CS 512: Advanced Machine Learning **Due: 12 noon, Feb 29, 2020**

Assignment 1: Graphical Models (Programming Questions)

Student: *Email:*

Wen-Yu Su(wsu23), ChiehHsi Lin(clin220), Jonas Szum(Jszum2), Jonathan Tso(Jtso2)

1 Conditional Random Fields

(1a)

$$p(\mathbf{y}^t|X^t) = \frac{1}{Z_X^t} \exp\left(\sum_{s=1}^{m} \left\langle \mathbf{w}_{y_s}^t, \mathbf{x}_s^t \right\rangle + \sum_{s=1}^{m-1} T_{y_s^t, y_{s+1}^t}\right) \tag{1}$$

From equation 1, we can simplify $\log p(\mathbf{y}^t|X^t)$ as follows:

$$\log p(\mathbf{y}^t|X^t) = -\log Z_X^t + \sum_{s=1}^{m} \left\langle \mathbf{w}_{y_s}^t, \mathbf{x}_s^t \right\rangle + \sum_{s=1}^{m-1} T_{y_s^t, y_{s+1}^t} \tag{2}$$

While taking the derivative of a dot product involving $\mathbf{w}_y$, X_s will only appear when $y_s = y$. And the sum of transitions will disappear because it does not depend on $\mathbf{w}$.

$$\implies \nabla_{\mathbf{w}_y} = \sum_{s=1}^{m} (\llbracket y_s^t = y \rrbracket) \mathbf{x}_s^t - \frac{\partial \log Z_X^t}{\partial \mathbf{w}_y} \tag{3}$$

$$\frac{\partial \log Z_X^t}{\partial \mathbf{w}_y} = \frac{1}{Z_X^t} \sum_{y \in \mathcal{Y}} \exp\left(\sum_{s=1}^{m} \left\langle \mathbf{w}_{y_s}^t \mathbf{x}_s^t, + \right\rangle \sum_{s=1}^{m-1} T_{y_s^t, y_{s+1}^t}\right) \sum_{s=1}^{m} \mathbf{x}_s^t \tag{4}$$

$$= p(y_s = y|\mathbf{x}^t) \sum_{s=1}^{m} \mathbf{x}_s^t \tag{5}$$

$$= \sum_{s=1}^{m} p(y_s = y|\mathbf{x}^t) \mathbf{x}_s^t \tag{6}$$

$$\therefore \frac{\partial}{\partial \mathbf{w}_y} \log p(\mathbf{y}^t|X^t) = \sum_{s=1}^{m} \left(\llbracket y_s^t - y \rrbracket - p(y_s = y)\right) \mathbf{x}_s^t \tag{7}$$

Similarly, for $\nabla_{T_{ij}} \log p(\mathbf{y}|X)$:

$$\frac{\partial}{\partial T_{ij}} \log p(\mathbf{y}^t|X^t) = \llbracket y_s^t = i, y^{s+1} = j|X^t \rrbracket - \frac{\partial \log Z_X^t}{\partial T_{ij}} \tag{8}$$

$$\frac{\partial \log Z_X^t}{\partial T_{ij}} = \sum_{s=1}^{m-1} p(y_s^t = i, y^{s+1} = j | X^t) \tag{9}$$

Using equation 8 and equation 9, we can get

$$\frac{\partial}{\partial T_{ij}} \log p(\mathbf{y}^t | X^t) = \sum_{s=1}^{m-1} \left[[\![y_s^t = i, y^{s+1} = j | X^t]\!] - p(y_s^t = i, y^{s+1} = j | X^t) \right] \tag{10}$$

(1b) If the features that depends on letters' label, it has the form

$$\mathbf{x}_i [\![y_i = j]\!] \quad \text{where} \quad i = 1 \dots m; j = 1 \dots 26 \tag{11}$$

If the features that depends on the transition between two letters, the feature takes the form

$$[\![y_i = k, y_{i+1} = z]\!] \quad \text{where} \quad i = 1 \dots m; k, z = 1 \dots 26 \tag{12}$$

The gradient of $\log Z_X$ are defined in equation 4 and 9. The feature function $\phi(X)$ will put the input in specific position to calculate dot productions and transitions correctly.

Consider features depending on labels, features with respect to $p(y|X)$ will be:

$$\sum_{s=1}^{m} p(y|X)\phi(X) = \sum_{s=1}^{m} p(y|X^t) x_s [\![y_s = y]\!] \tag{13}$$

The feature function will only be non-zero when $y_s = y$, otherwise it goes to zero. Therefore, the $p(y|X)$ will become $p(y_s = y|X)$ via marginalization, and x_s will be selected by feature function. Likewise, for the gradient respect to T, $[\![y_s = i, y_{s+1} = j]\!]$ is the important feature marginalizing $p(y|X)$ to $p(y_s = i, y_{s+1} = j|X)$.

2 Training Conditional Random Fields

$$\min_{\{\mathbf{w}_y\}, T} \; -\frac{C}{n} \sum_{i=1}^{n} \log p(\mathbf{y}^i | X^i) + \frac{1}{2} \sum_{y \in \mathcal{Y}} \|\mathbf{w}_y\|^2 + \frac{1}{2} \sum_{ij} T_{ij}^2. \tag{14}$$

(2a) **Provide** the value of $\frac{1}{n} \sum_{i=1}^{n} \log p(\mathbf{y}^i | X^i)$ for this case in your report.
The value is -28.56971.

(2b) In your report, provide the optimal objective value of (14) found by your solver. The value of optimal objective function is 3815.435975

3 Benchmarking with Other Methods

(3a) SVM-MC was taken from *Scikit-Learn*, where the LinearSVC was utilized. *Scikit-Learn* is an adapted version of 'liblinear', where the minute differences were not of concern in this assignment since the differing parameters were not utilized.

The training and testing data were stripped and split into the alphabet, word number, and features. The features were then used to train the model and create predictions on the testing data set using the *.predict()* function. Letter and word-wise accuracy were obtained and the C regularizer was tested for optimality.

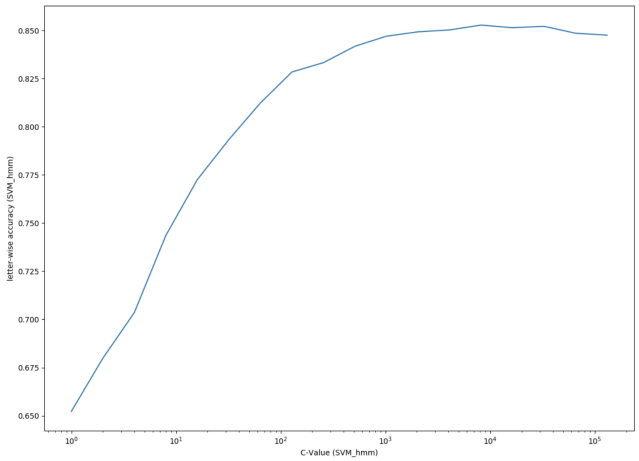

Figure 1. Letter-wise accuracy of SVM HMM: accuracy slowly increased as C increased

SVM-hmm was taken directly from the provided webpage and downloaded. Taking into account the updated notation from the `struct.txt` file, each row in the `train_struct.txt` file was parsed and broken down into the alphabet and `qid` to check if an alphabet character was relevant to a word.

The SVM-hmm learn was run to properly create and train a new model, and classify was run shortly after into a test.outtags file. Each classified row was compared to the truth value and the letter and word-wise accuracy were obtained, and the C regularizer was again tested.

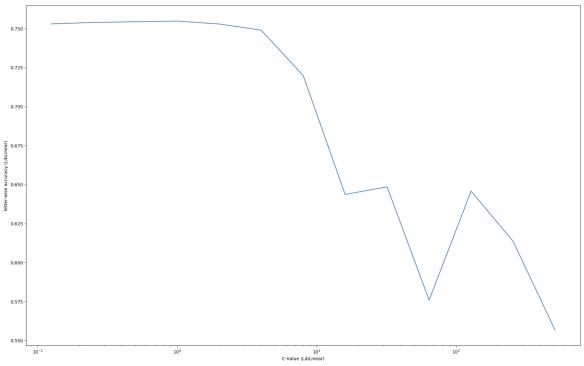

Figure 2. Letter-wise accuracy of LibLinear: accuracy decreased as C increased

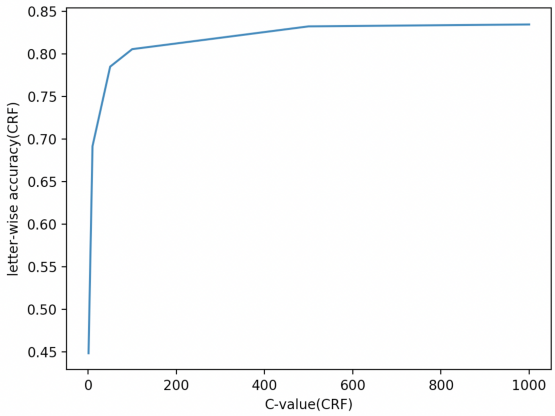

Figure 3. Letter-wise accuracy of CRF: accuracy starts low and increases quickly with C

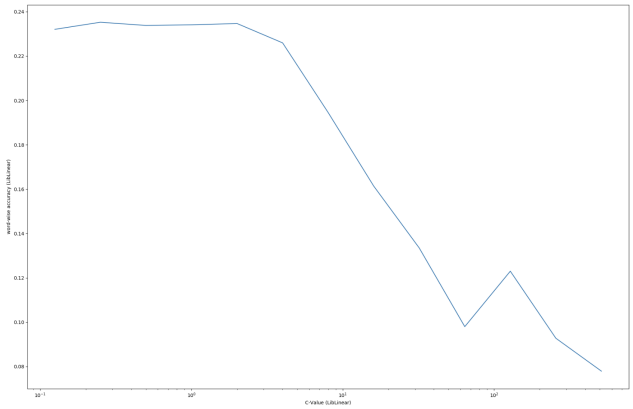

Figure 4. Word-wise accuracy of LibLinear: accuracy starts level then logarithmically decreases with C

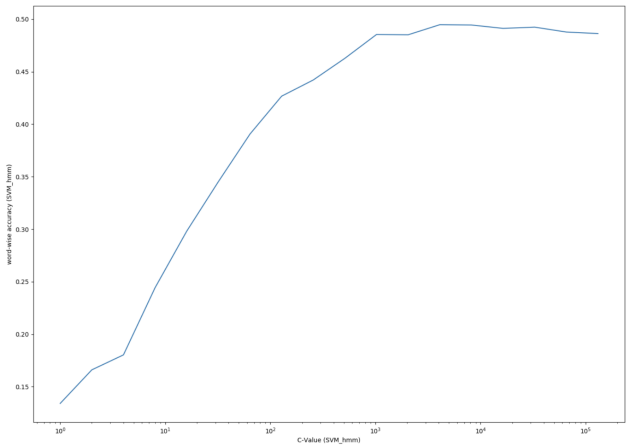

Figure 5. Word-wise accuracy of SVM HMM: accuracy slowly increases with C

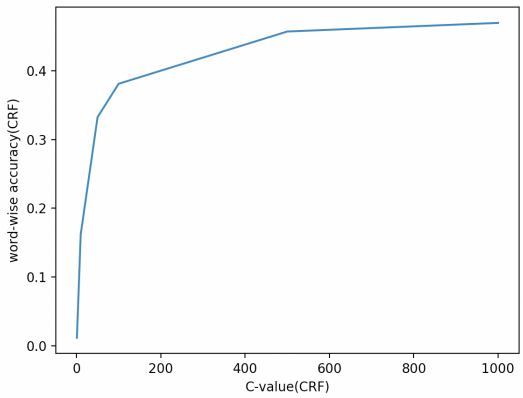

Figure 6. Word-wise accuracy of CRF: accuracy starts low but increases quickly with respect to C

4 Robustness to Distortion

(4a) Plot the following two curves where the y-axis is the letter-wise prediction accuracy on test data. We will apply to the training data the first x lines of transformations specified in `data/transform.txt`. x is varied in $0, 500, 1000, 1500, 2000$ and serves as the value of x-axis.

To properly defend against the forces of evil, tests were conducted against changes to the features set of the training data. Translations and rotations were applied to the training data based on information provided through the transform.txt file. This was then returned as a 2d array of truth values (alphabet letters) and features (updated through transformation).

The new 2d array was then passed into the CRF, SVM-MC and SVM-hmm models. The models used their found optimal C value while updating the amount of transformations done to the training set.

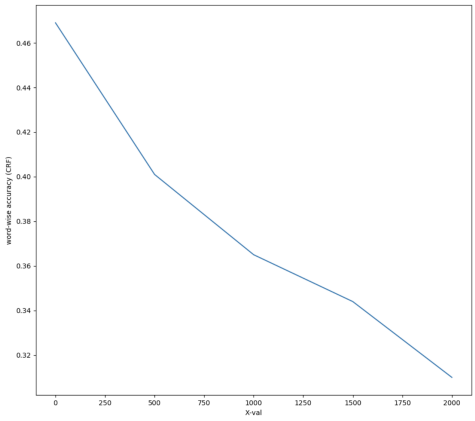

Figure 7. CRF Word-wise accuracy vs distortion

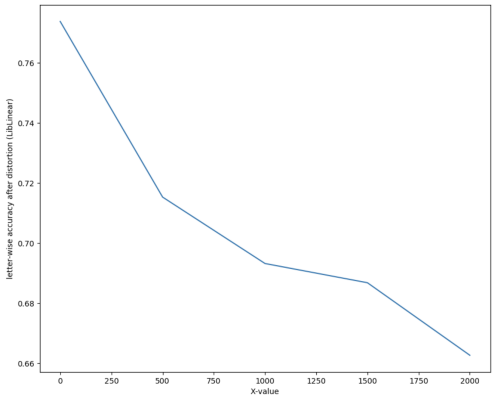

Figure 8. SVM LibLinear Accuracy vs distortion

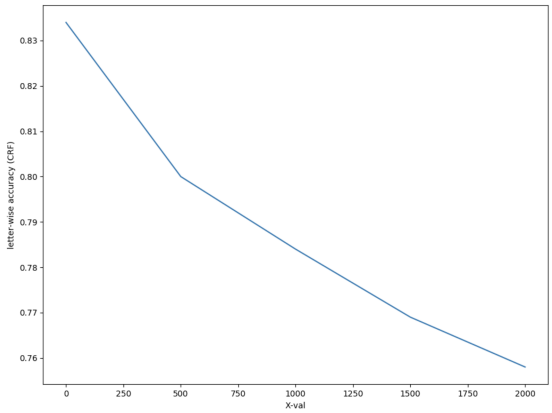

Figure 9. CRF Letter-wise accuracy vs distortion