Semi Supervised Learning for Classification of Forest Coverage Type

Sara Misra
Pittsburgh, PA
saram1@andrew.cmu.edu

Junyan Pu
Pittsburgh, PA
junyanp@andrew.cmu.edu

Wenyu Huang
Pittsburgh, PA
wenyuh@andrew.cmu.edu

1 Introduction

Accurate natural resource information is vital to any private or state land management agency. Natural resource managers responsible for developing ecosystem management strategies require basic descriptive information including inventory data for forested lands to support their decision-making processes. One of the most basic and useful characteristics is the forest cover type. Our project aims to provide this information by predicting the forest cover type.

Considering forest cover types and classification, it is natural to visualize RGB images and a computer vision problem. However, to capture this type of data is extremely expensive and time-consuming. A cheaper type of data is the cartographical features of the area itself, for example, what elevation is the land area on, what the soil type is, etc. As such the question naturally arises that is it possible to classify a forest cover type based on these inexpensive features. An additional challenge to this question is the presence of a large number of unlabeled data in a large data set in the real world.

In this report, we aim to predict the forest cover type of an area given cartographical features of the area using a semi-supervised approach to deal with the lack of labeled data. We explore multiple methods of semi-supervised learning, particularly the Naive Bayes and Expectation Maximization algorithm, S^3VM, and co-training using neural networks.

1.1 Features and Data

The data used in this report was taken from the UCI Machine Learning Repository[1]. The data set consists of 581,012 samples of 30m x 30m patches of forest located in northern Colorado's Roosevelt National Forest. Each data sample includes 54 attributes: elevation (in meters), slope, aspect (compass direction of slope face), the vertical distance from water, the horizontal distance from water, sunlight intensity at 9 am, 12 pm, and 3pm, 4 binary wilderness area designators, and 40 binary soil type designators. Each sample was classified into one of seven forest cover types: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas Fir, or Krummholz. There are 83,000 samples classified as each of the seven cover types.

2 Background

We tested three methods for the baseline: Naïve Bayes and Expectation Maximization(EM), Label Spreading and Label Propagation. Using Naïve Bayes and EM, we achieved test accuracy around 50% when we vary the percentage of labeled data in the training set, as shown in Figure 1. For a typical semi-supervised dataset with 80% unlabeled and 20% labeled data, we achieved 55% train accuracy 45% test accuracy. Due to the unstable results and low accuracy of the graph-based methods (Label Spreading and Label Propagation) on this dataset, we discarded the graph-based method.

Submitted as Introduction to Machine Learning(Phd) (10-701, Spring 2019) Project Final Report. Do not distribute.

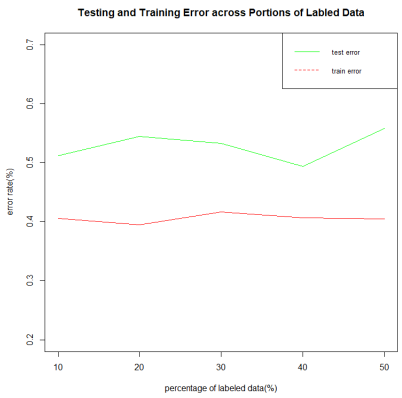

Figure 1: Test and Train error using EM on different percentage of labeled data in training dataset

Instead, we went with the higher accuracy stable result from the NB + EM algorithm as our baseline model to improve upon.

3 Related Work

3.1 Naive Bayes and Expectation-Maximization

Kamal Nigam, Andrew McCallum, and Tom Mitchell[2] discussed the approach of doing semi-supervised learning using generative models and Expectation-Maximization in text-classification. According to Nigam et al, a generative model is a good metric for semi-supervised learning when the assumptions of generative classifiers are met. For example, if the words of a document are conditionally independent of other words in the same document given the class label, then the Naive Bayes Assumptions are met. Therefore, if we can make certain assumptions about our dataset, we can relate this approach to our problem. In our problem, however, the conditionally independence assumption between features are not met. Thus, the performance of our baseline model using Naive Bayes and EM algorithm is limited due to the false assumption.

3.2 Graph Based Semi-Supervised Model

A Label propagation algorithm[4] creates a graph over all nodes in the dataset, and iteratively propagate labels through unlabeled nodes, by assuming that nodes close to each other have similar labels. Label propagation is effective because "some dataset are naturally represented by a graph, and this algorithm is scalable to large dataset". However, the disadvantage of Label propagation is that it produces no unique solution.

3.3 Semi-Supervised Support Vector Machine

K. Bennett and A. Demiriz[5] introduced a method using semi-supervised support vector machines. As they stated in their paper "Semi-Supervised Support Vector Machines", compared to SVM, S^3VM incorporates unlabeled data, and "requires the maximum margin to separate the labeled data and unlabeled data. It is also possible to add kernel functions in S^3VM algorithm[5]. However, since finding an exact solution to transductive SVM is NP-hard, we need to find appropriate optimization strategies, including the mainstream optimization algorithms including Combinatorial Optimization, Continuous Optimization, and Combinatorial Optimization[6]. In this report, we apply Quasi-Newton method (BFGS scheme) to perform a continuous optimization to find the optimal solution [9].

3.4 Co-training

Co-training algorithm proposed by Blum and Mitchel[7] first splits features into two views, and then uses two classifiers to train on these two views. Next, in each iteration, after two classifiers

predict unlabeled training data, for each predicted class, the algorithm will include predictions of the classifier who is more "confident" about its predictions as labeled data.

The intuition behind this algorithm is dependent on two assumptions. The first one is the conditional independence between 2 views. Therefore, on average each predicted labeled instances can be more informative and learning rate will increase. This method is first applied in text classification where 2 views are naturally chosen to be texts in web page and anchor text attached to hyperlinks pointing to this web page. The two classifiers in this case are Naive Bayes which have already achieved fair performance.

In text classification example, there are only 2 features and 2 classes. There have been some variations of co-training to adapt to multiple settings in real world. Du and Ling[8] explore co-training in empirical settings where the 2 assumptions are not entirely met. They tested co-training using different kinds of feature selection and various datasets. Their results show that most datasets did not have a better performance using co-training. One of the most possible reasons is that there may exist high redundancy in features. Although we can achieve a great performance by training on two views separately, the dependency between 2 views makes co-training less applicable to most datasets.

4 Methods

4.1 Bernoulli Naive Bayes and Expectation Maximization

Let the labeled data be L, unlabeled data be U and Y be the labels, where $K = 7$. There are n instances of labeled data $\mathbf{x}$ with corresponding labels and m instances of unlabelled data $\mathbf{z}$.

$$L = \{\mathbf{x}_1, \mathbf{x}_2..., \mathbf{x}_n\}, Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\}, U = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_m\}$$
$$\text{where } \mathbf{x}_i, \mathbf{z}_i \in \mathbf{R}_i^p, \mathbf{y}_i \in \{1, 2, ..., K\}$$

In this setting we use the Bernoulli Naive Bayes assumption to train the model on L, the labeled data set. The parameter is the seven forest covertypes class, which is $C_k, k = 1, 2, ...7$.

$$\hat{y} = \underset{k \in \{1,...,K\}}{\text{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

We use the Expectation Maximization algorithm to find the local optimal classifier. Here, in the E-step we use the model trained by Naive Bayes to predict the probabilistic labels $\mathbf{w}$ of the unlabeled data.
Then in the M-step retrain the model with the predicted labels and optimize the log likelihood of the function.

$$\text{E-step: for each } \mathbf{z}_i,\ \mathbf{w}_i = \texttt{max}_{\mathbf{w}_i} P(\mathbf{w}_i \mid \mathbf{z}_i)$$
$$\text{M-step: } \Theta^* = \texttt{argmax}_\Theta \sum_{i=1}^{m} logP(w_i, z_i \mid \Theta)$$

4.2 Semi Supervised Learning Support Vector Machine

4.2.1 Binary classification Semi-supervised SVMs

In binary classification setting, let $\mathbf{T}^\ell = \{(\mathbf{X}_1^l, \mathbf{Y}_1^l), ..., (\mathbf{X}_n^l, \mathbf{Y}_n^l)\}$ be the set of labeled instances. $\mathbf{X}_i^l = (\mathbf{X}_{i1}^l, ..., \mathbf{X}_{ip}^l)$ is a vector in p-dimension, and $\mathbf{Y}_i^l \in \{-1, 1\}$ is the corresponding label of $\mathbf{X}_i^l$. Let $\mathbf{T}^u = \{\mathbf{X}_1^u, ..., \mathbf{X}_m^u\}$ be the set of unlabeled instances. Let n denote the number of labeled instances, m denote the number of unlabeled instances, and $N = n + m$ be the total number of instances.
Semi-supervised SVM aims to find a prediction function that maximizes the separation on both labeled and unlabeled instances[3]. Let $\mathbf{f} \in \mathcal{F}$ be the decision function, and let $\mathbf{Y}^u = \{\mathbf{Y}_{n+1}^u, ..., \mathbf{Y}_N^u\} \in \{-1, 1\}^m$ be the predicted label vector on unlabeled instances $\mathbf{X}^u$. The S^3VM problem is equivalent to solving

$$\min_{\mathbf{f} \in \mathcal{F}, \mathbf{Y}^u} C^1 \sum_{i=1}^{n} \mathcal{L}^1(\mathbf{Y}_i^l, \mathbf{f}(\mathbf{X}_i^l)) + C^2 \sum_{j=n+1}^{N} \mathcal{L}^2(\mathbf{Y}_j^u, \mathbf{f}(\mathbf{X}_j^u)) + \lambda \mathcal{J}(\mathbf{f}) \tag{1}$$

where $C^1, C^2, \lambda \in \mathbb{R}^3$, $\mathcal{L}^1$ represents the loss function on labeled instances, $\mathcal{L}^2$ represents the loss function on unlabeled instances, and $\mathcal{J}(\mathbf{f})$ denotes the inverse of the margin. We choose hinge loss $\mathcal{L}^1(\mathbf{y}, \mathbf{f}(\mathbf{x})) = max(0, 1 - \mathbf{y}(\mathbf{f}(\mathbf{x})))$ to be the loss function for labeled instances, and effective loss(quote) $\mathcal{L}^2(\mathbf{f}(\mathbf{x})) = max(0, 1 - |\mathbf{f}(\mathbf{x})|)$ to be the loss function for unlabeled instances.

4.2.2 Multi-class classification Semi-supervised SVMs

In this project, we are solving a multi-class classification problem. We define $\mathbf{T}^\ell$ and $\mathbf{T}^u$, and the loss function the same way as in the binary case. In our problem, we have seven class labels, 10 continuous features, and 1 binary feature. Thus, we define $\mathbf{X}_i^l = (\mathbf{X}_{i1}^l, ..., \mathbf{X}_{ip}^l)$ to be a vector in p-dimension (with p = 10 if using only continuous features and p = 54 if using all features, which we will discuss later in Results section), and $\mathbf{Y}_i^l \in \{1, 2, 3, 4, 5, 6, 7\}$ to be the corresponding label of $\mathbf{X}_i^l$.
We use one-against-all approach in the multi-class classification task. In one-against-all approach, for each class j, a binary S^3VM classifier is trained to separate instances with label j from instances with other labels (i.e. $\mathbf{Y}_i = \begin{cases} 1 & \text{if} \quad \mathbf{Y}_i = j \\ -1 & \text{if} \quad \mathbf{Y}_i \neq j \end{cases}$). Specifically, we treat class 7 as the reference class, train six binary S3VM classifiers for class 1 - 6 separately, and get six decision functions f_1, ..., f_6. Then, given decision function f_1, ..., f_6, the overall decision function $\mathbf{f}$ is defined on $\mathbf{X}_i, i = 1, ..., N$ as follows:

$$\mathbf{f}(\mathbf{X}_i) = \begin{cases} 7 & \text{if } sign(\mathbf{f}_j(\mathbf{X}_i)) = -1 \quad \forall j = 1, 2, ..., 6 \\ \underset{k}{\operatorname{argmax}}\, \mathbf{f}_k(\mathbf{X}_i) & \text{otherwise} \end{cases}$$

In another word, if all six decision functions classify the instance $\mathbf{X}_i$ as -1, we classify $\mathbf{X}_i$ in class 7. Otherwise, we classify the instance into class j in which $\mathbf{X}_i$ has the largest margin.

4.2.3 Gradient Based Optimization

We use the Quasi-Newton Method (Broyden-Fletcher-Goldfarb-Shanno (BFGS scheme) to minimize the objective function (1.1) and find the optimal decision function $\mathbf{f}$. [9]
Since hinge loss and effective loss functions are not differentiable, we used differentiable surrogates loss functions in order to perform gradient based optimization. Specifically, we use $\mathcal{L}(y, t) = \frac{1}{\gamma} \log(1 + exp(\gamma(1 - yt)))$ with $\gamma = 20$ as the surrogate for hinge loss function, and $\mathcal{L}(t) = exp(-st^2)$ with $s = 3$ as the surrogate for effective loss function. It can be shown that the optimal decision function $\mathbf{f}(\mathbf{x})$ can be written as $\sum_{i=1}^N c_i K(\mathbf{x}_i, \mathbf{x})$, where K represents the kernel function, and $\mathbf{c} \in \mathbb{R}^N$. Thus, the optimization task is equivalent to finding coefficients $\mathbf{c} = \{c_1, ..., c_N\} \in \mathbb{R}^N$[9].

Plug in surrogates loss functions in the objective function (1.1) and we get:

$$\begin{aligned} \mathbb{F}(\mathbf{c}) = \mathcal{C}^1 \sum_{i=1}^{n} \frac{1}{\gamma} \log(1 - y_i \sum_{i=1}^{N} c_i K(\mathbf{x}_i, .))) \\ + \mathcal{C}^2 \sum_{i=n+1}^{N} \exp(-3(\sum_{i=1}^{N} c_i K(\mathbf{x}_i, .))^2) \\ + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_k K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

We optimize $\mathbb{F}(\mathbf{c})$ as follows:

Algorithm 1

procedure BFGS METHOD
  Initialize $\mathbf{c}_0$ by setting $\mathcal{C}^2 = 0$ and solve the supervised SVM problem.
  Initialize $\mathbb{B}_0$ the approximate Hessian matrix.
  $k = 0$.
  while termination criteria not fulfilled **do**
    obtain direction $\mathbf{p}_k$ by solving $\mathbb{B}_k \mathbf{p}_k = -\nabla \mathbf{F}(\mathbf{c}_k)$
    perform line search to obtain step size β_k
    Set $\mathbf{s}_k = \beta_k \mathbf{p}_k$ and Update $\mathbf{c}_{k+1} = \mathbf{c}_k + \mathbf{s}_k$
    Set $\mathbf{y}_k = \nabla \mathbf{F}(\mathbf{c}_{k+1}) - \nabla \mathbf{F}(\mathbf{c}_k)$
    Update $\mathbb{B}_{k+1} = \mathbb{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbb{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbb{B}_k^T}{\mathbf{s}_k^T \mathbb{B}_k \mathbf{s}_k}$

4.3 Deep co-training

We define $L = \{(\mathbf{x}_i, y_i) : 1 <= i <= N\}$ where $\mathbf{x}_i \in \mathbb{Z}^p$. L is the set of labeled data and p is the number of features which is 11. $y_i \in \mathbb{Z}^k$ where k is the number of classes which is 7. $U = \{(\mathbf{z}_i) : 1 <= i <= N\}$ and $\mathbf{z} \in \mathbb{Z}^p$. U is the set of unlabeled data.

4.3.1 Feature Selection

One important part of co-training is to select two views. In theory, two views should be conditionally independent and either of them as training data to either classifier should have a relatively good performance.

We tried two different approaches of feature selection. The first one is based on our intuition after the EDA analysis on different features. We try to divide 11 features into 2 sets where features in two sets are conditionally independent from the EDA analysis. The features in two views are {Elevation, Slope, Water Dist, Water Dist Vert, Soil type 1-20} and {Road dist, Hillshade 9AM, Hillshade Noon, Hillshade 3PM, Fire Point Dist} respectively.

The second one is using Information Gain. We first calculate the information gain of all features. We then sort them and take odd index feature as 1st view and features on the even index as 2nd view.

4.3.2 Algorithm

Algorithm 2 Co-training with Multi-Class

procedure FIT
  $U \leftarrow$ unlabeled data
  $L \leftarrow$ labeled data
  $k \leftarrow$ number of classes
  $U' \leftarrow$ randomly select u data points from U
  $i = 0$
  while i < T and $\mid U \mid > 7 * u$ **do**
    $i \leftarrow i + 1$
    fit U' using Classifier1
    fit U' using Classifier2
    for $1 <= j <= k$: **do**
      $P_{1j} =$ top n most confident data points in U' with predicted label k predicted by classifer1
      $P_1 = P_1 \cup P_{1j}$
      $P_{2j} =$ top n most confident data points in U' with predicted label k predicted by classifer2
      $P_2 = P_2 \cup P_{2j}$
    if $\mid p \in P_1 : p > \theta \mid > \mid p \in P_2 : p > \theta \mid$ **then**
      $U \leftarrow P_1 \cup U$
    else
      $U \leftarrow P_2 \cup U$
    update U' by redrawing samples randomly from U so that $\mid U' \mid = u$

Above is pseudocode for co-training algorithm. There are 2 classifiers and 2 views. For each iteration, each classifier will be trained using labeled data from 1 view. By comparing the confidence score of predictions on unlabled data, labeled data will be updated by adding the predictions of more confident predictions.

In our case, the two classifiers are both neural networks with 3 layers, first of which is a feedforward layer with 100 hidden ReLU units. The second layer is a feedforward layer with 50 hidden ReLU units. The last layer has 7 output softmax units that correspond with 7 classes. All of these hidden units are Rectified Linear Units(ReLU) and we use softmax as our activation function for output state. The loss function is categorical cross entropy.

We also include "early-stopping" in our neural network. This means that the neural network will stop training once validation error starts to decrease.

5 Results

Overall, given 20% labeled data and 80% unlabeled data, our models achieve approximately 70% test accuracy. On the other hand, because all instances in the dataset are labeled, prior work with respect to this dataset are all supervised learning. Their test accuracy is slightly above 80%.

5.1 Semi-Supervised Support Vector Machine

Fix percentage of labeled data = 20%, we use both linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ and radial basis function (rbf) kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ and obtain their train, validation, and test accuracy.

5.1.1 Continuous and discrete predictors

We have a total of 10 continuous features and 1 discrete feature. Since the discrete feature contains 44 binary categories, we can elaborate it as 44 binary features. Thus, in this case, we fit S^3VM algorithm on all the predictors, every instance $\mathbf{X}_i$ is a 54-dimensional vector.
Using linear kernel, we achieve 59.52% train accuracy, 60% validation accuracy, and 59.31% test accuracy.
Using rbf kernel, we use cross validation to choose the value of σ and achieve 68.69% train accuracy, 69.55% validation accuracy, and 69.49% test accuracy.

5.1.2 Continuous features only

Since large number of binary features would lead to inaccurate measure of kernel function, we remove the discrete feature, and run S^3VM on 10 continuous features only.
Using linear kernel, we achieve 58.02% train accuracy, 59.34% validation accuracy, and 58.01% test accuracy.
Using rbf kernel, we use cross validation to choose the value of σ and achieve 67.22% train accuracy, 65.76% validation accuracy, and 67.07% test accuracy.
In Figure 2 we can see the cross validation error curve versus different value of σ in rbf kernel.

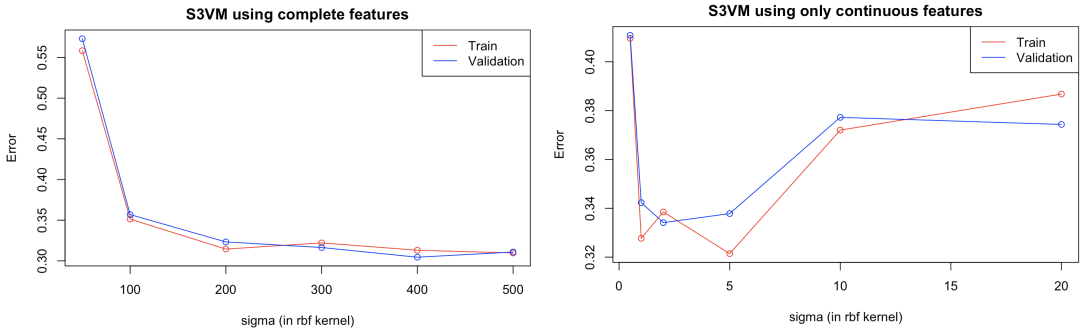

Figure 2: Cross Validation Error against σ parameter in RBF Kernel

5.2 Deep Co-training

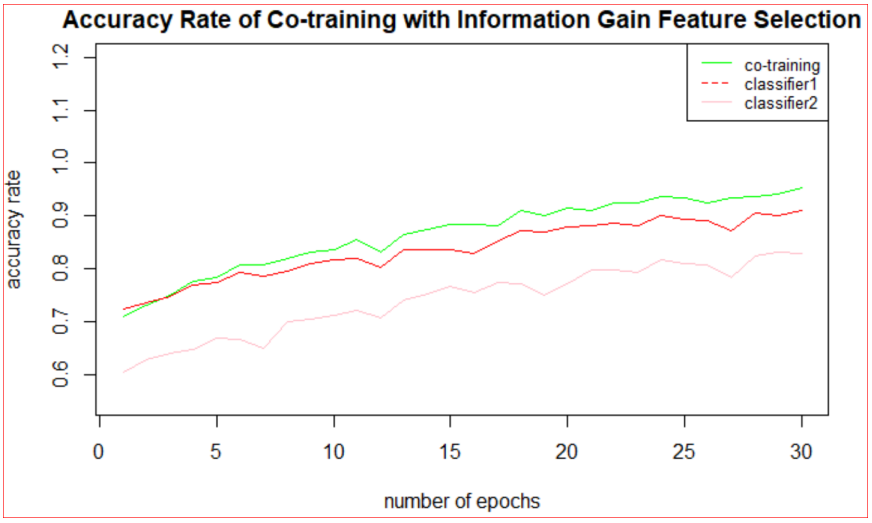

Figure 3: Co-training and implemented classifier's accuracy using information gain feature selection

From Figure 3, the training accuracy of both classifiers improve and therefore the combination of two classifiers has better performance as well. This graph shows that classifying with 2 views can improve the performance. However, it is also notable that as number of epochs increase, we observe signs of overfitting. This is expected because as number of epochs increase, some labeled training data has been used for training for multiple times.

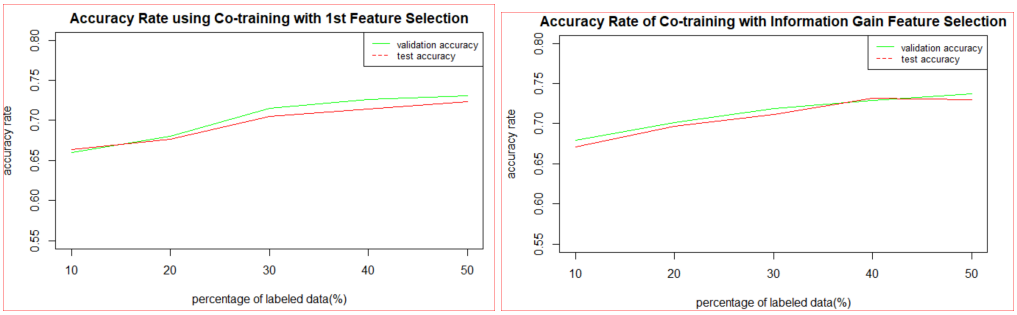

Figure 4: Co-training accuracy against percentage of labelled data using multiple feature selection methods

For both feature selection strategy, we plot the performance of co-training with respect to percentage of labeled data. We find that in general, accuracy rate of co-training using information gain is greater than feature selection based on EDA. The reason might be that we can evenly distribute high-entropy features evenly in the two views; hence, classification using two views separately are more sufficient to use in co-training. (sufficiency assumption)

6 Discussion and Analysis

Previous research work[10] on this dataset has been geared towards using supervised learning methods that have resulted in a prediction accuracy of around 80%. However, in general cases the challenge of classifying forest covertypes is the expensive cost of labelling such a large dataset which is not always worthwhile. Thus, this makes it an excellent choice for employing semi-supervised learning methods to solve this challenge and from our results we achieved around 70% test accuracy, incurring less cost for a high accuracy.

6.1 Deep Co-training

In general, we see a better performance of co-training classifier than that of either single classifier. The testing accuracy rate is about 70% using 20% of labeled data and 80% of unlabled data. This suggests the power of applying deep co-training on this dataset. However, despite the fact that it achieves a relatively better accuracy rate than the baseline, it has several limitations. We can see from Figure, there is a sign of overfitting when number of epochs increase. We summarize several possible sources of limitations below.

6.1.1 Independence and Sufficiency Assumption

As discussed in related work section, there are 2 fundamental assumptions in co-training. The first one is that features in 2 views should be conditionally independent (independence assumption). Second, when 2 classifiers train on 2 views separately, both of them should have a relatively good performance. (sufficiency assumption) The second one, sufficiency assumption, is fulfilled. Because we test neural network on two views separately, we achieve about 65% test accuracy which is much better than our baseline model accuracy about 50%. So we are confident enough that after co-training, co-training model will outperform our baseline EM model. However, with respect to the independence assumption, even though we do an EDA analysis on features, we mainly focus on one vs one comparisons. We are not able to guarantee the conditional independence between two Sets of features. Similarly, although information gain provides us with good perspective, information gain does not provide insights into conditional independence between features.

6.1.2 Relatively Small Input Space

Apart from strong assumptions in co-training, another drawback of our approach in this case is that the number of feature is relatively small. Neural network has good performance when given a large number of features so that it can find potential combination of features helpful in final prediction. However, in our dataset, we only have 11 features which seems to be one of the bottlenecks of performance.

6.2 Semi-Supervised Support Vector Machine

According to the results, we can tell that rfb kernel performs better than linear kernel in both cases (complete features and continuous features only). We can also see that including the discrete features leads to 1% improvement in accuracy, which is not significant. Thus, it is preferred that we remove the discrete feature (44 binary predictors) in order to trade for higher computational efficiency.

6.2.1 One-against-other vs one-against-one

In this project, we use One-against-other method to implement the multi-class S^3VM classifier. Another possible method for multi-class SVM is one-against-one approach, where for every combination of two different classes, we fit one classifier to separate the two classes. Since we have 7 classes, using one-against-one method would be computationally expensive and we therefore did not choose this method. However, one-against-one method might lead to higher accuracy, since one-against-one comparison between two classes is more specific than one-against-other comparison.

6.2.2 Constraint on label percentage

In our dataset, we have a total of seven classes uniformly distributed. Thus, when fitting one-against-other SVM for a particular class, the +1/-1 ratio for the labeled data is 1:6, which is unbalanced. One potential improvement is to add constraint on the percentage of positive labels fitted by the SVM classifier, which might lead to better result.

References

[1] UCI Forest Covertype Dataset: https://archive.ics.uci.edu/ml/datasets/covertype

[2] Kamal Nigam , Andrew McCallum , Sebastian Thrun , Tom Mitchell, Learning to classify text from labeled and unlabeled documents, Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, p.792-799, July 1998, Madison, Wisconsin, USA

[3]L. Wang, X. Shen, and W. Pan. On transductive support vector machines. In J. Verducci, X. Shen, and J. Lafferty, editors, Prediction and Discovery. American Mathematical Society, 2007.

[4]Zhu, X., Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation (p. 1). Technical Report CMU-CALD-02-107, Carnegie Mellon University.

[5] Bennett, K. P., Demiriz, A. (1999). Semi-supervised support vector machines. In Advances in Neural Information processing systems (pp. 368-374).

[6]Chapelle, O., Sindhwani, V., Keerthi, S. S. (2008). Optimization techniques for semi-supervised support vector machines. Journal of Machine Learning Research, 9(Feb), 203-233.

[7]A. Blum, T. Mitchell, "Combining labeled and unlabeled data with Co-training", Proc. 11th Annu. Conf. Comput. Learn. Theory, pp. 92-100, Jul. 1998

[8]J. Du, C. X. Ling and Z. Zhou, "When Does Cotraining Work in Real Data?," in IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 5, pp. 788-799, May 2011.

[9]Gieseke, F.; Airola, A.; Pahikkala, T.; Kramer, O. Fast and Simple Gradient-Based Optimization for Semi-Supervised Support Vector Machines. Neurocomputing 2014, 123, 23–32.

[10]Blackard, Jock A. and Denis J. Dean. 2000. "Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables." Computers and Electronics in Agriculture 24(3):131-151.