CS688: Graphical Models - Spring 2018

Assignment 2

Assigned: Tuesday, Feb 20th. Due: Tuesday, Mar 6th 11:55pm

Getting Started: You should complete the assignment using your own installation of Python 2.7. The only modules you are permitted to use in your implementations are Numpy and SciPy. To get started with the code portions of the assignment, download the assignment archive from Moodle and unzip the file. The data files for this assignment are in the `data` directory. Code templates are in the `code` directory. The README file contains additional descriptions of the download. **Note**: The autograder uses Numpy 1.14 and SciPy 1.0.0. Please use this version to minimize compatibility issues. Note that the autograder environment will fail if you attempt to load other packages.

Deliverables: This assignment has two types of deliverables: a report and code files.

- **Report:** The solution report will give your answers to the homework questions. Items that you should include in your report are marked with **(report)**. The maximum length of the report is 5 pages in 11 point font, including all figures and tables. You can use any software to create your report, but your report must be submitted in PDF format. You will upload the PDF of your report to Gradescope under `HW02-Report` for grading. It is strongly recommended that you typeset your report. To assist with this if you wish to use Latex, the Latex source of the handout is also included in the homework archive.
- **Code:** The second deliverable is your code. Items that you should include in your code are marked with **(code)**. Your code must be Python 2.7 (no iPython notebooks, other formats, or code from other versions of Python). You will upload a zip file (not rar, bz2 or other compressed format) containing all of your code to Gradescope under `HW02-Programming` for autograding. When unzipped, your zip file should produce a directory called `code`. If your zip file has the wrong structure, the autograder may fail to run.

Academic Honesty Statement: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Sharing your solutions with other students is also considered cheating. Collaboration indistinguishable from copying is a violation of the course's collaboration policy and will be treated as cheating. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Introduction: In this assignment, you will experiment with different aspects of modeling, learning, and inference with chain-structured conditional random fields (CRFs). This assignment focuses on the task of optical character recognition (OCR). We will explore an approach that bridges computer vision and natural language processing by jointly modeling the labels of sequences of noisy character images that form complete words. This is a natural problem for chain-structured CRFs. The node potentials can capture bottom-up information about the character represented by each image, while the edge potentials can capture

information about the co-occurrence of characters in adjacent positions within a word.

Data: The underlying data are a set of N sequences corresponding to images of the characters in individual words. Each word i consists of L_i positions. For each position j in word i, we have a noisy binary image of the character in the that position. In this assignment, we will use the raw pixel values of the character images as features in the CRF. The character images are 20×16 pixels. We convert them into 1×320 vectors. We include a constant bias feature along with the pixels in each image, giving a final feature vector of length $F = 321$. x_{ijf} indicates the value of feature f in position j of word i. The provided training and test files *train_img<i>.txt* and *test_img<i>.txt* list the character image $\mathbf{x}_{ij}$ on row j of file i as a 321-long, space-separated sequence.[1] The data files are in the column-major format. Given the sequence of character images $\mathbf{x}_i = [\mathbf{x}_{i1}, ..., \mathbf{x}_{iL_i}]$ corresponding to test word i, your goal is to infer the corresponding sequence of character labels $\mathbf{y}_i = [y_{i1}, ..., y_{iL_i}]$. There are $C = 26$ possible labels corresponding to the lower case letters "a" to "z." The character labels for each training and test word are available in the files *train_words.txt* and *test_words.txt*. The figure below shows several example words along with their images.
Model: The conditional random field model is a conditional model $P_W(\mathbf{y}_i|\mathbf{x}_i)$ of the sequence of class labels $\mathbf{y}_i$ given the sequence of feature vectors $\mathbf{x}_i$ that depends on a collection of parameters W. The CRF graphical model is shown below for a sequence of length 4.

The probabilistic model for the CRF we use in this assignment is given below. The CRF model contains one feature parameter W^F_{cf} for each of the C labels and F features. The feature parameters encode the compatibility between feature values and labels. The CRF also contains one transition parameter $W^T_{cc'}$ for each pair of labels c and c'. The transition parameters encode the compatibility between adjacent labels in the sequence. We parameterize the model in log-space, so all of the parameters can take arbitrary (positive or negative) real values. We have one feature potential $\phi^F_j(y_{ij}, \mathbf{x}_{ij})$ for each position j in sequence i and one transition potential for each pair of adjacent labels $\phi^T_j(y_{ij}, y_{ij+1})$ in sequence i.

$$\phi^F_j(y_{ij}, \mathbf{x}_{ij}) = \sum_{c=1}^{C} \sum_{f=1}^{F} W^F_{cf}[y_{ij} = c]x_{ijf}$$

$$\phi^T_j(y_{ij}, y_{ij+1}) = \sum_{c=1}^{C} \sum_{c'=1}^{C} W^T_{cc'}[y_{ij} = c][y_{ij+1} = c']$$

Given this collection of potentials, the joint energy function on $\mathbf{x}_i$ and $\mathbf{y}_i$ is defined below.

$$\begin{aligned} E_W(\mathbf{y}_i, \mathbf{x}_i) &= -\left(\sum_{j=1}^{L_i} \phi^F_j(y_{ij}, \mathbf{x}_{ij}) + \sum_{j=1}^{L_i-1} \phi^T_j(y_{ij}, y_{ij+1}) \right) \\ &= -\left(\sum_{j=1}^{L_i} \sum_{c=1}^{C} \sum_{f=1}^{F} W^F_{cf}[y_{ij} = c]x_{ijf} + \sum_{j=1}^{L_i-1} \sum_{c=1}^{C} \sum_{c'=1}^{C} W^T_{cc'}[y_{ij} = c][y_{ij+1} = c'] \right) \end{aligned}$$

The joint probability of $\mathbf{y}_i$ and $\mathbf{x}_i$ is given by the Gibbs distribution for the model.

[1] Images are also provided for each training and test word as standard PNG-format files *train_img<i>.png* and *test_img<i>.png*. These are for your reference and not for use in training or testing algorithms.

$$P_W(\mathbf{y}_i, \mathbf{x}_i) = \frac{\exp(-E_W(\mathbf{y}_i, \mathbf{x}_i))}{Z_W}$$
$$Z_W = \sum_{\mathbf{y}} \sum_{\mathbf{x}} \exp(-E_W(\mathbf{y}, \mathbf{x}))$$

However, as the name implies, a conditional random field is not trained to maximize the joint likelihood of $\mathbf{x}$ and $\mathbf{y}$. Instead, the model is trained to maximize the conditional likelihood of $\mathbf{y}$ given $\mathbf{x}$ similar to a discriminative classifier like logistic regression. The conditional probability of $\mathbf{y}$ given $\mathbf{x}$ is shown below. Note that the partition function $Z_W(\mathbf{x}_i)$ that results from conditioning on a sequences of feature vectors $\mathbf{x}_i$ will generally be different for each sequence $\mathbf{x}_i$.

$$P_W(\mathbf{y}_i|\mathbf{x}_i) = \frac{\exp(-E_W(\mathbf{y}_i, \mathbf{x}_i))}{Z_W(\mathbf{x}_i)}$$
$$Z_W(\mathbf{x}_i) = \sum_{\mathbf{y}} \exp(-E_W(\mathbf{y}, \mathbf{x}_i))$$

1. (*10 points*) **Basics:** To begin, implement the following basic methods. While we will only experiment with one data set, your code should be written to work with any label set and any number of features.

1.1. (*2 pts*) Implement the function `get_params`, which returns the current model parmeters.

1.2. (*2 pts*) Implement the function `set_params`, which sets the current model parmeters.

1.3. (*6 pts*) Implement the function `energy`, which computes the joint energy of a label and feature sequence $\mathbf{y}$ and $\mathbf{x}$.

2. (*30 points*) **Inference:** Efficient inference is the key to both learning and prediction in CRFs. In this question you will describe and implement inference methods for chain-structured CRF.

2.1. (*10 pts*) Explain how factor reduction and the sum-product algorithm can be combined to enable efficient inference for the single-node distributions $P_W(y_j|\mathbf{x})$ and the pairwise distribution $P_W(y_j, y_{j+1}|\mathbf{x})$. These distributions are technically conditional marginal distributions, but since we are always conditioning on $\mathbf{x}$ in a CRF, we will simply refer to them as marginal, and pairwise marginal distributions. Your solution to this question must have linear complexity in the length of the input, and should not numerically underflow or overflow even for long sequences and many labels. **(report)**

2.2. (*10 pts*) Implement the function `log_Z`, which computes the log partition function for the distributin $P_W(\mathbf{y}|\mathbf{x})$. **(code)**

2.3. (*10 pts*) Implement the function `predict_logprob`, which computes the individual $P_W(y_j|\mathbf{x})$ and pairwise marginals $P_W(y_j, y_{j+1}|\mathbf{x})$ for each position in the sequence. **(code)**

3. (*30 points*) **Learning:** In this problem, you will derive the maximum likelihood learning algorithm for chain-structured conditional random field models. Again, this algorithm maximizes the average conditional log likelihood function $\mathcal{L}(W|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \log P_W(\mathbf{y}_i|\mathbf{x}_i)$, not the average joint log likelihood, but the learning approach is still typically referred to as maximum likelihood.

3.1. (*2 pts*) Write down the average conditional log likelihood function for the CRF given a data set consisting of N input sequences $\mathbf{x}_i$ and label sequences $\mathbf{y}_i$ in terms of the parameters and the data. **(report)**

3.2. (*5 pts*) Derive the derivative of the average conditional log likelihood function with respect to the feature parameter W^F_{cf}. Show your work. **(report)**

3.3. (*5 pts*) Derive the derivative of the average conditional log likelihood function with respect to the transition parameter $W^T_{cc'}$. Show your work. **(report)**

3.4. (*3 pts*) Explain how the average conditional log likelihood function and its gradients can be efficiently computed using the inference method you developed in the previous question. **(report)**

3.5. (*5 pts*) Implement the function `log_likelihood` to efficiently compute the average conditional log likelihood given a set of labeled input sequences. **(code)**

3.6. (*5 pts*) Implement the function `gradient_log_likelihood` to efficiently compute the gradient of the average conditional log likelihood given a set of labeled input sequences. **(code)**

3.7. (*5 pts*) Use your implementation of `log_likelihood` and `gradient_log_likelihood` along with a numerical optimizer to implement maximum (conditional) likelihood learning in the `fit` function. The reference solutions were computed using the `fmin_bfgs` method from `scipy.optimize` using default optimizer settings. It is recommended that you use this optimizer and the default settings as well to minimize any discrepancies.
(code)

4. (*10 points*) **Prediction:** To use the learned model to make predictions, implement the function `predict`. Given an unlabeled input sequence, this function should compute the node marginal $P_W(y_j|\mathbf{x})$ for every position j in the label sequence conditioned on a feature sequence $\mathbf{x}$, and then predict the marginally most likely label. This is called *max marginal prediction*. **(code)**.

5. (*20 points*) **Experiments:** In this problem, you will use your implementation to conduct basic learning experiments. Add your experiment code to `experiment.py`

5.1. (*10 pts*) Use your CRF implementation and the first 100, 200, 300, 400, 500, 600, 700, and 800 training cases to learn eight separate models. For each model, compute the average test set conditional log likelihood and the average test set prediction error. As your answer to this question, provide two separate line plots showing average test set conditional log likelihood and average test error vs the number of training cases. **(report)**

5.2. (*10 pts*) Using your CRF model trained on all 800 data cases, conduct an experiment to see if the compute time needed to perform max marginal inference scales linearly with the length of the feature sequence as expected. You should experiment with sequences up to length 20. You will need to create your own longer sequences. Explain how you did so. You should also use multiple repetitions to stabilize the time estimates. As your answer to this question, provide a line plot showing the average time needed to perform marginal inference vs the sequence length. **(report)**