Using modern machine learning techniques for improved post-processing of probabilistic NWP forecasts

Stephan Rasp and Sebastian Lerch

December 1, 2017

This will be the abstract.

1 Introduction

Numerical weather forecasts continue to be unreliable and biased (**?**) making post-processing necessary.

1.1 Aims

The main aim of this study is to test modern machine learning techniques for the problem of post-processing probabilistic forecasts. Our aim is not to get the best possible score. For this reason we did not conduct a lot of hyper-parameter tuning.

1.2 Notation

We now introduce the notation we are using throughout the paper. $x_{m,s,t}$ is the 48 h-forecast for a variable, or in machine learning terms a *feature*, x for an ensemble member $m \in \{1, \dots, M\}$, a station $s \in \{1, \dots, S\}$ and a verification time $t \in \{1, \dots, T\}$. In the simplest case the feature vector $\mathbf{x} = (x_1, \dots, x_{N_{\text{features}}})$ contains only the temperature forecasts, either all members or reduced to the mean and the standard deviation. In this study however we also add further variables to $\mathbf{x}$. $y_{s,t}$ is the corresponding observation of 2 m temperature, or the *target*. The post-processed forecasts, or *predictions*, made by our models is denoted by $\hat{y}_{s,t}$. Since we are dealing with probabilistic predictions, our $\hat{y}$ can take the form of a parametric PDF or binned probabilities. In case of temperature a normal distribution is a good approximation. In this case $\hat{y} = \mathcal{N}(\mu, \sigma^2)$ and we are predicting two values, the mean μ and the standard deviationσ. In case of binned probabilities $\hat{y} = (p_1, \dots, p_B)$ for (not necessarily uniform) bins $b \in \{1, \dots, B\}$ under the constraint that $\sum_{b=1}^{B} p_b = 1$. Lastly, we are merging the station and time dimensions into a single sample dimension $i \in \{1, \dots, N_{\text{sample}}\}$.

2 Dataset

For this study, we focused on 2-meter temperature forecasts at synop stations in Germany at a forecast lead time of 48 h. Data is available from 1 January 2008 to 31 December 2016 every day. For details on the data used for training and verification, see Section 6

2.1 ECWMF TIGGE forecast dataset

The forecasts for this study are taken from the TIGGE dataset (**?**). In particular, we used the ECMWF 50-member ensemble forecasts started at 00UTC every day. The data in the TIGGE archive is upscaled to a 40 km horizontal grid. For comparison with the station observations, the gridded data were interpolated to the observation locations using some sort of algorithm.

In addition to the forecasts for the target variable, we have several auxiliary variables.

How are 2m temperatures interpolated?

2.2 DWD SYNOP data

Observations from 537 (is this correct) weather stations in Germany are used (see Fig. ???). Some data points are missing. These data were omitted.

Where did we get it from?

We are not taking into account observation error. Reference for how large that is.

2.3 Data augmentation

3 Traditional post-processing techniques

3.1 EMOS

3.2 Boosting

4 Modern machine learning techniques

4.1 (Neural) networks[1]

Rather than explaining all the basics of neural networks, we refer to two great resources on neural networks (**??**) and keep our discussion brief and visual (Fig???). The parameters of a neural network are trained using stochastic gradient descent (SGD). In particular we are using a version of SGD called Adam (**?**).

The traditional EMOS technique can also be seen from a network perspective (Fig???a). In this case not all the nodes in the input and output layers are connected. It is simple to extend this approach to a fully connected linear network (Fig???b) Which allows to incorporate arbitrarily many input and output nodes. A neural network includes one or more hidden layers with non-linear activation functions. For this study we use the relu activation function $g(z) = \max(0, z)$.

[1] A note on terminology: Many of our simpler models are not neural network or even networks in the traditional sense since they linearly connect the features and variables. In this regard we are basically doing linear regression using stochastic gradient descent. We will nevertheless use the term "network" to describe all models, but only use the term "neural" if there is a hidden layer and a non-linear activation function in the model.

If we want probability bins as an output, we use the Softmax function as an activation for the output layer:

$$g_b(\mathbf{z}) = \frac{e^{z_b}}{\sum_k e^{z_k}} \tag{1}$$

This function squashes an input vector $\mathbf{z}$ to a vector of probabilities between zero and one.

4.1.1 CRPS loss function

As a loss function for our probabilistic forecasts we use the Continuous Ranked Probability Score (CRPS; **?**), which for one forecast-observation pair is defined as

$$\mathrm{crps}(F, y) = \int_{-\infty}^{\infty} [F(t) - H(t-y)]^2 \mathrm{d}t, \tag{2}$$

where $F(t)$ is the forecast CDF and $H(t-y)$ is the Heaviside function. For a Gaussian forecast PDF it is possible to write down an analytic version of the CRPS (**?**):

$$\mathrm{crps}[\mathcal{N}(\mu, \sigma^2), y] = \sigma \left\{ \frac{y-\mu}{\sigma} \left[2\Phi\left(\frac{y-\mu}{\sigma}\right) - 1 \right] + 2\varphi\left(\frac{y-\mu}{\sigma}\right) - \frac{1}{\sqrt{\pi}} \right\}, \tag{3}$$

where $\Phi\left(y-\mu/\sigma\right)$ and $\varphi\left(y-\mu/\sigma\right)$ are the CDF and PDF of $\mathcal{N}(0,1)$ evaluated at the normalized prediction error $(y-\mu)/\sigma$.

4.1.2 Embeddings

Embeddings

4.1.3 Auxiliary variables

Add extra features

4.1.4 Using previous errors as features

Add previous errors

4.2 Gradient boosted trees

We are using XGBoost

5 Things we tried but didn't work

5.1 Recurrent neural nets

6 Results

6.1 Feature importance

Simple linear neural networks and XGBoost allow us to get an idea about how important each feature is.

7 Discussion and conclusion