FEATURE DISCOVERY USING SNAP-DRIFT NEURAL NETWORKS

Sin Wee Lee, Dominic Palmer-Brown
Innovative Informatics Research Group, School of Computing and Technology
{s.w.lee, d.palmer-brown}@uel.ac.uk

Abstract: This paper introduces an application of Snap-Drift Neural Networks (SDNNs), which employs the complementary concepts of fast, minimalist (snap) learning and slow (drift towards the input pattern) learning, for feature discovery and classification of speech waveforms from non-stammering and stammering speakers. The speech waveforms are drawn from a phonetically annotated corpus, which facilitates phonetic interpretation of the classes of patterns discovered by the SDNN. The results show that SDNN groups the phonetics speech input patterns meaningfully and extracts properties which are common to both non-stammering and stammering speech, as well as distinct features that are common within each of the utterance groups, thus supporting classification. SDNN is also being applied in a virtual learning environment to categorise students' test responses and thereby support individualised feedback.

1. Introduction

Stuttering (stammering) is a highly variable condition which occurs across ages and cultures. There is a lack of consensus in establishing the criteria for a definition. Finding a way of identifying exactly what phonetic characteristics are associated with stammering, as opposed to non-stammering speech, has proved elusive. Perceptual analysis is known to be compromised by its subjectivity (Aylett). In contrast, a correlative data analysis to characterise the acoustic properties of stammering is realisable. There are four classes of sound pressure wave that form the acoustic structure of utterances (Ladefoged, 2001): Periodic 'voice': regular repeating fluctuations produced by vocal fold vibration; Aperiodic 'noise': ongoing irregular fluctuations in voiceless fricatives; Transient 'burst': brief irregular fluctuations as in voiceless plosives; or Silent: no acoustic energy is emitted. The speech sounds used in human languages are made up of combinations of the four categories.

The *snap-drift* learning algorithm first emerged as an attempt to overcome the limitations of ART learning in non-stationary environments where self-organisation needs to take account of periodic or occasional performance feedback. Since then, the *snap-drift* algorithm has proved invaluable for continuous learning in several applications. The *reinforcement* versions (Lee, 2003, 2004) of *snap-drift* are used in the classification of user requests in an active computer network simulation environment whereby the system is able to discover alternative solutions in response to varying performance requirements. Furthermore, the *unsupervised snap-drift* algorithm, without any form of reinforcement, has been used in the analysis and interpretation of data representing interactions between trainee network managers and a simulated network management system (Donelan, 2004). New patterns of the user behaviour were discovered.

The further exploration of *snap-drift*, in the form of a classifier (Lee, 2005) has been used in attempting to discover and recognize phrases extracted from Lancaster Parsed Corpus (LPC) (Garside, 1987). Comparisons carried out between *snap-drift* and MLP with backpropagation, show that the former is faster and just as effective.

This paper describes the further exploration of snap-drift, in unsupervised form, in attempting to discover the defining and unique millisecond features in the speech patterns, which will be used to help understand the language learning of non-stammering and stammering speakers.

2. The Snap-Drift Neural Network (SDNN) Architecture

The modular neural network modified from Performance-guided Adaptive Resonance Theory (P-ART) network, first introduced by Lee & Palmer-Brown (Lee, 2004) is shown in Figure 1.

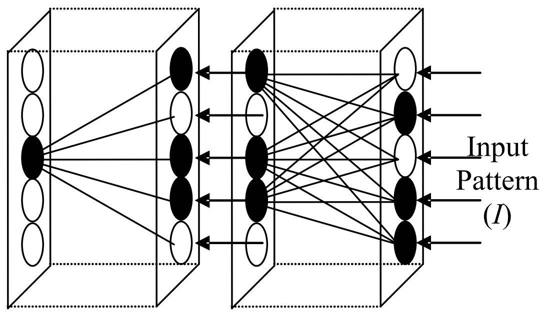

Fig. 1. SDNN Architecture

On presentation of an input pattern at the input layer, dSDNN will learn to group the input patterns according to their general features. In this case, 10 $F2_1$ nodes, whose weight prototypes best match the current input pattern, are used as the input data to the sSDNN module for feature classification. In both of the modules, the standard matching and reset mechanism of ART (Lee, 2003, 2004) is discarded. Instead, in the dSDNN module, the output nodes with the highest net input are *always* accepted as winners. In the sSDNN module, a quality assurance threshold is introduced. If the net input of a sSDNN node is above the threshold, the output node is accepted as the winner, otherwise a new uncommitted output node will be selected as the new winner and initialised with the current input pattern.

In this version of SDNN we introduce *weight re-initialisation*. The main purpose of weight re-initialisation is to enable unused output nodes to be reinstated into the competition for winning nodes. Weight re-initialization is invoked after many epochs since the SDNN must first allow input patterns to settle into their categories. After a duration defined by a certain number of input patterns, called a learning era (an era is a number of epochs), the weights of nodes unused during the preceding era will be reinitialised to enable them to participate again in the competition for the best winning nodes. In effect, re-initialisation is a neuron pruning algorithm. It removes weight vectors that are redundant.

The following is a summary of the steps that occur in SDNN:

Step 1: Initialise parameters: ($\alpha = 1$, $\sigma = 0$), era =2000

Step 2: For each epoch (t)

Test: **Weights re-initialization condition**

For each input pattern

Step 2.1: Find the D (D = 10) winning nodes at $F2_1$ with the largest net input

Step 2.2: Inhibit the $F2_1$ node for weights re-initialization

Step 2.3: Weights of dSDNN adapted according to the alternative learning procedure: (α,σ) becomes Inverse(α,σ) after every successive epoch

Step 3: Process the output pattern of $F2_1$ as input pattern of $F1_2$

Step 3.1: Find the node at $F1_2$ with the largest net input

Step 3.2: Test the threshold condition:

IF (the net input of the node is greater than the threshold)

THEN

Weights of the sSDNN output node adapted according to the alternative learning procedure: (α,σ) becomes Inverse(α,σ) after every successive epoch

ELSE

An uncommitted sSDNN output node is selected and its weights are adapted according to the alternative learning procedure: (α,σ) becomes Inverse(α,σ) after every successive epoch

Weights re-initialization condition:
After 'era' input patterns
IF ($F2_1$ node not used for the past era input presentations)
THEN
Re-initialize the $F2_1$ node with randomly selected input pattern
Inhibit the $F2_1$ node for weights re-initialization for the next era input pattern presentation
ELSE
No action taken.

3. The Snap-Drift Algorithm

The learning algorithm combines a modified form of Adaptive Resonance Theory (*snap*) (Carpenter, 1987) and Learning Vector Quantisation *(drift)* (Kohonen, 1990). In general terms, the snap-drift algorithm can be stated as:

$$Snap\text{-}drift = \alpha(Fast\_Learning\_ART) + \sigma(LVQ) \quad (1)$$

The top-down learning of both of the modules in the neural system is as follows:

$$w_{Ji}^{(new)} = \alpha(I \cap w_{Ji}^{(old)}) + \sigma(w_{Ji}^{(old)} + \beta(I - w_{Ji}^{(old)})) \quad (2)$$

where w_{Ji} = top-down weights vectors; I = binary input vectors, and β = the drift speed constant = 0.5.

In successive learning epochs, the learning is toggled between the two modes of learning. When $\alpha = 1$, fast, minimalist (snap) learning is invoked, causing the top-down weights to reach their new asymptote on each input presentation. (2) is simplified as:

$$w_{Ji}^{(new)} = I \cap w_{Ji}^{(old)} \quad (3)$$

This learns sub-features of patterns. In contrast, when σ = 1, (2) simplifies to:

$$w_{Ji}^{(new)} = w_{Ji}^{(old)} + \beta(I - w_{Ji}^{(old)}) \quad (4)$$

which causes a simple form of clustering at a speed determined by β.

The bottom-up learning of the neural system is a normalised version of the top-down learning:

$$w_{IJ}^{(new)} = w_{Ji}^{(new)} / | w_{Ji}^{(new)} | \quad (5)$$

where $w_{Ji}^{(new)}$ = top-down weights of the network after learning.

In SDNN, as described in section 2, snap-drift is toggled between snap and drift on each successive epoch. The effect of this is to capture the strongest clusters (holistic features), sub-features, and combinations of the two.

4. Simulations

The *snap-drift* algorithm is used for learning and discovering the features embedded in the utterances of two speaker groups, non-stammering and stammering. Before any simulations, pre-processing of the utterances is completed. In this research, each point of a speech utterance waveform collected represents 1 ms of speech data. In this research, in order to analyze and recognise the acoustic properties of the speaker with sufficient precision, each utterance is sampled every 10 points for a total of 1000 points, which represents about 1 second of speech information. This is considered sufficient by a phonetics expert. Figure 2 shows the example of sampled utterance used in the simulations. Each of the sampled waveforms is used to generate a number of input patterns for SDNN. The input patterns are generated using a sliding window of size 100 samples points. The sliding window is shifted to the right by 25 sample points to create a new input. This provides some overlapping of features among the input patterns. Then, each input pattern is converted into a 1400 bit coarse coded binary pattern. 5 utterances are used from 2 speakers, 3 utterances from the non-stammering speaker and 2 from the stammering speaker. Table 1 shows the range and properties of the input set, making the total number of input patterns, 1873 input vectors. These test input patterns are presented in sequence to SDNN. The number of input patterns for each speaker varies because:

1. Each speaker is asked to speak using different types of statements.
2. Non-stammering speaker will produce more fluent speech utterances with shorter or no delay between phrases.
3. Stammering speakers always produce longer utterances due to the delay in the voiceless fricative.

The input patterns, which are also quite noisy, provide a real world test for unsupervised SDNN as a feature discovery and classification system.

For SDNN to act as a viable classifier, and to demonstrate the utility of the features it acquires, it should be able to estimate or predict whether a speaker in a real-time scenario is non-stammering or stammering when a speech utterance is fed into the system. An estimation will be made of how long it takes to be certain that a speaker is non-stammering or stammering.

Table 1. Range and Properties of the input sets

Speaker Group	Total number of inputs
Non-Stammering	256
Stammering	644
Non-Stammering	162
Stammering	467
Non-Stammering	229

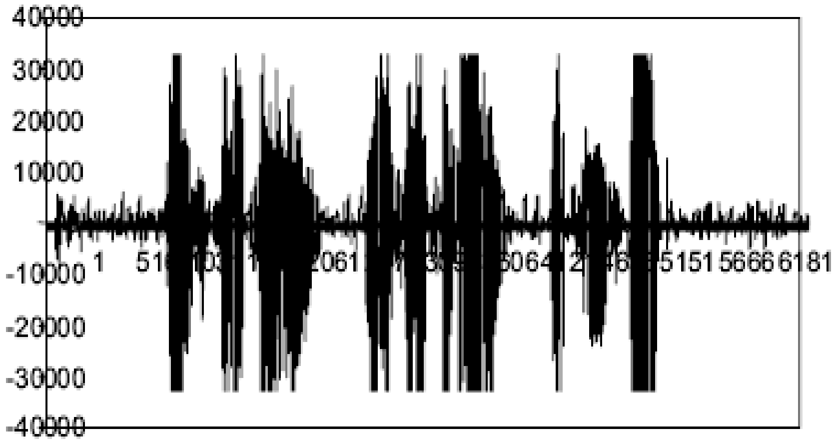

Fig. 2. Example utterance waveform used in simulation

4.1 Results

The results are presented in Table 2 to Table 4; each of the tables shows the example category types formed by the SDNN network with their acoustic properties. The acoustic properties record is obtained from a phonetics expert's annotation of the speech waveform corpus. Each of the sampled sequence of the speech utterance is identified with one or more acoustic properties: *Silent, Periodic, Aperiodic* and *Transient.*

By looking at the tables, it is clear that the SDNN has categorised the input patterns into 3 distinctive types, stammering speech, non-stammering speech, and a category type with a mixture of the two speaker types. The three category types were identified since they corresponded to different non-overlapping sets of sSDNN output nodes.

Figure 3 - 5 show the example input waveforms being grouped into the same category, in this case example category type 1 (Stammering).

By comparing these waveforms, the similarities can be easily identified. In order to understand the learned features of the speech utterances, a comparison of the input patterns of the system and the learned weight templates is performed.

The input patterns received by the SDNN are binary coarse coded representations of the fragments of speech input utterances, such as those shown in Fig. 3 – 5. Each point in the speech input is represented by a 14 bit binary representation. So, the weights learned are the results of processing these binary input patterns. As a means of visualization, the weights learned are thresholded as a first order approximation to produce a binary representation of the weights learned. Then, the 14 bit coarse binary representation of the weights learned are decoded to show the actual waveform features that have been acquired from the original waveforms.

Fig. 6 and Fig. 7 show the weights learned. Although these weights graphs are drawn using approximation for visualization, the figures clearly show that system has learned the features in the input patterns of the categories. In fig. 6 and 7, the graphs show a noisy sinusoid of about 3 Hz.

By comparing with the original waveforms, it has clearly shown that what these waveforms have in common is a sinusoid of approximately 3Hz.

The phonetics expert has identified that these parts of the utterances are often associated with silence or pauses or gaps between words where there is some sound perhaps but no clear articulation. This is indeed known to be the case for stammerers.

Table 2. Accoustic properties of example category type 1 (Stammering)

Input	Speaker group	Silent	Periodic	Aperiodic	Transient
195	Non-stammering	✓		✓	✓
211	Non-stammering	✓			✓
377, 456	Stammering		✓		
432, 68	Stammering		✓		✓
473, 485, 575, 585	Stammering	✓			
570	Stammering	✓	✓		
595	Stammering	✓			✓
609	Stammering	✓		✓	

Table 3. Accoustic properties of example category type 2 (Non-Stammering)

Input	Speaker group	Silent	Periodic	Aperiodic	Transient
21, 34, 3699, 142, 175	Non-stammering		✓	✓	
27, 32, 231, 253	Non-stammering		✓		
38, 187	Non-stammering			✓	
48	Non-stammering				✓
56	Non-stammering	✓			✓
200	Non-stammering	✓			
304	Stammering	✓	✓		✓
310	Stammering		✓	✓	

Table 4. Accoustic properties of example category type 3 (Mixture of both type of speakers)

Input	Speaker group	Silent	Periodic	Aperiodic	Transient
45, 108	Non-stammering		✓	✓	
165	Non-stammering	✓			
131, 135	Non-stammering		✓		✓
204, 123	Non-stammering		✓		
283, 504	Stammering		✓		✓
304, 442	Stammering	✓	✓		✓
615, 565, 370, 457	Stammering		✓		
546, 547	Stammering	✓			

5. Unique Sequences and Classifications

As mentioned, during each learning epoch, the speech utterances are fed into the system in sequence, one speaker utterance at a time. In order to do the analysis and thus determine the time it takes to identify the speaker type, one epoch after convergence is randomly selected. By randomly selecting one sequence of sSDNN winning nodes to start with, the whole epoch is examined to find any repeated occurrences of the sequence.

These repeated occurrences of winning nodes sequences are called *unique sequences* if they are unique to only stammering or non-stammering speakers. Then, the speaker input utterances which caused the *unique sequence*, is examined.

With this method of analysis, the length of unique sequence of winning nodes which system takes to be certain of the speaker group for a particular speech utterance.

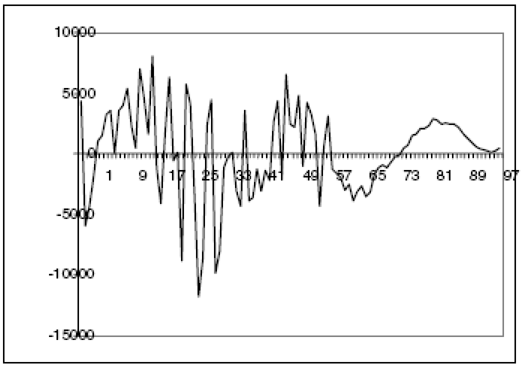

Fig. 3. Example input waveform for category type 1 (Input 377)

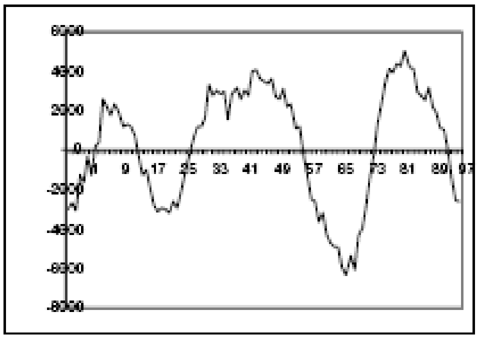

Fig. 5. Example input waveform for category type 1 (Input 456)

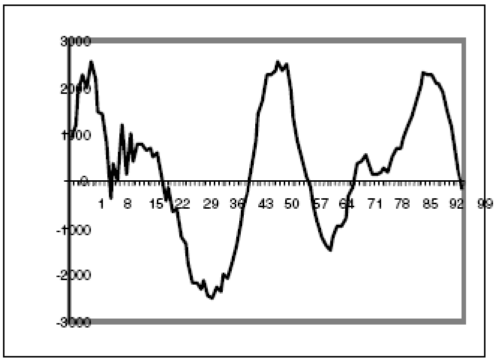

Fig. 4. Example input waveform for category type 1 (Input 595)

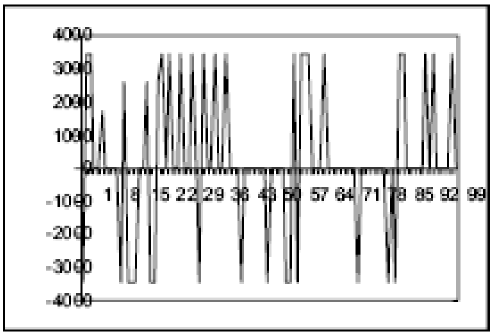

Fig. 6. Example weights learned for category type 1 (Winning node 42)

only occurred in a particular group of speakers, either stammering or non-stammering, will determine the time the system takes to be certain of the speaker group for a particular speech utterance.

These repeated occurrences of winning nodes sequences are called *unique sequences* if they are unique to only stammering or non-stammering speakers. Then, the speaker input utterances which caused the *unique sequence*, is examined. With this method of analysis, the length of unique sequence of winning nodes which only occurred in a particular group of speakers, either stammering or non-stammering, will determine the time the

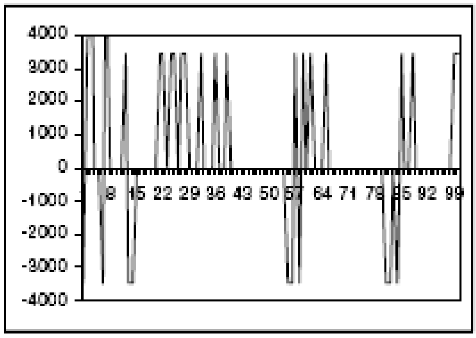

Fig. 7. Example weights learned for category type 1 (Winning node 13)

Table 4 shows the sequence occurrence of winning nodes for non-stammering or stammering group input patterns. The sequences for analysis are randomly selected. In the table, most of the sequences with the length less than 3 tend to have a mixture of occurrence of both types of speaker groups. By increasing the length of the sequence, some form of bias arises. With the sequence length of more than 5 winning nodes, these sequences only occur in one of the speaker types, either non-stammering or stammering. For example, the sequence {45, 52, 43, 19, 65} only exists in the speech input of the stammering speaker. Obviously, this sequence is *unique* to the stammering speaker. By plotting the average ratio of the speaker type over the sequence length, the length of the sequence which can be labelled as unique can be identified. This is illustrated in Fig. 8. In fig. 8, the average ratio of the speaker group for sequence length of 5 and 6 is the lowest. With this number of randomly selected sequences for consideration, it confidently shows that input patterns for particular speaker groups can be identified when a unique sequence, with the length of 5 winning nodes is used for analysis.

By identifying this *unique sequence*; we mean SDNN is capable of identifying the speaker group of input patterns after system convergence is achieved. As mentioned in section IV, each input pattern roughly represents about 1 second of speech information, thus, SDNN is capable of distinguishing the type of speaker by analysis of about 4 seconds of speech, which is analogous to the a person identifying a speaker as stammering or non-stammering after hearing several words. Since not all words are stammered by stammerers, this figure is also of the order of 5 seconds of speech for humans.

Thus, SDNN has shown the capability of a classifier, in this case, categorizing the input patterns according to their features and classifying and estimating the time it takes to be certain that a speaker is non-stammering or stammering by using unique sequences of sSDNN winning nodes.

Table 5. Randomly selected sequence occurrence of winning nodes for non-stammering /stammering group input patterns

Sequence	No. of Occurrences	Non-Stammering	Stammering
63,65	22	13	9
1,36, 31	15	9	6
7,5,3	19	11	8
45,52,43	15	6	9
12,23,34,34	13	6	7
7,5,3,54,39	4	4	0
42,34,46,10,59	3	3	0
45,52,43,19,65	7	0	7
7,7,2,6,49	3	3	0
39,36,56,16,32	4	0	4
6,32,40,4,23,58	6	1	5
69,68,56,68,69	3	0	3
54,69,55,11,46,50	3	0	3
11,63,45,37,56,68	4	4	0
6,32,46,23,4,33	2	0	2

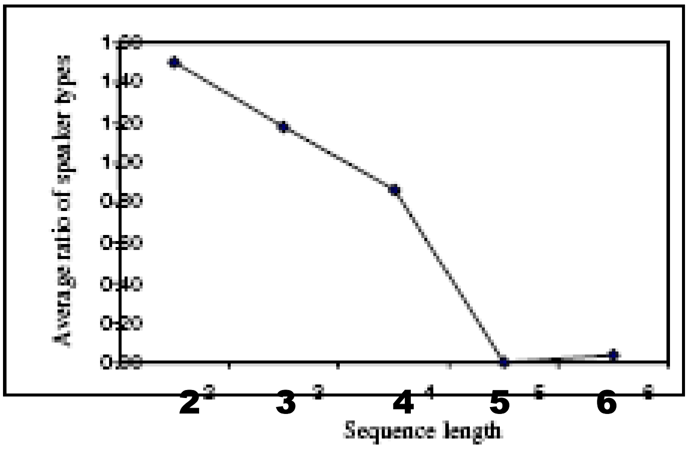

Fig. 8. The average ratio of the speaker type over the length of the winning node sequence

6. SDNN for Diagnostic Feedback in a Virtual Learning Environment

Currently, the version of SDNN described in this paper is being incorporated as an online diagnostic tool for automatic generation of diagnostic feedback for students within an e-learning environment. By discovering the hidden features in the students' responses, such as those to multiple choice questions, including typical error patterns, in multiple choice answers, the SDNN will facilitate automatic diagnosis of deficits in areas of knowledge and thereby select the most appropriate feedback mix from a corpus to guide the student towards a greater understanding of particular concepts. Because the feedback is concept based rather than tied to a given question, the learner can be encouraged to retake the same test and receive different feedback depending on their evolving state of knowledge.

7. Conclusion

This paper presents the new application of feature discovery in phonetics speech using the *snap-drift* algorithm. It also gives the opportunity to test the performance of SDNN without a performance feedback in a purely unsupervised mode. SDNN categorizes the input patterns according to their general and distinct features. By examining the phonetic and waveform properties of the input patterns in each of the categories formed, it has been shown that without any performance feedback, the SDNN modules group the input patterns sensibly and extract properties which are general between non-stammering and stammering speech, as well as distinct features within each of the utterance groups, thus supporting classification.

8. References

Aylett, M., Turk, A.: Vowel Quality in Spontaneous Speech: What makes a good vowel. Proc. of Int. Conf. of Spoken Language Processing. Sydney, Australia.

Carpenter, G. A., Grossberg, S.: A Massively Parallel Architecture for a Self-Organising Neural Pattern Recognition

Machine. Com. Vision, Graphics and Image Proc., Vol. 37 (1987) 54-115.

Donelan, H., Pattinson, C., Palmer-Brown, D., Lee, S. W.: The Analysis of Network Manager's Behaviour using a Self-Organising Neural Networks. Proc. of ESM, Magdeburg, Germany, (2004) 111 – 116.

Garside, R., Leech, G., Varadi, T.: Manual of Information to Accompany the Lancaster Parsed Corpus: Department of English, University of Oslo (1987).

Klatt, D. H.: Review of Text-to-Speech Conversion for English. Online collection (1987)

Kohonen, T.: Improved Versions of Learning Vector Quantization. Proc. of IJCNN, Vol. 1 (1990) 545 – 550.

Ladefoged, P.: A Course in Phonetics. 4th ed., Boston, Heinle & Heinle (2001)

Lee, S. W., Palmer-Brown, D., Roadknight, C. M.: Performance-guided Neural Network for Rapidly Self-Organising Active Network Management. Neurocomputing, Vol. 61C (2004) 5 – 20.

Lee, S. W., Palmer-Brown, D., Tepper, J., Roadknight, C. M.: Snap-Drift: Real-time Performance- guided Learning. Proc. of IJCNN, Portland, Oregon, Vo1. 2 (2003) 1412 – 1416.

Lee, S. W., Palmer-Brown, D., Roadknight, C. M.: Reinforced Snap-Drift Learning for Proxylet Selection in Active Computer Networks. Proc. of IJCNN, Budapest, Hungary, Vo1. 2 (2004) 1545 – 1550.

Lee, S. W., Palmer-Brown, D.: Phrase Recognition using Snap-Drift Learning Algorithm. Proc. of IJCNN, Montreal, Canada (2005).