CS 512: Advanced Machine Learning **Due: 17:00, Mar 11, 2019**

Project 2: Conditional Random Fields for Structured Output Prediction

Group Members:
Omid Memarrast <omemar2@uic.edu>
Sanket Gaurav <sgaura2@uic.edu>
Raj Shekhar <rshekh3@uic.edu>
Sarit Adhikari <sadhik6@uic.edu>

1 Conditional Random Fields

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

$$\text{where} \quad Z_X = \sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{\hat{y}_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{\hat{y}_s, \hat{y}_{s+1}}\right). \tag{2}$$

(1a)

$$\log p(\mathbf{y}|X) = \log \frac{1}{Z_X} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{y_s, y_{s+1}}\right) = \sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{y_s, y_{s+1}} - \log Z_x \tag{3}$$

$$\nabla_{\mathbf{w}_y} \sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{y_s, y_{s+1}} = \sum_{s=1}^{m} [\![y_s = y]\!] x_s^t \tag{4}$$

This is because while taking the derivative of a dot product involving w_y, X_s will only appear whenever $y_s = y$. And, the sum of transitions will disappear because it does not depend on w.

For $\nabla_{\mathbf{w}_y} \log Z_X$ we can use chain rule:

$$\nabla_{\mathbf{w}_y} \log Z_X = \frac{\sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{\hat{y}_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{\hat{y}_s, \hat{y}_{s+1}}\right)}{\sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{\hat{y}_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{\hat{y}_s, \hat{y}_{s+1}}\right)} * \sum_{s=1}^{m} [\![y_s = y]\!] x_s^t \tag{5}$$

Also, we can substitute by substituting $p(y|X)$ into equation 5 and rearranging the sums, in the following way:

$$\sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} p(y|X) \sum_{s=1}^{m} [\![y_s = y]\!] x_s^t = \sum_{s=1}^{m} \sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} p(y|X) [\![y_s = y]\!] x_s^t \tag{6}$$

Finally, we recognize that the inner summation is a marginalization over y except the label we are taking the gradient against. Therefore we can further reduce the equation to:

$$\sum_{s=1}^{m} p(y_s = y|X^t)) x_s^t \tag{7}$$

Using equations 4 and 7, the gradient is

$$\nabla_{\mathbf{w}_y} \log p(\mathbf{y}^t|X^t) = \sum_{s=1}^{m} ([\![y_s^t = y]\!] - p(y_s = y|X^t))\mathbf{x}_s^t, \tag{8}$$

Now computing gradient for T

The $\nabla_{Tij} \sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{y_s, y_{s+1}}$ is the following:

$$\sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] \tag{9}$$

This is because when we differentiate with respect to T_{ij}, the only terms that remain are the ones that specify a transition between i and j. There is only a single weight that lies at this transition point so when we differentiate it, it becomes 1. Also, the dot product goes away because it does not depend on T.

The $\nabla_{T_{ij}} \log Z_X$ is computed via the chain rule in the following way:

$$\nabla_{T_{ij}} \log Z_X = \frac{\sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{\hat{y}_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{\hat{y}_s, \hat{y}_{s+1}} \right)}{\sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{\hat{y}_s}, \mathbf{x}_s \rangle + \sum_{s=1}^{m-1} T_{\hat{y}_s, \hat{y}_{s+1}} \right)} * \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] \tag{10}$$

This can be further reduced, by substituting $p(y|X)$ into equation 10 and rearranging the sums, in the following way:

$$\sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} p(y|X) \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] = \sum_{s=1}^{m-1} \sum_{\hat{\mathbf{y}} \in \mathcal{Y}^m} p(y|X) [\![y_s = i, y_{s+1} = j]\!] \tag{11}$$

Finally, we recognize that the inner summation is a marginalization over y except the label we are taking the gradient against. Therefore we can further reduce the equation to:

$$\sum_{s=1}^{m-1} p(y_s = i, y_{s+1} = j | X^t) \tag{12}$$

Using equations 9 and 11 the gradient is:

$$\sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] - p(y_s = i, y_{s+1} = j | X^t) \tag{13}$$

(1b) If the feature depends on the letter's label and pixel values, then it will take the following form:

$$x_i [\![y_i = j]\!] \tag{14}$$

where $i \in 1...m$ and $j \in 1...26$. Following are the features that capture transition between two consecutive letters,

$$[\![y_i = a, y_i + 1 = b]\!] \tag{15}$$

where $i \in 1...m$ and $a, b \in 1...26$

The gradient of $\log Z_x$ is shown in Equations (5) and (10). The above feature function make sure that only relevant parameters participate in the prediction. For example, the feature function enables the selection of w_4 to be used in $\langle x_i, w_4 \rangle$ when $y_i = 4$.

For the features described in Equation (15), the conditional expectation with respect to $p(\mathbf{y}|X)$ is given by:

$$\sum_{s=1}^{m} p(\mathbf{y}|X)\phi(X) \tag{16}$$

which is,

$$\sum_{s=1}^{m} p(\mathbf{y}|X^t) x_s [\![y_s = y]\!] \tag{17}$$

The indicator function $[\![y_s = y]\!]$ is non-zero for terms $y_s = y$ and the others will be zero. Therefore, the above Equation (17) is equivalent to Equation (5) via marginalization of $p(\mathbf{y}|X)$. Likewise, the feature functions $[\![y_s = i, y_{s+1} = j]\!]$ will marginalize $p(\mathbf{y}|X)$ to $p(y_s = i, y_{s+1} = j)$.

(1c) We implemented the max-sum algorithm using the dynamic programming approach described in the project write-up. The implementation for this part is inference.py. The implementation includes optimized inference using max-sum algorithm and also brute force inference using recursive function. As mentioned in the project, optimized version has the order of complexity of $O(m|Y|^2)$ compared to $O(|Y|^m)$ for brute force inference. Hence, max-sum is much faster and finds the word maximizing object value in a few seconds but brute force is only working for words with 5 or 6 letters in our system.
The max objective value = 199.418

2 Training Conditional Random Fields

Finally, given a training set $\{X^t, \mathbf{y}^t\}_{t=1}^n$ (n words), we can estimate the parameters $\{\mathbf{w}_k : k \in \mathcal{Y}\}$ and T by maximizing the likelihood of the conditional distribution in (1), or equivalently

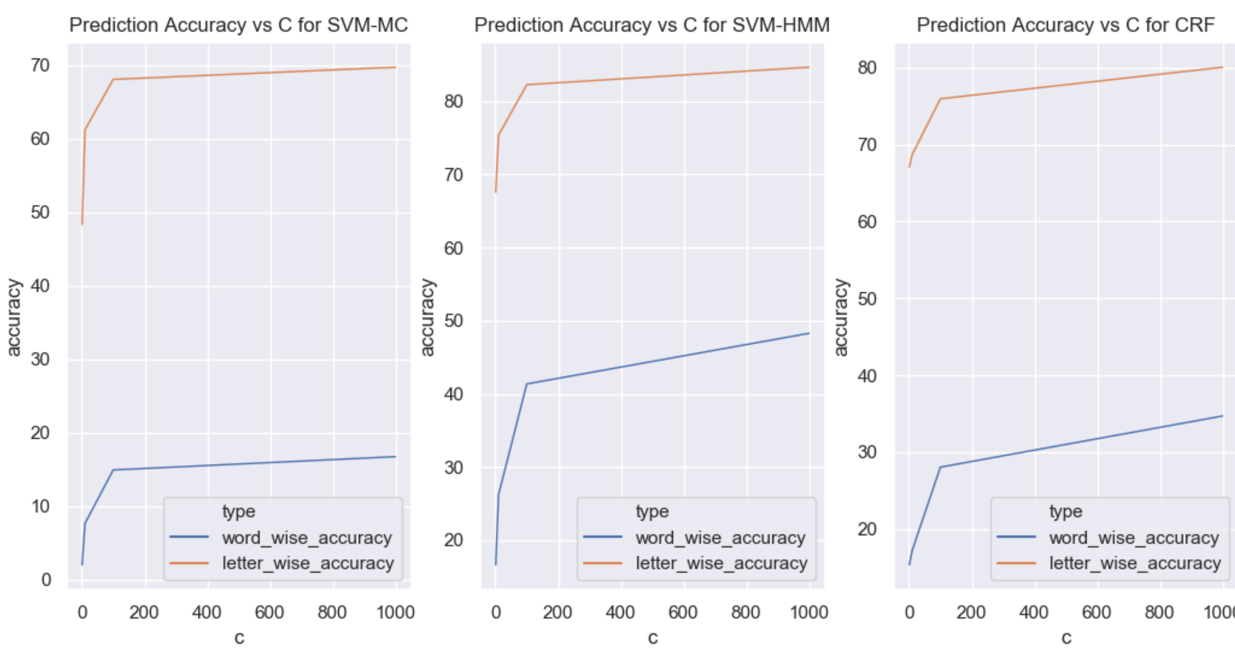
$$\min_{\{\mathbf{w}_y\},T} -\frac{C}{n}\sum_{i=1}^{n}\log p(\mathbf{y}^i|X^i) + \frac{1}{2}\sum_{y\in\mathcal{Y}}\|\mathbf{w}_y\|^2 + \frac{1}{2}\sum_{ij}T_{ij}^2. \tag{18}$$

Here $C > 0$ is a trade-off weight that balances log-likelihood and regularization.

(2a) Average log-likelihood = -31.2884

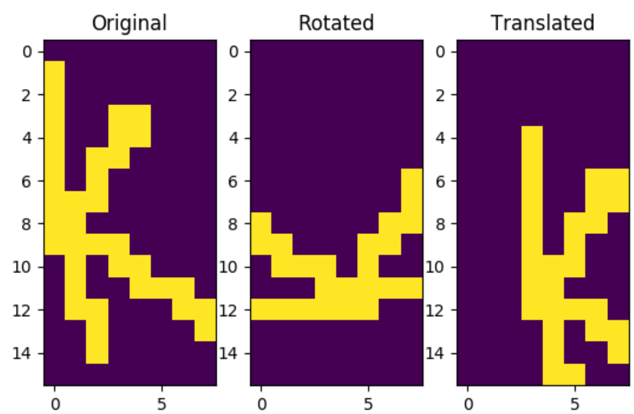
(2b) Optimal Objective Value = 3701.154

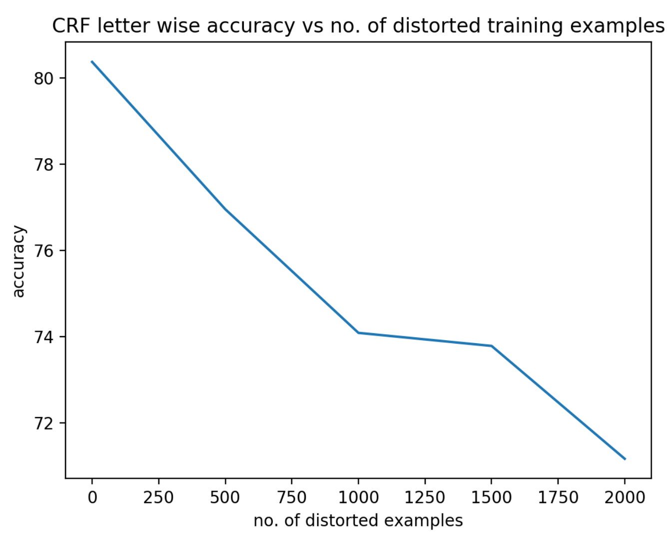
3 Benchmarking with Other Methods



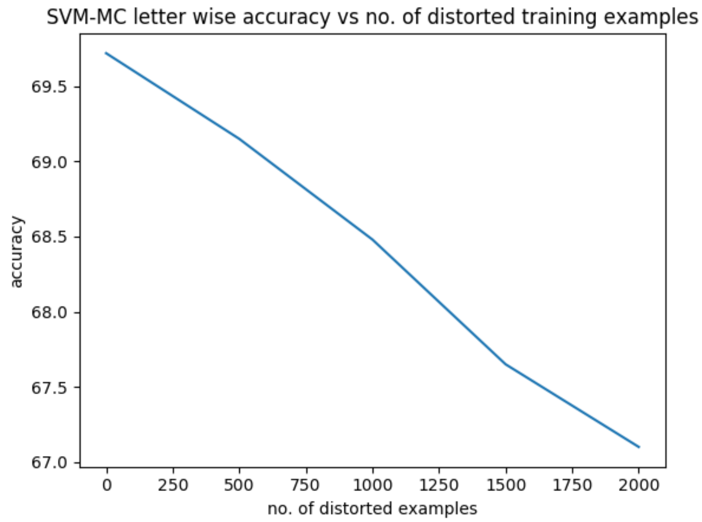
3 shows the change in accuracy as we vary C for three different models SVM-MC, SVM-HMM and CRF. Each graph has two line-plots corresponding to word wise accuracy and letter wise accuracy. The former is lower than the latter becausr for a word to be accurate all the letters in the word should match the label. SVM-MC model has the lower accuracy compared to SVM-HMM and CRF because it doesn't take relation betweeen letters in the word into account. SVM-HMM model slightly outperformed CRF model. In general, as we increase value of C, the accuracy seems to increase.

4 Robustness to Distortion

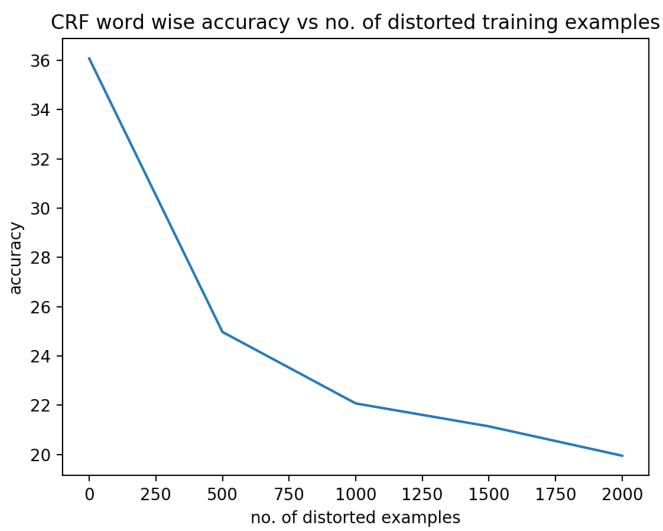




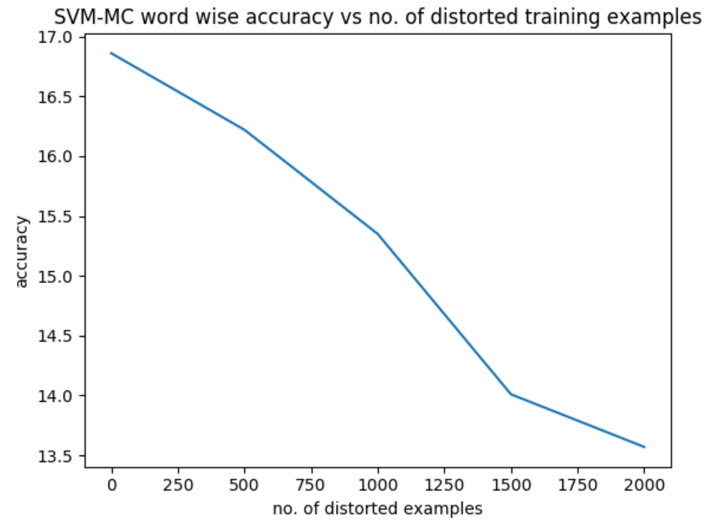
(4a-1)


SVM-MC letter wise accuracy vs no. of distorted training examples
accuracy
69.5
69.0
68.5
68.0
67.5
67.0
0
250
500
750
1000
1250
1500
1750
2000
no. of distorted examples

(4a-2)


CRF word wise accuracy vs no. of distorted training examples
accuracy
36
34
32
30
28
26
24
22
20
0
250
500
750
1000
1250
1500
1750
2000
no. of distorted examples

(4b-1)



(4b-2)

4.1 Observation for 4(a)

Value of C used = 1000
The letter-wise prediction accuracy decreases for both CRF and SVM-MC as more and more transformations are applied on the test-data, which is not surprising. However, the performance of CRF is relatively more robust to transformations as compared to SVM-MC.

4.2 Observation for 4(b)

Value of C used = 1000
The word-prediction accuracy also decreases for both CRF and SVM-MC as more and more transformations are applied on the test-data. The performance of CRF shows a steeper decrease in accuracy when compared to letter-wise accuracy. Though it still is always better than SVM-MC.