Multimodel forecasting multiscale time series in Internet of things

Radoslav Neychev

Moscow Institute of Physics and Technology
Supervised by Vadim Strijov

October 14, 2016

Test bench for multiscale time series forecasting

The goal

is to create a test-bench, which makes an accurate and stable forecast of a set of multi-scale time series.

The method

- resample time series to construct autoregressive matrix,
- generate features,
- select features,
- make multimodel,
- compute the error.

The project compares models and their expert mixtures to understand a role of each model in the adequate forecast.

Multiscale data

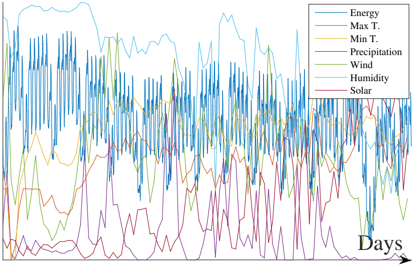
Consider a large set of time series $\mathfrak{D} = \{\mathbf{s}^{(q)} | \ q = 1 \ldots, Q\}$.

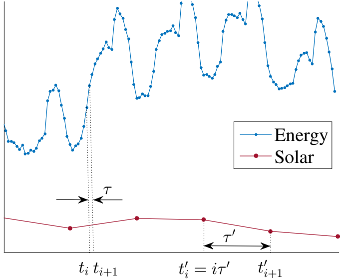
Each real-valued time series $\mathbf{s}$

$$\mathbf{s} = [s_1, \ldots, s_i, \ldots, s_T], \quad s_i = s(t_i), \quad 0 \leq t_i \leq t_{\max}$$

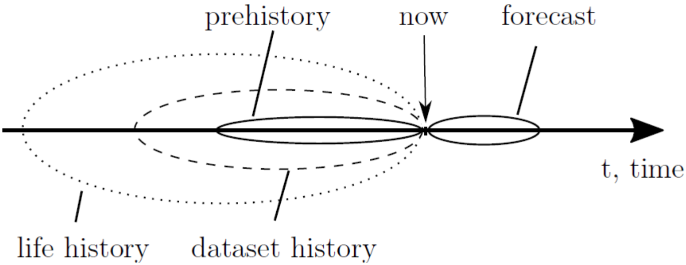
is a sequence of observations of some real-valued signal $s(t)$.

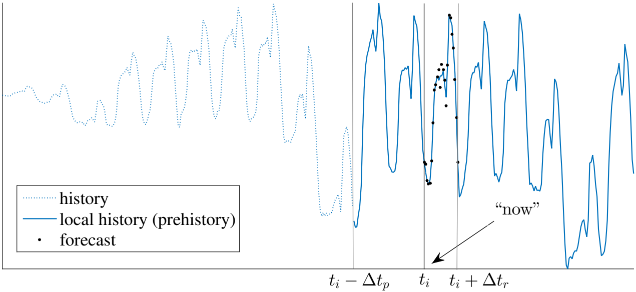
Each time series $\mathbf{s}^{(q)}$ has its own sampling rate $\tau^{(q)}$.





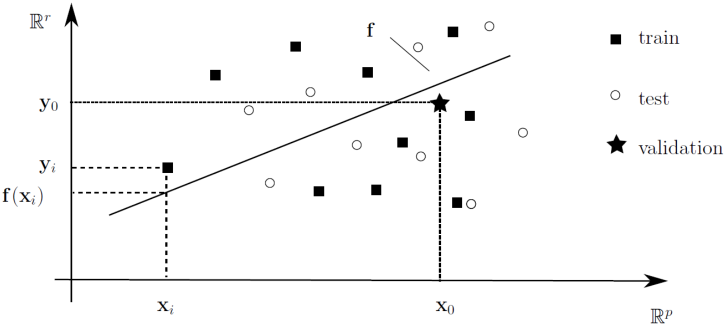
Time series forecasting



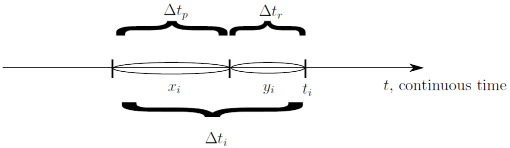


Design matrix

Forecast is a mapping from p-dimensional objects space to r-dimensional answers space.



$$\mathbf{X}^* = \left[\begin{array}{c|c} \underset{1\times n}{\mathbf{x}} & \underset{1\times r}{\mathbf{y}} \\ \hline \underset{m\times n}{\mathbf{X}} & \underset{m\times r}{\mathbf{Y}} \end{array}\right]$$



Forecasting problem

Regression problem is stated as follows:

$$\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x}, \hat{\mathbf{w}}), \text{where } \hat{\mathbf{w}} = \arg\min_{\hat{\mathbf{w}}} S\big(\mathbf{w}|\mathbf{f}(\mathbf{w}, \mathbf{x}), \mathbf{y}\big).$$

The error function $S\big(\mathbf{w}|\mathbf{f}(\mathbf{w}, \mathbf{x}), \mathbf{y}\big)$ averages forecasting errors of $[\mathbf{x}_i|\mathbf{y}_i]$ over all segments $i = 1, \dots, m$ in the test set.

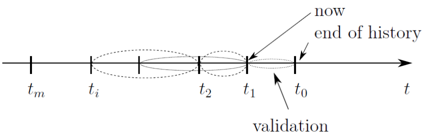
Types of forecasting errors:

- scale-dependent metrics: mean absolute error
$$MAE = \frac{1}{r}\sum_{j=1}^{r}|\varepsilon_j|,$$
- percentage-error metrics: (symmetric) mean absolute percent error
$$MAPE = \frac{1}{r}\sum_{j=1}^{r}\frac{|\varepsilon_j|}{|y_j|}, \quad sMAPE = \frac{1}{r}\sum_{j=1}^{r}\frac{2|\varepsilon_j|}{|\hat{y}_j + y_j|},$$

$\boldsymbol{\varepsilon}$ denotes residual vector

$$\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_r] = \mathbf{y} - \mathbf{f}(\mathbf{w}, \mathbf{x}).$$

Rolling validation



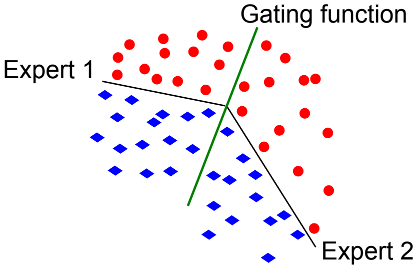
1) Construct the validation vector $\mathbf{x}^*_{\mathrm{val},k}$ for time series of the length Δt_r as the first row of the design matrix $\mathbf{Z}$,
2) construct the rest rows of the design matrix $\mathbf{Z}$ for the time after t_k and present it as

$$\mathbf{Z} = \left[\begin{array}{c|c} \cdots & \cdots \\ \hline \underset{1\times n}{\mathbf{x}_{\mathrm{val},k}} & \underset{1\times r}{\mathbf{y}_{\mathrm{val},k}} \\ \underset{m_{\min}\times n}{\mathbf{X}_{\mathrm{train},k}} & \underset{m_{\min}\times r}{\mathbf{Y}_{\mathrm{train},k}} \\ \hline \cdots & \cdots \end{array}\right], \Big\uparrow_k$$

3) optimize model parameters $\mathbf{w}$ using $\mathbf{X}_{\mathrm{train},k}, \mathbf{Y}_{\mathrm{train},k}$ and compute residues $\varepsilon_k = \mathbf{y}_{\mathrm{val},k} - \mathbf{f}(\mathbf{x}_{\mathrm{val}_k}, \mathbf{w})$ and MAPE,
4) increase k and repeat.

Gating function

Consider there are K models that are used to describe the data. *Gating function* is mapping $\pi_k : \mathbf{x} \mapsto [0, 1]$, which shows the likelihood of k-th model given vector $\mathbf{x} \in \mathbf{X}$.



The gating function:

$$\pi_k(\mathbf{x}, \mathbf{V}) = \frac{\exp(\mathbf{v}_k^\mathsf{T}\mathbf{x})}{\sum_{i=1}^{K} \exp(\mathbf{v}_i^\mathsf{T}\mathbf{x})}, \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K]$$

Mixture of Experts

Assume models $f_1, \dots, f_K$ with gaussian noise:

$$\mathbf{y} = \mathbf{f}_k(\mathbf{x}, \mathbf{w}) + \varepsilon, \quad \mathbf{y} \sim \mathcal{N}(\mathbf{f}_k(\mathbf{x}, \mathbf{w}), \beta_k).$$

Denote the vector of hyperparameters as $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = [w_1, \dots, w_K, \mathbf{V}, \beta]$$

Likelihood of $\mathbf{f}_k$ model on input $(\mathbf{x}, \mathbf{y})$ is $p(k|\mathbf{x}, \mathbf{w})$. Then the $\mathbf{y}$ distribution looks like

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(\mathbf{y}, k|\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(k|\mathbf{x}, \boldsymbol{\theta}) p(\mathbf{y}|k, \mathbf{x}, \boldsymbol{\theta}) =$$

$$= \sum_{k=1}^{K} \frac{\exp(\mathbf{v}_k^{\mathsf{T}} \mathbf{x})}{\sum_{k'=1}^{K} \exp(\mathbf{v}_{k'}^{\mathsf{T}} \mathbf{x})} \exp\left(-\frac{1}{2\beta_k} (\mathbf{y} - \mathbf{f}_k(\mathbf{x}, bw))^2\right).$$

EM algorithm

Let γ_{ik} be the likelihood of $\mathbf{f}_k$ on input $\mathbf{x}_i$, $\mathbf{\Gamma} = [\gamma_{ik}]$. The optimal values of the hyperparameters can be estimated using two iterative steps:

E-step: Fix $\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{V}, \boldsymbol{\beta}$ and recompute matrix

$$\mathbf{\Gamma} = [\pi_1(\mathbf{X}), \ldots, \pi_K(\mathbf{X})].$$

M-step: Re-estimate the parameters using new values of γ_{ik}:

$$\mathbf{v}_k = \arg\max_{\mathbf{v}} \sum_{i=1}^{m} \gamma_{ik}^{r+1} \ln \pi_k(\mathbf{x}_i, \mathbf{v}),$$

$$\mathbf{w}_k = \arg\max_{\mathbf{w}_k} \left[-\sum_{i=1}^{m} \gamma_{ik}^{r+1} \left(\mathbf{y}_i - \mathbf{f}_k(\mathbf{x}_i, \mathbf{w}_k)\right)^2 \right],$$

$$\beta_k = \arg\max_{\beta} \left[n \ln \beta - \sum_{i=1}^{m} \frac{1}{\beta} \left(\mathbf{y}_i - \mathbf{f}_k(\mathbf{x}_i, \mathbf{w}_k)\right)^2 \right].$$

Gating fuction as NN

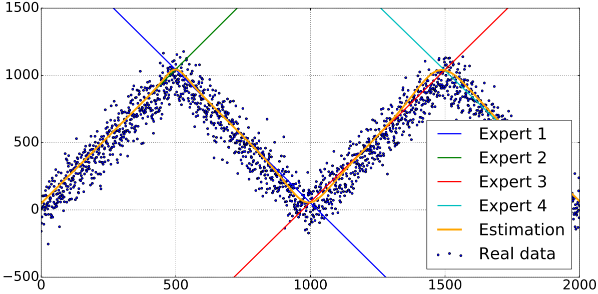
Instead of direct optimization of $\mathbf{V}$ using gradient methods *Neural network* with 3 layers structure is used.
Model $\mathbf{f} = \mathbf{a}(\mathbf{h}_N(\dots\mathbf{h}_1(\mathbf{x})))(\mathbf{w})$ contains autoencoders $\mathbf{h}_k$ and softmax classifier $\mathbf{a}$:

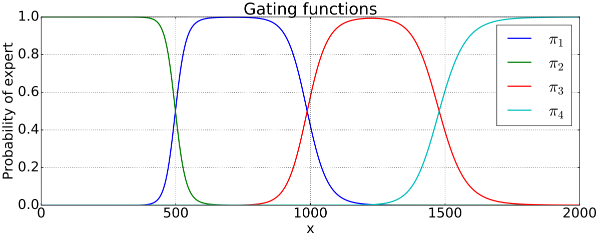
$$\mathbf{f}(\mathbf{w}, \mathbf{x}) = \frac{\exp(\mathbf{a}(\mathbf{x}))}{\sum_j \exp(a_j(\mathbf{x}))}, \qquad \mathbf{a}(\mathbf{x}) = \mathbf{W}_2^{\mathsf{T}}\mathbf{tanh}(\mathbf{W}_1^{\mathsf{T}}\mathbf{x}),$$

$$\mathbf{h}_k(\mathbf{x}) = \sigma(\mathbf{W}_k\mathbf{x} + \mathbf{b}_k),$$

where $\mathbf{w}$ minimizes the error function.

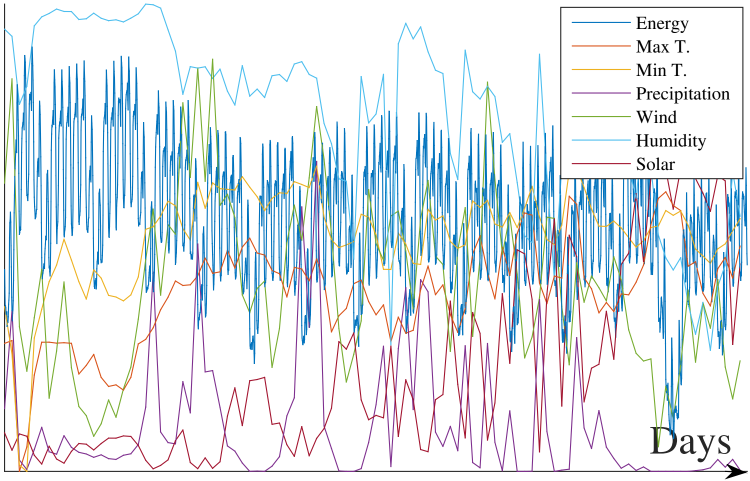
Four linear experts fitting toy data



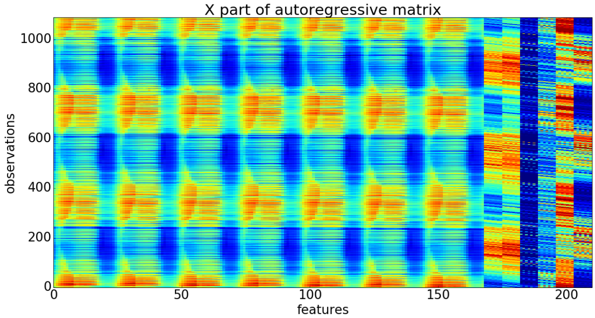


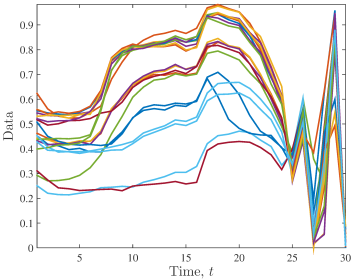
Computational experiment

Data from Poland about energy consumption and weather conditions in 2000-2004.

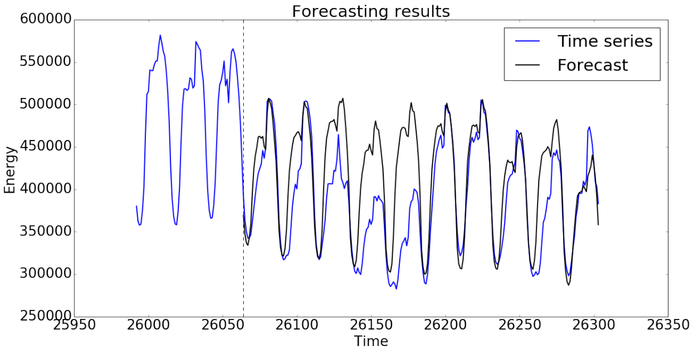


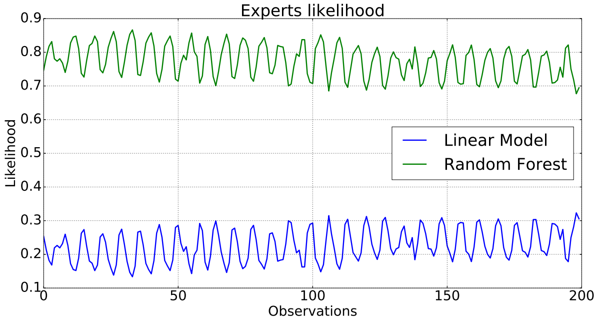
The design matrix.





Target variables for energy consumption time series.


Forecasting results
Energy
Time
Time series
Forecast
600000
550000
500000
450000
400000
350000
300000
250000
25950
26000
26050
26100
26150
26200
26250
26300
26350


Experts likelihood
Likelihood
Observations
Linear Model
Random Forest
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200

Comparison with other models

Model	Train MAE	Test MAE
Random Forest	6680.813	20213.763
MoE (RF+Lin.reg)	8613.395	17640.5
ElasticNet	68185.367	64458.609
Neural network	11274.041	14036.056

Model	Train MAPE	Test MAPE
Random Forest	0.021	0.066
MoE (RF+Lin.reg)	0.026	0.057
ElasticNet	0.229	0.229
Neural network	0.035	0.046

Conclusion

A framework for multiscale time-series forecast is suggested in this project. It allows to test different forecasting techniques on multiple time-series.

Forecasting models are compared to each other and to their expert mixtures. Comparison shows promising results.

Mixture of Experts approach development includes following steps:

- Enhance the convergence of gating function parameters to reach global optimum.
- Consider neural networks of different structure as gating function.