CS753 : Automatic Speech Recognition
Assignment 3

Arka Sadhu
140070011

Abstract

The problem statement considered here is that given an audio recording from a radio station we want to isolate the parts of the recordings that correspond to human speech. This will be useful for further downstream tasks like sentiment classification, audio summarization etc. The dataset we have includes audio clips annotated with intervals that contain human speech. Here we present two methods for this task. First is the use of Conditional Random Fields (CRF) which uses a discriminative model as opposed to the generative model used by Hidden Markov Models (HMM) and also circumvents bias problems inherent in other discriminative models like Maximum Entropy Markov Models (MEMMs). Second is the use of Segmental Recurrent Neural Networks which defines a joint probability distribution over segmentations of the input and labellings of the sequence.

Problem Statement

We define the observation sequence as observation vectors upto time T(which is assumed to be fixed) as $O = [o_1, o_2, ..., o_T]$. We also assume that we already know the mapping from the observation vector index to the actual speech duration which is to say that we know some observation vector o_i would correspond to the duration $t_i : t_j$. Therefore our only task is to label the observation sequence vectors. The label space we consider here is $l = \{l_h, l_n\}$ with l_h corresponding to human speech and l_n corresponding to anything other than human speech which we are not bothered with. So our task reduces to predicting the label sequence $Y = [y_1, y_2, ..., y_T]$ such that each $y_i \in l$. Another formulation we introduce is to explicitly group together consecutive same labels. In this case we would like to predict the sequence $Z = [z_1, z_2, ..., z_p]$ and $Y = [y_1, y_2, ..., y_p]$ such that $z_i \in \mathbb{Z}_+$ and $y_i \in l$ where z_i represents the duration of a segment and y_i is the label corresponding to the segment. The first formulation is used in the conditional random fields and the second formulation is used in the segmental recurrent neural networks. The training data which is available to us is audio clips with intervals of annotated human speech. Thus for training we can get the data for either formulation.

Methodology

Here we describe both methods Conditional Random Fields(CRF) (Lafferty, McCallum, and Pereira 2001) and Segmental Recurrent Neural Networks (sRNN) (Kong, Dyer, and Smith 2015) in more detail.

Conditional Random Field (CRF)

Conditional Random field is a discriminative framework which is able to model the sequence of data better in the sense of relaxing strong independence conditions in Hidden Markov Models (HMM) and overcoming the label bias problem inherent in Maximum Entropy Makov Models (MEMMs).

The independence condition in HMM is required because we are modeling the joint probability distribution $P(Y, X)$. Here Y is the label sequence and X is the input sequence. It is necessary to make the independence assumption for the probability distribution to be tractable. CRF overcome this by instead modelling $P(Y|X)$ which is the probability distribution of Y conditioned over X and not explicitly modeling the distribution $P(X)$. CRF gets away by not modeling $P(X)$ because at test time the observations are anyway fixed. This also allows for context dependent information to be captured because the label sequence Y is no longer needed to be decided only by the current observation.

The paper also compares CRF to other discriminative framework like MEMMs and claims like models like MEMMs are inherently biased towards states which have smaller number of output branches. The reason for this is that when it is at a particular state the competition of the next state is only from the current state. In other words we are only comparing transitions to the next state from the current state instead of looking at every other transition from any state to any other state (including self loops as well). To avoid this either the finite state machine can be determinized but this can lead to explosion of number of states or to start with fully connected model but that doesn't allow us to use our domain knowledge or priors. This problem is circumvented in CRF because the individual transitions do not explicitly choose the transition rather only amplify or dampen the probability mass they receive.

Actual Model Let X be the random variable over data sequence and Y be the random variable over label sequence.

Even though X and Y are jointly distributed we will try to get the probability distribution $p(Y|X)$ and we choose not to model explicitly the distribution $p(X)$ as it is already fixed during run-time.

We consider a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ characterized by vertex set $\mathbf{V}$ and edge set $\mathbf{E}$ and let $\mathbf{Y} = (\mathbf{Y_v})_{\mathbf{v} \in \mathbf{V}}$ be indexed by the vertices of the vertex set. Then we define $(\mathbf{X}, \mathbf{Y})$ to be a conditional random field if the random variables $\mathbf{Y_v}$ when conditioned on $\mathbf{X}$ obey the Markov property with respect to the graph: $\mathbf{p}(\mathbf{Y_v}|\mathbf{X}, \mathbf{Y_w}, \mathbf{w} \neq \mathbf{v}) = \mathbf{p}(\mathbf{Y_v}|\mathbf{X}, \mathbf{Y_w}, \mathbf{w} \in \mathbf{nbd}(\mathbf{v}))$ Here $nbd(v)$ is the set of vertices one-hop distance away from the vertex v or in the 1-neighbourhood of v.

An important thing to note is that the CRF is globally conditioned on the variable X. Since our primary task is to build the ASR system to extract human speech we consider a specific case of the graph G which is a simple chain with T vertices and each consecutive vertices are joined by a single edge. Let $X = (X_1, X_2, ..., X_T)$ and $Y = (Y_1, Y_2, ..., Y_T)$. Now using the fundamental theorem of random fields we know :

$$p_\theta(y|x) \propto exp(\sum_{e \in E} \lambda_k f_k(e, y_e, x) + \sum_{v \in V} \mu_k g_k(v, y_v, x)) \tag{1}$$

Here x is the whole sequence data, y is a label sequence and $y_{v/e}$ is the set of components associated with the vertex or edge in other words the sub-graph consisting of the vertex. For edge it will be simply the two vertices being connected. We assume that f and g are fixed and the parameter set is the $\{\lambda_i, \mu_j | i \in E, j \in V\}$ which will be estimated from the training data.

We note here that specific choice of f which depends only the transition probabilities and g which depends on the observation probability, we can get HMMs that is to say, HMMs can be considered as a special class of CRF and in general CRF is much more expressive. We also note that CRFs have inherently convex loss functions.

As mentioned earlier CRF looks at the set of all transition probabilities at once. While in HMM the transition probabilities considered at a particular state is deciding which is the next state to go, in the case of CRF it is considering all transitions from any state x to any other state y. For convenience we consider first and last state as start and stop state respectively. So we have a matrix of transition probabilities at each vertex (our graph is a chain) as M_i which is of size mxm where m is the number of labels. We define

$$M_i(x) = [M_i(y', y|x)]$$

which is the probability of the particular transition happening at vertex i.

$$M_i(y', y|x) = exp(\Lambda_i(y', y|x))$$

$$\Lambda_i(y', y|x) = \sum_k \lambda_k f_k(e_i, y_{e_i}{=}(y', y), x) + \sum_k \mu_k g_k(v_i, y_{v_i}{=}y, x)$$

The second formula follows from (1). Clearly the normalizing factor to make it a probability distribution will be the product of all the matrices

$$Z_\theta = \prod_{i=1:n+1} M_i(x)$$

Here we use θ to denote that there are parameters to be learnt. This gives us

$$p_\theta(y|x) = \frac{\prod_{i=1}^{n+1} M_i(y_{i-1}, y_i|x)}{\prod_{i=1}^{n+1} M_i(x)}$$

Segmental Recurrent Neural Networks (SRNN)

Segmental Recurrent Neural Networks (SRNN) is a generative framework. Given an input sequence it defines the joint probability distribution over segmentations of the input and labelings of the corresponding segements. The segments are contiguous subsequences of the input sequence and their representation are computed using a Bidirectional Recurrent Neural Networks (BiRNN). The corresponding sentence embeddings are used to get the output labels. Locally semi-markov conditional random fields are used to get the compatibility scores of the output labels. Moreover this can be used for both fully supervised settings where the training data consists of segmented labeled data and partially supervised setting where the segment boundaries are not explicitly given. The authors claim that even when segmentation of the original data is not explicitly required it is better to do so and this explicit segmentation leads to significantly higher accuracy.

Actual Model The actual model is fairly straightforward. The input sequence is $x = (x_1, ..., x_T)$. The output sequence consists of two variables z which characterizes the duration of each segment and y which is the label of the corresponding segment. Therefore $z = (z_1, ..., z_p)$ where $z_i \in \mathcal{Z}_+$ and $y = (y_1, ..., y_p)$ where $y_i \in l$. It is interesting to note the difference with the conditional random field where the output sequence of data was of the same length as that of the input sequence. Here each output label corresponds to a segment instead and therefore we can get away with a much smaller output sequence ($p \leq T$). In cases where the input sequence is much longer and the duration of each label is much larger we can have $p \ll T$. And clearly we have $\sum_{i=1}^{p} z_i = T$. We also define the start time of segment i as $s_i = 1 + \sum_{i<j} z$

We want to solve

$$y* = argmax_y p(y|x)$$

which can be approximately written as

$$y* = argmax_y \max_z p(y, z|x)$$

where we jointly maximize over y and z so as to make the problem tractable. However when explicit duration are not available we can only marginalize over z. We consider the negative log-likelihood which is to be minimized.

$$\mathcal{L} = -\log p(y|x) = \log \sum_z p(y, z|x)$$

We further note that

$$p(y, z|x) = \frac{\prod_{i=1}^{p} exp(f(y_{i-k:i}, z_i, x))}{Z(x)}$$

Here $Z(x)$ is a normalization function to make it an appropriate probability distribution and f is basically an affine

transformation of the features of its arguments. In simpler words $f(x) = w^T\phi(\alpha(x)) + b$. Here ϕ is just the non-linear activation and α maps the input to meaningful vector space representation. In the paper the authors use α as another affine transformation of a concatenation of the vectors $[g_y(y_{i-k}); ...g_y(y_i); g_z(z_i); c_f; c_b]$. Here $g_{y/z}$ outputs the vector representation of the input and the $c_{f/b}$ is the forward and backward context from the BiRNN. c_f, c_b are the encoding of the contents from $s_i : s_i + z_i - 1$ obtained from forward and backward passes of the BiRNN.

Task

Here we describe how the two models explained can be used for the speech extraction task described in the problem statement.

Conditional Random Field (CRF)

The task to be accomplished is described in the problem statement. For CRF we use the first formulation that is we have the observation vector $x = (x_1, ..., x_T)$ and we need to get the label sequence $y = (y_1, ..., y_T)$. We consider both training and testing.

Training The parameters of the CRF are the M matrices which consequently depend on the parameters $(\lambda_i, \mu_j | i \in E, j \in V)$. The training data consists of segmented data with corresponding labels of being a human speech or not.

To learn the parameters we use the algorithm Improved Iterative Scaling Algorithms (IIS). The exact derivations of these are slightly involved and hence it is not being covered in the report. The overall idea is that the scales that are used in the updates is obtained by equating the expectation of a particular feature with the observed expectation. This step requires the calculation of the **Total feature count** and this comes in the exponent and calculating the exponential sums in often problematic.

Thus there are two algorithms working together Algorithm S and T. Algorithm S has a new feature called the slack feature so that we don't have to keep track of the exact total feature count and yet get away with it. The slack feature is given simply by

$$s(x, y) = S - \sum_i \sum_k f_k(e_i, y_{e_i}, x) - \sum_i \sum_k g_k(v_i, y_{v_i}, x)$$

and S is chosen so that $s(x^{(i)}, y) \geq 0$ for $x^{(i)}$ seen in training data. The slack feature is global feature and doesn't belong to any one vertex. Algorithm T on the other hand remembers the partial T totals.

Now quite similar to the Baum-Welch algorithm, we define forward and backward vectors to deal with the large number of parameters. This is done for efficient computations of the matrices $M_i(x)$ for each i using dynamic programming. Combining both the forward and backward passes along with the matrices M_i and the features f_k and g_k it is possible to get the weight updates for the parameters λ and μ.

An important point to be noted is that a single iteration of the Algorithm S and Algorithm T has nearly the same time and space complexity as that of Baum-Welch.

A complication that can occur during the train time is that the feature s considered in the Algorithm S contains the constant S which is sometimes quite large as it is proportional to the length of the sequence. In such cases keeping track of feature totals for each observation sequence separately is helpful.

Testing At test time we are given the whole sequence. Then the problem reduces to simply finding the label sequence with the largest probability. We can directly use the MLE on $p(y|x)$ and this will give us the best label sequence data and each x_i for $0 \leq i \leq T$ will have a corresponding y_i and each $y_i \in l$ which is either human speech or not.

Segmental Recurrent Neural Networks (SRNN)

Again the main task to be done is in the problem formulation. For SRNN we use the second problem formulation. So we have the input sequence as $x = (x_1, ..., x_T)$ and output label sequence $y = (y_1, ..., y_p)$ as well as the duration sequence $z = (z_1, ...z_p)$.

Training To make use of the training data, we would first need to calculate the sentence embeddings (both forward and backward). Then we want to get the most likely segmentation and labeling. To get the sentence embeddings if we naively do a forward and backward pass it will take $O(n^3)$ steps but efficient calculation using dynamic programming allows us to do it in $O(n^2)$. This basically utilizes the computation from state i to state j without repeating the same calculation more than once.

To get the partition function $Z(x)$ we again use dynamic programming to calculate it in polynomial time.

$$\alpha_0 = 1$$

$$\alpha_j = \sum_{i<j} \alpha_i \sum_{y \in l} exp(f(.))$$

$$Z(x) = \alpha_T$$

Here . refers to the same arguments used for the calculation of the probability distribution $p(y, z|x)$. This in turn would have required us to calculate the sentence embeddings which we calculated earlier. The updates are pretty similar to forward pass recursion in HMM. Changing all summation parameters to max operators we can do the viterbi-like pass which will result in the maximal posteriori segmentation and labeling.

Again to calculate the posterior marginal which is $Z(x, y)$ we use a similar dynamic programming algorithm by introducing an extra indexing of the y variable.

Now that we have $Z(x)$ and $Z(x, y)$ we start the training procedure. We are only concerned with the supervised training part since we have such data and it will obviously give better results. We want to reduce the loss function described earlier.

$$\mathcal{L} = \sum_{(x,y,z) \in D} -\log p(y, z|x)$$

$$\mathcal{L} = \sum_{(x,y,z) \in D} \log Z(x) - \log Z(x, y, z)$$

Here $Z(xy, z)$ is the un-normalized conditional probability of the corresponding segmentation and labeling given the input.

Following backpropagation, standard optimization algorithms like Adam can be used for updating the weights of the BiRNN. Another equivalent way of backpropagation would be to use the backward variants of the above dynamic programming algorithms.

Testing We would need to repeat all the parts of the training process other than the backpropagation and updating weights. Then choose the pair (y, z) with the largest probability when conditioned on the input sequence. Here also we do take the MLE estimate of $p(y, z|x)$ as the resulting segmentation and label. Then we simply extract out the corresponding frames which lie in the segments tagged as human speech.

Benefits and Shortcomings

Here both Benefits and Shortcomings of the proposed models are given.

Conditional Random Field (CRF)

Benefits CRF comes with very obvious benefits some of which have already been discussed.

- It is able to relax the strong assumption made in the HMM about independence of observation from previous observation. In CRF it takes in the whole sequence and therefore sees the whole context.
- The features do not need to completely specify a state or observation so we can expect it to require less training data.
- It also circumvents the label-bias problem inherent in many other discriminative framework like MEMMs by considering all the transition possible rather than considering only the transitions from the present state.

Shortcomings CRF while is an excellent tool for speech recognition, there are some shortcomings.

- Firstly the transition matrices can depend as $O(n^2)$ on the number of labels required. In our case it is not a problem since there are only two labels, but in the case with large number of labels this can be an obvious shortcoming.
- The other is related to its inherent model in that it requires to see the whole sequence and therefore it may miserably fail with only partial observability like seeing only the first 3 observation vectors would not allow the model to make any good decision.

Segmental Recurrent Neural Networks (SRNN)

Benefits SRNN gives some interesting insights and interesting benefits.

- Firstly it gives both the segment duration as well as the segment labels jointly. Moreover it can work even in the partially supervised case as well.
- It also shows that in cases where explicit segmentations are not required, when this is done it leads to accuracy improvement.
- Output sequence of labels and durations can be much smaller than the input sequence length.
- Using BiRNN it is able to encode contextual information as well.

Shortcomings SRNN are not devoid of shortcomings.

- SRNN are essentially generative framework. In general it is found that discriminative framework gives better results and hence it may give sub-par performance on some tasks.
- While it can accommodate k^{th} markov model, the dynamic procedure may blow up and it may take a lot of time for individual updates.
- It uses BiRNN which inherently have the problem of large latency.

References

Kong, L.; Dyer, C.; and Smith, N. A. 2015. Segmental recurrent neural networks. *CoRR* abs/1511.06018.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 282–289. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.