Self-training classification model for semi labeled data

Hafez Asgharzadeh

(02/02/2018)

Abstract:

Semi-supervised method is a class of supervised learning techniques applied on a data with small amount of labeled and large amount of unlabeled data. Here, self-training method is applied to efficiently label unlabeled data. Effect of different parameters (probability threshold, unlabeled data ratio) are investigated versus the cost of human labeling. The model is developed in both sequential (with python) and distributed (with PySpark) systems. At the end, accuracy of the developed model is compared with label propagation model from Scikit-learn package.

Self-training model and method:

In self-training technique, a supervised model is trained based on the labeled data. The trained model is applied on the unlabeled data. Then, some of the labeled samples using trained model (in initial unlabeled data) are transferred to the training data based on the predict probability threshold. In other words, the model is improved gradually by adding samples of unlabeled data, which are labeled with high confidence, to the labeled data set. This algorithm iterates until the correct chance of unlabeled samples are below probability threshold, i.e., cannot be added to training data (labelled data) with high confidence.

Self-training model is wrapped on support vector model from Scikit-learn package and logistic regression model from PySpark (MLLIB) package, in sequential and distributed systems, respectively.

Dataset:

Iris dataset, which has150 observations, is used for evaluation of developed model similar to Scikit-learn package, i.e., iris dataset is used to evaluate label propagation model [1].

Result and discussion:

Figure 1 plots unlabeled data ratio (the ratio of unlabeled samples to the total number of samples) as a function of self-training iteration for probability threshold of 0.9 with initial unlabeled data

ratio equal to 0.88. As it is obvious from figure 1, self-training model can reduce unlabeled data ratio significantly from 0.88 to 0.15 with more than 90% confidence.

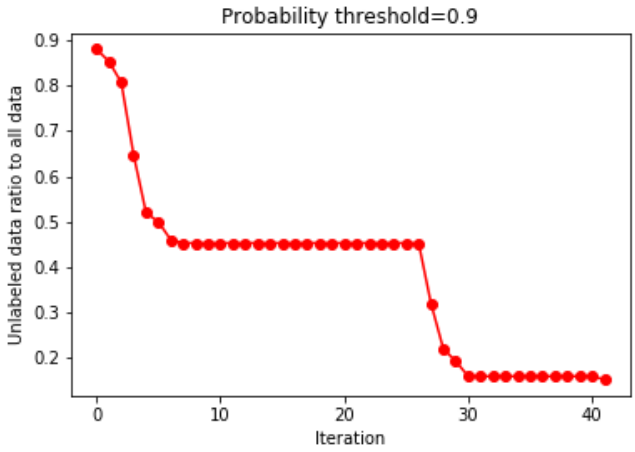

Figure 1. Unlabeled data ratio as a function of self-training iteration

Figure 2 shows human expert cost as a function of probability threshold. Note that labor cost for each label is assumed to be $5. From figure 2, labor cost changes slightly in the range of 0.6 to 0.9 probability threshold. However, labor cost notably increases after 0.9 probability threshold. As a consequence, 0.9 could be an optimal probability threshold since it benefits from both reasonable cost and high confidence for transferring samples from unlabeled to training data (labeled data).

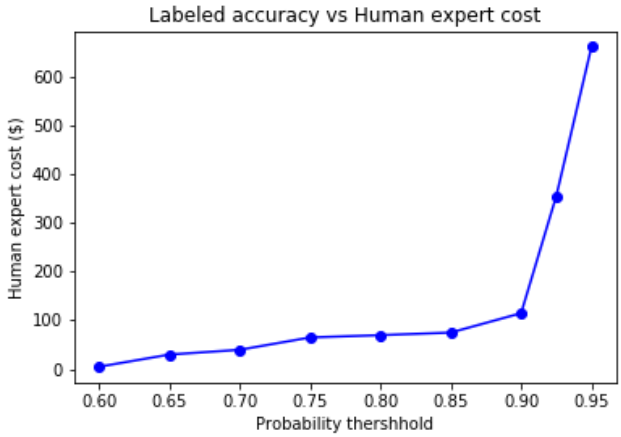

Figure 2. Human expert cost as a function of probability threshold

Figure 3 shows data ratio required to be labeled by a human expert, i.e., model can not label it by desirable confidence, as a function of initial unlabeled data ratio. The optimal probability threshold (=0.9 from figure 2) is used. Figure 3 shows capability of the developed model in labeling samples based on initilal unlabeled data ratio. As expected, the model requires enough labeled samples to start training. From figure 3, the model needs at least 10 % labeled data ratio, i.e., 90% unlabeled data ratio, to transfer most of the unlabeled samples with predict probability of higher than 0.9. As aconsequence, if the dataset has less than 10% labeled data, human expert is required to label samples in order to have 10% to start training. Note that all the results presented here are highly depend on the data.

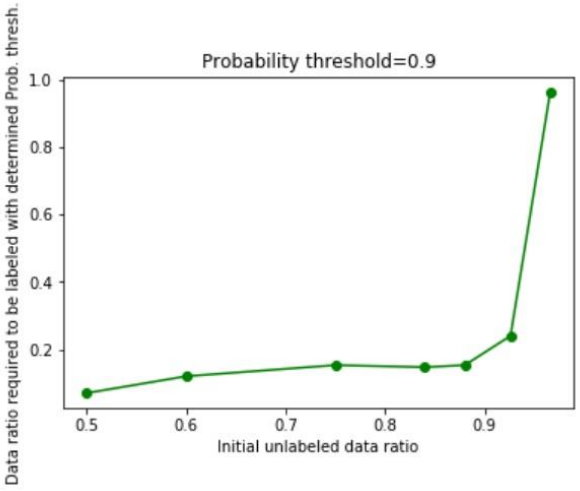

Figure 3. Data ratio required to be labeled as a function of Initial unlabeled data ratio.

Model comparison:

Although true labels of unlabeled samples are unknown in reality (it is considered throughout this documentation), it is a good idea to compare accuracy of the developed model with other existing models on a toy data (here, it is iris data) for the sake of validation [1]. Here, after labeling process is completed, the predicted labels are compared with true labels to compute accuracy, similar to [1]. My model (self-training) had a higher accuracy compared to label propagation model (graph-based) from Scikit-learn package, e.g., the accuracy of the developed model is approximately 95%, which is higher than label propagation model from Scikit-learn package (89%).

[1] *http://scikit-learn.org/stable/auto_examples/semi_supervised/plot_label_propagation_versus_svm_iris.html#sphx-glr-auto-examples-semi-supervised-plot-label-propagation-versus-svm-iris-py*