Learning ℓ^1-Penalized Logistic Regressions with Smooth Approximation

Jacek Klimaszewski, Michał Sklyar, Marcin Korzeń
West Pomeranian University of Technology in Szczecin
Faculty of Computer Science and Information Technology
Żołnierska 49 street, 71-210 Szczecin
Email: jklimaszewski@wi.zut.edu.pl

***Abstract*—The paper presents comparison of learning logistic regression model with different penalty terms. Main part of the paper concerns sparse regression, which includes absolute value function. This function is not strictly convex, thus common optimizers cannot be used directly. In the paper we show that in those cases smooth approximation of absolute value function can be effectively used either in the case of lasso regression or in fussed-lasso like case. One of examples focuses on two dimensional analogue of fussed-lasso model. The experimental results present the comparison of our implementations (in C++ and Python) on three benchmark datasets.**

***Index Terms*—classification, logistic regression, regularization.**

I. Introduction

We consider a supervised learning task with following notations: $\mathbf{X}_{n\times d}$ is an input part of the dataset and $\mathbf{y}_{n\times 1}$ is the binary decision with values $\{-1,+1\}$, n stands for number of observations and d is number of attributes. The goal is to build such a model, that for unseen examples it would predict correct answers. We consider the only one model i.e. logistic regression, and its parameters are denoted as $\mathbf{w}$. In fact the logistic regression (LR) is a group of models, which have common loss function i.e. negative logarithm of likelihood function $l(\mathbf{w};\mathbf{X},\mathbf{y})$ and differs with the regularization term $R(\mathbf{w})$. The goal of learning is to minimize the cost function that consists of loss function and penalty function in the following:

$$Q(\mathbf{w}) = l(\mathbf{w};\mathbf{X},\mathbf{y}) + R(\mathbf{w}), \tag{1}$$

where ($\langle\cdot,\cdot\rangle$ denotes dot product):

$$l(\mathbf{w};\mathbf{X},\mathbf{y}) = \sum_{i=1}^{n} \log(1+\exp(-y_i\cdot\langle\mathbf{x}_i,\mathbf{w}\rangle)), \tag{2}$$

and $R(\mathbf{w})$ is usually referred to as a penalty term. There are many penalty terms known, some are presented in the Table I. Strength of regularization is controlled via λ parameter, multiplied by n.

In the case when penalty term is smooth and convex, usually standard methods, such as IRLS, conjugate gradient etc. can be used [4], [5]. In case of absolute value function, one loses smoothness and standard derivative-based methods cannot be used directly. However there are many ways to optimize ℓ^1-regularized logistic regression known in the literature, such as IRLS-LARS [6], Coordinate Descent [7], Pathwise Coordinate Descent [8], Interior-Point method [9].

TABLE I
Commonly used penalty functions.

Name	Formula
Ridge	$\lambda\cdot\|\mathbf{w}\|_2^2$
Lasso	$\lambda\cdot\|\mathbf{w}\|_1$
Elastic-net [1]	$\lambda\cdot\left(\alpha\|\mathbf{w}\|_2^2+(1-\alpha)\|\mathbf{w}\|_1\right)$
Total Variation (TV) [2]	$\lambda\cdot\sum_{j=2}^{d}\|w_j-w_{j-1}\|$
Fused-lasso (FL) [3]	$\lambda_1\cdot\|\mathbf{w}\|_1+\lambda_2\cdot\sum_{j=2}^{d}\|w_j-w_{j-1}\|$
Generalized FL	$\lambda_1\cdot\|\mathbf{w}\|_1+\lambda_2\|\mathbf{D}\mathbf{w}\|_1$

Our approach is to replace the absolute value function with a smooth function. We consider following smooth approximations of the absolute value function:

$$\text{soft\_abs}(x;a) = \frac{1}{a}\log\left(\cosh\left(ax\right)\right), \tag{3}$$

and respectively Huber function [10] defined by

$$\text{Huber}(x;a) = \begin{cases} \frac{x^2}{2a}, & \text{for } |x|\le a, \\ |x|-\frac{a}{2}, & \text{for } a<|x|, \end{cases} \tag{4}$$

for $a>0$.

To avoid numerical instability, we replace (3) with equivalent numerically stable version:

$$\text{soft\_abs}(x;a) = |x| + \frac{1}{a}\log\left(1+\exp\left(-2a|x|\right)\right) - \frac{1}{a}\log(2). \tag{5}$$

It is worth to notice, that (3) is twice continuously differentiable, which is not true for (4). Penalty functions with their derivatives are presented in the Fig. 1 and 2.

II. Algorithm

Solution for ℓ^1-regularized logistic regression, using substitution via smoothed absolute value is too straightforward. After replacing $|\cdot|$ with smooth convex function we obtain a common convex optimization problem that can be easily solved.

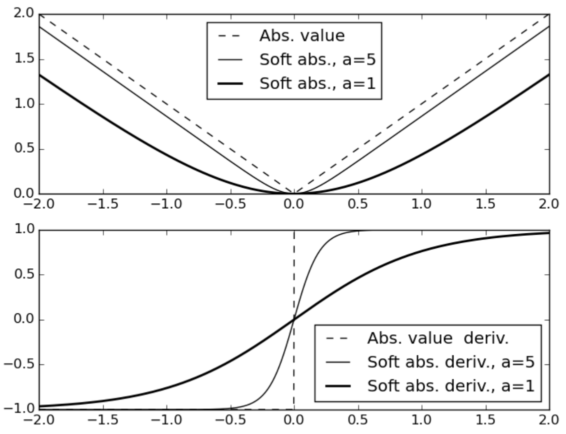

Fig. 1. The soft absolute value function and its derivative function compared to absolute value function respectively with its derivative.

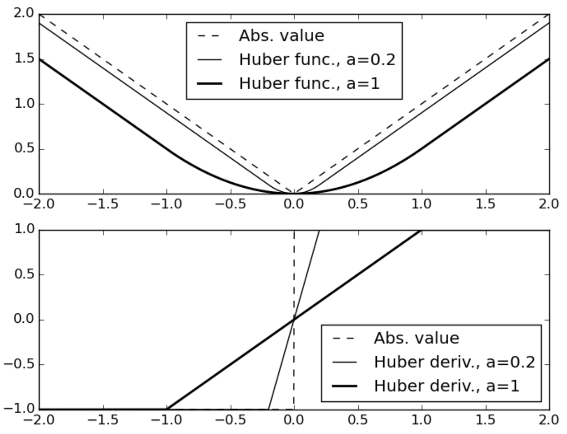

Fig. 2. The Huber function and its derivative function compared to absolute value function respectively with its derivative.

In case of fused-lasso the penalty term has the form $\|\mathbf{D}\mathbf{w}\|_1$, where $\mathbf{D}$ is some matrix. When we assume that input data are signals (1-D case), this imply that neighbouring variables are correlated and corresponding weights should be shrunk. The model will prefer such solutions when one uses the difference operator as the matrix $\mathbf{D}$, where matrix has -1 values on the main diagonal, 1 on the super-diagonal and 0 elsewhere. Also higher order differences can be computed using the same idea.

For 2-D data one has to vectorize it before calculation. Using row-major order it is possible to transform any matrix into vector — suppose that input images are matrices 3×3:

$$X^i = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}_{p \times q}. \tag{6}$$

After conversion to vector using row-major order we obtain:

$$X^i = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{bmatrix}^{\mathrm{T}}. \tag{7}$$

In this case we assume that neighbouring values (e.g. pixels in the image) are correlated vertically and horizontally, thus the matrix $\mathbf{D}$ consists of two parts: vertical and horizontal. For matrices 3×3 matrix $\mathbf{D}$ is as follows:

$$\mathbf{D} = \begin{bmatrix} H \\ V \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}_{p' \times q'}. \tag{8}$$

Let

$$\mathbf{z} = \mathbf{D} \cdot \mathbf{w}, \qquad z_j = \sum_{k=1}^{q'} D_{jk} \cdot w_k, \tag{9}$$

then penalty term (after substitution of "smooth" absolute value function) and its derivatives are:

$$R(\mathbf{w}) = \sum_{j=1}^{p'} \frac{1}{a} \log\left(\cosh\left(a z_j\right)\right), \tag{10}$$

$$\nabla R(\mathbf{w}) = \mathbf{D}^{\mathrm{T}} \cdot \begin{bmatrix} \tanh(a z_1) \\ \tanh(a z_2) \\ \vdots \\ \tanh(a z_{p'}) \end{bmatrix}, \tag{11}$$

$$\nabla^2 R(\mathbf{w}) = \mathbf{D}^{\mathrm{T}} \cdot \operatorname{diag}\left(\begin{bmatrix} a \cdot \left(1 - \tanh^2\left(a z_1\right)\right) \\ a \cdot \left(1 - \tanh^2\left(a z_2\right)\right) \\ \vdots \\ a \cdot \left(1 - \tanh^2\left(a z_{p'}\right)\right) \end{bmatrix} \right) \cdot \mathbf{D}. \tag{12}$$

Penalty term and the gradient in the case of Huber function are similar:

$$R(\mathbf{w}) = \sum_{j=1}^{p'} \mathrm{Huber}\left(z_j; a\right), \tag{13}$$

$$\nabla R(\mathbf{w}) = \mathbf{D}^{\mathrm{T}} \cdot \begin{bmatrix} \mathrm{diff\_Huber}(z_1; a) \\ \mathrm{diff\_Huber}(z_2; a) \\ \vdots \\ \mathrm{diff\_Huber}(z_{p'}; a) \end{bmatrix}, \tag{14}$$

where

$$\mathrm{diff\_Huber}(x; a) = \begin{cases} \frac{x}{a}, & \text{for } |x| \le a, \\ \mathrm{sign}(x), & \text{for } a < |x|. \end{cases} \tag{15}$$

TABLE II
ACCURACIES AND AREA UNDER ROC OF TESTED REGULARIZERS. IN CASE OF *Thrombin* DATASET, BALANCED ACCURACY IS PRESENTED.

	Regularizer	arcene		Thrombin		MicroMass	
		Acc	AUC	B_Acc	AUC	Acc	AUC
A	$\frac{\lambda}{2} \cdot \|\mathbf{w}\|_2^2$	0.84	0.9421	0.515	0.5624	0.979	0.9928
B	$\frac{\lambda}{2} \cdot \|\mathbf{Dw}\|_2^2$	0.82	0.9168	0.6206	0.6417	0.979	0.9952
C	$\frac{\lambda_1}{2} \cdot \|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2} \cdot \|\mathbf{Dw}\|_2^2$	0.82	0.9387	0.5127	0.5614	0.9825	0.9928
D	$\lambda \cdot \mathrm{Huber}(\mathbf{w})$	0.82	0.9395	0.515	0.5624	0.9965	0.9999
E	$\lambda \cdot \mathrm{soft\_abs}(\mathbf{w})$	0.82	0.9403	0.515	0.5624	0.9965	0.9999
F	$\lambda \cdot \mathrm{Huber}(\mathbf{Dw})$	0.83	0.9338	0.6164	0.6446	0.9755	0.9852
G	$\lambda \cdot \mathrm{soft\_abs}(\mathbf{Dw})$	0.83	0.933	0.6162	0.623	0.979	0.9977
H	$\lambda_1 \cdot \mathrm{Huber}(\mathbf{w}) + \lambda_2 \cdot \mathrm{Huber}(\mathbf{Dw})$	0.82	0.9355	0.5127	0.5614	0.972	0.9838
I	$\lambda_1 \cdot \mathrm{soft\_abs}(\mathbf{w}) + \lambda_2 \cdot \mathrm{soft\_abs}(\mathbf{Dw})$	0.82	0.9399	0.5127	0.5614	0.986	0.9982

Implementation Details: The implementation was performed in Python with some parts written in C++. Intercept w_0 was included in the weights $\mathbf{w}$ and was not regularized. To minimize cost function for soft_abs penalty function, Trust Region Newton method [11] was used and in the case of Huber function — the conjugate gradient. Testing procedures and all testing environment use `scikit-learn` library [12].

III. EXPERIMENTAL RESULTS

Datasets: We compare different regularization techniques on following datasets:

- **Thrombin** — binding to Thrombin dataset [13] was used in the 2001 KDD Cup data mining competition. It was produced by DuPont Pharmaceuticals Research Laboratories and concerns drug design. It has 2545 samples (1909 for training, 636 for testing) and about 130000 attributes.
- **MicroMass** — MicroMass spectrometry dataset [14] is a reference panel of 20 Gram positive and negative bacterial species. Each species was represented by 11 to 60 mass spectra obtained from 7 to 20 bacterial strains, constituting altogether a dataset of 571 spectra obtained from 213 strains. In this case as decision we use only distinction between Gram positive and Gram negative.
- **arcene** — arcene dataset [15] contains 200 samples (100 for training, 100 for testing) and 10000 attributes (it is known that 3000 of them is artificial noise)
- **MNIST** — MNIST dataset [16] is a large dataset of handwritten digits, containing 60 000 training and 10 000 testing images on matrices 28×28, which results in 784 attributes.
- **Art1000** — artificial dataset of size 1000×1000 generated as a random sparse array with the density factor set to 0.1 and decision defined by $d = \mathrm{sign}\left(X_{800} - \frac{1}{3}\sum_{i=50}^{55} X_i + \frac{1}{21}\sum_{i=500}^{550} X_i + \mathcal{N}(0, 0.05)\right)$

For experiments we choose following classifiers:

A logistic regression with ridge penalty: $\frac{\lambda}{2} \cdot \|\mathbf{w}\|_2^2$
B logistic regression with squared differences of weights: $\frac{\lambda}{2} \cdot \|\mathbf{Dw}\|_2^2$
C logistic regression with ridge penalty and squared differences of weights: $\frac{\lambda_1}{2} \cdot \|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2} \cdot \|\mathbf{Dw}\|_2^2$
D logistic regression with ℓ^1 penalty approximated by Huber function: $\lambda \cdot \mathrm{Huber}(\mathbf{w})$
E logistic regression with ℓ^1 penalty approximated by soft_abs function: $\lambda \cdot \mathrm{soft\_abs}(\mathbf{w})$
F logistic regression with total variation of weights approximated by Huber function: $\lambda \cdot \mathrm{Huber}(\mathbf{Dw})$
G logistic regression with ℓ^1 penalty and total variation of weights approximated by Huber function: $\lambda_1 \cdot \mathrm{Huber}(\mathbf{w}) + \lambda_2 \cdot \mathrm{Huber}(\mathbf{Dw})$
H logistic regression with total variation of weights approximated by soft_abs function: $\lambda \cdot \mathrm{soft\_abs}(\mathbf{Dw})$
I logistic regression with ℓ^1 penalty and total variation of weights approximated by soft_abs function: $\lambda_1 \cdot \mathrm{soft\_abs}(\mathbf{w}) + \lambda_2 \cdot \mathrm{soft\_abs}(\mathbf{Dw})$

In the first experiment we test our classifiers on three datasets: arcene, MNIST, Thrombin. In case of MNIST dataset, we did not fit intercept, because it was lowering accuracy. Results are presented in the Table II.

The second experiment concerns Art1000 and MNIST datasets. We look at accuracy according to a size of a training set. Testing procedure consists of two loops: outer loop iterates over training sets with increasing sizes, while inner loop iterates over considered classifiers. For any classifier tested in the inner loop we find optimal parameters using Grid Search method with 5 fold cross-validation procedure and then evaluate it on the test set. An important part of choosing a training data is that classes in our training set are equinumerous. The quality measure is the accuracy of classification (probability of a correct prediction). Because LR is a binary classifier, for MNIST dataset we transform it to multi-label classifier using one-vs-rest strategy [17]. In the Fig. 3 and 4 one can see the performance of different classifiers as a function of training sample's size. Moreover in the Fig. 5 there are images representing weights that each classifier generates for training data of size 3 000. One can notice that models D and E give sparse solution, and models F, G, I obtain smoothed weights.

Fig. 6 presents weights of classifiers for Art1000 data set. One can see that models trained with both components of penalty function (regularization of individual weights and its differences) better recognize true parameters than the other models.

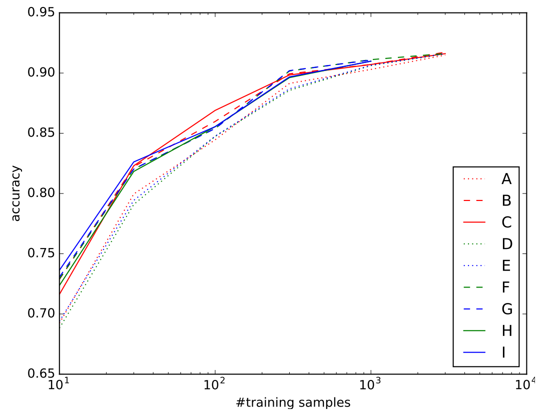

Fig. 3. Accuracy as a function of size of train set (number of examples from each class) in case of MNIST dataset.

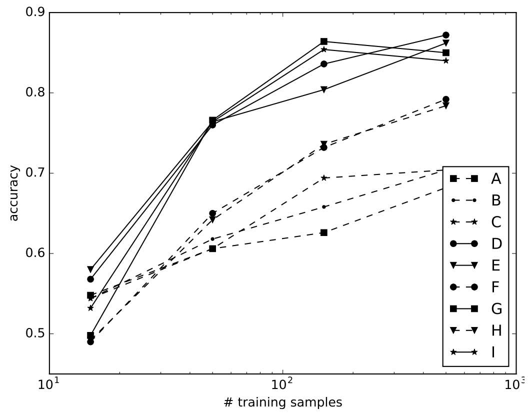

Fig. 4. Accuracy as a function of size of training set in case of Art1000 dataset.

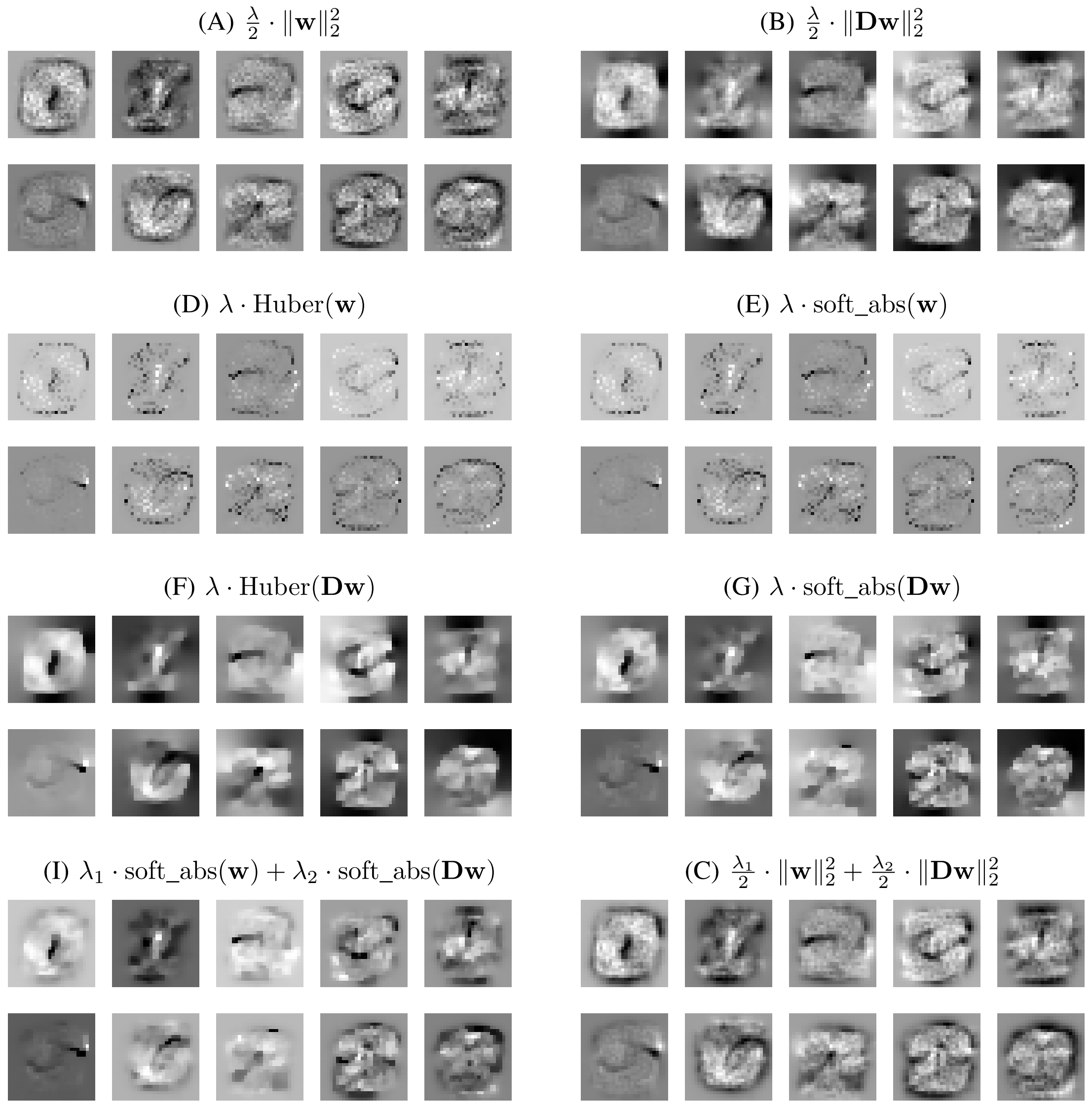
(A) $\frac{\lambda}{2} \cdot \|\mathbf{w}\|_2^2$
(B) $\frac{\lambda}{2} \cdot \|\mathbf{D}\mathbf{w}\|_2^2$
(D) $\lambda \cdot \mathrm{Huber}(\mathbf{w})$
(E) $\lambda \cdot \mathrm{soft\_abs}(\mathbf{w})$
(F) $\lambda \cdot \mathrm{Huber}(\mathbf{D}\mathbf{w})$
(G) $\lambda \cdot \mathrm{soft\_abs}(\mathbf{D}\mathbf{w})$
(I) $\lambda_1 \cdot \mathrm{soft\_abs}(\mathbf{w}) + \lambda_2 \cdot \mathrm{soft\_abs}(\mathbf{D}\mathbf{w})$
(C) $\frac{\lambda_1}{2} \cdot \|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2} \cdot \|\mathbf{D}\mathbf{w}\|_2^2$

Fig. 5. Depiction of models' parameters as a grey-scale images obtained for classifiers trained on MNIST dataset with a training data of size 30 000.

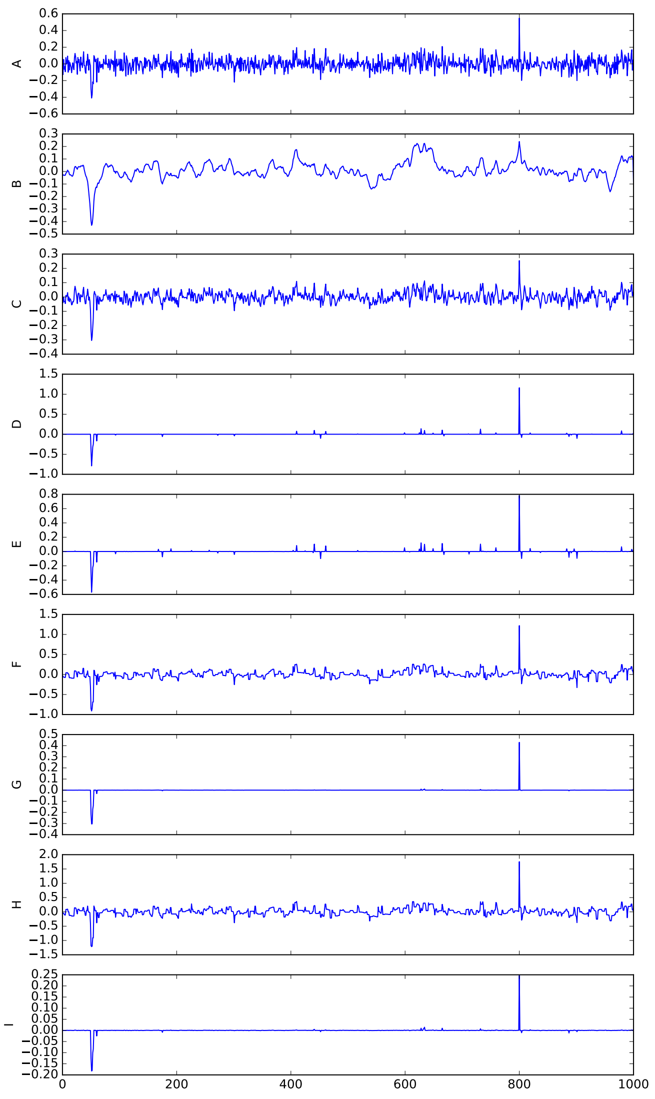

Fig. 6. Weights obtained for classifiers on Art1000 dataset after fitting with a training data of size 150. Horizontal axis corresponds to index of coefficient and vertical axis presents its value.

IV. Conclusion

In the paper we present approach to learning logistic regression model via smooth approximation of ℓ^1–norm. We compare two variants of approximation with a use of Huber and soft_abs functions. Overall conclusion is that the approach is effective, however difference in time of optimization between ℓ^2-regularized LR and smoothed version of ℓ^1 LR ($\mathrm{Huber}(\mathbf{w})$, $\mathrm{soft\_abs}(\mathbf{w})$) are quite similar. More precisely, time of a unique iterations is comparable, but the number of iterations depends on convergence, and for sure ℓ^2 regression converges faster. Comparison of time of computation of Huber function and soft_abs is more difficult. Huber function is simpler but it has not continuous second derivative, however in both cases Hessian is a diagonal matrix. Adding the restriction on differences of weights makes Hessian into a block banded matrix. In this case computing of an inverse of Hessian is more complicated. Looking at results of the experiments we cannot point out the best approach, because it strictly depends on kind of a data. In cases considered in the first experiment ℓ^2 penalties give quite good solutions, probably because of obvious correlation inside data, not necessarily corresponding to neighbouring variables.

In the second experiment neighbouring pixels have clear correlations and all methods using restrictions on differences of weights give better results, what can be seen especially for a small number of training examples. The most distinct differences we can see on artificial dataset Art1000, where learning with both components of penalties gives better results.

One should notice that here we consider only linear classifiers and without any selection of attributes between the experiment, thus the presented results are a bit weaker than those reported by machine learning community, however even in such cases we can obtain quite good results.

References

[1] H. Zou and T. Hastie, “Regularization and variable selection via the Elastic Net,” *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301–320, 2005.

[2] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear Total Variation Based Noise Removal Algorithms,” *Phys. D*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[3] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, “Sparsity and smoothness via the fused lasso,” *Journal of the Royal Statistical Society Series B*, pp. 91–108, 2005.

[4] T. Minka, “A comparison of numerical optimizers for logistic regression,” March 2003. [Online]. Available: https://www.microsoft.com/en-us/research/publication/comparison-numerical-optimizers-logistic-regression/

[5] D. Freedman, *Statistical Models: Theory and Practice*. Cambridge University Press, Aug. 2005.

[6] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng, “Efficient L1 regularized logistic regression,” in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[7] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, “An Improved GLMNET for L1-regularized Logistic Regression,” *Journal of Machine Learning Research*, vol. 13, pp. 1999–2030, Jun 2012.

[8] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, “Pathwise Coordinate Optimization,” *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[9] K. Koh, S.-J. Kim, and S. Boyd, “An Interior-Point Method for Large-Scale ℓ_1-Regularized Logistic Regression,” *Journal of Machine Learning Research*, vol. 8, pp. 1519–1555, Jul 2007.

[10] P. J. Huber, “Robust estimation of a location parameter,” *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, Mar. 1964.

[11] C.-J. Lin, R. C. Weng., and S. S. Keerthi, “Trust Region Newton Method for Logistic Regression,” *Journal of Machine Learning Research*, vol. 9, pp. 627–650, Apr 2008.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] [Online]. Available: http://pages.cs.wisc.edu/ dpage/kddcup2001/

[14] M. Pierre, “Automatic identification of mixed bacterial species fingerprints in a MALDI-TOF mass-spectrum,” *Bioinformatics*, 2014.

[15] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Eds., *Feature Extraction, Foundations and Applications*. Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2006.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[17] C. Bishop, *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.