Co-training and Learning with Noise

Wee Sun Lee
Department of Computer Science and Singapore-MIT Alliance, National University of Singapore, Singapore 117543, Republic of Singapore

Abstract

Blum and Mitchell introduced a model of learning in (Blum & Mitchell, 1998) where each instance $X$ is composed of two views $(X_1, X_2)$ with $X_1$ conditionally independent of $X_2$ given the label of $X$. They showed that the target concept is learnable using $X_2$ and the labels of a weak predictor $f$ on $X_1$ because the labels generated by $f$ behave like the actual label corrupted by independent classification noise. Most works on learning with two views have not exploited this result but concentrated instead on maximizing the agreement of two functions trained on the two views $X_1$ and $X_2$. In this paper, we show that it is possible to develop an effective learning algorithm for learning with two views by learning a linear function with noisy labeled data. This approach has the advantage of requiring a correct target concept to exist in only one but not both of the views. We then explore the consequence of the loss of conditional independence of the views when the positive and negative examples are drawn from a mixture distribution, where the views are conditionally independent given the component of the mixture but not conditionally independent given the label of the example.

1. Introduction

Co-training (Blum & Mitchell, 1998) is a method for exploiting redundancy in natural occuring data to perform learning using only unlabeled data and a weak predictor. In co-training, we assume that each instance $X$ is composed of two views $(X_1, X_2)$ where $X_1$ is conditionally independent of $X_2$ given the label of $X$. Under this condition, a weak predictor $f$ on $X_1$ (in the sense of $\Pr[f(X_1) = 1|Y = 1] \geq \Pr[f(X_1) = 1] + \epsilon$, where $Y$ is the actual label) can be used to generate labels for $X$. The labels generated by the weak predictor behave like the actual label corrupted by independent classification noise where a positive label is randomly flipped with probability $\alpha$ and a negative label is randomly flipped with probability $\beta$, and $\alpha + \beta < 1$ (Blum & Mitchell, 1998).

One example where the co-training model seems reasonable is in learning the class of a web page where the text of the page forms one view and the anchor text of links pointing to the same web page forms another view (Blum & Mitchell, 1998). Another example is in learning the named-entity class of a word where the spelling features of the word form one view and the context surrounding the text forms another (Collins & Singer, 1999).

Blum and Mitchell (Blum & Mitchell, 1998) showed that given $\alpha + \beta < 1$, function $h$, noisy observed label $Y'$, actual label $Y$ and input $X$, $\Pr[h(X) = 1|Y' = -1] + \Pr[h(X) = -1|Y' = 1]$ is linearly related to $\Pr[h(X) = 1|Y = -1] + \Pr[h(X) = -1|Y = 1]$. This is useful as it means that minimizing the observed sum of false positive and false negative frequencies will also minimize the actual sum of false positive and false negative frequencies when the sample size is large enough. Following (Blum & Mitchell, 1998), various authors have worked on the dual-view learning problem (Collins & Singer, 1999; Dasgupta, 2001; Abney, 2002) but most of them have concentrated on minimizing the disagreements between the two functions that are being learned from the two views. One disadvantage of minimizing disagreement between the functions from the two views is that both views must contain enough information to learn the correct target functions. By posing the problem as learning with noise, only one of the views need to contain enough information while the other view need to contain enough information only for a weak predictor.

In this paper, we look at learning a linear function in the dual-view framework where one of the views is used to generate noisy labels to be learned using the other view. Optimizing $\Pr[h(X) = 1|Y' = -1] + \Pr[h(X) = -1|Y' = 1]$ is equivalent to optimizing expected weighted errors where errors on observed positive examples are multiplied by $\Pr[Y' = 0]$ and errors on observed negative examples are multiplied by $\Pr[Y' = 1]$. Unfortunately, minimizing disagreement is NP-hard for linear functions (Hoffgen et al., 1995). We observe that with this weight-
ing on the observed positive and negative examples, the actual positive examples have conditional probability greater than 0.5 of being labeled positive while the actual negative examples have conditional probability less than 0.5 of being labeled positive. With this observation, we take the approach of using logistic regression with the weighted examples to learn the real valued conditional probability and threshold the resulting conditional probability at 0.5 for classification. With appropriate regularization, we obtain a convex cost function to optimize. Our limited experiments indicates that this approach performs well on the web page classification domain (Blum & Mitchell, 1998).

We then consider relaxing the conditional independence assumption. We assume that the positive and negative examples are each generated by mixture models where the views are independent given the component of the mixture but not independent given only the label (positive or negative) of the example (see Figure 1).

\[
\begin{align*}
X_1 & \rightarrow X_2 \\
Y & \\
\end{align*}
\]

\[
\begin{align*}
X_1 & \rightarrow X_2 \\
C & \\
Y & \\
\end{align*}
\]

Figure 1. Graphical models. The left side shows the conditional independence of the views given the label. The right side shows conditional independence given the component of the mixture which does not necessarily mean conditional independence given the label.

This model is more reasonable when the positive and negative examples can be further partitioned into subclasses. For example, consider the task of classifying an image into vehicle and non-vehicle categories using the image features as one view and the text of the web page that contains the image as the other view. An object of type vehicle can be further partitioned into cars, motorcycles, trucks, boats and various other vehicles. Similarly, non-vehicles can be partitioned into various subclasses. Under the assumption that the support of the distributions of the components are disjoint in \( X_2 \), a predictor on \( X_1 \) will provide labels with constant noise rates in each component in \( X_2 \) although the rate may be different in each component. We analyse the robustness of using the proposed algorithm for learning linear functions under this condition by considering the conditional probability of the labels after the examples have been reweighted. Our analysis suggests that if the actual probability of the positive examples is small, then the algorithm is robust to variations in the noise levels in the positive examples but is not robust to variations in the noise levels of the negative examples. Our analysis also suggests trying to find weak predictors with one-sided errors - predictors that will often classify positive examples as negative but will never classify negative examples as positive. If such predictors are available, we can learn even when the examples are not conditionally independent but are instead drawn from conditionally independent components by placing sufficiently large weights on the observed positive examples.

2. Learning with Conditional Independence

2.1. Conditional Probability

Blum and Mitchell (Blum & Mitchell, 1998) have shown that under the conditional independence assumption, a weak predictor on \( X_1 \) will produce labels for \( X_2 \) with independent classification noise where positive examples are misclassified with probability \( \alpha \), negative examples are misclassified with probability \( \beta \) and \( \alpha + \beta < 1 \).

Assume that examples observed to be labeled positive are reweighted by multiplying by \( \Pr[Y' = 0] \) and examples observed to be labeled negative are reweighted by multiplying by \( \Pr[Y' = 1] \), where \( Y' \) is the observed noisy label. We consider the conditional probability of the label given the input under the modified distribution to show that actual positive examples have conditional probability greater than 0.5 of being labeled positive while actual negative examples have conditional probability smaller than 0.5 of being labeled positive.

Let \( \gamma \) be the probability of an actual positive instance before reweighting. The expected fraction of examples that are observed to be labeled positive is

\[
p = \gamma (1 - \alpha) + (1 - \gamma) \beta.
\]

Similarly, the expected fraction of examples that are observed to be labeled negative is

\[
(1 - p) = \gamma \alpha + (1 - \gamma) (1 - \beta).
\]

Let \( x \) be an actual positive instance. Then the probability that it is labeled positive is \( 1 - \alpha \) while the probability that it is labeled negative is \( \alpha \). The expected positive weight on \( x \) is

\[
(1 - \alpha)(1 - p) = (1 - \alpha) [\gamma \alpha + (1 - \gamma) (1 - \beta)].
\]

The expected negative weight on \( x \) is

\[
\alpha p = \alpha [\gamma (1 - \alpha) + (1 - \gamma) \beta].
\]

Subtracting (2) from (1), we get

\[
(1 - \alpha)(1 - p) - \alpha p = (1 - \gamma)(1 - \alpha - \beta).
\]
Hence, there is more positive weight than negative weight as long as $\gamma < 1$ and $\alpha + \beta < 1$. Now, let $x$ be an actual negative example. The expected negative weight on $x$ is

$$(1 - \beta)p = (1 - \beta) \left[ \gamma(1 - \alpha) + (1 - \gamma)\beta \right],$$

while the expected positive weight on $x$ is

$$\beta(1 - p) = \beta \left[ \gamma\alpha + (1 - \gamma)(1 - \beta) \right].$$

This gives

$$(1 - \beta)p - \beta(1 - p) = \gamma(1 - \beta) \alpha,$$

which is again greater than zero as long as $\gamma > 0$ and $\alpha + \beta < 1$. Hence thresholding at 0.5 on the conditional probability of the reweighted distribution will give the correct classification.

### 2.2. Learning linear functions

Linear functions of the form $g(x) = \sum_{j=1}^{k} w_j x_j + b$, where $x_j$, $j = 1, \ldots, n$ are the components of the input vector $x$ and $b$ is the bias, are practically effective and commonly used in machine learning.

To learn the conditional probability, we compose the linear function $g(x)$ with the sigmoid function to obtain $h(x) = 1/(1 + e^{-g(x)})$. We then perform maximum likelihood estimation on reweighted positive and negative examples, where the reweighting can be interpreted as multiple copies of the same examples. For each unweighted example, we obtain the logit loss $l(y, g(x)) = \ln(1 + e^{-yg(x)})$. Summing over reweighted examples, we obtain

$$-\sum_{y_i=1}^{n^{(+)}} l(y_i, g(x^i)) + \sum_{y_i=-1}^{n^{(-)}} l(y_i, g(x^i))$$

where $n^{(+)}$ is the number of positive examples, $n^{(-)}$ is the number of negative examples and $(x^1, y^1), \ldots, (x^n, y^n)$ is the set of examples. This is equivalent to minimizing the cost

$$\sum_{y_i=1}^{n^{(+)}} \frac{1}{n^{(+)}} l(y_i, g(x^i)) + \sum_{y_i=-1}^{n^{(-)}} \frac{1}{n^{(-)}} l(y_i, g(x^i)).$$

Finally, to prevent overfitting, we add the sum of squared values of the weights to the minimization process to obtain the final cost function

$$\sum_{y_i=1}^{n^{(+)}} \frac{1}{n^{(+)}} l(y_i, g(x^i)) + \sum_{y_i=-1}^{n^{(-)}} \frac{1}{n^{(-)}} l(y_i, g(x^i)) + c \left( \sum_{j=1}^{k} w_j^2 + b^2 \right),$$

where $c$ is a constant that can be adjusted to prevent overfitting. This cost function is convex.

If the function class is powerful enough to represent the conditional probability, then maximum likelihood estimation will give us accurate approximation of the conditional probability function when the sample size is large enough. In the case where the function class is not powerful enough, it is useful to view the logit loss as an upper bound to the $0 - 1$ loss (Mason et al., 1999). In this case, we are trying to minimize an upper bound to the sum of false positive and false negative frequencies, which still makes good sense even when the function class is not powerful enough to give a good approximation to the real valued conditional probability but can give a good approximation to the classifier.

To optimize the cost, we do simple gradient descent. The gradient of the loss function $l(y, g(x))$ with respect to $w_j$ is simply

$$\frac{\partial l(y, g(x))}{\partial w_j} = -x_j y e^{-yg(x)}/(1 + e^{-yg(x)}).$$

Let $w_{j,t}$ be the $j$th component of the weight vector at epoch $t$ and let $g_t(x) = \sum_{j=1}^{k} w_{j,t} x_j + b$. Let

$$\Delta_{j,t} = \sum_{y_i=1}^{n^{(+)}} x_j y_i e^{-y_i g_t(x^i)}/(1 + e^{-y_i g_t(x^i)}) + \sum_{y_i=-1}^{n^{(-)}} x_j y_i e^{-y_i g_t(x^i)}/(1 + e^{-y_i g_t(x^i)}),$$

be the $j$th component of the negative gradient of the weighted sum of losses at epoch $t$. The $j$th component negative gradient of the sum of squared weights is simply $-w_j$. We add a momentum term $\gamma \Delta_{j,t-1}, \gamma < 1$, to the gradient of the sum of losses (see e.g. (Mitchell, 1997)) to accelerate convergence of the gradient descent. Our update at each epoch $t$ then becomes

$$w_{j,t} = (1 - c)w_{j,t-1} + \eta(\Delta_{j,t} + \gamma \Delta_{j,t-1}),$$

where $\eta$ is the learning rate. Bias is implemented by extending the feature vector by an additional component and setting the additional component to a constant for all examples. Updates for the bias is treated in the same way as updates for the weights.

### 2.3. Experiment

Previous works on co-training typically assume that both views $X_1$ and $X_2$ contain enough information on their own to perfectly classify the instances. We perform a simple experiment to illustrate that

- a linear function on one view can be trained using noisy training data generated by another view on a domain where we expect the conditional independence condition to be reasonably well approximated
- the linear function can be successfully trained even when one of the views cannot approximate the target function well.

We use the task of classifying university web pages into “Course” and “Non-course” web pages from (Blum & Mitchell, 1998) for the experiment. In this task, two views of each web page is given - the page itself and the anchor text of links pointing to the web page. In this domain, it is easy to imagine a scenario where the anchor text from links does not provide sufficient information for classification. For example, in an environment where students do not have access to build their own home pages, many course
pages may not have any links to them at all! However, this does not prevent us from doing training the classifier effectively as illustrated in the experiments below.

We use the same data set as Blum and Mitchell (Blum & Mitchell, 1998) in our experiments. The data set consist of 230 course web pages and 821 non-course web pages collected from computer science departments in various universities together with the anchor text on the inlinks to those pages. For one view, we use only one feature from the inlinks - whether the anchor text in the links contains a digit. The other view consists of the bag of words in the web page after all HTML tags and stop words have been removed. Intuitively, using the feature of whether the anchor text contains a digit is likely to better approximate conditional independence than using the bag of words on the links (for course pages, links often contain the course codes, words that are likely to appear only in the corresponding web pages of the same courses).

We label the document as positive if the links contains a digit and negative otherwise. This turns out to be quite a good predictor with 209 out of 230 course pages labeled correctly. Furthermore, 811 out of 821 non-course pages are also labeled correctly giving a total accuracy of 97.1%. To simulate situations where many pages may not have links to them, we created two more data sets by randomly removing all links from 30% and 70% of the pages respectively. We use term frequencies of the words, normalized so that the length of each vector is one, as the feature vectors. An additional constant component of size 0.02 was added to each vector to implement the bias of the linear function.

In the experiments, we randomly selected 60% of pages as training data, 20% as validation data and 20% as test data. A linear function was trained on the training data for \( T = 500 \) epochs with learning rate \( \eta = 1/1051 \), momentum parameter \( \gamma = 0.99 \) and four decay parameters \( c \in \{0.01,0.05,0.1,0.5\} \). The sum of the false positive and false negative frequencies on the validation set was used to select the best \( c \) parameter. The linear function is then retrained on the combined training and validation set with the selected \( c \) parameter and tested on the test data. The results, averaged over 5 random splits of the data, shows that the accuracies are 95.9%, 95.1% and 93.6% respectively for the cases of no links removed, all links removed from 30% of pages and all links removed from 70% of pages. In contrast, the labels have expected accuracies of 97.1%, 91.4% and 83.8% respectively (a page is labeled a course page if a digit is present in the link anchor text). The expected accuracies of the labels are close to the best that can be done purely from the links view as pages with no links at all are best labeled as non-course. When the links are removed from a large number of web pages (e.g. 70%), the resulting labels are only slightly better than predicting that all pages are non-course pages (with accuracy 78.1%). In spite of that, the degradation in performance of the linear function is small. Consequently, we can view this as an example where dual-view learning works even though no classifier on the link view can classify all the examples perfectly.

3. Learning without Conditional Independence

3.1. Multiple Noise Partitions

To consider the robustness of the algorithm, we relax the conditional independence assumption of Blum and Mitchell (Blum & Mitchell, 1998) to allow the positive and negative examples to be drawn from multiple components where the views are conditionally independent given the component. If we assume that the support of the distributions of the components are disjoint in \( X_1 \), a linear predictor on \( X_1 \) can create labeled examples on \( X_2 \) will result in both the positive examples and negative examples being partitioned into multiple partitions with constant but different noise rates.

Assume that the positive examples can be partitioned into \( k_1 \) different partitions where in partition \( i \), the labels are flipped with probability \( \alpha_i \). The probability that a randomly drawn example belongs to partition \( i \) is \( \gamma_i \) for \( i = 1, \ldots, k_1 \). Let \( \gamma = \sum_{i=1}^{k_1} \gamma_i \). Similarly, assume that the negative examples can be partitioned into \( k_2 \) different partitions where in partition \( j \), the labels are flipped with probability \( \beta_j \). The probability that a randomly drawn example belongs to partition \( j \) is \( \zeta_j \) for \( j = 1, \ldots, k_2 \) and \( 1 - \gamma = \sum_{j=1}^{k_2} \zeta_j \).

The expected fraction of examples that are labeled positive is

\[
p = \sum_{i=1}^{k_1} \gamma_i (1 - \alpha_i) + \sum_{j=1}^{k_2} \zeta_j \beta_j.
\]

Similarly, the expected fraction of examples that are labeled negative is

\[
(1 - p) = \sum_{i=1}^{k_1} \gamma_i \alpha_i + \sum_{j=1}^{k_2} \zeta_j (1 - \beta_j).
\]

In the weighting scheme, the weight of an example with positive observed label is multiplied by \( 1 - p \) while the weight of an example with negative observed label is multiplied by \( p \).
Let \( x \) be a positive instance in partition \( r \). Then the probability that it is labeled positive is \( 1 - \alpha_r \) while the probability that it is labeled negative is \( \alpha_r \). The expected positive weight on \( x \) is

\[
(1 - \alpha_r)(1 - p) = (1 - \alpha_r) \left[ \sum_{i=1}^{k_1} \gamma_i \alpha_i + \sum_{j=1}^{k_2} \zeta_j (1 - \beta_j) \right].
\]

The expected negative weight on \( x \) is

\[
\alpha_r p = \alpha_r \left[ \sum_{i=1}^{k_1} \gamma_i (1 - \alpha_i) + \sum_{j=1}^{k_2} \zeta_j \right].
\]

Subtracting (4) from (3), we get

\[
(1 - \alpha_r)(1 - p) - \alpha_r p = \sum_{j=1}^{k_2} \zeta_j (1 - \alpha_r - \beta_j) + \sum_{i=1}^{k_1} \gamma_i (\alpha_i - \alpha_r)
\]

\[
= \gamma (1 - \gamma)(1 - \alpha_r - \bar{\beta}) - \gamma (\alpha_r - \bar{\alpha})
\]

\[
= (1 - \gamma) \left[ 1 - \alpha_r - \bar{\beta} - \frac{\gamma}{1 - \gamma} (\alpha_r - \bar{\alpha}) \right],
\]

where \( \bar{\alpha} = \sum_{i=1}^{k_1} \frac{\gamma \alpha_i}{\gamma} \) is the conditional expectation of \( \alpha_i \) given that the example is positive and \( \bar{\beta} = \sum_{j=1}^{k_2} \frac{\zeta_j \beta_j}{\gamma} \) is the conditional expectation of \( \beta_i \) given that the example is negative. The right hand side of (6) is greater than zero when

\[
\gamma < 1 \text{ and } \alpha_r + \bar{\beta} + \frac{\gamma}{1 - \gamma} (\alpha_r - \bar{\alpha}) < 1.
\]

Consider now a negative instance \( x \) in partition \( s \). The probability that it is labeled negative is \( 1 - \beta_s \) while the probability that it is labeled positive is \( \beta_s \). The expected negative weight on \( x \) is

\[
(1 - \beta_s)p = (1 - \beta_s) \left[ \sum_{i=1}^{k_1} \gamma_i (1 - \alpha_i) + \sum_{j=1}^{k_2} \zeta_j \beta_j \right],
\]

while the expected positive weight on \( x \) is

\[
\beta_s (1 - p) = \beta_s \left[ \sum_{i=1}^{k_1} \gamma_i \alpha_i + \sum_{j=1}^{k_2} \zeta_j (1 - \beta_j) \right].
\]

This gives

\[
(1 - \beta_s)p - \beta_s (1 - p) = \gamma \left[ 1 - \beta_s - \bar{\alpha} + \frac{1 - \gamma}{\gamma} (\beta_s - \bar{\beta}) \right]
\]

\[
= \gamma \left[ 1 - \beta_s - \bar{\alpha} + \frac{1 - \gamma}{\gamma} (\beta_s - \bar{\beta}) \right].
\]

The right hand side of (8) is greater than zero when

\[
\gamma > 0 \text{ and } \beta_s + \bar{\alpha} + \frac{1 - \gamma}{\gamma} (\beta_s - \bar{\beta}) < 1.
\]

Assuming that (7) is true for all positive partitions and that (9) is true for all negative partitions, we can see that thresholding at 0.5 will give the correct classification for both positive and negative examples.

Instead of learning a real valued conditional probability, we can also minimize the expected sum of false positive and false negative frequencies. In this case, it is easy to see that the target function minimizes the expected error and hence the expected sum of false positive and false negative frequencies. Conversely, if the classification of any function is different from the classification of the target function on any subset of points that has nonzero probability of being observed, the expected sum of false positive and false negative frequencies of the function will be greater than that of the target function. Hence, we can either try to learn the conditional probability using a real valued function or use a classifier that minimizes the sum of false positive and false negative frequencies.

In practice, the probability of positive examples \( \gamma \) is often much smaller than the probability of negative examples \( 1 - \gamma \). In such cases, variations in the noise in the negative partitions can easily cause condition (9) to be violated while variations in the noise in the positive partitions are better tolerated in condition (7).

### 3.2. Weak Predictors with One-sided Errors

Practitioners often have the options of which weak predictor to select when more than one exists. Under the condition that the positive and negative examples are drawn from multiple components where the views are conditionally independent given the component, weak predictors that make one-sided errors may be advantageous. Assume that we have a weak predictor that always classify negative examples correctly but often make mistakes on positive examples. Assume further that the weak predictor will classify some of the examples in each positive component as positive. Then, by increasing the weight on the examples observed to be labeled positive sufficiently, we can modify the distribution such that the conditional probability of being labeled positive is greater than 0.5 for all positive components and less than 0.5 for all negative components. We give a simple analysis of this situation.

As in Section 3.1, we assume that the positive examples can be partitioned into \( k_1 \) partitions where in partition \( i \), the labels are flipped with probability \( \alpha_i \) while the negative examples are never labeled as positive.

The expected fraction of examples that are labeled positive is

\[
p = \sum_{i=1}^{k_1} \gamma_i (1 - \alpha_i).
\]

Similarly, the expected fraction of examples that are la-
We performed some experiments to explore the following ability, minimizing the expected sum of weighted false positives and negatives. As before, thresholding at 0.5 will also label negative examples correctly. Since the probability that it is labeled negative is \(1 - \gamma\), the expected positive weight on \(x\) is

\[
(1 - p) = \sum_{i=1}^{k_1} \gamma_i \alpha_i + (1 - \gamma).
\]

We assume a weighting scheme where an error on an example that is labeled positive is multiplied by \(1 - p + v\) while an error on an example that is labeled negative is multiplied by \(p\). The variable \(v\) here can be adjusted to change the distribution and hence the conditional probability of a positive example.

Let \(x\) be a positive instance in partition \(r\). Then the probability that it is labeled positive is \(1 - \alpha_r\) while the probability that it is labeled negative is \(\alpha_r\). The expected positive weight on \(x\) is

\[
(1 - \alpha_r)(1 - p + v) = (1 - \alpha_r) \left[ \sum_{i=1}^{k_1} \gamma_i \alpha_i + (1 - \gamma) + v \right].
\]  
(10)

The expected negative weight on \(x\) is

\[
\alpha_r p = \alpha_r \left[ \sum_{i=1}^{k_1} \gamma_i (1 - \alpha_i) \right].
\]  
(11)

Subtracting (11) from (10), we get

\[
(1 - \alpha_r)(1 - p + v) - \alpha_r p = (1 - \gamma + v)(1 - \alpha_r) - \gamma (\alpha_r - \bar{\alpha})
\]

\[
= (1 - \gamma) \left[ 1 - \alpha_r - \frac{\gamma}{1 - \gamma + v} (\alpha_r - \bar{\alpha}) \right].
\]

Hence, there will be more positive weight than negative weight on \(x\) if

\[
v > \frac{\gamma (\alpha_r - \bar{\alpha})}{1 - \alpha_r} + \gamma - 1.
\]  
(12)

Assuming that \(\alpha_r < 1\) for all \(r\), substituting the largest value of \(\alpha_r\) in all the positive components will give the value of \(v\) which will allow the threshold of 0.5 to work correctly for all positive examples. Since the probability that a negative example is labeled negative is 1, thresholding at 0.5 will also label negative examples correctly. As before, instead of learning a real function for the conditional probability, minimizing the expected sum of weighted false positive and false negative with the weights given above will also give the correct target function.

3.3. Experiments

We performed some experiments to explore the following factors which are not taken into account in the analysis:

- finite sample size
- very high input dimension
- target function that may not be very well approximated by functions in the class.

We use the 20 Newsgroup dataset (Lang, 1995) in our experiments. The dataset consist of documents from 20 newsgroups with roughly 1000 documents in each group. We use the term frequency of the bag of words after removal of stop words, words that occurred no more than than 5 times in the collection and the headers of each document. The vectors were scaled to have length 1. An additional component of size 0.01 was added to each vector to implement the bias. We performed a random splitting of the data into 3 sets: the training set containing 50% of the documents, the validation set containing 20% of the documents and the test set containing 30% of the documents.

We performed the following experiments:

**Exp1**: The groups comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware and comp.windows.x are considered as positive groups with \(\alpha_r\) values of 0.6, 0.45, 0.3 and 0 respectively. All the other groups are considered as negative groups with \(\beta_r = 0\). This is a case where the values of \(\alpha_r\) varies in the positive class but analysis indicates that thresholding at 0.5 should give the correct classifier for the conditional probability of the modified distribution.

**Exp2**: The same groups are considered as positive with the same \(\alpha_r\) values as in Exp1. All the other groups are considered as negative with \(\beta_r = 0\) except comp.graphics which has \(\beta_r = 0.1\). Again thresholding at 0.5 should give the correct classifier for the conditional probability of the modified distribution.

**Exp3**: Same as Exp2. However, comp.graphics now has \(\beta_r = 0.3\) and analysis indicated that it will be incorrectly classified as positive with the threshold at 0.5.

**Exp4**: Same as Exp2. However, rec.sport.hockey now has \(\beta_r = 0.1\) and comp.graphics has \(\beta_r = 0\).

**Exp5**: Same as Exp4. However, rec.sport.hockey has \(\beta_r = 0.3\).

**Exp6**: Same as Exp5. However, rec.sport.baseball has \(\beta_r = 0.3\) in addition to rec.sport.hockey having \(\beta_r = 0.3\). Analysis indicates that both these groups should be misclassified as positive with the threshold at 0.5.

**Exp7**: Same as Exp1. However all the negative groups have \(\beta_r = 0.2\). Analysis indicates that all groups should be classified correctly with the threshold at 0.5.

**Exp8**: Same as Exp2. However, this time we are assuming that comp.graphics is a positive class with \(\alpha_r = 0.9\) instead of a negative group with \(\beta_r = 0.1\). We also add weight to the observed positive examples by setting the \(v\) parameter described in Section 3.2 to 2.

For each experiment, a linear function was trained on the training data using the algorithm described
in Section 2.2 for $T = 500$ epochs with learning rate $\eta = 1/(\text{number examples})$, momentum parameter $\gamma = 0.99$ and four decay parameters $c = \{0.003, 0.005, 0.007, 0.01, 0.03, 0.05\}$. The only exception is in Exp8 where a value $v = 2$ is used. We tested the sum of the false positive and false negative frequencies on the validation set to select the best $c$ parameter. The linear function is retrained on the combined training and validation set using the selected $c$ parameter and then tested on the test data. We then measure the fraction of examples in each newsgroup that is classified as 1 in the test set.

The results are shown in Table 1. We interpret the results below.

**Exp1:** The classification is quite good for most groups. The group comp.ms-windows has accuracy of more than 0.8 even though its training error rate is 0.6 due to the reweighting. However, classification on comp.graphics has error of 0.35. This is not unexpected as it belongs to the comp family of newsgroups along with the other positive class and may be harder to separate from them compared to the other newsgroups. The other groups with significant errors are sci.electronics and misc.forsale, both of which we expect to find some overlap in the types of articles from the comp family of newsgroups.

**Exp2:** Increasing the noise level of comp.graphics to 0.1 increases its noise level in the test set by approximately the same amount, which is not too bad.

**Exp3:** Analysis indicates that comp.graphics will now be misclassified and the test set shows a fairly high error rate of 0.68 as expected.

**Exp4:** The same experiment as Exp2, except this time we use rec.sport.hockey instead of comp.graphics. We also see only a small increase of error to 0.08 when the group’s training noise level is raised to 0.1.

**Exp5:** The same experiment as Exp3, except with rec.sport.hockey instead of comp.graphics. We expect a fairly high test error rate but only obtained a rate of 0.44 which is lower than expected.

**Exp6:** The same as Exp5 but this time rec.sport.baseball is also given a training error rate of 0.3. The error rate of rec.sport.hockey now goes up to 0.6 indicating that in Exp5 there was some difficulty distinguishing it from rec.sport.baseball.

**Exp7:** All the negative groups were given a moderate error rate of 0.2. Analysis indicates that all groups should have fairly low error rates. However this is true only for some of the groups such as alt.atheism, talk.politics.guns, talk.politics.mideast, talk.politics.misc and talk.religion.misc. Groups such as comp.graphics, sci.electronics and misc.forsale now have fairly high error rates while the other negative groups have moderate error rates. This indicates that we need to be careful with errors on the negative set with high dimensional data such as text.

**Exp8:** In this experiment, comp.graphics is treated as a positive group with a high error rate of 0.9 but our labels only have one-sided errors on the positive examples. On

<table>
<thead>
<tr>
<th>Group</th>
<th>Exp1</th>
<th>Exp2</th>
<th>Exp3</th>
<th>Exp4</th>
<th>Exp5</th>
<th>Exp6</th>
<th>Exp7</th>
<th>Exp8</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.057</td>
<td>0.007</td>
<td></td>
</tr>
<tr>
<td>rec.autos</td>
<td>0.019</td>
<td>0.019</td>
<td>0.019</td>
<td>0.016</td>
<td>0.016</td>
<td>0.198</td>
<td>0.084</td>
<td></td>
</tr>
<tr>
<td>sci.space</td>
<td>0.024</td>
<td>0.024</td>
<td>0.027</td>
<td>0.017</td>
<td>0.017</td>
<td>0.173</td>
<td>0.095</td>
<td></td>
</tr>
<tr>
<td>comp.graphics</td>
<td>0.349</td>
<td>0.439</td>
<td>0.678</td>
<td>0.287</td>
<td>0.287</td>
<td>0.322</td>
<td>0.682</td>
<td>0.747</td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>0.013</td>
<td>0.020</td>
<td>0.023</td>
<td>0.023</td>
<td>0.040</td>
<td>0.395</td>
<td>0.144</td>
<td>0.067</td>
</tr>
<tr>
<td>soc.religion.christian</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
<td>0.007</td>
<td>0.007</td>
<td>0.108</td>
<td>0.047</td>
<td></td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>0.806</td>
<td>0.777</td>
<td>0.784</td>
<td>0.781</td>
<td>0.755</td>
<td>0.803</td>
<td>0.871</td>
<td>0.942</td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>0.020</td>
<td>0.020</td>
<td>0.023</td>
<td>0.023</td>
<td>0.040</td>
<td>0.395</td>
<td>0.144</td>
<td>0.067</td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>0.003</td>
<td>0.003</td>
<td>0.007</td>
<td>0.010</td>
<td>0.007</td>
<td>0.010</td>
<td>0.063</td>
<td>0.028</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>0.803</td>
<td>0.794</td>
<td>0.816</td>
<td>0.794</td>
<td>0.778</td>
<td>0.816</td>
<td>0.930</td>
<td>0.943</td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>0.003</td>
<td>0.007</td>
<td>0.007</td>
<td>0.081</td>
<td>0.443</td>
<td>0.591</td>
<td>0.077</td>
<td>0.044</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.003</td>
<td>0.058</td>
<td>0.026</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>0.858</td>
<td>0.858</td>
<td>0.874</td>
<td>0.861</td>
<td>0.868</td>
<td>0.883</td>
<td>0.962</td>
<td>0.962</td>
</tr>
<tr>
<td>sci.crypt</td>
<td>0.035</td>
<td>0.035</td>
<td>0.047</td>
<td>0.022</td>
<td>0.028</td>
<td>0.025</td>
<td>0.151</td>
<td>0.088</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>0.003</td>
<td>0.003</td>
<td>0.000</td>
<td>0.003</td>
<td>0.007</td>
<td>0.007</td>
<td>0.058</td>
<td>0.010</td>
</tr>
<tr>
<td>comp.windows.x</td>
<td>0.899</td>
<td>0.924</td>
<td>0.931</td>
<td>0.910</td>
<td>0.913</td>
<td>0.917</td>
<td>0.972</td>
<td>0.979</td>
</tr>
<tr>
<td>sci.electronics</td>
<td>0.144</td>
<td>0.150</td>
<td>0.163</td>
<td>0.121</td>
<td>0.108</td>
<td>0.141</td>
<td>0.451</td>
<td>0.405</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>0.000</td>
<td>0.007</td>
<td>0.007</td>
<td>0.004</td>
<td>0.000</td>
<td>0.000</td>
<td>0.064</td>
<td>0.032</td>
</tr>
<tr>
<td>misc.forsale</td>
<td>0.175</td>
<td>0.155</td>
<td>0.162</td>
<td>0.125</td>
<td>0.142</td>
<td>0.172</td>
<td>0.413</td>
<td>0.353</td>
</tr>
<tr>
<td>sci.med</td>
<td>0.036</td>
<td>0.046</td>
<td>0.043</td>
<td>0.030</td>
<td>0.026</td>
<td>0.026</td>
<td>0.164</td>
<td>0.095</td>
</tr>
</tbody>
</table>

*Table 1.* Results of the sets of experiments performed on the 20 Newsgroup data set where each entry gives the fraction of the group classified as positive in the test set.
the identical labeling in Exp2, only 44% of the examples in comp.graphics are classified as positive in the test set when the observed positive and negative examples have equal total weight. With the increased weighting on the observed positive examples, 75% of the comp.graphics test examples are now classified as positive in the test set. However, there is also an increase in false positive in all negative groups, particularly in sci.electronics and misc.forsale.

Overall, the results of the experiments indicate that the analysis gives reasonable insights on the behaviour of the algorithm on very high dimensional data where the target function can only be approximated. However, care needs to be taken regarding overfitting and the ability of the approximating class to approximate the target function. In particular, false positives can be particularly numerous when the percentage of positive examples are small compared to the percentage of negative examples.

4. Related Works

Co-training was introduced in (Blum & Mitchell, 1998) where the reduction to learning with noise was given. An experiment using the naive bayes algorithm was also performed on the domain used in Section 2.3. The experiment there appears to be iteratively trying to maximize the agreement between the two functions being learned in the two views rather than trying to learn with noise. In (Collins & Singer, 1999), experiments on learning the named-entity class of a word where the spelling features of the word form one view and the context surrounding the text forms another was performed. There, the authors explicitly tried to maximize the agreement between the two functions in the two views, again through an iterative algorithm. Theoretical works analysing the idea of maximizing the agreement of the functions in the two views appear in (Dasgupta, 2001; Abney, 2002).

Learning with constant classification noise was studied in (Bylander, 1994; Blum et al., 1996) for variants of the perceptron algorithm. The algorithm for learning with noise that is used in this paper was also used for learning with positive and unlabeled examples in (Lee & Liu, 2003).

5. Conclusion

We have shown that linear functions can be successfully trained in dual-view learning scenarios by converting the problem into a problem of learning with noise. This method is particularly useful when only one of the views can represent the target function well. We have also examined the robustness of the algorithm by considering the case where the positive and negative examples are drawn from mixture distributions with conditional independence of the view holding given the component of the mixture and not the label of the example. Our work indicates that in such situations, a modified version of the algorithm will still work well given a weak predictor that makes only one-sided errors.

References


