Robust Logistic Regression using Shift Parameters

Julie Tibshirani and Christopher D. Manning
Stanford University
Stanford, CA 94305, USA
{jtibs, manning}@cs.stanford.edu

Abstract

Annotation errors can significantly hurt classifier performance, yet datasets are only growing noisier with the increased use of Amazon Mechanical Turk and techniques like distant supervision that automatically generate labels. In this paper, we present a robust extension of logistic regression that incorporates the possibility of mislabelling directly into the objective. Our model can be trained through nearly the same means as logistic regression, and retains its efficiency on high-dimensional datasets. Through simulations as well as experiments on natural language data, we demonstrate that our approach can provide a significant improvement over the standard model when annotation errors are present.

1 Introduction

Almost any large dataset has annotation errors, especially those complex, nuanced datasets commonly used in machine learning research. Low-quality annotations have become even more common in recent years with the rise of Amazon Mechanical Turk, as well as methods like distant supervision and co-training that involve automatically generating training data.

Although small amounts of noise may not be detrimental, in some applications the level can be high: upon manually inspecting a relation extraction corpus commonly used in distant supervision, Riedel et al. (2010) report a 31% false positive rate. In cases like these, annotation errors have frequently been observed to hurt performance. Digare et al. (2005), for example, conduct error analysis on a system to extract relations from biomedical text, and observe that over half of the system’s errors could be attributed to inconsistencies in how the data was annotated. Similarly, in a case study on co-training for natural language tasks, Pierce and Cardie (2001) find that the degradation in data quality from automatic labelling prevents these systems from performing comparably to their fully-supervised counterparts.

Despite this prevalence, little work has been done in designing models that are aware of annotation errors. Moreover much of the previous work focuses on heuristic techniques to filter the data before training, which might discard valuable examples simply because they do not fit closely with the model assumptions.

In this work we argue that incorrect examples should be explicitly modelled during training, and present a simple extension of logistic regression that incorporates the possibility of mislabelling directly into the objective. Our model introduces sparse ‘shift parameters’ to allow datapoints to slide along the sigmoid, changing class if appropriate. It has a convex objective, can handle high-dimensional data, and we show it can be efficiently trained with minimal changes to the logistic regression pipeline.

Experiments on large, noisy NLP datasets show that our method can provide an improvement over standard logistic regression, both in manually and automatically annotated settings. The model also provides a means to identify which examples were mislabeled: through experiments on biological data, we demonstrate how our method can be used to ac-
curately identify annotation errors. This robust extension of logistic regression shows promise in handling annotation errors, while remaining efficient on large, high-dimensional datasets.

2 Related Work

Much of the previous work in dealing with annotation errors centers around filtering the data before training. These approaches can be roughly split into two categories: supervised and unsupervised filtering.

Brodley and Friedl (1999) introduce what is perhaps the simplest form of supervised filtering. In this method the training data is split into folds, and each fold is cleaned one at a time. To clean a fold $f$, various classifiers are trained on the remaining folds, then their predictions on $f$ are compared to the provided labels. If too many classifiers disagree with an example’s current label, then that example is marked as incorrect and eliminated from the training set. Finally, a single classifier is trained over the remaining instances. Verbaeten and Assche (2003) extend this idea to other ensemble methods like bagging, while Freund (2000) proposes a variant of boosting that effectively ‘gives up’ on frequently misclassified examples as it trains. In a similar vein Venkataraman et al. (2004) filter using SVMs, training on different subsets of the feature space to create multiple ‘views’ of the data.

One obvious issue with these methods is that the noise-detecting classifiers are themselves trained on noisy labels. Such methods may suffer from well-known effects like masking, where several mislabelled examples ‘mask’ each other and go undetected, and swamping, in which the mislabelled points are so influential that they cast doubt on the correct examples (She and Owen, 2011). Unsupervised filtering tries to avoid this problem by clustering training instances based solely on their features, then using the clusters to detect labelling anomalies (Rebbapragada et al., 2009). Rebbapragada and Brodley (2007) present an interesting extension to this approach in which these cluster assignments are converted into class probabilities, then used as instance weights in training the final classifier.

Unsupervised filtering, however, relies on the perhaps unwarranted assumption that examples with the same label lie close together in feature space. Moreover filtering techniques in general may not be well-justified: if a training example does not fit closely with the current model, it is not necessarily mislabeled. Although the instance itself has a low-likelihood, it might represent an important exception that would improve the overall fit. An example may also appear unusual simply because we have made poor modelling assumptions, and in discarding it valuable information could be lost.

Perhaps the most promising approaches are those that directly model annotation errors, handling mislabelled examples as they train. This way, there is an active trade-off between fitting the model and identifying suspected errors. Bootkrajang and Kaban (2012) present an extension of logistic regression that models annotation errors through flipping probabilities. For each example the authors posit a hidden variable representing its true label, and assume this label has a probability of being flipped before it is observed. While intuitive, this approach has shortcomings of its own: the objective function is non-convex and the authors note that local optima are an issue, and the model can be difficult to fit when there are many more features than training examples.

In the realm of statistics, researchers have long been interested in methods that can handle outliers.
Called ‘robust statistics’, the field seeks to develop estimators that are not unduly affected by deviations from the model assumptions \cite{Huber2009}. Since mislabelled points are one type of outlier, this goal is naturally related to our interest in dealing with noisy data, and it seems many of the existing techniques would be relevant. In a recent advance, \cite{She2011} demonstrate that introducing regularized ‘shift parameters’ can help increase the robustness of linear regression, and \cite{Candes2009} propose a similar technique for principal component analysis. It is this idea that inspired the present work.

3 Model

Recall that in binary logistic regression, the probability of an example $x_i$ being positive is modeled as

$$g(\theta^T x_i) = \frac{1}{1 + e^{-\theta^T x_i}}$$

For simplicity we assume the intercept term has been folded into the weight vector $\theta$, so $\theta \in \mathbb{R}^{m+1}$ where $m$ is the number of features.

We propose the following robust extension: for each datapoint $i = 1, \ldots, n$, we introduce a real-valued shift parameter $\gamma_i$ so that the sigmoid becomes

$$g(\theta^T x_i + \gamma_i) = \frac{1}{1 + e^{-\theta^T x_i - \gamma_i}}$$

Since we believe that most examples are correctly labelled, we $L_1$-regularize the shift parameters to encourage sparsity. Letting $y_i \in \{0, 1\}$ be the label for datapoint $i$ and fixing $\lambda \geq 0$, our objective is now given by

$$l(\theta, \gamma) = \sum_{i=1}^{n} \left[ y_i \log g(\theta^T x_i + \gamma_i) \right. + (1 - y_i) \log (1 - g(\theta^T x_i + \gamma_i)) \left. \right] + \lambda \sum_{i=1}^{n} |\gamma_i|$$

These parameters $\gamma_i$ let certain datapoints to shift along the sigmoid, perhaps switching from one class to the other. If a datapoint $i$ is correctly annotated, then we would expect its corresponding $\gamma_i$ to be zero. If it actually belongs to the positive class but is labelled negative, then $\gamma_i$ might be positive, and analogously for the other direction.

One way to interpret the model is that it allows the log-odds of select datapoints to be shifted. While it perhaps does not have as natural an interpretation as a model based on label-flipping, handling errors in this way allows us to retain much of logistic regression’s simplicity.

It is worth noting that there is no difficulty in regularizing the $\theta$ parameters as well. For example, if we choose to use an $L_1$ penalty then our objective becomes

$$l(\theta, \gamma) = \sum_{i=1}^{n} \left[ y_i \log g(\theta^T x_i + \gamma_i) \right. + (1 - y_i) \log (1 - g(\theta^T x_i + \gamma_i)) \left. \right] + \lambda \sum_{i=1}^{n} |\gamma_i|$$

Finally, it may seem concerning that we have introduced a new parameter for each datapoint. But in many applications the number of features already exceeds $n$, so this increase is actually quite reasonable.

3.1 Training

Notice that adding these shift parameters is equivalent to introducing $n$ features, where the $i$th new feature is 1 for datapoint $i$ and 0 otherwise. With this observation, we can simply modify the design matrix and parameter vector and train the logistic model as usual. Specifically, we let $\theta' = (\theta_0, \ldots, \theta_m, \gamma_1, \ldots, \gamma_n)$ and $X' = [X|I_n]$ so that the objective simplifies to

$$l(\theta') = \sum_{i=1}^{n} \left[ y_i \log g(\theta'^T x'_i) \right. + (1 - y_i) \log (1 - g(\theta'^T x'_i)) \left. \right] + \lambda \sum_{j=m+1}^{m+n} |\theta'^{(j)}|$$

Upon writing the objective in this way, we immediately see that it is convex, just as standard $L_1$-penalized logistic regression is convex.

One small complication is that the parameters corresponding to $\gamma$ are now regularized, while those corresponding to $\theta$ are not (or perhaps we wish to regularize them differently). In practice this situation does not pose much difficulty, and in the ap-
in selecting $\lambda$ in simulations that the errors largely do not interfere in evaluations may require more careful treatment, as will collections in which labels are flipped uniformly at random. The $p_0$ column gives the probability of class 0 flipping to 1, and vice versa for $p_1$.

Notice that $\lambda$ directly controls the number of nonzero shifts $\gamma$ and hence the suspected number of errors in the training set. So if we have information about the noise level, we can directly incorporate it into the selection procedure. For example, we may believe the training set has no more than 15% noise, and so would restrict the choice of $\lambda$ during cross-validation to only those values where 15% or fewer of the estimated shift parameters are nonzero.

We now consider situations in which the $\theta$ parameters are regularized as well. Assume, for example, that we use $L_1$-regularization as in equation (2). We would then need to optimize over both $\kappa$ and $\lambda$. In cases like these it is common to first construct a one-dimensional family, so we can then cross-validate a single parameter (Friedman et al., 2009; Arlot and Celisse, 2010). In addition to being faster to compute, this method gives more accurate estimates of the true error rate.

Concretely, we perform the following procedure:

1. For each $\kappa$ of interest, find the value of $\lambda$ that, along with this choice of $\kappa$, maximizes the robust model’s accuracy on the train set.

2. Cross-validate to find the best choice for $\kappa$, using the corresponding values for $\lambda$ found in the first step.

### Table 1: Accuracy of standard vs. robust logistic regression for various levels of noise.

| $p_0$ | $p_1$ | regularized | standard | robust |
|-------|-------|-------------|----------|--------|
| 0.0  | 0.0  | no          | 96.56 ± 0.09 | 96.60 ± 0.10 |
| 0.1  | 0.0  | no          | 93.48 ± 0.18 | 93.58 ± 0.18 |
| 0.2  | 0.0  | no          | 87.49 ± 0.24 | 89.22 ± 0.23 |
| 0.3  | 0.0  | no          | 80.40 ± 0.25 | 84.15 ± 0.28 |
| 0.3  | 0.1  | no          | 84.16 ± 0.35 | 86.63 ± 0.33 |
| 0.3  | 0.0  | yes         | 75.89 ± 0.50 | 76.98 ± 0.56 |
| 0.3  | 0.1  | yes         | 74.98 ± 0.56 | 76.16 ± 0.57 |

Note that it is fine to choose $\lambda$ based on training accuracy, since it is not used in making predictions and so there is little risk of overfitting.

For large, high-dimensional datasets even this procedure may be too costly, and training accuracy is not always informative. In the natural language processing experiments below, we adopt a simpler strategy:

1. Cross-validate using standard logistic regression to select $\kappa$.

2. Fix this value for $\kappa$, and cross-validate using the robust model to find the best choice of $\lambda$.

Although not as well-motivated theoretically, this method still produces reasonable results.

### 4 Experiments

We now present several experiments to assess the effectiveness of the approach, ranging from simulations in which labels are flipped uniformly at random, to experiments on natural language datasets where annotation errors are quite systematic.

#### 4.1 Simulated Data

In our first experiment, we simulate logistic data with 10 features drawn Uniform(-5, 5), letting $\theta_j = 2$ for $j = 1, \ldots, m$ and the intercept be zero. We create training, development, and test sets containing 500 examples each and introduce noise into both the training and development sets by flipping labels uniformly at random. The regularization parameter $\lambda$ is chosen simply by minimizing 0-1 loss on the development set. For all simulation experiments we use glmnet, an R package that trains both...
method | suspects identified | false positives
---|---|---
Alon et al. (1999) | T2 T30 T33 T36 T37 N8 N12 N34 N36 | | T6, N2
Furey et al. (2000) | • • • • • • | T8, N2, N28, N29
Kadota et al. (2003) | • • • • • | |
Malossini et al. (2006) | • • • • • • • | |
Bootkrajang et al. (2012) | • • • • • • • | |
robust LR | • • • • • • • | |

Table 2: Results of various error-identification methods on the colon cancer dataset. The first row lists the samples that are biologically confirmed to be suspicious, and each other row gives the output from an automatic detection method. Bootkrajang et al. report confidences, so we threshold at 0.5 to obtain these results.

The results for standard versus robust logistic regression are shown in Table 1, for various levels of noise.

Using the tuning procedure described in Section 3.3, we next perform simulations in which the original features are $L_1$-penalized as well (see Table 1). We generate logistic data with 20 features, only 5 of which are relevant, and again set $\theta_j = 2$ for $j = 1, \ldots, m$ and the intercept to zero. The training, development, and test sets are each of size 100, and label noise is added to all data but the test set. The regularization parameter for the baseline model is tuned on the development set. Additional implementation details can be found in the appendix.

As the results show, robust logistic regression provides a consistent improvement over the baseline. The performance difference grows larger with the amount of label noise, and is also evident when labels have been flipped in both directions. A one-dimensional example of this improvement is seen in Figure 1.

We also note that the nonzero shift parameters have a significant connection to which training examples are mislabelled: in a typical run with $p_0 = 0.2, p_1 = 0.0$ the model identifies 53 out of 57 flipped labels, albeit with fairly low precision. (Upon inspection, the false positives are largely unusual cases from the generating distribution that are difficult to distinguish from truly mislabelled examples.)

Importantly, the model does not perform worse than standard logistic regression when no errors are present. Inspecting the learned parameters, we see that almost all $\gamma$ have been set to 0.

| partition | # pos | # neg | # features | $p_0$ | $p_1$
|---|---|---|---|---|
| train | 24,002 | 200,000 | 393,633 | 0.004 | 0.075
| test | 3,149 | 48,429 | 125,062 | - | -

Table 3: Statistics about the data used in Wikipedia NER experiments. The $p_0$ column represents the fraction of examples that the majority agreed were negative, but that the chosen annotator marked positive (and analogously for $p_1$). We still include examples for which there was no majority consensus, so these noise estimates are quite conservative.

## 4.2 Contaminated Data

We next apply our approach to a biological dataset with suspected labelling errors. Called the colon cancer dataset, it contains the expression levels of 2000 genes from 40 tumor and 22 normal tissues (Alon et al., 1999). There is evidence in the literature that certain tissue samples may have been cross-contaminated. In particular, 5 tumor and 4 normal samples should have their labels flipped.

Since the dataset is so small, it is difficult to accurately measure the performance of our model against the baseline. We instead examine its ability to identify mislabelled training examples. Because there are many more features than datapoints and it is likely that not all genes are relevant, we choose to place an $L_1$ penalty on $\theta$.

Using glmnet, we again select $\kappa$ and $\lambda$ using the cross-validation procedure from Section 3.3. Looking at the resulting values for $\gamma$, we find that only 7 of the shift parameters are nonzero and that each one corresponds to a suspicious datapoint. As further confirmation, the sign of the gammas correctly match the direction of the mislabelling. Compared to previous attempts to automatically detect errors
| model   | precision | recall  | F1     |
|---------|-----------|---------|--------|
| standard| 76.99     | 85.87   | 81.19  |
| robust  | 77.04     | 90.47   | 83.22  |

Table 4: Performance of standard vs. robust logistic regression in the Wikipedia NER experiment.

In this dataset, our approach identifies at least as many suspicious examples but with greater consistency and fewer false positives. A detailed comparison is given in Table 2.

### 4.3 Manually Annotated Data

In these experiments we focus on a classic task in NLP called *named entity recognition*. In the traditional set-up, the goal is to determine whether each word is a person, organization, location, or not a named entity (‘other’). Since our model is binary, we concentrate on the task of deciding whether a word is a person or not. (This task does not trivially reduce to finding the capitalized words, as the model must distinguish between people and other named entities like organizations).

For training, we use a large, noisy NER dataset collected by Jenny Finkel. The data was created by taking various Wikipedia articles and giving them to five Amazon Mechanical Turkers to annotate. Few to no quality controls were put in place, so that certain annotators produced very noisy labels. To construct the train set we chose a Turker who was about average in how much he disagreed with the majority vote, and used only his annotations. Negative examples are subsampled to bring the class ratio to a reasonable level (around 1 to 10). We evaluate on the development test set from the CoNLL shared task ([Tjong Kim Sang and Meulder, 2003]). This data consists of news articles from the Reuters corpus, hand-annotated by researchers at the University of Antwerp. More details about the dataset can be found in Table 3.

We extract a set of features using Stanford’s NER pipeline ([Finkel et al., 2005]). This set was chosen for simplicity and is not highly engineered – it largely consists of lexical features such as the current word, the previous and next words in the sentence, as well as character n-grams and various word shape features. We choose to $L_2$-regularize the features, so that our penalty now becomes

$$
\frac{1}{2\sigma^2} \sum_{j=0}^{m} |\theta_j|^2 + \sum_{i=1}^{n} |\gamma_i|
$$

This choice is natural as $L_2$ is the most common form of regularization in NLP, and we wish to verify that our approach works for penalties besides $L_1$.

The robust model is fit using Orthant-Wise Limited-Memory Quasi Newton (OWL-QN), a technique for optimizing an $L_1$-penalized objective ([Andrew and Gao, 2007]). We tune both models through 5-fold cross-validation to obtain $\sigma^2 = 1.0$ and $\lambda = 0.1$. Note that from the way we cross-validate (first tuning $\sigma$ using standard logistic regression, fixing this choice, then tuning $\lambda$) our procedure may give an unfair advantage to the baseline.

The results of these experiments are shown in Table 4 as well as Figure 2. Robust logistic regression offers a noticeable improvement over the baseline, and this improvement holds at essentially all levels of precision and recall.

### 4.4 Automatically Annotated Data

We now turn to a setting in which training data has been automatically generated. The task is the same as in the previous experiment: for each word in a sentence we must identify whether it represents a person or not. For evaluation we again use the de-
Table 5: Statistics about the NER data generated by MUSE. The $p_0$ column gives the fraction of examples that are marked positive in the official CoNLL train set, but that MUSE labelled negative ($p_1$ is defined analogously).

| partition | # pos | # neg | # features | $p_0$ | $p_1$ |
|-----------|-------|-------|------------|-------|-------|
| train     | 8,392 | 80,000| 190,185    | 0.371 | 0.007 |
| test      | 3,149 | 48,429| 125,062    | -     | -     |

Table 6: Performance of standard vs. robust logistic regression in the Muse NER experiment.

|       | model  | precision | recall | F1    |
|-------|--------|-----------|--------|-------|
| standard | 84.52  | 70.91     | 77.12  |
| robust  | 84.64  | 72.44     | 78.06  |

As for the training data, we take the sentences from the official CoNLL train set and run them through a simple NER system to create noisy labels. We use a system called MUSE, which makes use of gazetteers and hand-crafted rules to recognize named entities (Maynard et al., 2001). The software is distributed with GATE, a general purpose set of tools for processing text, and is not tuned for any particular corpus (Cunningham et al., 2002). We have again subsampled negatives to achieve a ratio of roughly 1 to 10. More information about the data can be found in Table 5. Somewhat expectedly, we see that the system has a high false negative rate.

5 Discussion

As evidenced in the experiments section, the robust model can provide improvement in certain situations. We now consider in more detail the factors that affect its performance.

5.1 Distribution of Errors

In the simulation experiments from Section 4.1, the robust model offers a notable advantage over the baseline if the features are uniformly distributed. But when we rerun the experiments with features drawn Normal(0, 1), the improvement in accuracy decreases by as much as 1%. One explanation is as follows: in this situation, the datapoints, and therefore the annotation errors, tend to cluster around the border between positive and negative. Logistic regression, by virtue of its probabilistic assumptions, is naturally forgiving toward points near its decision boundary. So when noise is concentrated at the border, adding shift parameters does not provide the same benefit. In short, the robust model seems to perform best when there is a good number of mislabelled examples that are not close cases.

We also notice in the NER experiments that the robust model shows less of an improvement when the training data is generated automatically rather than manually. One likely explanation is that more than human annotators, rule-based systems tend to make mistakes on examples with similar features. For example if a certain word was not in MUSE’s gazetteers, and so it incorrectly labelled every instance of this word as negative, we might have a good number of erroneous examples that are close together in feature space. In this setting can be hard for any robust classifier to learn what is mislabelled.

5.2 Cross-Validating with Noisy Data

We previously noted that in simulation, cross-validating on noisy data did not pose an issue. In fact, experiments confirm that the procedure selected nearly the same parameter values as it would have with a clean development set.

In working with the NER data, however, we observed that tuning on the noisy train set may be suboptimal. This observation held even for the MUSE
experiments, where both the train and test data come from the Reuters corpus. Here, cross-validation resulted in a large value for $\lambda$, so that nearly all $\gamma$ parameters were zero. It was not until we used our knowledge of the error rate to guide the choice of $\lambda$ that we obtained a fit making use of the shift parameters. With clean held-out data the algorithm would likely have chosen this smaller value, as the robust model ended up outperforming the baseline on the error-free test set.

It therefore seems that if labels are flipped uniformly at random, then cross-validating with noisy data is acceptable. But in situations like the NER experiments where errors are systematic, the procedure can result in suboptimal parameter values. A useful way to characterize the distinction may be whether or not an example’s label being flipped depends on the values of its features.

6 Aside: Comparison to SVMs

It is interesting to observe the similarity between this model and a soft-margin SVM:

$$\min_{w,\xi,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

s. t. $\forall i \ y_i(w^T x_i - b) \geq 1 - \xi_i$, $\xi_i \geq 0$

The $\gamma$ parameters correspond to slack variables $\xi_i$, which allow certain datapoints to lie on the wrong side of the separating hyperplane. As in our model, these slack variables are $L_1$-penalized to promote sparsity.

The relationship suggests that an SVM can be helpful in identifying mislabelled examples, and indeed this technique has been successfully used in the past (Nakagawa and Matsumoto, 2002). However soft-margin SVMs may still struggle to learn in the presence of errors: the mislabelled points actually become support vectors and so have a large influence on the decision boundary. This potential sensitivity to noise is also supported empirically (Xu et al., 2006; Wu and Liu, 2007).

7 Future Work

A natural direction for future work is to extend the model to a multi-class setting. One option is to introduce a $\gamma$ for every class except the negative one, so that there are $n(c-1)$ shift parameters in all. We could then apply a group lasso, with each group consisting of the $\gamma$ for a particular datapoint (Yuan and Lin, 2007; Meier et al., 2008). This way all of a datapoint’s shift parameters drop out together, which corresponds to the example being correctly labelled. A simpler approach is to use one-vs-all classification: we train one binary robust model for each class, and have them vote on an example’s label. We have found preliminary success with this method in a relation extraction task.

As mentioned in the related work section, She and Owen (2011) demonstrate that shift parameters can help increase the robustness of linear regression. The fact that a similar approach works for logistic regression suggests that shift parameters may prove useful for other GLMs. Since the extra parameters can be neatly folded into the linear term, convexity is preserved and each model could be trained as usual.

Finally, we noted in the discussion that cross-validating on noisy data may pose a problem when the annotation errors are systematic. It is an important area of future work to determine precise conditions under which standard cross-validation is acceptable, and in cases where it is not, to provide a sound alternative.

8 Conclusion

We presented a robust extension of binary logistic regression that can outperform the standard model when annotation errors are present. Our method introduces shift parameters to allow datapoints to move across the decision boundary. It largely maintains the efficiency and scalability of logistic regression, but is better equipped to train with noisy data and can also help identify mislabelled examples.

As large, noisy datasets continue to gain prevalence, it is important to develop classifiers with robustness in mind. Most promising seem to be models that incorporate the potential for mislabelling as they train. We presented one such model, and demonstrated that explicitly accounting for annotation errors can provide significant benefit.

Acknowledgments

We acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Read-
ing Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government. We are especially grateful to Rob Tibshirani and Stefan Wager for their invaluable advice and encouragement.

References

U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, A. J. Levine. 2009. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *National Academy of Sciences of the United States of America*, 96(12):6745–6750

Galen Andrew and Jianfeng Gao. 2007. Scalable Training of $L_1$-Regularized Log-Linear Models. *ICML*, 24:33–40

Sylvain Arlot and Alain Celisse. 2010. A survey of cross-validation procedures for model selection. *Statistical Surveys*, 4:40–79

Jakramate Bootkrajang and Ata Kaban. 2012. Label-noise Robust Logistic Regression and Its Applications. *ECML PKDD*, 143–158

Carla E. Brodley and Mark A. Friedl. 1999. Identifying Mislabeled Training Data. *JAIR*, 11:131–167.

Emmanuel J. Candès, Xiaodong Li, Yi Ma, John Wright. Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablaan. 2009. Robust Principal Component Analysis? *arXiv preprint arXiv:0912.3599*

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablaan. 2002. GATE: an Architecture for Development of Robust HLT Applications. *ACL*

Shipra Dingare, Malvina Nissim, Jenny Finkel, Christopher Manning, and Claire Grover. 2005. A system for identifying named entities in biomedical text: How results from two evaluations reflect on both the system and the evaluations. *Comparative and Functional Genomics*, 6:77–85

Jenny Rose Finkel, Trond Grenager, Christopher Manning. 2005. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *ACL*, 43:363–370

Yoav Freund. 2000. An adaptive version of the boost by majority algorithm. *COLT*, 12:102–113.

Jerome Friedman, Trevor Hastie, Rob Tibshirani. 2009. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software*, 33(1): 122

Terrence S. Furey, Nello Cristianini, Nigel Duffy, David W. Bednarski, Michel Schummer, David Haussler. 2000. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914

Peter J. Huber and Elvezio M. Ronchetti. 2000. *Robust Statistics*. John Wiley & Sons, Inc., Hoboken, NJ.

Koji Kadota, Daisuke Tominaga, Yutaka Akiyama, Katsutoshi Takahashi. 2003. Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample. *ChemBio Informatics Journal*, 3(1):3045

Andrea Malossini, Enrico Blanzieri, Raymond T. Ng 2006. Detecting potential labeling errors in microarrays by data perturbation. *Bioinformatics*, 22(17), 2114–2121

Diana Maynard, Valentin Tablan, Cristian Ursu, Hamish Cunningham, Yorick Wilks. 2001. Named Entity Recognition from Diverse Text Types *Recent Advances in Natural Language Processing*

Lukas Meier, Sara van de Geer, Peter Bühlmann. 2008. The group lasso for logistic regression. *Journal of the Royal Statistical Society*, 70(1):53–71

Tetsuji Nakagawa and Yuji Matsumoto. 2002. Detecting Errors in Corpora Using Support Vector Machines. *COLING*

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. *EMNLP*

Umaa Rebbapragada and Carla E. Brodley. 2007. Class Noise Mitigation Through Instance Weighting. *ECML*, 18:708–715

Umaa Rebbapragada, Lukas Mandrake, Kiri L. Wagtstaff, Damhnait Gleeson, Rebecca Castano, Steve Chien, Carla E. Brodley. 2009. Improving Onboard Analysis of Hyperion Images by Filtering Mislabeled Training Data Examples. *IEEE Aerospace Conference*

Sebastian Riedel, Limin Yao, Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. *ECML PKDD*

Yiyuan She and Art Owen. 2011. Outlier Detection Using Nonconvex Penalized Regression. *Journal of the American Statistical Association*, 106(494):626–639

Erik F. Tjong Kim Sang, Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition *CoNLL*, 142–147

Sundara Venkataraman, Dimitris Metaxas, Dmitriy Fradkin, Casimir Kulikowski, Ilya Muchnik. 2004. Distinguishing Mislabeled Data from Correctly Labeled Data in Classier Design. *IEEE International Conference on Tools with Artificial Intelligence*, 16:668–672.
Sofie Verbaeten, Anneleen Van Assche. 2003. Ensemble Methods for Noise Elimination in Classification Problems. *International Conference on Multiple Classifier Systems* 4:317–325.

Yichao Wu and Yufeng Liu. 2003. Robust Truncated Hinge Loss Support Vector Machines. *Journal of the American Statistical Association* 102(479):974–983.

Ming Yuan, Li Yin. 2007. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, 68(1):49–67.

Linli Xu, Koby Crammer, Dale Schuurmans. 2006. Robust Support Vector Machine Training via Convex Outlier Ablation. *AAAI*
Appendix. Implementation Details

We now describe how to train the robust model using \texttt{glmnet}, starting with the case where the $\theta$ parameters are not penalized. Equation (3) from Section 3.2 shows our reparametrized training objective. It can equivalently be written as

$$l(\theta') = \sum_{i=1}^{n} y_i g(\theta'^T x_i') + (1 - y_i) \left(1 - g(\theta'^T x_i')\right) + \lambda \sum_{j=0}^{m+n} p_j |\theta'^{(j)}|$$

where $p = (0, \ldots, 0, 1, \ldots, 1)$ is a vector of penalty factors, commonly used to allow differential shrinkage.

The following code snippet trains such a model for a full path of $\lambda$:

```r
robust.train.data = cbind(train.data, diag(N))
penalties = append(rep(0, times=P), rep(1, times=N))
robust.fit = glmnet(robust.train.data, as.factor(train.labels), family='binomial', penalty.factor=penalties, standardize=FALSE)
```

When $\theta$ is $L_1$-regularized as well, we instead adopt the following trick. Factoring out $\kappa$ in equation (2) gives us

$$l(\theta, \gamma) = \sum_{i=1}^{n} y_i g(\theta^T x_i + \gamma_i) + (1 - y_i) \left(1 - g(\theta^T x_i + \gamma_i)\right) + \kappa \left(\sum_{j=1}^{m} |\theta_j| + \frac{\lambda}{\kappa} \sum_{i=1}^{n} |\gamma_i|\right)$$

Letting $X' = [X|\kappa I_n]$ and $\theta' = (\theta_0, \ldots, \theta_m, \frac{\lambda}{\kappa} \gamma_1, \ldots, \frac{\lambda}{\kappa} \gamma_n)$, we can now train the model as usual. If desired, it is simple recover the correct values for $\gamma$. These commands train a regularized model for fixed $\kappa$ and $\lambda$:

```r
relative.penalty = lambda / kappa
robust.train.data.local = cbind(train.data, diag(N)/relative.penalty)
robust.fit = glmnet(robust.train.data.local, as.factor(train.labels), lambda=kappa, family='binomial', standardize=FALSE)
```

It may seem that one could also use the strategy of supplying a vector of penalty factors, but \texttt{glmnet} internally rescales these factors to sum to $n$. Moreover, the provided technique can be used with practically any software for $L_1$ regularization.