Humanitas

A prediction tool for volatile commodity prices in developing countries

Students: Alexander John Busser, Anton Ovchinnikov, Ching-Chia Wang, Duy Nguyen, Fabian Brix, Gabriel Grill, Julien Graisse, Joseph Boyd, Stefan Mihaila

Time Series data

Different sources Price categories: Retail prices, wholesale prices

Wholesale prices

Wholesale price index

(taken from investopedia.com)
An index that measures and tracks the changes in price of goods in the stages before the retail level. Wholesale price indexes (WPIs) report monthly to show the average price changes of goods sold in bulk, and they are a group of the indicators that follow growth in the economy.

Although some countries still use the WPIs as a measure of inflation, many countries, including the United States, use the producer price index (PPI) instead.

Price sequences

Other sources

distribution & production
exchange rate crude oil

Social Media data

Twitter

Historical tweets

Approach 1: Fetching "historical" tweets via user graph

Using the Twython package for python we are able to interface with the Twitter API. Our methodology will be to select the twitter accounts of a number of regional celebrities as root nodes. These are likely to 'followed' by large numbers of local users. In a first phase, from each of these roots we may extract a list of followers and filter by various characteristics. Once a substantial list has been constructed, we may proceed to download the tweet activity (up to 3200 tweets) of each of these users in a second phase.

Despite recent updates allowing developers greater access, Twitter still imposes troublesome constraints on the number of requests per unit time window (15 minutes) and, consequently, the data collection rate. It is therefore necessary to: 1) optimise the use of each request; and 2) parallelise the data collection effort.

As far as optimisation is concerned, the **GET statuses/user_timeline** call may be called 300 times per 15 minute time window with up to 200 tweets returned per request. This sets a hard upper bound of 60000 tweets per time window. This is why the filtering stage of the first phase is so crucial. Using the **GET followers/list** call (30 calls/time window), we may discard in advance the majority of twitter users with low numbers of tweets (often zero), so as to avoid burning the limited user timeline requests on fruitless users, thus increasing the data collection rate. With this approach we may approach optimality and achieve 4-5 million tweets daily per process. However, it may be prudent to strike a balance between collection and diversity of account. Therefore a nominal filter is currently set to 50 tweets minimum rather than 200. It is furthermore necessary to install dynamic time-tracking mechanisms within the source code so as to monitor the request rates and to impose a process 'sleep'

Parallelisation will begin with obtaining N (≈ 10) sets of developer credentials from Twitter (https://dev.twitter.com/). These N credentials may then be used to launch N processes collecting tweet data in parallel. Given the decision to divide the follower collection and tweet collection into two separate phases (this may alternatively be done simultaneously), there is no need for distributed interaction between the processes to control overlap, as each process will simply take $1/N$ th of the follower list produced in phase 1 and process it accordingly. It should be relatively simple to initiate this parallel computation given the design of the scripts.

Approach 2: Filtering tweets provided by webarchive.org

https://archive.org/details/twitterstream

Daily? tweet aggregator

Issue of localization

Geolocalized tweets

Approximation: Mapping tweets to user location

Processing

Merging Series

Crafting indicators from tweets

Price Transmission Analysis

Interpretation

automate interpretation to a certain extent by learning about circumstances through online data.

Time Series Analysis

Time series data has a natural temporal relation between different data points. It is important in the analysis to extract significant temporal statistics out of data. We will focus on analyze stationarity, autocorrelation, trend, volatility change, and seasonality of our price datasets in R.

Stationarity of a series guarantees that the mean and variance of the data do not change over time. This is crucial for a meaningful analysis, since if the data is not stationary, we can not be sure that anything we derive from the present will be consistent in the future. We can transform our data into a stationary one by taking k-th difference to remove the underlying trend, and then apply standard test procedures such as KPSS test [1] to see if the differenced series is stationary.

Autocorrelation is another important trait in time series data. It suggests the degree of correlation between different time periods. By plotting correlograms (autocorrelation plots) of our data, we will be able to identify if the fluctuation of prices may be due to white noise or other hidden structures.

Seasonality is reasonably expected in our agricultural related time series. Several methods might help us to detect seasonality, such as common run charts, seasonal subseries plots, periodograms, and the correelograms we mentioned before.

(trend and volatility change is straightforward and can be concluded once we have the datasets)

[1] Kwiatkowski, D.; Phillips, P. C. B.; Schmidt, P.; Shin, Y. (1992). "Testing the null hypothesis of stationarity against the alternative of a unit root". Journal of Econometrics 54 (13): 159178.

Prediction Models

Time Series Forecasting

ARMA Model

The classical Time series forecasting approach is to use the ARMA (Auto-Regressive Moving Average) model to predict the target variable as a linear function which consists of the auto-regressive part (lag variables) and the moving average part (effects from recent random shocks).

The ARMA(p,q) model: (will refine math representations later)

$Phi(B) * Y_t = Theta(B) * eps_t$

The fitting of the model and the historical data can be accomplished by maximum likelihood estimation.

Regression

We can also apply ARMA to the linear regression model. It is formulated as such:

$Y = Beta * X + eps, eps\ ARMA(p, q)$

Through OLS (Ordinary Leasat Square) or GLS (General Least Square) processes, we can obtain an optimal Beta.

Multilayer Perceptrons

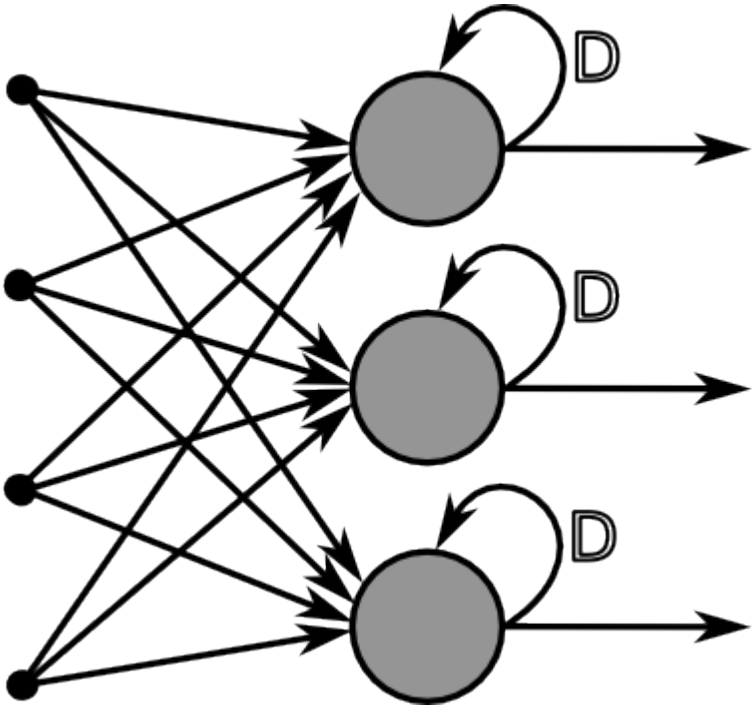
taken from M. Seegers course on Pattern Recognition and ML

Recurrent Neural Networks (RNN)

source: scholarpedia A recurrent neural network (RNN) is a special network in which neurons send feedback signals to each other. The key feature is that the weight matrix for each layer l in the network contains input weights from *all* other neurons in the network and not just the neurons from the previous layer.

Simple Recurrent Networks

These links enable signals to be fed back from a layer to a previous layer. The most simple form of an aRNN consists of an input, an output and one hidden layer.



General description of a discrete time aRNN

graph with K input units $\mathbf{u}$, N internal network units $\mathbf{x}$ and L output units $\mathbf{y}$. Activation (per layer) vectors at point n in time are denoted by $\mathbf{u}(n) = (u_1(n), ..., u_n(n))$, $\mathbf{x}(n) = (x_1(n), ..., x_n(n))$, $\mathbf{y}(n) = (y_1(n), ..., y_n(n))$. Edges between the units in these sets are represented by weights $\omega_{ij} \neq 0$ which are gathered in adjacency matrices. There are four types of matrices:

- $\mathbf{W}^{in}_{N\times K}$ contains inputs weigths for an internal unit in each row respectively
- $\mathbf{W}_{N\times N}$ contains the internal weights. This matrix is usually sparse with densities $5\% - 20\%$
- $\mathbf{W}^{out}_{L\times(K+N+L)}$ contains the weights for edges, which can stem from the input, the internal units and the outputs themselves, leading to the output units.
- $\mathbf{W}^{back}_{N\times L}$ contain weights for the edges that project back from the output units to the N internal units

In a *fully recurrent network* every unit receives input from all other units neurons and therefore input units can have direct impact on output units and output units can further be interconnected.

Evaluation

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{W}^{in}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{back}\mathbf{y}(n))$$

where $f = (f_1, ..., f_N)$

Exploitation

$$\mathbf{y}(n+1) = f^{out}(\mathbf{W}^{out}(\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n)))$$

where $f^{out} = (f_1^{out}, ..., f_L^{out})$

A *Hopfield network* is an RNN all connections of which are symmetric and which requires stationary inputs.

Echo State Networks (ESN)

Echo State Networks are a type of Recurrent Neural Network (RNN) applicable to many domains because unlike other RNNs they are easy to train. For our task we need a discrete time neural network which is incidentally also the constraint in which Echo State Networks are defined. They have sparsely connected *random* hidden layer consisting of the internal units with the effect that the weights of the output neurons are the only part of the network that can change and be trained. This property allows for the rest of the network to be topologically unrestricted. They are good at reproducing time series. (wikipedia)

Echo State Property Intuition: "If the network has been running long enough, its internal state is uniquely determined by the history of the input signal the teacher forced output."

Finding the right topology for the specific prediction problem

Converge

conceive network that converges fast to speed up training

"Known supervised training techniques for RNNS comprise Back Propagation Through Time (BPTT), Real Time Recurrent Learn- ing (RTRL) or Extended Kalman Filtering (EKF) all of which have some major drawbacks." *Application of BPTT to RNNs requires stacking identical copies of the network thus unfolding the cyclic paths in the synaptic connections. Unlike back-propagation used in*

feed-forward nets, BPTT is not guaranteed to con- verge to a local error minimum, computational cost is O(TN2) per time step where N is the number of nodes, T the number of epochs. In contrast RTRL needs O((N + L)4) (L denotes number of output units), which makes this algorithm only applicable for small nets. The algorithm complexity of EKF is O(LN2). EKF is mathematically very elaborate and only a few experts have trained predefined dynamical system behaviors successfully

Avoiding overfitting

[**?**] The third section explains how echo state networks can be trained in a supervised way. The natural approach here is to adapt only the weights of network-to-output connections. Essentially, this trains readout functions which transform the echo state into the desired output signal. Technically, this amounts to a linear regression task.

Echo States

Training ESN