Using Matrix Factorization Methods for Multiclass Classification Tasks

Abstract

In this paper we introduce a new approach to multiclassification based on simultaneous build of weighting functions for all classes. The proposed algorithm provides more lightweight decision function while being on par with state-of-the-art methods in terms of prediction quality. The method is based on multinomial logistic regression and gradient boosting. The first part allows the researcher not to bother about balancing of different classes in dataset, and the second give her the opportunity to control the time to model quality ratio.

1. Introduction

There are three basic approaches to multiclassification task: one-vs-rest, multinomial logistic regression (Hardin & Hilbe, 2001), and using complex system of binary classifiers with associated modeling matrix (Allwein et al., 2000). Each of these basis ideas have their advantages. The simplicity of one vs. rest makes it one of the most popular methods, on the other hand using this approach forces the researcher to balance class representatives in such a way that probabilities provided by different classes could be compared with each other (Weiss & Provost, 2001; Japkowicz & Stephen, 2002). Multinomial logistic regression allows to build the scoring functions comparable by their design (Oommen et al., 2011), but it takes ages to learn it. Modeling matrix approach generalize the one-vs-rest allowing one to build binary classifiers complimentary to each other. This way allows us to build smaller number of decision functions than the $K-1$, where K is number of classes. There are some works about the building of the modeling matrix (Zhao & Xing, 2013; Crammer & Singer, 2000b), however there is no good way to build this matrix universally and it should be constructed in ad-hoc manner in most cases.

Our first goal in this work is to combine approaches to get the method [as simple as one-vs-rest???, but]less sensitive to class imbalance. The second goal is to get lighter decision function to be able to use it for large number of classes. To achieve these goals we employ two methods: gradient boosting for multinomial logistic regression model and matrix factorization. The multinomial logistic regression model particularly allows to forget about class imbalance; gradient boosting allows one to control the time to model quality ratio; and matrix factorization allows to reduce the count of weak models in the gradient boosting.

Our contribution as follows:

- We propose the GradFac multiclassification algorithm that is based on gradient boosting and matrix factorization. We show that in most cases it can reduce total count of weak models without sufficient quality degradation.
- We describe the main disadvantage of the new algorithm and propose two fixes: matrix columns bootstrapping and *ElasticNet* factorization.
- We apply the GradFac algorithm to the classic gradient boosted tree classifier on benchmark datasets.

2. Multinomial logistic regression

In this section we consider application of gradient boosting algorithm to the multinomial logistic regression problem. This problem was well described in [...], so we just reformulate already known. Assume we have a training instances set $D = (x^i, y^i)_{i=1}^N$, where $x \in \mathbb{R}^n$ is an instance's feature vector and $y \in \{1, \dots, K\}$ is an instance's class label. According the multinomial regression model definition (Hardin & Hilbe, 2001), we need to build the decision in the next probabilistic form:

$$\mathbb{P}(Y = c|x) = \begin{cases} \frac{e^{s_c(x)}}{1+\sum_{j=1}^{K-1} e^{s_j(x)}}, & c \in \{1, \dots, K-1\} \\ \frac{1}{1+\sum_{j=1}^{K-1} e^{s_j(x)}}, & c = K \end{cases} \tag{1}$$

Here $s_j(x), j \in \{1, \dots, K-1\}$ are real functions (a.k.a. *scoring functions*) of some class $\mathbb{F}$:

$$s_j(x) : \mathbb{R}^n \to \mathbb{R}, c = 1, \dots, K-1.$$

Note that final decision function is fully determined by these functions. Consider corresponding log-likelihood

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

function for (4):

$$L(s_1,\dots,s_{K-1}|X) = \sum_{i=1}^{N} \ln \mathbb{P}(y_i|x_i) = \\ = \sum_{i=1}^{N} \ln \frac{e^{s_{y_i}(x_i)}}{1+\sum_{j=1}^{K-1} e^{s_j(x_i)}} = \\ = \sum_{i=1}^{N} \left(s_{y_i}(x_i) - \ln\left(1+\sum_{j=1}^{K-1} e^{s_j(x_i)}\right)\right)$$

And the problem is formulated as follows: find $K-1$ functions $s_1^*(x),\dots,s_{K-1}^*(x)$ of class $\mathbb{F}$ such that

$$(s_1^*,\dots,s_{K-1}^*) = \arg\max_{s_j\in\mathbb{F}} L(s_1,\dots,s_{K-1}|X). \quad (2)$$

We employ gradient boosting for solving (2). Note that with fixed functions $s_1,\dots,s_{K-1}$ target function L becomes a $N\times(K-1)$ multivariate function of $s_1(x_1),\dots,s_{K-1}(x_N)$ variables. Consider the gradient[1] of that function:

$$\nabla L = \begin{pmatrix} \frac{\partial L}{\partial s_1(x_1)} & \cdots & \frac{\partial L}{\partial s_{K-1}(x_1)} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial s_1(x_N)} & \cdots & \frac{\partial L}{\partial s_{K-1}(x_N)} \end{pmatrix}_{N\times(K-1)}$$

Partial derivatives of L are:

$$\frac{\partial L}{\partial s_j(x_i)} = \frac{\partial L(s_1(x_i),\dots,s_{K-1}(x_i))}{\partial s_j(x_i)} = \\ = \frac{e^{s_j(x_i)}}{1+\sum_{k=1}^{K-1} e^{s_k(x_i)}} - I\{y_i = j\} \quad (3)$$

According the gradient boosting algorithm, on each iteration we have to train a L2-approximation of gradient of the target function. However, in our case we have to approximate the full gradient by the *set* of $K-1$ functions. Therefore, we have to train $K-1$ approximation models: one model per each column of the gradient matrix. See Algorithm 1 for details.

3. GradFac

3.1. Motivation

We will look for decision function in the class of the following functions:

$$H_c(x) = \sum_t b_{ct} h_t(x)$$

[1]Formally, the matrix notation for partial derivatives is reserved by Jacobian. We use the gradient's matrix notation for the convenience here.

Algorithm 1 Gradient boosting for MLR

```
  **Input:** step $\alpha$, iterations count $T$.
  $H^{(0)}(x) := \mathbb{O} \in \mathbb{F}^{K-1}$ {initial zero model}
  $\overline{x}^{(0)} := \mathbb{O} \in \mathbb{R}^{N\times(K-1)}$ {initial cursor}
  **for** $t = 1$ **to** $T$ **do**
    Evaluate $\nabla L$ at $\overline{x}^{(t-1)}$ using (3).
    **for** $j = 1$ **to** $K-1$ **do**
      Train weak model $h_j^{(t)}(x)$ using $\{X, \nabla L^{[j]}\}$ as a training set and MSE as a target function:
      $$h_j^{(t)}(x) = \arg\min_{h\in\mathbb{F}} \sum_{i=1}^{N} (\nabla L_{ij} - h(x_i))^2$$
    **end for**
    Update model: $H^{(t)}(x) = H^{(t-1)}(x) + \alpha h^{(t)}(x)$.
    Update cursor: $\overline{x}^t = \overline{x}^{(t-1)} + h^{(t)}(X)$.
  **end for**
```

where $H_c(x)$ is a scoring function for class c at point x, b_c is scalar value, associated with the class c at step t, and $h_t(x)$ is some real-valued function of x. The probability of class c will look exactly like one in multinomial logistic regression:

$$P(Y=c|x,H) = \begin{cases} \frac{e^{H_c(x)}}{1+\sum_{j=1}^{K-1} e^{H_j(x)}}, & c\in\{1,\dots,K-1\} \\ \frac{1}{1+\sum_{j=1}^{K-1} e^{H_j(x)}}, & c = K \end{cases} \quad (4)$$

This type of decision function look very similar to those using modeling matrix, the difference is that we use real values instead of those from $\{-1,0,1\}$ set in modeling matrix $\mathbb{B} = \{b_{ct}\}$ and will build regression on each step instead of classification.

To build a sequence of b_t and h_t we use the idea of Friedmans' gradient boosting. The loss function we use is negative likelihood of the data based on introduced probability $P(Y=c|x,H)$:

$$\hat{H}(x) = \arg\min_H - \sum_{(x,y)\in X} \log P(Y=c|x,H)$$

where (x,y) is an example coming from training set X, H—vector of size $k-1$. On each step we build a weak model $(b_t, h_)$ to be as close to likelihood gradient matrix $\frac{\partial L}{\partial H_c(x)}(X)$ as possible:

$$(b_t,h_t) = \arg\min_{b,h} \left\| \frac{\partial L}{\partial H_c(x)}(X) - b_t h_t(X) \right\|_F$$

3.2. Main idea

Algorithm 1 allows to build an approximation of the gradient's matrix as the set of $h_j(x)$ functions:

$$\nabla L \approx \begin{pmatrix} h_1(x_1) & \cdots & h_{K-1}(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_{K-1}(x_N) \end{pmatrix}_{N\times(K-1)}$$

We employ matrix factorization to reduce the set of functions to the single function. Consider rank-1 approximation of the ∇L matrix:

$$\nabla L \approx \overline{u}\,\overline{v}^T, u \in \mathbb{R}^N, v \in \mathbb{R}^{K-1},$$

where

$$\overline{u}, \overline{v} = \arg\min_{u,v} \sum_{i,j} (\nabla L_{ij} - uv)^2. \quad (5)$$

Given the real vector $\overline{u}$, we can train weak model on that:

$$u(x) = \arg\min_{u\in\mathbb{F}} \sum_{i=1}^{N} \left(\overline{u}_i - u(x_i)\right)^2.$$

Desired functions $h_1(x), \ldots, h_{K-1}(x)$ could be expressed as a product of $u(x)$ and corresponding j-th element of the constant vector $\overline{v}$:

$$h_j(x) = u(x) \cdot \overline{v}_j.$$

Therefore, the actual gradient's matrix approximation could be written as follows:

$$\nabla L \approx \begin{pmatrix} u(x_1)\cdot\overline{v}_1 & \cdots & u(x_1)\cdot\overline{v}_{K-1} \\ \vdots & \ddots & \vdots \\ u(x_N)\cdot\overline{v}_1 & \cdots & u(x_N)\cdot\overline{v}_{K-1} \end{pmatrix}_{N\times(K-1)}$$

Now we can rewrite the weak models training stage in Algorithm 1:

1. Factorize the gradient's matrix:
$$\overline{u}, \overline{v} = \arg\min_{u,v} \sum_{i,j} (\nabla L_{ij} - u_i v_j)^2.$$

2. Train a weak model on the vector $\overline{u}$:
$$u(x) = \arg\min_{u\in F} \sum_{i=1}^{N} \left(\overline{u} - u(x_i)\right)^2. \quad (6)$$

3. Compose vector function $h(x)$:
$$h(x) = (u(x)\cdot\overline{v}_1, \ldots, u(x)\cdot\overline{v}_{K-1}) = u(x)\overline{v}.$$

Note that (5) may be effectively solved by ALS(Hu et al., 2008).

3.3. Matrix factorization

(This problem could be efficiently solved by *alternating least squares* method (Hu et al., 2008).)

According to the Eckart-Young-Mirsky theorem (Eckart & Young, 1936), solving (5) means finding the left and the right singular vectors of ∇L associated with the largest singular value of ∇L. Therefore, one may apply the next algorithm for solving this problem:

1. Evaluate the singular decomposition of gradient's matrix: $\nabla L = U\Sigma V^T$
2. Take the largest singular value σ_1 and associated singular vectors u and v.
3. Return $\overline{u} = \sigma_1 \|v\|_2 u$ and $\overline{v} = \frac{1}{\|v\|_2} v$ as solution.

However, experiments show that starting from the some iteration, singular values become too close to each other and choice of singular vectors associated with the largest singular value becomes non-trivial. It leads to factorization error growth because the single pair of the left and the right singular vectors is no longer meaningful characteristic of the matrix. We call this negative effect "the spreading of singular values", as the matrix is spreading across several pairs of singular vectors. To deal with this effect we propose two methods: regularized factorization and columns bootstrap.

3.3.1. Regularization

Instead of solving (5), consider the next problem:

$$u^*, v^* = \arg\min_{u,v} \sum_{i,j} \left(\nabla L_{ij} - u_i v_j\right)^2 + \\ + \alpha_1\|u\|_1 + \alpha_2\|u\|_2^2 + \beta_1\|v\|_1 + \beta_2\|v\|_2^2. \quad (7)$$

Added terms are called *Elastic-Net regularization* for (5) (Zou & Hastie, 2005). Usually similar problems are considered for non-negative matrix factorization (Guan et al., 2012),(Yeuntyng et al., 2013), however we don't need such constraint.

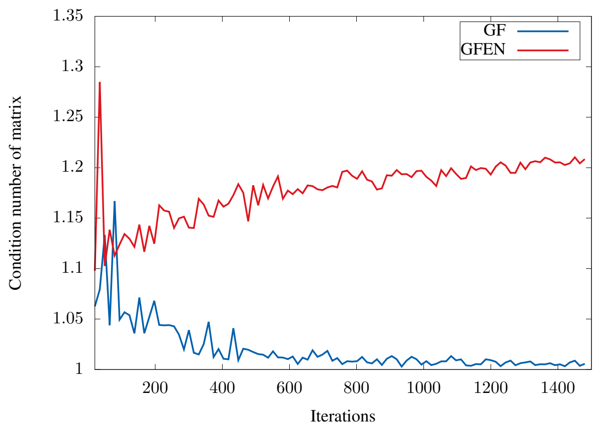
We employ alternating iterations idea [ref to ALS] for solving (7). For example, consider (7) with fixed $\overline{v}$. Then:

$$u^* = \arg\min_{u} \sum_{i,j} \left(\nabla L_{ij} - u_i\overline{v}_j\right)^2 + \\ + \alpha_1\|u\|_1 + \alpha_2\|u\|_2^2 + \beta_1\|\overline{v}\|_1 + \beta_2\|\overline{v}\|_2^2 = \\ = \arg\min_{u} \sum_{i,j} \left(\nabla L_{ij} - u_i\overline{v}_j\right)^2 + \alpha_1\|u\|_1 + \alpha_2\|u\|_2^2$$

Therefore, we have reduced the source problem to the linear regression problem with *Elastic-Net* regularization. Indeed, we can rewrite it in canonical form:

$$x^* = \arg\min \|y - Ax\|_2^2 + \alpha_1\|x\|_2^2 + \alpha_2\|x\|_1,$$

Figure 1. Condition number of the gradient matrix during training GF and GFEN models on *segmentation* dataset.



where $A \in \mathbb{R}^{N(K-1)\times N}$ and $y \in \mathbb{R}^{N(K-1)}$:

$$A = \begin{pmatrix} v_1 & & \\ \vdots & \mathbb{O} & \\ v_{K-1} & & \\ & \ddots & \\ & & v_1 \\ & \mathbb{O} & \vdots \\ & & v_{K-1} \end{pmatrix}, y = \begin{pmatrix} \nabla L_{1,1} \\ \vdots \\ \nabla L_{1,K-1} \\ \vdots \\ \nabla L_{N,1} \\ \vdots \\ \nabla L_{N,K-1} \end{pmatrix}$$

Detailed algorithm for *Elastic-Net* problem could be found at (Zou & Hastie, 2005) or (Hastie et al., 2001).

Similarly, one could derive the linear system in case of fixed vector u. Hence, the final algorithm is similar to ALS but instead of alternating gradient steps one needs to alternate solving *Elastic-Net* problem. Due to the simple structure of matrix A, these problems could be solved very efficiently.

3.3.2. Columns bootstrapping

We employ statistical bootstrap idea (Efron, 1992). On each iteration we assign random integer weights to columns of the gradient's matrix. It ensures that singular values will be "shake-up" and consequently it particularly protects us from the problem described above. Required weights could be generated by discrete random variable ξ that has a Poisson distribution with $\lambda = 1$. Due to the fact that $\mathbb{E}\xi = 1$, this weighing approach has a simple physical meaning: in the most of cases we consider all columns of the gradient's matrix but sometimes we amplify ($\xi > 1$) or mute ($\xi = 0$) some of them.

Table 1. Statistics for the classification datasets.

Data set	Examples	Features	Classes
wine	178	17	3
letter	20000	16	26
MNIST	60000	785	10
pendigits	7494	21	8
segmentation	2300	23	6

4. Discussion

The main advantage of the *GradFac* algorithm is the independence of the number of classes on the training stage: it's only required to train a single model instead of $K-1$ models. However, factorization stage depends on the number of classes. Consider impact of factorization on quality of the final classification model. Obviously, matrix factorization increases total error because of replacement the whole matrix to outer product of two estimated vectors u and v. Should we decrease the algorithm's quality on purpose? To answer this question one should remember the ability of the gradient boosting method to accumulate weak models in order to obtain the strong. Therefore, to compensate introduced error, we have to increase boosting iterations count and train some additional weak models (*one* per iteration). Suppose $L(H_T(x)|X,Y) \leq \varepsilon$ is true for the source algorithm after T_1 iterations and for the *GradFac* algorithm after T_2 iterations. Note that $T_1 < T_2$ because we need to compensate factorization error. Also note that the source algorithm requires to train $K-1$ weak models and the *GradFac* algorithm requires to train 1 model. It's hard to say definitely which model includes less weak models count:

$$\begin{array}{rccc} \text{Iterations count:} & T_1 & < & T_2 \\ \text{Weak models count:} & T_1 \times (K-1) & ?? & T_2 \end{array}$$

We will come back to this issue in the experiments section.

5. Experiments

We have tested the *GradFac* algorithm with natural data from the UCI repository (A. Asuncion, 2007). Table 1 shows characteristics of different datasets used.

5.1. Experimental setup

Let us remind that our main goal is minimization of the total weak models count. In each experiment we compare micro-F_1-score that could be reached with fixed count of weak models.

All considered models are multinomial logistic regression models that were trained with gradient boosting method. We use oblivious decision tree with depth = 6 [ref] as a weak model. The main difference between compared models is learning method for scoring functions $h_j(x)$ on each iteration of the gradient boosting algorithm. We consider three learning methods:

1. Multinomial logistic regression. For each j-th column of the gradient's matrix we train separate decision tree and use this tree as $h_j(x)$.

2. *GradFac*. Using ALS we factorize the gradient's matrix to product of two vectors $\overline{u}$ and $\overline{v}$. The scoring function $h_j(x)$ is expressed as a product of $u(x)$ and $\overline{v}_j$, where $u(x)$ is a decision tree trained on vector $\overline{u}$.

3. *GradFac* with *Elastic-Net* regularization. Similar to previous, but factorization is performed via alternating of *Elastic-Net* problems.

Also we include classic one-vs-rest approach to comparison. For each class we train binary logistic regression model with gradient boosting method. Again, we use oblivious decision tree with depth = 6 as a weak model.

5.2. Results

The results are presented in Table 2. Each cell contains mean and standard deviation of the micro-F_1-score which evaluated with 10-folds cross-validation. We find that our proposed algorithm GFEN almost always achieves the highest F_1-score compared to other models. Also we see that in most cases we could sufficiently reduce the weak models count for GFEN or GF methods and stay competitive with state-of-the-art methods like MLR or OVR.

6. Conclusion

We have introduced in this paper a new multiclassification algorithm GradFac that is based on the idea of gradient's matrix factorization. Experiments demonstrated that our algorithm allows to build up to 3 times easier model than state-of-the-art models like OVR or MLR without quality degradation. Of course, more experiments are needed to better understand applicability limits of this method, especially for tasks with large class count.

There are several avenues for future research. One of the most simple ideas - variation of considering eigen vectors count (instead of 1). GradFac is also appealing for multilabel tasks because there are several target functions for such tasks (Tsoumakas & Katakis, 2007) that allow to represent their gradients as matrix and consequently allow to apply factorization techniques.

Table 2. Micro-averaged F_1 scores for the multinomial logistic regression (MLL), *GradFac* (GF), *GradFac* with *Elastic-Net* regularization (GFEN), One-vs-Rest (OVR) models on benchmark datasets.

DATASET	# MODELS	MLL	GF	GFEN	OVR
WINE	30	**0.949** $\pm$ **0.047**	0.948 $\pm$ 0.041	0.931 $\pm$ 0.051	0.945 $\pm$ 0.085
	60	**0.970** $\pm$ **0.037**	0.967 $\pm$ 0.039	0.955 $\pm$ 0.060	0.951 $\pm$ 0.067
	90	**0.975** $\pm$ **0.037**	0.969 $\pm$ 0.046	0.955 $\pm$ 0.060	0.955 $\pm$ 0.069
LETTERS	3120	0.915 $\pm$ 0.005	0.946 $\pm$ 0.005	**0.947** $\pm$ **0.002**	0.922 $\pm$ 0.006
	6240	0.933 $\pm$ 0.004	0.958 $\pm$ 0.004	**0.958** $\pm$ **0.003**	0.943 $\pm$ 0.004
	9100	0.940 $\pm$ 0.005	0.960 $\pm$ 0.004	**0.962** $\pm$ **0.004**	0.950 $\pm$ 0.004
MNIST	1300	0.952 $\pm$ 0.002	**0.964** $\pm$ **0.002**	0.963 $\pm$ 0.002	0.957 $\pm$ 0.002
	2600	0.963 $\pm$ 0.002	**0.970** $\pm$ **0.001**	0.970 $\pm$ 0.002	0.966 $\pm$ 0.002
	4000	0.967 $\pm$ 0.002	0.972 $\pm$ 0.001	**0.973** $\pm$ **0.002**	0.970 $\pm$ 0.001
PENDIGITS	1300	0.988 $\pm$ 0.003	0.991 $\pm$ 0.004	**0.992** $\pm$ **0.003**	0.990 $\pm$ 0.003
	2600	0.990 $\pm$ 0.003	0.992 $\pm$ 0.004	**0.993** $\pm$ **0.003**	0.991 $\pm$ 0.002
	4000	0.991 $\pm$ 0.003	0.992 $\pm$ 0.003	**0.993** $\pm$ **0.003**	0.992 $\pm$ 0.002
SEGMENTATION	490	0.981 $\pm$ 0.010	0.984 $\pm$ 0.008	**0.985** $\pm$ **0.007**	0.977 $\pm$ 0.008
	1050	0.982 $\pm$ 0.008	0.984 $\pm$ 0.008	**0.986** $\pm$ **0.008**	0.980 $\pm$ 0.008
	1750	0.982 $\pm$ 0.009	0.985 $\pm$ 0.008	**0.987** $\pm$ **0.008**	0.981 $\pm$ 0.008

Acknowledgments

References

A. Asuncion, D.J. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Allwein, Erin L., Schapire, Robert E., and Singer, Yoram. Reducing multiclass to binary: A unifying approach for margin classifiers. *JOURNAL OF MACHINE LEARNING RESEARCH*, 1:113–141, 2000.

Crammer, Koby and Singer, Yoram. On the learnability and design of output codes for multiclass problems. In *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pp. 35–46, 2000a.

Crammer, Koby and Singer, Yoram. On the learnability and design of output codes for multiclass problems. In *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pp. 35–46, 2000b.

Eckart, Carl and Young, Gale. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3): 211–218, 1936.

Efron, Bradley. *Bootstrap Methods: Another Look at the Jackknife*, pp. 569–593. Springer New York, 1992.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

Friedman, Jerome H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29: 1189–1232, 2000.

Guan, Naiyang, Tao, Dacheng, Luo, Zhigang, and Shawe-Taylor, John. Mahnmf: Manhattan non-negative matrix factorization. *CoRR*, abs/1207.3438, 2012.

Gulin, Andrey, Kuralenok, Igor, and Pavlov, Dmitry. Winning the transfer learning track of yahoo!s learning to rank challenge with yetirank. In *JMLR Workshop and Conference Proceedings*, pp. 63–76, 2011.

Hardin, James W. and Hilbe, Joseph. *Generalized Linear Models and Extensions*. College Station, Texas: Stata Press, 2001.

Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. The elements of statistical learning – data mining, inference, and prediction, 2001.

Hu, Yifan, Koren, Yehuda, and Volinsky, Chris. Collaborative filtering for implicit feedback datasets. In *In IEEE International Conference on Data Mining (ICDM 2008*, pp. 263–272, 2008.

Japkowicz, Nathalie and Stephen, Shaju. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, pp. 429–449, 2002.

Koren, Yehuda, Bell, Robert, and Volinsky, Chris. Matrix factorization techniques for recommender systems, 2009.

Lee, Daniel D. and Seung, H. Sebastian. Algorithms for non-negative matrix factorization. In *In NIPS*, pp. 556–562. MIT Press, 2001.

Oommen, Thomas, Baise, Laurie G., and Vogel, Richard M. Sampling bias and class imbalance inmaximum-likelihood logistic regression. *Mathematical Geosciences*, 43(1):99–120, 2011.

Rifkin, Ryan and Klautau, Aldebaro. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Tsoumakas, Grigorios and Katakis, Ioannis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

Weiss, Gary and Provost, Foster. The effect of class distribution on classifier learning: An empirical study. Technical report, 2001.

Yeuntyng, Lai, Morihiro, Hayashida, Luo, Zhigang, and Tatsuya, Akutsu. Survival analysis by penalized regression and matrix factorization. *The Scientific World JournalRR*, 2013, 2013.

Zhao, Bin and Xing, Eric P. Sparse output coding for large-scale visual recognition. *International Journal of Computer Vision*, 119:60–75, 2013.

Zou, Hui and Hastie, Trevor. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.