Automation of Biological Research Course Project-2016fall

Yanyu Liang
yanyul@andrew.cmu.edu

Priyanka Raja
priyankj@andrew.cmu.edu

December 10, 2016

1 Methods Overview

In this project, the data is (x, y) and our task is to build a classifier to infer the label y given x under pool based active learning setting. We used uncertainty sampling as our active learning algorithm. To implement this approach, one requirement for the base learner is that it should assign probability to each label, namely $\Pr(Y = c_i|X = x, \text{model}), \forall i = 1, ..., k$.

1.1 Active learning strategy - Uncertainty sampling

Specifically, the uncertainty sampling approach we implemented is Best-versus-Second Best (BvSB) [1]. At each iteration, for each unlabeled sample in the pool, we first compute $\vec{p} = (p_1, ..., p_k)^T$, where $p_i = \Pr(Y = c_i|X = x, \text{model})$ using our current classifier. And then the BvSB score is computed as follow:

$$\text{BvSB} = p_{(1)} - p_{(2)}$$
$$,\text{where } p_{(1)} = p_j = \arg\max_j p_j$$
$$p_{(2)} = p_l = \arg\max_{l \neq j} p_l$$

Namely, BvSB is the difference between the probabilities of the most probable class and the second most probable class. Although entropy measure (EP) is widely used in uncertainty sampling, in multi-class problem, EP might not be suitable because in the case where $vecp$ has a heavy tail (even if the most probable class clearly stands out). In this sense, BvSB is more robust and we implemented BvSB as the uncertainty measure. And the active learner always queries the data point which has lowest BvSB score.

1.2 Base learners

The uncertainty sampling approach requires the base learner has probabilistic interpretation of label assignment. To achieve this, logistic regression is a natural choice. Other than this, with the use of softmax function at the output layer, neural network can be seen as a machinery to fit the function $\Pr(Y = y|X = x)$ which is also suitable in this set up. Hence, in these two classifiers, the learning objective is the negative log likelihood function.

Alternatively, we also applied SVM as base learner. SVM is widely used in binary classification and it is often known as a maximum margin classifier without probabilistic explanation. However, in one-vs-one approach for multi-class SVM implementation, the distance to each decision hyperplane can be transferred into a probabilistic interpretation [2]. So, we also used SVM as one of our base learners and we directly applied `libsvm` package [3].

Furthermore, to overcome overfitting, we regularized our logistic regression classifier and neural network classifier (one hidden layer with the number of nodes equals to 3 times the number of features) using both l_1 and l_2 penalties. We implemented the optimization algorithms of these two from sketch and in every iteration, we treated the previous model as a warm start and this made tremendously saved the computing power.

1.3 Implementation details

In the implementation, we used warm start, namely to randomly select 10 points from the pool to build the initial classifier. The reason why we use warm start instead of cold start is that SVM is built from a collection of binary classifiers and most of the sub-classifiers will be empty in cold start, which is potentially a technical issue. Furthermore, at the very beginning, the classifier is not very trustable for the sake of lack of data.

For the base learners, we set up kernels and hyperparameters as follow. The kernel of SVM was set as RBF kernel with default parameters. The hyperparameters of logistic regression is 10 for l_2 regularizer and 50 for group lasso regularizer. And the hyperparameters of neural network, namely group lasso, l_2 regularizer of W_1 and l_2 regularizer of W_2 are all set to 1, where W_i is the linear connection between ith layer and $(i+1)$th layer. The design of group lasso penalty is as follow:

$$\text{group lasso on } W = \sum_i \|W_i\|_2$$

, where W_i is the ith row of W which are the coefficients of ith feature. In another word, this term penalizes all of the coefficients of ith feature as a group and this can, on one hand, avoid overfitting and on the other hand, perform feature selection. Furthermore, the proximal gradient descent with backtracking was used for the training of both logistic regression and the neural network.

2 Results

In this section, we show the results of our three base learners along with uncertainty sampling one by one. Here, logistic regression is referred as sparseLogit and neural network is referred as sparseNN.

2.1 Base learner - SVM

The results of uncertainty sampling with SVM as base learner is shown in Figure 1. The y-axis is the rate of test error along learning and the x-axis is the number of queries (same for all the figures below if not specified).

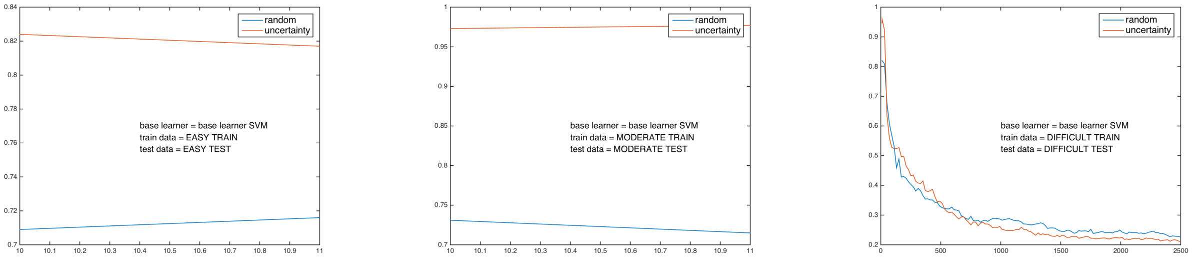

Figure 1: The results of SVM base learner (from left to right are the results of easy, moderate, and difficult datasets)

2.2 Base learner - sparseLogit

The results of uncertainty sampling with sparseLogit as base learner is shown in Figure 2. The test error is high in sparseLogit, a possible reason of it is that the model complexity of sparseLogit is low (linear classifier), which might not be powerful enough for our datasets.

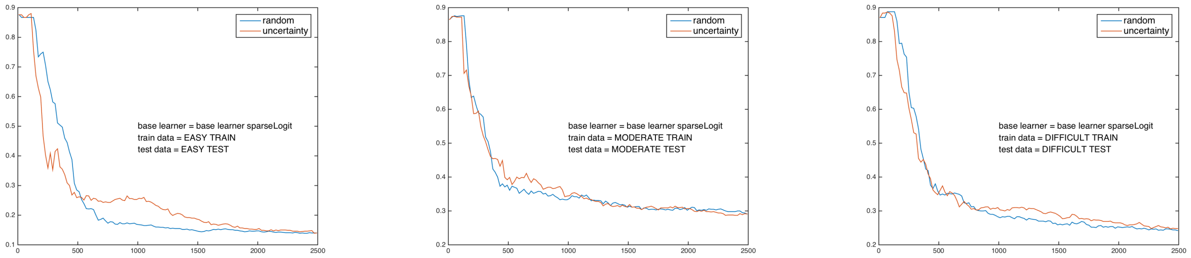

Figure 2: The results of sparseLogit base learner (from left to right are the results of easy, moderate, and difficult datasets)

2.3 Base learner - sparseNN

The results of uncertainty sampling with sparseNN as base learner is shown in Figure 3. As an ”advanced

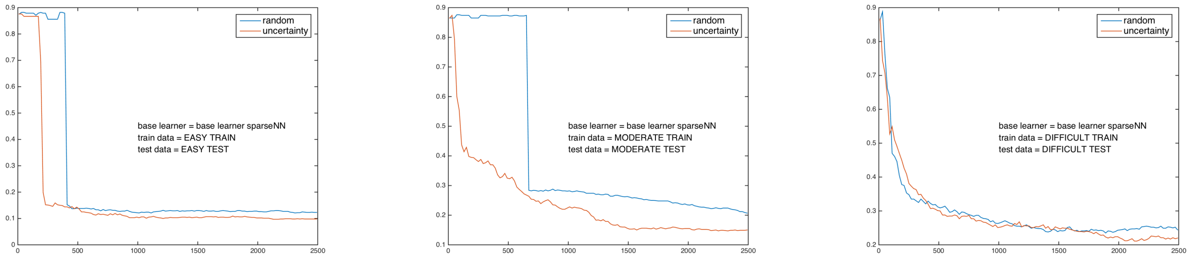

Figure 3: The results of sparseNN base learner (from left to right are the results of easy, moderate, and difficult datasets)

” or ”deep” version of sparseLogit, sparseNN

3 Conclusion

References

[1] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, “Multi-class active learning for image classification,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2372–2379, IEEE, 2009.

[2] T.-F. Wu, C.-J. Lin, and R. C. Weng, “Probability estimates for multi-class classification by pairwise coupling,” *Journal of Machine Learning Research*, vol. 5, no. Aug, pp. 975–1005, 2004.

[3] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.