Feature-selecting Logistmar Gramression on Synthetic FOL Data

Bill McDowell
Stanford University
450 Serra Mall
Stanford, CA 94305

Abstract

This paper discusses machine learning models that learn to perform supervised learning tasks over potentially infinite dimensional input feature spaces defined inductively through "feature grammars". A "feature grammar" consists of a set of rules for constructing increasingly complex features from simpler features, and it implicitly defines a discrete space of features where two features are neighbors if one can be constructed from another by the application of some deterministic feature transformation rules. The learning models use such rules to navigate the feature space in search of relevant features under the assumption that the grammatical relationship between two features in the space encodes some information about their relevance a task. In this project, I implement and experiment with sparse logistic regression inspired feature grammar models on a synthetic data set whose feature spaces consist of first-order logic formulas.[1]

In supervised machine learning tasks, we are given a set of training data $D = \{(x^1, l^1), (x^2, l^2), \ldots, (x^n, l^n)\}$ where each data instance $x^d \in X$ has a target label $l^d \in L$, and we wish to learn a function $c : X \to L$ (Mitchell 1997). Many learning algorithms assume that data instances can be represented by some finite feature vector in an m-dimensional space given by a function $\mathbf{f} : X \to \mathbb{R}^m$. The particular choice of featurization $\mathbf{f}$ is task specific, and is usually chosen according to prior knowledge of the relevant domain.

For many tasks, our prior knowledge of the relevant domain is insufficient for precisely specifying the relevant set of features to include in $\mathbf{f}$, and so feature selection algorithms might be used to determine a relevant set of features automatically (Guyon and Elisseeff 2003). These algorithms typically assume that we begin with some large finite set containing both relevant and irrelevant features, and selects a smaller subset from these based on their use with respect to approximating the target function.

In this paper, I propose "feature grammar" models that weaken the requirement of feature selection algorithms that we must start with a finite set of features from which to select those that are relevant. Rather than starting with a finite feature set from which to select, these models will assume that we have access to an infinite set of features describable by some formal language. Furthermore, these models will assume that the features describable in this language can be generated by some grammar, and that we have some heuristic rules for navigating the space of feature derivations by this grammar in search of relevant features. Intuitively, these rules should encode assumptions that if a feature of some form is relevant, then a feature of another grammatically related form is also likely to be relevant. This assumption will allow a feature selection model to traverse the grammatical feature space in search of relevant features.

As a concrete example, imagine a model that is learning to classify mentions of people in text documents into whether they are business owners. We might define a set of features for a given noun-phrase as the set of all unigrams, bigrams, trigrams, $\ldots n$-grams that occur in the same sentence. k-grams in this space might be related to superstring $(k+1)$-grams by heuristic rule. Specifically, a rule might allow construction of the bigram feature "owns businesses" from the unigram feature "owns". If the learning model discovers that the "owns" feature is relevant to the business owner classification task, then it might consider selecting the "owns business" feature under the assumption that the grammatical relationship between the two features indicates that one feature's task-relevance is indicative of the other feature's relevance. In contrast, a traditional feature selection model would require selecting a few k-gram features from a predetermined finite set.

In practice, models that operate over a potentially infinite grammatical feature space might be able to discover more compositional features (e.g. logically complex features) than traditional feature selection algorithms given finite resources. Traditional algorithms would require starting with a set of all possible compositions to be considered, and the desired set may be too large to practically consider in a reasonable amount of time. The heuristic rules for traversing a grammatical feature space might allow such features to be explored more efficiently.

The particular models considered in this paper build on my previous work on constructing a "feature grammar" version of logistic regression. In the next section, I will briefly summarize this past work, and suggest possible relationships

[1] My implementation for this project can be found at `https://github.com/forkunited/grampy`.

to other types of machine learning models. Following the review, I will propose an L1-regularized variation on my previous model, and describe an approximation algorithm for tuning the parameters. Next, I will describe the results of applying this model to a synthetic data set in a first-order logic feature space. Lastly, I will conclude with some general discussion of possible directions for future work.

Prior work

In the past, I constructed L2-regularized "logistmar gramression"—a variation on logistic regression that learns to perform a classification task given an initial seed feature vocabulary and a set of heuristic rules for expanding this vocabulary with other grammatically related features. The heuristics given to logistmar gramression are a set of rules for navigating the grammar's infinite feature space in search of relevant features, and the logistmar gramression training procedure uses these rules to determine which features in the space should be selected to have non-zero weight. If the heuristics guide the model to all relevant parts of the space prior to irrelevant parts of the space, then logistmar gramression will produce the same classifier as L2-regularized logistic regression trained with a vocabulary consisting of mostly relevant features.

In standard logistic regression (Ng 2000), we assume binary labels $l \in \{0, 1\}$ for the data , and for each data instance (x, l) featurized by $\mathbf{f} : X \to \mathbb{R}^m$, we assume the data is given by distributions of the form:

$$p_{d,\mathbf{w},\mathbf{f}} = P(L = 1 \mid x^d; \mathbf{w}, \mathbf{f}) = \frac{e^{\mathbf{w}^T \mathbf{f}(x^d)}}{1 + e^{\mathbf{w}^T \mathbf{f}(x^d)}}$$

In these distributions, the ratios of parameters in the weight vectors $\mathbf{w}_+$ and $\mathbf{w}_-$ determine how highly weighted a feature is with respect to predicting that the label $L = 1$ for a given x. For training, we use gradient ascent to choose $\mathbf{w}$ that maximizes the log-likelihood of the data set D under these distributions:

$$L(\mathbf{w}) = \sum_{d \in D} \Big(l^d \log p_{d,\mathbf{w}} + (1 - l^d) \log(1 - p_{d,\mathbf{w}}) \Big)$$

In practice, for this work, we approximately maximize variations on this objective using *stochastic* gradient descent (SGD)—which only computes gradients with respect to single or small batches of training examples rather than the full data set from the log-likelihood objective[2]. Furthermore, we consider adding L2 or L1 regularization terms to the objective which ensure many small or zero magnitude feature weights in the trained model—preventing overfitting in the L2 case, and performing feature selection in the L1 case (Ng 2004).

The first version of logistmar gramression chose parameters to maximize an L2 regularized objective similar to the one given above. However, the model assumed that we have access to a feature space F_G determined by a formal language characterized by a grammar G, a seed feature vocabulary $F_0 \subset F_G$, and a set of heuristic functions for $H = \{h : F_G \to F_G\}$ for navigating the grammatical space of features starting at the seed vocabulary. Furthermore, the model assumed that F_0 consisted of mostly relevant features, and that the heuristic rules in H encode assumptions that a feature of some form might be relevant given that a feature of another form is relevant. Given these assumptions, the logistmar gramression algorithm starts by performing SGD to maximize the L2-regularized version of the log-likelihood objective described above with respect to the features in the seed vocabulary F_0. While ascending the gradient, if the magnitude of a feature weight in $\mathbf{w}$ goes above a certain threshold t, the algorithm applies the rules in H to the corresponding feature, and adds the resulting features to the feature vectors produced by $\mathbf{f}$. After new features are added to $\mathbf{f}$, SGD continues adjusting an extended weight vector for the extended featurization according to the usual gradient update rule.[3]

The first formulation of logistmar gramression used L2 regularization to prevent overfitting and to ensure that the algorithm converged to a solution where only a finite set of features in F_G have non-zero weight. L2 regularization was originally chosen over L1 regularization in order to keep the gradient updates and the SGD procedure relatively simple, and to make it easier to prove theoretical results about the algorithm's properties. The non-differentiability of the L1 regularization term makes it more difficult to theorize about and correctly perform SGD updates. However, the feature selecting behavior of L1 regularization seems more intuitively appropriate given that logistmar gramression can be thought of as selecting a finite set of features from F_G. So, in the present paper, I modify the original logistmar gramression algorithm to use L1 regularization. As a result of the non-differentiability of the objective given by modification, the weight updates in SGD must be performed in a slightly more complicated way. I describe this complication in more detail below.

Other related work Several existing machine learning paradigms seem intuitively related to the idea of grammatically constructing features and compositions of features within a supervised learning model. The precise relation-

[2] Although the algorithm actually maximizes the objective using gradient *ascent*, I refer to it as stochastic gradient *descent* to respect conventions.

[3] In order to ensure that the logistmar gramression training algorithm converges and maximizes a sensible objective, the algorithm has to be a bit more complicated the one described here. In particular, it's necessary to ensure that a new feature resulting from applying the heuristics is only considered to have non-zero weight as long as its ancestors in the heuristic search space also have non-zero weight. This requirement changes the form of the logistic regression quite a bit, and the gradient updates become rather complicated. However, in practice, I think that this additional complexity is likely to be unnecessary and counter-productive, and it's better to consider the simpler algorithm described here. See my project proposal for details about the more complicated form of the objective used by the original form of logistmar gramression and its properties. I skip the details of this objective in this paper since they were not particularly relevant to my newer alternative version of the model or its implementation.

ships between these approaches remains blurry, but below I give some vague suggestions.

First, deep neural networks models that solve supervised learning tasks can be thought of as constructing complex features from simpler features in the input vector (LeCun and Ranzato 2013). This feature construction can be thought of as similar to the feature construction carried out by a feature grammar model in the application of the heuristic rules the search of the feature space. Logistmar gramression might use the heuristics in H to create logically complex features based on simpler features in F_0, and these features might reflect the features learned by a deep neural network.

Feature grammar models also bear resemblances to other learning paradigms like decision trees (Quinlan 1986) and inductive logic programming (Muggleton, Otero, and Tamaddoni-Nezhad 1992). These paradigms impose some logical structure over the input features to approximate a function which computes the output class or label. However, these paradigms are typically limited to a propositional or first-order logical formalism for constructing these structures, whereas the feature grammar models described in this paper can be defined independent of any particular formalism for specifying the space of features.

Finally, structure learning for probabilistic graphical models like Bayesian networks and Markov logic networks both seem related to the feature grammar models proposed here (Koller and Friedman 2009; Richardson and Domingos 2006). Traditional feature selection models can be thought of as solving the sub-task of structure learning where the goal is just to construct the Markov blanket for a single variable of a network (Koller and Sahami 1996). This suggests that the feature grammar models in the current paper might be thought of as finding a Markov blanket for some variable when the graphical model contains a potentially infinite number of other variables represented by a grammar.[4]

L1-regularized Feature-selecting Model

In my implementation on synthetic data for the current project, I modified the "logistmar gramression" model described in the previous section to use feature-selecting L1-regularization instead of L2-regularization. Since I'm using SGD to approximately maximize the likelihood of the data, and the L1-regularization term is non-differentiable, the training algorithm is a bit more complicated than typical SGD. In particular, naively applying SGD subgradient weight updates will tend to approximately optimize the objective in a way that does not yield the sparse weight vectors desired by feature selection, so a more subtle approach is necessary.

[4]There might be additional similarities to Markov logic networks, but I still haven't had time to try to understand MLNs well enough to try to figure this out. I also discovered papers on "structural logistic regression" over the past couple days, and it seems like it might be similar to what I have been doing (Popescul et al. 2002). I still haven't had time to understand the relationships between all of these different approaches. If I continue work on this in the future, I'll probably need to do a more thorough review of the literature to figure out how "feature grammar" models fit in with other approaches.

To deal with this issue, I take the approach given in (Tsuruoka, Tsujii, and Ananiadou 2009) of smoothing out the effects of fluctuating gradient estimates by considering the cumulative L1 penalty over multiple updates, and clipping the resulting weight updates according to zero when a weight changes sign. Algorithm 1 shows a modified version of this approach adapted from (Tsuruoka, Tsujii, and Ananiadou 2009) that incorporates expansion of the feature vocabulary through the heuristic rules.

Algorithm 1 L1-regularized Logistmar Gramression SGD

function TRAIN-LG-L1(D, $\mathbf{f}_0$, H, t, K, C η_0, α)
  $u \leftarrow 0$
  $\mathbf{w} \leftarrow 0$
  $\mathbf{q} \leftarrow 0$
  $\mathbf{f} \leftarrow \mathbf{f}_0$
  for $k = 0$ to K **do**
    $j \leftarrow k \mod |D|$
    if $j = 0$ **then**
      Shuffle D
      $\mathbf{f}, \mathbf{w}, \mathbf{q} \leftarrow$ APPLYRULES($D, \mathbf{f}, H, t, \mathbf{w}, \mathbf{q}$)
    $\eta \leftarrow \eta_0 \alpha^{k/N}$
    $u \leftarrow u + \eta \frac{C}{|D|}$
    $\mathbf{w}, \mathbf{q} \leftarrow$ UPDATEWEIGHTS($D, j, \mathbf{f}, \eta, u, \mathbf{w}, \mathbf{q}$)
return w, F

function APPLYRULES(D, $\mathbf{f}$, H, t, $\mathbf{w}$, $\mathbf{q}$)
  $\mathbf{f}_{>t} \leftarrow \{f_i \mid w_i > t\}$
  $\mathbf{f}_h \leftarrow \{h(f_{>t,i}) \mid h \in H\}$
  $\mathbf{f} \leftarrow (\mathbf{f}, \mathbf{f}_h)$
  $\mathbf{w} \leftarrow \mathbf{w}$ extended with $|\mathbf{f}_h|$ zeros
  $\mathbf{q} \leftarrow \mathbf{q}$ extended with $|\mathbf{f}_h|$ zeros
return $\mathbf{f}$,$\mathbf{w}$, $\mathbf{q}$

function UPDATEWEIGHTS(D, j, $\mathbf{f}$, η, u, $\mathbf{w}$, $\mathbf{q}$)
  $(\mathbf{x}^j, l^j) \leftarrow D[j]$
  $\mathbf{g} \leftarrow \mathbf{x}^j(l^j - p_{j,\mathbf{w},\mathbf{f}})$
  for $i = 1$ to $|\mathbf{f}|$ **do**
    $w_i \leftarrow w_i + \eta g_i$
    $w_i, q_i \leftarrow$ APPLYPENALTY(w_i, q_i, u)
return $\mathbf{w}$,$\mathbf{q}$

function APPLYPENALTY(w_i, q_i, u)
  $z \leftarrow w_i$
  if $w_i > 0$ **then**
    $w_i \leftarrow \max(0, w_i - (u + q_i))$
  else if $w_i < 0$ **then**
    $w_i \leftarrow \min(0, w_i + (u - q_i))$
  $q_i \leftarrow q_i + (w_i - z)$
return w_i,q_i

The algorithm takes as input the following parameters:

- D: A labeled training data set
- $\mathbf{f}_0$: An initial seed vocabulary of feature functions
- H: A set of heuristic rules for navigating the grammatical feature space

- t: A weight (relevance) threshold above which to apply rules in H to features to construct new features
- K: Number of training iterations
- C: Weight of the l1 regularization term in the objective
- η_0: Initial learning rate
- α: Learning rate exponential decay parameter

The algorithm exponentially decays the learning rate η according to parmaeters η_0 and α, and applies stochastic gradient weight updates estimated by single examples from D using the "UpdateWeights" function. Over successive updates, the u variable keeps track of the total L1 penalty that any weight could have received so far, and $\mathbf{q}$ keeps track of the amount of L1 penalty that weights have actually received. To ensure that the stochastic updates correctly optimize the objective, D is shuffled after each pass.

The heuristic rules in H could be applied after every iteration, but for the sake of efficiency, they are only applied by "ApplyRules" once after each pass over the data. The "ApplyRules" function finds the subset of features in the present vocabulary $\mathbf{f}$ that have weights greater than threshold t, and applies the rules in H. The resulting features are appended to $\mathbf{f}$, and the weight vector $\mathbf{w}$ and cumulative L1 penalty vector $\mathbf{q}$ are extended to contain dimensions corresponding to the new features. Note that the pseudocode in Algorithm 1 is written as though a rule in H can only apply to a single single feature, but the actual implementation also allows for rules that construct features from pairs of features.

Some observations about the L1 model

This L1 regularized version of logistmar gramression has some intuitive qualitative benefits over the original L2 regularized version described in the previous section. Namely, the L2 model would likely assign a small weight to irrelevant features mistakenly added to the vocabulary by the heuristics. In contrast, the L1 model will tend to send the irrelevant added feature weights to zero, allowing these features to be removed from the vocabulary in the final model. Intuitively, this means that the rules will allow the model to search the feature space, and the L1 regularization will cleanly prune irrelevant search trajectories. Somewhat interestingly, this behavior loosely mirrors the actor critic metaphor employed by some reinforcement learning algorithms (Konda and Tsitsiklis 1999)—the likelihood gradient updates along with the rule applications *act* to search the feature space, and the regularization *criticizes* the search trajectories.

On the other hand, this L1 regularized version has some theoretical downsides, but these downsides may not matter in practice. Whereas the original L2 regularized version of logistmar gramression was guaranteed to converge precisely to a known objective, this L1 feature may not have a well-defined optimization problem which it solves. In fact, modulo the decaying learning rate, the search performed by the heuristic rules might proceed down an infinite path of increasingly relevant features. In such a case, the regularizer might send features early in the search to zero, allowing the increasing relevance of later discovered features to guide the search endlessly away from $\mathbf{f}_0$. However, in practice, it might typically be the case that there is a closed boundary on the set of relevant features surrounding $\mathbf{f}_0$, which would eliminate the possibility of diverging infinite search trajectories. In this case the algorithm might be converging to an objective that it is possible to describe precisely.

Implementation and Experiment

In this section, I describe my implementation of the L1 regularized logistmar gramression model described above, and a simple experiment to test whether it behaves reasonably on a toy example task involving synthetic data. To construct the synthetic data, I computed labels on randomly generated relational structures using a ground-truth log linear model with weighted first order logic features. My implementations of L1 regularized logistic regression and logistmar gramression models were both able to approximately recover the ground-truth model feature weights given the synthetic training data. A more thorough description of the task and models is given below.

Task and Data

For my experiment, the supervised learning task is to recover the ground truth model weights $\mathbf{w}_{true}$ used within conditional distributions $P(L \mid x, \mathbf{w}_{true})$ to construct a labeled data set $D \subseteq X \times \{0, 1\}$ where instance X consists of random relational structures. In the construction of X, my implementation allows for specifying a finite domain $\mathbf{Dom}$, a set of properties $\mathbf{Prop}$, and a set of binary relations $\mathbf{Bin}$. Given this specification, each relational structure in X is constructed by assigning each object in $\mathbf{Dom}$ a property in $\mathbf{Prop}$ with probability 0.5, and each pair of objects from $\mathbf{Dom}$ relation from $\mathbf{Bin}$ with probabililty 0.5. A label is assigned to each random $x \in X$ drawn randomly from the ground truth model distribution $P(L \mid x, \mathbf{w}_{true})$.

Given the choice of data instances X, it makes sense to consider a grammatical feature space consisting of closed formulas from a fragment of first-order logic. Specifically, given that we have constant symbols $\mathbf{Con}$ for each element of $\mathbf{Dom}$ and variables $\mathbf{Var} = \{x_1, x_2, \ldots\}$ my implementation allows for features determined by closed formulas in the language $\mathcal{L}$ specified by the following grammatical rules:

- For each constant or variable $t \in \mathbf{Con} \cup \mathbf{Var}$ and each $P \in \mathbf{Prop}$, $P(t) \in \mathcal{L}$.
- For constants or variables $t_1, t_2 \in \mathbf{Con} \cup \mathbf{Var}$ and each for each $R \in \mathbf{Bin}$, $R(t_1, t_2) \in \mathcal{L}$.
- If $\phi, \psi \in \mathcal{L}$, then $\phi \vee \psi \in \mathcal{L}$ and $\phi \wedge \psi \in \mathcal{L}$.
- If $\phi \in \mathcal{L}$ then $\neg\phi \in \mathcal{L}$.
- If $x \in \mathbf{Var}$ and $\phi \in \mathcal{L}$, then $\forall x\phi \in \mathcal{L}$ and $\exists x\phi \in \mathcal{L}$
- $\top, \bot \in \mathcal{L}$

To parse and evaluate features corresponding to formulas in $\mathcal{L}$ on relational structures in X, I use the first-order logic module from NLTK (Bird 2006).

My implementation allows for the easy construction of feature sets consisting of features computed by closed forms in $\mathcal{L}$ by the specification of open form feature *types*. Given an open form formula, the implementation will construct all

closed form features in $\mathcal{L}$ corresponding to substitutions of constants for its free variables. This allows for easy construction of ground truth models over large complex feature sets.

In spite of the flexibility of my implementation, I only had time to experiment with a data set where $\mathbf{Dom} = \{0, 1, 2\}$, $\mathbf{Prop} = \{P_0, P_1, P_2\}$, and $\mathbf{Bin}$ is empty. X is randomly constructed from $\mathbf{Prop}$ and $\mathbf{Dom}$, and labels are sampled from a ground truth model with parameters $w_{true,\top} = -1$ and $w_{true,i} = \frac{2^i}{10}$ specifying the distribution:

$$P(L \mid x, \mathbf{w}_{true}) \propto e^{w_{true,\top}[\![\top]\!](x) + \sum_{i=0}^{4} w_{true,i}[\![P_i(0)]\!](x)} \tag{1}$$

This ground-truth model was chosen as a simple experimental test case for the learning model implementations described below.

Learning Models

I implemented the SGD approximation of L1 regularized logistic regression described in (Tsuruoka, Tsujii, and Ananiadou 2009), and the logistmar gramression modification shown in Algorithm 1. For the learning task specified above, the logistic regression model is given feature vocabulary $\mathbf{f} = \{\top\} \cup \{P_i(j) \mid 0 \leq i \leq 4, 0 \leq j \leq 2\}$, and logistmar gramression is given the seed feature vocabulary $\mathbf{f}_0 = \{\top\} \cup \{P_0(j) \mid 0 \leq j \leq 2\}$.[5]

The implementation allows for the heuristic rules given to logistmar gramression to match closed form features against an open form expression, and if they match, produce a new set of closed form features output by some function applied to the input features. In addition, the implementation allows for heuristic rules that match a pair of open forms against a pair of closed form features, and produce a set of features that is some function of the inputs.[6] However, for the simple experiment considered here, logistmar gramression takes a simple set of heuristics $H = \{P_i(x) \rightarrow P_{i+1}(x) \mid 0 \leq i \leq 3\}$. Each of these heuristics matches input features against an open formula $P_i(x)$, and upon matching, produces a feature $P_{i+1}(x)$ with x assigned to the same constant as in the input feature (so for example, one of the rules in H could take $P_2(1)$, and produce $P_3(1)$.

The idea for the experiment is to confirm the expectation that both L1 regularized logistic regression and logistmar gramression are able to recover the ground truth model described above. Most of the features in the vocabulary $\mathbf{f}$ given to logistic regression are irrelevant to the ground truth labeling, but the L1 regularization should send the weights of these irrelevant features to zero. The seed vocabulary $\mathbf{f}_0$ given to logistmar gramression contains two irrelevant features $P_0(1)$ and $P_0(2)$ and one relevant feature $P_0(0)$ with a low weight in the ground-truth model. Since the weights are increasing from $P_i(0)$ to $P_{i+1}(0)$ in the ground truth, it seems likely that logistmar gramression should apply the $P_i(x) \rightarrow P_{i+1}(x)$ rules to discover that all $P_i(0)$ features are relevant while using the L1 regularization to ignore the other irrelevant features.

[5]The $\top$ feature acts as a bias term, and so regularization is not applied to it in either of the learning models.

[6]It was only for the sake of simplicity that the earlier sections of this paper talked about the heuristic rules as applying to one feature and returning one feature. It's a straightforward modification to allow them to apply to multiple features and return multiple features.

$\mathbf{f}$	$\mathbf{w}_{true}$	$\hat{\mathbf{w}}_{LR}$	$\hat{\mathbf{w}}_{LG}$
$\top$	-1.00	-0.86	-1.00
$P_0(0)$	0.10	0.04	0.10
$P_0(1)$		0.00	0.00
$P_0(2)$		0.00	0.00
$P_1(0)$	0.20	0.22	0.28
$P_1(1)$		0.00	
$P_1(2)$		0.00	
$P_2(0)$	0.40	0.35	0.41
$P_2(1)$		0.00	
$P_2(2)$		0.00	
$P_3(0)$	0.80	0.65	0.71
$P_3(1)$		0.00	
$P_3(2)$		0.00	
$P_4(0)$	1.60	1.56	1.63
$P_4(1)$		0.00	
$P_4(2)$		0.00	

Table 1: Grouth truth weight vector $\mathbf{w}_{true}$ along with the final trained weight vectors $\hat{\mathbf{w}}_{LR}$ and $\hat{\mathbf{w}}_{LG}$ from logistic regression and logistmar gramression after training for $140,000$ iterations. The missing values in logistmar gramression and the ground-truth model indicate that the models did not use those features.

Results

After using the ground truth model to generate 3000 random training examples in D, I ran both logistmar gramression and logistic regression with initial weights $\mathbf{w}$ set to 0. I set the regularization constant $C = 16.0$ for logistic regression and $C = 8.0$ for logistmar gramression. For both models, the learning rate parameters were set to $\eta_0 = 1.0$ and $\alpha = 0.8$. The weight heuristic threshold for logistmar gramression was set to $t = 0.04$. I ran both models for $140,000$ iterations.

Both learning models successfully recover an approximation of the ground truth model from the data. Table 1 shows that the final feature weights for both models approximate the feature weights of the ground truth model. Logistic regression correctly sends all irrelevant features to zero, but slightly underestimates all of the relevant feature weights. The low weight estimates are likely due to C being set slightly too high. Logistmar gramression correctly sends $P_0(1)$ and $P_0(2)$ to zero, and runs heuristic rules to add the remaining relevant $P_i(0)$ features to the vocabulary. The weight threshold t appears to have been set high enough that the rules never misfire on $P_0(1)$ or $P_0(2)$ during the course of training.

Figure 1 shows log likelihood over successive iterations for both algorithms. Both algorithms noisily increase their log-likelihood until they converge after about $50,000$ iterations. Figure 2 shows how both algorithms increase the total weight magnitude rapidly and then jitter it up and down

before stabilizing. In Figure 3, both algorithms gradually increase the number of features with non-zero weight.

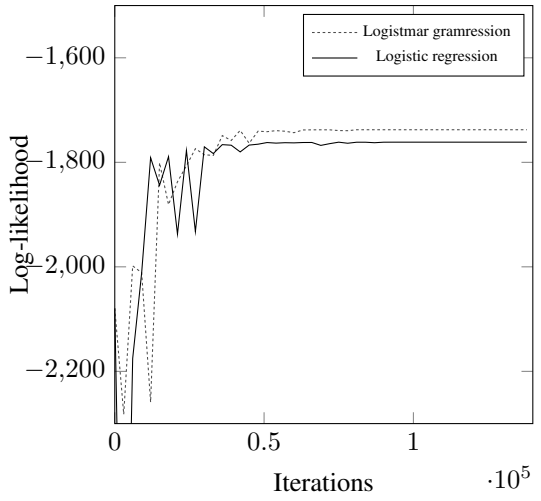

Figure 1: Log-likelihood for the two feature selection algorithms over successive iterations. Note that these are not directly comparable since each algorithm used different values for the L1 regularization weight hyper-parameter C.

The results show that the algorithms behave as we might expect in recovering the ground truth model. The figures also suggest that both algorithms converge in a qualitatively similar way, possibly taking similar trajectories through weight space in spite of the fact that they are initialized with different feature vocabularies. This similarity is likely an artifact of the simplicity of the experiment, and the fact that logistmar gramression did not mistakenly apply rules to any of the irrelevant features.

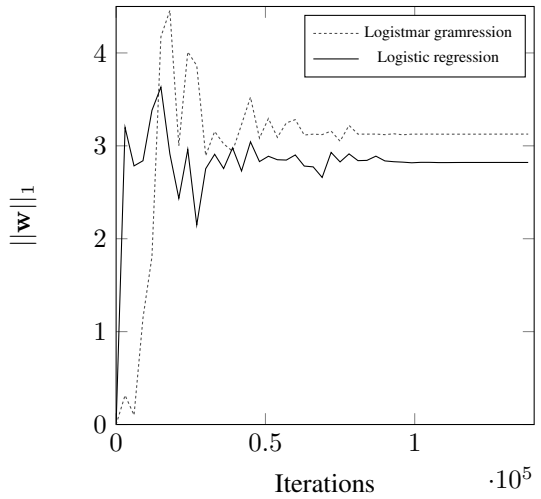

Figure 2: $L1$ norm of the weights for the two feature selection algorithms over successive iterations.

Discussion

The results from the experiment described here are not particularly surprising, but they confirm that logistmar gramression behaves in a reasonable way on a simple artificial data set. However, it is interesting that logistic regression and logistmar gramression converge in a qualitatively similar way (as shown in the plots), but that they require different values for the regularization constant C. In the future, it would be interesting to characterize how the value of C interacts with the weight threshold t and the seed vocabulary $\mathbf{f}_0$, and how this interaction relates to the behavior of standard logistic regression running with the same C value.

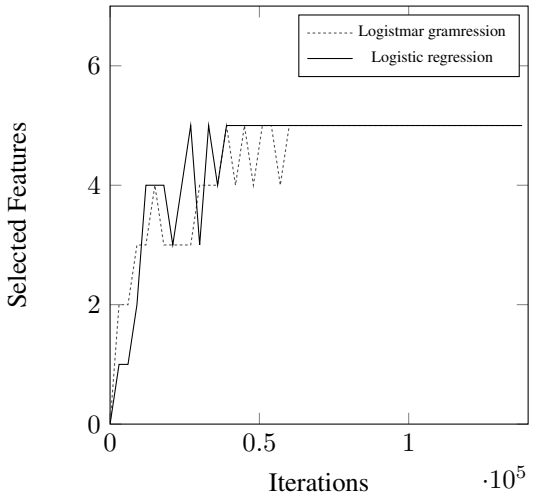

Figure 3: Number of non-zero weights for the two feature selection algorithms over successive iterations.

Although the experiment performed above was extremely simple, the framework that I have implemented allows for easily specifying models that contain more complex first-order logic features and heuristics for exploring them.[7] In the future, I would like to try some logistmar gramression experiments in this framework that require more complicated rules for introducing and eliminating various logical connectives between features.

In addition to the implementation described above, I spent some time during this project thinking of ideas for alternative models that operate under weaker assumptions than logistmar gramression about the heuristics in H, and I would like to implement some of these models in the framework that I have setup. In the following sections, I describe some of these alternative feature grammar models, and also give some suggestions for theoretical properties of the models that might be worth thinking about.

Alternative Models There are several possible ways to weaken the assumptions required for logistmar gramres-

[7] For example, there is some code at `https://github.com/forkunited/grampy/blob/master/test_fol_rules.py` for testing heuristics that introduce quantifiers and construct conjunctions and disjunctions over arbitrary pairs of features.

sion to discover relevant features in the grammatical feature space. In particular, it might be possible to weaken the assumption that we can easily hand-construct reliable heuristic rules that say that if a feature of some type is relevant, then a feature of some other type will be relevant. I have been considering the following directions for weakening this assumption:

1. **Learn the heuristics from data.** Along this direction, it's possible to treat the usefulness of various heuristic rules as latent variables, and use some kind of EM-style algorithm to estimate distributions over these latent variables (in the E-step), and then repeatedly retrain logistmar gramression under different weighted instantiations of these latent rules (in the M-step).
2. **Allow a model to use the heuristics under weaker assumptions.** In this direction, a model might treat the heuristics as encoding weaker distributional assumptions about the weights rather than using them as hard constraints on the feature search. In particular, it's possible to think of the heuristics as representing edges in an infinitely large graph over features, and think of possible weight distributions over the nodes in this graph. Given this graph, an edge (f_1, f_2) from feature f_1 and f_2 might represent some conditional independence assumptions about f_1 and f_2 with respect to the rest of the graph (so that the graph represents the Bayes net over feature weights if the graph is a tree). Related to this idea, it might be possible to model the distribution of weights of the features in this infinitely large graph using some kind of nonparametric Bayesian method.

Theoretical Directions

If I continue working on this project in the future, it would be my first priority to understand whether the connections between the logistmar gramression algorithm and the related work mentioned above on inductive logic programming, structural logistic regression, and learning structures for MLNs. There is some chance that the work in this paper is reinventing something that already exists in those topics, and it would be good to understand if this is the case before proceeding.

In addition to understanding the relationship between this work and other learning paradigms, there are several theoretical properties of logistmar gramression that it would be worth exploring:

1. **Optimization and Convergence** As mentioned above in the description of the logistmar gramression algorithm, it is difficult to characterize whether logistmar gramression is optimizing a well-defined objective. In theory, there are cases where the heuristics guide the model down an endless path of increasingly relevant features, and so the algorithm will not converge. However, in practice, we might be able to make an assumption that the relevant features are all within a finite distance from the $\mathbf{f}_0$. Under this condition, it might be possible to articulate an objective that L1 regularized logistmar gramression is optimizing, but this objective will depend on the initialization of $\mathbf{w}$. Intuitively, it should be possible to write some proofs about this similar to the ones given about the L2 regularized version of logistmar gramression described in my project proposal.
2. **Search and Retraining** Under some conditions on C, $\mathbf{f}_0$, and H, it should be possible to prove that logistmar gramression selects all relevant features in the grammatical feature space, and that it sends all irrelevant features to 0. I would like to characterize these conditions. Related to this, I've been considering the implications of repeatedly retraining logistmar gramression while initializing its seed vocabulary to the set of features selected on the previous iteration. Under the assumption that the learning rate decays to zero, this retaining procedure might be able to discover more relevant features than running the algorithm a just once. It would be interesting to precisely describe when retraining might be necessary given some learning rate parameters.

Conclusion

In the previous sections, I describe an L1 regularized "feature selecting" version of the logistmar gramression algorithm that I had previously developed, and I implement the algorithm to run in experiments on synthetic data in a first-order logic feature space. The work for this project allowed me to ensure that the model behaves sensibly on a simple toy example. In developing the model, I clarified my sense of the appropriate type of regularization in this context (L1 seems intuitively more appropriate than L2), and I gained some sense that the L1 approximate version of the model described here might work well in practice in spite of the fact that it might diverge and fail to optimize a well-specified objective. I also realized more possible connections to related machine learning paradigms. If I continue this work in the future, I would like to clarify these connections. I would also like to test logistmar gramression against more complicated ground truth models and in realistic domains.

Acknowledgements

Tom Mitchell at CMU suggested the "feature grammar" idea to me when I was working for him a few years ago, and so he is responsible for inspiring the overarching theme of this work. He also gave advice and suggestions when I was first developing the ideas for logistmar gramression.

References

[Bird 2006] Bird, S. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, 69–72. Association for Computational Linguistics.

[Guyon and Elisseeff 2003] Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3(Mar):1157–1182.

[Koller and Friedman 2009] Koller, D., and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[Koller and Sahami 1996] Koller, D., and Sahami, M. 1996. Toward optimal feature selection. Technical report, Stanford InfoLab.

[Konda and Tsitsiklis 1999] Konda, V. R., and Tsitsiklis, J. N. 1999. Actor-critic algorithms. In *NIPS*, volume 13, 1008–1014.

[LeCun and Ranzato 2013] LeCun, Y., and Ranzato, M. 2013. Deep learning tutorial. In *Tutorials in International Conference on Machine Learning (ICML13)*. Citeseer.

[Mitchell 1997] Mitchell, T. M. 1997. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill* 45(37):870–877.

[Muggleton, Otero, and Tamaddoni-Nezhad 1992] Muggleton, S.; Otero, R.; and Tamaddoni-Nezhad, A. 1992. *Inductive logic programming*, volume 38. Springer.

[Ng 2000] Ng, A. 2000. Cs229 lecture notes. *CS229 Lecture notes* 1(1):1–3.

[Ng 2004] Ng, A. Y. 2004. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, 78. ACM.

[Popescul et al. 2002] Popescul, A.; Ungar, L. H.; Lawrence, S.; and Pennock, D. M. 2002. Towards structural logistic regression: Combining relational and statistical learning.

[Quinlan 1986] Quinlan, J. R. 1986. Induction of decision trees. *Machine learning* 1(1):81–106.

[Richardson and Domingos 2006] Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1-2):107–136.

[Tsuruoka, Tsujii, and Ananiadou 2009] Tsuruoka, Y.; Tsujii, J.; and Ananiadou, S. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, 477–485. Association for Computational Linguistics.