Conditional Random Fields for Punctuation Prediction
Learning Algorithms, Project 2

Mohsen Malmir, Erfan Sayyari

February 12, 2014

1 Abstract

2 Introduction

In this paper, we provide the results and details of implementation of a Conditional Random Field (CRF) model for predicting English language text punctuations. CRFs are a variant of undirected graphical models that are well suited for predicting structured labels. We train our model by maximizing log conditional likelihood (LCL) of the training data. During our experiments, we found that the model could predict individual tags by more than 94% accuracy. However, as we performed more and more experiments, we discovered the over fitting to some prevalent tags in the training set. Using several experiments and methods that will be described later in this paper, we could overcome the over-fitting problem to some degrees.

We choose two techniques to train the proposed model: Collin's Perceptron and Contrastive Divergence. Both of these methods are approximations to the general gradient following method. We choose these two methods because of their two characteristics: simplicity and elegance. Both Collin's Perceptron and Contrastive Divergence provides simplified updates rule for the parameters, which are less expensive compared to the general gradient following for maximizing LCL. Despite simplicity, both methods have very clear and intuitive explanation behind them: they maximize the LCL of the data by throwing spurious samples at the model. However, as we explain our experiments in more details, we do not stick to the basic algorithms and tweak them in several ways to improve their performance.

One of the most important parts of the CRF model for predicting text punctuation is the choice of feature functions as it effects both the performance and accuracy of the model. The feature functions we develop here use Part-of-Speech (POS) tags for the input sentence. The main characteristic for

the proposed feature functions is their efficiency in computation time: training of our method only takes a few minutes on a single core machine. We show that using these feature functions, the model can predict the punctuation tags with high accuracy.
For the given training and test data, danger of over-fitting is high because of the imbalanced distribution of punctuation tags. We deal with this problem with different techniques. We show how bounding the weight parameters and early stopping can improve the accuracy for the Collin's Perceptron. By introducing random sampling and guided sampling, we achieve similar improvements in Contrastive Divergence. Then we introduce the Turn-taking train procedure: for each tag we train a different predictor and then combine them using .... We show this last technique is much more effective and achieves the highest accuracy for punctuation prediction for the given data set.

3 Design and Analysis of Algorithm

3.1 The general log-linear model

3.2 Feature functions

3.2.1 Part of Speech tagging

3.3 Conditional random fields

3.4 The Collins perceptron

3.5 Contrastive divergence

3.5.1 Gibbs sampling

4 Design of Experiments

To test the proposed model, we perform several experiments on the given dataset, which is an English language punctuation dataset that consists of 70115 training and 28027 test sentences. We use only training data to fit the model, one third for validation and two thirds for training. After training is complete, the test data is used to assess prediction and generalization of the model. Each sentence in the training data consists of a set of words without any punctuation symbols. For multiple sentences, there was a few form of syntax complications such as use of '$' in place of the word 'money', or words that were split to two parts by space *****. We ignored all such cases, as they were few sentences and finding a complete list of them required going through the entire dataset which was very time

consuming.

4.1 Initialization

For Collin's Perceptron, as pointed out in the lecture, the learning rate can be fixed to $\lambda = 1$. This is because scaling scaling the weight vector w does not affect the predicted label by the model. Using different learning rates leads to different weight vectors that are scaled versions of each other. However, one should note that this is correct only if the model converges to the global optimum ******.
For both experiments on Collin's Perceptron and Contrastive Divergence, we fix $\lambda \leftarrow 1$.
For initializing the weight vector w, we use small random Normal numbers from $\mathcal{N}(0., 1e-5)$. This is because as we describe later, limiting the weight vector entries w_j to be in a small range $[-a, a]$ helps to prevent over-fitting. Because both a and $-a$ represents the extreme learned situation, a good option for initial weights is a random vector that is not biased for specific feature functions, that is w_j better be close to 0 for all j.

4.2 Preprocessing

4.3 Performance Measure

We report the performance of the models in two different ways. First, we report the prediction of individual target tags, That is, how accurate the model can predict the individual punctuations tags in corpus sentences. We report the accuracy of prediction for the entire tags as wells as individual target tags. The benefit of looking into individual punctuation tag prediction is that it reveals if over-fitting has happened due to imbalanced distribution of tags in the training data. Next, we report the accuracy of model for predicting sentence punctuation. The accuracy for this case should be lower because each sentence punctuation is composed of multiple tags.

5 Results of Experiments

5.1 Collins perceptron

For training Collin's Perceptron, we divide the training data into two thirds for train and one third for validation. The initial value for the weight vector entries w_j is chosen to be small random numbers from $\mathcal{N}(0, 1e-5)$. We use early stopping as a criteria for terminating the training procedure. Because Collin's Perceptron has no hyper parameters, there is a chance of over fitting to the training data. In

the train procedure, we divide the data into train and validation and start training only with train data. After each epoch, we measure the accuracy for predicting individual punctuation tags on the validation data. If the accuracy has decreased, we stoop the training and roll back the weights to previous values.

5.2 Gibbs sampling

6 Findings and Lessons Learned

6.1 Numerical Issues and Preprocessing

6.2 Overfitting

6.3 Model Selection

6.4 Future works