

# $\epsilon$ -Machine Estimation and Forecasting

## Comparative Study of Inference Methods

D. Shemetov<sup>1</sup>

<sup>1</sup>Department of Mathematics  
University of California, Davis

Natural Computation, 2014

# Outline

## 1 Motivation

- $\epsilon$ -Machines and Time Series
- Baum-Welch Algorithm
- The Future, The Comparison Project

# Outline

## 1 Motivation

- $\epsilon$ -Machines and Time Series
- Baum-Welch Algorithm
- The Future, The Comparison Project

# $\epsilon$ -Machines and Time Series.

- $\epsilon$ -machines eat sequences of alphabet symbols:  
 $\{\dots, X_{-2}, X_{-1}, X_0, X_1, X_2, \dots\}$

## $\epsilon$ -Machines and Time Series.

- $\epsilon$ -machines eat sequences of alphabet symbols:  
 $\{\dots, X_{-2}, X_{-1}, X_0, X_1, X_2, \dots\}$
- These look way too much like time series:  $\{X_t | t \in \mathbb{Z}\}$

## $\epsilon$ -Machines and Time Series.

- $\epsilon$ -machines eat sequences of alphabet symbols:  
 $\{\dots, X_{-2}, X_{-1}, X_0, X_1, X_2, \dots\}$
- These look way too much like time series:  $\{X_t | t \in \mathbb{Z}\}$
- Let's compare with time series methods?

## Time Series.

- A time series is a sequence of measurements taken from a system, spaced at regular time-intervals:  $\{X_t | t \in \mathbb{Z}\}$

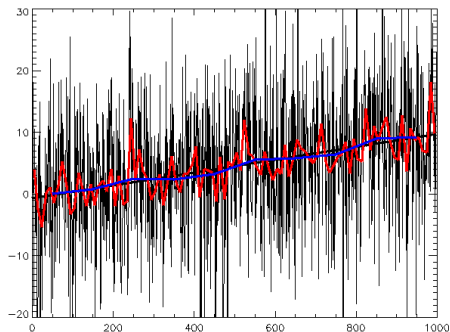
# Time Series.

- A time series is a sequence of measurements taken from a system, spaced at regular time-intervals:  $\{X_t | t \in \mathbb{Z}\}$
- Examples:
  - ▶ Atmospheric/climate data
  - ▶ Seismic activity data
  - ▶ Stock price data
  - ▶ Life



# Time Series.

- A time series is a sequence of measurements taken from a system, spaced at regular time-intervals:  $\{X_t | t \in \mathbb{Z}\}$
- Examples:
  - ▶ Atmospheric/climate data
  - ▶ Seismic activity data
  - ▶ Stock price data
  - ▶ Life



## Time Series Method Ecosystem.

- There are a lot of methods for studying and forecasting these things:

## Time Series Method Ecosystem.

- There are a lot of methods for studying and forecasting these things:
- Autocorrelation

## Time Series Method Ecosystem.

- There are a lot of methods for studying and forecasting these things:
- Autocorrelation
- Spectral analysis

## Time Series Method Ecosystem.

- There are a lot of methods for studying and forecasting these things:
- Autocorrelation
- Spectral analysis
- Principal component analysis

## Time Series Method Ecosystem.

- There are a lot of methods for studying and forecasting these things:
- Autocorrelation
- Spectral analysis
- Principal component analysis
- Neural networks

# Time Series Method Ecosystem.

- There are a lot of methods for studying and forecasting these things:
- Autocorrelation
- Spectral analysis
- Principal component analysis
- Neural networks

Tools for investigating time-series data include:

- Consideration of the [autocorrelation function](#) and the [spectral density function](#) (also [cross-correlation functions](#) and cross-spectral density functions)
- [Scaled](#) cross- and auto-correlation functions to remove contributions of slow components<sup>[8]</sup>
- Performing a [Fourier transform](#) to investigate the series in the [frequency domain](#)
- Use of a [filter](#) to remove unwanted [noise](#)
- [Principal component analysis](#) (or [empirical orthogonal function analysis](#))
- [Singular spectrum analysis](#)
- "Structural" models:
  - [General State Space Models](#)
  - [Unobserved Components Models](#)
- [Machine Learning](#)
  - [Artificial neural networks](#)
  - [Support Vector Machine](#)
  - [Fuzzy Logic](#)
- [Hidden Markov model](#)
- [Control chart](#)
  - [Shewhart individuals control chart](#)

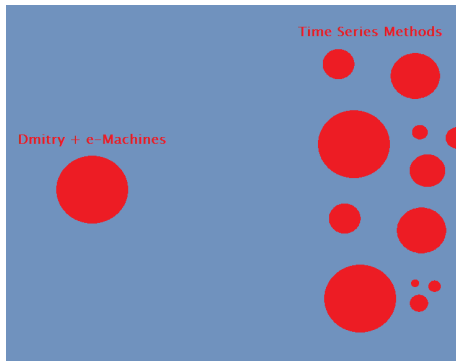
# [Dmitry + $\epsilon$ -machines] vs. [Time series]

- Let's compare!



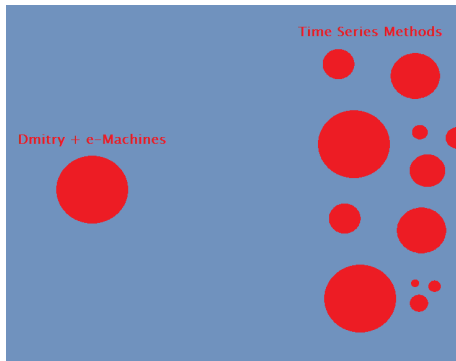
# [Dmitry + $\epsilon$ -machines] vs. [Time series]

- Let's compare!



# [Dmitry + $\epsilon$ -machines] vs. [Time series]

- Let's compare!



- Too many! Pick battles you can win!

# Outline

## 1 Motivation

- $\epsilon$ -Machines and Time Series
- **Baum-Welch Algorithm**
- The Future, The Comparison Project

## Baum-Welch - What is it?

The Baum-Welch algorithm - a method for inferring the most likely parameters of a Hidden Markov Model responsible for some given string of data.

### Example

#### Input

- Number of states -  $n$
- Network topology and initial guess for the parameters of the HMM -  $\theta$ 
  - ▶ Parameters are the start distribution, state transition probabilities, and symbol emission probabilities at each state.
- A sequence of outputted symbols -  $Y = \{Y_1, Y_2, \dots, Y_n\}$

### Example

#### Output

- New parameters  $\theta'$  with the property that  $P(Y|\theta') \geq P(Y|\theta)$ .

## Baum-Welch - How does it work?

- Belongs to a broad class of EM algorithms - expectation maximization.

## Baum-Welch - How does it work?

- Belongs to a broad class of EM algorithms - expectation maximization.
- Too many details:
  - ▶ Makes a forward pass to compute  $\alpha_i(t) = P("y_1, \dots, y_t", X_t = i | \theta)$ 
    - ★ Probability of seeing "word" and ending up in state  $i$  at time  $t$ .
  - ▶ Makes a backwards pass to compute  $\beta_i(t) = P("y_{t+1}, \dots, y_T" | X_t = i, \theta)$ 
    - ★ Probability of having the last  $T - t$  symbols be the specified "word", assuming that you ended up in state  $i$  at time  $t$ .
  - ▶ Combined these in the update step to calculate  $\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta)$ 
    - ★ Probability of transitioning  $i \rightarrow j$  at time  $t$ .
    - ★  $\xi_{ij}(t)$  can be computed as a function of  $\alpha_i$ 's and  $\beta_j$ 's
  - ▶ Where this eventually gets us:
    - ★ Summing  $\xi_{ij}(t)$  over time, we can compute the expected number of transitions from state  $i \rightarrow j$ , as well as  $i \rightarrow *$ .
    - ★ This will allow us to compute the expected transition probabilities between states.
    - ★ And! Similarly, we can compute the expected symbol emission matrix for each state.
    - ★ Starting probabilities.
    - ★ We now have  $\theta'$

## Baum-Welch - More.

- We can run this procedure again.

## Baum-Welch - More.

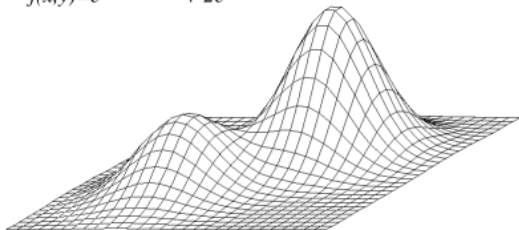
- We can run this procedure again.
- Thus, Baum-Welch algorithm is an iterative, gradient-based optimizer in the space of HMM parameters.



## Baum-Welch - More.

- We can run this procedure again.
- Thus, Baum-Welch algorithm is an iterative, gradient-based optimizer in the space of HMM parameters.
- “Hill-climber” algorithm.

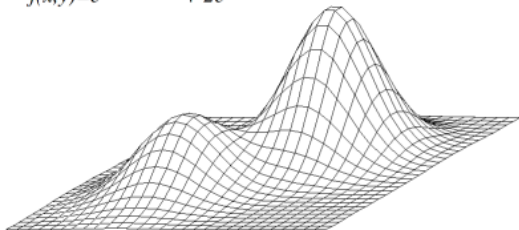
$$f(x,y) = e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$$



## Baum-Welch - More.

- We can run this procedure again.
- Thus, Baum-Welch algorithm is an iterative, gradient-based optimizer in the space of HMM parameters.
- “Hill-climber” algorithm.

$$f(x,y) = e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$$



- Gets stuck in local optima, so you have to seed it and rerun it with many starting conditions.

# Outline

## 1 Motivation

- $\epsilon$ -Machines and Time Series
- Baum-Welch Algorithm
- The Future, The Comparison Project

## $\epsilon$ -Machines and Baum-Welch.

- We can ask a number of questions.

## $\epsilon$ -Machines and Baum-Welch.

- We can ask a number of questions.

Pause

- Given impoverished data [small sample], do they differ in their rate of convergence to the true HMM as sample size is increased?

## $\epsilon$ -Machines and Baum-Welch.

- We can ask a number of questions.

Pause

- Given impoverished data [small sample], do they differ in their rate of convergence to the true HMM as sample size is increased?
- What about their error bars or confidence intervals?

## $\epsilon$ -Machines and Baum-Welch.

- We can ask a number of questions.

### Pause

- Given impoverished data [small sample], do they differ in their rate of convergence to the true HMM as sample size is increased?
- What about their error bars or confidence intervals?
- Maybe run-time speed, but that's tough - have to prove to C-language speed demons that you have a fast implementation.

## $\epsilon$ -Machines and Baum-Welch.

- We can ask a number of questions.

### Pause

- Given impoverished data [small sample], do they differ in their rate of convergence to the true HMM as sample size is increased?
- What about their error bars or confidence intervals?
- Maybe run-time speed, but that's tough - have to prove to C-language speed demons that you have a fast implementation.
- As fun side-project / implementation, would like to translate HMMs into some audio signal [for ex. each symbol corresponds to a pitch being emitted].
  - ▶ See if the HMMs the algorithms produce “sound” different.



## Sidenote: IPython.

- IPython is great for Windows machines that have run VM's.
- It is Sage-like, here is my workflow:

# Sidenote: IPython.

IPy Generate Machine and Data - More States (autosave)

```
In [1]: import numpy as np
from sklearn import hmm
import time as t

In [9]: states = 1
alphabet = 2
startprob = np.random.random(states)
startprob = startprob/sum(startprob)
transmat = np.random.random((states,states))
for i in range(len(transmat)):
    transmat[i] = transmat[i]/sum(transmat[i])

emissionprob = np.random.random((alphabet,alphabet))
for i in range(len(emissionprob)):
    emissionprob[i] = emissionprob[i]/sum(emissionprob[i])

print startprob
print transmat
print emissionprob

[[ 1.]]
[[[ 1.]]]
[[[ 0.34097063  0.65902937]
 [ 0.62161656  0.37838344]]]

In [14]: print startprob
print transmat
print emissionprob

[[ 1.]]
[[[ 1.]]]
[[[ 0.34097063  0.65902937]
 [ 0.62161656  0.37838344]]]

In [10]: model = hmm.MultinomialHMM(states, startprob, transmat)
model.startprob = startprob
```

```
f = urllib.urlopen("https://dl.dropbox.com/s/dn9qnr719ditaxe/X.txt")
dataDmitry = []
for line in f:
    temp = int(line)
    dataDmitry.append(str(temp))
print dataDmitry

['1', '0', '1', '1', '1', '0', '1', '0', '0', '0', '1', '0', '1',
'0', '1', '1', '0', '0', '1', '0', '1', '0', '1', '1', '1',
'1', '0', '1', '1', '1', '1', '1', '1', '1', '0', '1', '0', '1',
'0', '1', '1', '1', '1', '1', '1', '1', '1', '0', '1', '1',
'1', '0', '1', '0', '1', '1', '0', '1', '1', '0', '1', '1',
'1', '0', '1', '1', '0', '1', '1', '1', '1', '0', '0', '1',
'1', '1', '1', '1', '0', '1', '1', '1', '0', '1', '0', '1', '1',
'0', '1']

# Use from_string to define a very biased Biased Coin in CMPY
bcoin_str = "Q Q 0 0.1; Q Q 1 0.9"
bcoin = cmm.from_string(bcoin_str,
                        name='biased coin',
                        style=1)

# draw machine
bcoin.draw()

Couldn't import dot_parser, loading of dot files will not be possible

Q
0.10|0 0.90|1

bcoin_prior = bem.InferEM(bcoin)
print bcoin_prior.summary_string()

Epsilon Machine Inference
```

Thank you!

Thank you! Everybody!