Stony Brook University
CSE512 – Machine Learning – Spring 16
Homework 4, Due: 21 Apr 2016, 11:59PM

This homework contains 3 questions. The last question requires programming. The maximum number of points is 100 plus 20 bonus points.

1 Manual calculation of one round of EM for a GMM [30 points]

(Extended version of: Murphy Exercise 11.7) In this question we consider clustering 1D data with a mixture of 2 Gaussians using the EM algorithm. You are given the 1-D data points $x = [1 \;\; 10 \;\; 20]$.

M step

Suppose the output of the E step is the following matrix:

$$R = \begin{pmatrix} 1 & 0 \\ 0.4 & 0.6 \\ 0 & 1 \end{pmatrix}$$

where entry $R_{i,c}$ is the probability of observation x_i belonging to cluster c (the responsibility of cluster c for data point i). You just have to compute the M step. You may state the equations for maximum likelihood estimates of these quantities (which you should know) without proof; you just have to apply the equations to this data set. You may leave your answer in fractional form. Show your work.

1. *[3 points]* Write down the likelihood function you are trying to optimize.
2. *[6 points]* After performing the M step for the mixing weights π_1, π_2, what are the new values?
3. *[6 points]* After performing the M step for the means μ_1 and μ_2, what are the new values?
4. *[6 points]* After performing the M step for the standard deviations σ_1 and σ_2, what are the new values?

E step

Now suppose the output of the M step is the answer to the previous section. You will compute the subsequent E step.

1. *[3 points]* Write down the formula for the probability of observation x_i belonging to cluster c.
2. *[6 points]* After performing the E step, what is the new value of R?

2 HMM with tied mixtures (25 points)

(Adapted from Murphy Exercise 13.4) Consider an HMM where the observation model has the form:

$$p(O_t|X_t = j, \boldsymbol{\theta}) = \sum_{k=1}^{K} w_{jk} \mathcal{N}(O_t|\mu_k, \Sigma_k) \quad \forall j \in \{1, \cdots, M\}. \tag{1}$$

In this model, we assume there are M types of hidden states. The observation for each hidden state is given above; it is a mixture of K Gaussians. However, the Gaussians are shared between the states, and the state influences the mixing weights but not the means and covariances. This is called semi-continuous HMM or tied-mixture HMM.

1. (5 points) List all the parameters of this HMM model
2. (10 points) Derive the E step. What do we need to estimate in the E step?
3. (10 points) Derive the M step. How do we update the parameters of the model?

3 (Implementation) Linear-Chain Hidden CRF for gesture recognition (45 points)

In this question, you will implement Hidden CRF [1] and use it for gesture recognition.

3.1 General Hidden CRF

Consider the general form of of Hidden CRF, which assumes the following probability model:

$$P(y, \mathbf{s}, \mathbf{X}|\mathbf{w}) = \frac{1}{Z} \exp(\mathbf{w} \cdot \phi(y, \mathbf{s}, \mathbf{X})) \tag{2}$$

where $\mathbf{X}$ is the observed variable or set of variables, y is the target, and $\mathbf{s}$ is hidden variable(s). The parameter vectors of the CRF is $\mathbf{w}$, which is what we need to learn. Define:

$$Z(y, \mathbf{X}) = \sum_{\mathbf{s}} \exp(\mathbf{w} \cdot \phi(y, \mathbf{s}, \mathbf{X})) \tag{3}$$

$$Z(\mathbf{X}) = \sum_{y} Z(y, \mathbf{X}) \tag{4}$$

A Hidden CRF models the conditional probability of a class label y given a set of observations $\mathbf{X}$ by:

$$P(y|\mathbf{X}, \mathbf{w}) = \sum_{\mathbf{s}} P(y, \mathbf{s}|\mathbf{X}, \mathbf{w}) = \sum_{\mathbf{s}} \frac{P(y, \mathbf{s}, \mathbf{X}|\mathbf{w})}{P(\mathbf{X}|\mathbf{w})} = \frac{Z(y, \mathbf{X})}{Z(\mathbf{X})} \tag{5}$$

Given a training set $\{(\mathbf{X}^i, y^i)\}$, we train the parameters by minimizing the regularized negative log likelihood:

$$L(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||^2 - \frac{1}{n} \sum_{i=1}^{n} \log P(y^i|\mathbf{X}^i, \mathbf{w}). \tag{6}$$

To minimize this objective function, we can use a gradient-based optimization method. This requires computing the gradient for a single training example:

$$\frac{\partial \log P(y|\mathbf{X}, \mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \log Z(y, \mathbf{X})}{\partial \mathbf{w}} - \frac{\partial \log Z(\mathbf{X})}{\partial \mathbf{w}} \tag{7}$$

We have:

$$\frac{\partial \log Z(y, \mathbf{X})}{\partial \mathbf{w}} = \frac{1}{Z(y, \mathbf{X})} \sum_{\mathbf{s}} \exp(\mathbf{w} \cdot \phi(y, \mathbf{s}, \mathbf{X})) \phi(y, \mathbf{s}, \mathbf{X}) = \sum_{\mathbf{s}} P(\mathbf{s}|y, \mathbf{X}, \mathbf{w}) \phi(y, \mathbf{s}, \mathbf{X}) \tag{8}$$

We can also show

$$\frac{\partial \log Z(\mathbf{X})}{\partial \mathbf{w}} = \sum_{y} \left(\sum_{\mathbf{s}} P(\mathbf{s}|y, \mathbf{X}, \mathbf{w}) \phi(y, \mathbf{s}, \mathbf{X}) \right) P(y|\mathbf{X}, \mathbf{w}) \tag{9}$$

Thus to use gradient-based optimization technique, we need an efficient way to compute:
$\sum_{\mathbf{s}} P(\mathbf{s}|y, \mathbf{X}, \mathbf{w}) \phi(y, \mathbf{s}, \mathbf{X})$ and $P(y|\mathbf{X}, \mathbf{w})$ for all y.

3.2 Linear-chain Hidden CRF

Consider the sequence classification problem where the target label is one of m classes, $y \in \{1, \cdots, m\}$. The observed variables X is a sequence of T frames $\mathbf{X} = \mathbf{X}_{1:T}$. Each frame $\mathbf{X}_t$ corresponds to a hidden state s_t, which belong to one of k states. For convenience of notation, we introduce a special state s_0 which always assumes value $k+1$. For the linear-chain CRF, the feature vector is defined as:

$$\phi(y, \mathbf{s}, \mathbf{X}) = \sum_{t=1}^{T} \phi_t(y, s_t, s_{t-1}, \mathbf{X}) \tag{10}$$

Then:

$$\sum_{\mathbf{s}} P(\mathbf{s}|y, \mathbf{X}, \mathbf{w})\phi(y, \mathbf{s}, \mathbf{X}) = \sum_{\mathbf{s}} P(\mathbf{s}|y, \mathbf{X}, \mathbf{w}) \sum_{t} \phi_t(y, s_t, s_{t-1}, \mathbf{X}) \tag{11}$$

$$= \sum_{t} \sum_{\mathbf{s}} P(\mathbf{s}|y, \mathbf{X}, \mathbf{w})\phi_t(y, s_t, s_{t-1}, \mathbf{X}) \tag{12}$$

$$= \sum_{t} \sum_{s_t, s_{t-1}} P(s_t, s_{t-1}|y, \mathbf{X}, \mathbf{w})\phi_t(y, s_t, s_{t-1}, \mathbf{X}) \tag{13}$$

The forward-backward algorithm:

$$\alpha_1(y, s_1) = \exp(\mathbf{w} \cdot \phi_1(y, s_1, s_0, \mathbf{X})) \tag{14}$$

$$\alpha_i(y, s_i) = \sum_{s_{i-1}} \exp(\mathbf{w} \cdot \phi_i(y, s_i, s_{i-1}, \mathbf{X}))\alpha_{i-1}(y, s_{i-1}) \tag{15}$$

$$\beta_T(y, s_T) = 1 \tag{16}$$

$$\beta_i(y, s_i) = \sum_{s_{i+1}} \exp(\mathbf{w} \cdot \phi_{i+1}(y, s_{i+1}, s_i, \mathbf{X}))\beta_{i+1}(s_{i+1}) \tag{17}$$

And

$$P(s_t, s_{t-1}|y, \mathbf{X}, \mathbf{w}) \propto \alpha_{t-1}(y, s_{t-1}) \exp(\mathbf{w} \cdot \phi_t(y, s_t, s_{t-1}, \mathbf{X}))\beta_t(y, s_t) \tag{18}$$

$$P(y|\mathbf{X}, \mathbf{w}) \propto \sum_{s_t} \alpha_t(y, s_t)\beta_t(y, s_t) \tag{19}$$

Note that the above equation are proportional. You will need to compute the exact values for optimization. In general, the above formula works for any feature function $\phi(y, s_t, s_{t-1}, \mathbf{X})$. One possibility is to use a feature vector that resembles HMM. Suppose the dimension of feature vector for each time step of is d, the number of states is k, and the number of classes is m. HMM-like feature is as follows:

$$\phi_t(y, s_t, s_{t-1}, \mathbf{X}) = [\Phi^s(:); \Phi^l(:); \Phi^t(:)] \tag{20}$$

$$\Phi^s = zeros(d, k); \Phi^s(:, s_t) = \mathbf{X}(:, t) \tag{21}$$

$$\Phi^l = zeros(k, m); \Phi^l(s_t, y) = 1 \text{ \% Observing } s_t \text{ in Class } y \tag{22}$$

$$\Phi^t = zeros(k, k+1, m); \Phi^t(s_t, s_{t-1}, y) = 1 \text{ \% Transition from } s_{t-1} \text{ to } s_t \text{ in Class } y \tag{23}$$

Note that s_0 is a special state, it is always $k+1$. The feature vector is concatenation of three vectors. Φ^s encodes the fact that $\mathbf{X}(:, t)$ is associated with state s_t. Φ^l encodes that s_t belongs to class y. and Φ^t encodes State s_{t-1} transition to State s_t in class y.

One way to understand this feature vector is to look at the weight vector:

$$\mathbf{w} = [\mathbf{W}^s_{d \times k}(:); \mathbf{W}^l_{k \times m}(:); \mathbf{W}^t_{k \times (k+1) \times m}(:)] \tag{24}$$

The value $\mathbf{W}^s(:, i)^T\mathbf{x}$ is how likely an observation $\mathbf{x}$ belongs to State i. The value $\mathbf{W}^l(i, y)$ is how likely we observe States i in Class y. The value $\mathbf{W}^t(i, j, y)$ is how likely we observe the transition from State j to State i in Class y.

3.3 What are provided

To help you start, the following functions are provided:

1. Data and starter code:
   `https://dl.dropboxusercontent.com/u/3172165/mlCourse/hw4data.zip`
2. Feature computation function class: `HW4_HcrfFeat`
3. Forward-backward algorithm: `HW4_Hcrf.forwardBackward()`
4. Function to compute the $\frac{\partial \log Z(y,\mathbf{X})}{\partial \mathbf{w}}$: `HW4_Hcrf.cmpDerOfLogZ()`
5. Utility functions for log-sum-exponential trick and converting from unnormalized log probabilities to normalized probabilities are given in `HW4_Utils.m`.
6. Run `HW4_Hcrf.test_cmpDerOfLogZ()` for a demo of how to use some other functions.
7. `MHW_ChaLearnData.m` contains functions to load and display data. Run `showRandom` to display a random sequence of data; it's the 3D coordinates of human joints over time. Use `load3ClassData()` to load data for three classes 5,6,7. We will refer to this data as `3Classes dataset`. Use `load20ClassData()` to load data for all 20 classes, which will be referred to as `20Classes dataset`.

3.4 What to implement

1. Implement a function that returns the objective value and derivative with respect to $\mathbf{w}$ of the function defined in Eq. 6.
2. Write code to learn the parameters of HCRF by optimizing the objective function (Eq. 6). To optimize the function, you can use `fminunc` of Matlab. Set the 'Algorithm' option to 'quasi-newton', and this corresponds to using BFGS algorithm. To avoid your program to run for a long time, you might want to limit the number of iterations by setting 'MaxIterations' (e.g., setting to 50 or 100). You might want to set 'Display' to 'iter-detailed' to track the progress. A good explanation of BFGS can be found here `http://aria42.com/blog/2014/12/understanding-lbfgs`. If you want, you can also use SGD for optimization.
3. Write code for prediction and evaluation

3.5 What to report

1. (5 points) Derive the formula to compute the gradient of the training objective (Eq. 6) w.r.t. $\mathbf{w}$. Your formula must be based on the output of the forward-backward algorithm (i.e., $\alpha(\cdot,\cdot), \beta(\cdot,\cdot)$) and the derivative $\frac{\log Z(y,\mathbf{X}^i)}{\partial \mathbf{w}}$
2. (10 points) Derive the formula to compute the training objective (Eq. 6) based on $\alpha(\cdot,\cdot), \beta(\cdot,\cdot)$.
3. (10 points) Use $\lambda = 0.001$ and learn a HCRF (i.e., $\mathbf{w}$). Report the objective value (Eq. 6) both training and validation data for 3Classes dataset.
4. (10 points) Report: (a) the training accuracy, (b) the validation accuracy, and (c) the confusion matrix for validation data for 3Classes dataset
5. (10 points) Use your model to predict the label for the test set of the 3Classes dataset. Save the predicted labels on a CSV file `predTestLabels_3classes.csv`. The order of predicted labels should correspond to the order of the reviews in `tstData_3classes.mat`. Submit this `predTestLabels_3classes.csv` file on Kaggle and report classification accuracy.

6. (10 bonus points) Submit `predTestLabels_3classes.csv` and enter our Kaggle competition for fame. You must use your Stony Brook email to register on Kaggle. We will maintain a leader board, and the top three entries at the end of the competition (due date) will receive 10 bonus points. The ranking is based on classification accuracy in your PDF submission file. You must use HCRF. You can use different feature types or different number of hidden states. You can use all training data or combine train and validation data for training. You can also increase the number of iterations of BFGS or use different optimization algorithms.

7. (10 bonus points) Can you handle all 20 classes? Submit `predTestLabels_20classes.csv` and enter our Kaggle competition for fame and bonus point. The rules and restrictions for item 4 above also apply here.

4 What to submit?

4.1 Blackboard submission

You will need to submit both your code and your answers to questions on Blackboard. Do not submit the provided data. Put the answer file and your code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_heeyoung_kwon*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SUBID_FirstName_LastName.zip. The answer file should be named: answers.pdf, and it should contain:

1. Answers to Question 1 and 2
2. Answers to Question 3.5

4.2 Kaggle submission

For Questions 3.5.5, 3.5.6, and 3.4.7, you must submit a CSV file to get the classification accuracy from the competition site `inclass.kaggle.com/c/hw4-gesture-recognition-3classes` and `inclass.kaggle.com/c/hw4-gesture-recognition-20classes`. A submission file should contain two columns: ImgId and Prediction. The file should contain a header and have the following format.

$$\begin{array}{cc} ID, & Class \\ 1, & 3 \\ 2, & 1 \\ \ldots & \ldots \end{array} \tag{25}$$

A sample submission file is available from the competition site and our handout.

5 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You must use your SBU ID as your file name for the competition. Do not fake your Stony Brook ID to bypass the submission limitation per 24 hours. Doing so will be considered cheating.

References Cited

[1] S. B. Wang, A. Quattoni, L. P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.