Predicting the ionosphere

Knut Murå & Seméli Papadogiannakis

Abstract

GPS positioning rely on precise timing of signals sent from satellites to the receiver. Disturbances of the ionosphere change the propagation velocity of GPS signals, and can be one of the largest sources of positioning error. In this report, we attempt to reconstruct one ionosphere parameter measured over Tromsø, Norway, using the GPS errors measured by stations across Sweden operated by the Swedish Maritime Administration (SMA).

1 Introduction

We have attempted to predict ionospheric conditions using Global Positioning system (GPS) errors using machine learning regression. The GPS data consists of about 700 data point. Each data point is the average GPS error and root mean square deviation of the receiver in the latitude, longitude and altitude directions; six numbers per station. Our results will be done using four receivers, located at Bjuröklubb, Järnäs, Kapellskär and Göteborg.

Our ionosphere data is from EISCAT station in Tromsø [1] and is measured by radio, independent of GPS errors. The observable concerned is the electrons in a column through the atmosphere; Total Electron Content or TEC. It is measured in TECU; $10^{16}\mathrm{m}^{-2}$.

Large-scale measurements of TEC is done using GPS, see for example 1. This is done using the phase difference of two signal frequencies, essentially making a Faraday rotation measure. For the purposes of this report, we wish to see if we can reproduce TEC using only the actual position errors.

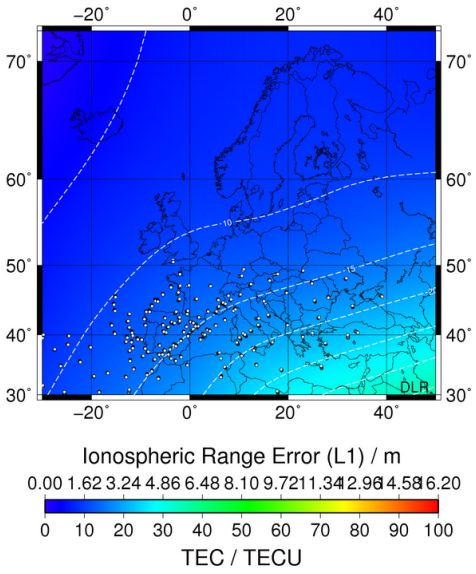


Figure 1: An extrapolated TEC map of Europe by ESA. The dots show the location of the GPS stations used.

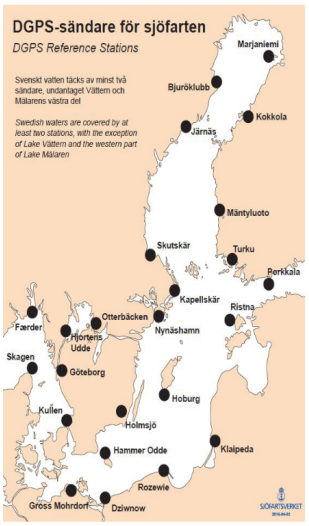


Figure 2: A map of DGPS stations in Sweden operated by SMA

[1] http://dynserv.eiscat.uit.no/

2 Machine learning

This problem uses supervised learning since we aspire to predict one of our variables based on the others. We use the sun angle and the mean squared DGPS errors in x, y, and z for each station in the analysis. In some cases we ommited the sun angle since that seems to have the clearest correlation with TEC to see if any other feature would be picked up. This is further explained in section 3.

Many different regression algorithms could be used for our problem; we have chosen to use linear regression and Support vector machine regression (see sections 2.1 and 2.2), since our results similar and not very predictive in either case we postulate that the results depend on the input data more than the method itself used.

2.1 Linear regression

We used the *scikit learn*[2] implementation of multivariate linear regression and apply it to our data. We have a data set with $6n+1$ parameters, 3 errors for each of n stations and the sun angle. As the distance errors and RMS are of 1m scale, and the cosine is between 0 and 1, no preconditioning is needed. We can write the regression function, y, for the k^{th} sample unit as:

$$y = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + ... + \theta_k x_{ik} + \epsilon_i \tag{1}$$

where θ_i are the regression coefficients and ϵ_i is a random error. We can re-write this in vector form for our N observations:

$$\vec{Y} = Z\vec{\theta} + \vec{\epsilon} \tag{2}$$

One way to estimate the vector θ is by minimising the sum of squared residuals, which is the method we used.

$$(\vec{Y} - Z\vec{\theta})'(\vec{Y} - Z\vec{\theta}) \tag{3}$$

We use cross-validation to choose the hyper-parameters (θ_i) that give us the best fitted function for our data.

The advantage of using linear regression is that the results are very interpretable, we are just minimising the squared residual when fitting a hyperplane on our parameters. If our data is not linearly separable or not well fitted with a hyperplane linear regression, however interpretable, will give us poor predictive power.

2.2 Support vector machine regression

Support Vector Machines (SVM), mostly known for being used in classification, attempts to maximise the distance between the closest data points in two samples, and constructs a hyperplane perpendicularly to the *support vector* between these datapoints. Using *kernels*, a metric function in our data space. To complement the linear regression, the RBF kernel was used: $d_{\text{rbf}} = \exp(-\gamma|\vec{x_1} - \vec{x_2}|^2)$. In regression mode, SVM works similarly, in that *all* data points are required to lie some ϵ from the estimate. The estimate $\hat{y}(x)$ is a constant, plus the inner product between weights and the feature vector x_i: $\hat{y}(x_i) = b + < w, x_i >$. The function to minimize is:[3]:

$$c \equiv \frac{1}{2} \cdot |w|^2 + C \sum (\xi_i + \xi_i^*) \tag{4}$$

$$\text{Given constraint:} \tag{5}$$

$$|y_i - \hat{y}(x_i)| < \epsilon \tag{6}$$

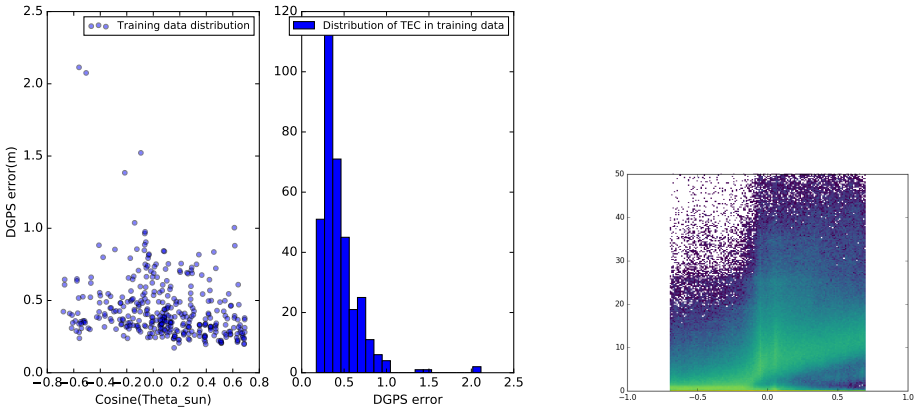
More sophisticated implementations, as the one in `scikit-learn`, will allow some outliers beyond ϵ subject to a (linear) penalty term. A parameter C regulates the relative importance of the two, with $C = 0$ corresponding to the above expression.

[2] `http://scikit-learn.org/stable/index.html`

[3] w is the weights assigned to each variable; adding this term is meant to promote a flat solution in the absence of data

3 Results

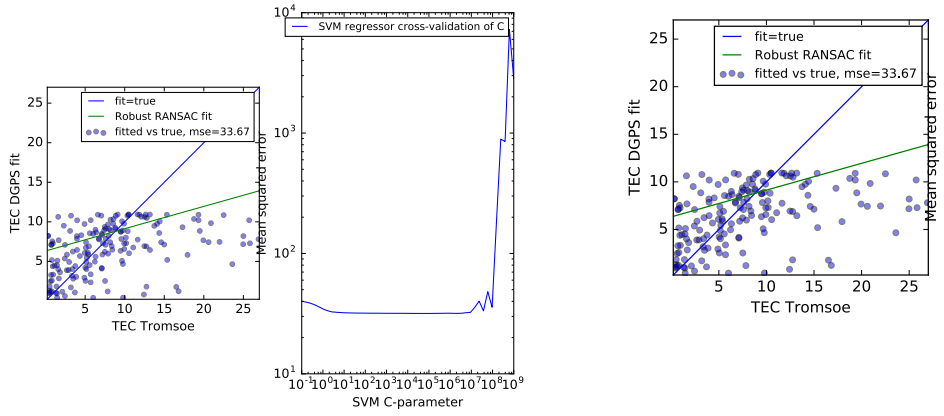
We divide our data to a training set (50 %), testing set (25%) and validation set (25 %). The testing set is used to choose C with cross-validation, while the validation set is used for final plots. Figure 3 shows the distribution of TEC. Note the strong day/night dependence. Figure 4 show fitted vs true data to the left. Some correlation is evident. However, the rightmost plot shows that the same result may be gotten by fitting to the sun angle alone. As measure of performance, the mean squared error between truth and fit is included in the plot, and used for the cross-validation.



(a) TEC in data

(b) TEC vs sun angle

Figure 3: Distribution of TEC in the data, and for a two-year period to the right, clearly showing the day-night variation.



(a) SVM fit

(b) Sun angle only

Figure 4: Fitted vs true TEC for all DGPS(left), and sun angle only(right). In the middle, the cross-validation of C. The large value corresponds to a harsh outlier penalty.

As a comparison, we also attempted to fit the Kapellskär station total position error with the two closest stations. In this case, there is a definite fitted correlation, shown in figure 5

Lastly, we have checked the fit using a linear regression, shown in figure 6:

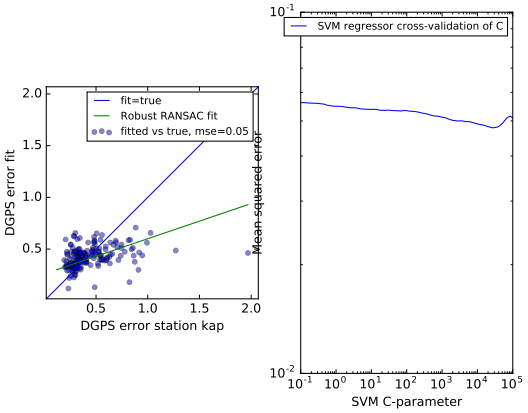


Figure 5: Fit to total DGPS error in kap using two closest stations: sku and nyn. Definitely some correlation there.

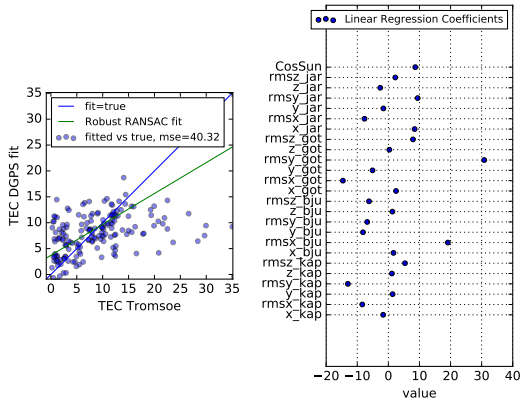


Figure 6: Fitted vs true TEC for all DGPS using a linear fit. To the right, the fitted weights in each dimension is shown. Performance is similar to the `svm` regression.

4 Discussion

Our analysis show that the position error in Sweden is not strongly correlated with `tec` over Tromsø. It should be noted that this is true for the data studied. A more comprehensive dataset would include the comparatively rare geomagnetic storms that are known to affect `gps` accuracy, perhaps lending itself to classification: Is there a geomagnetig storm, or not? The conclusion that `gps`, using four or six satellite fixes is resistant to the northernmost one being affected by the ionosphere is heartening, if not as exiting as the alternative.

Possible further development would be to access the individual phase delays of satellites. That data is used to compute professionally used `tec` maps. Alternatively, one could investigate using lassoo regression and see how many parameters have an impact. Lastly, to see what ionospheric indices does correlate with `gps` accuracy could be performed[4].

Acknowledgements

We would like to thank Jesper Backstedt from Sjöfartsverket for generously providing us with data of DGPS errors, as well as help in parsing the files.

[4]note that `tec` has a comprehensive literature to suggest its importance to `gps` positioning