Energy-Based Temporal Neural Networks for Imputing Missing Values

Philemon Brakel and Benjamin Schrauwen

Department of Electronics and Information Systems, Ghent University,
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
{philemon.brakel,benjamin.schrauwen}@elis.ugent.be

Abstract. Imputing missing values in high dimensional time series is a difficult problem. There have been some approaches to the problem [11, 8] where neural architectures were trained as probabilistic models of the data. However, we argue that this approach is not optimal. We propose to view temporal neural networks with latent variables as energy-based models and train them for missing value recovery directly. In this paper we introduce two energy-based models. The first model is based on a one dimensional convolution and the second model utilizes a recurrent neural network. We demonstrate how ideas from the energy-based learning framework can be used to train these models to recover missing values. The models are evaluated on a motion capture dataset.

Keywords: neural networks, energy-based models, time series, missing values, machine learning, optimization

1 Introduction

Many interesting datasets in fields like meteorology, finance, and physics are high dimensional time series. To make optimal use of such data, it is often necessary to impute missing values. Unfortunately, this can be a very challenging task because many multivariate time series are generated by complex non-linear processes. Simple techniques like nearest neighbour interpolation treat the data as independent, and ignore the temporal component.

Recently, a model for high dimensional non-linear time series called the Conditional Restricted Boltzmann Machines (CRBMs) [11] has been used to reconstruct motion capture data. This model is trained to learn a density over a frame in a sequence, conditioned on a fixed number of previously observed frames. Since inference in these models is intractable, sampling or mean-field approximations are required.

When the task is to reconstruct some missing values based on the remaining observed data, it is only this conditional inference that needs to be optimized. If an optimization method is used for inference, we don't have to care about the density the model assigns to regions that are far away from the data, because we will not start the approximate inference procedure in those regions anyway.

By using temporal energy models that are directly trained to reconstruct a subset of the input data conditioned on the remaining observations, we circumvent some of the problems probabilistic models have to deal with. Moreover, these kind of models make it easier to use features that are generated by complicated functions that can be chosen to represent prior knowledge about the problem domain. To demonstrate how temporal energy-based models can be trained to reconstruct sequential data, we will train both a convolutional energy-based model and one that is based on a recurrent neural network.

Dynamic factor graphs [8] are temporal energy-based models as well and have also been used to reconstruct motion capture data. This approach is somewhat similar to ours but like the CRBM, the model was trained to maximize the data likelihood.

2 Energy-Based Models

An Energy-Based Model (EBM) [6] defines a function $E(\cdot)$ that maps an observation vector $\mathbf{v}$ to an energy value. Many probabilistic and deterministic models can be seen as models that capture dependencies between variables by assigning a score to their joint configuration. For probabilistic models, this score is the negative log-likelihood and for Principal Component Analysis, for example, this is the reconstruction error. Recently, energy-based models have been used to train deep models of images [9].

To model complex processes, it is common practice to introduce latent or 'hidden' variables that represent interactions between the observed variables. This leads to an energy function of the form $E(\mathbf{v}, \mathbf{h})$, where $\mathbf{h}$ are the hidden variables, which need to be marginalized out to obtain the energy value for $\mathbf{v}$. This summation (or integration) can be greatly simplified by designing models such that the hidden variables are conditionally independent given an observation. A model that satisfies this independence condition is the Restricted Boltzmann Machine (RBM) [3, 4] which is a often used to build deep belief networks [5].

The energy-based framework allows one to come up with new architectures or optimization algorithms for statistical models without being constrained by probabilistic assumptions. Optimizing the likelihood makes sense if the goal is to learn a good statistical model of the data. Unfortunately, the maximum likelihood objective is often intractable for more complex architectures because of the requirement to compute a normalization constant. This intractability has led to the development of learning algorithms that rely on approximations of maximum likelihood learning like contrastive divergence [4] and pseudo-likelihood learning [1]. However, in an energy-based framework there is no reason to rule out alternatives for the likelihood objective that circumvent the normalization problem.

In the current paper we are not interested in the estimation of a good model of the data itself but in a model that performs well at restoring missing values given other values that have been observed. We will describe a way to discrim-

inatively train energy-based models for this task that is inspired by work on image processing [2].

3 The Convolutional Energy-Based Model

We will call the first model we will investigate the Convolutional Energy-Based Model (CEBM). Let $\mathbf{V}$ be a sequence of data vectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_T\}$ indexed by time index t. The CEBM has an energy function that is defined as a one dimensional convolution over a sequence of data combined with a quadratic term and is given by

$$E(\mathbf{V}, \mathbf{H}) = \sum_{t=1}^{T} \left(\frac{\|\mathbf{v}_t - \mathbf{b}_v\|^2}{2\sigma^2} - \mathbf{h}_t^\mathsf{T} \mathbf{g}_{\text{conv}}(\mathbf{V}, t) \right), \tag{1}$$

where $\mathbf{b}_v$ is a vector with biases for the visible units and $\mathbf{H}$ are a set of binary hidden units. The function $\mathbf{g}_{\text{conv}}(\cdot)$ is defined by

$$\mathbf{g}_{\text{conv}}(\mathbf{V}, t) = \mathbf{W}(\mathbf{v}_{t-k} \oplus \cdots \oplus \mathbf{v}_t \oplus \cdots \oplus \mathbf{v}_{t+k}), \tag{2}$$

for $k < t < T - k$ and equal to zero for all other values of t. The matrix $\mathbf{W}$ contains trainable connection weights and the operator $\oplus$ signifies concatenation. The value k determines the number of frames that each hidden unit is a function of. This model has the same energy function as the convolutional RBM [7] and the main difference is the way in which training and inference are done. See Fig. 1a for a graphical representation of the convolutional architecture.

To compute the free energy of a sequence $\mathbf{V}$, we have to sum over all possible values of $\mathbf{H}$. Fortunately, because the hidden units are independent given the output of the function $\mathbf{g}_{\text{conv}}$, the total free energy can be calculated analytically and is given by

$$E(\mathbf{V}) = \sum_{t}^{T} \left(\frac{\|\mathbf{v}_t - \mathbf{b}_v\|}{2\sigma^2} - \sum_j \log(1 + \exp(g_{\text{conv}j}(\mathbf{V}, t)) \right). \tag{3}$$

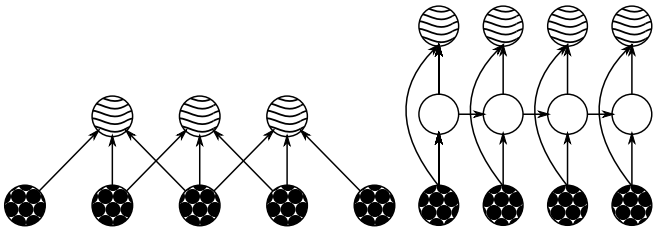
The index j points to the jth hidden unit.

The gradient of the free-energy function with respect to the input of the network is given by the negative sigmoid function:

$$\frac{\partial E(\mathbf{V})}{\partial g_{\text{conv}j}(\mathbf{V}, t)} = -(1 + \exp(g_{\text{conv}j}(\mathbf{V}, t))^{-1}. \tag{4}$$

The chain rule can be used to calculate the derivatives with respect to the parameters of the network. The derivative of the free energy with respect to the input variables is defined as

$$\frac{\partial E(\mathbf{V})}{\partial \mathbf{v}_t} = \frac{\partial E(\mathbf{V})}{\partial \mathbf{g}_{\text{conv}}(\mathbf{V}, t)} \frac{\partial \mathbf{g}_{\text{conv}}(\mathbf{V}, t)}{\partial \mathbf{v}_t} + \frac{\mathbf{v}_t - \mathbf{b}_v}{\sigma^2}. \tag{5}$$



(a) Convolutional structure. (b) Recurrent structure.

Fig. 1: The two model structures that are used in this paper. The wavey circles represent the hidden units, the circles filled with dots the visible units and empty circles represent deterministic functions. The time dimension runs from the left to the right and each circle represents a layer of units.

4 The Recurrent Energy-Based Model

The second model we propose in this paper is the Recurrent Energy-Based Model (REBM). The energy function of the REBM is again defined by equation 1 but $\mathbf{g}_{\text{conv}}$ is replaced by

$$\mathbf{g}_{\text{rec}}(\mathbf{V}, t) = \mathbf{A}\mathbf{x}_t + B\mathbf{v}_t + \mathbf{b}_r \tag{6}$$

$$\mathbf{x}_t = \tanh(\mathbf{C}\mathbf{x}_{t-1} + \mathbf{D}\mathbf{v}_t + \mathbf{b}_x) \quad 1 < t \leq T, \tag{7}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are matrices with network connection parameters and $\mathbf{b}_r$ and $\mathbf{b}_x$ are the bias vectors of, respectively, the output variables and the hidden state variables $\mathbf{X}$. This function can be replaced by any differentiable function of choice. See Fig. 1b for a graphical representation of the recurrent architecture. This model is similar to the Recurrent Temporal Restricted Boltzmann Machine [10] but in our model the units that define the energy are in a separate layer and the visible variables are not independent anymore given the hidden variables. Simple Gibbs sampling is not possible anymore but this will also not be required.

5 Training for Missing Value Recovery

Given a sequence $\mathbf{V}$, let Ω be the set of indices that point to elements of data vectors that have been labeled as missing. For real-valued data, a sound objective to minimize, is the mean squared error between the values we predicted $\hat{\mathbf{V}}$ and the actual values $\mathbf{V}$:

$$L = \frac{1}{2} \sum_{j \in \Omega} (\mathbf{V}_j - \hat{\mathbf{V}}_j)^2. \tag{8}$$

To obtain a prediction, we use an optimization procedure that uses the gradient with respect to $\{\hat{\mathbf{V}}_j | j \in \Omega\}$ to find a set of values that minimize this energy.

Running this optimization procedure until convergence takes very long and it can also not be guaranteed that a global optimum will be found. Recently, it has been proposed to run the optimizer for just a couple of steps and compute gradients with respect to the optimization procedure itself [2]. The rationale behind this is that we train the model to predict the correct values given that we consistently use the same optimizer for inference. Parts of the energy landscape that the optimizer never reaches will not affect the performance of the model.

To train our models, we first sampled a set of random indices to label as missing for every sequence. After that, we used a couple of steps of gradient descent with step size λ on the free energy to obtain predictions. Subsequently, the gradient of the mean squared error loss L with respect to the parameters is computed by propagating errors back through the gradient descent steps in holder variables $\bar{\mathbf{V}}$ and $\bar{\theta}$ that in the end will contain the gradient of the error with respect to the input variables and the parameters of the model (we use θ as a placeholder for both the biases and weights of the model). The gradient with respect to the parameters is used to train the model with stochastic gradient descent. Hence, the model is trained to improve the predictions that the optimization procedure comes up with directly. To perform the backpropagation steps, second order derivatives are required. However, the product of these with a vector can be estimated efficiently using finite differences and only requires routines that provide the gradient of the free energy with respect to the input variables and model parameters [2]. See Algorithm 1 for more details. In the algorithm we omitted that all references to the data only concern the missing values to avoid cluttered notation.

Algorithm 1 Compute the error gradient through gradient descent

Initialize $\hat{\mathbf{V}}^0$
for $k = 0$ **to** $N - 1$ **do**
  $\hat{\mathbf{V}}^{k+1} \leftarrow \hat{\mathbf{V}}^k - \lambda \nabla_{\hat{\mathbf{V}}} E(\hat{\mathbf{V}}^k; \theta)$
end for
$\bar{\mathbf{V}}^N \leftarrow \nabla L(\hat{\mathbf{V}}^N) = \hat{\mathbf{V}}^N - \mathbf{V}$
$\bar{\theta} \leftarrow \mathbf{0}$
for $k = N - 1$ **to** 0 **do**
  $\bar{\theta} \leftarrow \bar{\theta} - \lambda \frac{\partial^2 E(\hat{\mathbf{V}}^k; \theta)}{\partial \theta \partial \mathbf{V}^\intercal} \bar{\mathbf{V}}^{k+1}$
  $\bar{\mathbf{V}}^k \leftarrow \bar{\mathbf{V}}^k - \lambda \frac{\partial^2 E(\hat{\mathbf{V}}^k; \theta)}{\partial \mathbf{V} \partial \mathbf{V}^\intercal} \bar{\mathbf{V}}^{k+1}$
end for
Return $\nabla_\theta L = \bar{\theta}$

Adding artificial corruption to the data and training a model to reconstruct it is similar to the way denoising autoencoders are trained [12]. The most important difference is that our models only focus on recovery of the missing values given the observed variables and not on reconstructing both.

6 Experimental Evaluation

6.1 Data

The data consisted of three motion capture recordings from 17 marker positions represented as three 49-dimensional sequences of joint angles. The data was downsampled to 30Hz and the sequences consisted of 3128, 438, and 260 frames. The first sequence was used for training, the second for validation and the third for testing. The sequences were derived from a subject who was walking and turning and come from the MIT dataset provided by Eugene Hsu[1]. The data was preprocessed by Graham Taylor [11] using parts of Neil Lawrence's Motion Capture Toolbox.

6.2 Implementation Details

Good settings of the hyper-parameters of the models were found by doing a random search on the parameter space, followed by some manual fine-tuning to find the settings that led to minimal error on the validation set. After this, the models were trained again on both the train and the validation data with these settings. Since the CEBM is very parallel in nature, we used a GPU for the convolution. All models were trained on randomly selected mini batches of 140 frames.

Both the CEBM and the REBM were trained by labeling random sets of dimensions as missing. The number of missing dimensions was sampled uniformly. The specific dimensions were then randomly selected without replacement. The duration of the data loss was sampled uniformly between 60 and 125 frames. This adds some bias towards situations in which the same dimensions are missing for a certain duration. This seems to be a sensible assumption in the case of motion capture data and it is an advantage of the training method that it is possible to add this kind of information.

The CEBM had 200 hidden units and its window size was set to 15 time frames. The model was trained for 50000 iterations with a linearly decreasing learning rate initialized at .005. The inference optimization was set to three iterations with a step size of .2. The REBM had 200 units in the recurrent layer and 50 hidden units. It was trained for 25000 iterations with a linearly decreasing learning rate that started at .001. The step size of the optimization procedure was .2 and 5 iterations were done for inference. The CRBM had 200 hidden units, used the 10 previous frames as context and was trained using contrastive divergence with three sampling steps. The model was trained for 50000 iterations with a linearly decaying learning rate initialized at 10^{-4}.

Nearest neighbour interpolation was done by selecting the frame from the train set with minimal Euclidean distance to the test frame according to the observed dimensions. The distances were computed in the normalized joint angle space.

[1] `http://people.csail.mit.edu/ehsu/work/sig05stf/`

6.3 Evaluation

To evaluate the models, a set of dimensions was removed from the test data for a duration of 120 frames (4 seconds). This was done for either the markers of the left leg or the markers of the whole upper body (everything above the hip). Because the offset of this gap was chosen randomly and because the CRBM had a stochastic inference procedure, this process was repeated 500 times to obtain average mean squared error values for both the train and the test data.

Both the energy-based models used the same inference procedure as they used during training to fill in the missing values. The CRBM was used in a generative way by conditioning it on the samples it generated at the previous time steps while clamping the observed values and only sampling those that were missing. Preliminary results showed that this led to similar results as the use of mean-field or minimization of the model's free energy to do inference.

Table 1 shows the mean squared error between the reconstructed dimensions of the data and their actual values. The convolutional and recurrent models clearly outperform the CRBM and nearest neighbour interpolation on the reconstruction of the left leg. The CEBM has the best performance but a comparison of the train and test error scores suggests that the REBM appears to display better generalization properties. The CRBM seems to suffer most from over fitting. Surprisingly, the trained models all seem to perform equally well for reconstructing the whole upper body and far better than nearest neighbour interpolation.

Table 1: Results in mean squared error on the motion capture data.

	Left leg		Upper body	
	Train	Test	Train	Test
CEBM	.18	.29	.45	.45
REBM	.24	.33	.47	.44
CRBM	.23	.42	.46	.47
Nearest neighb.	N/A	.45	N/A	.76

7 Discussion

In this work we proposed two models for imputing missing values in time series and a way to train them for that task. Both models outperformed the baseline models on a motion capture task and we conclude from this that the approach is promising and worthy of further research. By showing that neural architectures can be trained to impute missing values without the need for approximations to complicated probability distributions, if they are cast in an energy-based learning framework, we hope to inspire more work in this area.

A downside of the training methodology is that it introduces a couple of extra parameters that need to be tuned. This could be solved by replacing the gradient descent inference procedure with an optimizer that doesn't require a step size parameter. Another limitation is that the gradient of the energy can only be computed for continuous data but for many datasets this is not a problem. Future work should also investigate the success of the methods on data sets from other domains, like video, music or geophysical records.

Acknowledgments. This article was written under partial support by the FWO Flanders project G.0088.09.

References

1. Besag, J.: On the statistical analysis of dirty pictures. Journal of the Royal Statistical Society B-48, 259–302 (1986)
2. Domke, J.: Generic methods for optimization-based modeling. Journal of Machine Learning Research - Proceedings Track 22, 318–326 (2012)
3. Freund, Y., Haussler, D.: Unsupervised Learning of Distributions on Binary Vectors Using Two Layer Networks. Tech. rep., Santa Cruz, CA, USA (1994)
4. Hinton, G.E.: Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation 14(8), 1771–1800 (2002)
5. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
6. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. In: Bakir, G., Hofman, T., Schölkopf, B., Smola, A., Taskar, B. (eds.) Predicting Structured Data. MIT Press (2006)
7. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 609–616. ICML '09, ACM, New York, NY, USA (2009)
8. Mirowski, P., LeCun, Y.: Dynamic factor graphs for time series modeling. In: Proc. European Conference on Machine Learning (ECML'09) (2009)
9. Ngiam, J., Chen, Z., Koh, P.W., Ng, A.: Learning deep energy models. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning. pp. 1105–1112. ACM, New York, NY, USA (June 2011)
10. Sutskever, I., Hinton, G.E., Taylor, G.W.: The recurrent temporal restricted boltzmann machine. In: Advances in Neural Information Processing Systems. vol. 21. MIT Press, Cambridge, MA (2008)
11. Taylor, G.W., Hinton, G.E., Roweis, S.: Modeling human motion using binary latent variables. In: Advances in Neural Information Processing Systems. vol. 19. MIT Press, Cambridge, MA (2007)
12. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine learning. pp. 1096–1103. ICML '08, ACM, New York, NY, USA (2008)