifiant les équations de rétropropagation. Cette solution n'est pas entièrement satisfaisante, car elle brise la relation entre le gradient et la surface d'erreur.

D'autres travaux ont porté sur l'apprentissage d'un seul neurone à réponse impulsionnelle (Davis *et al.*, 2003; Gütig & Sompolinsky, 2006; Mohemmed *et al.*, 2011). Cependant, ces derniers n'apportent ni résultat pouvant s'appliquer facilement à un réseau ni façon d'éviter les calculs numériques.

Enfin, nous notons que, pour des modèles simples, on peut parfois aborder les problèmes de classification par l'analyse géométrique de la frontière de décision. C'est le cas, en particulier, pour de petits réseaux de perceptrons (Labib & Khattar, 2010), pour des neurones suivant le modèle d'Ising (Fitzgerald & Sharpee, 2009), et pour le perceptron à impulsions dont les entrées sont des processus de renouvellement (Rowcliffe *et al.*, 2006). Toutefois, les méthodes utilisées dans ces travaux ne s'appliquent pas au quantron ou aux neurones à réponse impulsionnelle. En particulier, le perceptron à impulsions utilise les statistiques temporelles des entrées, comme le délai moyen entre les impulsions. Or, les neurones à réponse impulsionnelle modélisent des phénomènes de transmission rapide d'information dans le cerveau, de sorte que la réponse d'un neurone ne peut directement dépendre de telles statistiques (Maass, 1997).

Pour clore cette revue de littérature, nous notons qu'il existe certaines similitudes entre le quantron et les neurones impulsionnels utilisés dans Spikeprop. Par conséquent, pour justifier l'originalité de notre recherche, nous décrivons dans les trois prochaines sections les neurones impulsionnels, l'algorithme Spikeprop et le problème des neurones silencieux de manière détaillée. En particulier, nous mettons en évidence les différences entre les neurones impulsionnels et le quantron, nous illustrons les problèmes de convergence de Spikeprop, et nous expliquons pourquoi les neurones silencieux nuisent à l'apprentissage dans les réseaux de neurones impulsionnels et de quantrons.

Neurones impulsionnels

Nous comparons ici le modèle de neurone impulsionnel au quantron. Pour s'assurer de la clarté de la comparaison, nous utilisons les mêmes symboles pour représenter des éléments similaires des deux modèles. Par exemple, $A(t)$ désigne à la fois la fonction d'activation d'un neurone impulsionnel et d'un quantron, de même que Γ désigne le seuil. Cette notation diffère de celle de Bohte *et al.* (2002).

La fonction d'activation d'un neurone impulsionnel à deux entrées est donnée par

$$A(t) = \sum_{j=1}^{2} \sum_{i=1}^{N_j} w_{j,i} p(t - x_j - \theta_{j,i}),$$

où la fonction $p(\cdot)$, représentant la forme des potentiels post-synaptiques, est donnée par

$$p(t) = \begin{cases} \frac{t}{s} e^{1-\frac{t}{s}} & \text{si } t \geq 0, \\ 0 & \text{sinon.} \end{cases}$$

Comme pour le quantron, le temps d'activation t^*, c'est-à-dire la plus petite solution de l'équation $A(t) = \Gamma$, est utilisé comme réponse du neurone.

Les différences entre ce neurone et le quantron sont les suivantes :

- l'entrée x_j représente le délai initial d'un train de potentiels et non le délai successif entre les potentiels ;
- chaque potentiel possède son propre paramètre $w_{j,i}$ au lieu d'un seul paramètre pour un train de potentiels ;
- chaque potentiel possède son propre délai $\theta_{j,i}$;
- la forme des potentiels ne possède pas de largeur finie (s ne détermine plus la largeur du potentiel) ;
- dans la version originale de Spikeprop, seul les paramètres $w_{j,i}$ sont entraînés ($d_{j,i} = i$ et s est fixé pour l'ensemble du réseau).

Ainsi, le quantron transmet comme réponse un délai inter-potentiel (l'inverse d'une fréquence) tandis que le neurone impulsionnel transmet un délai initial. Il s'ensuit que, pour le quantron, la valeur de l'entrée x_j influence le chevauchement au sein d'un train de potentiels, alors que, pour le neurone impulsionnel, cette valeur n'affecte que le décalage des trains de potentiels par rapport au temps initial. La figure 3 présente deux frontières de décision, tirées de notre premier article, pour illustrer cette différence en comparant un neurone impulsionnel et un quantron possédant tous deux une fonction $p(t)$ de forme exponentielle décroissante.

On remarque que l'état d'activité du neurone impulsionnel ne varie pas le long des droites $x_2 = x_1 + b$ (où b est un nombre réel), ce qui limite l'intéret de la frontière de décision pour effectuer de la classification. Néanmoins, on voit que ce dernier peut résoudre le problème du OU-exclusif. Cependant, ce n'est pas la frontière de décision basée sur l'état d'activité du neurone qui est utilisé dans les travaux sur Spikeprop. En effet, pour pouvoir rétropropager l'erreur, la classification se base sur la valeur de t^* : l'état actif (respectivement inactif) est remplacé par une activation rapide (respectivement lente). Par exemple, pour résoudre le problème du OU-exclusif, on peut utiliser 10 millisecondes comme temps d'activation désiré pour une activation rapide et 16 millisecondes comme temps d'activation désiré pour une activation lente.

En analysant la fonction d'activation du neurone impulsionnel pour $x_2 = x_1 + b$, on trouve

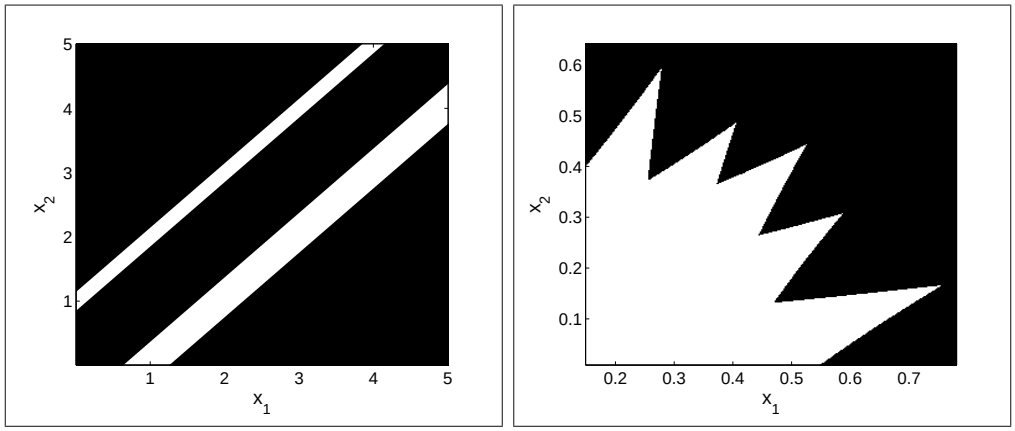

Figure 3 Frontière de décision d'un neurone impulsionnel avec $p(t) = e^{-3t}$ pour $t \geq 0$ (gauche) et d'un quantron avec $p(t) = e^{-t}$ pour $t \geq 0$ (droite). Quatre potentiels sont utilisés pour chaque entrée ($N_1 = N_2 = 4$). La région blanche (respectivement noire) représente les valeurs d'entrées pour lesquelles le neurone est actif (respectivement inactif).

que la valeur de t^* (en fonction de x_1 et x_2) s'exprime comme suit :

$$t^*(x_1, x_2) = t^*(\max(-b, 0), \max(b, 0)) + \min(x_1, x_2).$$

Il est alors impossible de représenter le OU-exclusif tel que décrit par la figure 2, car le temps d'activation $t^*(1, 1)$ est strictement inférieur à $t^*(2, 2)$ lorsque le neurone est actif alors que dans chaque cas il devrait prendre une valeur égale à 16 millisecondes. Cette limitation s'observe aussi dans un réseau de neurones impulsionnels (Sporea & Grüning, 2011), de sorte qu'il faut recourir à une entrée additionnelle (un temps de référence $x_3 = 0$) pour permettre une classification correcte. Cette modification ajoute des paramètres au neurone (ou au réseau). Avec le quantron, cette difficulté ne se présente pas, et l'utilisation d'une entrée supplémentaire n'est pas requise.

Spikeprop

Nous décrivons maintenant l'algorithme Spikeprop, qui est une adaptation de la rétropropagation de l'erreur aux neurones impulsionnels. Le principal obstacle à l'application directe de la rétropropagation est l'inexistence d'une expression analytique pour t^*. En effet, les règles de mise à jour des paramètres en rétropropagation sont basées sur les dérivées partielles de la sortie d'un neurone par rapport à ses paramètres et à ses entrées. Ainsi, pour calculer $\frac{\partial t^*}{\partial w_{j,i}}$ et $\frac{\partial t^*}{\partial x_j}$, les créateurs de Spikeprop ont proposé l'utilisation d'une approximation linéaire de la fonction d'activation proche du seuil. Cette approche est, à toute fin pratique, équivalente à l'utilisation de la dérivée implicite de $A(t) = \Gamma$ (Yang *et al.*, 2012). Pour un

neurone impulsionnel à deux entrées, on trouve

$$\frac{\partial t^*}{\partial w_{j,i}} = -\left.\frac{\frac{\partial}{\partial w_{j,i}}A(t)}{\frac{\partial}{\partial t}A(t)}\right|_{t=t^*} = -\frac{p(t^* - x_j - \theta_{j,i})}{\sum_{j=1}^{2}\sum_{i=1}^{N_j} w_{j,i}p'(t^* - x_j - \theta_{j,i})}$$
$$\frac{\partial t^*}{\partial x_j} = -\left.\frac{\frac{\partial}{\partial x_j}A(t)}{\frac{\partial}{\partial t}A(t)}\right|_{t=t^*} = \frac{\sum_{i=1}^{N_j} w_{j,i}p'(t^* - x_j - \theta_{j,i})}{\sum_{j=1}^{2}\sum_{i=1}^{N_j} w_{j,i}p'(t^* - x_j - \theta_{j,i})}$$

en dérivant $A(t)$ par morceaux. On voit que ces dérivées dépendent de la valeur de t^*, et elles prennent la valeur 0 si le neurone ne s'active pas. Dans Spikeprop, t^* est calculé numériquement en évaluant $A(t)$ pour $t = 0, \Delta t, 2\Delta t, ...$ jusqu'à l'atteinte ou le dépassement du seuil. Ainsi, la valeur de t^* est approximée avec un pas de temps Δt. En se basant sur des considérations biologiques, les créateurs de Spikeprop ont fixé Δt à 0.1 milliseconde.

Nous décrivons maintenant un premier problème qui affecte Spikeprop : les discontinuités de t^* (Takase *et al.*, 2009). La figure 4 montre un exemple de fonction d'activation où une petite variation du paramètre w associé à un potentiel entraîne un grand changement dans la valeur de t^*.

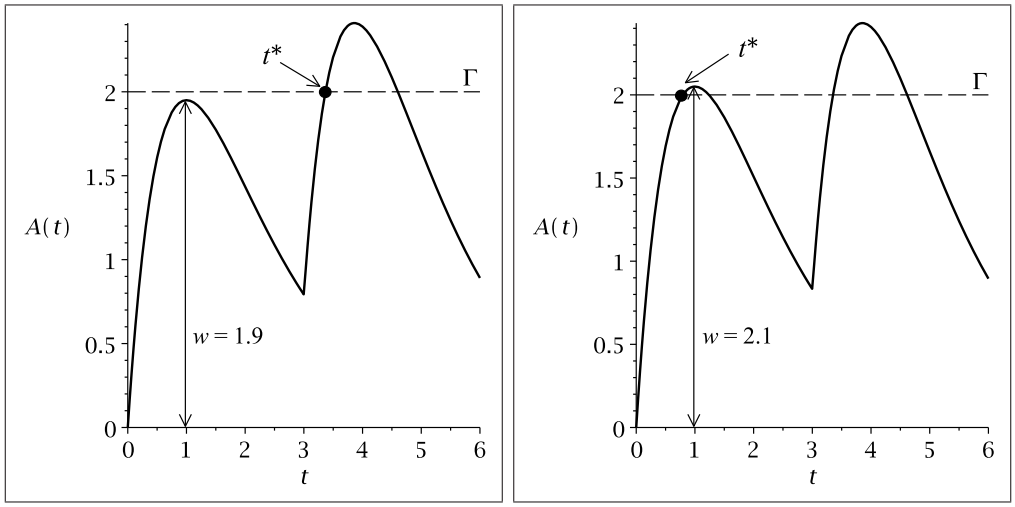

Figure 4 Fonction d'activation d'un neurone impulsionnel. Le paramètre w représente la hauteur du premier potentiel post-synaptique qui est généré au temps $t = 0$. Le seuil Γ est fixé à 2. À gauche, $w = 1.9$ ne permet pas au premier potentiel d'atteindre le seuil, alors qu'à droite, $w = 2.1$ lui permet d'atteindre le seuil.

La figure 5 illustre la relation entre w et t^* (à gauche), et présente une courbe d'erreur définie par $E = (t^* - 2.5)^2$, où 2.5 représente la valeur désirée pour t^* (à droite). La courbe d'erreur se généralise en surface d'erreur lorsque plusieurs paramètres sont modifiés simultanément. Avec cette courbe d'erreur, la descente du gradient, sur laquelle se base l'apprentissage par rétropropagation, ne converge pas. Bien que nous présentions ici une situation

pathologique (t^* ne peut jamais prendre la valeur 2.5), de manière générale les discontinuités nuisent à la convergence de la descente du gradient.

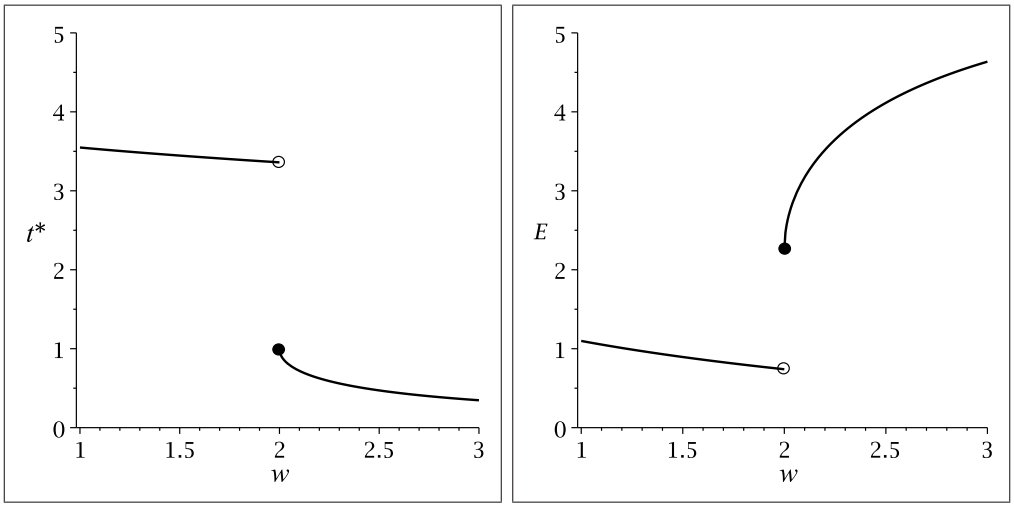

Figure 5 Discontinuité de t^*. À gauche, la courbe qui décrit la variation de t^* en fonction de w présente une discontinuité en $w = 2$. À droite, la courbe d'erreur $E = (t^* - 2.5)^2$ ne peut être minimisée correctement par descente du gradient, car l'équation $\frac{dE}{dw} = 0$ ne possède pas de solution.

Un deuxième problème qui affecte Spikeprop est le recours au calcul numérique (Fujita *et al.*, 2008). En effet, l'approximation de t^* avec un pas de temps fixé perturbe la courbe d'erreur. La figure 6 illustre la courbe d'erreur obtenue en modifiant la fonction d'activation présentée à la figure 4 comme suit : on exprime la fonction d'activation avec le premier potentiel débutant soit en $t = 0$, soit en $t = 0.5$. Ces deux cas correspondent à une valeur d'entrée x pouvant prendre les valeurs 0 ou 0.5. Pour $x = 0$, la valeur désirée pour t^* est fixée à 1.4. Pour $x = 0.5$, la valeur désirée est fixée à 3.9. La courbe d'erreur est définie par $E = (t^*(x = 0) - 1.4)^2 + (t^*(x = 0.5) - 3.9)^2$, et elle est calculée pour un pas de temps Δt fixé soit à 0.1, soit à 0.02. On remarque que, pour $\Delta t = 0.1$, la courbe d'erreur est irrégulière ; ceci est dû au calcul numérique. Pour $\Delta t = 0.02$, les irrégularités ont une amplitude plus faible, mais non négligeable. Ces irrégularités limitent l'utilisation des valeurs de la courbe d'erreur pour guider la descente du gradient (par exemple, en effectuant une recherche linéaire).

Une autre sorte de perturbation, liée à un phénomène d'instabilité (*hair-trigger* en anglais), peut se produire lorsque la fonction d'activation atteint tout juste le seuil (Maass, 1997; Booij & tat Nguyen, 2005). Dans la figure 7, nous reprenons le calcul de t^* basé sur la fonction d'activation de la figure 4 avec $w = 2.001$. En faisant varier le délai x du premier potentiel entre 0 et 1 et en calculant t^*, nous observons plusieurs transitions de valeurs basses (activation provoquée par le premier potentiel) vers des valeurs élevées (activation provoquée

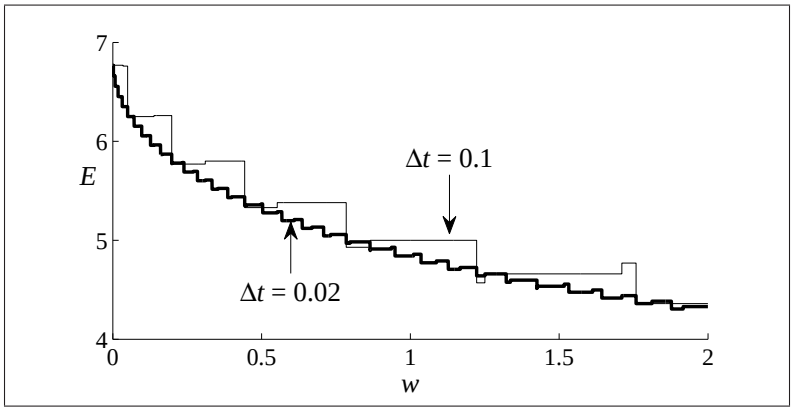

Figure 6 Perturbation de la courbe d'erreur. L'erreur est déterminée par t^*, qui est calculé avec un pas de temps $\Delta t = 0.1$ (ligne mince) et $\Delta t = 0.02$ (ligne épaisse). La courbe d'erreur exacte serait lisse et monotone décroissante pour $w < 2$, comme dans la figure 5.

par le deuxième potentiel). Le deuxième potentiel provoque l'activation du neurone lorsque la partie du premier potentiel dépassant le seuil se trouve entre les temps $k\Delta t$ et $(k+1)\Delta t$ pour un certain entier k. Booij & tat Nguyen (2005) ont montré que l'apprentissage du OU-exclusif par un neurone impulsionnel est sujet à de telles instabilités.

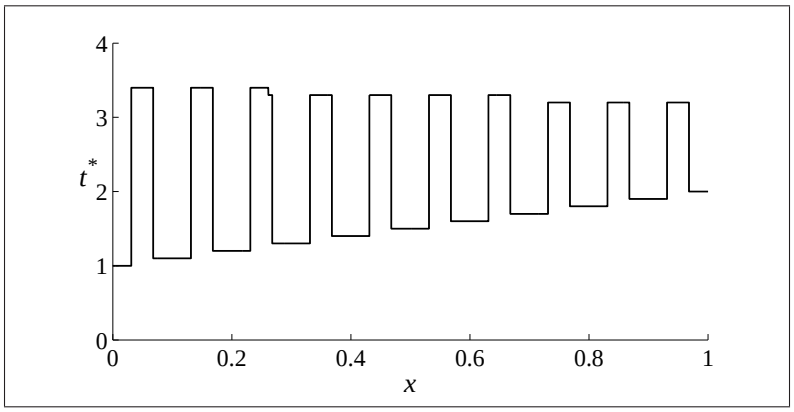

Figure 7 Perturbation de t^* avec $w = 2.001$. Lorsqu'un délai x $(0 \leq x \leq 1)$ est appliqué au premier potentiel de la fonction d'activation, la courbe qui décrit la variation de t^* en fonction de x présente des transitions entre des valeurs basses et élevées.

À la lumière de cette analyse, nous considérons que les discontinuités et les perturbations rencontrées dans Spikeprop ne doivent pas être ignorées. Notre approche analytique, qui a pour but de contourner ces deux problèmes, est donc pertinente pour l'étude du quantron.

Neurones silencieux

Nous abordons maintenant le problème des neurones silencieux qui a été mentionné par plusieurs auteurs dans les travaux sur Spikeprop. Un neurone silencieux est un neurone qui ne s'active pour aucun exemplaire d'entrée. Puisque les dérivées $\frac{\partial t^*}{\partial w_{j,i}}$ et $\frac{\partial t^*}{\partial x_j}$ prennent une valeur nulle lorsque le neurone est inactif, aucune modification aux paramètres d'un neurone silencieux n'est effectuée après la présentation d'un exemplaire, et aucune rétropropagation ne peut avoir lieu de ce neurone vers les neurones antérieurs. Donc, pendant l'apprentissage, un neurone silencieux tend à rester silencieux, à moins que d'autres chemins de rétropropagation ne viennent modifier suffisamment les neurones antérieurs à ce dernier. Enfin, il est possible qu'un neurone qui n'est pas silencieux initialement le devienne éventuellement.

Dans un réseau de neurones avec une seule couche cachée et un neurone de sortie silencieux, l'apprentissage du réseau entier est bloqué. Notons qu'en classification, l'utilisation du maximum de la fonction d'activation pour le neurone de sortie permet d'éviter un tel blocage. Si c'est plutôt un neurone caché qui est silencieux, le réseau se comporte comme s'il y avait un neurone en moins.

On voit donc que la présence de neurones silencieux réduit la capacité de classification du réseau, ce qui peut le rendre incapable de converger vers une solution acceptable lors de l'apprentissage (Ghosh-Dastidar & Adeli, 2007). Par conséquent, il est important de résoudre ce problème, qui affecte autant les réseaux de neurones impulsionnels que les réseaux de quantrons. Deux solutions ont été proposées au problème des neurones silencieux : la modification des règles de rétropropagation dans le but de forcer les neurones à s'activer (Schrauwen & Van Campenhout, 2004; Booij & tat Nguyen, 2005; Wu *et al.*, 2006) et l'utilisation du maximum de la fonction d'activation comme réponse de substitution lorsque le seuil n'est pas atteint (Ghosh-Dastidar & Adeli, 2007, 2009a). La première solution brise le lien entre le gradient et la surface d'erreur, alors que la seconde modifie les calculs dans le réseau. Notre approche analytique propose une autre voie : le lissage des discontinuités de la surface d'erreur. Si le lissage est raffiné en cours d'apprentissage, la surface d'erreur approximée se rapproche progressivement de la vrai surface d'erreur.

Une remarque finale s'impose au sujet de la capacité de classification. Comme mentionné précédemment, l'utilisation d'un réseau avec moins de paramètres permet théoriquement une meilleure généralisation. Cependant, le choix du nombre de paramètres et du modèle de neurone se fait de manière contrôlée, avant l'apprentissage. En général, plusieurs configurations de réseau sont évaluées dans le cadre d'un plan d'expérience. La perte de capacité due à l'initialisation des paramètres du réseau ou à l'évolution des paramètres en cours d'apprentissage est indésirable, car elle n'est pas contrôlable.

Fil conducteur entre les articles

Nous décrivons maintenant le lien qui unit les trois articles compris dans notre thèse, et nous soulignons leur apport à la littérature existante.

Dans le premier article, intitulé *On the Learning Potential of the Approximated Quantron*, nous trouvons une approximation analytique du maximum de la fonction d'activation du quantron, permettant ainsi l'apprentissage par descente du gradient. L'originalité de cet article consiste en la démonstration qu'une approximation analytique peut améliorer les résultats d'un algorithme qui tente de minimiser directement la surface d'erreur.

Dans le deuxième article, intitulé *Learning Algorithms for a Specific Configuration of the Quantron*, nous montrons que, dans un cas particulier, il est possible d'exprimer de manière analytique la fonction d'activation du quantron, ce qui rend possible une description géométrique de sa frontière de décision. Ensuite, nous développons deux algorithmes d'apprentissage qui trouvent rapidement une solution presque optimale à un problème de classification, dans la mesure où il existe une solution exacte, c'est-à-dire sans erreur de classification. Ces algorithmes constituent indéniablement un apport original à l'étude du quantron.

Dans le troisième article, intitulé *New Approximation Method for Smooth Error Backpropagation in a Quantron Network*, nous obtenons une approximation analytique du temps d'activation du quantron, laquelle est utilisée pour adapter l'algorithme de rétropropagation de l'erreur à un réseau de quantrons. L'aspect innovateur de cet article est de contribuer à la résolution du problème des neurones silencieux.

Finalement, on voit se dessiner le fil conducteur de nos articles et de notre thèse. L'utilisation d'expressions analytiques pour le maximum de la fonction d'activation, pour la fonction d'activation elle-même et pour le temps d'activation permet de développer de nouveaux algorithmes d'apprentissage pour le quantron et pour un réseau de quantrons.

En comparant ces articles à l'ensemble des travaux antérieurs présentés dans la revue de littérature, on distingue clairement l'originalité et le caractère innovateur de notre recherche. En effet, nos méthodes viennent combler certaines lacunes des algorithmes d'apprentissage qui ont été appliqués aux réseaux de neurones impulsionnels : nous lissons les discontinuités dans la surface d'erreur, ce qui permet, entre autres, de contourner le problème des neurones silencieux, et nous évitons aussi le recours à une méthode numérique. De plus, dans un cas particulier, nous pouvons décrire géométriquement les frontières de décision du quantron à l'aide d'une expression analytique de la fonction d'activation. Ces résultats viennent enrichir le corpus de la recherche sur l'apprentissage supervisé des réseaux de neurones biologiquement réalistes. En particulier, les résultats de notre troisième article pourraient être adaptés aux réseaux de neurones impulsionnels.

Plan de la thèse

Dans la suite de ce document, nous reprenons d'abord chacun de nos articles. Le premier chapitre contient l'article *On the Learning Potential of the Approximated Quantron*, le deuxième chapitre contient l'article *Learning Algorithms for a Specific Configuration of the Quantron*, et le troisième chapitre contient l'article *New Approximation Method for Smooth Error Backpropagation in a Quantron Network*. Chacun de ces chapitres débute par une courte introduction qui situe l'article en question dans le cadre général de notre recherche et se termine par un résumé des principaux résultats. Le quatrième chapitre comprend une discussion générale portant sur l'ensemble de nos travaux. Enfin, le cinquième chapitre présente la conclusion de notre thèse.

Note : Nous présentons les articles tels que publiés en anglais. Pour clarifier certains passages, nous avons ajouté quelques notes de bas de page en français. Les annexes des articles se retrouvent en annexe de la thèse.

CHAPITRE 1

ARTICLE 1 : ON THE LEARNING POTENTIAL OF THE APPROXIMATED QUANTRON

Présentation : Ce chapitre reprend un article publié dans *International Journal of Neural Systems* par R. Labib et S. de Montigny (vol. 22, no. 3, 2012).

Dans le cadre de notre approche analytique de l'apprentissage du quantron, nous nous intéressons d'abord au maximum de la fonction d'activation. On note qu'il n'existe pas d'expression analytique pour $\max[A(t)]$. De plus, considéré comme une fonction des paramètres du quantron, le maximum est non différentiable. Nous proposons donc une approximation analytique du maximum dans le but d'entraîner le quantron par descente du gradient. Nous démontrons l'utilité notre approximation en améliorant les résultats d'un algorithme de recherche directe par motif.

Abstract : The quantron is a hybrid neuron model related to perceptrons and spiking neurons. The activation of the quantron is determined by the maximum of a sum of input signals, which is difficult to use in classical learning algorithms. Thus, training the quantron to solve classification problems requires heuristic methods such as direct search. In this paper, we present an approximation of the quantron trainable by gradient search. We show this approximation improves the classification performance of direct search solutions. We also compare the quantron and the perceptron's performance in solving the IRIS classification problem.

Keywords : Quantron; Spiking Neuron; Learning Algorithm; Gradient Search; IRIS Classification Problem.

1.1 Introduction

The quantron is an innovative type of artificial neuron modeled after chemical neurotransmitters release in the synaptic cleft by diffusion processes (Labib, 1999; Labib *et al.*, 2005). The quantron shares some similarities with the advanced neuron model called spiking neuron (Maass, 1997; Ghosh-Dastidar & Adeli, 2009b; Nichols *et al.*, 2010; Jahangiriand & Durand, 2011), but the differences are crucial. The activation function of both the quantron and the spiking neuron is expressed as the sum of electrical signals varying in time, but their inputs have distinct forms. The input of a spiking neurons is usually a single firing

time or a sequence of firing times (Gerstner, 1995; Gerstner *et al.*, 1996; Ruf & Schmitt, 1997; Iglesias & Villa, 2008; Rossello *et al.*, 2009; Soltic & Kasabov, 2010; Vidybida, 2011), whereas the quantron uses regular firing delays similar to the firing rates used in classical perceptron networks (Rumelhart *et al.*, 1986; Haykin, 1999). However, training a quantron network in a supervised manner is difficult due to the presence of discontinuities in the error function which prevent direct application of gradient descent methods like the backpropagation algorithm. This situation is related to the problem of silent neurons encountered in the development of supervised learning algorithms for networks of spiking neurons, such as Spikeprop (Bohte *et al.*, 2002; Davis *et al.*, 2003; Schrauwen & Van Campenhout, 2004; Booij & tat Nguyen, 2005; McKennoch *et al.*, 2006; Ghosh-Dastidar & Adeli, 2007, 2009a). Another way to achieve supervised learning of spiking neurons is to use hybrid learning algorithms based on Spike Timing Dependent Plasticity (STDP) (Johnston *et al.*, 2010; Strain *et al.*, 2010; Wang *et al.*, 2011; Luque *et al.*, 2011), although in these methods supervised learning cannot easily scale on multiple neuron layers. Evolutionary algorithms have been used to allow the training of realistic neural networks (Belatreche *et al.*, 2003; Pavlidis *et al.*, 2005; Schliebs *et al.*, 2010; Vásquez & Garro, 2011), however these heuristic methods do not take advantage of the neuron's particular structure and thus lack the efficiency of the backpropagation algorithm.

As a first step towards adapting backpropagation to a network of quantrons, we focus on the behavior of a single neuron. The activation function of the quantron, noted $A(t)$, determines when the neuron fires neurotransmitters or stays silent via its maximum $\max[A(t)]$. Since this maximum is nondifferentiable in neuron parameters, we must use heuristic methods such as direct search to train the quantron. The main goal of this research is to improve upon direct search with an approximation of $\max[A(t)]$ for which we can apply a gradient search method. A wavelet-based multiscale scheme has been used to this end (Connolly & Labib, 2009), however it has not been applied yet in a learning algorithm.

Interestingly, in Multi-Spikeprop (a generalization of Spikeprop for the MuSpiNN model), the time where the internal state of a spiking neuron reaches its maximum has been used as the output value of inactive neurons to circumvent the silent neuron problem during learning (Ghosh-Dastidar & Adeli, 2009a). Noting this time τ, we have $\max[A(t)] = A(\tau)$. However, no analytical approximation is proposed to apply gradient descent directly to this time value.

Some supervised learning algorithms have been developped for single spiking neurons. The tempotron learns decision rules for spike patterns by using the time at which the maximum of the internal state is reached, but again no analytical approximation is used (Gütig & Sompolinsky, 2006). More recently, a new method was developped to match input and output spike train based on an integral of the absolute difference between a target internal

state and the output internal state, but it seems difficult to extend this method to train networks (Mohemmed *et al.*, 2011).

In this paper, we present the quantron model and we compare it to the spikeprop neuron model. Then, we propose a quadratic approximation to train the quantron with gradient search. We illustrate the potential of this approach in experimentations by showing that training the quantron with the proposed approximation helps improve the performance when solving classification problems.

1.2 Mathematical model

The numerical inputs of the quantron represent firing delays between consecutive potentials stemming from each synapse. For a given synapse, we define $R(t)$ as a function representing an input signal (a train of potentials), expressed likewise :

$$R(t) = \sum_{i=0}^{N-1} wp(t - \theta - ix) \tag{1.1}$$

where :

- N is the number of potentials;
- i is a potential counter;
- w is the weight of the synapse;
- θ is an initial delay;
- x is the input value fed to the synapse;
- $p(t)$ represents the potential's shape.

The function $p(t)$ can be derived from a model of neurotransmitter diffusion (Labib, 1999; Connolly & Labib, 2009), and is given by

$$p(t) = \begin{cases} cQ\left(\frac{\ln a}{\sqrt{t}}\right) & \text{if } 0 \leq t < s, \\ c\left[Q\left(\frac{\ln a}{\sqrt{s}}\right) - Q\left(\frac{\ln a}{\sqrt{t-s}}\right)\right] & \text{if } s \leq t < 2s, \\ 0 & \text{otherwise,} \end{cases} \tag{1.2}$$

where :

- $Q(z)$ is the probability that a random variable with standard normal distribution is greater than z;
- s is a parameter representing the half-width of the potential;
- the constant a is the width of the synaptic cleft which neurotransmitters have to cross to reach receptors on the neuron's dendrite;

- $c = 1/Q\left(\frac{\ln a}{\sqrt{s}}\right)$ gives a potential of unit height.

For computational tractability, we use surrogate potential functions (de Montigny & Labib, 2011). Figure 1.1 shows an example of both a biologically realistic and a rectangular potential function. For a rectangular potential of total width s, we have

$$p(t) = \begin{cases} 1 & \text{if } 0 \leq t < s, \\ 0 & \text{otherwise.} \end{cases} \tag{1.3}$$

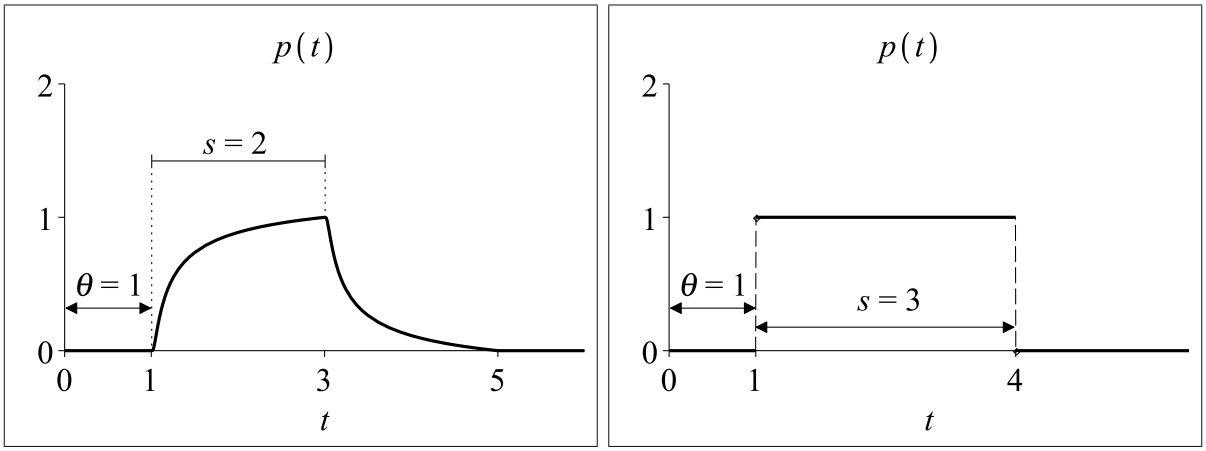

Figure 1.1 Examples of $p(t)$: realistic potential (left) with a rising part and a falling part both of width $s = 2$ (the constant a was set to 1.4918), and rectangular potential (right) of total width $s = 3$. The functions are shifted to the right with a delay $\theta = 1$.

Figure 1.2 shows the effect of x on $R(t)$. The overlapping of potentials represents temporal summation in a single synapse.

The activation function $A(t)$ of the quantron is obtained by adding the potential trains coming from M inputs :

$$A(t) = \sum_{j=1}^{M} \sum_{i=0}^{N_j - 1} w_j p_j(t - \theta_j - i x_j) \tag{1.4}$$

where $N_j, w_j, \theta_j, x_j, p_j$ are particular to the jth input. A common shape is used for all p_j, parameterized by a width parameter s_j. The additive form of $A(t)$ represents spatial summation over all the input synapses.

The output of the quantron is defined as the first time where the activation function reaches a certain threshold Γ. However, we can use the quantron as a binary classifier. Indeed, if $\max[A(t)]$ is higher than Γ, the quantron fires neurotransmitters. Otherwise, the quantron stays silent.

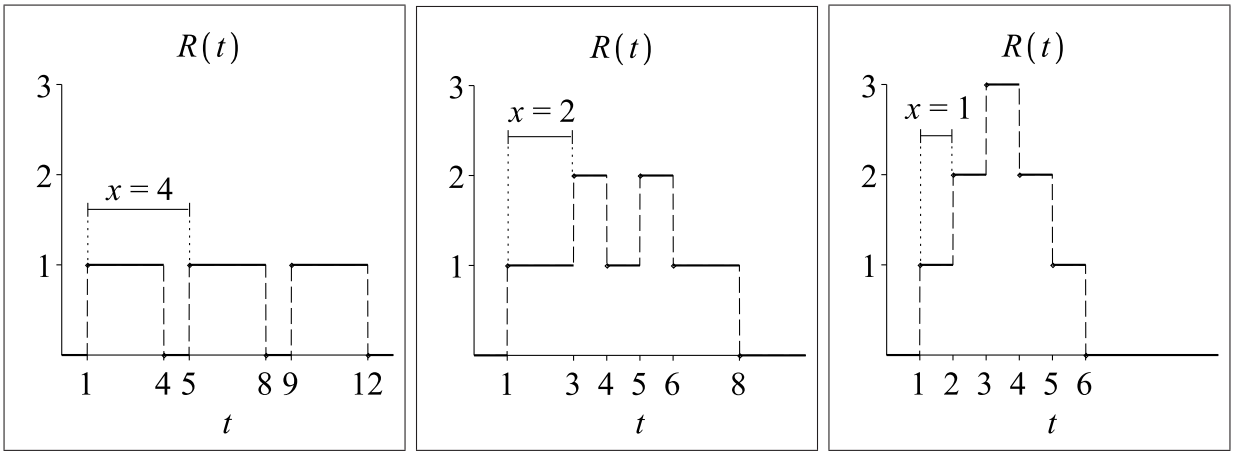

Figure 1.2 Example of $R(t)$ resulting from values $x = 4$, 2 and 1 with parameters $w = 1$, $s = 3$, $\theta = 1$ and $N = 3$.

In this work, we use rectangular functions, given by equation (1.3), to represent potentials, and we only use positive weights values. Thus, we can calculate $\max[A(t)]$ analytically by evaluating $A(t)$ only at the onset time of potentials. With this simplification, we avoid determining $\max[A(t)]$ with a time-consuming numerical algorithm.

To solve a classification problem with the quantron, we minimize the following error function :

$$E_{DS} = \sum_{k=1}^{n} \max\left(\frac{\Gamma - o_k}{d_k \Gamma}, 0\right) \tag{1.5}$$

where :

- n is the number of examples to classify;
- o_k corresponds to $\max[A(t)]$ evaluated with the kth example $(x_1, \ldots, x_M)_k$;
- d_k is an encoding of the desired state of the quantron for the kth example.

The firing state corresponds to $d_k = 1$ and the silent state to $d_k = -1$.

Since E_{DS} is nondifferentiable in the learning parameters w_j, s_j, θ_j, it must be minimized with an advanced optimization method. In this work we use a direct search method, namely a pattern search (Lewis & Torczon, 2000) implementation available in Matlab (the patternsearch function, in 64 bit version 7.10).

1.3 Comparison to the spikeprop neuron

To motivate the originality of this work, we compare the quantron to the neuron model used with the Spikeprop algorithm (Bohte *et al.*, 2002; Schrauwen & Van Campenhout, 2004). Both models are based on the summation of post-synaptic potentials. In both cases, the

neuron's output is defined as the first time where the sum of potentials reaches the neuron's threshold. However, the inputs are used differently in both models.

In the spikeprop neuron model, a single input is propagated to K synaptic terminals, and the contribution from the input x to the activation function is given by

$$V(t) = \sum_{k=1}^{K} w_k p_k(t - x - \theta_k) \tag{1.6}$$

where w_k and θ_k are respectively the weight and delay associated to the kth synaptic terminal. The functions $p_k(t)$ have a similar form, but they possess a decay rate parameter τ_k. This expression is the counterpart to equation (1.1) for the quantron.

In the plain Spikeprop algorithm, only the weights are trained; the delays are usually fixed using the rule $\theta_k = k$, and all the decay parameters are set to a constant value during training (Bohte *et al.*, 2002; Booij & tat Nguyen, 2005). An improved version of the algorithm allows delays and decay rates to be trained (Schrauwen & Van Campenhout, 2004).

Figures 1.3 to 1.8 present images produced by the spikeprop neuron and the quantron with input patterns (x_1, x_2) forming a cartesian grid. In those images, white pixels correspond to a firing state, and black pixels to a silent state. To obtain these decision boundaries, we used the following potential function in each model :

$$p(t) = \begin{cases} e^{-st} & \text{if } t \geq 0, \\ 0 & \text{if } t < 0. \end{cases} \tag{1.7}$$

To ensure that the maximum of the activation function occurs at the onset of a potential, we allowed only positive weights.

For the spikeprop neuron, we used $s = 3$, $K = 4$ and $\theta_k = k$ for each input. Thus, the neuron's activation function is given by :

$$S(t) = \sum_{k=1}^{4} w_{1,k} e^{-3(t-x_1-k)} + \sum_{k=1}^{4} w_{2,k} e^{-3(t-x_2-k)} \tag{1.8}$$

With a threshold value of 1, we obtain a firing state at coordinates (x_1, x_2) if

$$\max_{\substack{j=1,2 \\ k=1,2,3,4}} \{S(x_j + k)\} \geq 1 \tag{1.9}$$

and otherwise the neuron is silent. For the quantron, we used $s = 1$ for each input.

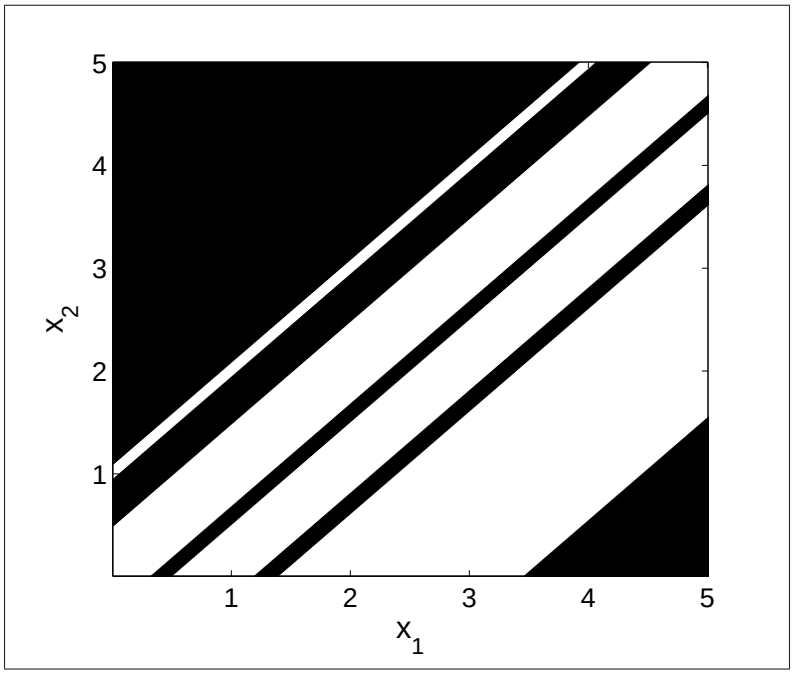

Figure 1.3 Decision boundary obtained with the two-input spikeprop neuron, with weights $w_{1,1} = 0.64$, $w_{1,2} = 0.40$, $w_{1,3} = 0.52$, $w_{1,4} = 0.90$, $w_{2,1} = 0.75$, $w_{2,2} = 0.42$, $w_{2,3} = 0.31$, $w_{2,4} = 0.29$.

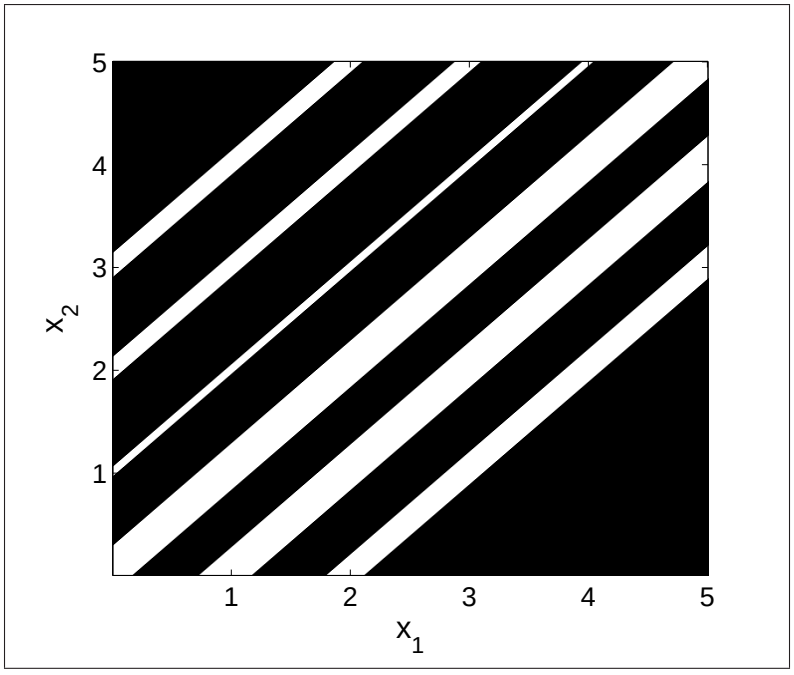

Figure 1.4 Decision boundary obtained with the two-input spikeprop neuron, with weights $w_{1,1} = 0.64$, $w_{1,2} = 0.20$, $w_{1,3} = 0.52$, $w_{1,4} = 0.75$, $w_{2,1} = 0.10$, $w_{2,2} = 0.42$, $w_{2,3} = 0.50$, $w_{2,4} = 0.50$.

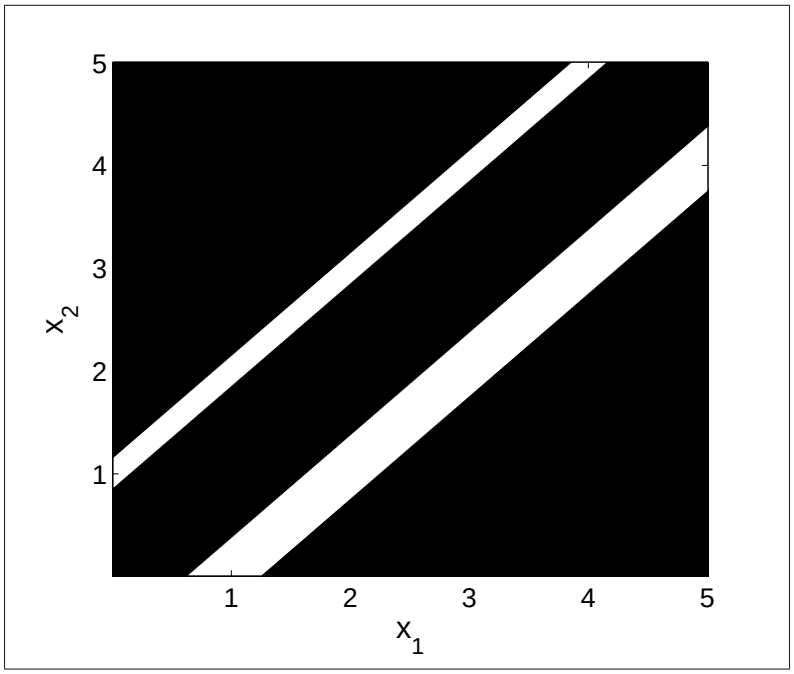

Figure 1.5 Decision boundary obtained with the two-input spikeprop neuron, with weights $w_{1,1} = 0.10$, $w_{1,2} = 0.60$, $w_{1,3} = 0.10$, $w_{1,4} = 0.80$, $w_{2,1} = 0.10$, $w_{2,2} = 0.10$, $w_{2,3} = 0.60$, $w_{2,4} = 0.10$.

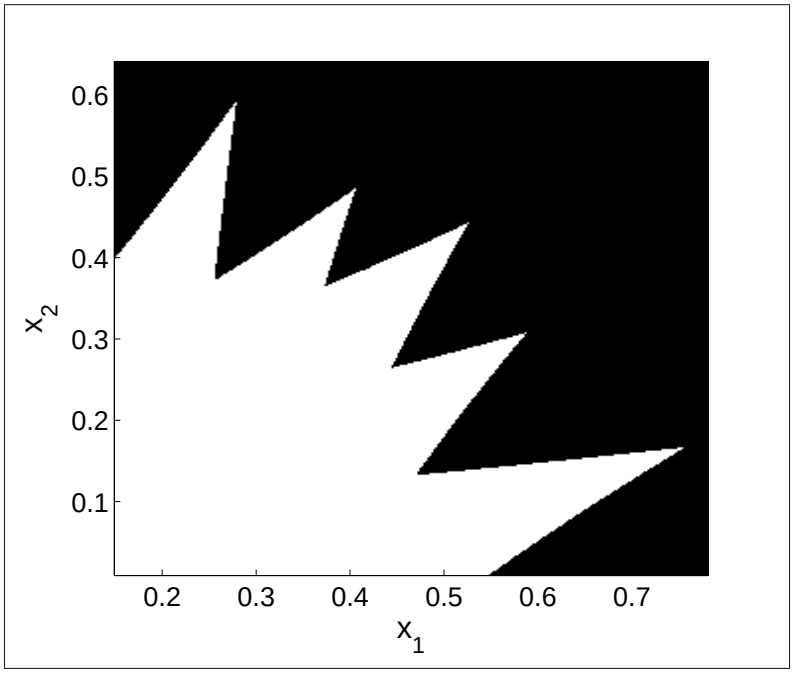

Figure 1.6 Decision boundary obtained with the two-input quantron. We used 4 potentials for each input ($N_1 = N_2 = 4$), and the following parameters : $w_1 = 0.25$, $w_2 = 0.20$, $\theta_1 = 1.00$, $\theta_2 = 1.25$.

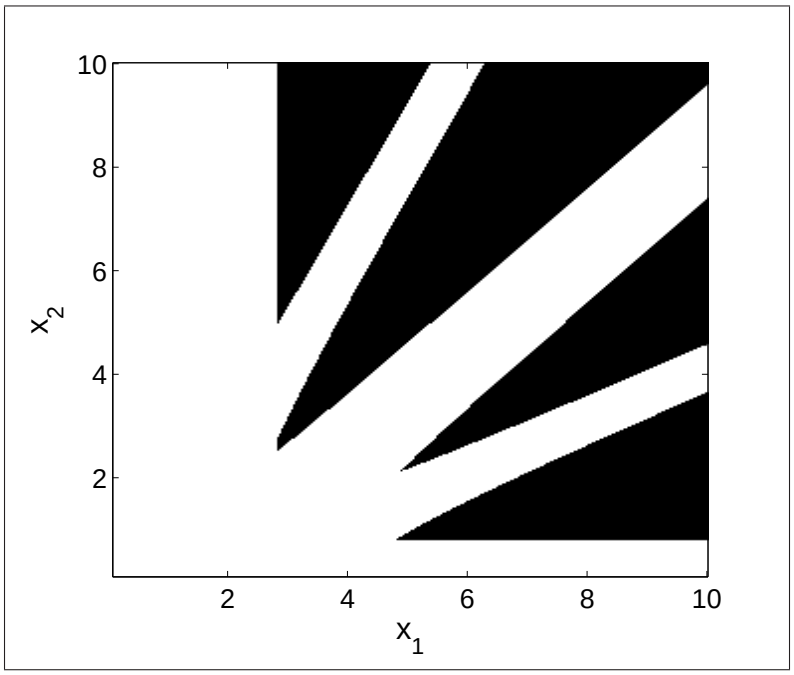

Figure 1.7 Decision boundary obtained with the two-input quantron. We used 3 potentials for each input ($N_1 = N_2 = 3$), and the following parameters : $w_1 = 0.80$, $w_2 = 0.60$, $\theta_1 = 1.00$, $\theta_2 = 2.50$.

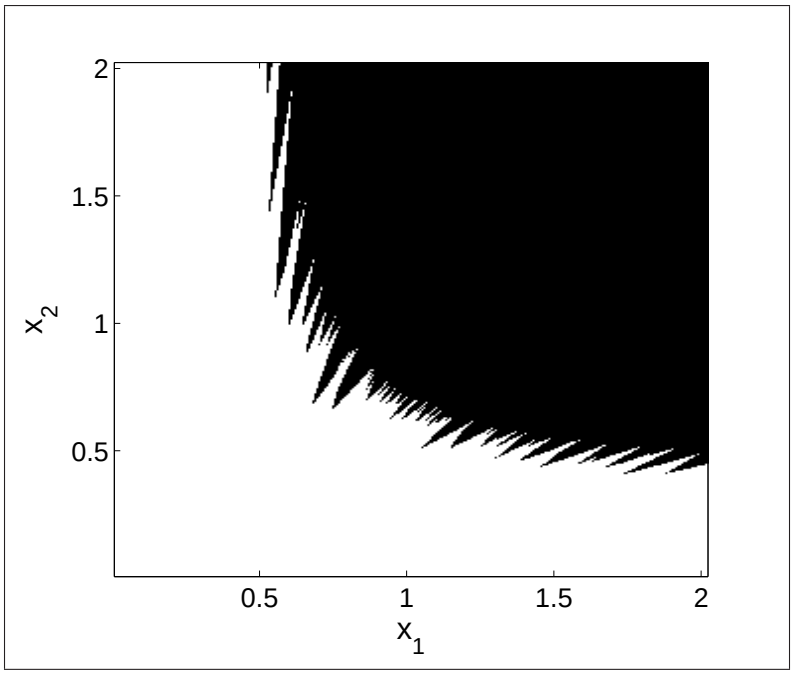

Figure 1.8 Decision boundary obtained with the two-input quantron. We used 10 potentials for each input ($N_1 = N_2 = 10$), and the following parameters : $w_1 = 0.33$, $w_2 = 0.25$, $\theta_1 = 1.00$, $\theta_2 = 4.00$.

With a threshold value of 1, the corresponding firing condition is given by

$$\max_{\substack{j=1,2 \\ i=0,\ldots,N_j-1}} \{A(\theta_j + ix_j)\} \geq 1 \tag{1.10}$$

The spikeprop neuron's decision boundaries are always composed of diagonal stripes, due to the form of equation (1.6), whereas the quantron's decision boundaries have a more complex structure. With less parameters than the spikeprop neuron, the quantron proposes more interesting decision boundaries. [1]

1.4 Approximated quantron

We now propose a new method to approximate $\max[A(t)]$ to apply gradient search while training a quantron with rectangular potentials. This method is distinct from the quantron's multiscale scheme (Connolly & Labib, 2009). While this scheme gives a very precise approximation of $\max[A(t)]$ when using biologically realistic potentials, the time needed for its calculation is a burden when testing various learning algorithms. This problem motivates the need for a computationally efficient approximation.

When using rectangular potentials and positive weights, we can approximate the input signals $R_j(t)$ by a quadratic function. Thus, $A(t)$ will also be a quadratic function, with a maximum that can be readily calculated.

To approximate $R_j(t)$, we use a quadratic functions with roots θ_j and $\theta_j + (N_j - 1)x_j + s_j$ which are the beginning time and the end time of the train of potentials. We can show that

$$\max\{R_j(t)\} = w_j \min\left\{ \left\lfloor \frac{s_j}{x_j} \right\rfloor + 1, N_j \right\}. \tag{1.11}$$

We propose to approximate this maximum with the following expression :

$$\mu_j = w_j \frac{\frac{x_j+2s_j}{2x_j} + N_j - \sqrt{\left(\frac{x_j+2s_j}{2x_j} - N_j\right)^2 + \epsilon}}{2} \tag{1.12}$$

where the $\min(a, b)$ function used in equation (1.11) was approximated by

$$\operatorname{smin}(a, b) = \frac{a + b - \sqrt{(a-b)^2 + \epsilon}}{2}. \tag{1.13}$$

Here, ϵ is a positive hyperparameter that controls the precision of the approximation. Thus,

1. À la lumière des frontières de décision présentées, le quantron semble être un classificateur plus flexible que le neurone impulsionnel de Spikeprop.

the quadratic function approximating $R_j(t)$ has μ_j for its maximum.

Finally, we obtain the following maximum for the sum of signals from M inputs :

$$\begin{aligned}\mu_{1,2,\ldots,M} =& \frac{\left(\sum_{j=1}^{M} \frac{4\mu_j(2\theta_j+(N_j-1)x_j+s_j)}{((N_j-1)x_j+s_j)^2}\right)^2}{\sum_{j=1}^{M}\left(\frac{4\mu_j}{((N_j-1)x_j+s_j)^2}\right)^2} \\ &- \sum_{j=1}^{M} \frac{4\mu_j\theta_j(\theta_j+(N_j-1)x_j+s_j)}{((N_j-1)x_j+s_j)^2}\end{aligned} \tag{1.14}$$

To solve a classification problem with this approximation, we minimize

$$E_{GS} = \sum_{k=1}^{n} \text{smax}\left(\frac{\Gamma - o_k}{d_k\Gamma}\right) \tag{1.15}$$

where o_k is now calculated with $\mu_{1,2,\ldots,M}$. Here, we approximated the max function used in E_{DS} with

$$\text{smax}(x) = \frac{x+\sqrt{x^2+\epsilon}}{2} \tag{1.16}$$

where we also have $\epsilon > 0$.

To minimize E_{GS}, we use an interior-point algorithm (Waltz *et al.*, 2006) implementation available in Matlab (the fmincon function, in 64 bit version 7.10). A constrained optimization method is needed to ensure that w_j, s_j and θ_j always remain positive.

1.5 Experimentation

We now present a comparative experiment between the original quantron and its quadratic approximation.

We proceed with three learning test sequences : a single direct search (DS), gradient search followed by direct search (GS+DS), and finally direct search, followed by gradient search and then by direct search (DS+GS+DS). We use direct search after gradient search to tune the solution to the error function E_{DS}.

Each search phase is allowed a maximum of 5000 iterations. This large number of iterations ensures that the algorithms will converge with very high confidence to a minimum (possibly local).

Our aim is to verify, in an ideal setting, if using gradient search can improve the best results of direct search. Thus, we do not take into consideration the convergence time of these methods.

In the following experiments, we first train a two-input and a three-input quantron to solve

nonlinear classification toy problems involving min and max functions. Then, we compare the quantron and perceptron performance on the IRIS classification problem from the UCI Machine Learning Repository (Blake *et al.*, 1998).

1.5.1 The quantron with two inputs

We use four toy classification problems with the two-input quantron and its approximation.

For each problem, the inputs (x_1, x_2) are in the square $(0,1) \times (0,1)$ and the class values are encoded as $\{-1, 1\}$. The four problems follow :

- $f_1(x_1, x_2) = \text{sign}[\frac{2}{3} - \max(x_1, x_2)]$;
- $f_2(x_1, x_2) = \text{sign}[\frac{1}{3} - \min(x_1, x_2)]$;
- $f_3(x_1, x_2) = \max[f_1(x_1, x_2), f_2(x_1, x_2)]$;
- $f_4(x_2, x_2) = \min[f_1(x_1, x_2), f_2(x_1, x_2)]$.

These problems are highly nonlinear due to the presence of min and max functions, and thus would be difficult to solve for a single perceptron neuron. [2]

Experimental setup

We use a regular sampling of the unit square $[0,1] \times [0,1]$ as training data to compare the original quantron and its approximation in an ideal setting. Here, we used input coordinates $(\frac{c_1}{9}, \frac{c_2}{9})$ for c_1 and c_2 between 1 and 8 (64 samples).

In our experiments, we use the same number (N) of potentials for all inputs. The parameters w_j, s_j, and θ_j are initialized at uniform random values in the following intervals :

- $w_j \in \left[\frac{\Gamma}{MN}, \frac{\Gamma}{M}\right]$;
- $s_j \in [0.01, N + 0.01]$;
- $\theta_j \in [0.01, 5]$.

These are chosen to give access to a large variety of decision boundaries.

The hyperparameter ϵ is set to 1 for computing μ_j and to 0.0025 for computing E_{GS}. The threshold Γ was set to 20. These values where chosen during preliminary tests.

Finally, we analyze the final error and the misclassification error obtained in the four problems for 100 random parameter initializations.

Thus, we measure the final error of DS, GS+DS and DS+GS+DS using equation (1.5), and the corresponding misclassification error (in %) on the 64 samples for DS, GS+DS and DS+GS+DS. We gather these statistics using 5, 10 and 15 potentials per input.

2. Les images correspondant à ces problèmes de classification sont présentées en annexe de la thèse.

The complete results of our experimentation with these four problems are presented in appendix A (tables A.1 to A.4). In the following analysis, we only report box plots of misclassification error and highlight the important aspects of the results.

In the box plots, errors for DS, GS+DS and GS+DS+GS are respectively noted M0(N), M1(N) and M2(N), where N is the number of potentials per input.

In each box, the median is represented by a dotted circle, and points located farther than 1.5 box height above or below their box are considered outliers (noted by the + symbol).

Many of the box plots we present miss parts of the box or some whiskers. This happens when a single value forms at least 25% of the data set. This is due to convergence of the training sequences to a same solution, or to different solutions with same error.

Results : Problem 1

Figure 1.9 shows box plots of the misclassification error.

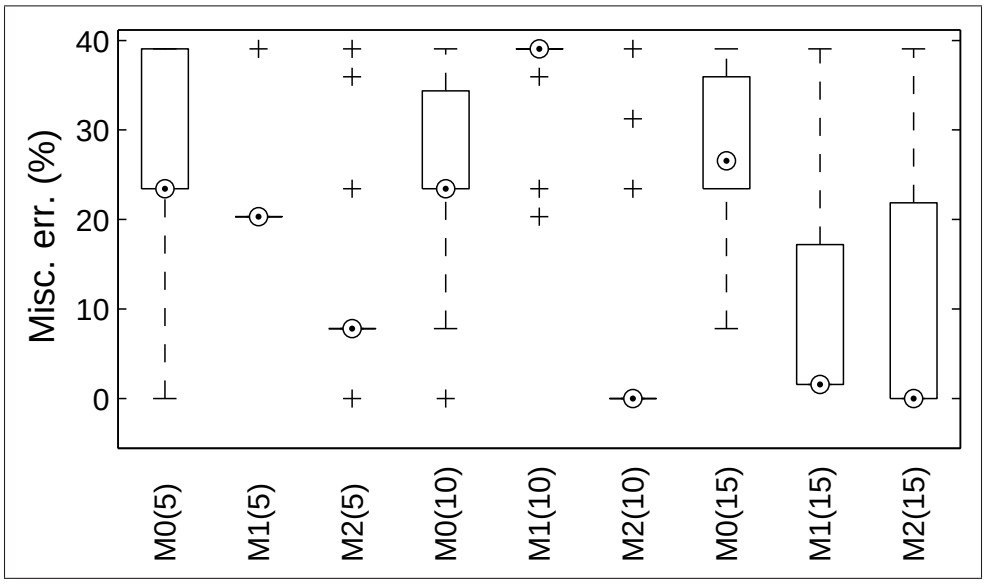

Figure 1.9 Box plots of misclassification error for problem 1.

The results from table A.1 show that final error was effectively reduced by using gradient search. Variation of the final error was also reduced.

Also, our results show that misclassification error is generaly also reduced, although less consistently. However, reduction of variation in final error doesn't necessarily correspond to a reduction in variation of the misclassification error. Indeed, the error function used is not always correlated to the effective misclassification error.

Augmentation of misclassification error variance by gradient search can be interpreted in the box plots. For example, in figure 1.9, for the quantron with 15 potentials per input, we

see that gradient search failed to reduce the maximal error of 100 runs while the minimal error was reduced. Thus, the range of the error data (and its variance) increased.

Results : Problem 2

Figure 1.10 shows box plots of the misclassification error.

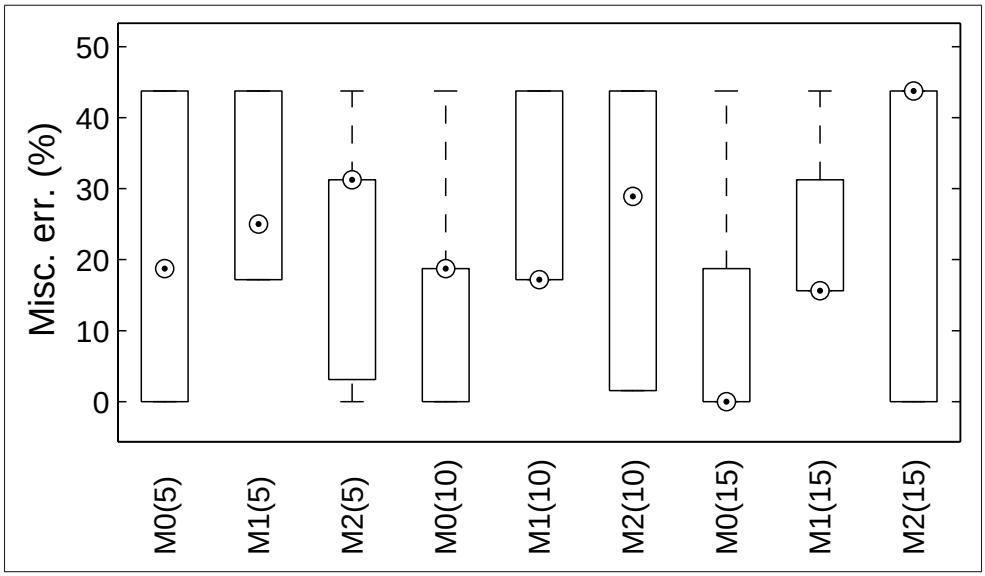

Figure 1.10 Box plots of misclassification error for problem 2.

The results from table A.2 show that gradient search does not consistently help to reduce the final error, in both median and mean. However, variation of the final error is reduced.

Here, our results show that misclassification error tends to increase when using gradient search. This could be caused by the incapacity of our approximation method to represent this particular problem, or by our experimental setup. However, we see that the original quantron can indeed solve this problem, as shown by a minimum misclassification error of zero in some cases. We also see that, as in problem 1, a reduction in final error variation does not necessarily correspond to a reduction in variation of the misclassification error.

Results : Problem 3

Figure 1.11 shows box plots of the misclassification error.

The results from table A.3 show that gradient search helps reduce the final error, except for $N = 10$. Variation of the final error is also reduced.

Also, our results show that misclassification error is consistently reduced, even when the final error is not reduced. As in both problems 1 and 2, the variation of the misclassification error increases when using gradient search.

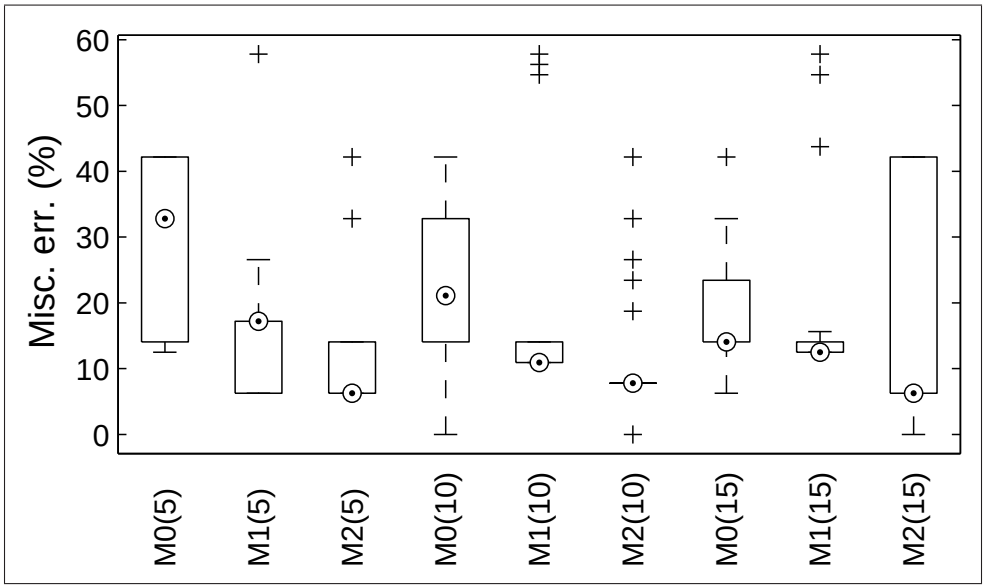

Figure 1.11 Box plots of misclassification error for problem 3.

Results : Problem 4

Figure 1.12 shows box plots of the misclassification error.

The results from table A.4 show that gradient search helps reduce the final error. Variation of the final error is also reduced.

Also, our results show that misclassification error is consistently reduced. As in others problems, the variation of the misclassification error increases when using gradient search.

1.5.2 The quantron with three inputs

We repeat the previous experiment with a three-input quantron and its approximation. Here, we solve only problem 4, generalized to three dimensions as follows :

- $f_4(x_1, x_2, x_3) = \min[f_1, f_2]$;
- $f_1(x_1, x_2, x_3) = \text{sign}[\frac{2}{3} - \max(x_1, x_2, x_3)]$;
- $f_2(x_1, x_2, x_3) = \text{sign}[\frac{1}{3} - \min(x_1, x_2, x_3)]$.

We use the sequences DS, GS+DS and DS+GS+DS with 15 potentials per input.

The only difference in the experimental setup is that we used input coordinates $(\frac{c_1}{6}, \frac{c_2}{6}, \frac{c_2}{6})$ for c_1, c_2 and c_3 between 1 and 5 (125 samples). We also limited the experiment to 50 repetitions of the learning sequences.

Figure 1.13 shows box plots of the misclassification error. As with problem 4 in two dimensions, these results also indicate an improvement in learning when using gradient search.

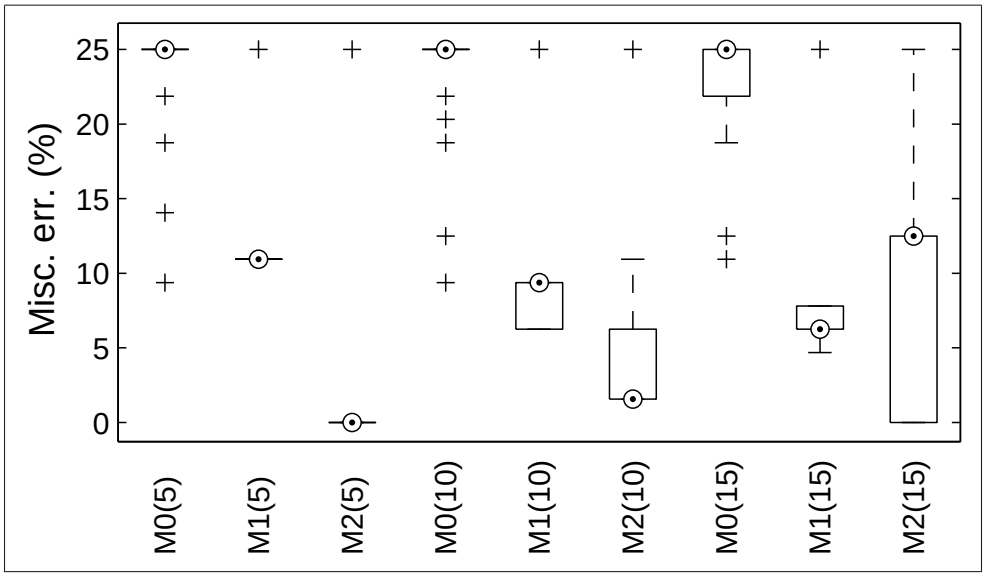

Figure 1.12 Box plots of misclassification error for problem 4.

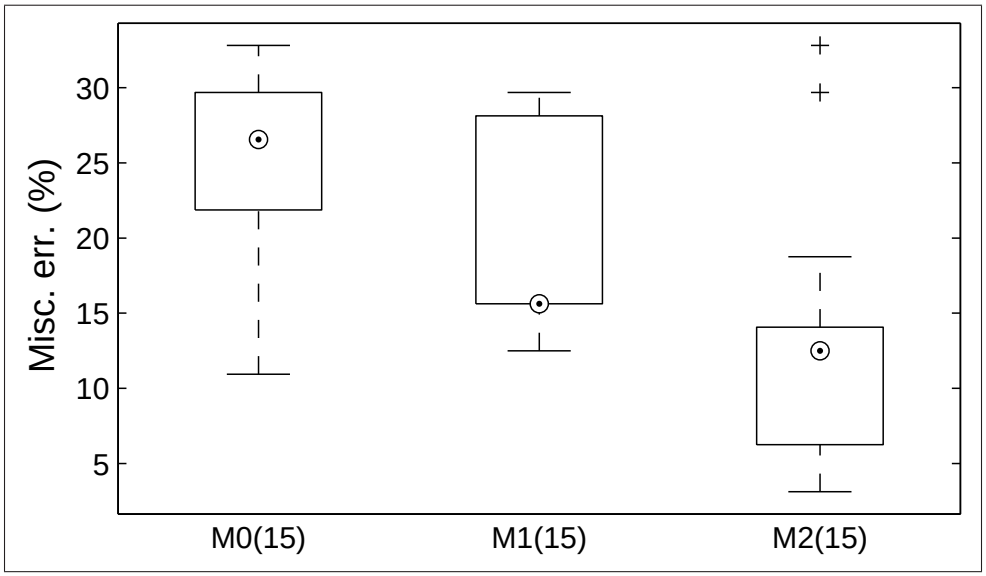

Figure 1.13 Box plots of misclassification error for problem 4 using a three-input quantron with 15 potentials per input.

1.5.3 Solving the IRIS problem

The IRIS classification problem consists of 50 samples from each of three types of iris plant : *Iris Setosa*, *Iris Versicolor* and *Iris Virginica*. The problem possesses four attributes, however the two attributes **petal lenght** and **petal width** are usually considered the most important predictors of iris type (Dash & Liu, 2000). Thus, we solve this problem by using all four attributes as input, and also by using only petal lenght and width.

To classify the samples, we use three quantrons, one for each class. For a given sample, the predicted class will be determined by the quantron with the highest activation maximum. To train this set of three quantrons, we will use modify the error function given by (1.5) as follows :

$$E_{DS} = \sum_{i=1}^{3} \sum_{k \in C_i} \sum_{j=1}^{3} \max(o_{j,k} - o_{i,k}, 0) \tag{1.17}$$

where C_1, C_2 and C_3 represent the three classes, and where $o_{i,k}$ (respectively $o_{j,k}$) is the output of the quantron associated with class C_i (respectively C_j) when processing the sample k. Terms with $i = j$ in equation (1.17) are always null. When using gradient search, the error function given by equation (1.15) will be modified likewise.

Experimental setup

We randomly select 15 samples from each of the three classes to form a training set. The other samples form a testing set to validate the generalization performance of the quantron. [3]

For this problem, we use only the DS and DS+GS+DS sequences, which we repeat 10 times using random sets of parameters (selected as in our first experiment) for each of the three neurons. We set ϵ as in the previous experimental setup. We use 10 potentials per input.

As a comparison, we will repeat the experiment with quantrons replaced by linear perceptrons (using the same error function and training sequences). The output of a M-input linear perceptron is given by $o(x_1, \ldots, x_M) = w_0 + w_1 x_1 + \ldots + w_M x_M$. In the experiment, the weights $w_0, w_1, \ldots, w_M$ will be randomly selected in the interval $[-2, 2]$.

Results

Tables 1.1 and 1.2 contain the results of the 10 runs for quantrons and perceptrons with 4 inputs. Tables 1.3 and 1.4 contain the results of the 10 runs for quantrons and perceptrons with 2 inputs.

3. Puisque ce travail est de nature exploratoire, nous n'avons pas jugé utile de procéder à une validation croisée.

Table 1.1: IRIS Problem : Results for training and testing misclassification error (in %) using three quantrons (with 4 inputs).

Run	Train DS	Train DS+GS+DS	Test DS	Test DS+GS+DS
1	0	0	11.43	11.43
2	0	0	15.24	16.19
3	0	0	13.33	12.38
4	0	2.22	7.62	16.19
5	0	0	10.48	9.52
6	0	0	19.05	14.29
7	8.89	0	27.62	13.33
8	0	0	8.57	9.52
9	2.22	0	13.33	14.29
10	0	0	6.67	9.52
Min	0	0	6.67	9.52
Med	0	0	12.38	12.86
Mean	1.11	0.22	13.33	12.67
Std	2.82	0.70	6.25	2.62

Table 1.2: IRIS Problem : Results for training and testing misclassification error (in %) using three perceptrons (with 4 inputs).

Run	Train DS	Train DS+GS+DS	Test DS	Test DS+GS+DS
1	6.67	0	8.57	2.86
2	11.11	0	15.24	3.81
3	0	0	10.48	13.33
4	4.44	0	10.48	2.86
5	2.22	0	7.62	6.67
6	4.44	0	11.43	6.67
7	4.44	0	4.76	4.76
8	8.89	0	10.48	4.76
9	8.89	0	13.33	3.81
10	11.11	0	20.00	3.81
Min	0	0	4.76	2.86
Med	5.56	0	10.48	4.29
Mean	6.22	0	11.24	5.33
Std	3.75	0	4.23	3.12

Table 1.3: IRIS Problem : Results for training and testing misclassification error (in %) using three quantrons (with 2 inputs).

Run	Train DS	Train DS+GS+DS	Test DS	Test DS+GS+DS
1	0	4.44	7.62	19.05
2	8.89	0	19.05	7.62
3	2.22	0	6.67	3.81
4	2.22	0	4.76	3.81
5	0	0	13.33	2.86
6	0	6.67	8.57	11.43
7	11.11	0	11.43	7.62
8	0	2.22	13.33	7.62
9	0	0	4.76	2.86
10	6.67	2.22	8.57	2.86
Min	0	0	4.76	2.86
Med	1.11	0	8.57	5.71
Mean	3.11	1.56	9.81	6.95
Std	4.22	2.35	4.49	5.14

Table 1.4: IRIS Problem : Results for training and testing misclassification error (in %) using three perceptrons (with 2 inputs).

Run	Train DS	Train DS+GS+DS	Test DS	Test DS+GS+DS
1	8.89	0	20.95	4.76
2	4.44	0	5.71	3.81
3	4.44	0	2.86	6.67
4	11.11	0	15.23	3.81
5	13.33	0	23.81	2.86
6	4.44	0	13.33	4.76
7	4.44	0	5.71	4.76
8	17.78	0	32.38	9.52
9	8.89	0	15.23	6.67
10	17.78	0	38.10	4.76
Min	4.44	0	2.86	2.86
Med	8.89	0	15.24	4.76
Mean	9.56	0	17.33	5.24
Std	5.35	0	11.63	1.92

The results show that, when using the four attributes of the IRIS problem, the quantron does not generalize as well as the perceptron. Furthermore, using gradient search in this case augmented the misclassification error in 5 out of 10 runs.

When using only the two most relevant attributes, gradient search actually improves the quantron's generalization. Then, the performance of the quantron is closer to that of the perceptron.

It is interesting to note that in the tests with two inputs, the quantron outperformed the perceptron when only direct search was used. [4]

1.6 Discussion

In our experiments, we demonstrated that using the approximated quantron can ameliorate the direct search solutions to classification problems by enabling gradient search. However, the approximated quantron model is better when many potentials overlap, so other problems might be more difficult to solve using gradient search.

When solving the IRIS classification problem, we showed that using relevant attributes, the quantron model compares favorably to the perceptron, and that gradient search helps reduce the testing misclassification error. When all attributes are used, the approximated quantron performs poorly even though there are good solutions in parameter space (we could set the weights corresponding to the uninformative attributes to 0). Thus, gradient search with uninformative attributes can cause an overfitting of the quantron. This behavior illustrates the importance of finding good heuristic rules to set the initial parameters of the quantron. The good performance of the perceptron in both situations is due to its simple mathematical model.

A theoretical advantage of our approximation is its computational complexity. For a quantron with N potentials per input, computing $\max[A(t)]$ is $O(M^2N^2)$ since we need to evaluate, at each of the MN potential onset times, a sum of MN terms. For the approximated quantron, the complexity is $O(M)$ only which can be seen by analyzing equation (1.14). However, this advantage only holds for large values of M and N. In the context of our experiments, the operations involved in using the approximation have a large hidden cost.

In this work, we used the quantron model to solve binary classification problems with a supervised training of the neural state (firing or silent). To our knowledge, this task has only been adressed in the spiking neuron litterature with the tempotron (Gütig & Sompolinsky, 2006), although the authors used spike trains as input whereas, with the quantron, we use a regular delay between potentials (the inverse of a firing rate). Other work based on the

4. Ceci pourrait être dû à l'initialisation des paramètres du perceptron qui ne donnerait pas un point de départ de qualité pour la recherche directe.

spikeprop neuron tends to avoid using silent neurons since it impedes backpropagation (Bohte *et al.*, 2002; Booij & tat Nguyen, 2005; Ghosh-Dastidar & Adeli, 2007, 2009a).

Finally, to solve more complicated classification tasks, we will need to generalize our approximation to networks of quantrons. For a quadratic model of the activation function, it is natural to consider the first solution to $A(t) = \Gamma$ as the activation time used as output value in a network (Labib, 1999). Thus, to adapt backpropagation to a network of quantrons, both the activation maximum and the activation time must be considered. This output could also be used to solve regression problems for which a continuum of values is needed.

1.7 Conclusion

In this paper, we reviewed the quantron model and compared it to the spikeprop neuron, showing that the former yields more interesting decision boundaries. We proposed a quadratic approximation to the input signals of the quantron, and we compared the resulting approximated quantron model to the original quantron and to the perceptron in our experiments. Our results indicate that gradient search with the approximated quantron can often improve solutions found by direct search with the original quantron. We used the IRIS problem to illustrate that, when the most relevant problem attributes are used, the quantron achieves a performance comparable to that of the perceptron. Our work shows the potential of using the neural states (firing or silent) to solve binary classification problems. Future work consists of developing heuristics to set the initial parameters of the quantron, extending our method to a network of quantrons, and applying a multiscale scheme (Connolly & Labib, 2009) to improve the approximation of $\max[A(t)]$.

En résumé : Dans ce premier article, nous avons proposé une approximation du maximum de la fonction d'activation qui a permis d'entraîner le quantron par descente du gradient. Nous avons comparé l'apprentissage du quantron par recherche directe seulement à l'apprentissage par succession de recherche par descente du gradient et de recherche directe. Les résultats obtenus montrent que, pour certains problèmes, une amélioration notable des solutions est apportée par l'ajout de la recherche par descente du gradient. Pour améliorer l'algorithme d'apprentissage, il faudrait utiliser des règles heuristiques d'initialisation des paramètres. Or, l'expression utilisée comme approximation du maximum ne se prête pas facilement à une analyse qui permettrait de trouver de telles règles. Il s'avère que ce problème peut être résolu pour une configuration spécifique du quantron, qui fait l'objet du prochain article.

CHAPITRE 2

ARTICLE 2 : LEARNING ALGORITHMS FOR A SPECIFIC CONFIGURATION OF THE QUANTRON

Présentation : Ce chapitre reprend un article publié dans *Proceedings of the International Joint Conference on Neural Networks* par S. de Montigny et R. Labib (2011).

Nous poursuivons notre approche analytique de l'apprentissage du quantron en obtenant deux expressions analytiques pour la fonction d'activation elle-même. À l'aide de ces simplifications, nous étudions les frontières de décision du quantron, et nous proposons deux algorithmes d'apprentissage tirant parti de leurs caractéristiques géométriques.

Abstract : The quantron is a new artificial neuron model, able to solve nonlinear classification problems, for which an efficient learning algorithm has yet to be developed. Using surrogate potentials, constraints on some parameters and an infinite number of potentials, we obtain analytical expressions involving ceiling functions for the activation function of the quantron. We then show how to retrieve the parameters of a neuron from the images it produced.

2.1 Introduction

The quantron is a new and biologically realistic neuron model that can solve nonlinear classification problems with few parameters (Labib, 1999). Therefore, using quantrons instead of perceptrons could improve a neural network's classification power. [1] However, the diversity of the classification boundaries of the quantron proves to be a challenge for the development of learning algorithms. Indeed, two boundaries whose structure is determined by the parameters can each provide a locally optimal solution. Thus, gradient descent procedures tend to get trapped in local minima.

Similar to the quantron, spiking neurons have been investigated as an improved model of the perceptron (Maass, 1997). The use of numerical procedures to simulate the temporal dynamics of these neurons limits the performance of backpropagation-like algorithms. Although supervised learning algorithms have been successfully applied to networks of spiking neurons (Bohte *et al.*, 2002; Pavlidis *et al.*, 2005), few analytical results on their behavior are

1. Le quantron est capable de résoudre des problèmes de classification non linéaires qui ne peuvent être résolus avec un seul perceptron. En particulier, un seul quantron peut résoudre le problème du OU-exclusif.

available. [2]

In this work, we develop learning algorithms for a specific configuration of the quantron by analyzing its decision boundary. These algorithms are able to separate the active and inactive pixels in an image which represents the state of a quantron evaluated with different input values. They achieve this by learning the hidden parameters that generated the image.

Other work related to the analysis of decision boundaries in the context of neural networks includes the study of the Multi-Layer Perceptron (MLP) used for bilinear separation (Labib & Khattar, 2010), the merging of the spiking neuron and the perceptron where the inputs represent rates of renewal processes (Rowcliffe *et al.*, 2006), and the boundary analysis of spiking neurons using information theory (Fitzgerald & Sharpee, 2009). However, in all these cases, the boundaries obtained are simple lines, circles or ellipses (or smooth combinations of these basic elements) whereas the algorithms proposed in this work are able to directly train a neuron possessing a non-smooth piecewise linear boundary. [3]

The rest of the paper is divided as follows. In section 2.2, we give a short description of the quantron's mathematical model. In section 2.3, we use surrogate potentials and constraints on some parameters to find analytical expressions for the decision boundary of a two-input quantron. In section 2.4, we investigate relations between the parameters of the quantron and the salient geometrical characteristics of its boundaries. In section 2.5, we use these relations to develop learning algorithms for the quantron. In section 2.6, we experiment with the proposed algorithms, showing they are able to retrieve the parameter values used to generate a particular output image. Finally, we discuss the results and provide the conclusion in sections 2.7 and 2.8.

2.2 Mathematical Model

The input value of a quantron determines the delay between consecutive potentials of similar shape $p(t)$. When the delay is shorter than the duration of the potentials, a spatial summation occurs. For a quantron with many inputs, the sequences of potentials stemming from each input overlap in time, which corresponds to temporal summation.

For a two-input quantron, the variables x and y represent the delays and the variables N_1 and N_2 represent the maximum number of potentials of each input. Two trains of potentials,

2. Par exemple, la convergence de l'algorithme de rétropropagation vers un minimum local est incertaine si la réponse du neurone est approximée numériquement.

3. Il est avantageux de pouvoir entraîner directement un neurone qui génère une frontière linéaire par morceaux. En effet, le comportement de ce neurone est plus facile à analyser et à interpréter que le comportement d'une approximation lisse du même neurone.

shifted by θ_1 and θ_2 time units respectively, are added to form the activation function

$$A(t) = \sum_{i=0}^{N_1-1} p_1(t - \theta_1 - ix) + \sum_{j=0}^{N_2-1} p_2(t - \theta_2 - jy). \tag{2.1}$$

The potentials $p_1(t)$ and $p_2(t)$ have height parameters w_1, w_2 and width parameters r_1, r_2.

The quantron is active when there exists a time t_0 for which $A(t_0) = \Gamma$, where Γ is a positive threshold value. Otherwise, it is inactive. We can also say that the quantron is active only when $\max\{A(t)\} \geq \Gamma$. Thus, we can define the classification function

$$g(x, y) = u(\max\{A(t; x, y)\} - \Gamma) \tag{2.2}$$

where $u(\cdot)$ is the Heaviside function. The decision boundary of the quantron can be defined as the solution in x and y to the equation $\max\{A(t; x, y)\} = \Gamma$.

2.3 Surrogate Potential Functions

Using realistic functions $p(t)$ to represent biological potentials makes it difficult to train the quantron since there is no simple analytical expression for $\max\{A(t)\}$ in that case. Thus, using surrogate potential functions is crucial to the development of a learning algorithm (Connolly & Labib, 2009). Throughout this work, we will use rectangular and ramp potentials.

2.3.1 Rectangular potentials

Considering a rectangular function of height w and width r to represent $p(t)$, equation (2.1) becomes

$$A(t) = \sum_{i=1}^{2} \sum_{j=0}^{N_i-1} w_i[u(t - \theta_i - jx_i) - u(t - \theta_i - r_i - jx_i)] \tag{2.3}$$

where $(x, y) = (x_1, x_2)$. Figure 2.1 illustrates particular shapes of an activation function with a single input x.

Since $A(t)$ is a piecewise constant function, we obtain an analytical expression for $\max\{A(t)\}$ by evaluating $A(t)$ on a discrete set of points. We can further simplify the analytical expression of $\max\{A(t)\}$ by imposing particular constraints to the parameters. We use

$$\begin{gathered} w_1 \geq 0,\ w_2 \geq 0, \\ \theta_1 = \max(r_1, r_2) - r_1,\ \theta_2 = \max(r_1, r_2) - r_2, \end{gathered} \tag{2.4}$$

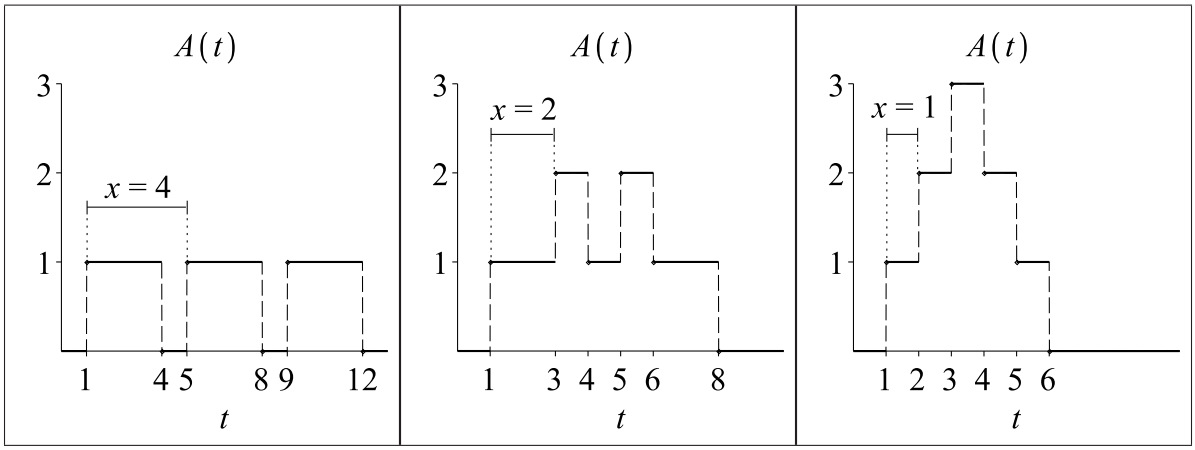

Figure 2.1 Activation function of the quantron for $x = 4, 2, 1$ with three rectangular potentials. Variation of input x can modify the observed overlap.

to obtain

$$\max\{A(t)\} = \lim_{t\uparrow\max(r_1,r_2)} A(t) = \sum_{i=1}^{2}\sum_{j=0}^{N_i-1} w_i u\left(r_i - jx_i\right) \tag{2.5}$$

where $u(0) = 0$. Also, with an infinite number of potentials, we obtain an analytical expression for $\max\{A(t)\}$ involving ceiling functions :

$$\max\{A(t)\} = w_1\left\lceil\frac{r_1}{x}\right\rceil + w_2\left\lceil\frac{r_2}{y}\right\rceil. \tag{2.6}$$

Finally, with the relation $\lceil x\rceil \geq y \iff x > \lceil y\rceil - 1$, we can isolate y in

$$w_1\left\lceil\frac{r_1}{x}\right\rceil + w_2\left\lceil\frac{r_2}{y}\right\rceil \geq \Gamma, \tag{2.7}$$

to find the following functional form for the decision boundary :

$$f(x) = \begin{cases} \dfrac{r_2}{\left\lceil\frac{\Gamma - w_1\left\lceil\frac{r_1}{x}\right\rceil}{w_2}\right\rceil - 1} & \text{if } x \geq \dfrac{r_1}{\left\lceil\frac{\Gamma - w_2}{w_1}\right\rceil - 1}, \\ \infty & \text{if } x < \dfrac{r_1}{\left\lceil\frac{\Gamma - w_2}{w_1}\right\rceil - 1}. \end{cases} \tag{2.8}$$

For the input pair (x, y), the quantron is active if $y \leq f(x)$.

2.3.2 Ramp potentials

Considering a ramp function of height w and width r defined by

$$p(t) = \begin{cases} \frac{wt}{r} & \text{if } 0 \leq t \leq r, \\ 0 & \text{elsewhere,} \end{cases} \tag{2.9}$$

to represent potentials, equation (2.1) becomes

$$\begin{aligned} A(t) = &\sum_{i=0}^{2} \sum_{j=0}^{N_i-1} \frac{w_i(t-\theta_i-jx_i)}{r_i} u(t-\theta_i-jx_i) \\ &- \sum_{i=0}^{2} \sum_{j=0}^{N_i-1} \frac{w_i(t-\theta_i-jx_i)}{r_i} u(t-\theta_i-r_i-jx_i). \end{aligned} \tag{2.10}$$

Figure 2.2 illustrates particular shapes of an activation function with a single input x.

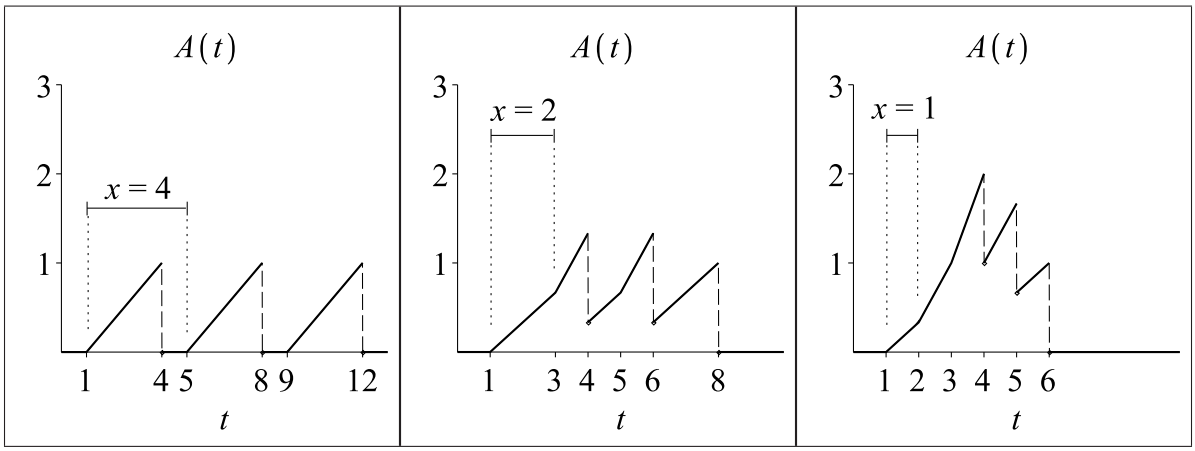

Figure 2.2 Activation function of the quantron for $x = 4, 2, 1$ with three ramp potentials.

Applying the constraints given in equation (2.4), we have

$$\begin{aligned} \max\{A(t)\} &= \lim_{t \uparrow \max(r_1, r_2)} A(t) \\ &= \sum_{i=1}^{2} \sum_{j=0}^{N_i-1} w_i \max\left(1 - \frac{jx_i}{r_i}, 0\right) \end{aligned} \tag{2.11}$$

and with an infinite number of potentials, we obtain the analytical expression

$$\max\{A(t)\} = \sum_{i=1}^{2} w_i \left\lceil \frac{r_i}{x_i} \right\rceil \left(1 - \frac{x_i}{2r_i}\left(\left\lceil \frac{r_i}{x_i} \right\rceil - 1\right)\right) \tag{2.12}$$

using the following lemma.

Lemma 2.1: We have

$$\mu(z) = \sum_{i=0}^{\infty} \max\left(1 - \frac{i}{z}, 0\right) = \lceil z \rceil \left(1 - \frac{\lceil z \rceil - 1}{2z}\right) \tag{2.13}$$

for $z > 0$.

Proof: We have $\mu(z) = 1$ if $0 < z \leq 1$, $\mu(z) = 2 - 1/z$ if $1 < z \leq 2$, $\mu(z) = 3 - 3/z$ if $2 < z \leq 3$, $\mu(z) = 4 - 6/z$ if $3 < z \leq 4$, etc. In general, we find that

$$\mu(z) = n - \frac{n(n-1)}{2z} \text{ if } n - 1 < z \leq n \tag{2.14}$$

for n integer. Setting $n = \lceil z \rceil$ in the previous equation, we obtain the announced result. □

The function $\mu(z)$ is not invertible on $(0, 1)$. However, we can use the relation

$$\mu(x) \geq y \iff x > \nu(y) \text{ for } x > 0 \text{ and } y > 0 \tag{2.15}$$

with

$$\nu(z) = \frac{(\lceil 2z \rceil - 1)(\lceil 2z \rceil - 2)}{2\lceil 2z \rceil - 2z - 2}, \tag{2.16}$$

which is the inverse of $\mu(z)$ for $z > 1$. Although $\nu(1)$ is undefined, we have $\lim_{z \downarrow 1} \nu(z) = 1$ and $\nu(z) = 0$ for $z < 1$, such that equation (2.15) is valid if we set $\nu(1) = 0$. Thus, we can isolate y in

$$w_1 \mu\left(\frac{r_1}{x}\right) + w_2 \mu\left(\frac{r_2}{y}\right) \geq \Gamma \tag{2.17}$$

to obtain the following functional form for the decision boundary :

$$f(x) = \begin{cases} \dfrac{r_2}{\nu\left(\frac{\Gamma - w_1 \mu\left(\frac{r_1}{x}\right)}{w_2}\right)} & \text{if } x > \dfrac{r_1}{\nu\left(\frac{\Gamma - w_2}{w_1}\right)}, \\ \infty & \text{if } x \leq \dfrac{r_1}{\nu\left(\frac{\Gamma - w_2}{w_1}\right)}. \end{cases} \tag{2.18}$$

For the input pair (x, y), the quantron is active if $y \leq f(x)$.

2.4 Decision boundary and image analysis

2.4.1 Rectangular potentials

Figure 2.3 shows an example of a decision boundary of the quantron with rectangular potentials. Over the extreme corner located at (a, b), the boundary goes straight up as y

grows, and at the right of the extreme corner located at (c, d), the boundary stays at the same level as x grows.

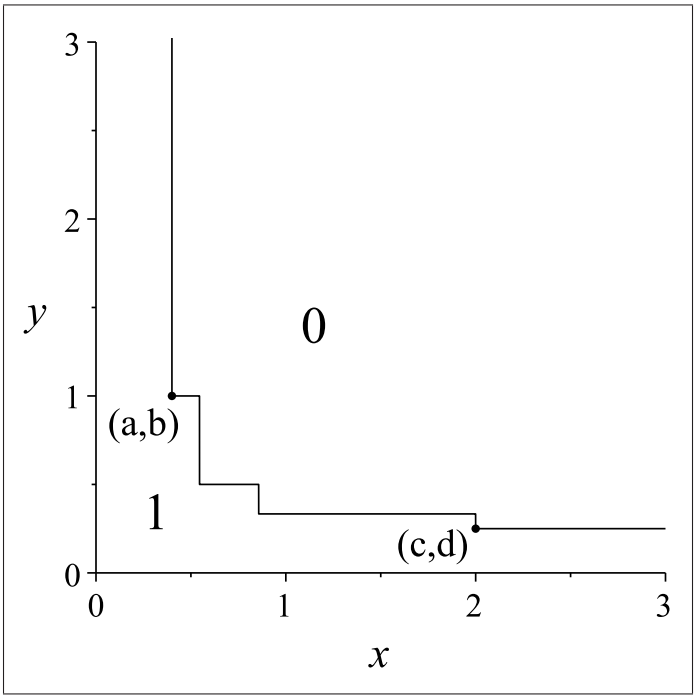

Figure 2.3 Regions 1 and 0 contain, respectively, the inputs for which the quantron is active or inactive. We plotted the boundary using equation (2.8) with $(w_1, w_2, r_1, r_2, \Gamma) = (0.05, 0.2, 6, 1, 1)$. Dots illustrate extreme corners.

Analysis of equation (2.8) reveals links between the extreme corner coordinates and the parameters of the quantron :

$$\begin{aligned}(a,b) &= \left(\frac{r_1}{\left\lceil \frac{\Gamma - w_2}{w_1} \right\rceil - 1} \ , \ \frac{r_2}{\left\lceil \frac{\Gamma - w_1 \left(\left\lceil \frac{\Gamma - w_2}{w_1} \right\rceil - 1 \right)}{w_2} \right\rceil - 1} \right), \\ (c,d) &= \left(\frac{r_1}{\left\lceil \frac{\Gamma - w_2 \left(\left\lceil \frac{\Gamma - w_1}{w_2} \right\rceil - 1 \right)}{w_1} \right\rceil - 1} \ , \ \frac{r_2}{\left\lceil \frac{\Gamma - w_1}{w_2} \right\rceil - 1} \right).\end{aligned} \tag{2.19}$$

Furthermore, the boundary is composed of

$$n = \min \left(\left\lceil \frac{\Gamma - w_1}{w_2} \right\rceil - 1, \left\lceil \frac{\Gamma - w_2}{w_1} \right\rceil - 1 \right) \tag{2.20}$$

visible steps, and possesses $2n - 1$ corners.

If $w_1 \leq w_2$, we find that $b = r_2$ and $d = b/n$. If $w_1 \geq w_2$, we have $c = r_1$ and $a = c/n$. Thus, the value of either r_1 or r_2 is present in the corner coordinates.

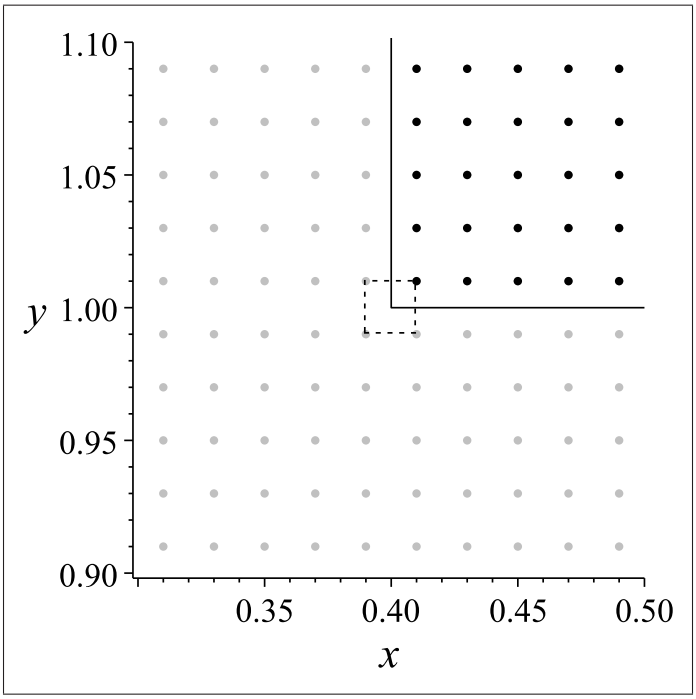

Figure 2.4 With a regular grid of pixels, corners are found inside the dashed square delimited by four adjacent pixels. We use grey pixels to represent inputs where the quantron is active, and black pixels where it is inactive.

By "an image produced by the quantron", we mean a regular grid of pixels (x, y, I), where x and y are inputs and where $I \in \{0, 1\}$ is determined by equation (2.2). We find ranges of possible values for a, b, c and d by searching for the extreme corners in the image. The range for a and d is determined by analyzing the topmost and rightmost columns of pixels in the image, and the range for b and c is determined by analyzing the pixel column to the right of a and also the one above d.

Figure 2.4 shows that we can choose the actual position of a corner anywhere in a square delimited by four pixels, as long as we respect the appropriate constraint (either $d = b/n$ or $a = c/n$).

2.4.2 Ramp potentials

Figure 2.5 shows an example of a decision boundary of the quantron with ramp potentials. The extreme corners shown here have the same interpretation as before.

As in section 2.4.1, we can express the corner coordinates with the parameters of the

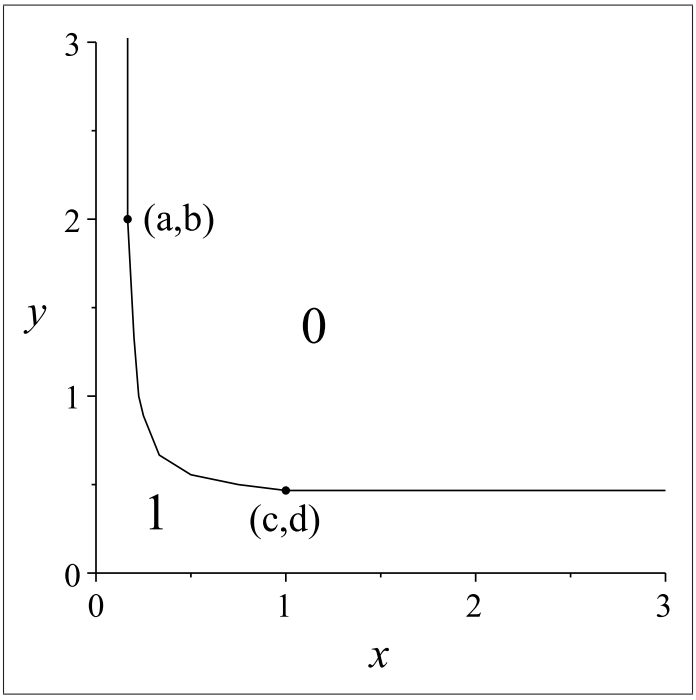

Figure 2.5 Regions 1 and 0 correspond, respectively, to the quantron being active or inactive. We plotted the boundary using equation (2.18) with $(w_1, w_2, r_1, r_2, \Gamma) = (0.2, 0.3, 1, 2, 1)$. Dots illustrate extreme corners.

quantron :

$$\begin{aligned}(a,b) &= \left(\frac{r_1}{\nu\left(\frac{\Gamma-w_2}{w_1}\right)}, r_2\right), \\ (c,d) &= \left(r_1, \frac{r_2}{\nu\left(\frac{\Gamma-w_1}{w_2}\right)}\right).\end{aligned} \tag{2.21}$$

In the following theorem, we prove that $b = r_2$ and $c = r_1$.

Theorem 2.1 : We have

$$b = \lim_{x\downarrow a} f(x) = r_2 \text{ and } c = \lim_{y\downarrow d} h(y) = r_1 \tag{2.22}$$

where $f(x)$ and $h(y)$ are the functional forms of the boundary with respect to x and y.

Proof : For $b = r_2$, we must prove that

$$\lim_{x\downarrow a} \nu\left(\frac{\Gamma - w_1\mu\left(\frac{r_1}{x}\right)}{w_2}\right) = 1. \tag{2.23}$$

To reach that limit, we must have

$$L = \lim_{x \downarrow a} \frac{\Gamma - w_1 \mu\left(\frac{r_1}{x}\right)}{w_2} = 1 \tag{2.24}$$

since we approach the discontinuity of $\nu(z)$ at $z = 1$ from the right. By continuity of $\mu(z)$, we can replace x by a, yielding

$$L = \frac{\Gamma - w_1 \mu\left(\nu\left(\frac{\Gamma - w_2}{w_1}\right)\right)}{w_2}. \tag{2.25}$$

Since the coordinate a exists, we have $\frac{\Gamma - w_2}{w_1} > 1$ (or else there would be no decision boundary as the quantron would always be active). Since $\mu(z)$ and $\nu(z)$ are inverse functions for $z > 1$, we have

$$L = \frac{\Gamma - w_1 \left(\frac{\Gamma - w_2}{w_1}\right)}{w_2} = 1 \tag{2.26}$$

as required. Similarly, we can show that $c = r_1$. □

Using this theorem, we have

$$\frac{c}{a} = \nu\left(\frac{\Gamma - w_2}{w_1}\right) \text{ and } \frac{b}{d} = \nu\left(\frac{\Gamma - w_1}{w_2}\right) \tag{2.27}$$

which can be transformed to

$$\mu\left(\frac{c}{a}\right) = \frac{\Gamma - w_2}{w_1} \text{ and } \mu\left(\frac{b}{d}\right) = \frac{\Gamma - w_1}{w_2}. \tag{2.28}$$

The solution to this system of equations is given by

$$w_1 = \frac{\Gamma\left(\mu\left(\frac{b}{d}\right) - 1\right)}{\mu\left(\frac{c}{a}\right)\mu\left(\frac{b}{d}\right) - 1} \text{ and } w_2 = \frac{\Gamma\left(\mu\left(\frac{c}{a}\right) - 1\right)}{\mu\left(\frac{c}{a}\right)\mu\left(\frac{b}{d}\right) - 1}. \tag{2.29}$$

These analytical expressions determine the quantron's parameters directly from the coordinates a, b, c and d. To locate these coordinates, we proceed as shown in figure 2.6 (where the left segment of the boundary can have either a flat or steep slope). In both cases, the small polygon R must contain the position of a corner. To find R, we select two pairs P and Q of adjacent pixels near the corner, one active and one inactive. The line J passes through the active pixel of P and the inactive pixel of Q, while the line K passes through the active pixel of Q and the inactive pixel of P. The lines J and K respectively meet lines J' and K' to form upper and lower bounds for the decision boundary near the corner. The equations for these lines can be found by a custom image analysis algorithm, and they are used to delimitate

the search for the extreme corner's location, since the region R must at least contain some coordinates that enable correct classification of all pixels.

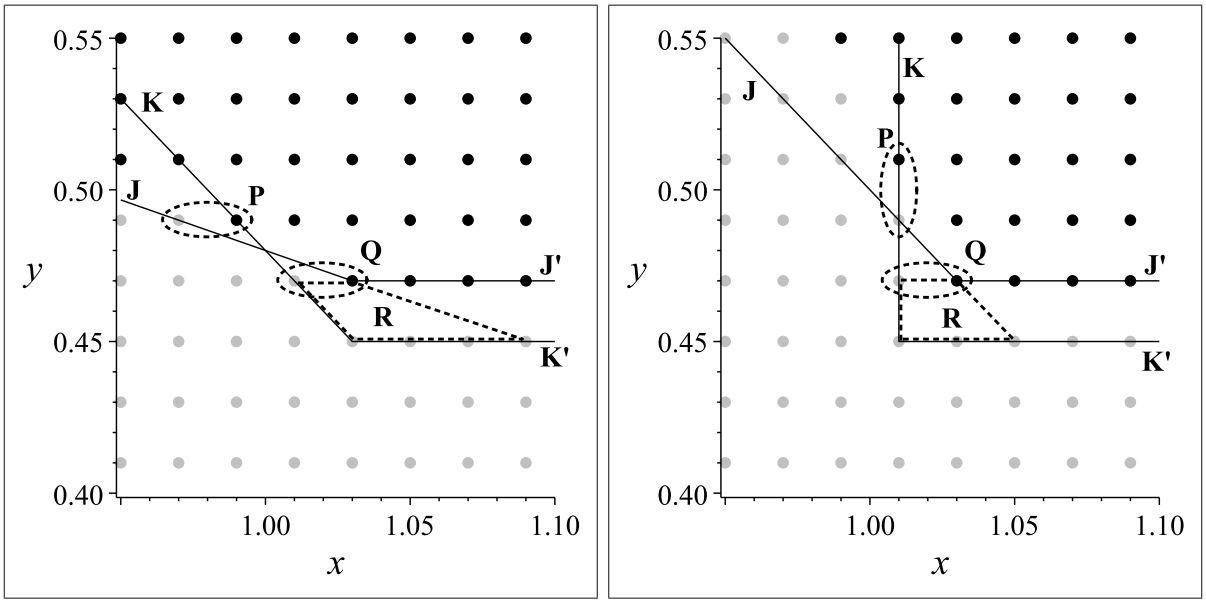

Figure 2.6 The boundary's corner must be inside the dashed polygon R. The dashed ellipses contain the pixel pairs used to find lines J, K, J' and K'. In the left graph, the boundary has a slope flatter than unity near the extreme corner, whereas in the right graph, it has a slope steeper than unity.

2.5 Learning algorithms

2.5.1 Rectangular potentials

The learning algorithm for the quantron with rectangular potentials is an hybrid discrete-continuous optimization procedure. The first step of the algorithm is to determine approximatively the coordinates a, b, c and d from the image. Then, we consider the two cases $w_1 \leq w_2$ and $w_1 \geq w_2$ separately. For $w_1 \leq w_2$, we have $\frac{b}{d} = n$ and $\left\lceil \frac{\Gamma - w_2}{w_1} \right\rceil - 1 = m$ with m an integer greater or equal than n. Thus, the parameter values that generate an image of extreme corners (a, b) and (c, d) are found in the polygonal region delimited by the constraints

$$\begin{aligned} w_1 + n w_2 < \Gamma,\ w_1 + (n+1) w_2 \geq \Gamma, \\ m w_1 + w_2 < \Gamma,\ (m+1) w_1 + w_2 \geq \Gamma. \end{aligned} \tag{2.30}$$

Since $r_2 = b$ and $r_1 = am$, the activation function's maximum is given by

$$\max\{A(t)\} = w_1 \left\lceil \frac{am}{x} \right\rceil + w_2 \left\lceil \frac{b}{y} \right\rceil \tag{2.31}$$

which is linear in w_1 and w_2. Thus, we define the average sum-of-squares error function between the D pixel binary training image $\{(x_k, y_k, I_k); k = 1, ..., D\}$ and the output of the quantron by

$$E = \frac{1}{2D} \sum_{k=1}^{D} \left(I_k - \sigma \left(w_1 \left\lceil \frac{am}{x_k} \right\rceil + w_2 \left\lceil \frac{b}{y_k} \right\rceil - \Gamma \right) \right)^2 \tag{2.32}$$

with $\sigma(z) = \frac{1}{1+e^{-\alpha z}}$ is a sigmoid function with a slope hyperparameter α.

If $w_1 \geq w_2$, we can switch m and n in the constraints of equation (2.30) and use $r_1 = c$ and $r_2 = dm$.

Minimizing the error function using gradient descent requires starting from a random point satisfying the constraints of equation (2.30) with $m = n$, then with $m = n + 1$ and so on until we find a solution where the gradient's norm is almost null. When learning parameters from images produced by the quantron, the algorithm should always converge to a solution with very low error since it uses a simple transformation of a function which is linear in w_1 and w_2.

2.5.2 Ramp potentials

We have obtained analytical expressions that determine the parameters of the quantron from the coordinates a, b, c and d when using ramp potentials, and we have shown how to identify a region R where to search for a corner. Thus, in this case, the learning algorithm is a procedure that first identifies the region R_1 containing (a, b) and the region R_2 containing (c, d). Then, a random sample of candidate coordinates can be generated and tested against the image by evaluating the pixel misclassification rate. For an image with a large number of pixels, the regions R_1 and R_2 should be small enough such that an acceptable solution will easily be found.

2.6 Experimentation

In both our experiments, we used an image of 500^2 pixels where the inputs corresponding to pixel (i, j) are given by $(x = 0.01i, y = 0.01j)$.

2.6.1 Rectangular potentials

Using test parameter values $w_1 = 0.11$, $w_2 = 0.19$, $r_1 = 4.1$ and $r_2 = 2.3$, we find that $n = 4$ by inspection of the image. Fixing the corner coordinates at random in the square of adjacent pixels, we minimize the sum-of-squares error function using batch gradient descent with a step of 0.005 during 100 iterations for $m = 4, 5, 6, 7, 8, 9$ under the hypothesis that $w_1 \leq w_2$, and again when $w_1 \geq w_2$. The hyperparameter α was set to 40. To speed up the evaluation of the error function, we ignored the upper right square of the image (pixels with i and j greater than 150, which are all inactive).

Figure 2.7 shows the 12 learning graphs obtained while running the algorithm. The graph for $m = 7$ (for $w_1 \leq w_2$) stands out as much lower than all the rest, and further analysis of this case shows convergence to a solution with zero classification error.

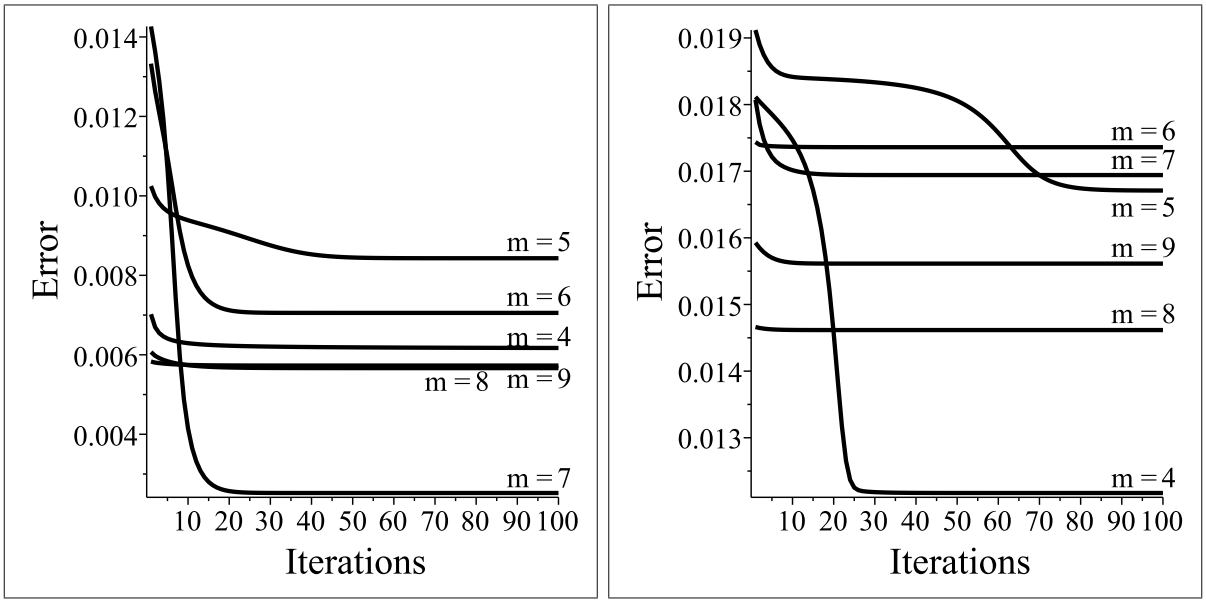

Figure 2.7 Rectangular potentials : learning graphs for $w_1 \leq w_2$ (left) and $w_1 \geq w_2$ (right). The graph for $m = 7$ from the left plot indicates the best solution.

2.6.2 Ramp potentials

Figure 2.8 shows an histogram of the misclassification rate calculated from 50 pairs of random points generated in the polygonal regions near the corners with test parameter values $w_1 = 0.25$, $w_2 = 0.40$, $r_1 = 2.606$ and $r_2 = 2.713$. We obtained an average misclassification rate of 0.0326% with a standard deviation of 0.0152%.

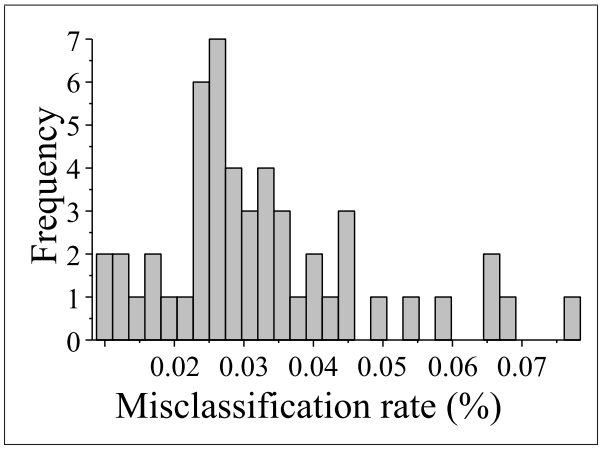

Figure 2.8 Ramp potentials : frequency histogram of the misclassification rate (in %).

2.7 Discussion

Our experiments show that the proposed learning algorithms successfully train the quantron in the restricted task of inferring parameter values from images. With rectangular potentials, the geometrical analysis of the decision boundaries divides the optimization problem into simple cases where gradient descent has good convergence properties. With ramp potentials, the analytical equations provide an efficient method to determine the values of the parameters in a gradient-free approach.

We emphasize that, in the scope of this paper, the sole goal of our learning algorithms is to retrieve parameter values responsible for generating a particular image. Thus, we did not use any validation procedure such as test sets and early stopping.

The more general problem of training the quantron with real classification data needs to be addressed by future research, since the algorithms presented in this work depend on precise characteristics of output images that might be difficult to determine with incomplete or noisy images. Generalization to inputs of higher dimension can also be considered. Furthermore, we need to do exhaustive experimentation with the algorithms to find heuristic rules to set the learning rate and the sigmoid slope in the rectangular case. Finally, we are interested in training networks of quantrons, so we will investigate the applicability of these algorithms to this difficult problem.

2.8 Conclusion

Using two types of surrogate potentials and a specific configuration of the quantron, we obtained analytical expressions for the decision boundaries of a single neuron and derived geometrical characteristics that are useful to develop learning algorithms. Our experiments

show the algorithms can find near optimal solutions when learning parameters from images. These results are a first step towards improved learning algorithms able to train quantrons and networks with more general data.

En résumé : Dans ce deuxième article, nous avons obtenu une relation explicite entre les paramètres et la frontière de décision pour une configuration spécifique du quantron qui admet une expression analytique de la fonction d'activation. L'étude des caractéristiques géométriques de la frontière de décision a mené au développement de deux algorithmes d'apprentissage qui trouvent des solutions presque optimales à des problèmes de classification constitués d'images générées par le quantron. Par contre, il est important de souligner que la nature géométrique de ces algorithmes pourrait limiter leur capacité à résoudre des problèmes issus de données réelles. De plus, rien n'indique que les discontinuités de la fonction d'activation et de ses dérivées peuvent s'incorporer harmonieusement à un réseau. Ainsi, le prochain article apporte une solution au problème de l'apprentissage dans un réseau de quantrons en présentant une nouvelle méthode d'approximation analytique permettant le lissage des discontinuités du temps d'activation.

CHAPITRE 3

ARTICLE 3 : NEW APPROXIMATION METHOD FOR SMOOTH ERROR BACKPROPAGATION IN A QUANTRON NETWORK

Présentation : Ce chapitre reprend un article accepté pour publication le 31 juillet 2014 dans *Neural Networks* par S. de Montigny.

Nous achevons notre approche analytique de l'apprentissage du quantron en proposant une approximation du temps d'activation. Nous prouvons que cette approximation, considérée comme une fonction des paramètres et des entrées du quantron, possède des dérivées partielles continues. Ainsi, nous obtenons un algorithme de rétropropagation lisse (*Smooth Backpropagation*) qui permet l'entraînement des neurones silencieux dans un réseau de quantrons.

Abstract : In this work, we propose a new approximation method to perform error backpropagation in a quantron network while avoiding the silent neuron problem that usually affects networks of realistic neurons. In our experiments, we train quantron networks to solve the XOR problem and other nonlinear classification problems. We achieve this while using less parameters than the number necessary to solve the same problems with networks of perceptrons or spiking neurons.
Keywords : Quantrons; Smooth Approximation; Backpropagation; Spiking Neurons; Classification.

3.1 Introduction

The quantron (Labib, 1999) is an advanced neuron model similar to spiking neurons (Gerstner, 1995; Maass, 1997), but with an important difference : spiking neurons compute precise spike times, whereas the quantron computes a regular firing interval. Due to this difference, the quantron generates classification boundaries with a more interesting structure than those generated by a single spiking neuron (Labib & de Montigny, 2012). In this work, we adapt the classical backpropagation algorithm (Rumelhart *et al.*, 1986) to the quantron's activation process to train a network efficiently. For spiking neurons based on the spike response model (Gerstner, 1995), the Spikeprop algorithm (Bohte *et al.*, 2000, 2002) has been devised to train a network via gradient descent. Being the first backpropagation-like supervised learning algorithm for spiking neural networks, Spikeprop attracted the attention

of several researchers (Xin & Embrechts, 2001; Schrauwen & Van Campenhout, 2004; Booij & tat Nguyen, 2005; Wu *et al.*, 2006; McKennoch *et al.*, 2006; Ghosh-Dastidar & Adeli, 2007, 2009a; Delshad *et al.*, 2010; Abiyev *et al.*, 2012; Thiruvarudchelvan & Bossomaier, 2012; Xu *et al.*, 2013). However, the creators of the algorithm (and others) have reported some convergence problems. The main difficulty with Spikeprop is the presence of "silent neurons", i.e. neurons that never fire and cannot be trained (Bohte *et al.*, 2002; Booij & tat Nguyen, 2005; Wu *et al.*, 2006; Schrauwen & Van Campenhout, 2006; Thiruvarudchelvan *et al.*, 2013). This problem can be alleviated with heuristic modifications to parameter update rules (Schrauwen & Van Campenhout, 2004; Booij & tat Nguyen, 2005; Wu *et al.*, 2006) or by using a surrogate output for silent neurons (Ghosh-Dastidar & Adeli, 2007, 2009a). Still, these two methods can be problematic : the first breaks the relation between the gradient and the error surface, and the second alters the network's computations. Other approaches avoiding backpropagation can be used for the supervised training of spiking neural networks, but they do not fully exploit the mathematical model of neural activity. These alternate methods are surveyed in Kasiński & Ponulak (2006) and Ghosh-Dastidar & Adeli (2009b).

The difficulties with backpropagation in networks of quantrons or spiking neurons are due to the computation of the neuron's activation time t^*. This time is the exact moment where a function $A(t)$, formed by the summation of post-synaptic potentials in the neuron, reaches an activation threshold noted Γ. Thus, t^* is found by solving $A(t) = \Gamma$ numerically. Figure 3.1 shows an example of a sum of potentials reaching the threshold and the corresponding activation time.

To train a network via backpropagation, it is necessary to differentiate t^* with regard to parameters such as height or width of potentials. However, explicit differentiation is impossible since t^* cannot be expressed analytically. In the Spikeprop algorithm, the derivative of t^* with regard to a parameter ω is given by

$$\frac{\partial t^*}{\partial \omega} = - \left. \frac{\frac{\partial}{\partial \omega} A(t;\omega)}{\frac{\partial}{\partial t} A(t)} \right|_{t=t^*} \tag{3.1}$$

where the notation $A(t;\omega)$ is used to emphasize that $A(t)$ depends on the parameter ω. This is tantamount to implicit differentiation of $A(t) = \Gamma$ with regard to ω, which can be justified formally if $A(t)$ is continuously differentiable in t and ω (Yang *et al.*, 2012). However, this expression is the source of the silent neuron problem since it is equal to zero if $A(t)$ does not reach the threshold. Indeed, this cutoff effect prevents the modification of parameters when a neuron remains silent for all network input patterns used in the training process. To circumvent this limitation, we propose an analytical approximation of t^* for which equation (3.1) can be computed explicitly, thus allowing error backpropagation in the network without

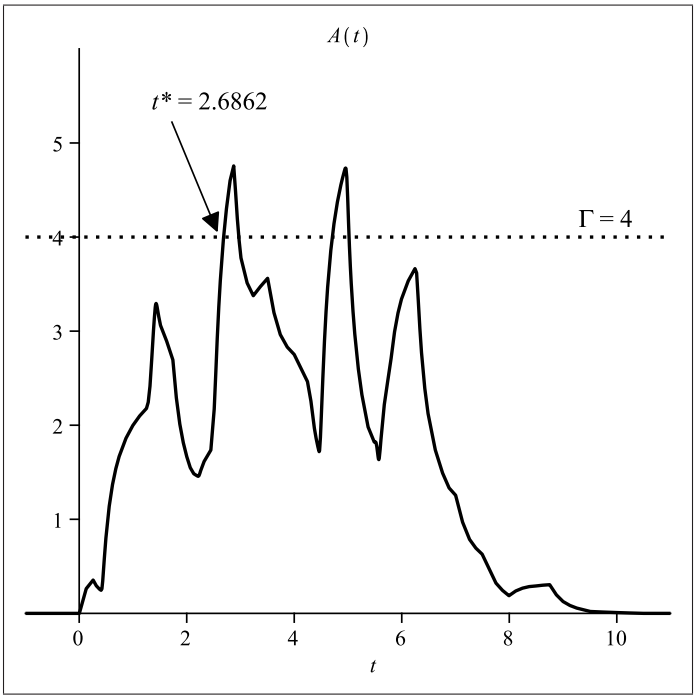

Figure 3.1 Sum of post-synaptic potentials with threshold $\Gamma = 4$ (represented by a dotted line) and activation time $t^* = 2.6862$ (the first solution to the equation $A(t) = \Gamma$).

cutoff. We call the resulting learning algorithm "Smooth Backpropagation".

We believe that this work is a useful contribution to the study of advanced neural network models and their applications. In particular, it should be of interest to researchers who are developing supervised learning algorithms for spiking neural networks.

The content of our paper is as follows. First, we present a simplified model of forward propagation for a network of quantrons. Then, we describe the proposed approximation method and we express the local gradient of a neuron via integrals which we compute explicitly. In our experiments, we train quantron networks via Smooth Backpropagation to successfully solve the XOR problem, and we compare its convergence to that of Spikeprop. Additionally, we show that Smooth Backpropagation can train a single quantron to solve the XOR problem using fewer parameters than a network of perceptrons. We also apply Smooth Backpropagation to two other nonlinear classification problems. Finally, we discuss our results and mention possible improvements.

3.2 Forward propagation in a quantron network

We present a brief review of the quantron artificial neuron model, which is inspired from the stochastic diffusion of neurotransmitters across the synaptic cleft (Labib, 1999; Connolly

& Labib, 2009). Then, we simplify the model for computational efficiency, while keeping its salient features. This surrogate model is the basis on which we build our smooth approximation of the activation time.

3.2.1 Quantron model

In a layered network of quantrons :

- the input values presented to the network are interpreted as the regular firing interval of the post-synaptic potentials arising in the neurons of the first layer;
- a neuron activates when the function $A(t)$, formed by the sum of post-synaptic potentials, reaches a threshold Γ;
- the activation time, noted t^* (the earliest time value where $A(t)$ reaches Γ) is used as the output value of the neuron;
- this output is propagated to the second layer, where it is also interpreted as a regular firing interval;
- if $A(t)$ never reaches Γ, the neuron stays inactive and does not generate post-synaptic potentials in the next layer.

For a network with a single neuron in its last layer, the final output can be the neuron's activation time (for regression), or a binary value ("1" or "0") encoding its active or inactive state (for classification).

The post-synaptic potentials of the quantron are represented by a function $p(t)$ given by

$$p(t) = \begin{cases} 2Q\left(\frac{\ln a}{\sqrt{t}}\right) & \text{if } 0 \leq t < s, \\ 2Q\left(\frac{\ln a}{\sqrt{s}}\right) - 2Q\left(\frac{\ln a}{\sqrt{t-s}}\right) & \text{if } s \leq t < 2s, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where :

- $Q(z)$ is the probability that a random variable with standard normal distribution is greater than z;
- s is a parameter representing the half-width of the potential;
- the constant a is the width of the synaptic cleft which neurotransmitters have to cross to reach receptors on the neuron's dendrite.

When a neuron receives a value x_j at its jth input terminal ($j = 1, ..., M$), N_j potentials are generated. The onset of each successive potential is delayed by x_j units of time. The potentials have a weight parameter w_j and the complete train of potentials is delayed by a parameter θ_j. Thus, for a single quantron, the training parameters are :

- the weights $w_1, \ldots, w_M$;

- the half-widths $s_1, \ldots, s_M$;
- the delays $\theta_1, \ldots, \theta_M$.

Although the neuron's threshold Γ is a physical constant, we will treat it as a parameter subject to an update rule. Previous work on Spikeprop showed that using a threshold update rule can improve the speed of learning algorithms (Schrauwen & Van Campenhout, 2004).

Although the parameters could take any coherent value (real numbers for weights and positive real numbers for half-widths, delays and thresholds), we find it useful to constrain the admissible values. This ensures that potentials from different inputs have a similar scale and position in time, and helps them interact to enable significant neural activation patterns. Thus, we introduce the hyper-parameters $w_{\min}, w_{\max}, s_{\min}, s_{\max}, \theta_{\min}, \theta_{\max}, \Gamma_{\min}, \Gamma_{\max}$ and implement the following constraints : $w_{\min} \leq w_j \leq w_{\max}$, $0 < s_{\min} \leq s_j \leq s_{\max}$, $0 \leq \theta_{\min} \leq \theta_j \leq \theta_{\max}$ and $0 < \Gamma_{\min} \leq \Gamma \leq \Gamma_{\max}$.

Let $A(t)$ represent the sum of post-synaptic potentials of a neuron. We call $A(t)$ the "activation function", since it determines if a neuron activates or not. We can express $A(t)$ as

$$A(t) = \sum_{j=1}^{M} \sum_{i=0}^{N_j - 1} w_j p\left(t - \theta_j - i x_j\right).$$

Figure 3.2 shows an example of the activation function for fixed parameter values.

The activation time of the quantron, noted t^*, is defined by

$$t^* = \inf\{t > 0 : A(t) = \Gamma\}.$$

When $A(t) < \Gamma$ for all t, t^* takes an infinite value and the quantron is inactive, producing no potential elsewhere in the network. Figure 3.3 gives a functional view of the quantron model, and figure 3.4 shows neural transmission across the synaptic cleft.

3.2.2 Surrogate model

To simplify $A(t)$, we represent potentials using the triangular function $p(t)$ given by

$$p(t) = \max\left(1 - |t - 1|, 0\right). \tag{3.3}$$

Figure 3.5 illustrates the realistic and triangular potentials given by equations (3.2) and (3.3).

The activation function is now given by

$$A(t) = \sum_{j=1}^{M} \sum_{i=0}^{N_j - 1} w_j \max\left(1 - \left|\frac{t - \theta_j - i x_j}{s_j} - 1\right|, 0\right). \tag{3.4}$$

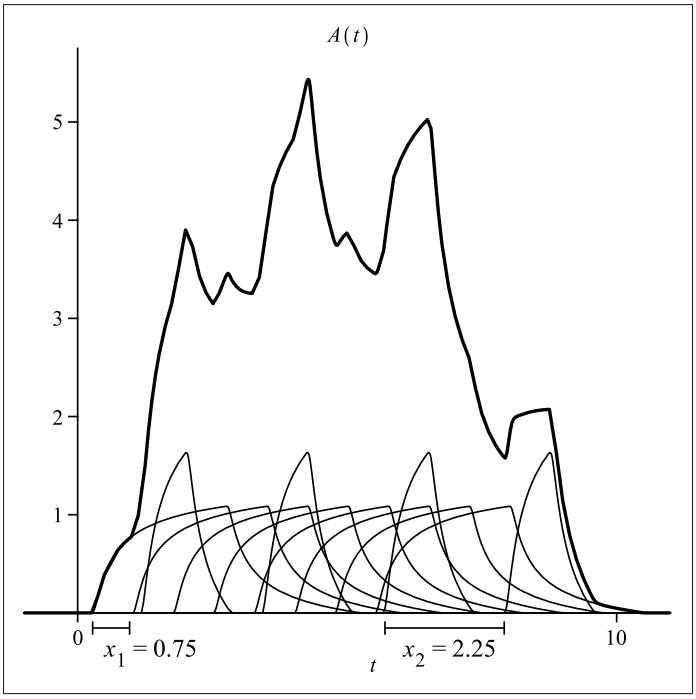

Figure 3.2 Two-input quantron : individual potentials (thin line) and activation function (bold line). Input and parameter values are $x_1 = 0.75$, $x_2 = 2.25$, $w_1 = 1.5$, $w_2 = 3$, $s_1 = 2.5$, $s_2 = 0.85$, $\theta_1 = 0.25$, $\theta_2 = 1.15$, $N_1 = 8$, $N_2 = 4$, $a = 1.75$.

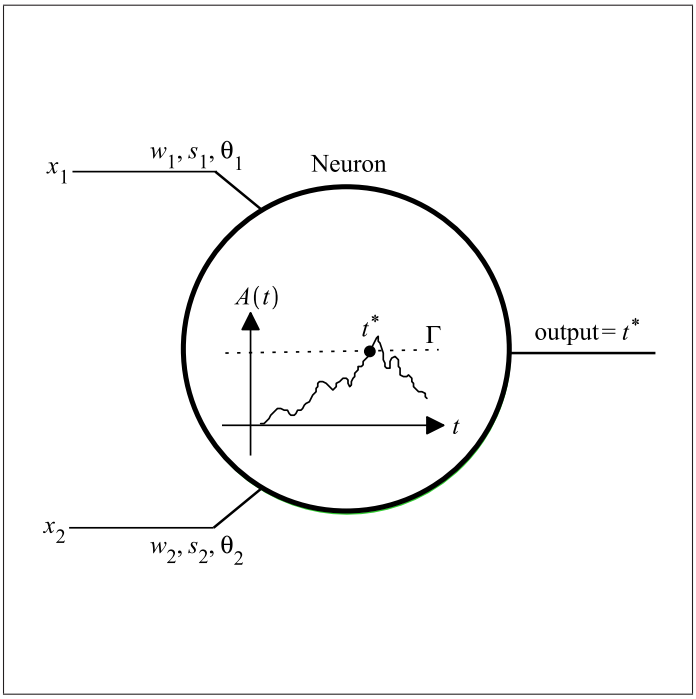

Figure 3.3 The activation time t^* is used as the output value of a single neuron. For a two-input quantron, t^* is determined by the inputs x_1, x_2 and parameters $w_1, s_1, \theta_1, w_2, s_2, \theta_2$.

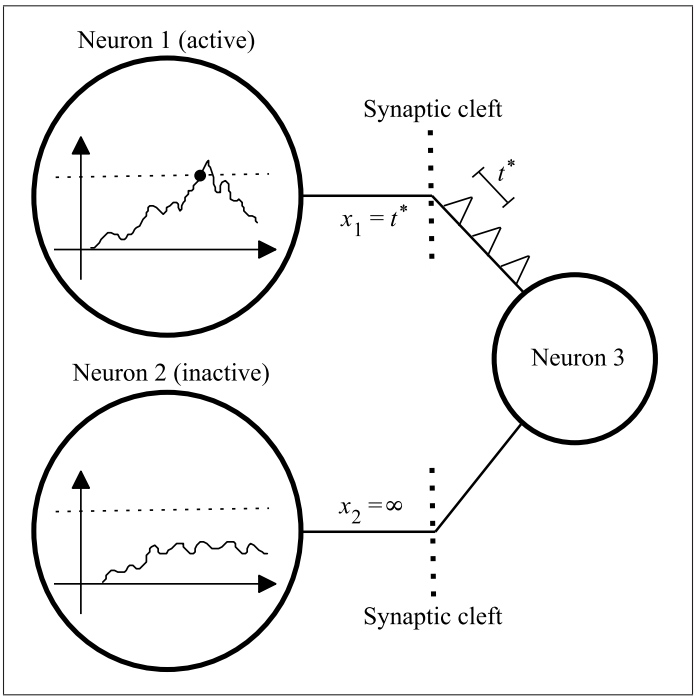

Figure 3.4 Neural transmission in the quantron model. If the activation threshold is reached, post-synaptic potentials are generated across the synaptic cleft.

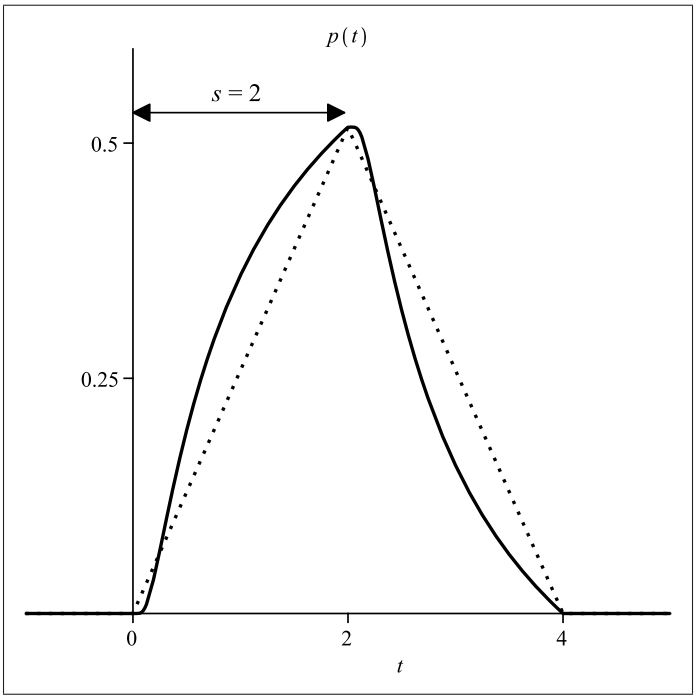

Figure 3.5 Realistic potential (solid line, $w = 1$, $a = 2.5$, $s = 2$) and triangular potential (dotted line, $w = 0.517$, $s = 2$).

To enable smooth computation in the network, we must avoid using infinite values for t^*. Therefore, we enforce a maximal value with a hyper-parameter T (a fixed time value). We have

$$t^* = \min(\inf\{t > 0 : A(t) = \Gamma\}, T) \tag{3.5}$$

so that $t^* = T$ for a quantron that does not become active before T units of time. Also, to distinguish a late activation from a failure to activate, we introduce a second output indicating the neuron's activation level (0 or 1). The activation level is found by comparing the maximum of $A(t)$ on $[0, T]$ with the threshold Γ. Thus, each neuron has a primary output o_1 and a secondary output o_2 respectively defined by

$$o_1 = t^*$$

and

$$o_2 = u\left(\max_{0<t\leq T}[A(t)] - \Gamma\right)$$

where $u(\cdot)$ is the Heaviside function.

To accommodate this second output, we modify the neurons in the middle layers of the network. In the jth synapse of such neurons, the input x_j receives the value provided by the output o_1 and a new input y_j receives the value provided by o_2. The post-synaptic potentials stemming from an inactive neuron are suppressed by using the following activation function

$$A(t) = \sum_{j=1}^{M} \sum_{i=0}^{N_j-1} y_j w_j \max\left(1 - \left|\frac{t - \theta_j - ix_j}{s_j} - 1\right|, 0\right) \tag{3.6}$$

where the weight w_j is canceled if $y_j = 0$ and is unchanged if $y_j = 1$.

3.3 Smooth approximation of activation time

To obtain smooth neuron outputs, we need to approximate the maximum of the activation function $A(t)$. A well-known approximation of the maximum of n numbers $A_1, A_2, \ldots, A_n$ is given by

$$\mu(\lambda) = \frac{1}{\lambda} \ln\left(\sum_{k=1}^{n} e^{\lambda A_k}\right) \tag{3.7}$$

where $\lambda > 0$ is a precision hyper-parameter, see Boyd & Vandenberghe (2004). It is easy to show that

$$\max_{k=1,\ldots,n}\{A_k\} \leq \mu(\lambda) \leq \max_{k=1,\ldots,n}\{A_k\} + \frac{\ln n}{\lambda}$$

which leads to the following limit

$$\lim_{\lambda\to\infty}\mu(\lambda)=\max_{k=1,\ldots,n}\{A_k\}.$$

Computing $\mu(\lambda)$ for large values of λ must be avoided due to the numerical overflow of the exponential function. In this work, we rely on the smoothness of the approximation for moderate values of λ.

Equation (3.7) can be adapted to approximate the global maximum of $A(t)$. Let

$$\mu(\lambda;T)=\frac{1}{\lambda}\ln\left(\int_0^T e^{\lambda A(t)}dt\right)$$

and let $\epsilon>0$. Since $A(t)$ is continuous, we have the following bounds for $\int_0^T e^{\lambda A(t)}dt$:

$$L_\epsilon\exp\left\{\lambda\left(\max_{0\le t\le T}[A(t)]-\epsilon\right)\right\}\le\int_0^T e^{\lambda A(t)}dt\le T\exp\left\{\lambda\max_{0\le t\le T}[A(t)]\right\} \tag{3.8}$$

where L_ϵ is the measure of the set

$$T_\epsilon=\left\{t\in[0,T]:\max_{0\le t\le T}[A(t)]-A(t)\le\epsilon\right\}.$$

Figure 3.6 illustrates the relation between T_ϵ and $A(t)$.

Equation (3.8) yields the following bounds for $\mu(\lambda;T)$:

$$\max_{0\le t\le T}[A(t)]-\epsilon+\frac{\ln L_\epsilon}{\lambda}\le\mu(\lambda;T)\le\max_{0\le t\le T}[A(t)]+\frac{\ln T}{\lambda}.$$

Taking the limit $\lambda\to\infty$, the bounds become

$$\max_{0\le t\le T}[A(t)]-\epsilon\le\lim_{\lambda\to\infty}\mu(\lambda;T)\le\max_{0\le t\le T}[A(t)]$$

for all $\epsilon>0$. Thus, we have the limit

$$\lim_{\lambda\to\infty}\mu(\lambda;T)=\max_{0\le t\le T}[A(t)],$$

which justifies the approximation. Again, we only intend to use moderate values of λ to prevent numerical overflow. Furthermore, when computing $\int_0^T e^{\lambda A(t)}dt$, we subtract the threshold Γ from $A(t)$, effectively computing $\int_0^T e^{\lambda(A(t)-\Gamma)}dt$ which also helps prevent overflow.

The activation time of the quantron can be computed by replacing $A(t)$ by the running

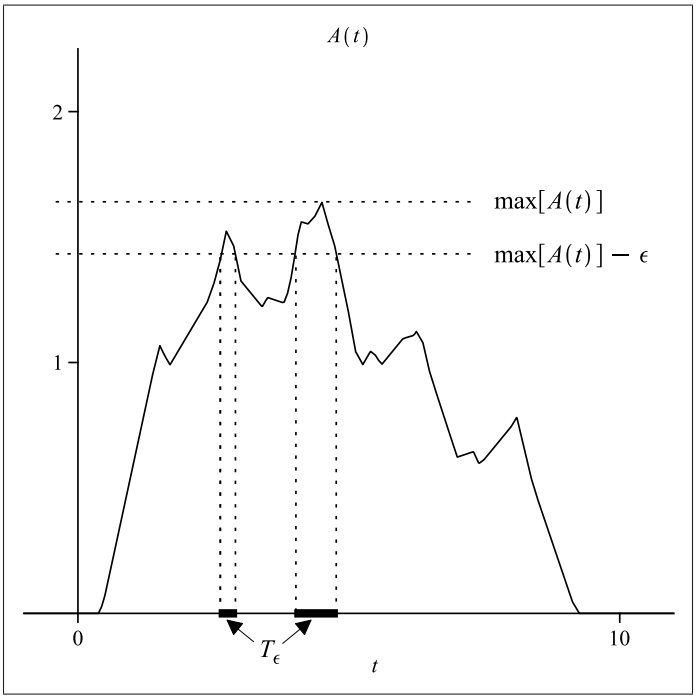

Figure 3.6 The set T_ϵ is comprised of the values of t for which the activation function $A(t)$ lies between $\max[A(t)] - \epsilon$ and $\max[A(t)]$.

maximum $\max_{0\leq v\leq t}[A(v)]$ in equation (3.5). Since $\max_{0\leq v\leq t}[A(v)]$ is monotone increasing for $t \in [0, T]$, we find that

$$t^* = \int_0^T u\left(\Gamma - \max_{0\leq v\leq t}[A(v)]\right) dt.$$

To obtain a smooth approximation of t^*, we replace $\max_{0\leq v\leq t}[A(v)]$ by $\mu(\lambda; t)$ and $u(z)$ by the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-\lambda z}}$$

with a slope hyper-parameter equal to λ. Thus, the primary output of a neuron is given by

$$o_1 = \int_0^T \frac{1}{1 + D(t)} dt \tag{3.9}$$

where $D(t) = \int_0^t e^{\lambda(A(v)-\Gamma)} dv$. Similarly, the secondary output o_2 is given by

$$o_2 = 1 - \frac{1}{1 + D(T)}. \tag{3.10}$$

Theorem 3.1 : The outputs o_1 and o_2, given by equations (3.9) and (3.10), are continuously differentiable functions of the variables $x_j, y_j, w_j, s_j, \theta_j$ and Γ.

Proof : The continuity of derivatives with regard to y_j, w_j and Γ is easily proven using results for integrals depending on a parameter (Dieudonné, 1960). For example, we have

$$\frac{\partial o_1}{\partial w_j} = -\lambda \int_0^T \frac{1}{(1+D(t))^2} \int_0^t e^{\lambda(A(v)-\Gamma)} \frac{\partial A(v)}{\partial w_j} dv dt,$$

which is continuous in w_j since

$$\frac{\partial A(v)}{\partial w_j} = \sum_{i=0}^{N_j-1} y_j \max\left(1 - \left|\frac{v-\theta_j-ix_j}{s_j} - 1\right|, 0\right)$$

is continuous in w_j and v. For x_j, s_j and θ_j, the partial derivative of $A(v)$ has discontinuities. Still, we obtain continuous derivatives for o_1 and o_2 due to the smoothing effect of the integral. More precisely, let $a < x < b$ and $c < y < d$. The derivative of $F(y) = \int_a^b f(x,y)dx$ at $y^* \in (c,d)$ is given by $F'(y^*) = \int_a^b [\frac{\partial}{\partial y} f(x,y)]_{y=y^*} dx$ if $[\frac{\partial}{\partial y} f(x,y)]_{y=y^*}$ exists for almost all $x \in (a,b)$, in the sense of measure theory. This result is valid under two conditions. First, we must have $\int_a^b |f(x,y)|\, dx < \infty$ for all $y \in (c,d)$; $f(x,\cdot)$ is said to be integrable. Second, we must have $|f(x,y) - f(x,y^*)| \leq K\,|y-y^*|$, where K is a constant, for all $y \in (c,d)$, $y^* \in (c,d)$ and $x \in (a,b)$; $f(\cdot,y)$ is said to be Lipschitz continuous. For a formal presentation of a more general version of this result, see Briane & Pagès (2004). □

Remark : The function $e^{\lambda(A(v)-\Gamma)}$ is integrable in v since $A(t)$ is a bounded function on $[0,T]$. The condition for Lipschitz continuity means that the derivative of $e^{\lambda(A(v)-\Gamma)}$ with regard to any input or parameter ω (where it exists) must not be too steep. Since we use the parameter constraints of section 3.2.1, and since the inputs are naturally bounded for neurons in the middle layers of the network ($0 < x_j < T$ and $0 < y_j < 1$), the condition is met.

We will now express the derivatives of o_1 and o_2 via several integrals. Let $[t_0, t_1]$ be an arbitrary interval. We define $D(t_0,t_1) = \int_{t_0}^{t_1} e^{\lambda(A(v)-\Gamma)} dv$, with $D(t) \equiv D(0,t)$, and

$$\begin{aligned}
E(t_0,t_1) &= \int_{t_0}^{t_1} v e^{\lambda(A(v)-\Gamma)} dv, \\
G(t_0,t_1) &= \int_{t_0}^{t_1} \frac{1}{(1+D(t))^2} dt, \\
H(t_0,t_1) &= \int_{t_0}^{t_1} \frac{1}{(1+D(t))^2} \int_{t_0}^{t} e^{\lambda(A(v)-\Gamma)} dv dt, \\
I(t_0,t_1) &= \int_{t_0}^{t_1} \frac{1}{(1+D(t))^2} \int_{t_0}^{t} v e^{\lambda(A(v)-\Gamma)} dv dt.
\end{aligned}$$

Let $t_{j,i,h} = \theta_j + ix_j + hs_j$ for $h = 0, 1, 2$. The gradient of output o_l (for $l = 1, 2$) can be expressed as

$$\begin{bmatrix} \partial o_l/\partial x_j \\ \partial o_l/\partial y_j \\ \partial o_l/\partial s_j \\ \partial o_l/\partial \theta_j \\ \partial o_l/\partial w_j \\ \partial o_l/\partial \Gamma \end{bmatrix} = \begin{bmatrix} \frac{\lambda w_j y_j}{s_j}\gamma_{l,j} \\ \frac{\lambda w_j}{s_j}\left(\theta_j\beta_{l,j} + x_j\gamma_{l,j} - \delta_{l,j}\right) - 2\lambda w_j\alpha_{l,j} \\ \frac{\lambda w_j y_j}{s_j^2}\left(\delta_{l,j} - \theta_j\beta_{l,j} - x_j\gamma_{l,j}\right) \\ \frac{\lambda w_j y_j}{s_j}\beta_{l,j} \\ \frac{\lambda y_j}{s_j}\left(\theta_j\beta_{l,j} + x_j\gamma_{l,j} - \delta_{l,j}\right) - 2\lambda y_j\alpha_{l,j} \\ -\lambda H(0,T) \end{bmatrix}$$

where

$$\begin{aligned}
\alpha_{l,j} &= \sum_{i=0}^{N_j-1} \Delta_{j,i,1}^{l,1}, \\
\beta_{l,j} &= \sum_{i=0}^{N_j-1} \left(\Delta_{j,i,0}^{l,1} - \Delta_{j,i,1}^{l,1}\right), \\
\gamma_{l,j} &= \sum_{i=0}^{N_j-1} i\left(\Delta_{j,i,0}^{l,1} - \Delta_{j,i,1}^{l,1}\right), \\
\delta_{l,j} &= \sum_{i=0}^{N_j-1} \left(\Delta_{j,i,0}^{l,2} - \Delta_{j,i,1}^{l,2}\right), \\
\Delta_{j,i,h}^{1,1} &= \int_0^T \frac{1}{(1+D(t))^2} \int_{\min(t,t_{j,i,h})}^{\min(t,t_{j,i,h+1})} e^{\lambda(A(v)-\Gamma)} dv dt, \\
\Delta_{j,i,h}^{1,2} &= \int_0^T \frac{1}{(1+D(t))^2} \int_{\min(t,t_{j,i,h})}^{\min(t,t_{j,i,h+1})} v e^{\lambda(A(v)-\Gamma)} dv dt, \\
\Delta_{j,i,h}^{2,1} &= \frac{-1}{(1+D(T))^2} \int_{\min(T,t_{j,i,h})}^{\min(T,t_{j,i,h+1})} e^{\lambda(A(v)-\Gamma)} dv, \\
\Delta_{j,i,h}^{2,2} &= \frac{-1}{(1+D(T))^2} \int_{\min(T,t_{j,i,h})}^{\min(T,t_{j,i,h+1})} v e^{\lambda(A(v)-\Gamma)} dv.
\end{aligned}$$

Let $r_{j,i,h} = \min(T, t_{j,i,h})$ for $h = 0, 1$. We find that $\Delta_{j,i,h}^{1,1}$, $\Delta_{j,i,h}^{1,2}$, $\Delta_{j,i,h}^{2,1}$ and $\Delta_{j,i,h}^{2,2}$ are given by

$$\begin{aligned}
\Delta_{j,i,h}^{1,1} &= H(r_{j,i,h}, r_{j,i,h+1}) + G(r_{j,i,h+1}, T)D(r_{j,i,h}, r_{j,i,h+1}), \\
\Delta_{j,i,h}^{1,2} &= I(r_{j,i,h}, r_{j,i,h+1}) + G(r_{j,i,h+1}, T)E(r_{j,i,h}, r_{j,i,h+1}), \\
\Delta_{j,i,h}^{2,1} &= \frac{-D(r_{j,i,h}, r_{j,i,h+1})}{(1+D(T))^2}, \\
\Delta_{j,i,h}^{2,2} &= \frac{-E(r_{j,i,h}, r_{j,i,h+1})}{(1+D(T))^2}.
\end{aligned}$$

For the neurons in the first layer, the derivatives $\frac{\partial o_l}{\partial x_j}$ and $\frac{\partial o_l}{\partial y_j}$ are not used.

3.4 Computation of integrals

In this section, we compute explicitly the integrals obtained in section 3.3. Depending on the form of $A(t)$, the evaluation of the resulting expressions on a computer is subject to numerical errors. Thus, we provide simple implementation rules to ensure the accuracy of the computations.

Let $\tau_0, \tau_1, \tau_2, \ldots, \tau_n$ be the ordered list of time values from the set

$$\tau = \{0, T\} \cup \{t_{j,i,h}\}$$

(with $j = 1, \ldots, M$; $i = 0, ..., N_j - 1$ and $h = 0, 1, 2$). For particular values of parameters and inputs, there could be duplicates in this list. For now, we consider that these duplicates can be removed, so that each value in the list is unique. We discuss the matter further in section 3.4.4.

The function $A(t)$ can be expressed as

$$A(t) = a_k t + b_k \text{ if } \tau_{k-1} \leq t < \tau_k$$

for $k = 1, \ldots, n$, where a_k and b_k are given by

$$a_k = \frac{A(\tau_k) - A(\tau_{k-1})}{\tau_k - \tau_{k-1}}, b_k = A(\tau_{k-1}) - a_k \tau_{k-1}$$

and where $A(\tau_k)$ and $A(\tau_{k-1})$ are calculated with equation (3.4) for neurons in the first layer and with equation (3.6) for middle layers.

We introduce some intermediate quantities. Let

$$D_k(t) = \int_{\tau_{k-1}}^{t} e^{\lambda(a_k v + b_k - \Gamma)} dv$$

with $D_k \equiv D_k(\tau_k)$. Let $C_1 = 1$, $C_k = C_{k-1} + D_{k-1}$ for $k = 2, \ldots, n+1$. Let

$$F_k = \int_{\tau_{k-1}}^{\tau_k} \frac{1}{C_k + D_k(t)} dt,$$

$$G_k = \int_{\tau_{k-1}}^{\tau_k} \frac{1}{(C_k + D_k(t))^2} dt,$$

$$H_k = \int_{\tau_{k-1}}^{\tau_k} \frac{D_k(t)}{\left(C_k + D_k(t)\right)^2} dt,$$

$$I_k = \int_{\tau_{k-1}}^{\tau_k} \frac{E_k(t)}{\left(C_k + D_k(t)\right)^2} dt.$$

First, we express outputs and derivatives with these quantities. Second, we compute these quantities explicitly.

3.4.1 Outputs

The output o_1, defined by equation (3.9), is computed as follows :

$$o_1 = \sum_{k\,:\,0<\tau_k\leq T} \int_{\tau_{k-1}}^{\tau_k} \frac{1}{1 + \sum_{m=1}^{k-1} \int_{\tau_{m-1}}^{\tau_m} e^{\lambda(a_m v + b_m - \Gamma)} dv + \int_{\tau_{k-1}}^{t} e^{\lambda(a_k v + b_k - \Gamma)} dv} dt.$$

In simplified notation, we have

$$o_1 = \sum_{k\,:\,0<\tau_k\leq T} F_k.$$

The output o_2, defined by equation (3.10), is directly given by

$$o_2 = 1 - \left(1 + \sum_{k\,:\,0<\tau_k\leq T} D_k\right)^{-1}.$$

3.4.2 Derivatives

The derivatives of o_1 and o_2 are expressed via the integrals $D(t_0, t_1)$, $G(t_0, t_1)$, $H(t_0, t_1)$ and $I(t_0, t_1)$. The first is given by

$$\begin{aligned} D(t_0, t_1) = \int_{t_0}^{t_1} e^{\lambda(A(v)-\Gamma)} dv &= \sum_{k\,:\,t_0<\tau_k\leq t_1} \int_{\tau_{k-1}}^{\tau_k} e^{\lambda(a_k v + b_k - \Gamma)} dv \\ &= \sum_{k\,:\,t_0<\tau_k\leq t_1} D_k. \end{aligned}$$

When computing these integrals, the value of t_0 is an element of τ. Let $E_k(t) = \int_{\tau_{k-1}}^{t} v e^{\lambda(a_k v + b_k - \Gamma)} dv$ with $E_k \equiv E_k(\tau_k)$. The three other integrals are given by

$$E(t_0, t_1) = \int_{t_0}^{t_1} v e^{\lambda(A(v)-\Gamma)} dv = \sum_{k\,:\,t_0<\tau_k\leq t_1} E_k,$$

$$G(t_0,t_1)=\int_{t_0}^{t_1}\frac{1}{(1+D(t))^2}dt=\sum_{k\,:\,t_0<\tau_k\le t_1}G_k,$$

$$\begin{aligned}H(t_0,t_1)&=\int_{t_0}^{t_1}\frac{1}{(1+D(t))^2}\int_{t_0}^{t}e^{\lambda(A(v)-\Gamma)}dvdt\\&=\sum_{k\,:\,t_0<\tau_k\le t_1}\left(\sum_{q\,:\,t_0<\tau_q\le\tau_{k-1}}G_kD_q\right)+\sum_{k\,:\,t_0<\tau_k\le t_1}H_k,\end{aligned}$$

$$\begin{aligned}I(t_0,t_1)&=\int_{t_0}^{t_1}\frac{1}{(1+D(t))^2}\int_{t_0}^{t}ve^{\lambda(A(v)-\Gamma)}dvdt\\&=\sum_{k\,:\,t_0<\tau_k\le t_1}\left(\sum_{q\,:\,t_0<\tau_q\le\tau_{k-1}}G_kE_q\right)+\sum_{k\,:\,t_0<\tau_k\le t_1}I_k.\end{aligned}$$

3.4.3 Computing F_k, G_k, H_k, I_k

The integrals $D_k(t)=\int_{\tau_{k-1}}^{t}e^{\lambda(a_kv+b_k-\Gamma)}dv$ and $E_k(t)=\int_{\tau_{k-1}}^{t}ve^{\lambda(a_kv+b_k-\Gamma)}dv$ are computed explicitly as follows :

$$D_k(t)=\begin{cases}(t-\tau_{k-1})e^{\lambda(b_k-\Gamma)} & \text{if } a_k=0,\\ \frac{1}{\lambda a_k}\left(e^{\lambda(a_kt+b_k-\Gamma)}-e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}\right) & \text{if } a_k\neq 0,\end{cases}$$

$$E_k(t)=\begin{cases}\frac{1}{2}\left(t^2-\tau_{k-1}^2\right)e^{\lambda(b_k-\Gamma)} & \text{if } a_k=0,\\ \frac{1}{\lambda a_k}\left(te^{\lambda(a_kt+b_k-\Gamma)}-\tau_{k-1}e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}-D_k(t)\right) & \text{if } a_k\neq 0.\end{cases}$$

The computation of F_k, G_k, H_k and I_k must be separated in the three following cases :

1) when $a_k\neq 0$ and $\lambda a_kC_k\neq e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}$;
2) when $a_k\neq 0$ and $\lambda a_kC_k=e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}$;
3) when $a_k=0$.

In the first case, we have

$$F_k=\frac{\lambda a_k(\tau_k-\tau_{k-1})-\ln\left(C_{k+1}/C_k\right)}{\lambda a_kC_k-e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}},$$

$$G_k=\frac{\lambda a_kF_k-\left(\frac{1}{C_k}-\frac{1}{C_{k+1}}\right)}{\lambda a_kC_k-e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}},$$

$$I_k = \frac{1}{\lambda a_k}\left(\frac{\tau_{k-1}}{C_k} - \frac{\tau_k}{C_{k+1}} + F_k - \tau_{k-1}e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}G_k - H_k\right).$$

In the second case, we have

$$F_k = e^{-\lambda(a_k\tau_{k-1}+b_k-\Gamma)} - e^{-\lambda(a_k\tau_k+b_k-\Gamma)},$$

$$G_k = \frac{\lambda a_k}{2}\left(e^{-2\lambda(a_k\tau_{k-1}+b_k-\Gamma)} - e^{-2\lambda(a_k\tau_k+b_k-\Gamma)}\right),$$

$$\begin{aligned} I_k = {} & \tau_{k-1}e^{-\lambda(a_k\tau_{k-1}+b_k-\Gamma)} - \tau_k e^{-\lambda(a_k\tau_k+b_k-\Gamma)} \\ & + \frac{1}{\lambda a_k}\left(F_k - \tau_{k-1}e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}G_k - H_k\right). \end{aligned}$$

In the third case, we have

$$F_k = e^{-\lambda(b_k-\Gamma)}\ln(C_{k+1}/C_k),$$

$$G_k = e^{-\lambda(b_k-\Gamma)}\left(\frac{1}{C_k} - \frac{1}{C_{k+1}}\right),$$

$$\begin{aligned} I_k = {} & \left(\tau_{k-1} - C_k e^{-\lambda(b_k-\Gamma)}\right)F_k + e^{-\lambda(b_k-\Gamma)}(\tau_k - \tau_{k-1}) \\ & - \frac{\tau_{k-1}(\tau_k - \tau_{k-1})}{C_{k+1}} - \frac{(\tau_k - \tau_{k-1})^2}{2C_{k+1}}. \end{aligned}$$

In all three cases, we have $H_k = F_k - C_k G_k$.

3.4.4 Implementation rules

Comparing floating-point numbers for equality is an unreliable operation. To avoid division by a very small number, we need a careful implementation in computer code of the expressions for F_k, G_k, H_k and I_k.

The conditions $\lambda a_k C_k = e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}$ and $a_k = 0$ can be implemented as

$$\left|\lambda a_k C_k - e^{\lambda(a_k\tau_{k-1}+b_k-\Gamma)}\right| \le \epsilon_1 \text{ and } |a_k| \le \epsilon_2,$$

where ϵ_1 and ϵ_2 are numerical tolerances.

Removing the duplicate time values from the set τ also requires equality comparisons. While time values with the same floating-point representation will be effectively removed,

there might remain some pair of adjacent values τ_{k-1} and τ_k which are numerically very close. In this case, the integrals F_k, G_k, H_k and I_k are approximately equal to zero. Indeed, we have the bounds

$$\begin{aligned} 0 \leq F_k &\leq \tau_k - \tau_{k-1}, \\ 0 \leq G_k &\leq \tau_k - \tau_{k-1}, \\ 0 \leq H_k &\leq \tau_k - \tau_{k-1}, \\ 0 \leq I_k &\leq T(\tau_k - \tau_{k-1}). \end{aligned}$$

Thus, we use another numerical tolerance, noted ϵ_3, and we remove either τ_{k-1} or τ_k from τ if $|\tau_k - \tau_{k-1}| \leq \epsilon_3$.

These bounds can also be used to check the numerical accuracy of the computations in floating-point arithmetic. If any quantity happens to be larger than its upper bound (or be negative), it is forced to take the value of the appropriate bound.

3.5 Experimentation

In this section, we apply our approximation method in a learning algorithm to solve binary classification problems. First, we describe the Smooth Backpropagation algorithm and we use it to train networks of quantrons to solve the XOR problem. Then, we pursue our experiment on harder classification problems. Finally, we study the accuracy of the approximation method.

3.5.1 Smooth Backpropagation

Using the outputs and derivatives developed in sections 3.3 and 3.4, we apply the standard error backpropagation algorithm on a quantron network with a hidden layer containing Q neurons and an output layer consisting of a single neuron. The network is represented by the notation M-Q-1 where M is the number of network inputs (for $Q = 0$, the 2-0-1 or 2-1 network contains a single neuron). The Smooth Backpropagation algorithm trains the network by minimizing the batch error which is determined by adding the cross-entropy error function for each sample pattern of the classification problem. The cross-entropy error function is given by

$$\mathcal{CE} = \begin{cases} -\ln(o_2) & \text{if } d = 1, \\ -\ln(1 - o_2) & \text{if } d = 0. \end{cases}$$

where o_2 is the secondary output of the network's last neuron. The batch error is minimized by gradient descent, which entails the following iterative update rules for the network's pa-

rameters.

The update rule for the parameter ω_k of the kth hidden neuron ($k = 1, 2, ..., Q$) is given by

$$\begin{aligned}\omega'_k &= \omega_k - \sum_{\text{input pairs}} \frac{\eta}{1-d-o_2}\frac{\partial o_2}{\partial \omega_k} \\ &= \omega_k - \sum_{\text{input pairs}} \frac{\eta}{1-d-o_2}\left(\frac{\partial o_2}{\partial x_k}\frac{\partial o_{k,1}}{\partial \omega_k} + \frac{\partial o_2}{\partial y_k}\frac{\partial o_{k,2}}{\partial \omega_k}\right)\end{aligned}$$

where η is a learning rate, ω'_k is the updated parameter, $o_{k,1}$ and $o_{k,2}$ are respectively the primary and secondary outputs of the kth hidden neuron, and x_k and y_k are the corresponding inputs for the last neuron.

The update rule for the parameter ω of the last neuron is given by

$$\omega' = \omega - \sum_{\text{input pairs}} \frac{\eta}{1-d-o_2}\frac{\partial o_2}{\partial \omega}.$$

To ensure error reduction at each iteration, we apply a simple line search method :

- when the error increases, we divide the learning rate η by 2 and reevaluate the error and derivatives;
- when the error decreases, we apply the update rules and proceed to the next iteration;
- after five iterations in which η remains the same, we multiply η by 2 (up to a maximum of 1).

The algorithm stops after a fixed number of iterations, or if η falls under 10^{-10}.

3.5.2 Hyper-parameters

Here, we list the values used for different hyper-parameters in our tests. These values were determined in preliminary experiments.

As mentioned in section 3.2.1, we enforce constraints on the parameters. Thus, parameters who break a constraint during learning are reset to the limit value of the broken constraint. For hidden neurons, we use the following limit values : $w_{\min} = -2$, $w_{\max} = 2$, $s_{\min} = 0.1$, $s_{\max} = 5$, $\theta_{\min} = 0$, $\theta_{\max} = 5$, $\Gamma_{\min} = 0.25$, $\Gamma_{\max} = 5$. For the last neuron of the network, we use the same values, except for $s_{\max}$ which is set to 8.

In all our tests, we use the hyper-parameter values $(\lambda, T) = (6, 15)$ for hidden neurons and $(\lambda, T) = (8, 30)$ for the last neuron (or for a single neuron if $Q = 0$).

Finally, we use the numerical tolerances $\epsilon_1 = \epsilon_2 = \epsilon_3 = 10^{-7}$ when computing the integrals of section 3.4.

3.5.3 XOR problem

The XOR problem is a well-known nonlinear classification task that can be solved by a network with a single hidden layer containing two perceptrons (linear threshold neurons) and an output layer consisting of a single perceptron. Also, it is known that a single quantron can solve the XOR problem (Labib, 1999). Whereas the perceptron network needs nine parameters for this task, the quantron uses six parameters (seven if Γ is trained). However, no learning algorithm has successfully trained the quantron to solve the XOR problem. We now show that the Smooth Backpropagation algorithm achieves this goal. Since the inputs of the quantron must take positive values, we use a special encoding for the XOR problem, as shown in table 3.1.

Table 3.1: XOR problem encoding.

Input 1	Input 2	Desired output d
1	1	0
1	2	1
2	1	1
2	2	0

In this first experiment, we test the Smooth Backpropagation algorithm with $Q = 0$, 2, 3, 4, 5 hidden neurons. For each case, we generate 10 network instances with initial parameters chosen randomly as follows. For a single neuron ($Q = 0$), we choose :

- w_1 in $[-0.5, 1]$ and w_2 in $[0, 1]$;
- s_j in $[0.25, 2.5]$ for $j = 1, 2$;
- θ_j in $[0, 3]$ for $j = 1, 2$;
- Γ in $[0.5, 1]$.

For $Q > 0$, we choose the parameters of hidden neurons in the same manner. For the last neuron, we choose :

- w_j in $[0, 1]$ for $j = 1, \ldots, Q$;
- s_j in $[1, 5]$ for $j = 1, \ldots, Q$;
- θ_j in $[0, 3]$ for $j = 1, \ldots, Q$;
- Γ in $[0.5, 1]$.

We repeat Smooth Backpropagation on each network instance with a different number of potentials. We set $N_1 = N_2 = N$ for neurons of the hidden layer (or for a single neuron), with $N = 2, 3, 4, 5, 6$. For $Q > 0$ we set $N_1 = \ldots = N_Q = N'$ for the neuron in the output layer, with $N' = 2, 3$.

In each test, we start with the learning rate $\nu = 1/256$ and execute the learning algorithm for 1000 iterations. When the algorithm ends, we evaluate the network's output for each input

pattern of the XOR problem. We say that the network has converged to a correct solution when its output is lower than 0.5 for inputs $(1, 1)$ and $(2, 2)$, and higher than 0.5 for inputs $(2, 1)$ and $(1, 2)$. We also evaluate the network's output with the exact t^* by equation (3.5). We refer to the former evaluation as "smooth" and the latter as "exact".

Results

Now, we give the results of this first experiment. Table 3.2 indicates the maximum and minimum number of convergent runs, for both smooth and exact network evaluation. Complete results are found in appendix B (tables C.1 to C.5). Figure 3.7 illustrates a typical error curve and learning rate curve.

Table 3.2: Maximum and minimum number of convergent runs obtained on the XOR problem for each network.

Network	Convergent runs (out of 10)			
	Smooth evaluation		Exact evaluation	
	Maximum	Minimum	Maximum	Minimum
2-1	9	3	9	3
2-2-1	9	4	9	3
2-3-1	10	7	8	3
2-4-1	10	7	9	3
2-5-1	10	9	10	3

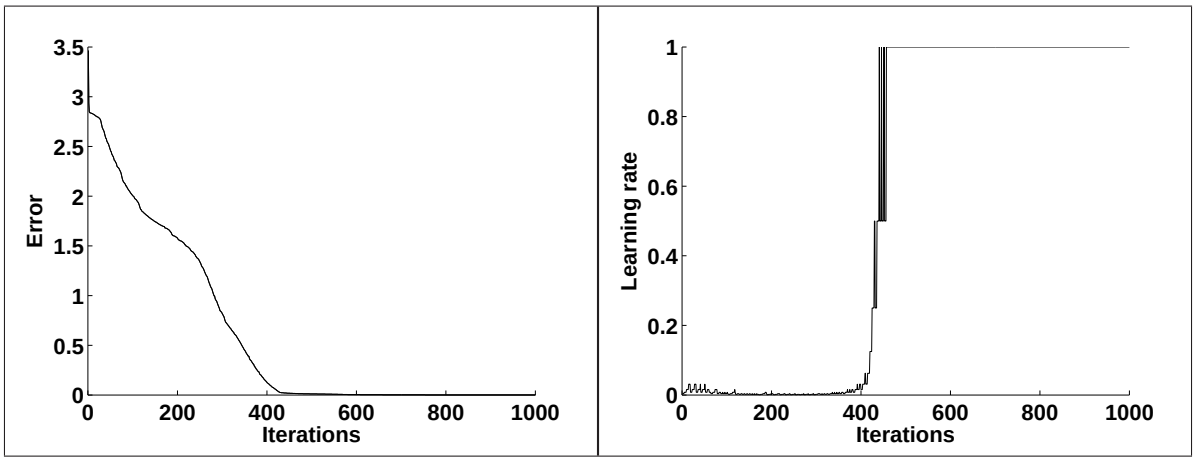

Figure 3.7 Example of error curve (left) and learning rate curve (right). The error curve is smooth and strictly decreasing due to line search. The learning rate curve shows adaptation of gradient descent to the steepness of the error surface.

The best configuration was the 2-5-1 network with $N = 5$ and $N' = 2$ for which all runs were convergent with smooth and exact evaluation. Even the 2-1 network (a single quantron)

with $N = 6$ solved the XOR problem correctly in 8 runs out of 10, also with smooth and exact evaluation. This result confirms that a single quantron can learn to solve the XOR problem while using less parameters than a perceptron network. In general, it seems that adding potentials and hidden neurons improves convergence.

Finally, to illustrate the capacity of Smooth Backpropagation to train silent neurons, we trained a 2-2-1 quantron network whose initial parameter values kept the hidden neurons inactive for all input patterns. Those values are given in table 3.3 and the corresponding values of $\max[A(t)]$ for the hidden neurons are given in table 3.4. After 1000 iterations, we find that one of the hidden neurons learned to solve the XOR by itself despite being previously silent. The parameters of the network after learning are given in table 3.5 and the corresponding values of $\max[A(t)]$ for the hidden neurons are given in table 3.6.

Comparison with Spikeprop

We now compare our results for Smooth Backpropagation to existing results for the Spikeprop algorithm. Bohte *et al.* (2002) showed that a network of 5 hidden neurons using 320 weight parameters could be trained via Spikeprop to solve a temporal version of the XOR problem. The best result they obtained is a convergence rate of 100% (10 convergent runs out of 10) using a complicated weight initialization method and particular values for hyperparameters of the network. Schrauwen & Van Campenhout (2004) extended Spikeprop to train other parameters in the network. Using a simple random weight initialization method, they obtained a convergence rate of 90% for a network of 5 hidden neurons (with 40 weight parameters and 86 other trainable parameters), and a convergence rate of 60% for a network of 3 hidden neurons (with 24 weight parameters and 54 other trainable parameters). The best comparable results for Smooth Backpropagation are contained in the "Maximum" column of table 3.2 with exact evaluation. Table 3.7 merges all these results.

We find that Smooth Backpropagation and Spikeprop achieve similar convergence rates. However, the networks trained by Smooth Backpropagation need fewer parameters to solve the XOR problem. Furthermore, the networks trained by Spikeprop use an additional input to represent a reference time. Without this third input, they cannot solve the XOR problem (Sporea & Grüning, 2011). This workaround is unnecessary in quantron networks.

Table 3.3: Initial parameter values.

Neuron	# 1	# 2	# 3
Layer	Hidden	Hidden	Output
w_1	0.4	0.2	0.8
w_2	0.3	0.5	0.7
s_1	0.5	1.4	2.0
s_2	1.5	1.2	3.0
θ_1	1.0	1.5	2.0
θ_2	2.0	1.0	3.0
Γ	1.0	1.0	1.0

Table 3.4: Exact value of $\max[A(t)]$ for hidden neurons with initial parameter values. Both neurons are silent for all input patterns.

Input pattern	Neuron # 1	Neuron # 2
$(1,1)$	0.9000	0.9381
$(1,2)$	0.7000	0.7714
$(2,1)$	0.9000	0.8238
$(2,2)$	0.7000	0.6143

Table 3.5: Final parameter values.

Neuron	# 1	# 2	# 3
w_1	1.7314	1.5995	2.0000
w_2	-1.6738	1.5868	1.1704
s_1	1.1983	1.1852	2.2547
s_2	1.2203	1.2002	1.5533
θ_1	1.4997	2.2006	2.1530
θ_2	1.4847	1.1897	2.8470
Γ	0.9851	2.3754	1.9978

Table 3.6: Exact value of $\max[A(t)]$ for hidden neurons with final parameter values. Maxima higher than the corresponding Γ value are in bold.

Input pattern	Neuron # 1	Neuron # 2
$(1,1)$	0.0612	**4.2101**
$(1,2)$	**1.7001**	**3.6808**
$(2,1)$	**1.7314**	**3.7103**
$(2,2)$	0.0672	2.1289

Table 3.7: Comparison of convergence rate for spiking neural networks trained with Spikeprop (Bohte *et al.*, 2002) and Extended Spikeprop (Schrauwen & Van Campenhout, 2004), and for quantron networks trained via Smooth Backpropagation. The convergence rate is based on 10 runs for Spikeprop and for each Smooth Backpropagation architecture. The number of runs for Extended Spikeprop is unknown.

Algorithm	Network	# parameters (weights/others/total)	Convergence rate (%)
Spikeprop	3-5-1	320/0/320	100
Extended Spikeprop	3-3-1	24/54/78	60
	3-5-1	40/86/126	90
Smooth Backpropagation	2-1	2/5/7	90
	2-2-1	6/15/21	90
	2-3-1	9/22/31	80
	2-4-1	12/29/41	90
	2-5-1	15/36/51	100

3.5.4 Other classification problems

In a second experiment, we test the Smooth Backpropagation algorithm on two other classification problems, both illustrated in figure 3.8.

Here again, a desired value of 1 (respectively 0) corresponds to an active (respectively inactive) state for the output neuron of the network. The first problem has a classification boundary with six linear pieces that is reminiscent of a triangular waveform, and the second problem has a boundary with six linear pieces that delimit three isolated points of desired value 0. To achieve perfect classification for both these problems, a perceptron network would need as many hidden neurons as linear pieces in the boundary, and at least another neuron with six inputs to form the boundary. Thus, using neurons with one weight per input and one additional bias weight, the network would require at least 25 weights.

To solve these problems, we use the 2-2-1 quantron network (with 21 parameters) and the 2-5-1 quantron network (with 51 parameters). We use $(N, N') = (5, 2)$. For each problem and each network, we generate 10 network instances with initial parameters chosen randomly in the same way as for the XOR problem, with the following exception : for hidden neurons $2, \ldots, Q$, we choose w_1 in $[0, 1]$ and w_2 in $[-0.5, 1]$. We start with the learning rate $\nu = 1/4096$ and execute the learning algorithm for consecutive blocks of 500 iterations until either the smooth evaluation of the network shows that 100% of patterns are classified correctly, or 25000 iterations have passed. When the algorithm ends, we proceed with the exact evaluation of the network.

For the 2-2-1 network trained on the triangular waveform problem, Smooth Backpropaga-

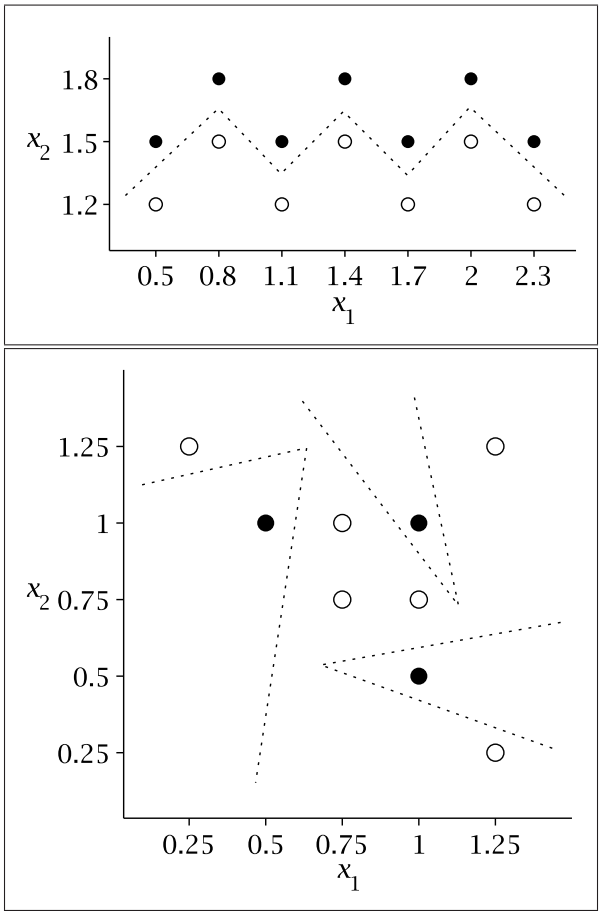

Figure 3.8 Triangular waveform problem (top) and isolated points problem (bottom). White dots (respectively black dots) represent inputs with a desired value of 1 (respectively 0). The dashed lines represent piecewise linear classification boundaries.

tion converged under 25000 iterations in 7 runs (out of 10), with an average of 5214 iterations needed to reach perfect classification. We obtained an average classification accuracy (the number of correctly classified patterns divided by the total number of patterns) of 93.6% for the smooth evaluation and 69.3% for the exact evaluation. For the 2-5-1 network, the algorithm converged in 5 runs (out of 10), with an average of 2600 iterations. We obtained an average classification accuracy of 92.1% for the smooth evaluation and 69.3% for the exact evaluation.

For the 2-2-1 network trained on the isolated points problem, Smooth Backpropagation converged under 25000 iterations in 6 runs (out of 10), with an average of 7500 iterations needed to reach perfect classification. We obtained an average classification accuracy of 90.0% for the smooth evaluation and 70.0% for the exact evaluation. For the 2-5-1 network, the algorithm converged in 9 runs (out of 10), with an average of 5167 iterations. We obtained an average classification accuracy of 98.9% for the smooth evaluation and 65.6% for the exact

evaluation.

These results indicate that Smooth Backpropagation can train quantron networks to solve difficult classification problems with less parameters than a perceptron network. However, this is only valid for the smooth evaluation of the network's output. Letting the algorithm execute for a longer time could improve the classification accuracy for the exact evaluation of the network's output, but improvements to the approximation method must also be considered. Finally, we note that using more hidden neurons improved the performance of the algorithm on the isolated points problem, but not on the triangular waveform problem.

3.5.5 Approximation accuracy

To illustrate the accuracy of our approximation, we propose a final experiment where we compute t^* and o_1 for a quantron with two inputs. We generate 200 sets of random parameter values in the parameter space defined by the constraints of section 3.5.2, with input values x_1 and x_2 each selected randomly in $[0.5, 2.5]$. We use $T = 30$ and successively set λ to $2, 4, 6, 8$.

Figure 3.9 presents scatter plots that illustrate the result of this experiment. We obtained 64 instances where the quantron was active and 136 instances where it was inactive. Where the quantron was active, points seem to follow a straight line. However, the presence of many outliers indicates poor approximation accuracy near the discontinuities of t^*, which is expected. The vertical strip of points in each plot corresponds to the instances where the quantron was inactive.

To complement the visual interpretation of figure 3.9, we give in table 3.8 the root mean square error (RMSE) and the mean absolute error (MAE) for the 64 instances where the quantron was active. We have

$$\text{RMSE} = \sqrt{\frac{1}{64}\sum_{k=1}^{64} e_k^2},$$

$$\text{MAE} = \frac{1}{64}\sum_{k=1}^{64} |e_k|,$$

where e_k is the error between the smooth and exact evaluation for the kth point. These errors measure the closeness of t^* and o_1, and help to compare the scatter plots.

When λ is increased, the scatter plots suggest that the approximation improves, but some outliers seem to drift away and affect the RMSE and MAE. Indeed, based only on the RMSE, we might conclude that the approximation is better with $\lambda = 2$. With the MAE, the approximation is slightly better with $\lambda = 8$. Overall, these results illustrate the potential of our approximation method and help to explain the performance of Smooth Backpropagation

and its limitations. We could ameliorate the approximation accuracy by augmenting the value of λ, but to do this we must circumvent the problem of numerical overflow mentioned in section 3.3.

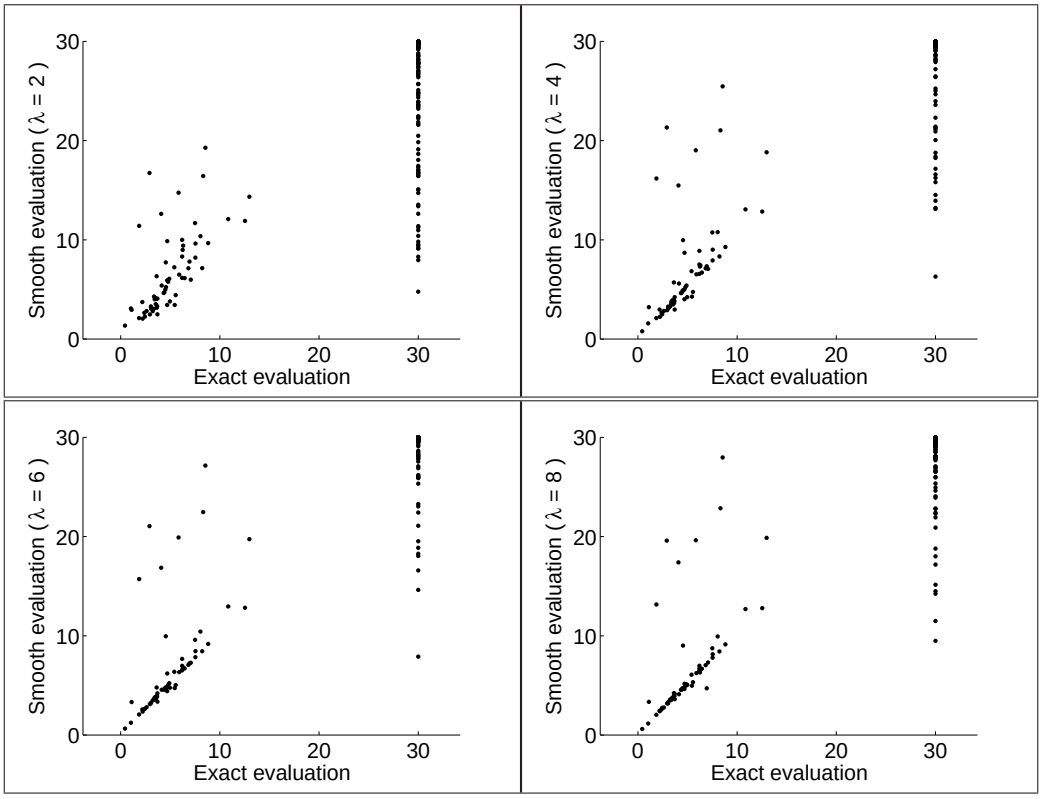

Figure 3.9 Scatter plots of smooth evaluation (o_1) against exact evaluation (t^*) for $\lambda = 2$ (top left), $\lambda = 4$ (top right), $\lambda = 6$ (bottom left) and $\lambda = 8$ (bottom right). Each plot has 200 points, 64 of which correspond to instances where the quantron was active.

Table 3.8: RMSE and MAE for the scatter plots of figure 3.9 (based on 64 points).

λ	RMSE	MAE
2	3.4571	1.9958
4	4.7311	2.1891
6	4.8991	2.0681
8	4.7656	1.9258

3.6 Discussion

Our experiments show that our approximation method successfully enabled the use of the backpropagation algorithm to train a network of quantrons. In particular, we solved the XOR problem with a single neuron, and with networks having two to five hidden neurons. Even though it was previously shown that a single quantron could solve the XOR problem with less parameters than a perceptron network (Labib, 1999), no systematic method to actually train the quantron to do so had been proposed. Furthermore, the solution often remained valid when using the exact output of the network. When comparing Smooth Backpropagation to Spikeprop, we found that quantron networks needed fewer parameters than spiking neural networks to solve the XOR problem with a high convergence rate. This result suggests that quantrons might have a better generalization capability than the spiking neuron model of Spikeprop. We mention that other types of spiking neurons, such as the spiking perceptron, can solve a version of the XOR problem in polar coordinates with even less parameters (Rowcliffe *et al.*, 2006; Xiang *et al.*, 2010). However, these models are based on statistical characteristics of interspike intervals which are incompatible with the fast computation of time-to-first-spike encoding (Maass, 1997).

We also solved harder nonlinear classification problems successfully, but only when performing the classification with the smooth output of the network. The exact output failed to solve these problems. It seems that the error between our approximation method and the exact value of t^* is sufficient to alter the behavior of the network. To reduce the impact of the approximation error, we could slowly increase the precision hyper-parameter λ during learning. This requires a careful analysis of the formulas of section 3.4, since errors in floating-point arithmetic could occur when using exponentials of large numbers.

The silent neuron problem is well known to researchers using the Spikeprop algorithm to train networks of spiking neurons. We successfully avoided this problem in a way similar to the way smooth transfer functions were used in learning algorithms for networks of perceptrons (Rumelhart *et al.*, 1986).

Finally, our work can be adapted to train networks of quantrons with realistic post-synaptic potentials. Indeed, we could approximate the function given in equation (3.2) by a continuous piecewise linear function with more pieces than a simple triangular function. As long as the derivative of this function exists for almost all t (for fixed values of inputs and parameters), theorem 3.1 still applies.

3.7 Conclusion

In this paper, we proposed an approximation method which enabled error backpropagation in a network of quantrons. Starting from a simple surrogate model, we obtained smooth neuron outputs which can be differentiated with regard to inputs and parameters. The outputs and their derivatives were expressed using integrals which can be computed explicitly in closed-form. In our experiments, we trained networks of quantrons to successfully solve three classification problems with less parameters than networks of perceptrons or spiking neurons. The most important contribution of our work is the experimental confirmation that smooth approximation helps avoid the silent neuron problem.

In future work, we plan to study heuristics to improve our approximation method, such as rules for the progressive augmentation of the precision hyper-parameter during learning. To do so, we must thoroughly analyze the numerical behavior of the integrals involved in gradient computations. Finally, we aim to train networks of quantrons with biologically realistic post-synaptic potentials by using a better surrogate model.

En résumé : Dans ce troisième et dernier article, nous avons proposé une approximation analytique du temps d'activation qui admet l'apprentissage d'un réseau de quantrons par rétropropagation lisse de l'erreur. Les résultats obtenus montrent que les neurones qui resteraient silencieux si l'on effectuait un calcul exact de leur réponse peuvent malgré tout être entraînés par un calcul approximatif. De plus, nous avons résolu des problèmes de classification avec moins de paramètres qu'il n'en faut pour des réseaux de neurones impulsionnels et des réseaux de perceptrons. Ce résultat indique que les quantrons pourraient avoir une meilleure capacité de généralisation que ces autres modèles. Cependant, la rétropropagation lisse doit être améliorée pour permettre une convergence de la réponse approximative des neurones vers leur réponse exacte en cours d'apprentissage. Finalement, nous prévoyons poursuivre les travaux sur le quantron en adaptant notre méthode d'approximation à des potentiels post-synaptiques biologiquement réalistes.

CHAPITRE 4

DISCUSSION GÉNÉRALE

Dans ce chapitre, nous établissons d'abord la synthèse de nos trois articles en évaluant l'atteinte de nos objectifs et en résumant les résultats intéressants obtenus. Ensuite, nous soulignons l'apport de cette recherche à l'étude du quantron et des réseaux de neurones biologiquement réalistes. Enfin, nous précisons les limites actuelles et les extensions possibles de nos algorithmes d'apprentissage.

4.1 Synthèse des articles

Rappelons l'essence de notre thèse : l'utilisation d'expressions analytiques pour le maximum de la fonction d'activation, pour la fonction d'activation elle-même et pour le temps d'activation permet de développer de nouveaux algorithmes d'apprentissage pour le quantron et pour un réseau de quantrons. De toute évidence, nos résultats soutiennent cet énoncé. Dans notre premier article, nous avons amélioré une méthode de recherche directe grâce à une approximation analytique du maximum. Dans notre deuxième article, nous avons obtenu des algorithmes donnant des solutions presque optimales pour une configuration particulière d'un quantron, et ce à l'aide d'une forme analytique de la fonction d'activation. Enfin, dans notre troisième article, nous avons proposé une approximation analytique du temps d'activation qui nous a permis d'entraîner un réseau de quantrons en évitant le problème des neurones silencieux, reconnu dans la littérature sur Spikeprop. Ainsi, notre approche analytique nous a indéniablement mené à des résultats originaux et innovateurs.

Nous mentionnons maintenant d'autres résultats connexes qui témoignent de la profondeur de notre recherche.

Dans notre premier article, nous avons illustré des frontières de décision basées sur l'activation du quantron et du neurone à réponse impulsionnelle pour différencier ces modèles. Les frontières du quantron présentent des contours plus intéressants pour la classification. De plus, nous avons comparé trois quantrons à trois perceptrons utilisés comme séparateurs des trois classes du problème IRIS. Dans cette expérience, les deux modèles ont fourni des résultats similaires, avec un avantage pour le perceptron. Ceci témoigne de la difficulté de faire mieux qu'un classificateur linéaire face à un problème simple.

Dans notre deuxième article, pour un quantron à potentiels en forme de rampe, nous avons obtenu un algorithme d'apprentissage qui ne nécessite aucune modification itérative des para-

mètres. Leur valeur est déterminée analytiquement par l'inversion d'un système d'équations.

Dans notre troisième article, nous avons montré qu'un seul quantron peut apprendre le problème du XOR par descente du gradient. Toujours pour le XOR, nous avons obtenu des taux de convergence pour des réseaux de quantrons entraînés par rétropropagation lisse qui sont similaires aux taux obtenus pour des réseaux de neurones à réponse impulsionnelle entraînés par Spikeprop, tout en nécessitant moins de paramètres. Enfin, nous avons illustré l'existence de problèmes de classification pour lesquels un réseau de quantrons, entraîné par rétropropagation lisse, atteint une solution optimale avec moins de paramètres qu'il n'en faudrait pour un réseau de perceptrons.

4.2 Apport de la recherche

Notre recherche constitue la première réussite convaincante d'apprentissage d'un quantron et d'un réseau de quantrons. De plus, nous avons montré l'intéret de notre approche analytique pour l'apprentissage supervisé des réseaux de neurones impulsionnels en contournant les problèmes de discontinuité de la réponse du neurone et de perturbation de la surface d'erreur causée par le calcul numérique. Enfin, nous fournissons une nouvelle façon d'éviter le problème des neurones silencieux.

Bien que nous ayons utilisé des fonctions simples pour représenter les potentiels post-synaptiques, nos résultats ont un intéret pratique pour l'étude des réseaux de neurones biologiquement réalistes. En effet, nous avons montré que l'analyse mathématique du quantron est grandement facilitée par la simplification des potentiels post-synaptiques. De manière générale, cette approche permet d'étudier séparément l'effet de la forme des potentiels et de leur mode d'interaction sur les capacités des neurones et sur le développement d'un algorithme d'apprentissage. En outre, des fonctions simples peuvent servir à la construction d'un potentiel fidèle au réalisme biologique. Dans le cas de notre troisième article, nous avons souligné la possibilité d'utiliser un potentiel linéaire par morceaux tout en conservant la nature lisse de notre approximation du temps d'activation.

4.3 Limites et extensions

Bien que nos résultats illustrent de manière satisfaisante la faisabilité de l'apprentissage du quantron, plusieurs aspects de nos algorithmes n'ont pas été approfondis. En premier lieu, nous soulignons le travail à accomplir en vue de l'amélioration du temps de calcul de nos algorithmes et de la qualité de nos approximations, ainsi qu'à la détermination de la capacité de classification et de généralisation du quantron. En second lieu, nous examinons les interactions possibles entre les différentes méthodes utilisées dans nos articles.

D'abord, nous portons attention au temps de calcul de nos algorithmes. Dans notre premier article, l'apprentissage est basé sur des fonctions déjà programmées et optimisées, de sorte que l'amélioration du temps de calcul passe par un choix judicieux du point de départ de l'algorithme. Ainsi, nous devons établir des règles heuristiques d'initialisation des paramètres du quantron. Dans notre deuxième article, le temps de calcul ne pose pas de problème. En particulier, l'algorithme d'apprentissage pour des potentiels post-synaptiques avec un profil en rampe calcule directement les valeurs des paramètres au lieu de procéder itérativement. Dans notre troisième article, l'amélioration du temps de calcul de l'algorithme de rétropropagation lisse peut se faire en adoptant des techniques de recherche plus efficaces que la simple descente du gradient, ainsi qu'en utilisant des règles heuristiques d'initialisation des paramètres. Nous remarquons que la complexité des expressions analytiques utilisées dans nos algorithmes d'apprentissage pose une difficulté importante à la recherche de telles règles. Pour contourner cet obstacle, il faut d'abord envisager l'étude de configurations particulières du quantron. Par exemple, l'utilisation d'une fonction d'activation qui est monotone croissante jusqu'à l'atteinte du seuil est prometteuse.

Une autre façon d'améliorer le temps de calcul de nos algorithmes est d'utiliser une fonction d'erreur appropriée. Dans nos travaux, nous avons employé différentes fonctions d'erreur : une fonction d'erreur de type « charnière » (*hinge error*) dans le premier article, une fonction d'erreur des moindres carrés (*least squares error*) dans le deuxième article, et une fonction d'erreur d'entropie croisée (*cross-entropy error*) dans le troisième article. Nous entrevoyons l'opportunité d'une étude comparative de ces différentes fonctions d'erreur et de la recherche d'une nouvelle fonction d'erreur adaptée à la réponse du quantron.

Ensuite, nous nous intéressons à la qualité de nos approximations. Dans notre premier article, la substitution des signaux d'entrée du quantron par des fonctions quadratiques peut être améliorée. On remarque que les sommes temporelles d'une quantité importante de potentiels rectangulaires a une forme de trapèze : on observe d'abord une rampe ascendante, ensuite un plateau, et enfin une rampe descendante. L'approximation du maximum pourrait être modifiée en conséquence. Par contre, il n'y aurait plus de relation directe entre le maximum de la fonction d'activation et les maximums des trains de potentiels. Dans notre troisième article, on remarque parfois une différence importante entre les réponses du réseau obtenues par calcul approximatif et par calcul exact. Pour réduire cette différence, nous proposons d'augmenter la valeur de l'hyper-paramètre λ pour rapprocher le temps d'activation approximatif de sa valeur exacte. Par contre, ceci risque d'occasionner des erreurs d'arithmétique en virgule flottante. D'autres erreurs peuvent aussi survenir dans le calcul d'intégrales qui détermine le gradient local d'un neurone. Donc, il faudrait étudier attentivement les opérations de calcul de la rétropropagation lisse sur ordinateur, et les réorganiser au besoin pour

assurer l'exactitude des calculs. Notons que les résultats obtenus dans notre expérimentation ne semblent pas avoir été affectés outre mesure par les erreurs d'arithmétique, car nous avons observé une convergence satisfaisante de l'algorithme d'apprentissage sous différentes conditions.

Enfin, nous cherchons à évaluer les capacités de classification et de généralisation du quantron et des réseaux de quantrons. Nos résultats à ce sujet sont de nature exploratoire. Dans notre premier article, la comparaison du quantron au neurone impulsionnel laisse croire que le quantron possède une meilleure capacité de classification. Cependant, la comparaison des quantrons aux perceptrons indique une performance similaire dans la résolution du problème IRIS. Dans notre troisième article, l'étude du problème du OU-exclusif (XOR) permet de croire que le quantron dispose aussi d'une meilleure capacité de généralisation.

Pour s'assurer de la validité de l'hypothèse que le quantron possède de meilleures capacités de classification et de généralisation que les neurones impulsionnels, il est impératif de tester nos algorithmes d'apprentissage sur une plus large sélection de problèmes de classification. De plus, l'utilisation d'un même algorithme d'apprentissage adapté aux deux modèles est essentielle pour que la comparaison soit effectuée de manière équitable. En considérant les différents problèmes de Spikeprop soulevés dans la littérature, il serait pertinent d'adapter l'algorithme de rétropropagation lisse aux réseaux de neurones à réponse impulsionnelle.

Pour terminer, nous relevons deux interactions possibles entre les méthodes utilisées dans nos articles, ajoutant ainsi aux extensions potentielles de nos travaux.

Premièrement, la forme quadratique de la fonction d'activation obtenue dans notre premier article permet le calcul du temps d'activation à l'aide des racines d'une équation quadratique. Cette approximation, moins complexe à calculer que celle du troisième article, pourrait faciliter l'apprentissage des réseaux de quantrons. Ainsi, nous envisageons l'intégration de cette autre approximation du temps d'activation à une variante de la rétropropagation lisse. Pour tenir compte des situations où la fonction quadratique n'atteint pas le seuil, il est possible de substituer le temps d'atteinte du maximum au temps d'activation.

Deuxièmement, l'approximation lisse du temps d'activation de notre troisième article se trouve simplifiée lorsqu'elle est appliquée aux expressions analytiques utilisées dans notre deuxième article. En effet, les deux formes de la fonction d'activation sont monotones croissantes jusqu'au temps d'activation, de sorte que le maximum courant $\max_{0 \leq v \leq t}[A(v)]$ est équivalent à $A(t)$ pour $0 \leq t \leq \max(r_1, r_2)$. En principe, cette simplification devrait s'appliquer aisément à la rétropropagation lisse dans le cas des potentiels en rampe.

CONCLUSION

Au cours de cette thèse, nous avons adopté une approche analytique pour développer de nouveaux algorithmes d'apprentissage pour le quantron. Nous avons présenté trois articles qui traitent successivement du maximum de la fonction d'activation, de la fonction d'activation elle-même, puis du temps d'activation. Il s'avère que l'utilisation d'approximations et d'expressions analytiques permet bien d'éviter les discontinuités de la réponse du quantron qui perturbent les algorithmes d'apprentissage basés sur la descente du gradient.

Dans le premier article, nous avons proposé une approximation analytique du maximum de la fonction d'activation pour un quantron possédant des potentiels post-synaptiques rectangulaires. Cette approximation, fondée sur la substitution des signaux des synapses d'entrée par des fonctions quadratiques, permet l'entraînement du quantron par descente du gradient. En appliquant successivement une recherche par descente du gradient et une recherche directe, nous avons amélioré les résultats obtenus par la recherche directe seule. De plus, nous avons différencié le quantron et les neurones à réponse impulsionnelle par leurs frontières de décision. Nous avons aussi comparé trois quantrons à trois perceptrons dans la résolution du problème de classification IRIS.

Dans le deuxième article, nous avons obtenu, pour une configuration spécifique du quantron, une expression analytique pour la fonction d'activation qui relie explicitement les paramètres à la frontière de décision. Les caractéristiques géométriques de cette dernière peuvent être utilisées pour obtenir des algorithmes d'apprentissage efficaces pour des problèmes de classification qui admettent une solution sans erreur. Cette étude des frontières de décision est particulièrement originale, puisque très peu de résultats de ce genre sont présents dans la littérature.

Dans le troisième article, nous avons développé une approximation analytique du temps d'activation basée sur le calcul d'intégrales. Nous avons démontré un théorème qui justifie l'utilisation de cette approximation pour obtenir une version lisse de l'algorithme de rétropropagation. Ensuite, nous avons appliqué la rétropropagation lisse à un réseau de quantrons et montré que les paramètres des neurones silencieux sont mis à jour lors de l'apprentissage.

De toute évidence, l'introduction d'une nouvelle méthode d'approximation lisse est une avancée notable permettant l'apprentissage d'un réseau de quantrons. Ceci a été fait en évitant le problème des neurones silencieux, et ce sans recourir à des modifications heuristiques aux règles de mise à jour itérative des paramètres. Par conséquent, ce résultat inédit constitue une contribution novatrice à l'étude de l'apprentissage supervisé des réseaux de neurones biologiquement réalistes.

Nos travaux sur le quantron ont débouché sur de nouveaux algorithmes d'apprentissage qui se distinguent clairement des méthodes proposées dans la littérature des réseaux de neurones à réponse impulsionnelle. Cependant, nous soulevons quelques limites à ces algorithmes. Les aspects les plus importants à améliorer en vue de leur application à des problèmes de classification basés sur des données réelles sont le temps de calcul, ainsi que la qualité des approximations. De plus, en dépit de résultats encourageants, nous ne pouvons affirmer avec certitude que le quantron permet une meilleure classification et une meilleure généralisation que les neurones à réponse impulsionnelle.

Ainsi, la suite logique de nos travaux consiste en l'utilisation de méthodes d'optimisation plus efficaces et en l'amélioration de nos approximations. De plus, pour confirmer l'avantage des capacités de classification et de généralisation du quantron, nous projetons une étude comparative entre ce dernier et les neurones à réponse impulsionnelle. L'adaptation d'un même algorithme d'apprentissage à chacun de ces modèles assurera la validité de la comparaison. Finalement, nous envisageons l'extension des méthodes utilisées dans nos deux premiers articles aux réseaux de quantrons.

En définitive, les algorithmes que nous avons proposés constituent un pas important vers une utilisation efficace du quantron en reconnaissance de formes et une meilleure compréhension de ses capacités. De plus, nous avons indéniablement démontré la fécondité de l'approche analytique pour l'apprentissage du quantron dans l'ensemble de nos travaux. Le futur de la recherche laisse entrevoir des améliorations potentielles à nos algorithmes qui pourront être développées en suivant à nouveau cette voie.

LISTE DES RÉFÉRENCES

ABIYEV, R., KAYNAK, O. & ONIZ, Y. (2012). Spiking neural networks for identification and control of dynamic plants. *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'12)*. 1030–1035.

BELATRECHE, A., MAGUIRE, L. P. & MCGINNITY, T. M. (2003). An evolutionary strategy for supervised training of biologically plausible neural networks. *Proceedings of the 7th Joint Conference on Information Sciences (JCIS'03)*. 1524–1527.

BIRIKUNDAVYI, S., LABIB, R., TRUNG, T. & ROUSSELLE, J. (2002). Performance of neural networks in daily streamflow forecasting. *Journal of Hydrologic Engineering*, 7, 5, 392–398.

BLAKE, C., KEOGH, E. & MERZ, C. J. (1998). UCI machine learning repository. http ://www.ics.uci.edu/ mlearn/MLRepository.html. Department of Information and Computer Sciences, University of California, irvine, CA.

BLUE, J. L., CANDELA, G. T., GROTHER, P. J., CHELLAPPA, R. & WILSON, C. L. (1993). Evaluation of pattern classifiers for fingerprint and OCR applications. *Pattern Recognition*, 27, 4, 485–501.

BOHTE, S. M., KOK, J. N. & LA POUTRÉ, H. (2000). Spikeprop : backpropagation for networks of spiking neurons. *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'00)*. 419–424.

BOHTE, S. M., KOK, J. N. & LA POUTRÉ, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48, 17–37.

BOOIJ, O. & TAT NGUYEN, H. (2005). A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, 95, 6, 552–558.

BOYD, S. & VANDENBERGHE, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

BRAULT, J.-M., LABIB, R., PERRIER, M. & STUART, P. (2011). Prediction of activated sludge filamentous bulking using ATP data and neural networks. *Canadian Journal of Chemical Engineering*, 89, 4, 901–913.

BRIANE, M. & PAGÈS, G. (2004). *Théorie de l'intégration*. Vuibert, troisième édition.

CLARK, J., KOPRINSKA, I. & POON, J. (2003). A neural network based approach to automated E-mail classification. *Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03)*. 702–705.

CONNOLLY, J.-F. & LABIB, R. (2009). A multiscale scheme for approximating the quantron's discriminating function. *IEEE Transactions on Neural Networks*, 20, 8, 1254–1266.

DASH, M. & LIU, H. (2000). Feature selection for clustering. *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*. 110–121.

DAVIS, B. A., ERDOGMUS, D., RAO, Y. N. & PRINCIPE, J. C. (2003). Supervised synaptic weight adaptation for a spiking neuron. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'03)*. vol. 4, 2558–2562.

DE MONTIGNY, S. & LABIB, R. (2011). Learning algorithms for a specific configuration of the quantron. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'11)*. 567–572.

DELSHAD, E., MOALLEM, P. & MONADJEMI, S. (2010). Spiking neural network learning algorithms : using learning rates adaptation of gradient and momentum steps. *Proceedings of the 5th International Symposium on Telecommunications (IST'10)*. 944–949.

DIEUDONNÉ, J. (1960). *Foundations of Modern Analysis*. Academic Press.

FITZGERALD, J. D. & SHARPEE, T. O. (2009). Maximally informative pairwise interactions in networks. *Physical Review E*, 80, 3, 031914.

FUJITA, M., TAKASE, H., KITA, H. & HAYASHI, T. (2008). Shape of error surfaces in spikeprop. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'08)*. 840–844.

GAMEZ, D., FIDJELAND, A. K. & LAZDINS, E. (2012). iSpike : a spiking neural interface for the iCub robot. *Bioinspiration & Biomimetics*, 7, 2, 025008.

GERSTNER, W. (1995). Time structure of the activity in neural network models. *Physical Review E*, 51, 738–758.

GERSTNER, W., KEMPTER, R., VAN HEMMEN, J. & WAGNER, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 384, 76–78.

GHOSH-DASTIDAR, S. & ADELI, H. (2007). Improved spiking neural networks for EEG classification and epilepsy and seizure detection. *Integrated Computer-Aided Engineering*, 14, 3, 187–212.

GHOSH-DASTIDAR, S. & ADELI, H. (2009a). A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, 22, 10, 1419–1431.

GHOSH-DASTIDAR, S. & ADELI, H. (2009b). Spiking neural networks. *International Journal of Neural Systems*, 19, 4, 295–308.

GÜTIG, R. & SOMPOLINSKY, H. (2006). The tempotron : a neuron that learns spike timing-based decisions. *Nature Neuroscience*, 9, 420–428.

HÄUSLER, C., NAWROT, M. P. & SCHMUKER, M. (2011). A spiking neuron classifier network with a deep architecture inspired by the olfactory system of the honeybee. *Proceedings of the 5th International IEEE EMBS Conference on Neural Engineering (NER'11)*. 198–202.

HAYKIN, S. (1999). *Neural Networks : A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, USA, seconde édition.

HODGKIN, A. & HUXLEY, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117, 500–544.

IGLESIAS, J. & VILLA, A. E. P. (2008). Emergence of preferred firing sequences in large spiking neural networks during simulated neuronal development. *International Journal of Neural Systems*, 18, 4, 267–277.

JAHANGIRIAND, A. & DURAND, D. M. (2011). Phase resetting analysis of high potassium epileptiform activity in ca3 region of the rat hippocampus. *International Journal of Neural Systems*, 21, 2, 127–138.

JOHNSTON, S. P., PRASAD, G., MAGUIRE, L. & MCGINNITY, T. M. (2010). An fpga hardware/software co-design towards evolvable spiking neural networks for robotics application. *International Journal of Neural Systems*, 20, 6, 447–461.

KASIŃSKI, A. & PONULAK, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, 16, 1, 101–113.

LABIB, R. (1999). New single neuron structure for solving nonlinear problems. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*. vol. 1, 617–620.

LABIB, R., AUDETTE, F., FORTIN, A. & ASSADI, R. (2005). Hardware implementation of a new artificial neuron. *International Journal of Neural Systems*, 15, 6, 427–433.

LABIB, R. & DE MONTIGNY, S. (2012). On the learning potential of the approximated quantron. *International Journal of Neural Systems*, 22, 03, 1250010.

LABIB, R. & KHATTAR, K. (2010). MLP bilinear separation. *Neural Computing and Applications*, 19, 2, 305–315.

LAPICQUE, L. (1907). Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *Journal de Physiologie et Pathologie Générale*, 9, 620–635.

LEWIS, R. M. & TORCZON, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal of Optimization*, 10, 3, 917–941.

LIU, J., PEREZ-GONZALEZ, D., REES, A., ERWIN, H. & WERMTER, S. (2009). A biomimetic spiking neural network of the auditory midbrain for mobile robot sound localisation in reverberant environments. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'09)*. 1855–1862.

LUQUE, N. R., GARRIDO, J. A., CARRILLO, R. R., TOLU, S. & ROS, E. (2011). Adaptive cerebellar spiking model embedded in the control loop : Context switching and robustness against noise. *International Journal of Neural Systems*, 21, 5, 385–401.

MAASS, W. (1997). Networks of spiking neurons : The third generation of neural network models. *Neural Networks*, 10, 9, 1659–1671.

MARTINEZ, D., ROCHEL, O. & HUGUES, E. (2006). A biomimetic robot for tracking specific odors in turbulent plumes. *Autonomous Robots*, 20, 3, 185–195.

MCCULLOCH, W. S. & PITTS, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133.

MCKENNOCH, S., LIU, D. & BUSHNELL, L. (2006). Fast modifications of the spikeprop algorithm. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'06)*. 3970–3977.

MOHEMMED, A., SCHLIEBS, S., MATSUDA, S. & KASABOV, N. (2011). Method for training a spiking neuron to associate input-output spike trains. *Proceedings of the12th Engineering Applications of Neural Networks / 7th Artificial Intelligence Applications and Innovations Joint Conferences (EANN/AIAI'11)*. vol. 1, 219–228.

NICHOLS, E., MCDAID, L. J. & SIDDIQUE, N. H. (2010). Case study on a self-organizing spiking neural network for robot navigation. *International Journal of Neural Systems*, 20, 6, 501–508.

PAVLIDIS, N. G., TASOULIS, D. K., PLAGIANAKOS, V. P., , NIKIFORIDIS, G. & VRAHATIS, M. N. (2005). Spiking neural network training using evolutionary algorithms. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'04)*. vol. 3, 2190–2194.

ROSSELLO, J. L., CANALS, V., MORRO, A. & VERD, J. (2009). Chaos-based mixed signal implementation of spiking neurons. *International Journal of Neural Systems*, 19, 6, 465–471.

ROWCLIFFE, P., FENG, J. & BUXTON, H. (2006). Spiking perceptrons. *IEEE Transactions on Neural Networks*, 17, 3, 803–807.

ROWLEY, H. A., BALUJA, S. & KANADE, T. (1996). Neural network-based face detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CPVR'96)*. 203–208.

RUF, B. & SCHMITT, M. (1997). Learning temporally encoded patterns in networks of spiking neurons. *Neural Processing Letters*, 5, 9–18.

RUMELHART, D., HINTON, G. & WILLIAMS, R. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

SANTOS, V. M. L., CARVALHO, F. & DE SOUZA JR, M. B. (2000). Predictive control based on neural networks : an application to a fluid catalytic cracking industrial unit. *Brazilian Journal of Chemical Engineering*, 17, 4-7, 897–906.

SCHLIEBS, S., KASABOV, N. & DEFOIN-PLATEL, M. (2010). On the probabilistic optimization of spiking neural networks. *International Journal of Neural Systems*, 20, 6, 481–500.

SCHRAUWEN, B. & VAN CAMPENHOUT, J. (2004). Extending spikeprop. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'04)*. vol. 1, 471–475.

SCHRAUWEN, B. & VAN CAMPENHOUT, J. (2006). Backpropagation for population-temporal coded spiking neural networks. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'06)*. 1797–1804.

SOLTIC, S. & KASABOV, N. (2010). Knowledge extraction from evolving spiking neural networks with rank order population coding. *International Journal of Neural Systems*, 20, 6, 437–445.

SPOREA, I. & GRÜNING, A. (2011). Reference time in spikeprop. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'11)*. 1090–1092.

STRAIN, T. J., MCDAID, L. J., MCGINNITY, T. M., MAGUIRE, L. P. & SAYERS, H. M. (2010). An STDP training algorithm for a spiking neural network with dynamic threshold neurons. *International Journal of Neural Systems*, 20, 6, 463–480.

TAKASE, H., FUJITA, M., KAWANAKA, H., TSURUOKA, S., KITA, H. & HAYASHI, T. (2009). Obstacle to training spikeprop networks - cause of surges in training process -. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'10)*. 3062–3066.

THIRUVARUDCHELVAN, V. & BOSSOMAIER, T. (2012). Towards realtime stance classification by spiking neural network. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'12)*. 1–8.

THIRUVARUDCHELVAN, V., CRANE, J. & BOSSOMAIER, T. (2013). Analysis of spikeprop convergence with alternative spike response functions. *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence (FOCI'13)*. 98–105.

VIDYBIDA, A. K. (2011). Testing of information condensation in a model reverberating spiking neural network. *International Journal of Neural Systems*, 21, 3, 187–198.

VÁSQUEZ, R. & GARRO, B. (2011). Training spiking neurons by means of particle swarm optimization. *Proceedings of the 2nd International Conference on Advances in Swarm Intelligence (ICSI'11)*. vol. 1, 242–249.

WAKAMATSU, T., TAKASE, H., KAWANAKA, H. & TSURUOKA, S. (2011). A training algorithm for spikeprop improving stability of learning process. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'11)*. 951–955.

WALTZ, R., MORALES, J., NOCEDAL, J. & ORBAN, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming*, 107, 3, 391–408.

WANG, J., BELATRECHE, A., MAGUIRE, L. P. & MCGINNITY, T. M. (2011). A new learning algorithm for adaptive spiking neural networks. *Proceedings of the International Conference on Neural Information Processing (ICONIP'11)*. vol. 1, 461–468.

WU, Q., MCGINNITY, T., MAGUIRE, L., GLACKIN, B. & BELATRECHE, A. (2006). Learning under weight constraints in networks of temporal encoding spiking neurons. *Neurocomputing*, 69, 16-18, 1912–1922.

XIANG, X. Y., OU, H. & DENG, Y. C. (2010). The learning of moment neuronal networks. *Neurocomputing*, 73, 2597–2613.

XIN, J. & EMBRECHTS, M. (2001). Supervised learning with spiking neural networks. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'01)*. vol. 3, 1772–1777.

XU, Y., ZENG, X., HAN, L. & YANG, J. (2013). A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Networks*, 43, 99–113.

YANG, J., YANG, W. & WU, W. (2012). A remark on the error-backpropagation learning algorithm for spiking neural networks. *Applied Mathematics Letters*, 25, 8, 1118–1120.

ANNEXE A

Annexe de l'article 1

For the two-input quantron applied on four toy problems, the following tables A.1 to A.4 present minimum, median, mean and standard deviation statistics for the final error and for the misclassification error measured over 100 random parameter initializations as described in Sec. 5.1.1. In the tables, data rows for the final error with the DS sequence are noted E0(N), where N is the number of potentials per input. For the GS+DS and DS+GS+DS sequences, rows for the final error are respectively noted E1(N) and E2(N). Data rows for the corresponding misclassification error (in %) are noted M0(N), M1(N) and M2(N).

Table A.1: Error statistics for problem 1.

Data	Min	Med	Mean	Std
E0(5)	0	1.57e-3	1.49e-2	1.20e-1
E1(5)	5.03e-4	5.09e-4	7.85e-4	6.64e-4
E2(5)	0	5.10e-4	7.17e-4	5.70e-4
E0(10)	0	1.64e-3	2.81e-1	1.27
E1(10)	0	0	7.60e-4	9.96e-4
E2(10)	0	0	3.51e-4	7.19e-4
E0(15)	5.32e-4	2.16e-3	1.37e-1	7.72e-1
E1(15)	0	0	5.51e-4	7.88e-4
E2(15)	0	0	6.18e-3	4.77e-2
M0(5)	0	23.44	28.03	8.48
M1(5)	20.31	20.31	23.31	6.91
M2(5)	0	7.81	10.98	8.65
M0(10)	0	23.44	26.80	8.21
M1(10)	20.31	39.06	35.56	6.56
M2(10)	0	0	5.47	11.19
M0(15)	7.81	26.56	28.31	6.69
M1(15)	1.56	1.56	9.34	11.95
M2(15)	0	0	8.20	12.23

Table A.2: Error statistics for problem 2.

Data	Min	Med	Mean	Std
E0(5)	0	1.24e-3	3.10e-3	1.55e-2
E1(5)	0	2.80e-3	4.79e-2	6.65e-2
E2(5)	0	2.88e-3	3.95e-2	7.15e-2
E0(10)	0	1.20e-3	2.52e-2	1.30e-1
E1(10)	0	1.38e-4	8.89e-4	1.20e-3
E2(10)	1.02e-4	2.32e-3	1.52e-3	1.36e-3
E0(15)	0	0	3.32e-2	1.79e-1
E1(15)	0	1.35e-5	1.28e-2	1.21e-1
E2(15)	0	2.80e-3	1.59e-3	1.37e-3
M0(5)	0	18.75	23.88	19.11
M1(5)	17.19	25.00	29.25	11.37
M2(5)	0	31.25	25.56	16.00
M0(10)	0	18.75	14.50	16.20
M1(10)	17.19	17.19	25.16	12.23
M2(10)	1.56	28.90	22.83	20.80
M0(15)	0	0	10.13	15.56
M1(15)	15.63	15.63	22.84	12.10
M2(15)	0	43.75	24.75	21.48

Table A.3: Error statistics for problem 3.

Data	Min	Med	Mean	Std
E0(5)	9.00e-4	2.71e-3	2.90e-2	1.27e-1
E1(5)	4.00e-4	9.01e-4	7.78e-4	5.45e-4
E2(5)	4.00e-4	4.07e-4	9.73e-4	8.69e-4
E0(10)	0	2.13e-3	1.61e-1	4.44e-1
E1(10)	0	3.33e-1	2.01e-1	1.63e-1
E2(10)	0	3.33e-1	2.69e-1	1.37e-1
E0(15)	9.00e-4	2.92e-3	2.96e-1	3.74e-1
E1(15)	0	9.57e-2	6.67e-2	4.36e-2
E2(15)	0	2.71e-3	4.44e-2	4.66e-2
M0(5)	12.50	32.81	31.38	11.99
M1(5)	6.25	17.19	14.56	11.46
M2(5)	6.25	6.25	15.13	13.58
M0(10)	0	21.09	23.72	11.33
M1(10)	10.94	10.94	15.63	12.36
M2(10)	0	7.81	12.08	11.68
M0(15)	6.25	14.06	19.86	10.01
M1(15)	12.50	12.50	17.66	13.72
M2(15)	0	6.25	22.27	18.31

Table A.4: Error statistics for problem 4.

Data	Min	Med	Mean	Std
E0(5)	6.02e-4	1.66e-3	6.98e-3	5.33e-2
E1(5)	0	0	3.05e-4	6.33e-4
E2(5)	0	0	2.60e-4	5.99e-4
E0(10)	1.20e-3	1.69e-003	4.10e-2	2.70e-1
E1(10)	1.60e-3	7.60e-003	1.97e-2	1.723e-2
E2(10)	1.60e-3	7.60e-003	1.95e-2	1.74e-2
E0(15)	1.20e-3	1.76e-003	1.36e-1	5.00e-1
E1(15)	0	0	7.08e-3	3.39e-2
E2(15)	0	1.61e-003	7.01e-2	8.55e-2
M0(5)	9.38	25.00	24.58	2.05
M1(5)	10.94	10.94	13.61	5.54
M2(5)	0	0	4.00	9.21
M0(10)	9.38	25.00	24.27	2.45
M1(10)	6.25	9.38	11.47	7.15
M2(10)	1.56	1.56	7.28	10.05
M0(15)	10.94	25.00	23.31	2.93
M1(15)	4.69	6.25	9.27	6.96
M2(15)	0	12.50	10.72	9.53

ANNEXE B

Figure supplémentaire de l'article 1

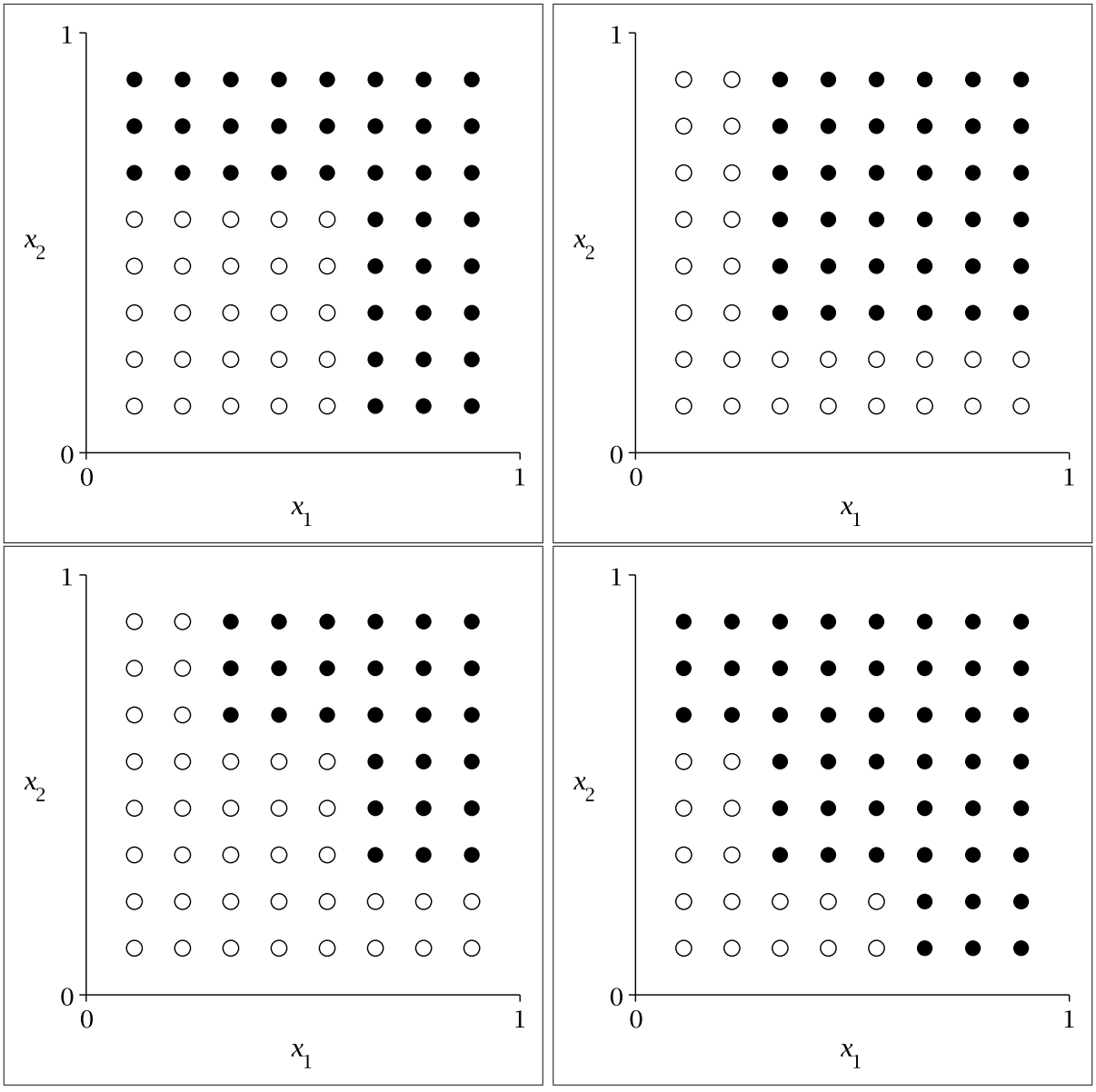

Figure B.1 Problème jouet 1 (en haut, à gauche). Problème jouet 2 (en haut, à droite). Problème jouet 3 (en bas, à gauche). Problème jouet 4 (en bas, à droite).

ANNEXE C

Annexe de l'article 3

In the following tables, we give the complete results of our experiment on the XOR problem. All results are based on 10 runs for each network configuration, starting from random parameter values.

Table C.1: Results for the 2-1 network (single quantron) on the XOR problem.

N	Convergent runs (out of 10)	
	Smooth evaluation	Exact evaluation
2	3	3
3	6	6
4	8	8
5	9	9
6	8	8

Table C.2: Results for the 2-2-1 network on the XOR problem.

(N, N')	Convergent runs (out of 10)	
	Smooth evaluation	Exact evaluation
$(2, 2)$	4	3
$(3, 2)$	7	7
$(4, 2)$	7	5
$(5, 2)$	9	9
$(6, 2)$	8	6
$(2, 3)$	6	3
$(3, 3)$	7	5
$(4, 3)$	8	7
$(5, 3)$	8	8
$(6, 3)$	7	7

Table C.3: Results for the 2-3-1 network on the XOR problem.

(N, N')	Convergent runs (out of 10)	
	Smooth evaluation	Exact evaluation
$(2,2)$	8	5
$(3,2)$	7	4
$(4,2)$	9	7
$(5,2)$	9	8
$(6,2)$	9	7
$(2,3)$	9	3
$(3,3)$	8	4
$(4,3)$	9	7
$(5,3)$	10	8
$(6,3)$	10	5

Table C.4: Results for the 2-4-1 network on the XOR problem.

(N, N')	Convergent runs (out of 10)	
	Smooth evaluation	Exact evaluation
$(2,2)$	7	5
$(3,2)$	10	9
$(4,2)$	10	8
$(5,2)$	10	7
$(6,2)$	10	7
$(2,3)$	8	3
$(3,3)$	9	5
$(4,3)$	10	6
$(5,3)$	10	7
$(6,3)$	9	7

Table C.5: Results for the 2-5-1 network on the XOR problem.

(N, N')	Convergent runs (out of 10)	
	Smooth evaluation	Exact evaluation
$(2,2)$	9	3
$(3,2)$	10	6
$(4,2)$	10	8
$(5,2)$	10	10
$(6,2)$	10	8
$(2,3)$	9	3
$(3,3)$	9	5
$(4,3)$	10	8
$(5,3)$	10	8
$(6,3)$	10	8