CS 4780/5780 Machine Learning Due November 19, 2013

Assignment 3: HMM & Statistical Learning Theory

Instructor: Thorsten Joachims

Course Policy:

Read all the instructions below carefully before you start working on the assignment, and before you make a submission

- Assignments are due at the beginning of class on the due date in hard copy.
- Write your NetIDs with submission date and time on the first page of the hard copy.
- No assignment will be accepted after the solution is publicized (about 4 days after due)
- The submission time of whatever you submit last (hard copy or CMS) is counted as the official submission timeand will determine the late penalty
- Upload only the code you wrote to CMS. Do not upload any additional libraries. Provide a README file with your submission that includes a list of all libraries and instructions needed to run your code.
- Late assignments can be submitted in class or to Ian Lenz in Upson Hall 5151. Since the fifth floor of Upson is locked on the weekends, weekend submissions (all code and answers to all questions) should be made digitally via CMS, with a hard copy delivered to Ian as soon as possible afterwards.
- All sources of material must be cited. Assignment solutions will be made available along with the graded homework solutions. The University Academic Code of Conduct will be strictly enforced, including running cheating detection software.
- No more than one submission per group.

Problem 1: Viterbi Algorithm [50 points]

We learned **Hidden Markov Models (HMMs)** as a generative model, and the **Viterbi Algorithm** is used to compute the most likely configuration of the hidden variables. HMMs are useful for various tasks such as *Speech Recognition, Machine Translation, and Signal Encoding*, and consequently the Viterbi algorithm is highly important technique in Machine Learning. In this assignment, we will investigate a code translation problem that can also be solved by the Viterbi algorithm.

In the beginning of the semester, Ian created a simple linguistic code that utilizes several Greek alphabets in order to safely encrypt his password. After a few months, he forgets the original English password, and only remembers the Greek encryption. He believes that strong similarity between English and Greek alphabets was incorporated for his encryption, whereas there was no deterministic mapping between English and Greek characters. Thus we hypothesize that sequences of Greek characters correspond to English words and plan to build an HMM to help him decipher his password.

Formally speaking, we represent a Greek word as $\mathbf{x}=(x_1, x_2...x_T)$, where $x_t \in \{\alpha, \tau, \eta, \gamma, \omega\}$ is the t^{th} character in the word (represented in Greek script). Given each Greek word, our goal is to predict the most probable English word $\mathbf{y}=(y_1, y_2...y_T)$, where $y_t \in \{a, i, p, s\}$ is the English translation of the t^{th} character (represented in English script). The following tables give the transition and emission probabilities of our HMM.

	a	**i**	**p**	**s**
a	0.05	0.1	0.15	0.7
i	0.1	0.05	0.25	0.6
p	0.45	0.15	0.05	0.35
s	0.35	0.2	0.15	0.3
START	0.1	0.4	0.2	0.3

Table 1: Transition Probabilities $P(y_t|y_{t-1})$ where y_{t-1} on each row, y_t on each column

	α	τ	η	γ	ω
a	0.4	0.2	0.1	0.2	0.1
i	0.3	0.1	0.4	0.1	0.1
p	0.1	0.1	0.1	0.2	0.5
s	0.1	0.4	0.1	0.3	0.1

Table 2: Emission Probabilities $P(x_t|y_t)$ where y_t on each row, x_t on each column

In HMMs, the transition probability $P(y_t|y_{t-1})$ decides the probability of the current English character given the previous English character, whereas the emission probability $P(x_t|y_t)$ determines the probability of a Greek translation given a English character. Based on these two model probabilities, we can compute the most likely English translation of each Greek word $\mathbf{x}=(x_1, x_2...x_T)$ via the following HMM formula:

$$\texttt{argmax}_{y_1,y_2...y_T}\ P(y_1, y_2...y_T|x_1, x_2...x_T) = \texttt{argmax}_{y_1,y_2...y_T}\ P(y_1)P(x_1|y_1)\prod_{t=2}^{T} P(x_t|y_t)P(y_t|y_{t-1})$$

Using the above equation and the Viterbi algorithm, answer the following questions:

(a) [**5 points**] Let $\delta_{s,t}$ be the probability of the most probable English sequence corresponding to the first t Greek observations $(x_1, x_2, ..., x_t)$ that end with the English character s. Write the recurrence relation of $\delta_{s,t}$ in terms of model probabilities[1] and specify the initial conditions[2] for 2D dynamic programming ($s \in \{a, i, p, s\}$, $1 \leq t \leq T$)

[1] We have three probabilities: transition, emission, and start probabilities given in the Table 1 & 2

[2] To perform dynamic programming, you have to initialize the base cases at $t = 1$ for all $s \in \{a, i, p, s\}$

(b) **[25 points]** Ian's encrypted password is "$\omega\alpha\gamma\tau$ $\eta\tau$ $\gamma\alpha\omega$". What are the most likely English translations for each of the Greek encrypted words: 1) $\eta\tau$, 2) $\gamma\alpha\omega$, 3) $\omega\alpha\gamma\tau$? For each Greek word, all you have to do is to fill the 2D dynamic programming table[3] whose entries are $\delta_{s,t}$ with specifying the backtracking path that corresponds to the most probable translation. What is indeed Ian's English password?

(c) **[10 points]** Suppose English has m characters and Greek has n characters in their alphabets. What is the running time of the translation from a Greek word of length T into an English word via the Viterbi algorithm? Compare it to the running time of the brute-force algorithm naively considering all possible combinations in terms of big-O complexity. (You could assume reading the model probabilities takes only a constant time)

(d) **[5 points]** Given the transition and emission probabilities of a first-order HMM model, describe how to compute the probability of a Greek word $\mathbf{x}=(x_1, x_2 ... x_T)$. (*i.e.,* $P(X = (x_1, x_2, ..., x_t))$) You have to clearly specify the equations you formulate for calculating this probability as well as a precise 2-4 sentence description of how your algorithm would work. (*note*: Solutions with exponential time complexity will get the full credits, but a clear description of sub-exponential time algorithm will get the bonus points)

(e) **[5 points]** We can similarly utilize HMMs to translate sentences of one language into another. In this sentence-level translation, each character in our problem corresponds to a word, and a sequence of characters corresponds to a sentence. How well would this HMM-based approach perform as compared to the state-of-the-art (like Google Translate) for languages common today, *say* English and Spanish? Briefly explain why you think so in 2-3 sentences.

Problem 2: Statistical Learning Theory [50 points]

For the questions below, clearly explain all steps in your derivations. You may use any result proved in class.

(a) Find the tightest lower bound that you can on the VC dimension for each of the following hypotheses classes. Present the sets of points that can be shattered for each of the following:

- **Intervals of the reals** For an instance space in $\mathbb{R}$, let H be the set of all intervals on the real line.
- **Pairs of intervals of the reals** For an instance space in $\mathbb{R}$, let H be the set of all pairs of intervals on the real line.

[3] Conventionally s will vary on the row side and t will vary on the column side in the table.

- **Rectangular hypotheses centered at the origin** For an instance space in $\mathbb{R}^2$, let $h \in H$ be a hypothesis: $h(x) = \mathbb{1}\{-a < x_1 < a, -b < x_2 < b\}$ for $a, b \in R$.
- **Rectangular hypotheses centered arbitrarily** For an instance space in $\mathbb{R}^2$, let $h \in H$ be a hypothesis: $h(x) = \mathbb{1}\{a < x_1 < b, c < x_2 < d\}$ for $a, b, c, d \in R$.
- **Circular hypotheses centered arbitrarily** For an instance space in $\mathbb{R}^2$, let $h \in H$ be a hypothesis: $h(x) = \mathbb{1}\{||x - c||_2 < r\}$ for $r \in R, c \in R^2$

(b) Let $\{0, 1\}^n$ be the instance space of a binary classification task. This means that an instance x consists of n binary-valued features. Let H_n be the class of all boolean functions over the input domain. What is $|H_n|$ and the VC dimension of H_n? Show all work leading you to your answer.

(c) Now, suppose we are interested in a binary classification task, where our examples $\mathbf{x}$ are all 100-dimensional binary vectors of the form $\mathbf{x} = (x_1, x_2, \ldots, x_{100})$, where each $x_i \in \{0, 1\}$. As above, we want to learn a binary hypothesis h over the instances. Consider a "restricted" linear hypothesis h that can be described using a 100-dimensional weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_{100})$ and a bias b, where all components of the weight vector are binary, i.e. $\forall i : w_i \in \{-1, 1\}$ and the bias in an integer between -15 and 15: $b \in \{-15, -14, \ldots, 0, \ldots, 14, 15\}$. The classification rule for classifying an example x is $\text{sign}(\mathbf{w}^T\mathbf{x} + b)$.

The hypothesis space H consists of all such $\mathbf{w}, b$. Given a training set S with $|S| = n$ examples, and a hypothesis with 0 training error, give a bound for the prediction error of this hypothesis that will hold with probability $1 - \delta$ in terms of δ and n.

(d) For H defined as above, let $\hat{\text{h}}_S = \text{argmin}_{h \in H} Err_S(h)$. How large must the training set size $|S|$ be for $P\left(|Err_S(\hat{\text{h}}_S) - Err_P(\hat{\text{h}}_S)| \geq 0.1\right) \leq 0.1$?