ML Reading Group (Fall 2015) **Lecture: 1**

Bringing machine learning and compositional semantics together

Lecturer: Jonathon Cai

1 Today

– Reading Group Mechanics (MIT/Stanford)

– Paper discussion

– Sempre code tutorial

– Example of linear algebra / theory (next time)

2 Quick Summary

1-Line Summary:

Authors describe toy examples and basic ML concepts (stochastic gradient descent [**SGD**], SVM, neural networks) in order to show how grammatical rules can be learned from data, bringing together logic and statistics.

Strengths:

1. Logic/stats not coherently united before; these mechanisms make it so that they are.

2. Very interesting application of ML to useful domain space

Weaknesses:

1. Can be difficult to implement things like "scope"

2. Logical forms can be tedious to annotate and find – i.e. they are not readily available – and hence denotations are preferred as a learning substrate.

3. Computational complexity of denotations is high.

Comments:

1. The lexical space takes up most of the grammar; in actuality, there is a small number of compositional rules. This likely reflects natural language.

3 Major Ideas

1. The context-free grammar and the linguistic triplet, consisting of utterance, semantic representation, and denotation, are ubiquitous concepts in computer science (Table 2).

2. Table 3 is a good example of the trade-offs we make when engineering features. A feature can be too coarse or too fine.

3. Figure 2 is probably the most important element in this paper. It gives us an explicit instantiation of why SGD (Figure 1) works.

4. The SVM latent objective function (Equation 4) for denotations is a slight modification of the SGD objective function for the logical form.

5. The distributed representation represents objects as matrices and vectors, which permits shades of meaning. A neural network and its corresponding parameters are described; in order to learn these parameters, one performs stochastic gradient descent via a cross-entropy objective function.

4 Formalism

The most important theoretical ingredient in this paper is SGD (Figure 1). It should be noted that this is not the most traditional formulation of SGD – the most common form involves a gradient symbol, while this form of SGD does not. It is not obvious to me why $\phi(x, y) - \phi(x, \tilde{y})$ should push the weight vector in the direction of the correct y. As the feature vector is pre-defined, conceivably, strange feature vectors could result in the weight vector being pushed in the "wrong" direction. I defer a formal proof of correctness to a reference in this paper (Bottou 2010). However, the explicit demonstration of the algorithm on a very small toy example in Figure 2 shows that the algorithm does push the weight vector in the direction of favoring the "correct" output from the training set. Going through this example carefully is profitable.

In this paper, the SVM and neural network objective functions are slight modifications of the original SGD optimization problem. The original SGD objective function for learning logical forms was:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{(x,y)\in D} \max_{y'\in Y}[\text{Score}_{\mathbf{w}}(x, y') + c(y, y')] - \text{Score}_{\mathbf{w}}(x, y)$$

where D represents the training set and Y the space of possible outputs. $\mathbf{w}$ represents the weight vector, which represents a linear combination of feature vector elements, used to predict logical forms and denotations, respectively. The weight vector $\mathbf{w}$ is what we seek to learn and hence this is why it is at the outermost part of the objective function. The weight vector should describe our training set somewhat accurately (that is to say, if we run a training example x through the weight vector, it should often output y).

It is important to understand the following sentences from this paper: "To understand the loss on a single (x, y) pair, first note that the loss is zero when the score of the correct output ($\text{Score}_{\mathbf{w}}(x, y)$) exceeds that of any incorrect output ($\text{Score}_{\mathbf{w}}(x, y')$ for $y' \neq y$) by at least $c(y, y') = 1$. Otherwise, we pay linearly in the amount by which the former fall shorts of the latter." As we seek to minimize the loss function, this gives us an intuitive idea of why solving this optimization problem does indeed push us in the direction of favoring correct outputs in our final weight vector.

The SVM objective function, which I will not list here, was used to learn denotations. It is quite similar to the logical form objective function. Logical forms are implicitly used in the SVM objective function, via the summation index $y' \in \text{GEN}(x)$.

Distributed representations are lightly touched upon in this paper – the learning algorithms for logical forms and denotations used "atomic" elements. That is to say, a denotation is a singular atomic element, and so is a logical form. In contrast, we may represents objects with more nuanced shades of meaning as matrices and vectors. The authors use a neural network to describe the function they learn. A neural network is a fairly intuitive object – it's essentially a layered system of functions. The algorithm used to learn the neural network parameters is really stochastic gradient descent multiple times, on a cross-entropy objective function, and also involves **backpropagation**.

I defer formal proofs of bounds and correctness, regarding SVM and neural network (equations 3 and 4), to references in the paper.

5 Sempre

I roughly went through the first three sections of the Sempre tutorial (on github), a semantic parser developed by Percy Liang's group. I demonstrated how SGD can be used to probabilistically learn a basic rule given a training example.

Relating to computer programming, a logical form may be considered the "program", the denotation a "return value".