

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

Democratic Echo State for Music Improvisation

Anonymous Author(s)

Affiliation

Address

email

Abstract

We propose an algorithm for music improvisation based on time-series prediction. Our learning algorithm consists of echo state networks trained on subsets of data and combined via majority vote to obtain a stronger classifier. The classifier is trained first on music drawn from MIDI files. Once the classifier has been trained, we will use this classifier to predict a probability distribution over the next note in a musical sequence. We will then draw a note from this distribution and feed it back into the classifier. Though the musical improvisations that our model produces is difficult to quantify, we are able to measure how effective our model is in doing time series prediction. Our Ensemble classifier only offers modest improvements in terms of classification accuracy over the single Echo State Machine. However, it still proves effective in music improvisation, learning and harmonizing musical phrases for use in musical improvisation.

The philosopher Leonard Meyer, in *Emotion and Meaning in Music* proposed his famous triple equation expectation = emotion = meaning[7]. Music exists, he claims, as a closed system - it does not make reference to objects outside of itself, only itself, other pieces of music. The principle emotional content of music, therefore he argues, is in only unfolding of musical possibilities in one piece in relation to other music. The listener listens to the music with some a certain degree of uncertainty, in statistical jargon a prior based on past musical compositions, stylistic elements, and so forth. The uncertainty of what comes next, Meyer claims, triggers a sense of apprehension and anxiety, which may be released or built up depending on the song's progression. In either case, there is a deviation from the expected progression which triggers an emotional response - the "meaning" of the musical composition[7, 11]. Music and emotion are linked, in other words, by manipulation of the predictive capacity of the human mind.

It seems plausible, therefore, that a system for music improvisation should involve two elements - *pattern recognition*, to learn the priors implicit in musical style, and *randomness* - an element of surprise. A sequence too coherent would bore the listener, a sequence too random would lack harmony, resulting in frustration. It is crucial that a musical generator should be able to correctly forecast music by detecting patterns within it, but have a sufficient amount of randomness to permit surprise and suspense.

1 Previous Work

Statistical models for music improvisation can be seen as a special case of Time Series prediction. Let us begin a formal treatment of the problem by seeing music as a sequence of events (x_1, x_2, \dots, x_n) . Each event can be seen as a "musical event", for the purpose of this project it will be (note, duration, delay). Let us assume that we are given a black box which can do the following. Given a sequence of events, (x_1, x_2, \dots, x_n) , it produces a probability distribution over what the next state should be $P(x_{n+1}|x_n, \dots, x_1)$. Now we draw a sample

054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

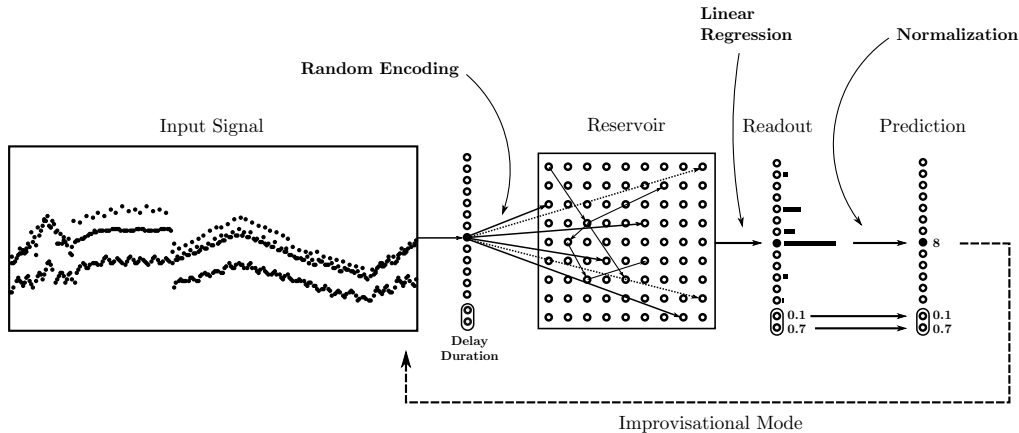


Figure 1: The echo state machine architecture.

from this probability distribution, x_{n+1} . We then feed the sequence $(x_1, x_2, \dots, x_n, x_{n+1})$ into our model, and repeat the process. The output then can be seen as a sequence of musical events which forms a melody.

The earliest models for music generation has been the Markov chain[2]. This assumes that music has a short memory memory, or is Markov in nature. Given a sequence of states (x_1, x_2, \dots, x_n) the probability of the next state, given the previous state depends only on k states in the past. Stated formally $P(x_{n+1}|x_n, \dots, x_1) = P(x_{n+1}|x_n, \dots, x_{n-k})$. Unfortunately, this model suffers the curse of dimensionality as the number of states grow exponentially with k , preventing the model from being explored beyond a small number of states.

More sophisticated models have been developed since. Recurrent Neural Networks,[3] including networks with Long Short Term Memory which hope to capture long range correlations and discard the short-term memory assumption . Stochastic Factor Oracles [13] are clever data structures which memorize musical sequences and are able to generalize the Markov Chain to large numbers of sequences. The PAQ [10] framework uses an internal model of data compression and a combination of a large Ensemble of predictors to predict the next state. Echo State Machines have also been employed to capture repeating motifs in music directly from sound data. Democratic Liquid State Machines[12] takes it’s inspiration from the last two paradigms - echo state machines and ensembles and combines them in a framework for time series prediction. This will also be the inspiration behind our project.

2 Echo State Ensembles

There is an old Indian parable in which six blind men are gathered around an elephant and asked to describe what they touch. Each man feels a different part, and though their observations appear to be absurd in isolation, they are in reality all correct, each is just partially so. When taken in concert, their observation describes the elephant more fully than the individual observations. This intuition has been exploited by a general class of algorithms known as *Random Subspace Methods*, *Bagging* and *Boosting*[5]. Most of these methods involve a set of weak learners which “probe” a part of the data. These learners, combined to form an Ensemble, generally outperform the individual learners, and are the basis of many state of the art algorithms available. Methods such as random forests, and the PAQ algorithm are examples of methods which use this paradigm.

2.1 Echo State Machines

The problem setup is as follows. We are given a series of musical events (x_1, x_2, \dots, x_n) . $x = (a, b)$ in this case is a tuple containing a discrete values, note $\in [1, 2, \dots, d]$, and real valued data $b \in \mathbb{R}$ At each point of time, we wish to find a probability distribution over

108 the next state $P(x_{n+1}|x_n, \dots, x_1)$. This probability distribution will take the form of a
 109 multinomial distribution over possible notes ($a \sim \text{Multinomial}(p_1, \dots, p_{61})$), and a Gaussian
 110 distribution with a fixed standard deviation over the duration and delays $b \sim \mathcal{N}(\mu_i, \sigma)$.

111 An echo state network consists of two parts - a *reservoir* and a *readout*. Let n be the number
 112 of neurons in the network, a tunable parameter. A random recurrent neural network (hereby
 113 referred to as the *reservoir*) is procured by generating a random matrix dense matrix W ,
 114 a *liquid* where each element W_{ij} is drawn from a standard normal. At each point in time t ,
 115 the random recurrent neural network is fed a time signal a_t . This signal has an encoding,
 116 a_t , $d \times 1$

117 E , another random sparse matrix which can be chosen arbitrarily. The neural network,
 118 $n \times d$ at each point, has a state s_t . At each time step, the states of the recurrent network are
 119 updated according to the following rule

$$s_{t+1} = \sigma \left(W s_t + E a_t \right)$$

121 Where

$$\sigma(x) = \frac{2}{1 + e^{-x}} - 1$$

122 The sequence of states, $\{s_1, s_2, \dots\}$ can be seen as a way of transforming the time series data
 123 with fading memory. To obtain useful information from the is the readout. The readout
 124 takes the form of a matrix of coefficients A for which the output will be $x_{t+1} = A s_t$. The
 125 readout is trained using linear regression from the input data. As a final step, the data is
 126 passed through a softmax function to normalize it into a probability distribution.

127 It is important to note that the random matrix W needs to be rescaled so that the spectral
 128 radius $\rho(A)$ is less than 1. To gain an intuition of why this is a good choice, notice that
 129 $\|W s_t\| \leq \|W\|_2 \|s_t\|_2 \leq \rho(A) \|s_t\|_2$. Since the sigmoid function keeps $\|s_{t+1}\|_2 \leq d$, repeated
 130 application of the W matrix for spectral radius too small will dampen signals quickly to 0.
 131 On the other hand, a spectral radius too large will send the neural network saturating at
 132 each timestep, making the behavior chaotic. Therefore setting it to ≈ 1 poses the neural
 133 network at the “edge of chaos”.

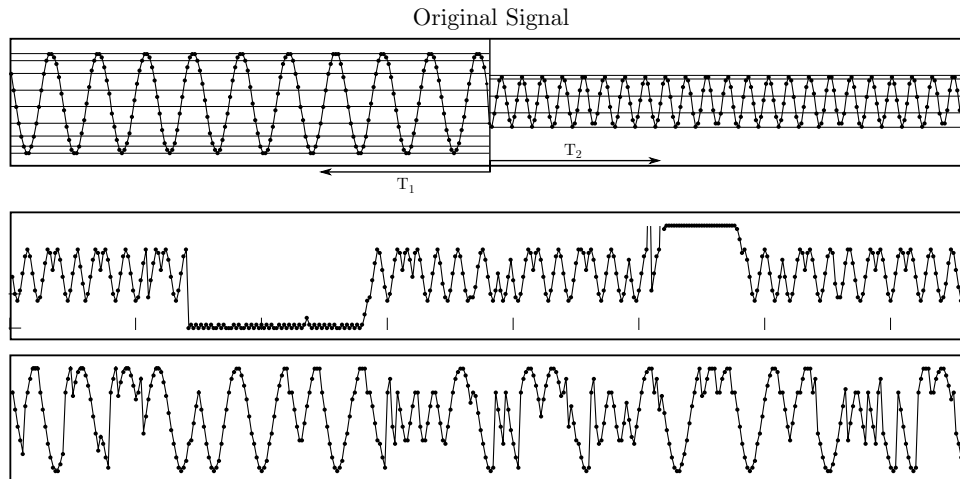
140 2.2 Comining Echo State Machines

141 Now that we have defined the Echo State Machines, we will treat them as a black box
 142 and proceed to combine them to form an Ensemble. Let us label each echo state machine
 143 with an integer $e \in \{1, \dots, E\}$. We randomly train each echo state machine on contigu-
 144 ous subsets of the data, hence each machine is hoped to capture a different “viewpoint”
 145 of the data. Given previous events (x_1, x_2, \dots, x_t) each echo state machine produces a
 146 random variable over the $X^{(e)} \sim P^{(e)}(x_{t+1}|x_1, x_2, \dots, x_t)$. Since the x 's contain both cat-
 147 egorical data and real valued data, we treat the two seperately. Let $X^{(e)} = (A^{(e)}, B^{(e)})$
 148 where A is categorical and B is real. For the categorical data, we average over the multi-
 149 nomial distributions. In other words, given $A^{(e)} \sim \text{Multinomial}(p_1^{(e)}, \dots, p_d^{(e)})$, we obtain
 150 $A \sim \text{Multinomial}(\sum_{e=1}^E p_1^{(e)}, \dots, \sum_{e=1}^E p_d^{(e)})$. We then take the maximum value of this dis-
 151 tribution, which can be interpreted as the democratic vote. For real-valued data, we simply
 152 take the average of the means.

153 3 Synthetic Data

154 We perform a simple experiment to convince ourselves the ensembles do indeed do some-
 155 thing. We train our improvisational tool on a artificially generated data-set of 6000 points
 156 containing sine waves $30 \sin(\frac{n}{3})$ and $20 \sin(\frac{n}{1.5}) + 10$. (the numbers are chosen so that they
 157 match the musical data in terms of scale). Note that the echo state network only sees *sym-*
 158 *bo*lic sequences, and does not take into account the spatial ordering of notes. This forces
 159
 160
 161

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177



178
179
180
181
182
183

Figure 2: The original (artificial) signal which consists of two parts. a) is the improvisation from the *Isolated Echo State Machine*. b) is an improvisation *Echo State Ensemble* with two echo state machines trained separately on T_1 and T_2 combined. The *Isolated Echo State Machine* fails to capture the long, low frequency sine waves, instead favoring the small high frequency sign waves, possibly due to it being easier to predict.

184
185
186
187
188
189

the echo state machine to memorize a huge repertoire of “phrases” in order to qualitatively reflect the input. We train our echo state machine on two networks.

Isolated Echo State Machine First, we train a single echo state machine with $n = 100$ neurons on the entire dataset.

190
191
192
193

Echo State Ensemble Next, we split time series is split into two components, T_1 and T_2 at the transition between the two sine waves and combine their readouts using the democratic procedure described above. We use $n = 50$ neurons each.

194
195
196
197
198
199
200

There is a striking difference between the two models is clear in this illustration. While the single echo state machine gets “stuck” and is unable to memorize both parts of the data, the ensemble is capable of capturing both signals and synthesizing them. It is interesting to note that the isolated echo state machine only captures the smaller, higher frequency component of the data - it is hypothesized that since the patterns in the high frequency part of the signal occur more frequently, it has valiance over the slower, lower memory low frequency component and hence dominates the slower low frequency parts of the wave.

201 202 4 Music Improvisation

203
204
205
206
207
208

We will now detail the exact experimental procedure employed for musical improvisation. Our data set consists of 48 pieces of piano pieces by Chopin downloaded from the *Classical Piano Midi Page*. We chose piano pieces largely for it’s simplicity as it only contains one channel and does not require the harmonization of multiple instruments, a more complex task.

209 4.1 Data Preprocessing

210 4.1.1 Data Clustering

211
212
213
214
215

We first preprocess our musical data by clustering it into musical parts. To do so, we take the raw MIDI data and perform spectral clustering on the data by building a similarity matrix S based on distances in time and pitch of the notes. In this case, the distance is simply the euclidean distance of the distance in $a(p - p')^2 + b(t - t')^2$ where $a = 1$ and

216 $b = 10$. This data can be interpreted as a graph, whose edges are the nodes and edge weights
 217 are the distances between nodes. Spectral clustering can be seen as an approximation of
 218 finding the k best cuts in the graph. By setting $b \gg a$, the cuts favour vertical rather
 219 than horizontal cuts, separating the music into contiguous components. We find spectral
 220 clustering surprisingly effective in clustering music into parts with independent textures.
 221 We run spectral clustering to separate the music into ψ different substrings. Abusing some
 222 language from music theory, we will call the segments “movements”.

223
 224 **4.1.2 Data Representation**

225 Raw MIDI data gives us data in terms of a 3-tuple, (note, start time, end time). We need to
 226 convert this data into a form which is conducive for our echo state machine. We hence take
 227 time differences to represent each note as a 3 element tuple, (note, duration, delay). Note
 228 is represented as a 1-of- k representation, where k is the total number of notes in the piece
 229 $[0, 0, \dots, 0, k, 0, \dots, 0]$. Duration is the duration of the note, which is end time-start time
 230 and the delay is the amount of delay before the next note starts playing. Note that delay
 231 helps the echo state machine learn chords automatically, as three notes played together will
 232 be represented a sequence with delay 0.

233
 234 **4.2 Data Analysis**

235 The Data is then analyzed using the Liquid State Ensembles described in the preceding
 236 section. Given a sequence of $48 \times \psi$ movements. We randomly select $\frac{2}{3}$ of the movements to
 237 be used as training, and $\frac{1}{3}$ of the movements to be used as testing. There are 85323 Notes
 238 in total, in $48 \times \psi$ different movements. Note that the exact split of training/test depends
 239 on the number of movements, and is different for each run. We train a classifier for each
 240 movement in the training set, and test it on the classifiers in the test set, resetting the state
 241 s of the echo state machine to 0 each time.

242
 243 **4.3 Results**

244
 245 As a measure of predictive accuracy, we use the average percent of misclassified notes or

246
 247
 248
$$100 \times \frac{1}{N} \sum_{i=1}^N \mathbb{I}(x_i \neq x_i^{(train)})$$

249
 250
 251 We also train a *base n-gram predictor*, which simply concatenates all the movements into a
 252 single, long piece, and does predictions based on counting the frequency of n -element tuples.
 253 The results are as follows

254

BASE PREDICTOR	UNIFORM	1	2	3
	0.9091%	2.9110%	4.1001%	4.2889%

256
 257

258 Now we run our model

259

		NEURONS			
		100	200	400	1000
SIZE	48 × 2	3.0710%	5.8286%	5.7487%	5.3943%
	48 × 3	4.4093%	5.4487%	5.0901%	5.5610%
	48 × 4	3.7093%	4.1559%	4.5767%	5.1559%

266
 267

268 Note that the predictive accuracy depends very heavily on the number of neurons used in
 269 the echo state machine. We get a modest improvement in accuracy by increasing the number
 of ensembles, and the improvements are more dramatic when there are less neurons. We

270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323



Figure 3: Sample music composition from echo state machine. Observe how it has captured increasing and decreasing progressions and chord repetitions. The music is slightly discordant but still pleasant.

hypothesize that smaller echo state networks reach it’s model capacity with more complex sequences, and hence has more to gain from the ensemble itself. To our dissapointment, the accuracy sometimes goes *down* when the ensemble size becomes too large. We guess that the averaging effects may drown out the signals from the noise, bring the model closer to the mean.

5 Conclusions

To our disappointment, a large, democratic collection of Echo State Machines trained on the same dataset, contrary to our expectations, only offers modest improvements in time series prediction. Fortunately, the musical improvisations (Figure 3) produced by the model are still quite pleasant and are recognizably musical and harmonious. Several suggestions come to mind on how to improve the Ensemble. One could use logistic regression to shift the weight onto predictors which have better predictive power. Or perhaps a Kalman Filter could be implemented on top of logistic regression to change the weights with time, perhaps allowing our improvisational tool to change musical textures slowly and consistently. Due to the constraints of time, however, we leave such investigations to the future.

References

- [1] N. Bertschinger and T. Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004.
- [2] J.E. Cohen. Information theory and music. *Behavioral Science*, 7(2):137–163, 1962.
- [3] J.A. Franklin. Recurrent neural networks for music computation. *INFORMS Journal on Computing*, 18(3):321, 2006.
- [4] B. Grychtol. Using echo state networks for modeling musical improvisation, 2006.
- [5] T.K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.
- [6] <http://www.piano.midi.de/chopin.htm>. Classical piano music page.
- [7] D.B. Huron. *Sweet anticipation: Music and the psychology of expectation*. The MIT Press, 2006.
- [8] H. Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007.
- [9] H. Jaeger and D. Eck. Can’t get you out of my head: A connectionist model of cyclic rehearsal. *Modeling Communication with Robots and Virtual Humans*, pages 310–335, 2008.
- [10] B. Knoll and N. de Freitas. A machine learning perspective on predictive coding with paq. *Arxiv preprint arXiv:1108.3298*, 2011.
- [11] L.B. Meyer. *Emotion and meaning in music*. University of Chicago Press, 1961.
- [12] L. Pape, J. de Gruijl, and M. Wiering. Democratic liquid state machines for music recognition. *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*, pages 191–215, 2008.
- [13] C. Rueda, G. Assayag, and S. Dubnov. A concurrent constraints factor oracle model for music improvisation. In *XXXII Conferencia Latinoamericana de Informtica CLEI*, 2006.