Multiple Instance Learning using EMDD

Tejas Khairnar
Arizona State University
Email: tkhairna@asu.edu

Vivin Paliath
Arizona State University
Email: tkhairna@asu.edu

***Abstract*—This midterm report details our understanding, approach, and progress in implementing the Expectation Maximization Diverse Density (EM-DD) [1] algorithm. EM-DD is an approach that has seen significant success in Multiple-Instances Learning problems and is a generalization of the supervised-learning classification-problem. Our goal is to demonstrate an understanding of this algorithm by providing a valid implementation and demonstrating its performance against the supplied data sets.**

I. Introduction

The Multiple Instances Learning (MIL) problem [2] is a variation of the general supervised-learning classification-problem. In MIL, instead of presenting a single instance, a bag of instances is presented to the algorithm, along with a label that describes the bag as either positive or negative. A bag is given a positive label if it contains at least one positive instance, and a negative label if it does not contain any positive instances. The goal of the MIL problem then, is to classify unseen bags or instances as either positive or negative, based on the training data. For this project, we specifically focus on the EM-DD algorithm and use it to solve the MIL problem.

II. EM-DD

Before looking at the EM-DD algorithm, it is useful to look at the definition of diverse density. If we assume that there is a hypothetical point t in the feature space that identifies the "true concept", the goal of DD is to identify the region that has positive instances from the most number of *different* positive bags, while also being far away from negative instances. Hence the diverse-density problem involves maximizing the probability $P(x = t \mid B_1^+, \ldots, B_n^+, B_1^-, \ldots, B_m^-)$, where B_i^+ is a positive bag and B_j^- is a negative bag. This is effectively equivalent to maximizing the following [3]:

$$\arg\max_x \prod_i P(x = t \mid B_i^+) \prod_i P(x = t \mid B_i^-) \qquad (1)$$

From this general definition of the maximum diverse-density, a noisy-or model is used to define the terms in the products. Hence the probability all points did not miss the target is $P(x = t \mid B_i^+) = P(x = t \mid B_{i1}^+, B_{i2}^+, \ldots) = 1 - \prod_j (1 - P(x = t \mid B_{ij}^+))$ and likewise $P(x = t \mid B_i^-) = \prod_j (1 - P(x = t \mid B_{ij}^-))$. The probability of a particular instance being the potential target-instance is then based on the distance between them: $P(x = t \mid B_{ij}) = \exp(-\|B_{ij} - x\|^2)$. The intuition is that if one of the instances in a positive bag is close to x, then $P(x = t \mid B_i^+)$ is high. Hence, if all positive bags have an instance close to x and no negative ones do, the diverse density of x will be high. Also observe that while the diverse density at the intersection of n bags is exponentially higher than at $n-1$ bags, the presence of a single negative instance is enough to drive down the diverse density.

The diverse density formulation also takes into account the relevance of features by learning a scaling vector s in addition to the location x that maximizes the diverse density ($\|B_{ij} - x\|^2 = \sum_k s_k^2 (B_{ijk} - x_k)^2$). This allows the algorithm to disregard irrelevant features and consider important ones, especially considering that it uses the Euclidean distance as a metric for "closeness".

In the standard diverse-density forumlation, the authors use gradient ascent with multiple starting-points to identify the target and scaling vector. But in the EM-DD formulation, the positive instance that corresponds to the bag label is viewed as a missing attribute that can be estimated using the expectation-maximization approach [1]. The algorithm starts with an initial guess of the target point t using instances from positive bags. It then repeatedly performs the *E*-step and the *M*-step to search for the maximum-likelihood hypothesis. In the *E*-step, the algorithm identifies the most-likely instance from each bag that is responsible for the label of the bag, based on the current target t. In the *M*-step, the algorithm uses a gradient-ascent search to find the new target t' that maximizes the diverse-density at h. After the maximization set, t is set to t' and the algorithm returns to the first step and runs until convergence.

III. Progress

We successfully implemented the EM-DD algorithm using the Python programming-language and are in the process of verifying its results against the provided data sets. To verify our implementation, we are comparing its performance against the EM-DD implementation in the MILL MatLab toolkit [4]. Our results largely agree with those from the MatLab implementation with the synthetic data-set, but further experimentation and fine-tuning is required to address some discrepancies. Once the discrepancies are addressed, we are planning to compare the performance of our implementation using public data-sets and also against the following algorithms in the MILL toolkit:

- Iterated-discrim APR
- Diverse Density
- Two SVM variants for MIL
- Citation-kNN for MIL

TABLE I
PROJECT TIMELINE

Deliverable	Due Date	Status
Understand EM-DD algorithm	11/10/16	Complete
Implement EM-DD algorithm in Python	11/18/16	Complete
Verify Python implementation	11/20/16	In progress
Analyze and compare results	11/24/16	Not started
Compare against other algorithms	11/26/16	Not started

Most issues that we ran into were due to unfamiliarity with SciPy, and were quickly resolved. One issue that we are currently dealing with is that our implementation doesn't seem to have as high as a precision as we would like on the synthetic data-set. We noticed that in the MatLab implementation, the `fpmincon` function was used, which doesn't have a very performant equivalent in SciPy; currently we are using an optimizer that performs gradient-ascent using the limited-memory variant of BFGS search with a numerically-estimated gradient. Documentation in the SciPy optimizer suggests that providing a function that calculates the gradient might provide better results than a numerically-estimated gradient. We are planning to implement this suggestion to see if it has a positive impact on our performance.

Another discrepancy we noticed was that by default, the EM-DD algorithm as implemented in the MILL toolkit, uses the average of the estimated target and scale vectors from each training-run during prediction. In contrast, our implementation uses the target and scale vector corresponding to the highest diverse-density. We feel that this could be a source of the discrepancy in results and plan to investigate this further.

A. Timeline

Our timeline for the project can be seen in table I. We're on track as far as finishing and verifying our implementation is concerned, and are looking forward to comparing its performance against the standard implementation, and other algorithms as well.

B. Workload

Workload was shared equally between both team-members. We started by reading the literature to gain an understanding of the material and then worked together on the implementation. While Vivin wrote the Python implementation, both team members were involved in designing it. Tejas worked on verifying that the implementation was accurate by comparing its results against the MatLab implementation. Both team members now plan on running experiments using more data, and also plan on comparing the performance of the implementation against other algorithms.

IV. CONCLUSION

We have achieved all the goals we have set ourselves thus far and are on track to achieve our remaining goals to successfully complete the project. We feel we have gained an appreciation and understanding of this particular statistical learning-method and look forward to performing additional experiments and comparisons using our implementation.

REFERENCES

[1] Q. Zhang and S. A. Goldman, "Em-dd: An improved multiple-instance learning technique," in *Advances in neural information processing systems*, 2001, pp. 1073–1080.

[2] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1, pp. 31–71, 1997.

[3] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," *Advances in neural information processing systems*, pp. 570–576, 1998.

[4] Jun Yang, "MILL: A Multiple Instance Learning Library," http://www.cs.cmu.edu/ juny/MILL/.

[5] SciPy, "SciPy Library," https://www.scipy.org/.

[6] Ragav Venkatesan, "Synthetic Dataset for MIL," https://github.com/ragavvenkatesan/np-mil/tree/master/data.

[7] J. Yang, "Review of Multi-Instance Learning and Its applications," https://www.cs.cmu.edu/ juny/MILL/review.htm.

[8] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," *Advances in neural information processing systems*, vol. 15, pp. 561–568, 2002.

[9] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml