Locally Sensitive Hashing for the Content Based Image Retrieval

Masoud Badiei Khuzani, Prof. Yinyu Ye, Prof. Sandy Napel, Prof. Lei Xing

---

July, 2020

Stanford University, Department of Radiation Oncology.

Content-based Image Retrieval

- Content-based image retrieval is the task of searching images by analyzing the contents of the image rather than the metadata such as keywords, tags, or descriptions associated with the image.

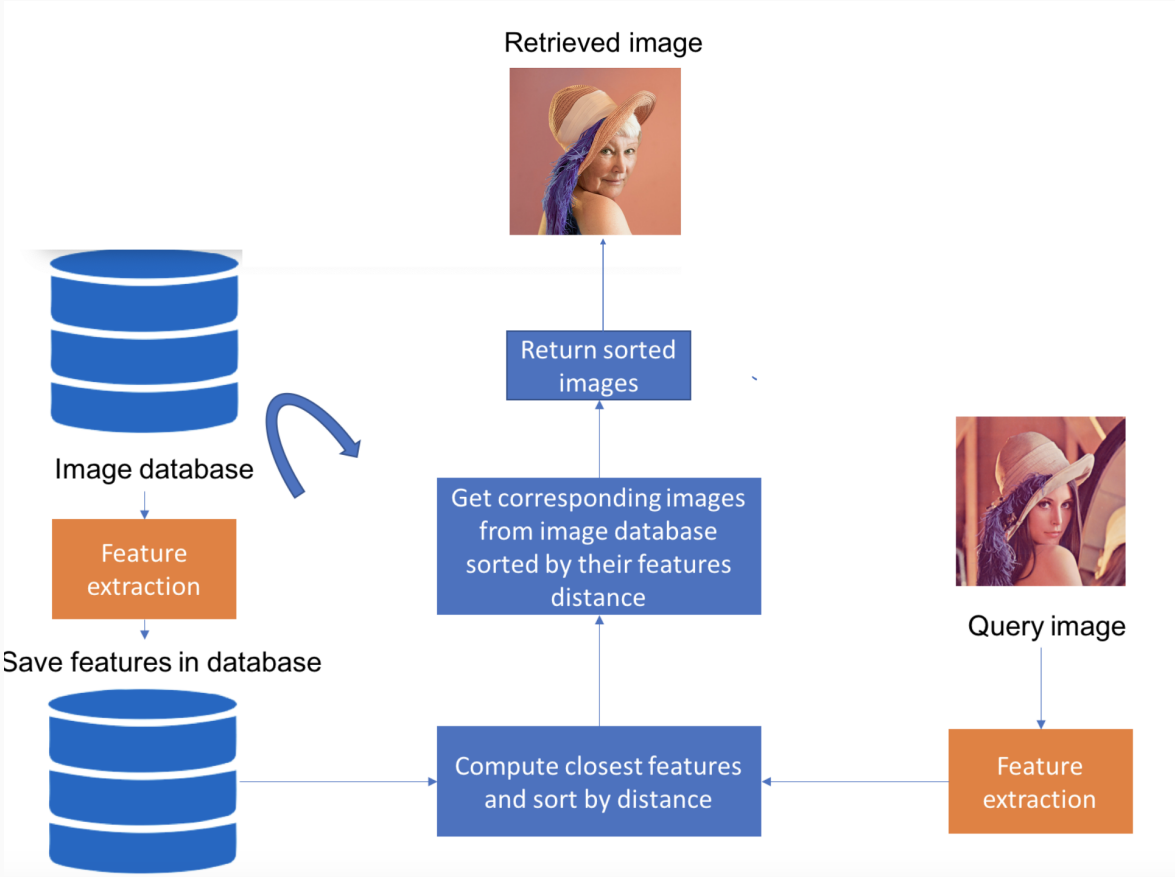


Figure 1: Content-Based Image Retrival

Content-based Image Retrieval for Medical Imaging

- Consider the image retrieval problem:

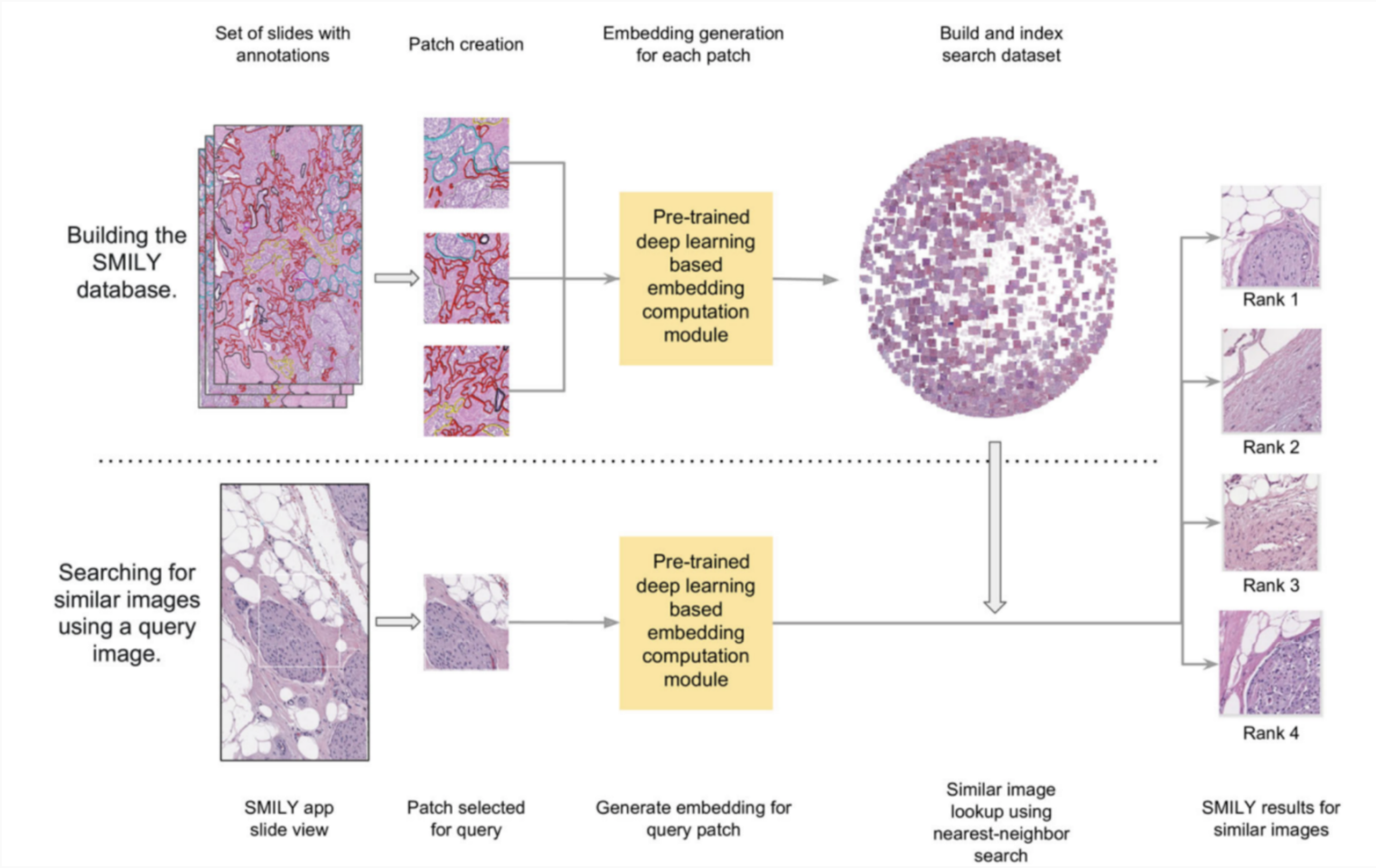


Figure 2: Image retrieval (N Hedge, et al., Nature 2019).

Other Variants of the Retrieval Systems.

- Such retrieval systems can be applied to histopathological images

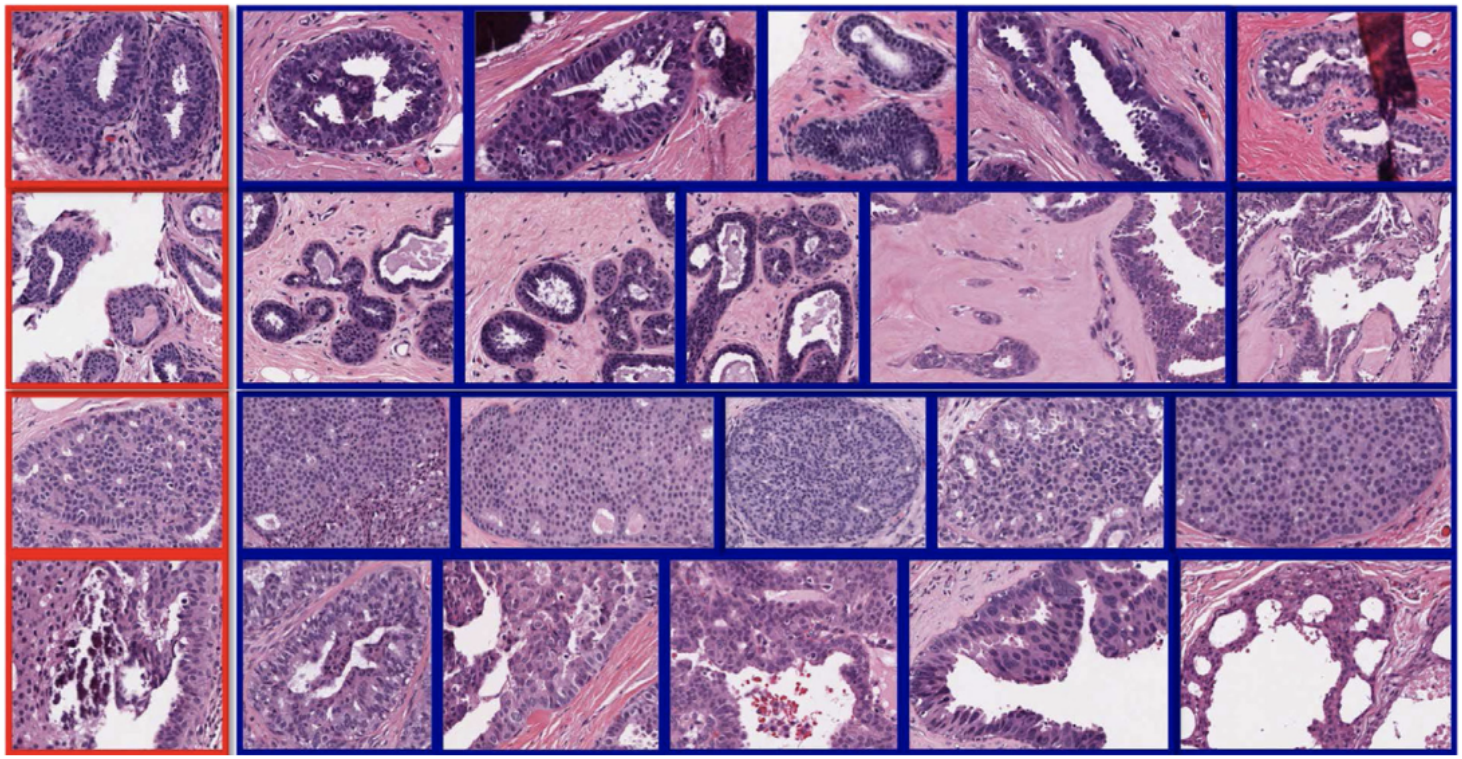


Figure 3: Image retrieval (Zhang, et al., IEEE Trans. Med 2015).

Other Variants of the Retrieval Systems.

- Such Retrieval system can also be applied to morphological neuron data-base

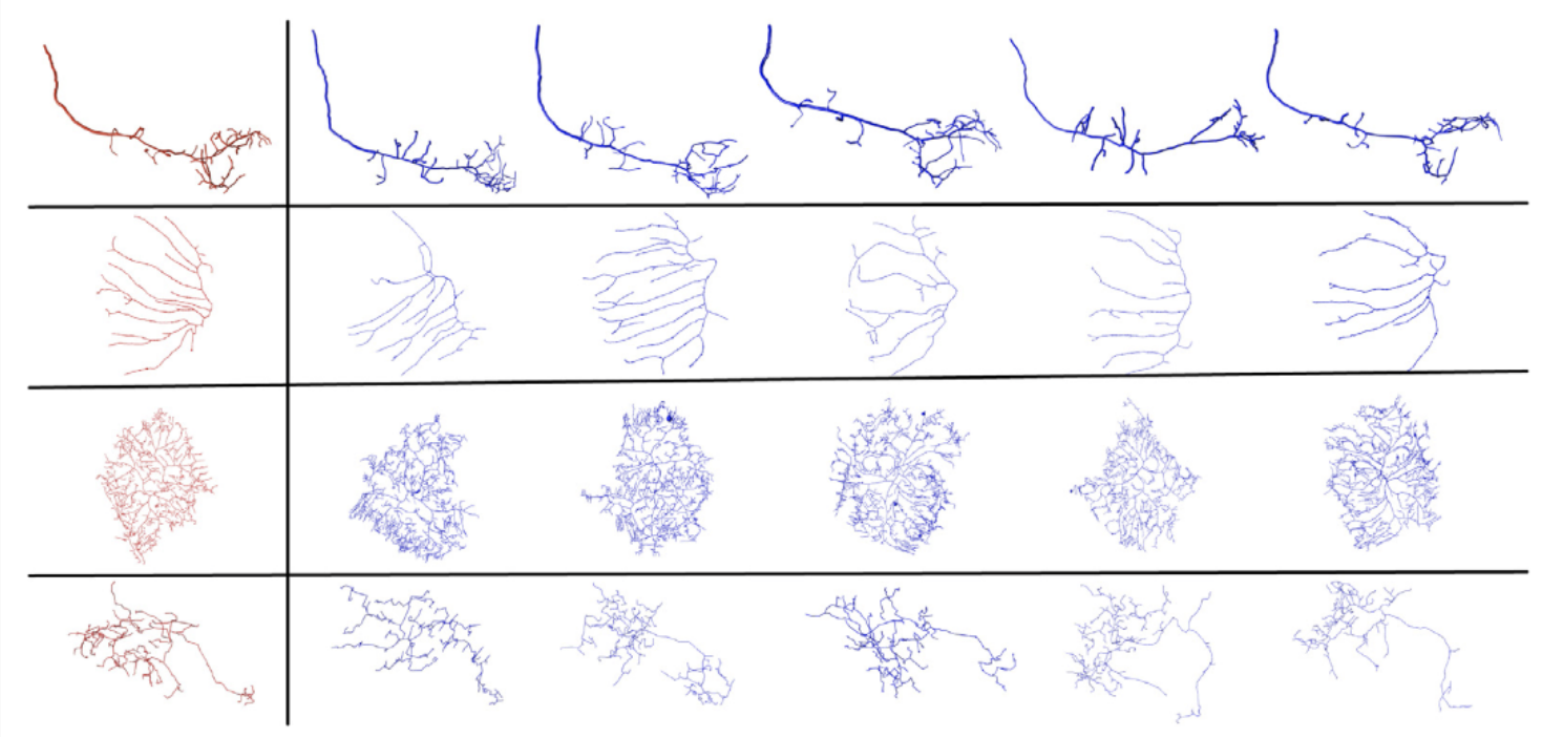


Figure 4: Image retrieval (Zhongyu Li, IEEE Trans. Pattern Recogn. 2017).

Other Variants of the Retrieval Systems.

- The crux of such retrieval system is a hash function that computes a binary code from the representations of the input

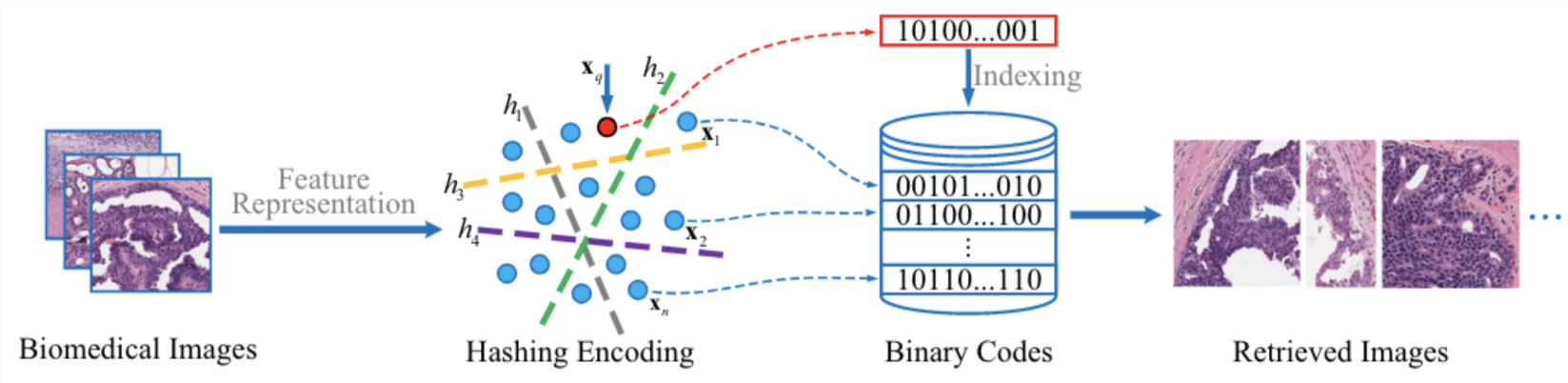


Figure 5: Image retrieval (Zhongyu Li, IEEE Trans. Pattern Recogn. 2017).

What is hashing?

- A hash function is a function takes a group of characters or numbers (called a key) and maps it into a value of a certain length.
- Hash functions and their associated hash tables are used in data storage and retrieval applications to access data in a small and nearly constant time per retrieval.
- A **collision** or **clash** occurs when two distinct pieces of data have the same hash value

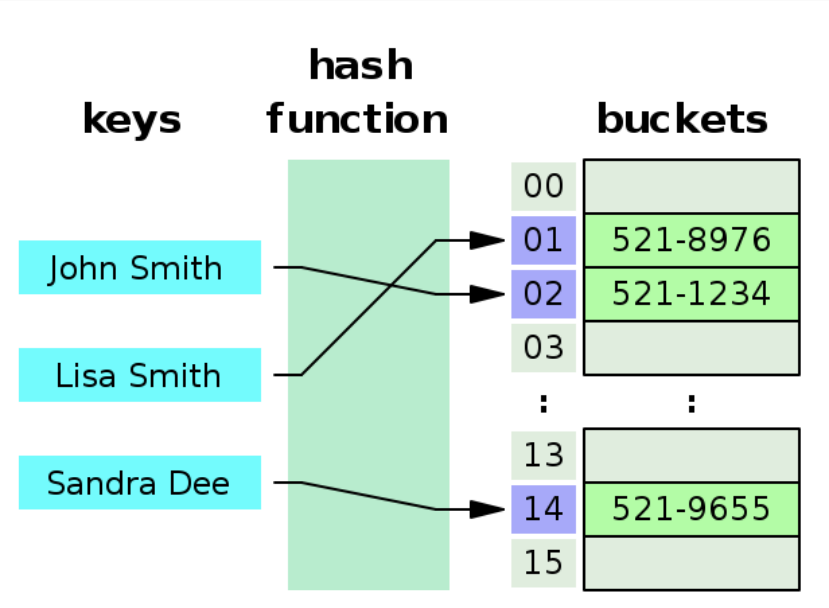


Figure 6: Hash function

What is the locally sensitive hashing (LSH)?

- A locally sensitive hash (LSH) function is an algorithmic technique that hashes similar input items into the same "buckets" with high probability.
- In a sense, we want a *controlled* collision in LSH.
- Mathematically,

$$\mathbb{P}_{h\in\mathcal{H}}[h(\boldsymbol{x}) = h(\tilde{\boldsymbol{x}})] = \mathrm{sim}(\boldsymbol{x}, \tilde{\boldsymbol{x}}). \tag{1}$$

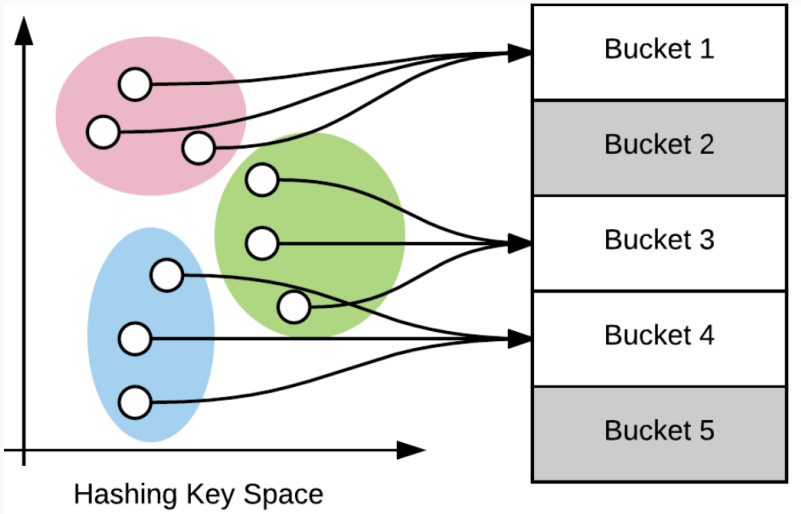


Figure 7: Locally Sensitive Hash function.

What is the use case of LSH?

- Given a query, we need to do a look up via nearest neighbor search.
- The complexity of k-NN using Euclidean distance is $\mathcal{O}(mdk)$, where m is the size of data-base and d is the dimension of data-points.

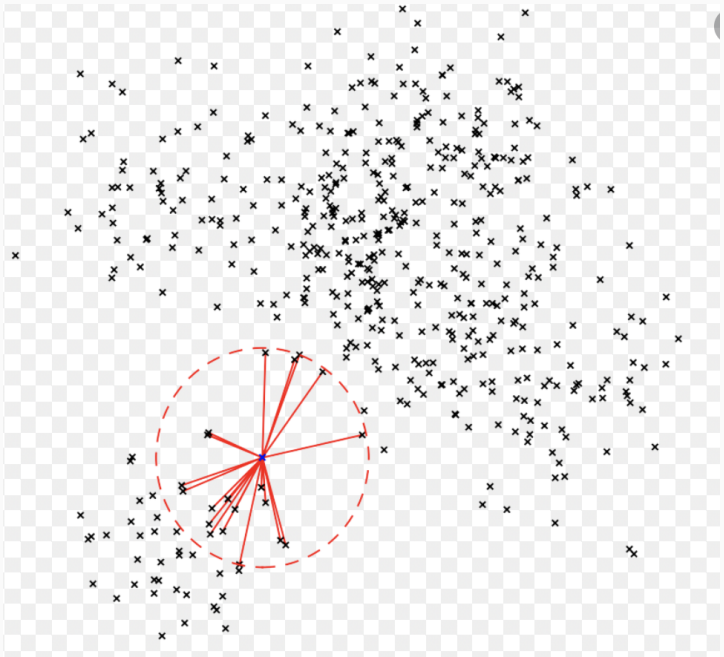


Figure 8: Image Lookup via the Nearest Neighborhood.

An example of LSH

- We can speed up the look up by using LSH that maps data points to a discrete set $h : \mathbb{R}^d \mapsto \{0, 1, 2, \cdots, q-1\}$.
- A possible hash function for $q = 2$ (Hamming code) is proposed by Raginsky, et al. [NIPS 2009], where

$$h_{t,b,\boldsymbol{\omega}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{1}{2}\Big[1 + \operatorname{sign}(\cos(\langle \boldsymbol{\omega}, \boldsymbol{x} \rangle + b) + t)\Big], \tag{2}$$

  where

  1. $t \sim \text{Uniform}[-1, 1]$.
  2. $b \sim \text{Uniofmr}[-\pi, \pi]$.
  3. $\boldsymbol{\omega} \sim \mathsf{N}(\mathbf{0}, \boldsymbol{I}_{d\times d})$.

- A hash code of length n can be constructed by sampling these random variables i.i.d., i.e., $h^n(\boldsymbol{x}) = (h_{t_1,b_1,\boldsymbol{\omega}_1}(\boldsymbol{x}), \cdots, h_{t_n,b_n,\boldsymbol{\omega}_n}(\boldsymbol{x}))$, where $t_1, \cdots, t_n \sim \text{Uniform}[-1, 1]$, $b_1, \cdots, b_n \sim \text{Uniofmr}[-\pi, \pi]$, and $\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_n \sim \mathsf{N}(\mathbf{0}, \boldsymbol{I}_{d\times d})$.

Simulations on MNIST Data-Set

- The hamming distance remains flat as the Euclidean distance change.

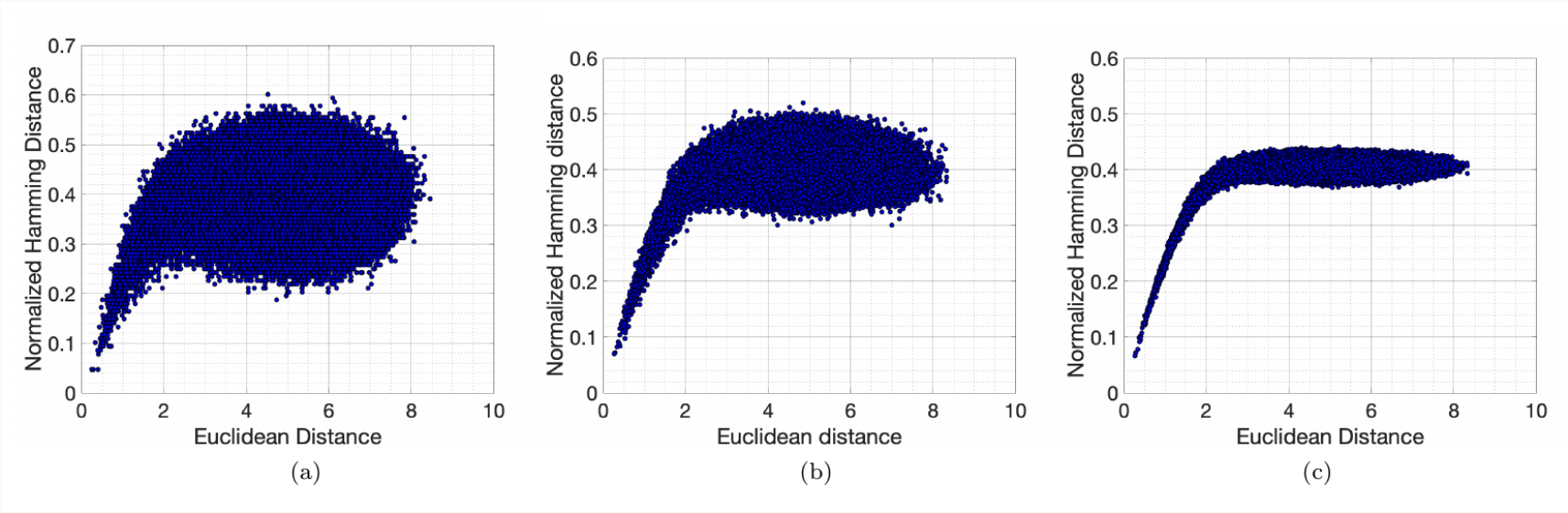


Figure 9: The scatter plots of normalized Hamming distance versus Euclidean distance for the hash function, where the hash functions. Panel (b): 512 bits, Panel (c): 4096 bits

Simulations on MNIST Data-Set

- Another way to look at the performance is via the Precision-Recall curve

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}.$$

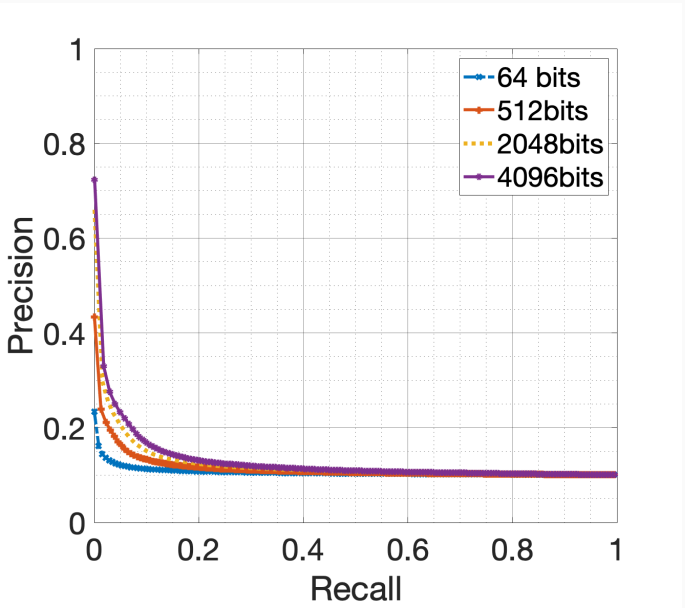


Figure 10: The precision-recall curve for different lengths of the hash codeword.

Simulations on MNIST Data-Set

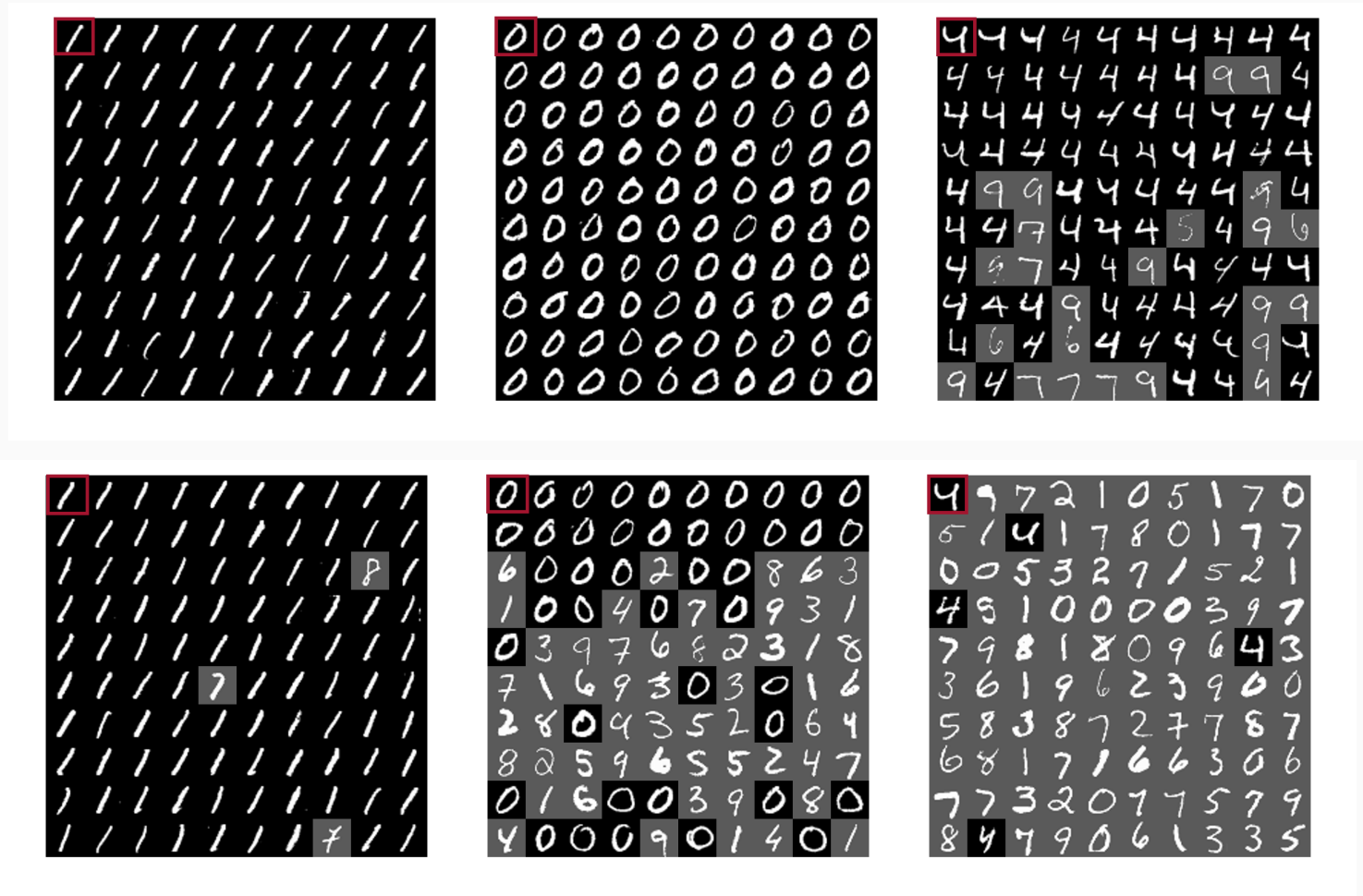


Figure 11: Examples of image retrieval for three query images on the MNIST database. The images are compressed via an autoencoder with 500 latent variables. The query image is in the top left of the collage (red box), and the incorrect retrieved images are shaded with the gray color. The top row shows top 100 neighbors for each query according to Euclidean distance. The bottom row shows nearest neighbors according to normalized Hamming distance with a 4096-bit code using an untrained

Why do we get a poor performance?

- The main theorem of Raginsky, et al. [NIPS 2009] computes the probability of collision in terms of a kernel function

$$\mathbb{P}\left[h_{t,b,\boldsymbol{\omega}}(\boldsymbol{x}) = h_{t,b,\boldsymbol{\omega}}(\tilde{\boldsymbol{x}})\right] = \frac{8}{\pi^2}\sum_{m=0}^{\infty}\frac{1-K(m\boldsymbol{x}, m\tilde{\boldsymbol{x}})}{4m^2-1},$$

  where we recall the hash function

$$h_{t,b,\boldsymbol{\omega}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{1}{2}\Big[1+\operatorname{sign}(\cos(\langle\boldsymbol{\omega},\boldsymbol{x}\rangle + b) + t)\Big], \tag{3}$$

- The main cause of the poor performance is due to the kernel $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \exp(-\|\boldsymbol{x}-\boldsymbol{x}\|_2^2)$ that is not refined for the hashing task.
- **Task 1**: We need to improve the performance of hash function by optimizing the kernel.
- **Task 2**: We need to extend the hamming codes to general non-binary alphabets.

A novel algorithm for the optimization of the radial kernels

- Since we would like to capture the Euclidean structure in image retrieval problems, we focus on the class of the *radial kernels*

$$\mathcal{K} \stackrel{\text{def}}{=} \left\{ K : \mathcal{X} \times \mathcal{X} \to \mathbb{R} : K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \psi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2), \psi \in C^1(\mathbb{R}) \right\}.$$

Theorem: (I. J. Schöenberg) A continuous function $\psi : \mathbb{R}_+ \to \mathbb{R}$ is positive semi-definite if and only if it admits the following integral representation

$$\psi(r) = \int_0^\infty e^{-tr^2} \mathrm{d}\mu(t),$$

for a finite positive Borel measure ν on $\mathbb{R}_+$. Moreover, if $\mathrm{supp}(\nu) \neq \{0\}$, then ψ is positive definite.

A novel algorithm for optimization of the kernel

- From the Schönberg's representation theorem, the following integral representation of the radial kernels follows

$$K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \int_0^\infty e^{-\xi \|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|_2^2} \mu(\mathrm{d}\xi), \quad \forall \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^d, \mu \in \mathcal{M}_+(\mathbb{R}_+),$$

- We optimize the kernel-target alignment over the distribution μ instead of the radial kernels. In particular,

$$\sup_{\mu \in \mathcal{P}} \frac{2}{n(n-1)} \sum_{1 \le i < j \le n} y_i y_j \int_0^\infty e^{-\xi \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2} \mu(\mathrm{d}\xi),$$

  where $(\boldsymbol{x}_i, y_i)_{1 \le i \le n} \sim P_{\boldsymbol{x},y}$ are labeled data.

- **Questions**:
  - How do we obtain labeled data $(y_i, \boldsymbol{x}_i)_{1 \le i \le n}$ in unsupervised tasks such as the image retrieval?
  - How do we optimize a distribution?

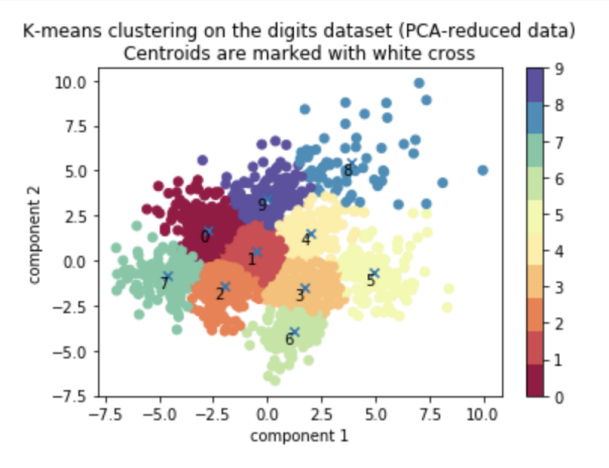
k-mean clustering for generating labeled data in unsupervised tasks

- We perform k-mean clustering on a small *subset* of the data-base.

$$\arg \min_{\mathcal{S}_1, \cdots, \mathcal{S}_k} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2,$$

  where $\boldsymbol{\mu}_\ell$ is the mean of points in the cluster $\mathcal{S}_\ell$.
- Choose an index $\ell \in \{1, 2, \cdots, k\}$, and assign the class labels

$$y_i = \begin{cases} +1 & \boldsymbol{x}_i \in \mathcal{S}_\ell \\ -1 & \boldsymbol{x}_i \notin \mathcal{S}_\ell. \end{cases} \tag{4}$$



A novel algorithm for the optimization of the radial kernels

- The main idea is to optimize the samples (particles) from the distribution $\boldsymbol{\xi} = (\xi^1, \cdots, \xi^N) \sim_{\text{i.i.d.}} \mu$.
- Let $\boldsymbol{z}_m = (y_m, \boldsymbol{x}_m)$ and $\boldsymbol{z}_m = (\tilde{y}_m, \tilde{\boldsymbol{x}}_m)$ denote two samples randomly drawn from the training data-set.
- We update the position of the particles using projected Langevin dynamics

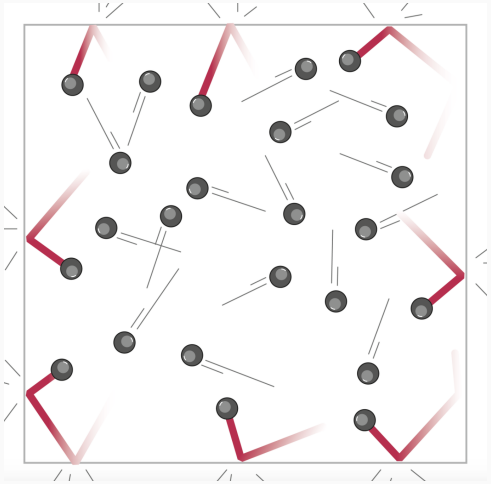
$$\boldsymbol{\xi}_m = P_{\Xi^N}\left(\boldsymbol{\xi}_{m-1} - \eta \nabla \widehat{J}^N(\boldsymbol{\xi}_{m-1}; \boldsymbol{z}_m, \tilde{\boldsymbol{z}}_m) + \sqrt{\frac{2\eta}{\beta}}\boldsymbol{\zeta}_m\right), \tag{5}$$

- $\boldsymbol{\xi}_m \sim \mathrm{N}(0, \boldsymbol{I}_{N\times N})$ is isotropic Gaussian noise, β is the inverse temperature, and $\widehat{J}^N(\boldsymbol{\xi}_{m-1}; \boldsymbol{z}_m, \tilde{\boldsymbol{z}}_m)$ is a cost function whose k-th partial derivative is defined as follows

$$\frac{\partial \widehat{J}^N_{\varepsilon,h}(\boldsymbol{\xi}_m; \boldsymbol{z}_m, \tilde{\boldsymbol{z}}_m)}{\partial \xi^k_m} \stackrel{\text{def}}{=} \frac{\partial E_\gamma(\boldsymbol{\xi}_m; \boldsymbol{z}_m, \tilde{\boldsymbol{z}}_m)}{\partial \xi^k_m} + \frac{h}{2}\frac{\partial W^2_{\varepsilon,2}(\widehat{\mu}^N_m, \widehat{\nu}^N)}{\partial \xi^k_m}. \tag{6}$$

A novel algorithm for the optimization of the radial kernels

- The intuition for this algorithm is from **the kinetic theory of non-ideal gases** in statistical physics.
- The gas has the inverse temperature of $T = \frac{1}{\beta}$, and the particles are interactive.
- Upon hitting the boundaries (wall), the confined particles are reflected **elastically**.



Histogram of the particles

- We can plot the histogram of the particles at each given iteration of the algorithm

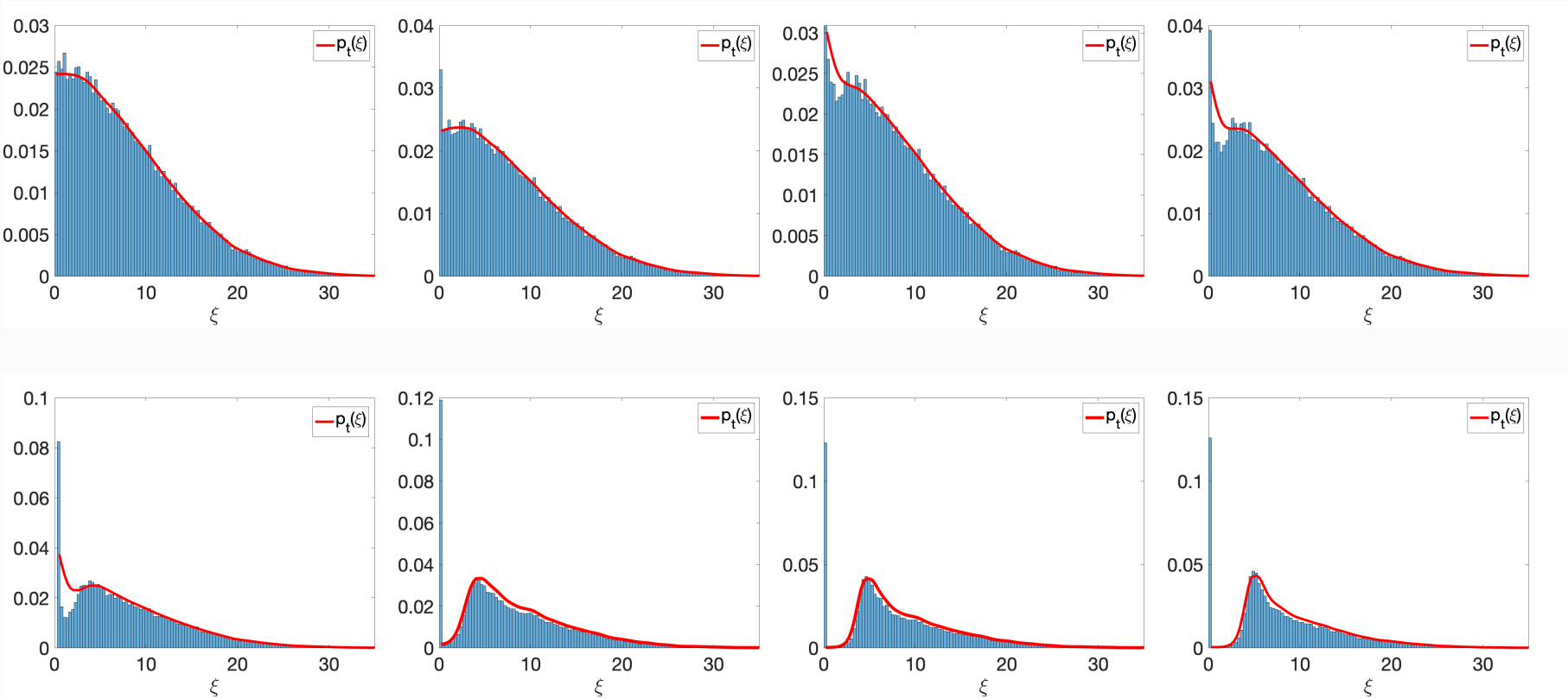


Figure 12: The evolution of the histogram of Langevin particles $\xi^1, \cdots, \xi^N \in \Xi = \mathbb{R}_+$

Mean-Field PDE

- Define the continuous-time embedding of the empirical measure $(\widehat{\mu}_k^N)_{k\in\mathbb{N}}$ associated with the projected Langevin particles in Eq. (**??**), *i.e.*,

$$\mu_t^N(\xi) \overset{\text{def}}{=} \widehat{\mu}_{\lfloor \frac{t}{\eta} \rfloor}^N = \frac{1}{N}\sum_{k=1}^{N} \delta(\xi - \xi_{\lfloor \frac{t}{\eta} \rfloor}^k), \quad 0 \le t \le T, \tag{7}$$

- Suppose the step size $\eta = \eta_N$ satisfies $\eta_N \to 0$, $N/\log(\eta_N/N) \to \infty$, and $\eta_N/\log(\eta_N/N) \to 0$ as $N \to \infty$.
- Furthermore, suppose the Lebesgue density of the initial particles $q_0(\xi) = \mathrm{d}\mu_0/\mathrm{d}\xi$ exists.
- Then, for any fixed $t \in [0, T]$, $\mu_t^N \overset{\text{weakly}}{\to} \mu_t$ as $N \to \infty$.

- The Lebesgue density of the limiting measure $p_t^*(\xi) = \mathrm{d}\mu_t^*/\mathrm{d}\xi$ is a solution to the following distributional dynamics with Robin boundary conditions as well as an initial condition

$$\frac{\partial p_t(\xi)}{\partial t} = \frac{\partial}{\partial \xi}\left(p_t(\xi)\frac{\partial}{\partial \xi}J(\xi, p_t(\xi))\right) + \frac{1}{\beta}\frac{\partial^2}{\partial \xi^2}\left(p_t(\xi)\right),$$

$$\frac{\partial p_t(\xi)}{\partial \xi} + \beta p_t(\xi)\frac{\partial}{\partial \xi}J(\xi, p_t(\xi))\Big|_{\xi=\xi_l} = 0, \quad \forall t \in [0, T],$$

$$\frac{\partial p_t(\xi)}{\partial \xi} + \beta p_t(\xi)\frac{\partial}{\partial \xi}J(\xi, p_t(\xi))\Big|_{\xi=\xi_u} = 0, \quad \forall t \in [0, T],$$

$$p_0(\xi) = q_0(\xi), \quad \forall \xi \in [\xi_l, \xi_u]$$

$$p_t(\xi) \geq 0, \quad \forall \xi \in [\xi_l, \xi_u], \quad \int_{\Xi} p_t(\xi)\mathrm{d}\xi = 1, \quad \forall t \in [0, T],$$

where the functional $J(\xi, p_t(\xi))$ is defined

$$J(\xi, p_t(\xi)) = \mathbb{E}\Big[y\hat{y}\exp(-\xi\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2)\Big] + \int_0^\infty \mathbb{E}\Big[\exp(-(\xi + \xi')\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2)\Big]p_t(\xi')\mathrm{d}\xi'.$$

Simulations on MNIST Data-Set

- Another way to look at the performance is via the Precision-Recall curve

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}.$$

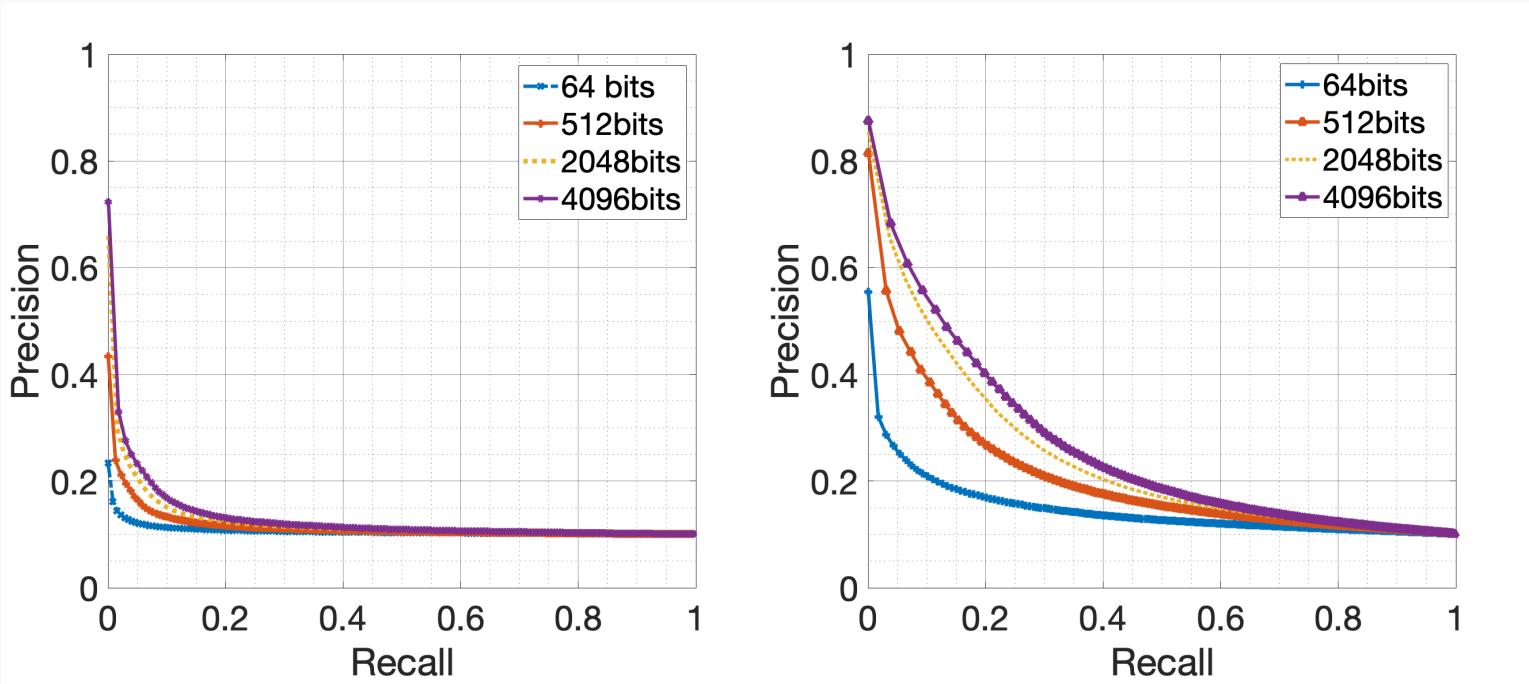
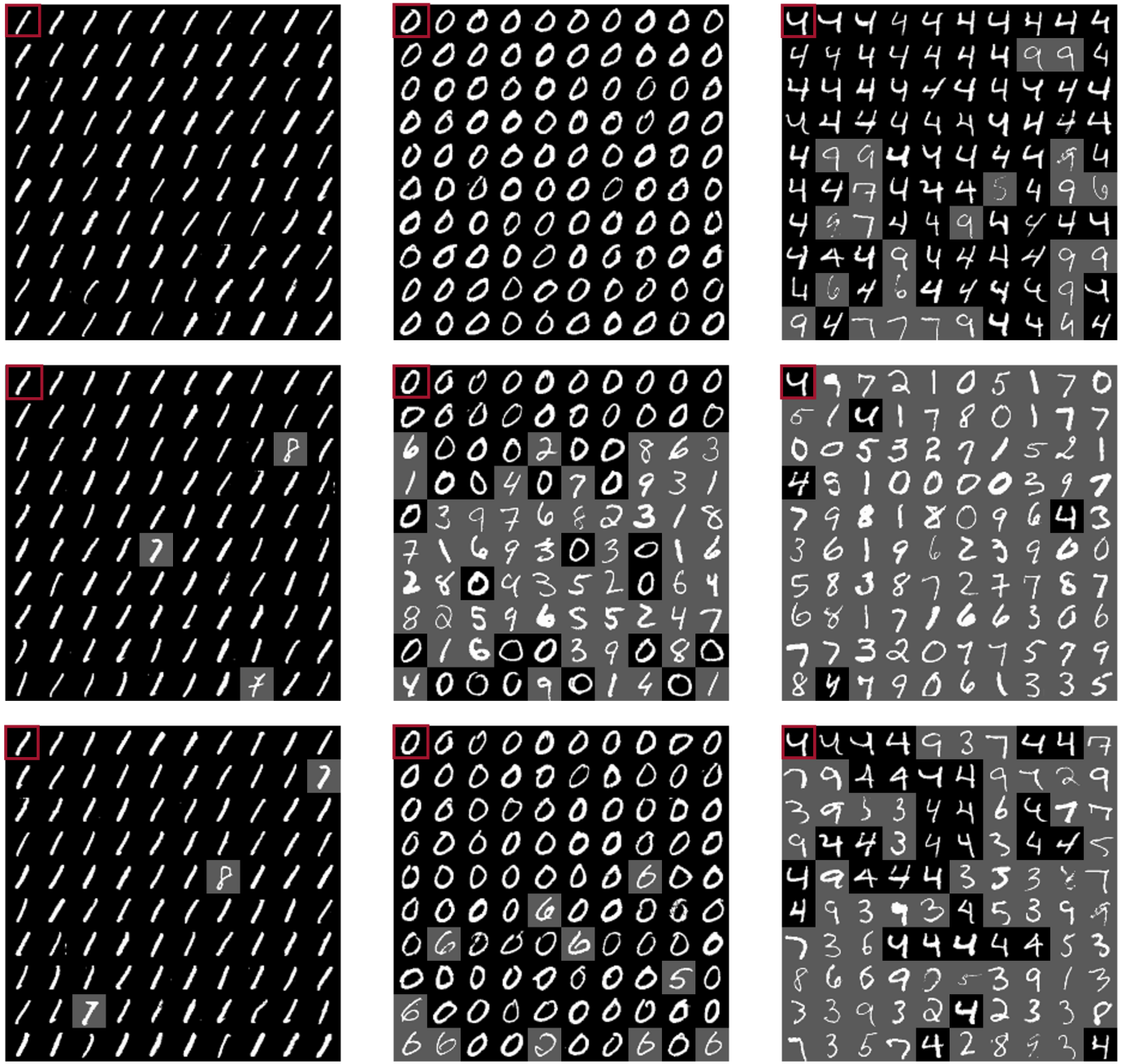


Figure 13: The precision-recall curve for different lengths of the hash codeword.

Simulations on MNIST Data-Set with 4096 bits Hash codes



Simulations on MNIST Data-Set

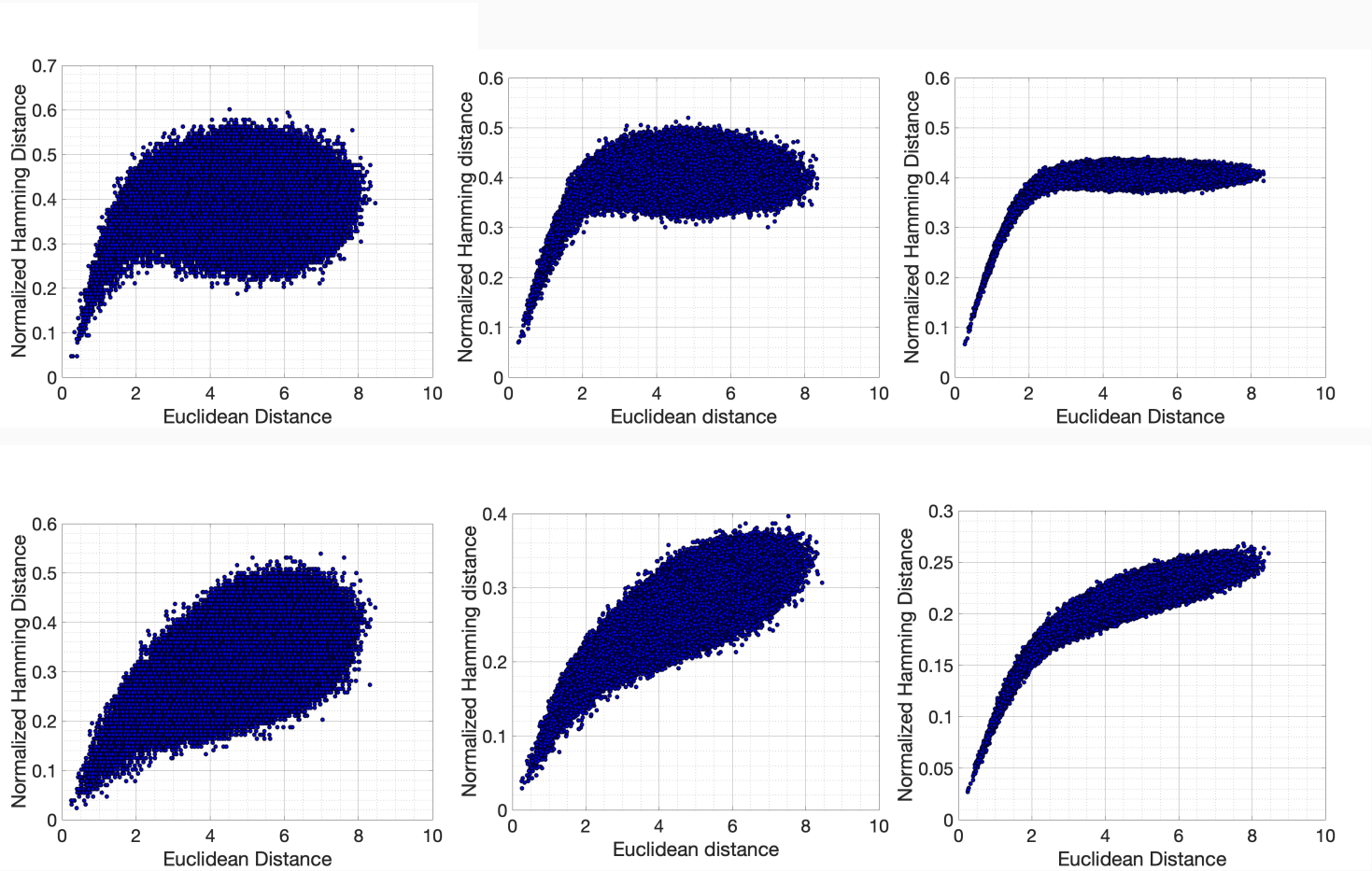


Figure 14: The scatter plots of normalized Hamming distance versus Euclidean distance, where the hash functions are generated with the random feature using untrained Gaussian kernel with the bandwidth parameter of $\gamma = 1$ (top row), and trained kernel with our algorithm in conjunction with a k-mean clustering (bottom row). Panel (a): 128 bits, Panel (b): 512 bits, Panel (c): 4096 bits.

Can we generalize hash function to other alphabets?

- **We propose** the following hash function

$$h_{t,\boldsymbol{w}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \lceil q(\langle \boldsymbol{w}, \varphi_{N_0}(\boldsymbol{x}) \rangle + t) \rceil \bmod q,$$

  where $\boldsymbol{w} \sim \mathrm{N}(0, \boldsymbol{I}_{N_0 \times N_0})$, $t \sim \mathrm{Uniform}[0, 1]$, and

$$\varphi_{N_0}(\boldsymbol{x}) \stackrel{\text{def}}{=} (\cos(\langle \boldsymbol{\omega}_k, \boldsymbol{x} \rangle + b_k))_{1 \le k \le N_0},$$

- To compute the distance between two hash codes, we leverage the Lee distance between two code-words $\boldsymbol{x} = (x_1, \cdots, x_n), \boldsymbol{y} = (y_1, \cdots, y_n) \in \mathbb{Z}_q^n$ of length n as follows

$$\begin{aligned} d_{\mathrm{Lee}}(\boldsymbol{x}, \boldsymbol{y}) &\stackrel{\text{def}}{=} \sum_{i=1}^{n} \min\{(x_i - y_i) \bmod q, (y_i - x_i) \bmod q\} \\ &= \sum_{i=1}^{n} \min\{|y_i - x_i|, q - |y_i - x_i|\}. \end{aligned}$$

  In the special case of $q = 2$ and $q = 3$, the Lee distance corresponds to the Hamming distance.

Histograms of Lee distance vs Euclidean distance

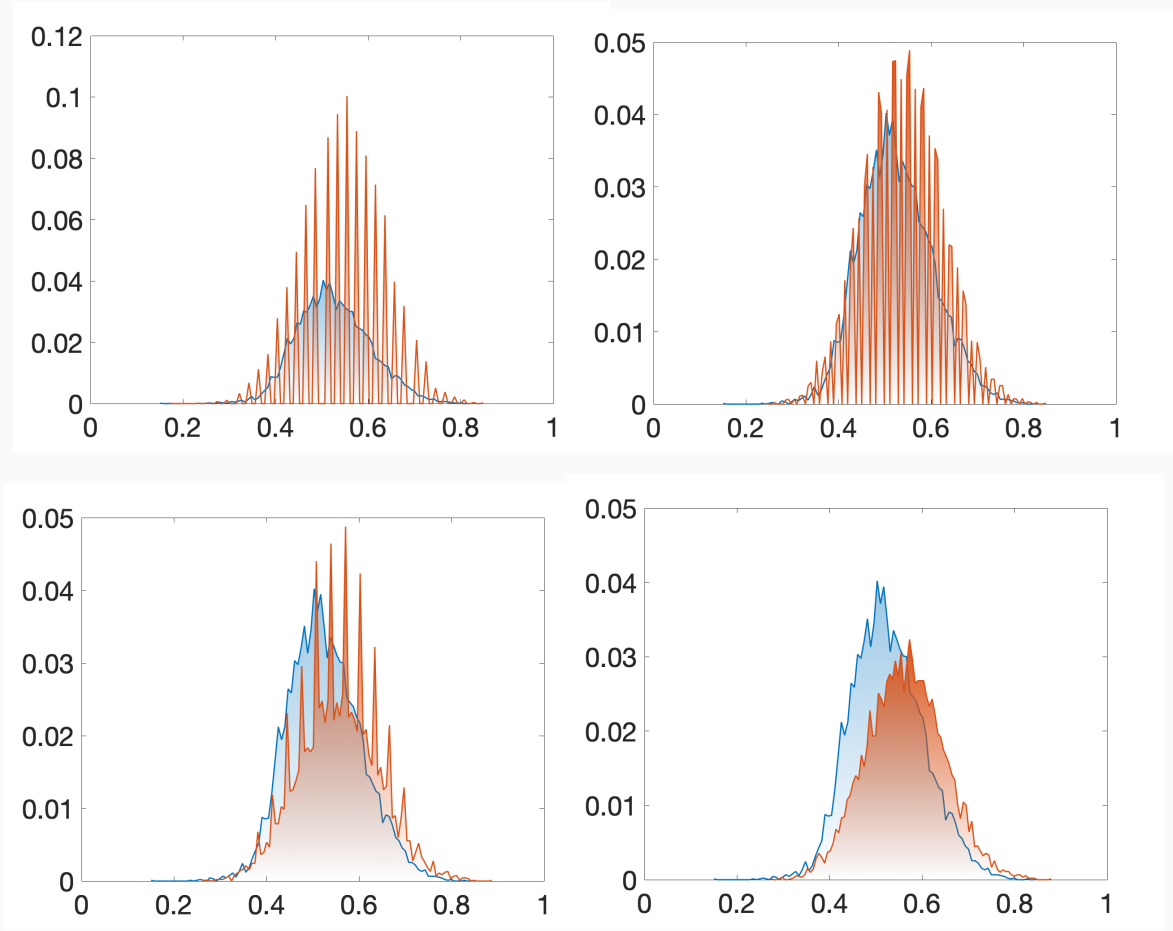


Figure 15: The (normalized) histogram of the (rescaled) Lee distance (red color) and the (rescaled) Euclidean distance (blue color) of a sample query from each point of the MNIST data-base for different values of q, Panel (a): $q = 2$, Panel (b): $q = 20$, Panel (c): $q = 50$. Increasing q in the Lee distance yields a better approximation for the Euclidean distance.

Simulations on MNIST Data-Set

- Another way to look at the performance is via the Precision-Recall curve

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}.$$

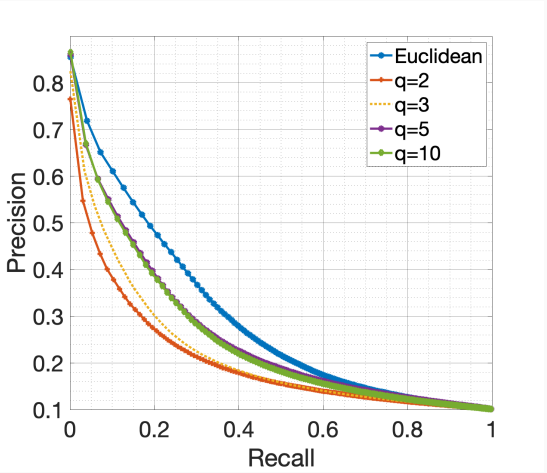
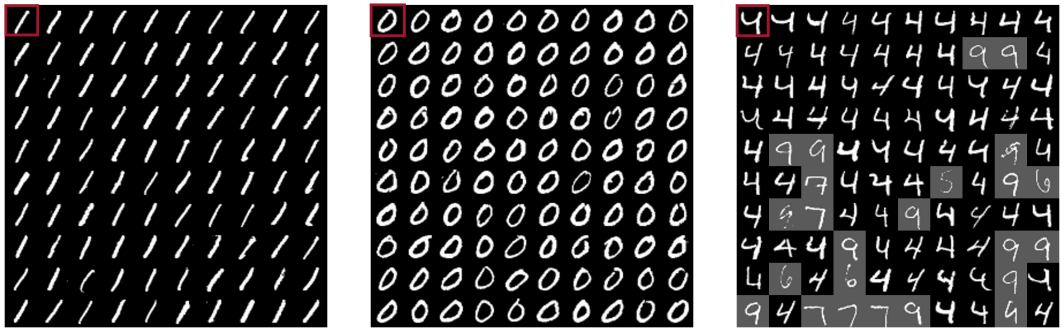
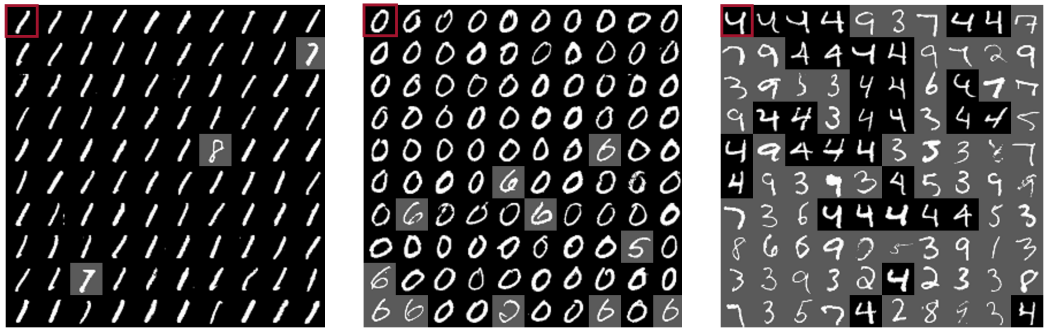


Figure 16: The precision-recall curve for different lengths of the hash codeword.

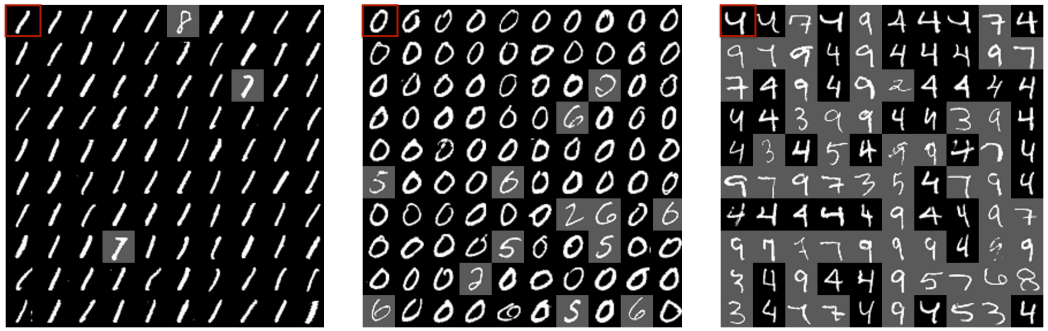
Simulations on MNIST Data-Set



(Euclidean distance)



(4096 Hamming Z_2^n codes)



(1024 Hamming Z_{10}^n codes)

Application to Classification Tasks

- The proposed method for optimizing RBF kernels can also be used for other machine learning tasks such as classification using kernel SVMs.
- Given the training samples $(y_i, \boldsymbol{x}_i)_{1 \leq i \leq n}$, the kernel SVMs solve the following optimization problem

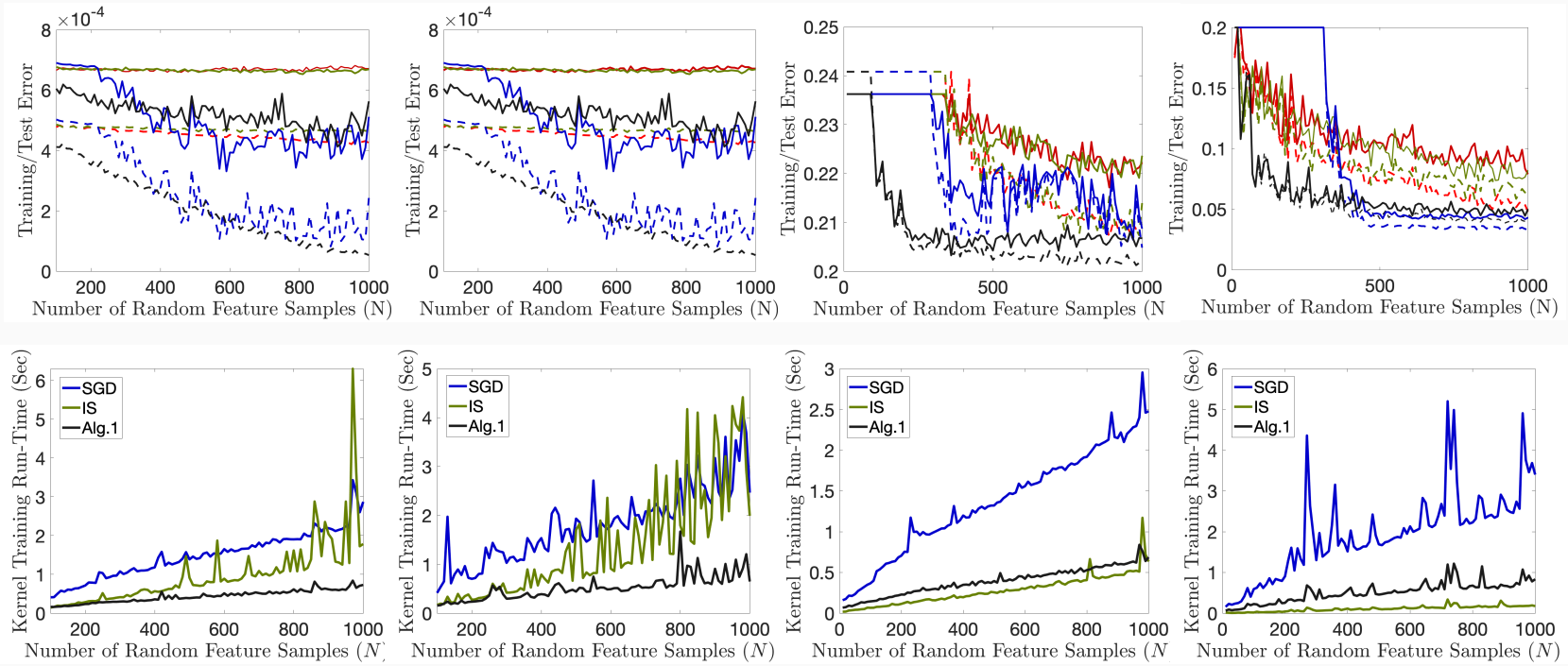
$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^D} \frac{1}{n} \sum_{i=1}^{n} \max \left\{ 0, 1 - \sum_{k=1}^{D} y_i \alpha_i \varphi(\boldsymbol{x}_i; \boldsymbol{\omega}_k) \right\} + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|_2^2.$$

- Here, $\varphi(\boldsymbol{x}_i; \boldsymbol{\omega}_k)$ is the *random feature* associated with a kernel in Rahimi and Recht integral representation of the translation invariant kernels

$$K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \int_{\Omega} \varphi(\boldsymbol{x}; \boldsymbol{\omega}) \varphi(\tilde{\boldsymbol{x}}; \boldsymbol{\omega}) \nu(\mathrm{d}\boldsymbol{\omega}), \tag{9}$$

where $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \phi(\boldsymbol{x} - \tilde{\boldsymbol{x}})$, and ν is the probability distribution of random variable $\boldsymbol{\omega}$.

Application to Classification Tasks



(a) (b) (c) (d)

Figure 17: The training and test errors (first row) and the run-times (second row) of kernel optimization algorithms using proposed method, the SGD optimization, and Importance Sampling (IS). Panel (a): BUZZ, Panel (b): ONLINE NEWS POPULARITY, Panel (c): ADULT, Panel (d): SEIZURE

Application to Classification Tasks

- Many ideas and techniques in statistical physics are relevant to machine learning problems.
- The proposed method for optimizing RBF kernels in our work is based on idea of non-ideal gases.
- Currently, we are trying to apply the methods we developed to medical imaging and the isodose distribution for radiation therapy treatment planning.