Fruit-fly Inspired Neighborhood Encoding for Classification

Kaushik Sinha*
kaushik.sinha@wichita.edu
Wichita State University
USA

Parikshit Ram*
parikshit.ram@ibm.com
IBM Research AI
USA

ABSTRACT

Inspired by the fruit-fly olfactory circuit, the Fly Bloom Filter [4] is able to efficiently summarize the data with a single pass and has been used for novelty detection. We propose a new classifier that effectively encodes the different local neighborhoods for each class with a per-class Fly Bloom Filter. The inference on test data requires an efficient `FlyHash` [6] operation followed by a high-dimensional, but *very sparse*, dot product with the per-class Bloom Filters. On the theoretical side, we establish conditions under which the predictions of our proposed classifier agrees with the predictions of the nearest neighbor classifier. We extensively evaluate our proposed scheme with 71 data sets of varied data dimensionality to demonstrate that the predictive performance of our proposed neuroscience inspired classifier is competitive to the nearest-neighbor classifiers and other single-pass classifiers.

CCS CONCEPTS

• **Computing methodologies** → **Bio-inspired approaches**; **Supervised learning by classification**; **Online learning settings**.

KEYWORDS

nearest-neighbor classification; randomized algorithm; bio-inspired

ACM Reference Format:
Kaushik Sinha and Parikshit Ram. 2021. Fruit-fly Inspired Neighborhood Encoding for Classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467246

1 INTRODUCTION

Neural circuits in the fruit-fly appear to assess the novelty of an odor in a two step process. An odor is first assigned a "tag" that corresponds to a small set of Kenyon Cells (KC) that get activated by the odor. Dasgupta et al. [6] interpret this tag generation process as a hash scheme, termed `FlyHash`, where the tag/hash is a very sparse point in a high dimensional space (2000 dimensions with 95% sparsity). The tag serves as input to a specific mushroom body output neuron (MBON), the MBON-$\alpha'3$, where the response of this neuron to the odor hash encodes the odor novelty. Dasgupta et al. [4] "interpret the KC→MBON-$\alpha'3$ synapses as a Bloom Filter" that effectively "stores" all odors exposed to the fruit-fly. This *Fly Bloom Filter* (`FBF`) generates continuous valued, distance and time sensitive novelty scores that have been empirically shown to be highly correlated to the ground-truth relative to other Bloom Filter-based novelty scores for neural activity and vision data sets. Theoretically, expected novelty scores of similar and dissimilar points have been analyzed for binary and exponentially distributed data.

This "learning" of the data distribution (for the purposes of novelty detection) has some interesting dynamics. First, the `FBF` encodes the data distribution in a *single-pass* manner without requiring to visit the same example twice – the relevant information for an example has been "stored" in the `FBF`– surfacing two advantages: (i) once processed, there is no need to retain an example in memory, allowing the encoding without much memory overhead, (ii) this mechanism allows the `FBF` encoding to happen in an *online* manner as more examples are seen. Second, this learning does not involve any explicit "loss minimization" or related gradient based optimization – the learning dynamic mimics a natural gradient-free process. Finally, the `FBF` learning can be accomplished *solely with additions,* `OR`*s and sort* – no complex mathematical operations are required.

Driven by immense empirical advantage, the current trend is of generating huge (deep learning) models with sophisticated learning procedures using complex novel compute hardware, mathematical operators and multiple epochs over the training examples. Inspired by the simplistic natural learning procedure, we pursue an opposite direction – we explore the extent to which *a simplistic learning procedure is successful in supervised classification.* Moreover, we believe the single-pass, online learning is critical in many situations such as (a) the learning happens in *low-memory "edge" devices* where we cannot retain a training set to repeatedly revisit, or (b) the examples have to be *discarded after a short amount of time due to privacy concerns.* There are situations where both reasons are valid – models need to be regularly updated with new data, but the retraining cannot access old data due to privacy regulations. This brings us to the questions we wish to address in this paper:

- *Can we devise a supervised classification scheme based on the simple learning dynamics of the* `FBF`*?*
- *Will such a supervised classification scheme be useful and competitive when learning needs to happen with a single pass?*
- *What generalization guarantees would such a learner have?*

To this end, we propose a **simple** algorithm using the `FBF` for classification, where we summarize each class with its own `FBF` and utilize the per-class novelty scores for inference. We theoretically study why this simple idea works, and empirically demonstrate that *simplicity does not preclude utility.* Specifically, we present:

- A novel `FBF` based classifier, `FBFC`, that is learned with an *additions-only, single-pass* of the training set without *any loss-minimizing*

*Both authors contributed equally to this research.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467246

Table 1: Properties of FBFC contrasted against standard machine learning models, namely, k-nearest-neigbor classifier (kNNC), prototype-based classifiers (CC, CC1), locality sensitive hashing based bloom filters (SBFC), linear models (LR), multi-layer perceptrons (MLPC), decision tree models (DT) and kernel machines (KM).

Classifiers	kNNC	CC1	CC	SBFC	LR	MLPC	DT	KM	FBFC
Single Pass		✓	✗	✓	✓ [a]	✓ [a]	✗	✓ [b]	✓
Infer w/o training data	✗	✓	✓	✓	✓	✓	✓	✓ [c]	✓
Online/streaming	✗	✓	✓ [d]	✓	✓	✓	✗	✓ [b]	✓
Parallel train		✓	✓	✓	✓ [e]	✓ [e]	✓ [e]	✓ [e]	✓
Gradient free learning	✓	✓	✓	✓	✗	✗	✓ [f]	✗	✓
Addition only training		✗	✗	✗	✗	✗	✗	✗	✓
Biologically inspired	✗	✗	✗	✗	✗	✓	✗	✗	✓

[a] A single pass generates a model that can be used. [b] For RBF & Polynomial kernels, randomized embeddings allow for approximate kernel learning to generate a model with a single pass. But it is not possible in general. [c] For RBF & Polynomial kernels, approximate kernel learning with randomized embeddings remove the need for the training data at inference. But it is not possible in general. [d] Approximate clustering with more than a single cluster is possible with streaming data. [e] Data-parallel training is possible but the optimization is either approximated or the objective is modified. [f] Decision trees perform a gradient-free combinatorial optimization; gradients are needed for gradient boosted decision trees.

optimization, and can be inferred from with an *efficient* FlyHash [6] followed by a *sparse* binary additions-only dot-product.

- A thorough empirical comparison of FBFC to standard classifiers on over 71 data sets in the single-pass learning setup, demonstrating significant gains over other single-pass schemes.
- A theoretical examination of the proposed scheme, establishing conditions under which FBFC *agrees* with the nearest-neighbor classifier, thereby inheriting its generalization guarantees.
- How the FBFC can *provide insights* into the problem structure in terms of a class hierarchy in classification problem.

The paper is organized as follows: We discuss related work in §2. We detail our proposed scheme and analyze its theoretical properties in §3. We empirically evaluate FBFC against baselines in §4 and conclude with a discussion in §5.

2 RELATED WORK

Neuroscience inspired techniques are now widely accepted in artificial intelligence to great success [14], especially in the field of deep learning with convolutional neural networks [18, 21], dropout [15] and attention mechanisms [22, 23] to name a few. Much like most machine learning methods, deep learning relies on loss-gradient based training in most cases. In contrast, our proposed FBFC learning does not explicitly minimize any "loss" function. Moreover, rather than learning a representation for the points that facilitates classification, the FBFC learns a representation for entire classes, allowing test points to be compared to classes for inference.

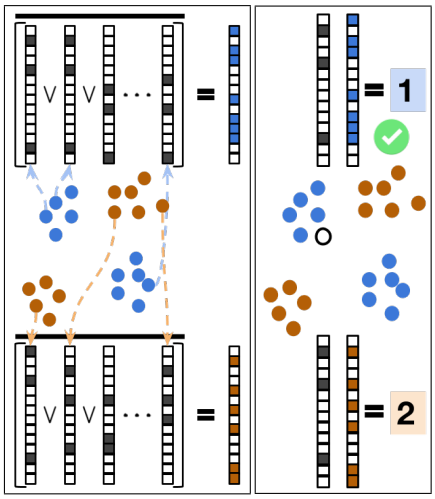
Given the correlation between a point x's FBF novelty score to its minimum distance from the set that the FBF summarizes [4], our proposed FBFC is perhaps closest to the nonparametric k-nearest-neighbor classifier (kNNC). Vanilla kNNC does not have an explicit loss or a training phase given a measure of similarity; all the computation is shifted to inference. FBFC does have an explicit training phase, but requires only a single pass of the training data – once a point is processed into the FBF, it can be discarded, making FBFC suitable for online learning with streaming data.

On a very high level, this is similar to cluster-based kNNC where class specific training data (data with same labels) is summarized as (multiple) cluster centers and used as a reduced training set on which kNNC is applied – this is also known as prototype-based classification (CC), with the simplest form where there is a single cluster/prototype per class (CC1). A variety of methods exists in literature that adopt this simple idea of data reduction [9, 11, 24, 25, 27, 35]. These algorithms are designed with the goal of reducing the high computational & storage requirements of kNNC. Orthogonally, various data structures have been utilized to accelerate the nearest-neighbor search in kNNC inference representing the data as an index such as space-partitioning trees [2, 5, 26, 31] and hash tables generated by *locality-sensitive* hashes [1, 10].

The closely related locality-sensitive Bloom filter (LSBF) [16, 20] also summarizes the data similar to FBF, relying on distance preserving random projection [34] to lower dimensions followed by quantizing the projected vector to an integer. Under this scheme, two inputs reset the same bit in the filter if they are assigned the exact same projected vector. Performance of LSBF heavily depends on the choice of hyper-parameters that control the projection dimensionality and the data-independent quantization scheme. FBF has been shown to empirically outperform LSBF for novelty detection [4]. LSBF can also be used for supervised classification in the same way we extend the use of FBF which we call SBFC.

Multinomial regression with linear models (LR) and multi-layered perceptrons (MLPC) can also be viewed as learning a set of weight vectors corresponding to each class, with the inner product of the test point with these vectors driving the class assignment. Kernel machines (KM) are generalizations of linear models to a (implicit) higher dimensional space where the inner product is defined by a pairwise kernel function. These learn weight vectors in this implicit kernel space. In general, these are not suited for single-pass learning, and require the training set (or a subset of it known as the support vectors) for inference. However, the seminal work of Rahimi and Recht [30] made KM more scalable with randomized explicit embeddings that allow for approximate kernel learning, making KM suitable for single-pass online settings and removing the need for the training set at inference. These approximations exist for radial basis function (RBF) kernels and polynomial kernels [13, 17, 29] but are not generally available for all kernel functions. Moreover, low levels of approximations often have a high memory overhead.

A comparison of our proposed method with the (related) existing methods across a wide variety of desirable properties is summarized in Table 1. We emphasize that in comparison to the existing methods, our biologically inspired proposed FBFC is simple to implement as it is gradient & optimization free and requires addition only operations while being single pass, and adaptive to streaming data all at the same time. Additionally, as we show in §3.2, the theoretical predictive performance of FBFC agrees with that of a



(a) FBFC training. (b) FBFC inference.

Figure 1: Visual depiction of FBFC training and inference (Algorithm 1). Colored circles are the labeled training set. In Figure 1a, the high dimensional sparse FlyHashes for the points (stacked ■ & □) in each class are used to generate the per-class FBF (NOT $\overline{(\cdot)}$ of the ORs $\vee$ of the hashes as per (2)). The ○ in Figure 1b is the unlabeled point we infer on based on the dot-product of its FlyHash with each of the per-class FBFs (3). *Please view in color.*

non-parametric classifier, 1NNC. But like parametric methods, it does not require access to training data during inference.

3 FlyHash BLOOM FILTER CLASSIFIER (FBFC)

The basic building block of our proposed algorithm is a fruit-fly olfactory circuit inspired FlyHash function, first introduced by Dasgupta et al. [6]. Here we consider the binarized FlyHash [4]. For $x \in \mathbb{R}^d$, the FlyHash function $h\colon \mathbb{R}^d \to \{0,1\}^m$ is defined as,

$$h(x) = \Gamma_\rho(M_m^s x), \tag{1}$$

where $M_m^s \in \{0,1\}^{m\times d}$ is the randomized sparse lifting binary matrix with $s \ll d$ nonzero entries in each row, and $\Gamma_\rho : \mathbb{R}^m \to \{0,1\}^m$ is the winner-take-all function converting a vector in $\mathbb{R}^m$ to binary one in $\{0,1\}^m$ by setting the highest $\rho \ll m$ elements to 1 and the rest to zero. Unlike random projection [34] which decreases data dimensionality after projection, FlyHash is an upward projection or a *lifting* which increases data dimensionality ($m \gg d$). The hyper-parameters for FlyHash are (i) the lifted dimensionality $m \in \mathbb{N}$, (ii) lifting matrix nonzero count per row $s \in \mathbb{N}$, and (iii) the number of nonzeros (NNZ) $\rho \in \mathbb{N}$ in the FlyHash. The FlyHash has been shown to be locality sensitive – similar points $x, x' \in \mathbb{R}^d$ will have matching nonzero elements in their hashes $h(x), h(x')$.

Using FlyHash as an algorithmic building block, Dasgupta et al. [4] construct a "Fly bloom filter" (FBF) $w \in \{0,1\}^m$ to succinctly summarize the data, and use it to effectively solve the unsupervised learning task of novelty detection, with the novelty score set as $w^\top h(x)$. Starting with $w = \mathbf{1}_m$ (the vector of all ones), for an "inlier" point x_{in}, the FBF encoding w is updated by zeroing the

Algorithm 1: FBFC training with labeled set $S \subset \mathbb{R}^d \times [L]$, lifted dimensionality $m \in \mathbb{N}$, NNZ per row in the lifting matrix $s \ll d$, NNZ in FlyHash $\rho \ll m$, and test point x.

1 **TrainFBFC:** $(S, m, \rho, s) \to (M_m^s, \{w_l, l \in [L]\})$
2 Sample $M_m^s \in \{0,1\}^{m\times d}$ with s NNZ/row
3 Initialize $w_1, \ldots, w_L \leftarrow \mathbf{1}_m \in \{0,1\}^m$
4 **for** $(x, y) \in S$ **do**
5 $h(x) \leftarrow \Gamma_\rho(M_m^s x)$ // Γ_ρ top-ρ WTA
6 $w_y \leftarrow w_y \wedge \overline{h(x)}$
7 **end**
8 **return** $(M_m^s, \{w_l, l \in [L]\})$
9 **end**
10 **InferFBFC:** $(x, M_m^s, \rho, \{w_l, l \in [L]\}) \to \hat{y}$
11 $h(x) \leftarrow \Gamma_\rho(M_m^s x)$
12 $\hat{y} \leftarrow \text{argmin}_{l\in[L]}(1/\rho)\, w_l^\top h(x)$
13 **return** $\hat{y}$
14 **end**

elements in w corresponding to the indices of the nonzero elements in $h(x_{\text{in}})$ (the FlyHash of x). This ensures that some $x = x_{\text{in}}$ or similar to x_{in} receives a low novelty score $w^\top h(x)$. For a novel point x_{nov} with FlyHash $h(x_{\text{nov}})$, which is not similar to any of the inliers, the indices of the nonzero elements of $h(x_{\text{nov}})$ will, with high probability, **not be set to zero in** w (so the element in w will be one), implying a high novelty score $w^\top h(x_{\text{nov}})$.

Given the high dimensionality $m \gg d$ of the FlyHash $h(x)$ and its potentially high sparsity (only $\rho \ll m$ nonzero elements in $h(x)$), the FBF w in $\{0,1\}^m$ can be a sparse encoding of the data distribution and motivates potential use in supervised classification – we can encode each individual class $l \in [L] = \{1, \ldots, L\}$ with its own FBF w_l – we posit that the FBF encodings would provide large inter-class separation on account of their very high dimensional and sparse representation. In this section, we will detail how we extend the use of FBF to supervised classification, discussing the learning and inference mechanisms in §3.1, and the theoretical guarantees for the presented algorithms to §3.2.

3.1 Learning & inference mechanics

Here we extend the use of FBF to classification, an instance of supervised learning. Specifically, we use FBF to summarize each class separately – the per-class FBF encodes the local neighborhoods of each class, and the high dimensional sparse nature of FlyHash (and consequently FBF) summarizes classes with multi-modal distributions while mitigating overlap between the FBFs of other classes.

3.1.1 FBFC *training.* Given a training set $S \subset \mathbb{R}^d \times [L]$, the learning of the per-class FBFs $w_l \in \{0,1\}^m, l \in [L]$ is detailed in the TrainFBFC subroutine in Algorithm 1. The FlyHash is a fundamental building block. We initialize the FlyHash by randomly generating the sparse binary $(m \times d)$ *lifting* matrix M_m^s with only s nonzero entries in each row of the matrix (line 2). The per-class FBF w_l are initialized to $\mathbf{1}_m \in \{0,1\}^m$, the all one vector (line 3).

For a training example $(x, y) \in S$ with point $x \in \mathbb{R}^d$ and label $y \in [L]$, we first generate the FlyHash $h(x)$ (line 5). Then, the FBF w_y (corresponding to x's class y) is updated with $h(x)$ as follows – the bit positions of w_y corresponding to the nonzero bit positions of $h(x)$ are set to zero, and the remaining entries of w_y are left as

is (line 6); the remaining FBFs $w_l, l \neq y \in [L]$ are not updated at all. This ensures that x (and points similar to x) are considered to be an "inlier" with respect to w_y. The precise mathematical update can be written as $w_y \leftarrow (w_y \oplus h(x)) \wedge w_y = w_y \wedge \overline{(h(x))}$, where $\oplus$, $\wedge$ and $\overline{(\cdot)}$ are the element-wise vector XOR, AND and NOT respectively. Starting with $w_y = \mathbf{1}_m$, using De Morgan's law, we can condense the FBF learning for a class $l \in [L]$ to

$$w_l = \mathbf{1}_m \bigwedge_{(x,y)\in S:\ y=l} \overline{(h(x))} \tag{2}$$

At the conclusion of the learning, the L per-class FBFs and the lifting matrix M_m^s constitute our proposed FlyHash bloom filter classifier (FBFC). Figure 1a visualizes the process for a toy example. Algorithm 1, equation (2), and the commutative nature of '$\wedge$' highlight couple of interesting aspects of this learning process:

- The learning scheme is *online* where an example can be used in isolation to update the model without any approximation. This is common with ML models trained via some form of stochastic gradient descent, but are not possible with decision tree methods.
- The learning process does not need to see an example (x, y) more than once – once the appropriate FBF w_y has been updated using $h(x)$, any subsequent update with (x, y) is redundant – the bit positions in w_y corresponding to the nonzero bits in $h(x)$ are already zero. This implies that, a single FBFC model can be learned with *a single pass of the training set* S.

Inter-class similarities. Given the per-class FBFs $w_l, l \in [L]$, we can use the cosine similarity $s(l_1, l_2) = \frac{\langle w_{l_1}, w_{l_2} \rangle}{\|w_{l_1}\| \|w_{l_2}\|}$ between the FBF pair (w_{l_1}, w_{l_2}) as an *inter-class similarity* score between classes l_1 and l_2 to quantify the hardness of differentiating these classes, and generate insights into the structure of the classification problem. For example, the per-class encodings $w_l, l \in [L]$ of the class conditional data-distributions can be used to generate a class hierarchy by performing hierarchical clustering of the per-class encodings.

3.1.2 FBFC inference. As discussed previously, the FBF w_l for a particular class $l \in [L]$ are learned in a way that any point x with a label l is treated as an inlier with respect to class l; the example x with label l does not affect the class encodings $w_{l'}, l' \neq l, l' \in [L]$. This implies that any point $x' \in \mathbb{R}^d$ similar to x will have a low novelty score $w_l^\top h(x')$. This motivates our inference rule – for a test point x, we compute the per-class novelty scores $f_l(x) \in [0, 1], l \in [L]$ and the predicted label as:

$$f_l(x) = (1/\rho) w_l^\top h(x), \quad \hat{y} = \arg\min_{l\in[L]} f_l(x) \tag{3}$$

A high $f_l(x)$ indicates that training examples with label l are very different from x. A small value of $f_l(x)$ indicates the existence of at least one training example with label l similar to x. The predicted label for x is simply the class with the smallest $f_l(x)$ (breaking ties randomly). This is visualized in Figure 1b and detailed in the InferFBFC subroutine in Algorithm 1. The per-class novelty $f_l(x), l \in [L]$ can be converted into class probabilities with a soft-max operation.

3.1.3 Robust FBFC against labeling noise with non-binary FBF. The proposed FBFC generates a binary encoding $w_l \in \{0, 1\}^m$ for each class $l \in [L]$ with a single pass of the data by zeroing all bits in w_l corresponding to the nonzero positions in the FlyHash $h(x)$ for every example $(x, y) \in S, y = l$. Given this encoding, the inference for a test point x generates the novelty score for a class l by counting the number of nonzero elements in $h(x)$ that are also nonzero in w_l – higher number of matches imply larger novelty scores and hence less chance of predicting label l for x. As we will see in our empirical evaluations, this process is quite effective even with a single pass. However, a single mislabeled example $(x_{\text{mis}}, y_{\text{mis}})$ can modify the FBF $w_l, y_{\text{mis}} = l$, in a way that all test points x similar to the mislabeled point x_{mis} may get misclassified since they would get a low novelty score with respect to the FBF w_l; we cannot correct this given the single pass nature of the FBFC learning. In the following, we will mathematically motivate this lack of robustness and provide a remedy utilizing a non-binary FBF.

In our binary FBF design, for any test point x and any $l \in [L]$, let $A_x = \{j \in [m] : h(x)_j = 1\}$ be the nonzero coordinates in $h(x)$. Each coordinate of A_x contributes in deciding the value of $f_l(x)$. For any $j \in A_x$, it is possible that a single (possibly mislabeled) training example x' from class l sets the contribution of the j^{th} coordinate to zero in the computation of $f_l(x)$ – it is only required that $h(x')_j = 1$. Even in the absence of labeling noise, the FlyHash $h : \mathbb{R}^d \to \{0, 1\}^m$ is randomized, and there is always a nonzero probability of this event. Also, for any $j, k \in A_x, j \neq k$, if $w_{lj} = w_{lk} = 0$ (the j^{th} and k^{th} element in the FBF w_l for class l), coordinates j and k are indistinguishable in terms of their contribution to $f_l(x)$. To address these limitations, we present a modified FBF design which aims to capture neighborhoods and distribution more effectively, by allowing coordinates of w_l to take value in $[0, 1]$. In this design, for any fixed $c \in (0, 1]$, the j^{th} coordinate w_{lj} of FBF w_l is set as:

$$w_{lj} = (1 - c)^{\left|\{(x,y)\in S:\ y=l \text{ and } (h(x))_j=1\}\right|}, l \in [L], j \in [m], \tag{4}$$

with $c = 1$ corresponding to the original binary FBF. At inference, the label for a test point $x \in \mathbb{R}^d$ is still computed as $\hat{y} = \arg\min_{l\in[L]} w_l^\top h(x)$. We term this form of the Fly Bloom Filter as FBF* and the corresponding classifier as FBFC*. While the entries w_{lj} in this non-binary FBF w_l can take values in $[0, 1]$, it does not completely lose the simplicity and interpretability of the binary FBF since the definition in equation (4) implies that the entries w_{lj} are either 1 or go to zero exponentially fast. The value of c is a *decay rate* controlling the rate at which entries w_{lj} in the FBF w_l go to zero. FBFC* training is detailed in Algorithm 2, with differences from Algorithm 1 highlighted in Maroon. Inference with FBFC* is exactly the same as with FBFC (Algorithm 1, InferFBFC).

To compute $w_l, l \in L$, the TrainNBFBFC subroutine in Algorithm 2 initializes the per-class count vectors $z_l \in \mathbb{R}^m$ to $\mathbf{0}_m$, the vector of all zeros (line 3). Every example $(x, y) \in S$ is processed sequentially (line 4) by first generating the FlyHash $h(x)$ for x (line 5) and the updating the count vector z_y corresponding to the class y by incrementing the counts of the indices $j \in [m]$ corresponding to the nonzero entries in $h(x)$ – this is equivalent to adding the sparse binary FlyHash $h(x)$ to the count vector z_y (line 6). After all the training examples are processed, the per-class final non-binary FBF $w_l, l \in [L]$ is generated by raising $(1 - c)$ to the power of z_l *elementwise* (line 8). Given the per-class FBFs $\{w_l \in [0, 1]^m, l \in [L]\}$, the inference procedure is exactly the same as Algorithm 1. The exponential decay in equation (4) allows $w_{lj}, l \in [L], j \in [m]$ to be determined by a local neighborhood of size dependent on c.

Algorithm 2: Non-binary FBFC training with training set $S \subset \mathbb{R}^d \times [L]$, lifted dimensionality $m \in \mathbb{N}$, NNZ for each row in the lifting matrix $s \ll d$, NNZ in the FlyHash $\rho \ll m$, decay rate $c \in (0, 1]$ and test point $x \in \mathbb{R}^d$.

TrainNBFBFC: $(S, m, \rho, s, c) \rightarrow (M^s_m, \{w_l, l \in [L]\})$
  Sample $M^s_m \in \{0,1\}^{m \times d}$ with s NNZ/row
  Initialize $z_1, \ldots, z_L \leftarrow \mathbf{0}_m \in \mathbb{R}^m$ // all z_l initialized to zero
  for $(x, y) \in S$ **do**
    $h(x) \leftarrow \Gamma_\rho(M^s_m x)$ // Γ_ρ top-ρ WTA
    $z_y \leftarrow z_y + h(x)$ // Sparse op $\because |h(x)|_0 = \rho \ll m$
  end
  $w_l \leftarrow (1-c)^{\odot z_l}, l \in [L]$ // elementwise exponentiation
  return $(M^s_m, \{w_l, l \in [L]\})$
end

3.2 Theoretical analysis

In this section we present theoretical analysis of FBFC, focusing on (i) the computaional complexities of the FBFC/FBFC* learning and inference mechanics presented in §3.1, and (ii) the learning theoretic properties of the proposed FBFC. All proofs are deferred to Supplement S1 and S2.

3.2.1 Computational Complexities. We provide the computational complexities of all the algorithms presented in §3.1 in terms of the runtime and memory requirement. We present the computational complexities for a specific hyper-parameter configuration of FBFC– (i) the lifted FlyHash dimensionality m, (ii) the number of nonzeros s in each row of M^s_m, (iii) the number of nonzeros ρ in a FlyHash after the winner-take-all operation, and (iv) (*only for* FBFC*) the decay rate $c \in (0, 1]$. We begin by presenting results for the FBFC training and inference in Algorithm 1.

Claim 1 (FBFC training). *Given a training set $S \subset \mathbb{R}^d \times [L]$ with n examples, the single pass Train*FBFC *subroutine in Algorithm 1 with the lifted* FlyHash *dimensionality m, number of nonzeros s in each row of the lifting matrix $M^s_m \in \{0,1\}^{m \times d}$, and number of nonzeros ρ in* FlyHash *$h(x)$ for any $x \in \mathbb{R}^d$, takes time $O(nm \cdot \max\{s, \log \rho\})$ and has a memory overhead of $O(m \cdot \max\{s, L\})$.*

Claim 2 (FBFC inference). *Given a trained* FBFC*, the inference for any test point $x \in \mathbb{R}^d$ with the Infer*FBFC *subroutine in Algorithm 1 takes time $O\left(m \cdot \max\left\{s, \log \rho, (\rho L/m)\right\}\right)$ with a memory overhead of $O(\max\{m, L\})$.*

Remark 1. *For any test point $x \in \mathbb{R}^d$ and corresponding* FlyHash *$h(x)$, with a large number of labels (large L), if the $\arg\min_{l \in [L]} w_l^\top h(x)$ can be solved via fast MIPS (maximum inner product search) algorithm in time $f(L)$ sublinear in L, then the overall inference time for x would be given by $O\left(m \cdot \max\left\{s, \log \rho, (\rho f(L)/m)\right\}\right)$ which would be sublinear in L.*

The computational complexities of FBFC* training are as follows:

Claim 3 (FBFC* training). *Given a training set $S \subset \mathbb{R}^d \times [L]$ with n examples, the single pass Train*NBFBFC *subroutine in Algorithm 2 with the lifted* FlyHash *dimensionality m, number of nonzeros s in each row of the lifting matrix $M^s_m \in \{0,1\}^{m \times d}$, number of nonzeros ρ in* FlyHash *$h(x)$ for any $x \in \mathbb{R}^d$, and decay rate $c \in (0, 1]$, takes time $O\left(nm \cdot \max\left\{s, \log \rho\right\}\right)$ and has a memory overhead of $O(m \cdot \max\{s, L\})$.*

Comparing this result to the computational complexities of binary FBFC (Claim 1), we see that the computational complexities are of the same order across the board. Since the inference with FBFC* is exactly the same as that of vanilla FBFC, the computational complexities of the inference is given by Claim 2.

3.2.2 Learning theoretic properties. Since FBFC is a completely novel classifier, we now establish theoretical guarantees of FBFC's predictive performance by relating it to a known classifier with well-studied theoretical guarantees. The FlyHash has been shown to be locality sensitive, and the Fly bloom filter (FBF) w creates an encoding of the data distribution (or in our case, the encoding w_l for the distribution of a class l). The novelty score $w^\top h(x)$ of any (test) point $x \in \mathbb{R}^d$ corresponds to how "far" the point is from the distribution encoded by w. In our FBFC, using the minimum novelty score $\arg\min_{l \in [L]} w_l^\top h(x)$ to label x is equivalent to labeling x with the class whose distribution/encoding is "closest" to x. This motivates us to study *how the* FBFC *is related to the well-studied nearest-neighbor classifier*. Specifically, we identify precise conditions under which FBFC agrees with the nearest-neighbor classifier 1NNC. The general setup, notations and proof sketches are described in Supplement S2.

We begin by analyzing the binary classification performance of FBFC trained on a training set $S = \{(x_i, y_i)\}_{i=1}^{n_0+n_1} \subset \mathcal{X} \times \{0, 1\}$, where $S = S^1 \cup S^0$, S^0 is a subset of S having label 0, and S^1 is a subset of S having label 1, satisfying $|S^0| = n_0$, $|S^1| = n_1$ and $n = \max\{n_0, n_1\}$. For appropriate choice of m, let $w_0, w_1 \in \{0,1\}^m$ be the FBFs constructed using S^0 and S_1 respectively.

Connection to 1NNC. For any test point x, without loss of generality, assume that its nearest neighbor from S has class label 1. Then 1NNC will predict x's class label to be 1. Therefore, if we are able to show that $\mathbb{E}_M(w_1^\top h(x)) < \mathbb{E}_M(w_0^\top h(x))$ then FBFC will predict, in expectation, x's label to be 1. While estimating expected novelty score is difficult, an upper and lower bound of class specific novelty scores can easily be estimated in terms of $\tau_x(f)$ – the top f-fractile value of the distribution $\theta^\top x$, where θ represents uniform sampling of the rows of M^s_m (see Lemma S1). This immediately provides us a sufficient condition for FBFC to agree with 1NNC on any test point x in expectation – the upper bound of $\mathbb{E}_M(w_1^\top h(x))$ should be strictly smaller than lower bound of $\mathbb{E}_M(w_0^\top h(x))$ (a high probability statement then follows using standard concentration bounds). Under mild structural and/or distributional assumptions on $\mathcal{X}$, we can readily establish a the following result. The assumptions mentioned above give rise to two special cases which are discussed in Supplement S2.2.

Theorem 4. *Fix any $\delta \in (0, 1)$, $s \ll d$, and $\rho \ll m$. Given a training set S as described above and a test example $x \in \mathcal{X}$, let x_{NN} be its closest point from S measured using ℓ_p metric for an appropriate choice of p. If (i) $\rho = \Omega(\log(1/\delta))$, (ii) $\|x - x_{NN}\|_p = O(1/s)$, and (iii) $m = \Omega(n\rho)$, then under mild conditions, with probability at least $1 - \delta$ (over the random choice of lifting matrix M), prediction of* FBFC *on x agrees with the prediction of* 1NNC *on x.*

Proof (sketch). If the structure of $\mathcal{X}$ allows us to choose a threshold τ_x that is identical for any $x \in \mathcal{X}$, resulting in a closed form solution for the quantity $q(x, x')$ (defined in Supplement S2.1) for any $x, x' \in \mathcal{X}$, or the distributional assumption on $\mathcal{X}$ sets the quantity $\mathbb{E}_x q(x, x')$

to be identical for all $x' \in S$, then all the three conditions mentioned in theorem are satisfied. This, in conjunction with Lemma S1, yields the desired result in expectation under mild conditions. The high probability result then follows from standard concentration bounds.

Remark 2. *We established conditions under which predictions of* FBFC *agrees with that of* 1NNC *with high probability.* 1NNC *is a non-parametric classification method with strong theoretical guarantee – as* $|S| = n \to \infty$*, the* 1NNC *almost surely approaches the error rate which is at most twice the Bayes optimal error. Therefore, by establishing the connection between* FBFC *and* 1NNC*,* FBFC *has the same statistical guarantee under the conditions of Theorem 4.*

Multi-class classification. The above results can be extended to multi-class classification problem involving $L > 2$ classes in a straight forward manner (see Supplement S2.4).

Note that the FBF guarantees for novelty detection are limited to two special cases: (i) examples with binary feature vectors containing fixed number of ones, and (ii) examples sampled from a permutation invariant distribution [4]. We extend this analysis with these two cases to provide guarantees for FBFC in multi-class classification (see Supplement S2), which is a distinct learning problem from novelty detection.

4 EMPIRICAL EVALUATIONS

In this section, we evaluate the empirical performance of FBFC. First, we demonstrate the dependence of FBFC on its hyper-parameters. Then, we compare FBFC to other classifiers that can be trained in a single pass on various data sets. Finally, we present some problem insights generated by a trained FBFC. The details on the implementation and compute resources are in Supplement S3.

4.1 Data sets and baselines

For the empirical evaluation of FBFC and FBFC*, we consider two groups of data sets:

- We consider **synthetic data** of varying sizes and properties in §4.3. These synthetic data are designed to favor local classifiers such as the kNNC– each class conditional data distribution consists of multiple separated modes, with enough inter-class separation [12]. We consider these synthetic data to see if our proposed FBFC is able to capture multiple separated local class neighborhoods in a single per-class FBF encoding while providing enough separation between FBFs of different classes.
- We consider **71 binary & multi-class classification data sets from OpenML** [33] in §4.4 to evaluate the performance of FBFC (and variants) on real data sets. We deliberately choose a large set of data sets, containing many where kNNC/1NNC have strong performance, and many where they are not as strong relative to other standard ML classifiers.

We compare our proposed FBFC and FBFC* to various baselines relative to kNNC. We consider a variety of baselines, including single-pass ones similar to FBFC. Further details on the baselines and their hyper-parameters are in Supplement S3:

- **kNNC:** This is the primary baseline. We tune over the neighborhood size $k \in [1, 64]$. We also specifically consider 1NNC ($k = 1$).
- **CC1:** Classification based on a single prototype per class – the geometric class center, computed with a single training set pass.
- **CC:** This generalizes CC1 with multiple prototypes per class – a test point is classified using its closest prototype. Per-class prototypes are obtained by K-means clustering, tuning over $K \in [1, 64]$. This is *not single pass.*
- **SBFC:** To ablate the effect of the high level of sparsity in FlyHash, we utilize SimHash [3] based LSBF for each class to get the SimHash Bloom Filter classifier (SBFC). We use this to demonstrate the utility of the high sparsity in FlyHash; SimHash is binary like FlyHash but not inherently sparse. We tune over the SimHash projected dimension m, considering $m < d$ (traditional regime where SimHash is usually employed) and $m > d$ (as in FlyHash). For the same m, SimHash is more costly ($O(md)$ per point) than FlyHash ($O(ms + m \log \rho)$), involving a dense matrix-vector product instead of a sparse matrix-vector one.
- **LR.** Logistic regression trained for a single epoch with an online algorithm, tuned over 960 hyper-parameter settings per data set.
- **MLPC.** Multi-layer perceptron trained for one epoch with Adam [19], tuned over 288 hyper-parameter settings per data set.
- **FBFC/FBFC*.** For a data set with d dimensions, we tune across 60 FBFC/FBFC* hyper-parameter settings in the following ranges: $m \in [2d, 2048d]$, $s \in (0.0, 0.5d]$, $\rho \in [8, 256]$, and $c \in [0.2, 1)$.

To normalize performance across data sets, we compute the relative performance of all methods on each data set as $(1 - A_M/A_{k\text{NNC}})$ where $A_{k\text{NNC}}$ is the best 10-fold cross-validated classification accuracy – accuracy aggregated over 10 train-test splits of any given data set – achieved by kNNC across all hyper-parameters and A_M is the best 10-fold cross-validated accuracy obtained by method M. This means that kNNC has a relative performance of 0. We perform this "normalization" to be able to compare the aggregate performance of different classifiers across multiple data sets (which themselves probably have different ranges for best achievable accuracies).

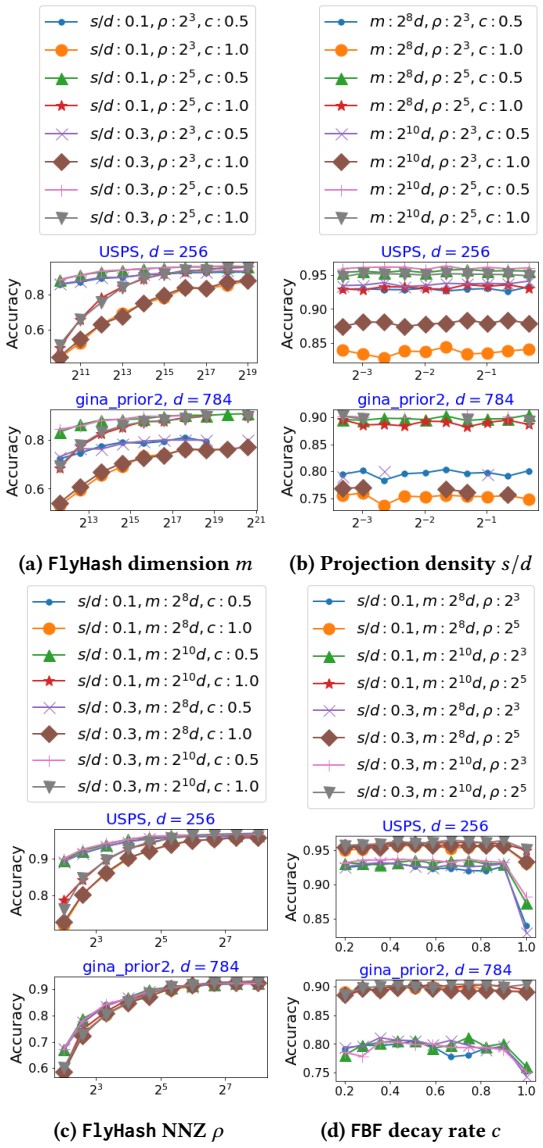
4.2 Dependence on hyper-parameters

We study the effect of the different FBFC hyper-parameters: (i) the FlyHash dimension m, (ii) the NNZ per-row $s \ll d$ in M^s_m, (iii) the NNZ ρ in the FlyHash, and (iv) the FBF decay rate c. We consider 2 OpenML data sets (see Table S1 in Supplement S3 for data details). For every hyper-parameter setting, we compute the 10-fold cross-validated accuracy. We vary each hyper-parameter while fixing the others. The results for each of the hyper-parameters and data sets are presented in Figure 2.

The results indicate that increasing m usually improves performance up to a point. FBFC performance is mostly not affected by s, allowing us to use small values for s ($\sim O(10)$). Increase in ρ improves performance up to point. The performance is not affected much by the value of the decay rate when $c < 1$, but there is a significant drop in performance as we move from $c < 1$ (non-binary FBFC*) to $c = 1$ (binary FBFC), indicating the advantage of our novel non-binary FBF; this behavior is pretty consistent and obvious across all data sets.

4.3 Synthetic Data

We first consider synthetic data designed for strong kNNC performance. We generate data for 5 classes with 3 clusters per class and points in the same cluster belong to the same class implying that a neighborhood based classifier will perform well. However, the

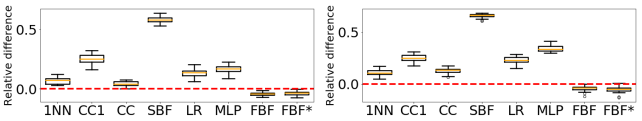


(a) FlyHash dimension m (b) Projection density s/d

(c) FlyHash NNZ ρ (d) FBF decay rate c

Figure 2: Effect of the different FBFC/FBFC* hyper-parameters on performance for 2 data sets – the horizontal axes correspond to the hyper-parameter being varied while fixing the remaining hyper-parameters. The vertical axes correspond to the 10-fold cross-validated accuracy for the given configuration (*higher is better*). Note the log scale on the horizontal axes. For the hyper-parameter c, $c = 1$ corresponds to the binary FBFC. *Please view in color.*

classes are not necessarily linearly separable. We select such a set to demonstrate that the proposed FBFC is able to encode multiple separate neighborhoods of a class within a single FBF while providing enough separation between the per-class FBFs for high predictive performance. We begin with binary data of the form considered in our theoretical results (Supplement S2.2.1) – points $x \in \{0,1\}^d$ with $|x| = b < d$. We then consider data in general $\mathbb{R}^d$. For each



(a) Synthetic $\{0,1\}^{100}$, $b = 20$ (b) Synthetic $\mathbb{R}^{100}$

Figure 3: Performance of FBFC/FBFC* and baselines relative to the kNNC performance on *synthetic* binary (Figure 3a) and real (Figure 3b) data. The 10-fold cross-validated accuracy is considered for each of the data sets. The box-plots correspond to performance relative to kNNC (*lower is better*) aggregated over 30 synthetic data sets (see text for details). The red dashed line denotes kNNC performance.

Table 2: FBFC vs FBFC* accuracy on synthetic data with label noise aggregated over 10 train/test splits. Note that the noise is only added to the train set in every split.

Label noise level	1.0%	5.0%	10%	25%
FBFC accuracy (%)	56.6±0.8	54.5±0.9	52.0±1.3	44.0±1.0
FBFC* accuracy (%)	60.1±0.7	58.9±0.9	57.1±1.2	50.3±1.2
Rel. improvement (%)	6.1±0.7	8.1±0.9	9.8±1.2	14.2±1.8

setting, we randomly generate 30 data sets with 1000 points each. The performance of all baselines, aggregated across all 30 sets, is presented in Figures 3a and 3b for $d = 100, b = 20$.

The results indicate that FBFC and FBFC* are able to match kNNC performance significantly better than all other single pass baselines. The binary FBFC matches the performance of FBFC*. As expected, CC performs significantly better than the other baselines on account of being able to properly compress multi-modal classes, albeit requiring multiple passes. CC1 performs significantly worse than CC since one cluster is not able to appropriately compress multi-modal classes while maintaining the separation between the classes. LR and MLPC perform similarly to CC1. The proposed FBFC and FBFC* significantly outperform SBFC, highlighting the need for sparse high dimensional hashes to summarize multi-modal neighborhoods while avoiding overlap between per-class FBFs. Moreover, note that in the absence of any labeling noise (which we have complete control over given we are generating these synthetic sets), FBFC and FBFC* perform very similarly as discussed in §3.1.3.

Effect of labeling noise. We consider an additional experiment with synthetic 5-class classification problem in $\mathbb{R}^d$ where we add increasing levels of noise to the labels. We compare FBFC with FBFC* for a single hyper-parameter configuration where the only difference between the two is that FBFC* uses $c = 0.9$. FBFC implicitly uses $c = 1$. The results presented in Table 2 indicate that as the label noise level goes up from 1% to 25%, the performance of both schemes drop as expected. However, the relative improvement of FBFC* over FBFC increases from around 6% to 14%, indicating that FBFC* is more robust to labeling noise in the training set.

Table 3: Fraction of data sets where FBFC* outperforms baselines with median difference in relative performance between FBFC* and baselines across data sets in brackets. The underlined methods are *not single-pass*. ▲ denotes FBFC* improvement against baseline; ▼ denotes FBFC* decline; • denotes we can reject the null hypothesis (FBFC* and baseline have similar performance) of the paired t-test at significance level 0.01; ○ denotes we cannot reject the null hypothesis.

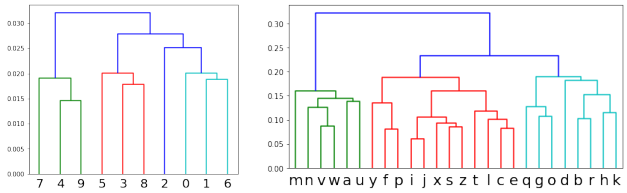
Method	All (71 sets)	Group A (37/71)	Group B (34/71)
kNNC	0.51 (▲0.05%) ○	0.24 (▼1.08%) ○	0.79 (▲3.98%) •
1NNC	0.62 (▲2.21%) •	0.38 (▼0.24%) ○	0.88 (▲12.1%) •
CC1	0.87 (▲7.64%) •	0.95 (▲11.8%) •	0.79 (▲4.96%) •
CC	0.38 (▼0.48%) ○	0.38 (▼0.31%) ○	0.38 (▼0.50%) ○
SBFC	0.99 (▲24.9%) •	0.97 (▲24.7%) •	1.00 (▲25.2%) •
LR	0.58 (▲1.34%) ○	0.78 (▲3.39%) •	0.35 (▼0.68%) ○
MLPC	0.73 (▲4.36%) •	0.81 (▲6.57%) •	0.65 (▲3.80%) •
FBFC	0.82 (▲5.87%) •	0.68 (▲0.73%) •	0.97 (▲9.13%) •

4.4 OpenML Data

We consider 71 classification (binary and multi-class) data sets from OpenML with d numerical columns and n samples; $d \in [10, 1024]$, $n \leq 50000$. Unlike the synthetic sets, these data sets do not guarantee strong kNNC performance. Hence we separate these data sets into 2 groups: (i) **Group A**: 37 data sets where the nonparametric kNNC performs the best among the baselines, (ii) **Group B**: the remaining 34 sets where parametric LR performs the best among the baselines. We study the performance of the proposed schemes in both these groups of data sets. The results are summarized in Table 3, where the relative performances of all baselines and FBFC are compared to FBFC*, aggregated over all the data sets, with paired t-tests for the null hypothesis that FBFC* and the method under consideration have similar performance at a significance level of 0.01.

Overall, FBFC* consistently outperforms FBFC across both groups on 58/71 or 82% of the data sets, with a median improvement of 5.87%, highlighting the value of the non-binary FBF*. As seen in Table 3, both FBFC and FBFC* significantly outperform SBFC, similar to the synthetic data sets, highlighting the need for sparsity with bloom filter based classifiers on real world data sets. In fact, SBFC performs the worst among all baselines across almost all data sets, indicating that vanilla locality sensitive bloom filters are not useful for multi-dimensional classification; both the high dimensionality and high sparsity are also necessary for real-world data.

Compared to the nearest-neighbor classifiers, kNNC and 1NNC, FBFC* performs comparably to the kNNC (null hypothesis cannot be rejected), while improving upon 1NNC by a median of around 2% across all 71 sets. Looking at the results on **Group A** (where kNNC performs best among baselines), FBFC* shows a median drop of around 1% over kNNC but the null hypothesis still cannot be rejected at a significance level of 0.01; FBFC* performs comparably to 1NNC. FBFC* is able to still outperform kNNC and 1NNC on 24% and 38% of the sets respectively in **Group A**. The results are more favorable with **Group B**, where kNNC is outperformed by LR. FBFC* significantly improves upon kNNC and 1NNC with a median difference of around 4% and 12% respectively while outperforming them on 79% and 88% of the sets respectively in **Group B**.



(a) MNIST – 10 classes (b) Letters – 26 classes

Figure 4: Class hierarchy as a dendogram generated with the per-class FBF based inter-class similarities for data sets with interpretable labels, highlighting that FBFs can encode classes in a way that the resulting inter-class similarities are semantically meaningful. See text for further discussion.

Comparing to the prototype-based classifiers, CC1 and CC, FBFC* outperforms the single-pass CC1 consistently and significantly across all groups of data sets, while slightly underperforming the multi-pass CC. Across all groups, FBFC* shows a median drop of at most 0.5% compared to CC, but the null hypothesis cannot be rejected at a significance level of 0.01, implying that the single-pass FBFC* performs comparably to the multi-pass CC. FBFC* is able to outperform CC on 38% of the data sets across all groups.

Among models, LR and MLPC, trained with (stochastic) gradient descent, Table 3 indicate that LR generally outperforms MLPC across all groups since a single pass is usually not sufficient to train MLPC to high accuracy without any pre-training or meta-learning. Over all data sets, FBFC* performs comparably to LR, showing an median improvement of around 1.3%, while outperforming LR on 58% of the data sets. However, we are unable to reject the null hypothesis here. With **Group A** data sets, FBFC* significantly outperforms LR on 78% of the data sets, showing a median improvement of around 3.4%. In **Group B**, where LR is the strongest baseline, FBFC* underperforms LR, showing a median drop of around 0.68%, but the difference is not significant as we are unable to reject the null hypothesis.

These results demonstrate that the proposed FBFC/FBFC* allow us to enjoy strong performance on (i) data sets where nonparametric kNNC perform well, as well as on (ii) data sets where parametric models are more favorable, enjoying best of both worlds with a single pass training. This behaviour is verified with a large number of data sets, implying the wide practical utility of FBFC/FBFC*.

4.5 Problem insights through class similarities

We wish to highlight the ability to generate problem insights in terms of class hierarchies utilizing the class encodings $w_l, l \in [L]$ generated during FBFC training. We consider FBFC generated on two data sets with already interpretable labels: MNIST data set for digits classification, and Letters data set for English letter classification. We wish to explore whether the FBFC is able to obtain a semantically meaningful hierarchy without any additional supervision. For the best FBFC* model, we generate dendograms for the classes using $s(l_1, l_2) = \frac{\langle w_{l_1}, w_{l_2} \rangle}{\|w_{l_1}\| \|w_{l_2}\|}$ as the similarity in Figure 4.

With MNIST (Figure 4a), we see three main clusters of digits (7,4,9), (5,3,8) and (0,1,6). The first two clusters are semantically

meaningful because of the structure of the digits. (0,6) in the (0,1,6) also have similar structure. With Letters (Figure 4b), of the 26 classes, (m,n,v,w), (a,u), (y,f,p), (i,j), (x,s,z), (t,l), (c,e), (q,g,o), (h,k) are some of the semantically meaningful groups discovered by FBFC without any extra supervision beyond just the training set. These results indicate that, for classification problems where we do not possess any semantically meaningful classes, we can utilize the class encodings generated by FBFC to find any underlying hierarchy if available.

5 CONCLUSIONS AND FUTURE WORK

In this paper we proposed a novel neuroscience inspired Fly Bloom Filter based classifier (FBFC) that can be trained in a single pass of the training set – a point never needs to be revisited, and the whole training data does not need to be in memory. The inference requires an efficient FlyHash followed by a very sparse dot product. On the theoretical side, we established conditions under which FBFC agrees with the 1NNC. We empirically validated our proposed scheme with 71 data sets of varied data dimensionality and demonstrated that the predictive performance of our proposed classifier is competitive with the kNNC and other single-pass classifiers.

We plan to pursue theoretical guarantees for FBFC and FBFC* in general $\mathbb{R}^d$ by exploring data dependent assumptions such as doubling measure. While our theoretical results connects FBFC to 1NNC, thereby inheriting its generalization guarantees, in our empirical evaluations, we also compared our proposed schemes to the more general kNNC (which has better generalization guarantees). Our empirical evaluations indicate that FBFC* significantly outperforms 1NNC, while matching kNNC in most cases and at times outperforming it. This motivates us to study the conditions under which FBFC/FBFC* matches kNNC in future work. Additionally, utilizing the sparse and randomized nature of FBFC, we will investigate differential privacy preserving properties of FBFC as well as robustness of FBFC to benign and adversarial perturbations. Finally, we believe that FBFC can be adapted to handle concept drifts and distribution shifts when learning with data streams (online learning) by being able to forget past examples.

Acknowledgement. KS gratefully acknowledges funding from NSF award FAIN 2019844.

REFERENCES

[1] Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 52(1) (2008), 117–122.

[2] A. Beygelzimer, S. Kakade, and J. Langford. 2006. Cover tree for nearest neighbor. In *International Conference on Machine Learning*. 97–104.

[3] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 380–388.

[4] Sanjoy Dasgupta, Timothy C Sheehan, Charles F Stevens, and Saket Navlakha. 2018. A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences* 115, 51 (2018), 13093–13098.

[5] Sanjoy Dasgupta and Kaushik Sinha. 2015. Randomized partition trees for nearest neighor search. *Algorithmica* 72(1) (2015), 237–267.

[6] Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. 2017. A neural algorithm for a fundamental computing problem. *Science* 358, 6364 (2017), 793–796.

[7] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*. 1646–1654.

[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research* 9, Aug (2008), 1871–1874.

[9] A-J. Gallego, J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan. 2018. Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation. *Pattern Recognition* 74 (2018), 531–543.

[10] A. Gionis, P. Indyk, and R. Motwani. 1999. Similarity search in high dimensions via hashing. In *International Conference on Very large Data Bases*. 518–529.

[11] J. Gou, W. Qiu, Z. Yi, Y. Xu, Q. Mao, and Y. Zhan. 2019. A Local Mean Representation-based K -Nearest Neighbor Classifier. *ACM Transactions on Intelligent Systems and Technology* 10(3) (2019), 1–25.

[12] Isabelle Guyon. 2003. Design of experiments of the NIPS 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction and feature selection*.

[13] Raffay Hamid, Ying Xiao, Alex Gittens, and Dennis DeCoste. 2014. Compact random feature maps. In *International Conference on Machine Learning*. 19–27.

[14] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. 2017. Neuroscience-inspired artificial intelligence. *Neuron* 95, 2 (2017), 245–258.

[15] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[16] Y. Hua, B. Veeravalli, and D. Feng. 2012. Locality-sensitive bloom filter for approximate membership query. *IEEE Trans. Comput.* 61, 6 (2012), 817–830.

[17] Purushottam Kar and Harish Karnick. 2012. Random feature maps for dot product kernels. In *Artificial Intelligence and Statistics*. 583–591.

[18] Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann L Cun. 2010. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*. 1090–1098.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] A. Kirsch and M. Mitzenmacher. 2006. Distance sensitive bloom filters. In *Meeting in Algorithm Engineering & Experiments*. 41–50.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[22] Hugo Larochelle and Geoffrey E Hinton. 2010. Learning to combine foveal glimpses with a third-order Boltzmann machine. In *Advances in neural information processing systems*. 1243–1251.

[23] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*. 2204–2212.

[24] S. Oigiaroglou and G. Evangelidis. 2013. Efficient k-NN classification based on homogeneous clusters. *Artificial Intelligence Review* 42 (2013), 491–513.

[25] S. Oigiaroglou and G. Evangelidis. 2016. RHC: A non-parametric cluster-based data reduction for efficient k-NN classification. *Pattern Analysis and Applications* 19 (2016), 93–109.

[26] S. M. Omohundro. 1989. *Five Balltree Construction Algorithms*. Technical Report. International Computer Science Institute, Berkeley, CA.

[27] Hamid Parvin, Moslem Mohamadi, Sajad Parvin, Zahra Rezaei, and Behrouz Minaei. 2012. Nearest Cluster Classifier. In *Hybrid Artificial Intelligent Systems*, Emilio Corchado, Václav Snášel, Ajith Abraham, Michał Woźniak, Manuel Graña, and Sung-Bae Cho (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 267–275.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[29] Ninh Pham and Rasmus Pagh. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 239–247.

[30] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems* 20 (2007), 1177–1184.

[31] Parikshit Ram and Kaushik Sinha. 2019. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1378–1388.

[32] Mark Schmidt, Nicolas Le Roux, and Francis Bach. 2017. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming* 162, 1-2 (2017), 83–112.

[33] Jan N Van Rijn, Bernd Bischl, Luis Torgo, Bo Gao, Venkatesh Umaashankar, Simon Fischer, Patrick Winter, Bernd Wiswedel, Michael R Berthold, and Joaquin Vanschoren. 2013. OpenML: A collaborative science platform. In *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 645–649.

[34] Santosh Vempala. 2004. *The Random Projection Method*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 65. DIMACS/AMS.

[35] L. Zhou, L. Wang, X. Ge, and Q. Shi. 2010. A clustering-based KNN improved algorithm CLKNN for text classification. In *International Asia Conference on Informatics in Control, Automation and Robotics*. 212–215.

S1 SUPPLEMENTARY MATERIAL FROM §3.2.1

*Proof for Claim 1 [*FBFC *training].* In TrainFBFC (Algorithm 1), line 2 takes $O(ms)$ time and memory, line 3 takes $O(mL)$ time and memory. Each FlyHash in line 5 takes $O(ms)$ time for computing $M_m^s x$, $O(m \log \rho)$ time for performing the WTA operation $\Gamma_\rho(\cdot)$, and $O(m)$ memory. FBF w_y update in line 6 takes $O(\rho)$ time since it is only updating ρ entries in w_y. Hence the loop 4-7 takes time $O(n(ms + m \log \rho + \rho))$ and maximum $O(m)$ memory. Given $\rho \ll m$ and $L \ll n$, the total runtime is given by $O(nm \cdot \max\{s, \log \rho\})$ time and $O(m \cdot \max\{s, L\})$ memory, proofing the claim of the statement.

*Proof of Claim 2 [*FBFC *inference].* In InferFBFC (Algorithm 1), the FlyHash operation in line 11 takes time $O(m(s + \log \rho))$ and $O(m)$ memory, while the operation in line 12 takes time $O(L\rho + L)$ since each $w_l^\top h(x)$ takes time $O(\rho)$ since $h(x)$ only have ρ nonzero entries. This leads to an overall runtime of $O(m \cdot \max\{s, \log \rho, (\rho L/m)\})$ and memory overhead of $O(\max\{m, L\})$, as per the claim.

*Proof for Claim 3 [*FBFC* *training].* In TrainNBFBFC (Algorithm 2), line 2 takes $O(ms)$ time and memory, line 3 takes $O(mL)$ time and memory. Each FlyHash in line 5 takes $O(m(s + \log \rho))$ time and $O(m)$ memory. FBF w_y update in line 6 takes $O(\rho)$ time since it is only adding 1 to ρ entries in w_y. Hence the loop 4-7 takes time $O(n(ms + m \log \rho + \rho))$ and maximum $O(m)$ memory. The elementwise exponentiation in line 8 takes atmost $O(mL)$. Given $\rho \ll m$ and $L \ll n$, the total runtime is given by $O(nm \cdot \max\{s, \log \rho\})$ time and $O(m \cdot \max\{s, L\})$ memory, proofing the claim of the statement.

S2 SUPPLEMENTARY MATERIAL FROM §3.2.2

S2.1 Preliminaries & notations

We denote a single row of a lifting matrix M_m^s by $\theta \in \{0,1\}^d$ drawn i.i.d. from Q, the uniform distribution over all vectors in $\{0,1\}^d$ with exactly s ones, satisfying $s \ll d$. For ease of notation, we use M instead of M_m^s and we use an alternate formulation of the winner-take-all strategy as suggested in Dasgupta et al. [4], where for any $x \in \mathbb{R}^d$, τ_x is a threshold that sets largest ρ entries of Mx to one (and the rest to zero) in expectation. Specifically, for a given $x \in \mathbb{R}^d$ and for any fraction $0 < f < 1$, we define $\tau_x(f)$ to be the top f-fractile value of the distribution $\theta^\top x$, where $\theta \sim Q$:

$$\tau_x(f) = \sup\{v : \Pr_{\theta \sim Q}(\theta^\top x \geq v) \geq f\} \tag{S1}$$

We note that for any $0 < f < 1$, $\Pr_{\theta \sim Q}(\theta^\top x \geq \tau_x(f)) \approx f$, where the approximation arises from possible discretization issues. For convenience, henceforth we will assume that this is an equality:

$$\Pr_{\theta \sim Q}(\theta^\top x \geq \tau_x(f)) = f \tag{S2}$$

For any two $x, x' \in \mathbb{R}^d$, we define

$$q(x, x') = \Pr_{\theta \sim Q}\left(\theta^\top x' \geq \tau_{x'}(\rho/m) \mid \theta^\top x \geq \tau_x(\rho/m)\right).$$

This can be interpreted as follows – with $h(x), h(x')$ as the FlyHash of x and x', respectively, $q(x, x')$ is the probability that $h(x')_j = 1$ given that $h(x)_j = 1$, for any specific $j \in [m]$ ($h(x)_j$ is the j^{th} entry of the FlyHash $h(x)$). Next, the following lemma establishes upper and lower bounds of class specific novelty scores.

Lemma S1. *Fix any $x \in \mathbb{R}^d$ and let $h(x) \in \{0,1\}^m$ be its* FlyHash *using equation 1. For class $i \in \{0,1\}$, let $x_{NN}^i = \text{argmin}_{(x',y') \in S^i} \|x - x'\|$ with any distance metric $\|\cdot\|$, and $A_{S^i} = \cap_{(x',y') \in S^i}\{\theta : \theta^\top x' < \tau_{x'}(\rho/m)\}$. Then the following hold for each class $i \in \{0,1\}$ separately, where the expectation is taken over the random choice of projection matrix M:*
(i) $\mathbb{E}_M\left(w_i^\top h(x)/\rho\right) = \Pr_{\theta \sim Q}\left(A_{S^i} \mid \theta^\top x \geq \tau_x(\rho/m)\right)$,
(ii) $\mathbb{E}_M\left(w_i^\top h(x)/\rho\right) \geq 1 - \sum_{x' \in S^i} q(x, x')$,
(iii) $\mathbb{E}_M\left(w_i^\top h(x)/\rho\right) \leq 1 - q(x, x_{NN}^i)$.

Proof. Part (i) follows from simple application of Lemma 2 of [4] to class specific FBFs $w_i, i \in \{0,1\}$. Part (ii) follows from simple application of Lemma 3 of [4] to class specific FBFs. For part (iii), simple application of Lemma 3 of [4] to FBF w_i ensures that for any $x' \in S^i, \mathbb{E}_M\left(w_i^\top h(x)/\rho\right) \leq 1 - q(x, x')$. Clearly, $\mathbb{E}_M\left(w_i^\top h(x)/\rho\right) \leq 1 - q(x, x_{NN}^i)$. □

S2.2 Two special cases

S2.2.1 Special case I: Binary data. In this section we consider a special case where examples from each class have binary feature vectors with fixed number of ones. In particular, let $\mathcal{X} = \mathcal{X}_b = \{x \in \{0,1\}^d : |x|_1 = b < d\}$.

Theorem S2. *Let S be a training set as given above. Fix any $\delta \in (0,1)$, and set $\rho \geq \frac{12}{\mu}\ln(4/\delta)$, $m \geq (d/b)n\rho$, and $s = \log_{d/b}(m/\rho)$, where $\mu = \min\left\{\mathbb{E}_M\left((w_0^\top h(x))/\rho\right), \mathbb{E}_M\left((w_1^\top h(x))/\rho\right)\right\}$ and $h(x)$ is the* FlyHash *(eq. (1)). For a test point $x \in \mathcal{X}$, let its closest point from S measured using ℓ_1 metric be x_{NN}, having label $y_{NN} \in \{0,1\}$, satisfies, (i) $\|x - x_{NN}\|_1 \leq 2b(1 - b/d)/3s$, and (ii) $\|x - x_i\|_1 \geq 2b(1 - b/d)$ for all $(x_i, y_i) \in S$, with $y_i \neq y_{NN}$. Let $w_0, w_1 \in \{0,1\}^m$ be the* FBF*s constructed using S^0 and S^1 respectively. Then, with probability at least $1 - \delta$ (over the random choice of lifting matrix M),* FBFC *prediction on x agrees with the* 1NNC *prediction on x.*

Remark 3. *Note that, in the worst case, when $b = \frac{d}{2}$ and n takes the maximum possible value, that is $n = \binom{d}{b}$, $\frac{s}{d} \leq 1 - \frac{\log(b+1)}{4b} = 1 - \frac{\log d/2+1}{2d}$, implying that the lifting matrix is not very sparse. Otherwise, when either $n \ll \binom{d}{b}$ or $b \ll d$, $s \ll d$, implying a very sparse lifting matrix M_m^s.*

S2.2.2 Special Case II: Permutation invariant distribution in $\mathbb{R}^d$. The previous result focused on binary data (that is, $\mathcal{X} \subset \{0,1\}^d$). Here we focus on *permutation invariant* distributions in $\mathbb{R}^d$ and present a similar result for $\mathcal{X} \subset \mathbb{R}^d$ – we show FBFC agrees with 1NNC in $\mathbb{R}^d$ with high probability. Permutation invariant distributions in the FBF context was introduced in Dasgupta et al. [4] and defined as a distribution P over $\mathbb{R}^d$ if, with any permutation σ of $\{1, 2, \ldots, d\}$ and any $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $P(x_1, \ldots, x_d) = P(x_{\sigma(1)}, \ldots, x_{\sigma(d)})$. Precisely, we show the following:

Theorem S3. *Let S be a training set as given above. Fix any $\delta \in (0,1)$, $s \ll d$, and set $\rho \geq \frac{48}{\mu}\ln(8/\delta)$ and $m \geq 14n\rho/\delta$, where $\mu = \min\left\{\mathbb{E}_M\left((w_0^\top h(x))/\rho\right), \mathbb{E}_M\left((w_1^\top h(x))/\rho\right)\right\}$, $h(x)$ is the* FlyHash *(1), and $w_0, w_1 \in \{0,1\}^m$ are the* FBF*s constructed using S^0 and S^1 respectively. For a test point $x \in \mathbb{R}^d$, sampled from a permutation invariant distribution, let x_{NN} be its nearest neighbor from S measured using ℓ_∞ metric, which satisfies $\|x - x_{NN}\|_\infty \leq \Delta/s$, where $\Delta = \frac{1}{2}\left(\tau_x(2\rho/m) - \tau_x(\rho/m)\right)$ and has label $y_{NN} \in \{0,1\}$.*

Then, with probability at least $1-\delta$ (over the random choice of lifting matrix M), FBFC *prediction on x agrees with* 1NNC *prediction on x.*

S2.3 Proof sketch of Theorem S2

Proof (sketch). Without loss of generality, assume that x_{NN} satisfies the relation $\|x - x_{NN}\|_1 = 2b\epsilon$ for some $0 < \epsilon < 1$ and $y_{NN} = 1$. Clearly, 1-NN classifier will predict x's class label to be 1. For any $x, x' \in \mathcal{X}_b$, the structural assumption of this lemma allows us to write $q(x,x') \approx \left(x^\top x'/b\right)^s$ and thereby, $s \approx \frac{\log(m/\rho)}{\log(d/b)} = \log_{d/b}(m/\rho)$. Combining this with part (iv) and (v) of lemma S1, the restriction on m as specified in the theorem and a simple algebraic manipulation yield, $\mathbb{E}_M\left(w_1^\top h(x)/\rho\right) \leq s\epsilon$ and $\mathbb{E}_M\left(w_0^\top h(x)/\rho\right) \geq 1 - b/d$. For appropriate choice of ϵ, and plugging the value of s, we get $\mathbb{E}_M\left(w_1^\top h(x)/\rho\right) < \mathbb{E}_M\left(w_0^\top h(x)/\rho\right)$. The desired result then follows by applying lemma S4, provided ρ is large.

The following concentration result is standard and a similar form has appeared in [4]. Due to space limitations we omit its proof.

Lemma S4. *Let $x_1, \ldots, x_{n_1} \in \mathcal{X}_b$ be the unlabeled examples of S^1 and let $\tilde{x}_1, \ldots, \tilde{x}_{n_0} \in \mathcal{X}_b$ be the unlabeled examples of S^0. Pick any $\delta \in (0,1)$ and $x \in \mathcal{X}_b$. With probability at least $1-\delta$ over the choice of random projection matrix M, the following holds,*
(i) $\frac{1}{2}\mathbb{E}_M\left((w_1^\top h(x))/\rho\right) \leq (w_1^\top h(x))/\rho \leq \frac{3}{2}\mathbb{E}_M\left((w_1^\top h(x))/\rho\right)$
(ii) $\frac{1}{2}\mathbb{E}_M\left((w_0^\top h(x))/\rho\right) \leq (w_0^\top h(x))/\rho \leq \frac{3}{2}\mathbb{E}_M\left((w_0^\top h(x))/\rho\right)$
provided $\rho \cdot \min\left\{\mathbb{E}_M\left(\frac{w_0^\top h(x)}{\rho}\right), \mathbb{E}_M\left(\frac{w_1^\top h(x)}{\rho}\right)\right\} \geq 12\ln(4/\delta)$.

S2.4 Result for multi-class classification

Theorem S2 can be easily extended to multi-class setting involving L classes in a straight forward manner by applying concentration result the terms $\left(\frac{w_i^\top h(x)}{\rho}\right)$, for $i \in [L]$, and using a union bound.

Corollary S5. *Given a training set $S = \{(x_i, y_i)\}_{i=1}^{n_0+\cdots+n_{L-1}} \subset \mathcal{X}_b \times \mathcal{Y} \subset \{0,1\}^d \times \{0,1,\ldots,L-1\}$ of size $\sum_{i=0}^{L-1} n_i$, let $S = \cup_{i=0}^{L-1} S^i$, where S^i is the subset of S with label i satisfying $|S^i| = n_i$ and $n = \max\{n_0, \ldots, n_{L-1}\}$. For any test example $x \in \mathcal{X}_b$, let its closest point from S measured using ℓ_1 metric be x_{NN} having label $y_{NN} \in \{0, \ldots, L-1\}$. Fix any $\delta \in (0,1)$ and set $\rho \geq \frac{12}{\mu}\ln(2L/\delta)$, $m \geq (d/b)n\rho$, and $s = \log_{d/b}(m/\rho)$, where,*
$\mu = \min\left\{\mathbb{E}_M\left((w_0^\top h(x))/\rho\right), \ldots, \mathbb{E}_M\left((w_{L-1}^\top h(x))/\rho\right)\right\}$ *and $h(x)$ is the* FlyHash *function from equation 1. Assume that for all $(x_i, y_i) \in S$, with $y_i \neq y_{NN}$, $\|x - x_i\|_1 \geq 2b(1 - b/d)$ and x_{NN} satisfies $\|x - x_{NN}\|_1 \leq \frac{2b(1-b/d)}{3s}$. Let $w_0, \ldots, w_{L-1} \in \{0,1\}^m$ be the* FBF*s constructed using $S^0, \ldots, S^{L-1}$ respectively. Then, with probability at least $1-\delta$ (over the random choice of projection matrix M), prediction of* FBFC *on x agrees with the prediction of* 1NNC *on x.*

S2.5 Proof sketch of Theorem S3

Proof (sketch). Without loss of generality, assume that $y_{NN} = 1$. We first show that $\mathbb{E}_M\left(w_1^\top h(x)/\rho\right) < \mathbb{E}_M\left(w_0^\top h(x)/\rho\right)$ with high probability and then using standard concentration bound presented in lemma S4, we achieve the desired result. Since $\|x - x_{NN}\|_\infty \leq \Delta/s$, using lemma 9 of [4], we get $q(x, x_{NN}) \geq 1/2$. Combining this with part (iv) of lemma S1, we get $\mathbb{E}_M\left(w_1^\top h(x)/\rho\right) \leq 1/2$. Next using properties of permutation invariant distribution, linearity of expectation, the Markov's inequality, and part (v) of lemma S1 we show that $\mathbb{E}_M\left(w_0^\top h(x)/\rho\right) \geq 1 - \alpha$ with probability at least $1 - \delta/2$, if $m \geq 2\rho n_0/(\alpha\delta)$. Choosing α appropriately and applying concentration bound from lemma S4, the result follows.

The above result can similarly be extended to a multi-class setting in a straight forward manner.

Table S1: Details of data sets used for FBFC/FBFC* specific evaluations. For MNIST, we flatten the 28×28 images to points in $\mathbb{R}^{784}$.

Data set	n	d	L	Experiment
Gina Prior 2	3468	784	10	OpenML
USPS	9294	256	10	OpenML
Letters	20000	16	26	OpenML
MNIST	60000	784	10	Vision

S3 SUPPLEMENTARY MATERIAL FROM §4

Implementation & Compute Resource. The proposed scheme is implemented in Python 3.8 to fit the `scikit-learn` API [28]. We use `scikit-learn` implementations of various baselines, and generate synthetic data with the `make_classification` function [12] in the `data` module of `scikit-learn`. The experiments are performed on a 16-core 128GB machine running Ubuntu 18.04. The code is available at https://github.com/rithram/fbfc.

Details on baselines. Here we detail all the baselines considered in our empirical evaluations and their respective hyper-parameter and the subsequent hyper-parameter optimization.

- **SBFC.** We tune over the `SimHash` dimensionality m in the range $m \in [1, d]$ (the traditional use) and projecting up $m \in [d, 2048d]$, where d is the data dimensionality.
- **LR.** We consider logistic regression trained with a single epoch of a stochastic algorithm. We utilize the `scikit-learn` implementation (`linear_model.LogisticRegression`) and tune over the following hyper-parameters – (a) penalty type (ℓ_1/ℓ_2), (b) regularization $\in \left[2^{-10}, 2^{10}\right]$, (c) choice of solver (liblinear [8], SAG [32], SAGA [7]), (d) with/without intercept, (e) one-vs-rest or multinomial for multi-class, (f) with/without class balancing. We consider a total of 960 hyper-parameter configurations for each experiment.
- **MLPC.** We consider a multi-layer perceptron trained for a single epoch with the "Adam" stochastic optimization scheme [19]. We use `sklearn.neural_network.MLPClassifier` and tune over the following hyper-parameters – (a) number of hidden layers $\{1, 2\}$, (b) number of nodes in each hidden layer $\{16, 64, 128\}$, (b) choice of activation function (ReLU/HyperTangent), (d) regularization, (e) batch size $\in \left[2, 2^8\right]$, (f) initial learning rate $\in \left[10^{-5}, 0.1\right]$. The remaining hyper-parameters are set to the defaults in `scikit-learn`. This results in a total of 288 hyper-parameters configurations per experiment.