Using Matrix Factorization Methods for Multiclass Classification Tasks

Abstract

We consider multiclass classification tasks with large number of classes. State-of-the-art methods like one-vs-rest try to reduce multiclass task to the set of binary classification tasks and build decision function as committee. However, learning and evaluation of such decision function may take a long time in case of the large number of classes. We combine gradient boosting and matrix factorization techniques in proposed *GradFac* algorithm. It allows to build decision function in ensemble manner and learns the only one regressor (instead of K or $K-1$ classifiers) on each iteration.

1. Introduction

Logistic regression was the one of the first binary classification algorithms and it still remains relevant in nowadays. The natural generalization of this method for the case of multiple classes is multinomial logistic regression. Friedman in 2001 described the way of training multinomial logistic model via gradient boosting machine. It allows to build the decision in the form of an ensemble of weak prediction models (e.g. decision trees). However, such model has a sufficient disadvantage: the model complexity is too big. On each iteration one needs to train K or $K-1$ regressors (see formal explanation below). For example, if one is solving the problem with 100 classes and setting 10^4 iterations for gradient boosting, then total weak models count will be equal to 10^6. Of course, training and evaluating such huge amount of models may take a long time.

In this paper, we show that multinomial model that was obtained via boosting is too redundant in the most cases. To reduce the complexity of the model, we propose *GradFac* algorithm. This is an extension of gradient boosting scheme for multinomial logistic regression. We employ matrix factorization techniques on each iteration of gradient boosting to decrease regressors count from K to 1. This property makes the approach appealing for problems with large class count or problems with hard limit on the model complexity.

To sum up, our contribution as follows:

- We propose the GradFac multiclassification algorithm that is based on gradient boosting and matrix factorization. We show that in case of the large class problem it can reduce total weak models count 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 [...], 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. *discriminant 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 function for (1):

$$L(s_1, \dots, s_{K-1}|X) = \sum_{i=1}^N \ln \mathbb{P}(y_i|x_i) =$$

$$= \sum_{i=1}^N \ln \frac{e^{s_{y_i}(x_i)}}{1 + \sum_{j=1}^{K-1} e^{s_j(x_i)}} =$$

$$= \sum_{i=1}^N \left(s_{y_i}(x_i) - \ln \left(1 + \sum_{j=1}^{K-1} e^{s_j(x_i)} \right) \right)$$

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
```

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

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

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:

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

According the gradient boosting scheme, we have to train a L2-approximation of gradient of the target function on each iteration. 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.

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

3. GradFac

3.1. Motivation

In the previous section we have shown the application of gradient boosting method to the multinomial logistic regression problem [describe that this is not our contribution]. One of the main disadvantages of this approach is model complexity. For example, if one is solving the multiclass problem with K classes using gradient boosting with T iterations, then total model count will be equal to $T \times (K-1)$. In practice, the number of boosting iterations is measured in thousands [cite YetiRank]. Consequently, in case of the problem with 100 classes total weak models count will be measured in hundreds of thousands.

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

$$\overline{u}, \overline{v} = \arg\min_{u,v} \sum_{i,j} (\nabla L_{ij} - uv)^2. \qquad (4)$$

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

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

Desired functions $h_1(x), \dots, h_{K-1}(x)$ 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 $\overline{u}$:
$$u(x) = \arg\min_{u \in F} \sum_{i=1}^{N} \left(\overline{u} - u(x_i)\right)^2. \qquad (5)$$

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 (4) may be effectively solved by ALS[ref].

3.3. Matrix factorization

(This problem could be efficiently solved by *alternating least squares* method [ref].)

According to the Eckart-Young-Mirsky theorem [...], solving (4) 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 (4), consider the next problem:
$$u^*, v^* = \arg\min_{u,v} \sum_{i,j} \left(\nabla L_{ij} - u_i v_j\right)^2 + \\ + \alpha_1 \|u\|_1 + \alpha_2 \|u\|_2^2 + \beta_1 \|v\|_1 + \beta_2 \|v\|_2^2. \qquad (6)$$

Added terms are called *Elastic-Net regularization* for (4) [ref to EN]. Usually similar problems are considered for non-negative matrix factorization [ref to MahNMF, another], however we don't need such constraint. [There are guys who solve this problem ......]

We employ alternating iterations idea [ref to ALS] for solving (6). For example, consider (6) 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:
$$x^* = \arg\min \|y - Ax\|_2^2 + \alpha_1 \|x\|_2^2 + \alpha_2 \|x\|_1,$$

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

Detailed algorithm for *Elastic-Net* problem could be found at [...].

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 [ref]. 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
GLASS	214	13	6
IRIS	150	8	3
LETTER	20000	16	26
MNIST	60000	785	10
PENDIGITS	7494	21	8
SEGMENTATION	2300	23	6
WINE	178	17	3

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 [ref, Murphy Aha 1994]. 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 weak models count that required to reach fixed micro/macro- F1-score on 10-fold cross-validation.

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 discriminant functions $h_j(x)$ on each iteration of the gradient boosting algorithm. We consider three learning methods:

1. Baseline. 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 discriminant 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.

5.2. Results

TBD

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 multi-label tasks because there are several target functions for such tasks (e.g. ...) that allow to represent their gradients as matrix and consequently allow to apply factorization techniques.

Acknowledgments

References

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.