BU-CEIS-9709
New Trends in Artificial Intelligence & Neural Networks, (Proceedings of TAINN'97),
Turhan Ciftcibasi, Mustafa Karaman and Volkan Atalay (Eds.),
Kizilcahamam, May 22-23, 1997. pp. 190-194

Differential Diagnosis of Erythemato-Squamous Diseases using Voting Feature Intervals [1]

Gülşen Demiröz, H. Altay Güvenir, and Nilsel İlter*

Department of Computer Engineering and Information Science, Bilkent University, Ankara
{demiroz, guvenir}@cs.bilkent.edu.tr
* Department of Dermatology, School of Medicine, Gazi University, Ankara

Abstract

A new classification algorithm, called VFI (for *Voting Feature Intervals*), is developed and applied to Differential Diagnosis of Erythemato-Squamous Diseases. The domain contains records of patients with known diagnosis. Given a training set of such records the VFI classifier learns how to differentiate a new case in the domain. VFI represents a concept in the form of *feature intervals* on each feature dimension separately. Classification in the VFI algorithm is based on a real-valued voting. Each feature equally participates in the voting process and the class that receives the maximum amount of votes is declared to be the predicted class. The performance of the VFI classifier is evaluated empirically in terms of classification accuracy and running time.

1 Introduction

Inductive learning is a well-known approach to automatic knowledge acquisition instead of extracting knowledge from human experts. In several medical domains the inductive learning systems were actually applied; for example, two classification systems are used in localization of primary tumor, prognostics of recurrence of breast cancer, diagnosis of thyroid diseases, and rheumatology [5].

In this work a new non-incremental classifier called VFI (for *Voting Feature Intervals*) is developed and applied to Differential Diagnosis of Erythemato-Squamous Diseases domain. VFI uses *Feature Projections* as the knowledge representation scheme, which is used in CFP [3] and k-NNFP[1]. The rationale behind this knowledge representation is that human experts maintain knowledge in this form, especially in medical domains. The input to VFI is a set of training instances that are descriptions of patients with known diagnoses. Knowledge is represented as projections of the training dataset by *feature intervals* on each feature dimension separately. When classifying a new patient, each feature equally participates in the voting process and the diagnosis that receives the maximum amount of votes is predicted as the diagnosis of that patient. Since each feature participates in learning and classification independently, VFI enables an easy and natural way of handling missing feature values by simply ignoring those missing values.

The next section will describe the VFI algorithm in detail. In Section 3, the problem of Differential Diagnosis of Erythemato-Squamous Diseases is explained. In Section 4, the application of the VFI algorithm to this domain is discussed. Finally, the last section concludes with some remarks and plans for future work.

2 The VFI Algorithm

The VFI classification algorithm represents the concept with feature intervals, and makes a classification based on feature votes. It is a non-incremental classification algorithm; that is, all training examples are processed at once. Each training example is represented as a vector of feature values plus a label that represents the class of the example. From the training examples, the VFI algorithm constructs feature intervals for each feature. The term *interval* is used for feature intervals throughout the paper. An interval represents a set of values of a given

[1]This project is supported by TUBITAK (Scientific and Technical Research Council of Turkey) under Grant EEEAG-153.

```
train(TrainingSet):
begin
    for each feature f
        for each class c
            EndPoints[f] = EndPoints[f] ∪ find_end_points(TrainingSet, f, c);
        sort(EndPoints[f]);
        /* the range between the middle points of two consecutive endpoints form an interval */
        for each interval i /* on feature f */
            for each class c
                /* count the number of instances of class c falling into interval i */
                interval_class_count[f, i, c] = count_instances(f, i, c);
end.
```

Figure 1: Training in the VFI Algorithm.

feature, where the same subset of class values are observed. Two neighboring intervals contains a different set of classes. For each interval, a lower bound of the values and the number of examples of each class in that interval are maintained. Thus, an interval may represent several classes by storing the number of examples for each class.

The training process in the VFI algorithm is given in Figure 1. The lower bounds of intervals are learned by finding the *end points* for each feature and for each class. The procedure *find_end_points*(*TrainingSet, f, c*) finds the lowest and the highest values for feature *f* from the examples of class *c* in the *TrainingSet*. The lowest and highest values are called the *end points*, and for each feature there are $2k$ end points where k is the number of distinct classes. The list of end points is then sorted and the range between the middle points of each consecutive pair of end points in this sorted list constitutes an interval.

Each interval is represented by a vector of $< lower, count_1, \ldots, count_k >$ where $lower$ is the lower bound of that interval, $count_i$ is the number of training instances of class i that fall into that interval. When a training instance of class i falls on the boundary of two consecutive intervals of feature f, then $count_i$ of both intervals are incremented by 0.5 if f is a linear feature. On the other hand, if f is a nominal feature, $count_i$ of only the right interval is incremented. The $count_i$ values are computed by the $count_instances(i,\ c)$ function in Figure 1. The lower bounds are sufficient to represent an interval, because the upper bound of the interval is the lower bound of the next interval. In the training phase of the VFI algorithm the intervals for each feature dimension are constructed and these intervals make up the concept description. Note that since each feature is processed separately, no normalization of feature values is required.

The classification in the VFI algorithm is given in Figure 2. The process starts by initializing the votes of each class to zero. The classification operation includes a separate preclassification step on each feature. The preclassification of feature f involves a search for the interval on feature dimension f into which e_f falls, where e_f is the value test example e for feature f. If that value is unknown (missing), that feature does not participate in the classification process. Hence, the features containing missing values are simply ignored. Ignoring the feature about which nothing is known is a very natural and plausible approach.

If the value for feature f of example e is known, the interval i into which e_f falls is found. That interval may contain training examples of several classes. The classes in an interval are represented by their number of occurrences in that interval. For each class c, feature f gives a vote equal to

$$feature_vote[f, c] \ = \ \frac{interval_class_count[f,\ i,\ c]}{class_count[c]}$$

where $interval_class_count[f, i, c]$ is the number of examples of class c which fall into interval i of feature

```
classify(e):
/* e: example to be classified */
begin
    for each class c
        vote[c] = 0
    for each feature f
        for each class c
            feature_vote[f, c] = 0 /*vote of feature f for class c*/
        if e_f value is known
            i = find_interval(f, e_f)
            feature_vote[f, c] = interval_class_count[f, i, c] / class_count[c]
            normalize_feature_votes(f); /* such that sum_c feature_vote[f, c] = 1 */
            for each class c
                vote[c] = vote[c] + feature_vote[f, c];
    return class c with highest vote[c];
end.
```

Figure 2: Classification in the VFI Algorithm.

dimension f. If e_f falls on the boundary of two intervals i and $i+1$, then feature f gives a vote equal to

$$feature_vote[f, c] = \frac{interval_class_count[f,\ i,\ c] + interval_class_count[f,\ i+1,\ c]}{2 \times class_count[c]}$$

in cases when f is linear. On the other hand, if f is a nominal feature, only interval $i+1$ determines the vote of feature f. The individual vote of feature f for class c, $feature_vote[f, c]$, is then normalized to have the sum of votes of feature f equal to 1. Hence, the vote of feature f is a real-valued vote less than or equal to 1. Each feature f collects its votes in an individual vote vector $< vote_{f,1}, \ldots, vote_{f,k} >$, where $vote_{f,c}$ is the individual vote of feature f for class c and k is the total number of classes. After every feature completes their preclassification process, the individual vote vectors are summed up to get a total vote vector $< vote_1, \ldots, vote_k >$. Finally, the class with the highest vote from the total vote vector is predicted to be the class of the test instance.

3 Differential Diagnosis of Erythemato-Squamous Diseases

The differential diagnosis of erythemato-squamous diseases is a real problem in dermatology. They all share the clinical features of erythema and scaling, with very little differences. The diseases in this group are *psoriasis* (C_1), *seboreic dermatitis* (C_2), *lichen planus* (C_3), *pityriasis rosea* (C_4), *cronic dermatitis* (C_5), and *pityriasis rubra pilaris* (C_6). Usually a biopsy is necessary for the diagnosis but unfortunately these diseases share many histopathological features as well. Another difficulty for the differential diagnosis is that a disease may show the features of another disease at the beginning stage and may have the characteristic features at the following stages. Patients were first evaluated clinically with 12 features which are *erythema* (f_1), *scaling* (f_2), *definite borders* (f_3), *itching* (f_4), *koebner phenomenon* (f_5), *polygonal papules* (f_6), *follicular papules* (f_7), *oral mucosal involvement* (f_8), *knee and elbow involvement* (f_9), *scalp involvement* (f_{10}), *family history* (f_{11}), and *age* (f_{34}). Afterwards, skin samples were taken for the evaluation of 22 histopathological features which are *melanin incontinence*(f_{12}), *eosinophils in the infiltrate* (f_{13}), *PNL infiltrate* (f_{14}), *fibrosis of the papillary dermis* (f_{15}), *exocytosis* (f_{16}), *acanthosis* (f_{17}), *hyperkeratosis* (f_{18}), *parakeratosis* (f_{19}), *clubbing of the rete ridges* (f_{20}), *elongation of the rete ridges* (f_{21}), *thinning of the suprapapillary epidermis* (f_{22}), *spongiform pustule* (f_{23}), *munro microabcess* (f_{24}), *focal hypergranulosis* (f_{25}), *disappearance of the granular layer* (f_{26}), *vacuolisation and damage of basal layer* (f_{27}), *spongiosis* (f_{28}), *saw-tooth appearance of retes* (f_{29}), *follicular horn plug* (f_{30}), *perifollicular parakeratosis*

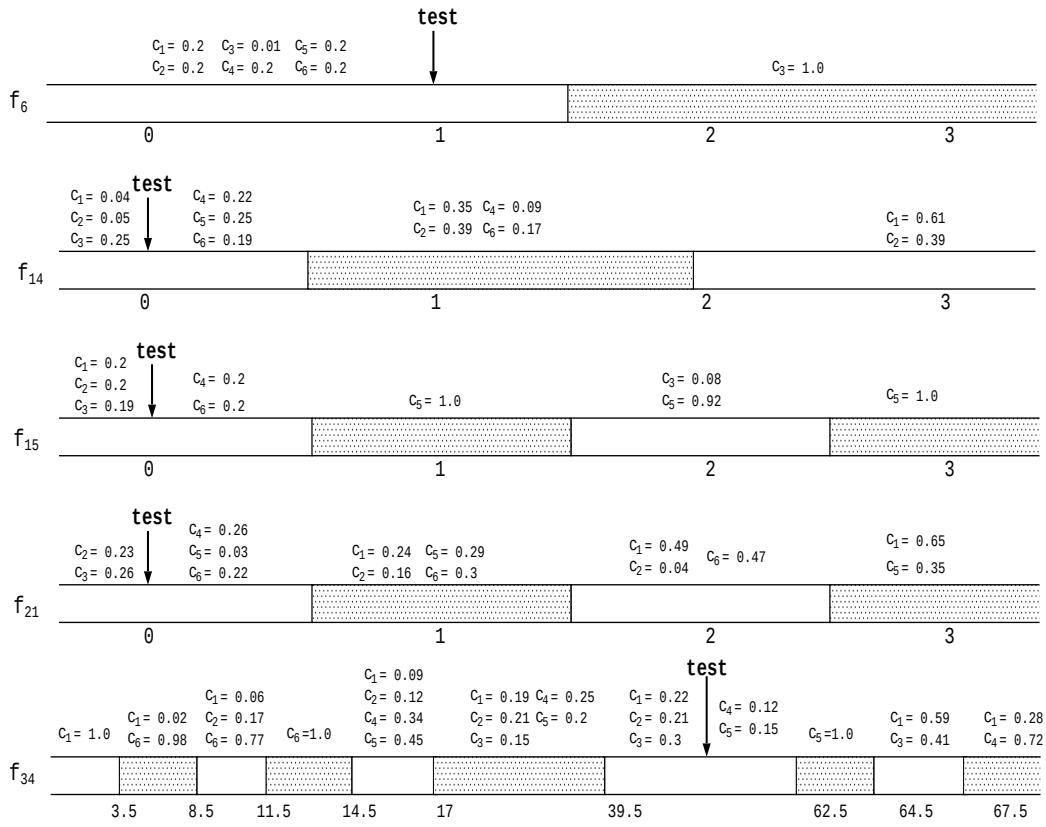

Figure 3: Some features and their intervals. The amount of vote for each class is given above the intervals. The classes that an interval gives 0 vote are not displayed.

(f_{31}), *inflammatory monoluclear inflitrate* (f_{32}), and *band-like infiltrate* (f_{33}). The values of the histopathological features are determined by an analysis of the samples under a microscope.

In the dataset constructed for this domain, the family history feature has the value 1 if any of these diseases has been observed in the family, and 0 otherwise. The age feature simply represents the age of the patient. Every other feature (clinical and histopathological) was given a degree in the range of 0 to 3. Here, 0 indicates that the feature was not present, 3 indicates the largest amount possible, and 1, 2 indicate the relative intermediate values.

4 Experiments

Currently, the dataset for the domain contains 287 instances. We first used all of these instances to obtain a description of the domain. The description consists of the feature intervals constructed for each feature. The intervals obtained for features f_6, f_{14}, f_{15}, f_{21} and f_{34} are shown in Figure 3.

It is clear from Figure 3 that the nonzero values of feature f_6 (polygonal papules) indicate the class C_3 (pityriasis rubra pilaris). On the other hand, the high values for f_{14} would suggest class C_1 or C_2. The feature f_{15} appears to a distinguishing feature for class C_5. However, high values of f_{21} can indicate both C_1 and C_5. Also, class C_6 appears to be a children's disease.

For a particular case, let us consider a patient, who has the following values for these features: $f_6 = 1$, $f_{14} = 0$, $f_{15} = 0$, $f_{21} = 0$, $f_{34} = 52$. For that patient, the vote vector for f_6 would be $< 0.2, 0.2, 0.01, 0.2, 0.2, 0.2 >$, for $f_{14} < 0.04, 0.05, 0.25, 0.22, 0.25, 0.19 >$, and so on. Then the votes for all classes received from all 34 features are summed. The class that receives the highest amount of votes is the class predicted.

We have also experimented the classification accuracy of the VFI algorithm. The classification accuracy of an algorithm is used as one measure of performance. The most commonly used classification accuracy metric is the percentage of correctly classified instances over all test instances. To measure the classification accuracy, 5-fold cross-validation technique is used in the experiments. That is, the whole dataset is partitioned into 5 subsets. The four of the subsets is used as the training set, and the fifth is used as the test set. This process is repeated 5 times once for each subset being the test set. Classification is the average of these 5 runs. This technique ensures that the training and test sets are disjoint. The VFI algorithm achieved 94.42% accuracy on this domain. The total time spent for training with 230 instances and testing with 57 instances was about 0.25 seconds on a Sparc20/61 computer.

5 Conclusions

In this paper, a new classification algorithm called VFI is developed and applied to Differential Diagnosis of Erythemato-Squamous Diseases. Since each feature is processed separately, the missing feature values that may appear both in the training and test instances are simply ignored in VFI. In other classification algorithms, such as decision tree inductive learning algorithms, the missing values require extra care [6]. This problem has been overcome by simply omitting the feature with the missing value in the voting process of VFI. Also note that the VFI algorithm, in particular, is applicable to concepts where each feature, independent of other features, can be used in the classification of the concept. One might think that this requirement may limit the applicability of the VFI, since in some domains the features might be dependent on each other. Holte has pointed out that the most datasets in the UCI repository are such that, for classification, their attributes can be considered independently of each other [4]. Also Kononenko claimed that in the data used by human experts there are no strong dependencies between features because features are properly defined [5]. Another advantage of the VFI classifier is that instead of a categorical classification, a more general probabilistic classification where the classifier returns a probability distribution over all classes is possible to implement with VFI.

For future work, we plan to integrate a feature weight learning algorithm to VFI, since both relevant and irrelevant features have equal vote in this version of the VFI algorithm. But more relevant features should have more voting power in classification. Genetic algorithms can be used to learn the optimum weights for VFI [2]. Another idea is to assign weights to intervals, since pure intervals representing only one class might be more effective in classification.

References

[1] Akkuş, A., & Güvenir, H. A. (1995). K Nearest Neighbor Classification on Feature Projections. *Proceedings of ICML'96*, 12–19.

[2] Demiröz, G., & Güvenir, H. A. (1996). Genetic Algorithms to Learn Feature Weights for the Nearest Neighbor Algorithm. *Proceedings of BENELEARN-96*, 117-126.

[3] Güvenir, H. A., & Şirin, İ. (1996). Classification by Feature Partitioning. *Machine Learning*, Vol. 23, 47–67.

[4] Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, Vol. 11, 63–91.

[5] Kononenko, I. (1993). Inductive and Bayesian Learning in Medical Diagnosis. *Applied Artificial Intelligence*, Vol. 7, 317-337.

[6] Quinlan, J. R. (1989). Unknown attribute values in induction. *Proceedings of 6th International Workshop on Machine Learning*, 164–168.