CSE 515, Statistical methods in CS, Spring 2013: Assignment 1
Due: Friday, 26th April, 11:59am
HMMs and CRFs

For this assignment, you will turn-in a write-up answering the questions below *and* all your code via catalyst dropbox `https://catalyst.uw.edu/collectit/dropbox/dvij/26655`.

Task description. The task we will consider in this assignment is optical character recognition (OCR). The dataset we provide consists of a sequence of words, one character per row.[1] The very first character of each word was capitalized in the original data and has been omitted for simplicity. The format of the data is described in `generate_hmm_plots.m`, so please look through that file before continuing. This file provides sample code structure to generate plots.

1 Programming: Learning/Inference in HMMs/CRFs [80 points]

In this problem you will implement maximum likelihood estimation and the forward-backward algorithm for Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs).

1. **Generative Models** **[40 points]**
   We will start with generative models that model the joint distribution of the observations (character images) and class labels (character names): HMMs and Naive Bayes fall into this category. Let X_t denote the t-th letter in a word and O_t^k the value of the k-th pixel for the t-th character.

   (a) **Parameter Estimation (MLE/MAP) in HMMs**: For this first part, you will set the parameters of the HMM using maximum likelihood and maximum a posteriori estimation using several values of pseudo-count/hyperparameter α. The result should be a stationary model (one that does not depend on t), i.e., you should have a single distribution $p(X_1)$, a single CPT $p(X_t|X_{t-1})$ and 64 CPTs $p(O_t^k|X_t)$ (one for each pixel).
   Your task is to fill in the missing code in the files `hmm_learn.m`. Note that `hmm_learn.m` goes over the specifics of what parameters you need to learn. Because we will be comparing HMM to Naive Bayes later, `hmm_learn.m` should also fit a probability model $p(X_t)$ which serves as the class prior for Naive Bayes.
   To help debug your code and generate results, `generate_hmm_plots.m` will plot the transition model that you learn, and the observation model for the letter 'a'. You should see that the transition model "makes sense", e.g. $p(X_t = u \mid X_{t-1} = q)$ should be high, and that the observation model looks like a blurry version of the desired letter.

   (b) **The Forward Backward Algorithm**: In this part, you will implement the forward-backward algorithm for HMMs and compare its performance to a Naive Bayes approach which classifies each character independently of all others. You have two programming tasks for this sub-part.

      i. **[15 points]** `hmm_fb.m` - In this file, you will implement the forward-backward algorithm to compute marginal probabilities $P(X_t \mid O_1, \ldots, O_T)$. The input to this function is the trained model from `hmm_learn.m` and the pixel data corresponding to a *single word* (not the entire test set). See the file `generate_hmm_plots.m` to see how `hmm_fb.m` is used.

[1] This dataset is a modified version of the dataset at `http://www.seas.upenn.edu/~taskar/ocr/`, which has been subsampled and slightly altered to simplify the processing.

ii. **[5 points]** `generate_hmm_plots.m` - Run this file to train the HMM model and evaluate it on the test data. Naive Bayes will serve as a baseline, but one critical line of code is missing in this file. Remember that Naive Bayes computes,

$$P(X_t \mid O_t) \propto P(O_t \mid X_t)P(X_t),$$

and that $P(X_t)$ was computed in `hmm_learn.m`, and $P(O_t \mid X_t)$ was computed in `hmm_fb.m`. You need to fill in this line before the code will run.

What to include in the write up: Try several values of the smoothing/pseudo-counts: $\alpha = 0, 1, 2, 4, 8$ and include the plots for the resulting observation model for 'a' and the transition model in the write-up. Describe in 1-2 sentences the effect of smoothing. Include a plot of accuracy on the test set vs. smoothing parameter α for HMM and NB. Next, **discuss** (3-4 sentences) how the two algorithms differ in performance, what their performance and errors are, and how/why they differ.

2. Discriminative Models **[40 points]**

Now, we move to discriminative models that only model the conditional likelihood of the observations given the class labels. Logistic Regression and CRFs fall in this category. Note that since we don't model the distribution over observations, it is easy to "featurize" these models: Work with features of the observations rather than the raw observations themselves. In this assignment, we will implement a simple featurization - we will consider features of the form $f(O, i) \in \mathbf{R}^{64\times 26}$

$$f(O,i)_{l,m} = \begin{cases} m^{th} \text{ pixel of } O & \text{if } i = l \\ 0 \text{ otherwise} \end{cases}.$$

For numerical stability, we will discard features whose sum over the dataset is small (the number of occurrences). The file `process_data.m` does this for you to construct goodFeatures, a logical 64×26 matrix storing the indices of the features that have non-negligible counts over the dataset.

Note that parameter estimation in these models is *not* closed form, but rather a convex optimization problem that needs to be solved numerically. We will use the minFunc package to perform numerical optimization (the package is included in the codebase given to you and just requires you to compute objective and gradient). Make sure that minFunc and its subfolders compiled and autoDif are in your MATLAB path when you are working on this part of the assignment (you will need to unzip `minFunc_2012.zip` and run the script `addpaths.m`).

(a) **Logistic Regresssion**: In this part, you will train a multiclass logistic regression model to classify images of single characters. The model is given by

$$P(X_t = i|O_t, \theta) = \frac{\exp\left(\langle \theta, f(O_t, i)\rangle\right)}{\sum_j \exp\left(\langle \theta, f(O_t, j)\rangle\right)}.$$

The file `logreg_learn.m` provides skeleton code for learning a logistic regression model. The objective is log-likelihood normalized by the total number of letters (not words), and regularized using L_2 norm:

$$\frac{1}{n}\log P(D|\theta) + \frac{\lambda}{2}||\theta||_2^2$$

.

You will need implement the function `infer_likelihood` to perform inference in the logistic regression model (compute log-likelihood of the observed class label given the features, and the gradient of this log-likelihood with respect to the model parameters $\{\theta\}$.

(b) **Conditional Random Fields (CRFs)**: Here you will train a CRF for the same task. Note that now the examples in the dataset are words (sequences of character labels and images). The CRF will use features of the form $f(X_t, X_{t+1}) \in \mathbf{R}^{26\times 26}$ to capture correlations between adjacent characters in a word (in addition to the ones used by logistic regression):

$$f(i,j)_{l,m} = \begin{cases} 1 & \text{if } i = l, j = m \\ 0 & \text{otherwise} \end{cases}$$

Under a CRF, the probability of observing a given sequence of characters $X_{1:T}$ given corresponding images $O_{1:T}$ is

$$P(X_{1:T}|O_{1:T}) \propto \exp\left(\sum_{t=1}^{T} \langle \theta_n, f(O_t, X_t) \rangle + \sum_{t=1}^{T-1} \langle \theta_e, f(X_t, X_{t+1}) \rangle \right)$$

The file `crf_learn.m` provides skeleton code for learning a CRF model. You will need to fill in the functions `infer_likelihood, crf_infer` in the file `crf_learn.m` to perform inference in the CRF (compute log-likelihood of the observed sequence of class images given the sequence of character images) , and the gradient of this log-likelihood with respect to the model parameters θ_n, θ_e.

What to include in the write up: Try several values of the regularization (the range $\lambda = [10^{-4}, .1]$ seems good for this dataset) and include the plots for the resulting node potential for 'a' and the edge potential in the write-up. Describe in 1-2 sentences the effect of smoothing. Include a plot of accuracy on the test set vs. smoothing parameter λ for CRF and log-reg (the file `generate_crf_plots.m` does generates these plots once you complete the inference code).

2 HMM Inference Written Questions [20 points]

1. In this question, you will derive how to compute the marginal probability of two adjacent states for HMMs, $P(x_{j-1}, x_j \mid o_1, \ldots, o_T)$.

   (a) **[10 points]** Prove that forward-backward as defined in class notes provides an easy way to compute $P(x_{i-1}, x_i, o_{1..T})$ from α and β as follows:

   $$P(x_{j-1}, x_j, o_{1..T}) = \alpha_{j-1}(x_{j-1}) P(x_j \mid x_{j-1}) P(o_j \mid x_j) \beta_j(x_j),$$

   where
   $$\alpha_{j-1}(x_{j-1}) = P(x_{j-1}, o_{1..j-1}), \qquad \beta_j = P(o_{j+1..T} \mid x_j).$$

   You must state explicitly which conditional independence properties of HMMs you use in each step of your proof. (For example, one useful conditional independence property of HMMs is that $X_j \perp O_{1..j-1} \mid X_{j-1}$.)

   Reminder: Directed Separation Theorem – Variables X and Y are independent given $\mathbf{Z}$ if there is no active trail between X and Y when $\mathbf{Z}$ variables are observed. The same theorem applies for sets of variables $\mathbf{A}, \mathbf{B}$: if there are no active trails between any pairs of variables $(X \in \mathbf{A}, Y \in \mathbf{B})$ when $\mathbf{Z}$ are observed, then $(\mathbf{A} \perp \mathbf{B} \mid \mathbf{Z})$.

   (b) **[2 points]** Given that you have run forward-backward already, what is the complexity (in Big-O notation) of computing a single *conditional* $P(x_{i-1}, x_i \mid o_{1..T})$. Use K to represent the number of possible values of a hidden state.

2. Suppose we have a regular HMM of length T over variables $X_1, \ldots, X_T, O_1, \ldots, O_T$, but we observe only the even Os, that is: $O_2, O_4, \ldots, O_T$ (assuming T is even). We want to compute $P(O_1)$ using variable elimination.

   (a) **[4 points]** Write down a variable elimination ordering that achieves linear time complexity in T.

   (b) **[4 points]** Now write down a variable elimination ordering that achieves exponential time complexity in T.