Efficient Application of Gabor Filters with Non-Linear Support Vector Machines

Ajitesh Srivastava[1], Pritish Mohapatra[2], and A. S. Mandal[2]

[1] Birla Institute of Technology and Science,
Pilani, Rajasthan, India
[2] Central Electronics Engineering Research Institute,
Pilani, Rajasthan, India
ajitesh.srivastava@live.in,pritish.eee@gmail.com,
atanu@ceeri.ernet.in

Abstract. Both Gabor filters and Support Vector Machines (SVMs) are widely used in computer vision tasks for feature extraction and classification respectively. However the method is usually plagued by the problems of high computational complexity and memory usage owing to the high dimensionality of the Gabor filter responses. There were methods proposed to mitigate this problem by truncating or finding a gist of the responses but such approaches also lead to loss of information. Ashraf et al. gave a reinterpretation of the whole method and proposed a way to eliminate the need for such approximations. But they only give an analysis for linear SVM. This paper extends their work and provides analysis for non-linear kernels within the same framework. The class of non-linear kernels that are compatible with this framework are derived and experimental results on the facial expression recognition task are reported.

Keywords: Gabor Filter, Support Vector Machines, Kernel functions, Expression Recognition

1 Introduction

Gabor filters have been used for feature extraction in computer vision for a long time now [1][2][3]. Their similarity to the receptive fields of mammalian cortical simple cells [4], provides researchers an extra inspiration for using them for representation of visual data. They are often used in conjunction with Support Vector Machine (SVM) classifiers for computer vision tasks [5][6][7][8]. SVMs are maximum margin classifiers which use sample data to construct a hyperplane classifier in a higher dimensional mapping of the original feature space. The mapping can be linear or nonlinear depending on the type of kernel used.

One of the key problems with the use of Gabor filters as feature extractors is the high computational complexity and memory usage, both during training of the classifier and during prediction. To counter this problem, often only a gist of the original concatenated Gabor response is used. The gist can be computed by using some kind of averaging or a feature selection algorithm like AdaBoost. In [9] Ashraf et al. proposed a method to circumvent the need for such approximations when using SVMs. It involves the use of some mathematical properties of the Fourier transform and the reinterpretation of the working of the SVMs when used in conjunction with the Gabor filters as maximizing the margin in a weighted Euclidian space. But [9] provides an analysis only for linear support vector machine.

In this paper, we provide an analysis of the extension of the method proposed in [9] to non-linear kernels. We find out the class of non-kernels that can be used in the same framework and to same effect towards reducing computation and memory usage. In congruence with [9] we chose the platform of facial expression recognition for our experiments.

The rest of the paper is organized as follows. Section 2 reviews the application of Gabor filters and SVMs to computer vision tasks and the reinterpretation provided by [9]. Section 3 analyses the extension of the method of Ashraf et al. to non-linear kernels. Experimental results are described in Sect. 4. Finally, the conclusions from the study have been drawn in Sect. 5.

2 Classification using Gabor filtering and SVM

2.1 Application of Gabor Filter

The Gabor filter is composed of two components, a complex sinusoidal carrier and a Gaussian envelope. The Gabor function can be written as $g_{\omega,\theta} = \frac{1}{2\pi\sigma} \exp(j\omega x') \exp\left(-\frac{x'^2+y'^2}{2\sigma^2}\right)$, where x', y' are the spatial coordinates, θ is the orientation of the Gabor filter and ω is the frequency of the complex sinusoid and σ is the standard deviation of the Gaussian. By varying, ω and θ, we can generate a set of 2D Gabor filters of different scales and orientations. A 2D Gabor filter can be represented as a vectorization of the matrix $\boldsymbol{g_{\omega,\theta}}$ with elements, $\{g_{\omega,\theta}(x,y)\}$. The response of an image to this filter in Fourier domain can be calculated as

$$\hat{\boldsymbol{r}} = \hat{\boldsymbol{x}} \circ \hat{\boldsymbol{g}}_{\boldsymbol{\omega},\boldsymbol{\theta}} \ . \tag{1}$$

Where, $\hat{\boldsymbol{x}}$ is the vectorized image matrix in Fourier domain and $\hat{\boldsymbol{r}}$ is the response in Fourier domain. Now the complete response of the image vector to the entire set of Gabor filters, in spatial domain can be represented in a concatenated form as,

$$\boldsymbol{z} = [\boldsymbol{r_1}^T, \boldsymbol{r_2}^T, \ldots, \boldsymbol{r_M}^T]^T \ . \tag{2}$$

2.2 Efficient Training of Gabor–SVM without Information Loss

SVMs are binary classifiers, popular for their ability to handle high dimensional data and is widely used in image classification. Consider a training data set

$\{(\boldsymbol{z}_1, y_1), (\boldsymbol{z}_2, y_2), \ldots, (\boldsymbol{z}_N, y_N)\}$, where $\boldsymbol{z_i}$ are the training samples and $y_i \in \{-1, +1\}$ are the labels to which each sample can be assigned to. The SVM tries to build a hyperplane, $\boldsymbol{w}^T\boldsymbol{z} - b = 0$ that best separates the data points (by widest margin). It does so by minimizing the objective function $\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$ subject to $\xi_i > 0$, $y_i(\boldsymbol{w}^T\boldsymbol{z} - b) \geq 1 - \xi_i$, $\forall i$.

Here ξ_i are slack variables that allow misclassification for data that are not linearly separable and C is the penalizing constant. The problem of optimization is simplified by using its dual representation:

$$\max_{0 \leq \alpha_i \leq C} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j {\boldsymbol{z}_i}^T \boldsymbol{z}_j \ , \tag{3}$$

subject to $\sum_i \alpha_i y_i = 0$.

In our case, $\boldsymbol{z_i}$ is the concatenated response of the image $\boldsymbol{x_i}$ on a set of Gabor filters. Using (1), (2) and Parseval's theorem[10] the dot product of concatenated response vectors ${\boldsymbol{z}_i}^T\boldsymbol{z}_j$ can be expressed as product of matrices of smaller dimensions[9]:

$${\boldsymbol{z}_i}^T\boldsymbol{z}_j = {\hat{\boldsymbol{x}}_i}^T \boldsymbol{S}\, \hat{\boldsymbol{x}}_j \ . \tag{4}$$

where,

$$S = \sum_m diag(\hat{\boldsymbol{g}}_m)^T\, diag(\hat{\boldsymbol{g}}_m) \tag{5}$$

The optimization problem, thus, becomes

$$\max_{0 \leq \alpha_i \leq C} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j {\hat{\boldsymbol{x}}_i}^T \boldsymbol{S}\, \hat{\boldsymbol{x}}_j \ . \tag{6}$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \, .$$

Which is same as training with the vectors $(\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i)$. Though there is one problem that seems to have crept in due to the use of Fourier vectors. The SVM is to be trained with complex vectors, while most SVM packages support real vectors only. To tackle this problem the vectors $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ of N dimension can be represented as a real vector $[\Re(\hat{\boldsymbol{x}}_{\boldsymbol{i}})\ \Im(\hat{\boldsymbol{x}}_{\boldsymbol{i}})]$ of $2N$ dimension [9] and the optimization problem in (6) can be solved.

2.3 Testing without Filtering

The classification of a test image x is theoretically based on its concatenated response $\boldsymbol{z}$, according to $sign(\boldsymbol{w}^T\boldsymbol{z} - b)$. The outcome of training SVM is the bias b and α_i values that enable us to calculate:

$$\boldsymbol{w}^T\boldsymbol{z} = \left(\sum_i \alpha_i y_i (\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i)^T\right) \sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}} \ . \tag{7}$$

If we let $\boldsymbol{W} = S\left(\sum_i \alpha_i y_i \boldsymbol{\hat{x}}_i\right)$ then we have $\boldsymbol{w}^T\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{\hat{x}}$. This requires the test image to be first transformed into Fourier domain. However, the step can be skipped by using Parseval's Theorem:

$$\boldsymbol{W}^T\boldsymbol{\hat{x}} = \boldsymbol{\tilde{W}}^T\boldsymbol{x} \tag{8}$$

where $\boldsymbol{\tilde{u}}$ represents inverse 2D fourier transform of the vector $\boldsymbol{u}$.

The value of $\boldsymbol{\tilde{W}}$ can be pre-computed and stored. The classification can now be performed by

$$class = sign(\boldsymbol{\tilde{W}}^T\boldsymbol{x} - b) \tag{9}$$

As a result, for testing, we do not need to compute the Fourier transform of the image and the classification can be performed on raw image. Thus, the testing time can be significantly reduced.

3 Extension to Non-Linear SVM

For a non-linear SVM a transformation $\boldsymbol{z} \to \boldsymbol{\phi}(\boldsymbol{z})$ is assumed and $\boldsymbol{\phi}(\boldsymbol{z_i})^T\boldsymbol{\phi}(\boldsymbol{z_j})$ is replaced with $K(\boldsymbol{z_i}, \boldsymbol{z_j})$, where K is the Kernel function. The optimization problem for non-linear SVM is

$$\max_{0\le\alpha_i\le C} \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j K(\boldsymbol{z}_i, \boldsymbol{z}_j)\ . \tag{10}$$

$$\text{subject to } \sum_i \alpha_i y_i = 0\,.$$

The technique that has been used in the linear case makes use of the property of the dot product $\boldsymbol{z_i}^T\boldsymbol{z_j}$. We cannot use the improvements in training and testing suggested for linear case if the objective function of the optimization step is dependent on $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$ in any other way. So, we need to select such a kernel which is a function of the dot product of its variables:

$$K(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{\phi}(\boldsymbol{u})^T\boldsymbol{\phi}(\boldsymbol{v}) = f(\boldsymbol{u}^T\boldsymbol{v})\ . \tag{11}$$

Due to the property of the Kernel function that we have assumed in (11) and the result of (4), we can write:

$$K(\boldsymbol{z}_i, \boldsymbol{z}_j) = f(\boldsymbol{z}_i{}^T\boldsymbol{z}_j) = f((\sqrt{\boldsymbol{S}}\boldsymbol{\hat{x}_i})^T(\sqrt{\boldsymbol{S}}\boldsymbol{\hat{x}_i})) = K(\sqrt{\boldsymbol{S}}\boldsymbol{\hat{x}_i}, \sqrt{\boldsymbol{S}}\boldsymbol{\hat{x}_j})\ . \tag{12}$$

Thus, once again, training with $\boldsymbol{z}_i$ becomes same as training with the vectors $(\sqrt{\boldsymbol{S}}\boldsymbol{\hat{x}}_i)$. Using the same technique of transforming from complex vector to real vector used in the linear case, we can get the corresponding values of α_i in (10) and do the classification.

The classification is based on $sign(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{z}) - b)$. Now, $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{z})$ can be evaluated as follows:

$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{z}) &= \sum_i \alpha_i y_i K(\boldsymbol{z}_i, \boldsymbol{z}) \\ &= \sum_i \alpha_i y_i f((\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_{\boldsymbol{i}})^T(\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}})) \\ &= \sum_i \alpha_i y_i f((\boldsymbol{S}\hat{\boldsymbol{x}}_i)^T\hat{\boldsymbol{x}}) \ . \end{aligned} \tag{13}$$

To reduce computational cost during testing, a method similar to the linear case can be employed to test without filtering the test image, i.e., without transforming the image in frequency domain. We make use of Parseval's Theorem that leads to

$$(\boldsymbol{S}\hat{\boldsymbol{x}}_i)^T\hat{\boldsymbol{x}} = \widetilde{(\boldsymbol{S}\hat{\boldsymbol{x}}_i)}^T \boldsymbol{x} \ . \tag{14}$$

Using this in (13) yields

$$\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{z}) = \sum_i \alpha_i y_i f(\widetilde{(\boldsymbol{S}\hat{\boldsymbol{x}}_i)}^T \boldsymbol{x}) \ . \tag{15}$$

Note that the values $\widetilde{(\boldsymbol{S}\hat{\boldsymbol{x}}_i)}$ can be precalculated and stored for all the support vectors during training. So, the testing involves, only taking their dot products with the test image $\boldsymbol{x}$.

4 RBF Kernels

A radial basis function (RBF) is a function of two vectors, which depends on only the distance between them, i.e., $K(\boldsymbol{u}, \boldsymbol{v}) = f(\|\boldsymbol{u} - \boldsymbol{v}\|)$. A very widely used RBF as SVM Kernel is the Gaussian Kernel $K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(\frac{\|\boldsymbol{u}-\boldsymbol{v}\|}{\sigma}\right)$. Here, we show that any RBF kernel can be used while training with transformed vectors as suggested in Sect. 2.2

$$\begin{aligned}\|\boldsymbol{z_i} - \boldsymbol{z_j}\| &= \sqrt{(\boldsymbol{z_i} - \boldsymbol{z_j})^T\,(\boldsymbol{z_i} - \boldsymbol{z_j})} \\ &= \sqrt{\boldsymbol{z_i}^T\boldsymbol{z_i} - \boldsymbol{z_i}^T\boldsymbol{z_j} - \boldsymbol{z_j}^T\boldsymbol{z_i} + \boldsymbol{z_j}^T\boldsymbol{z_j}} \\ &= \sqrt{(\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i - \sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i)^T\,(\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i - \sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i)} \\ \text{(Using equation 4)} & \\ &= \|\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i - \sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i\| \ . \end{aligned} \tag{16}$$

Again, we find that for any radial basis function, including Gaussian RBF, training with $\boldsymbol{z_i}$ is equivalent to training with the vectors $(\sqrt{\boldsymbol{S}}\hat{\boldsymbol{x}}_i)$. And as the size of the matrix $\boldsymbol{S}$ does not depend on the number of Gabor filters being used, it becomes possible to use as many Gabor filters as desired without increasing the amount of storage space required. However, as the RBF kernels do not satisfy equation (11), its not possible to do classification without filtering in this case. So, reduction in memory usage is still possible with RBF kernels, though reduction in computations is not obtained.

5 Results

We conducted our experiments on a combination of Cohn–Kanade FACS–Coded Facial Expression Database and JAFFE Database for expression recognition. The Cohn–Kanade database consists of about 500 video sequences of 100 subjects, while JAFFE database consists of 213 images of 10 subjects. Out of these, 1455 face images were extracted, in which eight expressions were identified: neutral, anger, contempt, disgust, fear, happiness, sadness and surprise. The face area of each of these images were cropped and resized to 100×100 image.

The implementaion was done using the SVM package of MATLAB from Bio–Informatics Toolbox. For each expression we learn a one vs all binary classifier as explained in Sect. 3, using Gabor filters of various sizes. The results of k-fold cross validation, with 32×32 Gabor filters are given in Table 1. This table shows the results of linear, MLP, polynomial and Gaussian RBF kernel after normalization or scaling the data to 0 mean and 1 standard deviation (which is default in MATLAB).

Table 1. Accuracies of one vs all classifiers, with normalization

Emotion	Linear Kernel	MLP Kernel	Polynomial Kernel	Gaussian RBF
Neutral	60.63	60.18	41.82	63.29
Anger	90.95	91.86	91.82	91.86
Contempt	95.93	95.02	95.45	95.39
Disgust	92.76	91.86	91.82	92.59
Fear	94.09	94.57	94.55	94.30
Happiness	91.36	87.33	88.07	88.25
Sadness	94.14	94.14	94.55	94.30
Surprise	89.59	86.94	87.16	90.14
Average	88.68	87.74	85.65	88.77

However, note that if we wish to test without filtering we have to use (14), which cannot be applied directly if we have used normalization during training. So, we also did some experiments without normalization (setting 'AutoScale' to false) for linear and MLP kernel. The results are shown in Table 2. We observe that there is no significant difference in with and without normalization. Also, kernels other than linear gave comparable results. In fact, Gaussian kernel produced better accuracy than the linear kernel.

Table 2. Accuracies of one vs all classifiers, without normalization

Emotion	Linear Kernel	MLP Kernel	Gaussian RBF
Neutral	72.07	57.47	74.50
Anger	90.45	80.54	90.96
Contempt	95.02	95.02	96.20
Disgust	92.31	91.86	92.95
Fear	95.00	89.64	94.48
Happiness	91.40	87.78	92.04
Sadness	94.57	92.31	94.58
Surprise	92.76	86.88	93.22
Average	90.45	85.19	91.12

For multiclass classification, instead of using the scores of the binary SVMs directly, we used the method proposed in [11]. The method maps the output of a binary SVM classifier, β into a posterior probability destribution $P(C_1|\beta)$:

$$P(C_1|\beta) = \frac{1}{1 + \exp{(A\beta + B)}} \ . \qquad (17)$$

Where A and B are parametes which can be learnt from the training data. For each of the binary classification we learnt the parameters A and B and used them to convert the output scores into probabilities of the image belonging to that class. The image was labeled to the class which had the highest probability. Implementation of this technique gave an overall error of 35.64 %.

6 Conclusion

In this paper, we have provided an analysis of the extension of the method proposed by Ashraf et al. to non-linear kernels. We observe that the same method can be applied to the class of non-linear kernels for which the kernel is a function of the inner product. Polynomial kernels, Multi-layer perceptron (MLP) kernels are some examples which fall into this class. We ran experiments for the task of facial expression recognition using the polynomial and MLP kernels apart from the linear kernel. The systems showed considerable improvement in memory usage and computation, albeit, slight decrease in accuracy compared to that of the linear kernel. This may be due to the comparatively small size of the dataset used in the experiment. Gaussian Kernel on the other hand, performed consistently better than linear kernel. With a more densely populated training dataset, the non-linear SVMs should be expected to give better result due to their higher model capacity. One of the key aspects of this work is that it exploits properties of specific non-linear kernels to enable more computationally efficient and memory efficient implementations of Gabor filters with SVM.

References

1. Wiskott, L., Fellous, J.M., Kruger, N., Malsburg, C.: Face Recognition by Elastic Bunch Graph Matching In: IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7):775–779, July (1997)
2. Feris, R.S., Gemmell, J., Toyama, K., Kruger, V.:Facial Feature Detection Using a Hierarchical Wavelet Face Database. Technical Report, MSR-TR-2002-05, Microsoft Research (2005)
3. Kruger, V., Sommer, G.: Gabor Wavelet Networks for Object Representation. In: Multi-Image Analysis, Springer - Lecture Notes in Computer Science.Vol. 2032/2001, pp. 115–128 (2001)
4. Daugman, J.G.: Complete discrete 2-D Gabor transforms by Neural Networks for Image Analysis and Compression. In: IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 36, pp. 11691179, July (1988)
5. Cheng, H., Zheng, N., Qin, J.: Pedestrian Detection Using Sparse Gabor filter and Support Vector Machine. In: Intelligent Vehicles Symposium. Proceedings, IEEE, pp. 583–587. (2005)
6. Zehang, S.,Bebis, G., Miller, R.: On–road Vehicle Detection Using Gabor Filters and Support Vector Machines. In: 14th International Conference on Digital Signal Processing, pp. 1019–1022 (2002)
7. Li, S., Kwok, J.T., Zhu, H., Wang, Y.: Texture Classification using the Support Vector Machines. In: Pattern Recognition, Elsevier, Volume 36, Issue 12, December, pp 2883–2893 (2003)
8. Bartlett, M.S., Littlewort, G., Frank, M., Lainscsek, C., Fasel, I., Movellan, J.: Recognizing Facial Expression: Machine Learning and Application to Spontaneous Behavior. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2, pp.568-573 (2005)
9. Ashraf, A.B., Lucey, S., Chen T.: Reinterpreting the Application of Gabor Filters as a Manipulation of the Margin in Linear Support Vector Machines. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 7, Mar., pp. 1335–1341 (2010)
10. Oppenheim, A.V., Willsky, A.S.: Signals & Systems, 2nd ed., Prentice Hall (1996)
11. Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D. (eds.): Advances in Large Margin Classifiers. Cambridge, MA (2000)