GNSS Pseudorange Error Characterization and Satellite Selection

Kazuma Gunning

INTRODUCTION

Background

Position, navigation, and timing (PNT) services are a critical part of global infrastructure, and to support this, many nations have built or are building Global Navigation Satellite Systems (GNSS). Among these are the American Global Positioning System (GPS) and Russian GLONASS systems, which are fully operational. The European Union is currently launching a system called Galileo and China has launched most of their system called Beidou. The number of fully operational satellites in orbit is expected to grow from the current approximately 60 to over 120 satellites or more over the course of the next decade. Most consumer GNSS receivers are equipped to track and position based on GPS and perhaps GLONASS. With this setup, a user can typically see somewhere between 10-15 satellites in the sky with varying geometry, and receivers are designed with such a number in mind with regards to the number of signal tracking channels. However, as the number of available satellites increases, the receiver's tracking resources become limited, and the receiver may want to make choices about which satellites to track.

For a user to compute the receiver position (and time), the receiver needs to track signals from at least four satellites. Those satellites are constantly broadcasting their current positions and their time offsets relative to the GPS constellation time. Given the ranging information collected and the knowledge of the precise positions of each satellite position and clock state, the user can solve for its four unknowns of the three position states and one clock state.

The GPS signal is comprised of three major components- the radiofrequency carrier wave (1.575 GHz), the spread spectrum ranging code (1.023 MHz), and the data bits (50 bits per second). The receivers used in this paper output three primary measurements: pseudoranges, carrier phases, and signal to noise ratios. The pseudorange data gives the ranging measurement using the spread spectrum code, and the carrier phase data gives the ranging measurement based on the RF carrier phase. The carrier phase data, while very precise, has an integer ambiguity associated with it and cannot be used as easily for ranging.

At any given moment for a GNSS receiver at sea level, 10-15 GNSS satellites may be in view, but the positioning error produced as a result of using these satellites is not always constant, and this effect is amplified when only a subset of those 10-15 satellites is used. The two factors that go into the total receiver position error are the geometry of the satellites relative to the user and the ranging errors of each of the satellites in view. Poor geometry could be a situation where all of the satellites are in the same direction relative to the user, which gives little information about the user position in the non-line-of-sight direction. It is desirable to have satellites in all directions. There are multiple contributors to the ranging error from a specific satellites: poor tracking of the signal by the receiver due to low signal strength (perhaps due to the satellite being particularly far away or low on the horizon), a bouncing signal that does not go directly from satellite to the user (this is known as multipath), or even bad information being broadcast by the satellite itself. The ranging error from each satellite, combined with the geometry of the situation, can be used to estimate total receiver position error.

Goal

The goal of this study is to use machine learning techniques to estimate the position error produced by the use of a certain subset of GNSS satellites in view taking into account both the geometry of the satellites as well as the ranging errors from each of those satellites. Previous studies [1-4] have used neural networks to estimate the error contribution from geometry only, but this study seeks to extend these ideas by incorporating the differences in ranging error on a per-satellite basis and to use only very simple features to make the aggregate error estimation. Broadly, the problem is that in order to select a desirable subset of GNSS satellites in view, one must iterate through subsets and evaluate performance of each one, and that evaluation traditionally involves a matrix inverse, which is computationally expensive. Rather than use this traditional method, we seek to use machine learning to develop an approximate method that uses simple, computationally inexpensive features to estimate the aggregate expected position error.

DATASET/FEATURES

Measurements

The dataset spawns from logged GNSS receiver data from the International GNSS Service (IGS) network [5]. Each receiver logs data at 30 second intervals. The logs from two days of twenty selected Trimble NetR9 GNSS receivers were used. The logs are in the RINEX format [6].

The data logged at each epoch for each satellite and frequency tracked was the pseudorange, carrier phase, and signal to noise ratio. Pseudoranges are noisy and biased direct range measurements from the receiver to the satellite and are used by essentially every GPS receiver for positioning. The carrier phase will be largely excluded from this study. The signal to noise ratio will be one of the primary features of interest, as a lower SNR generally results in a larger ranging error.

In addition to features extracted from the receiver measurements, a few features from the satellite geometry and satellite navigation message data can be found. First, the elevation angle of the satellite can be found given an approximate location of the receiver and the locations of the satellites in view (given by the navigation message). Second, the navigation message contains a parameter called the User Range Accuracy (URA) [7], which is an estimate of the ranging error due to the error in the navigation message estimate of the satellite clock and position.

The features to be used in the machine learning process are statistics that reflect the selected subset on aggregate rather than the individual measurements of the parameters of the subset. That is, for a subset of six satellites, a feature might be the maximum value of the elevation angles rather than a vector of six elevation angles. This is done to reduce the complexity of the problem and avoid the problem of the order of the satellites not mattering. Most of the full set of features is made up of the maximums, minimums, and standard deviations of the elevation angles (el), azimuths (az), SNRs, and URAs of the satellites in the selected subset. An additional four elements of the feature set have been previously developed [4] and are values built from the user geometry matrix G. The user geometry matrix is an N x 4 matrix that contain the unit line of sight vectors to each satellite in the first three columns and ones in the fourth column [8].

Table 1: Full set of features

$h_1 = tr(G^T G)$	σ_{el}	$\max(SNR)$
$h_2 = tr((G^T G)^2)$	σ_{az}	σ_{SNR}
$h_3 = tr((G^T G)^3)$	$\min(el)$	$\max(URA)$
$h_4 = \lvert G^T G \rvert$	$\max(el)$	$\min(URA)$
$\max(el) - \min(el)$	$\min(SNR)$	

To aid in the training process, Precise Point Positioning (PPP) is used to find the truth position and clock bias for our receiver logs in order to find the ranging error on each of the pseudorange measurements. PPP uses precise estimates of the satellite clock and position states as well as the very precise carrier phase measurements to compute a centimeter-level accurate estimate of the receiver position and clock bias. A PPP service provided by the Canadian Department of Natural Resources [9] was used. One simply submits the receiver measurement log, and the service sends the user an email containing PPP positioning results as well as the measurement residuals, which are the errors that will be used as the output of our training data.

Ranging measurement error

In order use our machine learning techniques to predict the position error for each subset, we must have a "truth" position error. This section develops that truth data on which our predictions are trained and evaluated. Given the user geometry matrix described previously and a covariance matrix R, where the ranging error variances of each satellite in view are uncorrelated and lie on the diagonals, the total covariance of the position error (P) as well as an estimate of the receiver position error (ε) can be described:

$$P = (G^T R^{-1} G)^{-1} \qquad \epsilon = tr(P)^{\frac{1}{2}} \qquad [8]$$

$$R = \begin{bmatrix} \sigma_{sat\ 1}^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_{sat\ N}^2 \end{bmatrix}$$

The receiver position error ε is ultimately the parameter that is to be estimated. We have a dataset that consists now of 20 receivers, where at each point some number of satellites in view, and for each satellite in view we have the SNR, elevation angle, URA, and ranging measurement residual error available. The first step is to predict the ranging error variances for each single ranging signal as a function of the SNR, satellite elevation angle, and URA. The approach to doing this is to use a locally weighted standard deviation estimator:

$$\sigma_{sat\,i} = \sqrt{\frac{1}{\sum w^{(i)}} \sum w^{(i)} (y^{(i)} - \mu)^2}$$

$$w^{(i)} = exp\left(-\frac{\left\|x^{(i)} - x\right\|^2}{\tau^2}\right)$$

Here, x is a vector consisting of the SNR, elevation angle, and URA of the satellite, and y is the ranging error as reported by the PPP results. τ is a bandwidth parameter that was tuned to produce the best distribution normalization. This technique proved to be very successful in normalizing the ranging error distribution and gives us access to the $\sigma_{sat\,i}$. For each subset, the R matrix and thus ϵ can be computed. The final dataset can be produced by, at each epoch in the receiver observation logs, choosing an n-satellite subset of the satellites in view and computing the receiver position error ε using the $\sigma_{sat\,i}$ estimated using the locally weighted standard deviation estimator. For each of these subsets the features listed in Table 1 are recorded as our predictors.

We also set an error threshold, below which a subset is deemed acceptable, and we change this problem into a classification problem. The low cost receivers that this problem is best suited for only need to find a subset of satellites that are "good enough," and do not need necessarily the optimal subset, so a classification problem is appropriate. Ultimately, the problem is now to train machine learning classifiers on the simple, computationally inexpensive features developed in Table 1 and the position error classification of each subset of satellites in view by the IGS receivers.

The error classification threshold is set to be $\varepsilon = \sqrt{15}$ meters, and the subset size is set to be 6 satellites. This classification threshold results in approximately 15% of all subsets resulting in an "acceptable" classification.

MODELS AND METHODOLOGY

Several models were used to attempt to capture the relationship between satellite subset and overall positioning performance: support vector machines, logistical regression, and k-nearest neighbors. For the SVM approach, a linear kernel, a second order polynomial kernel, and a radial basis function kernel were each used and evaluated. Gaussian discriminant analysis was also attempted and gave reasonable results but was ultimately not included in this report. Gaussian discriminant analysis models each class as a separate distribution. This is not an appropriate model for this classification problem, which takes a single distribution and classifies whether or not each element of that distribution lies above a certain threshold.

Each model was tuned using 5000 random training examples, normalized to zero mean and unit variance, and k-fold cross validation with k = 10. The SVM used was L^1-regularized with L^2-loss. Using the linear kernel, an optimal regularization parameter C was found to be 0.031. Similarly, for the 2nd order polynomial and RBF kernels, C parameters were found to be 4 and 1, respectively. For the RBF kernel, the σ was also tuned to be 5. The C parameter found for the linear kernel was much smaller than for the other two kernels, indicating that the linear kernel is less able to perfectly separate the classes, which is one of the results that was found.

The k-NN was similarly optimized. Two distance weighting functions were tried: equal and squared-inverse. The optimal result was found using squared-inverse weighting with 20 neighbors. k-NN is not a particularly computationally cheap method for prediction and is perhaps not a desirable method to be ultimately implemented, but it does a reasonable job of capturing the decision boundaries.

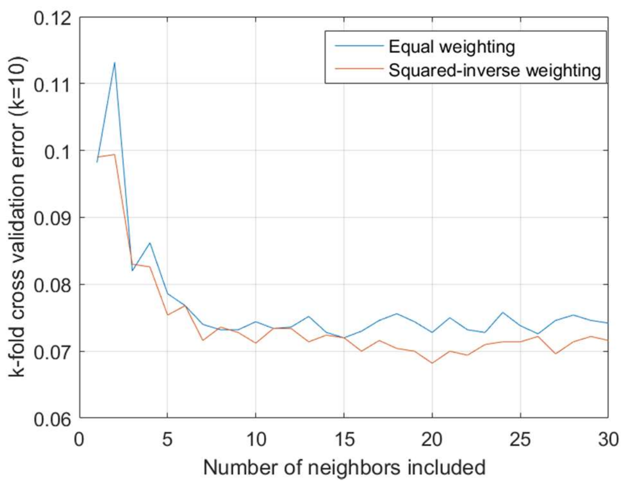

Figure 1: k-NN Tuning: num. of neighbors and distance weighting

The number of training examples used was found by examining Figure 2, where it is clear that the training and test error have converged roughly to steady state across all models. Additionally, the 2nd order polynomial kernel SVM shows some overfitting for lower numbers of training examples, but it seems to be acceptable once the number of training examples exceeds 2000.

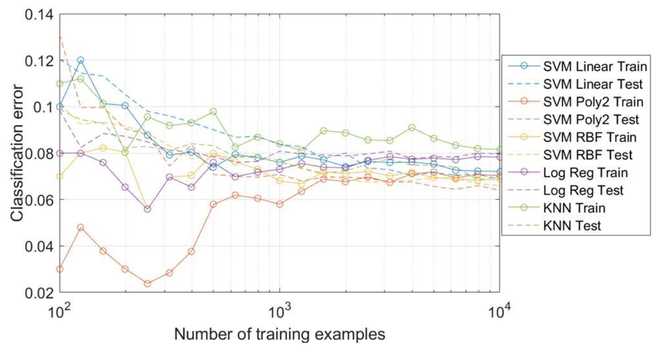

Figure 2: Test and training error vs number of training examples

Feature selection

In order to decrease complexity and overfitting, sequential forward feature selection was performed for each model. Feature selection was also performed using 5000 normalized training examples and k-fold validation with k=10. This was a necessary step, as many of the features were selected intuitively and needed to be screened for usefulness as not to simply introduce extra features for overfitting. Figure 3 shows an example of this process for the RBF SVM.

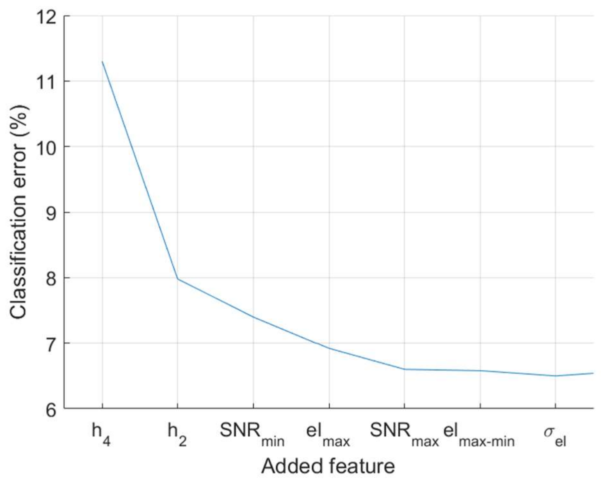

Figure 3: Feature selection for RBF SVM

The features that most commonly were selected across all models are shown in Table 2:

Table 2: Most commonly selected features across models:

h_4
SNR_{min}
el_{max}
h_2
SNR_{max}

Feature selection produced a very nice result that only a handful of features were required to bring the classification error down to the minimum level. Because of this, the overall complexity of the problem can be reduced.

RESULTS

Each of the models was evaluated using the same training set of 5000 examples and a test set of 2500 examples with feature selection and normalization. The misclassification rates are shown in Figure 4 and Table 3. Misclassification was the primary error metric.

Table 3: Model performance

	Training Err.	Test Err.
SVM Linear	0.0662	0.0668
SVM 2nd order poly	0.0542	0.0648
SVM RBF	0.0599	0.0644
Log. Reg.	0.0724	0.0736
k-NN	0.0726	0.070

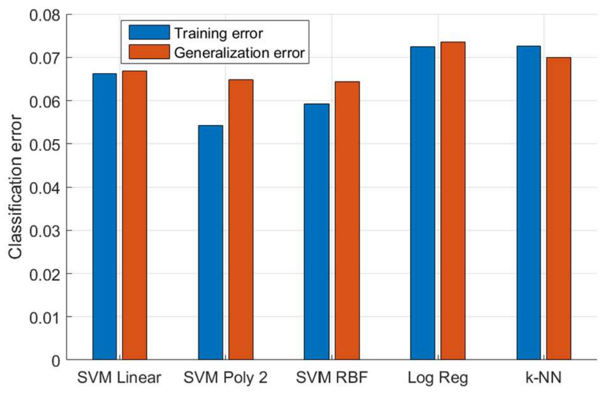

Figure 4: Model performance

The confusion matrix for the RBF result is shown in Figure 5. The training error for k-NN is actually leave-one-out cross validation. This is used because the distance weighting used is squared inverse and thus would always select the same training example and would result in a trivial training error of 0.

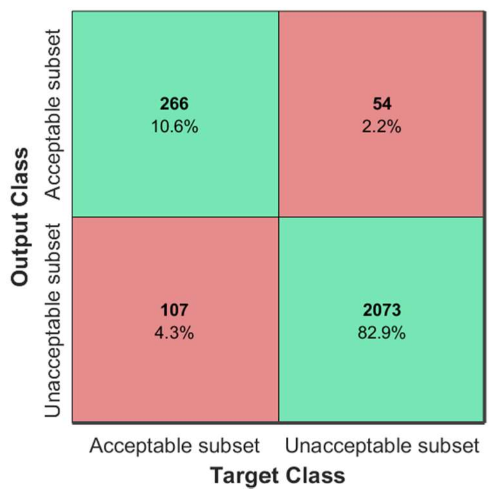

Figure 5: Confusion matrix for RBF SVM model

DISCUSSION

Overall, the classification error results shown in Figure 4 are encouraging. None of the methods give above 8% classification error for true distribution that is approximately 15% positive classifications. The RBF kernel SVM was the highest performing model with respect to test error. The 2nd order polynomial kernel SVM showed lower training error with higher test data, which was also visible in Figure 2. This may indicate that there is still some overfitting occurring, even after feature selection. The linear kernel SVM, logistic regression, and k-NN appear to have trouble capturing the nonlinearities in the data. Examinations of the data show that there is mixing of the classifications near the decision boundary that these models have trouble capturing.

The confusion matrix of the RBF SVM shown in Figure 5 shows a 4.3% false negative rate and 2.2% false positive rate. For our purposes, the false negative rate is not overly worrisome, but the false positive rate could potentially be, as it would lead to a receiver using a poor performing subset. However, a closer look at the false negatives show them all to be near the decision boundary and not significant outliers. It could be possible to predict the bounds for these outliers or simply pull the classification boundary to a low enough level that even these outliers are at an acceptable level.

The features eventually selected revealed that the h_i terms that have been computed in previous studies [4] for similar purposes seem to contain redundant information and are not all necessary for this level of performance. None of the models, after feature selection, utilize all four h terms. This is a great computational performance bonus. Another interesting note regarding the feature selection is that no features related to the URA broadcast by each satellite were incorporated. This is likely because only GPS satellites were used, which are all very high performance and thus have low and similar broadcast URA values. If the Russian GLONASS system were incorporated, which often broadcasts URA values 2-4 times as large as those broadcast by GPS and has similarly larger ranging errors, then the URA features might be selected more often. These might be more useful features for the multi-GNSS solutions that this technique overall would be most useful for. Unsurprisingly, the SNR shows up in the most common features, as the SNR is a key indicator of ranging error. Low signal strength typically indicates a poor ranging accuracy!

Looking forward, there are a number of extensions to this work that could be done. First and foremost, the inclusion of multiple GNSS constellations would be important. Additionally, it might be more useful to look at this problem from the perspective of planning which signals to use over the next hour or other period of time, where given knowledge of the current signal state and the propagated orbital positions, one could select which satellites to track over that period rather than simply selecting which satellites to use at the current epoch. It would be interesting to see what performance gains in both computational time and positioning error are reaped from using this technique, as this paper only sought to evaluate the machine learning classification aspects of the problem. Ultimately, very simple features were used in this study to predict whether or not a specific subset of satellites would produce an acceptable level of position error based on not only the geometry but the specific ranging errors from each satellite.

REFERENCES

[1] D.-J. Jwo and K.-P. Chin, "Applying back-propagation neural networks to GDOP approximation," *Journal of navigation*, vol. 55, no. 01, pp. 97-108, 2002.

[2] D.-J. Jwo and C.-C. Lai, "Neural network-based GPS GDOP approximation and classification," *GPS Solutions*, journal article vol. 11, no. 1, pp. 51-60, 2007.

[3] M. Mosavi and H. Azami, "Applying Neural Network Ensembles for Clustering of GPS Satellites," *International Journal of Geoinformatics*, vol. 7, no. 3, 2011.

[4] D. Simon and H. El-Sherief, "Navigation satellite selection using neural networks," *Neurocomputing*, vol. 7, no. 3, pp. 247-258, 1995.

[5] I. G. Service. *IGS Products*. Available: http://www.igs.org/products

[6] W. G. a. L. Estey, "The Receiver Independent Exchange Format Version 3.00," 2007.

[7] (2013). *IS-GPS-200H*.

[8] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[9] N. R. Canada. *Precise Point Positioning Service*. Available: https://webapp.geod.nrcan.gc.ca/geod/tools-outils/ppp.php?locale=en