Multi-Label Imbalanced Data Enrichment Process in Neural Net Classifier Training

Gorn Tepvorachai, *Student Member, IEEE*, and Chris Papachristou, *Fellow, IEEE*

***Abstract*— Semantic scene classification, robotic state recognition, and many other real-world applications involve multi-label classification with imbalanced data. In this paper, we address these problems by using an enrichment process in neural net training. The enrichment process can manage the imbalanced data and train the neural net with high classification accuracy. Experimental results on a robotic arm controller show that our method has better generalization performance than traditional neural net training in solving the multi-label and imbalanced data problems.**

I. Introduction

SEMANTIC scene classification, robotic state recognition, and many other real-world applications and optimizations involve multi-label classification, where a set of inputs is associated with multiple labels (outputs). For example, in semantic scene, a detected object (inputs) can belong to more than one object class (multiple labels). In robotic arm controller, a robot arm angle position may exist in more than one controller state. Unfortunately, most traditional classifiers [1], [2] can only handle single-label problems—, where a set of inputs is mapped to only one label, such as identification for one object class or the recognition of one controller state—or balanced data, where the classifier training data is uniformly distributed over the data space.

On the other hand, many classification problems also involve imbalanced data, where sampled data for the classifier training is non-uniformly distributed over the data space. The imbalanced data problem can take two distinct forms: either one class is under-sampled relatively to other classes, or it is over-sampled but too sparse in the sampling space. For example, a robotic arm controller uses the arm angle position and the sub-controller circuit gain to determine the controller states. The sampling of the controller has one state sparsely over-sampled relatively to other states due to the application design specification. Other applications, such as facial image associative memory [3] and adaptive self-configurable filters [4], [5], are also compatible with this approach.

Most learning algorithms, like traditional neural nets [6] and support vector machines [1], are designed for well-balanced data and do not work well on imbalanced data. While a traditional classifier can achieve very high accuracy by simply ignoring the minority samples, this is obviously undesirable because the minority samples may contain critical information which makes such a classifier useless in practice.

In this paper, we introduce a data enrichment process for neural net training to address the multi-label and imbalanced data problem. This enrichment process manipulates the imbalanced data into a smaller subset of more balanced data which will be used in the training of a neural net classifier. This subset is initialized through imbalanced data clustering and cluster re-sampling to obtain equally represented data in Euclidean space. The enrichment process iteratively updates the subset throughout the training phase by incrementally adding and removing data to maintain and improve classifier performance. The experimental outputs of the classifier show that our method has better generalization performance than the traditional classifier training.

This paper is structured as follows. In the following section, Section II, some related works and our enrichment process overview are briefly introduced. In Section III, the proposed enrichment processes, including initialization, updates, and termination, are described in detail. In Section IV, we perform experiments on robotic arm control problems and compare our methods with traditional classifiers. Finally, some conclusions are drawn in Section V.

II. Background

A. Related works

In the recent literature, some promising works have been reported with the development of multi-label classification algorithms, such as non-mutually exclusive class definition in Boutella [7], relevance feedback with supervised learning paradigm in Dorado [8], and multi-label conditional random field classification model in Ghamrawi [9]. These techniques have been shown to work with support vector machines but not for neural nets. Additionally, they assume a sparsely distribution of the training data set.

A number of approaches have also been proposed to address the imbalanced data problem. Examples include over-sampling of the minority class samples in Chawla [10] and adjusting the misclassification costs of the two classes in Japkowicz [11]. Nonetheless, these works have shown to be effective for Bayesian classifiers and decision tree systems–not neural nets.

Regarding works on data enrichment on neural net training, a few techniques to improve the training process have been proposed and proven to be effective under certain application constrains. Liu [12], Patil [13], and Salazar [14]

This work was supported, in part, by School of Graduate Studies at Case Western Reserve University.

Gorn Tepvorachai and Chris Papachristou are with the Department of Electrical Engineering and Computer Science, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, Ohio 44106, USA (email: gxt16@case.edu and cap2@case.edu).

978-1-4244-1821-3/08/$25.00©2008 IEEE

propose the use of various training data clustering techniques to find cluster centroids using the cluster centroids as a training set for a single neural net. The results show good improvement; however, it is uncertain on how to select an appropriate number of clusters.

Another effort is to use multiple neural nets (ensemble) training on data clusters reported by Arslan [15], Cunningham [16], El-Gamal [17], Hartono [18], and Liu [19]. Each neural net is dedicated for training one data cluster. The results show better neural net performance in generalizing new inputs. Nevertheless, these approaches suffer from the use of multiple neural nets and their interpolation between the transition region of two adjacent clusters.

B. Overview of the Enrichment Process

In this work, we introduce the training data enrichment process (otherwise known as enrichment process), which is responsible for managing the balanced/imbalanced training data during the neural net classifier training. The enrichment process re-samples the imbalanced data into a smaller subset of more balanced data. This subset is initialized through imbalanced data clustering and cluster re-sampling to obtain equally represented data in Euclidean space. The enrichment process iteratively updates the subset throughout the training phase by incrementally adding and removing data to maintain and improve classifier performance. The enrichment process operates solely during the neural net training phase–not during the normal operation.

The difference between our approach and other works are that we show the multi-label and imbalanced data problem can be addressed with a single neural net. Our result accuracy and capability are comparable to support vector machines, Bayesian classifiers, and decision tree systems mentioned earlier. Additionally, we propose a data selection technique to improve neural net classifier training from the available imbalanced training set, which has not been addressed directly in existing works. Our approach avoids the assumption of sparsely distribution of the available training set as in [7], [8], [9]; it does not need data fusion techniques to integrate multiple neural net estimations (from ensemble) as in [16], [17], [18]. Though our approach uses a clustering technique, it does not require knowing the number of initial clusters in advance as in [12], [13], [14]; however, having that knowledge can be useful.

The enrichment process manages the training data within three steps:

1) *Enrichment Initialization* re-samples the available imbalanced training data or training set scope ($\mathbb{T}$) to create a subset of more balanced training data or active training set (τ) for the first time neural net classifier training.
2) *Enrichment Update* incrementally adds and removes training data to/from the current active training set (τ) at the end of an enrichment training iteration. Additional training data are requested from the training set scope ($\mathbb{T}$) to train the neural net. The modified active training set (τ) (after adding and removing of some data) remains a subset of the training set scope ($\mathbb{T}$).
3) *Enrichment Termination* controls the enrichment process to iterate for a pre-defined number of enrichment training iterations.

We will describe the first two steps of managing the imbalanced data in Section III. Then, we will illustrate their mechanisms and the benefits of the enrichment process in Section IV.

III. Our Approach

Our enrichment process to improve the performance of a single neural net is distinctly different from other existing works mentioned earlier. The enrichment process manipulates the available imbalanced training data or the training set scope ($\mathbb{T}$) by, first, systematically generating an initial subset of training data or an active training set (τ) by cluster re-sampling. Then, it incrementally modifies the subset during the neural net classifier training process. The data for neural net training is continuously added and removed with respected to the neural net bias to certain training data space (regions). The training data points which are relatively well-trained in the neural net should be dropped. On the other hand, each training data point which is relatively under trained should recruit more neighboring training data points. The incremental data modification process repeats to a predefined number of enrichment iterations. Figure 1 presents the pseudo code illustrating the enrichment process in three steps.

Step 1: Initialization
a) Select a number, g.
b) Cluster the training set scope into g clusters.
c) Select equal number of data from each cluster as active training set.

Step 2: Updates
a) Train the neural net with the active training set.
b) Separate the active training set into 3 groups respecting to their energy errors.
c) Add neighbors for high−group and remove half of the low−group.

Step 3: Termination
a) Check if the enrichment process repeats for a pre−defined number of iterations.
b) If NO, repeat Step 2. Otherwise, stop.

Figure 1
Pseudo code for enrichment process

A. Enrichment Initialization

The enrichment process starts from the available imbalanced training data or the training set scope ($\mathbb{T}$) provided by a human investigator or a knowledge base. As for the first step, the enrichment process re-samples an initial subset training data or active training set (τ) from the training set scope ($\mathbb{T}$) by applying a clustering algorithm based on Euclidean distance [1] . The clustering technique ($\mathcal{C}$) is initiated with an arbitrary initial number of clusters (g). If we have a priori knowledge of how the training set scope ($\mathbb{T}$) is scattered

in space, we can use the visual estimation for the initial number of clusters. Regardless of how the initial number of clusters is selected, the active training set (τ) will be balanced later on during the neural net training. These g clusters are solely used during the enrichment initialization; they will be discarded afterwards.

Once the training set scope ($\mathbb{T}$) has been formed into g clusters, the enrichment process selects m training data from each cluster to produce the initial active training set (τ) where m is the least number of training data points in all g clusters. The selection technique ($\mathcal{S}$) of m training data can be done by applying the same clustering algorithm ($\mathcal{C}$) to the cluster with m sub-clusters. The training data points, which are closest (by Euclidean distance [1]) to the sub-cluster centers, represent the selected m points of the cluster. Hence, we can write the initializing process of the more balanced subset training data or the initial active training set (τ) in the enrichment process as

$$\tau = \mathcal{S}\left(\mathcal{C}(\mathbb{T}, g)\right) \tag{1}$$

About the computational complexity, the enrichment initialization performs the clustering algorithm ($\mathcal{C}$) and the selection algorithm ($\mathcal{S}$) on the training set scope ($\mathbb{T}$). According to Jain [20] and Berkhin [21], the clustering algorithm, based on k-mean Euclidean distance, has complexity of $O(n \cdot \ln n \cdot r)$ order where n is the number of data ($|\mathbb{T}|$) and r is the information dimensionality of data (including input and output data). The selection algorithm applies the clustering algorithm onto all g clusters; this requires complexity $g \cdot O(n \cdot \ln n \cdot \frac{r}{g}) = O(n \cdot \ln n \cdot r)$. The enrichment initialization only occurs once at the beginning of the enrichment training. Therefore, the enrichment initialization computational complexity is of the order

$$O(n \cdot \ln n \cdot r) \tag{2}$$

B. Enrichment Update

The training process will use the active training set (τ) to train the neural net. Once the training is suspended after a pre-defined number of training iterations, the enrichment process re-examines the current training data by analyzing the energy error [2] (e_j) of each training data (t_j) in the active training set (τ). The energy error of the training data is normalized and compared against two preset thresholds (e^h, e^l where $e^h > e^l$). These thresholds separate the subset training data (τ) into three groups: high error (τ^h), middle error (τ^m), and low error (τ^l) groups, in Equation (3).

$$\begin{aligned} \tau^h &= \left\{t_j | t_j \in \tau, e_j \geq e^h\right\} \\ \tau^m &= \left\{t_j | t_j \in \tau, e^h > e_j > e^l\right\} \\ \tau^l &= \left\{t_j | t_j \in \tau, e^l \geq e_j\right\} \end{aligned} \tag{3}$$

With the three groups, we can concentrate neural net training on particular data which have not been very well-trained (indicated in high-error group). At the same time, we can ignore some data which are already well-trained (indicated in low-error group). The high-error group represents all current training data which are under trained in the neural net. The middle-error group is doing acceptable. On the contrary, the lower-error group represents all current training data for which the neural net is already well-trained or is biased.

The enrichment process ($\mathcal{E}$) enriches the active training set (τ) by adding (at most doubling) training data in the high-error group (τ^h) and removing (at most half) training data in the lower-error group (τ^l). The middle-error group is left unmodified. The enrichment process for adding training data finds the nearest neighbor (defined by Euclidean distance [1]) for each training data in the high-error group (τ^h). The nearest neighbors are added to the current training data for the next enrichment training iteration.

On the other hand, the enrichment process for removing training data uses similar method of selecting m training data points from a cluster in the enrichment initialization. We need to select half number of training data in the low-error group ($n = \left\lfloor \frac{|\tau^l|}{2} \right\rfloor$). To select n training data points from the low-error group, we apply the same clustering algorithm ($\mathcal{C}$) to the low-error group with n sub-clusters. The training data point, which are closest (by Euclidean distance [1]) to the sub-cluster centers, will be kept for the next enrichment training iteration. The rest of the training data are removed from the current data for the next enrichment training iteration. Hence, we can represent the enrichment process of the training data as

$$\tau = \tau_{i+1} = \mathcal{E}\left(\tau_i, \{e_j\}, e_h, e_l\right) \tag{4}$$

where i is the enrichment iteration index and j is training data index in the current active training set τ_i. The newly modified, more balanced training data or the new active training set ($\tau = \tau_{i+1}$) will be used in neural net training of the next enrichment iteration.

The computational complexity of the enrichment update depends on two main factors: the number of data ($n = |\mathbb{T}|$) and the number of enrichment iterations (p). The three-group separation (τ^h, τ^m, τ^l) is a simple comparison against two preset threshold (e^h, e^l), which takes $O(n)$ complexity. Then a neighbor for each high-group data points (τ^h) is added to the active training set; the computation of finding an appropriate neighbor takes $O(n)$ for one data point. Thus, the total addition of the active training set requires $O(n \cdot |\tau^h|) = O(n^2)$ complexity. Next step is to half the low-group (τ^l) by applying the clustering technique ($\mathcal{C}$); this needs $O(|\tau^l| \cdot \ln |\tau^l| \cdot r) = O(n \cdot \ln n \cdot r)$ complexity. The enrichment update is repeated for a predefined number of enrichment iterations (p). Therefore, the enrichment update computational complexity is of the order

$$O(p \cdot n^2 \cdot r) \tag{5}$$

C. Enrichment Termination

The enrichment process iterates through the enrichment update step for a pre-defined number of enrichment training iterations. The enrichment update uses the active training set (τ) to train the neural net using the evolutionary training technique in Tepvorachai [4] which is based on a modified back propagation. Similar to a conventional back propagation in Haykin [6], the evolutionary training repeats the training process for a pre-defined number of iterations. Like the neural net training, the longer the enrichment process repeats, the better the trained neural net can estimate the active training set. However, if we have a priori knowledge of how well the training neural net performs after a number of iterations, we can select the number of iterations that trains the neural net to a desired level of accuracy.

IV. Experimentation

In this section, we are going to illustrate the results and the benefits of our enrichment process in neural net training. The scenario is the following. We face a great need to define a mapping between two input parameters and an output parameter. Such mapping is used in various applications, such as classification problems (i.e. image, fluorescence spectra, and gas dynamics) and control problems (i.e. robot arm, state space). The relationship between the two input parameters and the output is generally non-linear. Fitting and calculating an explicit mathematical 3-dimension function can be extremely tedious and computationally intensive. We choose, as an alternative, to use neural net as a candidate mapping function. In this example, we sampled data of a robotic arm controller in Figure 2 and plot them in 3 dimensions in Figure 3, where Angle and Gain are the two input parameters and State is the output parameter. Angle represents the current robot arm angle measured from a reference point (normalized from 0 to 1). Gain is an adaptive sub-controller unit input-output gain (normalized from 0 to 1). State refers to the expected system state of the robot arm.

Figure 3 shows the same sampled imbalanced control data of 100 sampling points. To calculate and fit all the sampling points with an explicit mathematical 3-dimensional function, we need to do trial-and-error on various modeling functions or require a certain high level of data modeling expertise. Alternatively, we can use the sampled points to train a neural net which also supports a powerful interpolation function between the sampled points [2], [6]. However, using all 100 sampled points in conventional training can prove to deteriorate the training process (slow convergence rate and imbalanced data effect) [10], [11], [14], [15]. We will be using our training data enrichment process on a single neural net to speed up the training process with imbalanced data while improving or maintaining the neural net accuracy in multi-label problems. Our results will be compared against a training process using all 100 sampled points on a single neural net and another training process with a randomly selected subset of the sampling points on a single neural net.

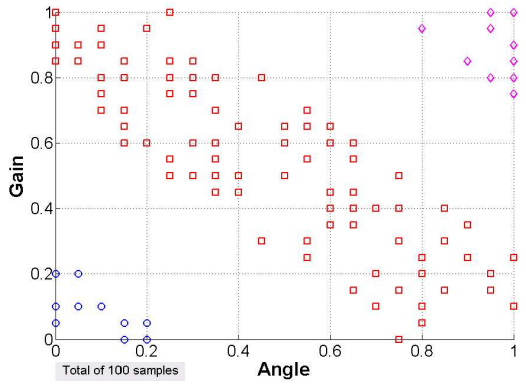

Figure 2
Sampled imbalanced control data (multi-label) [note: blue "◯" represents robot arm in state 0; red "□" represents state 1; and magenta "◇" represents state 2]

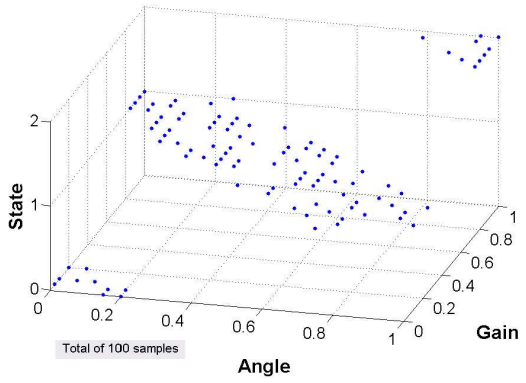

Figure 3
Same sampled imbalanced control data as in Figure 2 in 3-dimensional plot (multi-label)

A. Setup

The following section describes the setups of the enrichment process. Besides the difference in the training data, all trainings share the following parameters. The neural net trainings are repeated for 80 iterations. The neural nets are setup with 2 inputs, 1 output, and 4 hidden neurons in 1 hidden layer. Note that, for the enrichment process, we do data enrichment update every 20 iterations. This means the enrichment initialization is at iteration 0; the enrichment updates are at iteration 20, 40, and 60. The reason we choose to do 4 enrichments at 20 iterations is to generate a fair comparison among the trainings to the total sum of 80 iterations (arbitrary number). The 4 enrichment splits the 80 iterations into even interval of 20 iterations. From our prior

knowledge, the interval of 20 iterations is sufficient to train a neural net with the given architecture to a stable accuracy.

Figure 4 illustrates the imbalanced training data (sampled control) or the training set scope ($\mathbb{T}$) for neural net training where blue "•" marks all 100 training data (sampled control points), red "◯" around a blue "•" means the selected training data will be used for initial enrichment neural net training, and dotted splines indicate initial enrichment clusters. The enrichment initialization divides the 100 training data into 3 initial clusters, from the visual inspection of the training data scattering in space. As indicated by the dotted splines, each cluster consists of 10, 42, and 48 training data points, respectively, from left to right. The least number of training data in all 3 initial clusters (m) is 10. Thus, 10 data points are selected from each cluster contributing to the initial more balanced training data or the initial active training set (τ) (total of 30 points).

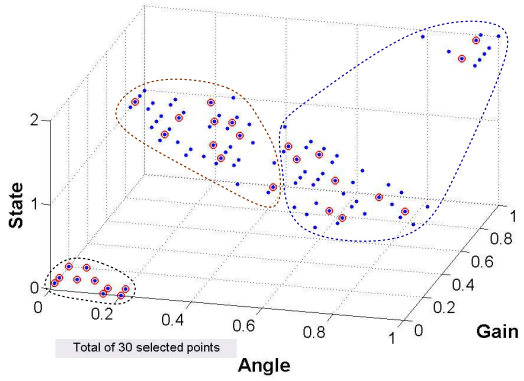

Figure 4
INITIAL ENRICHMENT ACTIVE TRAINING SET AT ENRICHMENT ITERATION 1 [NOTE: THE DOTTED SPLINES INDICATE 3 INITIAL CLUSTERS.]

B. Results and Measurements

In the following section, we describe the neural nets evaluation after their trainings. During the enrichment training, we observe the trained neural net performance. We measure neural net performance in two ways:

1) *Control effort accuracy (Ac)* defined as the percentage of accepted effort in the sampled points. This measures how acceptable the neural net controller is within a given tolerance level. Our robotic arm control process has the control state tolerance of 0.07 normalized units. If the estimated control state generated by the neural nets is within the tolerance range, it is considered acceptable. Otherwise, it will be rejected. The control effort accuracy can be mathematically defined as the percentage of acceptable control states ($accepted$) from all sample control states (all):

$$Ac = \frac{accepted}{all} * 100\% \tag{6}$$

2) *Control effort energy error (Ee)* defined as the Euclidean distances (energy error^2) between the estimated control states and the sampled target control states. This measures how closely the neural net is trained to the given target regardless of the tolerance. The smaller the energy error, the better the neural net performance is. The control effort energy error can be mathematically defined as:

$$Ee = \sqrt{\sum_j (t_j - o_j)^2} \tag{7}$$

where j is all sampled control points.

We measure the neural net's performance after enrichment iteration. The total number of enrichment iterations for this example is 4. Table I shows the two neural net performance measurements at the end of iterations. Figure 5 illustrates the last enrichment update training data where blue "•" marks all 100 training data (sampled control points), red "◯" around a blue "•" means the selected training data for enrichment iteration 4. We show Figure 5 as the enriched data selection comparable to the initial data selection in Figure 4.

Iterations	Accuracy	Energy Error
1	27.00%	2.0759
2	87.00%	0.4353
3	89.00%	0.3958
4	98.00%	0.3104

TABLE I
NEURAL NET PERFORMANCE OVER ENRICHMENT TRAINING PROCESS

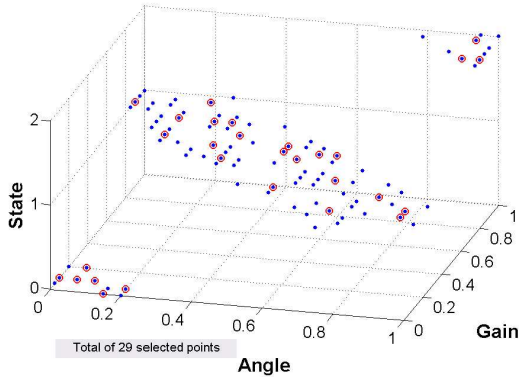

Figure 5
ENRICHMENT TRAINING DATA UPDATE AT ITERATION 4

During the enrichment training process, we observe the number of training data points being clustered into three error groups: high-error, middle-error, and low-error groups, as mentioned in Subsection III-B. The number of training data points in the three group assignment is illustrated in Figure 6. The first enrichment iteration (iteration 1) is the enrichment

initialization. The enrichment update occurs after enrichment iteration at 2, 3, and 4.

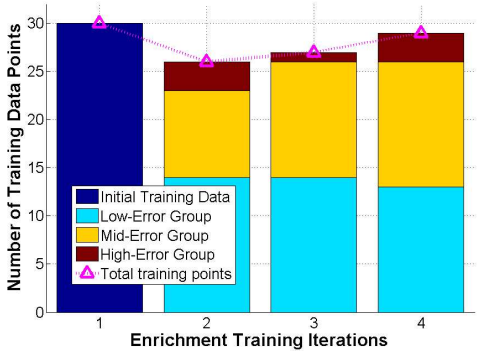

Figure 6
NUMBER OF TRAINING DATA POINTS IN HIGH-ERROR, MIDDLE-ERROR, AND LOW-ERROR GROUPS

In addition to the three error group observation, a few training data have been added and removed to/from the active training set due to the enrichment update. The number of training data points being added and removed to/from the active training set is shown in Figure 7. The first enrichment iteration (iteration 1) is the enrichment initialization; there is no enrichment update (no training data being added or removed).

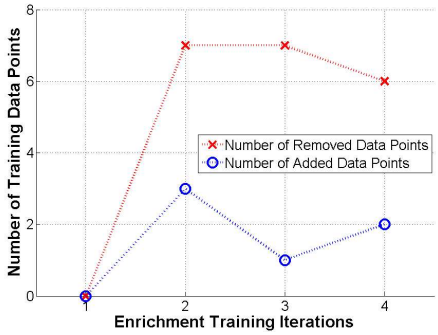

Figure 7
NUMBER OF TRAINING DATA POINTS BEING ADDED AND REMOVED

From the result in Figure 6 (magenta "△" marker) and Figure 7, we observe the number of training data points during the training process. Enrichment initialization produces training data with 30 points (1^{st} enrichment iteration); enrichment updates ($2^{nd} - 4^{th}$ iterations) have 26, 27, and 29 data points, respectively.

Moreover, we measure how the active training set (τ) from the enrichment initialization and the enrichment update distributed over the training data space. We measure the distribution by the training data variance. Recall that the variance (ρ^2) of 1-dimensional data is defined as the average of the Euclidean distance square from data mean in Equation (8) [22].

$$\rho^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{8}$$

We can calculate the variance of our 3-dimensional data by replacing the 1-dimensional Euclidean distance square by 3-dimensional Euclidean distance square. Hence, we obtain our training data distribution (variance) measurement in Equation (9).

$$\rho^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{3}(x_{i,j} - \bar{x}_j)^2 \tag{9}$$

The training data distribution and data mean (average)—as a vector of the low-error, middle-error, and high-error groups, respectively,—for the active training set (τ) in enrichment iteration is shown in Table II.

Iterations	Distribution	Mean
1	0.5311	(0.3700 , 0.4183 , 0.7303)
2	0.4734	(0.5019 , 0.4827 , 0.9591)
3	0.6185	(0.4056 , 0.4278 , 0.7748)
4	0.4958	(0.4500 , 0.4931 , 0.8940)

TABLE II
ACTIVE TRAINING SET DISTRIBUTION AND TRAINING DATA MEAN (AVERAGE)

Besides the neural net training with the enrichment process, we created two comparison neural net trainings. We refer to the neural net with enrichment training as "enriched" neural net. One of the comparison trainings, indicated as "all-data" neural net, refers to a neural net training with all sampled imbalanced control data from Figure 3. Another comparison training, indicated as "random" neural net, refers to a neural net training with some randomly selected training points from the sampled imbalanced control data in Figure 3. The "random" neural net uses 29 randomly selected training data which is about the same number of training data points that "enriched" neural net uses. The comparison of neural net performance at iteration 80 is shown in Table III. The Accuracy, Energy Error, and training data Distribution are defined in Equation (6), (7), and (9).

Neural net	Accuracy	Energy Error	Distribution
"enriched"	98.00%	0.3104	0.5297*
"all-data"	75.00%	1.2655	0.3903
"random"	72.00%	1.4792	0.4958

TABLE III
COMPARISON NEURAL NET PERFORMANCE AT ITERATION 80 [* IS THE AVERAGE DISTRIBUTION OVER THE ENRICHMENT PROCESS IN TABLE II]

C. Analysis and Discussion

From Figure 6, the "enriched" neural net training (with more balanced data about 26-30 points) uses much less training data than the "all-data" training (with 100 imbalanced training data points). The "enriched" and "random" (29 randomly selected training data points) neural net use about one-third of the available imbalanced data.

Regardless of the number of training data points for neural net training, the measurement in Table III shows the "enriched" neural net has higher accuracy (98.00%) than the "all-data" (75.00%) and "random" (72.00%) neural net classifier; at the same time, it has lower energy error (0.3104) than the others (1.2655 and 1.4792). Additionally, the training data distributions of "enriched" and "random" training (0.5297 and 0.4958) are almost indistinguishable; however, they are differentiated from the "all-data" distribution (0.3903).

Additionally, with regards to the result in Table I, the enrichment process has incrementally improved the neural net classifier performance. These results demonstrate that the enrichment process contributes significant improvement to the neural net classifier training process on multi-label imbalanced training data. The enrichment training loop can also be extended into higher information dimension neural net training (including input and output data).

V. Conclusion

We have used the enrichment process for neural net training to address the multi-label and imbalanced data problems. We proposed enrichment initialization, update, and termination processes. From the results, we can see that the enrichment process has better generalization performance than the traditional neural net training. Moreover, in order to achieve the best performance, the initial enrichment cluster size and update error thresholds should be selected according to the given imbalanced training data. In the future, we will incorporate this enrichment process with other neural net training techniques to improve the neural net performance and reduce the training time.

Notes

[1]Other clustering techniques and distance measurements are also possible as mentioned in Jain [20] and Berkhin [21].

[2]Energy error (Ee) is defined as a measurement of difference (or closeness or similarity) between the neural net estimated output (y) and the training data desired outputs (t) as described by

$$Ee = \frac{1}{2}\sum_{i=1}^{v}(t_i - y_i)^2$$

where t and y are vectors of same lengths.

References

[1] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, 1st ed. Cambridge University Press, 2000.

[2] R. Rojas, *Neural Networks - A Systematic Introduction*. Berlin, New-York: Springer-Verlag, 1996.

[3] G. Tepvorachai and C. Papachristou, "Facial image associative memory model," in *NASA/ESA Conference on Adaptive Hardware and Systems*, 2007.

[4] ——, "Self-configurable neural network processor for fir filter applications," in *NASA/ESA Conference on Adaptive Hardware and Systems*, 2006.

[5] ——, "Configurable fir filter scheme based on an adaptive multilayer network structure," in *NASA/ESA Conference on Adaptive Hardware and Systems*, 2007.

[6] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1998.

[7] M. R. Boutella, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, pp. 1757–1771, 2004.

[8] A. Dorado, D. Djordjevic, E. Izquierdo, and W. Pedrycz, "Supervised semantic scene classification based on low-level clustering and relevance feedback," in *European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology*, London, UK, 2004.

[9] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Conference on Information and Knowledge Management*, Bremen, Germany, 2005, pp. 195–200.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[11] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[12] Y. Liu, X. Dai, and Y. Shi, "Self-organizing fuzzy clustering neural networks controller for robotic manipulators," in *International Conference on Innovative Computing, Information and Control*, vol. 2, 2006, pp. 171–174.

[13] P. M. Patil, M. P. Deshmukh, and P. M. Mahajan, "A novel fuzzy clustering neural network," in *IEEE International Joint Conference on Neural Networks*, vol. 3, 2005, pp. 1989–1994.

[14] H. Salazar, R. Gallego, and R. Romero, "Artificial neural networks and clustering techniques applied in the reconfiguration of distribution systems," *IEEE Transactions on Power Delivery*, vol. 21, no. 3, pp. 1735–1742, 2006.

[15] M. A. Arslan, "The bp neural networks with data clustering enhancement - an emerging optimization tool," in *International Symposium on Intelligent Control*, Dearborn, MI, 1996.

[16] P. Cunningham, J. Carney, and S. Jacob, "Stability problems with artificial neural networks and the ensemble solution," *Artificial Intelligence in Medicine*, vol. 20, no. 3, pp. 217–225, 2000.

[17] M. A. El-Gamal and M. D. A. Mohamed, "Ensembles of neural networks for fault diagnosis in analog circuits," *Journal of Electronic Testing*, vol. 23, no. 4, pp. 323–339, 2007.

[18] P. Hartono and S. Hashimoto, "Adaptive neural network ensemble that learns from imperfect supervisor," in *International Conference on Neural Information Processing*, vol. 5, 2002, pp. 2561–2565.

[19] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory under-sampling for class-imbalance learning," in *International Conference on Data Mining*, 2006, pp. 965–969.

[20] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[21] P. Berkhin, "A survey of clustering data mining techniques," in *Computer Science*. Springer Berlin Heidelberg, 2006, pp. 25–71.

[22] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and Statistics for Engineers and Scientists*, 7th ed. Prentice Hall, 2002.