Pruning Hidden Markov Models with Optimal Brain Surgeon

Brian Mak and Kin-Wah Chan

Abstract

A method of pruning hidden Markov models (HMMs) is presented. The main purpose is to find a good HMM topology for a given task with improved generalization capability. As a side effect, the resulting model will also save memory and computation costs. The first goal falls into the active research area of model selection. From the model-theoretic research community, various measures such as Bayesian information criterion, minimum description length, minimum message length have been proposed and used with some success. In this paper, we are considering another approach in which a well-performed HMM, though perhaps oversized, is optimally pruned so that the loss in the model training cost function is minimal. The method is known as Optimal Brain Surgeon (OBS) that has been applied to pruning neural networks in the past. In this paper, the OBS algorithm is modified to prune HMMs. While the application of OBS to neural networks is a constrained optimization problem with only equality constraints that can be solved by Lagrange multipliers, its application to HMMs requires significant modifications, resulting in a quadratic programming problem with both equality and inequality constraints.

The detailed formulation of pruning an HMM with OBS is presented. It was evaluated by two experiments: one simulation using a discrete HMM, and another with continuous density HMMs trained for the TIDIGITS task. It is found that our novel OBS algorithm was able to "re-discover" the true topology of the discrete HMM in the first simulation experiment; in the second speech recognition experiment, up to about 30% of HMM transitions were successfully pruned, and yet the reduced models gave better generalization performance on unseen test data.

Keywords

model pruning, hidden Markov model, optimal brain surgeon, quadratic programming.

Corresponding Author: Dr. Brian Kan-Wing Mak.

Dr. Brian Mak is with the Department of Computer Science, the Hong Kong University of Science and Technology (HKUST), Clear Water Bay, Hong Kong. E-mail: mak@cs.ust.hk.

Mr. Kin-Wah Chan finished this work when he was an MPhil student of the Department of Electrical & Electronic Engineering, HKUST, Clear Water Bay, Hong Kong. E-mail: eeivan@cs.ust.hk.

I. Introduction

In recent years, hidden Markov model (HMM) has been applied successfully to statistical modeling of sequential (temporal or spatial) data such as acoustic signal for automatic speech recognition (ASR) [1], [2], [3], handwritings for handwritten word recognition [4], [5], [6], and protein or DNA sequences [7] in bioinformatics. In practice, the topology of an HMM — the number of states and their connectivity[1] — for a given task is usually pre-set by experience or heuristics, or found by trials-and-errors. As in any data modeling problem, one faces the modeling dilemma: if a model has too many parameters, it risks over-fitting the training data and results in poor generalization (on unseen data); but if the model has too few parameters, it may compromise its capability to represent the underlying data distribution. In modeling theory, this is the model selection problem. Common selection measures include the Bayesian information criterion (BIC) [11], Akaike information criterion (AIC) [12], minimum description length (MDL) [13], and minimum message length (MML) [14]. Recently Biem *et al.* [15] proposed an HMM-oriented BIC, called HBIC, for HMM selection.

In the speech recognition community, there are not many works about searching for an HMM of the right topology. Nevertheless, the works usually fall into one of the three major incremental approaches:

(1) Growing approach : One starts with a small HMM and iteratively increments the number of states and/or transitions until the desirable size is reached. For example, in each iteration of Sagayama's *successive state splitting algorithm* (SSS) [16] that derives a hidden Markov network (HMnet), a state with the largest variance is chosen to split either by the temporal domain information or contextual domain information . A maximum likelihood variant of SSS is described in [17].

(2) Pruning approach : A possibly oversized model is trained and then pruned to the right size according to an optimality criterion. For example, Vasko *et al.* [18] started with an ergodic HMM and state transitions were pruned in an iterative algorithm. At each iteration, one state transition was pruned after an exhaustive search in which all possible HMMs with one less transition were trained, and the model that gave the highest likelihood was chosen.

[1]In this paper, we do not deal with the number of distribution mixtures in the HMM topology. The reason is that an HMM state with mixture density is equivalent to multiple single-mixture-density states [8]. Without loss of generality, we may simply assume each state has only one single mixture in our ensuing discussion. On the other hand, quite a few works have been done to determine the optimal number of mixtures in an HMM state. For example, the Bayesian information criterion has been used to determine the right number of Gaussian mixtures in an HMM state in speech recognition [9], [10].

(3) Hybrid approach : One first grows a big HMM and then reduces the model size by state pruning or merging. For example, Casacuberta [19] used his *error-correcting grammatical inference algorithm* (ECGI) to create a huge discrete HMM and then pruned away states that have the least occupancy counts. On the other hand, Stolcke [20], [21] derived multiple-pronunciation models by HMM induction in which again a versatile HMM was first built from data and equivalent states were then merged using a Bayesian approach.

This paper investigates a method belonging to the second approach, called *Optimal Brain Surgeon* (OBS) [22], [23] to reduce the topology of an HMM. The purpose is two-fold:

- to select a good HMM topology for a given task with improved generalization capability, and/or
- to reduce the model complexity so as to save memory and computation costs. This is particularly needed in big HMMs such as multi-stream HMMs commonly used in multi-band ASR [24], [25], [26], [27] or audio-visual ASR [28].

OBS belongs to a class of sensitivity-based weight-pruning methods that make use of second-order derivatives (of some cost function) to eliminate the least "important" weights in a neural network (NN). It does not only remove weights but also re-adjusts the remaining weights optimally. The method has been shown effective in refining the complex topology of an over-fitted neural network in [29]. Although it has been shown that an HMM is a Boltzmann chain [30] and the OBS algorithm can be applied to a Boltzmann chain as well [31], unfortunately the result cannot be applied directly to an HMM to prune HMM transitions. The reason is that a zero weight in a Boltzmann chain does not translate to an equivalent HMM transition of zero probability but unity (and an HMM transition of zero probability is equivalent to a connection weight of $-\infty$ in a Boltzmann chain). In this paper, we would like to apply the theory of OBS and work out its application to pruning HMM transitions or states from the first principle. When applied to a neural network, OBS is an equality-constrained optimization problem that can be solved analytically by the method of Lagrange multipliers. When OBS is applied to an HMM, due to the various HMM constraints, it becomes a quadratic programming problem with equality and inequality constraints that has to be solved by more advanced methods.

This paper is organized as follows: In the following section, we will first review the theory of OBS and its predecessor, *Optimal Brain Damage* (OBD) [32] when they are applied to neural networks. Section III presents the formulation details of how OBS may be applied to an HMM. The major cost of the method is the computation of the Hessian which is detailed in Section IV. Since assumptions made in OBS are often found invalid in practice in neural

networks [33], but the method still deletes the right weights — that is, the assumptions are good approximations — most of the time, empirical evaluation of the method is necessary. In Section V, the application of OBS to HMMs was evaluated on two tasks: to re-discover the true topology of a discrete HMM, and to prune an expanded topology of the strictly left-to-right continuous-density HMMs that were trained for the TIDIGITS task. The paper ends with discussions and conclusions in Section VI.

II. Optimal Brain Surgeon on Neural Network

In 1990, Le Cun *et al.* proposed the Optimal Brain Damage (OBD) [32] and started the proliferation of a new class of weight pruning methods that make use of the second-order derivatives of an error function of a neural network[2]. In these methods, a neural network with a set of J weights $\boldsymbol{w}$ is first trained to convergence according to an error function $E(\boldsymbol{w})$. By the Taylor's expansion, a change in the error function, δE induced by a change in weights $\delta \boldsymbol{w}$ is given by

$$\delta E = \delta \boldsymbol{w}^T \, \boldsymbol{g} + \frac{1}{2} \delta \boldsymbol{w}^T \, \boldsymbol{H} \, \delta \boldsymbol{w} + \cdots \tag{1}$$

where $\boldsymbol{g} = \frac{\partial E}{\partial \boldsymbol{w}}$ is the gradient vector and $\boldsymbol{H} = \frac{\partial^2 E}{\partial \boldsymbol{w}^2}$ is the Hessian matrix[3]. Two assumptions are made:

1. Extremal Assumption: The first term in Eqn. (1) vanishes by the assumption that the network has converged to a local, if not global, minimum of the error function and the gradient $\boldsymbol{g}$ should be zero.

2. Quadratic Assumption: If all terms higher than the third order are negligible, then only the second-order derivative term remains and the change in error can be approximated as

$$\delta E = \frac{1}{2} \delta \boldsymbol{w}^T \, \boldsymbol{H} \, \delta \boldsymbol{w} \ . \tag{2}$$

A. Optimal Brain Damage (OBD)

OBD further assumes that the Hessian matrix is diagonal. That is equivalent to assume that the total change in E when several weights are deleted is the sum of δE caused by deleting each of the weights individually. Thus, the change in error δE_j induced by the elimination of

[2]Note that magnitude-based pruning methods that simply delete weights of the smallest magnitude give non-optimal solution [22], [32].

[3]We adopt the usual notation that bold-faced quantities are vectors or matrices. Vectors are in small cases and matrices are capitalized.

the jth weight, viz. $\delta w_j = -w_j$, is given by

$$\delta E_j = \frac{1}{2} H_{jj} w_j^2 \ , \quad j = 1, 2, \ldots, J. \tag{3}$$

Eqn. (3) is used to represent the saliency of each network weight, and weights of the least saliencies are eliminated to improve the neural network performance.

Experiments in [32] showed that OBD could delete over half of the weights in a neural network, resulting in a network whose speed was improved significantly and its recognition accuracy (generalization) increased slightly on the test data.

B. Optimal Brain Surgeon (OBS)

OBS is more general than OBD and works with the full Hessian matrix. The deletion of a single weight, say, the jth weight w_j is formulated as a constrained optimization problem with the following constraint on Eqn. (2):

$$\boldsymbol{e}_j^T \left(\delta \boldsymbol{w} + \boldsymbol{w} \right) = 0 \ , \tag{4}$$

where $\boldsymbol{e}_j$ is a unit vector with all components except the jth one being zeros. Thus, the objective of OBS becomes

$$\min_{1 \le j \le J} \left\{ \min_{\delta \boldsymbol{w}} \left(\frac{1}{2} \, \delta \boldsymbol{w}^T \, \frac{\partial^2 E}{\partial \boldsymbol{w}^2} \, \delta \boldsymbol{w} \right) \middle| \ \boldsymbol{e}_j^T \left(\delta \boldsymbol{w} + \boldsymbol{w} \right) = 0 \right\} \ . \tag{5}$$

The equality-constrained problem can be solved by the standard method of Lagrangian multipliers. Unlike OBD, OBS does not only delete a single weight, say, w_j, but it will also adjust the remaining weights optimally to give the least increase in the error function by the following formula

$$\delta \boldsymbol{w} = - \frac{w_j}{[\boldsymbol{H}^{-1}]_{jj}} \boldsymbol{H}^{-1} \, \boldsymbol{e}_j \ . \tag{6}$$

One may generalize the method to delete several weights at the same time. For example, in unit-OBS [34], one may prune a node by writing down a constraint like Eqn. (4) for each of its (in-coming and out-going) connection weight, and solve Eqn. (2) with the multiple constraints.

C. Miscellaneous

Other weight pruning methods in neural network that use second-order derivatives differ mainly on their objective functions [35], [36], [33], [37] or on the approximations used [38].

III. Optimal Brain Surgeon on HMM

The Optimal Brain Surgeon algorithm described in the last section may be modified to prune a transition in an HMM. The major difficulty is that while the application of OBS to a neural network is an equality-constrained problem that has a simple analytical solution, its application to an HMM has to preserve various equality and inequality constraints on HMM parameters. As will be shown below, the modification changes a Lagrange optimization problem to a quadratic programming problem.

A. Theory

We take the simple view that a connection weight in a neural network is analogous to a transition arc (hereafter, we will simply call it a transition) in an HMM and a node in a neural network is analogous to an HMM state. The log-likelihood of training data is used as the cost function, but other cost functions pertinent to the performance or model complexity of HMM may be used as well. These choices are summarized in Table I. Thus, the goal may be stated as follows:

"To prune the HMM transition or state that induces the least decrease in the log-likelihood of the training data, and at the same time to optimally re-adjust the probabilities of the remaining transitions while preserving all HMM constraints."

Let us denote the log-likelihood of the training data as $\log L$, and arrange all HMM transition probabilities in a vector $\boldsymbol{w}$. From Eqn. (2), the application of OBS to an HMM may be formalized as an optimization problem that minimizes

$$-\delta \log L = -\delta \boldsymbol{w}^T \frac{\partial^2 \log L}{\partial \boldsymbol{w}^2} \delta \boldsymbol{w} \tag{7}$$

subject to the following three constraints:

I. Selection Constraint : To prune the jth transition, that is, $\delta w_j = -w_j$, we have

$$\boldsymbol{e}_j{}^T (\boldsymbol{w} + \delta \boldsymbol{w}) = 0 \ . \tag{8}$$

II. Sum-to-one Constraint : The sum of probabilities of all out-going transitions from a state must be one. We will use an indicator matrix $\boldsymbol{M}$ to specify the outgoing transitions of all HMM states. If an HMM has J transitions and N states, then it will have an $J \times N$ indicator matrix $\boldsymbol{M}$. The indicator matrix consists of only 1's or 0's; if the entry $M_{ij} = 1$, then transition i is an out-going arc from state j. For example, all states except the last one of the strictly

left-to-right HMM in Fig. 1 has 3 out-going transitions (self transition, next-state transition, and 1-state-skip transition) as shown, then its indicator matrix $\boldsymbol{M}$ can be written as

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}_{J \times N} . \tag{9}$$

The sum-to-one constraint can be expressed as

$$\boldsymbol{M}^T (\boldsymbol{w} + \delta\boldsymbol{w}) = \mathbf{1}_N \ ,$$

where $\mathbf{1}_N$ is an N-dimensional vector of 1's. Or, equivalently, we have

$$\boldsymbol{M}^T \delta\boldsymbol{w} = \mathbf{0}_N \ , \tag{10}$$

where $\mathbf{0}_N$ is an N-dimensional vector of 0's.

III. Non-negativity Constraint : All transition probabilities must be non-negative. That is,

$$\boldsymbol{w} + \delta\boldsymbol{w} \geq \mathbf{0}_J \ . \tag{11}$$

One may think that we need another constraint to make sure all transitions are less than one. However, the non-negativity constraint together with the sum-to-one constraint already implicitly require all transitions to be less than one.

It is the last inequality constraint that turns the optimization to a quadratic programming problem.

B. Algorithm

The whole OBS algorithm for pruning HMM transitions is shown in Algorithm 1. It is an iterative procedure that removes one transition — the least salient transition — at a time. The saliency of a transition is defined as the decrease in the log-likelihood of the training data, $-\delta \log L$, if the transition is pruned. Due to the assumption that the original HMM must have been trained to convergence (in the maximum likelihood sense), any small perturbation of its transition probabilities will decrease the likelihood. Thus, the quantity $-\delta \log L$ should be non-negative, and we would like it to be as small as possible. The OBS algorithm is terminated when

- the least saliency is greater than a threshold θ_s, or
- the number of deletions exceeds a threshold θ_d.

The first stopping criterion prevents a dramatic drop in the performance of the pruned model while the second criterion guarantees a minimum model size. These thresholds were determined empirically in the experimental evaluations described in Section V.

Algorithm 1: Optimal Brain Surgeon algorithm on HMM

STEP 1. Train a sufficiently large HMM until it converges.

STEP 2. Compute its full Hessian $\mathbf{H}$ using the training data.

STEP 3. Solve the quadratic programming problem of Eqn. (7) for the deletion of each possible transition, and record the corresponding saliency $-\delta \log L$ and $\delta \boldsymbol{w}$.

STEP 4. Sort the saliencies. If the least saliency is greater than the threshold θ_s, then stop.

STEP 5. Delete the transition that has the least saliency and update other transition probabilities by the $\delta \boldsymbol{w}$ found in STEP 3.

STEP 6. Repeat STEP 2 – 5 until the number of deletions exceeds θ_d.

C. Unit-OBS

The foregoing discussion only deletes one transition in each iteration of the OBS algorithm. To achieve faster and greater pruning effect, one may try to delete several transitions at a time. In particular, one may prune an HMM state at each iteration in a fashion similar to the use of *unit-OBS* in pruning a node in a neural network [34] by removing all transitions going into or coming out of the pruned state.

The extension to unit-OBS requires two simple modifications to the selection constraint and sum-to-one constraint as follows:

I. Selection Constraint : Pruning an HMM state requires the deletion of all its in-coming and out-going transitions. For example, Fig. 2 illustrates the deletion of 4 arcs in order to prune state S2 of the HMM in Fig. 1. Mathematically, we may replace the selection vector $\boldsymbol{e}_j$ of Eqn. (8) by a selection matrix $\boldsymbol{S}$ that indicates all the relevant transitions to be pruned. For instance, if a state has totally K in-coming and out-going transitions that are represented by $\boldsymbol{e}_{j_1}, \boldsymbol{e}_{j_2}, \ldots, \boldsymbol{e}_{j_K}$, then the selection matrix $\boldsymbol{S}$ can be written as

$$\boldsymbol{S} = [\boldsymbol{e}_{j_1}, \boldsymbol{e}_{j_2}, \ldots, \boldsymbol{e}_{j_K}] \tag{12}$$

and the *selection constraint* for deleting that state is given by

$$\boldsymbol{S}^T (\boldsymbol{w} + \delta \boldsymbol{w}) = \mathbf{0}_K \ . \tag{13}$$

II. Sum-to-one Constraint : The constraint is similar to the one given by Eqn. (10) except that since a state is deleted its corresponding column vector in the indicator matrix $\boldsymbol{M}$ should also be deleted. Continuing with the example shown in Fig. 2, if state S2 is pruned, the indicator matrix to use will be modified as

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \cdots \end{bmatrix}_{J\times(N-1)} . \tag{14}$$

Notice that the 2nd column of $\boldsymbol{M}$ in Eqn. (9) corresponding to state S2 is removed to give the $\boldsymbol{M}$ matrix in Eqn. (14) whose number of columns is also reduced by 1.

The OBS algorithm given in Algorithm 1 is then modified to find the saliency of each state and the state that gives the least saliency is pruned.

D. Active-Set Method

The application of OBS to pruning HMM transitions or states is a quadratic programming problem. In all experiments presented in this paper, the Matlab optimization toolbox was used to solve the problem; it basically implements the *active-set method*. Details of the method is beyond the scope of this paper and the readers are referred to [39]. In brief, it is an iterative method which maintains a working set of *active constraints* in each iteration. All inequality constraints in an active set are converted into equality constraints, and the subsequent equality-constrained problem is solved by standard Lagrange multiplier method. If the Lagrange multiplier corresponding to a currently active inequality constraint is negative, the inequality constraint will be removed (or become inactive) from the working set. On the other hand, some direction search method is used to determine a new constraint to be added to the active set. The procedure iterates until a feasible solution is found and the Lagrange multipliers for all inequality constraints in the active set are non-negative.

IV. Hessian Calculation

The computation of the Hessian $\boldsymbol{H} = \frac{\partial^2 \log L}{\partial \boldsymbol{w}^2}$ plays an important role in the OBS algorithm. The Hessian matrix is derived from the first principle by differentiating the log-likelihood of the training data with respect to each state transition probability. This is done efficiently and iteratively by making use of the forward probability in the Baum-Welch forward-backward training algorithm [40].

Given an observation sequence $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T\}$ and an HMM with N states and model parameters λ, let's denote the likelihood of the observation sequence $P(\mathbf{O} \mid \lambda)$ by L, and its log-likelihood by $\log L$. The likelihood may be computed efficiently by the iterative Forward Procedure as summarized in the following formulas:

$$\alpha_s(1) = \pi_s b_s(\mathbf{o}_1) \ , \quad s = 1, \ldots, N \tag{15}$$

$$\alpha_s(t) = \left[\sum_{r=1}^{N} \alpha_r(t-1)\, a_{rs}\right] b_s(\mathbf{o}_t) \ , \quad t = 2, \ldots, T, \text{ and } r, s = 1, \ldots, N \tag{16}$$

$$L = P(\mathbf{O}|\lambda) = \sum_{r=1}^{N} \alpha_r(T) \tag{17}$$

where $\alpha_r(t)$ is the probability of observing the sub-sequence $\{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_t\}$ by the model, ending up in state r at time t; π_s is the initial probability of state s; $b_s(\mathbf{o}_t)$ is the probability of observing $\mathbf{o}_t$ in state s; and a_{rs} is the probability of making a transition from state r to state s.

A. Hessian of the Log-Likelihood

A general term in the Hessian of the log-likelihood is given by

$$\frac{\partial^2 \log L}{\partial a_{im} \partial a_{jn}} = \frac{1}{L} \cdot \frac{\partial^2 L}{\partial a_{im} \partial a_{jn}} - \frac{1}{L^2} \cdot \frac{\partial L}{\partial a_{im}} \cdot \frac{\partial L}{\partial a_{jn}} \ , \quad i, j, m, n = 1, \ldots, N \ . \tag{18}$$

The calculation of the Hessian of log-likelihood requires the calculation of the first- and second-order derivatives of the likelihood in linear domain. Differentiating the forward probabilities given by Eqn. (16), we have, for $t = 2, \ldots, T$:

$$\frac{\partial \alpha_s(t)}{\partial\, a_{im}} = \sum_{r=1}^{N} \left[\frac{\partial \alpha_r(t-1)}{\partial a_{im}} a_{rs} + \alpha_r(t-1)\, \delta(r,i)\, \delta(s,m)\right] b_s(\mathbf{o}_t) \tag{19}$$

and

$$\begin{aligned} \frac{\partial^2 \alpha_s(t)}{\partial a_{im} \partial a_{jn}} = \sum_{r=1}^{N} \Bigg[& \frac{\partial^2 \alpha_r(t-1)}{\partial a_{im} \partial a_{jn}}\, a_{rs} + \frac{\partial \alpha_r(t-1)}{\partial a_{im}}\, \delta(r,j)\, \delta(s,n) \\ & + \frac{\partial \alpha_r(t-1)}{\partial a_{jn}}\, \delta(r,i)\, \delta(s,m)\Bigg] b_s(\mathbf{o}_t) \ , \end{aligned} \tag{20}$$

where we have made use of the fact that

$$\frac{\partial a_{rs}}{\partial a_{im}} = \delta(r,i)\,\delta(s,m) = \begin{cases} 1 & \text{if } r=i \text{ and } s=m \\ 0 & \text{otherwise} \end{cases} , \tag{21}$$

and any second-order derivatives of a transition probability $\frac{\partial^2 a_{rs}}{\partial a_{im}\partial a_{jn}}$ are zeros.

Finally, using Eqn. (17), elements of the Hessian matrix in Eqn. (18) are given by

$$\frac{\partial L}{\partial a_{im}} = \sum_{r=1}^{N} \frac{\partial \alpha_r(T)}{\partial a_{im}} \tag{22}$$

and

$$\frac{\partial^2 L}{\partial a_{im}\partial a_{jn}} = \sum_{r=1}^{N} \frac{\partial^2 \alpha_r(T)}{\partial a_{im}\partial a_{jn}} . \tag{23}$$

Notice that, in practice, the computation of the Hessian is not as expensive as it may seem: Many of the Kronecker delta values in Eqn. (19) and Eqn. (20) are zeros, and the first-order derivatives at $t=1$ and the second-order derivatives at $t=1$ or $t=2$ are all zeros. That is,

$$\begin{aligned} \frac{\partial \alpha_s(1)}{\partial a_{im}} &= \frac{\partial(\pi_s b_s(\mathbf{o}_1))}{\partial a_{im}} = 0 , \\ \frac{\partial^2 \alpha_s(1)}{\partial a_{im}\partial a_{jn}} &= \frac{\partial^2(\pi_s b_s(\mathbf{o}_1))}{\partial a_{im}\partial a_{jn}} = 0 , \\ \text{and} \quad \frac{\partial^2 \alpha_s(2)}{\partial a_{im}\partial a_{jn}} &= \sum_{r=1}^{N} \left[\frac{\partial^2 \alpha_r(1)}{\partial a_{im}\partial a_{jn}} a_{rs} + \frac{\partial \alpha_r(1)}{\partial a_{im}} \frac{\partial a_{rs}}{\partial a_{jn}} + \frac{\partial \alpha_r(1)}{\partial\, a_{jn}} \frac{\partial a_{rs}}{\partial a_{im}} \right] b_s(\mathbf{o}_2) = 0 . \end{aligned}$$

If the computational complexity of the Forward Procedure is denoted as $O(\text{fwd\_proc})$ then the computational complexity of the Hessian is about $J^2 \times O(\text{fwd\_proc})$ (where J, once again, is the number of transitions). Although the computation of Hessian is expensive, it can be mitigated by the following two observations:

- the Hessian is symmetric so that only about half of its elements are computed.
- in practice, most computation is actually due to the calculation of the observation probabilities $b_s(\mathbf{o}_t)$. In our experiments, we pre-computed the probabilities of all observations for all states which were then used by all iterations of OBS.

V. Experimental Evaluation

The OBS algorithm has two major assumptions: the *"extremal"* assumption that the starting model has been trained to convergence, and the *"quadratic"* assumption that the third and higher-order terms in the Taylor expansion of the cost function are negligible. As the OBS algorithm iterates, these assumptions will become weaker and weaker. In practice, researchers in the neural network community find that OBS works reasonably well on neural networks even

when these assumptions are violated. In this section, we designed two experiments to study the behaviour of OBS on pruning HMM transitions empirically.

Before we describe the two evaluation experiments, one procedural modification to the OBS algorithm is worth mentioning. Depending on the particular topology of the HMM in a problem, certain transition deletions will render some states in the reduced HMM *useless* — *unreachable* or *non-propagating*. This is particularly problematic if the unreachable state is the final (null) state of the HMM. Hence in all our experiments, the OBS algorithm of Algorithm 1 was amended with the following steps to deal with the possible appearance of useless states:

- if a deletion will render a state *unreachable* — an HMM state with no in-coming transitions, and/or *non-propagating* — an HMM state with no out-going transitions, then the deletion is undone and unit-OBS is carried out on the state instead.
- deletions that will render an HMM *infeasible* — that is, an HMM with an unreachable final null state — are not allowed. As a matter of fact, those deletions will not be selected unless there are no other choices; in such case, the algorithm will stop.

Moreover, we always made sure that the OBS algorithm started with a feasible HMM.

A. Evaluation I: Application of OBS to Pruning Transitions of a Discrete HMM

This experiment is similar to the one described in [18]. In the paper, an HMM was pruned iteratively, one transition at a time. During each iteration, all possible HMMs having one transition less than the current HMM were trained and the one with the highest likelihood was selected. In our current case, we replaced the HMM training of all possible smaller HMMs by finding the saliencies of all transitions (as defined in Eqn. (7)), and pruned the one with the least saliency. The objective is to check if the OBS algorithm is effective in pruning a "wrongly" overgrown model to the true model.

Specifically, 1,000 discrete observation sequences of lengths ranging from 3 to 103 were generated from the 3-state strictly-left-to-right discrete HMM of Fig. 3 for a total of 26,829 training observations. The state observation probabilities of the true generating discrete HMM are given in Table II. Then a 4-state ergodic discrete HMM of Fig. 4(a), which is larger than the true model, was trained using the 1,000 observation sequences. The ergodic HMM always starts from S1 and ends in S4[4]. The ergodic HMM was trained using the HTK software [41] with flat-start initialization until the model converged. The OBS algorithm was then performed iteratively, pruning one transition at a time, and adjusting the remaining transitions optimally.

[4]There is a minor difference between our 4-state ergodic HMM and the one described in [18]. Ours always starts from S1 while the latter may start in any of the four states.

Fig. 4 shows the evolution of the topology as well as the transition probabilities of the pruned HMM during the first seven iterations of the OBS algorithm. In the figure, only non-zero transitions are drawn; the number next to an arrow represents the transition probability between the two states that the arrow connects in the direction of the arrow. For example, Fig. 4(a) indicates that the HMM originally had a transition from S1 to S2 with a probability of 0.257 and a transition from S2 to S1 with a probability of 0.234. However, after one iteration of the OBS algorithm, the transition probability from S1 to S2 was reduced to 0.103, and the transition from S2 to S1 was deleted. The deletion of transitions at each OBS iteration and the corresponding least saliency are summarized in Table III.

We have the following observations from Fig. 4 and Table III:

- Although the OBS algorithm starts with the aim of deleting one transition at each iteration, it is possible that more than one transition is actually deleted. The requirement of a single transition deletion is only a constraint, and the subsequent modification of all the remaining transitions may suggest more deletions. This is the reason for the additional transition deletions at the first iteration.
- The saliencies should always be positive because of the extremal assumption. The very small negative saliency in the first iteration is due to round-off errors in the computation of the Hessian.
- In this particular example with one excessive HMM state, the extraneous state was modeled by the HMM training as a repeated state: S2 and S3 are basically identical with the same relationship with S1 and S4 throughout the first four iterations.
- One of the two repeated states (S2 and S3), S2, was finally pruned at the 5th iteration — all its in-coming and out-going transitions were pruned. The details of the deletion of S2 are described as follows: at the 5th iteration, the OBS algorithm originally suggests to prune the transition S1 $\rightarrow$ S2. However, the suggested deletion renders S2 unreachable. As a result, the deletion is undone and unit-OBS is invoked to delete S2 instead; thus, the three transitions, S1 $\rightarrow$ S2, S2 $\rightarrow$ S2, and S2 $\rightarrow$ S4 are pruned.
- There is a big increase in the saliency — the least induced drop in the log-likelihood of the training data — of the pruned transition at the 6-th iteration: from 58.1 in the 5th iteration to 20,800 in the 6th iteration. The big change may be used as an indication to stop HMM pruning.

Thus, we see that the OBS algorithm effectively "finds" the true HMM of the training data in this experiment after 5 iterations.

B. Evaluation II: Application of OBS to Pruning Transitions of Continuous Density HMMs

In this experiment, the OBS algorithm was evaluated on a real speech recognition task. We investigate if the most commonly used HMM topology — strictly left-to-right HMM with no skip-transitions as shown in Fig. 5 — in the ASR community is a good choice for the TIDIGITS recognition task. To do that, we added single-state skip-transitions to each state as shown in Fig. 1; thus, each digit HMM has about 47 transitions[5]. The hypothesis is that if skip transitions are not needed, they will be the first to be pruned by the OBS algorithm.

The adult data set of TIDIGITS was used for evaluation. It consists of 8,623 training utterances and 8,700 testing utterances. Whole digit HMMs were trained. Each digit HMM had 16 states with 16 Gaussian components per state. The HMM transitions are numbered as follows (as shown in Fig. 1):

- self transitions are numbered as 1, 4, 7, ..., 46.
- next-state transitions are numbered as 2, 5, 8, ..., 47.
- single-state skip-transitions are numbered as 3, 6, 9, ..., 48.

That is, if the transition number is n, then $n \bmod 3 = 0, 1, 2$ for single-state skip-transitions, self transitions, and next-state transitions respectively. In addition, there were a 3-state "sil" model to capture silence speech and a 1-state "sp" model to capture short pauses between digits.

Feature extraction was performed over a window of 25ms at a frame rate of 10ms over all training and testing data. The acoustic vector was the conventional 39-dimensional cepstral vector containing 12 mel-frequency cepstral coefficients (MFCCs) and normalized energy plus their first- and second-order derivatives. All the eleven digit models, the noise model, and the short pause model were trained by the EM algorithm until convergence. The training data were first force-aligned, and then the OBS procedure of Algorithm 1 was used to compute the saliency of each transition for each digit HMM *separately* and the transition with the minimum saliency was deleted from each HMM.

To gauge our results, we also tried to delete transitions *manually* from each HMM. In this manual pruning (MP) method, the saliency of a transition is the decrease in log-likelihood if it is deleted and the transition probabilities of the affected state is simply re-normalized; the transition with the minimum saliency is pruned. The major difference between OBS and MP

[5]Although the topology of each digit HMM was initialized with the structure in Fig. 1, subsequent HMM training might decide to remove those transitions that had very small probabilities. That indeed happened in our case. As a result, some of the digit HMM did not have exactly 47 transitions.

is that the latter will only update the remaining transition probabilities of the single affected state by simple re-normalization, while the former will update the probabilities of all remaining transitions of any states optimally. Thus, the choice of transition deletion made by MP is not expected as good as the one found by OBS. Looking from another perspective, the MP method replaces the two assumptions (extremal and quadratic assumptions) made by the OBS algorithm by the approximation that the remaining transitions of the pruned state may be done by simple re-normalization and all other transitions remain unaffected. Using the notations of Section IV, since each possible transition deletion requires running the Forward Procedure once to determine its saliency, the complexity of the MP method is about $J \times O(\text{fwd\_proc})$, where J is the number of transitions and $O(\text{fwd\_proc})$ is the complexity of the Forward Procedure. Thus, it seems that MP is about J times faster than OBS. However, in practice, OBS may compute the whole Hessian in one pass of the training data and determine which transition to prune, while MP will have to go through J passes of the data. Since the data, in general, have to be read from files, MP requires J times more file I/O than OBS. In practice, we found that the OBS method was slightly faster than the MP method in this experiment.

B.1 Recognition Evaluation

The generalization performance of the reduced models after each iteration of the OBS algorithm and manual pruning method is plotted in Fig. 6. Notice that since it is possible to delete more than one transition at one iteration should a useless state appear and the OBS algorithm will then invoke the unit-OBS procedure to prune a whole state instead of a single transition, the average number of deletions (instead of the number of iterations) in each HMM is used in the abscissa in Fig. 6. We have the following observations:

- OBS not only reduces the topology of the digit HMMs but also improves their generalization performance on unseen test data during the first 13 OBS iterations, in which about 15 or 31% transitions are deleted from each of the 11 digit HMMs.
- During the first 9 iterations, the generalization performance only improves slowly. On closer examination of the deleted transitions, we find that they all have small magnitudes.
- The best performance occurs when about 11 transitions are deleted from each HMM, giving a recognition accuracy of 99.4%. Compared with the baseline accuracy of 99.2%, there is a word error rate reduction of 25%; the result is statistically better than the original model at the 0.05 confidence level.
- The OBS performance drops significantly after 13 iterations (or about 15 transition deletions per model). Again closer scrutiny reveals that useless states started to appear after 13 iterations

in the models of "oh", "1", "3" and "5", and some OBS iterations were replaced by unit-OBS iterations on those models. It seems that unit-OBS may have induced drastic changes in the models, resulting in greater deviation from the OBS assumptions.

• In general, the performance of OBS is only slightly better than that of the simpler MP method. This seems to confirm the finding in the automatic speech recognition (ASR) community that the transition probabilities are not very helpful in ASR because their dynamic range is much smaller than that of the observation probabilities. Nevertheless, the performance of OBS is consistently better than that of MP, probably due to the optimal update of the remaining transitions after each transition deletion. The result suggests that it is always better (even if by a small margin) to estimate the transition probabilities in a manner consistent with the HMM modeling theory. On the other hand, the performance of both algorithms starts to degrade at roughly the same time, but OBS degrades faster. Again it may be caused by the unit-OBS iterations after the 13th iteration.

B.2 Analysis of the Deleted Transitions

We also examine the three types of transitions that were deleted during the first 11 iterations of the OBS algorithm over the 11 digit HMMs. The result is summarized in Table IV. As mentioned in the beginning of this Section, the numbers of self-transitions, next-transitions, and skip-transitions are not 176, 176, and 165 respectively as one may have deduced from the initial HMM topology shown in Fig. 1. The reason is that after HMM training, some transitions were too small and were removed.

During the first 11 iterations, most of the deleted transitions are the single-state skip-transitions as one would expect, except for the digit "oh" which has 9 next-state transitions deleted. This suggests that fewer states are needed to model "oh". When we check the pruning results of model "oh" in the subsequent OBS iterations up to the 15th iteration, we find that unit-OBS was invoked eight times due to the appearance of many useless states. At the end, only eight states of the "oh" HMM remained in a strictly left-to-right HMM topology as shown in Fig. 7. On the other hand, among the remaining ten digits, only six states were deleted in the subsequent four iterations. This is reasonable as "oh" consists of only one single phoneme which probably does not require as many states as other digits. Altogether, the OBS pruning results confirms that the strictly left-to-right HMM topology with no skips, which is most commonly used in current ASR, is sufficient.

B.3 Savings in Computation

A side effect of HMM pruning is savings in computation time and memory. Fig. 8 shows the relationship between the number of deleted transitions or states and the decrease in decoding CPU time on the test set using the pruned HMMs obtained after 11–16 OBS iterations. In general, if N is the number of states and T is the duration of the decoding speech, the complexity of the Viterbi decoding algorithm is $O(N^2T)$. However, in our current case with a left-to-right HMM plus 1-state skip-transitions, the complexity is actually $O(3NT)$ which is linearly related to the number of states. This explains why the decoding time decreases linearly with the number of state deletions. Thus, after 13 OBS iterations, we have a smaller set of HMMs with no degradation in accuracy but 5.3% savings in computation time; after 16 OBS iterations, the set of pruned HMMs reduces decoding time by about 10% at the expense of 0.25% (absolute) drop in accuracy.

VI. Conclusions

In this paper, we adopt the theory of optimal brain surgery (OBS) to prune state transitions in an HMM. Although the theory of OBS had been well studied in the neural network community, its application to an HMM requires significant modification, resulting in a quadratic programming problem. We designed two experiments to study its empirical behaviour. In the first experiment, the OBS algorithm was able to "re-discover" the true topology of an overgrown discrete HMM. In the second experiment, on the TIDIGITS task, the OBS algorithm successfully pruned the digit models so that the resulting models generalize better than their initial models on unseen test data: the word error rate is reduced by 25% after 11 OBS iterations when 25% of the HMM transitions are pruned. It also suggests that the common strictly left-to-right HMM topology is reasonably good, except that the model "oh" may perhaps use fewer states than the other digits.

This paper represents our first study of applying OBS to pruning an HMM. The algorithm can be further improved in several directions. Firstly, as more and more transitions are pruned, the OBS assumptions will become less valid. This may be mitigated by model re-training. Another reason for re-training is that the OBS algorithm does not modify the state observation probabilities. As the model is pruned, the initial observation probabilities will no longer be optimal. Since model re-training is computationally expensive, it should be done only when it is necessary. For example, when a transition deletion will render a state useless, it is now replaced by an unit-OBS step; one may as well replace it by deleting the affected state and then re-

training the model. Secondly, right now all the models in a problem are pruned synchronously in the sense that each model goes through one iteration of OBS at the same time. The OBS algorithm may also be run asynchronously, so that some models are allowed to be pruned more aggressively than the others. Thirdly, it will be interesting to apply the OBS or unit-OBS algorithm to more complex HMMs, such as product HMMs commonly used in multi-band ASR and audio-visual ASR.

VII. Acknowledgements

This work is supported by the Hong Kong Research Grants Council under the grant number HKUST6201/02E.

References

[1] J. K. Baker, "The Dragon System: An overview," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, no. 1, pp. 24–29, February 1975.

[2] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of IEEE*, vol. 64, pp. 532–536, 1976.

[3] K. F. Lee and H. W. Hon, "Large-vocabulary speaker-independent continuous speech recognition using HMM," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1988, pp. 123–126.

[4] R. Nag, K. H. Wong, and F. Fallside, "Script recognition using hidden Markov models," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Japan, April 1986, vol. 3, pp. 2071–2074.

[5] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, and K. S. Nathan, "A fast statistical mixture algorithm for on-line handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 12, pp. 1227–1233, 1994.

[6] A. Kundu, Y. He, and P. Bahl, "Recognition of handwritten word: first and second order hidden Markov model based approach," *Pattern Recognition*, vol. 22, no. 3, pp. 283–297, 1989.

[7] D. Haussler, A. Krogh, S. Mian, and K. Sjölander, "Protein modeling using hidden Markov models: Analysis of globins," in *In Proceedings of the 26th Annual Hawaii International Conference on System Sciences*, T. N. Mudge, V. Milutinovic, and L. Hunter, Eds. 1993, pp. 792–802, IEEE computer Society Press.

[8] B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 307–309, March 1986.

[9] S. S. Chen and P. S. Gopalakrishnan, "Clustering via the Bayesian information criterion with applications in speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1998, pp. 645–648.

[10] Y. C. Chan, M. Siu, and B. Mak, "Pruning of state-tying tree using Bayesian information criterion with multiple mixtures," in *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China, 2000, vol. IV, pp. 294–297.

[11] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[12] H. Akaike, "A new look at statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[13] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[14] C. S. Wallace and D. Boulton, "An information measure for classification," *The Computer Journal*, vol. 11, no. 2, pp. 195–209, 1968.

[15] A. Biem, J. Y. Ha, and J. Subrahmonia, "A Bayesian model selection criterion for HMM topology optimization," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002, vol. 1, pp. 13–17.

[16] J. Takami and S. Sagayama, "A successive state splitting algorithm for efficient allophone modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992, vol. 1, pp. 573–576.

[17] H. Singer and M. Ostendorf, "Maximum likelihood successive state splitting," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1996, vol. 2, pp. 601–604.

[18] F. C. Vasko, Jr., A. El-Jaroudi, and J. R. Boston, "An algorithm to determine hidden Markov model topology," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 1996, vol. 6, pp. 3577–3580.

[19] F. Casacuberta, E. Vidal, B. Mas, and H. Rulot, "Learning the structure of HMM's through grammatical inference techniques," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1990, vol. 2, pp. 717–720.

[20] A. Stolcke and S. Omohundro, "Hidden Markov model induction by Bayesian model merging," in *Advances in Neural Information Processing Systems 5*, pp. 11–18. Morgan Kaufmann, 1993.

[21] C. Wooters and A. Stolcke, "Multiple-pronunciation lexical modeling in a speaker independent speech understanding system," in *Proceedings of the International Conference on Spoken Language Processing*, 1994, vol. 3, pp. 1363–1366.

[22] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds., San Mateo, CA, 1993, vol. 5, pp. 164–171, Morgan Kauffman.

[23] B. Hassibi and D. G. Stork, "Optimal brain surgeon and general network pruning," in *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, April 1993, pp. 293–300.

[24] B. Logan and P. Moreno, "Factorial HMMs for acoustic modeling," in *Proceedings of the International Conference on Spoken Language Processing*, 2000, vol. 3, pp. 143–146.

[25] H. J. Nock and S. J. Young, "Loosely coupled HMMs for ASR," in *Proceedings of the International Conference on Spoken Language Processing*, 2000, vol. 3, pp. 143–146.

[26] N. Mirghafori, *A Multi-Band Approach to Automatic Speech Recognition*, Ph.D. thesis, Department of Computer Science, University of California, Berkeley, January 1999.

[27] Y. C. Tam and B. Mak, "Development of an asynchronous multi-band system for continuous speech recognition," in *Proceedings of the European Conference on Speech Communication and Technology*, Aalborg, Denmark, 2001, vol. 1, pp. 575–578.

[28] S. Nakamura, "Statistical multimodal integration for audio-visual speech processing," *IEEE Transactions on Neural Networks*, vol. 13, pp. 854–866, July 2002.

[29] T. Ragg, H. Braun, and H. Landsberg, "A comparative study of neural network optimization techniques," in *Proceedings of ICANNGA*. 1997, Springer-Verlag.

[30] L. K. Saul and M. I. Jordan, "Boltzmann chains and hidden Markov models," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds., vol. 7, pp. 419–426. The MIT Press, 1995.

[31] M. W. Pedersen and D. G. Stork, "Pruning Boltzmann networks and hidden Markov models," in *Record of the Thirtieth Asilomar Conference on Signals, Systems and Computers*, Nov 1996, vol. 1, pp. 258–261.

[32] Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed., San Mateo, CA, 1990, vol. 2, pp. 598–605, Morgan Kauffman.

[33] Pierre van de Laar and T. Heskes, "Pruning using parameter and neuronal metrics," *Neural Computation*, vol. 11, pp. 977–993, 1999.

[34] A. Stahlberger and M. Riedmiller, "Fast network pruning and feature extraction using the unit-OBS algorithm," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., Cambridge, MA, 1997, vol. 9, pp. 655–661, The MIT Press.

[35] A. U. Levin, T. K. Leen, and J. E. Moody, "Fast pruning using principal components," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., San Mateo, CA, 1994, vol. 6, pp. 35–42, Morgan Kauffman.

[36] M. W. Pedersen, L. K. Hansen, and J. Larsen, "Pruning with generalization based weight saliencies: γOBD, γOBS," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds., Cambridge, MA, 1996, vol. 8, pp. 521–527, The MIT Press.

[37] G. Castellano, A. M. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 519–531, May 1997.

[38] V. Tresp, R. Neuneier, and H. G. Zimmermann, "Early brain damage," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., Cambridge, MA, 1997, vol. 9, p. 669, The MIT Press.

[39] R. Fletcher, *Practical Methods of Optimization*, chapter 10, John Wiley & Sons, Reading, Massachusetts, 2nd edition, 1987.

[40] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Annals of Mathematical Statistics*, vol. 41, pp. 164–171, 1970.

[41] Steve Young et al., *The HTK Book (Version 3.2)*, University of Cambridge, 2002.

TABLE I

A comparison of the application of OBS to a neural network and to an HMM.

Item	NN	HMM
Pruning Unit	connection weight	transition arc
	NN node	HMM state
Cost Function	training error	log-likelihood of training data
Problem Type	Lagrange optimization	quadratic programming

TABLE II

Discrete observation probabilities of the HMM in Fig. 3.

k-th Symbol	State 1	State 2	State 3
1	0.375	0.125	0.125
2	0.375	0.375	0.125
3	0.125	0.375	0.375
4	0.125	0.125	0.375

TABLE III

PRUNED TRANSITIONS AND STATES DURING EACH OBS ITERATION IN EVALUATION 1.

Iteration	Pruned Transition (Additional Deletions; Reason)	OBS Saliency
1	S2 → S3 (S2 → S1, S3 → S1, S4 → S2, S4 → S3; after probabilities re-adjustment)	-5.96e-13
2	S3 → S2	4.38e-16
3	S1 → S4	8.62e+00
4	S4 → S1	1.32e+00
5	S1 → S2 (S2 → S2, S2 → S4; unit-OBS on S2)	5.81e+01
6	S1 → S1	2.08e+04
7	S3 → S3	6.36e+04

TABLE IV

TYPES OF HMM TRANSITIONS DELETED BY OBS DURING THE FIRST 11 ITERATIONS IN EVALUATION II.

Types	Self-transition	Next-transition	Skip-transition
Original no.	174	176	148
#Deletions	13	15	93
Deletion%	$\frac{13}{174} = 7.47\%$	$\frac{15}{176} = 8.52\%$	$\frac{931}{148} = 62.8\%$

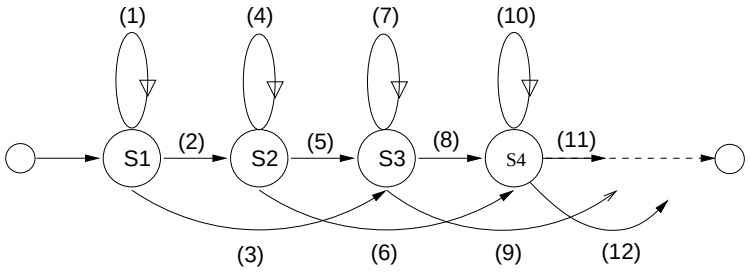

Fig. 1. A left-to-right HMM with single-state skips. Only 4 states are shown and the transitions are numbered.

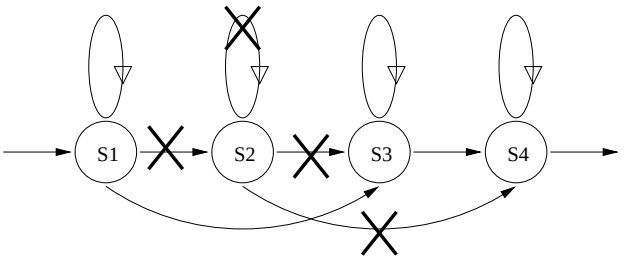

Fig. 2. Deletion of multiple transitions in unit-OBS.

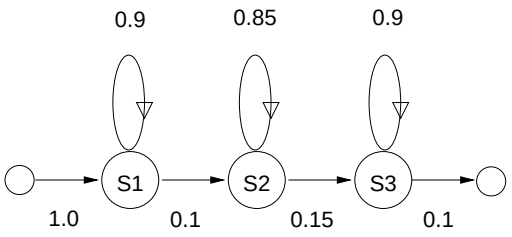

Fig. 3. A strictly left-to-right 3-state discrete HMM

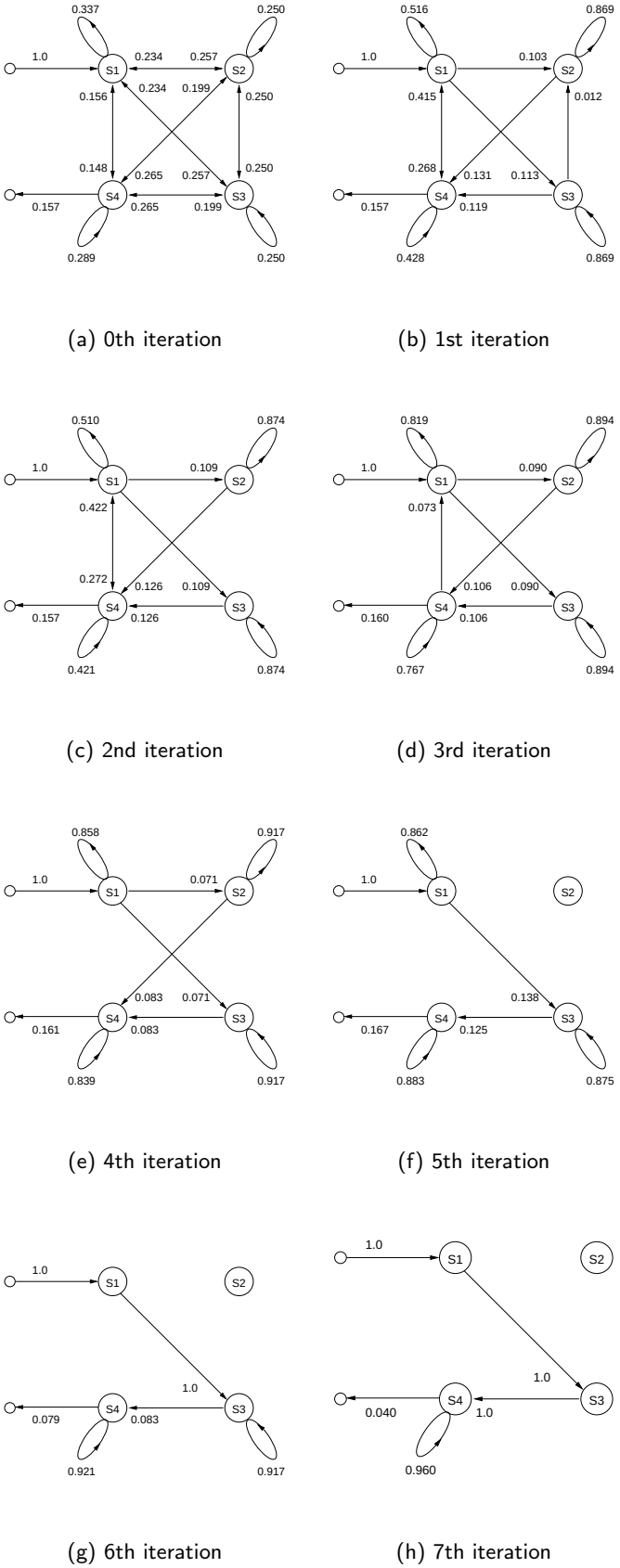

Fig. 4. Evolution of the OBS algorithm on pruning the discrete HMM in Evaluation I.

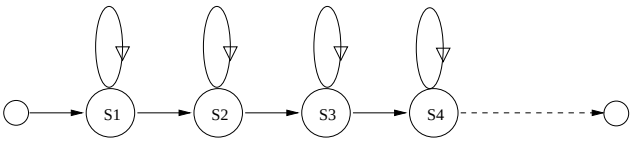

Fig. 5. A left-to-right HMM with no skips. Only 4 states are shown.

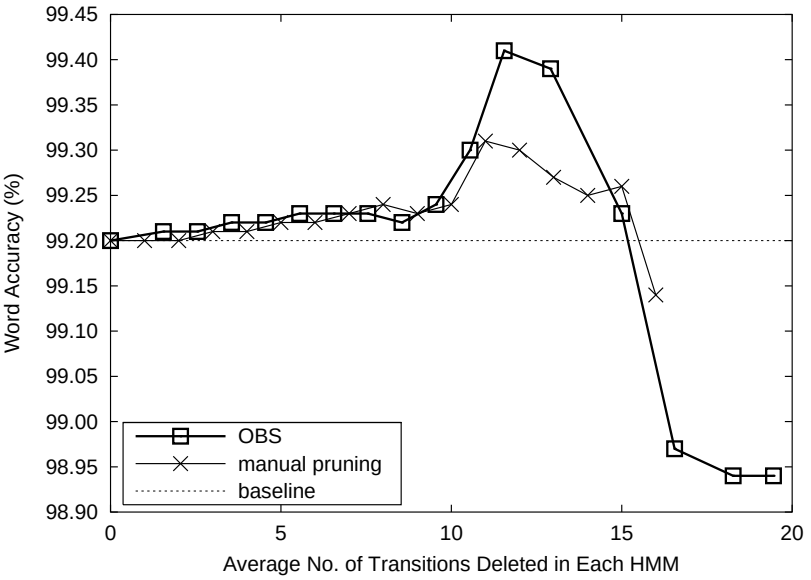

Fig. 6. Recognition performance on TIDIGITS after pruning HMM transitions by OBS.

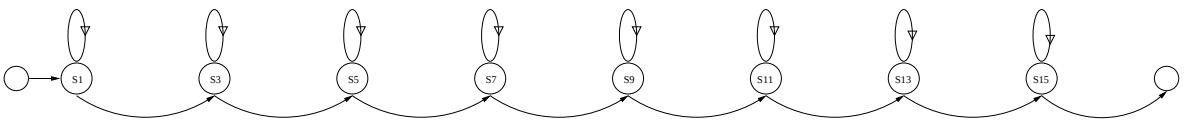

Fig. 7. The topology of HMM "oh" after 15 iterations of OBS.

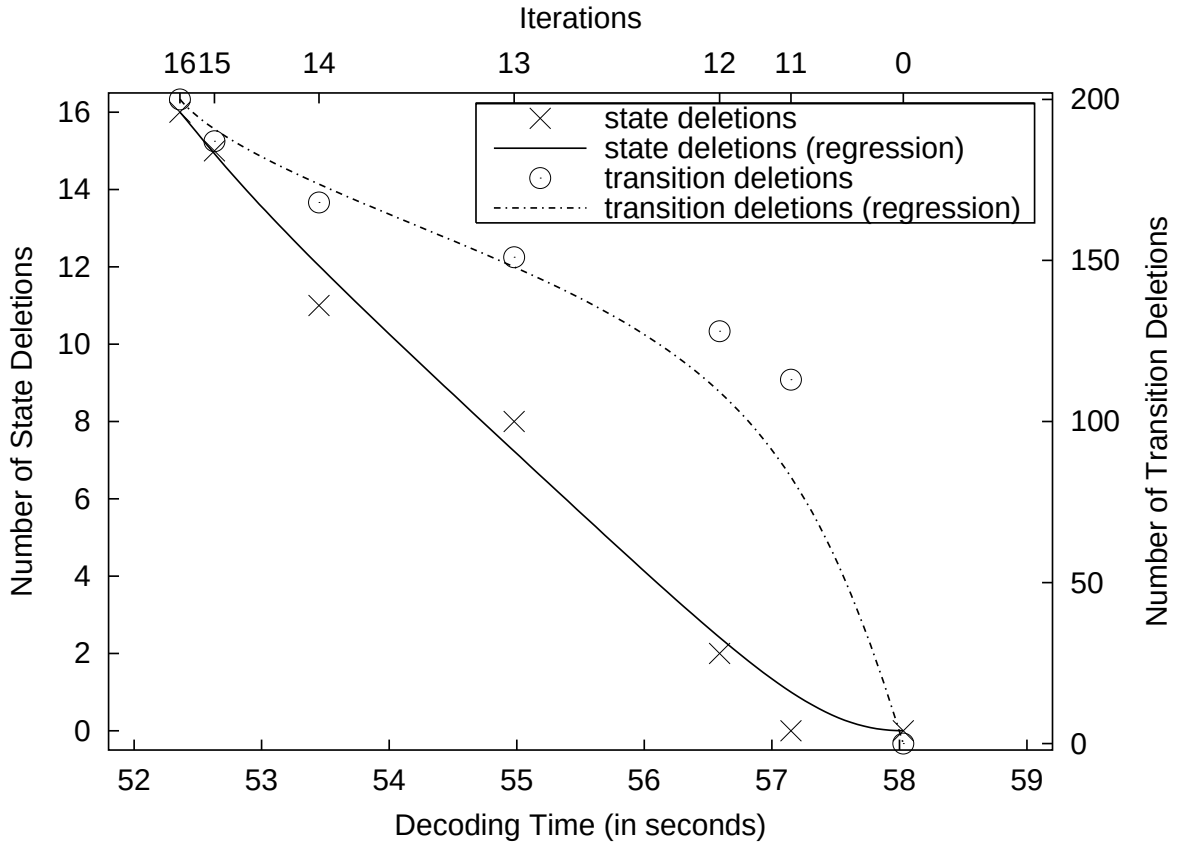

Fig. 8. Savings in computation time by OBS.

List of Tables

List of Figures