Assignment 2

Structured Prediction

Due: April 24, 2016, 11:59 PM PDT (grace period: we will not assign penalty this time if you turn it in by April 26, 11:59Pm, 2016).

Late policy: Every 5% of the total points will be deducted for every extra day past due.

Submit your report to Ted.

The aim of this project is experiment on different solutions for various structured predication algorithms. Please choose two out of the methods described below: SVM_struct, CRF, M3N, and FixedPoint.

Write a report about your experiments, comparison of the testing results, and any conclusions you might be able to draw from the results.

Datasets: The datasets to be used in the assignment are listed below. Source code (matlab) for running experiments on both the OCR and POS has been provided in the fixed-point paper. You are expected to report the results on both datasets. For the second one (either svm-struct, CRF, M3N), having result on the POS data is optional (bonus points).

	# features	Data source
Optical Character Recognition (OCR)	128	R. Kassel. A Comparison of Approaches to On-line Handwritten Character Recognition. PhD thesis, MIT, 1995. http://ai.stanford.edu/~btaskar/ocr/
Description (from Nguyen and Guo 2007)	"The OCR data set contains ~ 6000 handwritten words, with average length of ~ 8 characters, from 150 human subjects, from the data set collected by Kassel (Kassel, 1995). This data set is divided into 10 folds of ~ 600 training, ~ 100 validation, and ~ 5400 testing examples. The input features for each token is a vector representation of a 16 × 8 binary image of a letter."	
Part-of-speech tagging (POS)	446054	http://www.cis.upenn.edu/~treebank/home.html Treebank, The Penn. Penn's linguistic data consortium. http://www.cis.upenn.edu/treebank. http://www.machinelearning.org/proceedings/icml2007/papers/206.pdf
Description (from Nguyen and Guo 2007)	"POS tagging consists simply of labeling each word in some text with its part of speech. Our POS data set (The Penn Treebank, 2002) is separated into five different training sizes: 500, 1000, 2000, 4000, and 8000 sentences. For each training size, we leave out 10% of the sentences as the validation data. All trained models including SLE are evaluated on a separate test set of ~ 1600 sentences. The input features for each token (i.e, a word in POS task) vary with its position in the sentence. The features are defined by the user and are usually predictors like "previous word ends with -ation". In the dataset, the total number of such simple lexical features is around 450,000."	

1. For the fixed-point algorithm, you can just learn the code, run the experiments and report the results. Note that in practice, each structured input sample may have different lengths. In principle a scanning

window strategy (see an explanation of page 15 of lecture notes on 04/21, https://sites.google.com/site/ucsdcogs185spring2016/calendar-notes) is needed to deal with structured input of varying lengths.

The fixed-point algorithm we provide you with already is able to deal with the varying length problem but it might be trick when implementing your own algorithm when using svm-struct.

2. When experimenting on other methods, e.g. svm-struct, to simplify the problem, you can trim the features of samples in your dataset, so that every sequence has a fixed length (2 or 3). In this way, you don't have to worry about sequences with varying length (otherwise you should using a sliding window approach and average the predictions for every sequence).

2. Window size is a commonly-used term in structured prediction when designing your feature function. Window size means how many neighbors a structured classifier is considering during training. If you are exploring varying window size, just try 1 (look at position i-1 and i+1) or at most 2 (look at position i-2 i-1 i+1 i+2) to reduce the training time.

Note that in the code the context window size is set to 14:-2:14, which is basically 14. a context window size M means you look M/2 letters to the left, and M/2 to the right. You can try to modify this to test performance of varying window size. Auto-context and fixed-point methods are consider as only one method in this assignment.

1. Fixed-point/Auto-context

Download the matlab code and data for the auto-context algorithm and the fixed-point method at:

http://pages.ucsd.edu/~ztu/publication/fixed-point_release.zip
A Mac version (adapted by Zeyu Chen) is available at:

http://pages.ucsd.edu/~ztu/publication/fixed-point_mac.zip

Another Mac version (adapted by Yilun Liu) is available at:

http://pages.ucsd.edu/~ztu/publication/fixed-point_mac_v2.zip

The package includes the OCR and POS data for training and testing your structured prediction problem. Since the both datasets are relatively large and it might be time consuming to perform a full-scale training and testing, you are allowed to scale down the data to relatively small sets that you can handle.

Try to vary the window size (e.g. 0, 1, 2) for the context and observe different results. Note that the window size (context range) should be within the limit of the length of your structured data (number of elements).

2. Structural SVM

Compare with the structural SVM algorithm (varying the window size as well)

Pre-compiled Mac and Windows mex files are available at:

https://sites.google.com/site/ucsdcogs185spring2016/assignments/assignment2

The full svm-struct package can be found:

http://www.robots.ox.ac.uk/~vedaldi//svmstruct.html
To report classification errors (accuracy) from the training function, you just to make very small modification to the function, constraintCB by e.g. defining

function ypredict = predict(param, model, x)
  for all possible yhat
   find the ypredict that maximizes dot(featureCB(param, x, yhat), model.w)
end

Make sure that you also report errors from the other method you are comparing with. In general, it should be generally not hard to compile the c code of this package to make prediction on test data.

You need to pay particular attention to three functions in svm-struct, which are meant to provide you with the transparency to the algorithm, as well as a level of flexibility in defining your own function. Note that each x below refers to one sample of structural input. If you are dealing with OCR of length 2, then x is a 128*2 long vector; y is then a vector of length 2. $y \in \{1,2,3,\ldots,26\}$ in the OCR case.

Feature computation to make your feature representation transparent.
If x contains a structural input of two letters then, you can see an example below. Note the example provided below is not the most efficient but you get an idea of how it is done.

function psi = featureCB(param, x, y)

```
x = x';
psi = zeros(1,size(x,2)*26+1);
psi1 = zeros(1,size(x,2)*13);
psi2 = zeros(1,size(x,2)*13);
if(y(1)~=0)
    psi1(1,(y(1)-1)*128+1:(y(1)*128)) = x(1,1:128);
end
if(y(2)~=0)
    psi1(1,(y(2)-1)*128+1:(y(2)*128)) = x(1,129:256);
end
psi(1,1:size(psi1,2)) = psi1(:,:);
psi(1,size(psi2,2)+1:end-1) = psi2(:,:);

% important: please write your own code to append the encoding of y(1) and
y(2) to psi, the next line of code is one type of encoding by comparing the
two neighboring elements
%psi(1,size(x,2)*26+1) = (y(1,1)==y(1,2));

psi = sparse(psi');

if param.verbose
    fprintf('w = psi([%8.3f,%8.3f], %3d) = [%8.3f, %8.3f]\n', ...
        x, y, full(psi(1)), full(psi(2))) ;
end
```

end

Loss function to account for the structured prediction loss:

```
function delta = lossCB(param, y, ybar)
  delta = sum(double(y ~= ybar)) ;
end
```

To look for the most violating examples:

```
function yhat = constraintCB(param, model, x, y)
  x = x';
yhat = zeros(1,size(y,2));
yhat1 = zeros(1,2);
y1 = zeros(7,7);
for i = 0:26
    for j =0:26
        yhat1(1,1) = i;
        yhat1(1,2) = j;
        y1(i+1,j+1) =
lossCB(param,y,yhat1)/size(y,2)+dot(featureCB(param,x',yhat1),model.w);
    end
end
[~,yhat(1,1)] = max(max(y1,[],2));
[~,yhat(1,2)] = max(max(y1));
yhat(1,1) = yhat(1,1)-1;
yhat(1,2) = yhat(1,2)-1;
if param.verbose
    fprintf('yhat = violslack([%8.3f,%8.3f], [%8.3f,%8.3f], %3d)
= %3d\n', ...
        model.w, x, y, yhat) ;
end
end
```

To make your training process simple, you can chop your training data into fixed length of elements, e.g. 2 or 3.

3. CRF

CRF model (with different length of the windows for the binary terms)

http://www.di.ens.fr/~mschmidt/Software/UGM.html
or here:
http://www.cs.ubc.ca/~murphyk/Software/CRF/crf.html

You can also compare with the MRF model also in

http://www.di.ens.fr/~mschmidt/Software/UGM.html

4. M3N

Bonus will be given if you can come up with a new variation (method) or implement the M3N work:

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In NIPS, 2003.

The OCR dataset can be directly downloaded from the link below, if you have trouble loading the data from provided package:

http://ai.stanford.edu/~btaskar/ocr/

Penn's linguistic data consortium. http://www.cis.upenn.edu/treebank.