Large-Scale Learning with Less RAM via Randomization

Daniel Golovin DGG@GOOGLE.COM
D. Sculley DSCULLEY@GOOGLE.COM
H. Brendan McMahan MCMAHAN@GOOGLE.COM
Michael Young MWYOUNG@GOOGLE.COM
Google, Inc., Pittsburgh, PA, and Seattle, WA

Abstract

We reduce the memory footprint of popular large-scale online learning methods by projecting our weight vector onto a coarse discrete set using randomized rounding. Compared to standard 32-bit float encodings, this reduces RAM usage by more than 50% during training and by up to 95% when making predictions from a fixed model, with almost no loss in accuracy. We also show that randomized counting can be used to implement per-coordinate learning rates, improving model quality with little additional RAM. We prove these memory-saving methods achieve regret guarantees similar to their exact variants. Empirical evaluation confirms excellent performance, dominating standard approaches across memory versus accuracy tradeoffs.

1. Introduction

As the growth of machine learning data sets continues to accelerate, available machine memory (RAM) is an increasingly important constraint. This is true for training massive-scale distributed learning systems, such as those used for predicting ad click through rates (CTR) for sponsored search (Richardson et al., 2007; Craswell et al., 2008; Bilenko & Richardson, 2011; Streeter & McMahan, 2010) or for filtering email spam at scale (Goodman et al., 2007). Minimizing RAM use is also important on a single machine if we wish to utilize the limited memory of a fast GPU processor, or to simply use fast L1-cache more effectively. After training, memory cost remains a key consideration at prediction time as real-world models are often replicated to multiple machines to minimize prediction latency.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

Efficient learning at peta-scale is commonly achieved by online gradient descent (OGD) (Zinkevich, 2003) or stochastic gradient descent (SGD), (e.g., Bottou & Bousquet, 2008), in which many tiny steps are accumulated in a weight vector $\beta \in \mathbb{R}^d$. For large-scale learning, storing β can consume considerable RAM, especially when datasets far exceed memory capacity and examples are streamed from network or disk.

Our goal is to reduce the memory needed to store β. Standard implementations store coefficients in single precision floating-point representation, using 32 bits per value. This provides fine-grained precision needed to accumulate these tiny steps with minimal roundoff error, but has a dynamic range that far exceeds the needs of practical machine learning (see Figure 1).

We use coefficient representations that have more limited precision and dynamic range, allowing values to be stored cheaply. This coarse grid does *not* provide enough resolution to accumulate gradient steps without error, as the grid spacing may be larger than the updates. But we can obtain a provable safety guarantee through a suitable OGD algorithm that uses randomized rounding to project its coefficients onto the grid each round. The precision of the grid used on each round may be fixed in advance or changed adaptively as learning progresses. At prediction time, more aggressive rounding is possible because errors no longer accumulate.

Online learning on large feature spaces where some features occur very frequently and others are rare often benefits from per-coordinate learning rates, but this requires an additional 32-bit count to be stored for each coordinate. In the spirit of randomized rounding, we limit the memory footprint of this strategy by using an 8-bit randomized counter for each coordinate based on a variant of Morris's algorithm (1978). We show the resulting regret bounds are only slightly worse than the exact counting variant (Theorem 3.3), and empirical results show negligible added loss.

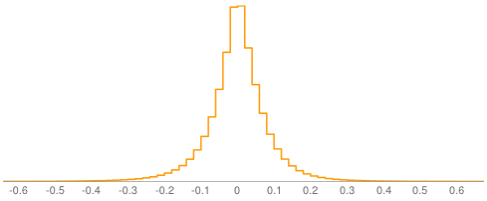

Figure 1. Histogram of coefficients in a typical large-scale linear model trained from real data. Values are tightly grouped near zero; a large dynamic range is superfluous.

Contributions This paper gives the following theoretical and empirical results:

1. Using a pre-determined fixed-point representation of coefficient values reduces cost from 32 to 16 bits per value, at the cost of a small linear regret term.
2. The cost of a per-coordinate learning rate schedule can be reduced from 32 to 8 bits per coordinate using a randomized counting scheme.
3. Using an adaptive per-coordinate coarse representation of coefficient values reduces memory cost further and yields a no–regret algorithm.
4. Variable-width encoding at prediction time allows coefficients to be encoded even more compactly (less than 2 bits per value in experiments) with negligible added loss.

Approaches 1 and 2 are particularly attractive, as they require only small code changes and use negligible additional CPU time. Approaches 3 and 4 require more sophisticated data structures.

2. Related Work

In addition to the sources already referenced, related work has been done in several areas.

Smaller Models A classic approach to reducing memory usage is to encourage sparsity, for example via the Lasso (Tibshirani, 1996) variant of least-squares regression, and the more general application of L_1 regularizers (Duchi et al., 2008; Langford et al., 2009; Xiao, 2009; McMahan, 2011). A more recent trend has been to reduce memory cost via the use of feature hashing (Weinberger et al., 2009). Both families of approaches are effective. The coarse encoding schemes reported here may be used in conjunction with these methods to give further reductions in memory usage.

Randomized Rounding Randomized rounding schemes have been widely used in numerical computing and algorithm design (Raghavan & Tompson, 1987). Recently, the related technique of randomized counting has enabled compact language models (Van Durme & Lall, 2009). To our knowledge, this paper gives the first algorithms and analysis for online learning with randomized rounding and counting.

Per-Coordinate Learning Rates Duchi et al. (2010) and McMahan & Streeter (2010) demonstrated that per-coordinate adaptive regularization (*i.e.*, adaptive learning rates) can greatly boost prediction accuracy. The intuition is to let the learning rate for common features decrease quickly, while keeping the learning rate high for rare features. This adaptivity increases RAM cost by requiring an additional statistic to be stored for each coordinate, most often as an additional 32-bit integer. Our approach reduces this cost by using an 8-bit randomized counter instead, using a variant of Morris's algorithm (Morris, 1978).

3. Learning with Randomized Rounding and Probabilistic Counting

For concreteness, we focus on logistic regression with binary feature vectors $x \in \{0,1\}^d$ and labels $y \in \{0,1\}$. The model has coefficients $\beta \in \mathbb{R}^d$, and gives predictions $p_\beta(x) \equiv \sigma(\beta \cdot x)$, where $\sigma(z) \equiv 1/(1+e^{-z})$ is the logistic function. Logistic regression finds the model that minimizes the logistic–loss $\mathcal{L}$. Given a labeled example (x, y) the logistic–loss is

$$\mathcal{L}(x, y; \beta) \equiv -y \log(p_\beta(x)) - (1-y) \log(1 - p_\beta(x))$$

where we take $0 \log 0 = 0$. Here, we take log to be the natural logarithm. We define $\|x\|_p$ as the ℓ_p norm of a vector x; when the subscript p is omitted, the ℓ_2 norm is implied. We use the compressed summation notation $g_{1:t} \equiv \sum_{s=1}^{t} g_s$ for scalars, and similarly $f_{1:t}(x) \equiv \sum_{s=1}^{t} f_s(x)$ for functions.

The basic algorithm we propose and analyze is a variant of online gradient descent (OGD) that stores coefficients β in a limited precision format using a discrete set $(\epsilon\mathbb{Z})^d$. For each OGD update, we compute each new coefficient value in 64-bit floating point representation and then use randomized rounding to project the updated value back to the coarser representation.

A useful representation for the discrete set $(\epsilon\mathbb{Z})^d$ is the `Qn.m` fixed-point representation. This uses `n` bits for the integral part of the value, and `m` bits for the fractional part. Adding in a sign bit results in a total of $K = \mathtt{n} + \mathtt{m} + 1$ bits per value. The value `m` may be fixed in advance, or set adaptively as described below. We use the method RandomRound from Algorithm 1 to project values onto this encoding.

The added CPU cost of fixed-point encoding and randomized rounding is low. Typically K is chosen to correspond to a machine integer (say $K = 8$ or 16),

Algorithm 1 OGD-Rand-1d

input: feasible set $\mathcal{F} = [-R, R]$, learning rate schedule η_t, resolution schedule ϵ_t
define fun Project $(\beta) = \max(-R, \min(\beta, R))$
Initialize $\hat{\beta}_1 = 0$
for t=1, ..., T **do**
  Play the point $\hat{\beta}_t$, observe g_t
  $\beta_{t+1} = \text{Project}\left(\hat{\beta}_t - \eta_t g_t\right)$
  $\hat{\beta}_{t+1} \leftarrow \text{RandomRound}(\beta_{t+1}, \epsilon_t)$

function RandomRound(β, ϵ)
  $a \leftarrow \epsilon \left\lfloor \frac{\beta}{\epsilon} \right\rfloor$; $b \leftarrow \epsilon \left\lceil \frac{\beta}{\epsilon} \right\rceil$
  return $\begin{cases} b & \text{with prob.} \quad (\beta - a)/\epsilon \\ a & \text{otherwise} \end{cases}$

so converting back to a floating point representations requires a single integer-float multiplication (by $\epsilon = 2^{-m}$). Randomized rounding requires a call to a pseudo-random number generator, which may be done in 18-20 flops. Overall, the added CPU overhead is negligible, especially as many large-scale learning methods are I/O bound reading from disk or network rather than CPU bound.

3.1. Regret Bounds for Randomized Rounding

We now prove theoretical guarantees (in the form of upper bounds on regret) for a variant of OGD that uses randomized rounding on an adaptive grid as well as per-coordinate learning rates. (These bounds can also be applied to a fixed grid). We use the standard definition

$$\text{Regret} \equiv \sum_{t=1}^{T} f_t(\hat{\beta}_t) - \arg\min_{\beta^* \in \mathcal{F}} \sum_{t=1}^{T} f_t(\beta^*)$$

given a sequence of convex loss functions f_t. Here the $\hat{\beta}_t$ our algorithm plays are random variables, and since we allow the adversary to adapt based on the previously observed $\hat{\beta}_t$, the f_t and post-hoc optimal β^* are also random variables. We prove bounds on *expected* regret, where the expectation is with respect to the randomization used by our algorithms (high-probability bounds are also possible). We consider regret with respect to the best model in the *non-discretized* comparison class $\mathcal{F} = [-R, R]^d$.

We follow the usual reduction from convex to linear functions introduced by Zinkevich (2003); see also Shalev-Shwartz (2012, Sec. 2.4). Further, since we consider the hyper-rectangle feasible set $\mathcal{F} = [-R, R]^d$, the linear problem decomposes into n independent one-dimensional problems.[1] In this setting, we consider OGD with randomized rounding to an adaptive grid of resolution ϵ_t on round t, and an adaptive learning rate η_t. We then run one copy of this algorithm for each coordinate of the original convex problem, implying that we can choose the η_t and ϵ_t schedules appropriately for each coordinate. For simplicity, we assume the ϵ_t resolutions are chosen so that $-R$ and $+R$ are always gridpoints. Algorithm 1 gives the one-dimensional version, which is run independently on each coordinate (with a different learning rate and discretization schedule) in Algorithm 2. The core result is a regret bound for Algorithm 1 (omitted proofs can be found in the Appendix):

Theorem 3.1. *Consider running Algorithm 1 with adaptive non-increasing learning-rate schedule η_t, and discretization schedule ϵ_t such that $\epsilon_t \leq \gamma \eta_t$ for a constant $\gamma > 0$. Then, against any sequence of gradients $g_1, \ldots, g_T$ (possibly selected by an adaptive adversary) with $|g_t| \leq G$, against any comparator point $\beta^* \in [-R, R]$, we have*

$$\mathbf{E}[\text{Regret}(\beta^*)] \leq \frac{(2R)^2}{2\eta_T} + \frac{1}{2}(G^2 + \gamma^2)\eta_{1:T} + \gamma R\sqrt{T}.$$

By choosing γ sufficiently small, we obtain an expected regret bound that is indistinguishable from the non-rounded version (which is obtained by taking $\gamma = 0$). In practice, we find simply choosing $\gamma = 1$ yields excellent results. With some care in the choice of norms used, it is straightforward to extend the above result to d dimensions. Applying the above algorithm on a per-coordinate basis yields the following guarantee:

Corollary 3.2. *Consider running Algorithm 2 on the feasible set $\mathcal{F} = [-R, R]^d$, which in turn runs Algorithm 1 on each coordinate. We use per-coordinate learning rates $\eta_{t,i} = \alpha/\sqrt{\tau_{t,i}}$ with $\alpha = \sqrt{2}R/\sqrt{G^2 + \gamma^2}$, where $\tau_{t,i} \leq t$ is the number of non-zero $g_{s,i}$ seen on coordinate i on rounds $s = 1, \ldots, t$. Then, against convex loss functions f_t, with g_t a sub-gradient of f_t at $\hat{\beta}_t$, such that $\forall t$, $\|g_t\|_\infty \leq G$, we have*

$$\mathbf{E}[\text{Regret}] \leq \sum_{i=1}^{d} \left(2R\sqrt{2\tau_{T,i}(G^2 + \gamma^2)} + \gamma R\sqrt{\tau_{T,i}} \right).$$

The proof follows by summing the bound from Theorem 3.1 over each coordinate, considering only the rounds when $g_{t,i} \neq 0$, and then using the inequality $\sum_{t=1}^{T} 1/\sqrt{t} \leq 2\sqrt{T}$ to handle the sum of learning rates on each coordinate.

The core intuition behind this algorithm is that for features where we have little data (that is, τ_i is small, for

[1] Extension to arbitrary feasible sets is possible, but choosing the hyper-rectangle simplifies the analysis; in practice, projection onto the feasible set rarely helps performance.

Algorithm 2 OGD-Rand

input: feasible set $\mathcal{F} = [-R, R]^d$, parameters $\alpha, \gamma > 0$
Initialize $\hat{\beta}_1 = 0 \in \mathbb{R}^d$; $\forall i, \tau_i = 0$
for t=1, ..., T **do**
  Play the point $\hat{\beta}_t$, observe loss function f_t
  for i=1, ..., d **do**
    let $g_{t,i} = \nabla f_t(x_t)_i$
    if $g_{t,i} = 0$ **then** continue
    $\tau_i \leftarrow \tau_i + 1$
    let $\eta_{t,i} = \alpha/\sqrt{\tau_i}$ and $\epsilon_{t,i} = \gamma\eta_{t,i}$
    $\beta_{t+1,i} \leftarrow \text{Project}\left(\hat{\beta}_{t,i} - \eta_{t,i} g_{t,i}\right)$
    $\hat{\beta}_{t+1,i} \leftarrow \text{RandomRound}(\beta_{t+1,i}, \epsilon_{t,i})$

example rare words in a bag-of-words representation, identified by a binary feature), using a fine-precision coefficient is unnecessary, as we can't estimate the correct coefficient with much confidence. This is in fact the same reason using a larger learning rate is appropriate, so it is no coincidence the theory suggests choosing ϵ_t and η_t to be of the same magnitude.

Fixed Discretization Rather than implementing an adaptive discretization schedule, it is more straightforward and more efficient to choose a fixed grid resolution, for example a 16-bit `Qn.m` representation is sufficient for many applications.[2] In this case, one can apply the above theory, but simply stop decreasing the learning rate once it reaches say ϵ ($= 2^{-\mathtt{m}}$). Then, the $\eta_{1:T}$ term in the regret bound yields a linear term like $\mathcal{O}(\epsilon T)$; this is unavoidable when using a fixed resolution ϵ. One could let the learning rate continue to decrease like $1/\sqrt{t}$, but this would provide no benefit; in fact, lower-bounding the learning-rate is known to allow online gradient descent to provide regret bounds against a moving comparator (Zinkevich, 2003).

Data Structures There are several viable approaches to storing models with variable–sized coefficients. One can store all keys at a fixed (low) precision, then maintain a sequence of maps (*e.g.*, as hash-tables), each containing a mapping from keys to coefficients of increasing precision. Alternately, a simple linear probing hash–table for variable length keys is efficient for a wide variety of distributions on key lengths, as demonstrated by Thorup (2009). With this data structure, keys and coefficient values can be treated as strings over 4-bit or 8-bit bytes, for example. Blandford & Blelloch (2008) provide yet another data structure: a compact dictionary for variable length keys. Finally, for a fixed model, one can write out the string s of all coefficients (without end of string delimiters), store a second binary string of length s with ones at the coefficient boundaries, and use any of a number of rank/select data structures to index into it, *e.g.*, the one of Patrascu (2008).

[2] If we scale $x \to 2x$ then we must take $\beta \to \beta/2$ to make the same predictions, and so appropriate choices of `n` and `m` must be data-dependent.

3.2. Approximate Feature Counts

Online convex optimization methods typically use a learning rate that decreases over time, *e.g.*, setting η_t proportional to $1/\sqrt{t}$. Per-coordinate learning rates require storing a unique count τ_i for each coordinate, where τ_i is the number of times coordinate i has appeared with a non-zero gradient so far. Significant space is saved by using a 8-bit randomized counting scheme rather than a 32-bit (or 64-bit) integer to store the d total counts. We use a variant of Morris' probabilistic counting algorithm (1978) analyzed by Flajolet (1985). Specifically, we initialize a counter $C = 1$, and on each increment operation, we increment C with probability $p(C) = b^{-C}$, where base b is a parameter. We estimate the count as $\tilde{\tau}(C) = \frac{b^C - b}{b-1}$, which is an unbiased estimator of the true count. We then use learning rates $\eta_{t,i} = \alpha/\sqrt{\tilde{\tau}_{t,i} + 1}$, which ensures that even when $\tilde{\tau}_{t,i} = 0$ we don't divide by zero.

We compute high-probability bounds on this counter in Lemma A.1. Using these bounds for $\eta_{t,i}$ in conjunction with Theorem 3.1, we obtain the following result (proof deferred to the appendix).

Theorem 3.3. *Consider running the algorithm of Corollary 3.2 under the assumptions specified there, but using approximate counts $\tilde{\tau}_i$ in place of the exact counts τ_i. The approximate counts are computed using the randomized counter described above with any base $b > 1$. Thus, $\tilde{\tau}_{t,i}$ is the estimated number of times $g_{s,i} \neq 0$ on rounds $s = 1, \ldots, t$, and the per–coordinate learning rates are $\eta_{t,i} = \alpha/\sqrt{\tilde{\tau}_{t,i} + 1}$. With an appropriate choice of α we have*

$$\mathbf{E}[\text{Regret}(g)] = o\left(R\sqrt{G^2 + \gamma^2} T^{0.5+\delta}\right) \qquad \text{for all } \delta > 0,$$

where the o-notation hides a small constant factor and the dependence on the base b.[3]

4. Encoding During Prediction Time

Many real-world problems require large-scale *prediction*. Achieving scale may require that a trained model be replicated to multiple machines (Buciluǎ et al., 2006). Saving RAM via rounding is especially attractive here, because unlike in training accumulated

[3] Eq. (5) in the appendix provides a non-asymptotic (but more cumbersome) regret bound.

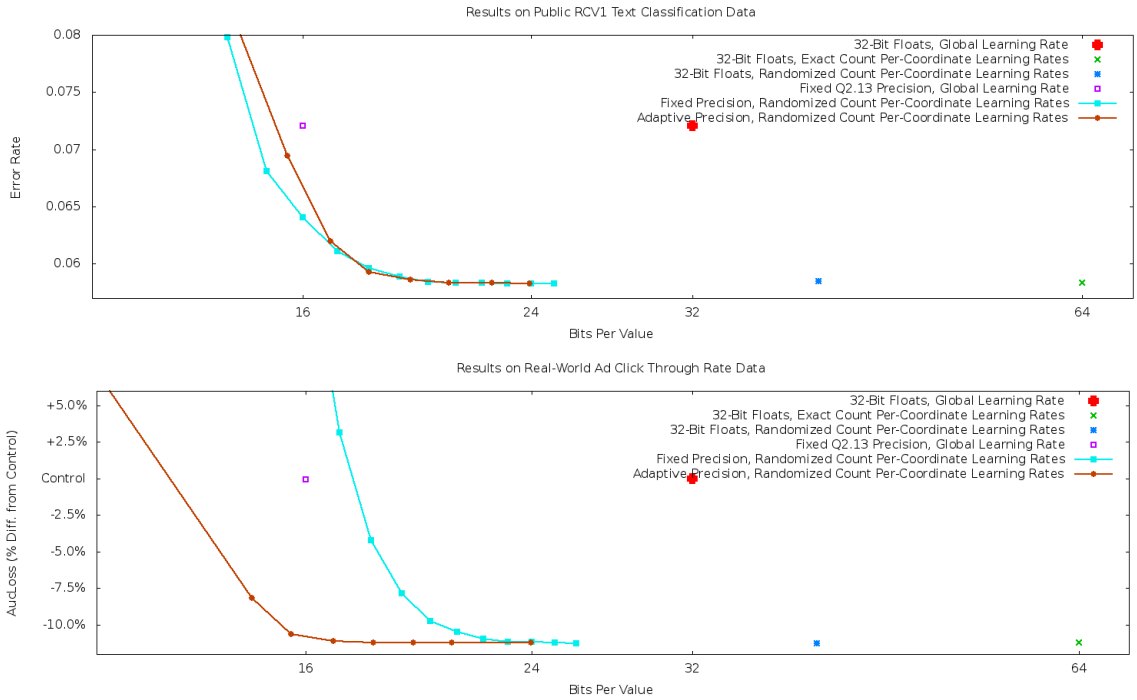

Figure 2. Rounding at Training Time. The fixed `q2.13` encoding is 50% smaller than control with no loss. Per-coordinate learning rates significantly improve predictions but use 64 bits per value. Randomized counting reduces this to 40 bits. Using adaptive or fixed precision reduces memory use further, to 24 total bits per value or less. The benefit of adaptive precision is seen more on the larger CTR data.

roundoff error is no longer an issue. This allows even more aggressive rounding to be used safely.

Consider a rounding a trained model β to some $\hat{\beta}$. We can bound both the additive and relative effect on logistic–loss $\mathcal{L}(\cdot)$ in terms of the quantity $|\beta \cdot x - \hat{\beta} \cdot x|$:

Lemma 4.1 (Additive Error)**.** *Fix $\beta, \hat{\beta}$ and (x, y). Let $\delta = |\beta \cdot x - \hat{\beta} \cdot x|$. Then the logistic–loss satisfies*

$$\mathcal{L}(x, y; \hat{\beta}) - \mathcal{L}(x, y; \beta) \leq \delta.$$

Proof. It is well known that $\left|\frac{\partial \mathcal{L}(x,y;\beta)}{\partial \beta_i}\right| \leq 1$ for all x, y, β and i, which implies the result. □

Lemma 4.2 (Relative Error)**.** *Fix $\beta, \hat{\beta}$ and $(x, y) \in \{0, 1\}^d \times \{0, 1\}$. Let $\delta = |\beta \cdot x - \hat{\beta} \cdot x|$. Then*

$$\frac{\mathcal{L}(x, y; \hat{\beta}) - \mathcal{L}(x, y; \beta)}{\mathcal{L}(x, y; \beta)} \leq e^{\delta} - 1.$$

Proofs for results in this section can be found in the extended version of this paper. Now, suppose we are using fixed precision numbers to store our model coefficients such as the `Qn.m` encoding described earlier, with a precision of ϵ. This induces a grid of feasible model coefficient vectors. If we randomly round each coefficient β_i (where $|\beta_i| \leq 2^n$) independently up or down to the nearest feasible value $\hat{\beta}_i$, such that $\mathbf{E}[\hat{\beta}_i] = \beta_i$, then for any $x \in \{0, 1\}^d$ our predicted log-odds ratio, $\hat{\beta} \cdot x$ is distributed as a sum of independent random variables $\{\hat{\beta}_i \mid x_i = 1\}$.

Let $k = \|x\|_0$. In this situation, note that $|\beta \cdot x - \hat{\beta} \cdot x| \leq \epsilon \|x\|_1 = \epsilon k$, since $|\beta_i - \hat{\beta}_i| \leq \epsilon$ for all i. Thus Lemma 4.1 implies

$$\mathcal{L}(x, y; \hat{\beta}) - \mathcal{L}(x, y; \beta) \leq \epsilon \, \|x\|_1.$$

Similarly, Lemma 4.2 immediately provides an upper bound of $e^{\epsilon k} - 1$ on relative logistic error; this bound is relatively tight for small k, and holds with probability one, but it does not exploit the fact that the randomness is unbiased and that errors should cancel out when k is large. The following theorem gives a bound on expected relative error that is much tighter for large k:

Theorem 4.3. *Let $\hat{\beta}$ be a model obtained from β using unbiased randomized rounding to a precision ϵ grid as described above. Then, the expected logistic–loss relative error of $\hat{\beta}$ on any input x is at most $2\sqrt{2\pi k} \exp\left(\epsilon^2 k/2\right) \epsilon$ where $k = \|x\|_0$.*

Additional Compression Figure 1 reveals that coefficient values are not uniformly distributed. Storing these values in a fixed-point representation means that individual values will occur many times. Basic information theory shows that the more common val-

Table 1. Rounding at Prediction Time for CTR Data. Fixed-point encodings are compared to a 32-bit floating point control model. Added loss is negligible even when using only 1.5 bits per value with optimal encoding.

ENCODING	AucLoss	Opt. Bits/Val
Q2.3	+5.72%	0.1
Q2.5	+0.44%	0.5
Q2.7	+0.03%	1.5
Q2.9	+0.00%	3.3

ues may be encoded with fewer bits. The theoretical bound for a whole model with d coefficients is $\frac{-\sum_{i=1}^{d} \log p(\beta_i)}{d}$ bits per value, where $p(v)$ is the probability of occurrence of v in β across all dimensions d. Variable length encoding schemes may approach this limit and achieve further RAM savings.

5. Experimental Results

We evaluated on both public and private large data sets. We used the public RCV1 text classification data set, specifically from Chang & Lin (2011). In keeping with common practice on this data set, the smaller "train" split of 20,242 examples was used for parameter tuning and the larger "test" split of 677,399 examples was used for the full online learning experiments. We also report results from a private CTR data set of roughly 30M examples and 20M features, sampled from real ad click data from a major search engine. Even larger experiments were run on data sets of billions of examples and billions of dimensions, with similar results as those reported here.

The evaluation metrics for predictions are error rate for the RCV1 data, and AucLoss (or 1-AUC) relative to a control model for the CTR data. Lower values are better. Metrics are computed using progressive validation (Blum et al., 1999) as is standard for online learning: on each round a prediction is made for a given example and record for evaluation, and only after that is the model allowed to train on the example. We also report the number of bits per coordinate used.

Rounding During Training Our main results are given in Figure 2. The comparison baseline is online logistic regression using a single global learning rate and 32-bit floats to store coefficients. We also test the effect of per-coordinate learning rates with both 32-bit integers for exact counts and with 8-bit randomized counts. We test the range of tradeoffs available for fixed-precision rounding with randomized counts, varying the number of precision `m` in `q2.m` encoding to plot the tradeoff curve (cyan). We also test the range of tradeoffs available for adaptive-precision rounding with randomized counts, varying the precision scalar γ to plot the tradeoff curve (dark red). For all randomized counts a base of 1.1 was used. Other than these differences, the algorithms tested are identical.

Using a single global learning rate, a fixed `q2.13` encoding saves 50% of the RAM at no added loss compared to the baseline. The addition of per-coordinate learning rates gives significant improvement in predictive performance, but at the price of added memory consumption, increasing from 32 bits per coordinate to 64 bits per coordinate in the baselines. Using randomized counts reduces this down to 40 bits per coordinate. However, both the fixed-precision and the adaptive precision methods give far better results, achieving the same excellent predictive performance as the 64-bit method with 24 bits per coefficient or less. This saves 62.5% of the RAM cost compared to the 64-bit method, and is still smaller than using 32-bit floats with a global learning rate.

The benefit of adaptive precision is only apparent on the larger CTR data set, which has a "long tail" distribution of support across features. However, it is useful to note that the simpler fixed-precision method also gives great benefit. For example, using `q2.13` encoding for coefficient values and 8-bit randomized counters allows full-byte alignment in naive data structures.

Rounding at Prediction Time We tested the effect of performing coarser randomized rounding of a fully-trained model on the CTR data, and compared to the loss incurred using a 32-bit floating point representation. These results, given in Table 1, clearly support the theoretical analysis that suggests more aggressive rounding is possible at prediction time. Surprisingly coarse levels of precision give excellent results, with little or no loss in predictive performance. The memory savings achievable in this scheme are considerable, down to *less than two bits per value* for `q2.7` with theoretically optimal encoding of the discrete values.

6. Conclusions

Randomized storage of coefficient values provides an efficient method for achieving significant RAM savings both during training and at prediction time.

While in this work we focus on OGD, similar randomized rounding schemes may be applied to other learning algorithms. The extension to algorithms that efficiently handle L_1 regularization, like RDA (Xiao, 2009) and FTRL-Proximal (McMahan, 2011), is rela-

tively straightforward.[4] Large scale kernel machines, matrix decompositions, topic models, and other large-scale learning methods may all be modifiable to take advantage of RAM savings through low precision randomized rounding methods.

A. Appendix: Proofs

A.1. Proof of Theorem 3.1

Our analysis extends the technique of Zinkevich (2003). Let β^* be any feasible point (with possibly infinite precision coefficients). By the definition of β_{t+1},

$$\|\beta_{t+1} - \beta^*\|^2 = \|\hat{\beta}_t - \beta^*\|^2 - 2\eta_t g_t \cdot (\hat{\beta}_t - \beta^*) + \eta_t^2 \|g_t\|^2.$$

Rearranging the above yields

$$\begin{aligned} g_t \cdot (\hat{\beta}_t - \beta^*) &\leq \frac{1}{2\eta_t}\left(\|\hat{\beta}_t - \beta^*\|^2 - \|\beta_{t+1} - \beta^*\|^2\right) + \frac{\eta_t}{2}\|g_t\|^2 \\ &= \frac{1}{2\eta_t}\left(\|\hat{\beta}_t - \beta^*\|^2 - \|\hat{\beta}_{t+1} - \beta^*\|^2\right) + \frac{\eta_t}{2}\|g_t\|^2 + \rho_t, \end{aligned}$$

where the $\rho_t = \frac{1}{2\eta_t}\left(\|\hat{\beta}_{t+1} - \beta^*\|^2 - \|\beta_{t+1} - \beta^*\|^2\right)$ terms will capture the extra regret due to the randomized rounding. Summing over t, and following Zinkevich's analysis, we obtain a bound of

$$\text{Regret}(T) \leq \frac{(2R)^2}{2\eta_T} + \frac{\|g_t\|_2^2}{2}\eta_{1:T} + \rho_{1:T}.$$

It remains to bound $\rho_{1:T}$. Letting $d_t = \beta_{t+1} - \hat{\beta}_{t+1}$ and $a_t = d_t/\eta_t$, we have

$$\begin{aligned} \rho_{1:T} &= \sum_{t=1}^{T} \frac{1}{2\eta_t}\left((\hat{\beta}_{t+1} - \beta^*)^2 - (\beta_{t+1} - \beta^*)^2\right) \\ &\leq \sum_{t=1}^{T} \frac{1}{2\eta_t}\left(\hat{\beta}_{t+1}^2 - \beta_{t+1}^2\right) + \beta^* a_{1:T} \\ &\leq \sum_{t=1}^{T} \frac{1}{2\eta_t}\left(\hat{\beta}_{t+1}^2 - \beta_{t+1}^2\right) + R\,|a_{1:T}|\,. \end{aligned}$$

We bound each of the terms in this last expression in expectation. First, note $|d_t| \leq \epsilon_t \leq \gamma\eta_t$ by definition of the resolution of the rounding grid, and so $|a_t| \leq \gamma$. Further $\mathbf{E}[d_t] = 0$ since the rounding is unbiased. Letting $W = |a_{1:T}|$, by Jensen's inequality we have $\mathbf{E}[W]^2 \leq \mathbf{E}[W^2]$. Thus, $\mathbf{E}[|a_{1:T}|] \leq \sqrt{\mathbf{E}[(a_{1:T})^2]} = \sqrt{\text{Var}(a_{1:T})}$, where the last equality follows from the fact $\mathbf{E}[a_{1:T}] = 0$. The a_t are not independent given an adaptive adversary.[5] Nevertheless, consider any a_s and a_t with $s < t$. Since both have expectation zero, $\text{Cov}(a_s, a_t) = \mathbf{E}[a_s a_t]$. By construction, $\mathbf{E}[a_t \mid g_t, \beta_t, \text{hist}_t] = 0$, where hist_t is the full history of the game up until round t, which includes a_s in particular. Thus

$$\text{Cov}(a_s, a_t) = \mathbf{E}[a_s a_t] = \mathbf{E}\big[\mathbf{E}[a_s a_t \mid g_t, \beta_t, \text{hist}_t]\big] = 0.$$

For all t, $|a_t| \leq \gamma$ so $\text{Var}(a_t) \leq \gamma^2$, and $\text{Var}(a_{1:T}) = \sum_t \text{Var}(a_t) \leq \gamma^2 T$. Thus, $\mathbf{E}[|a_{1:T}|] \leq \gamma\sqrt{T}$.

Next, consider $\mathbf{E}[\hat{\beta}_{t+1}^2 - \beta_{t+1}^2 \mid \beta_{t+1}]$. Since $\mathbf{E}[\hat{\beta}_{t+1} \mid \beta_{t+1}] = \beta_{t+1}$, for any shift $s \in \mathbb{R}$, we have $\mathbf{E}\big[(\hat{\beta}_{t+1} - s)^2 - (\beta_{t+1} - s)^2 \mid \beta_{t+1}\big] = \mathbf{E}\big[\hat{\beta}_{t+1}^2 - \beta_{t+1}^2 \mid \beta_{t+1}\big]$, and so taking $s = \beta_{t+1}$,

$$\begin{aligned} \frac{1}{\eta_t}\mathbf{E}\big[\hat{\beta}_{t+1}^2 - \beta_{t+1}^2 \mid \beta_{t+1}\big] &= \frac{1}{\eta_t}\mathbf{E}\big[(\hat{\beta}_{t+1} - \beta_{t+1})^2 \mid \beta_{t+1}\big] \\ &\leq \frac{\epsilon_t^2}{\eta_t} \leq \frac{\gamma^2\eta_t^2}{\eta_t} = \gamma^2\eta_t. \end{aligned}$$

Combining this result with $\mathbf{E}[|a_{1:T}|] \leq \gamma\sqrt{T}$, we have

$$\mathbf{E}\left[\rho_{1:T}\right] \leq \gamma^2\eta_{1:T} + \gamma R\sqrt{T},$$

which completes the proof. □

A.2. Approximate Counting

We first provide high–probability bounds for the approximate counter.

Lemma A.1. *Fix T and $t \leq T$. Let C_{t+1} be the value of the counter after t increment operations using the approximate counting algorithm described in Section 3.2 with base $b > 1$. Then, for all $c > 0$, the estimated count $\tilde{\tau}(C_{t+1})$ satisfies*

$$\mathbf{Pr}\left[\tilde{\tau}(C_{t+1}) < \frac{t}{bc\log(T)} - 1\right] \leq \frac{1}{T^{c-1}} \qquad (1)$$

and

$$\mathbf{Pr}\left[\tilde{\tau}(C_{t+1}) > \frac{et}{b-1} b^{\sqrt{2c\log_b(T)}+2}\right] \leq \frac{1}{T^c}. \qquad (2)$$

Both T and c are essentially parameters of the bound; in the Eq. (2), any choices of T and c that keep T^c constant produce the same bound. In the first bound, the result is sharpest when $T = t$, but it will be convenient to set T equal to the total number of rounds so that we can easily take a union bound (in the proof of Theorem 3.3).

[4] Some care must be taken to store a discretized version of a scaled gradient sum, so that the dynamic range remains roughly unchanged as learning progresses.

[5] For example the adversary could ensure $a_{t+1} = 0$ (by playing $g_{t+1} = 0$) iff $a_t > 0$.

Proof of Lemma A.1. Fix a sequence of T increments, and let C_i denote the value of the approximate counter at the start of increment number i, so $C_1 = 1$. Let $X_j = |\{i : C_i = j\}|$, a random variable for the number of increments for which the counter stayed at j.

We start with the bound of Eq. (1). When $C = j$, the update probability is $p_j = p(j) = b^{-j}$, so for any ℓ_j we have $X_j \geq \ell_j$ with probability at most $(1-p_j)^{\ell_j} \leq \exp(-p_j)^{\ell_j} = \exp(-p_j\ell_j)$ since $(1-x) \leq \exp(-x)$ for all x. To make this at most T^{-c} it suffices to take $\ell_j = c(\log T)/p_j = cb^j \log T$. Taking a (rather loose) union bound over $j = 1, 2, \ldots, T$, we have

$$\mathbf{Pr}\big[\exists j,\ X_j > cb^j \log T\big] \leq 1/T^{c-1}.$$

For Eq. (1), it suffices to show that if this does not occur, then $\tilde{\tau}(C_t) \geq t/(bc\log(T))-1$. Note $\sum_{j=1}^{C_t} X_j \geq t$. With our supposition that $X_j \leq cb^j \log T$ for all j, this implies $t \leq \sum_{j=1}^{C_t} cb^j \log T = cb \log T \left(\frac{b^{C_t}-1}{b-1}\right)$, and thus $C_t \geq \log_b\left(\frac{t(b-1)}{bc\log T}+1\right)$. Since $\tilde{\tau}$ is monotonically increasing and $b > 1$, simple algebra then shows $\tilde{\tau}(C_{t+1}) \geq \tilde{\tau}(C_t) \geq t/(bc\log(T)) - 1$.

Next consider the bound of Eq. (2). Let j_0 be the minimum value such that $p(j_0) \leq 1/et$, and fix $k \geq 0$. Then $C_{t+1} \geq j_0 + k$ implies the counter was incremented k times with an increment probability at most $p(j_0)$. Thus,

$$\begin{aligned}
\mathbf{Pr}[C_t \geq j_0 + k] &\leq \binom{t}{k} \prod_{j=j_0}^{j_0+k-1} p(j) \\
&\leq \left(\frac{te}{k}\right)^k \left(\prod_{j=0}^{k-1} p(j_0) b^{-j}\right) \\
&= \left(\frac{te}{k}\right)^k p(j_0)^k b^{-k(k-1)/2} \\
&\leq k^{-k} \cdot b^{-k(k-1)/2}
\end{aligned}$$

Note that $j_0 \leq \lceil \log_b(et) \rceil$. Taking $k = \sqrt{2c\log_b(T)}+1$ is sufficient to ensure this probability is at most T^{-c}, since $k^{-k} \leq 1$ and $k^2 - k \geq 2c\log_b T$. Observing that $\tilde{\tau}\left(\lceil \log_b(et) \rceil + \sqrt{2c\log_b(T)}+1\right) \leq \frac{et}{b-1} b^{\sqrt{2c\log_b(T)}+2}$ completes the proof. □

Proof of Theorem 3.3. We prove the bound for the one-dimensional case; the general bound then follows by summing over dimensions. Since we consider a single dimension, we assume $|g_t| > 0$ on all rounds. This is without loss of generality, because we can implicitly skip all rounds with zero gradients, which means we don't need to make the distinction between t and $\tau_{t,i}$. We abuse notation slightly by defining $\tilde{\tau}_t \equiv \tilde{\tau}(C_{t+1}) \approx t = \tau_t$ for the approximate count on round t. We begin from the bound

$$\mathbf{E}[\text{Regret}] \leq \frac{(2R)^2}{2\eta_T} + \frac{1}{2}(G^2+\gamma^2)\eta_{1:t} + \gamma R\sqrt{T}.$$

of Theorem 3.1, with learning rates $\eta_t = \alpha/\sqrt{\tilde{\tau}_t + 1}$. Lemma A.1 with $c = 2.5$ then implies

$$\mathbf{Pr}[\tilde{\tau}_t + 1 < k_1 t] \leq \frac{1}{T^{1.5}} \quad \text{and} \quad \mathbf{Pr}[\tilde{\tau}_t > k_2 t] \leq \frac{1}{T^{2.5}},$$

where $k_1 = 1/(bc\log T)$ and $k_2 = \frac{eb^{\sqrt{2c\log_b T}+2}}{b-1}$. A union bound on $t = 1, ..., T$ on the first bound implies with probability $1 - \frac{1}{\sqrt{T}}$ we have $\forall t,\ \tilde{\tau}_t + 1 \geq k_1 t$, so

$$\eta_{1:T} = \sum_{t=1}^{T} \frac{\alpha}{\sqrt{\tilde{\tau}_t+1}} \leq \frac{1}{\sqrt{k_1}} \sum_{t=1}^{T} \frac{\alpha}{\sqrt{t}} \leq \frac{2\alpha\sqrt{T}}{\sqrt{k_1}}, \quad (3)$$

where we have used the inequality $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 2\sqrt{T}$. Similarly, the second inequality implies with probability at least $1 - \frac{1}{T^{2.5}}$,

$$\eta_T = \frac{\alpha}{\sqrt{\tilde{\tau}_T+1}} \geq \frac{\alpha}{\sqrt{k_2 T+1}}. \quad (4)$$

Taking a union bound, Eqs. (3) and (4) hold with probability at least $1 - 2/\sqrt{T}$, and so at least one fails with probability at most $2/\sqrt{T}$. Since $f_t(\beta) - f_t(\beta') \leq 2GR$ for any $\beta, \beta' \in [-R, R]$ (using the convexity of f_t and the bound on the gradients G), on any run of the algorithm, regret is bounded by $2RGT$. Thus, these failed cases contribute at most $4RG\sqrt{T}$ to the expected regret bound.

Now suppose Eqs. (3) and (4) hold. Choosing $\alpha = \frac{R}{\sqrt{G^2+\gamma^2}}$ minimizes the dependence on the other constants, and note for any $\delta > 0$, both $\frac{1}{\sqrt{k_1}}$ and $\sqrt{k_2}$ are $o(T^\delta)$. Thus, when Eqs. (3) and (4) hold,

$$\begin{aligned}
\mathbf{E}[\text{Regret}] &\leq \frac{(2R)^2}{2\eta_T} + \frac{1}{2}(G^2+\gamma^2)\eta_{1:t} + \gamma R\sqrt{T} \\
&\leq \frac{2R^2\sqrt{k_2 T+1}}{\alpha} + (G^2+\gamma^2)\frac{\alpha\sqrt{T}}{\sqrt{k_1}} + \gamma R\sqrt{T} \\
&= o\Big(R\sqrt{G^2+\gamma^2}\,T^{0.5+\delta}\Big).
\end{aligned}$$

Adding $4RG\sqrt{T}$ for the case when the high-probability statements fail still leaves the same bound. □

It follows from the proof that we have the more precise but cumbersome upper bound on $\mathbf{E}$[Regret]:

$$\frac{2R^2\sqrt{k_2 T+1}}{\alpha} + (G^2+\gamma^2)\frac{\alpha\sqrt{T}}{\sqrt{k_1}} + \gamma R\sqrt{T} + 4RG\sqrt{T}. \quad (5)$$

References

Bilenko, Mikhail and Richardson, Matthew. Predictive client-side profiles for personalized advertising. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

Blandford, Daniel K. and Blelloch, Guy E. Compact dictionaries for variable-length keys and data with applications. *ACM Trans. Algorithms*, 4(2), May 2008.

Blum, Avrim, Kalai, Adam, and Langford, John. Beating the hold-out: bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, 1999.

Bottou, Léon and Bousquet, Olivier. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20. 2008.

Buciluǎ, Cristian, Caruana, Rich, and Niculescu-Mizil, Alexandru. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011. Datasets from `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

Craswell, Nick, Zoeter, Onno, Taylor, Michael, and Ramsey, Bill. An experimental comparison of click position-bias models. In *Proceedings of the international conference on Web search and web data mining*, 2008.

Duchi, John, Shalev-Shwartz, Shai, Singer, Yoram, and Chandra, Tushar. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, 2008.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, 2010.

Flajolet, Philippe. Approximate counting: A detailed analysis. *BIT*, 25(1):113–134, 1985.

Goodman, Joshua, Cormack, Gordon V., and Heckerman, David. Spam and the ongoing battle for the inbox. *Commun. ACM*, 50(2), 2 2007.

Langford, John, Li, Lihong, and Zhang, Tong. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10, June 2009.

McMahan, H. Brendan. Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

McMahan, H. Brendan and Streeter, Matthew. Adaptive bound optimization for online convex optimization. In *COLT*, 2010.

Morris, Robert. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10), October 1978. doi: 10.1145/359619.359627.

Patrascu, M. Succincter. In *IEEE Symposium on Foundations of Computer Science*, pp. 305–313. IEEE, 2008.

Raghavan, Prabhakar and Tompson, Clark D. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4), 12 1987.

Richardson, Matthew, Dominowska, Ewa, and Ragno, Robert. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

Shalev-Shwartz, Shai. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2), 2012.

Streeter, Matthew J. and McMahan, H. Brendan. Less regret via online conditioning. *CoRR*, abs/1002.4862, 2010.

Thorup, Mikkel. String hashing for linear probing. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009.

Tibshirani, Robert. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1), 1996.

Van Durme, Benjamin and Lall, Ashwin. Probabilistic counting with randomized storage. In *Proceedings of the 21st international jont conference on Artifical intelligence*, 2009.

Weinberger, Kilian, Dasgupta, Anirban, Langford, John, Smola, Alex, and Attenberg, Josh. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

Xiao, Lin. Dual averaging method for regularized stochastic learning and online optimization. In *NIPS*, 2009.

Zinkevich, Martin. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.