Feature Similarity: Geometrical Framework and Discriminative Kernel Learning

Canh Hao Nguyen, Tu Bao Ho

February 20, 2009
KS-RR-2009-001

Feature Similarity: Geometrical Framework and Discriminative Kernel Learning

Canh Hao Nguyen, Tu Bao Ho
{canhhao,bao}@jaist.ac.jp
School of Knowledge Science
Japan Advanced Institute of Science and Technology
1-8 Asahidai, Nomi, Ishikawa, 923-1211, Japan

February 20, 2009

Abstract

We propose geometrical models of features of a learning problem. We show that there are two alternative ways to interpret similarities between features. The syntactic way is well studied and applied in feature selection while the semantic way, in the context of kernel methods, is less understood. We show that the latter is equivalent to feature semantic similarity (FSS) and there are a number of methods that fall into this framework. We analyze to show relations among these methods and differences to feature selection ones. Our analysis shows a natural extension to all these methods. It automatically suggests that this framework can be applied in a general context.

We also note that all the methods using FSS are inherently unsupervised in nature. None of these methods make use of labels for classification or regression tasks. On the other hand, the feature selection counterparts consider labels as an important information. Therefore, we propose an algorithm to learn the feature proximity matrix for FSS for supervised tasks. We show the merit of our algorithm in various applications.

1 Introduction

With the understanding that kernel matrix should contain all the necessary information for learning method to exploit, it is crucial that a kernel matrix must carry the information of the semantics of the problem at hand. However, it is always difficult to incorporate background knowledge of the problem into kernels for two reason. First, we do not know what is the relevant information that could be helpful to kernel methods. Second, it is not always possible to incorporate some information into kernels, as kernels require it to be represented as dot products in some space.

There are various kernels proposed in practice, each of them, either implicitly or explicitly, incorporates some background knowledge in a very specific form. The background knowledge can be seen clearly from non-vectorial data. Examples can be seen as in discrete structures, substructures are used with the understanding that the semantics of substructures carries the semantics of the whole object (chemical interactions occur at substructures,

protein interactions happen at subsequences of amino acids). Knowing the data lying in a small subspace, kernels can be constructed with low rank using kernel PCA [19]. Assuming a latent semantics of data, kernels should represent the data like that in latent semantic analysis [3]. Feature selection [11], as a general tool for data preprocessing, should be used before kernels are computed. Having known a generative process of the data, kernels should take that into account to construct kernels such as Fisher kernels [7] or marginalized kernels [?]. Knowing a stochastic process on the graph constructed from data objects, kernel such as diffusion kernels [10], von Neumann kernels [?] are to used to take that knowledge into kernel construction.

One of the phenomena that has not received enough attention is that in data described by features, it is possible that features have certain semantic similarity among them. This phenomenon could be seen from textual semantic domain where words has semantic relations. In image domain, locations of pixels describing the images should reflect pixels' similarity, closer pixels should contribute more to image similarity given that images' similarity is made to be robust to small changes.

In this work, we propose two geometrical ways to interpret feature similarity (section 2) in the context of kernel methods [18]. Both these interpretations relied on the non-orthogonality of a basis (used to explain features geometrically) to encode similarity among features. While one is useful and well applied to feature selection, the other received a limited attention but equivalent to FSS. We show that there are many methods that fall into the latter framework. We draw its relation with Fisher kernels (section 3) and a probabilistic interpretation. With the notion of feature proximity matrix (FPM), we show similarities and differences between the methods in this framework (section 4), such as empirical kernels, von Newmann kernels, diffusion kernels, etc. We also show the difference to feature selection methods (section 5). We then propose a general kernel taking feature similarity into account with discriminative information of labels for classification (section 6). It has the property of the very first kernel with feature similarity based on label information (section 6). We present some experiments afterward on a simulation data and Information Retrieval data to show the merit of the general kernels and the merit of the framework itself (section 7). Conclusion comes at last.

2 A Geometrical Model for Feature Semantic Similarity

Given a training set $X = \{x_i\}_{i=1}^n$, where $x_i = (x_{i_1}, x_{i_2}, \cdots, x_{i_d})^T \in \mathbb{R}^d$ ($x_{i_j} \in \mathbb{R}$). We abuse the notion to denote X as the matrix of data as well, namely $X = (x_1|x_2|\cdots|x_n) \in \mathbb{R}^{d*n}$, each data point is a column in X. Denote the set of indices of features is $D = \{1, 2, \cdots, d\}$ and $I \subseteq D$ as a subset of D. Assume that we have label data, namely $Y = (y_1, y_2 \cdots, y_n)^T \in \mathbb{R}^n$ is the desired response vector. In case $y_i \in \{+1, -1\}$, we have a binary classification problem.

Denote $x_i^{'} = (x_{i_j}, \cdots, x_{i_k})^T$, $j \cdots, k \in I$ be the projection of x_i in the subset of indices I. Feature selection problem is to choose I such that the new data matrix $X^{'} = \{x_i^{'}\}_{i=1}^n$ gives a better estimation of the desired response (target) vector Y. A more general problem is to give a weight to each feature differently for a better estimation. It is to find

a vector $w = (w_1, w_2, \cdots, w_d)^T \in \mathbb{R}^d$, usually $w_i \geqslant 0$ such that $X' = \{x_i'\}_{i=1}^n$, where $x_i' = (w_1 x_{i_1}, w_2 x_{i_2} \cdots, w_n x_{i_d})^T$, gives a better estimation of the target vector Y.

2.1 Feature Interpretations

In this work, we take the interpretation of features as follows. We consider a feature as a measurement of a data object in some linear spaces. Suppose the training data $\S_i$ is in a linear space S. The k^{th} measurement of $\S_i$ is, in our model, a dot product of the training example $\S_i$ with a basic vector e_k of S. It is

$$x_{i_k} =< \S_i, e_k >_S . \tag{1}$$

Denote the matrix of all basis vectors $E = (e_1, e_2 \cdots e_d)$, (also $E = (e_1|e_2| \cdots e_d)$ in matrix form). Then the training data recorded is $x_i = E^T \cdot \S_i \in \mathbb{R}^d$.

Without preprocessing on features, one can assume that all features are independent. This is equivalent to assuming that all e_i are orthogonal, that is

$$< e_i, e_j >_S = \delta_{ij}, \tag{2}$$

where δ_{ij} is the Kronecker's *delta* notion. $\S$ is then estimated as

$$\S_i \cong \tilde{x_i} = \sum_{k=1}^{d} x_{i_k} e_k. \tag{3}$$

When E is orthogonal, (1) and (3) are equivalent. However, they are not the same if E is not orthogonal. There is a difference between interpretation in (1) and (3) that we exploit to provide a new model for feature similarity later. The difference can be seen from a linear kernel constructed using the two interpretations. The difference is illustrated in figure 1. We name the interpretation in (1) as *syntactic similarity*[1] and in (3) as *semantic similarity*. Mathematically, the two interpretations can be switched from one to another with appropriate linear transformations. However, the key difference is at the practical implications on the relations with basic vectors: in (1) feature values come from *projection*, as a kind of sensing, of data onto the basic vectors while in (3), feature values are used for the *reconstruction* of data from the basic vectors.

It is noteworthy that we do not claim which interpretation is correct. Rather, these are two *alternative* interpretations that are suitable to different problems. In fact, we will analyze to show that many other methods using feature similarity implicitly, follow either one of these interpretations.

[1] See section 5 for an explanation.

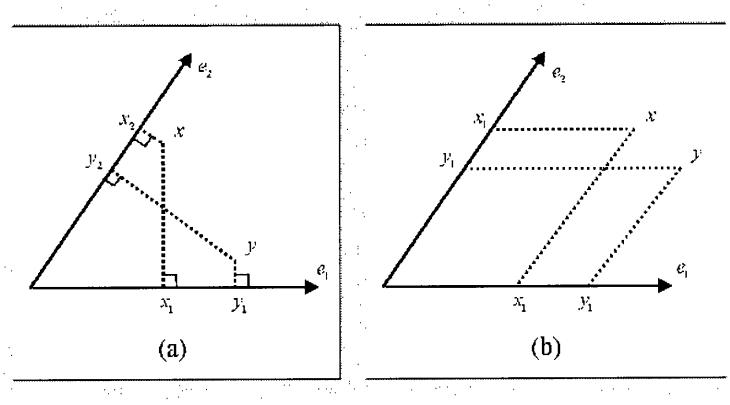

Figure 1: The difference between interpretations in a non-orthogonal basis. We show vectors $x = (x_1, x_2)^T$ and $y = (y_1, y_2)^T$ in (a): projection interpretation in (1) and (b): reconstruction interpretation in (3).

2.2 Feature Similarity and Kernels

According to the interpretation in (3), linear kernels [24, 18, 20] in Support Vector Machines are also estimated as

$$\begin{aligned} k(\S_i, \S_j) &= < \S_i, \S_j >_S \\ &\cong < \tilde{x_i}, \tilde{x_j} >_S \\ &= < \sum_{k=1}^{d} x_{i_k} e_k, \sum_{k=1}^{d} x_{j_k} e_k >_S \\ &= < x_i, x_j >_{\mathbb{R}^d} . \end{aligned}$$

The last equation is valid only if (2) holds true.

However, the assumption of orthogonality of E, equivalently independence of features is sometimes too strong as we do not know anything about the process of generating these features. Feature weighting (also feature selection) is an attempt to break this assumption. The basic vectors are not assumed to have length 1 (but still orthogonal to each other). That is to assume $< e_i, e_i >_S = w_i^2$ and $< e_i, e_j >_S = 0$ for $i \neq j$. The effect of this can be seen from the linear kernel matrix

$$\begin{aligned} < \S_i, \S_j >_S \cong & < \sum_{i=1}^{d} x_i e_i, \sum_{i=1}^{d} x_j e_j > \\ &= \sum_{k=1}^{d} w_k x_{i_k} \cdot w_k x_{j_k}. \end{aligned}$$

This is the dot product of the two vectors after feature weighting. Feature selection is to set $w_i = 1$ or $w_i = 0$, and is a special case of feature weighting.

In reality, feature weighting cannot represent certain kinds of relationship among features. One of the examples is semantic relation between words. Synonymy is a problem when two words have the same meaning in a given context [13, 8]. In a unigram language model, two words are considered two orthogonal dimensions. In the above interpretation, their basic vectors coincide. For example, the two basic vectors are $e_i = e_j$. The bag of word representations of two documents $(\cdots n_i, \cdots n_j, \cdots)^T$ and $(\cdots n_i^{'}, \cdots n_j^{'}, \cdots)^T$, which differ at only the i^{th} and j^{th} positions, would be semantically the same if $n_i + n_j = n_i^{'} + n_j^{'}$. The feature weighting techniques such as TFIDF [16, 21] would fail to recognize the synonyms. This means that document representation should be unchanged if synonym words are used interchangeably. This relationship is not representable by feature weighting.

When E is non-orthogonal, i.e. the features are semantically similar, to represent dot-products (kernels), it is sufficient to use the so-called *feature proximity matrix* (FPM). Feature proximity matrix is the matrix of pairwise dot-product between basic vectors e_i, denoted as

$$F = \{f_{ij}\}_{i,j=1}^{d} = \{\langle e_i, e_j \rangle\}_{i,j=1}^{d}. \tag{4}$$

The benefit of having F is that when computing dot product of two training examples, it does not concern the space that training examples lie in, or the basis on which measurements are carried out. In other words, one does not have to define explicitly what are the space S and the basis $E = \{e_1, e_2 \cdots e_d\}$. Instead, one can set the matrix F directly. This resembles the requirements of a kernel matrix of data objects [24]. For example, for $x = \{x_1, x_2 \cdots x_d\}^T$ and $x^{'} = \{x_1^{'}, x_2 \cdots x_d^{'}\}^T$, then

$$\begin{aligned}
\langle x, x^{'} \rangle_S &= \langle \sum_{i=1}^{d} x_i e_i, \sum_{i=1}^{d} x_i^{'} e_i \rangle \\
&= \sum_{i,j=1}^{d} x_i x_j^{'} \langle e_i, e_j \rangle \\
&= \sum_{i,j=1}^{d} x_i x_j^{'} f_{ij} \\
&= x^T F x^{'}.
\end{aligned} \tag{5}$$

One again, the difference between interpretations can be seen clearly from the construction linear kernels. As oppose to the above formula, dot product using the matrix F as:

$$\langle x, x^{'} \rangle_S = x^T F^{-1} x^{'},$$

where F^{-1} is the (possibly pseudo-) inverse of$F$.

The feature proximity matrix must be positive semidefinite, as the way it is defined to be. This also resembles the kernel matrix of data objects [24]. When $F = I$, a dot product is in the usual sense as in Euclidean space $\mathbb{R}^n$ with an orthogonal basis. When F is diagonal, it is equivalent to weighting features. To model similarity among features, F can be any positive

semidefinite matrix. By setting no restriction on F rather than positive semidefiniteness, this framework can add additional information.

Proposition 1. *All possible dot products on $\mathbb{R}^d$ can be represented as $\langle \cdot, A\cdot \rangle$.*

Proof can be found on any Linear Algebra textbook such as [?]. The implication of this theorem is as follows. All linear kernels on $\mathbb{R}^d$ imply some feature similarities in some linear spaces in the interpretation of this framework. This completes the picture of equivalent relations between three concepts: feature similarity, generalized linear kernels and the dot products of the form (5). It is noteworthy that [3], rooted in textual semantics, using LSA, also follows this form. The key difference in our work is that it is motivated from a general geometrical point of view. This is of vital importance as by coming from a general framework, we can safely induce that this framework is applicable to general context application, not textual semantics alone. This is later on discussed in the feature extraction subsection 4.3.

We take some simple examples to illustrate the utility of feature semantic similarity.

Example 1. When numerize ordinal data attributes for kernel methods, for example *low, normal, high, very-high*, each ordinal value usually becomes a new feature. By taking the order of values into account, it is necessary to say that *high* and *very-high* are semantically close. One can ensure this by having a high semantic similarity between them.

Example 2. Given the data set $X = \{x_1, x_2, x_3\}$ where $x_1 = (1,0,0)^T$, $x_2 = (0,1,0)^T$ and $x_3 = (0,0,1)^T$. Traditionally, as $\langle x_i, x_j \rangle = 0$, there is no similarity information encoded in this data set. If we know a priori that the first two features e_1 and e_2 are semantically close, then we wish to say that x_1 and x_2 are similar to each other, but not to x_3. The reason is that x_1 and x_2 score 1 in one of the two semantically similar features, then they should have some similarity. One of the way to encode this similarity information is to use the feature proximity matrix F. For example, take

$$F = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

,

then $x_1^T F x_2 = 1$, $x_1^T F x_3 = x_2^T F x_3 = 0$. It is evident that by forcing $\langle e_1, e_2 \rangle > 0$, we can model the similarity between features e_1 and e_2.

Example 3. In measuring similarity between images for recognition and retrieval, one important requirement of the similarity measure is that it must be robust to small changes in images. A dot in either of the neighboring pixels would give a visually similar perception. However, when considering an image as a vector and a pixel as a independent dimension, the representation does not reflect neighboring pixels to be semantically similar. Therefore, if we set the feature proximity matrix to have high similarity between neighboring pixels (features), the representation would be robust to small change in neighboring pixels. For example, in figure 2, images (a) and (b) should have a higher similarity to each other than to image (c).

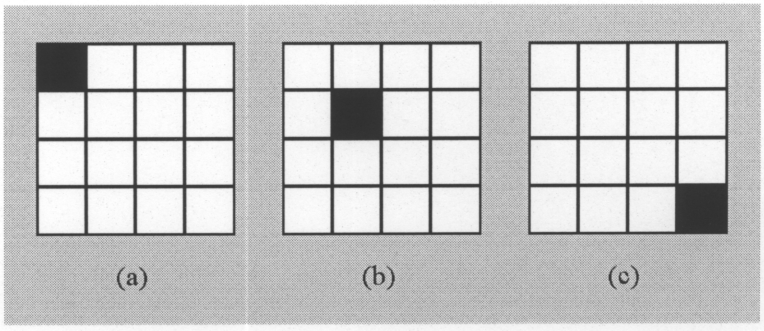

Figure 2: Image similarity: the images (a) and (b) should be more similar to each other than to (c). In vector representation, the three images have the same similarity to each other. By enforcing nearby pixels to be similar, images (a) and (b) can have a higher similarity.

3 Feature Semantic Similarity and Fisher Kernels

The proposed framework holds a tight connection with feature extractors from generative models, namely Fisher kernels [7]. The way to compute kernels immediately shows the resemblance between Fisher scores and measurements of an object. Moreover, the roles of the inverses of Fisher information matrices [2] and kernel matrices of features is to model correlations among Fisher scores or features. This already suggests that the proposed framework would be interpreted as Fisher kernels with some generative models. It also suggests that Fisher scores can be interpreted as projection of objects in some spaces onto appropriate basic vectors.

One of the ways to exploit generative models to learn kernels is proposed in the framework of Fisher kernels [7]. In this framework, Fisher scores are used as feature extractors for discriminative learning. A data object in a parametric probabilistic model, denoted $q(x|\theta)$, when the parameter is estimated as $\tilde{\theta}$, is measured by Fisher scores

$$f(x) = (\frac{\partial \ln p(x|\tilde{\theta})}{\partial \theta_1}, \frac{\partial \ln p(x|\tilde{\theta})}{\partial \theta_2}, \cdots \frac{\partial \ln p(x|\tilde{\theta})}{\partial \theta_d})^T.$$

Kernels induced from a generative model are based on the dot products while taking into account similarity among parameters of the generative model. It is defined to be

$$K(x_i, x_j) = f(x_i)^T I^{-1}(\theta) f(x_j) \tag{6}$$

where $I(\theta)$ is the Fisher information matrix of the generative model.

The formula (5) resembles (6). It suggests that the original feature values can be interpreted as Fisher score of some family of distributions. Denote that $x = \{x^{(1)}, x^{(2)}, \cdots x^{(d)}\}^T$

$$x^{(i)} = \frac{\partial \ln p(x|\theta)}{\partial \theta_i}.$$

Therefore

$$\ln p(x|\theta) = \theta_i x_i + g(x)$$
$$p(x|\theta) = h(x)exp\{\theta^T x\}$$

for some $g(x)$ and $h(x)$. The last equation is in form of an exponential family [4]. This shows that feature values can be interpreted as Fisher scores of the exponential family of multinomial distributions [1].

The difference from the Fisher kernel point of view to our framework is that we do not require a parametric distribution to be assumed beforehand. It gives the flexibility to choose a suitable distribution for the task at hand. Instead, we can just learn the Fisher information matrix.

From feature semantic similarity point of view, Fisher kernel is a special case where $F = I^{-1}(\theta)$. Recall that

$$I(\theta)_{ij} = E[\frac{\partial \ln p(x|\tilde{\theta})}{\partial \theta_i} \cdot \frac{\partial \ln p(x|\tilde{\theta})}{\partial \theta_j}] = E_l[x_{li} \cdot x_{lj}]$$
$$I(\theta) = Cov(X)$$

where $Cov(X)$ is the covariance matrix of X. Therefore, up to a constant, $I^{-1}(\theta)$ acts as the *whitening* operator for the distribution. Fisher kernel is equivalent to the dot-product kernel in the whitened distribution.

4 Analysis of Feature Semantic Similarity

There are some works that similarity among features are implicitly or explicitly exploited in different ways but still be viewed from this framework. In these methods, the feature proximity matrix is set fixed or based on some other criteria. FSS provides a geometrical interpretation and natural extension to all these methods.

4.1 Context vector for feature similarity

For the task of generating bilingual lexicon, similarity among words is learnt using comparable corpora from two languages [6]. The basic assumption behind all those works is that if two words are mutual translations, their collocates (the words that occur at the same location) are likely to be mutual translations as well (for example [15, 22]). On that assumption, standard approaches build context vectors for word pairs from the two languages and then compare context vectors utilizing translations. The problems of synonymy and polysemy can be solved by imposing similarity among features (words) when comparing context vectors.

To consider two words v and w from two languages to be mutual translations or not, context vectors $\vec{v}$ and $\vec{w}$ are generated from comparable corpora. A context vector of a word v is defined to be the vector of association scores between v and a set of words in the context ($e_i \in E$): $\vec{v} = \{a(v, e_1), a(v, e_2), \cdots a(v, e_p)\}^T$. Traditional approaches would compare two

words from two languages by the dot product: $S(v,w) = \sum_{(e,f)\in D} a(v,e)a(w,f)$, where D is a dictionary consisting of bilingual lexicons, used as seeds. It is formulated as:

$$s(v,w) = \langle \overrightarrow{v}, \overrightarrow{tr(w)} \rangle$$

In fact, this is a vector space model of a word by its context vectors with the basis $e_1, e_2, \cdots, e_n$, which is assumed to be orthogonal. Orthogonality means that no relations among basic vectors (words) are considered. To account for the problems of synonymy and polysemy, it is conjectured that the context vector can provide additional information. In [6], it is assumed that the context vector $\overrightarrow{v}$ is actually in the non-orthogonal basis of $(\overrightarrow{s_1}, \overrightarrow{s_2}, \cdots, \overrightarrow{s_m})$, $s_i \in E$ and the set of s_i-s is a subset of all the words in the context. The representation of the context vector of v in the original orthogonal basis spanned by $(s_1, s_2, \cdots, s_m)$ then becomes

$$\overrightarrow{v}' = Q_s \overrightarrow{v}$$

where

$$Q_s = \begin{pmatrix} a(s_1,e_1) & \cdots & a(s_m,e_1) \\ \cdots & \cdots & \cdots \\ a(s_1,e_p) & \cdots & a(s_m,e_p) \end{pmatrix}.$$

We name the matrix Q_s *association* matrix. This is equivalent to:

$$\overrightarrow{v}' = \sum_{i=1}^{p} a(v,e_i) \cdot \begin{pmatrix} a(s_1,e_i) \\ \cdots \\ a(s_m,e_i) \end{pmatrix}$$

In this case, the feature proximity matrix becomes $F = \{k(s_i,s_j)\}_{i,j=1}^{m} = Q_s Q_s^T$ and $k(s_i,s_j) = \langle \overrightarrow{s_i}, \overrightarrow{s_j} \rangle$. From our perspective, the philosophy behind is that if two words v and v' have a similar collocates, they form two basic vectors with large dot-product. The matrix F in this case is set using specific domain knowledge.

4.2 Smoothing and Sharpening

In image processing, smoothing and sharpening mean averaging out pixel intensities to neighboring pixels for some desired effects. In fact, these operators are linear on features (as pixels), therefore also a part of FSS framework. It can be characterized by a linear matrix $A = \{\alpha_{ij}\}_{i,j=1}^{d} \in \mathbb{R}^{d*d}$. In this matrix, α_{ij} encodes the neighborhood between i^{th} and j^{th} features. Smoothing or sharpening operators tranform x to x' as: $x = Ax'$.

Linear kernels on this data after smoothing or sharpening then become

$$\begin{aligned} k(x_i,x_j) &= \langle x_i', x_j' \rangle \\ &= \langle Ax_i, Ax_j \rangle \\ &= x_i^T \cdot (A^T A) \cdot x_j. \end{aligned}$$

This models feature semantic similarity with the similarity of e_i (i^{th} feature) and e_j as $f_{ij} = \sum_{l=1}^{d} \alpha_{il}\alpha_{jl}$.

However, FSS provides a richer family kernel functions on images than what obtained from dot-product kernels with these image operators. The evidence is as follows. Suppose that these operators use a distance-1 pixel neighborhood ($d(i,j) = 1$) as the smallest number of neighboring pixels for some pixel neighborhood function d. Then $\alpha_{ij} \neq 0$ if $d(i,j) \leqslant 1$. Hence, $f_{ij} \neq 0$ if $d(i,j) \leqslant 2$. In other word, by using the operators on a distance-1 pixel neighborhood, it is equivalent to imposing a feature similarity of distance-2 pixel neighborhood. Therefore, setting a distance-1 pixel neighborhood with the FPM is not representable with those image operators. Hence, FSS provides a richer family of kernels than these operators do.

4.3 Feature Extraction Methods

There are a large number of methods to extract a small number of features for a specific task, such as PCA, ICA, CCA and LSA [4]. The methods provide computational efficiency by working with a small set of (new) features, making expensive methods also applicable. They also improve overall quality of the task by preprocessing data beforehand, stripping off irrelevant or noisy information.

In these methods, data in $\mathbb{R}^d$ is projected into a subspace spanned by $\{w_1, e_2, \cdots, w_m\}$ where $w_i \in \mathbb{R}^d, i = 1, m$. The set of w_i is usually orthogonal. Usually, $m << d$. The projection matrix is $P = (w_1, w_2, \cdots, w_m)^T \in \mathbb{R}^{m*d}$ projecting $x_i \in \mathbb{R}^d$ into $Px_i = (w_1^T x_i, w_2^T x_i, \cdots, w_m^T x_i)^T \in \mathbb{R}^m$. Dot product, or linear kernel, in the reduced space is carried out as

$$\begin{aligned} k(x_i, x_j) &= \langle Px_i, Px_j \rangle \\ &= x_i^T P^T P x_j \\ &= x_i (P^T P) x_j. \end{aligned}$$

Therefore, a linear kernel on the subspace after projection has $KF = P^T P$ as the feature proximity matrix. Therefore, $f_{ij} = \sum_{l=1}^{d} w_{l_i} w_{l_j}$ encodes the similarity between i^{th} and j^{th} features. When $i = j$, it becomes a square of the weight of the i^{th} feature.

4.4 Empirical Kernel Maps

It is known that kernel methods map data into a high (sometimes infinite) dimensional space. The mapping induces a natural vector representation of (possibly) infinite dimension via Mercer's theorem [18]. To shed a light on the mapping, sometimes it is useful to approximate the mapping using landmarks. Empirical kernel map [23] represents the mapping by evaluating on training data as follows. Given a set $\{x_1, x_2, \cdots, x_n\}$, the mapping

$$\begin{aligned} \Phi : X &\rightarrow \mathbb{R}^n \\ x &\rightarrow k(., x)|_{\{x_1, x_2, \cdots x_n\}} \\ &= (k(x, x_1), k(x, x_2), \cdots k(x, x_n)) \end{aligned}$$

is called the empirical kernel map w.r.t $\{x_1, x_2, \cdots, x_n\}$.
Kernel evaluation of the empirical kernel k_{emp} is then carried out as

$$k_{emp}(x, x^{'}) = \sum_{i=1}^{n} k(x, x_i) \cdot k(x^{'}, x_i).$$

This dot product is equivalent to endowing the feature space $\mathbb{R}^m$ with a new dot product $\langle \cdot, \cdot \rangle$ [17]. Denote the images of data points in the first kernel map as $\phi(x_1), \phi(x_2) \cdots, \phi(x_n)$, and Q be matrix with these vectors as columns. Then,

$$\begin{aligned} k_{emp}(x, x^{'}) &= \sum_{i=1}^{n} k(x, x_i) \cdot k(x^{'}, x_i) \\ &= \sum_{i=1}^{n} \langle \Phi(x), \Phi(x_i) \rangle \cdot \langle \Phi(x_i), \Phi(x^{'}) \rangle \\ &= \Phi(x)^T \cdot \sum_{i=1}^{n} \Phi(x_i) \Phi(x_i)^T \cdot \Phi(x^{'}) \\ &= \Phi(x)^T \cdot S \cdot \Phi(x^{'}) \end{aligned} \tag{7}$$

Here we denote $S = \sum_{i=1}^{n} \Phi(x_i) \Phi(x_i)^T$ as the unnormalized scatter matrix of the data in the feature space. It is the scatter matrix of the original space if linear kernels are used. The above formula shows that empirical maps, belong to FSS framework where the feature proximity matrix is the unnormalized scatter matrix of the data. Note that an empirical kernel map is, up to a constant, an inverse of a Fisher kernel in terms of feature semantic similarity. Another thing to note is that regardless of data is still in original space or any feature space, the final kernel matrix remains unchanged.

4.5 von Neumann and Semantic Diffusion kernels

In [3], the authors proposed the latent semantic kernels aiming at taking into account semantical similarity between words in vector space models of documents. An automatic learning of those kernels is proposed in [9]. The von Neumann kernel is defined to be a kernel that satisfy the equilibrium between word similarity and document similarity.

$$\begin{aligned} \hat{K} &= K(I - \lambda K)^{-1} \\ &= K + \lambda K^2 + \lambda^2 K^3 + \lambda^3 K^4 + \cdots \\ &= X^T \cdot (I + \lambda S + \lambda^2 S^2 + \cdots) \cdot X \end{aligned}$$

where S is a unnormalized scatter matrix of the data as in (7). By making the weight of high order terms in the von Neumann kernels decaying faster [9], they came to the semantic diffusion kernels [10].

$$\hat{K} = K + \frac{\lambda}{1!}K^2 + \frac{\lambda^2}{2!}K^3 + \frac{\lambda^3}{3!}K^4 + \cdots$$
$$= X^T \cdot (I + \frac{\lambda}{1!}S + \frac{\lambda^2}{2!}S^2 + \cdots) \cdot X$$

These kernels can be casted as linear kernels with feature semantic similarity, where the FPM $F = I + \lambda S + \lambda^2 S^2 + \cdots$ and $I + \frac{\lambda}{1!}S + \frac{\lambda^2}{2!}S^2 + \cdots$, respectively. Empirical kernel maps (4.4) can be considered as a special case in the sense that empirical kernel maps take only the second term in these FPMs.

4.6 Summary

In summary, there are some methods that can be interpreted as special cases of feature semantic similarity where the FPM is hard coded or learn using a totally different objectives. We summarize methods that use feature similarity implicitly and set the feature proximity matrix fixed following other criteria in table 1.

Method	Feature proximity matrix	Note
Context vector	$Q_s Q_s^T$	Q_s: association matrix
Fisher kernels	I^{-1}	I: Fisher information matrix
Smoothing	$A^T A$	A: smoothing operator
Empirical kernels	S	$S = XX^T$, X: data matrix
Feature extractions	$P^T P$	P: projection matrix
von Neumann kernels	$I + \lambda S + \lambda^2 S^2 + \cdots$	$S = XX^T$
Diffusion kernels	$I + \frac{\lambda}{1!}S + \frac{\lambda^2}{2!}S^2 + \cdots$	$S = XX^T$

Table 1: Summary of methods incorporating feature semantic similarity.

As we analyzed and summarized in the table, these methods are special intances our framework of feature semantic similarity. The framework not only gives a natural way to generalize all these methods, it also suggests a natural way to combine these methods for desirable effect. For example, if onw assume a exponential process over the associations of words, context vector can be generated with von Neumann kernels by replacing S in the original von Neumann kernels with $Q_s Q_s^T$. If we assume a diffusion process on the smoothing operators, combining these kernel construction methodology would give a *deeper* smoothing operator.

It is noteworthy that in the methods that learn the feature proximity matrix (the last fourin the table 1), all of them rely on the matrix $S = XX^T$, which can be called uncentralized scatter matrix. Of course, being uncentralized, the matrix S is sensitive data translation in the feature space. By combining higher powers of the matrix S, the effect is to have a higher level of propagation of feature similarity. Also, by using S, kernels for both in and out-of-sample extension can be computed solely on the feature space, meaning that they can be computed based solely on a original kernels without refering to any concrete feature

space. This also means that by using different feature spaces, the matrix S may appear differently, but the final kernels remain the same.

One thing can be learnt from von Neumann and diffusion kernels is that the exponential or diffusion process at the object level is translated to the exponential or diffusion process at feature level. This comes from the dual formulae: $F_{vn} = I + \lambda S + \lambda^2 S^2 + \cdots = I + \lambda S(I - \lambda S)^{-1}$ as opposed to $K_{vn} = K(I - \lambda K)^{-1}$ with only a constant I differnt. Similarly, $F_{df} = I + \frac{\lambda}{1!}S + \frac{\lambda^2}{2!}S^2 + \cdots = exp(\lambda S)$ while $K_{df} = Kexp(\lambda K)$, which a scalar K different in the kernel computation. This is to say that if we assume such processes on objects, it is equivalent to understand that it is the process on feature. Therefore, feature similarity view give a dual interpretation of such processes, which could be useful in applications.

A very important merit of the framework is that it provides a general geometrical view on these methods. An knowledge of all these method that can be taken advantage of is that all these methods are unsupervised in nature. All of them do not take into account label information. However, almost all (if not all) of them are used for supervised purpose. This opens a research direction of making use of label information to learn the feature proximity matrix adaptively for each task, not fixed as in those methods. We consider this is a significant theoretical contribution as it opens a new research direction to many commonly used methods.

5 Feature Similarity versus Feature Selection

Feature selection methods usually make an assumption that some features are redundant [11]. Filtering them out will increase performances on the problem. The notion of feature redundancy is used to describe when one feature does not bring any more information for a task given that a set of other features is already in use. It is usually in form of correlation.

Feature redundancy does not follow the FSS framework. Istead, it makes use of the interpretation in (1). It can be described as follows. Given the feature set $E_1 = e_1, e_2 \cdots, e_k$ and $e_p \notin E_1$. If we assume that e_p is in the span of E_1, then

$$e_p = \sum_{i=1}^{k} \alpha_i e_i, \ \alpha_i \in \mathbb{R}. \tag{8}$$

For any $x \in S$, its representation is $(x_i =< x, e_i >_S)_{i=1}^k$. Then $x_p =< x, e_p >_S =< x, \sum_{i=1}^k \alpha_i e_i >_S = \sum_{i=1}^k \alpha_i < x, e_i >_S = \sum_{i=1}^k \alpha_i x_i$. In this case, the feature measured by e_p is completely determined by E_1, therefore does not bring any more information. It is noteworthy that vice verse, when $x_p = \sum_{i=1}^k \alpha_i x_i$, $e_p = \sum_{i=1}^k \alpha_i e_i$ is a candidate for feature basic vector. In an extreme case, if $x_p = x_1$, one can safely make $e_p = e_1$. In this case, the similarity of e_p and e_1 can be induced from x_p and x_1 if we use the interpretation in (1). This is the reason of naming this interpretation as *syntactic similarity.*

In fact, in practice, this interpretation is used in many feature selection methods [25, 14, 26, 5]. From the difference in interpretations, we can infer the difference in the families of methods, one for feature selection and the other for feature semantic similarity.

6 Supervised Feature Semantic Similarity

As analysed above, our framework opens a new research direction of learning the feature proximity matrix in a supervised way, that it to learn the matrix for discriminative purpose. In this section, we realize this analysis by proposing a general purpose method followed by some concrete application benefiting from the framework.

6.1 A General Discriminative Feature Proximity Matrix

Denote S as the within class covariance matrix. Denote M is the matrix containing corresponding classes' means from the original data matrix X. We have $S = (X - M)(X - M)^T$. Denote R as the between class covariance matrix. We restrict to the binary case, then $R = rr^T$ where r is the vector between classes' means. The objective is chosen to be the feature proximity matrix F that *on the direction of classes' means, the ratio between within class variance and between class variance is minimized.* Within class variance on the direction r is computed as the variance of $< (X - M), r >= (X - M)^T Fr$, hence equal to $((X - M)^T Fr)^T((X - M)^T Fr) = r^T FSFr$. Similarly, between class variance on the direction r is $r^T FRFr$. The problem is then formulated as maximizing some class separability criteria, as

$$\hat{F} = \arg\min_F f(F) = \frac{r^T FSFr}{r^T FRFr}. \tag{9}$$

Here we do not search for all possible F. Instead, we design the matrix F to have a special form so that the kernel is then computed from the original kernel. It is also our idea to use F of the form of a polynomial of S (with possibly negative degree). The reason is that S contains the label information (as opposed to the (uncentralized) covariance that does not contain label information). With some manipulation, we come up with the result $F = S^{-1}$.

Kernelization. We show that the having $F = S^{-1}$, kernels can be conveniently computed in the feature space, i.e., by using kernels themselves. Denote the kernels with feature similarity as K_{fs}, original kernel $K = X^T X$ for training data X, the kernel matrix is then:

$$\begin{aligned} K_{fs} &= X^T FX \\ &= X^T((X - M)(X - M)^T)^{-1}X \end{aligned}$$

Denote the class-conditioned centralized kernel as $K_{cn} = (X - M)^T(X - M) = X^T X + M^T M - X^T M - M^T X = X^T X - M^T M$, which is computable from the kernel matrix K [2]. Denote $K_{av} = (X - M)^T X$, then K_{av} can also be computed directly from the original kernel matrix K[3]. Suppose that Singular Value Decomposition of $X - M$ is

$$X - M = U\Lambda V^T,$$

[2]Suppose that K_{cn} is generated from the kernel function k_{cn} then $k_{cn}(x_i, x_j) = k(x_i, x_j) - k(\overline{x_i}, \overline{x_j})$, where $k(\overline{x_i}, \overline{x_j})$ is the average of all $k(x_k, x_l)$ that $y_k = y_i$, $y_l = y_j$.

[3]Suppose that K_{av} is generated from the kernel function k_{av} then $k_{av}(x_i, x_j) = k(x_i, x_j) - k(x_i, \overline{x_j})$, where $k(x_i, \overline{x_j})$ is the average of $k(x_i, x_l)$ as $y_l = y_j$.

then $(X - M)(X - M)^T = U\Lambda^2 U^T$, $U^T U = I$, $V^T V = I$ and i is an identity matrix in an appropriate space. Plugging in the above equation, we get

$$\begin{aligned} K_{fs} &= X^T((X - M)(X - M)^T)^{-1}X \\ &= X^T U\Lambda^{-2}U^T X \\ &= X^T U\Lambda V^T \cdot V\Lambda^{-4}V^T \cdot V\Lambda U^T X \\ &= X^T(X - M) \cdot ((X - M)^T(X - M))^{-2} \cdot (X - M)^T X. \end{aligned}$$

We have

$$K_{fs} = K_{av}^T \cdot K_{cn}^{-2} \cdot K_{av}. \tag{10}$$

For out-of-sample test data matrix A, the kernel function on them with training data are computed as

$$T_{fs} = T_{av} \cdot K_{cn}^{-2} \cdot K_{av} \tag{11}$$

where T_{av} is computed with the kernel function $k_{av}(x_m, x_j) = k(x_m, x_j) - k(x_m, \overline{x_j})$ and $k(x_m, \overline{x_j})$ is the average of all $k(x_m, x_l)$ that $y_l = y_j$. This kernel is computable from original kernels, therefore, it is free from concrete feature spaces.

Two dimensional example. We take a simple example in a two dimensional space to show the intuition of our supervised feature similarity. Suppose that data are as in the figure 3. In this case, the two features are highly correlated for each class, but not for the whole data set. The within class covariance matrix is

$$\begin{pmatrix} 1 & x \\ x & 1 \end{pmatrix}$$

for some $x \in \mathbb{R}^+$, $|x| < 1$. Our proposed feature similarity have the feature proximity matrix as:

$$F \stackrel{\text{def}}{=} \begin{pmatrix} 1 & x \\ x & 1 \end{pmatrix}^{-1} = \frac{1}{1 - x^2} \begin{pmatrix} 1 & -x \\ -x & 1 \end{pmatrix}$$

In this case, the degree of similarity is an inverse of class conditional correlation of feature. When $x \cong 1$, meaning that the two features are highly correlated, in our method, feature similarity is close to the smallest possible value. This means that in the geometric framework, the two features should be stretch away almost as far as possible, i.e. having opposite directions ($e_1 \cong -e_2$). We call this the two features complements each others. By using this feature proximity matrix when computing kernel, $k(x_i, x_j) = x_i^T F x_j$, it is equivalent to transforming the whole class to a point in the x axis, as shown in the figure 3. When having the two classes transformed to two different points, we are sure to be able to learn a perfect classification. It is noteworthy that by using other methods for feature similarity such as in [9], in this case, the two features are detected to have a positive similarity. Therefore,

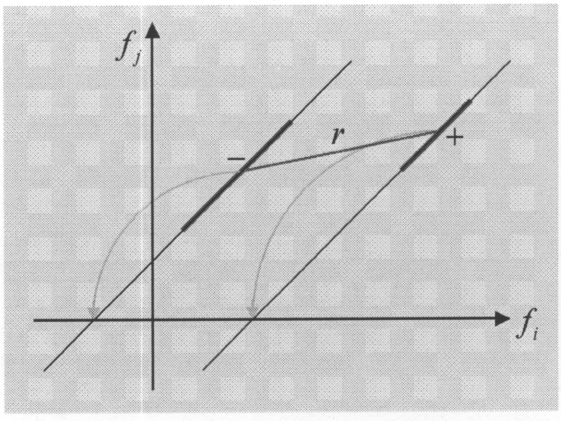

Figure 3: The case of perfect complementarity $x = -1$ while the two feature f_i and f_j are not correlated. Each thick red line segment represents data from a class (+ or −). Yellow arrows mean the transformations (to points) of the whole class via the effect of feature similarity. Note that the two features may *not* be correlated.

those methods do not transform data the way our method does to have a classes shrunken down to close to points, which results in high classification performance. The reason is straightforward, those methods do not take into account label information.

6.2 Properties

Some properties of the approach can be analyzed as follows.

- It is different from feature selection [25, 5] that we semantically similar features do not have a high correlation or contain similar information. Instead, they have anti-correlation or contain complementary information. This method is supposed to be a corresponding counterpart of those in [25, 5] when using an alternative feature interpretation.
- It is different from latent semantic kernels and [3] learning semantic similarity [9] that in our formulation, two features are said to be semantically similar if they show a *complementary behavior* with respect to the task at hand, not *semantical similarity* in the natural language sense as used in theirs. This can be a significant difference for the following reasoons. First, in natural language, synonyms or semantically similar words (in natural language senses) may not have complementary behavior. These words my appear in the same document with different functionality, not only as synonyms for each other. Second, semantically similar features may not have similar meaning in the natural language sense. For example, the words *buy* and *rent* would likely appear in the category *advertisement* complementarily but not synonyms. The pair of words *car* and *motorbike* in the category *vehicle* is another example. This difference also generalizes to the case of any kind of features, not necessarily words in previous applications [9, 3, 12].

7 Experiments

We conduct some experiments to show that the proposed general kernel actually improve performance over its baseline. In our case, baselines are linear kernels. However, the baselines could be any kernel as computing our proposed kernel is done completely in feature space, i.e. from baseline kernel only. We compare our proposed kernels with the baselines and empirical kernels, von Neumann kernel and diffusion kernels as well.

7.1 Synthetic data

Two classes: class $+1$ centers at $(0,0)^T$ and class -1 centers at $(1,0)^T$. Both classes are a Gaussian distribution with covariance matrix

$$\begin{pmatrix} 6 & 4 \\ 4 & 6 \end{pmatrix}.$$

We randomly generate 50 data points for each class following the distribution. Out of this data set, we sample different percentages of the whole data set for training while the rest are held for testing. The list of percentages used for split can be seen in the figure 5. For each percentage, we randomly split the data ten times. Support Vector Machines are used for classification with parameter C optimized by cross validation. The measure we show in the figure is the average of the ten accuracy from hold-out test set.

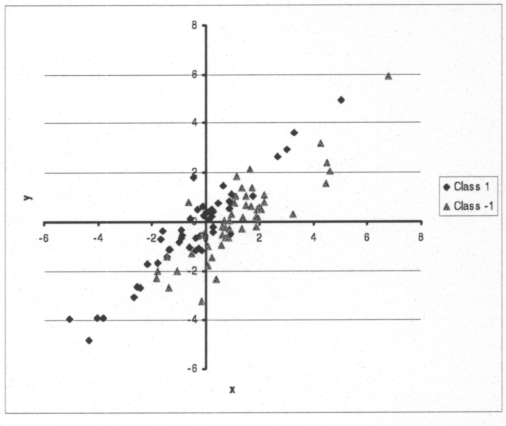

Figure 4: Two dimensional simulation data with each class is a Gaussian distribution.

Form the figure 5, we can observe that:

- Our proposed kernels show a consistently higher performance than any other kernels in comparison. This shows that by taking into account feature similarity in our method helps to improve accuracy over the baseline.
- Kernels proposed in [9] does not show any improvement over the baseline even though there is an additional parameter λ to generalize from the baseline. This means that

those method are not able to exploit feature semantic similarity for the purpose of classification. When we inspect the value of λ, it is usually set to close to 0, meaning that the additional information of feature semantic similarity are not helpful for discriminative purpose, therefore, down-weighted.

- Empirical kernels show the worst performance of all for a reason that it does not mean to exploit feature similarity.
- When there is a lesser amount of data, all methods perform not as well as when having more. In this case, our methods improve more over the baseline. This shows that our method is beneficial when there is aa few data, i.e., additional information is needed. It may not help when there is a large amount of data, i.e., having enough information.

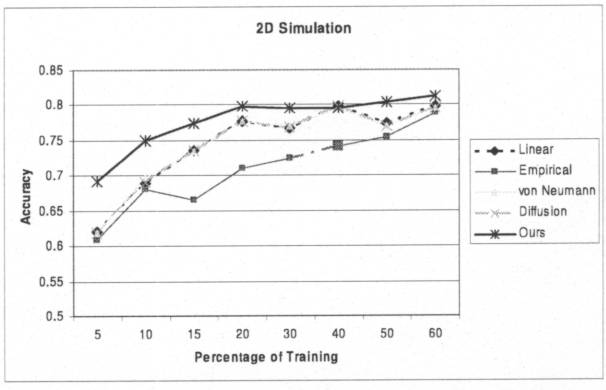

Figure 5: Classification accuracy of simulation data as a function of percentage of training data. Our proposed kernel usually gives a higher accuracy.

7.2 Text Retrieval with SVM

We follow the experiment conducted in the paper that proposes to learn semantic similarity of features, i.e., von Neumann and diffusion kernels [9]. Medline1033 [3] is the information retrieval data set used. The data set contains 1033 documents and 30 queries obtained from the national library of medicine. Here we concentrate on query20. For the data sets, each document is represented in a vector space that allows us to compute bag-of-word kernels as the baseline. Stop words and punctuation signs are removed from the documents. Poster stemmer was applied to the remaining words. Words are then are weighted according to a variance of tfidf scheme as in [9]. Support Vector Machines are used to learn to recognize relevant documents to a query. Parameter C is chosen using a cross validation scheme for the baseline kernel, then used for all other kernels. As a retrieval problem we use $F1$ as the performance measure as

$$F1 = \frac{2 \cdot pr \cdot rc}{pr + rc}$$

where pr is the precision and rc is the recall of retrieval result. F1 is a popular performance measure for Information Retrieval.

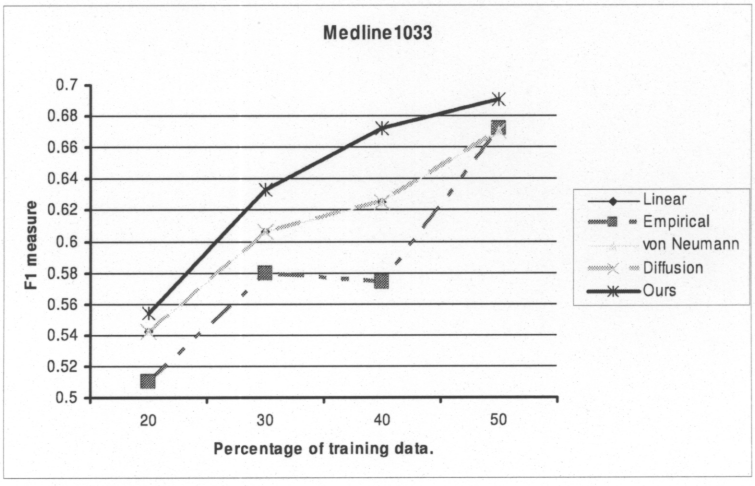

Figure 6: $F1$ measures of retrieval result on Medline1033 data as a function of percentage of training data taken for training. Our proposed kernel usually shows higher $F1$ measures. Linear, von Neumann and diffusion kernels give almost identical results.

We also use different splits of data with 20, 30, 40 and 50 percent of data for training. The rest are used for testing. Each split is repeated ten times. The averaged $F1$ results is shown in table 7.2 and visualized in figure 6.

	20	30	40	50
Linear	45.22 (± 17.63)	57.94 (± 15.94)	66.76 (± 5.69)	67.62 (± 5.65)
Empirical	41.09 (± 15.54)	56.19 (± 13.40)	61.78 (± 10.06)	66.48 (± 5.26)
von Neumann	45.22 (± 17.63)	57.94 (± 15.94)	66.76 (± 5.69)	67.62 (± 5.65)
Diffusion	45.22 (± 17.63)	57.94 (± 15.94)	66.76 (± 5.69)	67.62 (± 5.65)
Ours	46.05 (± 17.77)	60.35 (± 13.16)	69.64 (± 3.57)	71.06 (± 6.44)

Table 2: F1 results for comparing different kernels using feature similarity at different data split ratios.

From the figure and table, the following can be observed.

- Our proposed kernels show either a higher performance than any other kernels or equal performance in comparison. This shows that by taking into account feature similarity in our proposed kernels helps to improve accuracy over the baseline.
- Similar to the previous experiment, kernels proposed in [9] does not show any improvement over the baseline even though there is an additional parameter λ to generalize

from the baseline. When we inspect the value of λ, it is also usually close to 0, meaning that the additional information of feature semantic similarity are not helpful for discriminative purpose.

- Empirical kernels also show the worst performance of all.
- Due to the sparseness of data and the limited number of training data, retrieval results are usually highly variable.

8 Conclusion

In this part, we propose a framework that can incorporate feature similarity, a type of background knowledge that exists in various domain. The framework is based on a geometrical interpretation of the non-orthogonality of the space. The framework generalizes various previous methods for constructing kernels, e.g. Fisher kernel, von Neumann kernels, diffusion kernels, among others. It shows a possibility to generalize all those methods and sheds a light on their relations. It also allows us to see that all those methods, usually used for supervised purpose, does not take label information into account. We then utilize the framework to propose a general kernels incorporating feature similarity and label information at the same time. This is the first methods having this properties. The kernel is computed on feature space, i.e., from an original kernel. Evaluating the kernels on a simulation and text retrieval domain, we can see some merit of using the proposed kernel. The merit comes from the fact that the proposed kernel utilize label information and feature similarity, as a proof to show the practical benefit of the framework we proposed earlier on. The contribution here can be twofold, at the theoretical part with the framework, and at the practical part with the general method with many desirable properties.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 2006.

[3] Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. *Journal Intelligent Information Systems*, 18(2-3):127–152, 2002.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[5] François Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.

[6] Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. A geometric view on bilingual lexicon extraction from comparable corpora. In *ACL '04:*

Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, page 526, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[7] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[8] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, 2000.

[9] Jaz S. Kandola, John Shawe-Taylor, and Nello Cristianini. Learning semantic similarity. In *Neural Information Processing Systems 15 (NIPS 15)*, pages 657–664, 2002.

[10] Risi Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Confonference on Machine Learning (ICML)*, pages 315–322, 2002.

[11] Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[12] Ning Liu, Benyu Zhang, Jun Yan, Qiang Yang, Shuicheng Yan, Zheng Chen, Fengshan Bai, and Wei-Ying Ma. Learning similarity measures in non-orthogonal space. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 334–341, New York, NY, USA, 2004. ACM.

[13] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing.* MIT Press, Cambridge, MA, USA, 1999.

[14] Pabitra Mitra, C. A. Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 24(3):301–312, 2002.

[15] Reinhard Rapp. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 320–322, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

[16] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[17] Bernhard Schölkopf, Sebastian Mika, Chris J.C. Burges, Philipp Knirsch, Klaus-Robert Müller, Gunnar Rätsch, and Alexander J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[18] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[19] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.

[20] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, New York, NY, USA, 2004.

[21] Pascal Soucy and Guy W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 1130–1135. Professional Book Center, 2005.

[22] Kumiko Tanaka and Hideya Iwasaki. Extraction of lexical translations from non-aligned corpora. In *Proceedings of the 16th conference on Computational linguistics*, pages 580–585, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[23] Koji Tsuda. Support vector classifier with asymmetric kernel functions. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 183–188, 1999.

[24] N. Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, NY, 2000.

[25] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004.

[26] Zheng Zhao and Huan Liu. Searching for interacting features. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1156–1161, 2007.