Cross Validated

Time series prediction with non-constant sampling interval

Asked 6 years, 7 months ago Active 6 years, 7 months ago Viewed 981 times

6

2

I have some data which can be modelled as such: each data sample S is a series of discrete signal values $S(t_n) \in \{-1, 1\}$ measured at times $(t_{n,S})_{1 \leq n \leq N_S}$. The number of signal measurements N_S per sample and the times at which the measurements were made all vary from sample to sample: $N_S \neq N_{S'}$ and $t_{n,S} \neq t_{n,S'}$ in the general case.

In other words, each data sample is a series of "yes or no" answers asked at various times.

I have some training data. Now, given a data sample S, I would like to predict the answer to the next question, in other words: the value of $S(t_{N_S+1})$. Even better would be to have the probability of a "no" answer: $P(S(t_{N_S+1}) = -1)$.

I have no idea how to do this. Any hint? Starters?

EDIT: if the signals were measured always at the same times ($t_{n,S} = t_{n,S'}$), and there were a constant number of signal measurements for each data sample ($N_S = N$), then I could produce for each data sample S a vector of $N - 1$ dimensions that would contain the $N - 1$ first signal measurements: $S = [1, -1, -1, 1, 1, \ldots]$. Measurement $S(t_N) \in \{-1, 1\}$ would be the label associated to sample S. Then, the problem would boil down to a classification/regression problem that can be solved e.g: with linear SVM. Unfortunately, the time differences between each signal measurement are important for the experiment.

regression time-series machine-learning classification prediction

edited Jan 25 '13 at 11:22

asked Jan 24 '13 at 17:36

Régis B.
81 5

So if I'm interpreting this correctly, there are a number of series (A,B,C..), each of which is sampled at a number of points. However, the times at which A is sampled are not the same as the times at which B is sampled. Is that right? Can you say anything about how you would solve this problem if the sampling was on an even grid? Then maybe someone can help you generalize that. Or do you not know that? – alex Jan 24 '13 at 22:50

Does my edit answer your question? – Régis B. Jan 25 '13 at 11:23

2 Answers

Broadly speaking, I can think of three approaches:

use kernel methods to do your learning (eg. SVMs if you're doing classification). If you want to prove that your SVM will converge you'll need to ensure that your distance function is positive semi-definite, but in practice it may well work fine either way. Your performance here will depend on how well you design your distance function.

- Extract some features. Ie. map to an m-dimensional space. An example approach: divide your timeline into k (overlapping) bins for various values of k, and count three values for each bin 1 to k: the number of -1s, +1s and 0s (no signal). Then, concatenate all values (for all k, for 1 to k, all three frequencies) into a vector and use that as a representation of your instance. Then, you can use any basic classifier you like. Your performance here will depend on the quality of your feature extraction. It should capture as much relevant information and try to minimize the number of dimensions of the resulting vector.
- Use a temporal model. A recurrent neural network or a hidden markov model can read the signals coming in and learn to synchronize with the signal. It will then continuously predict the next value at any moment. The drawback is that there is much less of a general framework. You'll have to do more work to implement everything for your domain, and comparing results between models is more difficult. Echo state networks (or reservoir computing) are probably a good model to investigate if your signal has a long memory.

I would go for the second option if you want to keep things simple, the first if you want to reduce arbitrary choices and the third if the first two don't work.

answered Jan 29 '13 at 17:08
Peter
841 5 12

Aproaches 1) and 2) are both quite hard, as they require me to design either a meaningful distance or a feature space (which is essentially the same task). I'm awarding the bounty for the third answer, as recurrent neural networks and reservoir computing are unknown concepts to me. – Régis B. Feb 4 '13 at 13:22

Thanks for the bounty. I would personally consider the first two options far less work and more rewarding. Designing a distance function/feature space isn't usually that much work and it allows you to automatically test many classifiers (using toolkits like WEKA or Matlab) and compare them simply. With the inherently temporal models, comparison can be more difficult, and you may need to implement more of the models yourself. Also, a feature space will give you a distance function as well, but the converse is not necessarily true. – Peter Feb 4 '13 at 13:40

1

If I understand correctly, this is just standard sequential binary prediction. You have an alphabet $\Sigma = \{-1, 1\}$, and samples drawn from Σ^* (the set of all finite length strings over your alphabet) Given a string $s_{1:t-1} \in \Sigma^*$ (the observations from time 1 to $t-1$) you want to predict s_t.

There are many techniques you could apply. What's going to be best is largely dependent on the nature of the data, especially things like how long each sequence is and how complex the dependency structure is. For example, it may be the case that $s_{t+1} = \text{sign}\left(\sum_{i=1}^{t} s_i\right) =$ the most frequent symbol is a good predictor. On the other hand if your data features dependencies separated by long time periods, something like $s_i = -1 \implies P(s_{i+100} = 1) >> P(s_{i+100} = -1)$, you're going to have a bad time as these types of relationship are notoriously hard to learn.

Without knowing more it is difficult to suggest any particular approach but here are some example)

- hidden markov models
- recurrent neural networks
- context tree weighting

answered Jan 25 '13 at 14:55

alto
3,195 14 19

I fail to see how your answer takes into account the fact that the time differences that separate different signal measurements vary? – Régis B. Jan 25 '13 at 16:15

It doesn't, I missed that part of your question. If you believe this information is relevant for your task, there are several ways you could handle this and allow you to work within the above framework. Say you have a -1 at $t = 1$ and a 1 at $t = 5$. You could represent you sequence as $-1, -1, -1, -1, 1$, or introduce another symbol into your alphabet $\Sigma = \{-1, 0, 1\}$ and represent your sequence as $-1, 0, 0, 0, 1$. I'm really not sure how this would work, or which approach would work better. Just an idea. – alto Jan 25 '13 at 18:08

That means that I would have to discretize the time measurements, and that's not possible :) – Régis B. Jan 25 '13 at 18:23

Well, if your convinced such a discretization would result in poor performance, I'm not sure. You may want to dig through some of the HMM literature. I imagine there has to be some work which takes into account time between transitions or observations. Sorry. – alto Jan 25 '13 at 18:53

I am not sure discretization would result in poor performance. But the fact that time samples can be separated by times from 1 second to 7 days make it difficult to define a discretization step. Discretization might be an option, but it's a difficult subject in itself, in the present case. – Régis B. Jan 25 '13 at 23:27