First of all, we want to thank every reviewer for valuable notes and comments. We will do our best to accommodate all of them to reorganize the paper, make it more clear and thoughtful.

To Reviewer 1.

In particular, we will discuss tuning time of the algorithms. Since GOSS and MVS have one additional hyperparameter (large gradients size for GOSS and λ for MVS), the tuning times for them are approximately the same. We also found a parameter-free formula for λ (see the details in the answer to Reviewer 2).

To Reviewer 2.

– I assume you can incorporate the hessian the same way... Any reasons why you didn't do that?

Our paper is based on a standard GBDT score function (as, e.g., in [21]). Recently, we tested our method on the score function with hessians. The algorithm is easy to derive from our paper, when you replace a leaf size in Eq. 6 with sum of hessians in the leaf. The solution remains the same except for replacing $\sqrt{g_i^2 + \lambda}$ with $\sqrt{g_i^2 + \lambda h_i^2}$ in Theorem 2. Performance of this hessian-based sampling is even better (see Table 1), and we will add these results to the paper.

– replacing values on the leaves c_l with a constant ... The leaf value depends on the gradients of the instances that fall into that leaf ...

Since the structure of the tree is not known in advance we propose replacing c_l^2 with some constant upper bound. Recently, we also found a parameter-free formula for λ, which achieves near-optimal results (and even better than ones obtained via cross-validation, see Table 1, MVS Adaptive): we approximate c_l^2 by squared mean of all gradient's absolute values. We will add this to the paper.

– Line 24 ... Most implementations don't consider every possible value for each split.

We will fix this misleading statement, sorry.

– i am not sure how u go from 7 to 8. c_l can be either negative or positive...

The middle term (the covariance) in (7) is bounded by the sum of variances (Line 163). Since the variance is not negative the inequality is true for arbitrary c_l.

– in Table 3 or 4 you should also report complexities of methods and how many actual instances were sampled...

The algorithms differ only in their sampling stages. As it mentioned in section 4.3, all algorithms have $O(n)$ complexity here, so the complexities of all algorithms with same sampling ratio are the same. Unfortunately, we are not sure what do you mean by "sampling rate does not tell the whole story". Please, specify this issue, we will try to fix it.

To Reviewer 4.

– It would be more interesting to see how MVS affects the training time of large datasets.

We have experiments with large datasets and we will necessarily add them to the paper. The results are consistent with statements from paper, e.g. on Higgs dataset we have -15% learning time for MVS and -10% learning time for GOSS.

– it seems that the authors assume the tree structures are the same with subsampled and full training data...

Yes, paper derivations are conditioning on fixed previous splits of the tree. We will add this assumption.

– it would be better if the MVS could be merged into LightGBM repo, I am looking forward to using it.

The source code is available as a fork of LightGBM repo on github (see line 241). The github link is anonymized, and we think it will be possible to merge the code after review period ends.

– Line 175 to 186 is a little difficult for me to understand...

The idea is this: if only the first addend is optimized, then trees from close iterations will be trained on approximately the same instances (gradients do not change much), so this leads to higher correlation between them and as a result to higher variance of the ensemble. The second addend is minimized when all probabilities are the same.

Sample rate	0.02	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.5
MVS	+13.96%	+8.21%	+4.60%	+2.39%	+1.23%	+0.38%	+0.00%	-0.17%	-0.24%	-0.44%
MVS adaptive	+13.89%	+7.57%	+3.87%	+1.72%	+0.64%	+0.16%	-0.06%	-0.19%	-0.29%	-0.50%
MVS with hessians	**+13.72%**	**+7.47%**	**+3.71%**	**+1.70%**	**+0.55%**	**-0.03%**	**-0.07 %**	**-0.28%**	**-0.32%**	**-0.51%**

Table 1: Relative error change, average over datasets