# Additive Nearest Neighbor Feature Maps

Zhenzhen Wang[1,2], Xiao-Tong Yuan[1], Qingshan Liu[1] and Shuicheng Yan[2]
[1] Nanjing University of Information Science and Technology, China
[2]National University of Singapore, Singapore
wazhenzhen@gmail.com, {xtyuan,qsliu}@nuist.edu.cn, eleyans@nus.edu.sg

## Abstract

*In this paper, we present a concise framework to approximately construct feature maps for nonlinear additive kernels such as the Intersection, Hellinger's, and $\chi^2$ kernels. The core idea is to construct a set of anchor points for each individual feature and assign to every query the feature map of its nearest neighbor or the weighted combination of those of its $k$-nearest neighbors in the anchors. The constructed feature maps can be compactly stored by a group of nearest neighbor (binary) indication vectors along with the anchor feature maps. The approximation error of such an anchored feature mapping approach is analyzed. We evaluate the performance of our approach on large-scale nonlinear support vector machines (SVMs) learning tasks in the context of visual object classification. Experimental results on several benchmark data sets show the superiority of our method over existing feature mapping methods in achieving reasonable tradeoff between training time and testing accuracy.*

## 1. Introduction

Discriminative classification using kernel Support Vector Machines (SVMs) is one of the standard techniques used in supervised machine learning and computer vision. According to the linearity or nonlinearity of the kernel function $K(\cdot,\cdot)$, the kernel SVMs can be categorized as linear SVMs or nonlinear SVMs. It is commonly acknowledged that the nonlinear SVMs tend to achieve better performances than those linear ones. For instance, by using $\chi^2$ kernel SVMs, Yang *et al*. [31] observed $\sim 15\%$ improvement over linear SVMs on the Caltech-256 dataset [12], Li [18] achieved more than 5% improvement on MNIST dataset [15]. Maji and Berg [19] improved classification accuracy by more than $10\%$ using Intersection kernel on Caltech-101 dataset [10] and DaimlerChrysler pedestrian dataset [21]. Vedaldi and Zisserman [27] demonstrated the effectiveness of the Hellinger's kernel by an increase of $12\%$ in classification accuracy on the Caltech-101 dataset.

Although often being inferior to their nonlinear counterparts in accuracy, the linear SVMs have been shown to scale well to sample and/or feature size by using large-scale optimization tools such as stochastic optimization and consensus optimization [9, 24, 4, 5]. Models that operate on the nonlinear kernel matrices usually take much longer time to train and test than those linear ones. For example, a dataset with half a million training examples and each of which consists of thousands of features might take days to train with any nonlinear kernels, not to mention using billion or trillion features. In contrast, it only needs a few hours to train using linear SVMs. Thus, with the current trend of using ultra-high-dimensional representations and examples, huge computational cost and increasing storage space of nonlinear SVMs have gradually been a bottleneck for the extensive use of nonlinear classifiers.

The appealing classification performance of nonlinear SVMs and the scalability of linear SVMs inspire researchers to develop the so called *feature mapping* technique to bridge the gap between linear and nonlinear SVMs [27, 18, 30, 20]. The common spirit of this type of methods is to map a data sample $\boldsymbol{x} \in \mathbb{R}^D$ to a higher dimensional feature vector $\phi(\boldsymbol{x})$ such that the linear product of any two samples $\phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$ can well approximate the kernel function $K(\boldsymbol{x},\boldsymbol{y})$ evaluated on these two samples. After such a feature mapping, nonlinear SVMs can be converted into linear ones for efficient and scalable training and testing, while preserving the classification quality of the original nonlinear classifiers.

In this paper, we are particularly interested in constructing feature maps for the following nonlinear kernels with additive structure:

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_{d=1}^{D} K_d(x_d, y_d), \tag{1}$$

where $K_d$ are bi-variate nonlinear functions. For example, the $\chi^2$ kernel with $K_{\chi^2}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{D} \frac{2x_i y_i}{x_i+y_i}$ and the Intersection kernel with $K_{\min}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{D} \min(x_i, y_i)$ are two popular instances of additive kernels. The additive structure of these kernels allows us to perform feature mapping independently for individual features. That is, we may

construct for each feature $x_d$ a feature map $\phi(x_d)$ such that $K_d(x_d, y_d) \approx \phi(x_d)^\top \phi(y_d)$ and then simply set $\phi(\boldsymbol{x})$ as the concatenation of $\phi(x_d)$.

We develop in this paper a nearest neighbor coding based approach to accomplish the task of additive kernel feature mapping. *The intuition is that if a (scalar) feature space can be well covered by a small set of anchors, then the feature maps of that feature space are hopefully to be well approximated by those feature maps of the anchors.* Based on this idea, we propose to first find a group of anchor points to cover each individual feature dimension using, e.g., uniform partition or $k$-means clustering [6], and then construct the feature maps for the anchors through Singular Value Decomposition (SVD) [8]. Finally, we define the feature map of a query feature point as that of its nearest neighbor (or a linear combination of those of its $k$-nearest neighbors) in the corresponding anchors. Figure 1 illustrates the working mechanism of the proposed feature mapping strategy. Theoretic analysis shows that the approximation error of our method is controlled by the covering accuracy of the anchor points. The merits of the proposed nearest neighbor feature mapping method are highlighted as follows:

- **Generality:** the proposed method is a general framework applicable to any additive kernels whose components are Mercer kernels [11].
- **Theoretical guarantee:** the approximation error of the proposed feature maps is well bounded.
- **Sparsity:** The feature maps can be compactly stored as a group of nearest neighbor binary indication vectors along with the feature maps of anchors.

The performance of our method has been extensively evaluated on several visual classification benchmark datasets.

## 2. Related Work

As an appealing strategy to convert nonlinear kernel methods into linear ones, approximating kernel functions via explicit feature maps has recently gained huge popularity in computer vision and machine learning. For example, random Fourier feature maps [23, 2, 16, 29, 14] have attracted much attention by expressing the kernel as a Fourier expansion. Almost all these algorithms are designed for shift-invariant kernels and share a common spirit: the feature maps are often generated based on a finite set of random basis functions (i.e., cosine and sine functions) which are independent on the training data. In contrast, the Nyström method [32, 28] randomly samples from the training examples and it can yield better generalization error bound when there is a large gap in the eigen-spectrum of the kernel matrix. It has been shown in [17, 3] that Chebyshev polynomials can improve random Fourier approximation by removing the need to use periodic approximations to the $\chi^2$ function. Analogous to random Fourier feature maps, Yang *et al.* [30] developed random Laplace feature maps for a family of kernel functions adapted to the semigroup structure of $\mathbb{R}^d_+$. In order to extend feature maps to kernels suitable for histograms, Li *et al.* [16] developed approximate feature maps for arbitrary, locally compact Abelian groups, such as the Intersection and $\chi^2$ kernels. Vedaldi and Zisserman [27] extended feature maps to a family of additive homogeneous kernels of which the Intersection, $\chi^2$, Jensen-Shannon (JS), and Hellinger's kernels are special cases. Inspired by kernel PCA, Perronnin *et al.* proposed addkPCA [22] to construct feature maps for additive kernels by applying Nyström approximation to each dimension of the data independently. Although efficient, it should be noted that addkPCA is data-dependent and thus is of limited interests in applications such as online learning [27].

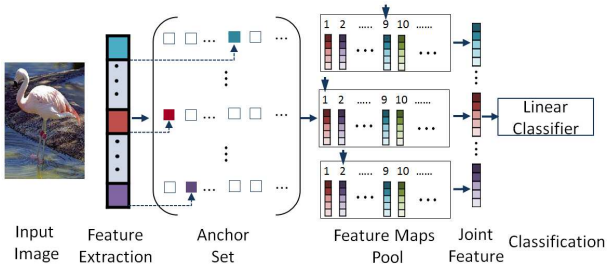


Figure 1. Schematic process of the key working mechanism. For each individual feature component, we find a group of anchor points to cover the training samples of that component and construct the feature maps of the anchor set as a feature maps pool of that component. Given an input image feature vector, we assign to each feature component the feature map of its nearest neighbor or the weighted combination of its $k$ nearest neighbors in the corresponding anchor set and concatenate them as a joint feature map for linear classifiers training/testing.

In addition to those general purpose feature maps for approximating certain kernel families, there are several feature maps proposed to approximate specific kernels. For instance, Maji and Berg [20] have proposed a sparse feature map for the Intersection kernel $K_{\min} = \min(x, y)$, obtaining up to $10^3$ speedup in the learning of corresponding SVMs. Li *et al.* [18] recently proposed a random projection based algorithm called *Sign Stable Projections* to speedup $\chi^2$ kernel SVMs.

In contrast to these prior methods which are more or less restricted to certain families of kernels, or are dependent on training data, we develop in this paper a general framework to construct feature maps applicable to a broader class of additive kernels with Mercer components.

## 3. Additive Nearest Neighbor Feature Maps

In this section we present a novel method to construct feature maps for additive kernels. The principle of our method is: for each individual feature component, we assign to a query sample the feature map of its nearest neigh-

bor (or a linear combination of the feature maps of its $k$ nearest neighbors) in a small set of anchor samples that approximately cover that feature space. In high level, our method contains the following three core steps:

- **Feature-wise anchors construction:** For each feature component, we construct a set of anchors which span as much as possible of the data variability (§ 3.2).
- **Feature maps for anchors:** We then build feature maps for each set of anchor points based on the SVD of a kernel matrix evaluated on these anchors (§ 3.3).
- **Nearest neighbor feature mapping:** Given a query vector, we assign to each feature component the feature map of its nearest neighbor or the linear combination of those of its $k$ nearest neighbors in the corresponding anchor set. The joint feature map is then constructed by concatenating the feature maps of individual features (§ 3.4).

Before elaborating these steps in details, we introduce in §3.1 a naive nearest neighbor feature mapping strategy that inspires our approach.

### 3.1. Warm up

**Naive nearest neighbor feature maps.** Let $\mathcal{X} \subseteq \mathbb{R}^D$ be a feature space of interest and $K(\cdot,\cdot)$ be a Mercer kernel defined over $\mathcal{X} \times \mathcal{X}$. Assume that we are given a set of anchor points $\hat{\mathcal{X}} \subset \mathbb{R}^D$ of size $N$ that well spans $\mathcal{X}$. For any element $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$, we assume that its feature map $\phi(\hat{\boldsymbol{x}})$ are available on hand, i.e., $K(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}') \approx \phi(\hat{\boldsymbol{x}})^\top \phi(\hat{\boldsymbol{x}}')$. Then for any $\boldsymbol{x} \in \mathcal{X}$, it is natural to define

$$\phi(\boldsymbol{x}) := \phi(\mathcal{N}(\boldsymbol{x})),$$

where $\mathcal{N}(\boldsymbol{x})$ is the nearest neighbor of $\boldsymbol{x}$ in $\hat{\mathcal{X}}$, i.e., $\mathcal{N}(\boldsymbol{x}) := \arg\min_{\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|$. It can be imagined that $\phi(\boldsymbol{x})$ should be a reasonable feature map of $\boldsymbol{x}$. Indeed, for any $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{X}$, since $\mathcal{X}$ is well approximated by $\hat{\mathcal{X}}$, it holds that $K(\boldsymbol{x}, \boldsymbol{y}) \approx K(\mathcal{N}(\boldsymbol{x}), \mathcal{N}(\boldsymbol{y})) \approx \phi(\mathcal{N}(\boldsymbol{x}))^\top \phi(\mathcal{N}(\boldsymbol{y})) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$. The closer $\boldsymbol{x}$ is to its nearest neighbor $\mathcal{N}(\boldsymbol{x})$, the more accurate the feature maps are expected to be.

**A curse of dimensionality.** The above anchor points based nearest neighbor feature mapping strategy is rather general in the sense that it can be defined for any Mercer kernel. Unfortunately, there is no free lunch: to ensure accurate approximation, the anchor set size $N$ is expected to have *exponential* dependency on the dimensionality $D$. To roughly justify this curse, we note that the accuracy of the above feature mapping strategy is largely controlled by representation quality of $\hat{\mathcal{X}}$ to $\mathcal{X}$. Intuitively, it is desirable that any $\boldsymbol{x} \in \mathcal{X}$ and its nearest neighbor $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ should be close enough so that $K(\boldsymbol{x}, \boldsymbol{y})$ can be well approximated by $K(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. The following concept of $\epsilon$-cover, which is standard in statistical learning theory [25], can be used to find an approximation to a rich data set:

**Definition 1** ($\epsilon$-Cover)**.** *Given any $\epsilon > 0$. An $\epsilon$-cover of a set $\mathcal{X} \subseteq \mathbb{R}^D$ is a set $\hat{\mathcal{X}} \subseteq \mathbb{R}^D$ such that for each $x \in \mathcal{X}$ there is a $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ such that $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \leq \epsilon$. The $\epsilon$-covering number of $\mathcal{X}$ is*

$$N(\epsilon, \mathcal{X}, D) := \min\{|\hat{\mathcal{X}}| : \hat{\mathcal{X}} \textit{ is an } \epsilon\textit{-cover of } \mathcal{X}\}.$$

It is reasonable to assume that the anchor set $\hat{\boldsymbol{\mathcal{X}}}$ is an $\epsilon$-cover of $\boldsymbol{\mathcal{X}}$ for some $\epsilon$. For a compact manifold $\mathcal{X}$ with intrinsic dimensionality $D$, it is well known that $N(\epsilon, \mathcal{X}, D) = O(\epsilon^{-D})$. This implies that the size of an $\epsilon$-cover $\hat{\boldsymbol{\mathcal{X}}}$ is expected to be exponential in $D$. Thus, a direct estimation of multivariate nearest neighbor feature maps $\phi(\boldsymbol{x})$ is neither computationally efficient nor storage economic even for moderately large $D$.

To summarize the descriptions so far, although general, the anchor points nearest neighbor assignment based feature mapping method suffers from the curse of dimensionality. In the following subsections, we will further explore the structure of additive kernels and introduce a feature-wise nearest neighbor feature mapping approach which successfully overcomes such a barrier of computation and storage.

### 3.2. Feature-wise anchor points construction

Recall that we are interested in the additive kernels as expressed in (1). Our proposal is to take advantage of such an additive structure to perform element-wise anchor set nearest neighbor feature mapping. Let $\boldsymbol{X} = \{\boldsymbol{x}_m\}_{m=1}^M$ be a set of data samples of cardinality $M$ and $\boldsymbol{x}_m = [x_{1,m}, \ldots, x_{D,m}]^\top$. For each feature $d \in [1, \ldots, D]$, we denote $\boldsymbol{X}_d = \{x_{d,m}\}_{m=1}^M$ as the one-dimensional set containing all the observed samples of feature $d$. Given $\epsilon > 0$, the following two strategies can be used to construct an $\epsilon$-cover of $\boldsymbol{X}_d$:

- **Uniform partition.** Assume $x_{d,m}$ are bounded and uniformly distributed. Without loss of generality we assume that $x_{d,m} \in [0, 1]$. In this case, we simply choose $N = \epsilon^{-1}$ and set

$$\hat{\boldsymbol{X}}_d = \left[0, \frac{1}{N}, \frac{2}{N}, ..., 1\right]. \tag{2}$$

  Apparently, $\hat{\boldsymbol{X}}_d$ is an $\epsilon$-cover of $\boldsymbol{X}_d$. The advantage of such a quantization strategy is that it is data independent whilst the disadvantage is that it makes uniform distribution assumption on the data which could be restrictive in real applications.

- **$k$-means-type partition.** If the distribution of $\{x_{d,m}\}$ is unknown, we consider the following $k$-means-type quantization:

$$\{\hat{\boldsymbol{v}}_d, \hat{\boldsymbol{u}}_{d,m}\} = \arg\min_{\boldsymbol{u}_{d,m}, \boldsymbol{v}_d} \sum_{m=1}^{M} \|x_{d,m} - \boldsymbol{v}_d^\top \boldsymbol{u}_{d,m}\|^2,$$
$$s.t.\ \boldsymbol{u}_{d,m} \in \{0,1\}^N, \textit{Card}(\boldsymbol{u}_{d,m}) = 1, \tag{3}$$

where $\boldsymbol{v}_d \in \mathbb{R}^N$ is the one-dimensional code book and $\mathbf{u}_{d,m}$ is the $N$-dimensional quantization code. The constraint in this formulation ensures that the $N$-bit code $\mathbf{u}_{d,m}$ is binary with only one nonzero element. Both the code book and the quantization codes are unknown and can be easily estimated by a $k$-means-type iteration procedure, i.e., to alternate between updating centers and updating nearest neighbor assignment until convergence. After the optimal $\hat{\boldsymbol{v}}_d$ is obtained, we set anchor set $\hat{\boldsymbol{X}}_d := \hat{\boldsymbol{v}}_d$. As a greedy scheme, we may solve a sequence of problem (3) with increasing code book size $N$ until the criterion $\|x_{d,m} - \hat{\boldsymbol{v}}_d^\top \hat{\mathbf{u}}_{d,m}\| \leq \epsilon, \forall m$ is met. This guarantees that the anchor set $\hat{\boldsymbol{X}}_d$ is an $\epsilon$-cover of $\boldsymbol{X}_d$. As an extreme case, we may directly assign $\hat{\boldsymbol{X}}_d := \boldsymbol{X}_d$ as a trivial $\epsilon$-cover of $\boldsymbol{X}_d$. This cover, however, is typically redundant. Since $\boldsymbol{X}_d$ is a scalar set, it is expected that $|\hat{\boldsymbol{X}}_d| = O(\epsilon^{-1})$. The advantage of such a $k$-means-type quantization method lies in that it adapts well to unknown data distribution.

### 3.3. Feature maps for anchors

The next step is to perform feature mapping for each individual anchor point set $\hat{\boldsymbol{X}}_d$, $d = 1, ..., D$. Generally speaking, any off-the-shelf feature mapping method (e.g. [27][18]) can be applied in this step. These existing methods, however, will inevitably introduce approximation error. Recall that in this paper we are particularly interested in the case where each kernel component $K_d(\cdot, \cdot)$ is a Mercer kernel. For this special class of kernels, in order to minimize the loss in this step, we propose to use a SVD based strategy to construct the anchor point feature maps. Let $\hat{\boldsymbol{K}}_d \in \mathbb{R}^{N \times N}$ be the kernel matrix evaluated over $\hat{X}_d \times \hat{X}_d$. Since $K_d(\cdot, \cdot)$ is a Mercer kernel, we have $\hat{\boldsymbol{K}}_d \succeq 0$. If $\hat{\boldsymbol{K}}_d$ has rank $r$, then there exists a matrix $\hat{\boldsymbol{Q}}_d = [\hat{\boldsymbol{q}}_{d,1}, ..., \hat{\boldsymbol{q}}_{d,N}] \in \mathbb{R}^{r \times N}$ such that $\hat{\boldsymbol{K}}_d = \hat{\boldsymbol{Q}}_d^\top \hat{\boldsymbol{Q}}_d$ (e.g., through SVD or Cholesky decomposition [13]). For any anchor point $\hat{x}_{d,n} \in \hat{\boldsymbol{X}}_d$, this implies an exact $r$-dimensional feature map $\phi(\hat{x}_{d,n}) := \hat{\boldsymbol{q}}_{d,n}$. In the following descriptions, we will denote $\hat{\boldsymbol{\Phi}}_d = [\phi(\hat{x}_{d,1}), \ldots, \phi(\hat{x}_{d,N})]$ as the anchor set feature maps matrix of feature $d$. If the anchor set $\hat{\boldsymbol{K}}_d$ is obtained in a data independent way, e.g., using the strategy of uniform partition, then the anchor point feature maps $\phi(\hat{x}_{d,n})$ will also be data independent. If $\hat{\boldsymbol{K}}_d$ is data dependent, e.g., it is obtained by $k$-means-type quantization, then the feature map will be data dependent as well.

### 3.4. Nearest neighbor feature mapping

For any $\boldsymbol{x}_m \in \boldsymbol{X}$, we define the feature map of its $d$-th element $x_{d,m}$ as

$$\phi(x_{d,m}) := \phi(\mathcal{N}(x_{d,m})),$$

where $\mathcal{N}(x_{d,m})$ denotes the nearest neighbor of $x_{d,m}$ in the anchor set $\hat{\boldsymbol{X}}_d$. The ANNFM of $\boldsymbol{x}_m$ can be constructed by concatenating its element-wise feature maps as follows:

$$\phi(\boldsymbol{x}_m) := [\phi(x_{1,m})^\top, ..., \phi(x_{D,m})^\top]^\top. \tag{4}$$

It is straightforward to verify that

$$\begin{aligned} K(\boldsymbol{x}_m, \boldsymbol{x}_{m'}) &= \sum_{d=1}^{D} K_d(x_{d,m}, x_{d,m'}) \\ &\approx \sum_{d=1}^{D} K_d(\mathcal{N}(x_{d,m}), \mathcal{N}(x_{d,m'})) \\ &= \sum_{d=1}^{D} \phi(\mathcal{N}(x_{d,m}))^\top \phi(\mathcal{N}(x_{d,m'})) \\ &= \phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_{m'}). \end{aligned}$$

Therefore, it is reasonable to regard $\phi(\boldsymbol{x}_m)$ as a feature map for $\boldsymbol{x}_m$. We will analyze the approximation error of such a feature mapping approach in the next section. It is noteworthy that the existing additive feature mapping methods can be taken as special cases of our method when the trivial anchor sets $\hat{\boldsymbol{X}}_d = \boldsymbol{X}_d$ are used.

An appealing merit of ANNFM is that it is storage economic when the anchor sets are relatively small. To see this point, note the feature-wise mapping can be expressed as

$$\phi(x_{d,m}) = \phi(\mathcal{N}(x_{d,m})) = \hat{\boldsymbol{\Phi}}_d \hat{\boldsymbol{u}}_{d,m},$$

where $\hat{\boldsymbol{u}}_{d,m}$ is the nearest neighbor indication vector, i.e., the entries of $\hat{\boldsymbol{u}}_{d,m}$ are all zeros except the entry corresponding to the nearest neighbor. We now rewrite $\phi(\boldsymbol{x}_m)$ as

$$\phi(\boldsymbol{x}_m) = [\hat{\boldsymbol{u}}_{1,m}^\top \hat{\boldsymbol{\Phi}}_1^\top, ..., \hat{\boldsymbol{u}}_{D,m}^\top \hat{\boldsymbol{\Phi}}_D^\top]^\top.$$

This suggests that we can store each data sample $\boldsymbol{x}_m$ as a set of extremely sparse binary codes $\{\hat{\boldsymbol{u}}_{d,m}\}_{d=1}^D$ and its feature map can be obtained by multiplying these binary codes with a set of pre-computed anchor point feature maps $\{\hat{\boldsymbol{\Phi}}_d\}_{d=1}^D$. In view of this, ANNFM is essentially a table-look-up method in which the indication vectors $\hat{\boldsymbol{u}}_{d,m}$ are indices while $\hat{\boldsymbol{\Phi}}_d$ are tables.

So far ANNFM simply assigns to a query scalar the feature map of its nearest neighbor in the anchors. As a variant of ANNFM based on $k$-nearest neighbors assigment, the $k$-ANNFM is defined as:

$$\phi_k(\boldsymbol{x}_m) := [\phi_k(x_{1,m})^\top, ..., \phi_k(x_{D,m})^\top]^\top, \tag{5}$$

in which the feature-wise mapping is given by

$$\phi_k(x_{d,m}) := \sum_{j=1}^{k} w_j \phi(\mathcal{N}_j(x_{d,m})).$$

Here $\{\mathcal{N}_j(x_{d,m})\}_{j=1}^k$ are the $k$-nearest neighbors of $x_{d,m}$ in the anchor set $\hat{\boldsymbol{X}}_d$ and $\{w_j\}_{j=1}^k$ are non-negative combination weights, e.g., $w_j = 1/k, j = 1, ..., k$. On one

hand, $k$-ANNFM is expected to be more stable if the kernel is approximately linear in a local manifold. On the other hand, $k$-ANNFM is more storage demanding as there are $k$ nonzero elements in a $k$-nearest neighbors indication vector. Specially, when $k = 1$ and $w_1 = 1$, $k$-ANNFM reduces to the ANNFM.

### 3.5. Application to Nonlinear SVMs

Among others, an important application of feature maps is for large-scale nonlinear SVMs learning. In Algorithm 1, we summarize the working flow of ANNFM in the context of nonlinear SVMs. Comparing to the existing additive feature maps, the advantages of ANNFM are highlighted in below:

- It is a general framework for additive kernel feature mapping with rather weak restriction (i.e., Mercer kernels) imposed on the kernels.
- ANNFM is essentially a binary coding scheme and thus only needs 1 bit per mapped feature for storage. In contrast, most existing additive feature maps are *dense* and may need 16 bits to store each mapped feature.

**Algorithm 1:** Additive Nearest Neighbor Feature Maps for Nonlinear SVMs Learning.

**Input** : A set of $D$-dimensional data samples $\boldsymbol{X} = \{\boldsymbol{x}_m\}_{m=1}^M$.

(**S1**) **for** *Every feature index* $d$ **do**

  (**SS1**) Construct an anchor set $\hat{\boldsymbol{X}}_d$ of size $N$ using, e.g., uniform partition or $k$-means-type partition;

  (**SS2**) Compute the kernel matrix $\hat{\boldsymbol{K}}_d \in \mathbb{R}^{N\times N}$ over $\hat{\boldsymbol{X}}_d \times \hat{\boldsymbol{X}}_d$;

  (**SS3**) Perform square-root decomposition $\hat{\boldsymbol{K}}_d = \hat{\boldsymbol{Q}}_d^\top \hat{\boldsymbol{Q}}_d$ by, e.g., SVD or Cholesky decomposition;

  (**SS4**) Define the anchor set feature maps as $\hat{\boldsymbol{\Phi}}_d := \hat{\boldsymbol{Q}}_d$;

**end**

(**S2**) Generate the ANNFM $\{\phi(\boldsymbol{x}_m)\}_{m=1}^M$ according to (4) (or alternatively $k$-ANNFM $\{\phi_k(\boldsymbol{x}_m)\}_{m=1}^M$ according to (5));

(**S3**) Train linear SVMs using ANNFM (or $k$-ANNFM) as input features;

(**S4**) Apply the same mapping strategy on testing data and evaluate the trained linear classifier.

**Output**: Classification error on testing set.

## 4. Error Analysis of ANNFM

In this section we analyze the approximation quality of ANNFM in (4). For each feature $d$, given a set of anchor points $\{\hat{x}_{d,n}\}$, we consider the following superposition stochastic model to generate the observed features $\{x_{d,m}\}$:

$$x_{d,m} = \mathcal{N}(x_{d,m}) + \delta_{d,m}, \tag{6}$$

where $\delta_{d,m}$ is a zero-mean sub-Gaussian random variable with parameter $\sigma > 0$ (i.e., $\mathbb{E}[\exp\{\eta\delta_{d,m}\}] \leq \exp\left\{\sigma^2\eta^2/2\right\}$, for all $\eta \in \mathbb{R}$). For example, $\delta_{d,m}$ can be zero-mean Gaussian variables and zero-mean bounded random variables.

The following assumption on the kernel function $K(\cdot,\cdot)$ is needed in our analysis.

**Assumption 1.** *For any feature index $d$, the kernel component $K_d(\cdot,\cdot)$ satisfies*

$$\triangle K_d(x, y; \delta_x, \delta_y) = O(\delta_x^2 + \delta_y^2),$$

*where* $\triangle K_d(x, y; \delta_x, \delta_y) := K_d(x + \delta_x, y + \delta_y) - K_d(x, y) + \frac{\partial K_d}{\partial x}\delta_x + \frac{\partial K_d}{\partial y}\delta_y$. *Also we assume that there exists a constant $L > 0$ such that $\|\nabla K_d(x, y)\| \leq L$, $\forall d$.*

For example, Assumption 1 is valid when $K_d(\cdot,\cdot)$ is strongly smooth and strongly convex, i.e., there exist $\rho_+ > 0$ and $\rho_- > 0$ such that $0.5\rho_-[(\delta_a)^2 + (\delta_b)^2] \leq \triangle K_d(a, b; \delta_a, \delta_b) \leq 0.5\rho_+[(\delta_a)^2 + (\delta_b)^2]$.

In the analysis to follow, in order to remove the scaling effect of summation, we alternatively consider a re-scaled variant the additive kernel $\bar{K}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{D}\sum_{d=1}^D K_d(x_d, y_d)$. Consequently, we define the element-wise feature map as $\bar{\phi}(x_{d,m}) = \frac{1}{\sqrt{D}}\phi(x_{d,m})$. The following theorem establishes the expectation and concentration bounds on the approximation error of ANNFM.

**Theorem 1.** *Given a feature index $d$. Assume that the anchor set $\hat{\boldsymbol{X}}_d$ is an $\epsilon$-cover of $\boldsymbol{X}_d$ and Assumption 1 holds.*

*(a) The expectation of approximation error is bounded by*

$$\mathbb{E}\left[\bar{K}(\boldsymbol{x}_m, \boldsymbol{x}_{m'}) - \bar{\phi}(\boldsymbol{x}_m)^\top \bar{\phi}(\boldsymbol{x}_{m'})\right] \leq O(\epsilon^2).$$

*(b) Moreover, assume that $\delta_{d,m}$ are independent sub-Gaussian random variables with parameters $\sigma$. Then for any $\eta \in (0, 1)$ and $\varepsilon = 2L\sigma\sqrt{\frac{\ln(2/\eta)}{D}}$, with probability at least $1 - \eta$ we have*

$$\left|\bar{K}(\boldsymbol{x}_m, \boldsymbol{x}_{m'}) - \bar{\phi}(\boldsymbol{x}_m)^\top \bar{\phi}(\boldsymbol{x}_{m'})\right| \leq \varepsilon + O(\epsilon^2).$$

A proof of this theorem is provided in Appendix A. The main message conveyed by the part(a) of this theorem is that if $\hat{\boldsymbol{X}}_d$ is an $\epsilon$-cover of $\boldsymbol{X}_d$, then under the stochastic model (6) the expectation of approximation error of ANNFM is bounded by $O(\epsilon^2)$. The part(b) further establishes that for any $\varepsilon > 0$, if $D$ is sufficiently large, then with overwhelming probability the approximation error of ANNFM is bounded by $\varepsilon + O(\epsilon^2)$. Figure 2 visualizes the kernel matrices of four popular additive kernels and their corresponding approximated ones by ANNFM. These visualization results confirm our theoretical prediction that ANNFM is an accurate approximation to the original nonlinear kernel.

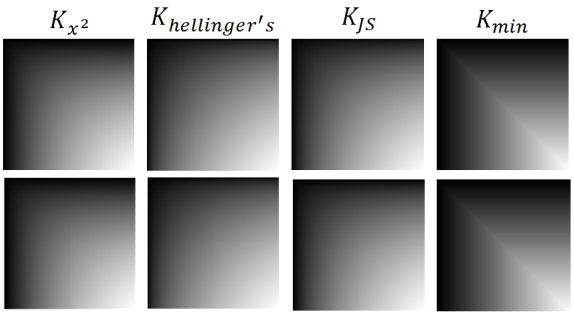


Figure 2. Heatmaps of four additive kernels, $K_{hellinger's}$, $K_{\chi^2}$, $K_{min}$, $K_{JS}$, over the interval $[0, 1]$. The first row shows the exact kernels, the second row shows the approximated kernels by ANNFM when the number of anchor points $N$ is 100.

## 5. Experiments

To evaluate the performance of ANNFM, we conduct experiments on three benchmarks: MNIST, Caltech-101 and DaimlerChrysler pedestrian. We compare ANNFM, in terms of training time and testing accuracy, with the corresponding exact kernels, the approximate feature maps by Vedaldi and Zisserman (VZ) [27], the sign-stable random projections by Li *et al*. [18], and the feature maps by Maji and Berg (MB) [19]. Parameters from these methods were tuned explicitly following their papers. For the feature maps based methods, we use Liblinear [9] to train the resultant linear SVMs, whilst for those baselines using exact kernels we use Libsvm [7] to train kernel SVMs. On MNIST, we conventionally use the low level image features for training. On Caltech-101 and DamilerChrysler datasets, we extract some improved dense SIFT descriptors as image features.

**Implementation Details:** There are three popular additive kernels involved in our experiments: $\chi^2$ kernel, Jensen-Shannon (JS) kernel, and Intersection kernel. Since all these kernels are evaluated on histograms, we mainly resort to the strategy of uniform partition as defined in (2) to construct the covering set. However, in order to evaluate the $k$-means-type partition (3), we also report the experimental results with the two strategy on MNIST dataset. We evaluate both ANNFM and 2-ANNFM in our experiments. The off-line results indicate that 2-ANNFM works favorably to $k$-ANNFM with $k \geq 3$.

Recall that when the anchor set kernel matrix $\hat{\boldsymbol{K}}_d \succ 0$, the aforementioned feature maps $\phi(\hat{x}_{d,n})$ in § 3.3 will have dimensionality $r = N$ which could be high for large $N$. To avoid such an undesirable high-dimensional expansion, in the following experiments we propose to only preserve the $r'$ elements of $\phi(\hat{x}_{d,n})$ corresponding to the top $r'$ eigenvectors of $\hat{\boldsymbol{K}}_d$ that preserve an overwhelming portion, e.g., 99%, of the total spectral energy. Our numerical experience indicates that such a simple trick can substantially speed up the processing in many cases without sacrificing accuracy.

| feat. | dm. | $\chi^2$ acc. (%) | $\chi^2$ time (s) | JS acc. (%) | JS time (s) | inters. acc. (%) | inters. time (s) |
|---|---|---|---|---|---|---|---|
| exact | - | 94.18 | 661 | 93.98 | 925 | 94.05 | 897 |
| VZ | 3 | 93.86 | 6.5 | 93.67 | 6.4 | 93.96 | 6.9 |
| | 5 | 93.83 | 10.3 | 93.67 | 9.6 | 93.92 | 9.9 |
| MB | 10 | - | - | - | - | 91.39 | 8.4 |
| | 20 | - | - | - | - | 91.42 | 16.9 |
| Li | 1k | 93.86 | 7.5 | - | - | - | - |
| | 2k | 93.83 | 13.9 | - | - | - | - |
| AN. | 30 | 93.26 | 2.7 | 93.14 | 2.8 | 93.52 | 16.1 |
| | 40 | 93.84 | 2.6 | 93.67 | 2.6 | 94.05 | 18.5 |
| | 50 | 93.98 | 2.6 | 93.66 | 2.5 | 94.01 | 19.6 |
| 2-AN. | 30 | 93.53 | 4.1 | 93.27 | 4.3 | 93.75 | 24.8 |
| | 40 | 94.06 | 4.0 | 93.81 | 4.0 | 94.11 | 27.2 |
| | 50 | 94.12 | 4.0 | 93.79 | 3.9 | 94.06 | 30.7 |

Table 1. Performance of the considered methods on MNIST. The average classification accuracy (%) and training time (s) are reported. The table compares the exact additive kernels (i.e., $\chi^2$, Intersection, and JS) and the approximations to them: VZ [27], MB [19], and Li [18]. We preserve 95% of the total spectral energy when decomposing the kernel matrix $\hat{\boldsymbol{K}}$. As a baseline, the accuracy of linear kernel is $87.2\%$ with training time about 1s.

| dm. | $\chi^2$ acc.(%) | $\chi^2$ time(s) | JS acc.(%) | JS time(s) | inters. acc.(%) | inters. time(s) |
|---|---|---|---|---|---|---|
| 30 | 92.90 | 2.8 | 92.87 | 2.7 | 92.99 | 14.4 |
| 40 | 92.92 | 3.0 | 92.94 | 2.8 | 93.33 | 16.7 |

Table 2. Performance of the proposed method on MNIST when using $k$-means-type partition to construct anchor sets. As the strategy of uniform partition is more appropriate for bounded features, the performance using $k$-means-type partition is inferior to those using uniform partition. Thus, in the following experiments, we only resort to the latter to construct our anchor sets.

We replicate the experiment 5 times with different random split of training and testing images to obtain reliable results. The mean of the per-class recognition rates were recorded for each run. We then report overall recognition rate as the average recognition rate of all the classes. To save space, we abbreviate ANNFM to "AN." and 2-ANNFM to "2-AN." in all the tables.

### 5.1. MNIST

The MNIST handwritten digits dataset [15] contains images of figures 0-9, with 60,000 training and testing examples in 780 dimensions. For each of the 10 classes, 10000 images are used for training and 10000 for testing. We directly use the intensity as image features. Reported time of the approximation methods includes time for feature mapping and training classifier. Detailed comparison results are listed in Table 1 in which the second column gives the dimensionality parameters used in the considered methods.

As shown in Table 1, all the feature mapping methods

can achieve satisfactory results except for MB [20] which is designed for piecewise const or piecewise linear features. It can be observed that ANNFM and 2-ANNFM take only a fraction time of the corresponding exact kernel for training. Also it is substantially faster than VZ [27] and Li [18]. It is noted, however, that we didn't gain much speedup with the Intersection kernel (see the last two columns) as the eigenvalues of $\hat{\boldsymbol{K}}_{inter}$ are observed to be uniformly distributed. Intuitively, the denser the anchor points are, the more accurate ANNFM will be for kernel linearization. In practice, we observed that satisfactory performance can be acquired when the anchor sets are of size $N = 30 \sim 50$. Concerning the storage size of feature maps, ANNFM needs $\sim$0.7M ($N = 30$) to store the training samples, while that size of VZ is $\sim$6.7M ($dm. = 3$).

## 5.2. Caltech-101

The Caltech-101 dataset [10] contains from 31 to 800 images per category. Most images are medium resolution, i.e., about $300 \times 300$ pixels. For each of the 101 classes, 30 images are used for training and at most 20 images for testing as some categories have fewer than 50 images. We optimize all the parameters by 5-fold cross validation. Features are extracted by an improved dense SIFT descriptor [26]. In particularly, we perform an over-segmentation operator on the image firstly, and then apply a saliency detection method [1] to estimate the importance of each segmented region. To keep the same sampling number of local features, dense features along the boundary of the important salient region with dense sampling are extracted as well as inside the region with random sampling according to its area and importance. Figure 3 shows an example of this feature extraction method. In the Caltech-101 dataset, features are extracted every six pixels with SIFT patches at $16\times16$ pixels; these are quantized in a 2048 visual words dictionary learned using $k$-means. Each image is described by a 43,008-dimensional histogram of visual words with $1\times1$, $2\times2$ and $4\times4$ spatial pyramid.

Table 3 lists accumulated training times and accuracies of various methods on Caltech-101. The advantage of the proposed method is obvious on this dataset, with competitive accuracies, we can respectively achieve about $\times4$ speedup with ANNFM and more than $\times2$ speedup with 2-ANNFM over VZ, and $\times10$ speedup over the exact kernels. In the meantime, ANNFM and 2-ANNFM outperform the method of Li *et al*. [18] by a large margin in both accuracy and running time. Concerning the performance on Intersection kernel, our proposed methods are comparable to MB and VZ in accuracy but inferior in speed. To compare the storage size of feature maps, ANNFM needs 13M ($N = 30$), VZ needs 369M ($dm. = 3$) and MB needs 399M ($dm. = 10$) to store the training set.

Figure 3. **Left** panel shows example of non-uniform spatial sampling. **Right** panel shows example of non-uniform spatial sampling + saliency detection. Sampling from the salient boundary are shown in yellow and sampling inside the region are in green.

| feat. | dm. | $\chi^2$ acc. (%) | $\chi^2$ time (s) | JS acc. (%) | JS time (s) | inters. acc. (%) | inters. time (s) |
|---|---|---|---|---|---|---|---|
| exact | - | 87.45 | 11068 | 86.62 | 13773 | 85.62 | 9748 |
| VZ | 3 | 86.53 | 1238.3 | 86.17 | 1240.1 | 86.12 | 1220.1 |
| | 5 | 87.03 | 1957.0 | 86.29 | 1933.3 | 87.17 | 1957.5 |
| MB | 10 | - | - | - | - | 86.29 | 402.7 |
| | 20 | - | - | - | - | 86.28 | 794.57 |
| Li | 40k | 84.93 | 716.5 | - | - | - | - |
| | 60k | 85.64 | 1090.4 | - | - | - | - |
| | 80k | 85.87 | 1893.5 | - | - | - | - |
| AN. | 20 | 86.52 | 311.1 | 86.58 | 313.9 | 86.35 | 1665.9 |
| | 30 | 87.0 | 294.4 | 86.88 | 317.5 | 86.41 | 2880.1 |
| 2-AN. | 20 | 86.68 | 486.4 | 86.72 | 521.7 | 86.5 | 2579.1 |
| | 30 | 87.09 | 443.9 | 87.0 | 473.4 | 86.62 | 4176.8 |

Table 3. Performance of the considered methods on Caltech-101. As a baseline, the accuracy of linear kernel is 76.28% and its training time about 200s.

## 5.3. DaimlerChrysler Pedestrian

This dataset is created by Munder and Gavrila [21], and is split into five disjoint sets, three for training and two for testing. Each training set has 5,000 positive and negative examples each, while each test set has 4,900 positive and negative examples each. The task is to discriminate $18 \times 36$ gray-scale image patches portraying a pedestrian (positive samples) or clutter (negative samples). Each classifier is trained using two out of three training sets at a time and one of the test sets. We use the same improved dense SIFT features as used in Caltech-101.

As shown in Table 4, ANNFM, 2-ANNFM and the VZ perform comparably to the exact kernels in testing accuracy. ANNFM is the fastest one among all the considered methods when $\chi^2$ and JS kernels are used. Again, it is observed that both ANNFM and 2-ANNFM are slightly inferior to VZ on the Intersection kernel. This is partially due to the fact that the eigenvalues of $\hat{\boldsymbol{K}}_{inter}$ are uniformly distributed. It is observable that even using the one nearest

| feat. | dm. | $\chi^2$ acc. (%) | $\chi^2$ time (s) | JS acc. (%) | JS time (s) | inters. acc. (%) | inters. time (s) |
|---|---|---|---|---|---|---|---|
| exact | - | 92.29 | 527 | 92.21 | 1687 | 92.63 | 463 |
| VZ | 3 | 92.70 | 12.4 | 92.79 | 12.9 | 92.78 | 11.9 |
| | 5 | 92.75 | 20.4 | 92.79 | 18.6 | 92.77 | 21.4 |
| MB | 10 | - | - | - | - | 91.41 | 49.0 |
| | 20 | - | - | - | - | 91.67 | 94.7 |
| Li | 6k | 88.55 | 110.1 | - | - | - | - |
| | 12k | 92.86 | 206.1 | - | - | - | - |
| AN. | 30 | 92.07 | 2.9 | 92.16 | 3.2 | 92.45 | 25.6 |
| | 50 | 92.69 | 6.9 | 92.24 | 8.5 | 92.37 | 34.5 |
| 2-AN. | 30 | 92.24 | 4.2 | 92.35 | 4.6 | 92.75 | 42.6 |
| | 50 | 92.83 | 10.5 | 92.48 | 13.2 | 92.59 | 51.3 |

Table 4. Performance of the considered methods on Daimler-Chrysler Pedestrian. We preserve 99% of the total spectral energy.

neighbor anchor point, ANNFM still outperforms MB and Li's feature maps in most cases. Concerning the storage efficiency, we again observe that ANNFM (21M, $N = 50$) is more storage efficient than VZ (41MB, $dm. = 3$) and MB (43M, $dm. = 10$).

## 6. Conclusion

In this paper, we presented ANNFM as a nearest neighbor search based feature mapping approach for additive kernels linearization. ANNFM is a general framework applicable to any additive kernels whose components are Mercer kernels. Theoretical analysis showed that the approximation error bound of ANNFM is controlled by the covering quality of the constructed anchor points. The obtained feature maps can be compactly stored by a group of nearest neighbor indication vectors (which are sparse) along with the anchor feature maps. Extensive experiments on real-world datasets confirmed that ANNFM is superior or comparable to the state-of-the-art feature mapping methods in accuracy and efficiency. We have also proposed and investigated $k$-ANNFM as a variant of ANNFM based on $k$-nearest neighbor search. Empirical results demonstrated that 2-ANNFM gains slight improvement over ANNFM in accuracy, while the former is slower and more storage demanding than the latter. To conclude, ANNFM is a computationally efficient and theoretically sound framework for additive kernel linearization.


## Acknowledgement

We would like to thank the three anonymous reviewers for their constructive comments on this paper. This work was supported in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK2012045 and Grant BK20141003, in part by the National Natural Science Foundation of China under Grant 61402232, Grant 61532009, Grant 61522308, and Grant 61328205.


## A. Proof of Theorem 1

*Proof.* **Part(a):** From the definition of NNAMF in (4) we have that

$$\begin{aligned}
&\bar{\phi}(\boldsymbol{x}_m)^\top \bar{\phi}(\boldsymbol{x}_{m'}) \\
= \;& \frac{1}{D}\sum_{d=1}^{D} \phi(\mathcal{N}(x_{d,m}))^\top \phi(\mathcal{N}(x_{d,m'})) \\
= \;& \frac{1}{D}\sum_{d=1}^{D} K_d(\mathcal{N}(x_{d,m}), \mathcal{N}(x_{d,m'})) \\
= \;& \frac{1}{D}\sum_{d=1}^{D} K_d(x_{d,m} - \delta_{d,m}, x_{d,m'} - \delta_{d,m'}) \\
= \;& \frac{1}{D}\sum_{d=1}^{D} K_d(x_{d,m}, x_{d,m'}) + \frac{\partial K_d}{\partial x}|_{x_{d,m}}\delta_{d,m} \\
& + \frac{\partial K_d}{\partial y}|_{x_{d,m'}}\delta_{d,m'} + \triangle K_d(x_{d,m}, x_{d,m'}; -\delta_{d,m}, -\delta_{d,m'}) \\
= \;& \bar{K}(\boldsymbol{x}_m, \boldsymbol{x}_{m'}) + \frac{1}{D}\sum_{d=1}^{D} \frac{\partial K_d}{\partial x}|_{x_{d,m}}\delta_{d,m} \\
& + \frac{\partial K_d}{\partial y}|_{x_{d,m'}}\delta_{d,m'} + \triangle K_d(x_{d,m}, x_{d,m'}; -\delta_{d,m}, \delta_{d,m'}) \\
= \;& \bar{K}(\boldsymbol{x}_m, \boldsymbol{x}_{m'}) + \frac{1}{D}\sum_{d=1}^{D} \frac{\partial K_d}{\partial x}|_{x_{d,m}}\delta_{d,m} \\
& + \frac{\partial K_d}{\partial y}|_{x_{d,m'}}\delta_{d,m'} + O(\delta_{d,m}^2 + \delta_{d,m'}^2). \qquad (7)
\end{aligned}$$

where the second "=" follows from the fact that $\phi(\mathcal{N}(x_{d,m}))$ is an exact feature map and the fourth "=" follows from the definition of $\triangle K_d$ in Assumption 1. Since $\hat{\boldsymbol{X}}_d$ is an $\epsilon$-cover of $\boldsymbol{X}_d$, it holds that $\mathbb{E}[\delta_{d,m}^2] \leq \epsilon^2$ for all $d$. Recall $\mathbb{E}[\delta_{d,m}] = 0$. By taking expectation operation on both sides of the preceding equality and with proper rearrangement we get

$$\begin{aligned}
& \mathbb{E}\left[\bar{K}(\boldsymbol{x}_m, \boldsymbol{x}_{m'}) - \bar{\phi}(\boldsymbol{x}_m)^\top \bar{\phi}(\boldsymbol{x}_{m'})\right] \\
= \;& O\left(\frac{1}{D}\sum_{d=1}^{D} \delta_{d,m}^2 + \delta_{d,m'}^2\right) \leq O(\epsilon^2).
\end{aligned}$$

This proves the part (a) of the theorem.

**Part(b):** Let us denote $\tilde{\delta}_{d,m} = \frac{\partial K_d}{\partial x}|_{x_{d,m}}\delta_{d,m}$ appeared in (7). Since $\delta_{d,m}$ are sub-Gaussian variables with parameter $\sigma$ and $\|\nabla K_d\| \leq L$, the terms $\tilde{\delta}_{d,m}$ are also independent sub-Gaussian with parameter $L\sigma$. It can be verified by Chernoff bound that

$$\mathbb{P}\left(\left|\frac{1}{D}\sum_{d=1}^{D} \tilde{\delta}(d,m) + \tilde{\delta}(d,m')\right| > \varepsilon\right) \leq 2\exp\left\{-\frac{D\varepsilon^2}{4L^2\sigma^2}\right\}.$$

The second claim in the theorem is then obtained by solving the inequality $\eta = 2\exp\left\{-\frac{D\varepsilon^2}{4L^2\sigma^2}\right\}$. □

## References


[1] R. Achanta and S. Susstrunk. Saliency detection for content-aware image resizing. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1005–1008. IEEE, 2009.

[2] E. G. Băzăvan, F. Li, and C. Sminchisescu. Fourier kernel learning. In *Computer Vision–ECCV 2012*, pages 459–473. Springer, 2012.

[3] A. Bhrawy and A. Alofi. The operational matrix of fractional integration for shifted chebyshev polynomials. *Applied Mathematics Letters*, 26(1):25–31, 2013.

[4] L. Bottou and C.-J. Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.

[5] O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[6] M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[7] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[8] L. De Lathauwer, B. De Moor, J. Vandewalle, and B. S. S. by Higher-Order. Singular value decomposition. In *Proc. EUSIPCO-94, Edinburgh, Scotland, UK*, volume 1, pages 175–178, 1994.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[11] C. Figuera, Ó. Barquero-Pérez, J. L. Rojo-Álvarez, M. Martínez-Ramón, A. Guerrero-Curieses, and A. J. Caamaño. Spectrally adapted mercer kernels for support vector nonuniform interpolation. *Signal Processing*, 94:421–433, 2014.

[12] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[13] E. G. Hohenstein and C. D. Sherrill. Density fitting and cholesky decomposition approximations in symmetry-adapted perturbation theory: Implementation and application to probe the nature of $\pi$-$\pi$ interactions in linear acenes. *The Journal of Chemical Physics*, 132(18):184111, 2010.

[14] P.-S. Huang, L. Deng, M. Hasegawa-Johnson, and X. He. Random features for kernel deep convex network. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3143–3147. IEEE, 2013.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] F. Li, C. Ionescu, and C. Sminchisescu. Random fourier approximations for skewed multiplicative histogram kernels. In *Pattern Recognition*, pages 262–271. Springer, 2010.

[17] F. Li, G. Lebanon, and C. Sminchisescu. Chebyshev approximations to the histogram $\chi$ 2 kernel. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2424–2431. IEEE, 2012.

[18] P. Li, G. Samorodnitsky, and J. Hopcroft. Sign stable projections, sign cauchy projections and chi-square kernels. *arXiv preprint arXiv:1308.1009*, 2013.

[19] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 40–47. IEEE, 2009.

[20] S. Maji, A. C. Berg, and J. Malik. Efficient classification for additive kernel svms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):66–77, 2013.

[21] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11):1863–1868, 2006.

[22] F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2297–2304. IEEE, 2010.

[23] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[24] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[25] V. Vapnik. *Estimation of dependencies based on empirical data*. Springer Verlag, 1982.

[26] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472. ACM, 2010.

[27] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):480–492, 2012.

[28] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*. Citeseer, 2001.

[29] J. Yang, V. Sindhwani, H. Avron, and M. Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. In *Proceedings of The 31st International Conference on Machine Learning*, pages 485–493, 2014.

[30] J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. W. Mahoney. Random laplace feature maps for semigroup kernels on histograms.

[31] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.

[32] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.