CS 4780/5780 Machine Learning Due Nov 15, 2011

Assignment 5: Generative Models & Learning Theory

Instructor: Thorsten Joachims *Total Points: 100*

Course Policy: Assignments are due at the beginning of class on the due-date in hard copy. Late assignments lose 5 points for each late day and can be submitted directly to the TAs. Assignments submitted after solutions are made available will not be accepted. Groups of at most two are allowed. 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.

Problem 1: Viterbi Algorithm [60 points]

You learned about **Hidden Markov Models (HMMs)** in class, and how the **Viterbi Algorithm** can be used to estimate the most likely configuration of the hidden variables. HMMs are used for a number of different tasks including *Speech Recognition, Machine Translation & Signal Encoding*. Consequently the Viterbi algorithm is one of the most important algorithms in Machine Learning. In this assignment we will look at a machine translation problem:

Scholars have discovered an ancient language called *Emelic*. They have found ancient writings in this language (which utilizes the Greek alphabet) and would like to decipher these writings. They believe that there is a strong similarity between English and Emelic and thus would like to translate to English. While there is no deterministic mapping between Emelic and English characters, we believe that sequences of Emelic characters are related to sequences of English characters. Thus we would like to use a HMM to help in this translation task.

More formally, we represent an emelic word as $\mathbf{x} = (x_1, x_2 ... x_k)$, where $x_i \in \{\alpha, \tau, \eta, \gamma, \omega\}$ is the i^{th} character in the word (represented in the Greek script). Given an emelic word we would like to predict the corresponding English word[1] $\mathbf{y} = (y_1, y_2 ... y_k)$, where $y_i \in \{a, n, o, t\}$ is the English translation of the i^{th} character (represented in the Latin script).

As required for HMMs we have come up with the transition probabilities $P(y_i|y_{i-1})$ (given in Table 1) which corresponds to the probability of what the next (English) character will be, given the current character. We also have the table of emission probabilities given in Table 2, which corresponds to the probability of an emelic character being the translation of a given English character. Using these, we can compute what is the most

[1] Do not worry if the words produced make sense or not

	a	**n**	**o**	**t**
a	0.05	0.1	0.25	0.6
n	0.35	0.05	0.5	0.1
o	0.1	0.5	0.1	0.3
t	0.4	0.1	0.4	0.1
START	0.1	0.4	0.2	0.3

Table 1: Transition Probabilities $P(y_i|y_{i-1})$ where y_i varies with columns and y_{i-1} varies with the row

	α	τ	η	γ	ω
a	0.4	0.2	0.1	0.2	0.1
n	0.3	0.1	0.4	0.1	0.1
o	0.1	0.1	0.1	0.2	0.5
t	0.1	0.4	0.1	0.3	0.1

Table 2: Emission Probabilities $P(x_i|y_i)$ where x_i varies with columns and y_i varies with the row

likely English translation of an Emelic word $\mathbf{x} = (x_1, x_2...x_k)$, via the HMM formula:

$$\texttt{argmax}_{y_1,y_2...y_k}\ P(y_1, y_2...y_k|x_1, x_2...x_k) = \texttt{argmax}_{y_1,y_2...y_k}\ P(y_1)P(x_1|y_1)\prod_{i=2}^{i=k} P(x_i|y_i)P(y_i|y_{i-1}) \tag{1}$$

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

1. **Most likely Translations:** (35 points) What are the most likely English translations for each of the following three emelic words: a)α, η, b) τ, ω, γ and c) $\gamma, \alpha, \omega, \eta$? You do not need to show all calculations, just the table of probabilities for partial paths suffices[2].

2. **Complexity:** (10 points) What is the complexity of translating an emelic word of length k into English using the Viterbi algorithm, if the English vocabulary was of size m and the Emelic vocabulary was of size n. You can assume that looking up the translation and emission probabilities is a constant time operation. Answer in terms of big-O notation. How does this compare to the naive, brute-force algorithm (in terms of complexity)?

3. **Probability of an observation:** (10 points) We now want to compute the probability of seeing a particular Emelic word, given the HMM. Given the transition and emission probabilities, clearly describe how you can compute the probability

[2]For consistency we recommend reporting the values via tables in a format similar to Table 1

of an Emelic word **x** *i.e* $P(x_1, x_2...x_k)$. You need to clearly specify the equations you are using to calculate this probability, as well as a precise 2-4 sentence description of how your algorithm would work. **NOTE**: Solutions with exponential time complexity are fine.

4. **Better translation model:** (5 points) We can similarly use HMMs to translate sentences of one language into another. This would involve replacing characters with words in the above formulation. How well do you think this HMM-based approach, for sentence-level translation, would do 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 [40 points]

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

1. **Using Restricted Linear Classifier:** (10 points) Suppose we are interested in a binary-classification task, where our examples x are all 100-dimensional binary vectors of the form $x = (x_1, x_2, ...x_{100})$ where each $x_i \in \{0, 1\}$. We want to be able to classify each example x into either of the two classes *i.e.* we want to learn a hypothesis $h : X \to \{0, 1\}$. As a first attempt we are using a (biased) linear hypothesis, which can be described using a 100-dimensional weight vector $\vec{w} = (w_1, w_2...w_{100})$ and a bias b, where all components of the weight vector are binary *i.e.* $\forall i : w_i \in \{0, 1\}$ and the bias is an integer between 0 and 20: $b \in \{0, 1, ...20\}$. The classification rule for classifying an example x is $sign(w^T x + b)$.

   The hypothesis space H consists of all such $\vec{w}, b$. 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.

2. **Using Unrestricted Linear Classifier:** (10 points) Suppose we remove the previous restriction we placed on w_i and b, *i.e.* we allow them to take all possible value in $\mathbb{R}$. Rederive the bound for the prediction error of this hypothesis (with 0 training error) that will hold with probability $1 - \delta$ in terms of δ and n.

3. **Training set size:** (5 points) For **H** as defined in part **1**, let $\hat{h}_S = \arg\min_{h \in H} Err_S(h)$. How large must the training set size (*i.e.* $|S|$) be for $P\Big(Err_S(\hat{h}_S) - Err_P(\hat{h}_S) \geq 0.1\Big) \leq 0.1$?

4. **Using Spherical classifiers:** (15 points) We will now use a spherical classifier instead of a linear classifier (in a d-dimensional space). Such a classifier can be defined by a centre $\vec{c} = (c_1, c_2, ..., c_d)$ and a radius R. The classification rule for

classifying an example x is $sign(R - ||x - \vec{c}||)$, where $||x - \vec{c}||$ is the distance of the point from the centre of the sphere. In other words if the point is within the sphere it is classified as positive else it is classified as negative.

(a) Prove that the VC-Dimension for these spherical classifiers in this d-dimensional space is at least d. If you are using a constructive proof, then clearly explain the example set you are using as well as your construction to shatter the set in all possible ways.

(b) Prove that the VC-Dimension for these spherical classifiers in this d-dimensional space is no more than $2d + 1$. HINT: Try to show that spherical classifiers are no more powerful than linear classifiers (in a higher-dimensional space).