# Improving Repeated Labeling for Crowdsourced Data Annotation

**Sergiu Goschin** SGOSCHIN@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

**Chris Mesterharm** MESTERHA@CS.RUTGERS.EDU
Applied Communication Sciences, One Telcordia Drive, Piscataway, NJ, USA

**Haym Hirsh** HIRSH@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

## 1 Introduction

The introduction of crowdsourcing platforms such as Amazon Mechanical Turk (AMT, mturk.com) has made it possible to harness cheap, albeit sometimes unreliable, human labor in a range of data annotation tasks. The most common approach for improving the quality of annotations is to seek multiple annotations for each item and use majority vote to assign a final output (Sheng et al., 2008). However, each annotation costs money, and this work targets the question of how to allocate annotation tasks to workers so as to boost annotation accuracy in a cost-sensitive fashion. We address two related problems in data annotation. The first concerns tasks in which the data annotation is itself the task of interest — for example, tasks that assess whether Web pages contain objectionable content. The second problem is to seek improved outcomes of classifier learning when given access to a resource such as AMT for labeling data.

Unlike other work that models individual worker ability, we consider the common case when only a small number of workers label more than one item and thus the vast majority of workers do one task and are never seen again. Our approach is thus to view AMT as a black-box oracle that takes examples as input, generates one of a fixed set of labels as output, and then corrupts the output by some random noise process. For the purpose of this work we assume that each item is assigned a binary label, 0 or 1. Rather than giving an item to some fixed number of workers and taking their majority vote, we instead present an approach that iteratively seeks labels for an item until the number of labels of one class exceeds by a given class-specific amount the number of labels given to the other. Our first contribution is to show, based on an insight from random walks, that this beat-by-k strategy performs better than the majority vote approach, in that the expected number of labels needed to reach a given accuracy threshold is less than that for the majority vote approach. Our second contribution is to give PAC-learning bounds on classification accuracy using the beat-by-k strategy to label data in the Random Classification Noise model (Angluin & Laird, 1988). We conclude with experimental results on the MNIST (Lecun et al., 1998) dataset that support our claims empirically.

## 2 Improving Label Accuracy

**Setup and Notation.** We assume that every time a new label is requested for a fixed instance the correct label is flipped with probability $p \in (0, \frac{1}{2})$ - the noise parameter. The underlying distribution over possible instances is skewed with parameter $s \in (0, \frac{1}{2}]$. Thus the probability of sampling a 0-instance from the underlying distribution is $s$ while the probability of sampling a 1-instance is $1 - s$.

**Majority Vote.** We use two baseline methods for repeated labeling that each take an instance and a budget $m$ ($m$ odd). **Majority Vote** (MV) obtains $m$ labels for the instance and returns the most frequent label among them. **Early Majority Vote** (EMV)) is identical to MV, but seeks labels incrementally and stops once either label appears at least $\lceil \frac{m}{2} \rceil$ times.

**Beat-by-k.** Let $\Delta_0^t = \#0\text{-labels} - \#1\text{-labels}$ be the difference between the number of 0-labels and the number of 1-labels for an instance after $t > 0$ labels have been obtained. We similarly define $\Delta_1^t$. The Beat-by-k strategy takes two parameters $k_0$ and $k_1$ and repeatedly seeks labels for a given instance until the first $t$ at which either $\Delta_0^t = k_0$, in which case it returns the label 0, or $\Delta_1^t = k_1$, when it returns a 1. $\tau(k_0, k_1)$ represents the number of labels obtained for an instance using the Beat-by-k strategy; a simple Markov Chain argument shows that $\tau$ is finite with probability 1 for fixed $k_0$ and $k_1$. We thus designate the expected cost of a $(k_0, k_1)$-strategy by $\tau$. The accuracy of the Beat-by-k strategy can then be described as $f(k_0, k_1) = P((k_0, k_1)\text{-strategy is accurate}) = P(\Delta_1^\tau = k_1|1)P(1) + P(\Delta_0^\tau = k_0|0)P(0) = sP(\Delta_1^\tau = k_1|1) + (1 - s)P(\Delta_0^\tau = k_0|0)$ where 1 and 0 in the formulas stand for the correct label of the instance.

We view the process of obtaining labels as a random walk on the number line with two absorbing barriers (at $-k_0$ and at $k_1$): every time a 0-label is sampled, we take a $-1$ step, otherwise we take a $+1$ step. Let $S_t$ be the sum of these steps until time $t$. $S_\tau$ is thus either $-k_0$ or $k_1$ (as this is the only way the random walk stops). The problem of assigning a label is thus equivalent to a random walk, but with two possible (and hidden to the algorithm) probability distributions—one for a 0-instance and the other for the 1-instance—which bias it in opposite ways.

Let $P$ denote the probability distribution over labels induced by a 1-instance and $N$ the probability distribution induced by a 0-instance. The above formula then becomes: $f(k_0, k_1) = sP_P(S_\tau = k_1) + (1 - s)P_N(S_\tau = k_0)$. If $t = \frac{1-p}{p}$, theorems for the probability of a random walk hitting the barriers (Feller, 1968) imply that $P_P(S_\tau = k_1) = \frac{t^{k_1}(t^{k_0}-1)}{t^{k_0+k_1}-1} P_N(S_\tau = k_0) = \frac{t^{k_0}(t^{k_1}-1)}{t^{k_0+k_1}-1}$ and thus: $f(k_0, k_1) = \frac{t^{k_0}(t^{k_1}-1)+s(t^{k_0}-t^{k_1})}{t^{k_0+k_1}-1}$.

Using (Feller, 1968) we can now derive a formula for the expected value of $\tau$ when the true label is 1:$\tau_1(k_0, k_1) = \tau(k_0, k_1, \text{label} = 1) = \frac{E_P[S_\tau]}{1-2p} = \frac{k_1 t^{k_1}(t^{k_0}-1)-k_0(t^{k_1}-1)}{(1-2p)(t^{k_0+k_1}-1)}$. Symmetrically then: $\tau_0(k_0, k_1) = \frac{k_0 t^{k_0}(t^{k_1}-1)-k_1(t^{k_0}-1)}{(1-2p)(t^{k_0+k_1}-1)}$, and thus $\tau(k_0, k_1) = \tau_1(k_0, k_1) + s(\tau_0(k_0, k_1) - \tau_1(k_0, k_1)$

Note that when there is no skew ($s = \frac{1}{2}$), $k_0$ and $k_1$ play a symmetrical role and thus the strategies are satisfyingly parameterized by a single parameter $k = k_0 = k_1$ with $f(k) = \frac{t^k}{t^k+1}$ and $\tau(k) = \frac{k(t^k-1)}{(1-2p)(t^k+1)}$, where both $f$ and $\tau$ are independent of $s$.

**Key Results**. In the full paper we will present our theorem that shows that for any fixed instance, **Beat-by-k** needs a smaller expected number of queries as compared to either **MV** and **EMV** strategies to reach a fixed desired accuracy threshold (we will also confirm this fact empirically in the next section), and, further, that **Beat-by-k** is an optimal strategy given our assumptions).

## 3 PAC-Learning Using Beat-by-k

**ERM algorithm**. As is typical in PAC learning, our results are depending on the parameters $\epsilon$ (accuracy), $\delta$ (failure probability), $\mathcal{F}$ (a hypothesis class), $\eta$ (random classification noise - the same for both instance labels). We state our results here for a finite $\mathcal{F}$, but (to be discussed in the full paper) they naturally extend to infinite concept classes by replacing the dependency on the log of the size of the class by a dependency on the VC-dimension of $\mathcal{F}$. The main result we are going to use is Theorem 2 from (Angluin & Laird, 1988) which states that the number of instances $m$ (and implicitly the total cost $C = m$) that need to be labeled so that an Empirical Risk Minimization (ERM) Learner will PAC-Learn a target hypothesis $f \in \mathcal{F}$ in the random noise classification model has to be at least: $C_{PAC} = m_{PAC} = \frac{2}{\epsilon^2(1-2\eta)^2} ln(\frac{2|\mathcal{F}|}{\delta})$. This is the result you would get if $k_0 = k_1 = 1$ in the **Beat-by-k** labeling strategy and $\eta = p$.

**Key Results**. We exploit the fact that repeated labeling permits a learner to decrease $\eta$ by sampling a label more than once — at the cost of possibly increasing the overall expected cost. We use a simple observation: if any instance is sampled until $k$ more 0's or 1's are seen, the resulting $\eta(k_0, k_1) = 1 - f(k_0, k_1)$ noise level will decrease (simply because the accuracy $f(k_0, k_1)$ will increase as $k$ increases). This in turn will lead to a smaller number of instances $m(k_0, k_1)$ that must be obtained to get the desired accuracy, while possibly increasing the expected cost of the labeling process: $E[C(k_0, k_1)]$. In the full paper we will show that you can choose $k_0$ and $k_1$ so as to minimize $h(k_0, k_1) = E[C(k_0, k_1)]$. So $h(k_0, k_1) = m(k_0, k_1)\tau(k_0, k_1) = m_{PAC}(1 - 2p)^2 \frac{\tau(k_0,k_1)}{(2f(k_0,k_1)-1)^2}$.

## 4 Experimental Results

We use Alma$_2$ (Gentile, 2001) as the learning algorithms for our experiments. Alma$_2$ is similar to the Perceptron algorithm (Block, 1962), but adjusts its updates as it learns in a way that tends to improve generalization performance. We furthermore make two modifications to Alma$_2$ that have been shown to improve its performance when used in a batch setting (Mesterharm, 2007). The first modification is a special type of local instance recycling that limits the number of times an instance can be used in an update. The second modification is based on the voting technique of Schapire and Freund (Freund & Schapire, 1998).

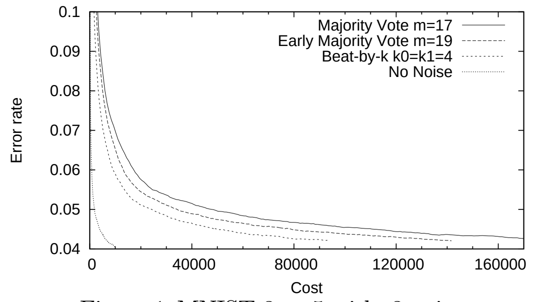

*Figure 1.* MNIST 3 vs 5 with .3 noise

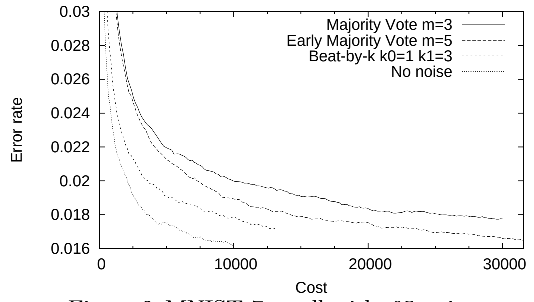

*Figure 2.* MNIST 7 vs all with .05 noise

We use the MNIST dataset of handwritten digits for our experiments (Lecun et al., 1998). Each instance is a 28 by 28 grid of pixels, resulting in a vector of 728 values, with each value reflecting the 8-bit grayscale of one pixel. 10000 instances are used for training and 3454 instances are used for testing. Each experiment is averaged over 50 permutations of the data.

Figure 1 uses data for the digits three and five, with minimal class skew of $s = .47$ and with a relatively high noise rate $p = .3$. Consistent with our theoretical results, label resampling is effective in cases of the relatively large noise rates often observed when using AMT. The graph compares best-case scenarios, in that each curve reflects the best parameter settings for each the three algorithms when $p = .3$. The x-axis in the graph is a measure of number of labels obtained; the different curves terminate at different points depending on how many labels each method seeks per instance. Figure 2 uses data for the digit seven versus data for all other digits, with class skew $s = .104$ and with a relatively low noise rate $p = .05$. There are again consistent with our theoretical results, with the random walk strategy being beneficial when the skew is far from .5.

**Ongoing work.** We are currently in the process of gathering data from running the algorithms with Amazon Turk as an oracle. We are also working on extending the **Beat-by-k** strategy to settings where $s$ is unknown and $p$ is possibly varying and unknown.

## References

Angluin, Dana and Laird, Philip. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988. ISSN 0885-6125.

Block, H. D. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34(1):123–135, 1962.

Feller, William. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.

Freund, Yoav and Schapire, Robert E. Large margin classification using the perceptron algorithm. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998.

Gentile, Claudio. A new approximate maximal margin classification algorithm. *Machine Learning*, 2:213–242, 2001.

Lecun, Yann, Bottou, Lon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Mesterharm, Chris. *Improving On-line Learning*. PhD thesis, Computer Science, Rutgers, The State University of New Jersey, 2007.

Sheng, Victor S., Provost, Foster, and Ipeirotis, Panagiotis G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD KDD*, KDD '08, pp. 614–622, New York, NY, USA, 2008. ACM.