Search r/MachineLearning

r/MachineLearning

Posts

Posted by u/Fender6969 8 months ago

[D] Machine Learning on Time Series Data?

Discussion

I am going to be working with building models with time series data, which is something that I have not done in the past. Is there a different approach to the building models with time series data? Anything that I should be doing differently? Things to avoid etc? Apologies if this is a dumb question, I am new to this.

107 Comments Share ••• 97% Upvoted

This thread is archived
New comments cannot be posted and votes cannot be cast

SORT BY BEST

412freethinker 185 points · 8 months ago · *edited 8 months ago*

Hey, I recently researched time series classification for a personal project. I'm going to dump some of my notes here, with the caveat that I'm not by any means an expert in the area.

First, the most helpful summary paper I've found in this area is The Great Time Series Classification Bake Off.

Here's another awesome resource with open source implementations and datasets to play with: http://timeseriesclassification.com/index.php

PREPROCESSING

- Z-normalization transforms time series values so that the mean is ~0, and the standard deviation is ~1. See https://jmotif.github.io/sax-vsm_site/morea/algorithm/znorm.html "in some cases, this preprocessing is not recommended as it introduces biases. For example, if the signal variance is significantly small, z-normalization will simply overamplify the noise to the unit of amplitude."
- Piecewise Aggregate Approximation (PAA) - It's kind of like a histogram where you add up all the values in a window, but you scale it down by averaging them together. Think

SUPERVISED ML

- Some neural models are already well-suited for spatial input out of the box, like RNNs/LSTMs, or CNNs

INSTANCE-BASED CLASSIFICATION

- Dynamic time-warping - this is often used as a baseline, because it's very simple and apparently performs well. Uses dynamic programming to align the two sequences according to their closest fit. It's hella slow, but there are ways to speed it up like FastDTW and the LB Keogh lower bound. Con: apparently performance degrades with long time series, relatively short features of interest, and moderate noise
- Weighted DTW - adds a multiplicative weight penalty based on the warping distance between points in the warping path
- Time Warp Edit distance (TWE) - an elastic distance metric that gives you control via a stiffness parameter, v. Stiffness enforces a multiplicative penalty on the distance between matched points in a manner similar to WDTW.
- Move-Split-Merge (MPM) - Kind of like string edit distance, but for parts of a time series.
- Derivative DTW (DDTW) - a weighted combination of raw series and first-order differences for NN classification with full-window DTW.

SVM + String Kernels

Most of these methods were originally applied to gene protein classification, but they should generalize.

- k-spectrum kernel - The kernel essentially asks "how many subsequences do they have in common?" The vector space is the set of all possible k-mers, and each value is 1 if the sequence contains the given subsequence, otherwise 0. The kernel function compares two examples by taking the inner product.
- Mismatch kernel - Similar to the k-spectrum, but allow for approximate matches by looking for subsequences within a local edit distance neighborhood.
- Spatial representation kernel - Similar to k-spectrum, but matches don't have to be contiguous, so ABCD matches with ABZZCD.
- Fisher kernel - I had a harder time following this one, but I think it trains a HMM on positive classes, then computes a feature vector for each example by taking the gradient of the HMM's model parameters at that point, and train a classic SVM in this new feature space.

SHAPELETS

Informally, shapelets are time series subsequences which are in some sense maximally representative of a class. You can use the distance to the shapelet, rather than the distance to the nearest neighbor to classify objects. Shapelets are local features, so they're robust to noise in the rest of the instance. They're also phase-invariant: location of a shapelet has no baring on the classification.

Basically, a random forrest is trained, where each split point of the tree is a shapelet. You slide a window across training examples, looking for shapelets (subsequences) that split the dataset in such a way that maximizes information gain.

algorithm discretises and approximates the shapelets using a dictionary of SAX words.

- Shapelet transform - separates the shapelet discovery from the classifier by finding the top k shapelets on a single run (in contrast to the decision tree, which searches for the best shapelet at each node). The shapelets are used to transform the data, where each attribute in the new dataset represents the distance of a series to one of the shapelets. Then you can train a new model on top of this dataset.
- Learned shapelets - adopts a heuristic gradient descent shapelet search procedure rather than enumeration. LS finds k shapelets that, unlike FS and ST, are not restricted to being subseries in the training data.

INTERVAL-BASED

- Time Series Forest (TSF) - Similar to shapelets. The authors overcome the problem of the huge interval feature space by employing a random forest approach, using summary statistics of each interval as features. Training a single tree involves selecting sqrt(m) random intervals, generating the mean, standard deviation and slope, then creating and training a tree on the resulting 3*sqrt(m) features.
- Learned Pattern Similarity (LPS)
- Piecewise-linear approximations (PLA) - Also similar to shapelets, but rather than use subsequences as candidate features, since there are too many to consider, they use linear approximations. They "discretize" each time series, using a regression tree to approximate it in a series of flat lines. They consider different sized portions of this approximation at each split of the decision tree, instead of just the whole thing.

BAG OF WORDS

These approaches can be better than shapelets when the *number* of patterns is important, instead of just finding the closest match to a pattern.

- Symbolic Aggregate approXimation (SAX) - SAX transforms a time-series X of length n into a string of arbitrary length w. Step 1: convert time series to Piecewise Aggregate Approximation representation (see notes above). Step 2: Z-normalize, then pick equal-areas-under-the-curve to decide on an alphabet, so that each word has approximately equal likelihood of occurring. Helpful example: https://jmotif.github.io/sax-vsm_site/morea/algorithm/SAX.html
- SAX Distance - a formula to compare two SAX representations of time series. Basically mean squared error between letters, while allowing close letters to count as equal.
- Bag-of-Words Vector Space Model (SAX-VSM) - Uses SAX representation like above, but then applies TF-IDF weighting to the resultsing word-document matrix.
- Bag-of-Patterns (BOP) - Applies SAX to each window to form a word (If consecutive windows produce identical words, then only the first of that run is recorded to avoid over counting trivial matches). The distribution of words over a series forms a count histogram. To classify new samples, the same transform is applied to the new series and the nearest neighbour histogram within the training matrix found.
- Time Series Bag-of-Features (TSBF) - Local information is captured from multiple subsequences selected from random locations and of random lengths and partitioned into shorter intervals to detect patterns represented by a series of measurements over shorter time segments. Local features are aggregated into a compact codebook, and

- Bag-of-SFA-Symbols (BOSS) - It first extracts substructures (patterns) from a time series using Symbolic Fourier Approximation (which like SAX, converts to words). After a discrete fourier transform, do low pass filtering and quantisation (via Multiple Coefficient Binning), which reduces noise and allows for string matching algorithms to be applied. Sequences of words are counted and histogrammed. Two time series are then compared based on the differences in the set of noise reduced patterns (a modified Euclidean distance)
- BOSSVS - BOSS with TF-IDF weighting and cosine similarity, similar to SAX-SVM
- WEASEL (Word ExtrAction for time Series cLassification) - similar to BoF/TSBF/BOSS

FEATURE-BASED METHODS

- FRESH - Extract lots of statistics about time series (min, max, median, peaks, etc), use statistical p-values to determine which them are useful for learning the problem at hand, and train another model on top of it.

ENSEMBLE METHODS

- DTW Features - Combines DTW distances to training cases with SAX histograms.
- Collection of Transformation Ensembles (COTE) - A combination of classifiers in the time, autocorrelation, power spectrum, and shapelet domain. According to the authors this performs best in general, of all methods.

Share **Report** **Save**

Fender6969 9 points · 8 months ago

Wow thank you so much! I will be reviewing this in detail!

Share **Report** **Save**

gerry_mandering_50 1 point · 8 months ago

Time series is time sequence. You can see that yourself, right? It's not happenstance. RNNs are designed for them. Particular types of RNN currently in use are GRU and LSTM.

A detailed discussion and solution code for the Rossman dataset at Kaggle, and the discussion, code, and approach are at or near the best in existence currently, are being provided freely to the public by the author of the fast.ai library which runs over pytorch.

Go to fast.ai and watch the first deep learning course series of videos for these things, with particular attention to the Rossman solution.

For deeper understanding of the sequence predictions then please also undertake the study of the course sequence Deep Learning by deeplearning.ai, which is Andrew Ng's company. It gives much more background understanding of RNNs. It's up to you how much you want to "get it" (understand it).

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Yeah I am able to see that. I ended up getting a copy of Deep Learning by Françis Colet and I reviewed some of the code and concepts there. I didn't really have a

NowanIlfideme 7 points · 8 months ago

Great post, and thanks for the supplementary links! I would only add "Always try an auto-ARIMA as a possible baseline" for forecasting. ;)

Share **Report** **Save**

91user 4 points · 8 months ago

Wow, this was awesome. Can you please say something about using 1D convolution?

Share **Report** **Save**

412freethinker 8 points · 8 months ago

Sure! I haven't trained 1D CNNs myself yet, but I plan to in the coming months.

Convolutions are almost always presented as operations on 2D data, but the theory is the same if you just take away a dimension. Instead of operating on square pixel windows of an image, you can operate on successive windows of a time series (using whatever step size you want). Each learned filter applies some function (e.g. dot product) to a window in order to produce a higher level representation of the input.

If you then apply a pooling layer like max pooling to reduce dimension, the filters act as feature detectors. And, you can stack multiple CNN layers on top of each other, to squeeze the time series into a smaller semantic representation, which can work great as input of some other model. Instead of the classic 3-D funnel shape that appears in image processing papers, picture a 2-D funnel.

I'm trying to figure out how to deal with multivariate time series, where the variables at each time step are highly correlated, like audio spectrum data.

Share **Report** **Save**

royal_mcboyle 8 points · 8 months ago · *edited 8 months ago*

If you are trying to predict the next time step, Wavenet is a really good example of a 1D CNN:

https://arxiv.org/pdf/1609.03499.pdf

Instead of using pooling it uses dilations, which allow it to increase the size of the receptive field exponentially and minimizes information loss. They have some great graphics in the paper illustrating how they work.

If you are doing something like classifying an entire series, like modulation detection for radio signals, Timothy O'Shea has done some great work:

https://arxiv.org/abs/1712.04578

He's experimented with both oblong 2D convolutions and 1D convolutions for modulation prediction on I/Q signal collection data, but seems to prefer the 1D variant. I assume it's because the I and Q series are highly correlated and the larger number of parameters the oblong convolutions needed wasn't improving model performance.

Share **Report** **Save**

Thanks for the links, I'll check em out!

Share Report Save

szpaceSZ 1 point · 8 months ago

Wrt last sentence, please keep us updated.

Share Report Save

89bottles 1 point · 8 months ago

Have a look at Daniel Holden's work with motion capture, it uses CNNs for multivariate time series data.

Share Report Save

412freethinker 2 points · 8 months ago

Thanks, will do! I had a feeling there was another domain I could borrow from

Share Report Save

89bottles 1 point · 8 months ago

Just came across this: https://machinelearningmastery.com/how-to-develop-convolutional-neural-networks-for-multi-step-time-series-forecasting/

Share Report Save

rlrgr 1 point · 8 months ago

Have you tried something along the lines of multiple sequence alignment (MSA) for signals? From what I understand DTW is pairwise so if you have many signals it would be slow

Share Report Save

412freethinker 1 point · 8 months ago

I haven't yet. And yeah, that's the unfortunate part of nearest neighbor methods

Share Report Save

eamonnkeogh 1 point · 8 months ago

DTW is not slow. In 2002 we did one trillion DTW comparisons [a]. For most domains, you can do DTW 1000 faster than real time.

[a] https://www.cs.ucr.edu/~eamonn/SIGKDD_trillion.pdf

Share Report Save

412freethinker 1 point · 8 months ago

:O very cool, that's faster than I would have thought

Share Report Save

animonein 2 points · 8 months ago

You are a messiah! I was just looking for the time series analysis even though wes mckinney book was good introduction, but i was searching for more in depth take.

Thank you.

Sure thing, hope it helped!

Share **Report** **Save**

Overload175 2 points · 8 months ago

Thanks, this is honestly more informative than the majority of time series analysis articles I was able to find online.

Share **Report** **Save**

ragulpr 1 point · 8 months ago

great post!

Share **Report** **Save**

spongepoc2 1 point · 8 months ago

for classification, have you come across class imbalance and strategies for dealing with it? at the moment i'm downsampling the majority class but i'm looking into data augmentation to generate samples for the minority class so i don't have to chuck away so much data

thanks for this btw, this is one of the most helpful comment i've seen on r ml

Share **Report** **Save**

412freethinker 1 point · 8 months ago

Sorry, I haven't. Maybe something here will inspire you https://florianhartl.com/thoughts-on-machine-learning-dealing-with-skewed-classes.html

Share **Report** **Save**

Guanoco 1 point · 8 months ago

Wow your post made mine look so half-assed that it makes me want to delete it.

Anyways great post. But did you also get the feeling that the COTE was completely over engineered? Like it's a collection of ensembles and then weighted based on train... It honestly felt like the researchers where trying to find the holy grail by just packing ensemble after ensemble.

I think their point was that in order to have a good general classifier you had to inspect the data in all those features and then have the collection... But in real world I would probably pick the two best individual classifies and use them since the overhead of COTE is just huge for marginal gains.

Share **Report** **Save**

412freethinker 1 point · 8 months ago

Ensembles are weird like that. It does kind of seem like cheating. And you're right that after incorporating a few models, each one you add means more overhead for the researcher (and training time) for probably a <1% gain in test error. But still, ensemble methods win Kaggle competitions almost every time, so it's hard to write them off. Here's some theory behind why:

Share **Report** **Save**

Guanoco 2 points · 8 months ago

Yeah I know what you mean and I have personally nothing against ensembles. But COTE just seemed to be a bruteforce approach which in academia would be fine but somewhat less relevant in practical applications.

I think a nice number of performance would be somehow relate the size of the learnt model, the number of operations needed for one inference and the accuracy. A simple example would be accuracy/ops and that would tell you how much overhead one has against another.

Anyways good discussion

Share **Report** **Save**

MaxBenChrist 1 point · 8 months ago

I miss feature based approaches on your list. You could use a library like https://github.com/blue-yonder/tsfresh (Disclaimer: I am the maintainer of tsfresh) to extract features from your whole time series or subwindows of it and then feed this to a normal classifier/regressor like light gbm or random forest.

Feature based approaches have several advantages over black box models. Normally, they allow to interpret and analyze the features themself. In contrast, go and try to analyze a complicated RNN.

I worked a lot of the methods from your list. which of those models work, depends on the application that you are looking at. Lately I had great success in combining tsfresh features with deep learning models on financial time series. For supply chain problems or dataset with saisonal effects , the theta method or prophet work well from my experience. On IoT time series I had great success using kNN with DTW.

Share **Report** **Save**

412freethinker 1 point · 8 months ago

Thanks for the info, I hadn't come across anything like that but I'll add it to the list

Share **Report** **Save**

meghalD 1 point · 4 months ago

I wanted to combine tsfresh features with LSTM. Is it possible ? I would like you to share some links or codes which can help me. If not LSTM can I use it along CNN or say autoencoders ?

Share **Report** **Save**

Wapook 67 points · 8 months ago · *edited 8 months ago*

While I'm sure another poster will detail many time series specific models or ways to perform time series feature extraction, I want to draw attention to an equally important aspect: test sets for time series data. While in a typical machine learning task you might randomly partition your data into train, test, and validation, in time series approaches you want to perform backtesting. In backtesting you hold aside some "future" data to predict on. So if the data you have access to is from 1980-2010, you may wish to hold aside 2005-2010 data to

access to information that will make the model appear better than it is. If you hold aside validation data you will wish to do the same.

Share **Report** **Save**

Wizard_Sleeve_Vagina 18 points · 8 months ago

As a follow up, make sure the sets dont overlap as well.

Share **Report** **Save**

Fender6969 3 points · 8 months ago

What do you mean by overlap? Using the previous example of years 1980-2015, we want each data set to have unique years and the test set to have the most recent years?

Share **Report** **Save**

Wizard_Sleeve_Vagina 9 points · 8 months ago

If you are predicting a year out, you need to leave a 1 year gap between train and validation (and test) sets. Otherwise, you get labels leaking across data sets.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Oh I see. So maintaining 1 year gap between training, validation, and testing set should be fine when partitioning?

Share **Report** **Save**

dzyl 4 points · 8 months ago

I will extend this excellent comment with saying this is proper validation in a lot of less obvious cases. When we are cases where labels are not automatically collected it can matter less but when you are trying to forecast demands, anything that relies on processes etc you should also attempt to do this. Due to underlying changing behaviour you will overestimate the performance of your model because you are interpolating in your evaluation but extrapolating in reality when you put it in production.

Share **Report** **Save**

Fender6969 3 points · 8 months ago

Hey thanks for the tip with the testing set that definitely makes sense!

Share **Report** **Save**

ragulpr 3 points · 8 months ago

Same goes for any type of analysis of customers or individuals over time. I've been in interviews with gotcha' questions w.r.t how you split their data in the take-home assignment. It's clear after thinking about it, that for a prediction task one needs to split customers before/after a certain date not just assign customers randomly to test/train.

Still, they told me most fail on this and also most papers I've come across assigns each customer/patient/whatever randomly into groups instead of splitting by time.

Share **Report** **Save**

Wapook 2 points · 8 months ago

po-handz 2 points · 8 months ago

I do this on my crypto dashboard, where most recent 2 weeks are held out, and models trained on all other data. Found this was one of best methods for getting real accuracy measurements

Share Report Save

slaweks 24 points · 8 months ago

You need to preprocess the data, and the preprocessing, especially normalization, needs to be more careful when working with NNs than when using tree-based algorithms. Also, avoid information leakage from future, you need to do backtesting, not the standard cross-validation. Finally, compare you results to say (Theta+ARIMA+ETS)/3 - this may be a humbling experience :-)

Share Report Save

Fender6969 1 point · 8 months ago

Using classification as an example, I can still do the same methodology of normalization and regularization but use backtesting instead of standard resampling techniques (k fold cross validation etc)?

Share Report Save

slaweks 3 points · 8 months ago

Hi, I do not have much experience in classification, but just thinking about it, no, it is not the same. For classification you are looking for some distinguishing features, perhaps much earlier in the sequence (e.g. a weird spike in ECG that portend a hart attack in 30-60 minutes). When exactly it happens matters less. But in forecasting, generally speaking, after taking care of seasonality, older data is less important.

Share Report Save

Fender6969 2 points · 8 months ago

I see. So how would I approach this if I was doing a regression problem in terms of normalization and Regularization?

Share Report Save

slaweks 0 points · 8 months ago

Normalize locally, e.g. as in https://gallery.azure.ai/Tutorial/Forecasting-Short-Time-Series-with-LSTM-Neural-Networks-2

Share Report Save

texinxin 22 points · 8 months ago

Time can be a tricky thing. One little trick I've done when using tree based models is some feature engineering. A timestamp of noon on Jan 15, 2017.. can be taken as 'one point in time' some 'x' days ago. And that's one way to create the time attribute. However, be careful with defining time in just one way. Be sure to create all the possible attributes from this one

Depending on the response of your output variable to time, it could respond to time defined from a different perspective. Are there seasonal effects? Day of week effects? Time of day effects? General day over day change?

By performing feature engineering like this it enables you to test which 'dimensions' of time are important. And you might find that another variable that changes with time is more important to include as a reference data set.

For example, let's say you learn that there is a strong correlation with time of year to your output variable. Could it be temperature causing the effect? Could it be budgets and fiscal quarters? Could it be vacation days?

If you find a trend like this it is helpful to determine if you can find data "truer" to the input. If you found that temperature was more important than the time stamp, then a look-up table to write temperature to rows given a time stamp could improve your accuracy. Then time just becomes a key for joining in better data.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

That is a wonderful idea thank you! Honestly didn't think of that. We will be creating dashboards using Tableau and this would be great to present to the client visually along with the model.

Share **Report** **Save**

eamonnkeogh 12 points · 8 months ago

Many time series problems, including time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering etc can be solved very efficiently and simply using just the Matrix Profile [a]. There is free code for this, in several languages [a].

A quick scan of this [b] tutorial will give you a hint as to what the Matrix Profile can do.

If you need to use the Dynamic Time Warping (DTW), this tutorial explains everything you need to know [c].

To help you more than this, we would need to know your domain and task.

- [a] https://www.cs.ucr.edu/~eamonn/MatrixProfile.html
- [b] https://www.cs.ucr.edu/~eamonn/Matrix_Profile_Tutorial_Part1.pdf
- [c] https://www.cs.unm.edu/~mueen/DTW.pdf

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Interesting I have not done Matrix Profiling before. I will be sure to check it out thank you for the links!

Share **Report** **Save**

281HoustonEulers 1 point · 8 months ago

[a] Is there a direct link to the C++ version?

Share **Report** **Save**

Sure! The C++ version, the matlab version, the GPU version are all here:
https://sites.google.com/site/scrimpplusplus/

Share **Report** **Save**

futureroboticist 1 point · 3 months ago

Can MP do real time online classification?

Share **Report** **Save**

eamonnkeogh 1 point · 3 months ago

You can maintain the MP in real time, for very fast moving streams, using STAMPi (see MP I)

On top of that, you can build a classifier, so yes

Share **Report** **Save**

Oberst_Herzog 7 points · 8 months ago

It all depends, what is the purpose of your models and what kind of data are you dealing with? My formal background is in econometrics, so my considerations are largely influenced by that but some considerations would be:

- You will usually have to consider whether there is any dependence across time, and if so, how long does this dependence carry? Depending on this you might be fine using a 'non-state based model' (i.e you can just use an NN, SVM etc. and include lags as opposed to for instance a LSTM/GRU)
- Weak stationarity is very desirable (While I don't see ML stressing it as much as ordinary TS stat, I've always gotten much better results using ML on stationary series).
- Depending on your purpose and data, you have to think very carefully about how to do cross validation and similar model selections (temporal dependence leaks across time, so some omit a period to sufficiently separate training and test data). Instead of the general CV, to consider performance one usually applies something termed a moving/expanding window, (But due to the computational intensity behind the estimation of some ML models, you might want to omit reestimation for every step etc.).
- Finally, depending on your data you might also want to consider simple ordinary time series models such as ARIMA & VARIMA (depending on dimensionality) - I usually find that they both perform surprisingly well and are robust.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

As of right now, I have not had access to the data but from my understanding, we are tracking a specific performance metric over time and the model will possibly be built to predict the metric into the future. In terms of regularization, can Lasso still be used for dimensionality reduction when working with time series data?

Share **Report** **Save**

are just interested in minimizing some risk. If you are looking for inference, I recall some ways to do ML with adaptive lasso for ARMA (but I think it would be extendable to including independent variables). And if your end goal with regularization is to reduce the number of variables required to forecast (as they might all add some business cost to capture, store etc.), you should do L1 reg on the sum of the coefficients corresponding to every lag of the variable.

Although more common approaches (in econometrics) for handling high-dimensional data are based on dynamic factor models (just think of something as a time-varying PCA) and partial least squares.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Perfect thank you. My goal using Lasso is simply for dimensionality reduction. I am thinking of starting with a simple linear regression, I don't know if I will be using PCA as I losing interpretability is a concern.

Share **Report** **Save**

coffeecoffeecoffeee 7 points · 8 months ago

To clarify, are you forecasting the future using time series data? Or are you using time series data as an input to a classification problem? An example of each:

- What will Amazon's stock look like in a month?
- Given heart rate monitor data, can you predict whether someone is having a heart attack?

Share **Report** **Save**

Franky1499 2 points · 8 months ago

Hi, I'm working on a personal project which is similar to the second example here.

I have some hardware equipments data, when it was sent for maintenance, when did a failure occurs, how much weight it's lifting etc. It has time stamps of all events. I need to predict when the next failure will occur or when will we need maintenance for certain equipments.

I am very new to this and I think it's a classification problem as you mentioned in the second example. Could you point me towards some resources to learn how to go about this project or some advice that you can give?

Thank you so much.

Share **Report** **Save**

jlkfdjsflkdsjflks 2 points · 8 months ago

> I need to predict when the next failure will occur or when will we need maintenance for certain equipments.

Then it is not like the second example (i.e. classification) at all. It is more of a regression problem, since you're trying to estimate *when* something happens (i.e. you're trying to estimate a *quantity* of time).

least a limited discrete set of things)

Regression/forecasting: estimating/predicting a quantity or value (which can take an infinite number of possible values).

You can convert your problem to a classification problem, but then you have to make your questions more specific (e.g. "will this equipment fail within the next 30 days?"), so that you can answer them with a simple YES/NO (or a probability of "yes", for instance). In case it's not obvious, a classification problem is usually easier to solve than a regression/forecasting problem.

Share **Report** **Save**

Franky1499 1 point · 8 months ago

Thank you so much for the explanation, I have a better understanding now. I will approach this problem as a regression problem.

I did some googling and I think it is a time series forecasting problem, from what I read so far it looks difficult for my knowledge. Do you have tips on how to approach this excercise?

Also please correct me if my understanding is incorrect.

Share **Report** **Save**

avalanchesiqi 2 points · 8 months ago

> I need to predict when the next failure will occur or when will we need maintenance for certain equipments.

The problem you described here, to me it falls naturally into the field of point/hawkes process. You can relate it to the problem "when will the next bus arrive?" IMO it's a typical Poisson point process (https://en.wikipedia.org/wiki/Poisson_point_process)

Share **Report** **Save**

WikiTextBot 1 point · 8 months ago

Poisson point process

In probability, statistics and related fields, a Poisson point process is a type of random mathematical object that consists of points randomly located on a mathematical space. The Poisson point process is often called simply the Poisson process, but it is also called a Poisson random measure, Poisson random point field or Poisson point field. This point process has convenient mathematical properties, which has led to it being frequently defined in Euclidean space and used as a mathematical model for seemingly random processes in numerous disciplines such as astronomy, biology, ecology, geology, seismology, physics, economics, image processing, and telecommunications.The Poisson point process is often defined on the real line, where it can be considered as a stochastic process. In this setting, it is used, for example, in queueing theory to model random events, such as the arrival of customers at a store, phone calls at an exchange or occurrence of earthquakes, distributed in time.

Share Report Save

coffeecoffeecoffeee 1 point · 8 months ago

Aha! That's a lot of important context. Unfortunately I've been on a quest for similar data but haven't found it yet.

Share Report Save

Franky1499 1 point · 8 months ago

Thank you for the response. I will try to find how to approach the problem :)

Share Report Save

harry_0_0_7 1 point · 8 months ago

Please update here if you got any resources

Share Report Save

Fender6969 1 point · 8 months ago

Very valid question. To be completely honest, I have a strong feeling that at some point of time I will be attempting to do both of the scenarios you described. I have not had a chance to look at the data or get a detailed explanation of the requirements from them, but it seems like there is some metric that they are tracking over time that is of importance. My naïve assumption would be that I want to predict this metric into the future.

Share Report Save

harry_0_0_7 1 point · 8 months ago

I am also working on the problem like second one..

I am searching on a viable way to split time dependent features and after classifying/predicting result, trying to add time metric..

Have you gone through google AI's recent one https://arxiv.org/abs/1708.00065

Share Report Save

coffeecoffeecoffeee 1 point · 8 months ago

I haven't but I'll take a look. One standard way to do this appears to be calculating characteristics of the time series over sliding, overlapping windows.

Share Report Save

harry_0_0_7 1 point · 8 months ago

Also my time series will have many events!

I dono some research is also going on HMM with time..

Share Report Save

anonamen 3 points · 8 months ago

412freethinker's answer is quite good; all I'd add is that the single most important thing in applied time-series work is preventing data leakage. Most of the canned versions of forecasting models (even sophisticated ones) will let this happen, one way or another.

look the same, on a fundamental level. Sometimes it's fine. Most times it creates bias and blows up your predictions. ML is curve-fitting. Better ML is better curve-fitting. If you're not very careful in constructing fair back-tests for your problem the results will be misleading.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Hey thanks for the response. I will be sure to keep this in mind. So general idea is instead of resampling techniques like k fold cross validation, I should split into train, validation, and test set ensuring my latest data is in the test? Won't my model be more prone to overfitting or poor performance without techniques like k fold cross validation?

Share **Report** **Save**

Shevizzle 6 points · 8 months ago

Two acronyms: LSTM RNN

Share **Report** **Save**

gerry_mandering_50 1 point · 8 months ago

Yea, a lot of other comments are directing the OP to go in so many different directions. You supplied the correct answer most succinctly. OP really needs to first understand machine learning, then deep learning, then sequences (series) using deep learning, then how to code that up with the best available software toolkits. It's a long road, and perhaps he's ready to begin.

Share **Report** **Save**

Guanoco 2 points · 8 months ago

Just adding my two cents. I think some aspects depend on if you're doing a regression or a classification problem. Also are they multimodal timeseries?

Some common tricks and I've seen for classification is to znormalize the sequence (so each sequence is mean 0 std 1). Some classical people do ensembles of nearest neighbors with some elastic measurement (DTW) and other techniques (but they mostly say the 1 NN WITH DTW is very hard to beat). Most of the work I'm referring to can be found at timeseriesclassification.com (I'm not joking... It literally exists and maintained by some Prof I think).

I've seen some work using 1d conv, preprocessing to create a 2d representation of the time series and then use 2d convs and lstm but that depends on how much data you have.

For regression idk the other comments seem to be more related.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Hey thanks for the response, will check that website out. To be honest, our team has not had the opportunity to see the dats or get a detailed overview of what our objectives are just yet. All we know is that there is some metric this company is tracking over time that is of importance to them.

Share **Report** **Save**

model. At the very least you can use it to figure out your inputs and get a baseline for performance

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Absolutely. This what I am going to be starting with (give that the task is regression based). Some of these methodologies using time series data is new so this would be a good way to understand the data.

Share **Report** **Save**

futureroboticist 1 point · 3 months ago

What are linear time series model?

Share **Report** **Save**

BlandBiryani 2 points · 8 months ago

You should read Time Series Forecasting literature regarding modeling your problem and possible approaches. Bontempi's papers should be a good starting point for deciding between Recursive, Direct, Dir-Recursive or MIMO approaches.

Depending on your particular task, literature related to adaptive filters might even come in handy.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Will check it out. By any chance, do you have a link to the literature?

Share **Report** **Save**

e_j_white 2 points · 8 months ago

I'm trying to train an ARIMA model on some time series data (say price of orange juice), and would like to add extraneous variables - e.g. sentiment data from twitter.

Anybody have a good resource/tutorial for understanding how to do this? Bonus points for a Python package that can easily accomplish this. :)

Share **Report** **Save**

cssbit 2 points · 8 months ago

https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html

Share **Report** **Save**

tehnokv 2 points · 8 months ago

IMO, echo-state networks (a simplified version of RNNs) are worth mentioning due to their simplicity.

Some references:

- https://en.wikipedia.org/wiki/Echo_state_network

Fender6969 2 points · 8 months ago

Have not heard of echo state networks. Thanks for the links I will be sure to check it out! A big goal I foresee would be to the importance of interpretability of the model. My fear with using neural networks or deep neural networks would be that I may not be able to properly explain what is happening from layer to layer in the model, and how exactly the model arrived at the prediction it outputted.

Share **Report** **Save**

jlkfdjsflkdsjflks 3 points · 8 months ago

If you're looking for interpretable models, then echo state networks (along with anything using random projections) is not for you.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Yeah interpretability is very important on my end.

Share **Report** **Save**

Jonno_FTW 2 points · 8 months ago

My PhD research is around time series prediction (basically a regression task). The main models I looked at are:

- Lstm/convlstm
- SARIMAX
- Htm

Looking at pacf, ACF and different will find you good first impressions of your data and how it relates to itself.

There are unsupervised models of you're looking for that sort of thing too.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Hey thanks for the post. I really doubt it will be an unsupervised learning case as there is some metric this company is already tracking over time that the project is revolving around (which does lead me to believe that this will be a regression based problem). Seems as if LSTM and SARIMAX are quite popular response for models in this post. I know that Keras package will allow for me to use LSTM, will have to research SARIMAX.

Share **Report** **Save**

Jonno_FTW 1 point · 8 months ago

SARIMAX is just a seasonal variant of ARIMA with extra variables (you don't need to use all the parameters). If you're using python, the statsmodels library has everything you need in the tsa submodule. The p,d,q,P,D,Q variables can be determined by scaling, differencing, ACF, PACF, there's plenty of tutorials out there about this. Determining s can be a bit difficult, you need to do t - t_s for whatever the length of a

Keep in mind that arima (or at least the statsmodels implementation) isn't that great on large datasets or with long seasons so you might want to use R or SPSS.

The annoying part of LSTM in keras is getting the shapes of your inputs right and using the stateful LSTM correctly. Be sure to read the docs thoroughly because there's a few gotchas. Also I found dropout and relu layers immediately after LSTM layer improves things and then a single dense layer at the end with relu activation. Here's a useful guide: http://philipperemy.github.io/keras-stateful-lstm/

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Yeah I will be using Python for this project, I don't think we will be using R. How would R handle this better than Python? Simply the flexibility of the packages for R?

And that was a problem I was having with Keras when working with it in the past. The size/shape was difficult for me to understand. Hopefully your link covers this.

Share **Report** **Save**

LoudStatistician 2 points · 8 months ago · *edited 8 months ago*

Other comments are good, especially the one about local validation with backtesting. Other tips:

Take a look at Facebook Prophet, which makes it very easy to get close to a good result, without any features/context.

I would stay clear from any neural based algo's as a novice (easy to overfit/underfit, overkill solution, resource-heavy, quite difficult to implement in continuous/real-world setting). If you architect a RNN/LSTM/Convnet and it does not beat simpler, more-production-friendly methods (say KNN with polynomial kernel logreg, or even a basic survival model), then consider the project a failure (or a learning exercise in neural based algo's).

XGBoost with lagged and quant features is very powerful and more difficult to shoot yourself in the foot. Online learning is good in a setting where you need continuous learning and don't want to constantly retrain to stay ahead of concept drift. RL is very powerful, but hard to set up and not much information available yet.

For state-of-the-art, revisit Kaggle's many forecasting competitions. You'll probably find something that is in between: "We created this new weird algorithm for timeseries forecasting that works really well on this artificial toy dataset", and "We just ran it through a ConvNet, tuned the window-size and architecture and this is what popped out after 180 AWS hours".

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Thank you so much for the tips, will keep these in mind! I think for the time being, I will start with some basic linear regression and see how I do. I have had really good success with XgBoost in the past, and am glad that model ideally works well!

Share **Report** **Save**

- [/r/tsdb] [D] Machine Learning on Time Series Data?

If you follow any of the above links, please respect the rules of reddit and don't vote in the other threads. (Info / ^Contact)

Share Report Save

xkalash1 1 point · 8 months ago

I'd say time series learning is quite different than say computer vision especially when you try to do at scale. You need to define your problem better for us to propose proper approaches. In general there is traditional approach that works well with single time series and modern RNN approaches for high dimensional time series.

The train test split should also require some special attention depending on your specific needs

Share Report Save

Fender6969 1 point · 8 months ago

That is fair but we are still awaiting a meeting for them to give us the data and problem statement. All we know at this point is that there is some metric that is important that they are tracking over time.

Share Report Save

mufflonicus 1 point · 8 months ago

Check for trends - time series with positive or negative trends are death traps. Apply transform or split series (ie detrend) before doing anything worthwhile!

Statistical models such as SARIMAX is good for getting a baseline when doing regression. Auto-regression and moving averages are also good data transforms in their own right. Kalman filters might also be a good read.

For classification you can't go wrong with wavelets.

Share Report Save

Fender6969 1 point · 8 months ago

Thank you for the response. I have not used some of the algorithms you mentioned, and will of further research on implementation. I will definitely be checking for trends and doing general descriptive analytics as soon as we get the data!

Share Report Save

slaweks 0 points · 8 months ago

A trend should not be a death trap if you are using ML and normalize properly.

Share Report Save

nickkon1 1 point · 8 months ago

Another point is to be careful with new features. Do you really have them in the future? Meaning can you confidently see what your feature looks like in a week if you want to predict that week? Weather can be used as a feature but it is an example of that. Weather itself is predicted for the future.

I should start by saying that we have not received the data yet and are waiting for the project overview meeting. From what we have been provided already, there is some sort of metric that this company has been tracking over time. My naïve assumption would be that the problem is a regression problem.

Share **Report** **Save**

hopticalallusions 1 point · 8 months ago · *edited 8 months ago*

Long Short Term Memory neural networks worked great for a project I was involved with. We used Caffe.

Share **Report** **Save**

Fender6969 1 point · 8 months ago

Hey thank you, seems to be a popular recommendation here. My only question concern is interpretability, which is why I am leaning towards starting with something a little simpler like SARIMAX.

Share **Report** **Save**

Ikuyas 0 points · 8 months ago

You may use the existing algorithm used for videos instead of static images.

Share **Report** **Save**

ruixif 0 points · 8 months ago

Starting with auto-regression will have you a lot of fun.

Share **Report** **Save**

whenmaster 0 points · 8 months ago

How would you classify time series of variable length? I know that HMMs can be used for this task, but I was wondering if it was possible with other classifiers as well.

Share **Report** **Save**

eamonnkeogh 1 point · 8 months ago

In many cases, you can just make them the same length. See section 3 of [a]

[a] https://www.cs.ucr.edu/~eamonn/DTW_myths.pdf]

Share **Report** **Save**