report for

Dynamic Programming and the Application in Speech Recognition System

EE 7003: Directed Individual Study

May 9, 2000

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

Yu Zeng

Mississippi State University
Box 9671
Mississippi State, Mississippi 39762
Tel: 601-325-8335
Fax: 601-325-8192
email: zeng@isip.msstate.edu

1 ABSTRACT

Since Bellman's first article introducing the theory of Dynamic Programming 50 years ago, this algorithm has had various application in the areas of engineering, economics, commerce, management etc. In the late 1960s and early 1970s, dynamic programming was initially introduced to speech recognition field. Until today, many speech recognition systems have been developped using this algorithm. In this paper, we first discuss the principle of dynamic programming algorithm. An example of sentence recognizer will be used to give us a straight-forward concept about this algorithm. Then we will discuss why dynamic programming is attractive for speech recognition system. The hierarchical dynamic programming search strategy will be introduced at the last part.

2 Elements of Dynamic Programming

Dynamic programming is usually used in optimization problems in which a set of decisions must be made to arrive at an optimal solution. As decisions are made, subproblems of the same form often appear. Dynamic programming is effective when a given subproblem may arise from more than one partial set of decisions. By storing, or memorizing the solution of each of the subproblem in case it may reappear, this algorithm can reduce the computational time [1]. Below are the points to evaluate if a problem can be solved using dynamic programming or not:

1. Optimal substructure

If an optimal solution to a problem contains optimal solutions to subproblems, we say that this problem exhibits optimal substructure. Whenever a problem exhibits optimal substructure, it is a good clue that dynamic programming might apply [1].

2. Overlapping suproblems

The space of subproblem must be relatively "small" so that dynamic programming is applicable. This way, when we develop a recursive algorithm to solve the problem, instead of always generating new subproblems, the same subproblems may arise over and over. We call these subproblems "overlapping subproblems". Dynamic programming algorithm takes advantage of overlapping subproblems by solving each of them once and then storing the solution in a table where it can be looked up when needed [1].

The development of a dynamic programming algorithm can be broken into a sequence of four steps:

1. Charactize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution in a bottom-up fashion.
4. Construct an optimal solution from computed information.

Now let's introduce the basic terms and variables of dynamic programming algorithm:

1. Stage

The optimization problem will be divided into a set of stages so that the decision can be ordered. In most cases, it's divided along the temporal or spacial ordinate, and the stage variable is

discrete.

2. State

This concept is used to describe the natural property of stages. Usually, a stage contains more than one state. We use s_k as the symbol of state variable. It can have different kinds of presentation, a number or a vector. The dimension of vectors at different stages can be different too. The state we are discussing here should hold the Markov property. This is to say that given a state at some stage, the states of previous stages will not affect the decision making after current stage. In another words, the state of a system at any stage is a conclusion of the history of the path ending at this state.

3. Decision

When the process comes to a state of some stage, it can make different decision (or take different choice) to determine the state of the next stage. $D_k(s_k)$ is used to denote the decision made at state s_k of stage k. The value of is $D_k(s_k)$ limited in the set of possible choices, usually the state set of the next stage.

4. Path

The set of a ordered decisions forms a path. $p_{k,n}(s_k)$ is a path starting from state s_k of stage k and ending at stage n.

5. Measure of performance

This concept is used to evaluate the performance of a path. Also it can be called as performance function or path value, it can represent distance, cost, profit or productivity. We denote the path value of $p_{k,n}(s_k)$ by $V(p_{k,n}(s_k))$. It can be the summation or product of the cost of each link in this path:

$$V(p_{k,n}(s_k)) = \sum_{i=k}^{n} V(p_{i,i+1}(s_i)) \qquad [2\text{-}2]$$

$$= V(p_{k,k+1}(s_k)) + V(p_{k+1,n}(s_{k+1}))$$

or:

$$V(p_{k,n}(s_k)) = \prod_{i=k}^{n} V(p_{i,i+1}(s_i)) \qquad [2\text{-}3]$$

$$= V(p_{k,k+1}(s_k)) \times V(p_{k+1,n}(s_{k+1}))$$

If we use $f(i,j)^*$ denote the path value of the optimal path from stage i to stage j, the

optimization problem can be written as:

Find the path which satisfies $f(i, j)^* = \max_{s_i} V(p_{i,j}(s_i))$.

3 Example of Sentence Recognizer

In this section, we use the example of sentence recognizer to illustrate the procedure of dynamic programming algorithm, which was introduced by H.F. Silverman and D.P. Morgan in [2].

Problem: Find the best four-word sentence from the output of a simple DUR recognizer which has a four word vocabulary -- cat, fat, sat, that -- referred to as word(1) to word(4), where $p(i, n)$ is the probability that word $W(i)$ was uttered at time n where $i, n \in [1, 4]$. Table 1 illustrates all of the 16 probabilities.

	Time(1)	Time(2)	Time(3)	Time(4)
$W(1)$ = 'cat'	0.229	0.242	0.300	0.226
$W(2)$ = 'fat'	0.257	0.258	0.200	0.290
$W(3)$ = 'sat'	0.243	0.227	0.267	0.290
$W(4)$ = 'that'	0.271	0.273	0.233	0.194

Table 1: Probability that word i is uttered at time n

Also we assume that we have analysized a large number of sentences made up from this four-word vocabulary. The knowledge extracted forms the bigram model $Q(i|j)$, which is the probability that word i is uttered at time $n+1$ given that word j is uttered at time n. Table 2 shows this language model, where the word $W(0)$ = silence is added in table 2 to give the

	Word uttered at time n				
word at $n+1$	$W(0)$ = silence	$W(1)$ = 'cat'	$W(2)$ = 'fat'	$W(3)$ = 'sat'	$W(4)$ = 'that'
$W(1)$ = 'cat'	0.10	0.05	0.50	0.10	0.45
$W(2)$ = 'fat'	0.40	0.10	0.20	0.30	0.45
$W(3)$ = 'sat'	0.10	0.50	0.10	0.30	0.05
$W(4)$ = 'that'	0.40	0.35	0.20	0.30	0.05

Table 2: Bigram $Q(i|j)$

problem a tractable starting condition.

We define the term "best sentence" as the most likely sentence given what is known from the

recognizer. The possibility of a sentence is defined as the summation of the probabilities of each two-word pair:

$$C(\boldsymbol{I}) = \sum_{n=1}^{4} Q(i_n | i_{n-1}) P(i_n, n) \qquad [3\text{-}1]$$

where $\boldsymbol{I}$ is the indices sequence of a sentence, i_n is the value of i at time n, and $i_0 = 0$, for the initial state. The optimization problem comes to finding the sentence which maximize the measure of C.

If we use exhaust numeric algorithm, then we simply evaluate all $4^4 = 256$ possible sentences, and pick up the one with maximum score of C. If what we need to recognize is not just a four-word sentence from a four-word vocabulary, but a 10-word sentence from a 100-word vocabulary, then the number of possible sentences is 100^{10}. We can see that as the size of the vocabulary and the length of the sentence increase, the search space will get out-of-hand very quickly.

Let's reflect this problem into figure 1, where each node is a word. The number to the lower right of the node is the recognition probability $P(i_n, n)$. The problem is now determining the optimal path starting from node 'init', through one node of each column and ending at column 4.

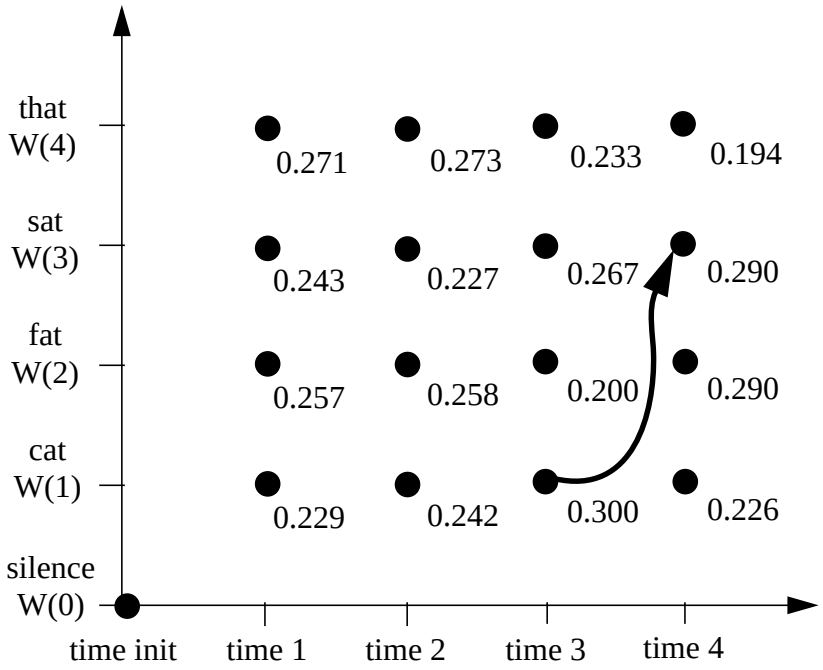

Figure 1. figure 1. Sentence recognition example

Since we defined the performance measure as the summation of the probabilities of each two-word pair, this forms a good optimal substructure. As we comes to time 4, we can see that the best 4-word sentence ending at word *sat* must be a path extending from one of the best 3-word sentences ending at each of the possible word at time 3. In another words, for each word of each time, we only need to think about one sentence extension from the previous time stage. If we

define $D(i, j)$ as the maximum value of C for the i-word sentences ending at word j, then the DP recursive equation can be constructed as:

$$D(i, j) = \max_{1 \le k \le 4} [D(i-1, k) + Q(j|k) \cdot p(j, i)] \qquad [3\text{-}2]$$

We compute the value of $D(i, j)$ for each word, and store the results, then for the next column, we do not need to recompute the value of $D(i-1, j)$, but simply find it out from the storage. At the last stage, comparing the values of $D(4, j)$, we get the possibility of the most likely path. Tracing back this path, this sentence is decoded as *That fat cat sat*. The procedure of the forward dynamic programming process and the back tracking result is illustrated in figure 2. The thin lines are the optimal path ending at word j from previous time stage. The thick lines are the final optimal path starting from initial time and ending at time 4.

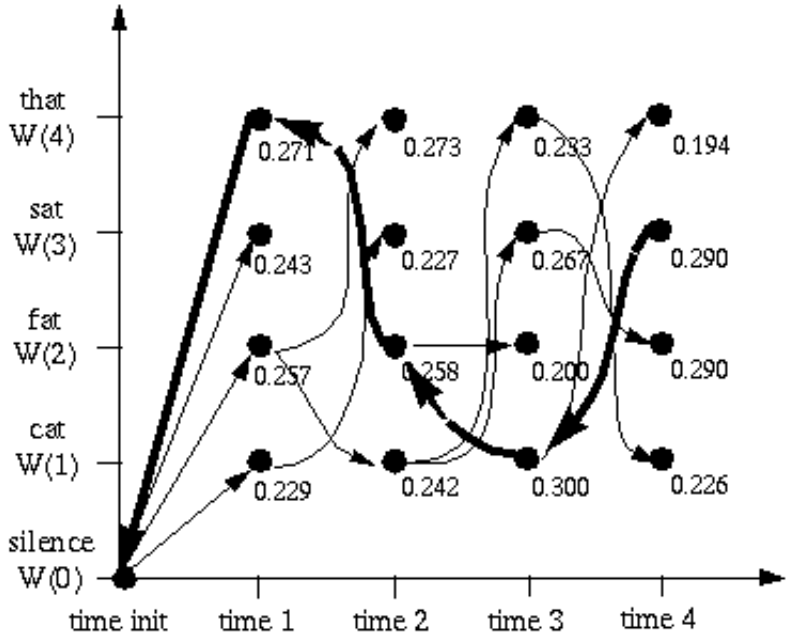

Figure 2. DP forward procedure and back-tracking result for sentence recognition example

Now let's discuss the complex of DP search. The principle cost of the DP search lies in the forward stage. To decode an N words sentence from an M words vocabulary, DP would require slightly more than $O[N \cdot N \cdot M]$ computations. This is much less than the full-search, which requires $O[N \cdot M^N]$ computations. What we need to point out here is that DP would not reduce the search space. It only reduce the computational time by storing the intermediate results [2].

4 Application of Dynamic Programming in Speech Recognition

After more than 30 years of development, dynamic programming has become an important tool for speech recognition systems. It has been successfully used in digit string recognition system, alphadigits recognition system, medium-size vocabulary recognition system using heavily constrained grammars and also, large vocabulary continuous speech recognition system with

virtually unconstrained speech input [3].

4.1 Statistical methods in speech recognition

The problem of speech recognition can be stated as: given an acoustic sequence A, choose a word sequence $\hat{W}$ that maximizes the probability that the word sequence W was spoken:

$$\hat{W} = \arg\max_{W} p(W|A) \qquad [4\text{-}1]$$

where $p(W|A)$ is the posteriori probability of the occurrence of a word sequence given that acoustic signal A was observed. Using Bayesian approach, equation [4-1] can be rewritten as:

$$\hat{W} = \arg\max_{W} p(A|W)p(W) \qquad [4\text{-}2]$$

In this equation, $p(A|W)$ is the probability that data A was observed if a word sequence W was spoken. We call this acoustic model. $p(W)$ is the probability that word sequence W was spoken,

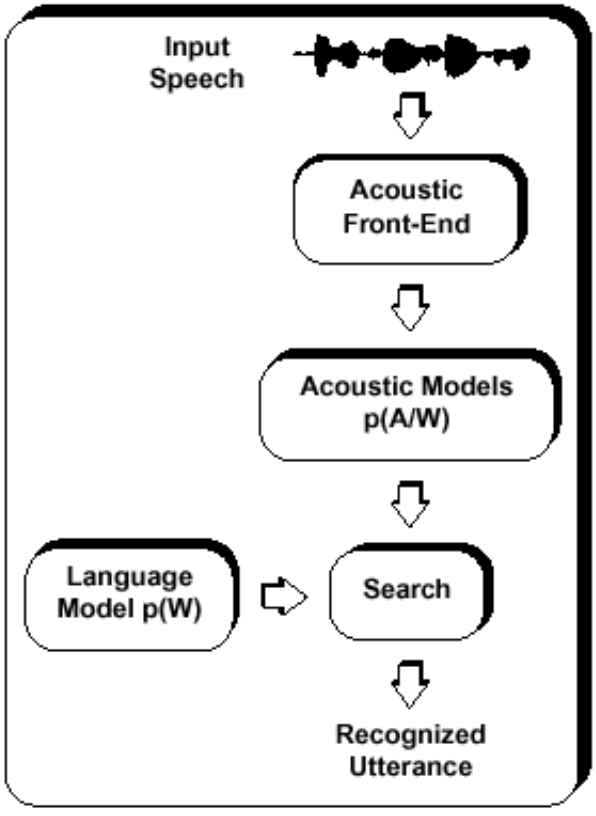

Figure 3. Statistical speech recognition system

which is called as language model [4].

The procedure of a typical statistical speech recognition system is illustrated in figure 3. The input

acoustic signal will be sliced into 5 to 50 ms frames, each of them will be converted into a sequence of feature vectors based on spectral and temporal measurements by the acoustic front-end. Combining the information from acoustic model and language model, the search engine determines the most probable word sequence $\hat{W}$ [4].

The major problems that make the speech recognition task so difficult are that it is hard to define the boundaries between phonemes and words, and there is a large variation in the speaking rates in continuous speech. Also, the feature vectors for each phone are heavily overlapped. These reasons result in the fact that the number of possible word sequences of all possible length for the same acoustic observation sequence is very large. Exhaustic numeric algorithm is obviously infeasible, since possible sentences can be infinite for a given language. Finding effective search algorithm is important for speech recognition system [3].

4.2 Hierarchical search based on dynamic programming

In a large vocabulary conversational speech recognition system (LVCSR), the sentence is a sequence of words W, each word w is a sequence of phones, and for each phone, we divide it into several sub-phone units, denoting as states s. Here, models at each level serve as the evaluator to assign a likely score to each hypotheses sequence, namely language model, pronunciation dictionary, acoustic model and context-dependent phone model such as diphone or triphone model. These knowledge sources help to reduce the search space [4]. Under this decomposition theme, the Bayesian approach of [4-2] results in the following optimization problem:

$$\hat{W}^N = \arg\max_{W^N} \left[p(W^N) \cdot \sum_{S^T} p(X^T, S^T | W^N) \right]$$

$$= \arg\max_{W^N} \left[p(W^N) \cdot \max_{S^T} p(X^T, S^T | W^N) \right] \qquad [4\text{-}3]$$

Where W^N means N-word sequence, S^T and X^T means sequence of N states and feature vectors respectively [3]. This equation implies that this problem satisfies the optimal policy for dynamic programming. We can develop a hierarchical search system to find the most likely word sequence using dynamic programming algorithm given those models.

Let's focus on the search procedure in the state level. After the training step, we assume that a prototype P for each state of each phone is generated. For each frame j, we define $D(j, s_k)$ as the distortion between the input feature vector $x(j)$ and the prototype of state s_k. The score of a path ending at state s_i of time stage i can be defined as $V(i, s_i) = \sum_j D(j, s_k) p(s_k | s_{j-1})$, where $p(s_k | s_{j-1})$ is the transition probability from state s_{j-1} of time $j-1$ to state s_k of time j. Using

distance $D(j, s_k)$, the DP recursive equation can be constructed as:

$$V(i, s_i) = D(i, s_i)p(s_i|s_{i-1}) + minV(i-1, s_{i-1}) \quad [4\text{-}4]$$

Here we use minimization of the distortion to replace the maximization of the probability. For other levels, the same procedure will be applied so as to achieve the best word sequence.

5 Summary

Dynamic programming is the most widely used algorithm in speech recognition. In this paper, we introduced the concept and procedure of dynamic programming using the sentence recognizer example. Then we discussed the reason why this algorithm was applied in speech recognition system, and the basic concept of hierarchical LVCSR system using dynamic programming algorithm.

REFERENCE

[1] T.H. Cormen, C.E.Leiserson and R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts London, England, USA, 1990.

[2] H.F. Silverman, D.P. Morgan, "The Application of Dynamic Programming to Connected Speech Recognition", *IEEE Acoustics, Speech, and Signal Processing Magzine*, July 1990.

[3] H. Ney, S. Ortmanns, "Dynamic Programming Search for Continuous Speech Recognition", *IEEE Signal Processing Magazine*, Vol. 1, No. 5, pp.64-83, September 1999.

[4] N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition", *IEEE Signal Processing Magazine*, Vol. 1, No. 5, pp.84-107, September 1999.