Learning Additive Kernel For Feature Transformation and Its Application to CNN Features

Takumi Kobayashi
takumi.kobayashi@aist.go.jp

National Institute of Advanced Industrial Science and Technology
Tsukuba, Japan

Motivation

Feature transformation is an important process following feature extraction and several methods such as L_2-Hellinger [1] and an explicit map of χ^2 kernel [5] favorably improve classification performance on the feature vectors represented in the forms of BoW and Fisher kernel. However, it is difficult to apply those top-down transformation methods, which are designed based on the inherent nature of the features, to the features whose characteristics are not fully disclosed, such as CNN features; actually, they fail to improve performance of `Alex` [2] CNN feature (Fig. 1). In this paper, we propose a method to learn the feature transformation function of high generality and discriminative power in a bottom-up manner based on actual (annotated) data. The learned function corresponds to an explicit map of an additive kernel and the bottom-up learning approach endows the kernel function with discriminative power, adapting it even to the features of unknown characteristics; this is the case of CNN features [4] which are of our main interest in this paper, though the enthusiastic studies are now underway to explore the CNN features' nature. Note that the proposed method transforms the feature vector itself so as to be applicable to linear classifications unlike the other kernels, *e.g.*, RBF.

Proposed method

Additive Kernel. Given a pair of feature vectors, $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathbb{R}^D$, an additive kernel is defined by

$$\bar{\mathtt{k}}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{D} \mathtt{k}(x_i,y_i) \approx \sum_{i=1}^{D}\sum_{l=1}^{K} \phi(\gamma_l;x_i)\phi(\gamma_l;y_i) = \sum_{i=1}^{D} \boldsymbol{\phi}(x_i)^\top \boldsymbol{\phi}(y_i), \quad (1)$$

where x_i and y_i are the i-th elements of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and $\boldsymbol{\phi}$ is the explicit map function of the kernel $\mathtt{k}$. While the explicit map $\boldsymbol{\phi}$ can be determined according to the predefined kernel $\mathtt{k}$ [5], we learn the function from data in a bottom-up manner, which eventually compose the additive kernel function (1).

Representation. Since it is quite difficult to built $\boldsymbol{\phi}$ from scratch, we resort to the approximated representation of $\boldsymbol{\phi}$ by using basis functions like the Fourier fashion:

$$\boldsymbol{\phi}(x) = \boldsymbol{W}^\top [\mathtt{f}_1(x), \cdots, \mathtt{f}_M(x)]^\top = \boldsymbol{W}^\top \mathbf{f}(x), \quad (2)$$

where $\boldsymbol{W} \in \mathbb{R}^{M\times K}$ is the coefficient matrix for the basis functions, and we define the basis functions $\{\mathtt{f}_m\}_{m=1}^M$ for CNN features as follows.

$$\mathtt{f}_{2m-1}(x) = x\cos(2\pi\eta_m x), \;\; \mathtt{f}_{2m}(x) = x\sin(2\pi\eta_m x), \quad (3)$$

where η_m are the frequency parameters; in this study, we set $\eta \in \{0, 0.1, \cdots, 0.9, 1, \cdots, 10\}$ to produce $M = 39$ basis functions. Thereby, our objective is to learn the coefficients $\boldsymbol{W}$ by utilizing the annotated data.

Learning. The feature vector $\boldsymbol{x}$ is transformed into $\boldsymbol{W}^\top \mathbf{F}(\boldsymbol{x}) \triangleq \boldsymbol{W}^\top [\mathbf{f}(x_1), \cdots, \mathbf{f}(x_D)]$ and thus the linear classifier is written by $\mathtt{tr}\{\boldsymbol{W}^\top \mathbf{F}(\boldsymbol{x})\boldsymbol{A}\} + b$ with a classifier weight $\boldsymbol{A}$ and a bias b. This induces a bilinear optimization having difficulty in scalability and generality due to aggressively minimizing the rank K of $\boldsymbol{W}$ on the particular training dataset. We apply an efficient approach to learn $\boldsymbol{W}$, providing a good trade-off between the generality and discriminativity. Our optimization scheme is composed of two steps. First, we optimize the joint weight $\boldsymbol{V}_{\mathcal{D}} = \boldsymbol{W}\boldsymbol{A}_{\mathcal{D}}$ on the dataset $\mathcal{D}$ by applying the off-the-shelf (linear) SVM solver, and repeat it for various datasets $\{\mathcal{D}_c\}_{c=1}^C$, such as MIT67 and VOC2007. Then, we extract $\boldsymbol{W}$ which is shared across $\{\boldsymbol{V}_{\mathcal{D}_c}\}_{c=1}^C$ as

$$\boldsymbol{V}^* = [\boldsymbol{V}^*_{\mathcal{D}_1}, \cdots \boldsymbol{V}^*_{\mathcal{D}_C}] = \boldsymbol{W}^*[\boldsymbol{A}^{*\top}_{\mathcal{D}_1}, \cdots, \boldsymbol{A}^{*\top}_{\mathcal{D}_C}] \in \mathbb{R}^{M\times CD}. \quad (4)$$

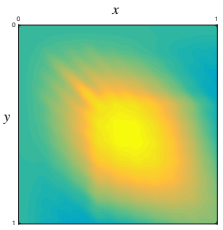
For that purpose, we utilize the singular value decomposition (SVD), $\boldsymbol{V}^* = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$, to produce

$$\boldsymbol{W}^* = \boldsymbol{P}\boldsymbol{\Lambda}^{\frac{1}{2}}. \quad (5)$$

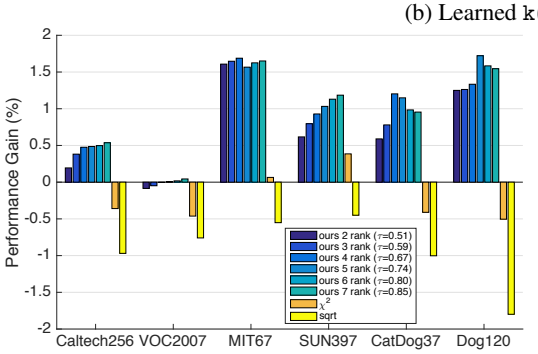
Actually, the rank K of the coefficient $\boldsymbol{W}^*$ is determined based on the contributing rate in SVD (singular values).

(a) Classification Accuracy (%)

Dataset	Orig.	χ^2 [5]	Hellinger [1]	Ours$_{6rank}$
CALTECH256	73.11	72.75	72.14	**73.61**
VOC2007	76.56	76.09	75.80	**76.57**
MIT67	62.99	63.06	62.44	**64.62**
SUN397	48.25	48.64	47.80	**49.38**
CATDOG37	80.83	80.42	79.83	**81.82**
DOG120	63.99	63.49	62.20	**65.58**



(b) Learned $\mathtt{k}(x,y)$



(c) Performance gain by the feature transformation

Figure 1: Performance results for `Alex` CNN feature.

Results

The proposed method is applied to transform the pre-trained CNN features using the models of `Alex` [2], `VGG` [3] and `C3D` [4]; the outputs of the first fully connected layer (`fc6`) are diverted to the feature vector of $D = 4096$ dimension.

We first assessed the generality by measuring how the learned (kernel) function fluctuates according to the training datasets, and found that the learned kernels exhibit quite high similarity, being almost identical, even though they are trained on different datasets of enough size. Such generality is rendered by exploiting the shared component $\boldsymbol{W}$ from various classifier weights via SVD (5) . Thus, we can say that the proposed method produces the *generic* additive kernel (or feature transformation).

Next, the proposed method is compared with the other feature transformation methods, explicit map of χ^2 kernel [5] and Hellinger (square root) transformation [1], both of which are successfully applied to the hand-crafted features, such as BoW and Fisher kernel features. The methods are tested on six image datasets using `Alex` [2] and `VGG` [3] features: CALTECH256, VOC2007, MIT67, SUN397, CATDOG37 and DOG120, and on four action datasets using `C3D` [4]: HMDB51, HOLLYWOOD2, UCF101 and UCF50. The performance results of `Alex` feature are shown in Fig. 1. In contrast to the case of hand-crafted features, the χ^2 and Hellinger transformations do not work well; the χ^2-based transformation [5] failed to improve performance in many cases and to make matters worse, the Hellinger transformation [1] degrades performance in all cases. On the other hand, the learned transformation favorably boosts the performance, outperforming the other methods. Through the bottom-up learning approach, the proposed method adapts the transformation function, *i.e.*, the additive kernel, to the CNN features whose characteristics are not fully revealed.

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, pages 2911–2918, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015.

[5] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.