REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Reprint

4. TITLE AND SUBTITLE

TITLE ON ~~REPRINT~~

5. FUNDING NUMBERS

DAAG55-98-1-0230

6. AUTHOR(S)

AUTHOR(S) ON ~~REPRINT~~

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)

University of Texas - Austin

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARO 37634.22-PH

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

ABSTRACT ON ~~REPRINT~~

20011023 052

14. SUBJECT TERMS

15. NUMBER IF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

A Bayesian Pairwise Classifier for Character Recognition

Shailesh Kumar[1], Joydeep Ghosh[1] and Melba Crawford[2]
Department of Electrical and Computer Engineering[1]
Department of Mechanical Engineering[2]
University of Texas at Austin

Abstract

In this chapter, we develop a Bayesian Pairwise Classifier framework that is suitable for pattern recognition problems involving a moderately large number of classes, and apply it to two character recognition datasets. A C class pattern recognition problem (e.g. $C = 26$ for recognition of English Alphabet) is divided into a set of $\binom{C}{2}$ two-class problems. For each pair of classes, a Bayesian classifier based on a mixture of Gaussians (MOG) is used to model the probability density functions conditioned on a single feature. A forward feature selection algorithm is then used to grow the feature space, and an efficient technique is developed to obtain a MOG in the larger feature space from the MOG's in the smaller spaces. Apart from improvements in classification accuracy, the proposed architecture also provides valuable domain knowledge such as identifying what features are most important in separating a pair of characters, relative distance between any two characters, etc.

1 Introduction

There are two phases in a typical pattern recognition problem: The *learning/training phase* and the *generalization phase.* In the learning phase, a predictor or classifier is designed from already labeled training examples. In the generalization phase, a novel example is assigned a class label by the trained classifier. The ability of the classifier to generalize to novel examples not seen during training is central to pattern recognition. It is typically measured in terms of the *empirical generalization accuracy* defined as the fraction of novel examples (test examples) that were assigned the right class label by the trained classifier.

Depending on the domain of application, the raw input could be a set of observed properties (e.g. symptoms of a disease), a one dimensional signal (e.g. voice recognition, text recognition etc.), or even an image (e.g. face recognition, character recognition etc.). It is neither feasible nor practical to learn a mapping from such complex input spaces to class labels. Hence, a preprocessing stage involving data conditioning followed by *feature extraction* is used to transform the raw sensory input into a small set of features that the classifier can operate on. This is all the more true for character recognition problems where the input is an image of handwritten characters. Thus, the two stage learning process can be expressed in terms of a pair of mappings:

$$\mathbf{x} \in \mathcal{I} \xrightarrow{\Psi} \mathbf{y} \in \mathcal{F} \xrightarrow{\Phi} \omega \in \mathbf{\Omega}, \tag{1}$$

where the first mapping $\Psi : \mathcal{I} \rightarrow \mathcal{F}$, called the *feature extractor*, transforms an input vector $\mathbf{x}$ in the *input space* $\mathcal{I}$ into a feature vector $\mathbf{y}$ in some feature space $\mathcal{F}$, and the second mapping $\Phi : \mathcal{F} \rightarrow \mathbf{\Omega}$, called the *classifier*, assigns a class label $\omega \in \mathbf{\Omega} = \{\omega_1, \omega_2, \ldots, \omega_C\}$ to the feature vector $\mathbf{y}$. For example, in the first character recognition problem that is considered in this chapter, the input images of characters are transformed into a set of 16 properties such as mean positions of on pixels, their variance, and mean edge count from left to right and bottom to top, etc. In the second problem, 30 tangent vectors are computed from each character image. Although domain knowledge is used to extract these features and reduce the 16384 dimensional (a 128× 128 character image) input space to a 30 dimensional feature space, it is not necessary that all the extracted features will be actually useful in classification. Hence a smaller set of these

VTRD Book chapter

1.2 Probabilistic learning framework

Once a suitable feature space $\mathcal{F}$ is obtained by feature extraction/selection methods, one has to discriminate among different classes in the feature space $\mathcal{F}$. The input to the classifier mapping $\Phi : \mathcal{F} \rightarrow \mathbf{\Omega}$ is a feature vector $\mathbf{y} = \Psi(\mathbf{x})$ for any input $\mathbf{x} \in \mathcal{I}$. The probabilistic learning framework, popular in the statistical pattern recognition community, is used for modeling Φ in this work. There are a number of classifiers with different properties that have evolved from this framework.

In the probabilistic learning framework [1, 6], input patterns and class labels are assumed to be stochastically independent and identically distributed random variables X and Ω respectively. Since feature vectors are obtained from the input patterns, they are also assumed to be random variables Y. In the following description of the probabilistic learning framework, only variables Y and Ω are used since once Φ is fixed, for every X there is a corresponding Y. The salient features of the probabilistic learning framework are as follows:

- Y and Ω are sampled from an unknown joint probability density function $p_{Y,\Omega}(\mathbf{y}, \omega)$.
- Input patterns belong to one of the C classes with the *prior probability* of a sample being in class ω_c given by $P(\Omega = \omega_c) \equiv P(\omega_c)$. The priors are constrained by $\sum_{c=1}^{C} P(\omega_c) = 1$.
- The overall probability density function $p(\mathbf{y})$ is a mixture of C class conditional probability density functions computed in the feature space i.e. $\{p(Y = \mathbf{y}|\Omega = \omega_c) \equiv p(\mathbf{y}|\omega_c)\}_{c=1}^{C}$:

$$p(\mathbf{y}) = \sum_{c=1}^{C} P(\omega_c) p(\mathbf{y}|\omega_c) \tag{4}$$

- The *posteriori probability* $P(\Omega = \omega_c|Y = \mathbf{y}) \equiv P(\omega_c|\mathbf{y})$ of pattern $\mathbf{y}$ belonging to class ω_c is given by the Bayes rule:

$$P(\omega_c|\mathbf{y}) = \frac{P(\omega_c) p(\mathbf{y}|\omega_c)}{p(\mathbf{y})}, \tag{5}$$

  where the denominator (see 4) is a normalizing factor such that $\sum_{c=1}^{C} P(\omega_c|\mathbf{y}) = 1$.

- The classifier $\Phi(\mathbf{y})$ tries to estimate these posterior probabilities $\{\hat{P}(\omega_c|\mathbf{y})\}_{c=1}^{C}$. Using these estimates, it assigns class label $\omega(\mathbf{y})$ based on the *maximum aposteriori probability* (MAP) rule,

$$\Phi(\mathbf{y}) = \omega(\mathbf{y}) = \arg \max_{c=1\ldots C} \hat{P}(\omega_c|\mathbf{y}). \tag{6}$$

- The *misclassification error* for the MAP rule is given by

$$\mathcal{E}_{\text{MAP}}(\Phi) = \int_{\mathbf{y}} (1 - \max_{c=1\ldots C} \hat{P}(\omega_c|\mathbf{y})) d\mathbf{y}. \tag{7}$$

- A *training set* $\mathcal{X} = \{\mathcal{X}_c\}_{c=1}^{C} \subset \mathcal{I}$, where $\mathcal{X}_c$ is the set of training inputs in class ω_c, is available for supervised learning of Φ. After feature extraction, the corresponding training data is denoted by $\mathcal{Y} = \{\mathcal{Y}_c\}_{c=1}^{C} \subset \mathcal{F}$, where for each sample $\mathbf{x} \in \mathcal{X}$, there is a corresponding $\mathbf{y} \in \mathcal{Y}$, such that $\mathbf{y} = \Psi(\mathbf{x})$.

1.3 Classifier Taxanomy

There are two broad categories into which most of the classifier architectures can be divided: DENSITY BASED and REGRESSION BASED [7].

1. DENSITY BASED classifiers estimate the class conditional probabiliy density functions $\{p(\mathbf{y}|\omega_c)\}_{c=1}^{C}$ and use these to compute the aposteriori probabilities using the Bayes rule (5). Once the estimated aposterior probabilities are available, the MAP rule (6) can be used to assign $\mathbf{y}$ a class label $\omega(\mathbf{y})$. In

2.1 Pairwise Classifier Architecture

Figure 1 shows the BPC framework. Each classifier ϕ_{ij} has an associated feature extractor denoted by $\psi_{ij} : \mathcal{I} \rightarrow \mathcal{F}_{ij}$ that transforms an input $\mathbf{x} \in \mathcal{I}$ into a feature vector $\psi_{ij}(\mathbf{x}) \in \mathcal{F}_{ij}$. The output of ϕ_{ij} is an estimate of the posterior probability $P_{ij}(\omega_i|\psi_{ij}(\mathbf{x}))$ ($P_{ij}(\omega_j|\psi_{ij}(\mathbf{x})) = 1 - P_{ij}(\omega_i|\psi_{ij}(\mathbf{x}))$).

Each ϕ_{ij} is implemented as a Bayesian classifier that uses two mixture of Gaussians (MOG), one for class ω_i and one for class ω_j, to model the probability density functions $p(\psi_{ij}(\mathbf{x})|\omega_k)$, $k = i, j$:

$$\hat{p}(\psi_{ij}(\mathbf{x})|\omega_k) = \sum_{\alpha=1}^{n_k^{(i,j)}} \pi_{k,\alpha}^{(i,j)} \mathcal{G}\left(\psi_{ij}(\mathbf{x}); \mu_{k,\alpha}^{(i,j)}, \Sigma_{k,\alpha}^{(i,j)}\right), \tag{10}$$

where $n_k^{(i,j)}$ is the number of Gaussians in the mixture for class ω_k, and $\mu_{k,\alpha}^{(i,j)}$ ($\in \mathcal{F}_{ij}$) and $\Sigma_{k,\alpha}^{(i,j)}$ are the mean vector and covariance matrix of the α^{th} Gaussian in the mixture of class ω_k for the classifier ϕ_{ij}. The Gaussian function $\mathcal{G}$ is given by:

$$\mathcal{G}(\mathbf{y}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left[-\frac{1}{2}(\mathbf{y} - \mu)^T \Sigma^{-1} (\mathbf{y} - \mu)\right], \tag{11}$$

where $\mathbf{y} = \psi_{ij}(\mathbf{x})$ and $d = |\mathcal{F}_{ij}|$ = the dimensionality of the feature space $\mathcal{F}_{ij}$. Bayes rule is used to compute the classifier output:

$$\phi_{ij}(\psi_{ij}(\mathbf{x})) = \hat{P}_{ij}(\omega_i|\psi_{ij}(\mathbf{x})) = \frac{\hat{p}(\psi_{ij}(\mathbf{x})|\omega_i)\hat{P}_{ij}(\omega_i)}{\hat{p}(\psi_{ij}(\mathbf{x})|\omega_i)\hat{P}_{ij}(\omega_i) + \hat{p}(\psi_{ij}(\mathbf{x})|\omega_j)\hat{P}_{ij}(\omega_j)}, k = i, j, \tag{12}$$

where $\hat{P}_{ij}(\omega_k)$ are the estimated class priors based on the training data:

$$\hat{P}_{ij}(\omega_k) = \frac{|\mathcal{X}_k|}{|\mathcal{X}_i| + |\mathcal{X}_j|}. \tag{13}$$

The problem of finding the right set of features ($\mathcal{F}_{ij}$) and the set of parameters $\{\pi_{k,\alpha}^{(i,j)}, \mu_{k,\alpha}^{(i,j)}, \Sigma_{k,\alpha}^{(i,j)}\}$, $\forall \alpha = 1 \ldots n_k^{(i,j)}$, $k = i, j$, and that of finding the right number of mixtures $n_k^{(i,j)}$ are discussed next.

2.2 Feature Selection

Feature selection is done separately and independently for each pairwise classifier. Let $\mathbf{F} = \{1, 2, \ldots, D\}$ denote the index set of all features and $\mathbf{y} = \psi_{ij}(\mathbf{x}) \in \mathcal{F}_{ij} \subset \mathbf{F}$ denote the feature vector corresponding to $\mathbf{x}$. In order to select the most discriminating features for the class pair (ω_i, ω_j), a relevance $R(\mathcal{F}_{ij})$ is assigned to the feature set $\mathcal{F}_{ij}$ based on the log odds of estimated class posteriors over the training set $\mathcal{X}_i \cup \mathcal{X}_j$:

$$R(\mathcal{F}_{ij}) = \frac{1}{|\mathcal{X}_i|} \sum_{\mathbf{x} \in \mathcal{X}_i} \log \frac{\hat{P}_{ij}(\omega_i|\psi_{ij}(\mathbf{x}))}{\hat{P}_{ij}(\omega_j|\psi_{ij}(\mathbf{x}))} + \frac{1}{|\mathcal{X}_j|} \sum_{\mathbf{x} \in \mathcal{X}_j} \log \frac{\hat{P}_{ij}(\omega_j|\psi_{ij}(\mathbf{x}))}{\hat{P}_{ij}(\omega_i|\psi_{ij}(\mathbf{x}))} \tag{14}$$

Note that the relevance depends on estimates of pairwise posteriors of $P_{ij}(\omega_k|\psi_{ij}(\mathbf{x}))$, which in turn depends both on the feature space $\mathcal{F}_{ij}$ and the parameters used for modeling the pdf in Equation (10). The algorithm for feature selection for the class pair (ω_i, ω_j) is summarized below:

1. Initialize $\mathcal{F}_{ij} = \text{argmax}_{f \in \mathbf{F}} R(f)$.
2. Augment the feature set sequentially as follows:
   (a) Find the next best feature $\tilde{f}$ to add to $\mathcal{F}_{ij}$:

$$\tilde{f} \leftarrow \arg \max_{f \in \mathbf{F} - \mathcal{F}_{ij}} R(\mathcal{F}_{ij} + f), \tag{15}$$

   where $(\mathcal{F}_{ij} + f)$ denotes the feature set formed by augmenting feature f in the feature set $\mathcal{F}_{ij}$.

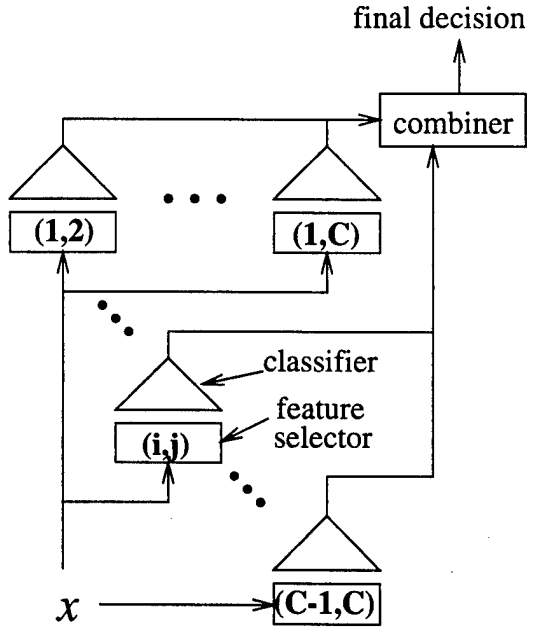

Figure 1: Pairwise classifier architecture: $\binom{C}{2}$ pairwise classifiers with respective feature selectors

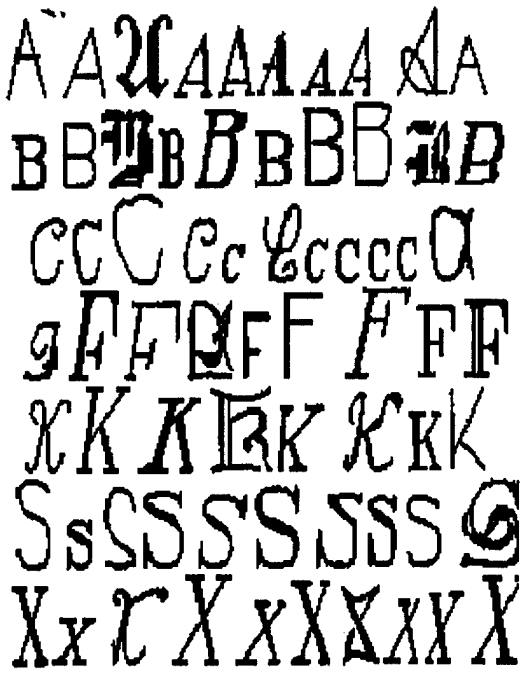

Figure 2: Some examples of letters in LETTER-I dataset [21]

2.3 Combining the pairwise classifiers

The outputs of the $\binom{C}{2}$ classifiers can be combined to obtain the final output in two ways: (i) by simple voting [22], or (ii) by using the MAP rule on an estimate of the overall aposterior probabilities obtained from the outputs of the pairwise classifiers [23]. In the voting combination scheme, a count $c(\omega_k|\mathbf{x})$ of the number of $\binom{C}{2}$ classifiers that labeled $\mathbf{x}$ into class ω_k,

$$c(\omega_k|\mathbf{x}) = \sum_{i<k} I(\phi_{ik}(\psi_{ik}(\mathbf{x})) < 0.5) + \sum_{i>k} I(\phi_{ki}(\psi_{ki}(\mathbf{x})) \geq 0.5), \tag{25}$$

is used. Here $I(bool)$ is the indicator function, which is 1 when the *bool* argument is true, and 0 otherwise. The input $\mathbf{x}$ is assigned the class label for which the count is maximum. i.e. $\omega(\mathbf{x}) = \text{argmax}_{k=1\ldots C}\, c(\omega_k|\mathbf{x})$.

In another approach to combining pairwise classifiers, proposed recently [23], the overall posterior probabilities $p_i = P(\omega_i|\mathbf{x})\ \forall i = 1\ldots C$ are estimated for some $\mathbf{x}$ from the $\binom{C}{2}$ posterior probabilities given by 12 as follows. Denote $m_{ij} = |\mathcal{X}_i| + |\mathcal{X}_j|$, $r_{ij} = \phi_{ij}(\psi_{ij}(\mathbf{x}))$ and $\nu_{ij} = \frac{\hat{p}_i}{\hat{p}_i + \hat{p}_j}$. The goal is to find an estimate $\hat{p}_i$ of true posteriors $P(\omega_i|\mathbf{x})$ such that ν_{ij} is close to r_{ij}, $\forall i \neq j$. Since there are $C-1$ independent parameters but $\binom{C}{2}$ equations, it is not possible in general to estimate $\hat{p}_i$ so that $\nu_{ij} = r_{ij}\ \forall i \neq j$. Hence only an approximate solution is sought. The closeness criteria that forms the objective function for finding $\mathbf{p} = (p_1, p_2, \ldots p_C)$ is the weighted KL-distance between r_{ij} and ν_{ij}:

$$J(\mathbf{p}) = \sum_{i<j} m_{ij} \left[r_{ij} \log \frac{r_{ij}}{\nu_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \nu_{ij}} \right] \tag{26}$$

This results in the following algorithm.

1. Start from an initial guess for $\hat{p}_i = \frac{|\mathcal{X}_i|}{|\mathcal{X}|}$, and evaluate corresponding ν_{ij} using the definition above.
2. Repeat the following updates for $i = 1, 2, \ldots, C, 1, 2, \ldots$ till convergence:

$$\hat{p}_i \leftarrow \hat{p}_i \frac{\sum_{j \neq i} m_{ij} r_{ij}}{\sum_{j \neq i} m_{ij} \nu_{ij}}, \tag{27}$$

Classifier	LETTER-I		LETTER-II	
	Train	Test	Train	Test
k-NN	91.2 (0.31)	89.9 (0.42)	91.9 (0.39)	89.5 (0.44)
MLP	80.2 (0.67)	79.3 (0.73)	79.3 (0.71)	76.2 (0.81)
MLC	84.4 (0.34)	82.7 (0.49)	81.4 (0.44)	79.5 (0.51)
BPC(1,V)	87.2 (0.45)	85.4 (0.57)	83.7 (0.52)	82.1 (0.63)
BPC(1,M)	87.6 (0.39)	85.3 (0.49)	84.9 (0.51)	83.3 (0.59)
BPC(n,V)	88.9 (0.26)	86.2 (0.33)	85.6 (0.36)	83.1 (0.40)
BPC(n,M)	89.5 (0.24)	87.6 (0.35)	87.9 (0.39)	86.3 (0.46)

Table 1: Average Training and test Accuracy (standard deviations) for multi layered perceptrons(MLP), Maximum likelihood classifier (MLC), Bayesian pairwise classifier with single Gaussian with voting combination method (BPC(1,V)), and MAP estimate combination (BPC(1,M)), and Bayesian pairwise classifier with mixture of Gaussian for voting (BPC(n,V)) and MAP estimate (BPC(n,M)) combination.

Classifier	LETTER-I		LETTER-II	
	FEATURES	GAUSSIANS	FEATURES	GAUSSIANS
BPC(1)	11.3	1.0	13.5	1.0
BPC(n)	8.2	1.1	9.8	1.5

Table 2: Number of features used (FEATURES) and number of Gaussians in the mixture of Gaussians pdf's (GAUSSIANS), averaged over all the $\binom{C}{2}$ Pairwise classifiers for both BPC(1) and BPC(n) case.

number of features required. Distribution for total usage of different features over all the pairwise classifiers is shown in Figure 3. For both BPC(1) and BPC(n), the distribution of usage of different features looks considerably similar for both datasets. In BPC(n) classifiers, the average number of Gaussians per pairwise classifier is more than 1 but the number of features required is significantly less than those required by BPC(1) classifier. Further, the fact that some pairwise classifiers required more than 1 Gaussians per class to model their pdf's, shows that the data sets were not exactly unimodal and this explains the difference in performance of BPC(1) and BPC(n) classifiers.

4 Domain Knowledge Extraction

The pairwise architecture with Bayesian classifiers based on mixture of Gaussians, makes it possible to extract several kinds of domain knowledge from the trained predictors, for example, features that are useful for distinguishing between particular pair of classes, a measure of distance between classes etc. Domain knowledge extracted from the character recognition datasets is described in this section.

4.1 Overall Importance of Features

Figure 3 shows the histogram of the number of times a feature was actually used in the pairwise classifiers for both BPC(1) and BPC(n) variants. For the LETTER-I dataset, the least used feature was vertical position of the box (feature 2), and the most used feature was edge count from bottom to top (feature 15). This kind of domain knowledge could reduce the cost of measuring different properties (features) of the objects once it is known what properties (features) are more useful than others for the overall task. Such domain knowledge is very useful in applications like remote sensing classification problems, where certain types of sensors are more useful than others for a given application [19].

-	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	-	37	39	45	45	42	44	36	45	38	43	**48**	37	**48**	33	45	39	37	40	**48**	38	40	35	28	39	36
B	9	-	37	21	**11**	25	30	21	38	33	29	34	33	29	25	19	25	**13**	18	36	44	37	45	33	39	19
C	7	11	-	44	**16**	38	**15**	21	29	38	**16**	33	33	41	18	40	30	49	41	40	27	40	44	42	22	47
D	13	10	12	-	26	38	23	25	31	33	31	34	33	31	18	18	42	22	32	31	45	39	39	40	34	41
E	8	2	5	8	-	**41**	22	19	29	28	34	36	32	46	27	33	41	26	29	34	33	47	45	19	26	20
F	8	10	12	12	11	-	28	40	37	42	33	38	32	41	38	**16**	37	39	24	22	43	41	38	33	25	31
G	13	16	10	10	9	8	-	18	40	30	21	34	23	36	**15**	36	20	31	38	34	26	42	32	32	45	43
H	10	10	5	7	4	11	4	-	42	29	18	44	34	35	**13**	39	17	18	38	37	26	38	33	22	35	**53**
I	11	8	5	7	8	12	10	8	-	22	31	46	36	44	44	32	35	39	25	**49**	42	40	44	31	36	41
J	10	12	10	11	8	13	8	7	4	-	29	38	39	42	20	38	28	42	26	39	34	41	37	30	36	28
K	10	9	6	11	14	8	12	5	7	7	-	36	37	33	34	33	35	35	22	37	34	42	38	**13**	47	29
L	12	10	7	11	12	7	13	13	10	8	11	-	35	39	43	39	36	32	**51**	35	38	38	42	31	33	47
M	14	10	6	7	4	6	9	12	5	12	11	6	-	23	25	34	26	28	29	41	29	38	27	30	40	33
N	10	5	11	10	8	10	13	12	8	7	8	9	10	-	28	33	37	40	**48**	41	36	44	29	**43**	40	42
O	11	9	8	11	5	8	13	5	8	3	9	11	8	7	-	28	**16**	30	36	31	27	39	43	33	36	29
P	11	6	11	3	7	11	13	10	12	12	7	7	7	8	9	-	28	20	30	38	41	**48**	35	31	36	36
Q	10	8	10	12	15	12	8	5	7	8	11	15	7	12	8	9	-	39	27	36	18	41	37	32	42	39
R	9	12	13	9	5	11	10	9	11	11	11	8	9	12	12	4	13	-	**13**	35	38	40	38	38	44	47
S	12	8	13	13	11	12	13	9	7	9	9	**14**	4	9	10	9	9	6	-	42	41	43	42	25	37	28
T	10	9	9	7	8	9	8	11	11	9	8	6	7	9	7	11	9	8	13	-	34	35	39	34	33	33
U	7	9	10	12	6	10	9	10	7	7	9	7	7	8	8	9	5	8	9	7	-	42	42	36	38	42
V	7	8	7	7	11	12	10	12	6	8	12	6	12	12	9	13	9	8	9	11	10	-	24	33	28	41
W	7	11	10	7	8	10	7	7	7	5	11	6	7	8	10	8	10	7	8	7	10	12	-	36	32	40
X	4	9	12	13	8	13	13	7	11	13	5	8	4	13	9	11	11	13	8	13	10	6	5	-	34	38
Y	7	9	5	7	3	8	13	9	7	6	11	5	12	10	8	11	11	9	11	11	12	13	10	11	-	36
Z	8	4	10	10	9	9	11	12	13	11	9	12	3	6	5	10	2	11	12	10	6	7	5	11	7	-

Table 3: The lower triangular matrix entries denote the number of features used for each pair of classes and the upper triangular matrix denotes the rounded relevance measure in the corresponding feature space.

number of modes for a class in a given feature space. The BPC framework addresses both these problems efficiently by the growing and pruning algorithm described in section 2.2. It also highlights the fact that the number of modes in a distribution is conditioned on the feature space.

Consider the scatter plots of class pairs (B/M), (B/W), (D/W), and (H/W) in Figures 4 and 5. For all these cases a single Gaussian would not have been able to model the desired pdf's. Moreover, different number of Gaussians are required, in general, for the two classes within each pairwise classifier. Thus the flexibility of the BPC architecture in not only choosing the right features, but also in automatically deciding the right number of Gaussians to model the pdf's for different classes, was found to be useful for the two datasets. Such flexibility is not available in conventional classifiers. Thus domain knowledge about the feature space together with information about the number of modes of each class in the corresponding feature space can be extracted from the BPC architecture.

4.4 Distance between classes

In conventional methods where a single classifier is used for the whole C class problem, an estimate of the distance between two classes could be obtained from the "confusion matrix" of training/validation set. Higher is the number of class ω_i examples getting classified into class ω_j and vice versa, the "closer" are the two classes. Unfortunately, this kind of estimate is influenced by the type of the classifier used, instead of solely being a property of the domain itself.

The BPC framework provides a classifier independent measure of distance between class pair (ω_i, ω_j) in terms of the relevance function $R(\mathcal{F}_{ij})$. If $\mathcal{F}_{ij}$ is a feature space in which the discrimination between two classes is high, then smaller relevance implies that it is harder to distinguish between the classes, which in turn implies that the two classes are "close" to each other in some sense. Table 3 contains the relevance measures between all pairs of classes for LETTER-I dataset. Since relevance is symmetric, only the upper triangular matrix is used for relevance measures. The lower triangular matrix contains the number of features required to distinguish between two classes. Relevance measures that are towards the lower and higher end are bold faced. The pair of classes that were found to be "close" to each other in terms of the relevance function using the LETTER-I dataset were (B/E), (B/R), (O/Q), (K/X), (P/F), etc. Classes that were found to be "distant" from each other were (H/Z), (L/S), (P/V), (A/L), etc. These results are particularly interesting as they show how the BPC framework is able to extract expected domain knowledge. The distance information is also useful, for example, to hierarchically cluster the characters.

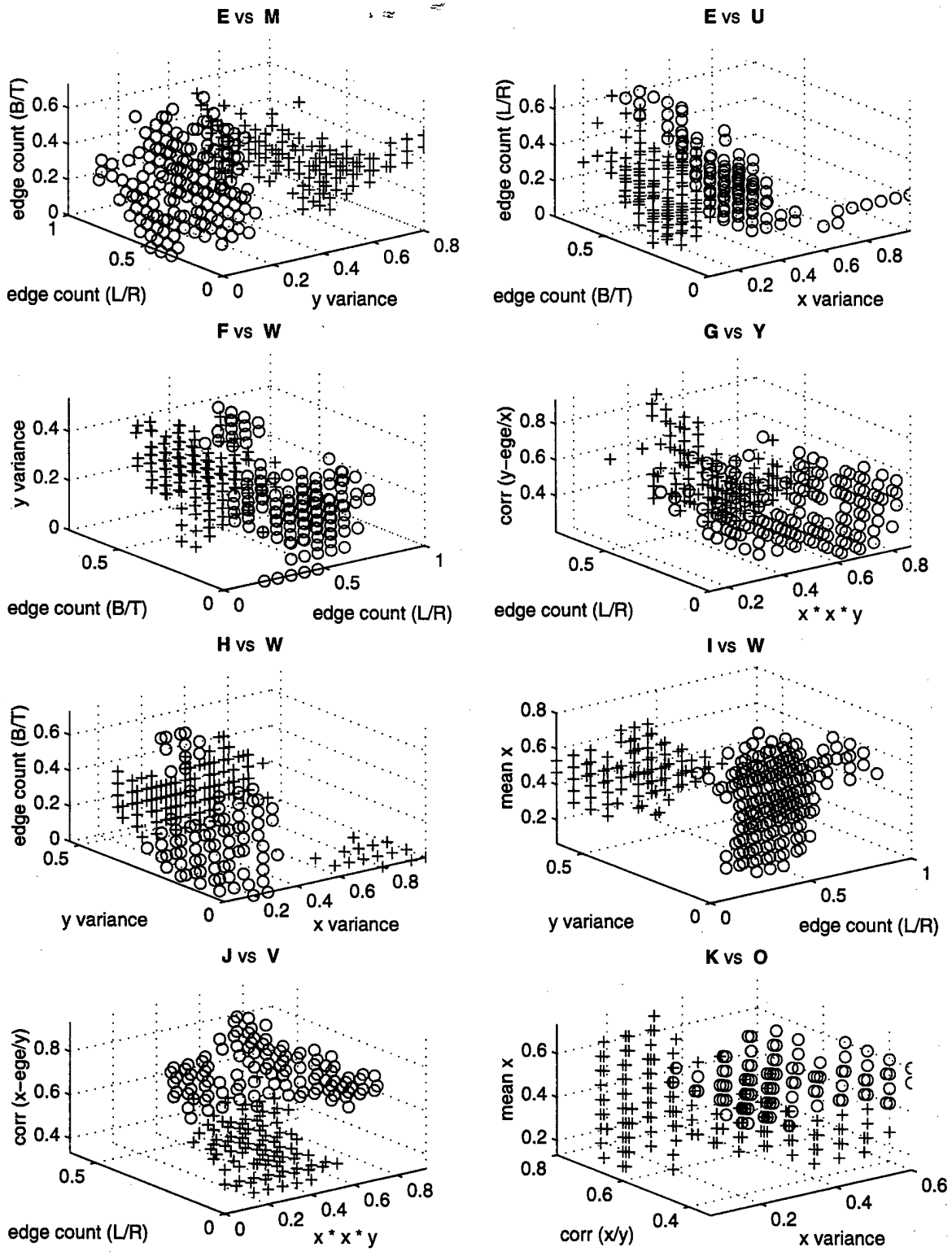

Figure 5: Data distribution for LETTER-I dataset in the first 3 dimensions of the feature space selected by BPC during the letter recognition task (+ vs *o* represent 2 classes)

[3] T. Marill and D. M. Green. On the effectiveness of receptors in recognition system. *IEEE Transactions on Information Theory*, 9:11–17, 1963.

[4] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26:917–922, 1977.

[5] J. Novovivova P. Pudil and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.

[6] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall, 1982.

[7] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja. Neural and statistical classifiers—taxonomy and two case studies. *IEEE Transactions on Neural Networks*, 8(1):5–17, January 1997.

[8] N. J. Nilsson. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems.* McGraw Hill, NY, 1965.

[9] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* Wiley, New York, 1992.

[10] J. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[11] K. Fukunaga and R. R. Hayes. The reduced parzen classifier. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):115–118, 1989.

[12] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[13] M.I. Jordan and R.A. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[14] P. Huber. Projection pursuit. *The Annals of Statistics*, 13:2:435–475, 1985.

[15] T. J. Hastie and R.J. Tibshirani. *Generalized Additive Models.* Chapman and Hall, 1990.

[16] J. H. Friedman. Multivariate adaptive regression splines. *Annal of Statistics*, 19:1–141, 1991.

[17] L. Xu, A. Krzyzak, and A. Yuille. On radial basis function nets and kernel regression: Statistical consistency, convergence rates and receptive field size. *Neural Networks*, 7:4:609–628, 1994.

[18] S. Kumar, M. M. Crawford, and J. Ghosh. A versatile framework for labeling imagery with large number of classes. In *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C., 1999.

[19] M. M. Crawford, S. Kumar, M.R.Ricard, J.C.Gibeaut, and A.Neuenshwander. Fusion of airborne polarimetric and interferometric SAR for classification of coastal environments. *IEEE Transactions on Geoscience and Remote Sensing*, 37(3), May 1999.

[20] A. Nag and J. Ghosh. Flexible resource allocating network for noisy data. In *SPIE Conf. on Applications and Science of Computational Intelligence, SPIE Proc. Vol. 3390*, pages 551–559, Orlando, Fl., April 1998.

[21] P. W. Frey and D. J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6(2), March 1991.

[22] Jerome H. Friedman. On bias, variance, loss, and the curse of dimensionality. Technical report, Department of Statistics, Stanford University, 1996.

[23] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In Michael J. Karens Michael I. Jordan and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 507–513, MIT Press, Cambridge Massachusetts, 1998.

[24] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1996. http://www.ics.uci.edu/ mlearn/MLRepository.html.