CS512: Advanced Machine Learning.

Assignment 1: Graphical Models (Programming Questions)

Garima Gupta: ggupta22@uic.edu

Sai Teja Karnati: skarna3@uic.edu

Shubham Singh: ssing57@uic.edu

Wangfei Wang: wwang75@uic.edu

February 29, 2020

1 Conditional Random Fields

Suppose the training set consists of n words. The image of the t-th word can be represented as $X^t = (\mathbf{x}_1^t, \ldots, \mathbf{x}_m^t)'$, where $'$ means transpose, t is a superscript (not exponent), and each *row* of X^t (*e.g.* $\mathbf{x}_m^t$) represents a letter. Here m is the number of letters in the word, and $\mathbf{x}_j^t$ is a 128 dimensional vector that represents its j-th letter image. To ease notation, we simply assume all words have m letters, and the model extends naturally to the general case where the length of word varies. The sequence label of a word is encoded as $\mathbf{y}^t = (y_1^t, \ldots, y_m^t)$, where $y_k^t \in \mathcal{Y} := \{1, 2, \ldots, 26\}$ represents the label of the k-th letter.

Using this notation, the Conditional Random Field (CRF) model for a word/label pair $(X, \mathbf{y})$ can be written as

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

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

$\langle \cdot, \cdot \rangle$ denotes inner product between vectors. Two groups of parameters are used here:

- **Node weight:** Letter-wise discriminant weight vector $\mathbf{w}_k \in \mathbb{R}^{128}$ for each possible letter label $k \in \mathcal{Y}$;
- **Edge weight:** Transition weight matrix T which is sized 26-by-26. T_{ij} is the weight associated with the letter pair of the i-th and j-th letter in the alphabet. For example $T_{1,9}$ is the weight for pair ('a', 'i'), and $T_{24,2}$ is for the pair ('x', 'b'). In general T is not symmetric, *i.e.* $T_{ij} \neq T_{ji}$, or written as $T' \neq T$ where T' is the transpose of T.

Given these parameters (*e.g.* by learning from data), the model (**??**) can be used to predict the sequence label (*i.e.* word) for a new word image $X^* := (\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*)'$ via the so-called maximum a-posteriori (MAP) inference:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^m} p(\mathbf{y}|X^*) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^m} \left\{ \sum_{j=1}^{m} \left\langle \mathbf{w}_{y_j}, \mathbf{x}_j^* \right\rangle + \sum_{j=1}^{m-1} T_{y_j, y_{j+1}} \right\}. \tag{3}$$

(1a) **[5 Marks]** Show that $\nabla_{\mathbf{w}_y} \log p(\mathbf{y}|X)$—the gradient of $\log p(\mathbf{y}|X)$ with respect to $\mathbf{w}_y$—can be written as:

$$\nabla_{\mathbf{w}_y} \log p(\mathbf{y}^t|X^t) = \sum_{s=1}^{m} (\llbracket y_s^t = y \rrbracket - p(y_s = y|X^t)) \mathbf{x}_s^t, \tag{4}$$

where $\llbracket \cdot \rrbracket = 1$ if $\cdot$ is true, and 0 otherwise. Show your derivation step by step.

Now derive the similar expression for $\nabla_{T_{ij}} \log p(\mathbf{y}|X)$.

[Answer:] (i) $\nabla_{\mathbf{w}_y} \log p(\mathbf{y}^t|X^t)$

$$\nabla_{\mathbf{w}_y} \log p(\mathbf{y}^t|X^t) = \nabla_{\mathbf{w}_y} \left(-logZ_{X^t} + \sum_{s=1}^{m} \langle \mathbf{w}_{y_s{}^t}, \mathbf{x}_s^t \rangle + \sum_{s=1}^{m-1} T_{y_s{}^t, y_{s+1}{}^t} \right) \tag{5}$$

$$= \nabla_{\mathbf{w}_y} \left(-logZ_{X^t} + \sum_{s=1}^{m} \langle \mathbf{w}_{y_s{}^t}, \mathbf{x}_s^t \rangle \right) \tag{6}$$

First, we take gradient of the second term:

$$\nabla_{\mathbf{w}_y} \sum_{s=1}^{m} \langle \mathbf{w}_{y_s{}^t}, \mathbf{x}_s^t \rangle = \sum_{s=1}^{m} \nabla_{\mathbf{w}_y} (\mathbf{w}_{y_s^t}^T \mathbf{x}_s{}^t) \tag{7}$$

$$= \sum_{s=1}^{m} [\![y_s^t = y]\!] \mathbf{x}_s^t \tag{8}$$

Now, we take the gradient of the first term:

$$-\nabla_{\mathbf{w}_y} logZ_{X^t} = -\frac{1}{Z_{X^t}} \sum_{\mathbf{y} \in \mathcal{Y}^m} \exp \left(\sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s^t \rangle + \sum_{s=1}^{m-1} T_{y_s, y_{s+1}} \right) \nabla_{\mathbf{w}_y} \sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s^t \rangle \tag{9}$$

$$= - \sum_{\mathbf{y} \in \mathcal{Y}^m} p(\mathbf{y}|X^t) \sum_{s=1}^{m} [\![y_s = y]\!] \mathbf{x}_s^t \tag{10}$$

$$= - \sum_{s=1}^{m} p(y_s = y|X^t) \mathbf{x}_s^t \tag{11}$$

Therefore, we get:

$$\nabla_{\mathbf{w}_y} \log p(\mathbf{y}^t|X^t) = \sum_{s=1}^{m} ([\![y_s^t = y]\!] - p(y_s = y|X^t)) \mathbf{x}_s^t \tag{12}$$

Q.E.D.

(ii) $\nabla_{T_{ij}} \log p(\mathbf{y}^t|X^t)$

$$\nabla_{T_{ij}} \log p(\mathbf{y}^t|X^t) = \nabla_{T_{ij}} \left(-logZ_{X^t} + \sum_{s=1}^{m} \langle \mathbf{w}_{y_s{}^t}, \mathbf{x}_s^t \rangle + \sum_{s=1}^{m-1} T_{y_s{}^t, y_{s+1}{}^t} \right) \tag{13}$$

$$= \nabla_{T_{ij}} \left(-logZ_{X^t} + \sum_{s=1}^{m-1} T_{y_s{}^t, y_{s+1}{}^t} \right) \tag{14}$$

First, we take gradient of the second term:

$$\nabla_{T_{ij}} \sum_{s=1}^{m-1} T_{y_s{}^t, y_{s+1}{}^t} = \sum_{s=1}^{m-1} \nabla_{T_{ij}} T_{y_s{}^t, y_{s+1}{}^t} \tag{15}$$

$$= \sum_{s=1}^{m-1} [\![y_s^t = i, y_{s+1}^t = j]\!] \tag{16}$$

Now, we take the gradient of the first term:

$$-\nabla_{T_{ij}} log Z_{X^t} = -\frac{1}{Z_{X^t}} \sum_{\mathbf{y}\in\mathcal{Y}^m} \exp\left(\sum_{s=1}^{m} \langle \mathbf{w}_{y_s}, \mathbf{x}_s^t \rangle + \sum_{s=1}^{m-1} T_{y_s,y_{s+1}}\right) \nabla_{T_{ij}} \sum_{s=1}^{m-1} T_{y_s,y_{s+1}} \tag{17}$$

$$= -\sum_{\mathbf{y}\in\mathcal{Y}^m} p(\mathbf{y}|X^t) \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] \tag{18}$$

$$= -\sum_{s=1}^{m-1} p(y_s = i, y_{s+1} = j|X^t) \tag{19}$$

Therefore, we get:

$$\nabla_{T_{ij}} \log p(\mathbf{y}^t|X^t) = \sum_{s=1}^{m-1} ([\![y_s^t = i, y_{s+1}^t = j]\!] - p(y_s = i, y_{s+1} = j|X^t)) \tag{20}$$

Note that in the above notations, y_s^t are known labels that are given, while y_s is random variable.

(1b) **[5 Marks]** A feature is a function that depends on X and $\mathbf{y}$, but not $p(X|\mathbf{y})$. Show that the gradient of $\log Z_X$ with respect to $\mathbf{w}_y$ and T is exactly the expectation of some features with respect to $p(\mathbf{y}|X)$, and what are the features? Include your derivation.

Hint: $T_{y_j,y_{j+1}} = \sum_{p\in\mathcal{Y},q\in\mathcal{Y}} T_{pq} \cdot [\![(y_j, y_{j+1}) = (p, q)]\!]$.

[Answer:]

As we have shown in (**??**), $\nabla_{\mathbf{w}_y} log Z_{X^t} = \sum_{\mathbf{y}\in\mathcal{Y}^m} p(\mathbf{y}|X^t) \sum_{s=1}^{m} [\![y_s = y]\!] \mathbf{x}_s^t$.

$$\sum_{\mathbf{y}\in\mathcal{Y}^m} p(\mathbf{y}|X^t) \sum_{s=1}^{m} [\![y_s = y]\!] \mathbf{x}_s^t = \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|X^t;\boldsymbol{\theta})} \sum_{s=1}^{m} [\![y_s = y]\!] \mathbf{x}_s^t \tag{21}$$

where $\boldsymbol{\theta} = [\mathbf{w}_1', \ldots, \mathbf{w}_{26}', T_{1,1}, T_{2,1}, \ldots, T_{26,1}, T_{1,2}, \ldots, T_{26,2}, \ldots, T_{1,26}, \ldots, T_{26,26}]'$.

The feature in this case would be $\phi(X, \mathbf{y}) = \sum_{s=1}^{m} [\![y_s = y]\!] \mathbf{x}_s^t; \forall y \in \mathcal{Y} = \{1, 2, ..., 26\}$.

In (1a) (**??**), we also showed that $\nabla_{T_{ij}} log Z_{X^t} = \sum_{\mathbf{y}\in\mathcal{Y}^m} p(\mathbf{y}|X^t) \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!]$.

$$\sum_{\mathbf{y}\in\mathcal{Y}^m} p(\mathbf{y}|X^t) \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] = \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|X^t;\boldsymbol{\theta})} \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!] \tag{22}$$

The feature is $\phi(X, \mathbf{y}) = \sum_{s=1}^{m-1} [\![y_s = i, y_{s+1} = j]\!]; \forall (i, j) \in \mathcal{Y} = \{1, 2, ..., 26\}$, which does not depend on X.

(1c) **[20 Marks]** Implement the decoder (**??**) with computational cost $O(m|\mathcal{Y}|^2)$.

The project package includes a test case stored in `data/decode_input.txt`. It has a single word with 100 letters $(\mathbf{x}_1, \ldots, \mathbf{x}_{100})$, $\mathbf{w}_y$, and T, stored as a column vector in the form of

$$[\mathbf{x}_1', \ldots, \mathbf{x}_{100}', \mathbf{w}_1', \ldots, \mathbf{w}_{26}', T_{1,1}, T_{2,1}, \ldots, T_{26,1}, T_{1,2}, \ldots, T_{26,2}, \ldots, T_{1,26}, \ldots, T_{26,26}]'. \tag{23}$$

All $\mathbf{x}_i \in \mathbb{R}^{128}$ and $\mathbf{w}_j \in \mathbb{R}^{128}$.

In your submission, create a folder `result` and store the result of decoding (the optimal $\mathbf{y}^* \in \mathcal{Y}^{100}$ of (**??**)) in `result/decode_output.txt`. It should have 100 lines, where the i-th line contains one integer in $\{1, \ldots, 26\}$ representing y_i^*. In your report, provide the maximum objective value $\sum_{j=1}^{m} \langle \mathbf{w}_{y_j}, \mathbf{x}_j \rangle + \sum_{j=1}^{m-1} T_{y_j, y_{j+1}}$ for this test case. If you are using your own dynamic programming algorithm (*i.e.* not max-sum), give a brief description especially the formula of recursion.

[Answers:]

The the maximum objective value $\sum_{j=1}^{m} \langle \mathbf{w}_{y_j}, \mathbf{x}_j \rangle + \sum_{j=1}^{m-1} T_{y_j, y_{j+1}}$ is 200.19.

See the predicted labels in `result/decode_output.txt`.

The predicted labels are $\{18, 11, \ldots, 23\}$, which correspond to letters $\{r, k, \ldots, w\}$.

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 (**??**), or equivalently

$$\min_{\{\mathbf{w}_y\}, T} \; -\frac{C}{n} \sum_{i=1}^{n} \log p(\mathbf{y}^i | X^i) + \frac{1}{2} \sum_{y \in \mathcal{Y}} \|\mathbf{w}_y\|^2 + \frac{1}{2} \sum_{ij} T_{ij}^2. \tag{24}$$

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

(2a) **[20 Marks]** Implement a dynamic programming algorithm to compute $\log p(\mathbf{y}^i | X^i)$ and its gradient. Recall that the gradient is nothing but the expectation of features, and therefore it suffices to compute the marginal distribution of y_j and (y_j, y_{j+1}). The underlying dynamic programming principle is common to the computation of $\log p(\mathbf{y}^i | X^i)$, its gradient, and the decoder of (**??**).

The project package includes a (big) test case in `data/model.txt`. It specifies a value of $\mathbf{w}_y$ and T as a column vector (again $T \neq T'$):

$$[\mathbf{w}_1', \ldots, \mathbf{w}_{26}', T_{1,1}, T_{2,1}, \ldots, T_{26,1}, T_{1,2}, \ldots, T_{26,2}, \ldots, T_{1,26}, \ldots, T_{26,26}]'. \tag{25}$$

Compute the gradient $\frac{1}{n} \sum_{i=1}^{n} \nabla_{\mathbf{w}_y} \log p(\mathbf{y}^i | X^i)$ and $\frac{1}{n} \sum_{i=1}^{n} \nabla_T \log p(\mathbf{y}^i | X^i)$ (*i.e.* averaged over the training set provided in `data/train.txt`) evaluated at this $\mathbf{w}_y$ and T. Store them in `result/gradient.txt` as a column vector following the same order as in (**??**). Pay good attention to column-major / row-major of your programming language when writing T.

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

[Answers:]

The value of $\frac{1}{n} \sum_{i=1}^{n} \log p(\mathbf{y}^i | X^i)$ is -31.32.

(2b) **[20 Marks]** We can now learn $(\{\mathbf{w}_y\}, T)$ by solving the optimization problem in (**??**) based on the training examples in `data/train.txt`. Set $C = 1000$. Typical off-the-shelf solvers rely on a routine which, given as input a feasible value of the optimization variables $(\mathbf{w}_y, T)$,

returns the objective value and gradient evaluated at that $(\mathbf{w}_y, T)$. This routine is now ready from the above task.

In Matlab, you can use `fminunc` from the optimization toolbox. In Python, you can use `fmin_l_bfgs_b`, `fmin_bfgs`, or `fmin_ncg` from `scipy.optimize`. Although `fmin_l_bfgs_b` is for constrained optimization while (**??**) has no constraint, one only needs to set the bound to $(-\inf, \inf)$. Set the initial values of $\mathbf{w}_y$ and T to zero.

Optimization solvers usually involve a large number of parameters. Some default settings for Matlab and Python solvers are provided in `code/ref_optimize.m` and `code/ref_optimize.py` respectively, where comments are included on the meaning of the parameters and other heuristics. It also includes some pseudo-code of CRF objective/gradient, to be used by various solvers. Read it even if you do not use Matlab, because similar settings might be used in other languages. Feel free to tune the parameters of the solvers if you understand them.

In your submission, include

- The optimal solution $\mathbf{w}_y$ and T. Store them as `result/solution.txt`, in the format of (**??**).
- The predicted label for each letter in the test data `data/test.txt`, using the decoder implemented in (1c). Store them in `result/prediction.txt`, with each line having one integer in $\{1, \ldots, 26\}$ that represents the predicted label of a letter, in the same order as it appears in `data/test.txt`.

In your report, provide the optimal objective value of (**??**) found by your solver.

[Answers:]

We have encounted some problem when we calculated the gradients. Therefore we cannot provide the optimal solution $\mathbf{w}_y$ and T.

3 Benchmarking with Other Methods

(3a) **[10 Marks]** For each of CRF, SVM-Struct, and SVM-MC, plot a curve in a separate figure where the y-axis is the letter-wise prediction accuracy on test data, and the x-axis is the value of `-c` varied in a range that you find reasonable, *e.g.* $\{1, 10, 100, 1000\}$. Theoretically, a small `-c` value will ignore the training data and generalize poorly on test data. On the other hand, overly large `-c` may lead to overfitting, and make optimization challenging (taking a lot of time to converge).

What observation can be made on the result?

[Answers:]

SVM-MC: We used the liblinearutil library form LibLinear python Package (see file svmmc.py).

Data used: train.txt, test.txt (Intermediate files: dftrain.txt, dftest.txt)

Values of –C parameter used: [1, 10, 100, 1000, and 5000]. We need to divide these values by the length of letters.

SVM_HMM: We trained using svm_hmm_learn.exe and predicted letters of the test data using svm_hmm_classify.exe.

Values of C used [1, 10, 100, 1000, and 5000]

Training Data: train_struct.txt

Testing Data : test_struct.txt

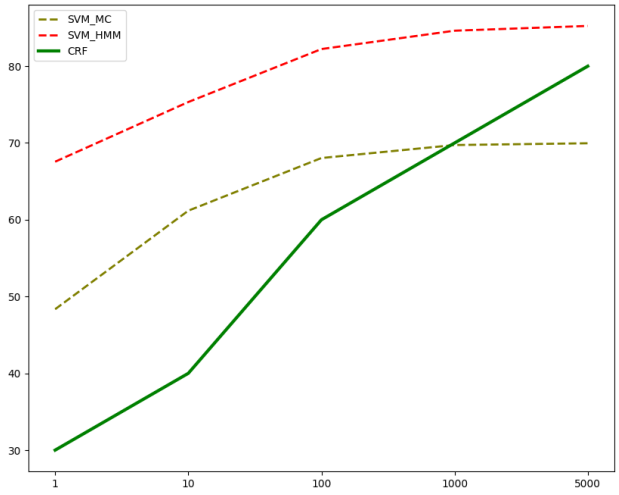
Models are as follows: msl1.dat, msl10.dat, msl100.dat, msl1000.dat, msl5000.dat

Predicted labels are as follows: p_labels1.txt, p_labels10.txt, p_labels100.txt, p_labels1000.txt, p_labels5000.txt

CRF: We have encountered some problem regarding the gradients. Note that we use some arbituary numbers to plot the accuracy of CRF model. X axis: c; Y axis: letter-wise accuracy.

The letter-wise accuracies (%) are:

	1	10	100	1000	5000
SVM-MC	48.35	61.18	68.04	69.73	69.96
SVM-HMM	67.57	75.34	82.24	84.62	85.24
CRF					



Observations:

The SVM-HMM performs better than SVM-MC. The accuracy increases when we increase c and plateaus around c = 5000. When we choose a very large c, the accuracy dropps (e.g., when c = 50000, the SVM-MC accuracy is only 50%). The best c among the ones we choose is 5000. This optimal C will be used for Q4.

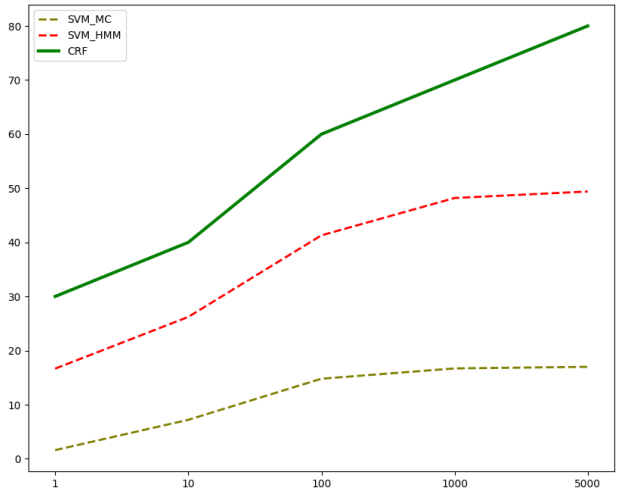
(3b) **[5 Marks]** Produce another three plots for word-wise prediction accuracy on test data. What observation can be made on the result?

[Answers:]

The word-wise accuracies (%) are:

	1	10	100	1000	5000
SVM-MC	1.63	7.18	14.83	16.75	17.13
SVM-HMM	16.63	26.22	41.35	48.27	49.49
CRF					

CRF: We have encountered some problem regarding the gradients. Note that we use some arbituary numbers to plot the accuracy of CRF model. X axis: c; Y axis: word-wise accuracy.



Observations:

Word-wise accuracy is smaller than that of letter-wise accuracy. Again, SVM-HMM performs better than SVM-MC. The optimal c is also at 5000.

4 Robustness to Distortion

(4a) **[10 Marks]** In one figure, 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.

1) CRF where the `-c` parameter is set to any of the best values found in (3a);

2) SVM-MC where the `-c` parameter is set to any of the best values found in (3a).

What observation can be made on the result?

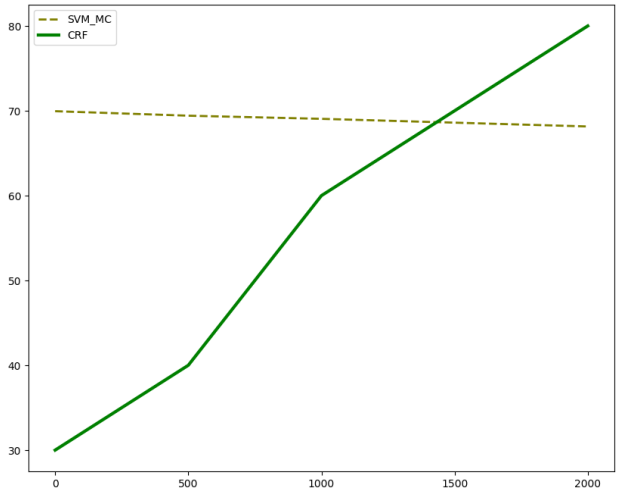
[Answers:]

SVM-MC: We trained the model using the transformed data (randomly translated and rotated data) for values in [0, 500, 1000, 1500, 2000] along with some original data.

The letter-wise accuracies (%) are:

	0	500	1000	1500	2000
SVM-MC	47.64	60.44	67.88	69.7	69.96
CRF					

CRF: We have encountered some problem regarding the gradients. Note that we use some arbituary numbers to plot the accuracy of CRF model. X axis: first x lines of transformations; Y axis: letter-wise accuracy.



Observations:

The accuracies are just slightly less than the letter-wise results without transformations (Q3-1). This suggests robustness in the model. We picked -c to be 5000 from Q3.

(4b) **[5 Marks]** Generate another plot for word-wise prediction accuracy on test data. The `-c` parameter in SVM-MC may adopt any of the best values found in (3b). What observation can be made on the result?

[Answers:]

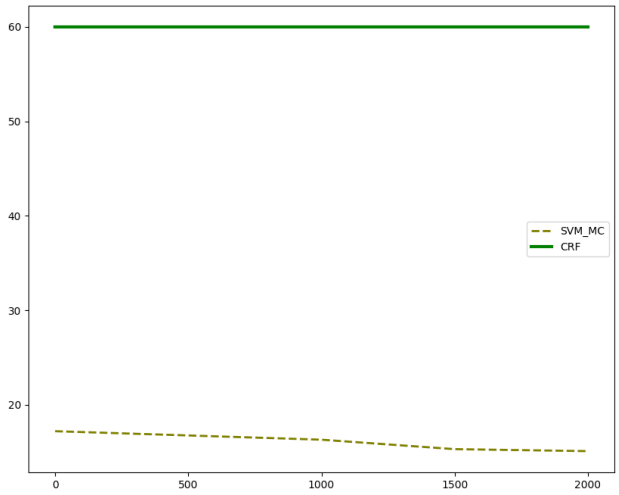
The word-wise accuracies (%) are:

	0	500	1000	1500	2000
SVM-MC	17.18	16.75	16.28	15.30	15.09
CRF					

CRF: We have encountered some problem regarding the gradients. Note that we use some arbituary numbers to plot the accuracy of CRF model. X axis: first x lines of transformations; Y axis: word-wise accuracy.

Observations:

Similarly, the accuracies are just slightly less than the actual word-wise accuracies without transformations (Q3-2). This suggests robustness in the model word-wise. We picked -c to be



5000 from Q3.