Fruit-fly Inspired Neighborhood Encoding for Classification

Kaushik Sinha (Wichita State University & IFML)
Parikshit Ram (IBM Research AI)

Neurobiological Inspiration

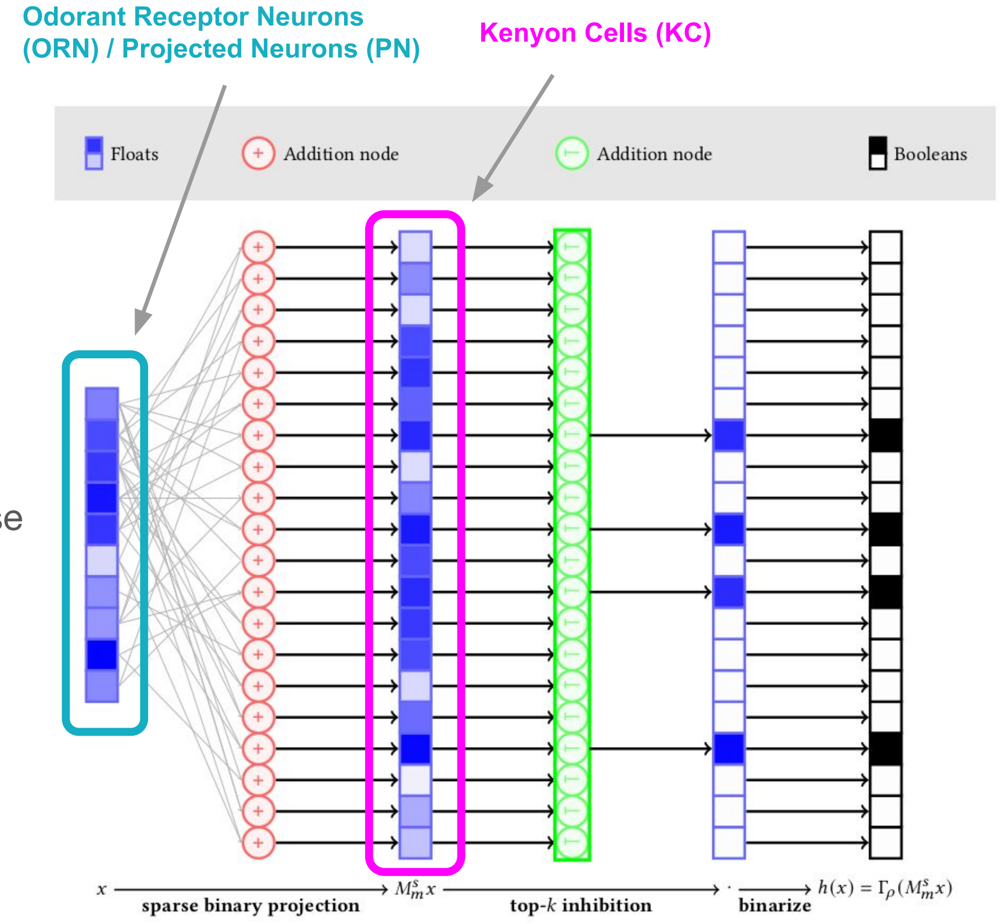
- ➔ Biological systems have served as inspiration to modern deep learning
  - ◆ neural networks, convolutions, dropout, attention mechanisms
  - ◆ amazing empirical performance in computer vision, NLP and reinforcement learning tasks
- ➔ Modern SOTA intelligent systems are not biologically viable anymore
  - ◆ however, the biological inspiration was critical
- ➔ This has motivated a lot of research into identifying other biological systems
  - ◆ that can inspire development of new and powerful learning mechanisms
  - ◆ provide novel critical insights into the workings of intelligent systems

What is FlyHash?

➔ A mapping from $\mathbb{R}^d$ to $\{0,1\}^m$ defined as:

$$h(x) = \Gamma_\rho(M_m^s x)$$

- $M_m^s \in \{0,1\}^{m \times d}$ is a randomized sparse binary lifting matrix with $s \ll d$ non-zero entries per row
- $\Gamma_\rho : \mathbb{R}^m \to \{0,1\}^m$ is the winner-take-all function that sets the $\rho \ll m$ largest entries of a vector in $\mathbb{R}^m$ to 1 and the rest to 0



What is Fly Bloom Filter?

- ➔ A Fly Bloom Filter (FBF) $w \in \{0,1\}^m$ succinctly summarizes data
- ➔ Start with $w = \mathbf{1}_m$ (the vector of all ones)
  - ◆ for an inlier point x_{in}, w is updated by zeroing the elements of w corresponding to the non-zero indices $h(x_{\text{in}})$ which can be compactly represented as

$$w \leftarrow w \wedge (w \oplus h(x_{\text{in}})) = w \wedge \overline{h(x_{\text{in}})}$$

Element-wise XOR operation

- ➔ The above construction ensures
  - ◆ any $x = x_{\text{in}}$ or similar to x_{in}, yields low novelty score $w^\top h(x)$
  - ◆ any novel point x_{nov} with FlyHash $h(x_{\text{nov}})$, which is not similar to any of the inliers yields high novelty score

Fruit-fly inspiration

- ➔ Recently, neurobiological mechanisms have been identified in the olfactory circuit of the brain in a common fruit-fly, where
  - ◆ an odor activates a small set of Kenyon Cells (KC) which represent a “tag” for the odor
  - ◆ the tag generation process can be viewed as a natural hashing scheme, termed FlyHash
    - *a very sparse high-dimensional representation (2000 dimensions with 95% sparsity) and the tag/hash creates a response in a specific mushroom body output neuron (MBON) corresponding to the perceived novelty of the odor*
  - ◆ Dasgupta et al. interpret the KC-MBON synapses as a Bloom Filter that creates a “memory” of all the odors encountered by the fruit-fly
    - *reprogrammed this Fly Bloom Filter (FBF) as a novelty detection mechanism that performs better than other locality sensitive Bloom Filter-based novelty detectors for neural activity and vision datasets*
  - ◆ FBFs have also been used for similarity search and word embedding tasks

Motivating questions

- ➔ Can we reprogram FBF to the supervised learning setting and devise a classifier 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?

Contributions

- ➔ Design of a novel FBF based classifier, FBFC
  - ◆ ***Learning:*** additions only operations, single-pass, no loss-minimizing optimization
  - ◆ ***Inference:*** an efficient FlyHash followed by a sparse binary additions-only dot-product
- ➔ A thorough empirical comparison of FBFC to standard classifiers
  - ◆ on over 71 datasets demonstrating significant gains over other single-pass schemes
- ➔ A theoretical examination of the proposed scheme
  - ◆ establish 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

Learning

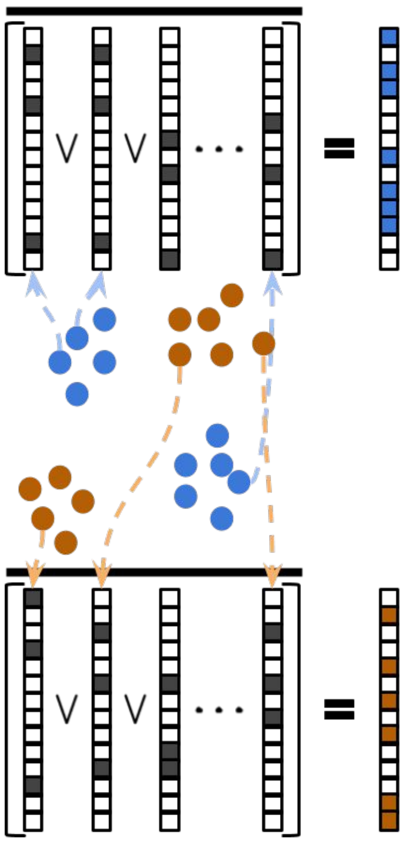
➔ Learns separate FBFs for each class

➔ The learning scheme is online

- an example can be used in isolation to update the model
- an example does not need to be seen more than once

TrainFBFC: $(S, m, \rho, s) \to (M_m^s, \{w_l, l \in [L]\})$
    Sample $M_m^s \in \{0,1\}^{m \times d}$ with s NNZ/row
    Initialize $w_1, \ldots, w_L \leftarrow \mathbf{1}_m \in \{0,1\}^m$
    for $(x, y) \in S$ **do**
        $h(x) \leftarrow \Gamma_\rho(M_m^s x)$
        $w_y \leftarrow w_y \bigwedge \overline{h(x)}$
    end
    return $(M_m^s, \{w_l, l \in [L]\})$
end

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



Inference

- Construction of w_l , $l \in [L]$ ensures
  - any point x with class label l is treated as inlier for class l and does not affect class encoding $w_{l'}, l' \neq l, l' \in [L]$
- For a test point x
  - compute per-class novelty score $f_l(x) \in [0, 1], l \in [L]$ and predict the class label to be the one with lowest novelty score

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

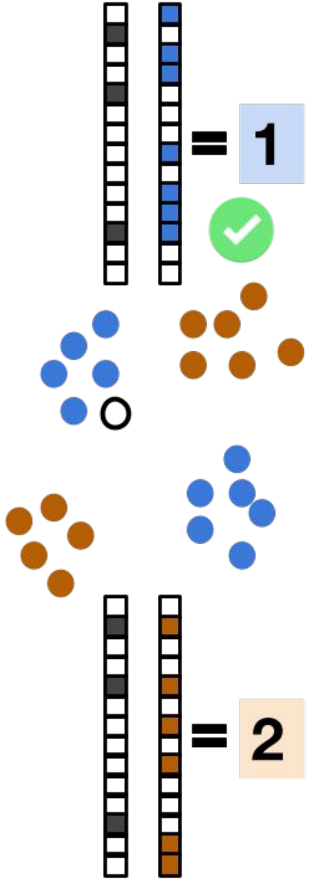
InferFBFC: $(x, M_m^s, \rho, \{w_l, l \in [L]\}) \to \hat{y}$

$h(x) \leftarrow \Gamma_\rho \left(M_m^s x\right)$

$\hat{y} \leftarrow \operatorname{argmin}_{l \in [L]} (1/\rho)\, w_l^\top h(x)$

return $\hat{y}$

end



Theoretical analysis

→ Computational complexities

- FBFC Training
  - Running time: $O(nm \cdot \max\{s, \log \rho\})$
  - Memory overhead: $O(m \cdot \max\{s, L\})$
- FBFC Inference
  - Running time: $O\left(m \cdot \max\left\{s, \log \rho, (\rho L/m)\right\}\right)$
  - Memory overhead: $O(\max\{m, L\})$
- For multi-class classification problem with large number of of labels (large L), inference can be made in time sub-linear in L using maximum inner product search (MIPS)

n : size of the training set
m : lifting dimensionality
s : # of non-zeros in each row of the lifting matrix
ρ : # of largest non-zero entries in FlyHash

Learning theoretic properties

- Intuition for connection to 1-NNC
  - novelty score of any test point corresponds to how ***“far”*** the point is from the distribution encoded by w
  - in FBFC, using minimum novelty score to label x is equivalent to labeling x with the class whose distribution/encoding is ***“closest”*** to x
  - motivates to study how the FBFC is related to the well-studied nearest-neighbor classifier
- Basic idea
  - we establish conditions under which the expected novelty score of the class associated with the nearest neighbor of a test point is the smallest among all expected novelty scores
  - we then use standard concentration argument to derive a high probability statement

Learning theoretic properties

- Main theorem
  - in the binary classification setting, let $S = \{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \{0, 1\}$ be a training set of size n
  - under mild structural and/or distributional assumption on $\mathcal{X}$ we prove

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* 1*NNC on* x.

- We consider two special cases
  - binary feature vectors with fixed number of ones
  - test point comes from a permutation invariant distribution in $\mathbb{R}^d$

Robust Learning

- Problem with mislabeled examples
  - a single mislabelled example $(x_{\text{mis}}, y_{\text{mis}})$ modifies the FBF $w_l, y_{\text{mis}} = l$
  - test point x similar to x_{mis} may get misclassified since it receives low novelty score with respect to FBF w_l
  - this lack of robustness can not be corrected due to single pass nature of FBFC
- Remedy
  - use non-binary FBF called FBF* to captures neighborhood and distribution more effectively
  - coordinates of FBF* w_l are decayed at a rate controlled by a parameter c based on neighborhood and distributional information of class l and takes values in [0,1]

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

Robust Learning

➔ Incorporating the FBF* learning dynamics give rise to robust classifier FBFC*

- training of FBFC* is slightly modified
- inference is still exactly the same as FBFC
- simplicity and interpretability of FBF is not completely lost in FBF*
- empirical performance of FBFC* is superior compared to FBFC

TrainNBFBFC: $(S, m, \rho, s, c) \rightarrow (M_m^s, \{w_l, l \in [L]\})$

$\quad$ Sample $M_m^s \in \{0,1\}^{m \times d}$ with s NNZ/row

$\quad$ Initialize $z_1, \ldots, z_L \leftarrow \mathbf{0}_m \in \mathbb{R}^m$

$\quad$ **for** $(x, y) \in S$ **do**

$\quad\quad$ $h(x) \leftarrow \Gamma_\rho(M_m^s x)$

$\quad\quad$ $z_y \leftarrow z_y + h(x)$

$\quad$ **end**

$\quad$ $w_l \leftarrow (1-c)^{\odot z_l}, l \in [L]$

$\quad$ **return** $(M_m^s, \{w_l, l \in [L]\})$

end

Robustness to noise: Synthetic data

➔ Setting

- synthetic 5-class classification problem in $\mathbb{R}^d$
- add increasing levels of noise to the labels
- for FBFC*, c=0.9, for FBFC, c is implicitly set to 1

➔ FBFC* is more robust to labeling noise

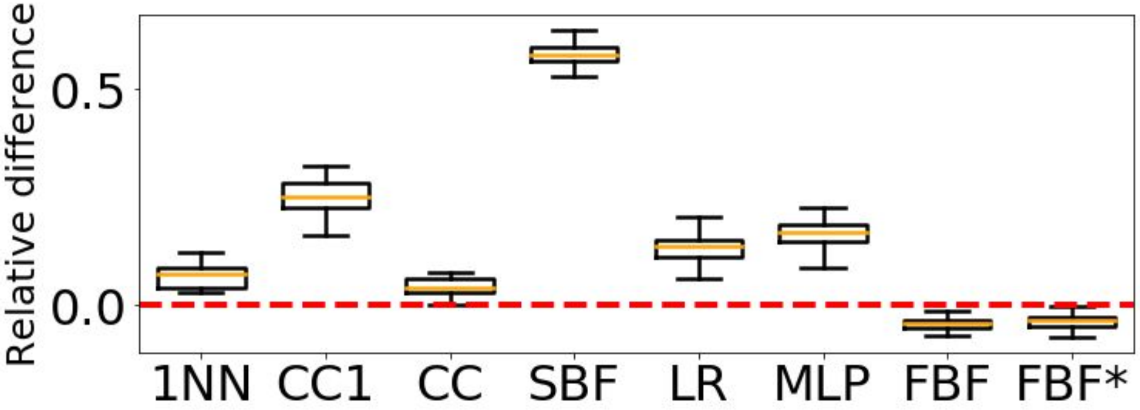
Label noise level	1.0%	5.0%	10%	25%
FBFC accuracy (%)	56.6±0.8	54.5±0.9	52.0±1.3	44.0±1.0
FBFC* accuracy (%)	60.1±0.7	58.9±0.9	57.1±1.2	50.3±1.2
Rel. improvement (%)	6.1±0.7	8.1±0.9	9.8±1.2	14.2±1.8

Data sets and Baselines

- Data sets
  - Synthetic binary and continuous data (see paper)
  - 71 binary and multiclass classification data from OpenML
- Baselines
  - K-nearest neighbor classifier (kNNC), nearest neighbor classifier (1NNC)
  - Prototype based classifiers (CC1, CC)
  - Locality sensitive bloom filter classifier using SimHash (SBFC)
  - Linear classifier (LR)
  - Multi-layered perceptron (MLPC)

Comparison to baselines: Binary synthetic data

- ➔ Setting
  - point $x \in \{0,1\}^d$ with $|x| = b < d$
  - generated 30 datasets for 5 classes with 3 clusters per class, d=100, b=20
  - points in the same cluster belonged to the same class
- ➔ All results are aggregated over 30 datasets and are relative to k-NNC (lower the better)

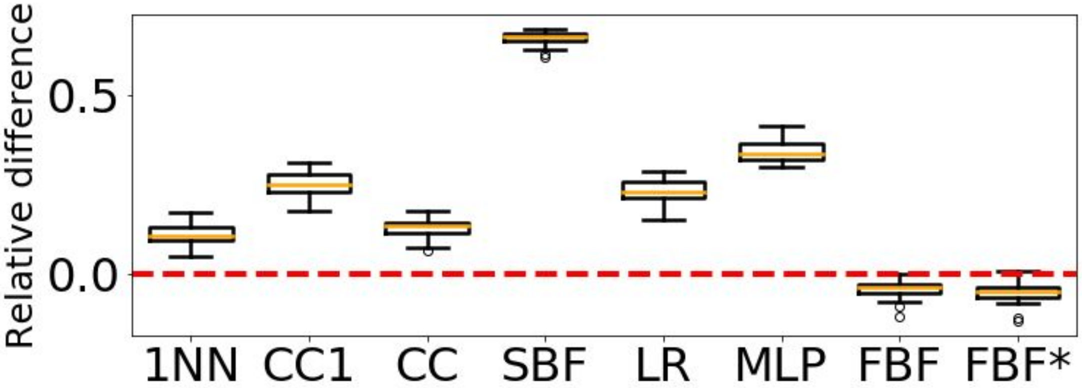


Comparison to baselines: Real synthetic data

➔ Setting

- point $x \in \mathbb{R}^d$, d=100
- generated 30 datasets for 5 classes with 3 clusters per class
- points in the same cluster belonged to the same class

➔ All results are aggregated over 30 datasets and are relative to k-NNC (lower the better)



Comparison to baselines: OpenML

- ➔ Three groups of data sets for comparisons:
  - All: over all 71 data sets
  - Group A: 37 data sets where kNNC performs best among baselines
  - Group B: 34 data sets where LR performs best among baselines
- ➔ All numbers relative to the FBFC* performance
  - Fraction of data sets where FBFC* beats baselines
  - Median margin of improvements aggregated over all data sets in the group in %
  - Statistical significance in terms of whether or not we can reject the null hypothesis $\mathbf{H_0}$ of the paired t-test that the performance of FBFC* and the baseline are similar at significance level 0.01 -- ● denotes we can reject; ○ denotes we cannot reject

Comparison to baselines: OpenML

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%) ●

➔ Robustness in FBFC* leads to significant gains on real data sets

Comparison to baselines: OpenML

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%) ●

➔ FBFC* significantly outperforms SBFC, indicating locality sensitivity is not sufficient for classification

➔ High dimensional sparse hash is critical

Comparison to baselines: OpenML

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%) ●

- ➔ Overall matches kNNC and outperforms 1NNC
- ➔ In Group A, FBFC* underperforms but not significantly (cannot reject $\mathbf{H}_0$)
- ➔ In Group B, FBFC* significantly improves upon kNNC and 1NNC

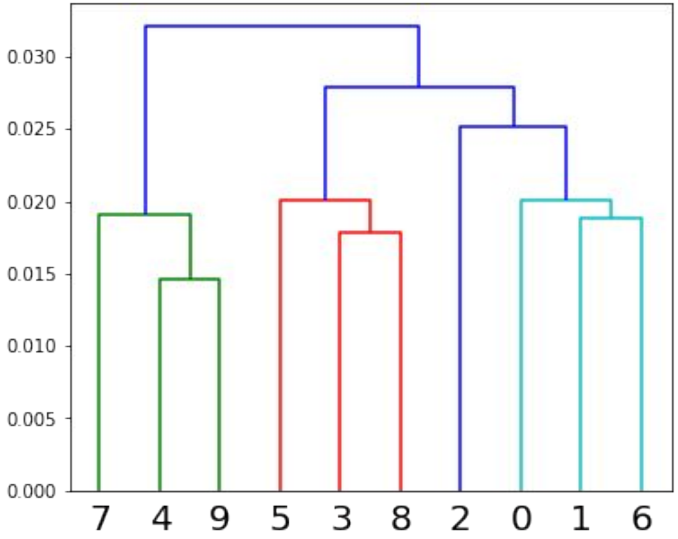
Comparison to baselines: OpenML

METHOD	ALL (71 SETS)	**Group A** (37/71)	**Group B** (34/71)
<u>kNNC</u>	0.51 (▲0.05%) ○	0.24 (▼1.08%) ○	0.79 (▲3.98%) ●
<u>1NNC</u>	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%) ●
<u>CC</u>	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%) ●

- ➔ Overall FBFC* matches parametric LR and outperforms MLPC
- ➔ In Group A, FBFC* significantly outperforms LR and MLPC
- ➔ In Group B, FBFC* underperforms LR but not significantly (cannot reject $\mathbf{H}_0$)

Problem insights: MNIST

- ➔ Similarity between two classes is defined as $s(l, l') = \frac{\langle w_l, w_{l'} \rangle}{\|w_l\| \, \|w_{l'}\|}$
- ➔ Constructed dendrogram based on class similarity
- ➔ Observed meaningful semantic hierarchy without additional supervision

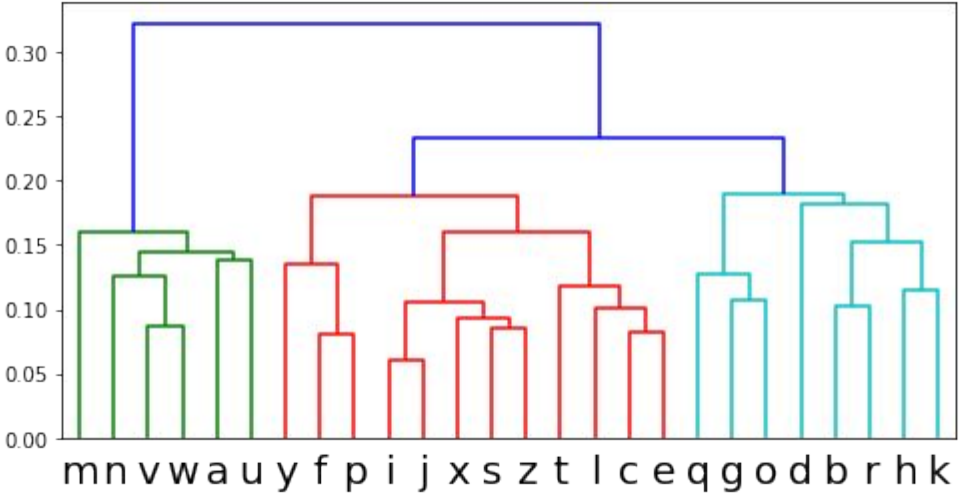


Problem insights: Letters

➔ Similarity between two classes is defined as $s(l, l') = \frac{\langle w_l, w_{l'} \rangle}{\|w_l\| \, \|w_{l'}\|}$

➔ Constructed dendrogram based on class similarity

➔ Observed meaningful semantic hierarchy without additional supervision



Conclusions

➔ We proposed a novel neuroscience inspired Fly Bloom Filter based classifier FBFC that can be trained in single pass over the training set

➔ Inference requires an efficient FlyHash followed by a very sparse dot product

➔ On the theoretical side, we established conditioned under which FBFC agrees with the well studied nearest neighbor classifier

➔ Empirically, we validated our proposed scheme with 71 datasets of varied data dimensionality and demonstrated effectiveness of our proposed classifier

Future Work

- ➔ Extend theoretical guarantee of FBFC and FBFC* for general $\mathbb{R}^d$ exploring various data dependent assumptions
- ➔ Extend theoretical results that connects FBFC to 1-NNC to the setting where FBFC connects to k-NNC
- ➔ Utilize sparse and randomized nature of FBFC to investigate differential privacy properties of FBFC

Thank you!

Positioning against existing ML models

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.