THESIS FOR THE DEGREE OF LICENTIATE OF TECHNOLOGY

Regularised Weights in Statistical Models

A General Strategy for Bias Reduction and Increased Stability

in Overparameterised Settings

Olof Zetterqvist

Division of Applied Mathematics and Statistics
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden 2021

Regularised Weights in Statistical Models: A General Strategy for Bias Reduction and Increased Stability in Overparameterised Settings
Olof Zetterqvist
Gothenburg 2021

Division of Applied Mathematics and Statistics
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31 772 1000

Typeset with LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Regularised Weights in Statistical Models
A General Strategy for Bias Reduction and Increased Stability in Overparameterised Settings

Olof Zetterqvist

Division of Applied Mathematics and Statistics
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

For flexible and overparameterised models like neural networks, overfitting can be a notorious problem that makes it hard to give accurate predictions in real-life usage. Overfitting is in particular likely in the presence of errors in the training data, such as misclassifications or outliers. Therefore it is essential to either carefully inspect the data or, more realistically, adapt the training algorithm to reduce variance and overfitting.

To reduce the risk of overfitting, a common approach is to manipulate the loss function. Either by adding a penalty on the model's flexibility, which reduces variance with a cost of an increased bias, or weight the loss contribution from different data points in order to reduce the influence of harmful data.

This thesis introduces a self-maintained method to reweigh different components (observations and/or parameter regularisation) in the loss function during training. With some care with the choice of model, these weights can be solved for, leading in the end to only a modification in the loss function. Due to this, the resulting method can easily be combined with other regularisation techniques.

Using the weighting technique on observations in a setting with mislabeled data produces more robust training than an unweighted model and detects mislabeled examples in data.

When used on the regularisation penalty, the weights reduces bias introduces by the regularisation term while keeping some crucial attributes from the original penalty.

Keywords: Deep Learning, Neural Networks, Noisy Labels, Lasso, Regularisation, Robust Statistics, Weighted loss.

List of publications

This thesis is based on the work represented by the following papers:

I. **Zetterqvist, O.**, Jörnsten, R., Jonasson, J. Robust Neural Network Classification via Double Regularization. *Manuscript*
II. **Zetterqvist, O.**, Jonasson, J. Entropy weighted regularisation, a general way to debias regularisation penalties. *Manuscript*

Acknowledgements

First and foremost, I would like to send a big thank you to my supervisor Johan for inspiration and excellent supervision. I would also like to thank my co-supervisor, Rebecka, for great inputs and ideas.

Thank you to my fellow PhD students of AI-batch 1 of the Wallenberg AI and Autonomous Systems and Software Program (WASP) for creating such a good community. Thank you to all friends at Mathematical Sciences: Anna, Anna, Anton, Barbara, Carl-Joar, Felix, Gabrijela, Helga, Henrik, Ivar, Jimmy, Johannes, Jonatan, Juan, Linnea, Mikael, Olle, Oskar, Sandra and Valentina. You all contribute to a warm and welcoming environment in the department. Thank you to Aila, Elisabeth and Marie for helping me when I'm confused about how things are done at the department.

I would like to thank my family. My parents for being of great support and giving me my curiosity for the mathematical field. My siblings for allowing me to break the "we should never move away from Skåne" pact. I would also like to thank Emilia, Sofia and Marta for giving me the courage to do so. Finally, I would like to send a big thanks to Johannes for countless hours of discussions about AI and for feeding my curiosity in the area.

Contents

1 Introduction

The first ideas behind artificial neural networks were introduced by McCulloch and Pitts (1943) and have since then become a popular tool in many data science applications. The reason for this is the power of these models and the increment of computational power making it able to use them. However, even if they show very impressive results, their flexibility leads to some demanding challenges in the form of overfitting. This decreases the performance of the model, which leads to uncertain predictions. This thesis introduces a general way of using weights in the loss function to attack these problems, making the training more robust.

This chapter is structured as follows; Section 1.1 describes the structure and concepts of neural networks and introduces the different loss functions used to train them. Next, section 1.2 describes some of the problems that can occur when using flexible models like neural networks, and section 1.3 describes how regularisation techniques can be used to reduce these problems. Finally, section 1.4 describes the concept of influential data points, how mislabeled data can be harmful in a classification setting and some methods used to overcome these problems.

1.1 Neural networks and their mathematical components

The structure of a neural network is highly inspired by the brain's architecture, which is built by billions of neurons. These neurons can be seen as small computational building blocks that can be used to build larger models. Even if one specific neuron cannot solve complex problems, the union of them can achieve remarkable things. Mathematically one neuron can be modelled as

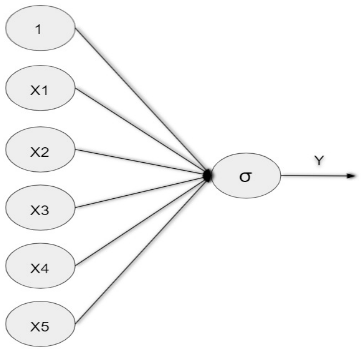

Figure 1.1: An illustration of the mathematical modulation of a neuron. The neuron takes in several inputs, process these to give one output. Setting $\sigma(x) = x$ gives a classical regression model and $\sigma(x) = \frac{1}{1+e^{-x}}$ gives a logistic regression model.

$$\hat{f}(X;\theta) = \sigma\left(\sum_{j=1}^{k} \theta_j X_j + \theta_0\right)$$

where $X = (X_1, \ldots, X_k)$ is the input vector and $\theta = (\theta_0, \ldots, \theta_k)$ parameters determining the behaviour of the neuron. The function σ corresponds to the neuron's activation function that transforms the input to the desired output domain. Some commonly used examples are $\sigma(x) = x$ which is simply linear regression and $\sigma(x) = \frac{1}{1+e^{-x}}$ which is used in logistic regression. One neuron is illustrated in figure 1.1 where an input vector of length 5 goes into the neuron and are used to calculate the output Y.

Even though ordinary regression and logistic regression models perform well in many situations, they have limitations since they can only model linear behaviours in data. However, combining many of these neurons creates a model with extensive flexibility that, with an appropriate choice of σ, can handle more complex data. In figure 1.2 is an example of how these neurons can be stacked together, creating a feed-forward neural network. Setting the σ's in the interior layers of the network, also called the hidden layers, to some non-linear function, the network can be very expressive. Neural networks can, as shown in Cybenko (1989), approximate any function to a given degree as long as it has enough neurons in its model. So, regardless of the complexity of the data, it is possible to create a neural network that can model its behaviour.

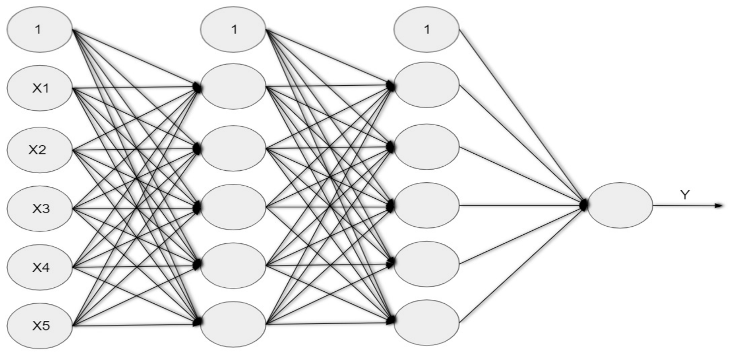

Figure 1.2: An illustration of a neural network. Each circle represents one neuron shown in figure 1.1. By stacking these together creating a neuron network, we get a more flexible model that can approximate more complex functions.

1.1.1 Loss functions

Given the data distributions (X, Y), the goal of training a neural network, $\hat{f}(X; \theta)$, is to tweak θ such that the network models the conditional expectation $E[Y|X]$. However, since the distributions of X and Y are unknown, the model is based on observed data points $(X_1, Y_1), \ldots, (X_n, Y_n)$ sampled from their joint distribution. Depending on the problem formulation, finding an estimate $\tilde{\theta}$ is typically done by a mean square error estimate, in the case of regression, or a maximum likelihood estimate, in classification. For example in regression, the goal is to find the parameters $\tilde{\theta}$ such that

$$E_{X,Y}\left[\frac{1}{2}(Y - \hat{f}(X;\theta))^2\right]$$

is minimised. Since the distributions of X and Y are not known, a common approach is to use the mean square error estimates $\tilde{\theta}$, which can be expressed as

$$\tilde{\theta} = \arg\min_{\theta} L(\hat{f}(X;\theta), Y) = \arg\min_{\theta} \frac{1}{2}||Y - \hat{f}(X;\theta)||_2^2 = \arg\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(Y_i - \hat{f}(X_i;\theta))^2.$$

In a classification setting, the problem of finding $\hat{\theta}$ is usually formulated in terms of maximum likelihood. In a binary setting, the goal is to find $\tilde{\theta}$ that maximises

$$E_{X,Y}\left[\prod_{i=1}^{n}\hat{f}(X_i;\theta)^{Y_i}(1-\hat{f}(X_i;\theta))^{1-Y_i}\right]$$

where $\hat{f}(X_i;\theta)$ models the probability of input X belonging to class one. This can easily be generalised to more than two classes.

Since the distribution of X and Y are unknown, we use the maximum likelihood estimate

$$\tilde{\theta}=\arg\max_{\beta}\prod_{i=1}^{n}\hat{f}(X_i;\theta)^{Y_i}(1-\hat{f}(X_i;\theta))^{1-Y_i}$$

which is equivalent to solving

$$\tilde{\theta}=\arg\min_{\theta}L(\hat{f}(X;\theta),Y)=\arg\min_{\theta}\sum_{i=1}^{n}-Y_i\log(\hat{f}(X_i;\theta))-(1-Y_i)\log(1-\hat{f}(X_i;\theta)).$$

Notice that in both the regression and classification problem, the minimisation problem is of the form

$$\tilde{\theta}=\arg\min_{\theta}L(\hat{f}(X;\theta),Y_i)=\arg\min_{\theta}\sum_{i=1}^{n}\ell(\hat{f}(X_i;\theta),Y_i).$$

where $\ell(\hat{f}(X_i;\theta),Y_i)$ represent the loss contribution from data point (X_i,Y_i). This form is of create importance for some algorithms as we will see later.

Usually, the minimisation problem is hard to solve analytically, meaning that we need to turn to numerical methods. The most common method is to use variants of gradient descent where the gradients are obtained through back-propagation for larger models Rumelhart et al. (1985).

1.2 Problems with flexible models

Neural networks have, in many setting, shown impressive results. The reason for this is the excellent approximation property of neural networks. However, these models can be hard to train if not appropriately handled. One obvious challenge is the problem of optimising the usually very high-dimensional and sometimes non-convex loss function, whose expression is very involved. This in itself is an extensive and active research field. It will, however, not be the focus here.

What will be the focus is the fact that the number of parameters is typically very large, which for proper estimation leads to a need for tremendous amounts of data, whereas in practice, the size of the data set is, strictly speaking, much too small compared with the complexity of the model. We will now discuss some problems that can follow from this.

1.2.1 Overfitting and the bias-variance tradeoff

Assume that the data $(X_1, Y_1), \ldots, (X_n, Y_n)$ follows the true model $E[Y_i|X_i] = f(X_i; \theta^*)$. Here θ^* is considered the true parameters of the model.

There are several ways to evaluate how good an estimation method is at producing good estimates $\tilde{\theta}$ based on these observations. One way is to measure the mean square error of the parameter difference

$$E_\theta\left[(\tilde{\theta} - \theta^*)^2\right] = E_\theta[(\tilde{\theta} - E_\theta[\tilde{\theta}])^2] + (E_\theta[\tilde{\theta}] - \theta^*)^2 = Var(\tilde{\theta}) + Bias(\tilde{\theta})^2.$$

We will now consider the consequences of having large, respectively, small values of both terms.

Having a high variance on the estimator $\tilde{\theta}$ indicates that minor differences in the training data give a significant difference in the estimated $\tilde{\theta}$. This comes from the fact that the model often adapts to the noise in data and memorise each data point separately. One example of this is the green prediction in figure 1.3. This phenomenon is called overfitting and can typically be seen with larger models where the flexibility is large, and the amount of data is insufficient. Having a low variance means that the estimate is not affected drastically by noise and is more robust, i.e. more stable in its predictions. On the other hand, a large bias term results in an estimator that will not, on average, predict the correct parameters, regardless of the amount of data.

Optimally, there exists an estimator that gives both low variance and a low bias. This is, however, often not the case in overparameterised settings. The reason for this is the flexibility of the model.

In order to reduce variance, several different techniques have been developed. These techniques are referred to as regularisation techniques and are designed to limit the model's flexibility, making it harder to overfit against data. Unfortunately, most of them also come with the side effect of increasing the bias. This means that using too much regularisation lowers the variance but introduce too much bias. On the other hand, using too little regularisation gives a small bias but a high variance. This phenomenon is called the bias-variance tradeoff and shows the importance of finding an optimal compromise in the amount of regularisation.

The terms in this tradeoff can differ between different types of regularisation techniques. Therefore it is essential to use a regularisation method that fulfils the required properties of the problem with an optimal tradeoff.

1.3 Regularisation penalties

One common regularisation technique is to introduce a constraint on the parameters θ. This is done by adding an additional term, called a regularisation penalty, to the loss function. The loss can then be written as

$$L(\hat{f}(X;\theta), Y) + g_\lambda(\theta;\phi)$$

where $g_\lambda(\theta;\phi)$ represents the regularisation penalty, λ the strength of the penalty and ϕ are parameters determining the shape of the penalty. Two common penalties are the ridge penalty $g = \lambda||\cdot||_2^2$ and the lasso penalty $g = \lambda||\cdot||_1$, Tibshirani (1996). By adding the regularisation term, we can penalise the model for using large values on θ and, by changing the value of λ, determine the flexibility of the model $f(x;\beta)$. This makes it possible to control how much variance that is removed. An example of the effect of regularisation is shown in figure 1.3 where an ordinary least square loss (OLS) with and without regularisation is used on a regression problem.

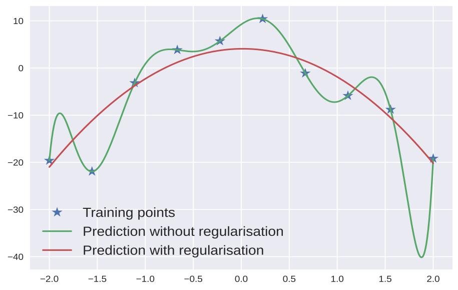

Figure 1.3: An illustration of overfitting with a model of the form $Y_i = \sum_{k=0}^{10} \theta_k X_i^k + \xi_i$ where ξ_i is normal distributed noise with parameters $\xi_i \sim N(0, 5^2)$. Based on the 10 marked data points we see the prediction of both an OSL estimate (green line) and a lasso estimate with $\lambda = 1$ (red line).

Even though the Lasso and Ridge penalties have turned out to be excellent at reducing variance and overfitting, they indeed come with a significantly increased bias.

Because of the challenges with the bias-variance tradeoff, several other and more refined regularisation penalties have been developed to reduce the variances with a minimal increase in bias. Some examples are non-convex penalties like SCAD, Fan and Li (2001), and some settings of Bridge,Frank and Friedman (1993), that use the non-convexity to reduce the bias of larger parameters. By having a penalty with a derivative close to zero for larger values, these parameters are not as affected by the penalty as smaller parameters. Another way to reduce bias is to weigh the regularisation terms differently for different parameters. One example of this is the adaptive lasso, Zou (2006) which uses an extra estimation step to calculate regularisation weights, ω_i for each parameter. To summarise the three examples, the regularisation penalties are

- **Bridge**: $g_\lambda(\theta; \gamma) = \lambda \sum_i |\theta_i|^\gamma; \gamma > 0.$
- **SCAD**: $g_\lambda(\theta; a) = \sum_i \left[\mathbf{1}(|\theta_i| < \lambda)\lambda|\theta_i| + \mathbf{1}(\lambda \leq |\theta_i| \leq a\lambda)\frac{|\theta_i|^2 - 2a\lambda|\theta_i| + \lambda^2}{2(a-1)} + \mathbf{1}(|\theta_i| > a\lambda)\frac{(a+1)\lambda^2}{2} \right]; a > 1.$
- **Adaptive lasso**: $g_\lambda(\theta; \gamma) = \lambda \sum_i \omega_i |\theta_i|$; where ω_i is weights based on a previous estimate $\hat{\theta}$. One example is $\omega_i = \frac{1}{|\hat{\theta}_i|^\gamma}; \gamma > 0$

It is crucial to keep in mind that also these models introduce additional bias to the estimates. However, the bias is more negligible in comparison with standard lasso or ridge.

Several methods to reduce variance other than adding a regularisation penalty are used. In particular, for neural network models where methods like early stopping and dropout, Srivastava et al. (2014), are very powerful.

1.3.1 Model selection

Even if regularisation methods are good at reducing the complexity in the models, they do not generally reduce the number of parameters used. However, there could be many benefits with a regularisation method that is able to reduce the number of parameters in the model. For example, it makes it easier to understand what features contribute to the result of the response and the predictions made by the model. Also, a regularisation method that can find the optimal set of features makes the cost of introducing new features less significant. Such methods are most attractive with linear models where the user can easily interpret the meaning of each parameter. Regularisation penalties that are good at reducing the model size are those that are non-differentiable at zero. Examples of this are lasso, Adaptive lasso, SCAD and Bridge when $\gamma \leq 1$.

1.4 Influence and mislabeled data

Overfitting can be a challenging problem in both regression and classification tasks and is even more likely to occur when data points are not following the "regular behaviour" of the data distribution, especially in the output space. Since every training point, by default, is treated as equally important, some data points affects the resulting model more than others. We say that these points have a strong influence on the model. A high influential point means that the resulting model would be very different if it is retrained on the same data set except for the specific point. Highly influential points are often considered bad for the model since they reduce robustness, which in turn leads to bad predictions. Examples of highly influential point could be an outlier in the input space or a data point with an incorrect label in a classification setting, which many studies have shown can be harmful to the resulting model, Koh and Liang (2017); Zhang et al. (2016); Arpit et al. (2017). An example can be seen in figure 1.4 where a neural network has been trained on a classification task with two classes, orange triangles and blue squares. With perfect data, the

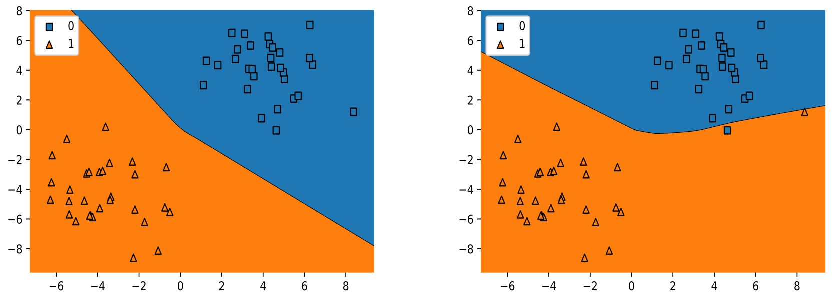

Figure 1.4: An illustration of what could happen when one training data point is miss classified. To the left is how a multilayered neural network classifies the two classes triangles and squares. The marks correspond to the training data. To the right is the result when using the same network and starting position but one training point has changed label.

model gives a very accurate classification of the two classes, but by changing the label of one data point, the prediction becomes very different, and the model cannot generalise.

Using neural networks requires a massive amount of data, making it hard to find mislabeled points by manual inspection. Since the impact can be so devastating, it is essential to use a method that reduces the influence of mislabelled data. To achieve this, several different techniques have been tried, Frénay and Verleysen (2014); Song et al. (2020). Like in a typical overfitting case, regularisation techniques like Lasso and Ridge can reduce the complexity of the model and, therefore, also reduce the influence. Also, methods more commonly used for neural networks like early stopping have been efficient in this regard, Li et al. (2020). Even if these methods reduce overfitting, it could mean that the model cannot learn the true behaviour in data due to its increased bias and reduced expressiveness.

Another approach is not to limit the model's flexibility but instead change the loss function or model architecture to reduce the influence of each point. Several such approaches have been suggested. One example is to locate misclassified data points by estimating the influence function of each point and then remove those with a high influence, Koh and Liang (2017). Some extend the model also to model the label noise. This can, for example, be done with a confusion matrix Goldberger and Ben-Reuven (2016); Hendrycks et al. (2018). The model should adapt to the noise and, therefore, learn which points are incorrect. Some use different kind's of prepossessing techniques to make the training more robust against mislabelled data. An example of this is mixup Zhang et al. (2017), which uses a linear combination of data points as an input to the model.

A further approach is to put a weight on each data point in the loss function Ren et al. (2018). This means to use a loss function like

$$L(X, Y; \beta) = \sum_{i}^{n} \omega_i \ell(\hat{f}(X_i; \theta), Y_i)$$

where ω_i is the individual weight corresponding to each point. The weights could, for example, be estimated by comparing gradients between training data and a separate dataset where the user knows that the labels are correct, Mengye Ren (2018). The goal is to give low weights to data that are harmful to the model and reduce their contribution to the total loss. Even though there are ways to detect mislabelled data and reduce their influence on the model, there is still much to learn about dealing with them optimally.

2 Summary of papers

In this theses, a general strategyto weight components in the loss function is introduced. This strategy can be used to give different weights to regularisation terms for different parameters as well as to give different weights to different data points. These weights are incorporated into the model by expanding the loss function to include terms to control the weights. By choosing the form of these terms wisely, one can solve for the weights in terms of the parameters and thus avoiding the apparent inclusion of a large set of new parameters. The particular form of weight controlling term that we have chosen to work with is $\gamma(\omega\log(\omega) - \omega + 1)$, where γ is a hyperparameter. This choice is analytically amenable and works well in experiments.

2.1 Article 1 Robust Neural Network Classification via Double Regularization

This article introduces our strategy when used to weigh different data points in a classification setting. It gives us the loss function

$$(\tilde{\theta}, \tilde{\omega}) = \arg\min_{\theta,\omega} L(X, Y; \beta) = \arg\min_{\theta,\omega} \sum_{i}^{n} \omega_i \ell(\hat{f}(X_i; \theta), Y_i) + \gamma(\omega_i \log(\omega_i) - \omega_i + 1)$$

with some appropriate constraints on ω within each class, such that the mean weight within each class c should be a number ρ_c. ρ_c is chosen to reflect the proportion of training data that belongs to c. We show that the minimization problem can be rewritten as

$$\tilde{\theta} = \arg\min_{\theta} -\gamma \sum_{c \in C} \rho_c |c| \log \sum_{i \in c} e^{-\frac{\ell(\hat{f}(X_i;\theta),Y_i)}{\gamma}}$$

where C corresponds to the different classes in the training set. In other words, the use of observation weights and regularisation terms for them translates to a certain modification of the loss function. We prove some analytical results that our methodology can indeed make the training more robust against mislabelled data. Numerical experiments are carried out on neural networks in a binary classification setting. These show that this loss function makes the overfitting against mislabelled data less significant and accurately finds which data points are classified incorrectly.

My contribution to this paper is

- contributing to the development of the methodology
- doing all implementations and simulations
- writing large parts of the article except for section four.

2.2 Article 2 Entropy weighted regularisation, a general way to debias regularisation penalties.

In this article, we investigate our methodology when applied to regularisation in a linear regression setting. This means to consider a regularised mean square loss function, e.g. lasso or ridge, and modify it by allowing different weights on the penalty for different parameters. The weights themselves are then controlled as before. Note the difference from paper 1, where the weights are on the observations whereas they are now on the model parameters. This reflects that the goal in paper 1 is to handle contaminated data, whereas in this paper, the goal is to handle overparametrisation. It gives us the loss function

$$(\tilde{\theta}, \tilde{\mathbf{u}}) = \arg\min_{\theta,u} = \frac{1}{2}||Y - X\theta||_2^2 + \sum_{i=1}^{p} u_i g_\lambda(\theta_i; \phi) + \gamma(u_i \log(u_i) - u_i + 1).$$

However, we show that this minimization problem can be rewritten as

$$\tilde{\theta} = \arg\min_{\theta} \frac{1}{2}||Y - X\theta||_2^2 + \gamma \sum_{i=1}^{p}(1 - e^{-\frac{g_\lambda(\theta_i;\phi)}{\gamma}}).$$

Hence, our procedure for individual parameter penalty weights translates to a modification to a more robust form of the original parameter penalty. We investigate how this regularisation behaves when $g_\lambda(\theta_i;\phi) = \lambda|\theta_j|$, called entropy weighted lasso (EWL), and $g_\lambda(\theta_i;\phi) = \frac{\lambda}{2}\theta_j^2$, called entropy weighted ridge (EWR). Since these regularisation's are extensions of Lasso and Ridge, they inherit some properties from these. We derive requirements on λ and γ for when the optimization problems are convex, and show that both are consistent. The EWL regression model is also proven to be sign consistent, giving it nice properties for model selection. We derive optimization algorithms for EWL and EWR and compare the resulting models with other methods like adaptive lasso, SCAD, lasso and ridge on simulated data.

My contribution to this paper is

- writing large parts of the theory. This includes both properties of EWL and EWR but also in finding efficient training algorithms
- implementing the algorithms and writing code for simulations
- writing large parts of the texts.

Bibliography

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

Frank, L. E. and Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135.

Frénay, B. and Verleysen, M. (2014). Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25:846–869.

Goldberger, J. and Ben-Reuven, E. (2016). Training deep neural-networks using a noise adaptation layer.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. (2018). Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in neural information processing systems*, pages 10456–10465.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia. PMLR.

Li, M., Soltanolkotabi, M., and Oymak, S. (2020). Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. volume 108 of *Proceedings of Machine Learning Research*, pages 4313–4324, Online. PMLR.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mengye Ren, Wenyuan Zeng, B. Y. R. U. (2018). Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018). Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Song, H., Kim, M., Park, D., and Lee, J.-G. (2020). Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Tibshirani, R. (1996). Regression selection and shrinkage via the lasso. *Journal of the Royal Statistical Society Series B*, 58(1):267–288.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

Robust Neural Network Classification via Double Regularization

Olof Zetterqvist OLOFZE@CHALMERS.SE
Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
Research sponsored by Wallenberg AI, Autonomous Systems and Software Program (WASP)

Rebecka Jörnsten JORNSTEN@CHALMERS.SE
Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
Research supported by SSF (Swedish Foundation for Strategic Research) and the Swedish Research Council.

Johan Jonasson JONASSON@CHALMERS.SE
Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
Research supported by Wallenberg AI, Autonomous Systems and Software Program (WASP) and Centiro Solutions AB.

Editor:

Abstract

The presence of mislabeled observations in data is a notoriously challenging problem in statistics and machine learning, associated with poor generalization properties for both traditional classifiers and, perhaps even more so, flexible classifiers like neural networks. Here we propose a novel double regularization of the neural network training loss that combines a penalty on the complexity of the classification model and an optimal reweighting of training observations. The combined penalties result in improved generalization properties and strong robustness against overfitting in different settings of mislabeled training data and also against variation in initial parameter values when training. We provide a theoretical justification for our proposed method derived for a simple case of logistic regression. We demonstrate the double regularization model, here denoted by DRFit, for neural net classification of (i) MNIST and (ii) CIFAR10, in both cases with simulated mislabeling. We also illustrate how DRfit, as a useful side-benefit, can be used to identify mislabeled data points. This provides strong support for the DRFit as a practical of-the-shelf classifier, since, without any sacrifice in performance, we get a classifier that reduces overfitting against mislabeling and gives an accurate measure of the trustworthiness of the labels.

Keywords: Deep Learning, Robust Statistics, Regularization, Mislabeling, Mislabeled data, Contaminated labels, DRFit, weighted loss.

1. Introduction

In supervised, semi-supervised or active learning with machine learning algorithms in general, and deep neural nets in particular, the default models assume that the given annotations of training data are correct. However, since well-annotated data sets can be expensive and time consuming to collect, a fair amount of recent research has focused on using larger but noisy sets of training data. Such data sets are much cheaper to collect, e.g. via crowd sourcing. The working assumption is that collecting larger amounts of data that are labeled with a decent accuracy can compensate for the noise (inaccurate labels).

Clearly, it is desirable that neural networks estimation is robust with respect to changing a small fraction of the labels of the training data. Results in the literature are conflicting, with findings depending on the noise distribution, the type of neural net, the amount of correctly labeled data and level of contamination. While some results indicate that classification performance on correct labels can be noise stable (e.g. (Rolnick et al., 2017)), there are many studies that have shown that neural networks are highly sensitive to label contamination (Koh and Liang, 2017; Zhang et al., 2016; Arpit et al., 2017).

There is a variety of methods for dealing with noisy labels (Frénay and Verleysen, 2014; Song et al., 2020). One approach is to try to identify training examples that are likely to be mislabeled and correct them manually (Koh and Liang, 2017; Pleiss et al., 2020). This is computationally costly and suffers from a chicken-and-egg problem; examples that stand out as likely to be incorrectly labeled do so on the basis of a model that is in itself trained on noisy data.

Some methods focus on using data augmentation in order to reduce the impact of noisy labels (Zhang et al., 2017), while others estimates a confusion matrix that compensates for the noise in the labels (Goldberger and Ben-Reuven, 2016; Hendrycks et al., 2018).

Another approach is to incorporate the noise directly into the loss function of the training model (Reed et al., 2015; Natarajan et al., 2013; Sukhbaatar et al., 2015; Liu and Tao, 2016; Tanaka et al., 2018; Arazo Sanchez et al., 2019), either via a surrogate loss function that augments the classification loss with noise rate parameters or with a data reconstruction term. Overfitting is still common for these methods which is usually addressed via early stopping. The intuition behind early stopping is that the optimization algorithm at its early stages finds the large scale classification boundaries and only after that starts to fine-tune to individual training examples and hence overfit. This intuition is supported by recent theoretical results (e.g. (Li et al., 2020)). Early stopping results in the final network parameters to be a mix of starting values and optimal values. Several questions arise; how should the starting values be chosen?, when should we stop?. Also, from a statistical point of view, it is very unsatisfactory to define a model (loss) that we in some sense do not ultimately use.

It is common to attempt to overcome overfitting through some form of regularization on the network parameters, e.g. lasso or ridge. However, we will here show that this does not fully alleviate the overfitting in the presence of training data mislabeling.

In this paper we suggest a double regularization (DR) technique where, in addition to using lasso or ridge, each term in the loss function is multiplied with an observation weight. The observation weights will be optimally penalized for deviating from equal. The weights can be seen as a new set of model parameters, different from the network parameters but may be trained in parallel with these. However, with the proper choice of regularization penalty for the observation weights, when optimizing this new loss function, it is easy to solve for the observation weights in terms of the original parameters and thereby obtain an explicit new loss function, where in fact no new parameters have been added and hence no extra complexity has been imposed. The benefits of the new model include; (a) a mathematical formulation that explains the weighting methodology as a regularization mechanism and (b) circumventing the need for early stopping since the loss function is explicitly optimized.

We claim that intuitively this model does not overfit, since early on in the training, the weights will adjust to leave only the large structure in feature space and hence works in itself as early stopping but without the need to actually stop. Experimental results very strongly suggest this intuition is correct. In addition, in section 4 we provide a theoretical justification, showing that for a simple version of logistic regression, DR performs strictly better than a standard regularized neural network.

2. Methods

Consider a binary classification in the space $X \times Y$ where X is the input space and Y the label space and let $z_i = (x_i, y_i) \in X \times Y$ be a realization. For inference we use a model $y = f(x;\theta)$, where $\theta \in \Theta$ are the model parameters and $x \in X$. In order to find the optimal θ, a loss function $C(\theta; z) = \sum_i L(\theta; z_i)$ is minimized.

Now consider uniform label noise such that each training point is contaminated with a probability that may depend on class but is otherwise independent of the features. In such a setting, a model with even a few parameters can easily overfit in the sense that it eventually learns a false and overly complex structure, trying to separate between instances in the training data that have different labels but are in truth of the same class. As a result, the model generalizes poorly to new data. To alleviate this problem, we propose a regularization technique that gives each training point an observation weight that reflects how much we "believe" in the annotation of that point, combined with a penalty for observation weights for deviating too much from one (since without such a penalty, the model would simply put all weight on one example from each class). The idea is that our model will learn observation weights that are close to 0 for mislabelled data while leaving other weights well away from 0. When the model is also over-specified, we combine this observation weighting with some standard regularization on the model parameters, such as a ridge or lasso regularization.

The minimization object of our doubly regularized model, DRFit, can in its general form thus be written as

$$(\theta^*, \omega^*) = \text{argmin}_{\theta,\omega} \sum_i \omega_i L(z_i, \theta) + \alpha g(\omega) + \lambda \tilde{g}(\theta), \tag{1}$$

where α and λ are hyper-parameters controlling the amount of regularization, and g and $\tilde{g}$ are regularization functions which are typically convex functions and where the last term may be dropped if the model is not over-parametrized. For the rest of this article we will consider $\tilde{g} = \frac{1}{2}||\cdot||_2^2$, i.e. a ridge regularization on the original model parameters.

There are some recent studies that utilize observation weights in the context of regularized or robust regression (Luo, 2016; Gao and Fang, 2016; Gao and Feng, 2017) as well as some methods on neural networks (Ren et al., 2018) where they calculate the observation weights based on a comparison between the gradients of training and validation data. With the extra forward and backward passes this algorithm requires the model takes approximately three times the training time to train.

The two hyperparameters in (1) control the trade-off between observation weighting and regularization of the network parameters. A larger λ reduces the complexity of the neural network and a larger α restricts the flexibility of observation reweighting. While it may appear as if we have introduced a large number of extra parameters with observation weights, as mentioned above, with a certain natural choice of regularization function g, we will be able to solve for the observation weights in terms of the model parameters, hereby getting a new modified minimization objective explicitly expressed in terms of the loss function $L(\theta; z)$.

3. Choice of weight regularisation function

In this paper we will use an entropy regularization on the observation weights: $g(\omega_i) = \omega_i \log(\omega_i) - \omega_i$ combined with the constraint that $\sum_{i:y_i \in C_k} \omega_i = \rho_k |C_k|$ for each class k. Here C_k is the set of training examples that have been labeled k and ρ_k is a scaling factor that can be tuned to fit the label noise. The main purpose of the ρ_k's is to make up for the difference in class proportions as they appear in the noisy training data and the true proportions. To estimate the true proportions, a natural assumption is that we have access to a clean set of validation data. If not, the natural choice is to set $\rho_k = 1$ for all k.

As promised, we can now solve for the observation weights in terms of the original model parameters as follows. With the given g, the DRFit optimization problem reads:

$$(\theta^*, \omega^*) = \operatorname{argmin}_{\theta,\omega} \sum_i [\omega_i L(z_i, \theta) + \alpha(\omega_i \log(\omega_i) - \omega_i)] + \lambda \frac{1}{2} ||\theta||_2^2. \tag{2}$$

Solving for ω in terms of θ (2) gives the following:

Proposition 1 *Solving the minimization problem*

$$(\theta^*, \omega^*) = \operatorname{argmin}_{\theta,\omega} \sum_i [\omega_i L(z_i, \theta) + \alpha(\omega_i \ln(\omega_i) - \omega_i)] + \lambda \frac{1}{2} ||\theta||_2^2$$

with the constraint $\sum_{i \in C_k} \omega_i = \rho_k |C_k|$ *is equivalent to solving*

$$\theta^* = \operatorname{argmin}_\theta - \alpha \sum_k \rho_k |C_k| \log \left(\sum_{i \in C_k} e^{-\frac{L(z_i,\theta)}{\alpha}} \right) + \frac{\lambda}{2} ||\theta||_2^2 \tag{3}$$

Proof For convenience, we let $\ell_i = L(z_i, \theta)$, $n_k = \rho_k |C_k|$ and

$$F(\theta, \omega) = \sum_i \omega_i \ell_i + \alpha(\omega_i \log(\omega_i) - \omega_i) + \lambda \frac{1}{2} ||\theta||_2^2 + r_k \left(\sum_{i \in C_k} \omega_i - n_k \right),$$

where r_k is our Lagrange factor associated with class k. Our goal is now to find stationary points of $F(\theta, \omega)$. Pick an ω_i, let k be the class to which y_i belongs and differentiate with respect to ω_i and solve $\frac{\partial}{\partial \omega_i} F(\theta, \omega) = 0$:

$$\frac{\partial F(\theta, \omega)}{\partial \omega_i} = 0 \Rightarrow \ell_i + \alpha \log(\omega_i) + r_k = 0 \Rightarrow \omega_i = -e^{-(\ell_i + r_k)/\alpha}$$

Inserting into the constraints $\sum_{i\in C_k}\omega_i = n_k$ and solving for r_k, we get

$$\omega_i = n_k \frac{e^{-\ell_i/\alpha}}{\sum_{j\in C_k} e^{-\ell_j/\alpha}}.$$

From the arguments it is clear that this stationary point is unique and must hence correspond to the minimum of (2). Plugging into (2) minus the ridge penalty term gives

$$\begin{aligned}
&\sum_i \omega_i\ell_i + \alpha(\omega_i\log(\omega_i) - \omega_i)\\
&= \sum_k \sum_{i\in C_k} \omega_i\ell_i + \alpha(\omega_i\log(\omega_i) - \omega_i)\\
&= \sum_k \sum_{i\in C_k} n_k \frac{e^{-\ell_i/\alpha}}{\sum_{j\in C_k} e^{-\ell_j/\alpha}}\ell_i + \alpha\left(n_k\frac{e^{-\ell_j/\alpha}}{\sum_{j\in C_k} e^{-\ell_j/\alpha}}\log\left(n_k\frac{e^{-\ell_i/\alpha}}{\sum_{j\in C_k} e^{-\ell_j/\alpha}}\right) - n_k\frac{e^{-\ell_i/\alpha}}{\sum_{j\in C_k} e^{-\ell_j/\alpha}}\right)\\
&= -\alpha\sum_k n_k\log\left(\sum_{j\in C_k} e^{-\ell_j/\alpha}\right) + n_k\log(n_k) - n_k.
\end{aligned}$$

Since n_k is a constant our preposition follows. ■

In summary, with this choice of g, we simply get a new minimization objective with no extra parameters, besides the hyperparameter α which still controls how much the observation weights are allowed to deviate from equal.

4. Theoretical results

In what follows, we will make a theoretical justification of the weight penalty in a simple setting, namely logistic regression for one-dimensional covariates without intercept term. The model will then not be overparametrized, so we discard the ridge penalty term from (3). The model may thus be written as

$$\mathbb{P}(y_i = 1) = \frac{e^{sx_i}}{e^{sx_i} + e^{-sx_i}}$$

for an unknown slope s. For standard logistic regression, s is estimated as

$$\hat{s}_n = \operatorname{argmax}\mathcal{L}(s), \tag{4}$$

where

$$\begin{aligned}
n\mathcal{L}_n(s) &= n\mathcal{L}_n(s;\mathbf{x},\mathbf{y})\\
&= s\left(\sum_{i:y_i=1} x_i - \sum_{i:y_i=0} x_i\right) - \sum_{i=1}^{n}\log(e^{sx_i} + e^{-sx_i}).
\end{aligned}$$

We claim that under various natural distributions of the covariates, one can always find an α such that (3) will produce an estimate of s that exactly coincides with $\hat{s}$, which is arguably best possible. In almost all cases, the proof relies on numeric computation of expectations. However in a special case of Gaussian distribution of covariates in each class, an analytic proof can be given.

We start by showing that under very mild assumptions, contamination of the labels will always cause standard logistic regression to underestimate s. This is formalized by a proposition below. Since the introduction of a fair amount of notation will be needed to state the proposition, we introduce it along with proving the proposition and state the result as a summary.

For data generation, assume that covariates whose true label is k are distributed according to density f_k, $k = 0, 1$. Write X_k for a random variable distributed according to f_k and assume that $\mathbb{E}[X_1] > 0$, $\mathbb{E}[X_0] < 0$, $\mathbb{P}(X_1 < 0) > 0$ and $\mathbb{P}(X_0 > 0) > 0$. These assumptions make sure that the two distributions are overlapping but not equal. They also entail that $0 \leq \hat{s} < \infty$. As $n \to \infty$, $\mathcal{L}_n(s)$ converges to

$$\mathcal{L}(s) := s(p_1\mathbb{E}[X_1] - p_0\mathbb{E}[X_0]) - \mathbb{E}[\log(e^{sX} + e^{-sX})], \tag{5}$$

where $p_k = n_k/n$, where n_k is the number of observations with $y_1 = k$, and X is chosen according to the overall distribution of the covariates, and $\hat{s} := \lim_{n\to\infty} \hat{s}_n$ is the argmax of (5).

Now assume that true labels 1 are misclassified as 0 and vice versa independently but potentially with a probability depending on x_i. This gives us contaminated labels y_i^* and we assume that $\mathbb{E}[X_1^*] \leq \mathbb{E}[X_1]$ and $\mathbb{E}[X_0^*] \geq \mathbb{E}[X_0]$. Assume also that $p_1^*\mathbb{E}[X_1^*] - p_0^*\mathbb{E}[X_0^*] < p_1\mathbb{E}[X_1] - p_0\mathbb{E}[X_0]$, where X_k^* is a random variable chosen according to the distribution, f_k^*, of covariates for which $y_i^* = k$ and p_k^* is the probability that a random observation has $y_i^* = k$. It can be shown in a straightforward manner that this assumption is satisfied e.g. whenever the probability of mislabeling is conditionally independent of x_i given y_i, provided that the first two assumptions hold. Let

$$\hat{s}^* = \operatorname{argmax} \mathcal{L}^*(s)$$

where

$$\mathcal{L}^*(s) = s(p_1^*\mathbb{E}[X_1^*] - p_0^*\mathbb{E}[X_0^*]) - \mathbb{E}[\log(e^{sX^*} + e^{-sX^*})].$$

Now $\hat{s}$ is the solution of

$$\frac{\partial\mathcal{L}(s)}{\partial s} = p_1\mathbb{E}[X_1] - p_0\mathbb{E}[X_0] - \mathbb{E}[X \tanh sX] = 0$$

and, since X is independent of contamination, $\hat{s}^*$ is the solution of

$$p_1^*\mathbb{E}[X_1^*] - p_0^*\mathbb{E}[X_0^*] - \mathbb{E}[X \tanh(sX)] = 0.$$

Since $p_1^*\mathbb{E}[X_1^*] - p_0^*\mathbb{E}[X_0^*] < p_1\mathbb{E}[X_1] - p_0\mathbb{E}[X_0]$ and $(\partial/\partial s)X\tanh(sX) = 4X^2/(e^{sX} + e^{-sX})^2 > 0$, so that (by the Dominated Convergence Theorem) $\mathbb{E}[X \tanh sX]$ is increasing in s, we get $\hat{s} > \hat{s}^*$. We have proved:

Proposition 2 *Under the above assumptions on the covariate and mislabel distributions, standard logistic regression underestimates s, i.e.*

$$\hat{s}^* < \hat{s}.$$

Next we claim, under the conditions of Proposition 2, optimizing (3) without the ridge penalty term with α sufficiently small will, in the limit as $n \to \infty$, estimate s to ∞. We will not prove the full claim as the proof is tedious and we judge that the given form is sufficient for making the point that DRFit behaves in a good way in the present setting.

Proposition 3 *Consider the optimization problem*

$$\hat{s}_w(\alpha) = argmax \mathcal{O}(s)$$

where

$$\mathcal{O}(s) = p_1^* \log \mathbb{E}\left[\left(\frac{e^{sX_1^*}}{e^{sX_1^*} + e^{-sX_1^*}}\right)^{1/\alpha}\right] + p_0^* \log \mathbb{E}\left[\left(\frac{e^{-sX_0^*}}{e^{sX_0^*} + e^{-sX_0^*}}\right)^{1/\alpha}\right].$$

Assume that $f_1^*(x) \geq f_1^*(-x)$ and $f_0^*(x) \leq f_0^*(-x)$ whenever $x \geq 0$, then $\hat{s}_w(\alpha) = \infty$ whenever $\alpha \leq 1$.

Proof Finding the maximum of $\mathcal{O}(s)$ is equivalent to finding the maximum of $\mathcal{E}(s) = \exp(\mathcal{O}(s))$, i.e. the maximum of

$$\mathcal{E}(s) := \mathbb{E}\left[\left(\frac{e^{sX_1^*}}{e^{sX_1^*} + e^{-sX_1^*}}\right)^{1/\alpha}\right]^{p_1^*} \cdot \mathbb{E}\left[\left(\frac{e^{-sX_0^*}}{e^{sX_0^*} + e^{-sX_0^*}}\right)^{1/\alpha}\right]^{p_0^*}.$$

Now we claim that both factors of $\mathcal{E}(s)$ are increasing in s when $\alpha \leq 1$. To see that this is so, observe that for $b \geq 0$,

$$\begin{aligned} &\int_0^\infty \left(\frac{e^{sx}}{e^{sx} + e^{-sx}}\right)^b f_1^*(x) + \left(\frac{e^{-sx}}{e^{sx} + e^{-sx}}\right)^b f_1^*(-x)dx \\ &= \int_0^\infty \left(\left(\frac{e^{sx}}{e^{sx} + e^{-sx}}\right)^b + \left(\frac{e^{-sx}}{e^{sx} + e^{-sx}}\right)^b\right) f_1^*(-x)dx \\ &+ \int_0^\infty \left(\frac{e^{sx}}{e^{sx} + e^{-sx}}\right)^b (f_1^*(x) - f_1^*(-x))dx. \end{aligned}$$

Since $f_1^*(x) \geq f_1^*(-x)$ by assumption, the second term is increasing in s. If also $b \geq 1$, the first term is nondecreasing in s since the integrated function is then nondecreasing. Taking $b = 1/\alpha$ now proves that the first factor of $\mathcal{E}(s)$ is increasing and the second factor follows analogously.

■

A consequence of Proposition 3 is that if it could be shown that $s_w^*(\alpha)$ is continuous in α for $\alpha \geq 1$ with $\lim_{\alpha \downarrow 1} = \infty$, would be that for some $\alpha = \alpha_0$ one has $s_w^*(\alpha) = \hat{s}$. However, $\mathcal{E}(s)$ is very difficult to analyze analytically, so it is hard to come up with explicit general conditions that ensure this. Numerically, we have verified that $\hat{s}_w$ is indeed continuous for numerous natural cases of distributions on X_1 and X_0. On the other hand, we have also found examples where continuity does in fact not hold. One such example is $\mathbb{P}(X_0 = -1) = \mathbb{P}(X_0 = 1) = 1/2$, $\mathbb{P}(X_1 = -1) = \mathbb{P}(X_1 = 1) = 1/10$, $\mathbb{P}(X_1 = 10) = 4/5$ and 20% mislabels for both classes. In this case we find that in the range $\alpha \in [1.54, 1.69]$, two local maxima in $\mathcal{E}(s)$ appear and the the global maximum shifts from one to the other at approximately $\alpha = 1.62$.

Making the strong assumptions that $X_0 =_d -X_1$, $p_1 = p_0 = 1/2$ and $p_1^* = p_0^* = 1/2$, maximizing $\mathcal{E}(s)$ boils down to maximizing the target function.

$$\mathbb{E}\left[\left(\frac{e^{sX_1^*}}{e^{sX_1^*} + e^{-sX_1^*}}\right)^{1/\alpha}\right] = \mathbb{E}\left[\frac{1}{(1 + e^{-2sX_1^*})^{1/\alpha}}\right].$$

over s. This is easier, but still very difficult, to analyze. However if we also add that X_1 is Gaussian with positive mean, the problem indeed turns out to be analyzable:

Proposition 4 *Assume that $X_0 =_d -X_1$, $p_0 = p_1$, $X_1 \sim N(\mu, \sigma^2)$ with $\mu > 0$ and that mislabels occur independently and with equal probability less than $1/2$ for the two classes. Then $s_w(\alpha)$ is continuous decreasing and $\lim_{\alpha \downarrow 1} s_w(\alpha) = \infty$.*

Proof Taking the derivative of the target function, the proposition follows once we have proved that the solution in s of

$$\mathcal{F}(s; b) := \mathbb{E}[G(s, X_1^*, b)] = 0,$$

where $b = 1/\alpha$ and

$$G(s, x, b) = \frac{xe^{-2sx}}{(1 + e^{-2sx})^{b+1}}$$

is unique and continuously increasing in b and converges to ∞ as $b \uparrow 1$. Assume with loss of generality that $\mu = 1$ and $\sigma^2 = 1/2$. Then $\mathcal{F}(s; b)$ equals $1/\sqrt{\pi}$ times

$$\gamma \int_{-\infty}^{\infty} G(s, x, b)e^{-(x-1)^2}dx + (1 - \gamma) \int_{-\infty}^{\infty} G(s, x, b)e^{-(x+1)^2}dx,$$

which in turn explicitly equals

$$\gamma \int_{-\infty}^{\infty} \frac{xe^{-2sx}e^{-(x-1)^2}}{(1 + e^{-2sx})^{b+1}}dx + (1 - \gamma) \int_{-\infty}^{\infty} \frac{xe^{2sx}e^{-(x+1)^2}}{(1 + e^{2sx})^{b+1}}dx,$$

where γ is the probability of having a training point correctly labeled. We claim that the integrated function in the first term (and hence the second) is antisymmetric in x about the origin for $s = s_0 := 2/(1 - b)$ and hence $\mathcal{F}(s)$ is 0 for s_0. A straightfoward computation of the partial derivative of $G(s, x, b)/G(s, -x, b)$ with respect to s, one finds this to be positive for $x > 0$ and hence $G(s, x, b)e^{-(x-1)^2} < -G(s, -x, b)e^{-(x+1)^2}$ for $s > 2/(1 - b)$ and vice versa for $s < 2/(1 - b)$. Hence, since $\gamma > 1/2$, $\mathcal{F}(s; b) < 0$ for $s > 2/(1 - b)$ and vice versa. Hence the solution $s = s_0$ is unique. It follows that once the claimed antisymmetry is proved, we are done.

Now, proving the claim means to show that for each $x \geq 0$,

$$\frac{xe^{-2s_0 x}e^{-(x-1)^2}}{(1+e^{-2s_0 x})^{b+1}} = \frac{xe^{2s_0 x}e^{-(x+1)^2}}{(1+e^{2s_0 x})^{b+1}}.$$

After shortening fractions and taking logarithms of both sides, this is equivalent to

$$\frac{4x}{1-b} = \log\left(\frac{1+e^{4x/(1-b)}}{1+e^{-4x/(1-b)}}\right).$$

Now the general observation that

$$\log\left(\frac{1+e^{y}}{1+e^{-y}}\right) = y$$

finishes the proof.

■

As already stated, we have numerically verified that under the same conditions, one has that $s_w(\alpha)$ i continuous and decreasing in α and $\lim_{\alpha\downarrow 1} = \infty$, for some natural assumptions on the distribution of X_1. These include

- $X_1 \sim$ logistic with positive mean.
- $X_1 \sim$ Cauchy centered at a positive number.
- $X_1 \sim$ uniform with positive mean.

5. Numerical solver for DRFit

In the general DRFit setting (i.e. with a penalty g different from the one given in Section 3, one cannot solve for ω in terms of θ in (1) and then has to settle for numerics. In that case, we will alternate between applying numerical fitting of θ given ω and vice versa. This is to say that we split the optimization into two sub-problems:

$$\text{Problem} \quad 1: \quad \theta^* = \operatorname*{argmin}_{\theta} \sum_i \omega_i L(z_i, \theta) + \alpha g(\omega) + \frac{\lambda}{2}||\theta||_2^2,$$

$$\text{Problem} \quad 2: \quad \omega^* = \operatorname*{argmin}_{\omega, \omega \geq 0} \sum_i \omega_i L(z_i, \theta) + \alpha g(\omega) + \frac{\lambda}{2}||\theta||_2^2.$$

We alternate between these two tasks during training. Problem 1 is a standard minimization problem for the network model parameters θ. The optimization of ω is done with gradient decent which gives us the updating procedure

$$\omega_i \leftarrow w_i - \beta[L(z_i, \theta) + \alpha g'(\omega)],$$

where β is the learning rate for the observation weights. In order to speed up convergence, we use a burn-in period where we do not update the weights ω. In this way, the model learns the more global structure of the data before we update the weights. In our experiments we found that a short burn-in of just a few epochs suffices.

In order to prevent the weights from becoming negative we use weight clipping to set negative weights to zero. In each iteration the weights are also normalized to sum to $\rho_c n$. This is to preserve the same learning rate throughout training. The training algorithm is summarized in algorithm 1.

Algorithm 1 Training algorithm with observation weights.

```
procedure TRAIN(Training set $D$, $\alpha$,burn_in,update_frequency, $\rho_c$, $\beta, N$)
    $\omega \leftarrow \mathbf{1}$
    for epoch $= 0 \ldots N-1$ do
        for Batch $S \subseteq D$ do
            Update $\theta$ with batch $S$ for current $\omega$.
            if (epoch > burn_in)
              and (mod(epoch,update_frequency) = 0) then
                $\omega_S \leftarrow \omega_S - \beta[L(S;\theta) + \alpha \nabla g(\omega_S)]$
                for i in S do
                    if ($\omega_i < 0$) then
                        $\omega_i \leftarrow 0$
                Normalize $\omega_S$ to mean $\rho_c$ for each class present in $S$
    return
```

6. Experiments

We will experimentally evaluate the performance of the DRFit method by training on different neural networks for classification tasks of varying difficulty. We will train DRFit both analytically (in the sense of solving for ω in terms of θ) and according to Algorithm 1. The reason for doing the latter is to see if there appears to be non-convergence phenomena that cannot be seen for the analytic solver. In each simulation we have used $\rho_c = p(c)/(1-q(c))$ where $p(c)$ is the proportion among those instances that have been classified as class c that are indeed of class c and $q(c)$ is the proportion of instances for which the correct label is c, but have been classified as something else. Good estimates of ρ_c can be obtained from a set well annotated validation data that only needs to be large enough that the proportions involved can be estimated with a decent precision. This scaling is used both for the DRFit method and standard regularized networks. Results for when we omit scaling (i.e. $\rho_c = 1$) are presented in the appendix. In all experiments a small burn-in period is used where we trained with a regular loss.

The experiments are the following;

- A small network on the MNIST dataset to classify 1's from 7's. The network has been chosen small enough that the number of parameters is small compared to the number of training examples; we used one hidden layer with eight nodes followed by a final logistic layer. Hence the network is not overparametrized per se and only the mislabels will cause it to overfit. We have hence here dropped the ridge regularization term and can study the refined effect of observation weighting in a standard network. The network we used has one hidden layer with eight nodes and ReLU activation followed by a final logistic layer.

- A subset of the well-known Cifar10 dataset. We have chosen to work with classifying cars vs airplanes. Here we use a convolutional neural network with three convolutional layers with 8, 16 and 32 filters, max pooling between each layer and Relu activations. This is followed by three dense layers with 124, 64 and 2 neurons. We have also used a cross-entropy loss for L.

For both the datasets, we have considered three levels of distribution of mislabels: (i) 20% mislabels in each class, (ii) 30% misalabels in one class and 10% in the other and (iii) 40% mislabels in one class and none in the other. We have also compared with classification without noise.

A small neural net on MNIST

In order to distill the introduction of observation weights, we have chosen to train a minimalistic network on a contaminated subset of the MNIST dataset. The network in question was taken small enough that overfitting via overparametrization in itself is not a problem and we have consequently dropped the l^2 penalty term from the target loss functions. We have selected two classes, images of ones and sevens, and then sampled down the images to a size of 14×14. The network we used has one hidden layer with eight nodes followed by a final logistic layer. We introduced random label noise where we have shifted the labels of the ones and sevens in the training set. The hyper-parameter α has been optimized on an separate validation set. We have then executed 100 training runs with the optimal α and compared this with 100 runs for the same network without observation weights. In figure 1 we find the results. We trained DRFit both by Algoritm 1 and by the analytic solver for the observation weights and compared with the same network network with no regularization. It is clearly seen that the unregularized network suffers heavily from overfitting while DRFit does not have this problem.

In figure 5 we can see that when we have 30% noise in ones and 10% noise in sevens, we almost get a perfect separation between the observation weights between the correctly classified data-points and the mislabeled ones. This explains how DRFit has virtually no problems at all with overfitting; DRfit learns the false labels early on and disregards them completely from then on and since the model is not overparametrized in itself, the test accuracy remains optimal. Some of the few misclassified images that DRFit could not detect can be seen in Figure 2 along with some correctly classified images that DRFit considered mislabeled.

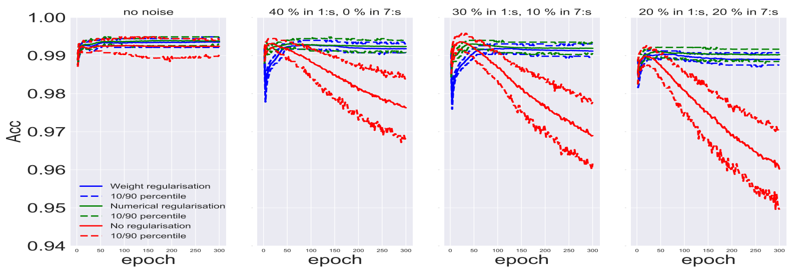

Figure 1: The mean accuracy on test data during training of 100 different random initializations, done with DRFit both with a numerical and a analytical solver and the same network without regularization. This is done in four different label noise settings. With no label noise, with 40% noise in class 1 and 0% noise in class 2, 30% noise in class 1 and 10% noise in class 2 and with 20% noise in class 1 and 20% noise in class 2

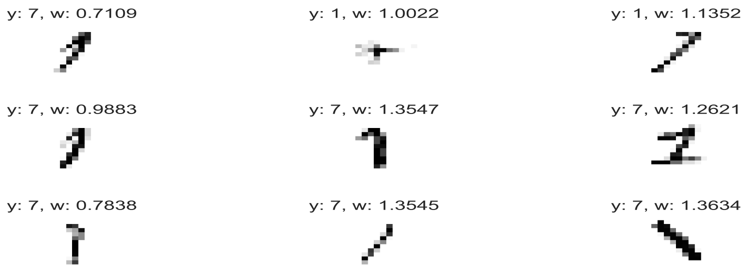

Figure 2: Example of images that where mislabeled but the DRFit method did not detect. Above of each image is the label the model was given and the weight the model set to the data point.

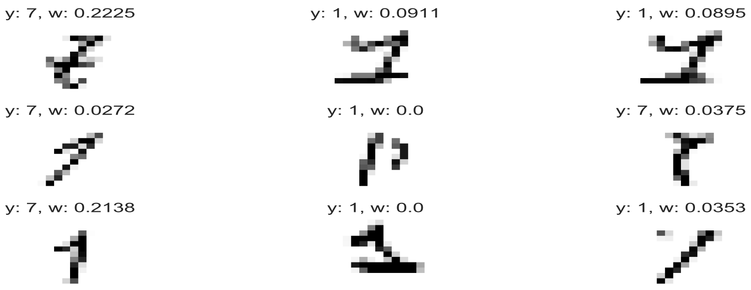

Figure 3: Example of images that are classified correctly but has been assigned a low observation weight by DRFit. Above each example you can see the label the model was given and the weight the model gave the data point

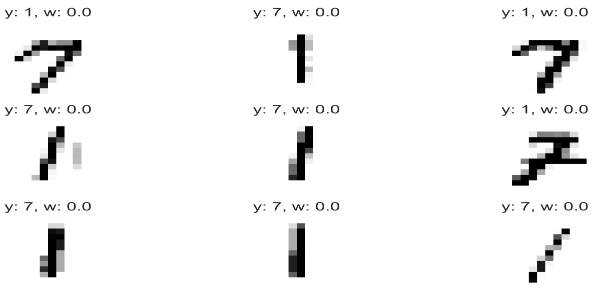

Figure 4: Example of images that where mislabeled and that DRFit could detect. Above each example is the label the model was given and the weight that DRFit assigned to the data point.

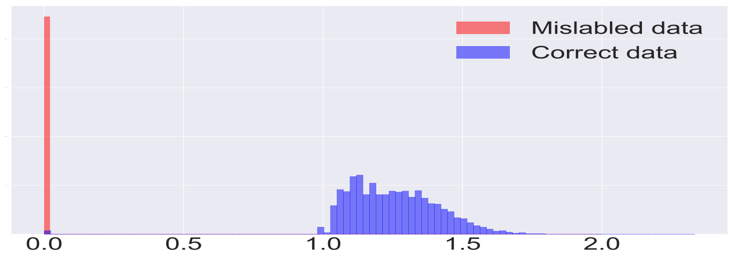

Figure 5: Histograms for the observation weight distributions for correctly labeled data and for mislabeled data. These are produced by training the DRFit on the subset of ones and sevens in the MNIST dataset.

A deep convolutional neural net on Cifar10

After finding the optimal parameters λ and α on a separate validation set, we trained the network 100 times with different random parameter initializations. We found that ridge regularization is highly sensitive to initialization and frequently crashes, meaning that during training the model starts to learn from data but suddenly drops to 50% accuracy and stays there for the rest of the training period. To make a direct comparison with DRFit we therefore remove all crashed runs from both DRFit and ridge. In these experiments, DRFit is by far more robust to parameter initialization. Indeed, DRFit did not crash for any of the 100 runs, whereas ridge for some noise settings crashed up to nearly 40 % of the runs and even 9 % of the runs without any label noise.

In Table 1 we can see the corresponding hyperparameters and the final mean accuracy on the test data for our different models on the different datasets.

Notice here that with a more uneven noise in the data, λ tends to increase leading to a stronger regularization with respect to the network parameters. Notice that α also increases with a more uneven noise in the training data leading to less regularization via observation weights, i.e. more uniformly distributed ω_i:s. So with more uneven noise in the data, more regularization is moved from the observations to the network parameters, i.e. it becomes more beneficial to restrict the network than the data itself.

In Figure 6, we can see that in the majority of the noise settings, the use of observation weights decreases overfitting. The optimal peak in accuracy seams to be unchanged between the two training algorithms.In Figure 7 we illustrate how the data weights are distributed for correct and mislabeled data respectively. This is done for the data set with 30/10-noise. Notice that α works as a scaling parameter for the weights. This means that with a lower α we would get a larger separation between correct and mislabeled data at the cost of having more correct data with lower weights; clearly too low an α will result in too much information being lost. By judging data points with observation weights below a fixed threshold as mislabeled, choosing this threshold will result in tuning the balance between the proportion of correct annotations rightly judged as correct and the proportion of mislabeled examples rightly judged as mislabeled. Figure 8 illustrates this. Notice e.g. that, when using optimal hyperparameters, one can find thresholds such that, when using the mean weight of data points over the 100 runs to compare with the threshold, more than 90% of correct labels are judged as correct and more than 90% of false labels are considered false.

class 1 (%)	class 2 (%)	Opt. λ	Opt. α	Acc test end	Acc test top	% crashes removed
			DRFit			
0	0	0.10281	3.0309	0.94734	0.95409	0
20	20	0.059512	1.1712	0.90177	0.91854	0
30	10	0.30801	1.1222	0.90878	0.91911	0
40	0	0.56716	3.1629	0.91544	0.92816	0
			RidgeNet			
0	0	0.50868	-	0.94773	0.95507	9
20	20	0.16716	-	0.82947	0.91690	38
30	10	0.26122	-	0.83802	0.91790	38
40	0	0.23280	-	0.82148	0.92814	24

Table 1: Table showing pptimal values of the hyperparameters λ and α for different noise settings, mean accuracy at the end of training, the top accuracy during training and the percentage of crashes of the runs. Both accuracies are calculated on test data over the 100 training runs with these optimal hyperparameter values.

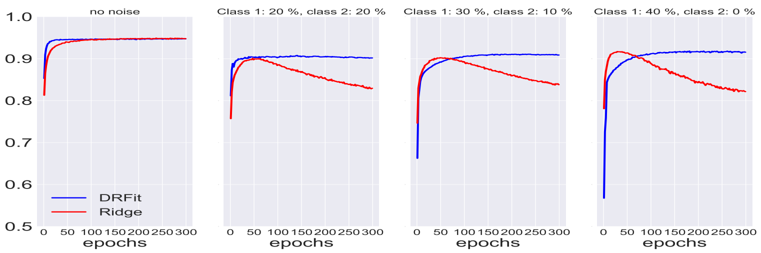

Figure 6: The average accuracy on a test data for DRFit and pure ridge regularization during training when trained on the classes cars and planes in the cifar10 dataset. In order to vary the noise we have done this in four different label noise settings. One with equally amount of noise in both classes, one with 30% noise in class one and 10% in class two, one with 40% noise in class one and no noise in class two and one setting with no label noise in the training data. In order to get a better comparison the runs that crashed are removed.

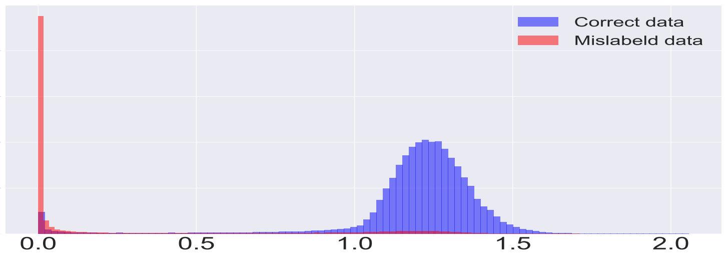

Figure 7: Histograms of the observation weight distributions over all simulations for mislabeled and correctly annotated data respectively for the 30/10-contaminated data set. The runs that crashed are removed.

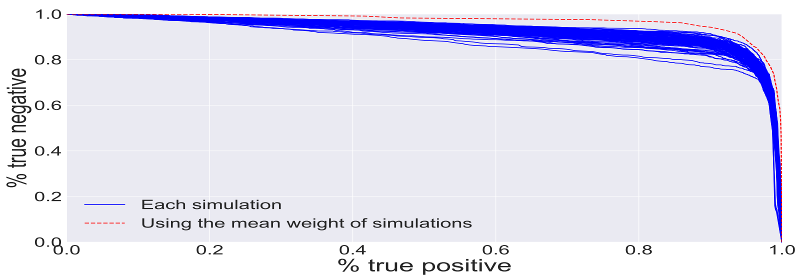

Figure 8: Curves that describe the balance between finding mislabeled points versus finding correctly labeled points while tuning a threshold t. For each value t the model classifies a data point as mislabeled if its observation weight $\omega_i < t$ and as correctly labeled if $\omega_i > t$. Each blue curve is for one training run and the red curve is the result of using the average ω_i over the training runs to compare with t.

7. Discussion

We have proposed a double regularization framework, DRFit, for the training of a predictive model in the presence of mislabeled training data. We have presented a theoretical result that supports the method and we have experimentally demonstrated that DRFit improves performance over standard regularizarion in two experiments on different data sets with different neural net predictive models and different noise distributions.

We found that combining regularization on observation weights with regularization on model parameters (or with no regularization when the model is not overspecified) results in higher, or at least as high, classification accuracy and significantly stronger robustness to overfitting.

In addition to test accuracy, we found, both on MNIST and Cifar10, that DRFit performs remarkably well for separating mislabeled training examples from those with correct labels.

In the MNIST case, we could also clearly see that mislabeled training examples that DRFit failed to detect were in fact very ambiguous.

Yet another benefit of DRFit is that early stopping does not have to be explicitly applied; we can run training until convergence. By contrast, standard neural networks with pure parameter regularization rely to a large extent on early stopping in the presence of label noise. (As shown in Li et al. (2020), early stopping will indeed make the model more robust to label noise, which is a result we also have seen in our simulations). DRFit is thus a more intelligible model in that we actually reach a minimum of the chosen loss minimization objective. We strongly believe that the reason that DRFit does not overfit is the effect that was intended, namely that mislabeled data is to a very large extent "turned off" early on in the training process and that this works as a replacement for early stopping. This means that we will get an optimization problem that we actually will optimize in contrast to the early stopping case.

The experiments on both datasets show that using observation weights does not have any adverse effects when training on a dataset without any mislabeled data. Moreover, in the MNIST case we see that DRFit is more consistent in the sense that the variation in test accuracy is smaller than for the nonregularized network.

The beneficial effect of double regularization seems to be pronounced in all tested noise distributions. While the inclusion of the scaling factors ρ_c improved performance overall, DRfit still outperformed standard regularization when $\rho_c = 1$ was used (see the appendix). In practise, we can obtain estimates of the noise level for the different classes from the validation set. Alternatively, as a two-stage training alternative, one could estimate the noise levels from weight distributions from initial training with no scaling. This latter option has been reserved for future work.

An unavoidable drawback of double regularization compared to standard ridge or lasso, is the need for two hyperparameters instead on one and hence optimization of the them is a more costly process. However experiments indicate that DRFit is fairly insensitive to hyper-parameter settings and our optimization algorithm was coarse-grained and far from a complete grid search. We also did not optimize the burn-in period or the learning rate for the observation weights in Algorithm 1, neither during hyperparameter optimization, nor during the test runs. This means that DRFit could potentially perform even better if further fine tuning in the hyperparameter optimization was applied. The optimization of these parameters is a non-trivial problem which we leave for further research.

In this paper, we restricted ourselves to binary classification. However the general formulation of DRFit was by no means restricted to that, or even to classification problems. We expect, however, that in cases of class imbalance in noise, the case of multiclass classification need more care.

Future research along the lines presented here will analyze other types of double regularization. This will potentially lead to new interesting findings on regularization synergies. Another obviously important extension is to go beyond the assumption of random noise. While random noise is a natural assumption in some situations, there are clearly situations where noise may be more pronounced for training examples that are "close" to other classes in feature space and yet other situations where it is natural to assume adversarial noise.

References

Eric Arazo Sanchez, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. 2019.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25:846–869, 2014.

Xiaoli Gao and Yixin Fang. Penalized weighted least squares for outlier detection and robust regression. *arXiv preprint arXiv:1603.07427*, 2016.

XL Gao and Yang Feng. Penalize weighted least absolute deviation regression. *Statistics and Its Interface*, 2017.

Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in neural information processing systems*, pages 10456–10465, 2018.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/koh17a.html`.

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. volume 108 of *Proceedings of Machine Learning Research*, pages 4313–4324, Online, 26–28 Aug 2020. PMLR. URL `http://proceedings.mlr.press/v108/li20j.html`.

Tonglian Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:447–461, 2016.

Bin Luo. *Robust High-dimensional Data Analysis Using a Weight Shrinkage Rule.* PhD thesis, University of North Carolina at Greensboro, 2016.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1196–1204. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5073-learning-with-noisy-labels.pdf`.

Geoff Pleiss, Tianyi Zhang, Ethan R Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *arXiv preprint arXiv:2001.10528*, 2020.

Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR 2015*, 2015. URL `http://arxiv.org/abs/1412.6596`.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020.

Sainbayar Sukhbaatar, Joan Bruna Estrach, Manohar Paluri, Lubomir Bourdev, and Robert Fergus. Training convolutional networks with noisy labels. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Appendix

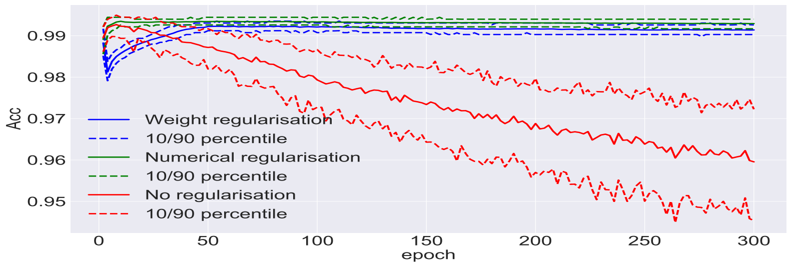

Figure 9: The mean accuracy on test data during training of 100 different random initializations, done with DRFit both with a numerical and a analytical solver and the same network without regularization. This is done in with 30% label noise in class ones and 10% noise in class sevens. In this simulation we have assumned that we don't know anything about the noise distribution so $\rho_k = 1$.

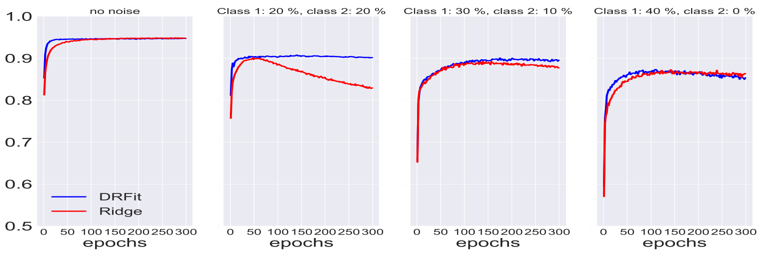

Figure 10: The average accuracy on a test data for DRFit and pure ridge regularization during training when trained on the classes cars and planes in the cifar10 dataset. In order to vary the noise we have done this in four different label noise settings. One with equally amount of noise in both classes, one with 30% noise in class one and 10% in class two, one with 40% noise in class one and no noise in class two and one setting with no label noise in the training data. In all settings we have assumed that we don't know anything about the noise distribution so $\rho_k = 1$. In order to get a better comparison the runs that crashed are removed.

Entropy weighted regularisation, a general way to debias regularisation penalties.

Zetterqvist, Olof olofze@chalmers.se
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

Jonasson, Johan jonasson@chalmers.se
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

May 21, 2021

1 Abstract

Lasso and ridge regression are well established and successful models for variance reduction and, for lasso, variable selection in regression problems where the number of estimated parameters is large compared to the number of observations. That however comes with a significant bias for large parameters. Many authors have, with varying degrees of success, suggested different modifications to come to terms with this without sacrificing the benefits.

Here we propose a general method that learns individual weights for each term in the regularisation penalty (e.g. lasso or ridge). To bound the amount of freedom for the model to choose the weights, a new regularisation term, that imposes a cost for choosing small weights, is introduced. If the form of this term is chosen wisely, the apparent doubling of the number of parameters vanishes, by means of solving for the weights in terms of the parameter estimates.

This paper is focused on the case where the regularisation on the weights is unnormalized entropy. The resulting optimisation problem is potentially nonconvex, but it is shown that convexity holds under reasonable assumptions. The model is shown to produce consistent parameter estimates as the number of observations $n \to \infty$, in the case where the number of parameters is independent of n. Moreover we show that the lasso version of the model is sign consistent in a very strong sense, even when the number of nonzero parameters grows linearly and the number of zero parameters grows like e^{n^δ}.

Experimentally, the proposed method in four different forms is compared with adaptive lasso, SCAD, standard lasso, standard ridge and standard OLS. The four forms are the lasso and ridge versions (EWL and EWR), both with and without forcing hyperparameters to be chosen to guarantee convexity of the corresponding optimisation problem. The comparisons are carried out on synthetic data with varying degrees of noise variance and varying degrees of correlation between covariates. It is found that EWL in both settings performs on par with adaptive lasso and SCAD, both in terms of mean squared error and deviation of estimated parameters from their true values. EWR performs well only when allowed to drop the convexity condition. The level of performance appears to be quite insensitive to the choice of hyperparameters.

2 Introduction

Consider the standard linear regression model

$$Y = X\beta^* + \xi$$

where $X \in R^{n\times p}$ is the design matrix, $\beta^* \in R^p$ is the unknown parameter vector and $\xi \in R^n$ a $N(0, \sigma^2\mathbf{I}_n)$-distributed noise vector. Then the standard OLS estimator $\tilde{\beta} = \arg\min_\beta \frac{1}{2}||Y - X\beta||_2^2 = (X^TX)^{-1}X^TY$ is the optimal estimator of β^* in several ways, provided that the number of parameters is small compared to the number of observations. However, when the number of parameters is large, a serious problem with the OLS estimator is its large variance and the risk of overfitting. Also, when one purpose with the analysis is variable selection, another drawback of OLS is that the estimator will never estimate any β_j^* as 0. In order to attack these problems, several different regularisation techniques have been suggested. The general idea behind these is that one adds a penalty term to the OLS loss. This term is typically increasing in the absolute values of the parameters. The regularised estimator in its general form can the be formulated as

$$\tilde{\beta} = \arg\min_\beta \frac{1}{2}||Y - X\beta||_2^2 + f_\lambda(\beta; \theta). \tag{1}$$

Here λ is a hyperparameter that determines the strength of regularisation, and θ a set of hyperparameters that determines the shape of the penalty term. These can be optimised via cross-validation on training data or performance on a separate validation set. For the following discussion, it is useful to keep in mind what conditions for considering an estimator to be good are. Common such conditions are the following.

- **Unbiasedness.** Parameter estimates should be unbiased or very close to unbiased, in particular for parameters that are far from zero.
- **Consistency** of the estimated parameters; the parameter estimator converges in probability to the true parameter as $n \to \infty$.
- **Continuity.** Parameter estimates should be continuous as functions of the data.

When the purpose of estimation is also variable selection, the following can be added.

- **Sparsity.** This means that (as many as possible of the) parameters whose true value is zero should be estimated as zero.
- **Sign consistency.** With probability converging to one in the number of observations, all parameter estimates have the same sign as the true parameter (where $sign(0) = 0$).

Two examples of regularisation penalties, that are arguably the most commonly used (along with the elastic net, a combination of the two), are ridge regression and lasso regression [Tibshirani, 1996]. These are both examples of the following special case of (1):

$$\tilde{\beta} = \arg\min_\beta \frac{1}{2}||Y - X\beta||_2^2 + \lambda\sum_{j=1}^p |\beta_j|^\alpha. \tag{2}$$

Here λ and α are two hyperparameters that determine the strength and shape of the regularisation. Taking $\alpha = 1$ gives lasso regression and $\alpha = 2$ gives ridge regression.

The lasso estimator gives rise to sparse estimates which are continuous with respect to data. In [Knight and Fu, 2000] it is shown that when p does not grow with n and under mild conditions lasso is consistent if λ is chosen wisely. In [Zhao and Yu, 2006] it is shown that under similar conditions, it also is sign consistent even when p grows with n provided that the growth satisfies some extra conditions.

(In [Zou, 2006] some cases where the parameter estimates are not consistent are also demonstrated.) However it is well known that in practice, balancing variance against bias optimally in lasso regression produces a large bias for large parameters.

For general $\alpha > 0$, the estimator (2) is known as the bridge estimator, [Frank and Friedman, 1993]. In [Knight and Fu, 2000], it is shown that the bridge estimator can be consistent. In [Huang et al., 2008] it is shown that when $\alpha < 1$, the model can be sign consistent. In this case, the estimates also have a small bias for larger parameters but fail to be continuous with respect to data. For $\alpha > 1$ the estimates are continuous, but cannot detect zero coefficients and have a larger bias.

Since no version of the bridge estimator satisfies all key properties, variants of the lasso estimator have been developed, with the goal of reducing the bias and guarantee consistency and sign consistency. One way to do this is to use a weighted version of the lasso estimator. Some examples of this are [Zou, 2006, Jung, 2011, Bergersen et al., 2011, Zhang, 2011, Javanmard et al., 2018, Bellec and Zhang, 2019]. The first of these is the well known Adaptive Lasso Estimator. The idea is to introduce a weight, ω_j, for each parameter β_j and estimate β as

$$\tilde{\beta} = \arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \lambda \sum_j \omega_j |\beta_j|.$$

The word "adaptive" refers to that the weights are data generated; first an initial estimate, e.g. an OLS estimate of β is computed, and then ω is set based on that result. One suggestion from the authors is to use $\omega_j = \frac{1}{|\hat{\beta}_j|^\gamma}$, where γ is a hyperparameter and $\hat{\beta}$ is the initial estimate.

Another approach is of course to use a loss penalty different from the Bridge loss. One successful example of this is the Smoothly Clipped Absolute Deviation estimator (SCAD) [Fan and Li, 2001], where

$$f_\lambda(\beta_j; a) = \begin{cases} \lambda|\beta_j| & it\ 0 \leq |\beta_j| \leq \lambda \\ -\frac{|\beta_j|^2 - 2a\lambda|\beta_j| + \lambda^2}{2(a-1)} & if\ \lambda \leq \beta_j \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} & if\ |\beta_j| \geq a\lambda \end{cases}.$$

Both the adaptive lasso and the SCAD estimator satisfy the criteria formulated above. However, adaptive lasso relies on the quality of the inital estimates $\hat{\beta}_j$. We also need to do an extra estimation step to find $\tilde{\beta}$, which may be costly. The SCAD estimator does not suffer from any of these drawbacks, but it does not appear to be as robust with respect to the choice hyperparameters, as we will illustrate with experiments in Section 7.

In this paper, we introduce a novel general strategy/point of view for reducing bias in regularisation penalties. Like other techniques, e.g. adaptive lasso, we also use different weights ω_j for different parameters β_j. Initially, we regard these as a set of new parameters, apparently doubling the number of parameters of the model. However, by introducing *regularisation on the weights* of a wisely chosen form on how the weights are penalized for being low and solving for them in terms of the β_j:s, we end up with a model with only two hyperparameters in addition to the β_j:s. In Section 4 we introduce the method and observe how solving for the weights gives an explicit expression for them in terms of the β_j:s. In Section 5 we present some properties when the method is used with lasso and ridge penalties. Section 6 presents the numerical algorithms for optimisation. Finally Section 7 contains results of numerical experiments and comparisons between ordinary least square (OLS), lasso, ridge, adaptive lasso, SCAD and a few instances of the new method. Proofs are given in the appendix.

3 Notation

We are considering the standard linear regression model of the form

$$Y = X\beta^* + \xi$$

As is common, we assume that the design matrix $X \in R^{n \times p}$ is standardised so that each column has mean 0 and standard deviation 1. The vector $\beta^* \in R^p$ contains the true unknown parameters and the ξ_k:s are independent normal with mean 0 and standard deviation σ. Without loss of generality, we assume that the first r elements in β^* are nonzero and that the next $q = p - r$ elements are zero. The matrix $C = X^T X$ corresponds to the correlation matrix for the columns in X. We sometimes write

$$X^T X = C_n := \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

where $C_{11} \in R^{r \times r}$ corresponds to the covariates of the nonzero parameters, $C_{22} \in R^{q \times q}$ the covariates of the zero parameters and $C_{12} = C_{21}^T \in R^{r \times q}$ contains the covariances between non zero and zero covariates. To consider consistency of the model, we use the notation β^*_n, λ_n and γ_n to make clear that these variables can change with respect n, even though we sometimes drop the indexes for better readability when these can be understood. We use the same definition as in [Zhao and Yu, 2006] for sign consistency.

Definition. An estimator $\tilde{\beta}_n$ of β^*_n is sign consistent if

$$\lim_{n \to \infty} P(sign(\tilde{\beta}_n) = sign(\beta^*_n)) = 1$$

□

Here $sign(a) = \mathbf{1}_{\{a \geq 0\}} - \mathbf{1}_{\{a \leq 0\}}$ and the sign of a vector is considered element-wise: $sign(x_1, \ldots, x_n) = (sign(x_1), \ldots, sign(x_n))$. Loosely speaking, being sign consistent means that the estimator in the limit works as a perfect variable selector.

4 Methods

Recall the general form (1) of regularization:

$$\tilde{\beta} = \arg\min_{\beta} \frac{1}{2}||Y - X\beta||^2 + \lambda f(\beta)$$

where λ sets the amount of regularisation. The function $f\lambda(\beta)$ can in principle be any function with global minimum at zero, but for the purpose of efficient optimisation it is often chosen convex. For lasso and ridge $f(\beta)$ is $||\beta||_1$ and $||\beta||_2^2$ respectively. Since both of these put an equal amount of penalty on all parameters, they lead to biased estimates. As for e.g. adaptive lasso and SCAD, but in a different way, we suggest a modification by extending the regularisation function to give each parameter its own unique penalty weight:

$$(\tilde{\beta}, \tilde{\mathbf{u}}) = \arg\min_{\beta, \mathbf{u}} \frac{1}{2}||Y - X\beta||_2^2 + \sum_{j=1}^{p} \left[\lambda u_j f(\beta_j) + \gamma(u_j \log(u_j) - u_j + 1)\right]. \tag{3}$$

By adding the weight parameters $u_1, \ldots, u_p$, the amount of regularisation of β is determined dynamically. The term $\gamma(u_i \log(u_i) - u_i + 1)$ is a regularisation penalty on the weights that hold them back from differing too much from 1 and the hyperparameter γ determines the strength of this regularisation. Since $\lim_{x \to 0+}(d/dx)x\log(x) - x = -\infty$, we get $\tilde{u}_j > 0$ for all j. By setting γ increasingly large, the regularisation term will have an effect more and more similar to $\lambda f(\beta_j)$.

The minimisation problem (3) now involves $2p$ parameters. However, it is easy to solve for $\mathbf{u}$ in terms of β:

Theorem 1. *Let $(\tilde{\beta}, \tilde{\mathbf{u}})$ be as in (3). Then*

$$\tilde{\beta} = \arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \gamma \sum_{i=1}^{p}(1 - e^{-\frac{\lambda}{\gamma} f(\beta_j)}) \tag{4}$$

Hence in fact, since Theorem 1 makes away with the u_j:s, we arrive at a model with only the two hyperparameters λ and γ in addition to the regression parameters. However, it should be noted that since $1 - e^{-x}$ is a non-convex function, we cannot guarantee that the minimisation objective in (4) is convex for an arbitrary choice of (λ, γ). In the following sections, we investigate key properties of (4) theoretically and experimentally for the cases $f(\beta_j) = |\beta_j|$ and $f(\beta_j) = \beta_j^2$. We name these instances of the model as *Entropy Weighted Lasso* (EWL) and *Entropy Weighted Ridge* (EWR).

5 Properties of EWL and EWR

First consider EWL. This gives us the minimisation problem

$$\tilde{\beta} = \min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \gamma \sum_{j=1}^{p} (1 - e^{-\frac{\lambda}{\gamma}|\beta_j|}) \tag{5}$$

Since the term $(1 - e^{-\frac{\lambda}{\gamma}|\beta_i|})$ is not convex with respect to β_j we do not know for sure that the minimisation objective has a unique local minimum. However in many situations, convexity holds with just a small measure of care with the choice of (λ, γ):

Theorem 2. *Let s_1^2 be the smallest eigenvalue of $X^T X$. The minimisation problem (5) is convex whenever $\gamma > \frac{\lambda^2}{s_1^2}$.*

So given some prior knowledge on the design matrix X, we can guarantee convexity of the minimisation problem and hence that there is an unique local minimum, which is then of course also the global minimum. As long as $p < n$, $X^T X$ is typically not singular and it is reasonable to expect that s_1^2 is of order n and the condition $\gamma > \frac{\lambda^2}{s_1^2}$ becomes that $\gamma/\lambda^2 \geq c/n$ for some finite constant c independent of n.

Choosing $\gamma < \frac{\lambda^2}{s_1^2}$ sometimes indeed leads to multiple local minima and also non-continuous $\tilde{\beta}$ with respect to data. This is demonstrated in Figure 1 for the $p = 1$ setting. (It could be noted however that in this case, the bias for large β is very small.)

The following result establishes that under mild conditions, the general estimator $\tilde{\beta}$ in (4) is consistent when the number of parameters p is independent of n.

Theorem 3. *Assume $\frac{\gamma_n}{n} \to \gamma_0 \geq 0$, $\frac{\lambda_n}{n} \to \lambda_0 \geq 0$, $\lim_{n\to\infty} \frac{X^T X}{n} = C$ is nonsingular and that f is a convex function. Let*

$$\tilde{\beta} = \arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \gamma_n \sum_{j=1}^{p} (1 - e^{-\frac{\lambda_n}{\gamma_n} f(\beta_j)}) \tag{6}$$

Then $\tilde{\beta} \to_p \arg\min(Z)$ where

$$Z(\beta) = (\beta - \beta^*)C(\beta - \beta^*) + \gamma_0 \sum_{j=1}^{p} (1 - e^{-\frac{\lambda_0}{\gamma_0} f(\beta_j)}),$$

where we define the second term to be 0 if $\lambda_0 = 0$ or $\gamma_0 = 0$. Hence if $\gamma_n = o(n)$ or $\lambda_n = o(n)$, then $\tilde{\beta}$ is consistent.

The following result proves that with $f(x) = |x|$, i.e EWL, $\tilde{\beta}$ is sign consistent as well as consistent in a very strong sense under conditions of a different nature than those of Theorem 3.

Theorem 4. *Assume that $||C_{11}^{-1}||_\infty < \frac{K_1}{n}$ and $||C_{21}||_\infty < K_2 n$ for some constants $K_1, K_2 < \infty$ independent of n, where $||\cdot||_\infty$ is the ∞-operator norm. Assume also that there is a constant $\delta > 0$*

such that for all n, $n > \lambda_n > n^{1/2+2\delta}$ and $\lambda_n|\beta_{nj}|/\gamma_n > n^{2\delta}$ and $|\beta_{nj}| > n^{-1/2+2\delta}$ for all $j = 1, \dots, r_n$. Assume in addition that $\gamma_n > 1$ and $q_n < e^{n^\delta}$.

Let L be the minimisation objective

$$L(\beta) = L_n(\beta; Y) = \frac{1}{2}||Y - X\beta)||_2^2 + \gamma_n \sum_{j=1}^{p_n}(1 - e^{-\frac{\lambda_n}{\gamma_n}|\beta_j|}).$$

Then with probability at least $1 - e^{-n^\delta}$, L has a local minimum $\bar{\beta}$ such that with probability $1 - e^{-n^\delta}$, $||\bar{\beta} - \beta^*||_\infty < n^{-1/2+\delta}$ and $sign(\bar{\beta}) = sign(\beta^*)$. Hence if L has a unique minimum, then

$$\tilde{\beta}_n = \arg\min_\beta \left[\frac{1}{2}||Y - X\beta)||_2^2 + \gamma_n \sum_{j=1}^{p_n}(1 - e^{-\frac{\lambda_n}{\gamma_n}|\beta_j|})\right]$$

satisfies with probability at least $1 - e^{-n^\delta}$ that $||\tilde{\beta} - \beta^*||_\infty < n^{-1/2+\delta}$ and $sign(\tilde{\beta}) = sign(\beta^*)$.

Note that even though we allow q_n to be very large (assuming that correlations between covariates for zero and nonzero parameters respectively are correspondingly small), the existence of C_{11}^{-1} implicitly assumes that $r_n < n$.

If the minimisation objective $L(\beta)$ of Theorem 4 is convex, then it has a unique minimum and $\tilde{\beta}$ is sign consistent. By Theorem 3, this holds if $\lambda_n^2/\gamma_n < s_1^2$. If $p_n < n$ it is natural to expect that s_1^2 is of order n and then there is room in Theorem 4 for choosing λ_n and γ_n in that way without violating the conditions of the theorem: e.g. $\gamma_n = n^{5\eta}$ and $\lambda_n = n^{1/2+2\eta}$ for a suitable $\eta > 0$.

Next we consider EWR, i.e. $f(\beta_i) = \beta_i^2$. This means that the regularisation term is now $\gamma(1 - e^{-\frac{\lambda}{\gamma}\beta_j^2})$. As in the case of EWL the minimisation objective is not necessarily convex for all λ and γ. The following theorem provides a sufficient condition for convexity.

Theorem 5. *The minimization problem*

$$\tilde{\beta} = \min_\beta L(\beta) = \min_\beta \frac{1}{2}(Y - X\beta)^2 + \gamma \sum_i (1 - e^{-\frac{\lambda}{\gamma}\beta_i^2}). \tag{7}$$

is convex if $\lambda < \frac{s_1^2 e^{\frac{3}{2}}}{4}$

Interestingly, the constraint of Theorem 5 does not depend on γ and if s_1^2 grows as order n. This leaves correspondingly more freedom for choosing λ.

Setting $\lambda > \frac{s_1^2 e^{3/2}}{4}$ can make the estimated parameters non-continuous with respect to data. When it comes to consistency, Theorem 3 applies to conclude that $\tilde{\beta}$ is consistent for constant p with $\gamma = o(n)$ and $\lambda_n = o(n)$. However, EWR can obviously not be sign-consistent. Figure 1 plots $\tilde{\beta}$ as a function of Y for $p = 1$.

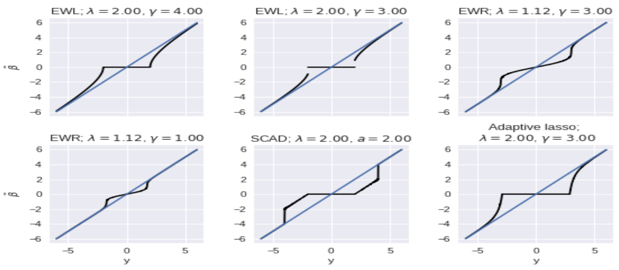

Figure 1: How $\tilde{\beta}$ depends on y with EWL, EWR, SCAD and adaptive lasso for a single data point y and a single parameter β. In this case $s_1^2 = 1$ and EWL is convex whenever $\gamma > \lambda^2$ and EWR is convex whenever $\lambda < e^{3/2}/4 \approx 1.12$.

Note from Figure 1 that as long as EWL is convex, EWL and adaptive lasso give very similar curves. Since both are based on the lasso penalty, this is not surprising. Also noticeable is the difference in "width" of the "plateau" between EWL and EWR. The width in EWL is determined by λ while γ determines how fast the bias is reduced as $|y|$ grows. The shape of EWR is determined by λ while γ determines the width of the plateau. (Of course strictly speaking EWR does not have a plateau, only a plateau-like part around 0.)

6 Training algorithms

Training algorithm for EWL

Let $L(\beta)$ be the minimisation objective in (5), To minimise L, we use coordinate descent: iteratively minimise with respect to β_j, keeping β_i, $i \neq j$, fixed, run through $j = 1, \ldots, p$ and repeat until convergence. The j'th component of the set of subgradients of L is

$$-\rho_j + n\beta_j + \begin{cases} -\lambda e^{\frac{\lambda}{\gamma}\beta_j} & \beta_j < 0 \\ [-\lambda, \lambda] & \beta_j = 0 \\ \lambda e^{-\frac{\lambda}{\gamma}\beta_j} & \beta_j > 0 \end{cases}$$

where $\rho_j = X_j^T(Y - X_{i\neq j}\beta_{j\neq i})$, X_j is the j'th column of X, $X_{i\neq j}$ is X matrix except X_j and $\beta_{j\neq i}$ is the vector β except β_j.

Set this to zero and solve for β_j. For this, we need to determine when the solution is nonzero and if so, determine its exact value. First consider if there is a solution $\beta_j > 0$. We claim that this is the case if and only if $\rho_j > \lambda$. A solution $\beta_j > 0$ exists if and only if

$$-\rho_j + n\beta_j + \lambda e^{-\frac{\lambda}{\gamma}\beta_j} = 0$$

By substituting $\hat{\beta}_j = \frac{\lambda}{\gamma}\beta_j$ the equation becomes

$$\hat{\beta}_j + \frac{\lambda^2}{n\gamma}e^{-\hat{\beta}_j} = \frac{\lambda}{n\gamma}\rho_j$$

which is equivalent to

$$\left(\hat{\beta}_j - \rho_j\frac{\lambda}{n\gamma}\right)e^{(\hat{\beta}_j - \rho_j\frac{\lambda}{n\gamma})} = -\frac{\lambda^2}{n\gamma}e^{-\rho_j\frac{\lambda}{n\gamma}} \tag{8}$$

Note that $0 \geq -\frac{\lambda^2}{n\gamma}e^{-\rho_j \frac{\lambda}{n\gamma}} \geq -\frac{1}{e}$ if and only if $\rho_j \geq \lambda$; the upper bound is trivial and the lower bound holds if and only if $\rho_j \geq \gamma$. Hence this equation has no real solutions for $\rho_j < \lambda$, which in particular rules out that there is a positive solution when $\rho_j \leq \lambda$ (for $\rho_j = \lambda$, the solution is 0). For $\rho_j > \lambda$, there are two real solutions, namely

$$\hat{\beta_j} = W\left(-\frac{\lambda^2}{n\gamma}e^{-\rho_j \frac{\lambda}{n\gamma}}\right) + \rho_j \frac{\lambda}{n\gamma}$$

which gives

$$\beta_j = \frac{\gamma}{\lambda} W\left(-\frac{\lambda^2}{n\gamma}e^{-\rho_j \frac{\lambda}{n\gamma}}\right) + \frac{\rho_j}{n}, \tag{9}$$

where W is either of the two branches W_+ and W_- (the positive and negative branch respectively) of the Lambert W-function, taking values in $(-\infty, -1)$ and in $[-1, \infty)$ respectively. However, as we will shortly prove, the solution with $W = W_-$ gives a negative value of β_j and since we are looking for a positive solution, this branch can be discarded. Now check that the solution with $W = W_+$ produces a positive value for β_j

Setting $W = W_+$ and requiring the right hand side of (9) to be positive, the resulting inequality for ρ_j gives

$$\frac{\gamma}{\lambda} W_+\left(-\frac{\lambda^2}{n\gamma}e^{-\rho_j \frac{\lambda}{n\gamma}}\right) + \frac{\rho_j}{n} > 0$$

which is equivalent to

$$W_+\left(-\frac{\lambda^2}{n\gamma}e^{-\rho_j \frac{\lambda}{n\gamma}}\right) > -\frac{\lambda\rho_j}{n\gamma}.$$

Since W_+ is increasing, taking the inverse yields

$$-\frac{\lambda^2}{n\gamma}e^{\rho_j \frac{\lambda}{n\gamma}} > -\rho_j \frac{\lambda}{n\gamma} e^{-\rho_j \frac{\lambda}{n\gamma}}$$

which is equivalent to

$$\left(\rho_j \frac{\lambda}{n\gamma} - \frac{\lambda^2}{n\gamma}\right) e^{-\rho_j \frac{\lambda}{n\gamma}} > 0,$$

i.e. $\rho_j > \lambda$. In short, whenever $\rho_j > \lambda$, one solution with respect to β_j of $(\partial/\partial\beta_j)L(\beta) = 0$ satisfies $\beta_j > 0$ and is given by (9). The solution is the unique positive solution provided that that using W_- in (9) gives negative values of β. For this, note that setting $\rho_j = \lambda$ gives

$$\beta_j = \frac{\gamma}{\lambda} W_-(-\frac{\lambda^2}{n\gamma}e^{-\frac{\lambda^2}{n\gamma}}) + \frac{\lambda}{n} = -\frac{\gamma}{\lambda}\frac{\lambda^2}{n\gamma} + \frac{\lambda}{n} = 0.$$

Taking the derivative of β_j with respect to ρ_j in (9) gives

$$\frac{d\beta_j}{d\rho_j} = \frac{1}{n\left(1 + W_-\left(-\frac{\lambda^2}{n\gamma}e^{-\rho_j \frac{\lambda}{n\gamma}}\right)\right)} < 0$$

as the output of W_- is in $(-\infty, -1)$. Hence, since $\rho_j > \lambda$, we get $\beta_j < 0$.

By symmetry, the solution satisfies $\beta_j < 0$ if and only if $\rho_j < -\lambda$ and is then given by

$$\beta_j = -\frac{\gamma}{\lambda} W(-\frac{\lambda^2}{n\gamma}e^{\rho_j \frac{\lambda}{n\gamma}}) + \frac{\rho_j}{n},$$

where W is again the positive branch of W.

Finally we need to establish that $\beta_j = 0$ is a solution whenever $-\lambda \leq \rho_j \leq \lambda$. The subgradients are $-\rho_j + n\beta_j + [-\lambda, \lambda]$. With $\beta_j = 0$ this set becomes $[-\lambda - \rho_j, \lambda - \rho_j]$, which includes 0 precisely when $-\lambda \leq \rho_j \leq \lambda$.

Putting all things together we end up with with the coordinate update

$$\beta_j = \begin{cases} \frac{\gamma}{\lambda} W(-\frac{\lambda^2}{n\gamma} e^{-\rho_j \frac{\lambda}{n\gamma}}) + \frac{\rho_j}{n} & \rho_j > \lambda \\ 0 & -\lambda \leq \rho_j \leq \lambda \\ -\frac{\gamma}{\lambda} W(-\frac{\lambda^2}{n\gamma} e^{\rho_j \frac{\lambda}{n\gamma}}) + \frac{\rho_j}{n} & \rho_j < -\lambda \end{cases} \tag{10}$$

In practice, in order to increase the numerical stability, we scale with n and make the substitutions $\hat{\rho} = \frac{\rho}{n}$, $\hat{\lambda} = \frac{\lambda}{n}$ and $\hat{\gamma} = \frac{\gamma}{n}$. The training algorithm is summarised in Algorithm 1. Note that if γ should be chosen so that (5) is not convex, the algorithm can still be used and converges. However, we are not guaranteed to converge to a global optimum.

Algorithm 1 Training algorithm with weighted L1 regularisation.

```
procedure TRAIN($X$,$Y$,$\lambda$,$\gamma$,$N$ = max number of iterations, $\epsilon$ = tolerance)
    $\lambda \leftarrow \lambda/n$
    $\gamma \leftarrow \gamma/n$
    $\beta \leftarrow \mathbf{0}$
    for iteration = $0 \dots N$ do
        $\hat{\beta} \leftarrow \beta$
        perm = random permutation of [1...$m$]
        for $j \in$ perm do
            $\rho \leftarrow X_j(Y - X_{i \neq j}\beta_{i \neq j})/n$
            if $|\rho| > \lambda$ then
                $\beta_j \leftarrow \frac{\gamma}{\lambda} sign(\rho) W(-\frac{\lambda^2}{\gamma} e^{-|\rho|\frac{\lambda}{\gamma}}) + \rho$
            else
                $\beta_j \leftarrow 0$
        if max($|\beta - \hat{\beta}|$) $< \epsilon$ then
            Break loop
    return $\beta$
```

Training algorithm for EWR

To solve the minimisation problem in (7) we start from equation (3) and solve for β and u iteratively. By fixing one and solving for the other, we end up with the solutions:

$$\arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \sum_i [\lambda u_i f(\beta_i) + \gamma(u_i \log(u_i) - u_i + 1)] = \left[X^T X + \lambda diag(\mathbf{u})\right]^{-1} X^T Y$$

$$\arg\min_{\mathbf{u}} \frac{1}{2}||Y - X\beta||_2^2 + \sum_{i=1}^{n} [\lambda u_i f(\beta_i) + \gamma(u_i \log(u_i) - u_i + 1)] = e^{-\frac{\lambda}{2\gamma}\beta^2}$$

By iterating between these two solutions, we get the solver in algorithm 2. If we use a λ that guarantees convexity, we end up in the global minimum of (7).

Algorithm 2 Training algorithm with weighted L2 regularisation.

```
procedure TRAIN($X$,$Y$,$\lambda$,$\gamma$,$N$ = max number of iterations, $\epsilon$ = tolerance)
    $\mathbf{u} \leftarrow \mathbf{1}$
    $\beta \leftarrow \mathbf{0}$
    for iteration $= 0 \dots N$ do
        $\hat{\beta} \leftarrow \beta$
        $\beta \leftarrow (X^TX + \lambda diag(\mathbf{u}))^{-1}X^TY$
        $\mathbf{u} \leftarrow e^{-\beta^2 \frac{\lambda}{2\gamma}}$
        if $\max(|\beta - \hat{\beta}|) < \epsilon$ then
            Break loop
    return $\beta$
```

7 Experiments

The experiments have been done with two collections of synthetic data sets. The collection for the first experiment contains data sets with uncorrelated covariates and varying noise variance. The collection for the second experiment consists of data sets that have a fixed noise variance, but varying degrees of correlation between covariates. In more detail, the collections of data are as follows.

1. **Experiment 1.** Independent covariates in the design-matrix $X \in R^{100\times 20}$ and varying noise variance σ^2 in the response: Each element in X is sampled uniformly in the interval $[-25, 25]$. The first 5 elements in the true parameter vector β^* are sampled uniformly in the interval $[-10, 10]$ and the rest of the 15 parameters are set to 0. Then Y is generated by $Y = X\beta^* + \xi$ where $\xi \in N(0, \sigma^2\mathbf{I})$. The standard deviation σ varies between 0 and 40 over the different data sets in this collection.

2. **Experiment 2.** We create β^* exactly as for the first experiment; $\beta_1, \dots, \beta_5$ are chosen independent and uniform on $(-10, 10)$ and $\beta_6 = \dots = \beta_{20} = 0$. The observations Y are sampled as $Y = X\beta^* + \xi$ with $\xi \sim N(0, \sigma^2\mathbf{I})$ with $\sigma = 30$. However, the design matrices are in this case constructed with correlated columns. More precisely, $X \in R^{100\times 20}$ is set to $X = ZD$, where $Z \in R^{100\times 20}$ with elements sampled from a uniform distribution on the interval $(-25, 25)$ and $D \in R^{20\times 20}$ is given by for each $k = 1, \dots, 5$, $D_{kk} = 1$, $D_{k,k+5} = D_{k,k+10} = D_{k,k+15} = \rho$ and $D_{ki} = 0$ for all other i. In other words, the parameters have been split into five groups of four parameters, each of which exactly one is nonzero and correlation is imposed within groups, but not between groups. The correlation controlling entity ρ varies between 0 and 0.8 over the data sets in this collection.

In both experiments, we evaluate each model by (i) normalizing the covariates, (ii) training it on training data, including a hyper-parameter search with cross-validation, and (iii) evaluating it on a separate test set. The evaluation is done in terms two different measures: (a) the L2 distance between the estimated parameters and the true parameters and (b) the mean square error of predictions on the test set.

Experiment 1. For each σ, the models have been run on 100 independently sampled data sets and the results displayed in Figure 2 are averages over these runs. For comparison, this has been done for seven different estimators: ordinary least square, lasso, ridge, adaptive lasso, SCAD, EWL and EWR, where the last two come with two hyperparameter settings each: one in the range where convexity of the loss function is guaranteed and one without that restriction. Some of the results displayed have been extracted in numeric form and are given in Tables 1 and 2 in the appendix.

Figure 2 displays the results of the evaluation. Figure 5 plots how the optimal hyper parameters change with σ. The latter plot has been put in the appendix as it does not seem to reveal any noteworthy observations.

In addition, Figure 3 displays proportions of parameters estimated correctly in terms of their signs as functions of the amount of regularisation. This is done with $\sigma = 30$ for the relevant models, i.e. SCAD, lasso, adaptive lasso and EWL.

Looking at the results in Figures 2, it is noteworthy that EWR greatly benefits when allowed to move into the non-convex setting, whereas EWL behaves well already with convex loss. Overall, SCAD, adaptive lasso, EWL in both settings and EWR in the non-convex setting show very similar performance.

The most interesting observation to make in Figure 3 is that SCAD stands out. While the curves for lasso, adaptive lasso and EWL are almost indistinguishable, SCAD is clearly more sensitive to the amount of regularisation for being sign-consistent.

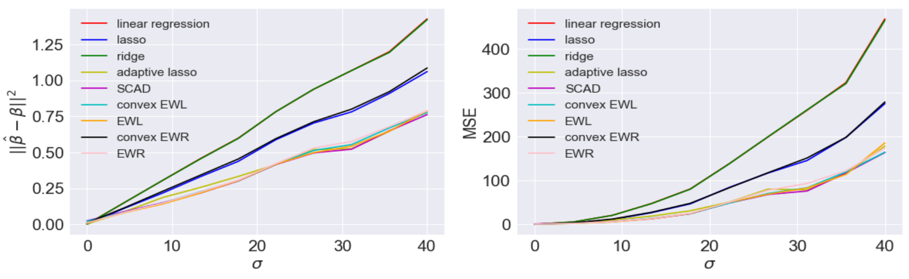

Figure 2: The average L_2 distance between the estimated parameters $\hat{\beta}$ and the true parameters β (**left**) and the mean squared error of predictions on test data (**right**) over 100 runs as functions of the noise standard deviation for nine models on uncorrelated covariates.

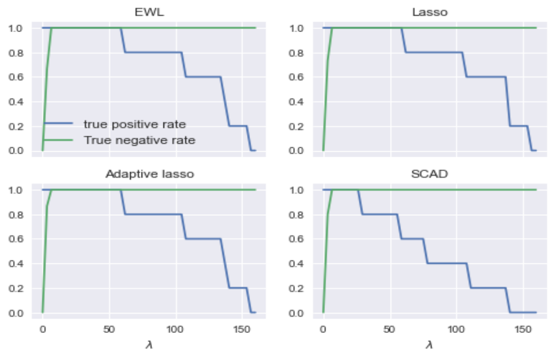

Figure 3: The proportion of correct signs on estimated parameters β as functions of λ. The blue curve represents the proportion of nonzero parameters that are estimated with the correct sign. The green curve shows the proportion of zero parameters that are estimated as zero. These models are trained on data where $X \in R^{100\times 20}$ where 5 of the true parameters are separate from zero. Each element in X are sampled from a uniform distribution on the interval $[-25, 25]$. The standard deviation of the noise is set to $\sigma = 30$. In each setting parameters except λ has been fixed to a constant or in the case of EWL such that the loss function is convex. Since these only determines the penalty shape these will not affect the amount of correct signs.

Experiment 2. The procedure is very similar to that of Experiment 1. For each ρ, the models have been run on 100 independently sampled data sets and the results displayed in Figure 4 are averages over these runs. For comparison, this has been done for the same estimators as for Experiment 1: ordinary least square, lasso, ridge, adaptive lasso, SCAD, EWL and EWR, where the last two again come with two hyperparameter settings each: one in the range where convexity of the loss function is guaranteed and one without that restriction.

As for Experiment 1, some of the displayed have been extracted in numeric form and are found in Tables 1 and 2 in the appendix.

Figure 6 plots how the optimal hyper parameters change with the amount of correlation. The plot has been deferred to the appendix.

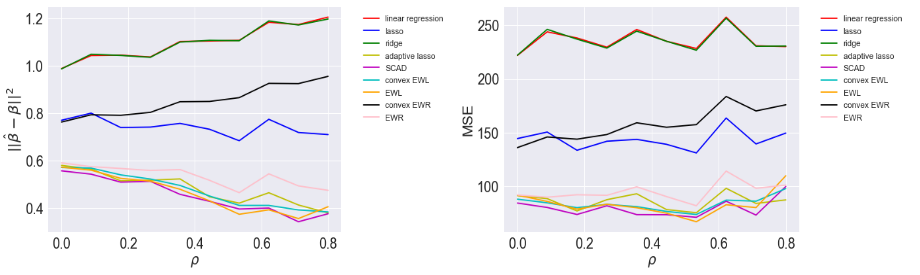

Figure 4: The L_2 distance between the estimated parameters $\hat{\beta}$ and the true parameters β^* (**left**) and the mean square error of predictions on the test set (**right**) as functions of correlation between covariates as ρ varies between 0 and 0.8. The results displayed are averages over 100 runs. The solid lines correspond to the mean distance and the dashed lines correspond to the 95% confidence intervals.

8 Discussion and open problems

We have introduced a novel method of reducing the bias of an estimators based on regularisation penalty on the parameter estimates and studied its properties theoretically and experimentally, mainly when used together with an L1 or an L2 penalty. This gave rise to the EWL and EWR losses, which under some conditions have been shown to be convex and the corresponding estimators have been shown to be consistent. Under other assumptions, EWL has also been shown to be sign consistent, making it an asymptotically consistent variable selector.

Comparing to adaptive lasso, EWL can be optimized without any pre-estimation of parameters, making EWL more efficient when trained on large data-sets and many parameters and also independent of the quality of the initial estimates. This gives EWL an edge. Experiments suggest that the two methods are equally robust with respect to hyperparameter settings and perform equally well with respect to bias and prediction. However as mentioned in [Zou, 2006] efficient algorithms such as LARS can be used to optimize the λ parameter in adaptive lasso.

In comparison with SCAD, EWL no longer has an edge in not having to pre-estimate the parameters, since this goes for SCAD too. In terms of bias and prediction, SCAD performs on par with EWL and adaptive lasso. However, experiments suggest that SCAD is more sensitive to the choice of hyperparameters for sign-consistency, giving EWL an advantage provided that this holds in general.

EWR does on average not seem to perform as well as the other three. However, to which extent this is true seems to depend heavily on if hyperparameters are allowed to be chosen such that the EWR loss becomes non-convex or not. EWL is not nearly as reliant on this and the experiments do not suggest that we should make any definite conclusion on whether or not the optimal hyperparameters are within the convex regimen or not. Regarding the non-convex settings, it should be noted that the experiments may neither have found the optimal hyperparameters nor the optimal $\tilde{\beta}$ for the given hyperparameters. Thus EWL and EWR are potentially better than what we have found here, given an efficient method for non-convex optimisation.

Open problems for future research are e.g.

- Is EWL optimal for hyperparameters that make the loss convex or not? When the EWL loss is not convex, can we find the optimum efficiently? The latter question also applies to EWR.
- How do EWL and EWR perform on more complex models, such as deep neural nets? How do they work for classification rather than regression, or a combination of the two?
- The regularisation function on the weights u_j, i.e. $x \to x\log(x) - x$, is natural and also convenient as it allows for a closed expression of u_j in terms of β_j. However strictly speaking, it is rather arbitrary. Are there other functions that perform better?

References

[Bellec and Zhang, 2019] Bellec, P. C. and Zhang, C.-H. (2019). De-biasing the lasso with degrees-of-freedom adjustment. *arXiv preprint arXiv:1902.08885.*

[Bergersen et al., 2011] Bergersen, L. C., Glad, I. K., and Lyng, H. (2011). Weighted lasso with data integration. *Statistical applications in genetics and molecular biology*, 10(1).

[Fan and Li, 2001] Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

[Frank and Friedman, 1993] Frank, L. E. and Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135.

[Huang et al., 2008] Huang, J., Horowitz, J. L., and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2):587–613.

[Javanmard et al., 2018] Javanmard, A., Montanari, A., et al. (2018). Debiasing the lasso: Optimal sample size for gaussian designs. *Annals of Statistics*, 46(6A):2593–2622.

[Jung, 2011] Jung, K. M. (2011). Weighted least absolute deviation lasso estimator. *Communications for Statistical Applications and Methods*, 18(6):733–739.

[Knight and Fu, 2000] Knight, K. and Fu, W. (2000). Asymptotics for lasso-type estimators. *Annals of statistics*, pages 1356–1378.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression selection and shrinkage via the lasso. *Journal of the Royal Statistical Society Series B*, 58(1):267–288.

[Zhang, 2011] Zhang, C.-H. (2011). Statistical inference for high-dimensional data. *Mathematisches Forschungsinstitut Oberwolfach: Very High Dimensional Semiparametric Models, Report*, 48:28–31.

[Zhao and Yu, 2006] Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563.

[Zou, 2006] Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

9 Appendix

Plots of model hyperparameters as functions of experimental settings

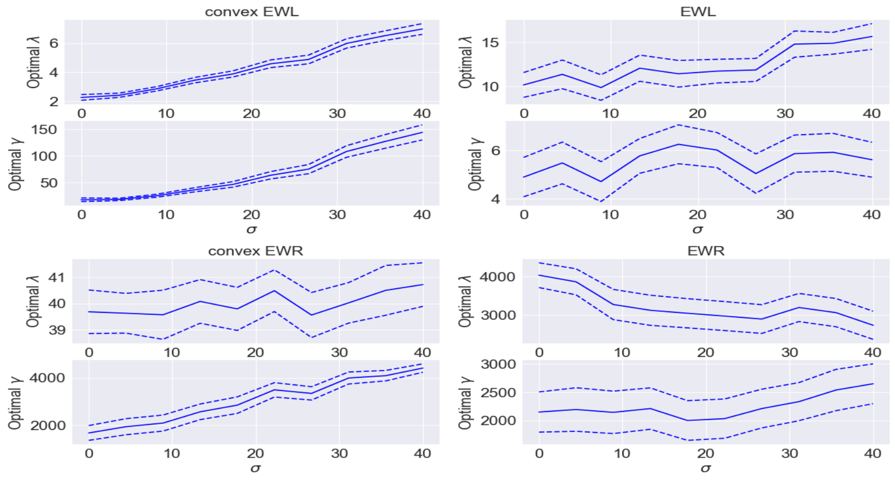

Figure 5: The average optimal hyper-parameters with respect to MSE for EWL and EWR, both in the convex and in the non-convex setting with uncorrelated covariates as functions of σ. The dashed lines correspond to the estimated 95% confidence intervals based on 100 sample points.

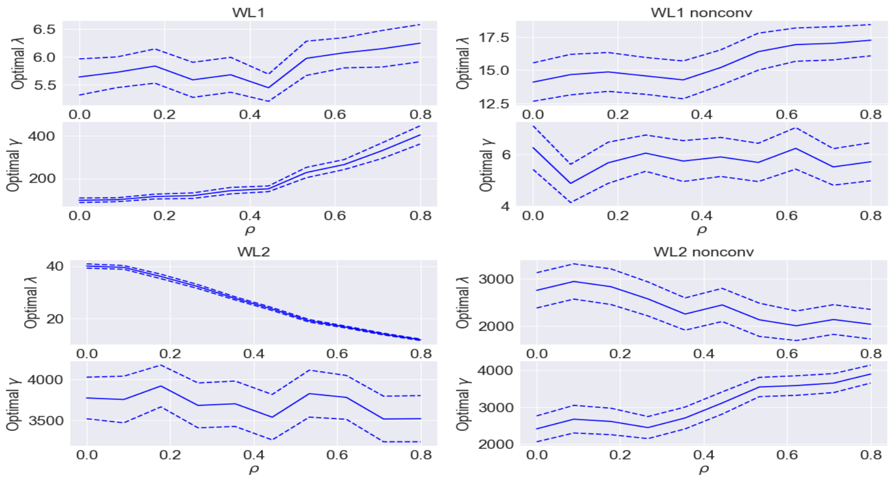

Figure 6: The average optimal parameters for EWL and EWR for correlated covariates as functions of ρ. The dashed lines correspond to the estimated 95% confidence intervals based on the 100 sampled points.

Tables

	$\sigma = 17.78,\ \rho = 0$	$\sigma = 35.56,\ \rho = 0$	$\sigma = 30,\ \rho = 0.36$	$\sigma = 30,\ \rho = 0.71$
Linear regression	246.12 ± 5.29	230.94 ± 20.4	246.12 ± 18.38	230.94 ± 15.45
Lasso	143.65 ± 4.54	139.37 ± 18.95	143.65 ± 16.29	139.37 ± 12.36
Ridge	244.6 ± 5.2	230.46 ± 20.7	244.6 ± 17.88	230.46 ± 15.41
Adaptive lasso	92.93 ± 4.93	83.89 ± 15.21	92.93 ± 19.14	83.89 ± 11.88
SCAD	73.49 ± 2.72	73.06 ± 16.38	73.49 ± 10.41	73.06 ± 10.13
EWL convex	80.91 ± 2.75	85.98 ± 14.46	80.91 ± 10.02	85.98 ± 10.58
EWL	79.9 ± 2.94	80.02 ± 14.16	79.9 ± 12.64	80.02 ± 16.32
EWR convex	159.13 ± 3.18	170.15 ± 14.1	159.13 ± 11.42	170.15 ± 11.75
EWR	99.41 ± 2.86	98.02 ± 14.73	99.41 ± 15.89	98.02 ± 14.15

Table 1: 95% confidence intervals of the mean square error for each method on four different datasets.

	$\sigma = 17.78,\ \rho = 0$	$\sigma = 35.56,\ \rho = 0$	$\sigma = 30,\ \rho = 0.36$	$\sigma = 30,\ \rho = 0.71$
Linear regression	0.6 ± 0.02	1.2 ± 0.04	1.1 ± 0.04	1.18 ± 0.05
Lasso	0.44 ± 0.02	0.91 ± 0.05	0.76 ± 0.04	0.72 ± 0.05
Ridge	0.6 ± 0.02	1.19 ± 0.04	1.1 ± 0.04	1.17 ± 0.05
Adaptive lasso	0.33 ± 0.03	0.65 ± 0.05	0.52 ± 0.06	0.41 ± 0.05
SCAD	0.3 ± 0.02	0.65 ± 0.05	0.46 ± 0.04	0.34 ± 0.02
EWL convex	0.3 ± 0.02	0.67 ± 0.05	0.5 ± 0.04	0.39 ± 0.03
EWL	0.3 ± 0.02	0.65 ± 0.05	0.48 ± 0.05	0.36 ± 0.04
EWR convex	0.45 ± 0.01	0.92 ± 0.03	0.85 ± 0.03	0.93 ± 0.03
EWR	0.31 ± 0.02	0.68 ± 0.05	0.56 ± 0.05	0.49 ± 0.06

Table 2: 95% confidence intervals of the L2 distance between true and estimated parameters on four different datasets.

Proofs

Theorem 1. *Let $(\tilde{\beta}, \tilde{\mathbf{u}})$ be as in (3). Then*

$$\tilde{\beta} = \arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \gamma \sum_{i=1}^{p} (1 - e^{-\frac{\lambda}{\gamma} f(\beta_i)}) \tag{11}$$

Proof. By optimizing over $\mathbf{u}$ in equation (3) we need to solve

$$\frac{\partial}{\partial u_j} \frac{1}{2}||Y - X\beta||_2^2 + \sum_i [\lambda u_i f(\beta_i) + \gamma(u_i \log(u_i) - u_i + 1)] = 0$$

for all j, which is equivalent to

$$\lambda f(\beta_j) + \gamma \log(u_j) = 0.$$

Solving for u_j gives us

$$u_j = e^{-\frac{\lambda f(\beta_j)}{\gamma}}.$$

Inserting this result into (3) we end up with

$$\arg\min_{\beta, \mathbf{u}} \frac{1}{2}||Y - X\beta||_2^2 + \sum_{i=1}^{p} [\lambda u_i f(\beta_i) + \gamma(u_i \log(u_i) - u_i + 1)] =$$

$$\arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \sum_{i=1}^{p} \left[\lambda e^{-\frac{\lambda f(\beta_i)}{\gamma}} f(\beta_i) + \gamma(e^{-\frac{\lambda f(\beta_i)}{\gamma}} \log(e^{-\frac{\lambda f(\beta_i)}{\gamma}}) - e^{-\frac{\lambda f(\beta_i)}{\gamma}} + 1)\right] =$$

$$\arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \gamma \sum_{i=1}^{p} (1 - e^{-\frac{\lambda}{\gamma} f(\beta_i)})$$

This finishes the proof.

□

Theorem 2. *Let s_1^2 be the smallest eigenvalue of $X^T X$. The minimisation problem (5) is convex whenever $\gamma > \frac{\lambda^2}{s_1^2}$.*

Proof. Let L be the minimisation objective in (5):

$$L(\beta) = \frac{1}{2}||Y - X\beta||_2^2 + \gamma \sum_{i=1}^{p} (1 - e^{-\lambda\gamma|\beta_i|}).$$

Consider the gradient and the Hessian at the points $\beta_j \neq 0 \, \forall j$. They are

$$\frac{\partial L(\beta)}{\partial \beta} = -X^T(Y - X\beta) + \lambda \begin{bmatrix} sign(\beta_1)e^{-\frac{\lambda}{\gamma}|\beta_1|} \\ \vdots \\ sign(\beta_n)e^{-\frac{\lambda}{\gamma}|\beta_n|} \end{bmatrix}$$

$$\frac{\partial^2 L(\beta)}{\partial \beta^2} = X^T X - diag\left(\left[\frac{\lambda^2}{\gamma}e^{-\frac{\lambda}{\gamma}|\beta_i|}\right]\right) \tag{12}$$

We want to determine for what values of λ and γ the Hessian is positive definite for each value of β with nonzero coefficients. Since

$$b^T \frac{\partial^2 L(\beta)}{\partial \beta^2} b \geq b^T \left(X^T X - \frac{\lambda^2}{\gamma}\mathbf{I}\right) b \geq s_1^2 - \frac{\lambda^2}{\gamma}$$

for all $b \in \mathbb{R}^p$, it follows that the Hessian is positive definite whenever $s_1^2 > \lambda^2/\gamma$. Hence L is convex within each (closed) orthant.

It remains to check that L is convex on the union of the 2^p orthants. To this end, fix an arbitrary $\epsilon > 0$. For a subset P of $\{1, \dots, p\}$, let

$$O_P = [0, \infty)^P \times (-\infty, 0]^{P^c}.$$

and

$$O_P^+ = [-\epsilon, \infty)^P \times (-\infty, \epsilon]^{P^c}.$$

Define

$$L_P(\beta) = ||Y - X\beta||_2^2 + \gamma(\sum_{j \in P} e^{-\lambda\beta_j/\gamma} + \sum_{j \in P^c} e^{\lambda\beta_j/\gamma}), \, \beta \in O_P^+.$$

Observe that $L = L_P$ on O_P for each P, but that L_P is defined on a slightly larger domain than O_P. In analogy with the above, L_P is convex on its domain whenever $s_1^2 > \frac{\lambda^2}{\gamma}e^{\frac{\lambda}{\gamma}\epsilon}$.

Consider now two adjacent orthants P_1 and P_2 (i.e. such that points in the interior of O_{P_1} and points in the interior of O_{P_2} have the same sign for each coordinate but one). It is easy to see that $L_{P_1} \geq L_{P_2}$ on $O^{P_1} \cap O_{P_2}^+$ The restriction, L_U, of L to $O_{P_1} \cup O_{P_2}$ can thus be written as

$$L_U(\beta) = \begin{cases} L_{P_1}(\beta) & \beta \in O_{P_1} \setminus O_{P_2} \\ \max(L_{P_1}(\beta), L_{P_2}(\beta)) & \beta \in O_{P_1} \cap O_{P_2} \\ L_{P_2}(\beta) & \beta \in O_{P_2} \setminus O_{P_1} \end{cases}$$

Since $O_{P_1}^+ \cap O_{P_2}^+$ is convex and the restriction, L_C, of L there is the maximum of two convex functions, L_C is convex. Now we have that the restriction of L_{P_1} to O_{P_1} and L_C agree on the intersection of their domains (which are O_{P_1} and $O_{P_1}^+ \cap O_{P_2}^+$ respectively). We also have that each line with one end point in $O_{P_1} \setminus O_{P_2}^+$ and the other end point in $O_{P_2} \setminus O_{P_1}$ passes through an open interval of points in the intersection $O_{P_1} \cap O_{P_2}^+$ of the two domains. From that observation, it is an easy exercise in convexity to conclude that the restriction of L_U to $O_{P_1}^+$ (i.e. the union of the two domains under discussion) is convex. By the same reasoning, it then follows that L_U is convex.

Now pair off the 2^p orthants into 2^{p-1} pairs of adjacent orthants. Then the restriction of L to each of these pairs is convex. Next pair off these pairs in turn into 2^{p-2} adjacent pairs. It then follows that

L restricted to each of these quadruples of orthants is convex. Now iterate this until all orthants have been united.

Since ϵ is arbitrary, this concludes the proof.

□

Theorem 3. *Assume* $\frac{\gamma_n}{n} \to \gamma_0 \geq 0$, $\frac{\lambda_n}{n} \to \lambda_0 \geq 0$, $\lim_{n\to\infty} \frac{X^TX}{n} = C$ *is nonsingular and that* f *is a convex function. Let*

$$\tilde{\beta} = \arg\min_{\beta} \frac{1}{2}||Y - X\beta||_2^2 + \gamma_n \sum_{j=1}^{p}(1 - e^{-\frac{\lambda_n}{\gamma_n}f(\beta_j)}) \tag{13}$$

Then $\tilde{\beta} \to_p \arg\min(Z)$ *where*

$$Z(\beta) = \frac{1}{2}(\beta - \beta^*)C(\beta - \beta^*) + \gamma_0 \sum_{j=1}^{p}(1 - e^{-\frac{\lambda_0}{\gamma_0}f(\beta_j)}),$$

where we define the second term to be zero if λ_0 *or* γ_0 *is zero. Hence if* $\gamma_n = o(n)$ *or* $\lambda_n = o(n)$*, then* $\tilde{\beta}$ *is consistent.*

Proof. We start by using $Y = X\beta^* + \xi$ and setting Z_n to

$$Z_n(\beta) = \frac{1}{2n}||Y - X\beta||^2 + \frac{\gamma_n}{n}\sum_{i=1}^{p}(1 - e^{-\frac{\lambda_n}{\gamma_n}f(\beta_i)}) = \frac{1}{2n}||X\beta + \xi - X\beta||^2 + \frac{\gamma_n}{n}\sum_{i=1}^{p}(1 - e^{-\frac{\lambda_n}{\gamma_n}f(\beta_i)})$$

$$= \left((\beta^* - \beta)^T\frac{X^TX}{2n}(\beta^* - \beta) + \frac{\xi^TX(\beta^* - \beta)}{n} + \frac{\xi^T\xi}{2n}\right) + \frac{\gamma_n}{n}\sum_{i=1}^{p}(1 - e^{-\frac{\lambda_n}{\gamma_n}f(\beta_i)}).$$

Note that $\tilde{\beta} = \arg\min_\beta Z_n(\beta)$. In order to prove the result, it suffices to show that

$$\sup_{\beta\in K}|Z_n(\beta) - Z(\beta) - \frac{\sigma^2}{2}| \to_p 0. \tag{14}$$

for any compact set $K \subset R^p$ and that

$$\tilde{\beta} = O_p(1). \tag{15}$$

Inserting Z_n into (14) gives

$$\sup_{\beta\in K}|Z_n(\beta) - Z(\beta) - \sigma^2| =$$

$$\sup_{\beta}\left[\frac{1}{2}(\beta^* - \beta)^T\left(\frac{X^TX}{n} - C\right)(\beta^* - \beta) + \frac{\xi^TX(\beta^* - \beta)}{n} + \frac{\xi^T\xi}{2n} - \frac{\sigma^2}{2}\right.$$
$$\left. + \frac{\gamma_n}{n}\sum_{i=1}^{p}(1 - e^{-\frac{\lambda_n}{\gamma_n}f(\beta_i)}) - \gamma_0\sum_{i=1}^{p}(1 - e^{-\frac{\lambda_0}{\gamma_0}f(\beta_i)})\right]$$

Since $\frac{X^TX}{n} \to C$, $\frac{\epsilon^T\epsilon}{n} \to \sigma^2$, $\frac{\lambda_n}{\gamma_n} \to \frac{\lambda_0}{\gamma_0}$ the first four terms vanish. For (14), it remains to consider the last two terms and show that the left term converges to the right term. When $\gamma_0 > 0$ the convergence is trivial. When $\gamma_0 = 0$ we use the fact that

$$0 \leq \frac{\gamma_n}{n}\sum_{i=1}^{p}(1 - e^{-\frac{\lambda_n}{\gamma_n}f(\beta_i)}) < \frac{\gamma_n}{n}p \to 0$$

This proves that (14) holds. Next we show $\tilde{\beta} = O_p(1)$. Observe first that for the OLS estimate:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2n}||Y - X\beta||_2^2 = O_p(1).$$

If we can show that there exists a δ_0 s.t. $\tilde{\beta}$ must be in the ball

$$B = \{\hat{\beta} + \delta\nu \ : ||\nu||_2 = 1, \ |\delta| \leq \delta_0, \}$$

the proof will be concluded. We will show this by finding a δ_0 s.t.

$$Z_n(\hat{\beta} + \delta\nu) - Z_n(\hat{\beta}) \geq 0 \tag{16}$$

whenever $\delta \geq \delta_0$, implying that Z_n can have no minimum outside of B. Using the definition of Z_n and simplifying gives us that (16) is equivalent to

$$\frac{\delta^2}{2n}\nu X^T X\nu + \frac{\xi^T X\delta\nu}{n} - (\beta^* - \hat{\beta})\frac{X^T X}{n}\delta\nu + \frac{\gamma_n}{n}\sum_{i=1}^{p}(e^{-\frac{\lambda_n}{\gamma_n}f(\hat{\beta}_i)} - e^{-\frac{\lambda_n}{\gamma_n}f(\hat{\beta}+\delta\nu)_i}) \geq 0$$

for every unit vector ν. Lower bounds of each term on the left hand side are

$$\frac{\delta^2}{2n}\nu X^T X\nu \geq \frac{\delta^2 s_1^2}{2}$$
$$\frac{\xi^T X\delta\nu}{n} \geq -\frac{||\xi||_2\delta s_p}{\sqrt{n}}$$
$$-(\beta^* - \hat{\beta})\frac{X^T X}{n}\delta\nu \geq -||\beta^* - \hat{\beta}||_2\delta s_p^2$$
$$\frac{\gamma_n}{n}\sum_{i=1}^{p}(e^{-\frac{\lambda_n}{\gamma_n}f(\hat{\beta}_i)} - e^{-\frac{\lambda_n}{\gamma_n}f(\hat{\beta}+\delta\nu)_i}) \geq \frac{\gamma_n}{n}\sum_{i=1}^{p}e^{-\frac{\lambda_n}{\gamma_n}f(\hat{\beta}_i)} - \frac{\gamma_n}{n}p.$$

where s_1^2 and s_p^2 are the smallest and largest eigenvalues of $\frac{X^T X}{n}$ respectively. Observe that $\frac{E[||\xi||_2]}{\sqrt{n}} = O_p(1)$ making the term bounded. Putting it all together gives us that it is suffice to find a δ s.t.

$$\frac{\delta^2 s_1^2}{2} - \left(\frac{||\xi||_2 s_p}{\sqrt{n}} + ||\beta^* - \hat{\beta}||_2 s_p^2\right)\delta + \frac{\gamma_n}{n}\sum_{i=1}^{p}e^{-\frac{\lambda_n}{\gamma_n}f(\hat{\beta}_i)} - \frac{\gamma_n}{n}p \geq 0.$$

Since this is a quadratic equation in δ, $s_1 \neq 0$ and each term is bounded in probability with respect to n, there exists a $\delta_0 = O_p(1)$ s.t. (16) holds for all $|\delta| \geq \delta_0$.

This shows that $\tilde{\beta}$ must be in B since for values outside B we have $Z_n(\hat{\beta} + \delta\nu) \geq Z_n(\hat{\beta})$. This completes the proof.

□

Theorem 4. *Assume that* $||C_{11}^{-1}||_\infty < \frac{K_1}{n}$ *and* $||C_{21}||_\infty < K_2 n$ *for some constants* $K_1, K_2 < \infty$ *independent of n, where* $||\cdot||_\infty$ *is the* ∞*-operator norm. Assume also that there is a constant* $\delta > 0$ *such that for all n,* $n > \lambda_n > n^{1/2+2\delta}$ *and* $\lambda_n|\beta_{nj}|/\gamma_n > n^{2\delta}$ *and* $|\beta_{nj}| > n^{-1/2+2\delta}$ *for all* $j = 1, \ldots, r_n$*. Assume in addition that* $\gamma \geq 1$ *and* $q_n < e^{n^\delta}$*.*

Let L be the minimisation objective

$$L(\beta) = L_n(\beta; Y) = \frac{1}{2}||Y - X\beta)||_2^2 + \gamma_n\sum_{j=1}^{p_n}(1 - e^{-\frac{\lambda_n}{\gamma_n}|\beta_j|}).$$

Then with probability at least $1-e^{-n^\delta}$*, L has a local minimum $\bar\beta$ such that with probability* $1-e^{-n^\delta}$*,* $||\bar\beta-\beta^*||_\infty < n^{-1/2+\delta}$ *and* $sign(\bar\beta)=sign(\beta^*)$*. Hence if L has a unique minimum, then*

$$\tilde\beta_n = \arg\min_\beta \left[\frac{1}{2}||Y-X\beta)||_2^2 + \gamma_n \sum_{j=1}^{p_n}(1-e^{-\frac{\lambda_n}{\gamma_n}|\beta_j|})\right]$$

satisfies with probability at least $1-e^{-n^\delta}$ *that* $||\tilde\beta-\beta^*||_\infty < n^{-1/2+\delta}$ *and* $sign(\tilde\beta)=sign(\beta^*)$*.*

Proof. For convenience, we drop the subscript n from γ, λ, β_j, p, r and q.

The SSE-term in L can be written as

$$SSE(\beta) = \frac{1}{2}||(Y-X\beta)||_2^2 = \frac{1}{2}||\xi||_2^2 + \xi^T X(\beta^*-\beta) + \frac{1}{2}(\beta^*-\beta)^T X^T X(\beta^*-\beta)$$

where the second term can be written as $\sum_j \sqrt{n}(\beta_j-\beta_j^*)Z_j$ where $Z=(Z_1,...,Z_{r+q})^T \sim N(0,X^TX/n)$. Now write $\beta=(\phi^T,\psi^T)^T$ where $\phi=(\beta_1,...,\beta_r)^T$, $\psi=(\beta_{r+1},...,\beta_{r+q})^T$ and $Z=(Z_\phi^T,Z_\psi^T)^T$. Write as above

$$X^TX = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}.$$

We can now write L as

$$L(\beta) = \frac{1}{2}||\xi||_2^2 + \frac{1}{2}(\beta-\beta^*)X^TX(\beta-\beta^*) + \sum_j \left[\sqrt{n}(\beta_j-\beta_j^*)Z_j + \gamma(1-e^{-\frac{\lambda}{\gamma}|\beta_j|})\right]$$

Now set $\psi=\psi^*=0$. We claim that there is a $\bar\phi$ in the same orthant as ϕ^* such that $(\bar\phi^T,0^T)^T$ is a critical point of L. Proving this, we will prove that

$$\sqrt{n}Z_\phi + C_{11}(\bar\phi-\phi^*) + \lambda[e^{-\frac{\lambda_n}{\gamma_n}|\phi|}]_{j=1}^r = 0 \tag{17}$$

$$\forall j: -\lambda < \sqrt{n}Z_{\psi,j} + (C_{21}(\bar\phi-\phi^*))_j < \lambda \tag{18}$$

Here (17) is the condition that the gradient of L as seen as a function of ϕ at $(\bar\phi^T,0^T)$ vanishes and then (18) that the zero vector is a subgradient of L seen as a function of ψ at $(\bar\phi^T,0^T)$. To prove this, we will first show that a solution to (17) is obtained sufficiently close to ϕ^* so as to be in the same orthant as ϕ^* and then that this solution automatically also satisfies (18). To prove the former claim, we will show that Newton's method for solving (17) started from $\phi^{(0)}=\phi^*$ converges well within the same orthant as the starting point.

Since by assumption $\lambda|\phi_j|/\gamma > n^{2\delta}$, we have with $L_1(\phi)=L((\phi^T,0^T)^T)$,

$$\nabla L_1(\phi^*) = \sqrt{n}Z_\phi + \lambda[sign(\phi_n^*)e^{-\frac{\lambda}{\gamma}|\phi_n^*|}]_{j=1}^r = \sqrt{n}Z_\phi + o(e^{-n^\delta})$$

and

$$\nabla^2 L_1(\phi^*) = C_{11} - \frac{\lambda^2}{\gamma}diag\left(e^{-\frac{\lambda}{\gamma}|\phi_n^*|}\right) = C_{11} - o(e^{-n^\delta})\mathbf{I}.$$

The first update step with Newton's method gives

$$\phi^{(1)} - \phi^{(0)} = \left[C_{11} - o(e^{-n^\delta})\mathbf{I}\right]^{-1}\left[\sqrt{n}Z_\phi + o(e^{-n^\delta})\right] = \sqrt{n}C_{11}^{-1}Z_\phi + o(e^{-n^\delta})\mathbf{1}.$$

Note that with probability $1 - o(e^{-n^{2\delta}})$ we have for a given j that $|Z_j| < n^\delta$ and since $q < e^{n^\delta}$, this holds simultaneously for all j with probability $1-o(e^{-n^\delta})$. This gives us that $|\phi_i^{(1)} - \phi_i^{(0)}| < K_1 n^{-1/2+\delta}$ and $||\phi^{(1)} - \phi^*||_\infty < K_1 n^{-1/2+\delta}$. We now proceed to show that after this, Newton's method converges very fast, in particular fast enough to ensure that $||\bar{\phi} - \phi^{(0)}||_\infty < 2K_1 n^{-1/2+\delta}$.

Note that as long as

$$||\phi - \phi^*||_\infty < \frac{1}{2}n^{-1/2+2\delta} \tag{19}$$

we have

$$\frac{\lambda^3}{\gamma^2}e^{-\frac{\lambda}{\gamma}|\phi|} < m = o(e^{-n^\eta}).$$

A Taylor expansion of $\nabla L_1(\phi)_j$ gives

$$\nabla L_1(\phi)_j = \nabla L(\phi^{(k-1)})_j + (C_{11}(\phi - \phi^{(k-1)}))_j + \frac{\lambda^3}{2\gamma^2}e^{-\frac{\lambda}{\gamma}|\phi^{(k-1)}|}(\zeta - \phi_j^{(k-1)})^2$$

where ζ is between ϕ_j and $\phi_j^{(k-1)}$. Since Newton's method by its nature chooses $\phi^{(k)}$ so as to make the sum of the first two terms on the right hand side sum to 0, it follows that

$$|\nabla L_1(\phi^{(k)})_j| < m|\phi_j^{(k)} - \phi_j^{(k-1)}|^2,$$

i.e.

$$||\nabla L_1(\phi^{(k)})||_\infty < m||\phi^{(k)} - \phi^{(k-1)}||_\infty^2. \tag{20}$$

Taking the $k+1$'th Newton step and provided that $||\phi^{(k)} - \phi^{(0)}||_\infty < 2K_1 n^{-1/2+\delta}$, we get

$$\phi^{(k+1)} - \phi^{(k)} = \left(C_{11} + o(e^{-n^\eta})\right)^{-1}\nabla L_1(\phi^k),$$

which implies

$$||\phi^{(k+1)} - \phi^{(k)}||_\infty \leq \frac{K_1}{n}||\nabla L_1(\phi^{(k)}))||_\infty. \tag{21}$$

Taken together with (20), (21) gives

$$||\phi^{(k+1)} - \phi^{(k)}||_\infty \leq \frac{K_1 m}{n}||\phi^{(k)} - \phi^{(k-1)}||_\infty^2$$

Now since $||\phi^{(1)} - \phi_n^{(0)}||_\infty < K_1 n^{-1/2+\epsilon}$, $\{\phi^{(k)}\}$ converges well within the marginal of (19). Taking $\tilde{\phi} = \lim_k \phi^{(k)}$, it follows that $\bar{\phi}$ solves (17) and satisfies

$$||\bar{\phi} - \phi^*||_\infty < \frac{2K_1 m}{n}||\phi^{(k)} - \phi^{(k-1)}||_\infty.$$

In particular $\bar{\phi}_j$ and ϕ^*_j have the same sign for all j with probability tending to 1 as n grows.

Since $||\bar{\phi} - \phi^*_n||_\infty < 2K_1 n^{-1/2+\epsilon}$ it also follows that with probability tending to 1,

$$||\sqrt{n}Z_\psi + C_{21}(\bar{\phi} - \phi^*_n)||_\infty < n^{1/2+\delta} + 2K_1K_2n^{1/2+\delta} < \lambda.$$

In other words, $(\tilde{\phi}^T, 0^T)^T$ satisfies (18).

Finally it needs to be shown that the given critical point is a local minimum. It then suffices to show that the function $L_1(\phi)$ is convex in a neighborhood of $\bar{\phi}$. To this end, recall from above that $e^{-\lambda|\bar{\phi}_j|/\gamma} < e^{n^\delta}$. Then exactly as in the proof of Theorem 2, it follows that the $\nabla^2 L_1(\phi)$ is positive definite in a neighborhood of $\bar{\phi}$ whenever $\gamma > \lambda^2 e^{-n^\delta}/\ell$, where ℓ is the smallest eigenvalue of C_{11}, which is true by assumption as $\gamma > \lambda/n$ and ℓ is of order n.

□

Theorem 5. *The minimisation problem*

$$\tilde{\beta} = \min_\beta L(\beta) = \min_\beta \frac{1}{2}(Y - X\beta)^2 + \gamma \sum_i (1 - e^{-\frac{\lambda}{\gamma}\beta_i^2}). \tag{22}$$

is convex if $\lambda < \frac{s_1^2 e^{\frac{3}{2}}}{4}$.

Proof. We start by calculate the Hessian with respect to β

$$\frac{\partial^2 L(\beta)}{\partial^2 \beta} = X^T X + diag\left[2\lambda e^{-\frac{\lambda}{\gamma}\beta_i^2} - 4\beta_i^2 \frac{\lambda^2}{\gamma} e^{-\frac{\lambda}{\gamma}\beta_i^2}\right] \tag{23}$$

In order for L to be convex we need the Hessian to be positive definite for all β. The eigenvalues of the right hand side can be no smaller than when the function

$$h(\beta_i) = 2\lambda e^{-\frac{\lambda}{\gamma}\beta_i^2} - 4\beta_i^2 \frac{\lambda^2}{\gamma} e^{-\frac{\lambda}{\gamma}\beta_i^2}$$

is minimised. We minimise by taking the derivative and set to zero.

$$\frac{\partial h}{\partial \beta_i} = -4\frac{\beta_i\lambda^2}{\gamma} e^{-\frac{\lambda}{\gamma}\beta_i^2} - 8\frac{\beta_i\lambda^2}{\gamma} e^{-\frac{\lambda}{\gamma}\beta_i^2} + 8\frac{\beta_i^3\lambda^3}{\gamma^2} e^{-\frac{\lambda}{\gamma}\beta_i^2} = -4\frac{\beta_i\lambda^2}{\gamma} e^{-\frac{\lambda}{\gamma}\beta_i^2}\left[3 - 2\beta_i^2\frac{\lambda}{\gamma}\right] = 0$$

This gives us the solutions $\beta_i = 0, \beta_i = \pm\sqrt{\frac{3\lambda}{2\gamma}}$ where $\beta = 0$ corresponds to a maximum and $\beta_i = \pm\sqrt{\frac{3\lambda}{2\gamma}}$ to a minimum. Substituting the minimum points into equation (23) for each β_i we end up with

$$\frac{\partial^2 L(\beta)}{\partial^2 \beta} = X^T X - 4\lambda e^{-\frac{3}{2}}\mathbf{I}.$$

The eigenvalues of this matrix are $\left\{(s_j^2 - 4\lambda e^{-\frac{3}{2}})\right\}$ where s_j^2 are the eigenvalues of $X^T X$. Now solving for λ we need that $\lambda < \frac{s_1^2 e^{\frac{3}{2}}}{4}$ where s_1^2 is the smallest eigenvalue of $X^T X$.

□