Hash-Based Feature Learning for Incomplete Continuous-Valued Data

Shuai Yuan* Pang-Ning Tan* Kendra S. Cheruvelil† C. Emi Fergus‡ Nicholas K. Skaff‡ Patricia A. Soranno‡

Abstract

Hash-based feature learning is a widely-used data mining approach for dimensionality reduction and for building linear models that are comparable in performance to their nonlinear counterpart. Unfortunately, such an approach is inapplicable to many real-world data sets because they are often riddled with missing values. Substantial data preprocessing is therefore needed to impute the missing values before the hash-based features can be derived. Biases can be introduced during this preprocessing because it is performed independently of the subsequent modeling task, which can result in the models constructed from the imputed hash-based features being suboptimal. To overcome this limitation, we present a novel framework called `H-FLIP` that simultaneously estimates the missing values while constructing a set of nonlinear hash-based features from the incomplete data. The effectiveness of the framework is demonstrated through experiments using both synthetic and real-world data sets.

1 Introduction

Real world data sets are often noisy, making it difficult to develop accurate prediction models from the data. In addition, the data are often high-dimensional and may contain redundant or correlated features, as well as missing values, which makes it crucial to derive a good set of features to represent the data. This has led to considerable interest in developing feature selection [1] and feature learning [2] methods to overcome the limitations of using the original features of the data.

Hashing is a popular feature learning technique for transforming high dimensional data into an alternative representation that preserves the similarities between instances in the original data [3]. There are two main advantages of using hashing for feature learning. First, it provides an effective dimensionality reduction technique, especially for applications such as large-scale image and multimedia retrieval problems [4]. Second, the similarity-preserving property of the hash functions enables the hash-based features to be used as an approximation to the features defined in the Reproducing Kernel Hilbert Space (RKHS). This allows us to construct linear models on the hash-based features with comparable accuracy as their nonlinear counterpart, but with substantial reduction in runtime complexity [5].

Hash-based feature learning methods can be generally classified into two categories, depending on how the features are created [3]. The first category corresponds to data-independent methods, which create the features by applying randomly generated hash functions to the data. This includes the min-hash [6], random hyperplane-based hashing [7], and shift-invariant kernel hashing [8] methods. One potential limitation of using data-independent methods is that the number of hash-based features needed to provide a good representation of the data can be very large since the hash functions are generated randomly. Thus, data-driven methods have been developed as an alternative to such methods as they can fit the salient properties of the data using a small set of hash functions. Methods that belong to this second category include spectral hashing [9] and minimal loss hashing [10].

Existing hash-based feature learning methods assume that the input data are complete. Any missing values present in the data must be imputed before the hash functions can be derived. Because the imputation is typically performed during preprocessing, the hash-based features created after the preprocessing may not be optimal for the subsequent modeling task. As an illustration, consider the example shown in Figure 1. Suppose P denotes a data point that has a missing value for its predictor variable (the x-axis), but the value of its response variable (the y-axis) is known. After applying mean imputation, the data point P is shifted to its estimated point P', which is much larger than its true value. This process introduces a bias into the data set, resulting in a new regression model (represented by a solid line in the diagram) that deviates from its true model (represented by a dashed line). At first glance, the bias may seem quite small. However, it may become

*Dept of Comp Science and Engr, Michigan State University
†Dept of Fisheries and Wildlife & Lyman Briggs College, Michigan State University
‡Dept of Fisheries and Wildlife, Michigan State University

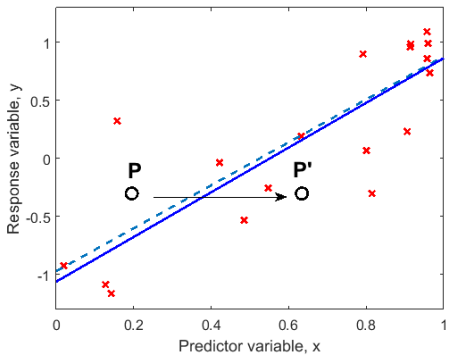


Figure 1: Effect of mean value imputation on regression.

an issue with more missing values in the data set. If P is imputed in a way that takes into account the effect of the imputation on the response variable, the regression model would be less affected by biases due to the imputation. This is the key motivation behind our proposed framework called `H-FLIP`, which combines missing value imputation with hash-based feature learning in a way that preserves the relationship between the hash-based features and the response variables of the data.

Many existing hash-based feature learning methods are designed for discrete-valued data, such as those encountered in image classification problems [4]. In this paper, we investigate the application of hash-based feature learning for regression problems, in which the hash-based features are continuous rather than binary. In addition, most of the existing methods create the hash-based features as a linear combination of the original features. These methods are ineffective for more complex prediction tasks, where the relationship between the response and predictor variables is often non-linear. Previous work has shown that the inner product of random Fourier features provides a good approximation to the shift-invariant kernels used for building nonlinear models [5]. However, a relatively large number of such features is needed to provide a succinct representation of the data since the Fourier features are generated randomly. To overcome this limitation, we train a set of hash functions to fit the response variable using random Fourier features. This enables our framework to model nonlinear relationships with a small set of supervised hash functions. Our framework can thus be regarded as a hybrid method that embeds a data-independent approach, i.e., random Fourier features, into a supervised learning setting.

In summary, the main contributions of this paper are as follows:

- We developed a supervised feature learning framework for regression problems using nonlinear, random Fourier features as its basis functions.
- We extended this framework to handle incomplete data by enabling the missing value imputation and hash-based feature learning to be performed jointly.
- We demonstrated the efficacy of our approach through extensive experiments using both synthetic and real-world data sets. Our empirical results suggested that the framework is more effective than the approach of imputing the missing values before applying feature learning in 7 out of 9 real-world data sets evaluated in this study.

2 Related Work

Feature learning [11] is the task of extracting an alternative feature representation of a given data set. Classical methods such as principal component analysis were developed to create features that preserve variability in the original data. These methods are mostly unsupervised, and thus, provide no guarantee about the effectiveness of their extracted features for prediction problems. More recently, methods such as deep learning [12, 13] have been proposed to extract hierarchical features from data. Although such methods have been successfully applied to applications such as image classification [12], they are expensive to train and require considerable human effort to design the right architecture for a given prediction problem.

Hashing is another feature learning technique that has attracted considerable attention in recent years. The goal of hash-based feature learning is to transform the data into easily computable features that preserve the underlying properties of the data. For example, the Min-hash method [6] was designed to create features that preserve the Jaccard similarity between instances. Hash-based methods have also been developed for other similarity measures [7, 9, 10]. However, they were mostly designed to represent complete data with binary hash codes unlike the framework proposed in this paper.

3 Preliminaries

This section reviews some of the fundamental concepts underlying the framework proposed in this paper.

3.1 Support Vector Regression (SVR) Support vector regression is a widely-used method to solve large scale prediction problems. The goal of SVR is to learn a linear function f that fits the training set $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ by solving the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_i (\xi_i + \xi_i^*) \\ \text{s.t.} \quad & -\epsilon - \xi_i^* \leq y_i - \mathbf{w}^T \mathbf{x}_i - b \leq \epsilon + \xi_i, \end{aligned}$$

Table 1: Comparison among the various regression methods for modeling lake water quality in terms of their mean square prediction error (MSE).

Method\Response variable	TP	TN	Chla	Secchi
Multiple linear regression	1.39	2.39	1.31	0.85
Ridge regression	0.78	0.72	**0.71**	0.59
Lasso regression	0.79	0.74	0.72	0.58
Linear SVR	0.78	0.71	**0.71**	0.58
Nonlinear SVR (RBF)	**0.76**	**0.69**	**0.71**	**0.56**

where ξ_i and ξ_i^* are the slack variables that can be used to relax the error bounds on the training instances while C controls the trade-off between minimizing the training error and the magnitude of $\mathbf{w}$.

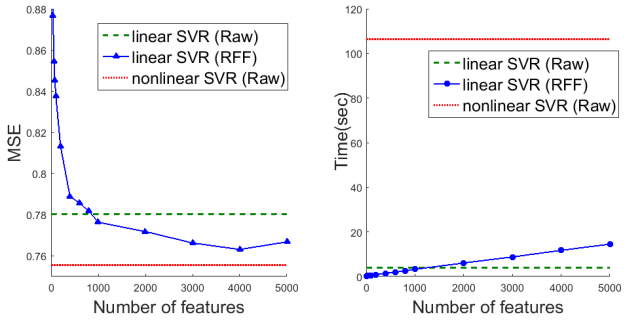
The linear SVR formulation can be extended to a nonlinear setting by projecting the original features $\mathbf{x}$ to a higher dimensional feature space, $\Phi(\mathbf{x})$, such that the inner product between instances in the new, projected space is equivalent to computing their similarity in the original space using a nonlinear kernel function. For example, the Gaussian radial basis function (RBF), $k(\mathbf{x}_i, \mathbf{x}_j) = \exp[-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}]$ is a popular choice for nonlinear SVR. The implicit mapping to a high-dimensional feature space facilitated by the kernel function enables SVR to capture non-linear dependencies in the data.

To illustrate the advantages of using SVR, we have compared its performance to other regression methods on four lake water quality data sets obtained from the LAGOS database [14]. Details about the data sets can be found in Section 5. Table 1 shows the mean square error (MSE) obtained using 10-fold cross validation. Due to the noise present in the data, methods such as multiple linear regression may overfit the training data resulting in its high error rate. The performance of linear SVR is either comparable to or better than the results obtained using lasso and ridge regression. However, the best results are obtained using nonlinear SVR with Gaussian RBF kernel.

3.2 Random Fourier Features (RFF) Despite its superior performance in terms of modeling complex relationships in data, the nonlinear SVR method scales poorly with increasing data set size. This is because the method requires considerable computational resources to compute and store the kernel matrices. To overcome this limitation, Rahimi and Recht [5] proposed a mapping function known as random Fourier feature:

$$\phi(\mathbf{x}) = \sqrt{2/p}\cos(\mathbf{R}\mathbf{x} + \mathbf{t}) \tag{3.1}$$

where $\mathbf{R} \in \Re^{p \times d}$ is a random matrix drawn from a standard normal distribution while $\mathbf{t} \in \Re^p$ is a random vector drawn from a uniform distribution between $[0, 2\pi]$. Previous research has shown that the RFF provides an unbiased estimate of the RBF kernel in the original feature space [5]. Thus, instead of applying nonlinear SVR with an RBF kernel, comparable performance can be achieved by training a linear SVR on the RFF.



(a) Accuracy (b) Run time

Figure 2: Average runtime and accuracy comparison for linear SVR with raw features, nonlinear SVR with raw features, and linear SVR with RFF.

To illustrate this, Figure 2 compares the average runtime and accuracy for the following three approaches: (i) linear SVR trained on the original features, (ii) nonlinear SVR trained on the original features, and (iii) linear SVR trained on the random Fourier features. The experiment was performed on the lake water quality data set with total phosphorous (TP) as the response variable. For linear SVR with RFF, we vary the number of random Fourier features from 1 to 5000, each repeated 10 times. The average MSE and runtime using 10-fold cross validation are shown in Figure 2. The results suggest that the MSE of SVR decreases with increasing number of RFF, approaching the results of nonlinear SVR. Although its runtime increases with larger number of features, it is still significantly lower than the runtime for nonlinear SVR. This result shows the advantage of using RFF as hash-based features for training a linear SVR with comparable accuracy as nonlinear SVR. Nevertheless, the number of hash-based features needed is still large (in the order of several thousands) to produce an error rate that is comparable to nonlinear SVR.

3.3 Matrix Completion Matrix completion is an approach for recovering missing values by assuming that the original data is a low rank matrix. Let $\mathbf{A} \in \Re^{N \times d}$ be the original data matrix with incomplete entries and $\mathcal{P} : \mathcal{R}^{N \times d} \rightarrow \mathcal{R}^p$ be a linear map that identifies indexes of the non-missing entries in the matrix. The matrix completion approach is often cast into the following optimization problem [15, 16]:

$$\min_{\mathbf{X}} \ \frac{1}{2} \parallel \mathcal{P}(\mathbf{X}) - \mathcal{P}(\mathbf{A}) \parallel_2^2 + \mu \parallel \mathbf{X} \parallel_* \tag{3.2}$$

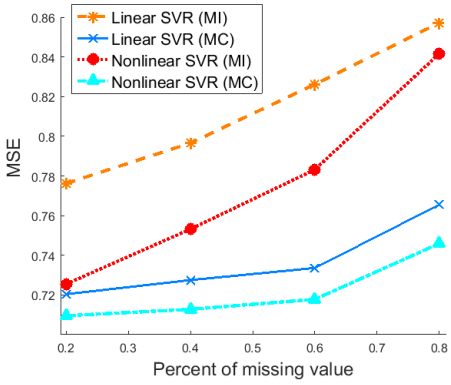


Figure 3: Comparing linear and nonlinear SVR with mean imputation (MI) and matrix completion (MC).

where $\|\cdot\|_*$ denote the trace-norm of a matrix, which is the sum of its singular values. Intuitively, the preceding objective function seeks to learn a "complete" matrix $\mathbf{X}$ of minimal rank that is consistent with the non-missing entries of the original data $\mathbf{A}$. The regularization parameter μ controls the trade-off between maintaining the consistency of the non-missing entries and minimizing the rank of the recovered matrix.

To demonstrate the effectiveness of this method, we performed an experiment on the lake water quality data set using total chlorophyll-a (chla) as the response variable. We introduced missing values randomly into the data set, varying the percentage of missing values from 20% to 80%. We then applied both mean imputation and matrix completion to the altered data followed by linear and nonlinear SVR. The results shown in Figure 3 suggested that matrix completion enabled the missing values to be recovered at a higher precision compared to the mean imputation method. These results hold true for all percentages of missing values introduced and for both linear and nonlinear SVR.

4 Proposed Framework

This section describes the detailed formulation of our proposed hash-based feature learning framework. The unique characteristics of our framework are as follows:

1. It uses a set of sparse random Fourier features as its basis function for creating the hash-based features. The RFF enables the framework to capture nonlinear relationships in the data.

2. It applies supervised learning to identify the best combination of RFF that fits the response variable. Our framework is thus a hybrid method that combines data-independent with data-driven methods.

3. It uses trace-norm regularization to deal with missing values present in the data. Our framework would simultaneously estimate the missing values while learning the hash-based features.

The resulting framework is called **H-FLIP**, which stands for **H**ash-based **F**eature **L**earning for **I**ncom**P**lete data. Although RFF has been used to approximate shift-invariant kernels [5], it has not been used for supervised feature learning. Thus, we first present our supervised hash-based feature learning framework in Section 4.1. The framework is extended to incomplete data in Section 4.2.

4.1 Supervised hash-based feature learning using RFF

Consider a data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where each $\mathbf{x}_i \in \Re^d$ denote a set of values for the predictor variables, y_i is the corresponding value for the response variable, and N is the number of observations. Our goal is to learn a set of hash functions $\mathbf{H} = \{h_k\}_{k=1}^K$ where each function $h_k : \Re^d \to \Re$ transforms the original data in d-dimensional feature space to a 1-dimensional feature space. Instead of using conventional linear hash functions, we employ RFF as our basis function in order to capture nonlinear relationships in the data. Formally, the k^{th} hash function is defined as

$$h_k(\mathbf{x}_i) = \mathbf{w}_k^T \phi_k(\mathbf{x_i}), \tag{4.3}$$

where each basis function $\phi_k(\cdot) : \Re^d \to \Re^p$ corresponds to a p-dimensional RFF defined in Equation (3.1). The parameters of the hash functions are trained to fit the training data $\mathcal{D}$ using a supervised learning algorithm.

Transforming the original data $\mathcal{D}$ to K RFF requires $\mathcal{O}(NKdp)$ operations, which is expensive when the number of features in the original (d) and projected (p) feature space are large. To reduce the computations, instead of using all d features, each hash function ϕ_k is generated by randomly selecting a subset of the d features and applying the nonlinear transformation given in Equation (3.1) to the selected features. The creation of sparse RFF is illustrated by the following example.

EXAMPLE 1. *Let $\{x_1, x_2, x_3, x_4\}$ be the set of predictor variables associated with the data instance $\mathbf{x}$. Consider the following RFF, $\phi(\mathbf{x}) = \cos(\mathbf{Rx} + \mathbf{t})$, where*

$$\mathbf{R} = \begin{bmatrix} 0.3 & -0.5 & 0 & 0 \\ -0.1 & 0.7 & 0 & 0 \\ 0.2 & 0.6 & 0 & 0 \end{bmatrix}, \ \mathbf{t} = \begin{bmatrix} 1.2 \\ 2.8 \\ 0.66 \end{bmatrix}.$$

This is a sparse RFF as it depends on x_1 and x_2 only.

The sparse RFF forms the basis function of our supervised hash-based features defined in Equation (4.3).

The weights of the hash functions are estimated by optimizing the following objective function:

$$J(\mathbf{W}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{N}(h_k(\mathbf{x}_i) - y_i)^2 + \lambda\sum_{k=1}^{K} \parallel \mathbf{w}_k \parallel_2^2$$

$$= \frac{1}{2}\sum_{k=1}^{K}\left\|\Phi_k(\mathbf{X})\mathbf{w}_k - \mathbf{y}\right\|_2^2 + \lambda \parallel \mathbf{W} \parallel_F^2, \tag{4.4}$$

where $\mathbf{W} \in \Re^{p\times K} = [\mathbf{w}_1\ \mathbf{w}_2 \cdots \mathbf{w}_K]$ denote the weight matrix associated with the supervised hash functions and $\Phi_k(\mathbf{X}) \in \Re^{N\times p}$ is the RFF representation of the input data matrix. As the K hash functions can be decoupled from one another in the formulation given in Equation (4.4), the parameter vector $\mathbf{w}_k$ of each hash function can be solved independently as follows.

$$\mathbf{w}_k = \arg\min_{\mathbf{v}} \frac{1}{2} \parallel \Phi_k(\mathbf{X})\mathbf{v} - \mathbf{y} \parallel_2^2 + \lambda \parallel \mathbf{v} \parallel_2^2 \tag{4.5}$$

The closed form solution for $\mathbf{w}_k$ is given by:

$$\mathbf{w}_k = \left[\Phi_k(\mathbf{X})^T\Phi_k(\mathbf{X}) + \lambda\mathbf{I}\right]^{-1}\Phi_k(\mathbf{X})^T\mathbf{y}$$

where $\mathbf{I}$ denotes the identity matrix.

4.2 Hash-based feature learning for incomplete data (H-FLIP) We now extend the previous formulation to data with missing values. Let $\mathbf{A}$ be the incomplete data matrix and $\mathbf{X}$ be the imputed data matrix. Furthermore, let $\mathbf{X} = [\mathbf{X}_l; \mathbf{X}_u]$, where $\mathbf{X}_l$ is the set of training instances whose response variable values are known and $\mathbf{X}_u$ is the set of test instances whose response variable values are unknown.

H-FLIP is designed to simultaneously learn the imputed matrix $\mathbf{X}$ and the weight matrix $\mathbf{W}$ of the sparse RFF by minimizing the following objective function:

$$\min_{\mathbf{W},\mathbf{X}} F(\mathbf{X}, \mathbf{W}) = F_1(\mathbf{X}) + \alpha F_2(\mathbf{X}_l, \mathbf{W}) \tag{4.6}$$

where

$$F_1(\mathbf{X}) = \frac{1}{2} \parallel \mathcal{P}(\mathbf{X}) - \mathcal{P}(\mathbf{A}) \parallel_F^2 + \mu \parallel \mathbf{X} \parallel_*$$

$$F_2(\mathbf{X}, \mathbf{W}) = \frac{1}{2}\sum_{k=1}^{K} \parallel \Phi_k(\mathbf{X}_l)\mathbf{w}_k - \mathbf{y}) \parallel_2^2 + \lambda \parallel \mathbf{W} \parallel_F^2,$$

F_1 measures the error in missing value imputation while F_2 measures the prediction error of using the hash function to fit the response variable y. The regularization parameter α controls the trade-off between minimizing both factors. A trace-norm regularization is applied to ensure that the recovered matrix $\mathbf{X}$ has a low rank. As the missing value imputation must be performed on the entire data set, F_1 involves instances from both the training and test sets while F_2 involves only instances from the training set.

Algorithm 1 H-FLIP Framework

Input: $\mathbf{A}, \mathbf{y}$,
Output: $\mathbf{X}, \mathbf{W}$
Initialize $\mathbf{X}^{(0)}$ by solving Equation (3.2).
Generate random matrices $\{\mathbf{R}_k\}$ and $\mathbf{T}$.
Create RFF: $\Phi_k(\mathbf{X}_l) = \sqrt{2/p}\cos(\mathbf{X}_l\mathbf{R}_k^T + \mathbf{T})$
Update $\mathbf{W}$ using Equation (4.9).
while stopping condition is not met **do**
  Initialize $\gamma^{(1)} = \gamma^{(0)} = 1$ and $\mathbf{X}^{(1)} = \mathbf{X}^{(0)}$.
  for $k = 1$ **to** maxIter **do**
    $\mathbf{Y}^{(k)} \leftarrow \mathbf{X}^{(k)} + \frac{\gamma^{(k-1)}-1}{\gamma^{(k)}}\left[\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}\right]$
    $\mathbf{Z}^{(k)} \leftarrow \mathbf{Y}^{(k)} - \frac{1}{\tau^{(k)}}\nabla f(\mathbf{Y}^{(k)})$
    $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^T] \leftarrow \text{SVD } (\mathbf{Z}^{(k)})$
    $\mathbf{X}^{(k+1)} \leftarrow \mathbf{U}\mathbf{D}_{\mu/\tau^{(k)}}(\mathbf{\Sigma})\mathbf{V}^T$
    Compute τ^k using line search.
    $\gamma^{(k+1)} \leftarrow \frac{1+\sqrt{1+4(\gamma^{(k)})^2}}{2}$
  end for
  $\mathbf{X}^{(0)} \leftarrow \mathbf{X}^{(k+1)}$.
  Update $\mathbf{W}$ using Equation (4.9)
end while
return $\mathbf{X}, \mathbf{W}$

An alternating minimization scheme is employed to solve the objective function given in Equation (4.6). A pseudocode of the H-FLIP framework is shown in Algorithm 1. The framework alternates between optimizing for $\mathbf{X}$ and $\mathbf{W}$, until the stopping criteria is satisfied. The optimization steps are outlined below.

4.2.1 Updating X When $\mathbf{W}$ is fixed, the objective function can be simplified as follows:

$$F(\mathbf{X}) = \frac{1}{2} \parallel \mathcal{P}(\mathbf{X}) - \mathcal{P}(\mathbf{A}) \parallel_F^2$$

$$+ \frac{\alpha}{2}\sum_{k=1}^{K} \parallel \Phi_k(\mathbf{X}_l)\mathbf{w}_k - \mathbf{y} \parallel_2^2 + \mu \parallel \mathbf{X} \parallel_* \tag{4.7}$$

Since the trace-norm regularization is a non-smooth function, we separate the objective function into a sum of two functions, $F(\mathbf{X}) = f(\mathbf{X}) + g(\mathbf{X})$, where

$$f(\mathbf{X}) = \frac{1}{2} \parallel \mathcal{P}(\mathbf{X}) - \mathcal{P}(\mathbf{A}) \parallel_F^2 + \frac{\alpha}{2}\sum_{k=1}^{K} \parallel \Phi_k\mathbf{w_k} - \mathbf{y} \parallel_2^2$$

$$g(\mathbf{X}) = \mu \parallel \mathbf{X} \parallel_*$$

We apply the accelerated proximal gradient descent method [15] to solve for $\mathbf{X}$. Let $\mathbf{X}^{(k)}$ denote the most

recent estimate after k iterations. The following two steps are performed to update the estimate.

1. Apply accelerated gradient descent method to the smooth part of the objective function.

$$\mathbf{Y}^{(k)} = \mathbf{X}^{(k)} + \frac{\gamma^{(k-1)}-1}{\gamma^{(k)}}\left[\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}\right]$$
$$\mathbf{Z}^{(k)} = \mathbf{Y}^{(k)} - \frac{1}{\tau^{(k)}}\nabla f(\mathbf{Y}^{(k)})$$

   The above equation reduces to the update formula for standard gradient descent when $\forall k : \gamma_k = 1$.

2. Apply singular value shrinkage operator to $\mathbf{Z}^{(k)}$. Let $\mathbf{Z}^{(k)} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{\Sigma} = \text{diag}(\sigma_i)$ is a diagonal matrix containing the singular values of $\mathbf{Z}^{(k)}$ while $\mathbf{U}$ and $\mathbf{V}$ are matrices containing its left and right singular vectors. We update $\mathbf{X}$ as follows:

$$\mathbf{X}^{(k+1)} = \mathbf{U}\mathbf{D}_{\mu/\tau^{(k)}}(\mathbf{\Sigma})\mathbf{V}^T \tag{4.8}$$

   where $\mathbf{D}_\nu(\mathbf{\Sigma})$ is a diagonal matrix whose i-th element is $\max(0, \sigma_i - \nu)$. This is equivalent to applying a threshold $\mu/\tau^{(k)}$ to each singular value of $\mathbf{Z}^{(k)}$.

The step size of the gradient descent is determined dynamically using a line search algorithm [15]. The matrix $\mathbf{X}$ is updated until one of the following stopping conditions is satisfied: (1) the maximum number of iterations is reached, (2) $\|\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}\|_F / \|\mathbf{X}^{(k)}\| < \epsilon$, or (3) the objective function given in Equation (4.7) no longer decreases significantly.

4.2.2 Updating W When $\mathbf{X}$ is fixed, we update $\mathbf{W}$ by minimizing the following objective function:

$$\min_{\mathbf{W}} \frac{1}{2}\sum_{k=1}^{K} \parallel \Phi_k(\mathbf{X}_l)\mathbf{w}_k - \mathbf{y}) \parallel_2^2 + \lambda \parallel \mathbf{W} \parallel_F$$

This is equivalent to solving the objective function for supervised hash-based feature learning as presented in Section 4.1. W can be updated using only instances that belong to the training set in the following way:

$$\mathbf{w}_k = (\Phi_k(\mathbf{X}_l)^T\Phi_k(\mathbf{X}_l) + \lambda\mathbf{I})^{-1}\Phi_k(\mathbf{X}_l)^T\mathbf{y} \tag{4.9}$$

4.3 Predictive modeling on hash based features Once the hash-based features are learned, we can train a linear regression model such as SVR on the data. If there are no missing values in the test data, we can simply extract their random Fourier features and apply the supervised hash functions given in Equation (4.3) to generate the hash-based features. The linear SVR model can then be applied to predict the values of the test data.

Table 2: Summary of data sets.

Response variable	TP	TN	Chla	Secchi
# instances (lakes)	3694	1961	4834	4684
# features	356	356	356	356
Mean	37.5	821.2	20.9	2.6
Std deviation	68.2	729.6	36.9	1.86

(a) Lake water quality data

Data	Housing	Wine	Parkinson	News	Concrete
# instances	506	4898	5875	5000	1030
# features	14	12	26	61	9

(b) UCI machine learning data

If the test data is incomplete, their missing values are simultaneously imputed along with the training data during the training phase of `H-FLIP`. This allows us to apply linear SVR to the random Fourier features extracted from the imputed test data. Nevertheless, one potential limitation of our framework is that it has to be re-trained whenever new data with missing values become available. An incremental version of `H-FLIP` is thus needed to overcome this limitation but this will be a subject for future research.

5 Experimental Evaluation

This section describes the experiments conducted to evaluate the performance of our proposed framework.

5.1 Data Sets We have performed experiments using both synthetic and real-world data sets.

5.1.1 Synthetic data We created a rank-20 data matrix $\mathbf{X}$ containing 5000 instances (rows) and 100 predictor variables (columns) in the following way: $\mathbf{X} = \mathbf{P}\mathbf{Q} + 0.1\mathbf{E}$, where $\mathbf{P} \in \Re^{5000\times 20}$, $\mathbf{Q} \in \Re^{20\times 100}$, and $\mathbf{E} \in \Re^{5000\times 100}$. The entries of the matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{E}$ are generated randomly from a standard normal distribution. Let x_i denote the i-th predictor variable. The value of the response variable y is computed as follows: $y = x_1x_2 + x_{10}x_{11} + x_{12} + \mathcal{N}(0, 0.1)$. All the columns in $\mathbf{X}$ and the vector $\mathbf{y}$ are standardized to have zero mean and unit variance.

5.1.2 Lake water quality data We used several lake water quality data sets from LAGOS [14], which is a geo-spatial database that contains landscape characterization features and lake water quality data measured at multiple scales covering 17 states in the United States. We used four lake water quality variables—total phosphorus (TP), total nitrogen (TN), total chlorophyll-a (chla) and Secchi depth (Secchi)—as response variables, creating 4 distinct data sets for our experiments. Our goal was to predict the response variables for each lake based on a set of predictor variables (features) that

included land cover, land use, and climate variables. Since there are multiple lake water quality measurements taken at different times for each lake, we computed a single value for each lake by taking the mean of all measurements during the summer months after 2010. The statistics for each data set are shown in Table 2.

5.1.3 Benchmark data from UCI Machine Learning Repository The benchmark datasets used include `housing` [17], `wine` [18], `Parkinson` [19], `online news` [20], and `concrete strength` [21] (see Table 2).

5.2 Experimental Setup We performed our experiments on a Dell PowerEdge R620 server with 2.7GHz Dual Intel Xeon processor. The proposed framework and other baseline methods were written in Matlab.

5.2.1 Baseline Methods. First, we compare our proposed supervised hash-based feature learning method for complete data against the following three baseline algorithms: (1) linear SVR trained on the raw features, (2) nonlinear SVR trained on the raw features, (3) linear SVR trained on the random Fourier features.

Second, we compare `H-FLIP`, which is an extension of our supervised hash-based framework to deal with incomplete data, against the following three methods:

- *MC+Raw*: Missing values are imputed during preprocessing using matrix completion. A linear SVR is then trained on the imputed data.
- *MC+PCA*: Missing values are imputed during preprocessing using matrix completion. We then apply principal component analysis (PCA) to extract features from the imputed data. A linear SVR is then trained to fit the PCA-reduced data.
- *MC+RFF* This is similar to the previous two approaches except we use random Fourier features as feature learning on the imputed data. A linear SVR is then trained on the unsupervised RFF.

For a fair comparison, we extract an equal number of features for all the methods, unless specified otherwise. Specifically, the number of PCA components, unsupervised RFF, and supervised hash-based features are set to 50. For `H-FLIP`, we first project the data to 200 sparse RFF before reducing it to the 50-dimensional hash-based features using our supervised learning framework. The regularization parameter λ in `H-FLIP` is determined using cross validation, while the parameter α in Equation (4.6) is set to 0.01. We apply SVR to the features generated by the baseline and proposed methods. The hyper-parameters of SVR are chosen using nested cross validation [22], in which an inner 3-fold cross validation is performed for hyper-parameter tuning and an outer 5-fold cross validation is performed for model assessment.

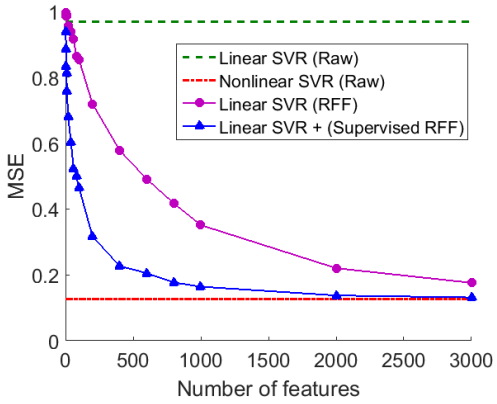


Figure 4: Average MSE for linear SVR with raw features, nonlinear SVR with raw features, linear SVR with RFF features, and linear SVR with supervised hash-based features on complete synthetic data.

5.2.2 Evaluation Metrics. We consider both the imputation error as well as the prediction accuracy of the induced SVR models. To assess the error in missing value imputation, let $\mathbf{A}_c$ be the true complete data matrix and $\mathbf{X}$ be the estimated (imputed) matrix. The imputation error is computed as follows:

$$\text{Imputation error} = \|\mathcal{P}(\mathbf{X}) - \mathcal{P}(\mathbf{A}_c)\|_2^2 / \|\mathcal{P}(\mathbf{A}_c)\|_2^2.$$

We also evaluate the performance of the SVR models in terms of their mean square prediction error, $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2$, where $\hat{y}_i$ is the predicted response value for the i-th data instance and y_i is its true value.

5.3 Experimental Results This section summarizes the results of our experimental evaluation.

5.3.1 Results for Synthetic Data We compared the proposed framework against linear SVR on raw features, nonlinear SVR (with RBF kernel) on raw features, and linear SVR on unsupervised RFF features. As expected, Figure 4 shows that linear SVR on the raw features is worse than other methods as it fails to capture the nonlinear relationships in the data. In addition, the accuracy of linear SVR on both RFF and our supervised hash-based features improves as the number of features increases. More importantly, they are comparable to the accuracy of nonlinear SVR. This supports the rationale for using RFF to capture nonlinear dependencies in the data. Finally, comparing RFF against the proposed supervised hash-based features, we observe that the supervised approach does not require as many features to achieve high accuracy compared to unsupervised RFF. This justifies the case for using supervised hash-based feature learning for nonlinear regression problems.

Table 3: Imputation error for synthetic data

Percent of missing value	10%	20%	30%
Mean imputation (MI)	1.0002	1.0002	1.0003
Matrix completion (MC)	0.0271	0.0276	0.0280
H-FLIP	0.0273	0.0280	0.0290

Table 4: MSE of linear SVR on synthetic data

% missing	10%	20%	30%	#features
MC+Raw	0.9680	0.9683	0.9679	100
MC+PCA	0.9681	0.9683	0.9683	50
MC+RFF	0.8960	0.8887	0.8997	50
H-FLIP	**0.4596**	**0.4610**	**0.4626**	50

Next, we add missing values randomly into the synthetic data and compare the imputation error of `H-FLIP` against methods that apply mean imputation and matrix completion during preprocessing. The results in Table 3 suggest that mean imputation has the highest imputation error while matrix completion has the lowest error. The imputation error of `H-FLIP` is very close to the imputation error for matrix completion, which is not surprising as `H-FLIP` is designed not only to recover the incomplete data, but also to fit the response variable as accurately as possible. The imputation errors of both matrix completion and `H-FLIP` also do not change significantly as we vary the percent of missing values from 10% to 30%, which shows the robustness of our proposed framework.

We also compare the MSE values of the regression models. The results in Table 4 suggest that the MSE for raw features and PCA-induced features are worse than unsupervised RFF. `H-FLIP` outperforms all the baseline methods because it learns the appropriate nonlinear features and imputes the missing values without adding significant bias that could degrade the performance of the regression model.

5.3.2 Results for Lake Water Quality Data First, we report the results of applying the various methods to the complete lake water quality data with no missing values using Secchi as response variable. Figure 5 shows that the MSE for linear SVR on the supervised hash-based features is slightly better than linear SVR on the raw features. More importantly, the supervised hash-based features can achieve a low MSE with fewer number of features compared to unsupervised RFF.

We repeat the experiments by adding 20% missing values to the predictor variables and compare the MSE for `H-FLIP` against the baseline methods. The results shown in Table 5 suggest that MC+RFF with only 50 features performs the worst, which is consistent with our previous observation that a large number of RFF

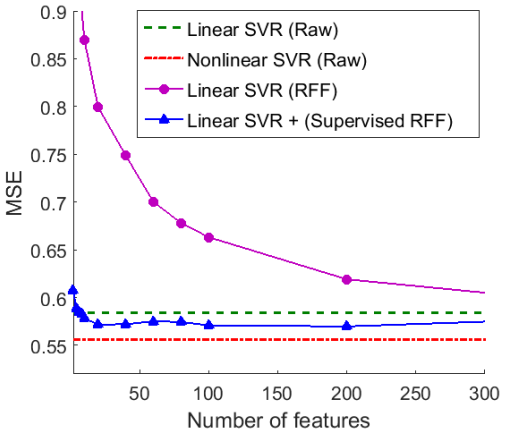


Figure 5: Comparing average MSE of linear SVR on the complete Secchi data.

Table 5: MSE of linear SVR for lake water quality data

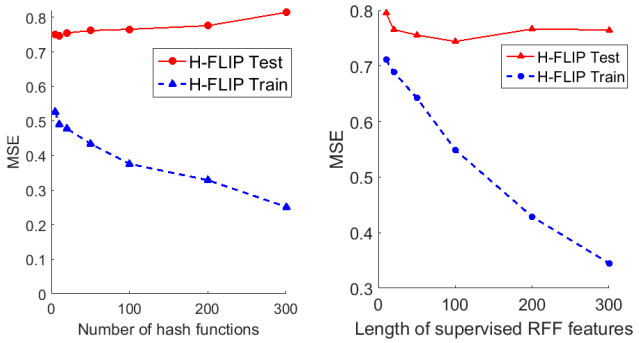
	TP	TN	Chla	Secchi	#features
MC+Raw	**0.78**	0.73	0.72	0.60	356
MC+PCA	0.81	0.75	0.74	0.63	50
MC+RFF	0.84	0.82	0.81	0.70	50
MC+RFF	0.79	0.74	0.72	0.60	300
H-FLIP	0.79	**0.70**	**0.71**	**0.58**	50

Table 6: MSE of linear SVR for UCI data with 20% missing values.

Method	Housing	Wine	Parkinson	News	Concrete
MC+Raw	0.35	0.76	0.80	**0.92**	0.50
MC+PCA	0.36	0.83	0.97	0.94	0.67
MC+RFF	0.38	0.76	0.85	0.94	0.42
H-FLIP	**0.32**	**0.69**	**0.74**	**0.92**	**0.36**

is needed to effectively represent the data. As we increase the number of RFF from 50 to 300, the MSE improves significantly, comparable to the results for MC+Raw. Nevertheless, `H-FLIP` with 50 hash-based features outperforms all other methods in 3 of the 4 data sets. In fact, the MSE of linear SVR with `H-FLIP` on the incomplete data is comparable to the results for nonlinear SVR on the complete data (see Table 1).

5.3.3 Results for UCI Benchmark Data The results in Table 6 show that `H-FLIP` outperforms the baseline methods in 4 of the 5 data sets. The improvement in `H-FLIP` is more significant here compared to the lake data as there are more nonlinear relationships in these data sets. Nonlinear SVR has a lower MSE than linear SVR by more than 0.10 in 3 of the 5 UCI benchmark data but in none of the lake data (see Table 1).



(a) RFF basis functions (b) Hash-based features

Figure 6: MSE of H-FLIP when varying the number of basis functions and supervised hash-based features

5.4 Sensitivity Analysis Using total nitrogen (TN) as response variable, we investigate how sensitive the `H-FLIP` results are when varying the number of hash functions (K) and the length of each sparse RFF (p). We first vary the number of hash functions from 5 to 300 while fixing the length of each sparse RFF to be 200. The results shown in Figure 6a suggest that the test MSE is not sensitive to the number of hash functions. Next, we vary the length of the sparse RFF from 10 to 300 while fixing the number of hash functions to be 50. The results given in Figure 6b suggest that the test error of `H-FLIP` is not that sensitive to the increasing length of the RFF compared to its training error.

6 Conclusion

This paper presents `H-FLIP`, a hash-based feature learning framework for incomplete data. The framework is designed to train a small set of hash functions using random Fourier features to effectively model nonlinear relationships in the data while simultaneously imputing all the missing values in the data.

7 Acknowledgements

This research was supported by National Science Foundation under grant #EF-1065786 and #IIS-1615612.

References

[1] J. Gui, Z. Sun, S. Ji, D. Tao, and T. Tan, "Feature selection based on structured sparsity: A comprehensive study," IEEE Trans Neural Netw Learn Syst, 2016.

[2] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in CVPR, 2015.

[3] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," CoRR, vol. abs/1408.2927, 2014.

[4] W. Kong, W.-J. Li, and M. Guo, "Manhattan hashing for large-scale image retrieval," in SIGIR, 2012, pp. 45–54.

[5] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in NIPS, 2008, pp. 1177–1184.

[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," J. of Comp. Sys. Sci., vol. 60, pp. 327–336, 1998.

[7] M. Charikar, "Similarity estimation techniques from rounding algorithms," in STOC, 2002, pp. 380–388.

[8] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in NIPS, 2009, pp. 1509–1517.

[9] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in NIPS 21, 2009, pp. 1753–1760.

[10] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in ICML, 2011, pp. 353–360.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," TPAMI, vol. 35, pp. 1798–1828, 2013.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc of NIPS, 2012.

[13] L. Zhao, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Hierarchical incomplete multi-source feature learning for spatiotemporal event forecasting," in KDD, 2016, pp. 2085–2094.

[14] P. Soranno et al., "Building a multi-scaled geospatial temporal ecology database from disparate data sources: Fostering open science through data reuse," Journal of Giga Science, 2015.

[15] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems," Pacific J. of Optimization, 2009.

[16] S. Ji and J. Ye, "An accelerated gradient method for trace norm minimization," in ICML, 2009, pp. 457–464.

[17] Housing Data Set, 1993. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Housing

[18] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," Decis. Support Syst., vol. 47, pp. 547–553, 2009.

[19] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig, "Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests." IEEE Trans. Biomed. Engr., vol. 57, pp. 884–893, 2010.

[20] K. Fernandes, P. Vinagre, and P. Cortez, "A proactive intelligent decision support system for predicting the popularity of online news," in EPIA, 2015, pp. 535–546.

[21] I.-C. Yeh, "Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research, vol. 28, pp. 1797–1808, 1998.

[22] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," BMC Bioinformatics, vol. 7, p. 91, 2006.