Feynman Machine: A Novel Neural Architecture for Cortical And Machine Intelligence

Eric Laukien, Richard Crowder, Fergal Byrne*

eric@ogmacorp.com, rcrowder@ogmacorp.com, *fergal@ogmacorp.com
Ogma Intelligent Systems Corp.
4425 Military Trail, Ste. 209, Jupiter, FL 33458

Abstract

Developments in the study of Nonlinear Dynamical Systems (NDS’s) over the past thirty years have allowed access to new understandings of natural and artificial phenomena, yet much of this work remains unknown to the wider scientific community. In particular, the fields of Computational Neuroscience and Machine Learning rely heavily for their theoretical basis on ideas from 19th century Statistical Physics, Linear Algebra, and Statistics, which neglect or average out the important information content of time series signals generated between and within NDS’s. In contrast, the Feynman Machine, our model of cortical and machine intelligence, is designed specifically to exploit the computational power of coupled, communicating NDS’s. Recent empirical evidence of causal coupling in primate neocortex corresponds closely with our model. High-performance software implementation has been developed, allowing us to examine the computational properties of this novel Machine Learning framework.

Introduction

Computer-assisted and numerical methods have only recently allowed us to study the properties of Nonlinear Dynamical Systems. Since Lorenz discovered his famous strange attractor (Lorenz 1963), applied mathematicians have steadily discovered an unexpected world of structure in the behaviour and interactions of such systems (Kantz and Schreiber 2004; Strogatz 2014). In particular, techniques based on delay embedding of trajectories on manifolds have been used to identify the true causal structure of numerous natural and artificial phenomena, either in the absence of classical correlation or despite spurious correlations (Sugihara et al. 2012). These techniques rely in the main on the Theorem of Floris Takens (Takens 1981), which proves (given certain assumptions) that a reconstruction of sufficient dimensionality is diffeomorphic (topologically equivalent) to the system generating a time series.

In simple terms, the mere representation in a computer or brain area, of a vector of lagged values \((x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-k})\) from the time series \(x_t\), will trace out a trajectory in \(k\)-dimensional space which is to all intents and purposes the same thing as the real NDS which produces the time series, in the sense that forecasts of the futures of the representation and that of the real system have the same mathematical properties. Importantly, this information transfer requires no knowledge of the underlying rules governing the evolution of the real system. This notion is extended, in brains and intelligent machines, to the interactions between brain regions and processing modules respectively.

The Feynman Machine (Figure 1) is a network or hierarchy of paired encoders and decoders, which operate as NDS’s and communicate spatiotemporal sparse distributed representations (SDR’s). It is superficially similar to a stacked autoencoder ladder network (Rasmus et al. 2015), but the encoders are designed to incorporate the past history of their activity and thus produce encodings which are predictive of the future evolution of the time series input.

In the hierarchy, inputs (which can be sensory or sensorimotor) are fed into the bottom layer encoder, which produces a sparse binary SDR to be passed up to the next higher layer, a process which continues up the hierarchy. Each decoder takes the output of its paired encoder (which forms a lateral prediction signal) and combines it with the output of the next higher layer’s decoder (a top-down feedback/context signal). At the bottom, the decoder outputs a prediction of the next (sensory or sensorimotor) input in the time series. During learning, the decoder compares its previous prediction with the actual input to its paired encoder, and this error vector is used to train both encoder and decoder.

Our model was originally inspired by previous work on a theory of neocortical function based on communicating NDS’s (Byrne 2015). There is now convincing evidence from empirical Neuroscience that this process is indeed occurring in primate neocortex (Tajima et al. 2015). Interestingly, the authors of this study used the methods of nonlinear time series analysis to extract the causal structure of interactions among patches of neocortex in the monkey brain. In more recent work, (Tajima and Kanai 2017), they have proposed a role for delay embedding dimensions in Integrated Information Theory (Tononi et al. 2016).

The artificial Feynman Machine is described in full detail in (Laukien, Crowder, and Byrne 2016), so here we will briefly describe its most important characteristics in the Methods section. The Results section describes some initial experimental evidence of self-organised unsupervised learning of high-dimensional spatiotemporal structure, in the case of colour video prediction. Our Conclusion includes some...
comments on ongoing and future work.

Source code for the architecture and experiments detailed here is available for non-commercial use at https://github.com/OgmaCorp.

Methods

We briefly describe the operation of an artificial Feynman Machine, which is a hierarchical architecture (Figure 1) of paired encoders and decoders, each pair forming a spatiotemporal predictive autoencoder module. A variety of encoder designs can be used, as long as they have certain properties: a) they must incorporate past history in their processing, and b) the output must be a nonlinear function of the input. Our experiments indicate that the best-performing encoders are those that produce sparse binary outputs; in this paper we chose the Delay Encoder, described below.

The decoder in each pair takes the top-down feedback from higher layers and the output of its paired encoder, and attempts to reconstruct an output, usually the next input in the time series. We have found that a simple linear decoder (which uses a weight matrix for each of its inputs) is sufficient for sequence learning. The decoder learning algorithm is simple perceptron learning on each weight matrix.

An Example Encoder Design: Delay Encoder

The Delay Encoder takes a vector x of (scalar or binary) inputs, passes them through a linear weight matrix W, combines the summed results with a bias b (producing the stimulus vector s), and applies a nonlinearity (a Rectified Linear Unit or ReLU function), giving the activation vector a. So far, this resembles the operation of a standard artificial neural network. The temporal memory is provided by adding a decayed copy of the activations from the previous timestep \( \hat{a} \) to the stimulus vector before entering the ReLU (Figure 2). The activation vector is then passed through a \( k \)-sparse inhibition stage, producing a sparse binary vector of “firing” units \( z \), which is the output of the encoder. The activations of any firing units are set to zero for use in the next timestep.

\[
  s_t = W x_t \tag{1}
\]

Update activation vector \( a \) element-wise: preserve only the previous non-active activations, add the stimulus from Eq. (1), subtract the biases, and apply a rectified linear unit function (ReLU):

\[
  a_t = max(0, (J_n - z_{t-1})a_{t-1} + s_t - b_{t-1}) \tag{2}
\]

Choose the top \( k \) activation elements:

\[
  \Gamma_k = supp_k(a_t) \tag{3}
\]

Set the appropriate elements to 1 or 0 to generate the encoder output \( z \):

\[
  z_t = \delta(i \in \Gamma_k) \tag{4}
\]

Determine the exponentially decaying traces of the inputs \( x_t \) and outputs \( z_t \):

\[
  \hat{x}_t = \lambda \hat{x}_{t-1} + x_t \tag{5}
\]

\[
  \hat{z}_t = \lambda \hat{z}_{t-1} + z_t \tag{6}
\]

Learning in the Delay Encoder combines a number of previously studied ideas in machine learning. In particular, a form of Real-Time Recurrent Learning, or RTRL (Williams and Zipser 1989), which uses local information to maintain a trace of past activity and inputs, is combined with a form of Spike-time Dependent Plasticity, or STDP (Sjstrm and Gerstner 2010; Markram, Gerstner, and Sjstrm 2012; Gilson, Burkitt, and Van Hemmen 2010), which cross-correlates inputs and output states and traces over time. Each

![Figure 1: The Sparse Predictive Hierarchy Feynman Machine (Laukien, Crowder, and Byrne 2016).](image)

![Figure 2: Activation diagram of the Delay Encoder.](image)
connection in the delay encoder computes a STDP function that indicates the preferred temporal direction the connection would like the corresponding cell to take. The STDP values of all connections are averaged together to get the total temporal direction the cell should move in order to maximize the number of inputs the cell is predicting. Once this direction is obtained, the cell can be updated to fire either sooner or later depending on the desired temporal direction. This is done by maintaining another set of per-connection traces, which accumulate inputs and are reset to 0 when the cell emits a spike. The final update to the connection weights is then a product of the desired temporal direction and the eligibility trace for a connection.

Maintain a trace matrix $T$ for previous activity taking reset into account (derived from RTRL (Williams and Zipser 1989)):

$$T_t = \lambda T_{t-1} \odot (J_{n,m} - Z_{t-1}) + X_{t-1}$$  \hspace{1cm} (7)$$

where $Z_{ij} = z_i$, $X_{ij} = x_j$ and $J_{ij} = 1$, $i = 1..n$, $j = 1..m$, when there are $n$ outputs and $m$ inputs. Each element $T_{ij}$ contains a decaying memory of all inputs $x_j$ since the last time the hidden cell $z_i$ fired, the trace is erased to zero each time cell $i$ fires.

Determine “importance” $Y$ of the feed forward weights:

$$Y_{ij} = e^{W_{ij}}$$ \hspace{1cm} (8)

Calculate the importance-weighted spike time dependent plasticity (STDP) value for each connection:

$$P = Y \odot (z_{t-1} \odot \tilde{x}_t - z_{t-1} \odot x_{t-1})$$ \hspace{1cm} (9)$$

The second term in Eq. (9) is the STDP matrix, formed as the difference of two outer products. The first outer product matrix $z_{t-1} \odot \tilde{x}_t$ cross-correlates the previous firing pattern with the input trace vector, and the second, $z_{t-1} \odot x_{t-1}$ cross-correlates the output traces with the previous input vector.

Determine the temporal direction using an average of the connection STDPs:

$$d_i = \frac{\sum_j P_{ij}}{\sum_j Y_{ij}}$$ \hspace{1cm} (10)$$

Finally, update the weights:

$$\Delta W_{ij} = \alpha (d_i T_{ij})$$ \hspace{1cm} (11)$$

To improve stability, normalize $W$ such that each node’s connection weights sum up to 1:

$$W_{ij} = \frac{W_{ij}}{\|W_i\|_2}$$ \hspace{1cm} (12)$$

Update the biases:

$$\Delta b_t = \beta (s_{t-1} - b_{t-1})$$ \hspace{1cm} (13)$$

Source code, documentation and a white paper on the Delay Encoder algorithms are available at https://github.com/OgmaCorp.

Results

The Feynman Machine is capable of learning to represent the hierarchical spatiotemporal structure of high-velocity, high-dimensional data such as streaming video, and to use the sparse representations at all levels to generate the imagined future of that data. In order to demonstrate this, we trained a system with clips of video of natural and artificial scenes. After fewer than twenty presentations, the hierarchy was capable of replaying an entire clip with excellent subjective fidelity, simply by providing the first few frames and then running the system off its own predictions (Figure 3).

The hierarchy used in this case was 8 layers (encoder-decoder pairs) of 512x512 units at the input layer, decreasing to 64x64 units in the top layer, trained on a downsampled video for 16 presentations. We used a desktop PC with a consumer GPU, training time was under 3 minutes. Videos of this and similar experiments can be accessed at https://www.youtube.com/ogmaai.

Figure 3: Video Prediction Experiment. An 8-second HD video sequence (example frame top left) was downscaled and fed as input to the hierarchy 16 times. The network was then fed the beginning of the sequence, and subsequently its predictions were fed back as input, producing a recalled sequence, of which 37 successive 400x300 pixel frames are shown.

We also apply the Feynman Machine architecture to the Noisy Lorenz Attractor as studied in (Hamilton, Berry, and Sauer 2016). The classic Lorenz system (Lorenz 1963) is made stochastic by adding noise terms $\eta$ to each equation in the system. This dynamic noise has variance $\sigma^2 = 0.8$ in each dimension, causing the resulting time series to jump to nearby trajectories of the attractor (Figure 4 left). A lagged time series vector $(x_t, x_{t-\tau}, x_{t-2\tau})$ is produced from the $x$-coordinates for each time $t$, resulting in a 3D delay embedded reconstruction (Figure 4 centre), where the lag $\tau$ is 13 timesteps. Each observation $x_t$ is further perturbed by adding a large amount of observation noise, with variance $\sigma^2 = 20$. The resulting observation time series is then fed to the Feynman Machine (a 12-layer hierarchy, each layer has 48x48 units). Starting from a randomly initialised network, the system’s predictions converge to within an RMSE of approximately 2.7 of the next true $x$-coordinate, comparable
with results reported in (Hamilton, Berry, and Sauer 2016). This convergence occurs within fewer than 1000 steps of noisy observations.

A recording of this experiment, from which Figure 4 was taken, is available at https://youtu.be/fUHnEzPqCJo and includes a visualisation of the noisy observations. The system runs at c.60 steps per second on a consumer laptop with a standard GPU.

Figure 4: Noisy Lorenz Attractor (based on the system studied in (Hamilton, Berry, and Sauer 2016). The signal source (left) is used to reconstruct a delay embedding (centre), and noise is added (not shown). The Feynman Machine forms a prediction (right) based on the noisy observations.

A high-performance library, OgmaNeo, has been built using C++ and OpenCL, and runs on CPU-only or compatible CPU/GPU PC’s. Bindings for C++, Python, and Java are publicly available, along with source code for experiments described here and several others. Source code for both the library and numerous experiments can be accessed at https://github.com/OgmaCorp.

**Conclusion**

We have described a novel neural architecture which exploits the information communication power of coupled dynamical systems, using a hierarchical network structure inspired by the mesoscale connectome of the mammalian neocortex. Our currently best-performing encoder design, the Delay Encoder, is capable of adaptively learning to model high-velocity, high-dimensional data streams such as natural video sequences, and forms excellent predictions based on its model.

The system has also been tested on a number of other spatiotemporal tasks, as detailed in (Laukien, Crowder, and Byrne 2016). The artificial Feynman Machine is a very new neural network architecture. While its function is based on solid theory in Dynamical Systems, and its structure is inspired by Neuroscience, the capability, range and limitations of cognitive systems built on these principles are only beginning to be explored.

Current work at this time is focused on applying the architecture in a Reinforcement Learning context. Initial exercising of an agent using the Feynman Machine as its core learning component has been carried out on the OpenAI Gym environments (OpenAI 2016), with encouraging results. We are also exploring the abilities of the architecture in sequence identification, anomaly detection, and vocalisation.

Future work includes exploration of further opportunities for cross-fertilisation with the fields of Nonlinear Dynamical Systems - in particular, causation analysis and detection (Sugihara et al. 2012), stochastic partial differential equations (Ovchinnikov and Wang 2015; Ovchinnikov et al. 2016), and hardware implementations (Appeltant 2012) - and Neuroscience (Tajima et al. 2015; Hawkins and Ahmad 2015).

**References**


