Feature Hashing for Large Scale Multitask Learning

Kilian Weinberger   KILIAN@YAHOO-INC.COM
Anirban Dasgupta   ANIRBAN@YAHOO-INC.COM
Josh Attenberg   JOSH@CIS.POLY.EDU
John Langford   JL@HUNCH.NET
Alex Smola   ALEX@SMOLA.ORG

Yahoo! Research, 2821 Mission College Blvd., Santa Clara, CA 95051 USA

Abstract

Empirical evidence suggests that hashing is an effective strategy for dimensionality reduction and practical nonparametric estimation. In this paper we provide exponential tail bounds for feature hashing and show that the interaction between random subspaces is negligible with high probability. We demonstrate the feasibility of this approach with experimental results for a new use case — multitask learning with hundreds of thousands of tasks.

1. Introduction

Kernel methods use inner products as the basic tool for comparisons between objects. That is, given objects $x_1, \ldots, x_n \in \mathfrak{X}$ for some domain $\mathfrak{X}$, they rely on

$$k(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle \qquad (1)$$

to compare the features $\phi(x_i)$ of x_i and $\phi(x_j)$ of x_j respectively.

Eq. (1) is often famously referred to as the *kernel-trick*. It allows the use of inner products between very high dimensional feature vectors $\phi(x_i)$ and $\phi(x_j)$ *implicitly* through the definition of a positive semi-definite kernel matrix k without ever having to compute a vector $\phi(x_i)$ directly. This can be particularly powerful in classification settings where the original input representation has a non-linear decision boundary. Often, linear separability can be achieved in a high dimensional feature space $\phi(x_i)$.

In practice, for example in text classification, researchers

frequently encounter the opposite problem: the original input space is almost linearly separable (often because of the existence of handcrafted non-linear features), yet, the training set may be prohibitively large in size and very high dimensional. In such a case, there is no need to map the input vectors into a higher dimensional feature space. Instead, limited memory makes storing a kernel matrix infeasible.

For this common scenario several authors have recently proposed an alternative, but highly complimentary variation of the kernel-trick, which we refer to as the *hashing-trick*: one *hashes* the high dimensional input vectors x into a *lower* dimensional feature space $\mathbb{R}^m$ with $\phi : \mathfrak{X} \to \mathbb{R}^m$ (Langford et al., 2007; Shi et al., 2009). The parameter vector of a classifier can therefore live in $\mathbb{R}^m$ instead of in $\mathbb{R}^n$ with kernel matrices or $\mathbb{R}^d$ in the original input space, where $m \ll n$ and $m \ll d$. Different from random projections, the hashing-trick preserves sparsity and introduces no additional overhead to store projection matrices.

To our knowledge, we are the first to provide exponential tail bounds on the canonical distortion of these hashed inner products. We also show that the hashing-trick can be particularly powerful in multi-task learning scenarios where the original feature spaces are the cross-product of the data, $\mathfrak{X}$, and the set of tasks, U. We show that one can use different hash functions for each task $\phi_1, \ldots, \phi_{|U|}$ to map the data into one joint space with little interference.

While many potential applications exist for the hashing-trick, as a particular case study we focus on collaborative email spam filtering. In this scenario, hundreds of thousands of users collectively label emails as *spam* or *not-spam*, and each user expects a personalized classifier that reflects their particular preferences. Here, the set of tasks, U, is the number of email users (this can be very large for open systems such as *Yahoo Mail*™ or *Gmail*™), and the feature space spans the union of vocabularies in multitudes

of languages.

This paper makes four main contributions: 1. In section 2 we introduce specialized hash functions with unbiased inner-products that are directly applicable to a large variety of kernel-methods. 2. In section 3 we provide exponential tail bounds that help explain why hashed feature vectors have repeatedly lead to, at times surprisingly, strong empirical results. 3. Also in section 3 we show that the interference between independently hashed subspaces is negligible with high probability, which allows large-scale multi-task learning in a very compressed space. 4. In section 5 we introduce collaborative email-spam filtering as a novel application for hash representations and provide experimental results on large-scale real-world spam data sets.

2. Hash Functions

We introduce a variant on the hash kernel proposed by (Shi et al., 2009). This scheme is modified through the introduction of a *signed* sum of hashed features whereas the original hash kernels use an *unsigned* sum. This modification leads to an unbiased estimate, which we demonstrate and further utilize in the following section.

Definition 1 *Denote by h a hash function $h : \mathbb{N} \rightarrow \{1, \ldots, m\}$. Moreover, denote by ξ a hash function $\xi : \mathbb{N} \rightarrow \{\pm 1\}$. Then for vectors $x, x' \in \ell_2$ we define the hashed feature map ϕ and the corresponding inner product as*

$$\phi_i^{(h,\xi)}(x) = \sum_{j:h(j)=i} \xi(i)x_i \qquad (2)$$

$$\text{and } \langle x, x' \rangle_\phi := \left\langle \phi^{(h,\xi)}(x), \phi^{(h,\xi)}(x') \right\rangle. \qquad (3)$$

Although the hash functions in definition 1 are defined over the natural numbers $\mathbb{N}$, in practice we often consider hash functions over arbitrary strings. These are equivalent, since each finite-length string can be represented by a unique natural number.

Usually, we abbreviate the notation $\phi^{(h,\xi)}(\cdot)$ by just $\phi(\cdot)$. Two hash functions ϕ and ϕ' are different when $\phi = \phi^{(h,\xi)}$ and $\phi' = \phi^{(h',\xi')}$ such that either $h' \neq h$ or $\xi \neq \xi'$. The purpose of the binary hash ξ is to remove the bias inherent in the hash kernel of (Shi et al., 2009).

In a multi-task setting, we obtain instances in combination with tasks, $(x, u) \in \mathcal{X} \times U$. We can naturally extend our definition 1 to hash pairs, and will write $\phi_u(x) = \phi(x, u)$.

3. Analysis

The following section is dedicated to theoretical analysis of hash kernels and their applications. In this sense, the present paper continues where (Shi et al., 2009) falls short: we prove exponential tail bounds. These bounds hold for general hash kernels, which we later apply to show how hashing enables us to do large-scale multitask learning efficiently. We start with a simple lemma about the bias and variance of the hash kernel. The proof of this lemma appears in appendix A.

Lemma 2 *The hash kernel is unbiased, that is $\mathbf{E}_\phi[\langle x, x' \rangle_\phi] = \langle x, x' \rangle$. Moreover, the variance is $\sigma_{x,x'}^2 = \frac{1}{m} \left(\sum_{i \neq j} x_i^2 x_j'^2 + x_i x_i' x_j x_j' \right)$, and thus, for $\|x\|_2 = \|x'\|_2 = 1$, $\sigma_{x,x'}^2 = O\left(\frac{1}{m}\right)$.*

This suggests that typical values of the hash kernel should be concentrated within $O(\frac{1}{\sqrt{m}})$ of the target value. We use Chebyshev's inequality to show that half of all observations are within a range of $\sqrt{2}\sigma$. This, together with an indirect application of Talagrand's convex distance inequality via the result of (Liberty et al., 2008), enables us to construct exponential tail bounds.

3.1. Concentration of Measure Bounds

In this subsection we show that under a hashed feature-map the length of each vector is preserved with high probability. Talagrand's inequality (Ledoux, 2001) is a key tool for the proof of the following theorem (detailed in the appendix B).

Theorem 3 *Let $\epsilon < 1$ be a fixed constant and x be a given instance such that $\|x\|_2 = 1$. If $m \geq 72 \log(1/\delta)/\epsilon^2$ and $\|x\|_\infty \leq \frac{\epsilon}{18\sqrt{\log(1/\delta)\log(m/\delta)}}$, we have that*

$$\Pr[|\|x\|_\phi^2 - 1| \geq \epsilon] \leq 2\delta. \qquad (4)$$

Note that an analogous result would also hold for the original hash kernel of (Shi et al., 2009), the only modification being the associated bias terms. The above result can also be utilized to show a concentration bound on the inner product between two general vectors x and x'.

Corollary 4 *For two vectors x and x', let us define*

$$\sigma := \max(\sigma_{x,x}, \sigma_{x',x'}, \sigma_{x-x',x-x'})$$

$$\eta := \max\left(\frac{\|x\|_\infty}{\|x\|_2}, \frac{\|x'\|_\infty}{\|x'\|_2}, \frac{\|x-x'\|_\infty}{\|x-x'\|_2}\right).$$

Also let $\Delta = \|x\|^2 + \|x'\|^2 + \|x - x'\|^2$. If $m \geq \Omega(\frac{1}{\epsilon^2} \log(1/\delta))$ and $\eta = O(\frac{\epsilon}{\log(m/\delta)})$, then we have that

$$\Pr\left[|\langle x, x' \rangle_\phi - \langle x, x' \rangle| > \epsilon\Delta/2\right] < \delta.$$

The proof for this corollary can be found in appendix C. We can also extend the bound in Theorem 3 for the maximal

canonical distortion over large sets of distances between vectors as follows:

Corollary 5 *If $m \geq \Omega(\frac{1}{\epsilon^2} \log(n/\delta))$ and $\eta = O(\frac{\epsilon}{\log(m/\delta)})$. Denote by $X = \{x_1, \ldots, x_n\}$ a set of vectors which satisfy $\|x_i - x_j\|_{\infty} \leq \eta \|x_i - x_j\|_2$ for all pairs i, j. In this case with probability $1 - \delta$ we have for all i, j*

$$\frac{| \|x_i - x_j\|_{\phi}^2 - \|x_i - x_j\|_2^2 |}{\|x_i - x_j\|_2^2} \leq \epsilon.$$

This means that the number of observations n (or correspondingly the size of the un-hashed kernel matrix) only enters *logarithmically* in the analysis.

Proof We apply the bound of Theorem 3 to each distance individually. Note that each vector $x_i - x_j$ satisfies the conditions of the theorem, and hence for each vector $x_i - x_j$, we preserve the distance upto a factor of $(1 \pm \epsilon)$ with probability $1 - \frac{\delta}{n^2}$. Taking the union bound over all pairs gives us the result. ∎

3.2. Multiple Hashing

Note that the tightness of the union bound in Corollary 5 depends crucially on the magnitude of η. In other words, for large values of η, that is, whenever some terms in x are very large, even a single collision can already lead to significant distortions of the embedding. This issue can be amended by trading off sparsity with variance. A vector of unit length may be written as $(1, 0, 0, 0, \ldots)$, or as $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, \ldots\right)$, or more generally as a vector with c nonzero terms of magnitude $c^{-\frac{1}{2}}$. This is relevant, for instance whenever the magnitudes of x follow a known pattern, e.g. when representing documents as bags of words since we may simply hash frequent words several times. The following corollary gives an intuition as to how the confidence bounds scale in terms of the replications:

Lemma 6 *If we let $x' = \frac{1}{\sqrt{c}}(x, \ldots, x)$ then:*

1. *It is norm preserving: $\|x\|_2 = \|x'\|_2$.*

2. *It reduces component magnitude by $\frac{1}{\sqrt{c}} = \frac{\|x'\|_{\infty}}{\|x\|_{\infty}}$.*

3. *Variance increases to $\sigma_{x',x'}^2 = \frac{1}{c}\sigma_{x,x}^2 + \frac{c-1}{c}2\|x\|_2^4$.*

Applying Lemma 6 to Theorem 3, a large magnitude can be decreased at the cost of an increased variance.

3.3. Approximate Orthogonality

For multitask learning, we must learn a different parameter vector for each related task. When mapped into the same hash-feature space we want to ensure that there is little interaction between the different parameter vectors. Let U be a set of different tasks, $u \in U$ being a specific one. Let w be a combination of the parameter vectors of tasks in $U \setminus \{u\}$. We show that for any observation x for task u, the interaction of w with x in the hashed feature space is minimal. For each x, let the image of x under the hash feature-map for task u be denoted as $\phi_u(x) = \phi^{(\xi,h)}((x, u))$.

Theorem 7 *Let $w \in \mathbb{R}^m$ be a parameter vector for tasks in $U \setminus \{u\}$. In this case the value of the inner product $\langle w, \phi_u(x) \rangle$ is bounded by*

$$\Pr\{|\langle w, \phi_u(x) \rangle| > \epsilon\} \leq 2e^{-\frac{\epsilon^2/2}{m^{-1}\|w\|_2^2 \|x\|_2^2 + \epsilon \|w\|_{\infty} \|x\|_{\infty}/3}}$$

Proof We use Bernstein's inequality (Bernstein, 1946), which states that for independent random variables X_j, with $\mathbf{E}[X_j] = 0$, if $C > 0$ is such that $|X_j| \leq C$, then

$$\Pr\left[\sum_{j=1}^{n} X_j > t\right] \leq \exp\left(-\frac{t^2/2}{\sum_{j=1}^{n} \mathbf{E}[X_j^2] + Ct/3}\right). \quad (5)$$

We have to compute the concentration property of $\langle w, \phi_u(x) \rangle = \sum_j x_j \xi(j) w_{h(j)}$. Let $X_j = x_j \xi(j) w_{h(j)}$. By the definition of h and ξ, X_j are independent. Also, for each j, since w depends only on the hash-functions for $U \setminus \{u\}$, $w_h(j)$ is independent of $\xi(j)$. Thus, $\mathbf{E}[X_j] = \mathbf{E}_{(\xi,h)}[x_j \xi(j) w_{h(j)}] = 0$. For each j, we also have $|X_j| < \|x\|_{\infty} \|w\|_{\infty} =: C$. Finally, $\sum_j \mathbf{E}[X_j^2]$ is given by

$$\mathbf{E}\left[\sum_j (x_j \xi(j) w_{h(j)})^2\right] = \frac{1}{m}\sum_{j,\ell} x_j^2 w_{\ell}^2 = \frac{1}{m}\|x\|_2^2 \|w\|_2^2$$

The claim follows by plugging both terms and C into the Bernstein inequality (5). ∎

Theorem 7 bounds the influence of unrelated tasks with any particular instance. In section 5 we demonstrate the real-world applicability with empirical results on a large-scale multi-task learning problem.

4. Applications

The advantage of feature hashing is that it allows for significant storage compression for parameter vectors: storing w in the raw feature space naively requires $O(d)$ numbers, when $w \in \mathbb{R}^d$. By hashing, we are able to reduce this to $O(m)$ numbers while avoiding costly matrix-vector multiplications common in Locally Sensitive Hashing. In addition, the sparsity of the resulting vector is preserved.

The benefits of the hashing-trick leads to applications in almost all areas of machine learning and beyond. In particular, feature hashing is extremely useful whenever large numbers of parameters with redundancies need to be stored within bounded memory capacity.

Personalization One powerful application of feature hashing is found in multitask learning. Theorem 7 allows us to hash multiple classifiers for different tasks into one feature space with little interaction. To illustrate, we explore this setting in the context of spam-classifier personalization.

Suppose we have thousands of users U and want to perform related but not identical classification tasks for each of the them. Users provide labeled data by marking emails as *spam* or *not-spam*. Ideally, for each user $u \in U$, we want to learn a predictor w_u based on the data of that user solely. However, webmail users are notoriously lazy in labeling emails and even those that do not contribute to the training data expect a working spam filter. Therefore, we also need to learn an additional global predictor w_0 to allow data sharing amongst all users.

Storing all predictors w_i requires $O(d \times (|U| + 1))$ memory. In a task like collaborative spam-filtering, $|U|$, the number of users can be in the hundreds of thousands and the size of the vocabulary is usually in the order of millions. The naive way of dealing with this is to eliminate all infrequent tokens. However, spammers target this memory-vulnerability by maliciously misspelling words and thereby creating highly infrequent but spam-typical tokens that "fall under the radar" of conventional classifiers. Instead, if all words are hashed into a finite-sized feature vector, infrequent but class-indicative tokens get a chance to contribute to the classification outcome. Further, large scale spam-filters (e.g. *Yahoo Mail*[TM] or *GMail*[TM]) typically have severe memory and time constraints, since they have to handle billions of emails per day. To guarantee a finite-size memory footprint we hash all weight vectors $w_0, \ldots, w_{|U|}$ into a joint, significantly smaller, feature space $\mathbb{R}^m$ with different hash functions $\phi_0, \ldots, \phi_{|U|}$. The resulting hashed-weight vector $w_h \in \mathbb{R}^m$ can then be written as:

$$w_h = \phi_0(w_0) + \sum_{u \in U} \phi_u(w_u). \qquad (6)$$

Note that in practice the weight vector w_h can be learned directly in the hashed space. All un-hashed weight vectors never need to be computed. Given a new document/email x of user $u \in U$, the prediction task now consists of calculating $\langle \phi_0(x) + \phi_u(x), w_h \rangle$. Due to hashing we have two sources of error – distortion ϵ_d of the hashed inner products and the interference with other hashed weight vectors

ϵ_i. More precisely:

$$\langle \phi_0(x) + \phi_u(x), w_h \rangle = \langle x, w_0 + w_u \rangle + \epsilon_d + \epsilon_i. \qquad (7)$$

The interference error consists of all collisions between $\phi_0(x)$ or $\phi_u(x)$ with hash functions of other users,

$$\epsilon_i = \sum_{v \in U, v \neq 0} \langle \phi_0(x), \phi_v(w_v) \rangle + \sum_{v \in U, v \neq u} \langle \phi_u(x), \phi_v(w_v) \rangle. \qquad (8)$$

To show that ϵ_i is small with high probability we can apply Theorem 7 twice, once for each term of (8). We consider each user's classification to be a separate task, and since $\sum_{v \in U, v \neq 0} w_v$ is independent of the hash-function ϕ_0, the conditions of Theorem 7 apply with $w = \sum_{v \neq 0} w_v$ and we can employ it to bound the second term, $\sum_{v \in U, v \neq 0} \langle \phi_u(x), \phi_u(w_v) \rangle$. The second application is identical except that all subscripts "0" are substituted with "u". For lack of space we do not derive the exact bounds.

The distortion error occurs because each hash function that is utilized by user u can self-collide:

$$\epsilon_d = \sum_{v \in \{u, 0\}} | \langle \phi_v(x), \phi_v(w_v) \rangle - \langle x, w_v \rangle |. \qquad (9)$$

To show that ϵ_d is small with high probability, we apply Corollary 4 once for each possible values of v.

In section 5 we show experimental results for this setting. The empirical results are stronger than the theoretical bounds derived in this subsection—our technique outperforms a single global classifier on hundreds thousands of users. We discuss an intuitive explanation in section 5.

Massively Multiclass Estimation We can also regard massively multi-class classification as a multitask problem, and apply feature hashing in a way similar to the personalization setting. Instead of using a different hash function for each user, we use a different hash function for each class.

(Shi et al., 2009) apply feature hashing to problems with a high number of categories. They show empirically that *joint* hashing of the feature vector $\phi(x, y)$ can be efficiently achieved for problems with millions of features and thousands of classes.

Collaborative Filtering Assume that we are given a very large sparse matrix M where the entry M_{ij} indicates what action user i took on instance j. A common example for actions and instances is user-ratings of movies (Bennett & Lanning,). A successful method for finding common factors amongst users and instances for predicting unobserved actions is to factorize M into $M = U^\top W$. If we have millions of users performing millions of actions, storing U
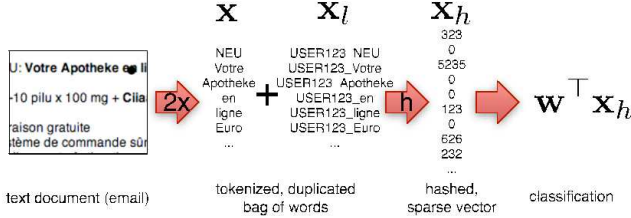
Figure 1. The hashed personalization summarized in a schematic layout. Each token is duplicated and one copy is individualized (e.g. by concatenating each word with a unique user identifier). Then, the global hash function maps all tokens into a low dimensional feature space where the document is classified.

and W in memory quickly becomes infeasible. Instead, we may choose to compress the matrices U and W using hashing. For $U, W \in \mathbb{R}^{n \times d}$ denote by $u, w \in \mathbb{R}^m$ vectors with

$$u_i = \sum_{j,k:h(j,k)=i} \xi(j,k)U_{jk} \text{ and } w_i = \sum_{j,k:h'(j,k)=i} \xi'(j,k)W_{jk}.$$

where (h, ξ) and (h', ξ') are independently chosen hash functions. This allows us to approximate matrix elements $M_{ij} = [U^\top W]_{ij}$ via

$$M_{ij}^\phi := \sum_k \xi(k,i)\xi'(k,j)u_{h(k,i)}w_{h'(k,j)}.$$

This gives a compressed vector representation of M that can be efficiently stored.

5. Results

We evaluated our algorithm in the setting of personalization. As data set, we used a proprietary email spam-classification task of $n = 3.2$ million emails, properly anonymized, collected from $|U| = 433167$ users. Each email is labeled as *spam* or *not-spam* by one user in U. After tokenization, the data set consists of 40 million unique words.

For all experiments in this paper, we used the Vowpal Wabbit implementation[1] of stochastic gradient descent on a square-loss. In the mail-spam literature the misclassification of *not-spam* is considered to be much more harmful than misclassification of *spam*. We therefore follow the convention to set the classification threshold during test time such that exactly 1% of the $not-spam$ test data is classified as $spam$ Our implementation of the personalized hash functions is illustrated in Figure 1. To obtain a personalized hash function ϕ_u for user u, we concatenate a unique user-id to each word in the email and then hash the newly generated tokens with the same global hash function.
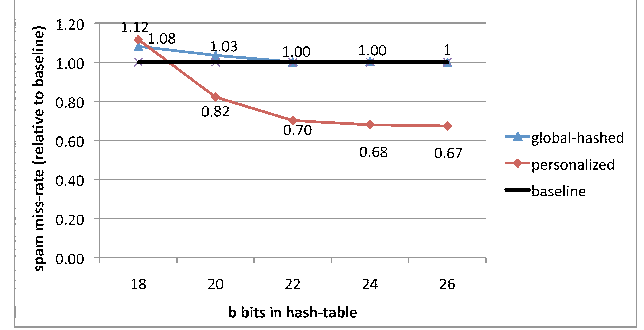
Figure 2. The decrease of uncaught spam over the baseline classifier averaged over all users. The classification threshold was chosen to keep the not-spam misclassification fixed at 1%. The hashed global classifier (*global-hashed*) converges relatively soon, showing that the distortion error ϵ_d vanishes. The personalized classifier results in an average improvement of up to 30%.

The data set was collected over a span of 14 days. We used the first 10 days for training and the remaining 4 days for testing. As *baseline*, we chose the purely global classifier trained over all users and hashed into 2^{26} dimensional space. As 2^{26} far exceeds the total number of unique words we can regard the baseline to be representative for the classification without hashing. All results are reported as the amount of spam that passed the filter undetected, relative to this baseline (eg. a value of 0.80 indicates a 20% reduction in spam for the user)[2].

Figure 2 displays the average amount of spam in users' inboxes as a function of the number of hash keys m, relative to the baseline above. In addition to the baseline, we evaluate two different settings.

The *global-hashed* curve represents the relative spam catch-rate of the global classifier after hashing $\langle \phi_0(w_0), \phi_0(x) \rangle$. At $m = 2^{26}$ this is identical to the baseline. Early convergence at $m = 2^{22}$ suggests that at this point hash collisions have no impact on the classification error and the *baseline* is indeed equivalent to that obtainable without hashing.

In the *personalized* setting each user $u \in U$ gets her own classifier $\phi_u(w_u)$ as well as the global classifier $\phi_0(w_0)$. Without hashing the feature space explodes, as the cross product of $u = 400K$ users and $n = 40M$ tokens results in 16 trillion possible unique personalized features. Figure 2 shows that despite aggressive hashing, personalization results in a 30% spam reduction once the hash table is indexed by 22 bits.
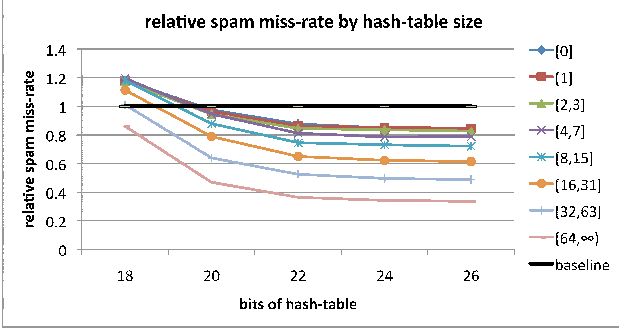
Figure 3. Results for users clustered by training emails. For example, the bucket [8, 15] consists of all users with eight to fifteen training emails. Although users in buckets with large amounts of training data do benefit more from the personalized classifier (up-to 65% reduction in spam), even users that did not contribute to the training corpus at all obtain almost 20% spam-reduction.

User clustering   One hypothesis for the strong results in Figure 2 might originate from the non-uniform distribution of user votes — it is possible that using personalization and feature hashing we benefit a small number of users who have labeled many emails, degrading the performance of most users (who have labeled few or no emails) in the process. In fact, in real life, a large fraction of email users do not contribute at all to the training corpus and only interact with the classifier during test time. The personalized version of the test email $\Phi_u(x_u)$ is then hashed into buckets of other tokens and only adds interference noise ϵ_i to the classification.

In order to show that we improve the performance of most users, it is therefore important that we not only report averaged results over all emails, but explicitly examine the effects of the personalized classifier for users depending on their contribution to the training set. To this end, we place users into exponentially growing buckets based on their number of training emails and compute the relative reduction of uncaught spam for each bucket individually. Figure 3 shows the results on a per-bucket basis. We do not compare against a *purely* local approach, with no global component, since for a large fraction of users—those without training data—this approach cannot outperform random guessing.

It might appear rather surprising that users in the bucket with none or very little training emails (the line of bucket [0] is identical to bucket [1]) also benefit from personalization. After all, their personalized classifier was never trained and can only add noise at test-time. The classifier improvement of this bucket can be explained by the subjective definition of *spam* and *not-spam*. In the personalized setting the individual component of user labeling is absorbed by the local classifiers and the global classifier

represents the *common* definition of spam and not-spam. In other words, the global part of the personalized classifier obtains better generalization properties, benefiting all users.

6. Related Work

A number of researchers have tackled related, albeit different problems.

(Rahimi & Recht, 2008) use Bochner's theorem and sampling to obtain approximate inner products for Radial Basis Function kernels. (Rahimi & Recht, 2009) extend this to sparse approximation of weighted combinations of basis functions. This is computationally efficient for many function spaces. Note that the representation is *dense*.

(Li et al., 2007) take a complementary approach: for sparse feature vectors, $\phi(x)$, they devise a scheme of reducing the number of nonzero terms even further. While this is in principle desirable, it does not resolve the problem of $\phi(x)$ being high dimensional. More succinctly, it is necessary to express the function in the dual representation rather than expressing f as a linear function, where w is unlikely to be compactly represented: $f(x) = \langle \phi(x), w \rangle$.

(Achlioptas, 2003) provides computationally efficient randomization schemes for dimensionality reduction. Instead of performing a dense $d \cdot m$ dimensional matrix vector multiplication to reduce the dimensionality for a vector of dimensionality d to one of dimensionality m, as is required by the algorithm of (Gionis et al., 1999), he only requires $\frac{1}{3}$ of that computation by designing a matrix consisting only of entries $\{-1, 0, 1\}$. Pioneered by **(Ailon & Chazelle, 2006)**, there has been a line of work (Ailon & Liberty, 2008; Matousek, 2008) on improving the complexity of random projection by using various code-matrices in order to preprocess the input vectors. Some of our theoretical bounds are derivable from that of **(Liberty et al., 2008)**.

A related construction is the CountMin sketch of **(Cormode & Muthukrishnan, 2004)** which stores counts in a number of replicates of a hash table. This leads to good concentration inequalities for range and point queries.

(Shi et al., 2009) propose a hash kernel to deal with the issue of computational efficiency by a very simple algorithm: high-dimensional vectors are compressed by adding up all coordinates which have the same hash value — one only needs to perform as many calculations as there are nonzero terms in the vector. This is a significant computational saving over locality sensitive hashing (Achlioptas, 2003; Gionis et al., 1999).

Several additional works provide motivation for the investigation of hashing representations. For example, **(Ganchev & Dredze, 2008)** provide empirical evidence that the hash-

ing trick can be used to effectively reduce the memory footprint on many sparse learning problems by an order of magnitude via removal of the dictionary. Our experimental results validate this, and show that much more radical compression levels are achievable. In addition, **(Langford et al., 2007)** released the Vowpal Wabbit fast online learning software which uses a hash representation similar to that discussed here.

7. Conclusion

In this paper we analyze the hashing-trick for dimensionality reduction theoretically and empirically. As part of our theoretical analysis we introduce unbiased hash functions and provide exponential tail bounds for hash kernels. These give further inside into hash-spaces and explain previously made empirical observations. We also derive that random subspaces of the hashed space are likely to not interact, which makes multitask learning with many tasks possible.

Our empirical results validate this on a real-world application within the context of spam filtering. Here we demonstrate that even with a very large number of tasks and features, all mapped into a joint lower dimensional hash-space, one can obtain impressive classification results with finite memory guarantee.

References

Achlioptas, D. (2003). Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, *66*, 671–687.

Ailon, N., & Chazelle, B. (2006). Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. *Proc. 38th Annual ACM Symposium on Theory of Computing* (pp. 557–563).

Ailon, N., & Liberty, E. (2008). Fast dimension reduction using Rademacher series on dual BCH codes. *Proc. 19th Annual ACM-SIAM Symposium on Discrete algorithms* (pp. 1–9).

Alon, N. (2003). Problems and results in extremal combinatorics, Part I. *Discrete Math*, *273*, 31–53.

Bennett, J., & Lanning, S. The Netflix Prize. *Proceedings of KDD Cup and Workshop 2007*.

Bernstein, S. (1946). *The theory of probabilities*. Moscow: Gastehizdat Publishing House.

Cormode, G., & Muthukrishnan, M. (2004). An improved data stream summary: The count-min sketch and its applications. *LATIN: Latin American Symposium on Theoretical Informatics*.

Dasgupta, A., Sarlos, T., & Kumar, R. (2010). A Sparse Johnson Lindenstrauss Transform. *Submitted*.

Ganchev, K., & Dredze, M. (2008). Small statistical models by random feature mixing. *Workshop on Mobile Language Processing, Annual Meeting of the Association for Computational Linguistics*.

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. *Proceedings of the 25th VLDB Conference* (pp. 518–529). Edinburgh, Scotland: Morgan Kaufmann.

Langford, J., Li, L., & Strehl, A. (2007). *Vowpal wabbit online learning project* (Technical Report). http://hunch.net/?p=309.

Ledoux, M. (2001). *The concentration of measure phenomenon*. Providence, RI: AMS.

Li, P., Church, K., & Hastie, T. (2007). Conditional random sampling: A sketch-based sampling technique for sparse data. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in neural information processing systems 19*, 873–880. Cambridge, MA: MIT Press.

Liberty, E., Ailon, N., & Singer, A. (2008). Dense fast random projections and lean Walsh transforms. *Proc. 12th International Workshop on Randomization and Approximation Techniques in Computer Science* (pp. 512–522).

Matousek, J. (2008). On variants of the Johnson–Lindenstrauss lemma. *Random Structures and Algorithms*, *33*, 142–156.

Rahimi, A., & Recht, B. (2008). Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*. Cambridge, MA: MIT Press.

Rahimi, A., & Recht, B. (2009). Randomized kitchen sinks. In L. Bottou, Y. Bengio, D. Schuurmans and D. Koller (Eds.), *Advances in neural information processing systems 21*. Cambridge, MA: MIT Press.

Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A., Strehl, A., & Vishwanathan, V. (2009). Hash kernels. *AISTATS 12*.

Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., & Smola, A. (2009). Feature hashing for large scale multitask learning. *26th International Conference on Machine Learning* (p. 140).

A. Mean and Variance

Proof [Lemma 2] To compute the expectation we expand

$$\langle x, x' \rangle_\phi = \sum_{i,j} \xi(i)\xi(j) x_i x_j' \delta_{h(i),h(j)}. \tag{10}$$

Since $\mathbf{E}_\phi[\langle x, x' \rangle_\phi] = \mathbf{E}_h[\mathbf{E}_\xi[\langle x, x' \rangle_\phi]]$, taking expectations over ξ we see that only the terms $i = j$ have nonzero value, which shows the first claim. For the variance we compute $\mathbf{E}_\phi[\langle x, x' \rangle_\phi^2]$. Expanding this, we get:

$$\langle x, x' \rangle_\phi^2 = \sum_{i,j,k,l} \xi(i)\xi(j)\xi(k)\xi(l) x_i x_j' x_k x_l' \delta_{h(i),h(j)} \delta_{h(k),h(l)}.$$

This expression can be simplified by noting that:

$$\mathbf{E}_\xi[\xi(i)\xi(j)\xi(k)\xi(l)] = \delta_{ij}\delta_{kl} + [1 - \delta_{ijkl}](\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}).$$

Passing the expectation over ξ through the sum, this allows us to break down the expansion of the variance into two terms.

$$\mathbf{E}_\phi[\langle x, x' \rangle_\phi^2] = \sum_{i,k} x_i x_i' x_k x_k' + \sum_{i \neq j} x_i^2 x_j'^2 \mathbf{E}_h[\delta_{h(i),h(j)}]$$

$$+ \sum_{i \neq j} x_i x_i' x_j x_j' \mathbf{E}_h[\delta_{h(i),h(j)}]$$

$$= \langle x, x' \rangle^2 + \frac{1}{m}\left(\sum_{i \neq j} x_i^2 x_j'^2 + \sum_{i \neq j} x_i x_i' x_j x_j'\right)$$

by noting that $\mathbf{E}_h[\delta_{h(i),h(j)}] = \frac{1}{m}$ for $i \neq j$. Using the fact that $\sigma^2 = \mathbf{E}_\phi[\langle x, x' \rangle_\phi^2] - \mathbf{E}_\phi[\langle x, x' \rangle_\phi]^2$ proves the claim. ∎

B. Concentration of Measure

We use the concentration result derived by Liberty, Ailon and Singer in (Liberty et al., 2008). Liberty et al. create a Johnson-Lindenstrauss random projection matrix by combining a carefully constructed deterministic matrix A with random diagonal matrices. For completeness we restate the relevant lemma. Let i range over the hash-buckets. Let $m = c \log(1/\delta)/\epsilon^2$ for a large enough constant c. For a given vector x, define the diagonal matrix D_x as $(D_x)_{jj} = x_j$. For any matrix $A \in \Re^{m \times d}$, define $\|x\|_A \equiv \max_{y:\|y\|_2=1} \|AD_x y\|_2$.

Lemma 2 (Liberty et al., 2008). *For any column-normalized matrix A, vector x with $\|x\|_2 = 1$ and an i.i.d. random ± 1 diagonal matrix D_s, the following holds: $\forall x$, if $\|x\|_A \leq \frac{\epsilon}{6\sqrt{\log(1/\delta)}}$ then, $\Pr[|\|AD_s x\|_2 - 1| > \epsilon] \leq \delta$.*

We also need the following form of a weighted balls and bins inequality – the statement of the Lemma, as well as

the proof follows that of Lemma 6 (Dasgupta et al., 2010). We still outline the proof because of some parameter values being different.

Lemma 8 *Let m be the size of the hash function range and let $\eta = \frac{1}{2\sqrt{m \log(m/\delta)}}$. If x is such that $\|x\|_2 = 1$ and $\|x\|_\infty \leq \eta$, then define $\sigma_*^2 = \max_i \sum_{j=1}^d x_j^2 \delta_{ih(j)}$ where i ranges over all hash-buckets. We have that with probability $1 - \delta$,*

$$\sigma_*^2 \leq \frac{2}{m}$$

Proof We outline the proof-steps. Since the buckets have identical distribution, we look only at the 1st bucket, i.e. at $i = 1$ and bound $\sum_{j:h(j)=1} x_j^2$. Define $X_j = x_j^2 \left(\delta_{1h(j)} - \frac{1}{m}\right)$. Then $E_h[X_j] = 0$ and $E_h[X_j^2] = x_j^4\left(\frac{1}{m} - \frac{1}{m^2}\right) \leq \frac{x_j^4}{m} \leq \frac{x_j^2 \eta^2}{m}$ using $\|x\|_\infty \leq \eta$. Thus, $\sum_j E_h[X_j^2] \leq \frac{\eta^2}{m}$. Also note that $\sum_j X_j = \sum_{j:h(j)=1} x_j^2 - \frac{1}{m}$. Plugging this into the Bernstein's inequality, equation 5, we have that

$$\Pr[\sum_j X_j > \frac{1}{m}] \leq \exp\left(-\frac{1/2m^2}{\eta^2/m + \eta^2/3m}\right)$$

$$= \exp(-\frac{3}{8m\eta^2}) \leq \exp(-\log(m/\delta)) \leq \delta/m$$

By taking union bound over all the m buckets, we get the above result. ∎

Proof [Theorem 3] Given the function $\phi = (h, r)$, define the matrix A as $A_{ij} = \delta_{ih(j)}$ and D_s as $(D_s)_{jj} = r_j$. Let x be as specified, i.e. $\|x\|_2 = 1$ and $\|x\|_\infty \leq \eta$. Note that $\|x\|_\phi = \|AD_s x\|_2$. Let $y \in \Re^d$ be such that $\|y\|_2 = 1$. Thus

$$\|AD_x y\|_2^2 = \sum_{i=1}^m \left(\sum_{j=1}^d y_j \delta_{ih(j)} x_j\right)^2$$

$$\leq \sum_{i=1}^m (\sum_{j=1}^d y_j^2 \delta_{ih(j)})(\sum_{j=1}^d x_j^2 \delta_{ih(j)})$$

$$\leq \sum_{i=1}^m (\sum_{j=1}^d y_j^2 \delta_{ih(j)}) \sigma_*^2 \leq \sigma_*^2.$$

by applying the Cauchy-Schwartz inequality, and using the definition of σ_*. Thus, $\|x\|_A = \max_{y:\|y\|_2=1} \|AD_x y\|_2 \leq \sigma_* \leq \sqrt{2} m^{-1/2}$. If $m \geq \frac{72}{\epsilon^2} \log(1/\delta)$, we have that $\|x\|_A \leq \frac{\epsilon}{6\sqrt{\log(1/\delta)}}$, which satisfies the conditions of Lemma 2 from (Liberty et al., 2008). Thus applying the above result from Lemma 2 (Liberty et al., 2008) to x, and

using Lemma 8, we have that $\Pr[|\|AD_s x\|^2 - 1| \geq \epsilon] \leq \delta$ and hence

$$\Pr[|\|x\|_\phi^2 - 1| \geq \epsilon] \leq \delta$$

by taking union over the two error probabilities of Lemma 2 and Lemma 8, we have the result. ∎

C. Inner Product

Proof [Corollary 4] We have that $2\langle x, x'\rangle_\phi = \|x\|_\phi^2 + \|x'\|_\phi^2 - \|x - x'\|_\phi^2$. Taking expectations, we have the standard inner product inequality. Thus,

$$|2\langle x, x'\rangle_\phi - 2\langle x, x'\rangle| \leq |\|x\|_\phi^2 - \|x\|^2|$$
$$+ |\|x'\|_\phi^2 - \|x'\|^2| + |\|x - x'\|_\phi^2 - \|x - x'\|^2|$$

Using union bound, with probability $1 - 3\delta$, each of the terms above is bounded using Theorem 3. Thus, putting the bounds together, we have that, with probability $1 - 3\delta$,

$$|2\langle \phi_u(x), \phi_u(x)\rangle - 2\langle x, x\rangle| \leq \epsilon(\|x\|^2 + \|x'\|^2 + \|x - x'\|^2)$$

∎

D. Refutation of the Previous Incorrect Proof

There were a few bugs in the previous version of the paper (Weinberger et al., 2009). We now detail each of them and illustrate why it was an error. The current result shows that the using hashing we can create a projection matrix that can preserve distances to a factor of $(1 \pm \epsilon)$ for vectors with a bounded $\|x\|_\infty/\|x\|_2$ ratio. The constraint on input vectors can be circumvented by multiple hashing, as outlined in Section 3.2, but that would require hashing $O(\frac{1}{\epsilon^2})$ times. Recent work (Dasgupta et al., 2010) suggests that better theoretical bounds can be shown for this construction. We thank Tamas Sarlos and Ravi Kumar for the following writeup on the errors and for suggestion the new proof in Appendix B.

1. The statement of the main theorem in Weinberger et al. (Weinberger et al., 2009, Theorem 3) is false as it contradicts the lower bound of Alon (Alon, 2003). The flaw lies in the probability of error in (Weinberger et al., 2009, Theorem 3), which was claimed to be $\exp(-\frac{\sqrt{\epsilon}}{4\eta})$. This error can be made arbitrarily small without increasing the embedding dimensionality m but by decreasing $\eta = \frac{\|x\|_\infty}{\|x\|_2}$, which in turn can be achieved by preprocessing the input vectors x. However, this contradicts Alon's lower bound on the em-

bedding dimensionality. The details of this contradiction are best presented through (Weinberger et al., 2009, Corollary 5) as follows.

Set $m = 128$ and $\delta = 1/2$ and consider the vertices of the n-simplex in $\Re^{n+1}$, i.e., $x_1 = (1, 0, ..., 0)$, $x_2 = (0, 1, 0, ..., 0)$, ... Let $P \in \Re^{(n+1)c \times (n+1)}$ be the naive, replication based preconditioner, with replication parameter $c = 512 \log^2 n$ as defined in Section 2 of our submission or (Weinberger et al., 2009, Section 3.2). Therefore for all pairs $i \neq j$ we have that $\|Px_i - Px_j\|_\infty = 1/\sqrt{c}$ and that $\|Px_i - Px_j\|_2 = \sqrt{2}$. Hence we can apply (Weinberger et al., 2009, Corollary 5) to the set of vectors Px_i with $\eta = 1/\sqrt{2c} = 1/(32 \log n)$; then the claimed approximation error is $\sqrt{\frac{2}{m} + 64\eta^2 \log^2 \frac{n}{2\delta}} = \frac{1}{8} + \frac{1}{16} \leq \frac{1}{4}$. If Corollary 5 were true, then it would follow that with probability at least $1/2$, the linear transformation $A = \phi \cdot P : \Re^{n+1} \to \Re^m$ distorts the pairwise distances of the above $n + 1$ vectors by at most a $1 \pm 1/4$ multiplicative factor. On the other hand, the lower bound of Alon shows that any such transformation A must map to $\Omega(\log n)$ dimensions; see the remarks following Theorem 9.3 in (Alon, 2003) and set $\epsilon = 1/4$ there. This clearly contradicts $m = 128$ above.

2. The proof of the Theorem 3 contained a fatal, unfixable error. Recall that δ_{ij} denotes the usual Kronecker symbol, and h and h' are hash functions. Weinberger et al. make the following observation after equation (13) of their proof on page 8 in Appendix B.

   "First note that $\sum_i \sum_j \delta_{h(j)i} + \delta_{h'(j)i}$ is at most $2t$, where $t = |\{j : h(j) \neq h'(j)\}|$."

   The quoted observation is false. Let d denote the dimension of the input. Then, $\sum_i \sum_j \delta_{h(j)i} + \delta_{h'(j)i} = \sum_j (\sum_i \delta_{h(j)i} + \delta_{h'(j)i}) = \sum_j 2 = 2d$, independent of the choice of the hash function. Note that t played a crucial role in the proof of (Weinberger et al., 2009) relating the Euclidean approximation error of the dimensionality reduction to Talagrand's convex distance defined over the set of hash functions. Albeit the error is elementary, we do not see how to rectify its consequences in (Weinberger et al., 2009) even if the claim were of the right form.

3. The proof of Theorem 3 in (Weinberger et al., 2009) also contains a minor and fixable error. To see this, consider the sentence towards the end of the proof Theorem 3 in (Weinberger et al., 2009) where $0 < \epsilon < 1$ and $\beta = \beta(x) \geq 1$.

   "Noting that $s^2 = (\sqrt{\beta^2 + \epsilon} - \beta)/4\|x\|_\infty \geq \sqrt{\epsilon}/4\|x\|_\infty, ...$"

Here the authors wrongly assume that $\sqrt{\beta^2 + \epsilon} - \beta \geq \sqrt{\epsilon}$ holds, whereas the truth is $\sqrt{\beta^2 + \epsilon} - \beta \leq \sqrt{\epsilon}$ always.

Observe that this glitch is easy to fix locally, however this change is minor and the modified claim would still be false. Since for all $0 \leq y \leq 1$ we have that $\sqrt{1+y} \geq 1 + y/3$, from $\beta \geq 1$ it follows that $\sqrt{\beta^2 + \epsilon} - \beta \geq \epsilon/3$. Plugging the latter estimate into the "proof" of Theorem 3 would result in a modified claim where the original probability of error, $\exp(-\frac{\sqrt{\epsilon}}{4\eta})$, is replaced with $\exp(-\frac{\epsilon}{12\eta})$. Updating the numeric constants in the first section of this note would show that the new claim still contradicts Alon's lower bound. To justify observe that counter example is based on a constant ϵ and the modified claim would still lack the necessary $\Omega(\log n)$ dependency in its target dimensionality.