Probabilistic Graphical Models, Spring 2012

Problem Set 6: EM and Structured Prediction

Due: Thursday, May 3, 2012 at 5pm (in class)

---

1. **HMM with mixture model emissions**.

   A common modification of the hidden Markov model involves using mixture models for the emission probabilities $p(\mathbf{y}_t|q_t)$, where q_t refers to the state for time t and $\mathbf{y}_t$ to the observation for time t.

   Suppose that $\mathbf{y}_t \in \mathbb{R}^n$ and that the emission distribution is given by a mixture of Gaussians for each value of the state. To be concrete, suppose that the q_t can take K discrete states and each mixture has M components. Then,

   $$p(\mathbf{y}_t \mid q_t) = \sum_{j=1}^{M} b_{q_t j} \left(\frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_{q_t j}|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} (\mathbf{y}_t - \mu_{q_t j})^T \Sigma_{q_t j}^{-1} (\mathbf{y}_t - \mu_{q_t j}) \right\} \right)$$

   where $\mathbf{b}_i \in [0,1]^M$ denotes the mixing weights for state i ($\sum_{j=1}^{M} b_{ij} = 1$ for $i = 1, \ldots K$), $\mu_{ij} \in \mathbb{R}^n$ and $\Sigma_{ij} \in \mathbb{R}^{n \times n}$.

   Let $\pi \in \mathbb{R}^K$ be the probability distribution for the initial state q_0, $A \in \mathbb{R}^{K \times K}$ the transition matrix of the q_t's. In this problem you will derive an EM algorithm for learning the parameters $\{b_{ij}, \mu_{ij}, \Sigma_{ij}\}$ and A, π.

   (a) The EM algorithm is substantially simpler if you introduce auxiliary variables $z_t \in \{1, \ldots, M\}$ denoting which mixture component the t'th observation is drawn from.

   Draw the graphical model for this modified HMM, identifying clearly the additional latent variables that are needed.

   (b) Write the expected complete log likelihood for the model and identify the expectations that you need to compute in the E step. *Show all steps of your derivation.*

   (c) Give an algorithm for computing the E step.

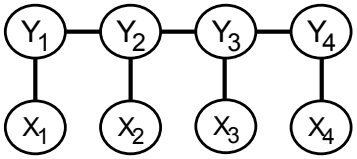
   Hint: Reduce the inference problem to something you know how to do, such as sum-product belief propagation in tree-structured pairwise MRFs.

   (d) Write down the equations that implement the M step.

2. **Structured prediction for part-of-speech tagging**

   In this question, you will experiment with structured prediction using the averaged perceptron algorithm on a chain-structured conditional random field (CRF). You will tackle the task of part-of-speech (POS) tagging, a problem from the natural language processing domain. POS tagging is a classification problem where the goal is to accurately predict the part of speech (e.g., noun, verb, adjective) of each word in a sentence.

   The CRF used in this prediction task is a Markov model. Let L_i be the length of sentence i. Then, the CRF for sentence i has variables $Y_1, \ldots, Y_{L_i}$, where Y_l is a discrete variable denoting the part-of-speech of token l. The tokens are denoted by the variables $X_1, \ldots, X_{L_i}$. The CRF for a sequence of length 4 is shown below.



Raw POS tagging data takes the form of a set of sentences and tag sequences. Each token (normally a word but sometimes a number or punctuation symbol) in each sentence is associated with exactly one tag. Standard POS tag sets typically include around $C = 40$ distinct tags. For the purpose of this assignment, we have prepared data with a simplified tag set consisting of $C = 10$ groups of tags. Each token in the data set is assigned one of these 10 tags. To learn prediction models for the POS tagging task, we need to select a feature space to represent the tokens. You will use simple features that we have extracted from the tokens for you.

The data consist of a set of training sentences *train-i.txt* and a set of test sentences *test-i.txt*. Each file contains one sentence. Each row in each file contains the raw token, the tag ID for that token, and five feature values in standard comma-separated-value (CSV) format. The format of each row is as follows: *<Token>, <TagId>, <Bias>, <InitCap>, <AllCap>, <Pre>, <Suff>*. A description of the fields is given below.

Column	Field	Description	Value
1	Token	The raw token string	Any string
2	TagID	The ID of the tag	$\{1, ..., 10\}$
3	Bias	Feature: Bias	1
4	InitCap	Feature: Initial Capital	$\{0, 1\}$
5	AllCap	Feature: All Capitals	$\{0, 1\}$
6	Pre	Feature: Prefix ID	$\{1, ..., 201\}$
7	Suff	Feature: Suffix ID	$\{1, ..., 201\}$

The 10 tag ID's correspond to the following 10 tag groups: verb, noun, adjective, adverb, preposition, pronoun, determiner, number, punctuation, and other. The Bias feature is a constant 1 for all tokens. The InitCap feature is 1 if the token string starts with a capital letter and 0 otherwise. The AllCap feature is 1 if the token string is all capital letters and 0 otherwise. The Pre feature takes one of 201 values corresponding to the most common two-character token string prefixes. The Suff feature takes one of 201 values corresponding to the most common two-character token string suffixes.

For each word l we have a feature vector $\mathbf{x}_l$ of dimension 5 (corresponding to rows 3–7 in the table), where x_{la} takes values in the set V_a (given in the last column of the table). The CRF model has one parameter w^A_{cav} for each of part-of-speech tag c, feature a, and feature value v (note that different features a have different numbers of values as given by the set V_a). These parameters encode the compatibility between feature values and class labels. The CRF also has one transition parameter $w^T_{cc'}$ for each pair of labels c and c'. The transition parameters encode the compatibility between adjacent class labels in the sequence. All of the parameters can take arbitrary (positive or negative) real values. To be clear, there are exactly $(1 + 2 + 2 + 201 + 201) \cdot 10 + 10^2 = 4170$ parameters to be learned.

The log-potentials are then given by:

$$\begin{aligned} \theta_l(y_l, \mathbf{x}_l) &= \sum_{a=1}^{5} w^A_{y_l, a, x_{la}} \\ \theta_{l,l+1}(y_l, y_{l+1}) &= w^T_{y_l, y_{l+1}} \end{aligned}$$

Given a new sentence of length L, we predict its part-of-speech tagging by MAP inference in this CRF,

$$\text{Predict\_POS}(\mathbf{x}; \mathbf{w}) = \arg\max_{\mathbf{y}} \sum_{l=1}^{L} \theta_l(y_l, \mathbf{x}_l) + \sum_{l=1}^{L-1} \theta_{l,l+1}(y_l, y_{l+1})$$

Since the CRF is chain-structured, MAP inference can be performed in linear time using variable elimination or max-sum belief propagation.

Let $\mathbf{w}$ denote weight vector (i.e., of dimension 4170), and let $\mathbf{f}(\mathbf{x}, \mathbf{y})$ be the feature vector (the sufficient statistics) for the CRF, such that the joint distribution is given by:

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{x}; \mathbf{w})} \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}$$

The *averaged* structured perceptron algorithm is given as follows:

1 **Input:** Training examples $(\mathbf{x}_i, \mathbf{y}_i)$
2 **Initialization:** Set $\mathbf{w} = 0$, $\overline{\mathbf{w}} = 0$
3 For $t = 1, \ldots, T$
4 For $i = 1, \ldots, N$
5 $\hat{\mathbf{y}} \leftarrow \text{Predict\_POS}(\mathbf{x}_i; \mathbf{w})$
6 $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}_i) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}})$
7 $\overline{\mathbf{w}} \leftarrow \overline{\mathbf{w}} + \frac{\mathbf{w}}{NT}$
8 **Output:** Parameters $\overline{\mathbf{w}}$

Notice that the weight vector in line 6 is only modified if $\hat{\mathbf{y}} \neq \mathbf{y}_i$. The averaging of the parameters can be understood as a type of regularization to prevent overfitting. In practice, one would choose the number of epochs T by evaluating performance on held-out data. However, for the purpose of this problem set, let $T = 50$.

(a) **Implement the averaged structured perceptron algorithm.** You may use any programming language. You may want to use the inference code that you wrote for the earlier problem sets.

(b) Run your algorithm using the first 100 training sentences. Using the learned parameters, report the average per-token error rate on the training set. Next, evaluate the learned parameters using the 1000 test sentences, again reporting the average per-token error rate. How do these differ?

(c) **Effect of Training Data Set Size:** Repeat the experiment in the previous part, but varying the number of training sentences from the first 100 to the first 1000 in steps of 100. Produce one graph showing the average per-token error rate on test data as a function of the amount of training data.

In addition to your answers, hand in all code that you write for this assignment.

Bonus questions (optional):

(d) **Structured SVM:** A different approach to structured prediction is to solve a structural SVM optimization problem. As we discussed in class, this has several advantages including a clear objective function (based on hinge loss, which is an upper bound on 0-1 loss) and max-margin regularization. Learn a structural SVM, and compare your results to those learned by the averaged structured perceptron algorithm. Choose the regularization parameter C by using a hold-out set of the last 100 sentences from the training data.

Although there are many algorithms for optimizing the structured SVM objective, we suggest using SVM^{struct},

`http://svmlight.joachims.org/svm_struct.html`

which is based on the cutting-plane approach. This code offers a Python, Matlab, and C++ interface for implementing the required functions (namely, MAP inference and loss-augmented MAP inference).

(e) **Features:** The accuracy of the CRF model is limited by the features used. Consider defining some of your own features and adding them to the existing set of features. Rerun the evaluation using your new features. Can you find new features that lead to a reduction in POS tag prediction error? If the features that you add make the CRF no longer chain-structured, you can use your MPLP implementation to do (approximate) MAP inference.

The amount of extra credit will be commensurate with the quality of your implementation and your analysis of the results. If you answer any of the bonus questions, please also submit all code electronically to the course instructors, with instructions on how to run.

Acknowledgement: This question is based on an assignment developed at UMass Amherst by Ben Marlin, Andrew McCallum, Sameer Singh and Michael Wick.