УДК 519.23

ZERO-INFLATED BOOSTED ENSEMBLE FOR SMALL COUNT PROBLEM

© Borisov A.E., Tuv E.V.

Intel corporation
Nizhnhy Novgorod, Russia

e-mail: alexander.borisov@intel.com, tuv.eugene@intel.com

Abstract. The article introduces a new approach for modeling "small count data" where distribution of the response variable is assumed to follow the zero-inflated Poisson (ZIP) model. ZIP model based on boosted ensemble is introduced. It combines and extends ZIP tree and gradient boosting tree (GBT) methods. Our algorithm, called ZIP-GBT, is at first introduced from theoretical perspective in the framework of Friedman's gradient boosting machine. Then it is compared empirically on two real data sets and two artificial data sets versus single tree approach (ZIP-tree). It is shown that ZIP-GBT outperforms ZIP tree in most cases both in terms of cross validated ZIP-likelihood and ZIP distribution parameters prediction.

Introduction

The analysis of count data is the primary interest in many areas including public health, epidemiology, sociology, psychology, engineering, and agriculture. Poisson distribution is typically assumed to model the distribution of the rare event counts. The Poisson regression model is commonly used to explain the relationship between the count (non-negative integer) response and input variables (predictors). However, it is often the case that the outcome of interest contains excess number of zeros which cannot be explained correctly by the standard Poisson model.

Lambert [1] successfully proposed a mixture of the distribution with a point mass at zero and a Poisson distribution, called zero-inflated Poisson (ZIP) regression, to handle zero-inflated count data in a number of defects in a manufacturing process. After Lambert [1] successfully introduced the zero-inflated Poisson (ZIP) model, many extensions or modified ZIP models were elaborated. For example Wang [18] proposed Markov zero-inflated Poisson regression (MZIP), Li et.al [3] introduced multivariate ZIP models, Lee and Jin [2] proposed a tree-based approach for Poisson regression, Chiogna and Gaetan [11] used semi-parametric ZIP in animal abundance studies, Hsu [13] proposed a weighted ZIP, and Famoye and Singh [12] used zero-inflated generalized Poisson (ZIGP) regression model when the count data is over-dispersed. ZIP regression is not only applied in the manufacturing, but it is also widely used in many other areas such as public health, epidemiology, sociology, psychology, engineering, agriculture, etc. ([17], [16], [14], [10], [19]).

In data mining, tree-based model is one of the most popular and common methods used for approximating target functions, in which a function can be learned by splitting the data set into subsets based on an response-attribute value test. This process is repeated on each derived subset in a recursive manner and is represented by a tree model. Each terminal node is assigned a response value. A popular method of tree-based regression and

classification is called CART (Classification and Regression Tree) [9,4]. In 2006, Lee and Jin [2] introduced ZIP-tree model. They modified CART algorithm splitting criteria by using the zero-inflated Poisson (ZIP) likelihood error function instead of residual sum of squares. Each terminal node of ZIP tree is assigned its own ZIP distribution parameters (zero inflation probability p and Poisson distribution parameter λ).

Further development of the idea of using trees for ZIP regression leads to using a tree ensemble instead of a single tree. Ensemble methods are very popular in literature and widely used in practice, especially parallel (Random Forest, or RF, see [7,8]) and boosted tree ensembles (like AdaBoost or GBT[20,21]). Tree ensembles are shown to have smaller prediction error (bias) than a single tree; parallel ensembles (RF) also offer more stability (smaller variance).

In this paper, we propose a boosted ensemble approach similar to GBT that fits ZIP distribution parameters p, λ using two tree ensembles. The algorithm minimizes ZIP log-likelihood loss function by gradient descent method similar to the one proposed by Friedman for multi-class logistic regression (MCLRT) [21]. Our algorithm uses the log-link function for λ and the logit-link function for p as proposed in [1] for standard ZIP regression.

1. Previous work : ZIP regression and ZIP tree

Lambert [1] used ZIP distribution for response variable y, where Poisson distribution parameters depend on the values of input variables:

$$y_i \sim \begin{cases} 0, \text{ with probability } p_i, \\ Poisson(\lambda_i), \text{ with probability } 1-p_i, \end{cases} \quad i = 1 \dots n,$$

where n is the number of samples. This model implies that

$$P(y_i = k) = P(p_i, \lambda_i, k) = \begin{cases} p_i + (1-p_i)e^{-\lambda_i}, k = 0, \\ (1-p_i)e^{-\lambda_i}\lambda_i^k/k!, k = 1, 2, \dots, \end{cases}$$

where parameters λ, p are obtained from the linear combinations of inputs via log- and logit-link functions :

$$\log(\lambda_i) = \beta x_i, \text{ and } \log it(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \gamma x_i.$$

Here x is input feature vector (for simplicity of notation we always assume that a "dummy" variable $x = 1$ is added as the first input variable to take the intercept term into account), β, γ are vectors of coefficients (same for each data point) to be fit. The ZIP model is usually fitted using the maximum likelihood estimation method.

Log-likelihood can be maximized using Newton-Raphson method, but usually EM algorithm is used because it is more robust and computationally simpler. The authors study the behavior of their algorithm on AT&T Bell Labs soldering data. Another article[6] applies the same ZIP regression model to DMFT(decayed, missing and filled teeth) data. They use piecewise constant model for p parameter. Both articles consider using mixture models (most popular is mixture of Poisson and negative binomial distributions), but

both authors claim that such models are more difficult to fit and usually provide worse predictions than ZIP models.

In 2006, Lee and Jin [2] used the ZIP likelihood as new splitting criteria for decision tree. They modified CART (classification and regression tree) algorithm by using negative ZIP likelihood as an impurity measure in a node. Negative ZIP likelihood of the data in node T can be expressed as

$$L_{ZIP}(T) = L_{ZIP}(p, \lambda, y) = -n_0 \cdot \log\left(p + (1-p)e^{-\lambda}\right) - (n - n_0) \cdot (\log(1-p) - \lambda) - \\ - \sum_{x_i \in T} y_i \cdot \log\lambda + \sum_{x_i \in T, y_i > 0} \log(y_i!),$$

where p, λ are estimates of Poisson distribution parameters in node T.

The new splitting criterion is based on the difference of the ZIP likelihood in the left-child node and the right-child node from the ZIP likelihood in the parent node. The expression for split weight can be written as $\phi(s, T) = L_{ZIP}(T) - L_{ZIP}(T_L) - L_{ZIP}(T_R)$, where T is the parent node, T_L, T_R are left and right children of T, s is the split in node T. The same best split search strategy can be applied as in CART.

Parameter λ for a tree node is estimated using zero-truncated Poisson distribution:

$$\frac{\lambda}{1 - e^{-\lambda}} = \bar{y} = mean(y_i | y_i > 0, x_i \in T).$$

After λ parameter is obtained, p can be estimated from the known proportion of zero-class samples in the node :

$$p = \frac{n_0/n - e^{-\lambda}}{1 - e^{-\lambda}},$$

where n_0 is the number of zero count samples and n is total number of samples.

2. Boosting framework and ZIP boosted ensemble

Trees usually provide robust models for complex target functions (not limited by linearity assumptions) and are not sensitive to noise and outliers. They also allow working with mixed type data (both numeric and categorical predictors) and handle missed values in a natural way. CART trees are also very fast to fit (they do not require complex matrix operations as MLE problem). That is why trees are widely used in real life applications where data sets are mixed-type, large in number of samples and predictors, and noisy (both input variables and the response).

However, a single tree often has low predictive power (especially if an underlying target model is complex and multivariate) and is not stable to small fluctuations in the data. So different authors proposed using ensembles of trees for regression and classification problems (L. Breiman introduced parallel ensembles, or Random Forests [7,8], and J.H. Friedman introduced boosted ensembles [20, 21]). Ensembles have much higher predictive accuracy and generalization ability while keeping all advantages of the single tree. So ensemble methods become more and more popular as "off-the-shelf" approach and often provide as good results as best state-of-art methods.

Random Forest, a parallel ensemble, is a set of trees, with each of the trees build on a different (random) subsample of training data. In each node when searching for best split

only a small subset of input variables is selected randomly. Prediction from a set of trees is obtained using averaging prediction over trees in regression or voting in classification.

Gradient boosting, in its general form, constructs an additive regression (or logistic regression) model by sequentially fitting a simple parameterized function (a base learner that can be a tree or any other model) to current "pseudo-residuals" at each iteration. The pseudo-residuals are the gradient of the loss functional minimized with respect to the current parameter values, with respect to the model values at each training data point evaluated at the current step. Let's describe gradient boosting framework more formally.

Suppose we have a training sample $\{y_i, x_i\}_{i=1...n}, x_i = \{x_{i1}, \ldots, x_{im}\} \in X, y_i \in Y$, where n is the number of samples, m is the number of input variables. Our goal is to find a function $F^*(x) : X \to Y$ that minimizes expected value of the specified loss function $L(y, F(x))$ over the joint distribution of x, y values :

$$F^*(x) = \arg\min_{F(x)} E_{x,y} L(y, F(x)).$$

Here the expectation term cannot usually be computed directly as the joint distribution of x, y is not known. So in practice it is replaced with expected risk, i.e. :

$$F^*(x) = \arg\min_{F(x)} \sum_{i=1}^{n} L(y_i, F(x_i)).$$

Boosting uses an additive model to approximate $F^*(x)$: $F^*(x) = \sum_{m=0}^{M} h(x, a_m)$, where function $h(x, a_m)$ is some simple function ("base learner") of parameter vector a. Base learner parameters $a_m, m = 1 \ldots M$ are fit in forward stepwise manner. It starts from some initial approximation $F_0(x)$, then proceeds as follows :

$$a_m = \arg\min_{a} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + h(x_i, a)\right), \tag{1}$$

$$F_m(x) = F_{m-1}(x) + h(x, a_m).$$

Gradient boosting solves optimization problem (1) using the stepwise steepest descent method. The function $h(x, a)$ is fit by least squares:

$$a_m = \arg\min_{a} \sum_{i=1}^{n} (\tilde{y}_{im} - h(x_i, a))^2$$

to the current "pseudo-residuals" or "pseudo-response" :

$$\tilde{y}_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}. \tag{2}$$

Gradient tree boosting is the specialization of this approach to the case where base learner is a CART regression tree :

Algorithm 1 : Gradient tree boosting

1. $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$
2. For $m = 1$ to M do:
3. $\tilde{y}_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, i = 1 \dots n$
4. $\{R_{lm}\}_{l=1\dots L} = L-$ terminal node tree
5. 5. $\gamma_{lm} = \arg\min_{\gamma} \sum_{x_i \in R_{lm}} L(y_i, F_{m-1}(x_i) + \gamma)$
6. $F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_{lm} I(x \in R_{lm})$
7. End for

Here γ_{lm} is the response (mean) in node R_{lm}. Parameter ν is "shrinkage rate" or "regularization parameter" that controls learning rate of the algorithm. Smaller values for shrinkage (0.01-0.1) have proven to reduce over-fitting, thus allowing building models with better generalization ability. Usually only random part of samples (about 60%) are used to learn a tree on step 4 (bootstrapping). This speeds up model building and also reduces over-fitting.

A particularly interesting case of the algorithm 1 is a two-class logistic regression (that also has multi-class generalization that we will omit). It is derived from gradient tree boosting framework when using CART tree as a base learner, and negative binomial log-likelihood as the loss function.

Assume that response is binary, $y \in \{-1, 1\}$, and the loss function is negative binomial log-likelihood : $L(y, F) = \log(1 + \exp(-2yF))$, where F is a two-class logistic transform : $F(x) = \frac{1}{2}\log\left[\frac{\Pr(y=1|x)}{\Pr(y=-1|x)}\right]$. So each tree approximates log-odd of class 1 probability, and the pseudo-response derived from formula (2) or step 3 of Algorithm 1 is $\tilde{y}_{im} = 2y_i/(1 + \exp(2y_i F_{m-1}(x_i))$.

Optimization problem on step 5 cannot be solved in closed form, so single Newton-Raphson step approximation is used :

$$\gamma_{lm} = \sum_{x_i \in R_{jm}} \tilde{y}_{im} = \sum_{x_i \in R_{jm}} |\tilde{y}_{im}| \cdot (2 - |\tilde{y}_{im}|) \tag{3}$$

To increase the robustness of GBT algorithm, influence trimming can be applied when selecting samples for building a subsequent tree. Suppose we want to estimate the response in a terminal node on step 5 of Algorithm (1) via equation

$$\sum_{x_i \in R_{lm}} \partial L(y_i, F_{m-1}(x_i) + \gamma)/\partial\gamma = 0.$$

Influence of i-th sample on the solution can be gauged by the second derivative of the loss function, i.e

$$w_i = w(x_i) = \partial^2 L(y_i, F_{m-1}(x_i)+\gamma)/\partial\gamma^2|_{\gamma=0} = \partial^2 L(y_i, f)/\partial f^2|_{f=F_{m-1}(x_i)} = |\tilde{y}_{im}|\cdot(2-|\tilde{y}_{im}|).$$

When building subsequent tree, we omit all observations with $w_i < w_{l(\alpha)}$, where $l(\alpha)$ is the solution to $\sum_{i=1}^{l(\alpha)} w_{(i)} = \alpha \cdot \sum_{i=1}^{N} w_i$ (here weights $w_{(i)}$ are w_i's sorted in ascending order), and α is usually chosen in [0.05,02] range.

Influence trimming not only speeds up the tree construction, but also improves robustness of Newton-Raphson method step (equation 4), preventing small denominator values for a tree node, because denominator is proportional to the sum of sample influences in the node. Influence trimming

Now we are ready to derive our own algorithm for the ZIP regression problem using negative ZIP-likelihood as a loss function, and CART model as a base learner. We use two ensembles of trees to approximate transformed Poisson distribution parameters (p, λ). We use the same transformation (link function) as used by Lambert[1], i.e log-link for p and logit-link for λ :

$$\mu = \log\left(p/(1-p)\right), p = e^{\mu}/(1+e^{\mu}),$$
$$\nu = \log(\lambda), \lambda = e^{\nu}.$$

So the first ensemble fits model for $\mu(x)$, the second one – for $\nu(x)$. Initial value for ν is estimated from zero truncated Poison distribution of the response :

$$\frac{\lambda_0}{1-e^{-\lambda_0}} = \bar{y} = mean(y_i|y_i > 0), \nu_0 = \log(\lambda_0),$$

then μ_0 as

$$p_0 = \frac{n_0/n - e^{-\lambda_0}}{1-e^{-\lambda_0}}, \mu_0 = \log it(p_0),$$

where n_0 is the number of zero-class samples $(y_i = 0)$.

The loss function to be minimized takes the form

$$L(y,p,\lambda) = \sum_{i=1}^{n} L(y_i, p_i, \lambda_i) = -\sum_{y_i=0} \log\left(p_i + (1-p_i)e^{-\lambda_i}\right) -$$
$$-\sum_{y_i>0}\left(\log(1-p_i) - \lambda_i\right) - \sum_{y_i>0} y_i \log\lambda_i + \sum_{y_i>0}\log(y_i!),$$

where we denoted $p_i = p(x_i), \lambda_i = \lambda(x_i)$ to simplify notation. Last term is not dependent on the model and can be dropped. In other terms,

$$L(y,p,\lambda) = L(y,\mu,\nu) = \sum_{i=1}^{n} L(y_i,\mu_i,\nu_i) =$$
$$= -\sum_{y_i=0}\left(\log(e^{\mu_i} + \exp(-e^{\nu_i}) - \log(1+e^{\mu_i})\right) + \sum_{y_i>0}\left(\log(1+e^{\mu_i}) + e^{\mu_i}\right) - \sum_{y_i>0} y_i\nu_i,$$

where $\mu_i = \mu(x_i), \nu_i = \nu(x_i)$.

Pseudo-responses are calculated as follows. Pseudo-response for p-ensemble is

$$\tilde{\mu}_{im} = -\left[\frac{\partial L(y_i,\mu_i,\nu_i)}{\partial\mu_i}\right]_{\mu_i=\mu_{m-1}(x),\nu_i=\nu_{m-1}(x)} =$$

$$= \left[\frac{\partial L(y_i,p_i,\lambda_{m-1}(x_i))}{\partial p_i}\right]_{p_i=p_{m-1}(x)} \cdot \left[\frac{\partial p(\mu_i)}{\partial\mu_i}\right]_{\mu_i=\mu_{m-1}(x)} =$$

$$= \begin{cases} \frac{(e^{-\lambda_i}-1)}{p_i+(1-p_i)e^{-\lambda_i}} \cdot p_i(1-p_i), & y_i = 0, \\ 1/(1-p_i) \cdot p_i(1-p_i) = p_i & y_i > 0, \end{cases}$$

where $p_i = p_{m-1}(x_i), \lambda_i = \lambda_{m-1}(x_i)$. Here pseudo-response is expressed in terms of p_i, λ_i to simplify notation.

Pseudo-response for λ-ensemble is derived in the same way (note that $\frac{\partial p(\mu)}{\partial \mu} = e^{\mu}/(1+e^{\mu})^2 = p(1-p)$, $\frac{\partial \lambda(\nu)}{\partial \nu} = e^{\nu} = \lambda$):

$$\tilde{\nu}_{im} = \begin{cases} \lambda_i e^{-\lambda_i} / \left(p_i/(1-p_i) + e^{-\lambda_i}\right), & y_i = 0, \\ \lambda_i - y_i, & y_i > 0. \end{cases}$$

Then node response optimization problem on step 5 of algorithm 1 is solved via single step of Newton-Raphson as in Friedman's two class logistic regression. Unfortunately in our case Hessian (second derivative) can be negative sometimes, although such occasions are rare and possibly indicate over-fitting or "self-contradictory" data i.e a case when data points with similar x values have very different (p, λ) values. Negative Hessian means that the target function is not concave and thus cannot be approximated by 2-nd order polynomial. In such case we use one step of steepest descent instead of Newton-Raphson step. Second derivatives for p-tree (which are summands in denominator in formula (4)) are:

$$\partial^2 L(y_i, \mu_{m-1}(x_i) + \gamma, \nu_{m-1}(x_i))/\partial \gamma^2|_{\gamma=0} = \partial^2 L(y_i, \mu_i, \nu_{m-1}(x_i))/\partial \mu_i^2|_{\mu_i = \mu_{m-1}(x_i)} =$$
$$= \tilde{\tilde{\mu}}_{im} = \begin{cases} -\frac{(1-p_i)^2 e^{-\lambda_i} - p_i^2}{(p_i+(1-p_i)e^{-\lambda_i})^2} \cdot p_i(1-p_i) \cdot (1-e^{-\lambda_i}), & y_i = 0, \\ p_i(1-p_i), & y_i > 0. \end{cases}$$

Same for λ-tree:

$$\partial^2 L(y_i, \mu_{m-1}(x_i), \nu_{m-1}(x_i) + \gamma)/\partial \gamma^2|_{\gamma=0} = \partial^2 L(y_i, \mu_{m-1}(x_i), \nu_i)/\partial \nu_i^2|_{\nu_i = \nu_{m-1}(x_i)} =$$
$$= \tilde{\tilde{\nu}}_{im} = \begin{cases} \lambda_i(1-p_i) \cdot \frac{1-p_i+p_i e^{\lambda_i} \cdot (1-\lambda_i)}{(p_i+(1-p_i)e^{\lambda_i})^2}, & y_i = 0, \\ \lambda_i, & y_i > 0. \end{cases}$$

The formula (4) for "optimal" response in p-tree terminal node will look like ($n(R_{jm})$ is the count of training samples in node R_{jm}):

$$\gamma^1_{lm} = \begin{cases} \sum_{x_i \in R_{jm}} \tilde{\mu}_{im} / \sum_{x_i \in R_{jm}} \tilde{\tilde{\mu}}_{im}, \sum_{x_i \in R_{jm}} \tilde{\tilde{\mu}}_{im} > \varepsilon = 10^{-6}, \\ \sum_{x_i \in R_{jm}} \tilde{\mu}_{im} / n(R_{jm}), \text{otherwise.} \end{cases}$$

same for λ-tree :

$$\gamma^2_{lm} = \begin{cases} \sum_{x_i \in R_{jm}} \tilde{\nu}_{im} / \sum_{x_i \in R_{jm}} \tilde{\tilde{\nu}}_{im}, \sum_{x_i \in R_{jm}} \tilde{\tilde{\nu}}_{im} > \varepsilon, \\ \sum_{x_i \in R_{jm}} \tilde{\nu}_{im} / n(R_{jm}), \text{otherwise.} \end{cases}$$

There are several tricks that we use to improve the numeric stability of the algorithm. To prevent μ_i, ν_i from causing numerical overflow or underflow we simply threshold them by a reasonable constant ($log(FLT\_MAX/2)$ for example). We also adopted influence trimming strategy to prevent very small Hessian by absolute value in a tree node. We found that one cannot remove samples with negative loss function second derivative because it can harm severely the performance of the algorithm. However one can trim samples with

second derivative small by absolute value in p-tree. So we do no influence trimming for λ-tree (as small absolute value of the second derivative of the loss function is not likely to happen there), and do influence trimming with weights $w_i = p_i(1 - p_i)$ for p-tree in the same way as it is described earlier for two-class logistic regression.

3. Evaluation

First we validate our algorithm and compare its performance with our implementation of ZIP tree on two artificial data sets. Both data sets are generated from a known model for ZIP distribution parameters (p, λ) with a small amount of random noise added, i.e

$$p = p(x_1, x_2) \cdot (1 + \varepsilon \cdot u_1), u_1 \in U(-1, 1),$$
$$\lambda = \lambda(x_1, x_2) \cdot (1 + \varepsilon \cdot u_2), u_2 \in U(-1, 1).$$

Then response value y_i is generated from ZIP distribution with parameters $(p_i = p(x_{1i}, x_{2i}), \lambda_i = \lambda(x_{1i}, x_{2i}))$. In all three experiments three values for noise level $\varepsilon = 0, 0.2, 0.5$ are used.

The first data set uses linear model for (p, λ) :

$$p = 0.2 + 0.6 \cdot (0.3x_1 + 0.7x_2),$$
$$\lambda = 1.5 + 7 \cdot (0.6x_1 + 0.4x_2).$$

The second data set uses more complex highly nonlinear model

$$\text{logit}(p) = 2\sin(20x_1) + 3x_2 \cdot (x_2 - 0.5),$$
$$\log(\lambda) = \sin(30x_1) + 3x_2.$$

For each model we report the base error (error for the best constant model), training error, and cross-validation error (5-fold), where error is average negative ZIP log-likelihood, and average absolute difference (on the training set) between "true" and "predicted" parameters (p, λ), we also report average relative difference for λ parameter. Three last numbers show how well ZIP distribution parameters are approximated by the model. In all experiments the model complexity (which is the pruning step for the tree and the number of iterations for GBT) is selected using best CV error. Size of all data sets is 10000 samples.

For artificial data sets, the following parameters are used :
ZIP TREE : tree_depth = 6, min_split = 50, min_bucket = 20.
ZIP GBT : nit = 1000, tree_depth = 3, min_split = 400, min_bucket = 200, shrinkage = 0.01, infl_trimming = 0.1.
Here tree_depth is a maximum tree depth (node is not split if it is at the specified depth), min_split is a minimum size of the node that will be split (if it has less observations it is NOT split), min_bucket is a minimum size of the terminal node (split is not accepted if it creates a terminal node with smaller size), nit = is a maximum number of iterations for an ensemble, shrinkage is the ν parameter (regularization) on step 6 of Algorithm 1, infl_trimming is α threshold for influence trimming.

Base error column in the following table shows negative ZIP log-likelihood for the best constant model, train and CV-error are train and 5-fold cross-validation errors (negative ZIP log-likelihood also), δp is average absolute difference in predicted

p parameter ($\delta p = \sum_{i=1}^{n} |p(x_{1i}, x_{2i}) - \hat{p}(x_{1i}, x_{2i})|/n$ where $\hat{p}(x_1, x_2)$ is prediction from the model), $\delta\lambda$ is an average absolute difference in predicted λ parameter ($\delta\lambda = \sum_{i=1}^{n} |\lambda(x_{1i}, x_{2i}) - \hat{\lambda}(x_{1i}, x_{2i})|/n$), $\delta\lambda_{rel}$ is an average relative difference in predicted λ parameter ($\delta\lambda_{rel} = \sum_{i=1}^{n} |1 - \hat{\lambda}(x_{1i}, x_{2i})/\lambda(x_{1i}, x_{2i})|/n$).

Table 1. Comparison of ZIP tree and ZIP GBT on two artificial data sets.

Data	Noise(ε)	Base error	Model	Train error	CV error	δp	$\delta\lambda$	$\delta\lambda_{rel}$	Best step
LINEAR	0	1.801	TREE	1.663	1.690	0.043	0.355	0.074	16
			GBT	1.653	1.675	0.027	0.182	0.038	413
	0.2	1.859	TREE	1.707	1.736	0.043	0.416	0.092	15
			GBT	1.702	1.721	0.032	0.179	0.040	284
	0.5	1.873	TREE	1.744	1.775	0.040	0.441	0.093	13
			GBT	1.733	1.754	0.032	0.234	0.049	319
NON-	0	2.920	TREE	1.535	1.675	0.146	3.105	0.403	49
LINEAR			GBT	1.360	1.413	0.058	1.844	0.255	999
	0.2	3.037	TREE	1.594	1.735	0.156	3.027	0.423	35
			GBT	1.425	1.492	0.064	1.810	0.247	999
	0.5	3.310	TREE	1.774	1.925	0.154	3.112	0.394	42
			GBT	1.577	1.663	0.073	1.812	0.253	998

This table shows that GBT is always superior to a single tree in terms of train error, CV error and ZIP distribution parameters prediction error. One can see that over-fitting (difference between CV and train errors) is much smaller for GBT, especially for bigger noise levels and more complex models.

Then we compared performance of ZIP GBT to ZIP tree on two public available real-life data sets. The first one is SOLDER, which is a part of rpart R free package, the second is DMFT (decayed, missing and filled teeth) data set used in [6]. On SOLDER data set, ZIP GBT is much better than a single tree in terms of cross-validated log-likelihood, on DMFT GBT is only slightly better. Parameters of both algorithms were adjusted manually to minimize cross-validation error :
ZIP TREE : tree_depth = 6, min_split = 15, min_bucket = 10.
ZIP GBT : nit = 1000, tree depth = 3, min_split = 30, min_bucket = 20, shrinkage = 0.02(0.005 for DMFT), infl_trimming = 0.1.

It can be seen that GBT has much smaller CV error on SOLDER data set and a little smaller on DMFT data set.

Conclusion

This article introduces gradient boosting model for small-count regression problem, where response is assumed to follow ZIP distribution. This model uses gradient tree boosting concepts introduced by Friedman for regression and classification and extends

Table 2. Comparison of ZIP tree and ZIP GBT on real-life data.

Data	Base error	Model	Train error	CV error	Best step
SOLDER	4.464	TREE	2.493	2.714	9
SOLDER	4.464	GBT	1.510	1.818	765
DMFT	1.789	TREE	1.525	1.577	8
DMFT	1.789	GBT	1.499	1.564	660

them to ZIP model. It is shown that the algorithm performance (both in terms of log-likelihood value and prediction of ZIP distribution parameters as function of inputs) is superior to the performance of ZIP tree.

The algorithm can be adapted to different problems using different link functions. Further analysis of the algorithm performance and comparison to other small count data models on large real-life data that comes from Intel manufacturing processes is of great interest.

References

1. *D. Lambert.* Zero-inflated Poisson regression with an application to defects in manufacturing// Technometrics, 34(1), pp. 1-14, 1992.
2. *S. Lee and S. Jin.* Decision tree approaches for zero-inflated count data// Journal of applied statistics, 33(8), pp. 853-865, 2006.
3. *C. Li, J. Lu, and J. Park.* Multivariate zero-inflated Poisson models and their applications// Technometrics, 41(1), pp. 29-38, 1999.
4. *Hastie, R. Tibshirani, and J. Friedman.* The Elements of Statistical Learning. Springer, New York , 2001.
5. *D. Bohning, E. Dietz, P. Schlattman, L. Mendonca, and U. Kirchner.* Testing parameter of the power series distribution of a zero inflated power series model// Statistical Methodology, 4, pp. 393-406, 2007.
6. *D, Bohning, E, Dietz ,P, Schlattman, L, Medonca and U,Kirchner.* The zero-inflated Poisson model and the decayed, missing and filled teeth index in dental epidemiology// J.R.Statist.Soc A(1999) 162, part 2, pp.195-209.
7. *L. Breiman.* Bagging predictors// Machine Learning, 24, pp. 123-140, 1996.
8. *L. Breiman.* Random forests// Machine Learning, 45, pp. 5-32, 2001.
9. *L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone.* Classification and regression trees. Chapman and Hall/CRC, New York, 1998.
10. *Y.B. Cheung.* Zero-inflated models for regression analysis of count data: a study of growth and development// Statistics in Medicine, 21, pp. 1461-1469, 2002.
11. *M. Chiogna and C. Gaetan.* Semiparametric zero-inflated poisson models with application to animal abundance studies// Environmetrics, 18, pp. 303-314, 2007.
12. *F. Famoye and K.P. Singh.* Zero-inflated generalized Poisson regression model with an application to domestic violence data.// Journal of Data Science, 4, pp. 117-130, 2006.
13. *C. Hsu.* A weighted zero-inflated poisson model for estimation of recurrence of adenomas// Statistical Methods in Medical Research, 16, pp. 155-166, 2007.
14. *K. Hur, D. Hedeker, W. Henderson, S. Khuri, and J. Daley.* Modeling clustered count data with excess zeros in health care outcomes research// Health Services and Outcomes Research Methodology, 3, pp. 5-20, 2002.

15. *C. Li, J. Lu, and J. Park.* Multivariate zero-inflated Poisson models and their applications// Technometrics, 41(1), pp. 29-38, 1999.
16. *C.S. Li, J.C. Lu, J. Park, K. Kim, P.A. Brinkley, and J.P. Peterson.* Multivariate zero-inflated Poisson models and their applications. // Technometrics, 41(1), pp. 29-38, 1999.
17. *R. Ramis Prieto, J. Garcia-Perez, M. Pollan, N. Aragones, B. Perez-Gomez, and G. Lopez-Abente.* Modelling of municipal mortality due to haematological neoplasias in Spain // Journal of epidemiology and community health, 61(2), pp. 165-171, 2007.
18. *P. Wang.* Markov zero-inflated poisson regression models for a time series of counts with excess Zeros// Journal of Applied Statistics, 28(5), pp. 623-632, 2001.
19. *K. K. Yau and A.H. Lee.* Zero-inflated poisson regression with random effects to evaluate an occupational injury prevention programme// Statistics in Medicine, 20(19), pp. 2907-2920, 2001.
20. *J.H. Friedman.* Greedy function approximation : a gradient boosting machine// Technical report, Dept. of Statistics, Stanford University, 1999.
21. *J.H. Friedman.* Stochastic gradient boosting// Computational Statistics and Data Analysis, 38(4), pp. 367-378, 2002.

Статья поступила в редакцию 25.04.2008