Finding Patterns in Timed Data with Spike Timing Dependent Plasticity

by

Alexandre Oliveira

B.S., Electrical Engineering and Computer Science Massachusetts Institute of Technology, 2010

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology

February, 2012

Author: [illegible]
Department of Electrical Engineering and Computer Science
February 1, 2012

Certified by: [illegible]
Dr. Patrick H. Winston, Thesis Supervisor
February 1, 2012

Certified by: [illegible]
Victor Chan, Thesis Co-Supervisor
February 1, 2012

Accepted by: [illegible]
Prof. Dennis M. Freeman, Chairman, Masters of Engineering Thesis Committee

Finding Patterns in Timed Data with Spike Timing Dependent Plasticity

by

Alexandre Oliveira

B.S., Electrical Engineering and Computer Science Massachusetts Institute of Technology, 2010

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

Abstract

My research focuses on finding patterns in events – in sequences of data that happen over time. It takes inspiration from a neuroscience phenomena believed to be deeply involved in learning. I propose a machine learning algorithm that finds patterns in timed data and is highly robust to noise and missing data. It can find both coincident relationships, where two events tend to happen together; as well as causal relationships, where one event appears to be caused by another. I analyze stock price information using this algorithm and strong relationships are found between companies within the same industry. In particular, I worked with 12 stocks taken from the banking, information technology, healthcare, and oil industries. The relationships are almost exclusively coincidental, rather than causal.

Contents

List of Figures

List of Tables

1 Introduction

The field of machine learning has had a huge impact on our lives over the past decade. Machine learning has allowed patterns that traditionally were visible only to humans to be found automatically. This means that certain information can now be extracted at speeds never before possible. Take webpage ranking for example. If the quality of web pages were not decided automatically, web search would be too slow for all practical purposes. Besides page ranking, machine learning has also made big advances in automating speech recognition, and facial detection and recognition.

While advances in these fields have certainly been very impactful, it's arguable that one of the most interesting kinds of data has been left out of the picture – in particular data where precise timing matters. Webpage rank, facial detection, and recognition, are all tasks that have no time component. Even speech, which is essentially a sequence of sounds over time, has been traditionally treated using Bayesian networks, which disregard timing and instead preserve only the ordering.

In real life, the things we see, hear, and feel every second are all pieces of data happening at particular times and in a particular order. With the greatest of ease, we as humans can see patterns in all this data – we can quickly get a feeling for the relationship between things. We can even make conclusions (or assumptions) about what events are causing what.

Machine learning has not focused on finding relationships between events in time. However it is a crucial piece of the vision to emulate humans' pattern-finding abilities. Even for problems that seem to be time-independent like object recognition, there is evidence that the human brain translates this task into a pattern matching problem of events over time[1]. This makes sense given that neurons transmit information through spikes whose timing is known to be crucial[1].

In this thesis, I took inspiration from a neuroscience phenomenon that is believed to be involved in learning called Spike Timing Dependent Plasticity (STDP). I implemented its functional aspects to capture patterns in timed data. The goal is was approach the problem from a machine learning perspective, rather than to simulate the biological aspects.

1.1 Drawbacks of Current Approaches

Typical approaches to solve problems such as speech, gesture recognition, and other kind of time-based problems use Hidden Markov Models (HMM) and Bayesian networks (more specifically Dynamic Bayesian Networks). These type of networks can detect relationships between events and can be trained to understand the conditional probabilities of an event happening.

While Bayesian networks are great at discovering conditional probabilities in data, they are not practical for discovering *unknown* relationships in data. This is because, in order for a Bayesian network to be useful, the researcher must typically provide an initial graph containing all the potential relationships between events. If the researcher provides a complete graph as the initial graph, this results in a combinatorial explosion – Bayesian networks are inefficient at dealing with graphs with a lot of edges. Furthermore, the computation of conditional probabilities typically requires complete

[1] Metzner W, Koch C, Wessel R, Gabbiani F (March 1998). "Feature extraction by burst-like spike patterns in multiple sensory maps". *The Journal of Neuroscience* **18** (6): 2283–300.

availability of the data, meaning that Bayesian networks are ill-suited in a situation where data is available incrementally. The final and main drawback is that, while Bayesian networks are great at dealing with ordering, they do not take into account the subtleties of timing information.

2 Steps and Thesis Outline

Animals are great at picking up patterns in events. Early experiments in classical conditioning showed that many animals are able to understand relationships between events. Pavlov famously showed that he could train a dog to associate a random event like a bell ring with food just by consistently ringing a bell before serving the dog food.

While animal brains have a staggering complexity, neuroscience research over the past ten years has begun to uncover some basic rules that govern neuron activity. One of them is called *Spike Timing Dependent Plasticity* (STDP), and it quantifies how the strength of neurons' connections changes as neurons fire[2].

I picked a particularly recent STDP model presented by Clopath[3] due to its computational simplicity, and implemented its functionality. This allowed me to simulate how neuron connections change as they fire. Because the original problem is to find patterns in timed data, I map each piece of timed data to a neuron. When an event is processed, I cause the corresponding neuron to fire. The idea is that the connections formed by the neurons can tell you something about the relationships between the events.

[2] Bi GQ, Poo MM (15 December 1998). "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type". *The Journal of Neuroscience.* **18** (24): 10464–10472.
[3] Clopath, C., Büsing, L., Vasilaki, E. & Gerstner, W. *Nat. Neurosci.* **13**, 344–352 (2010).

In section 4, I explain a little bit more about the functional aspects of the phenomenon that inspired the pattern finding algorithm. In Section 5, I explain the details of how the algorithm works and how it was implemented. In section 6, I apply the algorithm to automatically generated data that can be carefully controlled and that contains known relationships. In Section 6.6, the algorithm is applied to stock price fluctuations and finds relationships between companies within the same industry. Finally, Section 7 covers future work that can be done to expand on this thesis and understand the limits of this approach to pattern finding.

3 News

In this thesis, I show that it is possible to find relationships in timed-data using an algorithm inspired in STDP. In a landmark experiment, I show that even amongst overwhelming amounts of noise it is still possible to find timing patterns between events.

As a concrete example, when I generated data simulating 102 types of events where 2 were related in time and 100 were random, the algorithm easily identified the two that were related.

The events were related in time, event *A* and *B*, consistently happened closely together but in no particular order. More precisely, event *A* always caused event *B* to occur after a short interval, and vice versa. The exact interval amount was chosen to be the average timing between the random events. The other 100 events were random, distributed with a Poisson distribution. All 102 events had equal frequencies (number of occurrences per unit time). The algorithm was able to easily identify the two events that were related. Figure 1 shows the relative strength of the relationships found after an after an exposure to 1000 events (of which about 98% are the random events). Because the events are randomly presented, each event is shown only 10 times on average. The algorithm finds that the strength of the relationship between *A/B* is about 9 times stronger than the relationship between any of the other events.

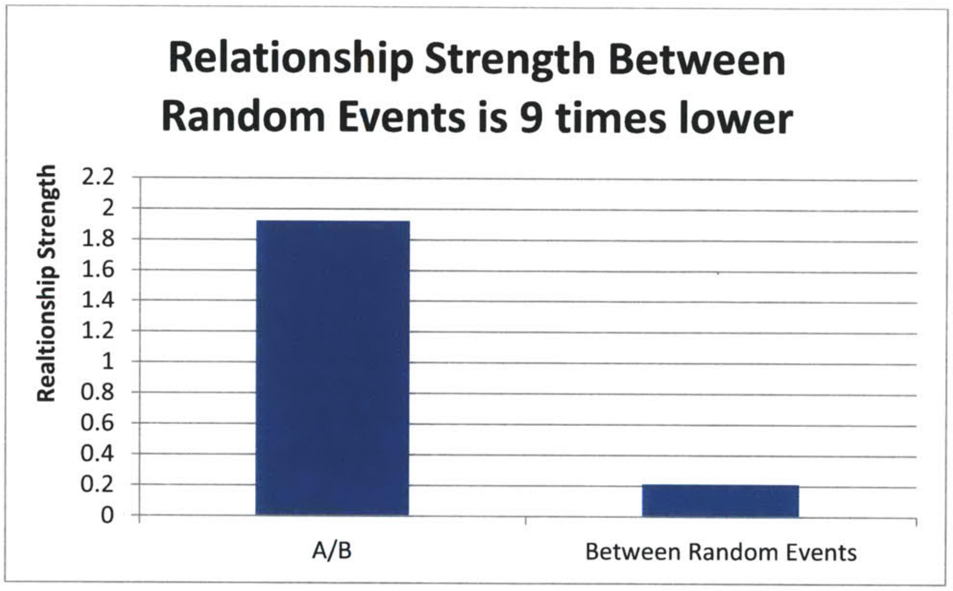

Figure 1 - Given 102 different types of events, where 2 of them (*A* and *B*) have a relationship between them, the algorithm unequivocally identifies the existing relationship with an input with only 1000 randomly generated events.

In this thesis I also show that the same algorithm is capable of finding relationships between company stock price data. Using information about which days stocks go up and down for 12 major U.S companies, it finds strong relationships between companies in the same industry. In fact, the top 10 strongest relationships are all between companies in the same industry (see Table 1). Oil and gas companies as well as banking companies had the strongest relationships among themselves.

Company 1	Company 2	Relationship Strength
Exxon Mobile UP	Chevron UP	13.97
Chevron UP	Exxon Mobile UP	13.94

Citigroup DOWN	JPMorgan DOWN	13.05
ConocoPhillips UP	Exxon Mobile UP	12.92
Exxon Mobile UP	ConocoPhillips UP	12.89
AIG DOWN	JPMorgan DOWN	12.87
Chevron UP	ConocoPhillips UP	12.85
JPMorgan DOWN	AIG DOWN	12.81
ConocoPhillips UP	Chevron UP	12.72
JPMorgan DOWN	Citigroup DOWN	12.62

Table 1 - List of the strongest relationships found between 12 major U.S. companies. The top 10 strongest relationships are all between companies in the same industry.

Furthermore, in this thesis I also demonstrate that the algorithm can indentify causal relationships (events that seem to be causing other events) as well as coincident relationships (events that tend to happen together in time).

Another feature of the algorithm is that it creates a graph of event relationships on the fly, and updates it for each piece of data – it is an online algorithm. This also means that the input can be a continuous stream of event data with no length restriction.

Finally, it does not require any *a priori* knowledge about the data. This means that arbitrary or poorly understood timing data can be fed in, and is less likely to be influenced by a human bias.

4 How Spike Timing Dependent Plasticity Works

The abilities of STDP were first highlighted a little over a decade ago[4], but STDP has been receiving more attention as mounting evidence shows that this rule plays a central

[4] Bi GQ, Poo MM (15 December 1998). "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type". *The Journal of Neuroscience.* **18** (24): 10464–10472.

role in learning. In particular, theories on the mechanism of STDP have recently been proposed[5].

STDP is a Hebbian learning rule that quantifies the change in connection strength between two neurons given their exact firing times. The graph in Figure 2 is typically used to summarize the effects of STDP. So, for example, if neuron *A* spikes and then neuron *B* spikes, the connection between *A->B* will be strengthened while the connection *B->A* will be weakened. The magnitude of the connection strengthening (LTP or Long-Term Potentiation) is thought to be stronger than the corresponding weakening (LTD or Long-Term Depression), thereby allowing connections to become stronger over time and over the course of many firings.

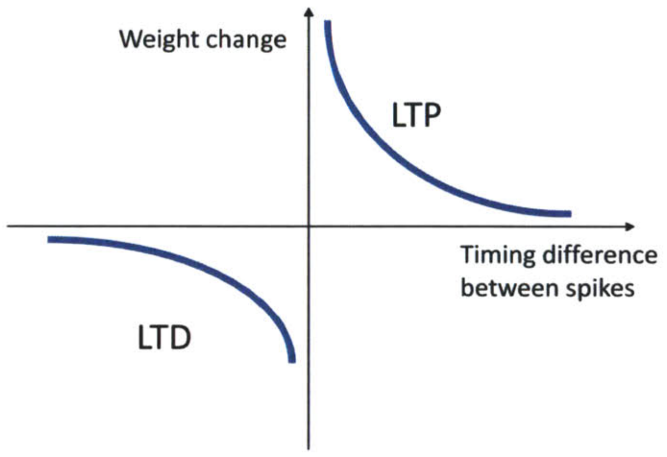

Figure 2 - Graph illustrating the connection weight change between a pair of neurons, given the timing difference between their firing. More specifically, this is the timing difference between the pre-synaptic and post-synaptic spikes. For example, if neuron *A* spikes and then *B* spikes, the connection between *A->B* will be strengthened and the connection *B->A* will be weakened. The label "LTP" refers to Long-Term Potentiation, or increasing connection weight, and "LTD" is Long-Term Depression, or decreasing connection weight.

[5] Clopath, C., Büsing, L., Vasilaki, E. & Gerstner, W. *Nat. Neurosci.* **13**, 344–352 (2010).

If you apply the weight changes described by this STDP curve to a sequence of firing neurons, strong connections will be created between pairs of neurons that tend to spike in a particular sequence. Figure 3 illustrates two different types of relationships: causal and coincidental relationships. For example, in a causal relationship a spiking in neuron *A* may be consistently followed by a spiking of *B* (Figure 3A). I call it causal because it seems like *A* is causing *B* to fire. On the other hand, a coincidental relationship would be one where, for example, *A* and *B* tend to spike together but not in a particular order (Figure 3B).

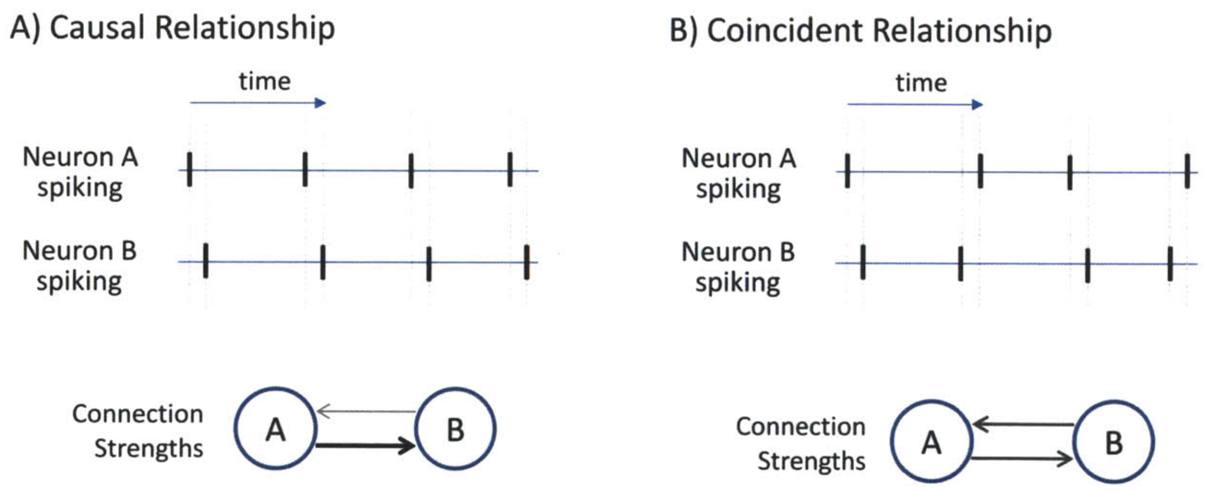

Figure 3 - Above, a timeline of spiking is shown for both neuron *A* and *B*. Each dark line on the timeline represents a spiking event. Below, the resulting connections as a result of STDP are shown. Thicker darker arrows represent stronger connections. Part A) shows a causal relationship between *A* and *B*. Part B) shows a coincidental relationship.

5 Method

I developed an algorithm that takes in a sequence of events happening over a period of time. It outputs a graph illustrating the relationships between the events.

5.1 Input

The input is a list where each element of the list has both a label and a time.

ex: [('falling glass', 1) , ('shattering sound', 2.2) , ('broken glass pieces', 2.5)]

Throughout this thesis, I will illustrate the input on a timeline for clarity. The x-axis is the time, and the labeled lines represent the events (see Figure 4 below).

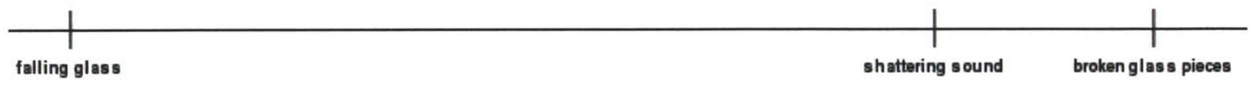

Figure 4 - Illustration of the input on a timeline. This is more intuitive to see than a sequence of numbers, so this will be the format shown throughout the thesis.

5.2 Output

The output is a graph with nodes and directed weighted edges.

Each node has a name and can have multiple directed edges. I represent the weight of the edges using both thickness and color (see Figure 5). Stronger edges are thicker and are shown in red. Weak edges are thin and are shown in yellow. See Figure 6 for a more precise color scale.

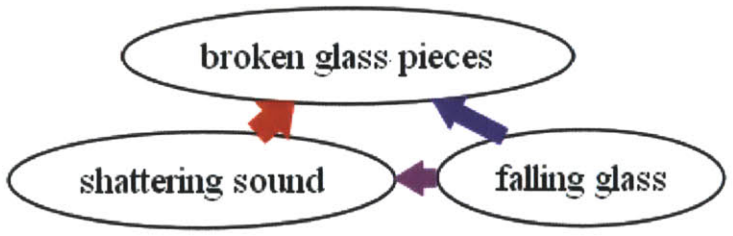

Figure 5 - Graph representation of the output of the algorithm. In this example you see that "shattering sound" is related to "broken glass pieces" very strongly.

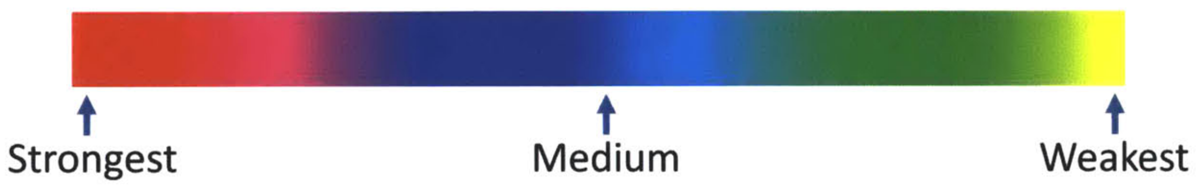

Figure 6 - How strength of edges relates to the color of the edge. Stronger edges are also drawn thicker than weaker edges for emphasis.

The output graph scale is normalized so that the strongest edges are always shown in red (and thick) and edges with zero weight are shown in yellow (and thin). This is merely for display purposes, to make the edge weight differences easier to perceive. This normalization has no effect on the underlying edge weights.

5.3 Algorithm

The input list is read one element at a time. Every time an element with a new label is seen, a new node is added to the output graph.

5.3.1 Node Potential

Every node stores a quantity called potential, a quantity that always decreases exponentially over time. This quantity is boosted up by a constant amount when the node's label is read from the input. Figure 7 illustrates this by placing the input and a graph of the node's potential side-by-side.

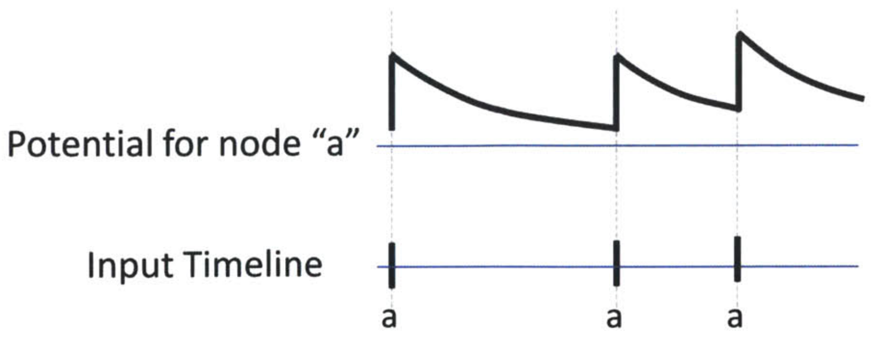

Figure 7 - Relationship between the input and the potential of a node. Each time the node "a" is seen on the input, the potential for node "a" is increased by a constant amount. Notice how potential decays

exponentially over time. As usual, time appears on the x-axis.

The potential at a given time t is calculated using a simple negative exponential:

$$potential(t) = potential(t_s)e^{-d(t-t_s)}$$

where t_s is the time of the last spike, d is a decay exponent, and e is Euler's number. The value for decay d is decided experimentally and depends on the input data.

This decay time constant can be thought of as the strength of memory in the system – the slower the decay, the longer the memory. For data that is more spread out over time, a d that is closer to zero (slower) is better for picking up the patterns. For data that happens quickly and in rapid succession, a larger (faster) d is preferred. Figure 8 shows how the quality of the pattern extraction can be affected by the constant d that is chosen. For values of d that are too small, the patterns are less clearly extracted. For values of d that are too large, the output (the relationships discovered) become very sensitive to small changes in the input (see Figure 9).

The "clarity" of the patterns extracted given the decay was measured using data that contained known relationships (more precisely, it is the exact data used in section 6.3). By knowing the relationships contained in the data, I know which edges should be present in the output graph. The "clarity" of the patterns found is given by:

$$\frac{Average\big(Weight(Correct)\big)}{Average\big(Weight(Incorrect)\big)}$$

Where $Correct$ is the set of edges that represent relationships present in the data, and $Incorrect$ is the set of edges that represent relationships that are not present in the data.

Weight returns the set of weights corresponding to the given set of edges. *Average* simply returns the average value of the set.

Unless stated otherwise, for all the examples throughout this thesis, a value of d=2.0 was picked as a compromise between these two extremes.

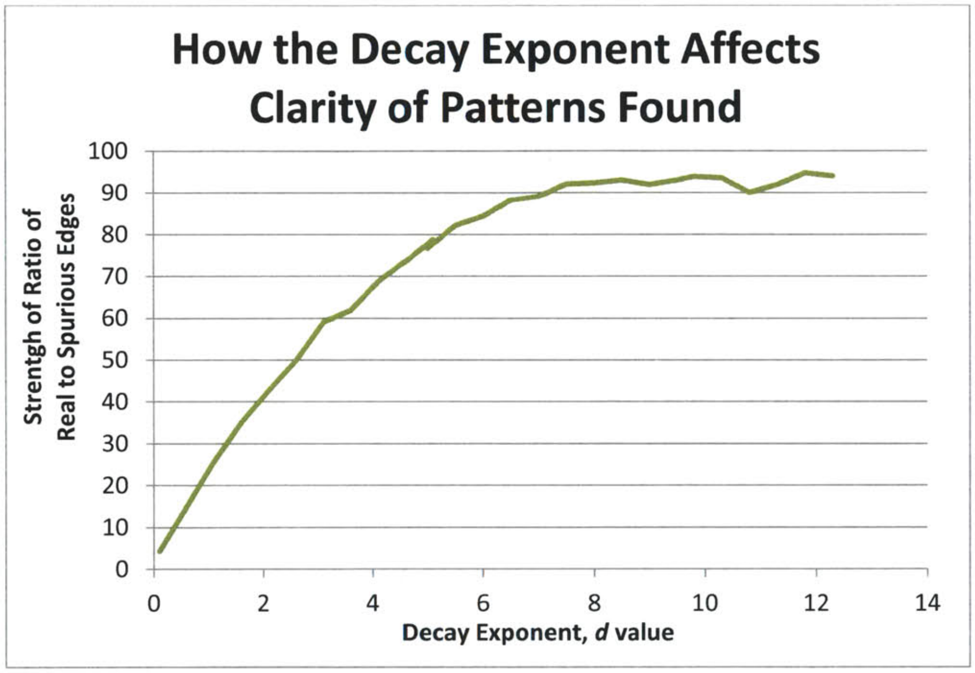

Figure 8 – As the decay exponent increases, the clarity of the pattern increases on average. 350 trials were used to come up with the average clarity value.

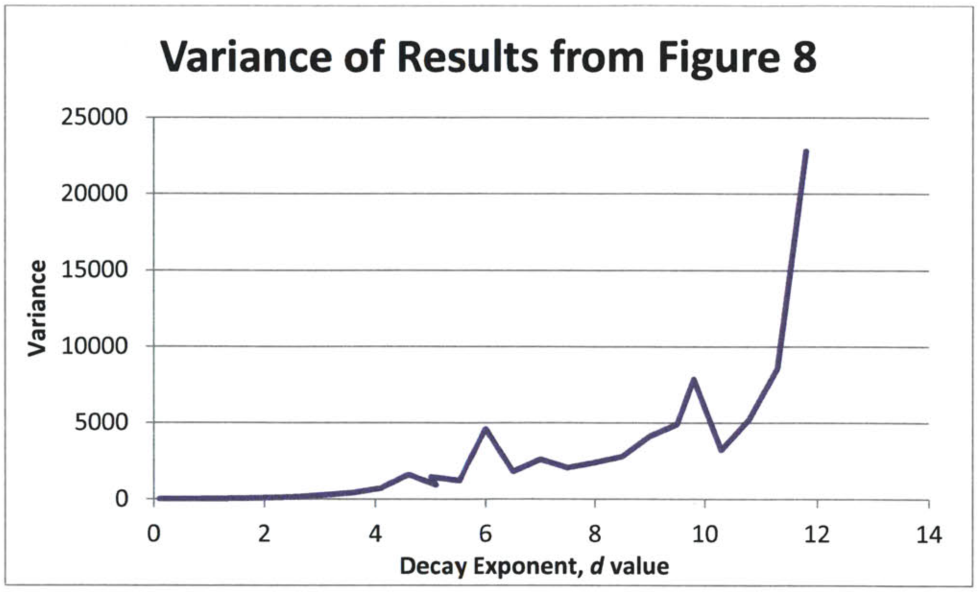

Figure 9 – Variance of the results from Figure 8. As the decay exponent increases, the variance of the data increases dramatically.

Now that I have gone over how the potential for nodes is determined, I will go over how edge weights are changed.

5.3.2 Edge Weights

Edge weights are changed every time an event is processed. For example, imagine you have event *A* coming in as input. If this event doesn't already have a corresponding node in the graph, that node is created and added to the graph. Otherwise, if it already exists in the graph, you consider its neighbors. We'll call neighbors the nodes that have edges going *into* node *A*. In Figure 10 for example, nodes N_1, N_2 and N_3 are neighbors of *A*. However *A* is not a neighbor of N_1 because you can't go from A to N_1 by following the arrow.

For each neighbor *N* of *A*, I adjust the connection from *N* to *A* depending on *N*'s potential. Following Clopath's model of STDP, there are 3 possible scenarios depending on the node's potential:

- If *N*'s potential is low, no edge weights are changed.
- If *N*'s potential is medium, the strength of the connection from *N->A* is *decreased.*
- If *N*'s potential is high, the strength of the connection from *N->A* is *increased.*

See Figure 10 for a representation of these thresholds.

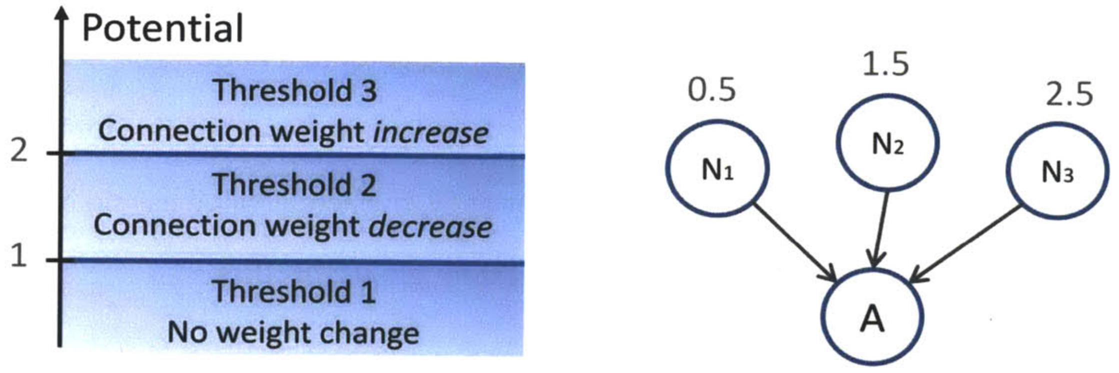

Figure 10 - On the right, a scale showing the different thresholds for different node potentials. The shown scale is arbitrary. On the right, a sample arrangement of nodes. The label above neighbors N_1, N_2, N_3 corresponds to an example current potential. Given the scale on the left, the N_1 -> A connection would be unchanged, the N_2 -> A connection would be weakened, and the N_3 -> A connection would be strengthened.

The exact amounts for the decrease and increase are proportional to *N*'s potential. The increase is made twice as large as the decrease, obeying the traditional STDP curve where potentiation is twice as strong as depression. Here is pseudo-code summarizing what was mentioned above:

```
if (N_potential > small_threshold):
   N_to_A_edge_strength += N_potential
else if (N_potential > large_threshold):
   N_to_A_edge_strength -= 0.5 * N_potential
```

For the purposes of this thesis, the exact threshold values they were inspired by the values observed in STDP measurement experiments[6] and were fine-tuned by a little

[6] Bi GQ, Poo MM (1998). "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type". *The Journal of Neuroscience.* **18** (24): 10464–10472.

experimentation. Empirically, the actual values have little effect on the output as long as `small_threshold > large_threshold`. Throughout this thesis, the values used are:

```
small_threshold = 0.01
large_threshold = 0.1
```

Another variable that goes hand-in-hand with these is the amount a node's potential is increased by when a corresponding event is observed. As long as it is a positive number and the potential decay exponent is chosen appropriately, the actual value has no impact on the relative weights of the edges. The value used throughout this thesis is:

```
potential_increase = 0.5
```

Finding the optimal values for these three combined variables for a given input would be an interesting area of further research.

Just to go over an example, consider event *A* followed by *B*. For simplicity, assume that these are not the first events in a sequence, meaning *A* and *B* are already connected together and both have a positive, but low, potential. When event *A* is processed, its potential is increased. Then its neighbors are considered. The only neighbor is *B*, whose potential is low. This means there are two possibilities:

1) The edge *B*->*A* is weakened if *B*'s potential is above `small_threshold` but below `large_threshold`
2) The edge *B*->*A* is not changed if *B*'s potential is below `small_threshold`

That completes the steps when processing event *A*. Now when the next event is processed, event *B*, things are a little different. The main difference is that *A* now has a very high potential (because the potential was recently increased). This means that *A*'s potential will be above `large_threshold` and the edge *A*->*B* will be strengthened strongly.

The overall effect is that edge *A*->*B* is strengthened. This was the desired effect because *A* was directly followed by *B*. The closer together in time *A* and *B* are, the higher *A*'s potential will be, and the stronger the connection increase will be.

5.4 How connections are initially formed

I've mentioned how connections strengths change, but not how they are originally formed. Working with a complete graph is too inefficient. Not only would a lot of memory be spent on storing unimportant edges, but it would mean each node would be a neighbor of all other nodes. If there are N nodes in the graph, each event would cause N edges to be updated, mostly with an insignificant effect.

A simple solution was implemented using a short queue that keeps track of recent events. This queue keeps the most recent events processed over a set interval. The length of the interval is set so that over the interval the potential decays down to an insignificant amount. Throughout this thesis I used an interval of 10 time units, at which point the potential decays to about 10^{-7} of a percent of the original value.

The graph starts with no connections. When an event is processed, edges are created between this event and all events in the queue. If the edge already exists, nothing is changed. Since edges weights are only increased when neighbors have a significant potential, the use of this queue allows the algorithm to avoid adding edges that would be very close to zero.

6 Results

In order to test the algorithm, I started out by running it on artificial data. This allowed me to carefully control the patterns contained in the input data.

6.1 Causal Relationships

To start out, I created data that simulated a caused event. Consider two events, "A" and "B".

Over a set span of time, "A" events were randomly distributed with a given frequency (a given rate parameter λ), using a Poisson distribution. I then made sure that after a certain interval every "A" event was followed by a "B" event (see Figure 11 for sample data).

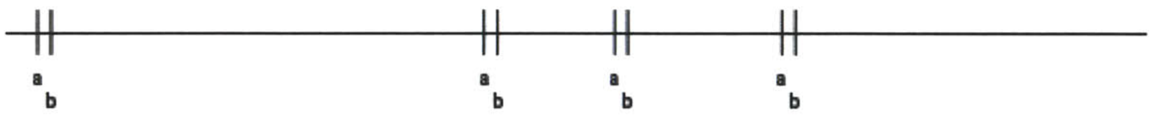

Figure 11 – Sample data generated automatically to simulate causal events. "A" events are distributed randomly (with a given frequency) and "B" events are forced to follow each "A" event after a given interval of time.

The auto-generated input data had on average 20 A/B pairs that took place over 200 time units. The Poisson rate was $\lambda = 0.1$, with an average of a pair every 10 time units. The interval between the "A" event and the "B" event was 0.1 time units.

Figure 12 shows the results of the algorithm. As you can see, it clearly extracts the causal relationship indicating that A seems to be causing B. Across different trials that used different instantiations of the data (different due to randomness of the distribution), the results were consistently the same.

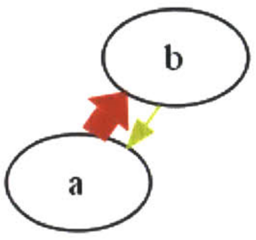

Figure 12 – Graph illustrating the relative strength of the connections formed given data with causal data. The strength of the connection between *A->B* is much stronger than *B->A*. This indicates that there is an asymmetric relationship from *A* to *B*. It is called a causal relationship because *A* seems to cause *B*.

The ratio of the edges weights *A->B* to *B->A* differed slightly across different trials. The value averaged over 100 trials was 29.7, implying a strong causal relationship.

6.2 Coincidental Relationships

The goal of the following experiment was to find out if the algorithm could identify symmetric relationships, also called coincident relationships. This is the kind of relationship that represents events that tend to happen together in time, but in no particular order.

Like before, there were two types of events, *A* and *B*. They were randomly spread out with an equal frequency over a set amount of time. Then, for each of these events, the opposite event was added after a certain interval. After each *A*, a *B* event was added. Likewise, after every *B* event an *A* event was added (see Figure 13 for sample data). The Poisson rates and the intervals were the same as those used before in the causal experiments.

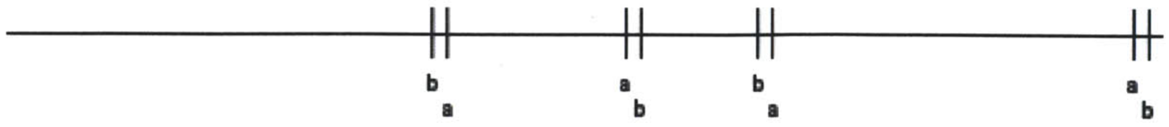

Figure 13 - Data generated automatically to simulate coincident events. Both "A" and "B" events are added randomly (with a given frequency) and then followed by the corresponding opposite event ("A" by "B" and "B" by "A") after a given interval.

Figure 14 shows the results of the algorithm. Unsurprisingly given the symmetry of our data, the relationships between our events now appear symmetric.

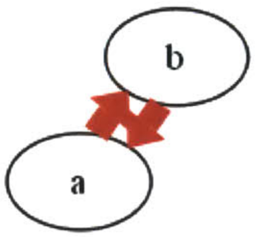

Figure 14 – The strength of the connections between *A->B* and *B->A* are very similar. This shows that the symmetric relationship between them was successfully captured. I call this a coincident relationship.

The ratio of the edges weights *A->B* to *B->A* differed slightly across different trials. The value averaged over 100 trials was 1.02, implying a coincident relationship.

6.3 Putting it together

Any arbitrary combination of these relationships can be found. For example, consider data with 4 events: *A, B, C* and *D*. Event *A* will cause *B* and events *C* and *D* will be coincident, meaning they will tend to happen close in time but in no particular order. Figure 15 shows a sample of the data, and Figure 16 shows the graph representing the results.

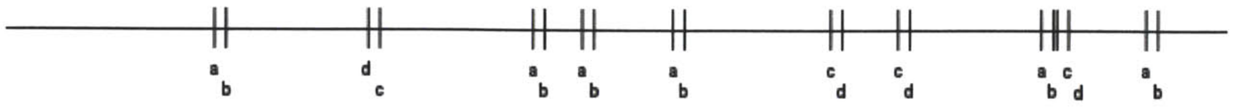

Figure 15 – Sample timeline of events simulating a causal relationship between *A->B*, and a coincident relationship between *C<->D*.

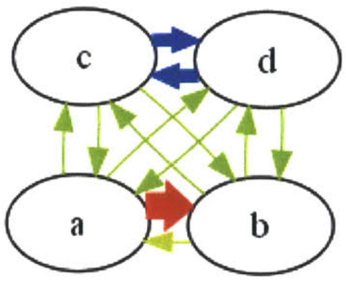

Figure 16 – The relationships present in the data are visible and easily extracted. The unwanted edges all appear in yellowish green, and the expected edges are much darker and thicker.

Here are the main ratios in this graph:

- *A->B* to *B->A*: 28.1
- *C->D* to *D->C*: 1.0
- *A->B* to *C->D*: 1.9
- Average(*A->B*, *C->D*, *D->C*) to Average(Remaining Edges): 9.6

The expected edges are on average almost 10 times stronger than the unwanted edges. This shows that the algorithm clearly extracts the patterns present in the data for this experiment.

Notice how the causal edge is stronger than the coincident edge (about 2 times stronger). This was expected given that there are just as many causal events as there are coincidental events. Causal events always strengthen the same edge (in this case A->B), while coincidental events are spread out strengthening two edges (in this case both C->D and D->C). Each of these pairs of events increases the connection strength by a constant amount since the interval between the pairs is the same. Therefore it follows that the causal edge ends up about twice as strong.

6.4 Introducing Noise

6.4.1 Missing Information

So far the patterns have been extracted under idealized conditions. The data is perfect in the sense that there is nothing missing – we observe all the relevant events. However, in real life we often miss important information. Sometimes events are not visible or our senses are imperfect.

More concretely, imagine that you are learning that a falling glass is followed by a shattering sound. It's possible that you hear the shattering sound but don't look to see what shattered. On the other hand, you might see a glass falling but it's too noisy for you to clearly hear the sound. A robust learning system should be able to handle these imperfections that sometimes cause information to be missing.

I used the three sets of data used for sections 6.1, 6.2 and 6.3, representing causal, coincident, and the combined relationships. I modified them so that 50% of the data is randomly removed (See Figure 17 for a sample).

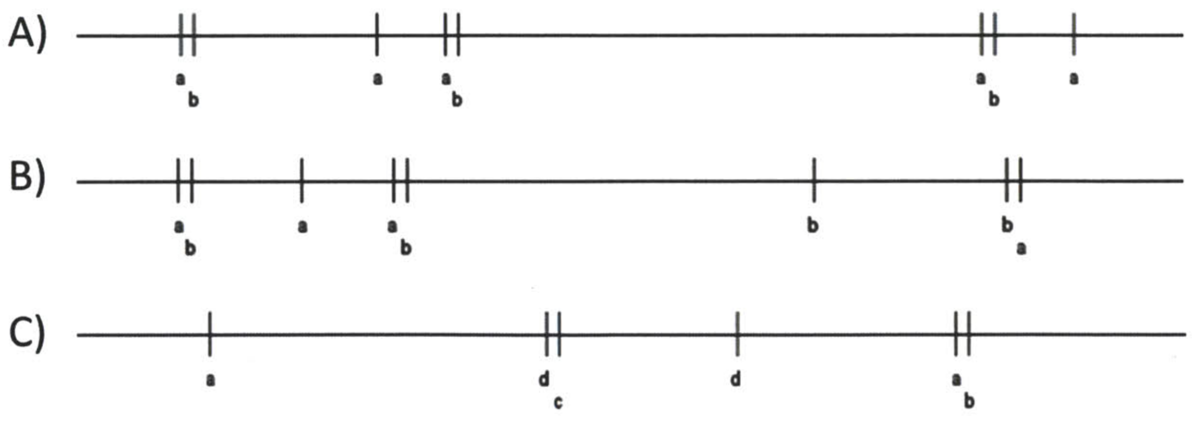

Figure 17 – Sample data for the A) causal relationship and B) coincident relationship. 50% of the original data was removed, so instead of pairs you can often see isolated "a" or "b" events.

Surprisingly, even with half of the data removed, the algorithm manages to fare well and clearly extract the patterns contained in the data. The patterns consistently resemble those extracted with complete information (see Figure 18).

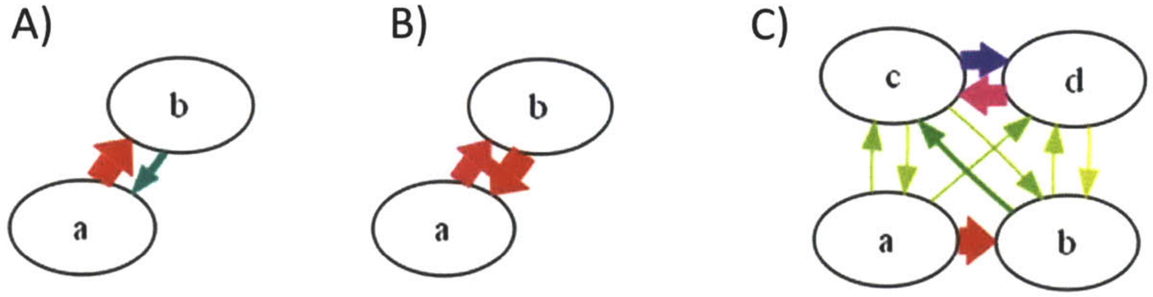

Figure 18 – Even though half of the data was removed, the relationships are still clearly discovered. A) shows the causal relationship B) shows the coincident relationship C) shows a combination of these. These are the same relationships used in Figure 16 where *A->B* and *C<->D*. The fact that the *C<->D* relationship doesn't appear perfectly symmetrical reflects the fact that when 50% of the data is removed, some result variance is already observed across trials.

This algorithm seems to be robust against missing information. It does well when up to 50% of the information is gone. However, when more than 50% is removed, the results start to quickly deteriorate. The results start to become unpredictable, or in other words, start having too much variance across trials (see Figure 19).

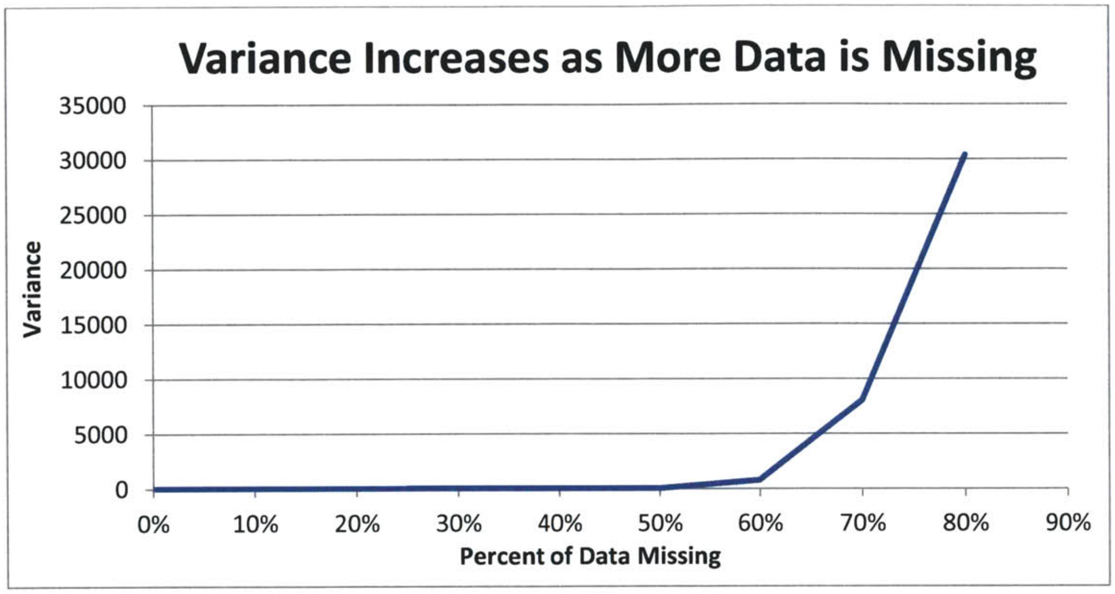

Figure 19 - As a greater percent of the input data is randomly removed, the algorithm's output becomes more unpredictable. I compared the strength of edges in the output that represent relationships present in the data, with the strength of edges that are not present in the data. The graph shows the variance of the ratio of these two quantities. This ratio quantifies the quality of relationship extraction. When the variance becomes too high, the relationship extraction becomes too erratic.

6.4.2 Extra Information

Still the pattern extraction is a little idealized. Often not only is there missing information, but there is *extra* information. Often, we observe information that has nothing to do with the pattern we are trying to extract.

Consider the example where you are learning that a falling glass is followed by a shattering sound. While learning, there will likely be other things going on simultaneously. There may be dogs barking, wind blowing, people chatting, colorful balls bouncing – generally things that could confuse the learning process. A robust algorithm should be able to learn even amid these unrelated distractions.

I re-generated the three data sets in 6.4.1, except this time I added 3 distracting events which I call n1, n2, and n3 (n stands for noise). These represent random events with no particular relationship with one another. For this experiment, I controlled the amount of distracting events so that the sum of the frequencies of n1, n2, and n3 are the same as the sum of the frequency of *A* and *B* events. This essentially means that there is as much noise as there is useful data. Figure 20 shows a sample of this input data.

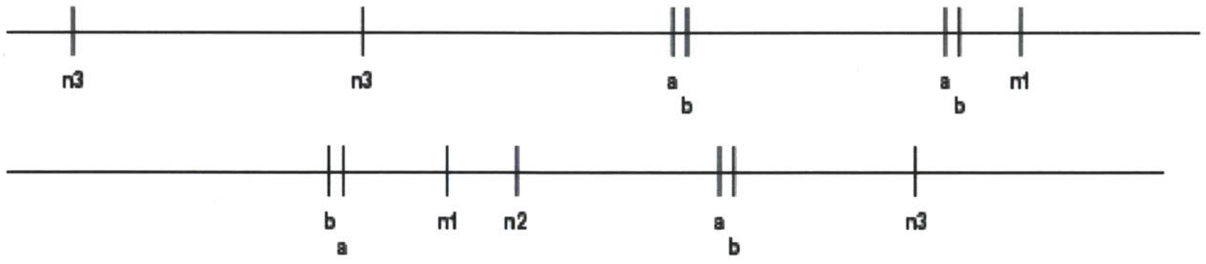

Figure 20 – Subset of the data illustrating the kind input that the algorithm receives. The top timeline represents the causal data, and the bottom one represents the coincidental data.

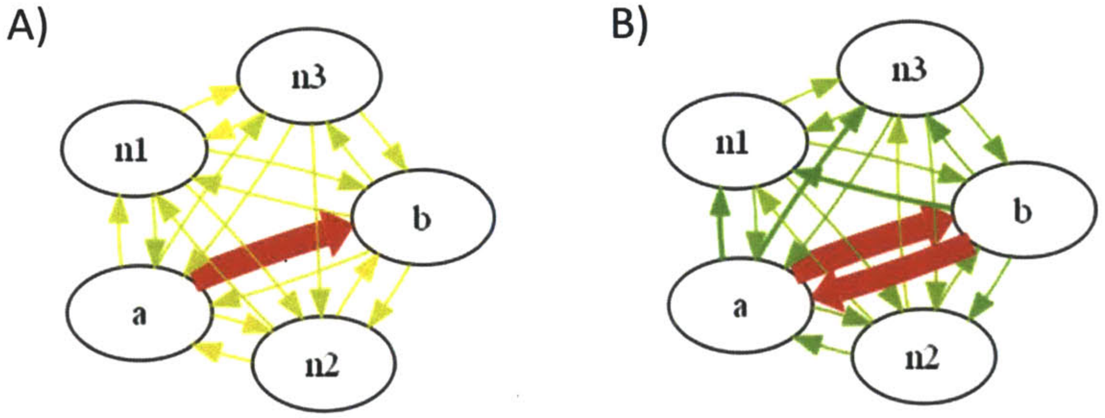

Figure 21 – Attempting to find patterns in noisy data. Three noise events (n1, n2, n3) are introduced at a frequency equal to the frequency of useful data (a, b). Even so the relationships present in the data are still clearly extracted (indicated by the strong red arrows). Part A) shows a causal relationship between a->b and part B) shows a coincident relationship between a<->b.

As you can see in Figure 21, even though there is as much random noise as useful data, the patterns are still clearly discovered. In the coincident case (part B in Figure 21), the weights of the strongest edges (shown in red) are on average 11 times stronger than those of the weaker ones (shown in green). In the causal case (part A in Figure 21), this value is 41 times.

Naturally, as the amount of noise is notched up, the patterns become harder and harder to extract. Figure 22 illustrates more precisely how the amount of noise affects the relationship extraction.

So far we've been using only 3 different types of noise for simplicity. It is important to point out that this is not very realistic. When learning patterns, there's a wide variety of distracting factors which are certainly not limited to 3 types. A realistic number might be closer to 100 or 1000. Figure 23 illustrates how the extracted relationships hold up to noise when there are 100 different types of noise. In this more realistic case, the algorithm can tolerate extremely high levels of noise while still clearly extracting the true relationships present in the data.

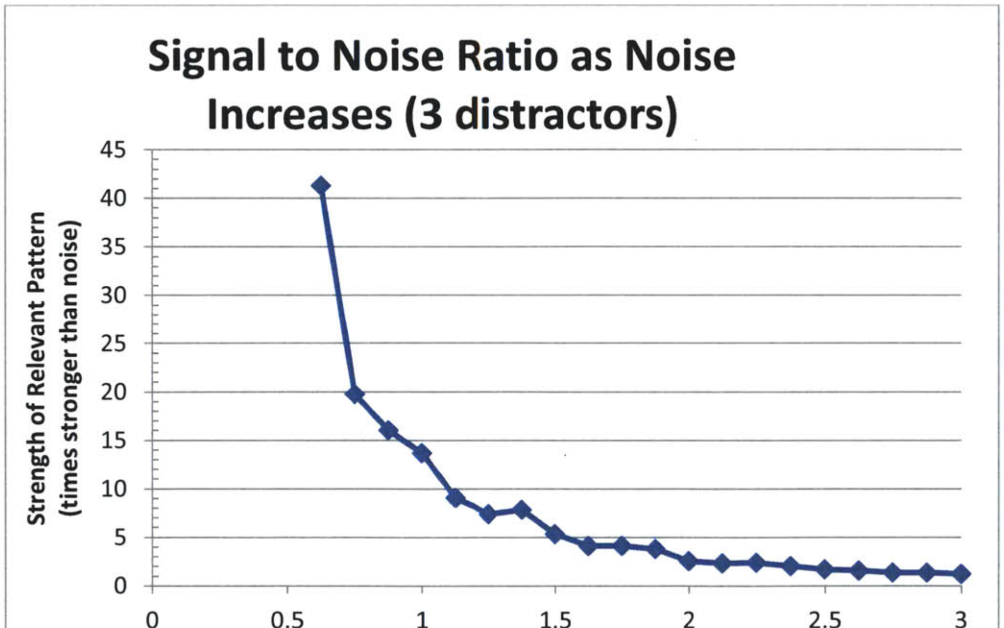

Figure 22 - How the strength of the extracted signal decays as amount of noise in the data increases. More precisely, the x-axis shows the ratio of the frequency of noisy events to the frequency of A/B event pairs. The y-axis shows the ratio of the average strength of relevant edges (red in Figure 23) to the average strength of noise edges (green in Figure 23). If the amount of noise is reduced to zero, the strength of the extracted pattern becomes infinitely stronger in comparison to the strength of the edges between the noisy events.

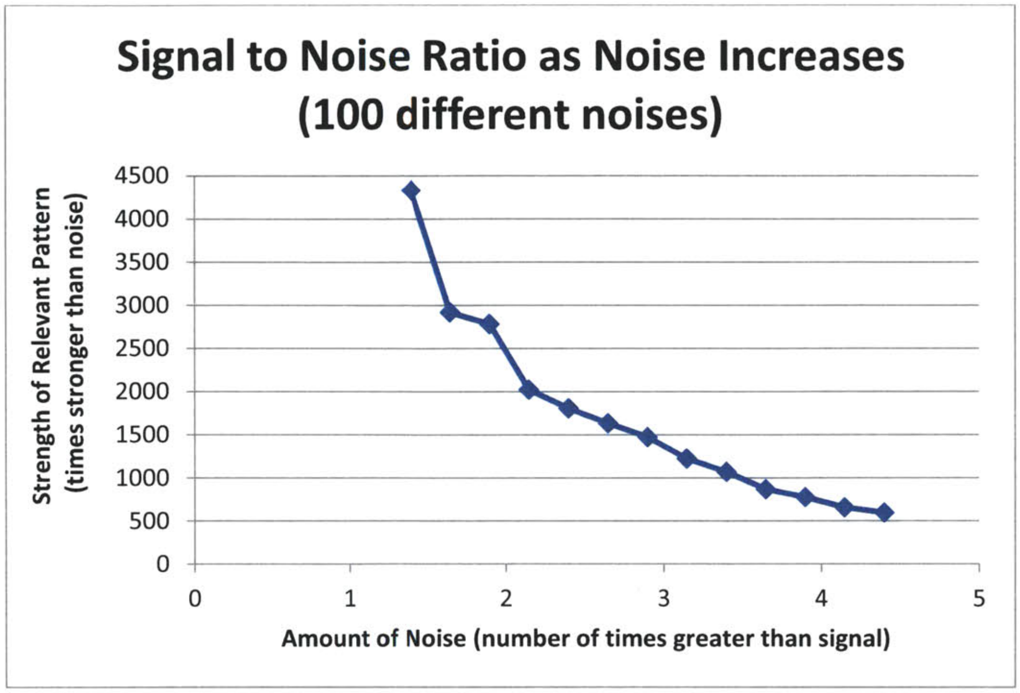

Figure 23 - Same as Figure 22, except using 100 different types of noise. It is now far more resistant to noise. Even when there is 4 times more noise than signal, the strength of the weights of the extracted relationships are still 500 times stronger than that of the other edges.

It's very encouraging to see that this simple algorithm can be resilient to very high amounts of noise, something that has traditionally been hard to deal with (in machine learning techniques such as Bayesian networks and support vector machine).

6.5 Extracting Patterns from a Story

One way to test the ability of the algorithm to find patterns is to apply it to data that is well understood and for which I can qualitatively gauge the results of.

An interesting such data set would be a well known children's story. When you listen to a story, you are essentially listening to sequences of sounds at particular points in time. These sounds form words, each of which could be thought of as an event in a sequence. I chose to work with the story "The Wizard of Oz" in particular because it is in the public domain.

To run the algorithm over a text, I did some basic pre-processing to generate the events from the story. I assumed the following:

- Each word is an event
- Periods represent long pauses
- Commas represent short pauses
- Period pauses are twice as long as commas pauses

Using these assumptions, I was able to turn the entire text into event-timings where each word is an event in time.

A first run through the text, gave uninteresting results. Most of the strong relationships were between very frequent words such as "the", "a" and "it." These words had hundreds of connections with other words, even though they are irrelevant to the plot of the story.

The problem with these words is that they form strong connections with a large variety of other words. This means that the list of strongest edges is overwhelmed by these connections.

This qualitative problem was solved by limiting the number of connections a node could make. More precisely, it limited the sum of weights of all outgoing connections. For simplicity, I will call this quantity the degree of the node.

Instead of removing nodes above a particular degree altogether (which also affects the connections to other important nodes), a softer option was taken. Once a node reaches a certain degree, the connection strengths are normalized as to keep the degree constant (see Figure 24 for a more intuitive illustration). This means that words with very high degree of connections quickly 'spread themselves too thin' and end up with very weak connections.

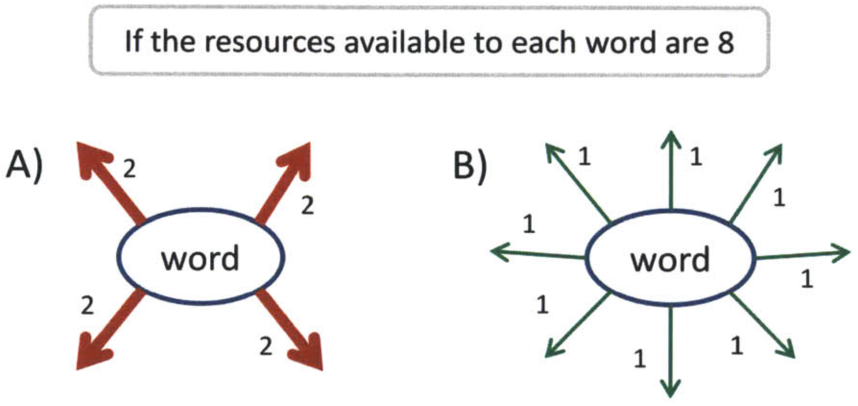

Figure 24 – Whenever a word reaches a certain limit, which I call the "available resources", the sum of the edge weights starts being normalized. This means the sum of all the edge weights never goes above this limit. This decreases the strength of the edges between very common words with very high degree like "the", "and" and "it".

When this normalization was put into place, the list of strongest relationships became very different. Surprisingly, the strong connections all came in pairs. The kind of pairs found such as "Emerald City" and "Tin Woodman" are satisfyingly relevant to the story, showing that qualitatively this algorithm can find basic patterns in simple stories (see Table 2 for the top 10 pairs).

Picked	Up
Winged	Monkeys
Emerald	City
Tin	Woodman
Guardian	Gates
Wicked	Witch
Uncle	Henry
Rule	Over
Ball	Fire
Aunt	Em

Table 2 - Top ten pairs of words found in "The Wizard of Oz" with strongest relationships between them. Aside from "Picked Up" the other relationships are relatively important elements of the story.

It makes sense that the strongest relationships are between common compound expressions or names because these are the words that when they appear, they tend to appear together. Because a maximum degree is set, words that are connected to many other words end up with weak edges. Words that are connected to only a few other words but appear often are the ones that end up with the strongest edges.

It turns out, for example, that every instance of "Uncle" is followed by "Henry" (because he is the only uncle in the story). The other pairs are in similar situations. "Picked," as another example, is followed by "Up" in 8 out of 10 cases. The other 2 cases are "picked them up" and "picked him up" which would also strengthen the "picked" -> "up" relationship (although not as strongly as simply the expression "picked up").

This short experiment showed that the algorithm is working as expected and finding patterns between events that happen together.

6.6 Extracting Patterns from Stock Prices

One type of data that lent itself well for analysis by this algorithm was stock prices. They are essentially a series of values changing over time which have potentially interesting correlations.

Stock prices are continuous values. In order to be processed by the algorithm, however, they needed to be transformed into discrete events. The way I did this was to simplify stock fluctuations as just "up" events and "down" events, depending on whether the stock value increased or decreased across trading days. This way, each company's stock fluctuations was reduced to just 2 types of events happening over time.

The stock price data was collected from Google Finance, which makes this data publicly available. I used values from over the course of one full year (from Jan 15, 2011 to Jan 14, 2012) and considered only the values at the start of each trading day. This provided about 250 individual events for each company's stock.

So, for example, if the stocks being analyzed were Citigroup and Microsoft, there would be 4 possible events: "Citigroup UP," "Citigroup DOWN," "Microsoft UP," "Microsoft DOWN." This sequence of events along with their timestamps was provided to the algorithm as input. The timestamps were simplified to be just the number of trading days since the first data point (Jan 15, 2011). This means that weekends and public holidays, for example, are not counted as days.

The companies that were chosen were picked from four different industries: finance, health care, oil and gas, and information technology. The companies were picked as the top three largest U.S. companies in their industries by revenue[7]. Table 3 lists the exact companies used.

Finance	Health Care	Oil and Gas	Information Technology
JP Morgan	McKesson	Exxon Mobil	Hewlett-Packard
Citigroup	Cardinal Health	Chevron	IBM
AIG	United Health	ConocoPhillips	Microsoft

Table 3 - List of companies whose stock was chosen to be analysed by the pattern finding algorithm.

One initial issue that needed to be addressed was how to handle simultaneous events – events that happen at precisely the same time. There were a lot of simultaneous events because each trading day provided several data points, all of which had exactly the same data stamp. The algorithm, however, has no well defined behavior for simultaneous

[7] "Fortune 500 2011." Fortune on CNNMoney.com. May 23, 2011. CNNMoney. 23 Jan. 2012. <http://money.cnn.com/magazines/fortune/fortune500/2011/full_list/>

events. The desired behavior would be to get coincidental relationships formed between these simultaneous events.

The way to get around this was to introduce very small random shifts in the timing of the events to break the ties. Because the shifts were random and very small, over time and on average the simultaneous events formed the desired coincidental relationships.

Something else that needed to be decided was the potential decay exponential. Because this data's distribution over time is different from the data I dealt with before, I needed to determine which was the best decay exponential to use. The decay exponential is important because it affects the algorithm's ability to detect patterns over different time spans. If the value is too high, the potential decays too fast and patterns happening over longer periods of time cannot be detected.

I wanted to find patterns on the order of a day because each entry of the data set is equivalent to one trading day. The way I handled this was by running experiments with different decay exponentials on data that I knew should have strong causal relationships on the order of a day.

The data that I used was stock price data for Shell alongside the same data shifted back by one day. This meant that the latter data would be able to predict the Shell stock increases and decreases perfectly. I then measured the strength of the causal relationship by taking the ratio of strengths of one edge direction to the other. More exactly:

$$\frac{Strength(Shell\,Shifted \rightarrow\ Shell)}{Strength(Shell \rightarrow Shell\,Shifted)}$$

Figure 25 shows how the potential decay exponential affected this quantity. It shows that the decay exponent of about 1.1 was ideal to detect causal relationships I was looking for. The choice of Shell's stock was arbitrary, it just needed to be data in the same format as the ultimate dataset but not included in it (to avoid any kind of overtraining).

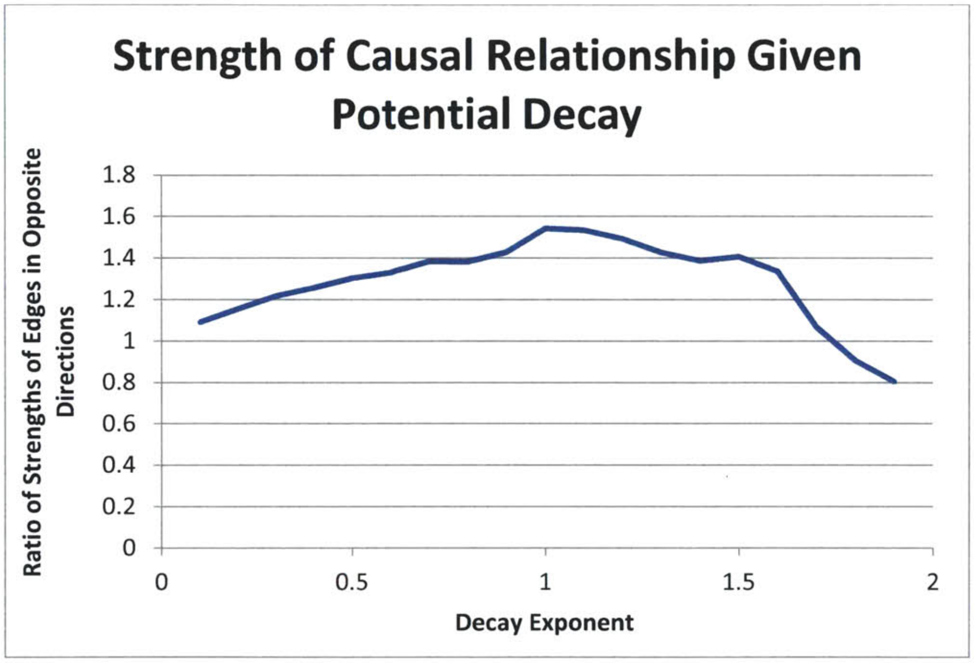

Figure 25 – Plot illustrating the how causal relationships can become clearer if the right potential decay is picked. A decay of 1.1 is ideal to find causal relationships where the interval between corresponding data is 1 (corresponding to 1 day). For decays that are larger (faster) than 1.7, the relationship is not even visible. Smaller (slower) decays make longer patterns more visible, while higher (faster) decays make patterns taking place over short periods more visible.

Table 4 shows the strongest 15 edges in the graph resulting from processing the stock price information. It is very encouraging to see that out of these top 15 strongest edges, the majority are between companies in the same industry. In fact, only three of the 15 are across different industries. The algorithm can essentially predict which companies are related simply by looking at how stocks go up and down each trading day.

Node 1	Node 2	Edge Weight
Exxon Mobile UP	Chevron UP	13.97
Chevron UP	Exxon Mobile UP	13.94
Citigroup DOWN	JPMorgan DOWN	13.05
ConocoPhillips UP	Exxon Mobile UP	12.92
Exxon Mobile UP	ConocoPhillips UP	12.89
AIG DOWN	JPMorgan DOWN	12.87
Chevron UP	ConocoPhillips UP	12.85
JPMorgan DOWN	AIG DOWN	12.81
ConocoPhillips UP	Chevron UP	12.72
JPMorgan DOWN	Citigroup DOWN	12.62
Citigroup DOWN	AIG DOWN	12.54
JPMorgan DOWN	Cardinal Health DOWN	12.46
JPMorgan DOWN	Hewlett-Packard DOWN	12.30
AIG DOWN	Citigroup DOWN	12.21
HewlettPackard DOWN	JPMorgan DOWN	12.20

Table 4 - List of strongest edges in the graph resulting from analyzing stock prices fluctuation events over the course of one year. Rows shaded in orange indicate edges between companies within the same industry. The 10 strongest connections are all between companies in the same industry.

It is interesting to see that all the 15 strongest edges are for companies going up together or going down together. This is certainly not a coincidence - this is consistently the case up to the 218th strongest edge (out of 678 edges). This may be a result of the fact that markets tend to be in sync to a large degree and tend go up and down together.

Another interesting thing to note is that the weight of the relationships is very close to symmetrical, the weight of edges in opposite directions are consistently within 5% of each other. The slight differences between symmetrical edge weights can easily be attributed to the small timing randomness that was introduced.

The top two strongest edges, for example, are "ExxonMobile UP" and "Chevron UP" and vice versa. This implies that the two events seem to be happening simultaneously, rather than being shifted in time (by a trading day for example). A delay between the two would create a causal (asymmetric) relationship between the two. The fact that these kinds of relationships are not observed is reasonable – if any strong causal relationships were detected, this would mean there existed market inefficiencies that could be exploited to predict stock prices. If there was a strong asymmetrical relationship between Exxon Mobile and Chevron, it would mean you could probabilistically predict the rise or falls of one price given the other. Unsurprisingly (and unfortunately) this is not the case.

While companies in the "Oil and Gas" and "Finance" seem to have strong relationships between each other, the same is not true of "Health Care" and "Information Technology."

One way to make it clearer which companies are related is organize the data a little differently. For each pair of companies I took the average weight of edges connecting two "UP" events or two "DOWN" events. I also took the average weight of edges connecting opposite events: one "UP" and one "DOWN". The difference between these quantities gives a measure of how in correlated the companies' stock is. I call this quantity "net correlation". Table 5 lists the top two net correlations for each company. Rows in orange indicate pairs of companies that are in the same industry.

Company 1	Company 2	Net Correlation
AIG	JPMorgan	10.35
AIG	Citigroup	9.52
Cardinal Health	McKesson	10.11
Cardinal Health	JPMorgan	9.83
Chevron	Exxon Mobile	12.63
Chevron	ConocoPhillips	11.05
Citigroup	JPMorgan	10.69
Citigroup	AIG	9.52
ConocoPhillips	Exxon Mobile	11.26
ConocoPhillips	Chevron	11.05
Exxon Mobile	Chevron	12.63
Exxon Mobile	ConocoPhillips	11.26
Hewlett-Packard	JPMorgan	9.37
Hewlett-Packard	Cardinal Health	8.93
IBM	IBM	2.42

IBM	McKesson	1.23
JPMorgan	Citigroup	10.69
JPMorgan	AIG	10.35
McKesson	Cardinal Health	10.11
McKesson	ConocoPhillips	7.66
Microsoft	Hewlett-Packard	8.52
Microsoft	AIG	7.71
UnitedHealth	Exxon Mobile	8.80
UnitedHealth	Cardinal-Health	8.74

Table 5 – List of the top two net correlations for each company. Entries shaded in orange indicate relationships between companies in the same industry.

It is encouraging to see that out of the 24 entries listed, 16 of them are for companies across the same industry, demonstrating that the algorithm is finding very plausible patterns. As for the remaining correlations, it is still possible that these represent real-life relationships but they are less straightforward to verify.

The relationships that emerged from running over stock price data have been very symmetrical relationships. To verify that this is not an artifact of the algorithm, I ran a simple experiment where I shifted one of the stocks (Exxon Mobile) over by a single day. Table 6 illustrates the relationships between the 3 oil and gas companies when Exxon Mobile was not shifted. Table 7 shows the result for the same experiment when the Exxon Mobile stock is shifted.

Node 1	Node 2	Edge Weight	Ratio of Edge Weights
Chevron DOWN	ExxonMobile DOWN	17.17	0.95
ExxonMobile DOWN	Chevron DOWN	18.03	
Chevron UP	ExxonMobile UP	20.98	0.95
ExxonMobile UP	Chevron UP	22.08	
ConocoPhillips DOWN	ExxonMobile DOWN	17.85	0.97
ExxonMobile DOWN	ConocoPhillips DOWN	18.42	
ConocoPhillips UP	ExxonMobile UP	20.15	0.97
ExxonMobile UP	ConocoPhillips UP	20.86	

Table 6 – Relationships between Exxon Mobile and both Chevron and Conoco Phillips are mostly symmetrical. If the relationships were perfectly symmetrical, the ratio of the edge weights would be 1.0. They are less than 5% away from perfectly symmetrical and the difference can be attributed to the randomness I introduced to resolve ties between simultaneous events.

Node 1	Node 2	Edge Weight	Ratio of Edge Weights
Chevron DOWN	ExxonMobile DOWN	17.23	1.41
ExxonMobile DOWN	Chevron DOWN	12.26	
Chevron UP	ExxonMobile UP	21.14	1.31
ExxonMobile UP	Chevron UP	16.19	
ConocoPhillips DOWN	ExxonMobile DOWN	18.07	1.39
ExxonMobile DOWN	ConocoPhillips DOWN	13.04	
ConocoPhillips UP	ExxonMobile UP	20.19	1.31
ExxonMobile UP	ConocoPhillips UP	15.38	

Table 7 - Relationships between Exxon Mobile and both Chevron and Conoco Phillips are not symmetrical anymore once the Exxon Mobile stock data is shifted by a day. The fact that the ratio of edge weights is different from 1.0, implies that there is a causal relationship between both of these companies and Exxon Mobile. The 3 companies tended to be in sync before, but when the Exxon Mobile was shifted forward by one day, its stock lagged in relation to the other oil companies. This gives the appearance it is the other companies stock changes that cause Exxon Mobile's stock changes to happen one day later.

Table 6 has edge weight ratios that are close to 1.0 (within less than 5%), which matches the expectation that the relationship would be symmetrical. Table 7 on the other hand, shows a causal relationship with an edge weight ratio that is over 30% larger than the perfectly symmetrical ratio of 1.0. Because the oil companies' stock prices are coincidentally related, shifting Exxon Mobile's data forward in time by one day created a causal relationship. With the data shifted, knowing the other companies stock trends gave information about Exxon Mobile's future stock trend.

This little debugging experiment confirms that the algorithm successfully picks up causal relationships between the stock prices, at least when causal relationships occur on the order of a day. If the stock data was more granular (order of hours or minutes), it is likely that the algorithm would be able to find relationships across smaller time intervals.

7 Future Work

There is a variety of next steps to do with this research. Right now the algorithm can only extract relationships between given events. It would be very interesting to give it the ability to look beyond the existing events, and allow it to create basic abstractions (ex: go from "apple", "pear", "peach" and relate them by the idea of "fruit").

7.1.1 Abstraction Nodes

One step towards creating abstractions would be to create new nodes automatically. When two nodes have a strong coincident relationship, it means that they tend to fire together (or equivalently, two events tend to happen together). If that's the case, it would make sense to create a new node that represents both these nodes. This new node could be created and connected to both of the old nodes, thereby implying a strong connection between them.

7.1.2 Potential Propagation

The idea of potential propagation is for the firing of one node to influence the firing of another related node. A potential threshold could be set above which the node would spontaneously "fire" (in the same sense that a neuron fires). So, in this model, if there is a node A and its close neighbor N, when A fires the potential of N would increase. If enough neighbors fire, that might be enough to raise the potential high enough to cause N to fire. The ultimate goal here would be to create more indirect associations. Imagine that nodes representing "round", "small", "lime color", "bouncy" all fire. It would be nice to have the related node "tennis ball" fire as well. It would serve as a first step towards abstraction.

8 Conclusion

Machine learning has traditionally focused on data that is isolated from its context in time. The algorithm presented in this thesis offers a new method to capture timing patterns in series of events. It has shown to be very robust to noisy inputs, which makes it promising in its usefulness to process sensor data. It is also an online algorithm, meaning that it is able to process data on the fly as it receives it.

There are multiple potential applications for this algorithm. Essentially any data that happens over time could benefit from analysis with this algorithm. I showed that it can find patterns in artificially generated data, as well as in stock price events.

I truly hope more research is done in this field. I believe there are some great insights to be made in this area. This research emulates a neuroscience phenomenon that is thought to have the incredible potential to carry out basic learning in the brain, and it would be fascinating to see how far this approach can be taken. Achieving the goal of mimicking the pattern-finding abilities of the brain is still far out, but I believe that taking inspiration from the new processes recently being discovered in neuroscience is the right approach.

9 Contributions

- Implemented a biologically inspired algorithm to capture relationships in time-based data.
- Showed by simulation that relationships can be extracted even in situations of extreme noise.
- Applied algorithm to a simple story text by turning words into events. Found that subjectively the patterns extracted are relevant to the story.
- Applied algorithm to stock price data. Found that companies within the same industry tend to have stronger coincident relationships between each other.