Client-Friendly Classification over Random Hyperplane Hashes

Shyamsundar Rajaram and Martin Scholz

Hewlett Packard Laboratories
1501 Page Mill Road
Palo Alto, CA
shyam.rajaram@hp.com, scholz@hp.com

Abstract. In this work, we introduce a powerful and general feature representation based on a locality sensitive hash scheme called random hyperplane hashing. We are addressing the problem of centrally learning (linear) classification models from data that is distributed on a number of clients, and subsequently deploying these models on the same clients. Our main goal is to balance the accuracy of individual classifiers and different kinds of costs related to their deployment, including communication costs and computational complexity. We hence systematically study how well schemes for sparse high-dimensional data adapt to the much denser representations gained by random hyperplane hashing, how much data has to be transmitted to preserve enough of the semantics of each document, and how the representations affect the overall computational complexity. This paper provides theoretical results in the form of error bounds and margin based bounds to analyze the performance of classifiers learnt over the hash-based representation. We also present empirical evidence to illustrate the attractive properties of random hyperplane hashing over the conventional baseline representation of bag of words with and without feature selection.

1 Introduction

In times of increasingly web-oriented information architectures, it becomes more and more natural to push analytical software down to clients, and to have them report back critical and prototypical events that require additional attention or indicate specific business opportunities. Examples of analytical software running on user PCs include spam filtering, malware detection, and diagnostic tools for different kinds of system failures.

We are concerned with a family of classification problems where high-dimensional, sparse training data is available on a large number of clients. We want to centrally collect data to train classifiers for deployment on the clients. Although the techniques we are studying apply to a broader set of problems, like eDiscovery, for the sake of this paper, we will exemplarily only consider the problem of classifying (reasonably large) text documents, like web pages a user visits. We aim to classify with respect to a quickly changing taxonomy of relevant concepts.

Our constraints in this setting stem from the natural goal of minimizing resource consumption on clients. This includes network bandwidth, but also memory and CPU footprint of classifiers and related software. During a cycle of training and deploying classifiers we go through the following phases: Data is preprocessed on clients before it is uploaded to a server. The preprocessing is required to reduce data volumes, and sometimes also to preserve privacy. The classifiers are learnt on the server. Clients download a potentially large number of classifiers, so we would like to minimize the required bandwidth. Finally, the models are deployed on the clients and triggered for each document under consideration, so we are concerned with the associated costs of preprocessing each document and of applying a linear classifier on top of that representation.

The questions we are addressing in this paper are therefore, which representations of sparse and high-dimensional data are **compact enough** to be transmitted over the web, **general enough** to be used for all kinds of upcoming multi-class classification problems, **cheap enough** to be applicable at deployment time, and are **close enough** to the performance of the models that are not narrowed down by operational costs.

The novelty of this work is in exploiting a locality sensitive hashing technique called random hyperplane hashing (cf. Algorithm 1) towards building a compact, general, cheap and powerful representation scheme. This technique has the additional benefit of being content-agnostic, which implies that it does not require any deep domain understanding in the preprocessing phase. Further, it can be used in the text domain, without any feature selection and for any language. The contributions of this paper are two fold: (i) we present key theoretical results in terms of error bounds and margin based bounds to quantify the loss of information due to random hyperplane hashing and (ii) we present experimental results to bring out the above mentioned attractive properties of the hashing-based representation scheme.

The closely related random projection technique [3,4,9,19,10] has been used successfully as a tool for dimensionality reduction because of its simplicity and nice theoretical Euclidean distance preserving properties. Though the random hyperplane hash method has a striking resemblance to the random projection method (as seen in Algorithm 1), there is a key difference as random hyperplane hashing preserves the angular distance while random projection preserves Euclidean distance, leading to a completely different representation scheme. More importantly, from a practical standpoint, the bit-based representation based on random hyperplane hashing turns out to be significantly more compact, which we elaborate later in the paper.

The rest of the paper is organized as follows. Sec. 2 provides the required technical preliminaries and reviews locality sensitive hashing with emphasis on the angular distance preserving random hyperplane representation scheme. In this paper, we confine ourselves to the problem of learning linear classification models on top of this representation; in Sec. 3 we discuss this setting and present theoretical results based on error bounds and margin based bounds. Sec. 3 includes a comparative study of our representation scheme with that of random projection

```
Require:
  – Input document $d$
  – Number $k$ of output dimensions
  – Type of transformation, either RP or RHH
Ensure:
  $k$-dimensional boolean (for RHH) or integer (for RP) vector representing $d$

  Computation:
  Create a $k$ dimensional vector $v$ with $v[i] = 0$ for $1 \leq i \leq k$
  for all terms $w$ in document $d$ do
    Set random seed to $w$            // cast $w$ to integer or use hash value
    for all $i$ in $(1, \ldots, k)$ do
      b ← sample random bit uniformly from $\{-1, +1\}$
      $v[i] \leftarrow v[i] + b$
  for all $i$ in $(1, \ldots, k)$ do
    $v[i] \leftarrow sign(v[i])$            // only for RHH, skip this step for RP
  return $v$
```

Algorithm 1. A random projection (RP) / hyperplane (RHH) algorithm

based representation. We complement our theoretical findings with a number of empirical results on benchmark datasets in Sec. 4. Finally, we summarize and conclude in Sec. 5.

2 Random Hyperplane Hashing

Locality Sensitive Hash functions are invaluable tools for approximate near neighbor problems in high dimensional spaces. Traditionally, nearest neighbor search in high dimensional spaces has been expensive, because with increasing dimensionality, indexing schemes such as KD Trees very quickly deteriorate to a linear scan of all the items. Locality Sensitive Hash Functions [12] were introduced to solve the approximate nearest neighbor problem in high dimensional spaces, and several advancements [6,2,5] have been done in this area.

Definition 1 (Locality Sensitive Hashing [5,12]). *A locality sensitive hashing scheme is a distribution on a family $\mathcal{F}$ of hash functions on a set of instances, such that for two instances x and y,*

$$\mathbf{Pr}_{h \in \mathcal{F}} [h(x) = h(y)] = f(\mathrm{sim}(x, y)) \tag{1}$$

where, $\mathrm{sim}(x, y)$ *is some similarity function defined on the instances and f is a monotonically increasing function i.e., more the similarity, higher the probability.*

Simply put, a locality sensitive hash function is designed in such a way that if two vectors are close in the intended distance measure, the probability that they hash to the same value is high; if they are far in the intended distance measure, the probability that they hash to the same value is low. Next, we provide a formal review of locality sensitive hash functions. In particular, we focus on the cosine similarity as it is a popular one for a variety of applications such as in document retrieval [17], natural language processing [16] and image retrieval [18].

Definition 2 (Cosine Similarity). *The cosine of two vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^m$ is defined as $cos(u, v) = \frac{u.v}{|u||v|}$.*

As we will discuss, the random hyperplane hashing algorithm provides a locality sensitive hash function that corresponds to the cosine similarity measure. Let us first introduce the notion of a cosine hash family.

Definition 3 (Cosine Hash Family [7]). *A set of functions $H = \{h_1, h_2, \ldots\}$ constitute a cosine hash family over $\mathbb{R}^m$ iff for some finite $\mathbf{U} \subset \mathbb{N}$,*

– $h_k \in H : \mathbb{R}^m \to \mathbf{U}$
– *For any four vectors u, u', v, v' in $\mathbb{R}^m$ where $\cos(u, u') = \cos(v, v')$, the following is true:*

$$\mathbf{Pr}_{h \in H}(h(u) = h(u')) = \mathbf{Pr}_{h \in H}(h(v) = h(v'))$$

– *For any four vectors u, u', v, v' in $\mathbb{R}^m$ where $\cos(u, u') > \cos(v, v')$, the following is true:*

$$\mathbf{Pr}_{h \in H}(h(u) = h(u')) > \mathbf{Pr}_{h \in H}(h(v) = h(v'))$$

The following definition is similar to Algorithm 1, but more convenient for analytical purposes. We will compare the two at a later point.

Definition 4 (Random Hyperplane Hash (RHH) algorithm [5]). *The random hyperplane hashing algorithm works as follows: for a desired hash length of k, generate an $m \times k$ projection matrix M, where each element is chosen i.i.d from a $\mathcal{N}(0, 1)$ distribution. The k dimensional hash of a vector $u \in \mathbb{R}^m$ is computed in two steps,*

– *Compute the vector $P = uM$*
– *The ith entry of the hash of u is -1 if $P_i < 0$, and 1 otherwise.*

We will conclude this section with a number of important properties of random projections that will be used throughout the paper.

Lemma 1. *Let u, v be vectors in $\mathbb{R}^m$ such that $cos(u, v) = \rho$ and their corresponding k-dimensional random hyperplane hash are denoted as u' and v' respectively. We denote the distance between u' and v' as $d'(u', v')$ which is defined below. Then,*

1. $P(u'_i = v'_i) = 1 - \frac{\arccos(\rho)}{\pi}$
2. $\frac{1}{4}(u'_i - v'_i)^2 \sim Bernoulli(\frac{\arccos(\rho)}{\pi})$
3. $d'(u', v') := \frac{1}{4}\sum_{i=1}^{k}(u'_i - v'_i)^2 \sim Binomial(k, \frac{\arccos(\rho)}{\pi})$

4. *Additive bound: For any $\epsilon > 0$,*

$$\mathbf{Pr}\left(\frac{1}{k}d'(u',v') - \frac{\arccos(\rho)}{\pi} > \epsilon\right) \leq e^{-2\epsilon^2 k} \tag{2}$$

$$\mathbf{Pr}\left(\frac{1}{k}d'(u',v') - \frac{\arccos(\rho)}{\pi} < -\epsilon\right) \leq e^{-2\epsilon^2 k} \tag{3}$$

5. *Multiplicative bound: For any $\lambda \in [0,1]$,*

$$\mathbf{Pr}\left(\frac{1}{k}d'(u',v') \geq (1+\lambda)\frac{\arccos(\rho)}{\pi}\right) \leq e^{-\frac{\lambda^2 k \arccos(\rho)}{3\pi}} \tag{4}$$

$$\mathbf{Pr}\left(\frac{1}{k}d'(u',v') \leq (1-\lambda)\frac{\arccos(\rho)}{\pi}\right) \leq e^{-\frac{\lambda^2 k \arccos(\rho)}{2\pi}} \tag{5}$$

Proof. The first result is a fundamental property of random hyperplanes exploited in [11] and was later used to develop the random hyperplane hashing algorithm in [5]. This key property results in random hyperplane hashes satisfying the properties of a cosine hash family. The second and third results follow from 1. The fourth and fifth results are obtained by applying Hoeffding and Chernoff bounds respectively on $d'(u',v')$ by expressing it as a sum of independent Bernoulli distributed random variables $(u'_1 - v'_1)^2, (u'_2 - v'_2)^2, \ldots, (u'_k - v'_k)^2$. □

The key contribution of this work is the use of random hyperplane hashing as a representation scheme over which classifiers can be learnt. The efficient random hyperplane hashing algorithm leads to a general and compact angular distance preserving representation. In the next section, we exploit the properties of random hyperplane hashing provided in Lemma 1 to obtain bounds on the performance of linear classifiers learnt over such a representation and further, compare it with the closely related random projection algorithm.

3 Classifier over Random Hyperplane Hashes

We start with a formal definition of a linear classifier in both spaces, Euclidean space and random hyperplane hash space.

Definition 5 (Linear classifier and its hash representation). *Let T represent a concept that maps instances x_i from an m-dimensional space of reals $\mathbb{R}^m$ to labels y_i that belongs to $\{-1,+1\}$. The concept T allows for a linear classifier $h \in \mathbb{R}^m$, if there exists a h satisfying, $y_i(h^T x_i) \geq 0$ which can alternately be written as $y_i\left(\frac{1}{2} - \frac{1}{\pi}\arccos(\frac{h^T x_i}{|h||x_i|})\right) \geq 0$. Since, $h \in \mathbb{R}^m$, it allows for a random hyperplane hash representation h'. In the hash space, we consider linear classifiers of the form $\frac{y_i}{k}(h'^T x'_i) \geq 0$* [1].

[1] The $\frac{1}{k}$ is for convenience and results in the classifier output to be in the range $[-1,1]$.

Our first important error bound requires a stronger notion of separability, however. In the next subsection, we will show that a slightly less restrictive constraint is not sufficient.

Definition 6 (ϵ-robust concept). *For any real $0 < \epsilon < 0.5$, a concept T along with a distribution $\mathcal{D}$ on $\mathbb{R}^m$ is ϵ-robust, if it allows for a linear classifier h that satisfies $\frac{1}{2} - \frac{1}{\pi}\arccos(\frac{h^T x}{|h||x|}) > \epsilon$ for positive and $\frac{1}{2} - \frac{1}{\pi}\arccos(\frac{h^T x}{|h||x|}) < -\epsilon$ for negative instances.*

Theorem 1. *Let h be a linear classifier that can correctly classify instances in $\mathbb{R}^m$ according to some ϵ-robust target concept T and let h' denote its hash representation. Consider $x \in \mathbb{R}^m$ and its k dimensional random hyperplane hash x', for $k \geq \frac{\alpha}{2\epsilon^2}$ and projection matrix M. Then,*

$$\forall x : \mathcal{D}(x) > 0, \mathbf{Pr}_{M \sim [N(0,1)]^{m \times k}} \left[label_h(x) \neq label_{h'}(x') \right] \leq e^{-\alpha} \tag{6}$$

Proof. Consider a positive instance x. Let the random hyperplane hash of x be x'. By definition of ϵ robustness, h satisfies

$$\frac{1}{\pi}\arccos(\frac{h^T x_i}{|h||x_i|}) \leq \frac{1}{2} - \epsilon. \tag{7}$$

Now for $0 < \epsilon < 0.5$, let us compute the following probability,

$$\mathbf{Pr}\left(\frac{1}{k}d'(h', x') > \frac{1}{2} + \epsilon\right) \tag{8}$$

The Hoeffding bound from Lemma 1,

$$\mathbf{Pr}\left(\frac{1}{k}d'(h', x') - \frac{1}{\pi}\arccos(\frac{h^T x_i}{|h||x_i|}) > \epsilon\right) \leq e^{-2k\epsilon^2} \tag{9}$$

and Eqn. 7, with the definition of d', can be combined to obtain,

$$\mathbf{Pr}\left(\frac{1}{4k}\left(|h'|^2 + |x'|^2 - 2h'^T x'\right) - (\frac{1}{2} - \epsilon) > \epsilon\right) \leq e^{-2k\epsilon^2} \tag{10}$$

and using the property that $|h'| = |x'| = \sqrt{k}$ (by construction[2]) leads to,

$$\mathbf{Pr}\left(\frac{1}{k}h'^T x' < 0\right) \leq e^{-2k\epsilon^2} \tag{11}$$

which corresponds to the probability that x' is mislabeled as negative by h'. A similar result can be shown for negative instances which leads to the error bound in Eqn. 6. □

Using Theorem 6, Fig. 1 illustrates the lower bounds on hash length k required for different label error rate constraints. It is important to note that the bound is obtained based on the hash of the classifier. A classifier explicitly learnt in the hash space can only lead to a better margin.

[2] The conventional random hyperplane algorithm results in a k dimensional vector of 0s and 1s, but we construct a vector of -1s and 1s to allow for a constant norm of $\sqrt{k}$ for hash length k.

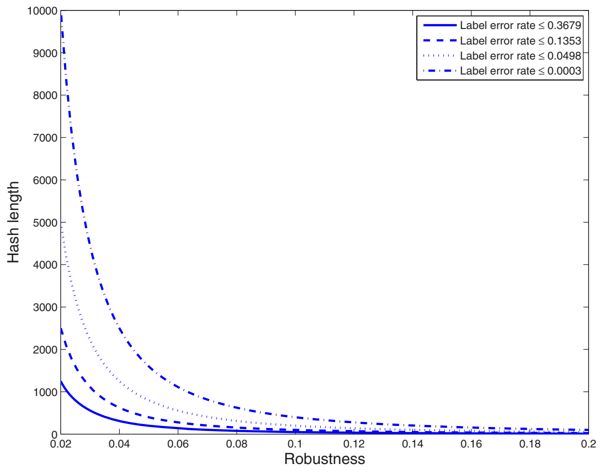

Fig. 1. Plot showing lower bounds on hash length for the random hyperplane hashing algorithm for different upper bounds on label error rates (set by varying α). The x-axis represents the level of robustness of the target concept in terms of ϵ.

3.1 Comparison with Random Projections

The method of random projections (RP) has been used extensively to perform dimensionality reduction [19] in a distance preserving manner. The random projection method is the same as the random hyperplane method described in Def. 1 without the thresholding performed in the second step. The thresholding step results in a key difference: Euclidean distance is preserved with high probability in the case of random projections [13] whereas the angular distance (Lemma 1) is preserved in the random hyperplane case. As mentioned before, cosine similarity is extensively used in domains like text and image retrieval. The work reported in [3] provides error bounds for classifiers learnt over random projections. The error bound (based on Lemma 3 from [3]), indicates the number of projections, $k_{RP} \geq \frac{8\alpha}{\epsilon^2}$ compared to our case where the number of hash elements $k_{RHH} \geq \frac{\alpha}{2\epsilon^2}$ while ensuring the same error bound $e^{-\alpha}$ for different notions of robustness. In [19], the authors define the notion of l-robustness which simply put, disallows instances with different labels to be less than l close to each other. Our notion of ϵ robustness is much stronger than the l-robustness. Next, we present empirical results to show the need for such a strong notion of robustness.

In this synthetic experiment, we demonstrate the label errors introduced by the random projection algorithm and random hyperplane hashing algorithm. Generate p different random hyperplanes $h_1, h_2, \ldots, h_p$. For every random hyperplane, we use the l robustness assumption used in [3] and generate N instances at random from $\mathbb{R}^m$ in such a way that no two instances with different labels are more than l close to each other. The labeling y_{ij} of each instance x_i is obtained

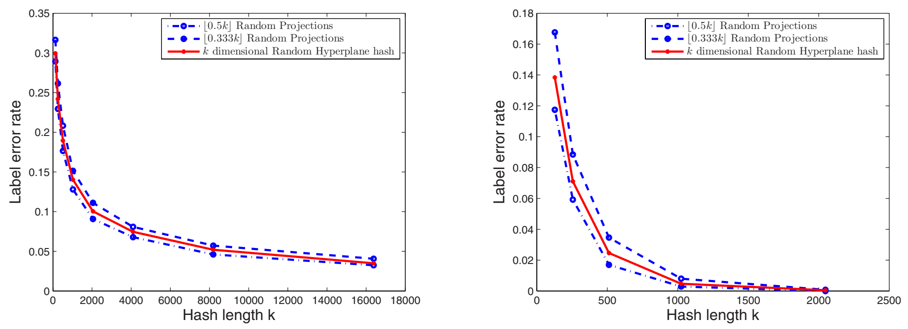

Fig. 2. Plot illustrating the drop in label error rate with increasing projections/hashlength where the classification obeys l-robustness (left plot) and ϵ-robustness (right plot)

based on $y_{ij}(h_j^T x_i) \geq 0$ [3]. We generate K different projection matrices of size $k \times m$ where each element is chosen i.i.d from $\mathcal{N}(0,1)$. The random projection and random hyperplane hashing algorithms are performed using all l projection matrices, on all x_{il}, h_{jl}, to obtain $\hat{x}_{il}$,$\hat{h}_{jl}$ and x'_{il}, h'_{jl} respectively. The labels in the hash space are obtained according to $\hat{y}_{ijl}(\hat{h}_{jl}^T \hat{x}_{il}) \geq 0$ and $y'_{ijl}({h'}_{jl}^T x'_{il}) \geq 0$. Now, we evaluate the label error rate given by, $\frac{1}{K \times N \times p} \sum_{i,j,l} I\left(y_{ij} \neq \hat{y}_{ijl}\right)$ and $\frac{1}{K \times N \times p} \sum_{i,j,l} I\left(y_{ij} \neq y'_{ijl}\right)$ for the random projection method and the random hyperplane case for different values of k. Fig. 2 (left) shows that, even with hashlength 16384, the label error rate is still around 4% which is clearly unexpected based on the lower bounds on hash length. The same inconsistency arises with the random projection case as well. The reason for this effect is that the notion of l-robustness does not account for disallowing instances that are very close to the hyperplane, which leads to high label error rates. In the following simulation, we enforce the ϵ-robustness constraint, which simply put, disallows instances to be too close to the hyperplane. We set ϵ to be 0.0319. The rest of the experimental setup is the same as the first experiment. Results presented in Fig. 2 (right) show a more desirable fast drop-off in label error rate and at the same time, comfortably satisfying the lower bounds presented in Theorem 6 and illustrated through Fig. 1.

It is important to note that one projection obtained through random projection requires a significantly larger chunk of memory compared to one bit required to represent one element of a random hyperplane hash; each hyperplane hash is just one bit long, whereas random projections are of continuous nature, and would usually be represented by 4 or even 8 bytes. The results from Fig. 2 (right) lead to another very interesting observation. The label error rate of the k-dimensional random hyperplane hashing is empirically bound by the label error rates of $\lfloor \frac{k}{2} \rfloor$ random projections and $\lfloor \frac{k}{3} \rfloor$ random projections. It also

[3] The random projection method and the random hyperplane algorithm are comparable only in the case of hyperplanes passing through the origin.

illustrates that less than 3 random hyperplane bits are more expressive than a "full" random projection that allocates multiple bytes. The random hyperplane algorithm obviously compresses the data better than random projections. In the experimental results section, we further substantiate the claim that this behavior also holds for real data.

On the technical side, it is worth noting that the elements of the projection matrix M used in Def. 4, are not required to be continuous in practice. As noted in [1] for the random projection case, sampling uniformly from $\{-1, +1\}$ instead of sampling from $\mathcal{N}(0, 1)$, yields very similar results due to the central limit theorem. In fact, unless the documents were very short, we found the results to be indistinguishable. This closes the gap between Def. 4 and Algorithm 1. Regarding preprocessing, random projections are cheaper. Even if we need only about 2.5 times more random hyperplane "bits" than random projections to get similarity approximations of the same quality, this still means that the algorithm has to generate more than twice as many random bits in the first place. This is required only once for each document, however, and whenever we expect to apply a large number of classifiers to that representation we may in return benefit from the cheaper complexity of the models: random hyperplane representations are just bits, so linear models just have to add associated weights. There is a trivial way of simplifying such classifiers: After transforming the decision function of the classifier from $\{-1, +1\}$ feature space to $\{0, +1\}$ space (which just requires to adapt the threshold) it only has to do an addition for every second bit on average. In contrast, random projections yield integer or real-valued vectors, which even requires additional multiplications. In practice, random hyperplanes should hence be cheaper, even if they operate on twice as many dimensions. When it comes to uploading models to clients, which happens only once per model-client pair, the higher dimensionality of the random hyperplanes algorithm is a disadvantage however, because models that get the same quality of results will usually have a larger number of associated weights.

Whether random hyperplanes or random projections are favorable depends on an application-dependent weighting of CPU and memory footprint, the expected number of examples to be classified per model, the number of examples to be uploaded, and the expected frequency of model upgrades.

3.2 Large Margin Classifiers over Random Hyperplane Hashes

The results previously derived in this section show that the preservation of labels in the random hyperplane hash space is closely related to the notion of a margin. In this subsection, we consequently analyze how margins of SVM classifiers in the hash space are affected by the hash length. Further, we determine the hash length which will preserve the margin with high probability. In [14], the authors present such a theorem on the effect of learning large margin classifiers over random projections. We first require a few more definitions.

Definition 7 (SVM classifiers, margin and angular margin). *Consider a linearly separable data set* $S = \{(x_i, y_i), i = 1 \ldots n\}, x_i \in \mathbb{R}^m, y_i \in \{+1, -1\}$*, the SVM optimization problem is set up as follows:*

$$\max_h \frac{1}{|h|} \; s.t. \; y_i(h^T x_i) \geq 1, i = 1, \ldots, n \tag{12}$$

where, $\frac{1}{|h|}$ *is defined as the margin and hence, the solution to the above optimization problem is termed the maximum margin solution. Traditionally, an offset is included in the constraint and we address this issue at the end of this section.*

For the sake of convenience, we define the angular margin l_a*, as*

$$l_a := \min_i \, y_i \left(\frac{1}{2} - \frac{1}{\pi} \arccos \frac{h^T x_i}{|h||x_i|} \right) \tag{13}$$

where we have exploited the fact that a classifier satisfying $y_i(h^T x_i) \geq 0$ *can be rewritten as,* $y_i \left(\frac{1}{2} - \frac{1}{\pi} \arccos(\frac{h^T x_i}{|h||x_i|}) \right) \geq 0$.

These definitions allow us to formulate the desired probabilistic bound on the margin:

Theorem 2. *Given a linearly separable data set* S *(*$|S| = n$*), let the optimal maximum margin solution on* S *be given by* h *with angular margin* l_a*. Let* S'_k *represent the set of random hyperplane hashes of instances in the set* S*. If* $k \geq \frac{\frac{1}{2}+l_a}{\gamma^2 l_a^2} \log \frac{n}{\delta}$*,* $0 < \gamma < 1$*,* $0 < \delta < 1$*, then the maximum margin solution on* S'_k *satisfies the following bound,*

$$\mathbf{Pr}(l_p > (1-\gamma) l_a) \geq 1 - \delta \tag{14}$$

where l_p *corresponds to the margin in the hash space.*

Proof. Consider a positive example x, and in particular we consider the worst case where the instance lies on the margin i.e., $\frac{1}{2} - \frac{1}{\pi} \arccos \frac{h^T x}{|h||x|} = l_a$. Let x' and h' represent the k-dimensional random hyperplane hash of x and the optimal solution h respectively. Now, let us compute the following probability:

$$\mathbf{Pr}\left(\frac{1}{k} d'(h', x') < \frac{1}{2} - (1-\gamma) l_a \right) \tag{15}$$

which can be reformulated in a more amenable multiplicative form:

$$\frac{1}{k} d'(h', x') < \frac{1}{2} - (1-\gamma) l_a \equiv \frac{1}{k} d'(h', x') < (1+\lambda) \left(\frac{1}{2} - l_a \right) \tag{16}$$

Solving for λ, we obtain $\lambda = \frac{\gamma l_a}{\frac{1}{2} - l_a}$. Eqn. 15, can be rewritten as:

$$\mathbf{Pr}\left(\frac{1}{2k} h'^T x' > (1-\gamma) l_a \right) = \mathbf{Pr}\left(\frac{h'^T x'}{\sqrt{k}} > 2\sqrt{k}(1-\gamma) l_a \right). \tag{17}$$

Using the multiplicative Chernoff bounds from Lemma 1, we obtain the following bound:

$$\begin{aligned}
\mathbf{Pr}\left(\frac{h'^T x'}{\sqrt{k}} > 2\sqrt{k}(1-\gamma)l_a\right) &= 1 - \mathbf{Pr}\left(\frac{1}{k}d'(h',x') > \frac{1}{2} - (1-\gamma)l_a\right) \\
&> 1 - e^{-\frac{\gamma^2 l_a^2}{\frac{1}{2}-l_a}\frac{k}{3}} \\
\Rightarrow \mathbf{Pr}\left(\frac{h'^T x'}{\sqrt{k}} > (1-\gamma)l_a\right) &> 1 - e^{-\frac{\gamma^2 l_a^2}{\frac{1}{2}-l_a}\frac{k}{3}} \qquad (18)
\end{aligned}$$

Similarly for any negative instance x', we obtain the bound:

$$\begin{aligned}
\mathbf{Pr}\left(\frac{h'^T x'}{\sqrt{k}} < -(1-\gamma)l_a\right) &= 1 - \mathbf{Pr}\left(\frac{1}{k}d'(h',x') < \frac{1}{2} + (1-\gamma)l_a\right) \\
&> 1 - e^{-\frac{\gamma^2 l_a^2}{\frac{1}{2}+l_a}\frac{k}{2}} \qquad (19)
\end{aligned}$$

By definition, the margin in the hash space expressed as l_p is given by $\min y' \frac{h'^T x'}{\sqrt{k}}$. The union bound can be used on Eqns. 18 and 19, to obtain guarantees of a hyperplane in the hash space with a margin l_p that is at least $(1-\gamma)l_a$ where

$$\gamma \leq \frac{\epsilon\sqrt{l_a + \frac{1}{2}}}{l_a} \qquad (20)$$

with probability at least $1 - ne^{-\epsilon^2 \frac{k}{3}}$. The corresponding value of k is given by:

$$ne^{-\frac{\gamma^2 l_a^2}{\frac{1}{2}+l_a}\frac{k}{3}} \leq \delta \Rightarrow k \geq \frac{\frac{1}{2}+l_a}{\gamma^2 l_a^2}\log\frac{n}{\delta} \qquad (21)$$

Notice that the sub-optimal h', which corresponds to the random hyperplane hash of the optimal classifier h achieves the bound. So, the optimal classifier on S'_k can only achieve a better margin than h'. □

Finally, we want to justify for not having included an offset term into Eqn. (12) for mathematical simplicity. The common SVM constraint is given by

$$y_i(h^T x_i - b) \geq 1, i = 1, \ldots, n. \qquad (22)$$

It is slightly more expressive in theory, but we found the difference in practice to be negligible. However, in doubt we can always include a "constant" component to each example vector, e.g. a new first component which is always equal to 1. In this case, the vector h effectively also provides the offset b. The only remaining difference to the regular SVM is then that having the offset b inside h makes it part of the margin. In the case of high-dimensional data we are discussing in this paper this obviously does not change matters too much. In the next section we will complement this argument with empirical evidence that the hashing-based representation is well suited for building text classifiers, even without spending any particular attention to the matter of offsets.

Table 1. LibSVM results for a number of multi-class classification problems previously used in the context of feature selection. Reported are 10fold cross-validated accuracies on different representations and corresponding standard deviations.

dataset	**classes**	**size**	**BOW**	**8192 RHH**	**2048 RHH**	**2048 RP**
Cora	36	1800	49.50 ± 2.24	66.56 ± 2.51	58.00 ± 3.45	52.22 ± 3.42
FBIS	17	2463	85.87 ± 1.61	84.94 ± 2.08	83.92 ± 1.29	84.65 ± 1.91
LA1	6	3204	90.20 ± 1.34	89.95 ± 1.32	85.55 ± 1.96	87.42 ± 0.60
LA2	6	3075	90.51 ± 2.28	90.41 ± 1.92	86.86 ± 1.70	87.51 ± 1.98
OH0	10	1003	89.53 ± 3.75	88.83 ± 2.69	87.13 ± 2.68	87.63 ± 2.65
OH10	10	1050	81.43 ± 3.79	81.71 ± 2.44	76.86 ± 2.37	78.19 ± 2.71
OH15	10	913	81.49 ± 2.69	81.71 ± 3.54	79.62 ± 5.09	80.83 ± 5.12
OH5	10	918	87.69 ± 3.76	88.57 ± 3.43	83.33 ± 3.80	85.08 ± 3.88
RE0	13	1504	84.17 ± 2.69	84.31 ± 2.56	83.04 ± 2.86	82.44 ± 2.45
RE1	25	1657	84.55 ± 2.32	80.87 ± 2.80	79.85 ± 3.01	82.62 ± 2.77
WAP	20	1560	85.38 ± 1.34	82.95 ± 2.84	80.06 ± 2.49	82.50 ± 2.48
OHSCAL	10	11162	78.07 ± 0.93	78.31 ± 1.14	71.58 ± 1.19	74.82 ± 0.97

4 Experimental Results

The main goal of the subsequently described experiments is to complement the theoretical guarantees by empirical evidence that for typical text classification tasks hashing-based representations are in fact able to capture enough of the similarities between documents to allow for competitive predictive classifier performance. More detailed questions of interest include how much information is lost when switching to less exact representations and how this loss qualitatively changes with the number of bits we allow for each of the representations. We stress that the experiments do *not* aim at reaching the best individual classifier performances reported for each dataset. We are hence not applying any sophisticated preprocessing methods and avoid the issue of parameter tuning by fixing the experimental setup. We just vary the representations of input data sets.

The representations we want to evaluate are random hyperplane hashing (RHH) and random projections (RP). We compare the classifiers learnt over these representations to the classifiers learnt over the regular bag of words vector space representation of text (BOW). In order to raise the bar of the baseline method, we also report results when classifying on top of a BOW representation after feature selection. We do not want to narrow down the setting to the somewhat unnatural case of strictly binary classifiers, so all of our experiments involve multi-class problems. For this kind of data, Information Gain feature selection is a widely used option.

All experimental results reported in this section are the result of tenfold cross-validation. We chose the LibSVM operator that is part of RapidMiner [15] as an inner learner, because it is capable of building competitive logistic models on top of SVM classifiers for multi-class problems. We used a linear kernel with default settings, and fixed the C parameter to 0.1 for all experiments to establish a common baseline. Please note that this low value of C introduces a slight

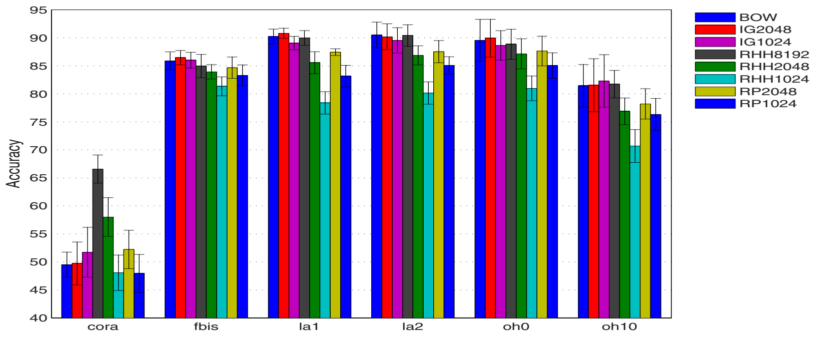

Fig. 3. Accuracies and standard deviations of different methods on benchmark datasets

disadvantage for hashing-based representations, since the original vector space contains up to $30,000+$ dimensions and for the most part much fewer data points in our experiments; the data can be expected to be linearly separable in the original space, but after the noisy projections there might be a considerable number of examples on the wrong side of the hyperplane, which requires a larger value of C. In our experimental setup we allow for the baseline information gain based feature selection to select features on the complete data set (including the test set) which results in a minor disadvantage for the hashing representation.

4.1 Medium-Sized Corpora

For brevity, we first report experiments on a *set* of publicly available benchmark data sets, namely corpora that were used in [8] to evaluate feature selection techniques[4]. Compression is not a concern for data sets at this scale, but they allow for estimates of how many projections or RHH bits are required to preserve similarities in hash space sufficiently well for classification purposes.

Table 1 lists the data sets and summarizes the results for the plain bag of words representation, random hyperplane hashing with 8192 and 2048 bits (1 KByte and 256 Bytes), and random projections with 2048 projections. We report results for random projections where we encoded each projection by 2 bytes, which means that the representation using only 2048 projections is still four times longer than the 8192 RHH bit representation. Figures 3 and 4 show the same results and in addition compare to Information Gain with 2048 and 1024 dimensions and 1024 random projections.

The results of our experiments suggest that 8192 RHH bits are enough to represent documents based on hashes with a negligible loss in classification accuracy (see bar plots). Only in rare cases, the RHH representation and the vector space model give significantly different results. Examples include the Cora data

[4] It is available at the JMLR website:
`http://www.jmlr.org/papers/volume3/forman03a/forman03a_data.tgz`.

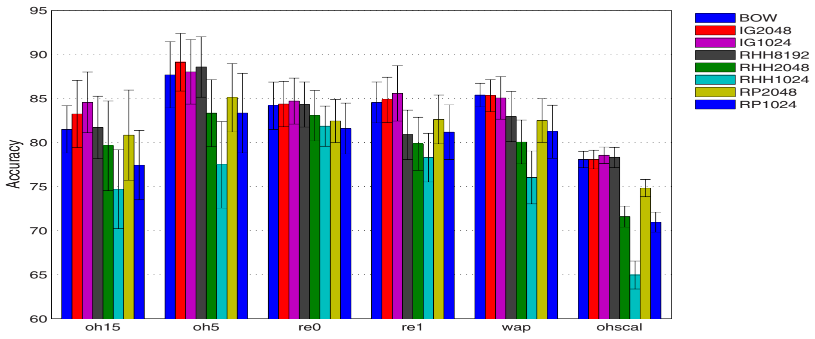

Fig. 4. Accuracies and standard deviations of different methods on benchmark datasets

set, where RHH performs much better, and RE1, where the original representations and feature selection perform better. Using only 2048 RHH bits reduces the accuracy typically by about 3 to 4%. Keeping the number of classes and the generality of the representation in mind, this is still a decent result. Interestingly, reducing the number of bits even further to 1024 (factor of 2) the performance drops more than when we reduced it by a factor of 4 before. This reflects the exponentially decreasing quality of approximations.

Using random projections in the order of 200 dimensions, a case that has been studied in the literature, clearly does not serve the purpose. It is interesting to note though that even if we assume just 4 bytes per projection (float, single precision) we end up with 8192 bits representing only 256 dimensions. In our experiments we generally found that – depending on whether accuracy or shorter representations are more important – choices between 100 bytes and 1KByte seem to work reasonably well. A pleasant property of the hashing-based representation we found is that the accuracy always increased monotonically with an increasing number of features. This does not hold for feature selection.

4.2 Experiments with Larger Corpora

The practically more relevant task is to scale up the sketched method to data sets of very high dimensionality and size. To study the effects with larger corpora we hence ran experiments on the widely used 20 newsgroups data set and on a hand-crafted corpus that contained examples of categorized web pages. The former contains about 20K examples, 40K terms, and 20 different categories. For the latter we crawled pages from the different sports subcategories of the Open Directory (DMOZ). We removed all non-textual parts from the web pages, and tokenized and Porter stemmed the plain text. To reduce noise, we deleted pages with less than 2 KBytes. Finally, we removed all categories with less than 20 remaining documents, leaving about 30K documents, 200K terms, and 52 skewed classes for the evaluation. The largest category is "Soccer" with 4858 documents. Due to the high dimensionality of the original corpus, we only compared the performance of random hyperplane projections to BOW after feature selection. For 20 newsgroups, the information gain based selection of 1024 features gave an accuracy of 77.11% $\pm$ 1.82; increasing the number of selected features up to 8192 reduced the accuracy. The same LibSVM classifier on a 2048 RHH bit representation still gave 73.26% $\pm$ .64 accuracy. The representation that gave best results used 8192 RHH bits; the accuracy was 83.28 $\pm$ .62 in that case.

On the sports corpora, varying the number of selected features between 2048 and 8192 did not change the performance much. The best result on information gain feature selection was 90.99% $\pm$ .5. With only 1024 bits of RHH data the same classifier still reached an accuracy of 84.75% $\pm$ .45, on 2048 bits it reached 88.68 $\pm$ .37. We got the best result of 91.66 $\pm$ .41 using 8192 RHH bits.

The results on both these tasks are very competitive in terms of predictive performance. They illustrate that global feature selection can be avoided without compromising accuracy, e.g., if transmitting raw data to a central server for learning is prohibitively expensive in client-server settings.

5 Conclusion

We studied a specific locality sensitive hashing scheme called random hyperplane hashing as a representation for building linear classifiers. It differs from random projections in that it preserves angular rather than Euclidean distances. A margin-based robustness criterion allowed us to both upper-bound the error rate and to lower-bound the margin of the resulting classifier in the hash-space in a strong probabilistic sense. We illustrated that (i) a certain pre-defined weaker notion of robustness fails to achieve desirably low label error rates, but that (ii) label errors when enforcing our margin-based robustness criterion decrease exponentially fast with increasing hash length (dimensionality). Moreover, we showed that less than 3 random hyperplane hash bits are as expressive in terms of preserving labels as a full, real-valued random projection.

The error and margin bounds coupled with the high density and expressiveness motivated an evaluation of the hash representation for learning text classifiers in a client-centered setting where the representation length of documents and models as well as the computational costs of classifiers are crucial. We demonstrated significant gains in representation length over bag of words and the random projection method, and a classification performance that is competitive to standard feature selection, despite using a generic, content-agnostic preprocessing scheme.

Acknowledgments

We would like to thank Kave Eshghi, Rajan Lukose, George Forman, Ram Swaminathan, and Jaap Suermondt for several fruitful discussions regarding this work. We would also like to thank an anonymous reviewer for helpful comments.

References

1. Achlioptas, D.: Database-friendly random projections. In: Symposium on Principles of Database Systems (PODS 2001), pp. 274–281. ACM Press, New York (2001)
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Symposium on Foundations of Computer Science (FOCS 2006), pp. 459–468. IEEE Computer Society, Los Alamitos (2006)
3. Arriaga, R.I., Vempala, S.: An algorithmic theory of learning: Robust concepts and random projection. In: IEEE Symposium on Foundations of Computer Science, pp. 616–623 (1999)
4. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Int. Conf. on Knowledge Discovery and Data Mining (KDD 2001), pp. 245–250. ACM Press, New York (2001)
5. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Symposium on Theory of computing (STOC 2002), pp. 380–388. ACM Press, New York (2002)
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Symposium on Computational geometry (SCG 2004), pp. 253–262. ACM Press, New York (2004)
7. Eshghi, K., Rajaram, S.: Locality-sensitive hash functions based on concommitant rank order statistics. In: Int. Conf. on Knowledge discovery and data mining (KDD 2008). ACM Press, New York (2008)
8. Forman, G.: An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research (JMLR) (3), 1289–1305 (2003)
9. Fradkin, D., Madigan, D.: Experiments with random projections for machine learning. In: Int. Conf. on Knowledge discovery and data mining (KDD 2003), pp. 517–522. ACM Press, New York (2003)
10. Goel, N., Bebis, G., Nefian, A.: Face recognition experiments with random projection. In: SPIE, Bellingham, WA, pp. 426–437 (2005)
11. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42(6), 1115–1145 (1995)
12. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Symposium on Theory of computing (STOC 1998), pp. 604–613 (1998)
13. Johnson, W., Lindenstrauss, J.: Extensions of lipschitz maps into a hilbert space. Contemporary Mathematics 26, 189–206 (1984)
14. Kumar, K., Bhattacharya, C., Hariharan, R.: A randomized algorithm for large scale support vector learning. In: Advances in Neural Information Processing Systems (NIPS 2007), pp. 793–800. MIT Press, Cambridge (2008)
15. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid prototyping for complex data mining tasks. In: Int. Conf. on Knowledge discovery and data mining (KDD 2006). ACM Press, New York (2006)
16. Ravichandran, D., Pantel, P., Hovy, E.: Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering. In: Association for Computational Linguistics (ACL 2005), pp. 622–629 (2005)
17. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (1975)
18. Singh, K., Ma, M., Park, D.W.: A content-based image retrieval using FFT & cosine similarity coefficient. Signal and Image Processing (2003)
19. Vempala, S.: The Random Projection Method. American Mathematical Society (2004)