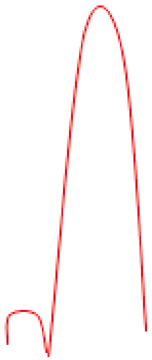
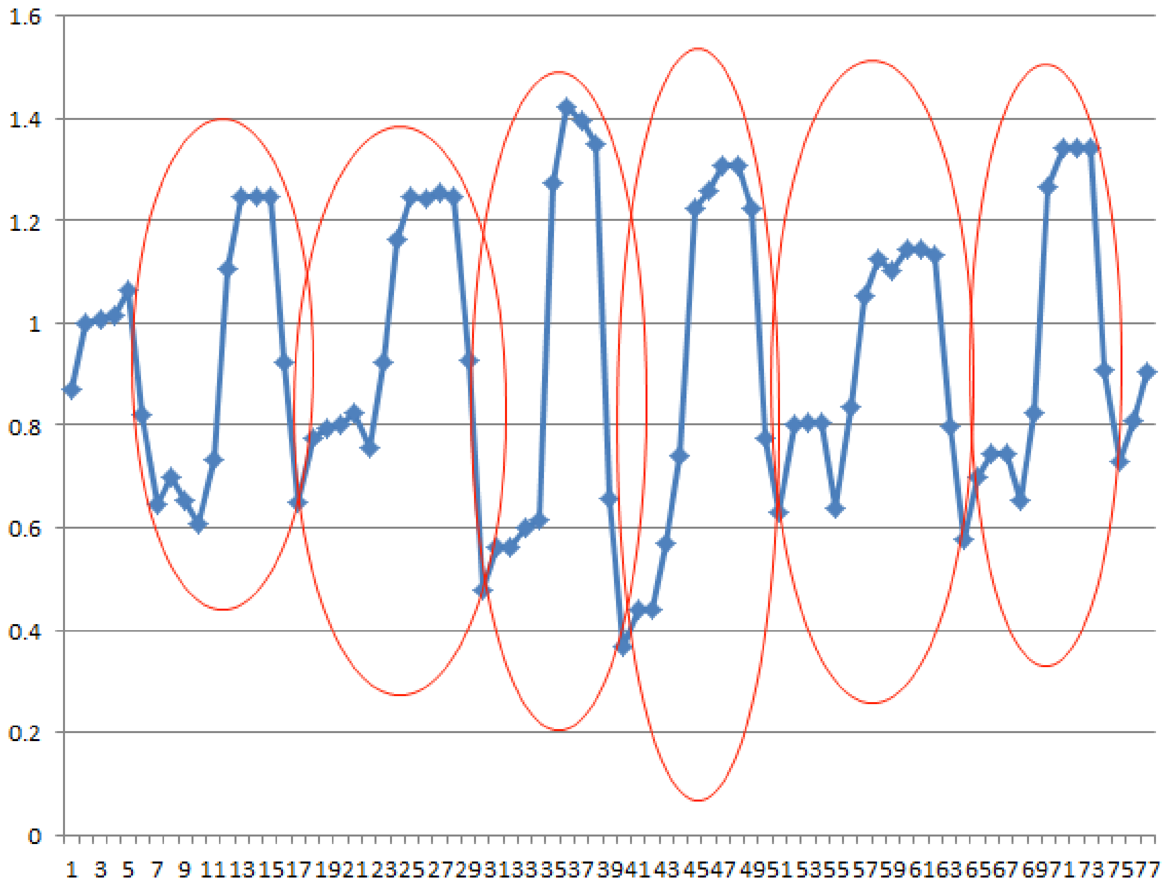
Pattern recognition in time series

By processing a time series graph, I Would like to detect patterns that look similar to this:



Using a sample time series as an example, I would like to be able to detect the patterns as marked here:



What kind of AI algorithm (I am assuming marchine learning techniques) do I need to use to achieve this? Is there any library (in C/C++) out there that I can use?

machine-learning time-series pattern-recognition

edited Aug 1 '12 at 5:13
JJPA
5,953 3 22 45

asked Aug 1 '12 at 4:53
Ali
271 1 6 10

Can you clarify: the patterns that you are looking for, are they fixed in size (always extend across the same number of time-steps)? Also, have you looked at sliding-window approaches? – schaul Aug 7 '12 at 19:34

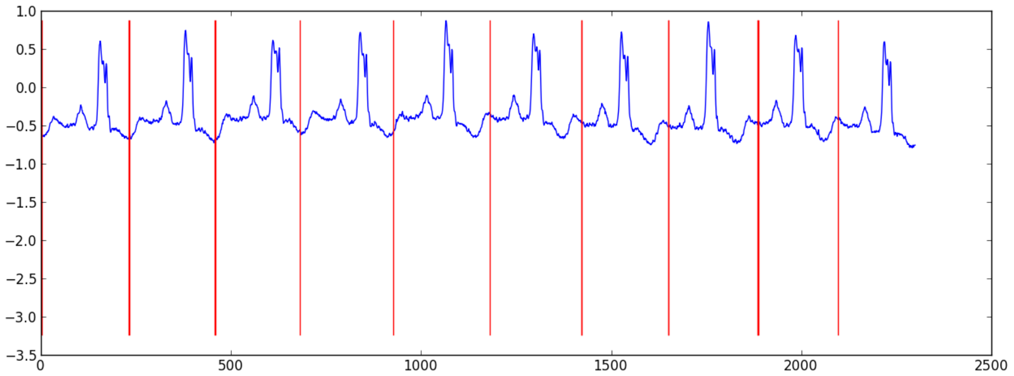
perhaps you should try R for this. As it's more statistically oriented. – marbel Nov 25 '13 at 19:00

@marbel what library in R should I use for this purpose? – Mona Jalal Sep 30 '15 at 16:45

That's a good question to ask. Why don't you write it up and ask this in SO. – marbel Oct 2 '15 at 17:01

5 Answers

Here is a sample result from a small project I did to partition ecg data.



My approach was a "switching autoregressive HMM" (google this if you haven't heard of it) where each datapoint is predicted from the previous datapoint using a Bayesian regression model. I created 81 hidden states: a junk state to capture data between each beat, and 80 separate hidden states corresponding to different positions within the heartbeat pattern. The pattern 80 states were constructed directly from a subsampled single beat pattern and had two transitions - a self transition and a transition to the next state in the pattern. The final state in the pattern transitioned to either itself or the junk state.

I trained the model with Viterbi training, updating only the regression parameters.

Results were adequate in most cases. A similarly structure Conditional Random Field would probably perform better, but training a CRF would require manually labeling patterns in the dataset if you don't already have labelled data.

Edit:

Here's some example python code - it is not perfect, but it gives the general approach. It implements EM rather than Viterbi training, which may be slightly more stable. The ecg dataset is from http://www.cs.ucr.edu/~eamonn/discords/ECG_data.zip

```
import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt
import scipy.linalg as lin
import re

data=np.array(map(lambda l: map(float,filter(lambda x:
len(x)>0,re.split('\\s+',l))),open('chfdb_chf01_275.txt'))).T
dK=230
pattern=data[1,:dK]
data=data[1,dK:]

def create_mats(dat):
    '''
    create
        A - an initial transition matrix
        pA - pseudocounts for A
        w - emission distribution regression weights
        K - number of hidden states
    '''
    step=5 #adjust this to change the granularity of the pattern
    eps=.1
    dat=dat[::step]
    K=len(dat)+1
    A=np.zeros( (K,K) )
    A[0,1]=1.
    pA=np.zeros( (K,K) )
```

```
        pA[0,1]=1.
        for i in xrange(1,K-1):
            A[i,i]=(step-1.+eps)/(step+2*eps)
            A[i,i+1]=(1.+eps)/(step+2*eps)
            pA[i,i]=1.
            pA[i,i+1]=1.
        A[-1,-1]=(step-1.+eps)/(step+2*eps)
        A[-1,1]=(1.+eps)/(step+2*eps)
        pA[-1,-1]=1.
        pA[-1,1]=1.

        w=np.ones( (K,2) , dtype=np.float)
        w[0,1]=dat[0]
        w[1:-1,1]=(dat[:-1]-dat[1:])/step
        w[-1,1]=(dat[0]-dat[-1])/step

        return A,pA,w,K

    #initialize stuff
    A,pA,w,K=create_mats(pattern)

    eta=10. #precision parameter for the autoregressive portion of the model
    lam=.1 #precision parameter for the weights prior

    N=1 #number of sequences
    M=2 #number of dimensions - the second variable is for the bias term
    T=len(data) #length of sequences

    x=np.ones( (T+1,M) ) # sequence data (just one sequence)
    x[0,1]=1
    x[1:,0]=data

    #emissions
    e=np.zeros( (T,K) )
    #residuals
    v=np.zeros( (T,K) )

    #store the forward and backward recurrences
    f=np.zeros( (T+1,K) )
    fls=np.zeros( (T+1) )
    f[0,0]=1
    b=np.zeros( (T+1,K) )
    bls=np.zeros( (T+1) )
    b[-1,1:]=1./(K-1)

    #hidden states
    z=np.zeros( (T+1),dtype=np.int )

    #expected hidden states
```

```
ex_k=np.zeros( (T,K) )

# expected pairs of hidden states
ex_kk=np.zeros( (K,K) )
nkk=np.zeros( (K,K) )

def fwd(xn):
    global f,e
    for t in xrange(T):
        f[t+1,:]=np.dot(f[t,:],A)*e[t,:]
        sm=np.sum(f[t+1,:])
        fls[t+1]=fls[t]+np.log(sm)
        f[t+1,:]/=sm
        assert f[t+1,0]==0

def bck(xn):
    global b,e
    for t in xrange(T-1,-1,-1):
        b[t,:]=np.dot(A,b[t+1,:]*e[t,:])
        sm=np.sum(b[t,:])
        bls[t]=bls[t+1]+np.log(sm)
        b[t,:]/=sm

def em_step(xn):
    global A,w,eta
    global f,b,e,v
    global ex_k,ex_kk,nkk

    x=xn[:-1] #current data vectors
    y=xn[1:,:1] #next data vectors predicted from current
    #compute residuals
    v=np.dot(x,w.T) # (N,K) <- (N,1) (N,K)
    v-=y
    e=np.exp(-eta/2*v**2,e)

    fwd(xn)
    bck(xn)

    # compute expected hidden states
    for t in xrange(len(e)):
        ex_k[t,:]=f[t+1,:]*b[t+1,:]
        ex_k[t,:]/=np.sum(ex_k[t,:])

    # compute expected pairs of hidden states
    for t in xrange(len(f)-1):
        ex_kk=A*f[t,:][:,np.newaxis]*e[t,:]*b[t+1,:]
        ex_kk/=np.sum(ex_kk)
        nkk+=ex_kk
```

```
    # max w/ respect to transition probabilities
    A=pA+nkk
    A/=np.sum(A,1)[:,np.newaxis]

    # solve the weighted regression problem for emissions weights
    #  x and y are from above
    for k in xrange(K):
        ex=ex_k[:,k][:,np.newaxis]
        dx=np.dot(x.T,ex*x)
        dy=np.dot(x.T,ex*y)
        dy.shape=(2)
        w[k,:]=lin.solve(dx+lam*np.eye(x.shape[1]), dy)

    #return the probability of the sequence (computed by the forward algorithm)
    return fls[-1]

if __name__=='__main__':
    #run the em algorithm
    for i in xrange(20):
        print em_step(x)

    #get rough boundaries by taking the maximum expected hidden state for each
position
    r=np.arange(len(ex_k))[np.argmax(ex_k,1)<3]

    #plot
    plt.plot(range(T),x[1:,0])

    yr=[np.min(x[:,0]),np.max(x[:,0])]
    for i in r:
        plt.plot([i,i],yr,'-r')

    plt.show()
```

edited Apr 13 at 12:44

Community ♦
1 1

answered Aug 10 '12 at 14:28

user1149913
3,324 1 10 19

1 sorry but I am very new in this field and I am still overwhelmed by the amount of information related to pattern recognition. I roughly understand what you are suggesting but might need a bit more of explanation if possible. I understand how HMM works, but my issue right now is how to map my problem into HMM, can you help me understand better by formating it similar to the example here: en.wikipedia.org/wiki/Hidden_Markov_model#A_concrete_example – Ali Aug 16 '12 at 9:39

2 Rabiner's "An introduction to hidden Markov models" is a good place to start. I have also included code to show how the HMM works in practice. – user1149913 Aug 18 '12 at 16:36

What's the content of the columns in chfdb_chf13_45590.txt – Ali Aug 29 '12 at 4:09

The first column is the time. Columns 2 and 3 are electrical signals. I was just using column 2 signals to simplify things. – user1149913 Aug 29 '12 at 14:56

Any chance an LSTM would be an appropriate tool for this kind of thing due to its memory features? – Norman H Oct 14 '15 at 12:34

Why not using a simple matched filter? Or its general statistical counterpart called cross correlation. Given a known pattern x(t) and a noisy compound time series containing your pattern shifted in **a,b,...,z** like `y(t) = x(t-a) + x(t-b) +...+ x(t-z) + n(t).` The cross correlation function between x and y should give peaks in a,b, ...,z

edited Sep 27 '16 at 10:02

Ruben de la Fuente
477 5 19

answered Sep 29 '15 at 16:46

Davide C
21 3

Weka is a powerful collection of machine-learning software, and supports some time-series analysis tools, but I do not know enough about the field to recommend a best method. However, it is Java-based; and you can call Java code from C/C++ without great fuss.

Packages for time-series manipulation are mostly directed at the stock-market. I suggested Cronos in the comments; I have no idea how to do pattern recognition with it, beyond the obvious: any good model of a length of your series should be able to predict that, after small bumps at a certain distance to the last small bump, big bumps follow. That is, your series exhibits self-similarity, and the models used in Cronos are designed to model it.

If you don't mind C#, you should request a version of TimeSearcher2 from the folks at HCIL - pattern recognition is, for this system, drawing what a pattern looks like, and then checking whether your model is general enough to capture most instances with a low false-positive rate.

Probably the most user-friendly approach you will find; all others require quite a background in statistics or pattern recognition strategies.

edited May 23 at 10:31
Community ♦
1 1

answered Aug 1 '12 at 5:08
tucuxi
10.3k 2 23 51

yes, I have labled dataset that can be used for supervised learning. Weka seems to be for Java language, any equivalent library for C/C++? – Ali Aug 1 '12 at 5:33

sorry, did not read that part of the question... what about cronos? stat.cmu.edu/~abrock/oldcronos – tucuxi Aug 1 '12 at 5:36

Sorry, do you know how to use cronos to do pattern recognition? I went through the website, all I see is estimation and forecast. – Ali Aug 1 '12 at 5:58

1 I assumed there is a strong relation between these. After training with part of the sequence, given the small spikes, GARCH and so on should generate a signal predicting the big spikes... – tucuxi Aug 1 '12 at 17:36

I'm not sure what package would work best for this. I did something similar at one point in college where I tried to automatically detect certain similar shapes on an x-y axis for a bunch of different graphs. You could do something like the following.

Class labels like:

- no class
- start of region
- middle of region
- end of region

Features like:

1. relative y-axis relative and absolute difference of each of the surrounding points in a window 11 points wide
2. Features like difference from average
3. Relative difference between point before, point after

answered Aug 2 '12 at 18:02

nflacco

2,741 4 33 59

I am using deep learning if it's an option for you. It's done in Java, Deeplearning4j. I am experimenting with LSTM. I tried 1 hidden layer and 2 hidden layers to process time series.

```
return new NeuralNetConfiguration.Builder()
                .seed(HyperParameter.seed)
                .iterations(HyperParameter.nItr)
                .miniBatch(false)
                .learningRate(HyperParameter.learningRate)
                .biasInit(0)
                .weightInit(WeightInit.XAVIER)
                .momentum(HyperParameter.momentum)
                .optimizationAlgo(
                        OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT  // RMSE:
????
                )
                .regularization(true)
                .updater(Updater.RMSPROP) // NESTEROVS
                // .l2(0.001)
                .list()
                .layer(0,
                        new
GravesLSTM.Builder().nIn(HyperParameter.numInputs).nOut(HyperParameter.nHNodes_1).ac

                .layer(1,
                        new
GravesLSTM.Builder().nIn(HyperParameter.nHNodes_1).nOut(HyperParameter.nHNodes_2).dr

                .layer(2,
                        new
GravesLSTM.Builder().nIn(HyperParameter.nHNodes_2).nOut(HyperParameter.nHNodes_2).dr

                .layer(3, // "identity" make regression output
                        new
RnnOutputLayer.Builder(LossFunctions.LossFunction.MSE).nIn(HyperParameter.nHNodes_2)
 // "identity"
                .backpropType(BackpropType.TruncatedBPTT)
                .tBPTTBackwardLength(100)
                .pretrain(false)
                .backprop(true)
                .build();
```

Found a few things:

- LSTM or RNN is very good at picking out patterns in time-series.
- Tried on one time-series, and a group different time-series. Pattern were picked out easily.
- It is also trying to pick out patterns not for just one cadence. If there are patterns by week, and by month, both will be learned by the net.

answered Jan 25 at 16:30

Neil Han
313 2 8