FEATURE EXTRACTION USING MUTUAL INFORMATION BASED ON PARZEN WINDOW

Nojun Kwak

Telecommunication R&D Center
Samsung Electronics, Suwon, Korea
nojunk@ieee.org

Chong-Ho Choi

School of Electrical Eng. & Computer Sci.
Seoul National University, Seoul, Korea
chchoi@csl.snu.ac.kr

ABSTRACT

In this paper, feature extraction for classification problems are dealt with. The proposed algorithm searches for a set of linear combinations of original features that maximizes the mutual information between the extracted features and the output class. The difficulties in the calculating mutual information between the extracted features and output class are resolved using Parzen window density estimate. Greedy algorithm with gradient ascent method is used to find the new feature. The computational load is proportional to the square of the number of the given samples. We have applied the proposed method to a simple classification problems and have observed that the proposed method gives better or compatible performance than the conventional feature extraction methods.

1. INTRODUCTION

For many pattern recognition problems, it is desirable to reduce the dimension of feature space via feature extraction because there may be irrelevant or redundant features that complicate the learning process, thus lead to erroneous results. Even when the features presented contain enough information about the problem, the result may be erroneous because the dimension of feature space can be so large that it may require numerous instances to obtain a generalized result.

Though mutual information is widely accepted as a good measure in the feature extraction problems, the computational complexity makes it difficult to use mutual information as a measure of extracting features. For this reason, in [1] Torkkola extracted output relevant features based on mutual information maximization using Renyi's entropy measure instead of that of Shannon.

Recently we have developed an effective way of calculating mutual information between an output class and continuous input features and applied it to feature selection problems [2]. In this paper, this method is applied to feature extraction problems by maximizing mutual information. In calculating the mutual information between the input features and the output class, instead of discretizing the input space, we use the Parzen window method to estimate the input distribution. With this method, more accurate mutual information is calculated and the projection direction which produces maximum mutual information is searched for.

In the following section, the method of calculating mutual information by Parzen window is presented. In Section III, we propose a new feature extraction method and in Section IV, the proposed algorithm is applied to a simple classification problem to show its effectiveness. And finally, conclusions follow in Section V.

2. CALCULATION OF MUTUAL INFORMATION WITH PARZEN WINDOW

In this section, the method of estimating the conditional entropy and the mutual information by the Parzen window in [2] is presented.

To calculate the mutual information between the input features and the output class, we need to know the probability density functions (*pdf*s) of the inputs and the output. The Parzen window density estimate can be used to approximate the probability density $p(\boldsymbol{x})$ of a vector of continuous random variables $\boldsymbol{X}$ [3]. It involves the superposition of a normalized window function centered on a set of samples. Given a set of n d-dimensional samples $D = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}\}$, the *pdf* estimate by the Parzen window is given by

$$\hat{p}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}-\boldsymbol{x}_i, h), \tag{1}$$

where $\phi(\cdot)$ is the window function and h is the window width parameter. Parzen showed that $\hat{p}(\boldsymbol{x})$ converges to the true density if $\phi(\cdot)$ and h are selected properly [3]. The window function is required to be a finite-valued non-negative

This work is partly supported by the Brain Neuroinformatics Research Program of Korean government.

density function where

$$\int \phi(\boldsymbol{y}, h) d\boldsymbol{y} = 1, \tag{2}$$

and the width parameter is required to be a function of n such that

$$\lim_{n\to\infty} h(n) = 0, \tag{3}$$

and

$$\lim_{n\to\infty} nh^d(n) = \infty. \tag{4}$$

For window functions, the rectangular and the Gaussian window functions are commonly used. The Gaussian window function is given by

$$\phi(\boldsymbol{z}, h) = \frac{1}{(2\pi)^{d/2} h^d |\Sigma|^{1/2}} \exp(-\frac{\boldsymbol{z}^T \Sigma^{-1} \boldsymbol{z}}{2h^2}), \tag{5}$$

where Σ is a covariance matrix of a d-dimensional vector of random variables $\boldsymbol{Z}$.

In classification problems, the class has discrete values while the input features are usually continuous variables. In this case, the mutual information between the input features $\boldsymbol{F}$ and the class C can be represented as follows:

$$I(\boldsymbol{F}; C) = H(C) - H(C|\boldsymbol{F}). \tag{6}$$

In this equation, because the class is a discrete variable, the entropy of the class variable $H(C)$ can be easily calculated. But the conditional entropy

$$H(C|\boldsymbol{F}) = -\int_{\boldsymbol{F}} p(\boldsymbol{f}) \sum_{c=1}^{N} p(c|\boldsymbol{f}) \log p(c|\boldsymbol{f}) d\boldsymbol{f}, \tag{7}$$

where N is the number of classes, is hard to get because it is not easy to estimate $p(c|\boldsymbol{f})$.

By the Bayesian rule, the conditional probability $p(c|\boldsymbol{f})$ can be written as

$$p(c|\boldsymbol{f}) = \frac{p(\boldsymbol{f}|c)p(c)}{p(\boldsymbol{f})}. \tag{8}$$

If there are N_c classes, we get the estimate of the conditional *pdf* $\hat{p}(\boldsymbol{f}|c)$ of each class using the Parzen window method as

$$\hat{p}(\boldsymbol{f}|c) = \frac{1}{n_c} \sum_{i\in I^c} \phi(\boldsymbol{f} - \boldsymbol{f}_i, h), \tag{9}$$

where $c = 1, \cdots, N_c$; n_c is the number of the examples belonging to class c; and I^c is the set of indices of the training examples belonging to class c. Because the sum of the conditional probability eqauls one, i.e.,

$$\sum_{k=1}^{N_c} p(k|\boldsymbol{f}) = 1,$$

the conditional probability $p(c|\boldsymbol{f})$ is

$$p(c|\boldsymbol{f}) = \frac{p(c|\boldsymbol{f})}{\sum_{k=1}^{N_c} p(k|\boldsymbol{f})} = \frac{p(c)p(\boldsymbol{f}|c)}{\sum_{k=1}^{N_c} p(k)p(\boldsymbol{f}|k)}.$$

The second equality follows from the Bayesian rule (8). Using (9), the estimate of the conditional probability becomes

$$\hat{p}(c|\boldsymbol{f}) = \frac{\sum_{i\in I^c} \phi(\boldsymbol{f} - \boldsymbol{f}_i, h_c)}{\sum_{k=1}^{N_c} \sum_{i\in I^k} \phi(\boldsymbol{f} - \boldsymbol{f}_i, h_k)}, \tag{10}$$

where h_c and h_k are window width parameters corresponding to class c and k.

Using the Gaussian window function (5) with the same window width parameter h and the same covariance matrix $\Sigma_{\boldsymbol{F}}$ for each class, (10) becomes

$$\hat{p}(c|\boldsymbol{f}) = \frac{\sum_{i\in I^c} \exp(-\frac{(\boldsymbol{f}-\boldsymbol{f}_i)^T \Sigma_{\boldsymbol{F}}^{-1} (\boldsymbol{f}-\boldsymbol{f}_i)}{2h^2})}{\sum_{k=1}^{N_c} \sum_{i\in I^k} \exp(-\frac{(\boldsymbol{f}-\boldsymbol{f}_i)^T \Sigma_{\boldsymbol{F}}^{-1} (\boldsymbol{f}-\boldsymbol{f}_i)}{2h^2})}. \tag{11}$$

Now in the calculation of the conditional entropy (7) with n samples, if we replace the integration with the summation of sample points and suppose that each sample has the same probability, then we get

$$\hat{H}(C|\boldsymbol{F}) = -\sum_{j=1}^{n} \frac{1}{n} \sum_{c=1}^{N_c} \hat{p}(c|\boldsymbol{f}_j) \log \hat{p}(c|\boldsymbol{f}_j), \tag{12}$$

where $\boldsymbol{f}_j$ is the jth sample. With (6) and (11), the estimate of the mutual information is obtained as follows:

$$\hat{I}(\boldsymbol{F}; C) = -\sum_{c=1}^{N_c} \hat{p}(c) \log \hat{p}(c) + \sum_{j=1}^{n} \frac{1}{n} \sum_{c=1}^{N_c} \hat{p}(c|\boldsymbol{f}_j) \log \hat{p}(c|\boldsymbol{f}_j), \tag{13}$$

where $\hat{p}(c)$ and $\hat{p}(c|\boldsymbol{f})$ can be replaced with $1/n_c$ and (12) respectively.

3. FEATURE EXTRACTION BY MAXIMIZING MUTUAL INFORMATION

Because the covariance matrix $\Sigma_{\boldsymbol{F}_i}$ has to be inverted in the calculation of $H(C|\boldsymbol{F}_i)$, it would be better if $\Sigma_{\boldsymbol{F}_i}$ takes a special form. To this end, the original N dimensional feature vector $\boldsymbol{X}$ is transformed into $N'(\leq N)$ dimensional vector $\boldsymbol{Y} = \boldsymbol{W}_{pca}^T \boldsymbol{X}$ using PCA. Note that the rank N' of $\boldsymbol{W}_{pca}$ becomes N if the covariance matrix $\Sigma_{\boldsymbol{X}}$ of the original features $\boldsymbol{X}$ is nonsingular. Note also that by the data processing inequality [4], $I(\boldsymbol{Y}; C) = I(\boldsymbol{X}; C)$ if $\boldsymbol{W}_{pca}$ has rank N. After PCA, the covariance matrix of $\boldsymbol{Y}$ becomes N' dimensional identity matrix, i.e., $\Sigma_{\boldsymbol{Y}} = I_{N'}$. Instead of using $\boldsymbol{X}$, if we use $\boldsymbol{Y}$, the feature extraction problem we are going to study is to find $\boldsymbol{v}_i^* \in \Re^{N'}$, $i = 1, \cdots, M$, such that

$$\boldsymbol{v}_i^* = \arg\min_{\boldsymbol{v}} H(C|\boldsymbol{F}_i) = \arg\min_{\boldsymbol{v}} H(C|\boldsymbol{V}_i^T \boldsymbol{Y}), \tag{14}$$

where $\boldsymbol{V}_i \triangleq [\boldsymbol{v}_1^*|, \cdots, |\boldsymbol{v}_{i-1}^*|\boldsymbol{v}]$ and $\boldsymbol{F}_i \triangleq \boldsymbol{V}_i^T\boldsymbol{Y}$.

In this case, the search space of $\boldsymbol{w}$ is restricted to N' dimensional subspace such that $\boldsymbol{w} = \boldsymbol{W}_{pca}\boldsymbol{v}$ and the ith newly extracted feature F_i becomes

$$F_i = \boldsymbol{v}_i^{*T}\boldsymbol{Y} = \boldsymbol{v}_i^{*T}\boldsymbol{W}_{pca}^T\boldsymbol{X}. \tag{15}$$

Henceforth, the covariance matrix of $\boldsymbol{F}_i$ becomes

$$\begin{aligned}\Sigma_{\boldsymbol{F}_i} &\triangleq E\{\boldsymbol{F}_i\boldsymbol{F}_i^T\} = E\{\boldsymbol{V}_i^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{V}_i\} \\ &= \boldsymbol{V}_i^T\boldsymbol{V}_i.\end{aligned} \tag{16}$$

To make the inversion of $\Sigma_{\boldsymbol{F}_i}$ easy, the candidate vector $\boldsymbol{v} \in \Re^{N'}$ for the last column of $\boldsymbol{V}_i$ is restricted only to the orthogonal direction to all the other $i-1$ columns of $\boldsymbol{V}_i$. In addition, because scaling $\boldsymbol{v}$ does not change the value of $\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y})$ and $\hat{H}(C|\boldsymbol{V}_i^T\boldsymbol{Y})$ in (11) and (12), $\boldsymbol{v}$ can always be normalized such that $\boldsymbol{v}^T\boldsymbol{v} = 1$. Then, the covariance matrix of $\boldsymbol{F}_i$ becomes i dimensional identity matrix I_i. These orthonormalization of the weight matrix is unnecessary but effective way to avoid matrix inversion in the calculation of the conditional entropy (12) and its derivative, alleviating the computational complexities.

Now a new greedy extraction algorithm is proposed. In this algorithm, the mutual information is calculated as in Section II using the Parzen window density estimation, thus it is named as 'Parzen window feature extractor (PWFX)'. The algorithm is as follows:

I. (Initialization) set $\mathcal{F} \longleftarrow$ "empty set."

II. (Sphering by PCA) transform the original features $\boldsymbol{X}$ into $\boldsymbol{Y} = W_{pca}^T\boldsymbol{X}$ to have zero mean and an $N' \times N'$ identity covariance matrix; $\Sigma_{\boldsymbol{Y}} = I_{N'}$.

III. (Greedy extraction) for $i = 1, \cdots, M$, repeat the following.

1. (Randomize weight) generate N' dimensional random weight $\boldsymbol{v}$.
2. (Orthonormalization) Orthonormalize the weight $\boldsymbol{v}$ by Gram-Schmidt method;

$$\begin{aligned}\boldsymbol{v} &\longleftarrow \boldsymbol{v} - \sum_{j=1}^{i-1}(\boldsymbol{v}^T\boldsymbol{v}_j^*/||\boldsymbol{v}_j^*||^2)\boldsymbol{v}_j^* \\ \boldsymbol{v} &\longleftarrow \boldsymbol{v}/||\boldsymbol{v}||.\end{aligned} \tag{17}$$

3. (Weight update) update weight by gradient descent method;
   (a) (Gradient calculation) calculate $\nabla_{\boldsymbol{v}}H(C|\boldsymbol{V}_i^T\boldsymbol{Y})$
   (b) (Weight update) reserve old weight and update weight;

$$\begin{aligned}\boldsymbol{v}_{old} &\longleftarrow \boldsymbol{v} \\ \Delta\boldsymbol{v} &\longleftarrow -\mu\nabla_{\boldsymbol{v}}H(C|\boldsymbol{V}_i^T\boldsymbol{Y}) \\ \boldsymbol{v} &\longleftarrow \boldsymbol{v} + \Delta\boldsymbol{v}.\end{aligned} \tag{18}$$

Here μ is the learning rate.

   (c) (Orthonormalization) Orthonomalize the weight $\boldsymbol{v}$ to make $\Sigma_{\boldsymbol{F}_i} = E\{\boldsymbol{F}_i\boldsymbol{F}_i^T\} = I_i$. The procedure is the same as (17).
   (d) (Convergence check) check if $||\boldsymbol{v} - \boldsymbol{v}_{old}|| < \epsilon$ or the number of iterations reached MAX_ITER. If it is, goto step 4. Otherwise, goto step 3-(a).
4. (Extraction of the next feature) set $\boldsymbol{v}_i^* = \boldsymbol{v}$, $F_i = \boldsymbol{v}_i^{*T}\boldsymbol{Y}$ and $\boldsymbol{w}_i^* = W_{pca}\boldsymbol{v}_i^*$. $\mathcal{F} \longleftarrow \mathcal{F} \cup \{F_i\}$.

IV. Output the set $\mathcal{F}$ containing the extracted features.

The step III.3-(c) is used to make sure that the new feature candidate is uncorrelated to the already extracted features.

In the calculation of the conditional probability (11), the denominator of the exponent increases approximately in proportion to the number of extracted features increases, thus we set $h = \hbar_1\sqrt{k}$ when the dimension of $\boldsymbol{F}$ is i. Here, $\hbar_1$ is a constant.

Now, the remaining part is the calculation of the gradient $\nabla_{\boldsymbol{v}}H(C|\boldsymbol{V}_i^T\boldsymbol{Y}) \in \Re^{N'}$ in III.3-(a). Replacing $\boldsymbol{F}$ with $\boldsymbol{V}_i^T\boldsymbol{Y}$, (12) becomes

$$\hat{H}(C|\boldsymbol{V}_i^T\boldsymbol{Y}) = -\sum_{j=1}^{n}\frac{1}{n}\sum_{c=1}^{N_c}\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y}_j)\,\log\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y}_j), \tag{19}$$

where $\boldsymbol{y}_j$ represents the jth sample of the original data transformed by PCA.

Differentiating this with respect to $\boldsymbol{v}$, the last column of $\boldsymbol{V}_i$, we obtain

$$\begin{aligned}&\nabla_{\boldsymbol{v}}\hat{H}(C|\boldsymbol{V}_i^T\boldsymbol{Y}) \\ &= -\sum_{j=1}^{n}\frac{1}{n}\sum_{c=1}^{N_c}\nabla_{\boldsymbol{v}}\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y}_j)\{1+\log\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y}_j)\}.\end{aligned} \tag{20}$$

Rewriting (11) leads to

$$\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y}) = \frac{\sum_{l\in I^c}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)}{\sum_{k=1}^{N_c}\sum_{l\in I^k}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)}, \tag{21}$$

where $\varphi(\boldsymbol{z}) \triangleq \exp(-\frac{\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z}}{2h^2})$ and $\tilde{\boldsymbol{y}}_l \triangleq \boldsymbol{y} - \boldsymbol{y}_l$.

By differentiating this w.r.t. $\boldsymbol{v}$, $\nabla_{\boldsymbol{v}}\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y})$ becomes

$$\begin{aligned}\nabla_{\boldsymbol{v}}\hat{p}(c|\boldsymbol{V}_i^T\boldsymbol{y}) &= \frac{\sum_{l\in I^c}\nabla_{\boldsymbol{v}}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)}{\sum_{k=1}^{N_c}\sum_{l\in I^k}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)} \\ &- \frac{[\sum_{l\in I^c}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)][\sum_{k=1}^{N_c}\sum_{l\in I^k}\nabla_{\boldsymbol{v}}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)]}{[\sum_{k=1}^{N_c}\sum_{l\in I^k}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l)]^2}.\end{aligned} \tag{22}$$

Because $\boldsymbol{v}$ is always orthonormalized in the step III.3-(c), new feature candidate is uncorrelated with already extracted features, resulting $\Sigma_{\boldsymbol{F}_i} = I_i$. Replacing Σ with this

in (5), and differentiating it w.r.t. $\boldsymbol{v}$, $\nabla_{\boldsymbol{v}}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l) \in \Re^{N'}$ can be obtained as follows:

$$\begin{aligned}\nabla_{\boldsymbol{v}}\varphi(\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l) &\simeq \nabla_{\boldsymbol{v}} \exp(-\frac{\tilde{\boldsymbol{y}}_l^T\boldsymbol{V}_i\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l}{2h^2}) \\ &= -\frac{1}{h^2}\exp(-\frac{\tilde{\boldsymbol{y}}_l^T\boldsymbol{V}_i\boldsymbol{V}_i^T\tilde{\boldsymbol{y}}_l}{2h^2})(\tilde{\boldsymbol{y}}_l\boldsymbol{v}^T\tilde{\boldsymbol{y}}_l).\end{aligned} \tag{23}$$

Finally, the computation of (20) is completed and the PWFX algorithm can be implemented. The computational complexity of this method is proportional to square of the number of samples, n^2.

4. EXPERIMENTAL RESULTS

Suppose we have four independent input features x_1, x_2, x_3 and x_4 uniformly distributed on [-1,1] for a binary classification, and the output class c is determined as follows:

$$c = \begin{cases} 0 & \text{if } x_1 + 4x_2 < 0 \\ 1 & \text{if } x_1 + 4x_2 \geq 0. \end{cases}$$

To see the performance of feature extraction algorithms to noisy data, five sets of data were generated where the class c was randomly flipped with probability of 0 to 0.4. Each dataset contains 500 samples on which PWFX was performed. Also features found from LDA [5] and ICA-FX [6] were used for comparison. These feature extraction methods were tested on a separate set of test data consisting of 500 samples with no flip of class information.

As can be seen, this problem is linearly separable and the optimal feature is $f^* = x_1 + 4x_2$. When we performed PWFX for no flipped case, the newly extracted feature is $f = -7.73x_1 - 30.39x_2 + 0.54x_3 - 1.07x_4$ which is very close to f^*.

Table 1 is the classification performances of various feature extraction methods on these datasets. One feature is extracted with each method. Averages and standard deviations of 10 experiments are reported here. The standard multi-layer perceptron (MLP) with one hidden layer was used for the classification. Three hidden nodes were used with learning rate of 0.1 and momentum of 0.9. The number of iterations was set to 100. In PWFX, the window width parameter for k extracted feature is computed $h = \hbar_1\sqrt{k}$ where $\hbar_1$ was set to 0.3.

In the table, the performances of LDA, ICA-FX, and PWFX becomes gradually worse as more training samples are flipped. In this example, PWFX is slightly better than ICA-FX which is again slightly better than LDA.

5. CONCLUSIONS

In this paper, we have proposed a new method of feature extraction for classification problems. The proposed method searches for the direction where the mutual information between the extracted features and the class labels are maximized. Although the mutual information is a very good indicator of the relevance between variables, the reasons why it is not widely used is its computational difficulties, especially for continuous multi-variables. To overcome this, the proposed method makes use of the Parzen window in getting the conditional density in a feature space. With this method, we can compute the mutual information between output class and multiple input features without requiring a large amount of memory. The stochastic gradient ascent method was used to maximize the mutual information. In addition, greedy extraction scheme was used in adding new features. Though the restriction that the newly extracted feature is orthogonal to the already extracted features is unnecessary, it simplifies the gradient calculation greatly. The computational complexity of the proposed method is proportional to the square of the sample size.

We have applied the proposed method for several classification problems including face recognition problems and obtained better or compatible performances than those of LDA and ICA-FX. These results will be reported in the future.

Table 1. Classification performance for the separable $x_1 + 4x_2$ problem (Averages of 10 experiments. Numbers in the parentheses are the standard deviations.)

% of flips	Classification error (%) (MLP)		
	LDA	ICA-FX	**PWFX**
0	2.90 (0.44)	3.17 (0.81)	**1.61 (0.49)**
10	4.22 (0.45)	3.45 (1.20)	**2.01 (1.04)**
20	5.25 (0.56)	5.01 (1.76)	**4.19 (1.18)**
30	9.08 (0.73)	8.22 (1.51)	**6.62 (1.20)**
40	15.48 (0.93)	11.56 (2.08)	**10.93 (1.52)**

6. REFERENCES

[1] K. Torkkola and W.M. Campbell, "Mutual information in learning feature transformations," in *Proc. Int'l Conf. Machine Learning*, Stanford, CA, 2000.

[2] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on parzen window," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1667–1671, Dec. 2002.

[3] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statistics*, vol. 33, pp. 1065–1076, Sept. 1962.

[4] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[5] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, second edition, 1990.

[6] N. Kwak and C.-H. Choi, "Feature extraction based on ica for binary classification problems," *To appear in IEEE Trans. on Knowledge and Data Engineering*, 2003.