Sparse Selection of Training Data for Touch Correction Systems

Daryl Weir, Simon Rogers
University of Glasgow
18 Lilybank Gardens, Glasgow, G12 8QQ
darylw@dcs.gla.ac.uk,
Simon.Rogers@glasgow.ac.uk

Daniel Buschek
University of Munich
Amalienstr. 17, 80333 Munich, Germany
buschek@cip.ifi.lmu.de

ABSTRACT
Touch offset models which improve input accuracy on mobile touch screen devices typically require the use of a large number of training points. In this paper, we describe a method for selecting training points such that high performance can be attained with fewer data. We use the Relevance Vector Machine (RVM) algorithm, and show that performance improvements can be obtained with fewer than 10 training examples. We show that the distribution of training points is conserved across users and contains interesting structure, and compare the RVM to two other offset prediction models for small training set sizes.

Author Keywords
Touch; Machine Learning; Sparse Methods

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): Input devices and strategies (e.g mouse, touchscreen)

INTRODUCTION
Recently, several approaches have been proposed to improve touch accuracy on small devices [4, 3, 7, 9]. Touch accuracy is often low in this domain for various reasons including, for example, the 'fat finger effect', occlusion of the target when performing the touch, and different mental models used by users when targeting [5, 6]. The approaches proposed can be split into two general categories: those that are population based, and those that are user based. Population based methods (such as [3]) take touch data from populations of users and use this to train models that apply a (hopefully) corrective offset to a touch input. Individual based models [9] use data from just one user. [9] showed that these generally outperform models trained on data from several users, due to user-specific touch offsets.

One problem with all of these approaches is the need for touch data on which to train the models. Data from e.g. typing is not typically acceptable as we must be sure where the user was aiming. In [4], a touch-based game was used, whilst [9] used a bespoke calibration application. The requirement for training data represents a hurdle for the use of such methods within mobile devices. The problem is particularly acute for individual based models as we cannot rely on data from other people – all training data must come from the current user. Given that the phone may be used in multiple modes (portrait, landscape, one hand, two hand, thumb, finger, etc), and that each mode will require its own set of training data, the burden on the user becomes far too onerous.

In this paper, we investigate whether it is possible to train individual offset models with very small quantities of data (i.e. fewer than 10 calibration touches). We are not interested, in this study, in finding out how many touches users might tolerate, although this is clearly important. Rather, we are looking to push the number required to train touch models down as low as possible. As well as answering the question 'how many points are required', we will also compare different touch models to see which models perform best in this small data setting. In particular, we propose that it is very important that any touch model, when presented with a very small set of calibration touches shouldn't make the touch problem worse.

Our work here has two strands. In the first, we use recorded touch data to investigate, in a perfect scenario, how few calibration touches can be used to define a model. For this, we use the Relevance Vector Machine (RVM) [8] as it is a regression algorithm (suitable for the touch offset problem) that produces a solution that is sparse in basis vectors (corresponding to training points). The solution is therefore defined with respect to a (typically) small subset of the total training data. This represents an ideal scenario as we perform the analysis on a large touch dataset. However, it provides an indication into how few a number of touches are required to produce improvements in touch accuracy of the baseline of no offset model. The results suggest that good performance can be achieved with fewer than 10 training points and that the location of the chosen calibration points is highly conserved across different users.

The second strand involves investigating how rapidly it is possible to train models starting from no data. In other words, if we sequentially add more data points to the model, how rapidly does the error drop? In [9], improvement was almost immediate when a Gaussian Process regression model was trained using the recorded touch position as an input and the intended touch position as an output. Here, we perform a similar analysis but focus in on the very small dataset sizes and look at data from different devices and different input algorithms. We discover that the RVM can get excellent performance even with datasets so small that it would be impossible to train existing parametric models.

The remainder of the paper is structured as follows. In the next section we describe the regression models used in this work. Following that, we present the results of our two analysis strands and then finish with some discussions and conclusions.

OUR MODEL

The relevance vector machine (RVM) is a technique for obtaining sparse regression and classification functions by linearly weighting a small number of basis functions from a large set of candidates. The RVM is a specialisation of a more general family of machine learning techniques called sparse Bayesian models. Given a set of observed inputs $\mathbf{x}$, the RVM makes predictions of the form

$$y(\mathbf{x}) = \sum_{m=1}^{M} w_m \phi_m(\mathbf{x}).$$

For the touch offset problem, $\mathbf{x}$ are the sensed touch locations, $y(\mathbf{x})$ is the approximation to the offset between the sensed and intended location, $\phi_m(\mathbf{x})$ are the basis functions and w_m are the weights. If we have a set of N training points $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$, the RVM operates by finding values for the weights such that $y(\mathbf{x})$ generalises well to test data but relatively few of the weights are non-zero. In this way, only a small proportion of the basis functions are used.

Touch Data

We evaluated the RVM on an existing set of touch data [2]. This data was gathered in a user study where users were shown a series of 300 crosshair targets on a smartphone and asked to touch them as accurately as possible. Participants were seated and held the phone in one hand, touching using the thumb on that hand. The dataset contains touches from 30 subjects on 13 different smartphones. All subjects repeated the study on multiple phones, but not all subjects used all phones. In this paper, we analyse 43200 touches from the 3 phones for which most data was collected — the iPhone 4 (25 subjects), Nokia N9 (24) and Nokia Lumia 900 (22).

Making Predictions

To train an RVM using this data, we chose the basis functions $\phi_m(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_m)$, where $K(\mathbf{x}, \mathbf{x}_m)$ is the radial basis function kernel between the test point $\mathbf{x}$ and the training example $\mathbf{x}_m$. In this way, we have a number of basis functions equal to the number of training points — 300 for each session. The basis functions given non-zero weights by the RVM (the 'relevance vectors') are those points which are important for making predictions of the intended target for new test touches.

The predictive function from the RVM is one-dimensional — that is, although (x_1, x_2) training pairs are used to compute the basis functions, the RVM can only predict either x_1 or x_2. Thus, we need to train two models for each session, each of which will have different relevance vectors. This is potentially interesting, as it allows us to to find which areas of the screen we need to collect data from to predict each of the dimensions, and also which points are used to predict both x_1 and x_2.

RESULTS

Predictive Performance

For each session, we train the two RVMs and evaluate their predictive performance. We perform 5-fold cross validation, holding out 60 touches for testing at each step. Our performance metric is the RMS error between the predicted location of the RVM and the intended touch location (position of the crosshair in the trial). As a baseline, we compare against a second order polynomial of the form $y = w_0 + w_1x_1 + w_2x_1^2 + w_3x_2 + w_4x_2^2$, fitted using least squares regression. This allows us to investigate how a sparse solution performs in comparison to a model learned on all the available data.

In [3] the authors learn an offset function using a fifth order polynomial. However, we found that the typical number of relevance vectors was small enough that it was not possible to train a well conditioned fifth order model (such a model would require 11 datapoints at least, due to the 11 features required). The number of relevance vectors was typically less than 5, hence our choice of a quadratic baseline, which can be trained with 5 points.

Our results are summarised in Table 1. We present results for the three most common devices in the dataset. For each device, the table shows the mean number of points n selected by the RVM to predict each dimension, along with the average RMS error across all sessions for the two models. We also show the RMS error between the recorded and intended locations in the absence of any correction.

Number of Points Required – the Ideal Case

We can see that for each device, the RVM typically selects between 2 and 5 relevance vectors for each dimension. That is, we need to evaluate the kernel function at fewer than 10 training points from the original 240 to make good predictions on test data. We see that the RVM and the polynomial both perform better than the baseline. This difference is statistically significant (paired t-test, $p < 0.05$). We also see that the RVM provides a small performance improvement over the polynomial model. For all but the x_1 conditions on the iPhone and N9, this improvement is also significant. The RVM performs as well or better than a model with many times more training data. Note that this minimum number of points is something that we may not be able to meet in practice as the RVM analysis has many points available to start with — a situation that is clearly not feasible in practice.

The RVM analysis also allows us to explore where these points are positioned in the input space. This can give insight into which areas of the screen are important when learning the offset function. In order to do this, we use a kernel density estimate to produce a probability density function of the

Phone	Dimension	n	RVM	Quadratic (all data)	Baseline (No correction)
iPhone 4	x_1	2.28	0.0289	0.029	0.0444
	x_2	3.28	0.0435	0.0444	0.0532
Nokia N9	x_1	2.52	0.0363	0.0365	0.0523
	x_2	3.02	0.0371	0.0375	0.0538
Nokia Lumia 900	x_1	2.70	0.0307	0.0313	0.0456
	x_2	4.65	0.0335	0.0340	0.0431

Table 1. Average RMS errors between the predicted and intended touch location for various phones and predictive models.

point location for each phone type. For a given phone, we collect all relevance vectors extracted for all users and sessions on that phone, and place a narrow Gaussian at each of these points. To estimate the density at a new point $\mathbf{x}'$, we take the average of all of the training point densities evaluated at $\mathbf{x}'$. By evaluating the density estimate at each point in a grid, we can visualise the surface over the screen. Figure 2 shows the density of the points used to make predictions on the iPhone 4 for both horizontal and vertical co-ordinate predictions.

There is a clear structure evident, showing which areas of the screen are important for making predictions. It is interesting to note that the points which are most important for predicting x_1 are distributed quite differently from those for x_2. For predicting x_1, the most important areas are at the right and left edges of the screen, whereas to predict x_2 the most important points are in the corners.

Generalising to New Users

The analysis above represents an ‘ideal world’ situation, in which we start with a fairly large dataset of touches for a user and use the RVM to extract the relevant training points. This is not the desired use case in general — we wish to learn a model for a new user which requires only a small number of training points.

An algorithm designed to work with a small training set needs to satisfy two properties. First, it must be stable. That is, when there is very little data (less than 5 examples) the predictions should not be worse than the baseline where no offset correction is performed. Secondly, the performance should improve quickly as training data is added. We have seen that the RVM produces good predictive performance using a small set of basis vectors, so we now investigate the performance of the algorithm when starting from no data.

To conduct the analysis, we take the training examples for a user, permute them into a random order, and add them to the training set in small increments. We train an RVM at each increment and compute the predictive performance on a held out test set. In addition, we also train a quadratic model using the same points and a least squares regression based on the kernel matrix passed to the RVM. We evaluate the performance of the models using 1, 2, 5, 10, 15 and 20 training points. Our performance measure is the root mean square error between the model predictions (x'_1, x'_2) and the true positions (x_1, x_2).

Figure 1 shows the results averaged across all users and phones. We rescale device coordinates to a unit square to

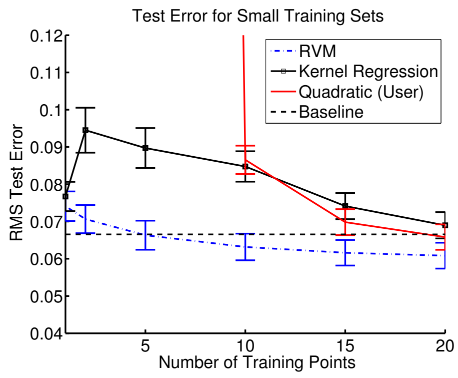
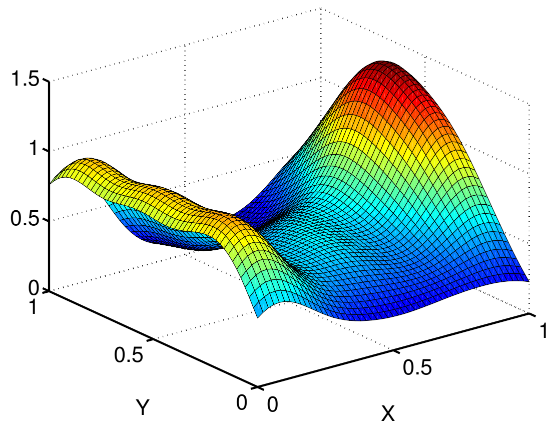


Figure 1. RMS error as a function of training set size for three prediction models. Results show mean and standard error across all subjects. The baseline is a quadratic model trained on the data for all subjects.

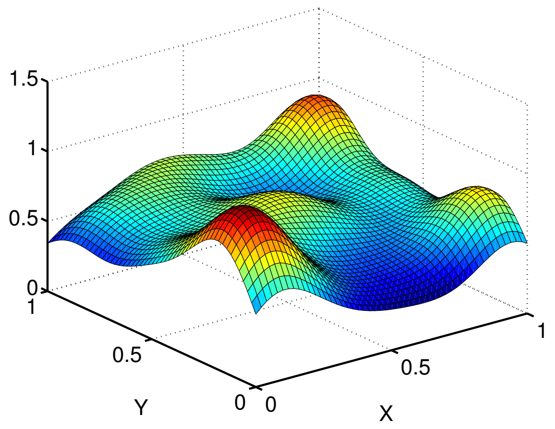
allow cross-device comparison. The performance value for each user was obtained by averaging the RMSE over 20 restarts of the process described above. This helps ensure we use a variety of subsets of the training data.

The plotted curves show the mean test RMS and standard error across all users. To provide a comparison to a more traditional population approach, we also fitted a quadratic predictor based on all of the collected training data. The performance of this baseline is shown by the dashed line. We can see that the RVM error is equivalent to the baseline with only 5 training points, and better with 10. The parametric models, by contrast, have high error for these values of n. The quadratic model in particular requires 5 features and so for fewer than 5 training points the model is poorly conditioned and the error is very large. For the RVM, the average improvement over the baseline at 10 training points was small but statistically significant (paired t-test, $p < 0.05$). This analysis shows the power of the RVM for very quickly building offset models that offer an improvement over commercial hardware and existing correction strategies.

It is interesting to note that the user specific quadratic model exceeds the performance of the population model for 20 or more training points. This is in line with the findings of [9], which suggested that user specific offset models performed better than population based techniques. When all the available training data are used, the RVM and quadratic models give similar performance. However, the RVM approaches the optimal performance more quickly, with little improvement after 50 training points, whereas the quadratic model improves steadily as data are added.



(a) X Relevance Vectors



(b) Y Relevance Vectors

Figure 2. Kernel density estimates for the distribution of relevance vectors on the iPhone 4 in portrait orientation. Higher values of the distribution indicate the location of points important in predicting touch offset surfaces. The important points for predicting X and Y offsets are quite different.

DISCUSSION AND FUTURE WORK

The RVM allows us to train offset models with a very small number of training points. This is potentially very useful to train these models in real time as a new user interacts with a device. User specific offset models are desirable [9] but existing offset correction systems have used hundreds or thousands of training points — too many for a typical calibration. The RVM offers a solution to this problem, since it is much more feasible to collect 5 or 10 training touches than 200.

We have also seen that the RVM can be used to produce a distribution of training point locations for these models. These distributions are potentially of interest for interaction design. The points where the distributions are peaked are important precisely because they are harder to model, and so this could be used to inform the placement of interface elements.

The small number of training points required by our model suggests the offset surface is quite simple. This may be because the data set used consists of taps from only one thumb. Two thumb tapping behaviour is likely to be more complex, and studying this is an important direction for future work. Touch behaviour changes based on the hand used, and also based on posture and grip[1]. A practical deployment of our system would need to be able to identify the current usage mode. [2] showed the currently used hand could be inferred from tapping behaviour — this might be used with our algorithm to determine when to train a new offset model.

CONCLUSION

In this paper, we have investigated the use of the RVM algorithm to learn a touch offset function using a sparse set of training data. We used the RVM to learn a minimal representation based on a full set of data, and explored the distribution of the important examples. We also studied the use of the algorithm to quickly learn offset models starting from no data. We found small but significant improvements over a polynomial baseline after only 5 or 10 training points.

The RVM, and sparse methods in general, are a powerful tool for improving the quality of touch screen interaction. This paper has demonstrated the utility and potential of this algorithm to quickly produce user specific touch models.

ACKNOWLEDGMENTS

Nokia provided equipment and funding to support this work. Daryl Weir is supported by a SICSA prize scholarship.

REFERENCES

1. Azenkot, S., and Zhai, S. Touch behavior with different postures on soft smartphone keyboards. In *MobileHCI '12*, 251–260.
2. Buschek, D., Rogers, S., and Murray-Smith, R. User-Specific Touch Models in a Cross-Device Context. In *MobileHCI '13*, To appear.
3. Henze, N., Rukzio, E., and Boll, S. 100,000,000 taps: Analysis and Improvement of Touch Performance in the Large. In *MobileHCI '11*, 133–142.
4. Henze, N., Rukzio, E., and Boll, S. Observational and experimental investigation of typing behaviour using virtual keyboards for mobile devices. In *CHI '12*, 2659–2688.
5. Holz, C., and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *CHI '10*, 581–590.
6. Holz, C., and Baudisch, P. Understanding touch. In *CHI '11*, 2501–2510.
7. Rogers, S., Williamson, J., Stewart, C., and Murray-Smith, R. AnglePose: robust, precise capacitive touch tracking via 3d orientation estimation. In *CHI '11*, 2575–2584.
8. Tipping, M. E. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research 1* (2001), 211–244.
9. Weir, D., Rogers, S., Murray-Smith, R., and Lochtefeld, M. A User-Specific Machine Learning Approach for Improving Touch Accuracy on Mobile Devices. In *UIST '12*, 465–476.