Learning when to try hard at learning

Anonymous Author(s)
Affiliation
Address
email

[PROBABLY NEED A BETTER TITLE. –BM]

[NOAH, I PUT YOUR NAME SECOND BECAUSE YOUR NAME SEEMS TO BE LAST ON ALL OF YOUR RECENT PAPERS. I DON'T CARE WHICH OUR NAMES IS FIRST THOUGH, SO FEEL FREE TO SWAP IF YOU WANT. –BM]

Abstract

[I HAVE TROUBLE WRITING THIS PART. –BM]

1 Introduction

[GENERAL IDEAS ABOUT COST LEARNING ABSTRACTED AWAY FROM THE SVM DETAILS IN THE NEXT SECTION...? –BM]

2 Background

[I INTRODUCED SVMS THROUGH THE QUADRATIC PROGRAMMING FORMULATION BECAUSE IT SEEMED EASIER TO SUMMARIZE MARGIN MAXIMIZATION ACROSS ALL THE RELEVANT SVM VARIANTS THAT WAY WHILE STAYING TRUE TO THE LITERATURE... BUT MAYBE I SHOULD HAVE JUST KEPT IT IN THE FORM OF AN UNCONSTRAINED OPTIMIZATION PROBLEM FOR CONTINUITY WITH THE REST OF THIS PAPER? –BM]

[CITE EXAMPLE OF UNRELIABLE OUTPUT LABELS –BM]

[MIGHT WANT TO GIVE SOME MOTIVATING EXAMPLES? NOT SURE IF THAT'S NECESSARY OR NOT. ONE EXAMPLE OF MULTICLASS DOMAIN, AND ONE EXAMPLE WHERE COST FUNCTIONS ARE USED IN STRUCTURED DOMAINS –BM]

[MENTION EXAMPLES OF MEASURES OF 'DIFFICULTY'? CITE. –BM]

For the multiclass classification problem, we are given a set of m labelled training data instances $D = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\} \subset \mathcal{X} \times \mathcal{Y}$ where each example $\mathbf{x}_i$ is assigned label $\mathbf{y}_i$, and we want to learn a function $f : \mathcal{X} \to \mathcal{Y}$ which gives the correct label for any input taken from $\mathcal{X}$.[1] Past work has developed a multiclass support vector machine (SVM) which generalizes the concept of margin maximization employed by classical SVMs for binary classification tasks [2][3][4].

In particular, assuming that $\mathbf{g} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^k$ computes a featurized representation of an input-output pair, the multiclass SVM gives a score to a labelled instance $(\mathbf{x}, \mathbf{y})$ as:

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) \tag{1}$$

[1] We borrow heavily from the notation used in [1].

And makes predictions according to:

$$\hat{f}(\mathbf{x}; \mathbf{w}) = \underset{\mathbf{y}' \in \mathcal{Y}}{\operatorname{argmax}} F(\mathbf{x}, \mathbf{y}'; \mathbf{w}) \tag{2}$$

Where the feature weights $\mathbf{w}$ are given by the solution to the following soft-margin maximizing quadratic program:

$$\begin{aligned} &\min_{\xi \geq 0, \mathbf{w}} \big(\frac{\lambda_2}{2} \|\mathbf{w}\|_2^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \big) \\ &\text{s.t. } \forall i : \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i : F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) - F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}) \geq 1 - \xi_i \end{aligned} \tag{3}$$

This quadratic program chooses weights that minimize the number of misclassified instances while simultaneously trying to increase the margin between the scores of correct and incorrect labels, and the λ_2 hyper-parameter determines the relative importance of these two goals.[2]

Tsochantaridis et al. and Taskar et al. further generalized the margin maximization to make sense for domains where some prediction mistakes are more costly than others (especially domains where $\mathcal{Y}$ contains a large number of structured outputs) [1][5]. They introduce a function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ that determines the cost $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ of predicting label $\hat{\mathbf{y}}$ when the correct label is $\mathbf{y}$. The $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ is larger for more inaccurate $\hat{\mathbf{y}}$ predictions, and $\Delta(\mathbf{y}, \mathbf{y}) = 0$ giving no cost for a correct prediction. This cost function is incorporated into the SVM learning by modifying the multiclass SVM quadratic program through 'margin re-scaling' as:[3]

$$\begin{aligned} &\min_{\xi \geq 0, \mathbf{w}} \big(\frac{\lambda_2}{2} \|\mathbf{w}\|_2^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \big) \\ &\text{s.t. } \forall i : \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i : F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) - F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}) \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \end{aligned} \tag{4}$$

Intuitively, this margin re-scaling quadratic program allows the model to choose margins between between scores of $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$ proportional to the prediction cost $\Delta(\mathbf{y}_i, \hat{\mathbf{y}})$. This choice biases the model toward making low cost predictions over high cost predictions.

Whereas previous work assumes that we hand-select domain-specific cost functions that seem like good measures of the distances between labels, we propose learning this function from the data jointly with the prediction function f. Similarly to the hand-selected cost functions, the learned cost functions should reflect some notion of the distance between two labels, but as seen from the perspective of the model and its features rather than from the perspective of the person engineering the model. In other words, the value of $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ should be proportional to the ease with which the model discriminates between $\mathbf{y}$ and $\hat{\mathbf{y}}$ given its features. This choice of cost gives the model the freedom to shift its weights so that there are wider margins between labels which it can easily discriminate, and smaller margins between labels which it has difficulty discriminating. Such a policy is particularly useful when there are classes of incorrect predictions that are difficult or impossible for the model to systematically resolve due to unreliably annotated output labels or a choice of features that is insufficient for the task.

There are several ways that we might quantify the ease with which the model discriminates between two labels, but for now, we will keep the notion of 'easiness' fuzzy while establishing its relation to the cost function. We propose that an incorrect prediction $\hat{\mathbf{y}}$ where the actual label is $\mathbf{y}$ can fall into some number of 'incorrect prediction classes', and the overall cost of the incorrect prediction is the sum over the easiness of resolving (shrinking) each of these classes. More formally, let $\mathcal{S} \subseteq 2^{\mathcal{Y}^2}$ be

[2] This SVM and all other models in this paper can include non-regularized bias terms which we omit from our descriptions for readability, but these biases were included in the implementations we used for our experiments.

[3] Tsiochantaridis et al. also introduce an alternative 'slack re-scaling' technique, but the present paper mainly builds off of the 'margin re-scaling' approach.

a collection of incorrect prediction classes that collectively exhausts $\mathcal{Y}^2$. Assume that the 'easiness' with which a model of type $\mathcal{M}$ (e.g. SVM) given features $\mathbf{g}$ and data D resolves incorrect prediction class $S \in \mathcal{S}$ is given by $\mathcal{E}(S, \mathcal{M}, \mathbf{g}, D) \in \mathbb{R}$. Then, we assume that the cost is given by:

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{S \in \mathcal{S}} \mathcal{E}(S, \mathcal{M}, \mathbf{g}, D) \mathbb{1}((\mathbf{y}, \hat{\mathbf{y}}) \in S) = \mathcal{E}^\top \mathcal{S}(\mathbf{y}, \hat{\mathbf{y}}) \tag{5}$$

There are many possible choices for prediction classes $\mathcal{S}$, but in this paper, we focus on the following two:

$$\begin{aligned} \mathcal{S}_{[\mathcal{Y}]^2} &= \{S_{\{\mathbf{y},\mathbf{y}'\}} | \mathbf{y}, \mathbf{y}' \in \mathcal{Y}\} \\ \text{where } S_{\{\mathbf{y},\mathbf{y}'\}} &= \{(\mathbf{y}_1, \mathbf{y}_2) | (\mathbf{y}_1 = \mathbf{y} \wedge \mathbf{y}_2 = \mathbf{y}') \vee (\mathbf{y}_1 = \mathbf{y}' \wedge \mathbf{y}_2 = \mathbf{y})\} \end{aligned} \tag{6}$$

$$\mathcal{S}_{\mathcal{Y}^2} = \{S_{(\mathbf{y},\mathbf{y}')} | \mathbf{y}, \mathbf{y}' \in \mathcal{Y}\} \text{ where } S_{(\mathbf{y},\mathbf{y}')} = \{(\mathbf{y}, \mathbf{y}')\} \tag{7}$$

$\mathcal{S}_{[\mathcal{Y}]^2}$ contains a prediction class for every unordered pair of labels, and $\mathcal{S}_{\mathcal{Y}^2}$ contains a prediction class for every ordered pair. We might choose $\mathcal{S}_{[\mathcal{Y}]^2}$ when factoring the cost function if we believe it is equally easy to resolve misclassifications of $\mathbf{y}$ as $\mathbf{y}'$ and misclassifications of $\mathbf{y}'$ as $\mathbf{y}$. Otherwise, if we suspect these two incorrect prediction types to differ in easiness, then we might choose $\mathcal{S}_{\mathcal{Y}^2}$.

3 A cost learning model

We desire a model which approximates $\mathcal{E}(S, \mathcal{M}, \mathbf{g}, D)$ (for some notion of 'easiness') to estimate prediction costs while simultaneous learning feature weights $\mathbf{w}$. Our proposal follows the intuition that the easiness $\mathcal{E}(S, \mathcal{M}, \mathbf{g}, D)$ of prediction class S for model class $\mathcal{M}$ is related to the size of the set:

$$S_{\mathcal{M},\mathbf{g},D} = \{(\mathbf{x}_i, \mathbf{y}_i) | (\mathbf{y}_i, \hat{f}^{\Delta}_{\mathcal{M},\mathbf{g},D}(\mathbf{x}_i; \mathbf{w}_{\mathcal{M},\mathbf{g},D})) \in S \text{ and } (\mathbf{x}_i, \mathbf{y}_i) \in D\} \tag{8}$$

Where $\mathbf{w}_{\mathcal{M},\mathbf{g},D}$ are the learned feature weights, and $\hat{f}^{\Delta}_{\mathcal{M},\mathbf{g},D}$ is the model's prediction function augmented with the cost:

$$\hat{f}^{\Delta}_{\mathcal{M},\mathbf{g},D}(\mathbf{x}_i; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \Big(F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}) + \Delta(\mathbf{y}_i, \mathbf{y}) \Big) \tag{9}$$

The size of $S_{\mathcal{M},\mathbf{g},D}$ is the number of training examples with margin violations in class S. If the size of this set is large, we might infer that the model has trouble shrinking it, and so it's not 'easy'. This might lead us to conclude that $S_{\mathcal{M},\mathbf{g},D}$ tends to shrink with 'easiness'. However, for many data sets and choices of $\mathcal{S}$, the size of each $S_{\mathcal{M},\mathbf{g},D}$ can be inherently biased by the data independently of 'easiness'. For example, for $S_{\{\mathbf{y},\mathbf{y}'\}} \in \mathcal{S}_{[\mathcal{Y}]^2}$, the size of $S_{\{\mathbf{y},\mathbf{y}'\},\mathcal{M},\mathbf{g},D}$ is biased by the number of examples in the training data which have output labels $\mathbf{y}$ and $\mathbf{y}'$–if there are few training examples of labels $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$, then the size of $S_{\{\mathbf{y},\mathbf{y}'\},\mathcal{M},\mathbf{g},D}$ will necessarily be small relative to other prediction classes. Furthermore, we expect output labels which occur infrequently in the training data to be more difficult for the model to predict correctly, so this will lead to the size of $S_{\{\mathbf{y},\mathbf{y}'\},\mathcal{M},\mathbf{g},D}$ increasing with the 'easiness' of $S_{\{\mathbf{y},\mathbf{y}'\}}$ which is opposite the conclusion that that we draw if we think of the size of $S_{\{\mathbf{y},\mathbf{y}'\},\mathcal{M},\mathbf{g},D}$ as increasing due to the model's difficulty in shrinking it. In general, this suggests that if we want the size of $S_{\mathcal{M},\mathbf{g},D}$ to vary with easiness, we need to normalize it to account for properties of the training data that introduce irrelevant biases.

3.1 A measure of 'easiness'

[MIGHT WANT TO DISCUSS OTHER POSSIBILITIES, OR GENERALLY REFER TO OTHER POSSIBILITIES IN FUTURE WORK –BM]

The above observations suggest the following as a possible measure of 'easiness':

$$\mathcal{E}(S, \mathcal{M}, \mathbf{g}, D) = \max\left(0, 1 - \frac{|S_{\mathcal{M},\mathbf{g},D}|}{n_{S,\mathcal{M},\mathbf{g},D}}\right) \tag{10}$$

Where $n_{S,\mathcal{M},\mathbf{g},D}$ is a normalization constant which gives the maximum possible value we expect for the size of $S_{\mathcal{M},\mathbf{g},D}$, accounting for irrelevant biases introduced by the data as discussed above. This measure of easiness is in $[0, 1]$, and it has the property that if $|S_{\mathcal{M},\mathbf{g},D}| \geq n_{S,\mathcal{M},\mathbf{g},D}$, then $\mathcal{E}(S, \mathcal{M}, \mathbf{g}, D) = 0$, indicating that $S_{\mathcal{M},\mathbf{g},D}$ is so difficult to shrink that its size is greater than our expected upper bound.

3.2 A cost learning objective

[CITE SELF-PACED LEARNING FOR INSPIRATION FOR NEW OBJECTIVE FUNCTION? –BM]

[ADD FOOTNOTE ABOUT RELATIONSHIP BETWEEN NORM IN OBJECTIVE AND MAHALANOBIS NORM –BM]

[IS THERE ANY MATH I SHOULD BE MORE EXPLICIT ABOUT? –BM]

We can modify the margin re-scaling SVM learning procedure given by Quadratic Program 4 to learn the cost function according to the easiness measure shown in Equation 10. First, we transform the quadratic program into the equivalent unconstrained optimization problem:

$$\min_{\mathbf{w}} \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{m}\left(-F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) + \max_{\mathbf{y}\in\mathcal{Y}}\left(F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}) + \Delta(\mathbf{y}_i, \mathbf{y})\right)\right) \tag{11}$$

And we modify this function to include the cost learning as:

$$\min_{\hat{\mathcal{E}}\geq 0, \mathbf{w}} \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{m}\left(-F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) + \max_{\mathbf{y}\in\mathcal{Y}}\left(F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}) + \hat{\mathcal{E}}^\top \mathcal{S}(\mathbf{y}_i, \mathbf{y})\right)\right) - \hat{\mathcal{E}}^\top \mathbf{n} + \frac{1}{2}\|\hat{\mathcal{E}}\|_{\mathbf{n}}^2 \tag{12}$$

Where $\mathbf{n}$ is the vector of normalization constants, and $\|\hat{\mathcal{E}}\|_n^2 = \sum_{S\in\mathcal{S}} n_S \hat{\mathcal{E}}_S^2$.[4] The idea for this new objective function is to use the $-\hat{\mathcal{E}}^\top \mathbf{n}$ term to select which $\hat{\mathcal{E}}_S$ should be non-zero (or correspondingly which S are not impossibly difficult), and use the $\|\hat{\mathcal{E}}\|_{\mathbf{n}}^2$ to limit the magnitude of $\hat{\mathcal{E}}_S$.

If the solution to Objective 12 is at point that is differentiable with respect to $\hat{\mathcal{E}}_S$, then it's easy to show that $\hat{\mathcal{E}}_S$ has exactly the value given by Equation 10. Otherwise, if the solution is at a non-differentiable point, then $\hat{\mathcal{E}}_S$ has a value that approximates Equation 10 in a sensible way. Specifically, the non-differentiable points of Objective 12 occur due to ties between multiple sequences of labels for the minimum value solution, and each of these equally good label sequences has differently sized incorrect prediction classes, giving different values of 'easiness' for each class according to Equation 10. The values of $\hat{\mathcal{E}}_S$ at the non-differentiable points are between the values given by Equation 10 for each tied best label sequence.

[IS THIS NON-DIFFERENTIABLE PART UNDERSTANDABLE? IS IT ENOUGH, OR DO I NEED TO GIVE THE PROOF? I STILL HAVEN'T ACTUALLY GONE THROUGH THE PROOF COMPLETELY, SO MAYBE WE SHOULD AT LEAST GO OVER THE REASONING TO MAKE SURE I'M NOT CRAZY. –BM]

3.3 Some 'easiness' normalization constants

[MENTION CHOOSING N USING BASELINE SVM ESPECIALLY IF THIS IS INCLUDED IN RESULTS. BUT OTHERWISE MAYBE PUT IT IN FUTURE WORK SINCE CHOOSING N BASED ON OTHER MODELS IS A GENERAL TOPIC TO EXPLORE –BM]

[4] We abbreviate $\hat{\mathcal{E}}(S, \mathcal{M}, \mathbf{g}, D)$ and $n_{S,\mathcal{M},\mathbf{g},D}$ as $\hat{\mathcal{E}}_S$ and n_S for readability.

The appropriate choice for the normalization vector $\mathbf{n}$ in Objective 12 depends on the prediction classes $\mathcal{S}$ and the type of irrelevant bias we are trying to remove from 'easiness' measure. For $\mathcal{S}_{[\mathcal{Y}]^2}$ and $\mathcal{S}_{\mathcal{Y}^2}$, we want to remove the bias introduced into the size of the prediction classes by non-uniform label distributions, as discussed at the beginning of Section 3. Otherwise, the model will tend to over-estimate the costs of incorrect predictions involving labels that occur infrequently in the training data. The following are two plausible choices of $\mathbf{n}$ to achieve this goal. In each, assume that $D_{\mathbf{y}}$ is the set of training examples for which $\mathbf{y}$ is the correct label.

1. **Logical** $\mathbf{n}$: Choose n_S as an upper bound on $|S|$ that cannot possibly be violated given the training data. For prediction classes $S_{\mathbf{y},\mathbf{y}'} \in \mathcal{S}_{\mathcal{Y}^2}$, choose $n_{S_{\mathbf{y},\mathbf{y}'}} = |D_{\mathbf{y}}|$, and for prediction classes $S_{\{\mathbf{y},\mathbf{y}'\}} \in \mathcal{S}_{[\mathcal{Y}]^2}$, choose $n_{S_{\{\mathbf{y},\mathbf{y}'\}}} = \max\left(|D_{\mathbf{y}}|, |D_{\mathbf{y}'}|\right)$.
2. **Expected** $\mathbf{n}$: Choose n_S as the expected number of times a model makes a mistake in S if it predicts labels at random according to their distribution in the training data. For prediction classes $S_{\mathbf{y},\mathbf{y}'} \in \mathcal{S}_{\mathcal{Y}^2}$, choose $n_{S_{\mathbf{y},\mathbf{y}'}} = \frac{|D_{\mathbf{y}}||D_{\mathbf{y}'}|}{m}$, and for prediction classes $S_{\{\mathbf{y},\mathbf{y}'\}} \in \mathcal{S}_{[\mathcal{Y}]^2}$, choose $n_{S_{\{\mathbf{y},\mathbf{y}'\}}} = \frac{2|D_{\mathbf{y}}||D_{\mathbf{y}'}|}{m}$. Other variations of this idea might choose alternative models by which to compute the expectation, but model that makes predictions at according to the training data label distribution seems reasonable for getting rid of the label distribution bias.

In general, the **Logical** choice of $\mathbf{n}$ is an upper bound on the **Expected** choice. The **Logical** choice will tend to over-estimate the maximum size of each prediction class, but we might choose it over **Expected** if we have reason to believe that it is difficult to estimate the baseline rate at which the model is biased to predict certain labels by the label distribution independent of the inputs.

[MORE DETAIL ON WHY EXPECTED MIGHT BE A BETTER CHOICE THAN LOGICAL –BM]

4 Experiments

[ADD GRAPHS SHOWING CONVERGENCE OF SGD? –BM]

We implemented the multiclass SVM and normalized cost learning SVM (SVMCLN) using stochastic (sub-)gradient descent (SGD) to approximate Objectives 11 and 12 (see [6] on SGD). Our SGD implementation used a learning rate of $\frac{1}{\lambda_2 t}$ at time-step t following the Pegasos algorithm in [7]. We ran several experiments to compare these models on standard text classification corpora. In each experiment, we compared the accuracies of the models on standard train/test split given by each respective data corpus while also randomly selecting a 10% subset of the training data to use as a dev set. We used the dev set to perform a grid search over 16 possible values ranging from 10^{-6} to 0.5×10^2 for the λ_2 hyper-parameter in each model. After the grid search, we fixed the best value for λ_2, and retrained on the full training data, evaluating the final performance of the model on the test set.

Due to the stochasticity of the optimization algorithm, we expected some noisiness in its convergence. In general, we ran SGD for 150 iterations over the random permutations of the full data. After the 150 iterations, the accuracy and predictions of each model on the dev/test sets were relatively stable; the accuracy varied by at most 0.001 and fewer 5% of the predictions changed during the last 10 iterations over the training data.

4.1 Datasets

[CITE HTTP://WWW.AAAI.ORG/PAPERS/AAAI/2006/AAAI06-121.PDF ON RELATIVELY LOW DIFFERENCE IN PERFORMANCE BETWEEN LOG(TF) AND TFIDF –BM]

[CITE HTTP://QWONE.COM/~JASON/WRITING/LOOCV.PDF FOR EXAMPLE USE OF LOG(TF) –BM]

[PREPROCESSED USING METHOD FROM HTTP://WEB.IST.UTL.PT/ACARDOSO/DOCS/2007-PHD-THESIS.PDF –BM]

We compared the performance of the models on the 20 Newsgroups and the Reuters-21578 (R52) data sets. We chose these two data sets because they both have a relatively large number of output labels, some of which might not be distinctive enough for the SVM to plausibly learn well given its features, in which case we might expect a gain in accuracy from SVMCLN. As shown below, we found out that this expectation was correct for the 20 Newsgroups data, but incorrect for the Reuters data, for reasons that became apparent to us after we reviewed the results.

For both of these data sets, the target labels $\mathcal{Y}$ are single document topics/categories, and the inputs $\mathcal{X}$ are documents. We preprocessed each text document from both corpora using the procedure given in [FILL IN] (convert to lower-case, remove symbols, etc), and then we computed $\mathbf{g}$ as normalized log unigram term-frequency features. In particular:

$$\mathbf{g}_{\mathbf{y}'}(\mathbf{x}, \mathbf{y}) = \mathbb{1}(\mathbf{y} = \mathbf{y}')\mathbf{f}(\mathbf{x}) \tag{13}$$

Where $\mathbf{f}(\mathbf{x})$ is a normalized vector whose elements are $\log(1 + tf(\mathbf{x}, v))$ when v is a unigram in the corpus vocabulary and $tf(\mathbf{x}, v)$ is the frequency of v in document $\mathbf{x}$.

4.1.1 Reuters 21578 (R52)

The Reuters-21578 R52 corpus contains 9100 documents from the original Reuters-21578 data, each of which is labeled with one of 52 possible topics. [5] The documents are divided along the Reuters-21578 'ModApte' split into $\tilde{7}0\%$ training and $\tilde{3}0\%$ test. The label distribution is extremely non-uniform, with 43% of the documents assigned to a single topic, and 71% of the other topics each assigned fewer than 50 (0.5%) documents.

4.1.2 20 newsgroups

The 'by date' version of the 20 Newsgroups data contains 18846 documents sorted by date into 60% for training and 40% for testing. [6] In comparison to the Reuters data, the newsgroups distribution of 20 categories is nearly uniform, with some topics highly related and some topics entirely unrelated. The relatedness of the topics is approximately captured by their hierarchical naming scheme (e.g. there is *rec.autos* and *rec.motorcycles* which are likely to be highly related to each other, but mostly unrelated to *sci.space*). We expected that this hierarchy would be encoded by the learned 'easiness' approximations, since the 'easiness' with which the model discriminates two categories likely decreases with their relatedness.

4.2 Results

[ALL 20NEWS TABLE NUMBERS ARE WRONG. NEED TO FIX THEM AFTER EXPERIMENTS RERUN. –BM]

Table 1 shows the micro-averaged accuracies on the Reuters and 20 newsgroups tasks for the SVM baseline model and versions of SVMCLN with different choices of normalization constants $\mathbf{n}$ and incorrect prediction classes $\mathcal{S}$. The last column in the table shows that none of the SVMCLN variations improve the ccuracy over the SVM on the Reuters task, but inspection of the confusion matrix for the SVM baseline provides some intuition for this lack of improvement. The matrix shows that 53% of the SVM's mistakes were on examples with whose topics had 10 or fewer test examples, all off which were predicted incorrectly. Furthermore, there is no non-diagonal element in the confusion matrix with more than 10 mistakes. These observations suggest that many of the SVM baseline's mistakes come from infrequent labels rather than systematic conflations between certain label pairs, and further that all of the incorrect prediction classes in the cost learning model will be extremely small to begin with. As a result, it's unlikely that SVMCLN would be able to re-scale the costs to greatly shrink any incorrect prediction classes. In hindsight, this provides a good example of a good example of how analysis of the errors made by the standard SVM can determine whether cost learning will be beneficial.

In contrast, the fourth column of Table 1 shows that every SVMCLN variation improves the accuracy over the SVM on the 20 Newsgroups data. Unsurprisingly, the **Logical** and **Expected** normalized

[5] See `http://www.csmining.org/index.php/r52-and-r8-of-reuters-21578.html`.
[6] See `http://qwone.com/~jason/20Newsgroups/`.

Table 1: Micro-averaged Accuracies

Model	n	$\mathcal{S}$	20news	20news level 2	20news level 1	Reuters
SVM	N/A	N/A	0.7760	0.8008	0.8654	0.9213
SVMCLN	None	$\mathcal{Y}^2$	0.8008	0.8259	0.8752	0.9213
SVMCLN	None	$[\mathcal{Y}]^2$	0.8011	0.8257	0.8751	0.9194
SVMCLN	Logical	$\mathcal{Y}^2$	0.8024	0.8271	0.8769	0.9210
SVMCLN	Logical	$[\mathcal{Y}]^2$	0.8376	0.8631	0.9142	0.9159
SVMCLN	Expected	$\mathcal{Y}^2$	0.8303	0.8557	0.9092	0.9159
SVMCLN	Expected	$[\mathcal{Y}]^2$	0.8307	0.8558	0.9084	0.9171

versions perform better than the non-normalized versions since they should give better estimates of the 'easiness' measure and cost function. Also, each $\mathcal{S}_{[\mathcal{Y}]^2}$ version performs slightly better than each $\mathcal{S}_{[\mathcal{Y}]^2}$ version, which makes sense given that the ease of resolving mistakes in $S_{\mathbf{y},\mathbf{y}'}$ should be the same as the ease of resolving mistakes in $S_{\mathbf{y}',\mathbf{y}}$, and so the ordering on $\mathbf{y}$ and $\mathbf{y}'$ gives the model unnecessary extra parameters. We also expected **Expected** to perform better than **Logical** since **Logical** over-estimates the maximum value of each incorrect prediction class, but this expectation was not met with $\mathcal{S}_{[\mathcal{Y}]^2}$ prediction classes, possibly because the **Expected** normalizers tend to under-estimate.

The hierarchical structure of the 20 Newsgroups topics encodes a notion of distance between topics as distance within the hierarchy. The fifth and sixth columns of Table 1 show the accuracies computed when nearby topics in the hierarchy are are collapsed into single topics–the fifth column shows the accuracies computed when all topics that are the same to two levels deep in the hierarchy are collapsed into a single topic, and the sixth column shows the accuracies computed when all topics that are the same at the first level of the hierarchy are collapsed into a single topic.

[I THINK THERE ARE WAYS TO MAKE THE FOLLOWING RESULT CLEARER IF WE COMPUTE OTHER NUMBERS... BUT I DON'T KNOW IF WE HAVE TIME FOR THAT. THIS MAY BE A BIT CONFUSING I THINK THOUGH –BM]

The cost learning should learn a notion of distance between topics approximated by the cost function, and we expect this notion of distance to approximate the notion of distance encoded by the hierarchy. The approximated notion of distance encoded in the cost function should cause SVMCLN to improve at distinguishing topics that it estimates to be far apart from each other. So if SVMCLN's learned distances approximate the distances through the hierarchy, then we should expect the SVMCLN's collapsed accuracy improvements to be nearly as great as the uncollapsed accuracy improvements. This is shown by the [FILL] accuracy improvements in the fourth, fifth, and sixth columns of Table 1 by the **Logical** $\mathcal{S}_{[\mathcal{Y}]^2}$ version of SVMCLN.

[FIX THE ABOVE PARAGRAPH WHEN GET NEW NUMBERS –BM]

[CITE HIERARCHICAL CLUSTERING. BE SPECIFIC ABOUT WHICH TYPE OF HIERARCHICAL CLUSTERING IS USED. –BM]

[ADD HIERARCHY FIGURE. –BM]

We also directly evaluated the extent to which the cost function approximates the newsgroup hierarchy by constructing a hierarchy from the 'easiness' approximations and comparing it to the newsgroup hierarchy. In order to construct the hierarchy, we ran hierarchical clustering on the newsgroups where each approximated 'easiness' value acts as a distance measure. The approximate hierarchy for the **Logical** $\mathcal{S}_{[\mathcal{Y}]^2}$ SVMCLN is shown in figure [FILL], and the true Newsgroup hierarchy is shown in figure [FILL].

[ADD MEASUREMENTS OF SIMILARITY BETWEEN HIERARCHIES –BM]

5 Discussion

5.1 Related literature

[I HAVE TROUBLE WRITING THIS PART. –BM]

5.2 Future work

[THERE'S PROBABLY LOTS OF STUFF I DON'T KNOW ABOUT WHICH COULD GO HERE. HERE ARE SOME IDEAS THOUGH: –BM]

[INCORPORATE INPUT FEATURES INTO COST –BM]

[OTHER CHOICES FOR N –BM]

[OTHER MEASURES OF 'EASINESS' –BM]

[STRUCTURED VERSION –BM]

[ALTERNATING MINIMIZATION VERSION –BM]

[RELATIONSHIP BETWEEN MODEL 'DIFFICULTY' AND 'TASK' DIFFICULTY –BM]

[CHARACTERIZE PROPERTIES OF DATA/FEATURES THAT DETERMINE WHETHER COST LEARNING IS USEFUL –BM]

[THE 'REFERENCES' HEADING GIVEN BY THE BIBLIOGRAPHY COMMAND IS THE WRONG SIZE FONT. NEEDS TO BE THE SIZE OF A 'THIRD LEVEL HEADING'. HOW TO CHANGE THIS? –BM]

References

[1] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[2] Vladimir N Vapnik. Statistical learning theory (adaptive and learning systems for signal processing, communications and control series), 1998.

[3] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[4] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998.

[5] Ben Taskar Carlos Guestrin Daphne Koller. Max-margin markov networks. 2003.

[6] G George Yin and Harold Joseph Kushner. *Stochastic approximation and recursive algorithms and applications*. Springer, 2003.

[7] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.