Lecture 8: Advanced Topics: Advanced Forecasting Methods, Ensembling, and Calibration

Introduction to Time Series, Fall 2023

Ryan Tibshirani

1 Advanced forecasters

- Thus far, we've learned in-depth about ARIMA and ETS as our two major forecasting frameworks. Undoutedbly, these are batted-tested frameworks that have been used for decades and will take you far, albeit with proper scrutiny
- Of course, there are actually many other forecasting methods out there, and this continues to be an active topic of research. Below, we briefly summarize three other forecasting methods, chosen based on their popularity (they also seem to constitute a common cast of characters, together with ARIMA and ETS, in R and Python forecasting packages)

1.1 Theta model

- First on our list is the *Theta model*, proposed by Assimakopoulos and Nikolopoulos (2000). This is on our list not as an advanced forecaster per se (as you will see, it's actually quite simple)
- Instead, it is a simple and popular method that began gaining a lot of attention in parts of the forecasting community, and is closely connected to something you already know: exponential smoothing
- Our presentation here follows Hyndman and Billah (2003), who give a nice, clear perspective on the Theta method and its connection to exponential smoothing
- Given data x_t, $t = 1, 2, 3, \ldots$, the Theta method starts by defining a smoothed sequence $y_{\theta,t}$, $t = 1, 2, 3, \ldots$ by the second-order difference equation:

  $$\Delta^2 y_{\theta,t} = \theta \Delta^2 x_t$$

  Here $\Delta^2 = (1 - B)^2$ is the second-order difference operator, just like in our ARIMA lecture, and $\theta \geq 0$ is a parameter. The solution to the above is given by

  $$y_{\theta,t} = a_\theta + b_\theta t + \theta x_t$$

  for some (constant in time) intercept and slope parameters a_θ, b_θ
- For fixed θ, the intercept and slope parameters are fit by minimizing the sum of squared errors to the original sequence

  $$\min_{a_\theta, b_\theta} \sum_{t=1}^n (x_t - y_{\theta,t})^2 \iff \min_{a_\theta, b_\theta} \sum_{t=1}^n \Big((1-\theta) x_t - a_\theta + b_\theta t \Big)^2$$

  which is simply a linear regression of $(1 - \theta)x_t$ on time t
- Once these estimates $\hat{a}_\theta, \hat{b}_\theta$ are found, forecasts are made by running simple exponential smoothing (SES) on the sequence

  $$\hat{y}_{\theta,t} = \hat{a}_\theta + \hat{b}_\theta t + \theta x_t$$

  if $\theta > 0$, or by extrapolating the line $\hat{a}_0 + \hat{b}_0 t$ forward in time if $\theta = 0$

- Assimakopoulos and Nikolopoulos (2000) make the following general recommendation:
  - produce forecasts $\hat{y}_{0,t+h|t}$ with $\theta = 0$ (recall this is just extending the line $\hat{a}_0 + \hat{b}_0 t$);
  - produce forecasts $\hat{y}_{2,t+h|t}$ with $\theta = 2$ (recall this is given by just running SES on $\hat{y}_{2,t}$)
  - return their average: $\hat{y}_{t+h|t} = (\hat{y}_{0,t+h|t} + \hat{y}_{2,t+h|t})/2$.

  Seasonality, if present, is estimated and removed before running this procedure, and added back in at the end. Figure 1 gives a visualization of the canonical "Theta lines": $\hat{y}_{0,t}$ and $\hat{y}_{2,t}$, for $\theta = 0, 2$

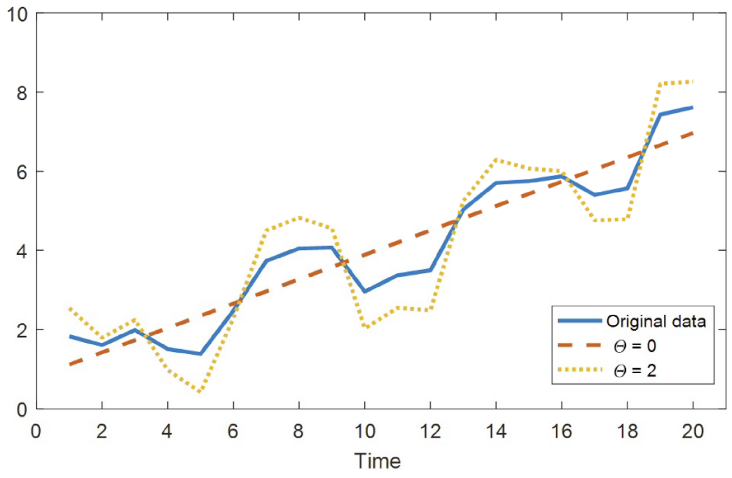

Figure 1: The canonical "Theta lines", from Dudek (2019).

- Hyndman and Billah (2003) show that this procedure is quite similar to running Holt's linear trend method with an estimated slope $\hat{b}_0/2$ and with $\beta = 0$ (no evolution of the slope over time)
- They also compare the Theta method with Holt's linear trend method on the daa from the M3 forecasting challenge (where the Theta method performed well and subsequently gained popularity), and observe that Holt's linear trend performs competitively
- It is worth knowing about the close connections between Theta and exponential smoothing, as much of the literature on the Theta model does not seem to emphasize this aspect. There has been more recent work on the Theta model (which may further distinguish it from exponential smoothing—we cannot say, because we have not followed it) that you may be interested in reading up on

1.2 Prophet model

- Next on our list is the *Prophet model*, proposed by Taylor and Letham (2018), from Facebook. This has become popular for large-scale forecasting enterprises, and the popular opinion seems to be that its advantages over traditional ARIMA or ETS models are twofold: flexibility and speed. But make sure to read on, especially to the end of this subsection, for further discussion of this
- The Prophet model is a particular type of signal plus noise model, where we decompose the signal into three components,

$$x_t = g_t + s_t + h_t + \epsilon_t$$

  Here ϵ_t, $t = 1, 2, 3, \ldots$ is a white noise sequence, and:
  - g_t represents a trend component
  - s_t represents a seasonal component
  - h_t captures holiday/calendar effects

- This can be seen as a particular type of smoother, based on a particular model for the trend (which we will describe shortly; the seasonal and holiday components are fairly generic). We could also refer to it as a particular type of additive model, where the regressor is time
- The seasonal component is parametrized by a Fourier (cosine and sine) basis at given fixed, known periods chosen by the user. For frequencies ω_j, $j = 1, \ldots, p$ (equivalently, periods $1/\omega_j$, $j = 1, \ldots, p$), recall, this is:

$$g_t = \sum_{j=1}^{p} \Big(a_j \cos(2\pi\omega_j t) + b_j \sin(2\pi\omega_j t) \Big)$$

  for coefficients a_j, b_j, $j = 1, \ldots, p$
- The holiday/calendar component is simply parametrized using indicator variables

$$h_t = \sum_{j=1}^{m} \alpha_j \cdot 1\{t \in D_j\}$$

  where α_j, $j = 1, \ldots, m$ are coefficients and each D_j is a set of dates representing a particular holiday or calendar event (e.g., Christmas, Thanksgiving, etc.)
- Finally, the trend component is modeled in one of two ways. The first way is for *saturating* trends. For this, Taylor and Letham (2018) propose to model g_t using a sigmoid function with a piecewise growth rate:

$$g_t = \frac{C(t)}{1 + \exp\Big(c_0 + c_1 t + \alpha \sum_{j=1}^{r} \beta_j \cdot (t - t_j)_+ \Big)}$$

  where $u_+ = \max\{u, 0\}$ denotes the positive-part of u. Here $C(t)$ is a (possibly) time-varying capacity, which it appears Taylor and Letham (2018) recommend be set externally (e.g., based on market size considerations)
- For *non-saturating* trends, Taylor and Letham (2018) propose to model g_t using a piecewise linear trend directly:

$$g_t = c_0 + c_1 t + \alpha \sum_{j=1}^{r} \beta_j \cdot (t - t_j)_+$$

- In either case (saturating or non-saturating), β_j, $j = 1, \ldots, r$ are coefficients to be estimated. Also, t_j, $j = 1, \ldots, r$ are knots (locations where the slope changes), which, in the simplest case, could be fixed ahead of time. Instead, Taylor and Letham (2018) recommend that knots be selected using ℓ_1 penalization from a large initial set of locations. That is, they use the ℓ_1 penalty

$$\sum_{j=1}^{r} |\beta_j|$$

  when fitting the model, which is like a special type of lasso regression. (In fact, though it may not be obvious at first pass, placing an ℓ_1 penalty on β_j, $j = 1, \ldots, r$ here is actually equivalent to reparametrizing the entire sequence as $g_t = \theta_t$, and then using an ℓ_1 penalty on second differences of θ_t; recall, this is the penalization scheme used in *trend filtering*, which you learned earlier in the course when we covered smoothing)
- Altogether, the Prophet model is fit by minimizing the sum of squared errors to the observed data, over all parameters $a_j, b_j, \alpha_j, c_0, c_1, \beta_j$ that determine the decomposition, with a squared ℓ_2 (ridge) penalty on the parameters a_j, b_j, α_j for the seasonal component s_t and holiday component h_t, and an ℓ_1 (lasso) penalty on the parameters β_j for the trend component g_t
- Forecasts are generated by extrapolating the fitted components forward in time. For s_t and h_t, this is straightforward, because they are periodic in nature. For g_t, this is done by holding the slope (i.e., growth rate in the saturating model) constant from its last value, as we move forward in time

- (Though unimportant for our purposes here, Taylor and Letham (2018) actually phrase all of this in the context of a hierarchical Bayesian model, with normal and Laplace priors that serve the purpose of regularization; this Bayesian machinery also provides added stochasticity in computation of prediction intervals)
- So, how does the Prophet model compare to ARIMA and ETS? It depends on who you ask. In their original paper, Taylor and Letham (2018) find that ARIMA and ETS models are too rigid in their motivating examples—take a look at Figure 3 in their paper, and compare Figure 2, which displays Prophet forecasts on the same data

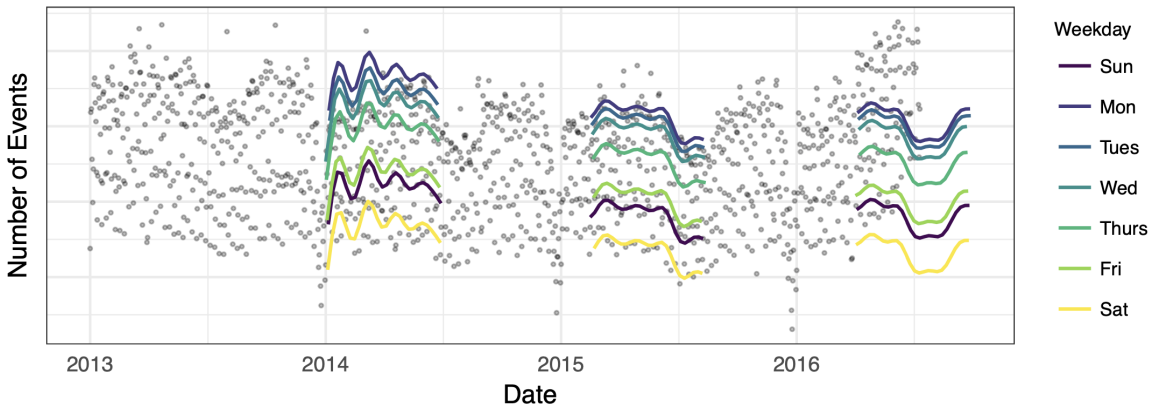

Figure 2: Prophet forecasts, from Taylor and Letham (2018).

- Indeed, in their traditional forms, ARIMA and ETS models lack the flexibility of the Prophet model, particularly the flexibility exhibited in the latter's trend component. However, both ARIMA and ETS can be extended to accommodate more sophisticated trends. With ARIMA, you have actually already seen a way to do this: we can phrase the problem as *regression with correlated errors* (where we use an ARIMA model for the errors), and can then use time for the regressor and invoke a flexible basis representation to model trends just like Prophet does
- Hyndman and Athanasopoulos (HA) call this a *dynamic regression model* and study it in Chapter 10 of their book. The advantage this has over Prophet is that it is able to capture auto-correlations in the errors, which can lead to narrower prediction intervals. The advantage Prophet has is speed: it is usually more efficient to fit the Prophet model, since its error model (white noise) is simpler and this makes the optimization a version of penalized least squares

1.3 Neural network autoregression

- A *neural network* describes a class of models that make real-valued predictions $f(x) \in \mathbb{R}$ from an input feature vector $x \in \mathbb{R}^p$ of the form:

$$
\begin{aligned}
f_1(x) &= \rho(W_1 x + b_1) \\
f_\ell(x) &= \rho\Big(W_{\ell-1} f_{\ell-1}(x) + b_{\ell-1}\Big), \quad \ell = 2, \ldots, L \\
f(x) &= f_L(x)
\end{aligned}
$$

- Here each $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is a matrix of weights that maps from the dimension $d_{\ell-1}$ of layer $\ell - 1$ to the dimension d_ℓ of layer ℓ. Note that $d_0 = p$, and $d_L = 1$ (for real-valued predictions)
- Each $b_\ell \in \mathbb{R}^{d_\ell}$ is a vector of intercepts (often called *biases* in the deep learning community). Generically, the parameters W_ℓ, b_ℓ are all learned by minimizing the sum of squared errors of predictions on the training data

- The function ρ is a nonlinear *activation function* that is interpreted as being applied componentwise. It is user-chosen; common choices are $\rho(u) = u_+$ and $\rho(u) = 1/(1 + e^{-u})$
- Lastly, L here is the number of layers, also a design choice, and often called the *depth* of the network
- For time series, one of the simplest things that can be done with neural networks is just to form input features by taking lags of the given response variable. We can use both nonseasonal and seasonal lags (as in ARIMA). This gives rise to what we call a *neural network autoregressive* (NNAR) model
- What we described above is actually just a particular type of neural network architecture, and indeed, the simplest kind, called a *feedforward* neural network. Many other architectures are possible, and some more appropriate for time series data, such as the *long short term memory* (LSTM) network, a type of recurrent neural network
- A popular time series forecaster based on LSTMs is called *DeepAR*, proposed by Salinas et al. (2020), from Amazon. Relative to all other methods you have learned thus far, DeepAR is quite complicated to describe precisely (and to train). Figure 3 gives a schematic: the left column for training, and the right for forecasting

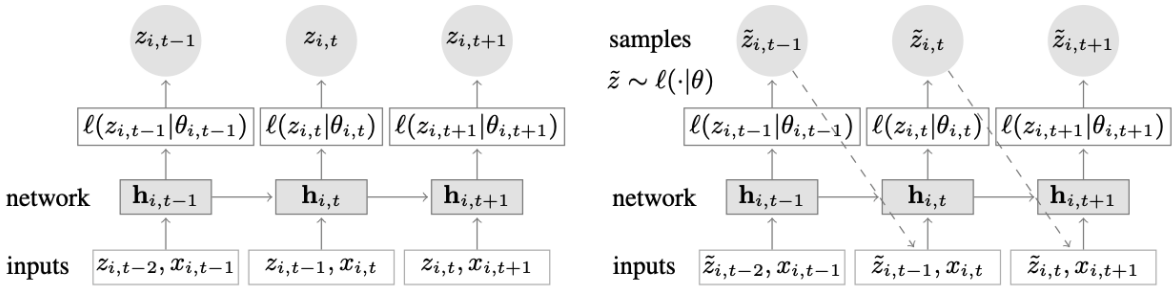

Figure 3: Schematic of DeepAR network, from Salinas et al. (2020), where each $h_{i,t}$ is the output of a recurrent neural network with LSTM cells.

- However, like many deep learning methods, it can work very well in data-rich prediction problems that have a high signal-to-noise ratio. Another strength it is often cited as having is that it can learn rich, joint correlation structures between *multiple* time series—think of spatiotemporal data, where we have one time series per location, and many locations—and it can and produce forecasts which respect this correlation structure
- Perhaps not surprisingly, some authors are now re-purposing transformers in order to turn them into time series forecasters. As deep learning continues to move forward, we will continue to see spillover into time series forecasting

2 Ensembling

- So, you have lagged regression, ARIMA, ETS, Theta, Prophet, and DeepAR in your forecast toolkit. You have data in front of you, and you're unsure which model(s) will work best. What do you do?
- The answer we have given all semester: *use time series CV*. That is, move the time index back to $t = t_0$, train each model on the past, make forecasts, compare to the unseen target values, march the time index forward sequentially, and repeat. In doing so, we can compute our chosen accuracy metric (MAE, MASE, etc.) for each model, and choose the one that emerges as the most accurate, perhaps using Occam's razor to help decide between models that perform similarly
- Here is an alternative that can often be advantageous: *combine* the models, which can be informed by their out-of-sample predictions from time series CV, rather than using CV to select one of them

- The combination of prediction models is often called *ensembling*, or *aggregation*. It has a long history in statistics and machine learning, including in forecasting. Still, it seems perhaps people don't quite appreciate enough just how effective it can be
- Roughly speaking, there are two types of ensembles: *untrained* and *trained* ones. An untrained ensemble simply combines a given set of base models (also called constituent models) without consideration of their individual past performance. Meanwhile, a trained ensemble combines the base models in a way that reflects their performance, typically upweighting more accurate base models
- At a high level, an ideal ensemble model would satisfy the following two properties:
  1. *compete-with-best*: the ensemble should have accuracy that is competitive with the best indvidual constituent model
  2. *robustness-over-all*: the ensemble should have greater robustness (make fewer large errors) than any individual constituent model
- Interestingly (though perhaps not surprisingly), these two are somewhat at odds. Untrained ensembles can be *very* robust, but typically cannot compete with the best constituent model. On the other hand, trained ensembles can compete with the best constituent model (and even outperform it, if the ensemble is flexible enough), but "heavier" training schemes tend to hurt robustness

2.1 Untrained methods

- The simplest untrained ensemble method is just to take a straight average of all constituent model forecasts. Denote these by $\hat{x}^j_{t+h|t}$, for $j = 1, \ldots, p$. Then the simple average ensemble forecast is

$$\hat{x}^{\text{avg}}_{t+h|t} = \frac{1}{p} \sum_{j=1}^{p} \hat{x}^j_{t+h|t}$$

- Another option is to take a median of point forecasts from the constituent models,

$$\hat{x}^{\text{med}}_{t+h|t} = \text{median}\Big\{\hat{x}^j_{t+h|t} : j = 1, \ldots, p\Big\}$$

- Despite their simplicity, these methods can be very effective, especially when we have a diverse set of constituent models (which have a healthy degree of diversity in their errors), as this can lead to very robust average or median ensembles
- Figure 4 shows a striking example of this, from 1- though 4-week ahead Covid-19 death forecasting. The COVIDhub-ensemble model was (for most of the pandemic) a simple median ensemble of all forecasts submitted to the US Covid-19 Forecast Hub which met a certain very basic set of inclusion criteria (ruling out truly wild or unrealistic forecasts). This ensemble model was the basis for all official CDC communications on short-term Covid-19 forecasting. The number of constituent models fluctuated up and down over time, in between about 25 to 50
- For state-level forecasts made over a 1.5 year period during the pandemic, the figure considers a notion called *standardized rank* that is defined as follows. For a given forecast task (given state, given target date, and given horizon), we rank all available forecasts using an accuracy measure called WIS (a generalization of absolute error for quantile-based forecasts) figure. We assign a standardized rank of 1 for the most accurate forecast of all available forecasts for that task, and 0 to the least accurate. Then for each forecaster, we average these standardized ranks over all forecast tasks, and plot a histogram of their standardized rank distribution
- As we can see in Figure 4, the distribution of the ensemble is *very* different from all others—in particular, look at its thin left tail. It is the only forecast model that was in the top half of forecasters (standardized rank ≥ 0.5) for 75% of the forecast tasks. Interestingly, we can also see that it is not in the top quartile of forecasters (standardized rank ≥ 0.75) as often as the best constituent models, hinting at the tradeoff described above

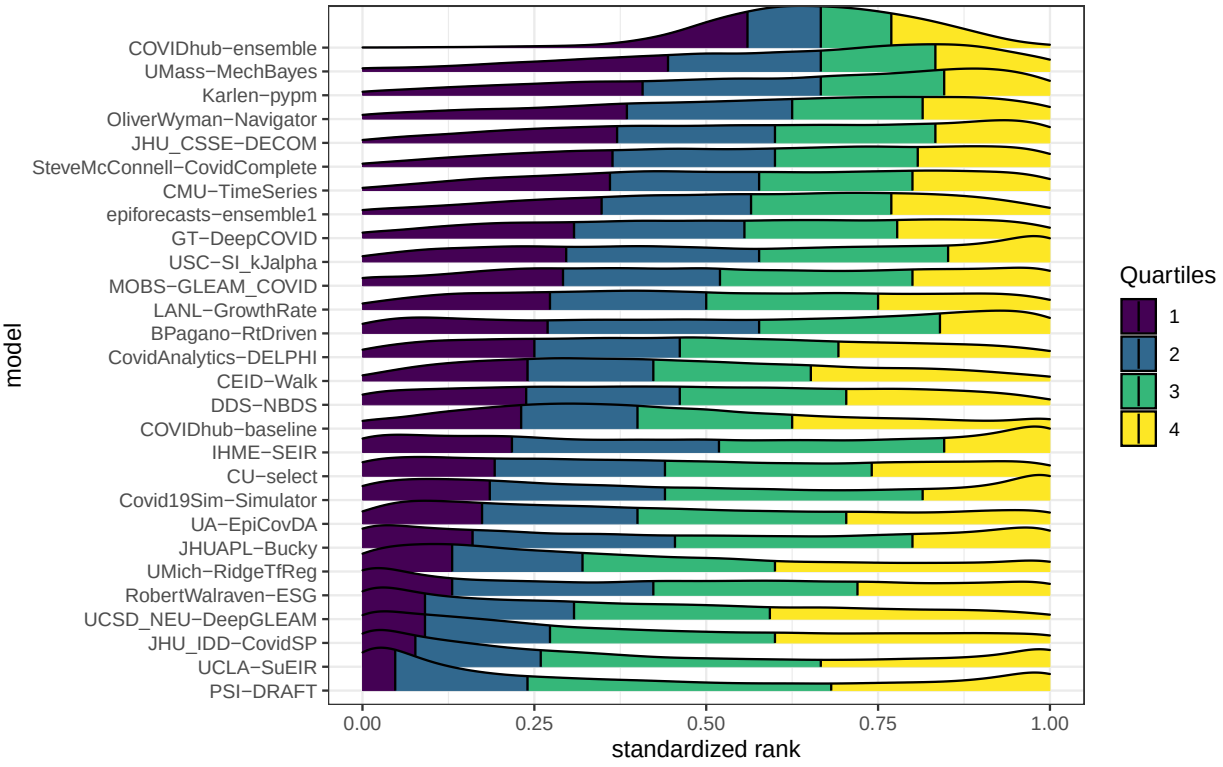

Figure 4: Comparison of Covid-19 death forecast models, from Cramer et al. (2022).

2.2 Trained methods

- One of the simplest trained ensemble methods is to directly fit a weighted combination of constituent forecasts in order to optimize accuracy (MSE, MAE, etc.). For example, going by MSE, to form the ensemble at time t (for a forecast of the target value at time $t+h$), we would first solve

$$\begin{aligned} \min_{w \in \mathbb{R}^p} \quad & \sum_{s=t_0+1}^{t} \Big(x_s - \sum_{j=1}^{p} w_j \cdot \hat{x}^j_{s|s-h} \Big)^2 \\ \text{subject to} \quad & \sum_{j=1}^{p} w_j = 1, \ \text{ and } \ w_j \geq 0, \ j = 1, \ldots, p \end{aligned}$$

  Note that this is simply a regression of the target values (response) on the forecasts made by constituent models (features), where we constrain the coefficients to be nonnegative and sum to 1

- After obtaining the solution $\hat{w}^t_j$, $j = 1, \ldots, p$, we then form the weighted ensemble to make forecasts

$$\hat{x}^{\text{stack}}_{t+h|t} = \sum_{j=1}^{p} \hat{w}^t_j \cdot \hat{x}^j_{t+h|t}$$

- This is sometimes called *linear stacking*: here "stacking" refers to the fact that we have stacked the predictions made by our constituent models into features for a second-level model, and "linear" refers to the fact that our second-level model is linear in the features (it is a linear regression subject to constraints)

- Instead of re-solving the optimization problem above at each forecast time t, we could also incrementally update the learned weights over time. A method called (online) *mirror descent* applied to the

above optimization problem yields the following intuitive weight updates, at time t:

$$\tilde{w}_j^t = \hat{w}_j^{t-1} \exp\left\{\eta \cdot \hat{x}_{t|t-h}^j \left(x_t - \sum_{j=1}^p \hat{w}_j^{t-1} \cdot \hat{x}_{t|t-h}^j\right)\right\}, \quad j = 1, \ldots, p$$

$$\hat{w}_j^t = \tilde{w}_j^t \Big/ \sum_{\ell=1}^p \tilde{w}_\ell^t, \quad j = 1, \ldots, p$$

Here $\eta > 0$ is a parameter called the learning rate. In words, if a forecaster's latest prediction $\hat{x}_{t|t-h}^j$ is correlated with the ensemble's latest residual $x_t - \sum_{j=1}^p \hat{w}_j^{t-1} \cdot \hat{x}_{t|t-h}^j$, then it is underrepresented in the ensemble, and we increase its weight. This is related to a general class of online learning algorithms called *exponential weights* algorithms; and there is a huge literature on related schemes

- Stacking—especially more flexible second-level models that are nonlinear, or use weights that depend on features—can lead to impressive accuracy in practice. Figure 5 shows a nice example of this (albeit in a batch prediction context, rather than time series). The figure shows a neural network ensemble model called *deep quantile aggregation* (DQA), that is trained as a second-layer model on top of many constituent models, including some very flexible ones (such as deep neural networks). Across 35 data sets, ordered by their signal-to-noise ratio along the x-axis, the WIS (recall this is a generalization of absolute error for quantile-based predictions) of all of the constituent models and other ensemble methods is compared to that of DQA. We can see that no method outperforms DQA (nothing below the red horizontal line), and for problems with more signal, towards the right end of the x-axis, DQA improves massively on a lot of the other models

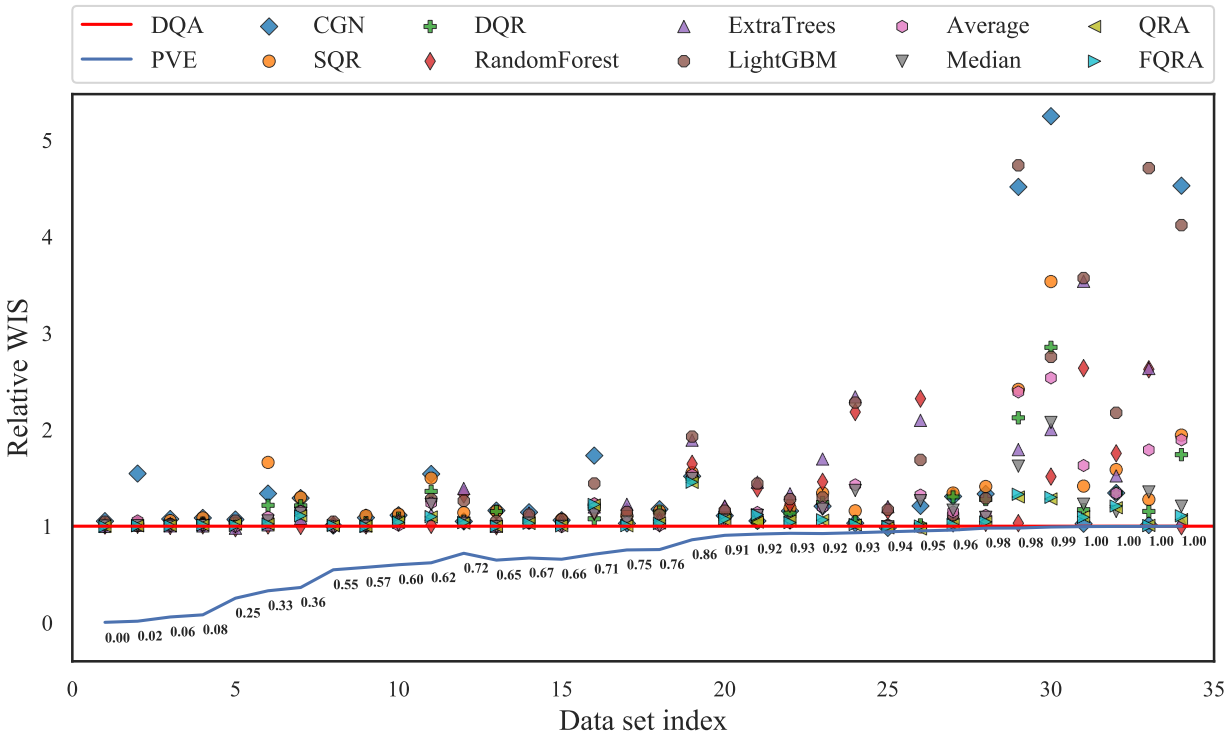

Figure 5: Deep stacking of quantile prediction models, from Fakoor et al. (2023).

3 Calibration

- Beyond point forecasts, all of the methods that we have learned thus far produce *prediction intervals*, which reflect uncertainty in the predictions that are made, useful for downstream decision-making

- How do we measure the accuracy of such prediction intervals? Let's fix a horizon h henceforth. Denote by $I_{t+h|t}^{1-\alpha}$ denote the level $1-\alpha$ prediction interval made by our forecaster at time t, for the target variable x_{t+h} at time $t+h$
- Recall, for a 90% level, this is often (though not necessarily) of the form

$$I_{t+h|t}^{1-\alpha} = \big[\hat{x}_{t+h|t} - \hat{\sigma}_h q_{0.95},\, \hat{x}_{t+h|t} + \hat{\sigma}_h q_{0.95}\big]$$

  where $\hat{x}_{t+h|t}$ is a point forecast of x_{t+h} made a time t, $\hat{\sigma}_h^2$ is an estimate of the variance of the h-step ahead forecast distribution, and $q_{0.95}$ is the 0.95 quantile of the standard normal distritbuion
- Given these intervals, we can measure their empirical *coverage*, in a time series CV sense:

$$\text{Coverage} = \frac{1}{n-t_0} \sum_{t=t_0+1}^{n} 1\big\{x_t \in I_{t|t-h}^{1-\alpha}\big\}$$

  In words, this the fraction of times that our prediction intervals cover their targets. Of course, we want the coverage of our prediction intervals to roughly match the nominal level, Coverage $\approx 1-\alpha$

  n important topic in probabilistic forecasting which we have not really covered thus far is *calibration*.

References

Vassilis Assimakopoulos and Konstantinos Nikolopoulos. The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, 16(4):521–530, 2000.

Estee Y. Cramer, Evan L. Ray, Velma K. Lopez, Johannes Bracher, Andrea Brennen, Alvaro J. Castro Rivadeneira, Aaron Gerding, Tilmann Gneiting, Katie H. House, Yuxin Huang, Dasuni Jayawardena, Abdul H. Kanji, Ayush Khandelwal, Khoa Le, Anja Mühlemann, Jarad Niemi, Apurv Shah, Ariane Stark, Yijin Wang, Nutcha Wattanachit, Martha W. Zorn, Youyang Gu, Sansiddh Jain, Nayana Bannur, Ayush Deva, Mihir Kulkarni, Srujana Merugu, Alpan Raval, Siddhant Shingi, Avtansh Tiwari, Jerome White, Spencer Woody, Maytal Dahan, Spencer Fox, Kelly Gaither, Michael Lachmann, Lauren Ancel Meyers, James G. Scott, Mauricio Tec, Ajitesh Srivastava, Glover E. George, Jeffrey C. Cegan, Ian D. Dettwiller, William P. England, Matthew W. Farthing, Robert H. Hunter, Brandon Lafferty, Igor Linkov, Michael L. Mayo, Matthew D. Parno, Michael A. Rowland, Benjamin D. Trump, Sabrina M. Corsetti, Thomas M. Baer, Marisa C. Eisenberg, Karl Falb, Yitao Huang, Emily T. Martin, Ella McCauley, Robert L. Myers, Tom Schwarz, Daniel Sheldon, Graham Casey Gibson, Rose Yu, Liyao Gao, Yian Ma, Dongxia Wu, Xifeng Yan, Xiaoyong Jin, Yu-Xiang Wang, YangQuan Chen, Lihong Guo, Yanting Zhao, Quanquan Gu, Jinghui Chen, Lingxiao Wang, Pan Xu, Weitong Zhang, Difan Zou, Hannah Biegel, Joceline Lega, Timothy L. Snyder, Davison D. Wilson, Steve McConnell, Robert Walraven, Yunfeng Shi, Xuegang Ban, Qi-Jun Hong, Stanley Kong, James A. Turtle, Michal Ben-Nun, Pete Riley, Steven Riley, Ugur Koyluoglu, David DesRoches, Bruce Hamory, Christina Kyriakides, Helen Leis, John Milliken, Michael Moloney, James Morgan, Gokce Ozcan, Chris Schrader, Elizabeth Shakhnovich, Daniel Siegel, Ryan Spatz, Chris Stiefeling, Barrie Wilkinson, Alexander Wong, Zhifeng Gao, Jiang Bian, Wei Cao, Juan Lavista Ferres, Chaozhuo Li, Tie-Yan Liu, Xing Xie, Shun Zhang, Shun Zheng, Alessandro Vespignani, Matteo Chinazzi, Jessica T. Davis, Kunpeng Mu, Ana Pastore y Piontti, Xinyue Xiong, Andrew Zheng, Jackie Baek, Vivek Farias, Andreea Georgescu, Retsef Levi, Deeksha Sinha, Joshua Wilde, Nicolas D. Penna, Leo A. Celi, Saketh Sundar, Sean Cavany, Guido España, Sean Moore, Rachel Oidtman, Alex Perkins, Dave Osthus, Lauren Castro, Geoffrey Fairchild, Isaac Michaud, Dean Karlen, Elizabeth C. Lee, Juan Dent, Kyra H. Grantz, Joshua Kaminsky, Kathryn Kaminsky, Lindsay T. Keegan, Stephen A. Lauer, Joseph C. Lemaitre, Justin Lessler, Hannah R. Meredith, Javier Perez-Saez, Sam Shah, Claire P. Smith, Shaun A. Truelove, Josh Wills, Matt Kinsey, RF. Obrecht, Katharine Tallaksen, John C. Burant, Lily Wang, Lei Gao, Zhiling Gu, Myungjin Kim, Xinyi Li, Guannan Wang, Yueying Wang, Shan Yu, Robert C. Reiner, Ryan Barber, Emmanuela Gaikedu, Simon Hay, Steve Lim, Chris Murray, David Pigott, B. Aditya Prakash, Bijaya Adhikari, Jiaming Cui, Alexander Rodríguez, Anika Tabassum, Jiajia Xie, Pinar Keskinocak, John Asplund, Arden Baxter, Buse Eylul Oruc, Nicoleta

Serban, Sercan O. Arik, Mike Dusenberry, Arkady Epshteyn, Elli Kanal, Long T. Le, Chun-Liang Li, Tomas Pfister, Dario Sava, Rajarishi Sinha, Thomas Tsai, Nate Yoder, Jinsung Yoon, Leyou Zhang, Sam Abbott, Nikos I. Bosse, Sebastian Funk, Joel Hellewel, Sophie R. Meakin, James D. Munday, Katherine Sherratt, Mingyuan Zhou, Rahi Kalantari, Teresa K. Yamana, Sen Pei, Jeffrey Shaman, Turgay Ayer, Madeline Adee, Jagpreet Chhatwal, Ozden O. Dalgic, Mary A. Ladd, Benjamin P. Linas, Peter Mueller, Jade Xiao, Michael L. Li, Dimitris Bertsimas, Omar Skali Lami, Saksham Soni, Hamza Tazi Bouardi, Yuanjia Wang, Qinxia Wang, Shanghong Xie, Donglin Zeng, Alden Green, Jacob Bien, Addison J. Hu, Maria Jahja, Balasubramanian Narasimhan, Samyak Rajanala, Aaron Rumack, Noah Simon, Ryan J. Tibshirani, Rob Tibshirani, Valerie Ventura, Larry Wasserman, Eamon B. O'Dea, John M. Drake, Robert Pagano, Jo W. Walker, Rachel B. Slayton, Michael Johansson, Matthew Biggerstaff, and Nicholas G. Reich. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the United States. *Proceedings of the National Academy of Sciences*, 119(15):e2113561119, 2022.

Grzegorz Dudek. Short-term load forecasting using theta method. In *E3S Web of Conferences*, volume 84, 2019.

Rasool Fakoor, Taesup Kim, Jonas Mueller, Alexander Smola, and Ryan J. Tibshirani. Flexible model aggregation for quantile regression. *Journal of Machine Learning Research*, 24(162):1–45, 2023.

Rob J. Hyndman and Baki Billah. Unmasking the theta method. *International Journal of Forecasting*, 19(2): 287–290, 2003.

David Salinas, Valentin Flunkert an Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.