CPTS 577 Structured Prediction and Intelligent Decision-Making, Spring 2019
Homework #2
Due Date: Mar 26

NOTE 1: Please use a word processing software (e.g., Microsoft word or Latex) to write your answers and submit a printed copy to me. The rationale is that it is sometimes hard to read and understand the hand-written answers. Thanks for your understanding.

NOTE 2: Please ensure that all the graphs are appropriately labeled (x-axis, y-axis, and each curve). The caption or heading of each graph should be informative and self-contained.

1. (**5 points**) Prove that if the learned recurrent classifier makes mistakes with a small probability ϵ at each decision step, the total number of mistakes grows qudratically $O(\epsilon T^2)$ instead of $O(\epsilon T)$, where T is the number of decision steps.

2. (**25 points**) Present all the imitation learning algorithms (exact-imitation, forward training, SEARN, DAgger, AggreVaTe) in a unified framework. Show how to instantiate this framework to get each of the imitation learning algorithms.

3. (**10 points**) Suppose we are dealing with a sequence labeling problem, and some of the labels are missing in the training data. How will you deal with this setting while applying imitation learning approaches in general? Specifically, how will you extend DAgger and AggreVaTe algorithms to handle the missing label setting?

4. (**60 points**) Please implement the recurrent classifier training approach using *exact imitation* and *DAgger* algorithms for sequence labeling problems and experiment with two sequence labeling datasets: handwriting recognition, and text-to-speech mapping.

   In a sequence labeling problem, the structured input $x = (x_1, x_2, \cdots, x_T)$ is a sequence of input tokens, where each input token x_i is represented as a m-dimensional feature vector; and the structured output $y = (y_1, y_2, \cdots, y_T)$ is a sequence of output labels, where each output label y_i comes from a label set $\{1, 2, \cdots, k\}$. You were provided with a set of training examples $\mathcal{D} = \{(x, y^*\}$, where y^* is the correct structured output for the structured input x.

   You need to learn a linear classifier that maps a state (structured input and partial structured output) to a label in $\{1, 2, \cdots, k\}$. You can try different feature choices as you see fit. For example, you can employ the same features as your first homework. Alternatively, you can use a much simpler representation: features of the input token for which you want to make a prediction PLUS a small history (say H) of previous labels.

   You will evaluate your classifiers using the hamming error, i.e., error over individual labels.

   (a) Learn a recurrent classifier via exact imitation algorithm on the training data; use the label of the previous character as a context in addition to the features of the current input token. (1) Compute the recurrent error of the classifier on the testing data; and (2) Compute the oracle error (IID error) of the learned classifier on the testing data, i.e., error of the learned classifier with correct (oracle) context.

Algorithm 1 Recurrent Classifier Learning via Exact Imitation

Input: $\mathcal{D}$ = Training examples
Output: h, the recurrent classifier

Initialize the set of classification examples $\mathcal{L} = \emptyset$
for each training example $(x, y = y_1 y_2 \cdots y_T) \in \mathcal{D}$ **do**
  for each search step $t = 1$ to T **do**
    Compute features f_n for search node $n = (x, y_1 \cdots y_{t-1})$
    Add classification example (f_n, y_t) to $\mathcal{L}$
  end for
end for
$h =$**Classifier-Learner**$(\mathcal{L})$ *// learn classifier from all the classification examples*
return learned classifier h

(b) Learn a recurrent classifier via DAGGER algorithm on the training data for different values of β (interpolation parameter): 0.5 to 1 in the increments of 0.1, for a fixed number of iterations (say 5) with an exponential decay over iterations. Compute the recurrent error of the classifier on the training data and testing data over different iterations of the DAGGER algorithm.

Algorithm 2 Recurrent Classifier Learning via DAGGER

Input: $\mathcal{D}$ = Training examples, d_{max} = dagger iterations
Output: $\mathcal{H}$, the heuristic function

Initialization: $\mathcal{L} = \mathcal{L}_{ei}$ *// classification examples from Exact imitation*
$\hat{\mathcal{H}}_1 =$**Classifier-Learner**$(\mathcal{L})$
for each dagger iteration $j = 1$ to d_{max} **do**
  Current Policy: $\mathcal{H}_j = \beta_j \mathcal{H}^* + (1 - \beta_j)\hat{\mathcal{H}}_j$ *// $\mathcal{H}^*$ is the oracle classifier*
  for each training example $(x, y = y_1 y_2 \cdots y_T) \in \mathcal{D}$ **do**
    for each search step $t = 1$ to T **do**
      Compute features f_n for search node $n = (x, \hat{y}_1 \cdots \hat{y}_{t-1})$
      if $\mathcal{H}_j(f_n) \neq \mathcal{H}^*(f_n)$ **then**
        Add classification example (f_n, y_t) to $\mathcal{L}$
      end if
      Predict the output label using current heuristic: $\hat{y}_t = \mathcal{H}_j(f_n)$
    end for
  end for
  $\hat{\mathcal{H}}_{j+1}=$**Classifier-Learner**$(\mathcal{L})$ *// learn classifier from aggregate data*
end for
return best classifier $\hat{\mathcal{H}}_j$ on the validation data

(c) Draw one plot for recurrent error of the classifier on training data and another on the testing data. On x-axis, we have iteration number of the DAGGER algorithm. On y-axis, we have the error-rate of the classifier. One curve for each interpolation parameter β value.

Please use a linear classifier (e.g., Perceptron, Linear SVM) from an existing machine learning software library (Weka, LibSVM, Vowpal Wabbit, scikit-learn etc.).

(d) Please explore the L2S system as part of the Vowpal Wabbit library (http://hunch.net/~l2s) for debugging and testing your code in addition to get yourself familiar with that system for future use. See https://arxiv.org/pdf/1406.1837.pdf for more information.