IN4320 Machine Learning Assignment 2

Dhruv Batheja
4716523

13/03/2018

Semi-Supervised Learning
Data preparation/preprocessing: For preparing the dataset, I used pandas to first read the dataset into a *data-frame*. Now, to make sure, consistency was maintained and the model we build was robust, I shuffled the dataset. Next, I normalized the feature values *(X vector)* to a mean of 0 and variance 1 by using this transformation: $(x - \mu_x)/\sigma_x$. I also used 0 and 1 as the labels for the classes *(y vector)* as these are the standard labels for 2-class classification problems. I used the last 2000 examples from the randomly shuffled dataset as the test-set for all of my following experiments. The training-set comprises of two components: 1) *Labeled* (fixed 25 samples) and 2) *Unlabeled* (Experimented with 0, 10, 20, 40, 80, 160, 320, 640, 1280 and 2560 samples).

a

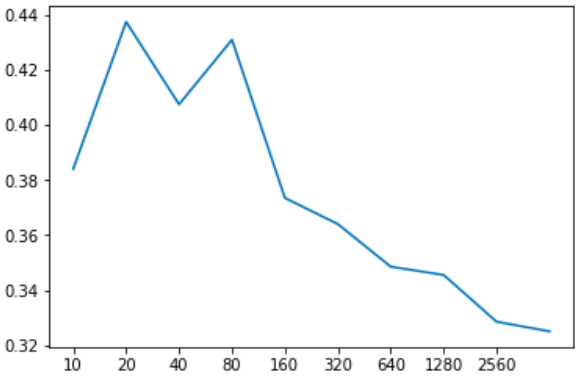
Self Training Algorithm: This method makes the assumption that one's own high confidence predictions are correct. Check reference link [1] for a good explanation of this approach. The classifier I used for this was the LDA classifier (Linear Discriminant Classifier) as suggested in the question. It is a pretty safe assumption to make, but is safe only when we have sufficient amount of training data. Before trying this approach, I tried scoring the classifier with no unlabeled data and just 25 training samples. I got a score of $\tilde{6}5\%$ i.e. an error of 35%. Here is the self-training algorithm:

- Train classifier from $(X_{labelled}, Y_{labelled})$
- Get predictions on $X_{unlabelled}$. (Let predictions $= f(x)$)
- Add the rows $(x, f(x))$ to the labeled dataset.
- Repeat

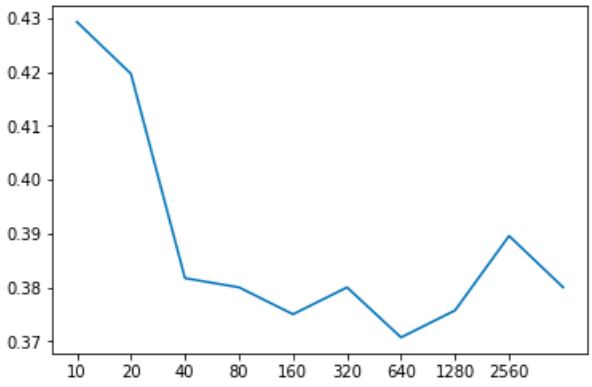
I tried two variations of this method: 1) Adding a few most confident $(x, f(x))$ to labeled data. Here I only added the rows that had a prediction confidence of 0.7 or more. 2) Adding all the predictions to the labeled data. I seemed to get better results with the second variant i.e. Adding all the predictions to the labeled data and I used that for my comparisons in the upcoming sections. It is probably the simplest semi-supervised learning method but many real world classification tasks show that simplicity is power. This method is often used in tasks like natural language processing.

Expectation Maximization for Gaussian Mixture Model: We need to estimate μ (mean) and $\sum$ (covariance) of the Gaussian distributions based on the assumption that our dataset is comprised of a mixture of Gaussians[2]. Since the LDA performs fairly well on this dataset, this is not a very unsafe assumption to make. The parameters μ and $\sum$ can be calculated using Expectation Maximization (EM) technique. EM algorithm is an iterative optimization technique which is operated locally. It has 2 steps: 1)*Estimation step*: for given parameter values we can compute the expected values of the latent variable. 2) *Maximization step*: updates the parameters of our model based on the latent variable calculated using ML method. This method can be summarized into the following steps[3]:

- Initialize the means μ_j, covariances $\sum_j$ and mixing coefficients π_j using $X_{labeled}$ and $Y_{labeled}$



(a) Self Training Algorithm



(b) EM for Gaussian Mixture Model

Figure 1: Test set error vs number of unlabeled samples

- *E step.* Evaluate the responsibilities using the current parameter values
  $\gamma_j(x) = \frac{\pi_k G(x|\mu_k, \sum_k}{\sum_{j=1}^{K} \pi_j G(x|\mu_j, \sum_j)}$
  where: $G(x|\mu, \sum)$ is a Multi-variate Gaussian.
- *M step:* Re-estimate the parameters using the current responsibilities.
  $\mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) x_n}{\sum_{n=1}^{N} \gamma_j(x_n)}$
  $\sum_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)}$
  $\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n)$

- Evaluate log likelihood like this:

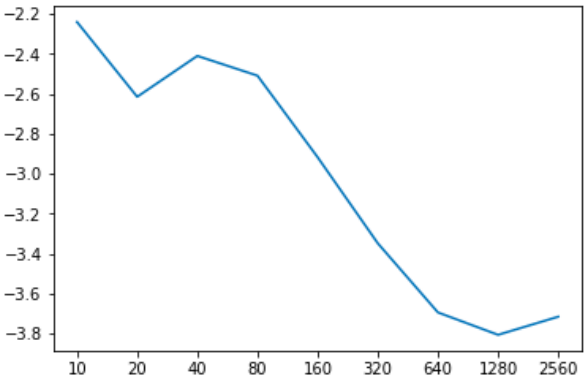
  $$lnp(X|\mu, \sum, \pi) = \sum_{n-1}^{N} ln\left\{ \sum_{k=1}^{K} \pi_k G(x_n|\mu_k, \sum_k) \right\}$$

  If there is no convergence, return to the second step.

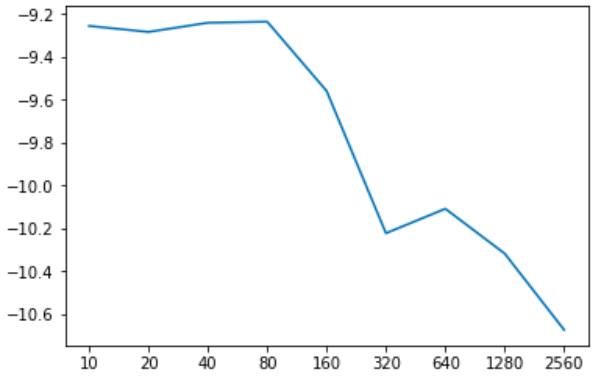
This algorithm iteratively improves the prediction and converges to a local maximum of the parameters. Some advantages of this approach include: It is a clear and well studied framework. And it can be extremely effective if the model is close to perfect.

b

I normalized the X vector using this transformation: $(x - \mu_x)/\sigma_x$. This ensures that the values for each feature have a mean of 0 and variance 1. I performed both experiments over 100 iterations each of the algorithms discussed in the section above. Figure 1 plots the mean of the 100 iterations. The general trend can be observed for both the algorithms i.e. the error rate seems to decrease for sometime and become somewhat constant after that. The error rate drops by about 10% for both the implemented approaches which fulfills the objective of Semi-Supervised learning. Making use of unlabeled data to get better predictions than using just labeled data.
Note in 2a the error initially goes up for less unlabeled samples and gradually goes down. This might be due to several reasons like: the initial samples that are added might have low confidence predictions (since we are adding all predictions to the training set as discussed in section **a**.) For the learning curve 2b, the behavior is predictable as the error just decreases and kind of stabilizes. The simpler Self training algorithm performs better (in terms of classification errors) than the Gaussian Mixture Model for high volumes of unlabeled data as can be seen from the plots.



(a) Self Training Algorithm



(b) EM for Gaussian Mixture Model

Figure 2: Log likelihood curves for sum of probabilities for both classes

c

There are several ways to plot the log likelihoods. Log likelihoods of the confidence values can give us a better feeling about how confident our model was while predicting a certain label for the vector X. The way I did it is not the conventional way to do it. Everyone is familiar with the log likelihood curve that goes up and converges. That will be the case if you just plot the logs of the higher probability class. I plotted the sum of the logs of the probabilities. If you plot the sum of logs of probabilities of a vector with increasing probabilities(for one class and decreasing for the other) example: p= [0.1, 0.2, 0.3 ... 0.9] and q = [0.9, 0.8, 0.7 ... 0.1], it will look like 3. This means that the sum will be maximum when the confidence of both classes is 0.5 and 0.5 respectively and as the confidence for any of the classes increases, the value of our function decreases. For instance $(log(0.5)+log(0.5)) > (log(0.7)+log(0.3)) > (log(0.9)+log(0.1))$. Finally our function can be formulated as:
$f(p) = log(p) + log(1-p)$ where p is the probability of predicting a class for a vector.

The figure 2 shows the function plots for both the algorithms and both show a curve that is going down that indicates that the confidence of prediction increases as the number of unlabeled samples increase. This is a good sign for our models. One anomaly I observed here is that the values for EM are lower than than for self training which means EM has better confidence in predictions even though self training ends up classifying more vectors correctly. This could be because of the fact that the model generally gives higher probabilities. This needs more introspection.

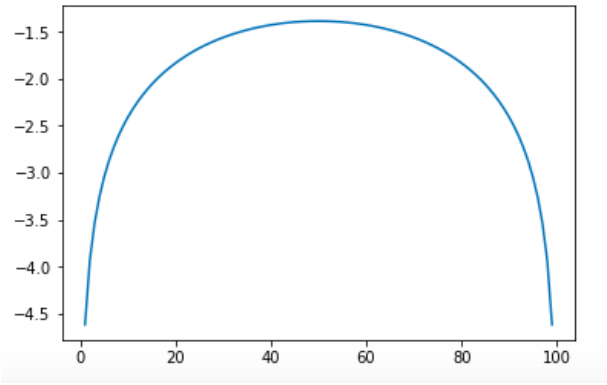


Figure 3: Plot of sum of probabilities

d

Construction of an artificial dataset such that the Self-training works well while Gaussian Mixture Model doesn't: Since we're using LDA as our classifier, we'll need the discrimination information to be in the variance of the distributions. LDA reduces the dimension of the dataset in the direction that maximizes the split in the dataset and finally splits them in lower dimensions. For the Gaussian mixture model to fail, the distributions has to be Non-Gaussian or overlapping. The standard implementation for producing well performing dataset for the self training algorithm assumes a Gaussian distribution of the input variables. Consider reviewing the univariate distributions of each attribute and using transforms to make them more Gaussian-looking. Conclusively, a Gaussian dataset which has high variance and is linearly separable in lower dimensions is what we need for this scenario as unlabeled data will hurt if generative model is wrong.
Construction of an artificial dataset such that the EM Gaussian Mixture Model works well while Self-training using LDA classifier doesn't: Some of the disadvantages of self-training can include: Early mistakes could reinforce themselves. Especially because we are adding all the predictions to the training set. Another disadvantage can be that a lot cannot be said about convergence of this algorithm like for EM GMM. LDA expects the discrimination information in the variance and doesn't perform that well when the discrimination information is in the means of the distributions. We could definitely exploit this weakness. We'll also have to make sure that the distributions are Gaussians as GMM expects that to perform well. Hence, we can safely conclude that the dataset that we need for this scenario needs to have discrimination information in the means of the distributions and the vector X has to be a mixture of Gaussians.

References

[1] Xiaojin Zhu "Semi-Supervised Learning Tutorial." *http://pages.cs.wisc.edu/˜jerryzhu/pub/sslicml07.pdf*

[2] Hastie & Tibshirani "Gaussian mixture models" *http://statweb.stanford.edu/ tibs/stat315a/LECTURES/em.pdf*

[3] Gaussian Mixture Model (GMM) using Expectation *http://www.cse.iitm.ac.in/ vplab/courses/DVP/PDF/gmm.pdf*