Pruning of Hidden Markov Model with Optimal Brain Surgeon

by

CHAN Kin Wah

A Thesis Submitted to

The Hong Kong University of Science and Technology

in Partial Fulfillment of the Requirements for

the Degree of Master of Philosophy

in Department of Electrical and Electronic Engineering

May, 2003, Hong Kong

Authorization

CHAN Kin Wah

Pruning of Hidden Markov Model with Optimal Brain Surgeon

by

CHAN Kin Wah

This is to certify that I have examined the above MPhil thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

Prof. Manhung SIU, Thesis Supervisor

Prof. Brian MAK, Thesis Co-Supervisor

Prof. Bertram SHI, Thesis Examination Chairman

Prof. James KWOK, Thesis Examination Examiner

Prof. Khaled BEN LETAIEF, Acting Head of Dept. of EEE

Department of Electrical and Electronic Engineering

May 2003

Acknowledgments

I should thank many people for helping me and encourage me to complete my study. First of all, I should thank my supervisors, Dr. Manhung Siu and Dr. Brian Mak, for giving me an opportunity to study my MPhil in HKUST. Dr. Siu taught me what speech technology is and supervised me during the final year project. Dr. Mak has been my guidance as well as my partner in research for three years. They are one of the most important teachers in my life.

Thanks God for guiding me through the path that I have ran. To complete this thesis, I have received many helps from the teammates of LASTRE. Thanks Wilson Tam and Franco Chong for providing the recognizer and sharing the experience of multi-band recognition. Thanks my family for understanding and supporting me during this period. Lastly, I would like to give a thousand thanks to Karen for her unlimited support. I would not finish my work without her encouragement.

I hope this thesis can make contribution to the Speech recognition community. Perhaps the contribution maybe minimal, I am pleased for having the chance of participation.

Contents

List of Figures

List of Tables

List of Algorithms

Pruning of Hidden Markov Model with Optimal Brain Surgeon

by CHAN Kin Wah

Department of Electrical and Electronic Engineering

The Hong Kong University of Science and Technology

Abstract

In the training process of hidden Markov model (HMM), the topologies of HMMs, which includes the number of states and the connectivity of states, are usually pre-set based on experience or heuristic. However, it is uncertain whether one topology is optimal in one particular task. Many model complexity measures have been proposed for model selection such as Bayesian information criterion, minimum description length, and minimum message length.

In this thesis, an alternate approach of reducing complexity of the well-trained HMMs is proposed. The classical neural network weight pruning technique, called the *Optimal Brain Surgeon* (OBS), is adopted to pruning HMMs. In the method, an HMM component is first pruned if it results in a minimal decrease in total log-likelihood of the training data. The decrease in total log-likelihood is approximated by the Taylor's Series with the second derivatives information in total log-likelihood function.

The experimental evaluation showed that pruning HMM transitions with OBS method is able to reduce the topology of well trained, though perhaps over-fitted, HMMs successfully. The algorithm removed transitions optimally and, as a result, the memory and computation costs was reduced and the recognition performance of the pruned HMM on unseen test data was also improved in some cases (and did not get worse in all cases).

The OBS method on HMM transition deletion is further extended to HMM state deletion by removing all incoming and outgoing transitions of the state simultaneously. This method has greatly reduced the computation cost of OBS as well by deleting multiple transitions in a single iteration. It was applied to a more complex HMM that was used in the multi-band automatic speech recognition. It showed that OBS in deleting states saves memory and computation costs by reducing the model complexity and is more efficient than OBS on transition deletion since it requires fewer number of iterations.

Chapter 1

Introduction

In recent years automatic speech recognition (ASR) technology has been able to achieve reasonable recognition performance and has been applied in many real-world applications. For example, ASR has been widely used in command and control systems, dictation software and telephony applications. The usage of the hidden Markov model (HMM) in ASR is one of the most important and dominant technologies used in speech research community. HMM provides a successful framework in modeling quasi-periodic speech signals and its recognition performance has been improved by many extensions. Nowadays, the trend of ASR research shifts to handle more complicated tasks with HMMs such as hybrid type of HMMs [1]. This requires a larger and more complex acoustic model with more parameters and the model structures of these new algorithms are neither simple nor trivial compared to simple recognition tasks.

One example which involves a complex topology is factorial HMM [2, 3]. It has recently been applied into audio-visual ASR and multiband ASR recently. Factorial HMM combines state sequences from many layers where the layers can be sub-band features of speech signal in multi-band ASR or audio stream and video stream in audio-visual ASR depending on the applications. It also defines a new term, meta-state S_t, which is composed of M layers of state sequences $S_t = S_1^{(1)}, \cdots, S_t^{(M)}$. When the transitions of K states in each layer is unrestricted, the dimension of the transition matrix in factorial HMM will be $K^M \times K^M$.

Another example which also involves a complex topology of HMM is multi-band automatic speech recognition (MBASR) [4, 5]. The target of MBASR is the robustness of speech recognition in noisy environments[6]. Researchers introduced composite HMM framework [7] to combine multiple HMMs which are trained for different frequency bands (see Figure 1.1) so that the multi-band HMM can be interpreted as an ordinary HMM. However, the composite HMM leads to a huge parameter size and complicated topology. For example, if two sub-band HMMs with n states are combined as a composite HMM with full asynchrony, the composite MBHMM contains n^2 number of states. These examples of HMM extensions show a common characteristic: both of the approaches construct new topologies of HMMs that require huge parameter size in modeling.

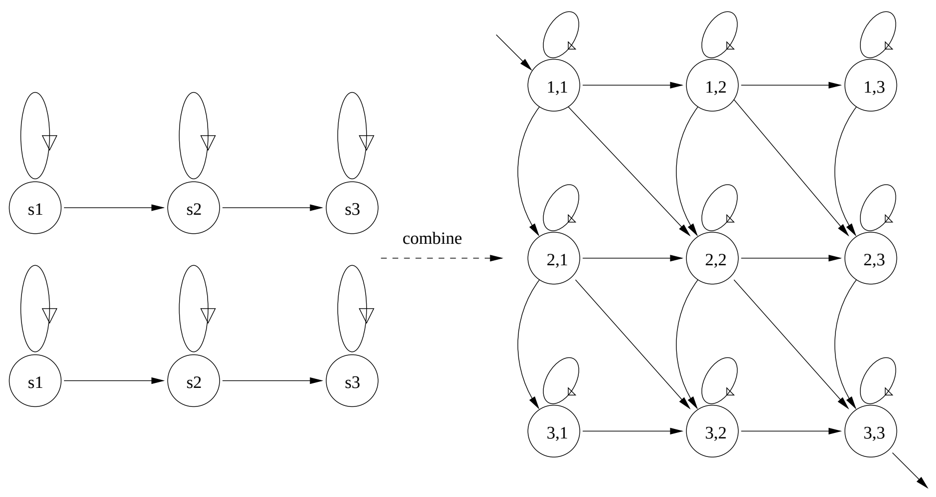

Figure 1.1: Composite HMM generated by combining 2 HMMs with 3 states respectively

Traditionally, the topology of an HMM such as the number of states and the connectivity of states are usually pre-defined by experience or heuristic. In a simple recognition task, a strictly left-to-right topology is usually employed since this topology has already performed satisfactorily in many ASR tasks. However, it is uncertain whether one topology is optimal for a specific task. The optimal topology mentioned is the one that gives the minimum error with the minimal parameter size and model complexity. Finding the optimal topology of HMM belongs to problem of model selection. This is an active research topic that focuses on the tradeoff between training error and model complexity[8]. It tries to determine an optimal balance in the dilemma of modeling: if the model has too many parameters which allow a higher degree of freedom, it may over-fit the training data; but, if the model has too few parameters, it may not have enough power to learn the training data well and, as a result, the recognition performance degrades because the model is unable to represent the training data accurately (Figure 1.2).

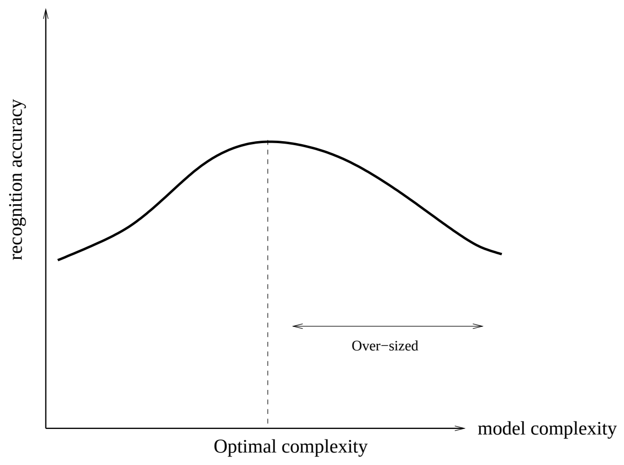

Figure 1.2: Relationship between model complexity and recognition performance

1.1 Model selection methods

In more complex recognition tasks such as the multiband ASR and factorial HMM mentioned above, the requirements for computation power and memory usage are greatly increased because the size of model is increased non-linearly. In such cases, finding the optimal topology or optimal model selection becomes more critical in order to reduce computation cost and memory usage. More importantly, a model that has a large parameter size may suffer from the over-fitting problem. Generally, there are at least two approaches which can be used to solve the regularization/generalization problem:

- In the model selection area, there are many well-known information-theoretic metrics for model selection. Some examples are Bayesian information criterion (BIC) [9], Akaika information criterion (AIC), minimum description length (MDL) [10] and minimum message length (MML) [11, 12].

- A data driven method which refines HMM topology by training data is also a possible solution of tackling generalization problem. J. Takami at el. proposed a successive state splitting algorithm [13, 14] and Singer at el. proposed a similar idea [15] based on the maximum likelihood criterion for optimizing HMM topology. Both approaches grow the topology to a optimal size from a small HMM according to their optimization criteria.

1.2 Proposed solution - Optimal Brain Surgeon

To tackle the risk of over-fitting in complex HMMs and to determine the optimal HMM topology for ASR tasks, a well-known neural network weight pruning technique, *Optimal Brain Surgeon* (OBS) [16], is adopted into topology pruning in the

HMM framework. Instead of growing or splitting of HMM states from a small topology [13, 15, 17], OBS achieves the same goal, but in the opposite way. It starts pruning from a sufficiently large topology until it reaches the optimal size according to some optimization criteria. This approach does not only address the problem of finding an HMM topology which is optimal in a particular recognition task, but also corrects and refines an over-fitted HMM topology by removing the least important parameter(s) such that the costs of memory and computation can be reduced and, in some cases, the generalization performance can be improved. This is particularly important for the state-of-the-art applications of HMM as they are usually more complex and yet the operating platforms such as hand-helds or wireless devices may have limited computation capacity. The balance of performance and computation cost is also a critical issue.

In this thesis, our focus is the removal of the connectivity between states; that is the transition probabilities of HMM. Theoretically, Optimal Brain Surgeon (OBS) deletes HMM state transition(s) so that it minimizes the loss on the cost function. In the implementation of OBS algorithm, the model topology is reduced by pruning away transition(s) that is(are) the least important. The saliency of each transition in an HMM is represented by the approximated change in the total log likelihood of the training data. The remaining transitions are re-adjusted so that the decrease in the total log likelihood is maximally compensated.

1.3 Outline of thesis

In this chapter, the thesis objectives and the basic idea of the OBS theory are stated. The rest of the chapters are divided into three parts: the background of OBS, the theory and formulation of OBS on the HMM framework and the experimental evaluations.

The first part of thesis, which give the background of OBS, is described in Chapter 2. The chapter is a literature review on OBS and its predecessor, Optimal Brain Damage (OBD). Since OBS is an improvement over OBD and it shares the same basis of OBD, we will explain the theory of OBD first. After that, we will cover the introduction of OBS.

In Chapter 3 and 4, we will present the theory and the formulation of OBS on the HMM framework. While OBS of pruning neural network weights is not subjected to any constraints, the OBS applications on the HMM framework is a constrained optimization problem. In Chapter 3, we will show that it can be formulated as a quadratic programming optimization problem subjected to both equality and inequality constraints. The usage of Lagrange multiplier and Kuhn Tucker (KT) conditions will be discussed. Chapter 4 is a short chapter which details the step in the calculation of the Hessian matrix and shows all key equations required in the calculation. It is derived from the first principle based on the forward-backward algorithm and Hessian terms are calculated from the differentiation of forward probabilities.

Chapter 5 gives details of the experimental setup and the evaluation result. The experiments will be focused on the impact of the OBS pruning algorithm on recognition accuracy and the change in the computation cost after pruning. In the experiments, both transition pruning and state pruning is evaluated with left-to-right HMM in digit recognition task and composite HMM in multi-band recognition task respectively. Finally, conclusion of the experiment results and contributions of this thesis is made in Chapter 6.

Chapter 2

Background of Optimal Brain Damage and Optimal Brain Surgeon

Before explaining the theory underlying the work in this thesis, this chapter will give a literature review on Optimal Brain Damage (OBD) [18] and Optimal Brain Surgeon (OBS) [16]. Both OBD and OBS are classical weight pruning techniques in the neural network community [19]. This chapter covers:

1. an introduction to OBD and OBS in the neural network (NN) field
2. the detailed formulation of OBS
3. a comparison of OBD, OBS and other methods

Since OBS is improved from the OBD method, this chapter will explain the theory of OBD before discussing the OBS theory. The ideas and the formulations of both methods will also be discussed. At the end of this chapter, a further-improved OBS technique, the Unit-OBS, is briefly introduced as it is important for developing the OBS in deleting states in the HMM framework.

2.1 History of Optimal Brain Damage and Optimal Brain Surgeon

The traditional neural network pruning method for dealing with the generalization problem is the magnitude based method. Weights of smaller magnitude in the network are considered to be less needed. However, this concept is non-optimal since network weights with small magnitudes sometimes are important in the learning procedure.

In 1990, Le Cun, Denker and Solla derived an information theoretic technique for pruning unnecessary neural network weights which is named as Optimal Brain Damage (OBD) [18]. The name refers to the technique that performs weight removal (damage) in neural networks (brain) optimally in order to reduce computation cost and , in some cases, the removal also improve the generalization performance. Theoretically, the weight removal is based on the second derivative information of the error function in the training set. The results showed that OBD method can delete over half of the network weights so that the network's speed improved significantly and its recognition accuracy increased slightly. Optimal Brain Surgeon (OBS) [16, 20], a descendant of the OBD technique was proposed by Hassibi and Stork (1993). It further imposes the constrained optimization idea into the prediction of change in the training error, given that a network weight is removed. Thus, instead of finding which network weight is the least important to be removed, it also re-weight other weights in a optimal way. Furthermore, OBS's weight removal is considered more optimal than OBD since OBS used a diagonal Hessian approximation while OBS uses a full Hessian.

In later years, several variants of OBD/OBS have been proposed. For example, the Early Brain Damage [21] which re-formulates the weight deletion solution without the assumption of a converged network and Unit-OBS [22] which makes further improvements on the efficiency of algorithm when a whole network unit is deleted.

2.2 Optimal Brain Damage

Optimal Brain Damage is the first neural network pruning technique which uses the information of the second order derivatives of the training error function. The idea of OBD is simple: it predicts the change in training error by approximating the training error function with the Taylor series and deletes weight so that the change in the training error is maintained as small as possible. Thus, for each network weight, OBD method measures its corresponding importance - also called the *saliency* - by measuring its influence on the training error. The saliencies are sorted and the least important weights, which are indicated by their low saliencies are removed.

2.2.1 Taylor Series

The first step in the formulation of OBD is to approximate the change in the error function (objective function) using the Taylor series. Recall from the basic principle that any continuous function of order $(n+1)^{th}$ and is differentiable can be expressed as a Taylor series with n^{th}- order derivatives about a point x^*. This case is illustrated by Equation 2.1. It is certain that higher-order terms in the series are less significant in approximating a function $f(x)$ about a point. In Figure 2.1, it is shown that the approximation is good around the point x^*, but it diverges from the true error function when it moves away from x^*.

$$f(x) = f(x^*) + \frac{f'(x^*)}{1!}(x - x^*) + \frac{f''(x^*)}{2!}(x - x^*)^2 + \cdots \tag{2.1}$$

The objective function of the OBD is the mean squared error of the neural network in the training process. Let's denote the target network weights in a converged neural network by the vector $\mathbf{w}^*$. The objective function - the training error - and the change in training error are denoted as $E(\mathbf{w})$ and δE respectively. The Taylor expansion of change in the training error is given by

$$\begin{aligned} \delta E &= E(\mathbf{w}) - E(\mathbf{w}^*) \\ &= \frac{E'(\mathbf{w}^*)}{1!}(\mathbf{w} - \mathbf{w}^*) + \frac{E''(\mathbf{w}^*)}{2!}(\mathbf{w} - \mathbf{w}^*)^{\mathbf{2}} + \cdots \\ &= \mathbf{g}^{\mathbf{T}} \cdot \delta\mathbf{w} + \frac{1}{2}\delta\mathbf{w}^T \cdot \mathbf{H} \cdot \delta\mathbf{w} + \cdots \end{aligned} \tag{2.2}$$

In equation 2.2, $\mathbf{g}$ is the gradient vector with elements $g_i = \frac{\partial E}{\partial w_i}$and $\mathbf{H}$ is the Hessian matrix with elements $h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$. An alternative representation of equation 2.2 is

$$\delta E = (\frac{\partial E}{\partial \mathbf{w}})^T \cdot \delta\mathbf{w} + \frac{1}{2} \cdot \delta\mathbf{w}^T \cdot \frac{\partial^2 E}{\partial \mathbf{w}^{\mathbf{2}}} \cdot \delta\mathbf{w} + \cdots \tag{2.3}$$

2.2.2 Assumptions of OBD

With equation 2.2 expanded with Taylor series, the following is the formulation of the saliency term which indicates its corresponding change in error if a weight is removed. It should be noted that the calculation of the Hessian matrix $\mathbf{H}$ is an expensive process. If there are N weights in the network, the total number of distinct elements in Hessian matrix will be $\frac{1}{2}N^2$(the number of elements is reduced by half because the Hessian matrix is symmetric). Modern applications of neural

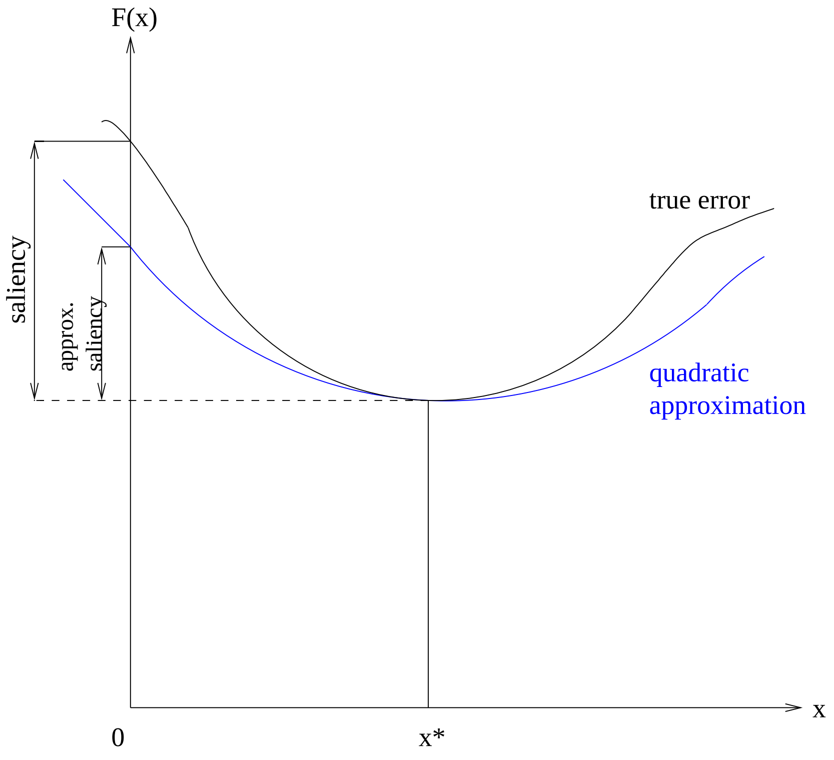

Figure 2.1: Taylor series approximates the error function about a point $\mathbf{x}^*$

network contains thousands of weights and its calculation dominates in the weight pruning algorithm. To solve this computation problem, Le Cun et al. introduced three further simplifying approximations [18].

2.2.2.1 The diagonal approximation

It assumes that the change in training error δE is caused by deleting each weights individually. The total δE term is the sum of each δE induced by deleting each weight individually. All cross terms in the Hessian matrix are negligible so that the Hessian matrix is diagonal. This assumption can greatly reduce the number of elements in Hessian to the dimension of the weight vector. Equation 2.2 then becomes

$$\delta E = \mathbf{g}^{\mathbf{T}} \cdot \delta \mathbf{w} + \frac{1}{2} \sum_i h_{ii} \delta w_i^2 + \cdots \tag{2.4}$$

2.2.2.2 The extremal approximation

In the OBD method, its target neural network is assumed to be well-trained. That is, the network training has converged to the (local) minimum of the error function. Mathematically, the gradient term in Equation 2.2 is zero at the converged point $\mathbf{w}^*$. Therefore, the first term in the Taylor series can be ignored, and the change in error function becomes

$$\delta E = \frac{1}{2} \sum_i h_{ii} \delta w_i^2 + \cdots \tag{2.5}$$

2.2.2.3 The quadratic approximation

In a Taylor series, it approximates the objective function around a point $\mathbf{w}^*$. If we assume that the error function is quadratic, terms of order greater than 2 are ignored. Finally, Equation 2.2 is further reduced to

$$\delta E = \frac{1}{2} \sum_i h_{ii} \delta w_i^2 \tag{2.6}$$

2.2.3 Algorithm of OBD

With the formulation of δE, the OBD is summarized in Algorithm 1.

Two criteria maybe used to terminate the OBD algorithm. The first one is setting a saliency threshold. Only the network weights that have saliencies less than the threshold are deleted. The second criterion is a minimum number of network weights in the network. It ensures that the network size is large enough to represent

Algorithm 1 The algorithm of Optimal Brain Damage

```
1. Define a sufficiently large network structure.
2. Train the network until its converges.
3. For each network weight $w_i$, determine its corresponding $h_{ii} = \frac{\partial^2 E}{\partial w_i \partial w_i}$
4. For each $h_{ii}$, compute its saliency $\frac{1}{2} h_{ii} w_i^2$
5. Sort all saliency values and delete network weight(s) with the lower saliencies
6. Iterate to step 2~5 until the terminating criterion is met
```

the training data. Both of the threshold and the minimum network size maybe determined by cross validation method with a separate set of data.

2.3 Optimal Brain Surgeon

Optimal Brain Surgeon (OBS) was proposed by Hassabi et al. in 1993. It is an extension of OBD which focuses on pruning network weights in the neural network area. Both OBD and OBS share the same basic idea in that they use Taylor series to approximate the error function. However, OBS makes several improvements over the original OBD. The key differences of them are:

- the diagonal approximation on the calculation of Hessian matrix is dropped in OBS
- OBS imposes the deletion of network weight as a constraint in the objective function
- OBS does not only delete a network weight, it also updates all the remaining network weights optimally after each deletion

In the following section, the derivation of OBS algorithm will be given. This includes solving the constrained optimization problem by Lagrangian technique.

2.3.1 Constrained optimization

According to OBD, the change in error function δE is expanded in the form of a Taylor series as in Equation 2.3.

$$\delta E = (\frac{\partial E}{\partial \mathbf{w}})^T \cdot \delta \mathbf{w} + \frac{1}{2} \cdot \delta \mathbf{w}^T \cdot \frac{\partial^2 E}{\partial \mathbf{w^2}} \cdot \delta \mathbf{w} + \cdots$$

Assume that the network is trained to a (local) minimum and the error function is quadratic, the first term and the high order (order greater than 2) terms in equation 2.3 are neglected. Furthermore, the goal of OBS is to set a network weight to zero (deletion) in order to minimize the change (increase) in training error. Suppose we have a weight vector $\mathbf{w}$ and the q^{th} network weight w_q will be deleted. The deletion can be interpreted as the following equality constraint

$$\begin{aligned} \delta w_q + w_q &= 0 \\ & \mathit{or} \\ \mathbf{e_q^T} \cdot \delta \mathbf{w} + w_q &= 0 \end{aligned} \tag{2.7}$$

where vector $\mathbf{e_q}$is an unit vector with the q^{th} element being 1. Then, the objective of OBS becomes minimizing the error function subject to the deletion constraint

$$\min_q \left\{ \min_{\delta \mathbf{w}} \left(\frac{1}{2} \cdot \delta \mathbf{w}^T \cdot \frac{\partial^2 E}{\partial \mathbf{w^2}} \cdot \delta \mathbf{w} \right) \middle| \ \mathbf{e}_q^T \cdot \delta w + w_q = 0 \right\} \tag{2.8}$$

A typical method used to solve the constrained minimization problems is the Lagrangian technique. A Lagrange multiplier λ with the constraint in Equation 2.7 is inserted as a penalty term into the objective function $\delta E = \frac{1}{2} \delta \mathbf{w} \cdot \mathbf{H} \cdot \delta \mathbf{w}$. The Lagrangian form of the approximated change of the training error $\delta E_q(\delta \mathbf{w}, \lambda)$ for the minimization problem becomes

At the minimum, we have

$$\delta E_q(\delta \mathbf{w}, \lambda) = \frac{1}{2}\delta \mathbf{w}^T \cdot \mathbf{H} \cdot \delta \mathbf{w} + \lambda \cdot (\mathbf{e}_q^T \cdot \delta \mathbf{w} + w_q) \tag{2.9}$$

Both $\frac{\partial L}{\partial \delta \mathbf{w}} = 0$ and $\frac{\partial L}{\partial \lambda} = 0$, that is

$$\begin{aligned} \mathbf{H}\delta \mathbf{w} + \lambda \mathbf{e}_q^T &= 0 \qquad (2.10) \\ \mathbf{e}_q^T \delta \mathbf{w} + w_q &= 0 \qquad (2.11) \end{aligned}$$

Substitute Equation 2.11 into Equation 2.10, we have

$$\begin{aligned} \lambda \mathbf{e}_q^T \mathbf{H}^{-1} \mathbf{e}_q^T &= w_q \\ => \lambda &= \frac{w_q}{[\mathbf{H}^{-1}]_{qq}} \qquad (2.12) \end{aligned}$$

Substitute back λ of Equation 2.12 into Equation 2.10, we get

$$\delta \mathbf{w} = -\frac{w_q}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} \cdot \mathbf{e}_q \tag{2.13}$$

Finally, the deletion of w_q will increase the error by

$$\delta E_q = \frac{1}{2} \frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}} \tag{2.14}$$

Note that $\mathbf{H}$ is a full Hessian matrix and $[\mathbf{H}^{-1}]_{qq}$ is the element in the q^{th} column and q^{th} row of inverse Hessian. In Equation 2.14, the saliency for deleting network weight $w_q$is represented by δE_q. Besides that, Equation 2.13 embeds the changes in all network weights which include the deletion of weight w_q and the optimal update of all remaining weights in order to minimize the objective function. Every deletion is applied to the weight of the least saliency.

2.3.2 Algorithm of OBS

The algorithm of OBS, shown in algorithm 2, is similar to that of OBD (section 2.2.3).

Algorithm 2 The algorithm of Optimal Brain Surgeon

```
1. Define a sufficiently large network structure
2. Train the network until its converge stage is reached
3. Compute the full inverse Hessian $\mathbf{H}^{-1}$
4. Find the network weight $w_q$ that has the smallest saliency $\delta E_q = \frac{1}{2}\frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}}$. If
   the smallest saliency is smaller than a threshold, $w_q$ is deleted and proceed to
   Step 5; otherwise, OBS terminates.
5. Use $q$ from Step 4. and Equation 2.13 to update all the remaining network
   weights
6. Iterate to Step 2~5 until the deletion of a weight will cause large increase in
   the objective function $\delta E_q$
```

2.4 An Improved variant: Unit-OBS

Although OBS is claimed to generalize better than OBD [23, 24], there is still a disadvantage: a calculation of full inverse Hessian matrix is needed to prune one single weight in a large neural network. Since the inverse Hessian calculation costs the most in terms of computation power, pruning of a large neural network would require enormous computational cost. Therefore, unit-OBS [22], which is used to remove one network unit for each Hessian calculation, was proposed by Stahlberger et al. In general, unit-OBS can remove a group of network weights in one single Hessian calculation, so using unit-OBS can reduce the computation time drastically.

2.4.1 Introduction of selection matrix

The key point of deleting a whole network unit with the unit-OBS method is to used a selection matrix as a generic indication of the multiple deletion constraints. Recall that in equation 2.7, $\mathbf{e}_q^T \cdot \delta\mathbf{w} + w_q = 0$, the equation enforces a single deletion on weight q by a unit vector $\mathbf{e}_q$. However, if unit vector $\mathbf{e}_q$ is replaced by a selection matrix $\mathbf{M}$, where $\mathbf{M} = (\mathbf{e}_{q1}\mathbf{e}_{q2} \cdots \mathbf{e}_{qm})$, then OBS can be generalized and used to delete any m selected weights. The deletion positions are indexed by $q1, q2, \cdots, qm$ and the constraint is written as

$$(\mathbf{w} + \delta\mathbf{w})^T \mathbf{M} = 0 \quad with \quad \mathbf{M} = (\mathbf{e}_{q1}\mathbf{e}_{q2} \cdots \mathbf{e}_{qm})$$

The constrained optimization problem can be solved as usual using the Lagrangian method. The corresponding solutions become

$$\delta E_u = \frac{1}{2}\mathbf{w}^T\mathbf{M}\left(\mathbf{M}^T\mathbf{H}^{-1}\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{w} \quad (2.15)$$

$$\delta\mathbf{w} = -\mathbf{H}^{-1}\mathbf{M}\left(\mathbf{M}^T\mathbf{H}^{-1}\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{w} \quad (2.16)$$

where $\delta E_u$is the saliency of deleting network unit u and $\delta\mathbf{w}$ is the corresponding update on all network weights

2.4.2 Unit deletion

Equation 2.15 and Equation 2.16 constructs a generalized functional form of the OBS algorithm. To remove a network unit, all outgoing connections of this unit are labelled $q1, q2, \cdots, qm$ in the selection matrix $\mathbf{M}$. Then the corresponding saliency is calculated by using equation 2.15. With the same procedures as OBS, the smallest saliency term implies the least increase in the training error induced by the unit removal or the corresponding unit is least important unit to be deleted. Finally, other weights are rearranged by Equation 2.16. The above procedures composes one iteration of the unit-OBS method. The next iteration begins with calculation of the Hessian and then the calculation of saliencies, and so on.

2.5 Comparison between OBD, OBS, Unit-OBS

A comparison of the neural network pruning method is shown in Table 2.1. The differences between the algorithms, their strengths and weaknesses are analyzed.

In OBD, the foundation method of both OBS and Unit-OBS, the deletion of the least important network weight is given by a closed form solution of the minimization of the error function. The output of the OBD method contains only the saliency term. Furthermore, it is also the only method that makes an assumption of the diagonal Hessian matrix. Although the diagonal matrix saves a lot of computation when compared to a full Hessian matrix operation, this assumption is proved to be the cause of improper deletions [23].

In OBS and unit-OBS, a full Hessian matrix and inverse Hessian matrix is used. Both methods consider the deletions of the network weight or the network unit as constraints in the optimization problem. In the algorithm, the constraints are reformulated and imposed into cost function by Lagrangian technique. The solutions of the Lagrangian function are the saliency for each deletion and the new rearrangement of network weights.

Unit-OBS has a further improvement in reducing the computational cost. In a large neural network, the deletion of a single weight by OBS requires calculating a

	OBD	**OBS**	**unit-OBS**
Goal	- reduce the network size by pruning network weights	- reduce the network by pruning network weights - all remaining weights will be re-adjusted optimally	- reduce the network size by pruning network units - all remaining weights will be re-adjusted optimally
Methods	- error function is expanded by Taylor series	- error function is expanded by Taylor series - cast as a constrained optimization problem with an equality constraint - solved by Lagrangian method	- error function is expanded by Taylor series - cast as a constrained optimization problem with an equality constraint - solved by Lagrangian method
Assumptions and approximations	- assume a optimal network - assume diagonal Hessian - quadratic approximation on Taylor series	- assume a optimal network - assume full Hessian - quadratic approximation on Taylor series	- assume a optimal network - assume full Hessian - quadratic approximation on Taylor series
Major calculations	- calculate saliency for each weight	- calculate saliency for each weight and update all weights optimally	- calculate saliency for each unit and update all weights optimally

Table 2.1: Comparison of OBD, OBS and Unit-OBS

Hessian matrix once, which is expensive and dominate the cost of OBS. To further reduce the computational cost of OBS and Hessian, the deletions of multiple weights is formulated as the deletion of a single network unit in the unit-OBS method . For a network unit that has x connecting weights, the unit-OBS method would be able to reduce the number of calculations of the Hessian by x times, as a result, unit-OBS is also approximately x times faster than iterating OBS x times.

Chapter 3

OBS in hidden Markov model

With the background of OBD and OBS was introduced in chapter 2, the theory of OBS in HMM pruning will be presented in this chapter. The details of how Optimal Brain Surgeon (OBS) is adopted to pruning transitions and states in the HMM will be showed.

3.1 Introduction

The idea of adopting OBS in HMM pruning was inspired by a paper by Pedersen and Stork [25], which is about pruning in HMM by using the pruning method of the Boltzmann network. In practice, one can never be sure what topology should be defined for a particular task before its training process. Without an efficient method to change the topology dynamically, the HMM maybe predefined as over-sized. We believed that the OBS idea for pruning a neural network could be borrowed and applied to HMM pruning as a result of re-fining the over-sized and perhaps over-fitted HMM. It works by reducing the redundant model parameters so that it saves memory and computational costs.

In this chapter, we highlights the objectives and general idea of the OBS approach. Later, we focus on the mathematical formulation of the OBS theory. Since the architecture of the HMM is different from that of the neural network, several modifications and assumptions on the theory are necessary to adopt OBS to the HMM framework. We introduce the constrained optimization method as a solution for OBS which is subjected to both equality and inequality constraints. Finally, an overall algorithm of the OBS in the HMM framework will be discussed.

3.2 Transition pruning in hidden Markov model

Recognition with HMMs do not solely rely on their transition probabilities. The log-likelihood probability of a observation sequence is mainly contributed by the probability density function (pdf) in each state (as shown in 3.1). Those pdfs maybe modeled by some Gaussian mixtures in a continuous HMM or the probability associated with the centroids of a vector quantization codebook in a discrete HMM.

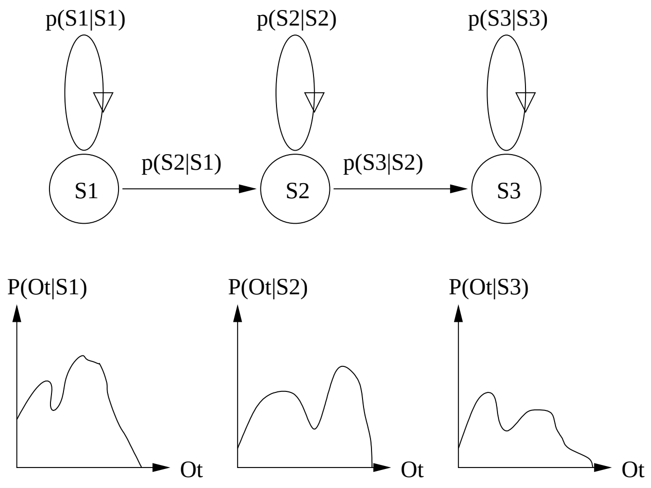

Figure 3.1: Example of a 3 states HMM with continuous probability density function in each state

3.2.1 Reasons of pruning transitions

Although transition probabilities are not the most important contribution when calculating the log-likelihood, there are strong reasons for choosing transition as the pruning element of HMM

- transitions act as a topology constraint of the HMM. Any change in transitions (addition or deletion) directly induce topology change in the HMM and the change in topology affects the search path of state sequence when calculating likelihood. In some cases, removal of transitions may lead to removal of a whole state in an HMM.
- transition probabilities have properties similar to those of weights in neural network. They are both scalar terms that join the network units (states) and construct the architecture of model. So that transition pruning in HMM framework is similar to the theory of OBS in a neural network and it can be adopted with only a few modifications.

Because of the reasons mentioned above, the focus in our research was to develop a method capable to delete the least important transition probabilities in an HMM so that the decrease of the total log-likelihood in the training data is minimal. It was expected that the pruned HMM will lead to a modification on transitions and, more importantly, on HMM topology.

3.2.2 Specific issues

Transition probabilities in the HMM controls the search of state sequence path during the training and decoding process. They are scalar quantities which indicate how likely the next state will be given that the accumulated probabilities in the previous time are identical. In the neural network, the network weights can be any real numbers and do not have any constraints or magnitude bounds. However, transitions in the HMM are constrained by the probability axioms. The first one is that probability values should be between 0 and 1. The second is that the sum of all out-going transitions from a state should be 1, since they are mutually exclusive events. Both the constraints cause the differences in pruning weights in the neural network. Table 3.1 summarizes all differences between transitions and network weights

	NN weights	**HMM transitions**
physical meaning	scalar as a multiplication factor	probability of state transition
constraints	none	should be between 0 and 1, all out-going transitions of a state should sum to 1

Table 3.1: Comparison of transitions in the hidden Markov model (HMM) and weights in the neural network (NN)

Another important pre-caution in pruning HMM is to maintain a feasible deletion of transition and ensure that the topology is reasonable. It is impossible that there is no reachable path going through from the starting state to the ending state in the pruned HMM. If this is the case, the HMM is broken by the pruning method. In OBS method, every choice of transition deletion should be verified and ensure that the HMM will not be broken by the OBS deletion.

3.3 Constrained optimization

According to the theory of OBS mentioned in Section 2.3 under the former work of neural work, the central idea of weight pruning is to delete the least important weight(s) so that the change in the cost function is minimized. In the HMM framework, pruning of a network weight in the neural network framework has changed to pruning of a transition and the cost function has changed from the training error to the likelihood of the training data. Since maximum likelihood is the criterion in our HMM training, our cost function is to maximize the likelihood of training data instead of minimizing th training error in the case of neural network. To summarize, OBS in the HMM framework implies the deletion of the least important transition(s); as a result, the decrease in the likelihood of training data is minimized.

Let us denote the change in log-likelihood as δL; the Hessian matrix and transition vector are represented by $\mathbf{H}$ and $\delta\mathbf{w}$ respectively. Thus, the cost function - the change in likelihood - is approximated by using the Taylor expansion as mentioned in equation 2.6 as the following

$$\delta L = \frac{1}{2}\delta\mathbf{w}^T \cdot \mathbf{H} \cdot \delta\mathbf{w} \tag{3.1}$$

It is noticeable that the Taylor expansion of the likelihood on the training data is identical to the cost function with training error. In fact, there are differences between training error and training likelihood in regard to assumptions and approximations. The approximations and assumptions will be analyzed in the following sections.

3.3.1 Approximations

Before deriving the formulas of the OBS in the HMM framework, the assumptions in the original OBS in NN should be modified. In section 2.3, the valid assumptions contains the quadratic approximation and the extremal approximation. The quadratic approximation refers to the fact that the change in cost function is assumed in a second order functional form so that the third and higher order terms are neglected in the equation 3.1. The extremal approximation assumes that the target model is converged to a (local) maximum point in the training process. Therefore, the gradient of the cost function is approximately zero and can be neglected.

It should be emphasized that full Hessian matrix is used in this approach. As the paper by Stork et al.[24] pointed out that the diagonal approximation in the Hessian leads to the wrong deletion in network weight. It is necessary to compute a Hessian with higher computation cost but this is more accurate. Another reason is because HMM, unlike NN, after deleting a transition of a state, the other out-going transition of this state should be re-adjusted so that they sum up to 1. Thus, the change in one transition should also update other related transitions.

3.3.2 Constraints

In the OBS method, a deletion in network weight is considered as a constraint in the minimization of the cost function. In other words, instead of minimizing the change in training error directly by using the differentiation method, the main idea of solving this problem becomes minimizing the change in training error by deleting a network weight and subject to a deletion constraint at the same time.

A well known approach for tackling constrained optimization is the Lagrangian method. It imposes constraint(s) as a penalty function(s) into the cost function so that the optimization technique such as differentiation method does not require extra modification. The OBS is one of the examples that makes use of Lagrangian as a solution for constrained minimization. A closed form solution of saliency and re-calculation of other weights is formulated by using this method.

In the HMM framework, transition probabilities have additional constraints because of their definitions. Therefore, it is obvious that additional constraints should be inserted into the Lagrangian. The three constraints of deleting a transition in an

HMM are:

1. delete a specific HMM transition within all transitions

2. the sum of all out-going transitions from each HMM state should equal to 1

3. any transitions should be between 0 and 1.

Before the formulation of these constraints, it should be noticed that the first two constraints are equalities and the third one is an inequality. With the introduction of the selection matrix in the unit-OBS method [22], all transitions are re-arranged in vector $\mathbf{w}$. Assume that $\mathbf{e}_q$ is an unit vector with the q^{th}element equal to 1. The equality constraint 1 which represents a deletion of the q^{th}element in transition vector $\mathbf{w}$ can be written as

$$\begin{aligned} \delta\mathbf{w}_q &= -\mathbf{w}_q \\ & \mathit{or} \\ \mathbf{e}_q^T \cdot (\delta\mathbf{w}_q + \mathbf{w}_q) &= 0 \end{aligned} \tag{3.2}$$

The equality constraint 2, which enforces the sum of all out-going transitions in each state equals one, can be written as

$$\begin{aligned} \mathbf{M}^T (\mathbf{w} + \delta\mathbf{w}) &= \vec{1} \\ & \mathit{or} \\ \mathbf{M}^T \delta\mathbf{w} &= \vec{0} \end{aligned} \tag{3.3}$$

where $\mathbf{M}$ is an $n \times m$ indicator matrix for an HMM with totally n transitions and m states. The elements of $\mathbf{M}$ are either 0 or 1. The 1's in its k^{th} column vector indicates positions of the out-going transitions in the k^{th} state. For example, if each HMM state has 2 out-going transitions. The matrix $\mathbf{M}$ will look like

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & & \\ 1 & 0 & 0 & & \\ 0 & 1 & 0 & \cdots & \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \\ 0 & 0 & 1 & & \\ & & \vdots & & \end{bmatrix}_{n\times m} \tag{3.4}$$

This indicator matrix plays an important role in formulation; it collects all out-going transitions in each state correspondingly so the "sum-to-one" constraint can be formulated in a simple matrix form (like equation 3.8).

The constraint 3 is an inequality about restricting non-negative values on transition probabilities. It is noticed that the upper bound of the transition probabilities can be eliminated because constraints 2 and 3 together implies the upper bound (less than one). So, the inequality constraint will be

$$\begin{aligned} \mathbf{w} + \delta\mathbf{w} &\geqslant \vec{0} \\ &\mathit{or} \\ \delta\mathbf{w} &\geqslant -\mathbf{w} \end{aligned} \tag{3.5}$$

3.3.3 Active-set method

With all constraints and cost function defined, the overall constrained optimization problem subjected to equality and inequality constraints is re-written as

maximize

$$\delta L = \frac{1}{2}\delta \mathbf{w}^T \cdot \mathbf{H} \cdot \delta \mathbf{w} \tag{3.6}$$

such that

$$\begin{aligned} \mathbf{e}_q^T \cdot (\delta \mathbf{w}_q + \mathbf{w}_q) &= 0 \\ \mathbf{M}^T \delta \mathbf{w} &= \vec{0} \\ \mathbf{w} + \delta \mathbf{w} &\geqslant \vec{0} \end{aligned}$$

This is a quadratic programming problem subjected to both equality and inequality constraints. In this thesis, we solve it by the active-set method. The active-set method [26, 27, 28] is considered to be an approach with the usage of Lagrangian, but it divides the problem into many sub-problems with equality constraint(s) by maintaining a set of active equality constraints. Suppose there are many inequality constraints which bounce the solution within a certain region. The optimal solution for the optimization problem, either minimization or maximization, must lie on the extreme point or constraint boundaries. To give a detailed explanation of the active-set method, the concept is illustrated in Figure 3.2, and the method is presented with the following example.

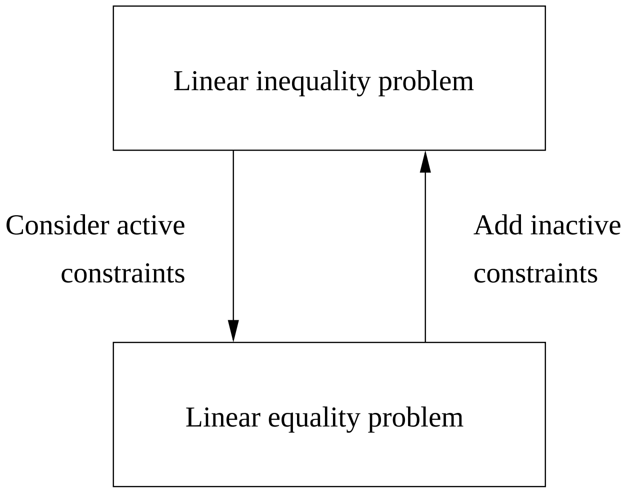

Figure 3.2: Active set method with linear inequality constraints

3.3.3.1 An Example in constrained optimization

Consider the problem of minimization, that is

minimize

$$f(x) = \frac{1}{2} \cdot x_1^2 + \frac{1}{2} \cdot x_2^2$$

Lagrangian subjected to constraints $x_1 + 2x_2 \geq 2$, $x_1 - x_2 \geq -1$ and $-x_1 \geq -3$

The feasible region of the optimal solution is shown in the shaded area in Figure 3.3. The abscissa and ordinate are $x_1$and x_2 respectively. The minimum point, notated as x_0, is the reference solution for this problem. It can be observed that the solution must lie on one of the equality constraints $x_1 + 2x_2 = 2$, $x_1 - x_2 = -1$ or $x_1 = 3$.

Therefore, the first step in solving this problem is to make an initial guess about the active equality constraint. Suppose the solution x_0 is on the line $x_1 + 2x_2 = 2$ (this implies only constraint $x_1 + 2x_2 \geq 2$ is active). This leads to a minimization

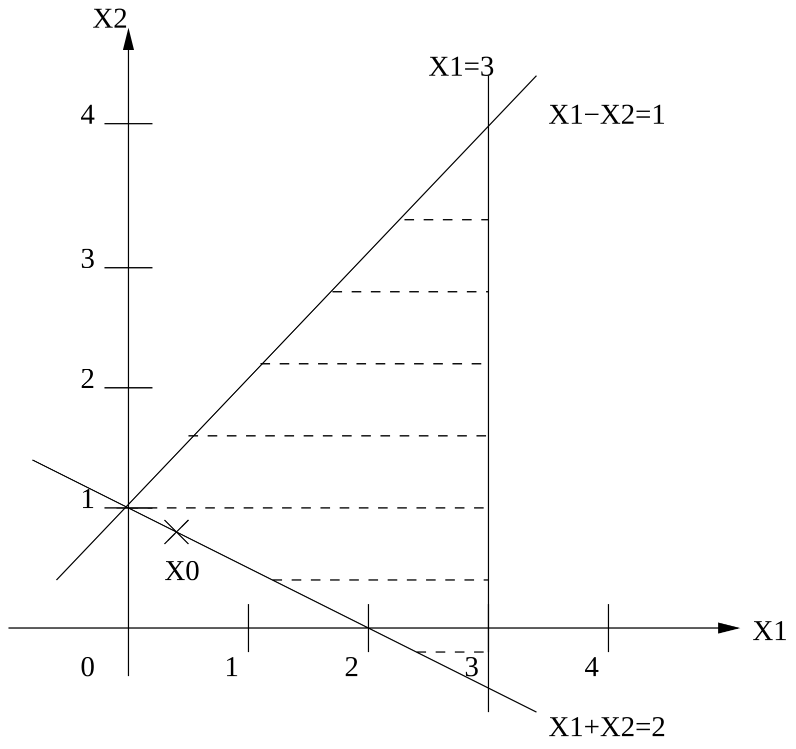

Figure 3.3: Plot of feasible solution and its feasible area

problem with equality constraints.

minimize

$$\begin{aligned} f(x) &= \frac{1}{2} \cdot x_1^2 + \frac{1}{2} \cdot x_2^2 \\ & s.t. \\ x_1 + 2x_2 &= 2 \end{aligned}$$

Using the Lagrangian method, we solve equations

$$f'(x) - \begin{pmatrix} 1 \\ 2 \end{pmatrix} \lambda = 0 \quad and \quad x_1 + 2x_2 = 2$$

and the solution is $x = (\frac{2}{5}, \frac{4}{5})^T$, $\lambda = \frac{2}{5}$.

Instead of solving this set of equations for a feasible solution, the active-set method continues other possible sets of active constraints and solve other feasible solutions. After all possible constrained optimization problems are solved, the solution that gives minimum $f(x)$ will be the global constrained minimization solution.

3.3.3.2 Kuhn Tucker conditions

According to the necessary conditions for optimality, solution x_0 is a local minimizer over $f(x)$ $s.t.$ $Ax \geq b$ if

- corresponding Lagrange multiplier $\lambda > 0$
- Hessian matrix $\mathbf{H}$ in the cost function is positive semi-definite
- $f'(x) = A^T \lambda$
- $\lambda^T(Ax - b) = 0$

Since the solution matches all the necessary conditions, this is the local minimized solution for this sub-problem. However, the initial guess in regard to the active constraint(s) was not sufficient to find the optimal solution for the overall optimization problem. In this case, the major difficulty in solving optimization problem with inequality constraints was discovered; that is, it is necessary to check a large number of possible combinations of active constraints in order to determine the optimal solution. Undoubtly, there will be an exhaustive search into the active constraints and the computation cost will be expensive.

To tackle this problem, the active-set method does not only use Lagrangian in solving each equality constrained optimization sub-problem, their Lagrange multipliers also measure the following quantities

- important information in sensitivity analysis
- in case of problems with inequality constraints, estimation of the multipliers indicates how to improve an estimation of the optimal solution

To conclude, the active-set method uses the Lagrange multipliers and its corresponding function values as the searching criterion of the active constraints so that the search times (iteration) can be reduced.

3.4 Algorithm of OBS in HMM

The overall method of OBS (as shown in Figure 3.4) is an iterative process which deletes a single transition in each iteration. The OBS is terminated by one of the following two criteria

- the largest saliency (δL) which indicates the least decrease in the log-likelihood should be greater than a threshold.
- a limit on the number of deletions

The first criterion is important for preventing a dramatic drop in likelihood as well as model performance. The second criterion guarantees a minimum model parameter size.

3.5 State deletion by Unit-OBS

Besides the adoption of OBS from weight deletion in neural network to transition deletion in HMM, the unit-OBS for unit deletion in network can also be adopted

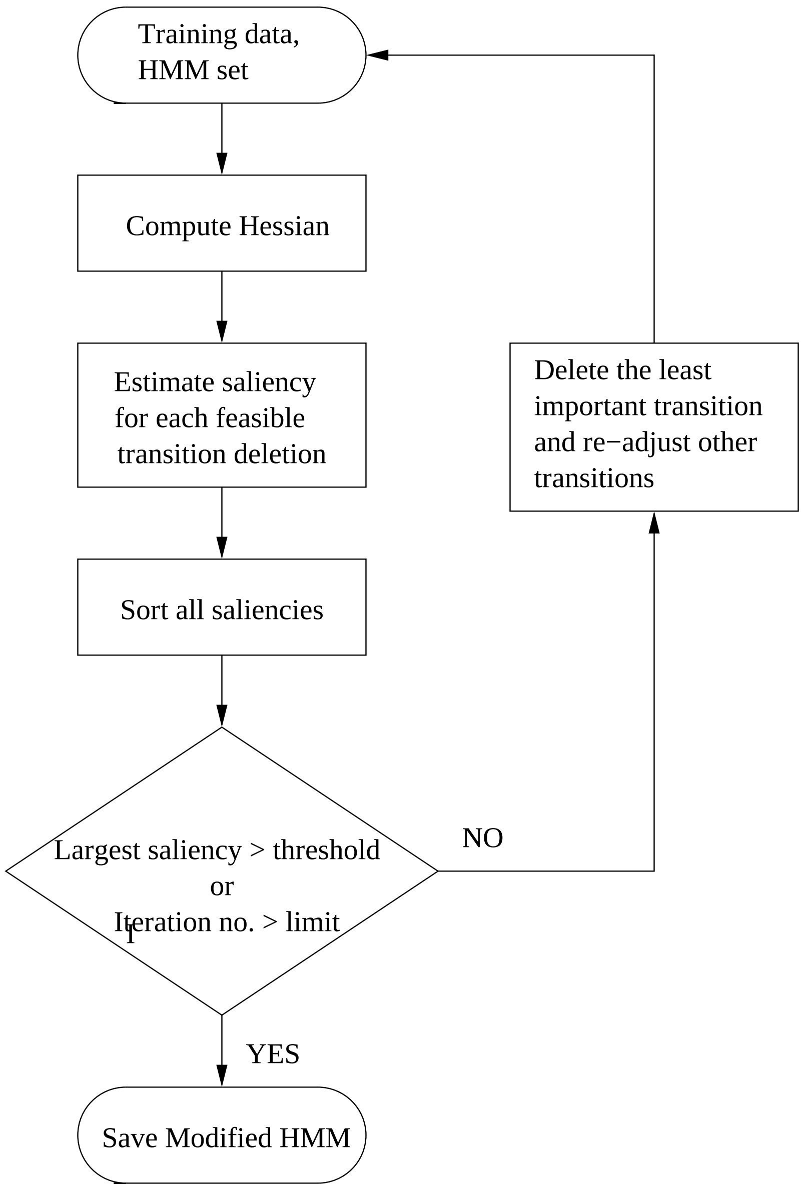
Training data,
HMM set
Compute Hessian
Estimate saliency for each feasible transition deletion
Sort all saliencies
Largest saliency > threshold
or
Iteration no. > limit
NO
Delete the least important transition and re−adjust other transitions
YES
Save Modified HMM

Figure 3.4: Flow chart of OBS algorithm in HMM pruning

to state deletion of HMM. With the capability of multiple deletions, unit-OBS is able to remove the least important state in the HMM in a single iteration. So that, the decrease in cost function, which is the log-likelihood of the training data, is minimized. Practically, removing a state S implies the removal of all incoming and outgoing transitions of state S. The example is illustrated in Figure 3.5.

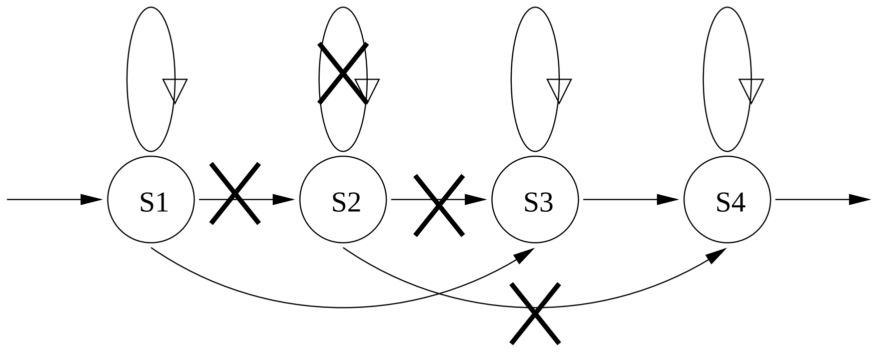

Figure 3.5: Illustration of deleting state S2 in HMM

For any state in an HMM, if no transition is connected to it, this state will not be considered in the decoding and training procedures. By specifying appropriate constraints, the general theory maybe applied directly to do unit-OBS in HMMs.

3.5.1 Modifications of constraints

The first modification is made to the deletion constraint. In multiple deletions, D is defined as the set of deletion indexes $\{q : q \in D\}$. D indicates the set of transitions to be removed. The elements in D are the incoming and outgoing transitions of the state S that is to be removed. Thus, S has $|D|$ connections and Equation 3.7 represents $|D|$ equations to delete those in-coming and out-going transitions.

$$\begin{aligned} \delta \mathbf{w}_q &= -\mathbf{w}_q \textit{ where } q \in D \\ &\textit{or} \\ \mathbf{e}_q^T \cdot (\delta \mathbf{w}_q + \mathbf{w}_q) &= 0 \end{aligned} \tag{3.7}$$

The second modification is made to the equality constraint (equation 3.8), which ensures the sum of all out-going transitions in each state is equal to one, can be written as

$$\begin{aligned} \mathbf{M}^T (\mathbf{w} + \delta \mathbf{w}) &= \mathbf{1} \\ &\textit{or} \\ \mathbf{M}^T \delta \mathbf{w} &= \mathbf{0} \end{aligned} \tag{3.8}$$

where matrix $\mathbf{M}$, the indicator matrix, is composed of column vectors of either 0 or 1. The 1's in its k^{th} column vector indicates the positions of the out-going transitions in the k^{th} state. Except the column which indicates the state to be deleted, the column vectors of $\mathbf{M}$ are the same as the indicator matrix in the OBS method. This matrix is used in formulation of unit-OBS for deleting a whole state. For example, if there is an HMM with m states , n transitions and each state has two outgoing transitions, then the indicator matrix $\mathbf{M}$ of deleting the third state (the third column vector is removed) becomes

$$\mathrm{M} = \begin{bmatrix} 1 & 0 & 0 & & & \\ 1 & 0 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 0 & 0 & . & . & . \\ 0 & 0 & 0 & & & \\ 0 & 0 & 1 & & & \\ 0 & 0 & 1 & & & \\ & & \vdots & & & \end{bmatrix}_{n\times(m-1)} \tag{3.9}$$

Finally, the inequality constraint on the boundaries of the transition probabilities is kept unchanged. The upper bound of transition probabilities can be eliminated because constraint 3.8 and 3.10 together implies the upper bound (less than one). So, the inequality constraint contains the lower bound of state transitions and can be notated as

$$\begin{aligned} \mathbf{w} + \delta\mathbf{w} &\geqslant \mathbf{0} \\ &\mathit{or} \\ \delta\mathbf{w} &\geqslant -\mathbf{w} \end{aligned} \tag{3.10}$$

With all constraints and cost function defined, the overall central optimization problem in OBS is rewritten as a maximization of

$$\delta logL = \frac{1}{2}\delta\mathbf{w}^T \cdot \mathbf{H} \cdot \delta\mathbf{w} \tag{3.11}$$

subject to

$$\begin{aligned} \mathbf{e}_q^T \cdot (\delta\mathbf{w}_q + \mathbf{w}_q) &= 0 \; where \; q \in \; deletion\, set\, D \\ \mathbf{M}^T\delta\mathbf{w} &= \mathbf{0} \\ \mathbf{w} + \delta\mathbf{w} &\geqslant \mathbf{0} \end{aligned}$$

This constrained optimization problem is a *quadratic programming problem.* The formulation of this problem can be solved by using many standard numerical methods such as the active-set method [26].

3.5.2 Implementation

The algorithm of the unit-OBS is similar to the algorithm of the OBS except that the deletion element is changed from a transition to a state in HMM. The flow of algorithm is illustrated in algorithm of unit-OBS (Algorithm 3).

One should note that the deletion of a state by unit-OBS cannot totally replace the deletion of a transition by OBS in the task of refining the topology of HMMs. The unit-OBS method performs state deletion by restricting those deletions around one particular state. However, it is unable to fine-tune the topology of the HMM with a single transition deletion. When comparing the unit-OBS and the original

Algorithm 3 The algorithm of unit-OBS in HMM

```
1. Define a sufficiently large HMM and train the HMM until it converges
2. Compute the full Hessian $\mathbf{H}$ of this HMM with training data
3. Construct and solve the constrained optimization problem as equation 3.11.
4. Find the $q^{th}$ state that has the largest saliency $\delta logL$. If the saliency is greater
   than a threshold $T$, then connections of the $q^{th}$ state are deleted and proceed
   to step 5; otherwise, OBS terminates.
5. Use $q$ from Step 4. and update remaining transitions by the corresponding
   constrained optimization result $\delta\mathbf{w}$ in Step 3
6. Iterate step 2~5 until the maximum number of iteration exceeded
```

OBS method, unit-OBS has the benefit of reduced computation cost, whereas the OBS method can fine-tune the topology of an HMM in more detail.

Chapter 4

Calculation of Hessian

The Hessian matrix plays an important role in our quadratic programming problem. It provides the second order derivative information for the optimization method such as the Active-set method used in this thesis. In this chapter, the detailed formulation and implementation on Hessian matrix will be discussed, which includes:

1. the formulation of gradient and Hessian

2. their corresponding initial and ending conditions

3. other implementation issues

The Hessian matrix is derived from the first principle. The cost function for the derivative calculation is the total log-likelihood of the training data which is calculated by the well known Forward algorithm.

4.1 Cost function

The Hessian matrix $\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{w}^2}$ is the second derivatives of the cost function E with respect to the change of the transition vector $\delta\mathbf{w}$. In this thesis, the total log-likelihood of the training data is chosen as the cost function. There are both theoretical and practical benefits for choosing the total log likelihood as the cost function. The reasons are

- Likelihood has already been the optimization (minimization/maximization) criterion for many classification problems such as HMM training, speaker adaptation, etc.
- Likelihood in log domain is efficient for computer programming of numerical algorithm. It solves the under-flow and over-flow problems in numerical calculations that require high precision. Multiplication of floating-point values can be also re-written as summation in log-domain.
- Log function is monotonic. The optimizations algorithms derived in likelihood can also be adopted to log likelihood domain.

4.2 Notations

For the reference purpose in the following sections, the notations which appear in formulas are defined in the following list

$O^{(u)}$	observation sequence of utterance u of duration $T^{(u)}$
O_t	observation vector in one utterance at time t , where $t = 1, \ldots, T$
λ	model parameters
a_{ij}	transition probability from state i to state j

$b_j(O_t)$	observation probability in state j at time t
$\alpha_j(t)$	forward probability of Baum Welch algorithm in state j at time t
N	number of states in the HMM
U	total number of utterances in training set

4.3 Multiple observations

With multiple observations or multiple utterances in the training data set, the total likelihood will be $P(O \mid \lambda) = \prod_u P(O^{(u)} \mid \lambda)$. When it is transformed into log domain, the total log likelihood becomes $logP(O \mid \lambda) = \sum_u logP(O^{(u)} \mid \lambda)$. It is noticed that the summation enable us to separate the calculation of the Hessian for each utterance individually. Therefore, the gradient terms $\frac{\partial logP(O|\lambda)}{\partial a_{ij}}$ and the Hessian $\frac{\partial^2 logP(O|\lambda)}{\partial a_{ij}\partial a_{mn}}$ terms for the whole training set can be written as

$$\begin{aligned}
\frac{\partial logP(O \mid \lambda)}{\partial a_{ij}} &= \sum_u \frac{\partial logP(O^{(u)} \mid \lambda)}{\partial a_{ij}} \\
&= \sum_u \left(\frac{\partial logP(O^{(u)} \mid \lambda)}{\partial P(O^{(u)} \mid \lambda)} \times \frac{\partial P(O^{(u)} \mid \lambda)}{\partial a_{ij}} \right) \\
&= \sum_u \left(\frac{1}{P(O^{(u)} \mid \lambda)} \times \frac{\partial P(O^{(u)} \mid \lambda)}{\partial a_{im}} \right) \qquad (4.1)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 logP(O \mid \lambda)}{\partial a_{ij}\partial a_{mn}} &= \sum_u \frac{\partial \left(\frac{\partial logP(O^{(u)}|\lambda)}{\partial P(O^{(u)}|\lambda)} \times \frac{\partial P(O^{(u)}|\lambda)}{\partial a_{ij}} \right)}{\partial a_{mn}} \\
&= \sum_u \left[\frac{1}{P(O^{(u)} \mid \lambda)} \times \frac{\partial^2 P(O^{(u)} \mid \lambda)}{\partial a_{ij}\partial a_{mn}} \right. \\
&\quad \left. - \frac{1}{P(O^{(u)} \mid \lambda)^2} \times \frac{\partial P(O^{(u)} \mid \lambda)}{\partial a_{ij}} \times \frac{\partial P(O^{(u)} \mid \lambda)}{\partial a_{mn}} \right] \qquad (4.2)
\end{aligned}$$

The transitions in the HMM are notated as a_{ij} and a_{mn}. In the formulation, the indexes i, j, m, n can be any real integer smaller than N. From the above equations, the Hessian matrix, with the log likelihood as cost function, can be decomposed into summation of gradient and Hessian terms of each utterance in linear domain. To simplify the representation, the notation O_t indicates the observation vector at time t in one particular utterance u in the following formulations.

4.4 General form

As mentioned in the beginning of this chapter, the log-likelihoods of the utterances are calculated with the forward-pass in the well-known Baum-Welch algorithm. This forward-pass algorithm is an iterative process where the forward probability term at previous time t depends on forward probability at time $(t-1)$. The details of the forward probability formula in the Baum-Welch algorithm is covered by [29, 30]. The following equations summarize the calculation of the forward probabilities:

- The initial condition : $\alpha_s(1) = \pi_s b_s(O_1)$
- The iterative form : $\alpha_s(t) = \sum_{r=1}^{N}[\alpha_r(t-1)a_{rs}]b_s(O_t)$ where $s = 1, \ldots, N$ and $t = 2, \ldots, T$
- The cost function (Likelihood) : $logP(O \mid \lambda) = \alpha_N(T) = \sum_{r=2}^{N-1} \alpha_r(T)a_{rN}$ where $s = 1, \ldots, N$

The formulation of both gradient terms and Hessian terms are separated as real state and null state because there are differences between their handling methods. In each real state, which is also called emitting state, the observation probabilities are required in calculation. Whereas, the calculation in null state does not require any observation probability.

In this thesis, the HMM topology always contains begin and end null node. This is illustrated by Figure 4.1 When there are N states in an HMM, state 1 will be the

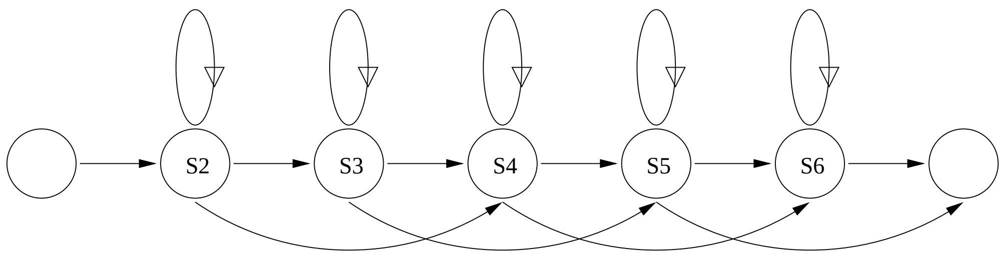

Figure 4.1: begin null state and end null state in HMM

begin null state and state N will be the end null node. Therefore, in the real state calculations, the previous terms are ranged from 2 to $N-1$.

4.4.1 for real state (node)

$$\alpha_s(t) = \sum_{r=2}^{N-1} [\alpha_r(t-1) a_{rs}] b_s(O_t) \tag{4.3}$$

$$\frac{\partial \alpha_s(t)}{\partial a_{ij}} = \sum_{r=2}^{N-1} \left[\frac{\partial \alpha_r(t-1)}{\partial a_{ij}} a_{rs} + \frac{\partial a_{rs}}{\partial a_{ij}} \alpha_r(t-1) \right] b_s(O_t) \tag{4.4}$$

$$\begin{aligned} \frac{\partial^2 \alpha_s(t)}{\partial a_{ij} \partial a_{mn}} &= \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(t-1)}{\partial a_{ij} \partial a_{mn}} a_{rs} + \frac{\partial \alpha_r(t-1)}{\partial a_{ij}} \frac{\partial a_{rs}}{\partial a_{mn}} \right. \\ &\quad \left. + \frac{\partial \alpha_r(t-1)}{\partial a_{mn}} \frac{\partial a_{rs}}{\partial a_{ij}} + \frac{\partial^2 a_{rs}}{\partial a_{ij} \partial a_{mn}} \alpha_r(t-1) \right] b_s(O_t) \\ &= \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(t-1)}{\partial a_{ij} \partial a_{mn}} a_{rs} + \frac{\partial \alpha_r(t-1)}{\partial a_{ij}} \frac{\partial a_{rs}}{\partial a_{mn}} \right. \\ &\quad \left. + \frac{\partial \alpha_r(t-1)}{\partial a_{mn}} \frac{\partial a_{rs}}{\partial a_{ij}} \right] b_s(O_t) \end{aligned} \tag{4.5}$$

4.4.2 for null state (node)

$$\alpha_s(t) = \sum_{r=2}^{N-1} [\alpha_r(t) a_{rs}] \tag{4.6}$$

$$\frac{\partial \alpha_s(t)}{\partial a_{ij}} = \sum_{r=2}^{N-1} \left[\frac{\partial \alpha_r(t)}{\partial a_{ij}} a_{rs} + \frac{\partial a_{rs}}{\partial a_{ij}} \alpha_r(t) \right] \tag{4.7}$$

$$\begin{aligned} \frac{\partial^2 \alpha_s(t)}{\partial a_{ij} \partial a_{mn}} &= \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(t)}{\partial a_{ij} \partial a_{mn}} a_{rs} + \frac{\partial \alpha_r(t)}{\partial a_{ij}} \frac{\partial a_{rs}}{\partial a_{mn}} + \frac{\partial \alpha_r(t)}{\partial a_{mn}} \frac{\partial a_{rs}}{\partial a_{ij}} + \frac{\partial^2 a_{rs}}{\partial a_{ij} \partial a_{mn}} \alpha_r(t) \right] \\ &= \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(t)}{\partial a_{ij} \partial a_{mn}} a_{rs} + \frac{\partial \alpha_r(t)}{\partial a_{ij}} \frac{\partial a_{rs}}{\partial a_{mn}} + \frac{\partial \alpha_r(t)}{\partial a_{mn}} \frac{\partial a_{rs}}{\partial a_{ij}} \right] \end{aligned} \tag{4.8}$$

In the differentiation of transitions with respect to transitions, such as $\frac{\partial a_{rs}}{\partial a_{ij}}$, these terms will equal 1 when $r = i$ and $s = j$. Otherwise, it will equal 0.

4.5 Initial and ending conditions

In forward pass probability, the initial and final terms are handled with the special equations 4.9 and 4.10 respectively. The Section 4.5.1 and Section 4.5.2 illustrates their corresponding derivatives.

$$\alpha_s(1) = \pi_s b_s(O_1) \tag{4.9}$$

$$\alpha_s(2) = \sum_{r=2}^{N-1} [\alpha_r(1) a_{rs}] b_s(O_2)$$

$$\alpha_s(3) = \sum_{r=2}^{N-1} [\alpha_r(2) a_{rs}] b_s(O_3)$$

$$\vdots$$

$$\alpha_N(T) = \sum_{r=2}^{N-1} \alpha_r(T) a_{rN} \tag{4.10}$$

4.5.1 1st derivatives

$$\begin{aligned}
\frac{\partial \alpha_s(1)}{\partial a_{ij}} &= \frac{\partial \pi_s b_s(O_1)}{\partial a_{ij}} \\
&= 0
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \alpha_s(2)}{\partial a_{ij}} &= \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial \alpha_r(1)}{\partial a_{ij}} a_{rs} \right] + \frac{\partial a_{is}}{\partial a_{ij}} \alpha_i(1) \right\} b_s(O_2) \\
&= \frac{\partial a_{is}}{\partial a_{ij}} \alpha_i(1) b_s(O_2) \\
&= \frac{\partial a_{is}}{\partial a_{ij}} \pi_i b_i(O_1) b_s(O_2)
\end{aligned}$$

$$\frac{\partial \alpha_s(3)}{\partial a_{ij}} = \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial \alpha_r(2)}{\partial a_{ij}} a_{rs} \right] + \frac{\partial a_{is}}{\partial a_{ij}} \alpha_i(2) \right\} b_s(O_3)$$

$$\vdots$$

$$\frac{\partial \alpha_s(T)}{\partial a_{ij}} = \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial \alpha_r(T-1)}{\partial a_{ij}} a_{rs} \right] + \frac{\partial a_{is}}{\partial a_{ij}} \alpha_i(T-1) \right\} b_s(O_T)$$

$$\frac{\partial \alpha_N(T)}{\partial a_{ij}} = \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial \alpha_r(T)}{\partial a_{ij}} a_{rN} \right] + \frac{\partial a_{iN}}{\partial a_{ij}} \alpha_i(T) \right\} \tag{4.11}$$

4.5.2 2nd derivatives

$$\frac{\partial^2 \alpha_s(1)}{\partial a_{ij} \partial a_{mn}} = 0$$

$$\begin{aligned} \frac{\partial^2 \alpha_s(2)}{\partial a_{ij} \partial a_{mn}} &= \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(1)}{\partial a_{ij} \partial a_{mn}} a_{rs} \right] + \frac{\partial \alpha_i(1)}{\partial a_{ij}} \frac{\partial a_{is}}{\partial a_{mn}} + \frac{\partial \alpha_i(1)}{\partial a_{mn}} \frac{\partial a_{is}}{\partial a_{ij}} \right\} b_s(O_2) \\ &= 0 \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 \alpha_s(3)}{\partial a_{ij} \partial a_{mn}} &= \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(2)}{\partial a_{ij} \partial a_{mn}} a_{rs} \right] + \frac{\partial \alpha_i(2)}{\partial a_{ij}} \frac{\partial a_{is}}{\partial a_{mn}} + \frac{\partial \alpha_i(2)}{\partial a_{mn}} \frac{\partial a_{is}}{\partial a_{ij}} \right\} b_s(O_3) \\ &= \left\{ \frac{\partial \alpha_i(2)}{\partial a_{ij}} \frac{\partial a_{is}}{\partial a_{mn}} + \frac{\partial \alpha_i(2)}{\partial a_{mn}} \frac{\partial a_{is}}{\partial a_{ij}} \right\} b_s(O_3) \end{aligned}$$

$$\vdots$$

$$\frac{\partial^2 \alpha_s(T)}{\partial a_{ij} \partial a_{mn}} = \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(T-1)}{\partial a_{ij} \partial a_{mn}} a_{rs} \right] + \frac{\partial \alpha_i(T-1)}{\partial a_{ij}} \frac{\partial a_{is}}{\partial a_{mn}} + \frac{\partial \alpha_i(T-1)}{\partial a_{mn}} \frac{\partial a_{is}}{\partial a_{ij}} \right\} b_s(O_{T-1})$$

$$\frac{\partial^2 \alpha_N(T)}{\partial a_{ij}\partial a_{mn}} = \left\{ \sum_{r=2}^{N-1} \left[\frac{\partial^2 \alpha_r(T)}{\partial a_{ij}\partial a_{mn}} a_{rN} \right] + \frac{\partial \alpha_i(T)}{\partial a_{ij}} \frac{\partial a_{iN}}{\partial a_{mn}} + \frac{\partial \alpha_i(T)}{\partial a_{mn}} \frac{\partial a_{iN}}{\partial a_{ij}} \right\} \tag{4.12}$$

4.6 Implementation issues

Since calculation of the Hessian matrix is computationally expensive and it costs most of the computation time in the whole OBS method, the implementation method which reduces the computation cost becomes critical. One of the simplest way that can save computational cost is to make use of the symmetric property of the Hessian matrix. The computation cost can be reduced by half by only calculating the upper tri-angular Hessian elements and copying them to their corresponding lower triangular elements.

Besides that, as the gradient and the Hessian matrix at time t depend on those at time $(t-1)$, it is perfect to borrow the idea of dynamic programming and implement the calculation across the time domain. The procedure stores intermediate terms at time $t-1$ and time t. Both simple but useful methods should reduce computation and memory storage considerably.

Chapter 5

Experimental evaluations

In this chapter, we will present some experiments to evaluate our OBS method and unit-OBS method in reducing HMM complexity. Although the theory of our OBS should be general enough that it can be applied to all HMM classification tasks, we will only focus on the automatic speech recognition (ASR) applications in this thesis.

This chapter is divided into 2 parts: the preliminary experiments and the application experiments. In the first part, several preliminary experiments were designed with a focus on proving focus on proof of the OBS theory and investigating some minor issues in the OBS algorithm such as the effect of Viterbi segments in the Hessian calculation and the deletion choice made by OBS. In the second part, OBS was used to reduced the complexity of a 16-state HMM, and unit-OBS was used to reduce the complexity of a composite HMM [7] in a multi-band speech recognition task.

5.1 Preliminary experiments

In the preliminary experiments, an HMM of simple topology was used to show the validity of our OBS-HMM algorithm, and to verify the approximations we made to speed up the computation of the Hessian matrix. The investigation focused on the measurement of saliency for each deletion. Since the saliency term indicates the importance of the deletion choices, we would like to compare the computed saliency and the actual change in the total log-likelihood of the training data. Another preliminary experiment was done to investigate the effect of forced alignment used in the calculation of the Hessian matrix. Since the calculation of the Hessian matrix is computationally expensive, we proposed to use speech segments generated by Viterbi forced alignments to greatly reduce its computation.

5.1.1 Configuration of experiments

The HMMs used in the preliminary experiments were evaluated on the adult data set of TIDIGITS [31]. This corpus contains a total of 8623 utterances of connected digits in the training set and 8700 utterances of connected digits in the testing set. The speech data were converted to 39 dimension mel-frequency cepstral coefficient (MFCC) vectors consisting of 12 mfcc + 1 normalized energy term and their first and second-order derivatives with cepstral mean subtraction. During MFCC conversion, the window size and target rate is 25 msec and 10 msec respectively. It uses hamming windowing and its pre-emphasis coefficient is 0.97.

There are 11 digit HMMs in the HMM set: digit “0” to digit “9” and ”oh”. Each of them has 16 states and 16 Gaussian mixture in each state. They were trained to convergence using the Baum-Welch re-estimation algorithm. In the testing phase, Viterbi algorithm was used in recognition without any grammar. In the experiments, the HTK software toolkit [29] was used for both HMM training and recognition test.

The topology of the digit HMMs is shown in Figure 5.1. In additional to transitions to the next state and self-loop transitions, we allowed single-state jumps from each states. The reason for defining this topology is that we wanted to allow more relaxation as well as more deletion choices for OBS. It is reasonable for more than

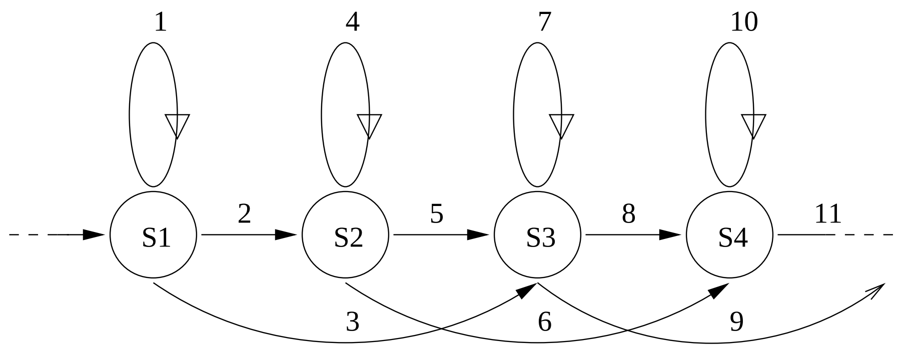

Figure 5.1: Topology of the HMMs used in the preliminary experiments

one deletion choice available in each state so that the result of OBS was not restricted within those self-loop transitions. The transitions are labelled with indexes as shown in Figure 5.1. The indexes allow us to locate each transition deletion. In this topology, the self-loop transitions are labelled as $1, 4, 7, 10, 13, \ldots$; the transitions to the next state are labelled as $2, 5, 8, 11, 14, \ldots$ and the transitions of state jump are labelled as $3, 6, 9, 12, 15, \ldots$.

Notice that the OBS algorithm should be applied to a converged or over-trained HMM set, otherwise the HMM performance will be significantly worse after deletion due to the violation of the extremal assumption explained in Section 3.3.1.

5.1.2 Saliency and change in log-likelihood

The goal of this experiment is to prove that saliency is a good indicator which can reflect the change in total log-likelihood after an HMM transition is deleted. We evaluated the exact change in the total log-likelihood of the training data, denoted as $\delta logL$, since it is the physical meaning of the saliency.

In this experiment, the top 5 ranks of the least important deletions were investigated in the HMM digit "**1**". The saliencies of OBS were generated by solving constrained optimization problem mentioned in Section 3.3 and its exact changes in total log-likelihood of the training data were generated by measuring the differences between $logL$ of HMM before deletion and $logL$ of the modified HMM after deletion.

5.1.2.1 Results and analysis

		OBS	
Rank	**Trans. index**	**saliency**	$\delta logL$
1st	12	-3.53e-36	-1.00e-02
2nd	18	-3.89e+00	-1.07e+01
3rd	39	-1.04e+01	-1.09e+02
4th	3	-1.19e+01	-1.13e+02
5th	33	-1.73e+02	-1.22e+02

Table 5.1: Comparison between OBS saliency and the exact change in (Baum Welch) log-likelihood if a transition is deleted from the digit “0” HMM

Table 5.1 compares the top 5 saliencies and the exact changes in the total log-likelihood of training data of both OBS method. According to the OBS results, the ranks of deletion given by saliency, which is an approximation of the change in log likelihood, is the same as the ranks given by the exact change in the log likelihood ($\delta logL$). This result justifies that saliency is a good approximation of the exact change in the log likelihood ($\delta logL$) in regarding the rank order and, as a result, it is also a good indicator for choosing the least important transition in an HMM.

5.1.3 Effects of Viterbi segmentation on the Hessian calculation

Since the calculation of Hessian is computationally expensive, an approach to speed up the computation was investigated. In computing the forward probabilities using the Baum-Welch algorithm, we suggested that only the utterance segments, which are extracted by Viterbi [32] forced alignment, are used in calculating the Hessian matrix. That is, Viterbi forced alignment was applied on all training utterances, then based on this alignment and the target HMM **X**, all segments of label **X** were extracted and used for computing the Hessian matrix. Since the lengths of utterances are greatly reduced, the computational cost of Hessian can be reduced proportionally.

In this experiment, the deletion results with top 12 saliencies were computed from segmented data and unsegmented data. OBS method was applied to both data set with the same experimental setting.

5.1.3.1 Results and analysis

	OBS on segmented data	**OBS on unsegmented data**	
Rank	**Trans. index (saliency A)**	**Trans. index (saliency B)**	**A:B**
1st	6 (-2.511e-74)	6 (-2.425e-74)	1.035
2nd	3 (-2.464e-48)	3 (-2.427e-48)	1.015
3rd	11(-4.195e+00)	11(-4.394e+00)	0.955
4th	5 (-1.307e+03)	5 (-1.297e+03)	1.008
5th	10(-1.492e+03)	10(-1.488e+03)	1.003
6th	2 (-1.506e+03)	2 (-1.516e+03)	0.993
7th	8 (-1.528e+03)	8 (-1.522e+03)	1.004
8th	4 (-6.690e+03)	4 (-6.487e+03)	1.031
9th	1 (-3.255e+04)	1 (-3.229e+04)	1.008
10th	7 (-4.657e+04)	7 (-4.640e+04)	1.004
11th	9 (-5.983e+04)	9 (-5.964e+04)	1.003
12th	12(-7.145e+04)	12(-7.141e+04)	1.000

Table 5.2: Comparison of the top 12 OBS saliencies computed from Viterbi segmented data and unsegmented data for the digit “1” HMM

The experimental result is shown in Table 5.2. To compare the of saliencies computed from segmented data (saliency A) and unsegmented data (saliency B), their ratios (saliency A:saliency B) was also computed. We can observe that their orders of sorted saliencies generated by segmented data and unsegmented data were exactly the same. Their magnitudes were also very similar to each other. According to their ratio, the maximum ratio was only 1.035, which indicated a difference smaller than 4% and it was in-sufficient to affect the saliency order that computed from segmented data. From the results, we can conclude that the Hessian calculation benefits from the Viterbi segmentation by reducing computational cost and it shows no difference from the result generated from unsegmented data. Therefore, Viterbi segmented data are preferred and will be used in the calculation of the Hessian matrix in all remaining experiments.

Algorithm 4 The algorithm of control method

```
1. for each transition, delete the transition and normalize other transitions of the
   same state in the HMM
2. calculate its corresponding change in the total log-likelihood by the Baum
   Welch algorithm
3. remove the least important transition according to the sorted list of changes in
   the total log-likelihood
4. go back to Step 1 until no transition can be removed.
```

5.2 Application experiments

In this section, we applied OBS to prune HMM transitions and states in a left-to-right HMM and a composite HMM respectively. The left-to-right HMM is used in common speech recognition tasks and the composite HMM topology has been used in multi-band recognition which integrates several left-to-right sub-band HMMs into one large complex HMM.

5.2.1 Control method of experiments

To compare the deletion choices and evaluate the approximations made in the OBS method, a control method is necessary. In transition deletion, the basic method in choosing the least important transition is to manually remove each transition and measure its corresponding change in the total log-likelihood of training data. This brute force method should give the true importance of transitions as it measures the exact change in the cost function. However, the control method requires an exhaustive search on all possible deletions within an HMM and, therefore, this method is time-consuming and computationally expensive. The detailed procedure of this control method is shown in Algorithm 4. In the later sections, the control method is named as manual pruning.

5.2.2 Complexity of manual pruning and OBS

To analyze the complexity of OBS and manual pruning method, several symbols should be defined. In an HMM, the number of states is labelled as N and the total number of transitions is labelled as M. The complexity of the Forward Algorithm in the calculation of log-likelihood is notated as $O(T_{BW})$ which also equals to $O(N^2 \times T)$.

In the manual pruning method, the main computational cost comes from the Forward Algorithm. In one iteration of manual pruning, we need to execute the compute Forward Algorithm for each transition. Therefore, the complexity of manual pruning method in terms of complexity of Forward pass algorithm T_{BW} is $O(M \times T_{BW})$.

If further parameter re-estimation is applied to the HMM in each iteration of manual pruning method, the transitions can be re-adjusted and the deletion choice will be made based on the re-adjusted HMM. The total complexity of manual pruning method plus re-training will be $O(M \times R \times T_{BW})$ where R is the average number of re-estimation iterations.

In the OBS method, the main computational cost comes from the calculation of Hessian matrix. The computational cost of quadratic programming problem is minimal when comparing with the cost of calculation of Hessian matrix. Recall from the equations of computing Hessian matrix in Section 4.4, the iterative equations of computing gradient and Hessian are also based on the Forward Algorithm. The complexity of the gradient computation and Hessian computation is $O(M \times N^2 \times T)$ and $O(M^2 \times N^2 \times T)$ respectively. To summarize, the overall complexity of the OBS method can be approximated by $O(M^2 \times N^2 \times T) = O(M^2 \times T_{BW})$.

Practically, the manual pruning method should also include additional computational cost. In each Forward Algorithm, there are overheads of file IO. Since the calculation of manual pruning method involves many times of Forward Algorithm, the overheads should be significant to the total computational complexity.

5.2.3 Pruning HMM transitions by OBS

We applied OBS method to pruning transitions of an HMM. In the following subsections, we will evaluate:

- the deletion choice of OBS
- the recognition result of unseen test data

5.2.3.1 Configuration of experiment

The OBS method was applied on digit HMMs which were trained with the adult training set of TIDIGITS. As mentioned in Section 5.1.2, the training set contains 8623 utterances of connected digits. The feature extraction of speech data is the same as that mentioned in Section 5.1.2. The HMM set contains 11 digits. Each of them is continuous density HMM consisting 16 real states with 16 mixtures per state. The HMM topology used in experiment was strictly left-to-right with single-state jumps illustrated in Figure 5.1.

Both of the OBS method and manual pruning method are applied to each HMM individually. The speech utterances of training data are segmented by Viterbi forced alignments. After one iteration round of all HMMs, the modified (pruned) HMMs are concatenated to form a new HMM set. In some cases, the pruning method of one HMM may be terminated but the other HMMs are continue the pruning.

5.2.3.2 Evaluation of deletion choice

The goal of the first evaluation is to verify the deletion choice based on the saliency order given by the OBS method. The manual pruning method was used as a control to compare with the OBS results. The manual pruning method shows the best deletions for pruning the HMM transitions without any optimization but a re-normalization on remaining transitions.

Rank	**OBS (saliency)**	**Manual pruning ($\delta logL$)**
1st	12 (-3.53e-36)	12 (-9.11e-02)
2nd	18 (-3.89e+00)	18 (-1.98e+01)
3rd	39 (-1.04e+01)	3 (-1.10e+02)
4th	3 (-1.19e+01)	39 (-1.13e+02)
5th	33 (-1.73e+01)	33 (-1.23e+02)

Table 5.3: Top 5 saliency based on OBS saliency and the manual pruning method for the digit "**0**" HMM

Table 5.3 shows the top 5 saliency in the HMM of digit "**0**" according to the OBS method and the manual pruning method. In the table, the saliency ranks are shown in term of transition indexes which are arranged in the HMM shown in figure 5.1. We can observe that the deletion choices of OBS method are similar to the manual pruning method. The OBS method does not only make the optimal deletion of transitions, but also re-weight other transitions optimally after each deletion. Whereas in the manual deletion method, the remaining transitions are just renormalized non-optimally after each transition deletion.

5.2.3.3 Evaluation of recognition test

The optimization criterion of the OBS method is the total change in log-likelihood in the training data. In Figure 5.2, it shows the total log-likelihood of training data across each OBS iteration and each manual deletion iteration. In one iteration of the OBS method and the manual method, one single transition was removed from each HMM. Therefore, in a left-to-right HMM with 46 transitions initially, the twelveth iteration implies pruning 12 out of 46 iterations which is about 25% of the transition parameters. On the whole, each iteration also implies a removal of eleven transitions in the HMM set of eleven digits.

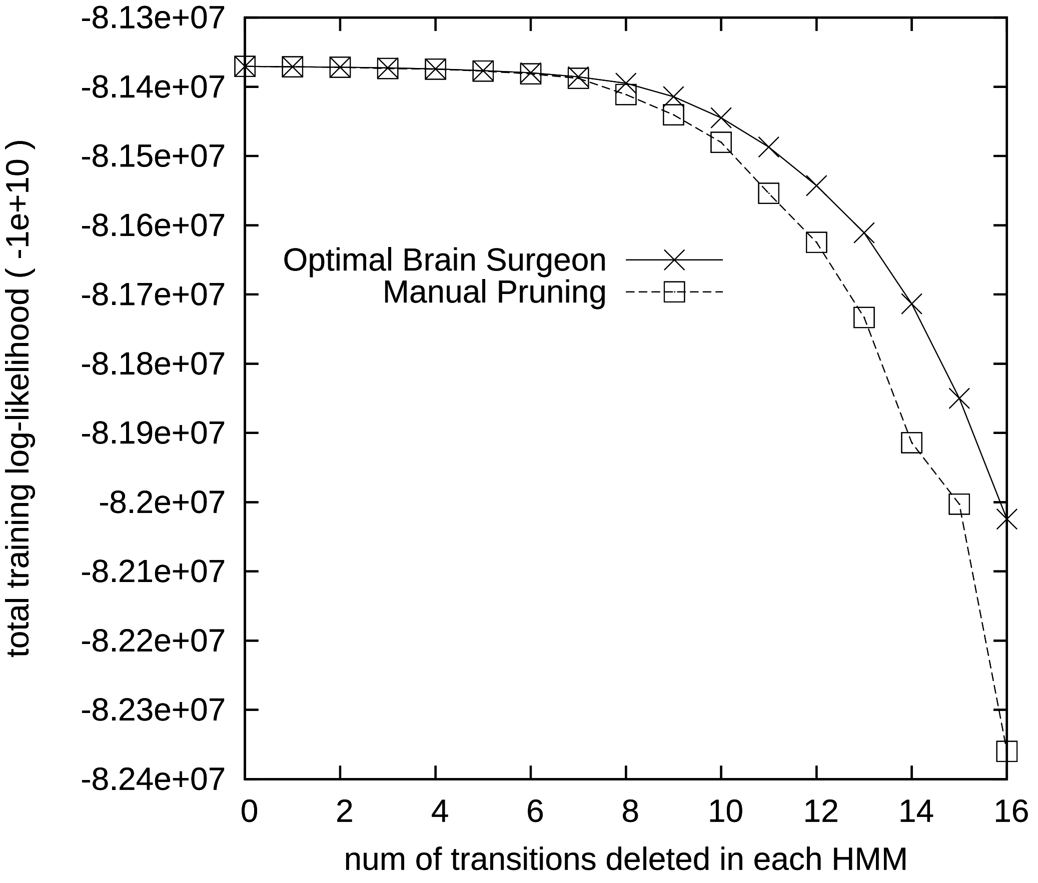

Figure 5.2: The change of total log-likelihood of training data by OBS

According to Figure 5.2, it shows an decreasing trend of training log-likelihood across each iterations. The decreases in the training log-likelihood were induced by each deletion of transition. In the first seven deletions, the deletion choices are unimportant to the HMM set so that the decrease in the training log-likelihood is minimal. However, starting from the seventh deletion, the deleted transitions became more significant and important to the HMM set, so the drops of the training log-likelihood were larger. This showed that the OBS method successfully chose the less important transitions to remove in each iteration. Furthermore, compared with the manual pruning method, the OBS method was able to pick the "better" transitions to remove so as to maintain a higher training log-likelihood afterwards. This was because OBS also re-adjusted other transition weights optimally in order to minimize the decrease in log-likelihood.

Although the goal of OBS in pruning transitions is to reduce the complexity of HMM optimally, the overall recognition performance is also our concern. We want to evaluate the generalization performance of the pruned HMM. Therefore, more iterations of OBS were applied to the HMM set and the modified HMM set were tested on the TIDIGITS testing set. We assumed that the deletion did not break the HMMs; there must be at least a feasible path join from the starting state to the ending state. The overall accuracy of this HMM set across each OBS and manual pruning iterations are shown in Figure 5.3.

From the result, the deletions of transition by OBS did not harm the recognition performance in the first 15 iterations. On the contrary, the recognition performance gradually increases from the baseline 99.2% to 99.4% in the 11th iteration. The main reason is that OBS reduced the model complexity and changed the topology of HMM, at the same time, it also re-distributed other transitions optimally. Starting from iteration 15, the word recognition accuracy dropped below the baseline. Perhaps, by then, the HMM set was over-pruned so that its generalization performance became worse. But, in general, the result showed that the generalization performance of HMM set was sightly improved in the recognition of the unseen test data.

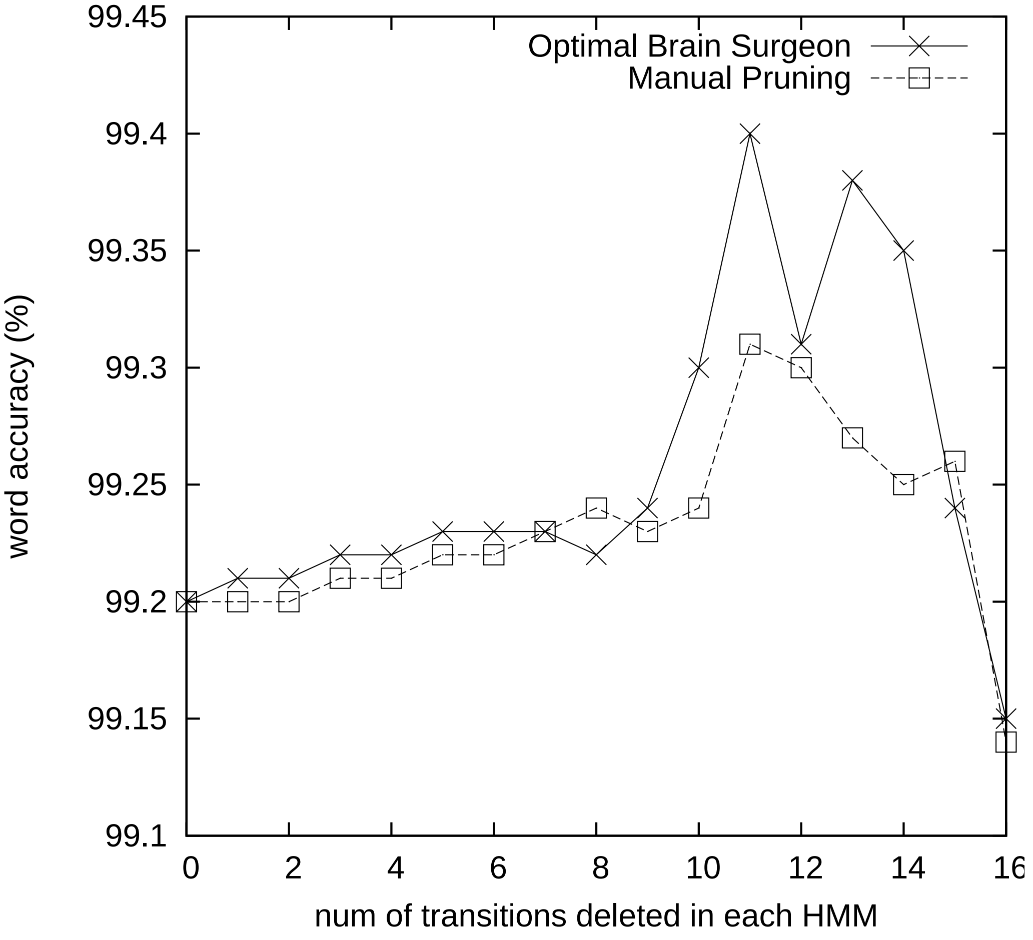

Figure 5.3: Recognition accuracy of HMM set after transition deletions

5.2.3.4 Types of transition pruned by OBS

To investigate the topology of the pruned HMM set, we counted the type of transitions deleted within 16 iterations of the OBS method. In the topology of HMM used in Figure 5.1, there are 3 types of transitions: self-loop transitions, next-state transitions and 1-state-jump transitions. The HMM set in iteration 16 was compared with the reference HMM set and the difference in transitions was recorded.

	self-loop	**next-state**	**1-state-jump**
number of transitions	174	176	148
number of deletion	16	32	111
percentage of deletion	$\frac{16}{174} = 9.2\%$	$\frac{32}{176} = 18.2\%$	$\frac{111}{148} = 75.0\%$

Table 5.4: Type of transitions deleted in the HMM by OBS

Table 5.4 shows that most of the transitions deleted (about 70%) by OBS is the transitions of 1-state-jump. It suggest that, in the TIDIGITS recognition task, 1-state-jump transitions are not useful. Obviously, the strictly left-to-right topology is adequate for this task. However, in many advance cases, there is no prior knowledge of what kind of topology is optimal for one particular task.

5.2.4 Multi-band HMM pruning by Unit-OBS

In a complex HMM topology, there can be hundreds of transitions in each HMM. The OBS method of deleting one transition per iteration is not an appropriate approach in pruning a complex HMM topology as it will require many iterations in order to delete a significant number of parameters. Thus, the unit-OBS method by deleting HMM state is introduced as a faster approach in pruning large and complex HMM. In each iteration of the unit-OBS method, multiple transitions around one particular state are removed instead of deleting a single transition in the OBS method. Therefore, the unit-OBS method leads to a large reduction in the number of iterations required as well as a reduction in the computational cost.

5.2.4.1 Configuration of experiment

In this section, experiments of the unit-OBS method was applied to the composite HMMs. The composite HMMs were used in the multiband HMM recognition tasks by combining multiple left-to-right HMMs from each frequency band to form a single HMM topology (Figure 1.1). The sub-band HMMs were trained with the multi-condition training set of the Aurora2 corpus [33]. In this multi-condition training set, both continuous digits speech data in both clean and noisy conditions with 8440 speech utterances were used to train 2 sub-band HMMs. There are 8440 utterances of both clean and noisy digits. Each sub-band HMM has a strictly left-to-right topology with 16 real states and 16 mixtures per state . The recognition performance of the multiband HMM set was tested on the unseen testing data set with a HTK compatible recognizer [7]. The match-condition test set A in Aurora2, which contains 28028 utterances, was chosen as the testing set in this experiment. In reporting the word accuracy of recognition tests, the utterances in clean condition and with SNR -5 are ignored.

5.2.4.2 Evaluation of recognition test

In Figure 5.4, the total log-likelihood of the training data set is plotted against each iteration (i.e. deletion of one state in each HMM) of the unit-OBS method and the manual deletion method. It shows that the declines in the training log-likelihood were gradually increased across each iteration. The plot also shows that the training log-likelihood levels given by the unit-OBS and the manual pruning method were similar in this experiment. This observation can be explained by the change in log-likelihood was not only produced by deletions of multiple transitions around the deleted state, but, more importantly, the removal of state also leaded to a removal of Gaussian mixtures. This removal of Gaussian mixtures was dominant in the change in log-likelihood.

In Figure 5.5, the HMM set pruned by the unit-OBS method gives better recognition performance than that of HMM pruned by manual pruning method during the first 11 iteration generally. It shows a recognition rate increase slightly from 89.28% to 89.35% in five iterations which is the best accuracy that this HMM set can achieve. We would like to emphasize that one state is deleted in an HMM in each iteration. In iteration 5, five states in each HMM were removed so that $5 \times 11 = 55$ states were removed in the whole HMM set. The percentage of reduction of state in each HMM is $5 / 46 \approx 11\%$. However, starting from iteration 12, the recognition performance of HMM pruned by OBS is worse than that of manual pruning method. It was noticed that, in the optimization criterion, the maximization of training log-likelihood does not guarantee a strictly increase in testing accuracy performance. To summarize, the result shows that the generalization performance slightly increased in this multiband recognition task and the generalization performance given by OBS was better than that of manual pruning.

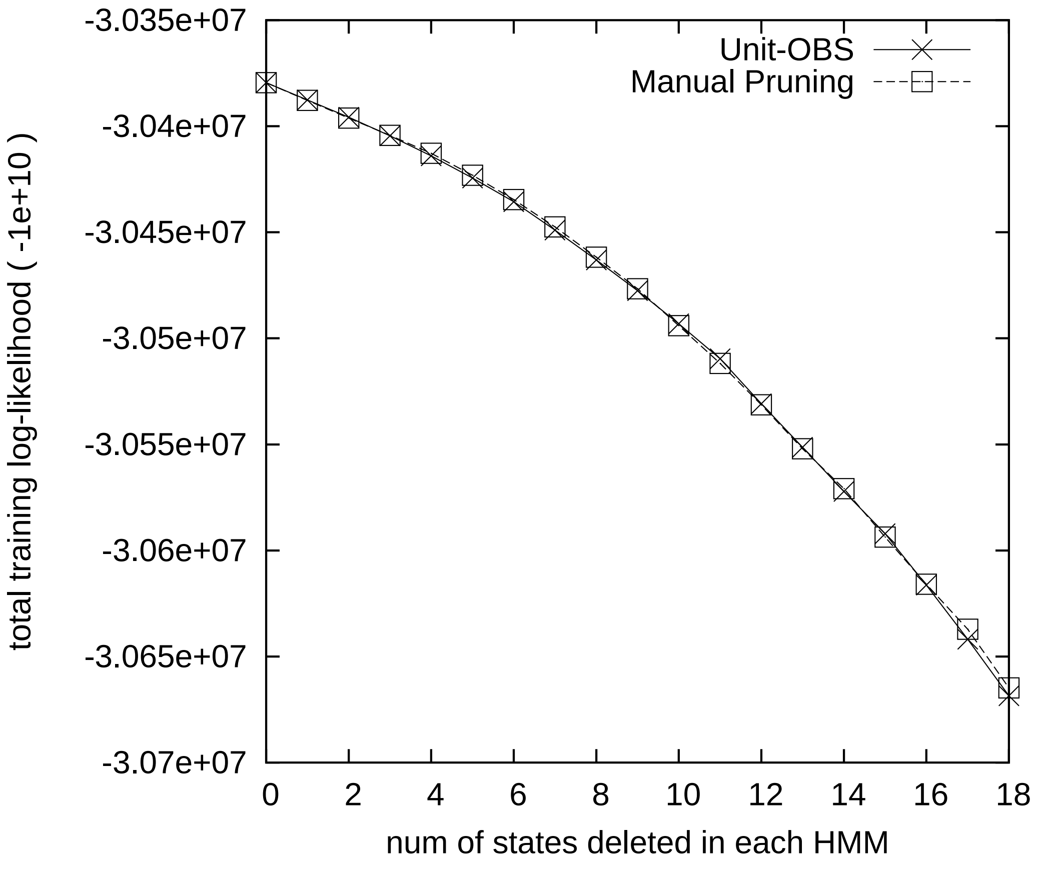

Figure 5.4: The change of total log-likelihood of training data by OBS

5.2.5 Recognition time of different HMM sets

The reduction of HMM complexity also reduce the computational cost in the processing time of the HMM. To investigate the save in computational cost given by OBS and unit-OBS, the user time used in the recognition test is recorded. In the

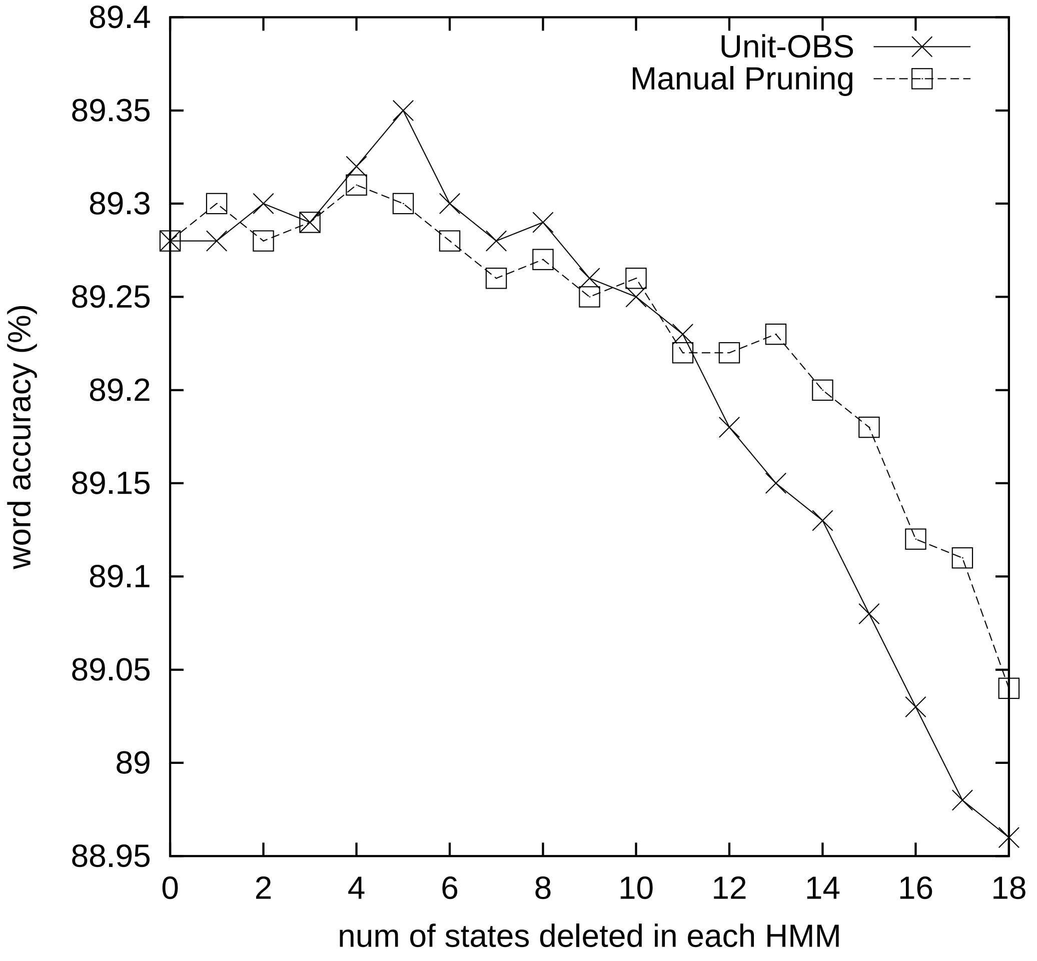

Figure 5.5: Recognition accuracy of HMM set with deletion of state by unit-OBS

experiment, it shows that recognition time of the reference HMM set, the HMM set with best generalization performance and the HMM set with maximum number of parameter deleted. The results are shown in Table 5.5.

From the results, we can conclude that more parameters, which are transitions or states in HMM, deleted, the recognition time became shorter. Although the reduction in recognition time was proportional to number of parameters removed,

OBS			**unit-OBS**		
HMM set	#**iter.**	**user time**	**HMM set**	#**iter.**	**user time**
reference	0	2780.80	**reference**	0	20651.79
best	11	2722.22(-2.11%)	**best**	5	19906.90(-3.61%)
smallest	16	2586.69(-6.98%)	**smallest**	18	18803.13(-8.95%)

Table 5.5: Recognition time of different pruned HMM set

the percentage of reduction in recognition time was not as large as percentage of removed parameters. For example, in the unit-OBS experiment, the reduction in number of states in an HMM is 39%, but the recognition time saved is only 8.95%. This is explained by the fact that recognition time depends not only on the number states or transitions, but also other factors such as the number Gaussian mixtures in used, number of active states in decoding, etc.

5.3 Summary

In this chapter, the results of experiment evaluations in both OBS and unit-OBS methods were shown. The OBS method performed transition deletion in a left-to-right HMM set while the unit-OBS method performed state deletion in a composite HMM set used in a multi-band speech recognition application. We can conclude that:

1. 11 out of 46 transitions from each of the left-to-right digit HMM may be pruned by the OBS method, which is nearly 25% of the total number of transitions in each HMM. The recognition performance on unseen test data was increased slightly from 99.2% to 99.4%. It shows that the OBS method does not only remove the less important transition parameters in order to save memory and computation, but it also improves the generalization of the resulting HMM set.

2. To speed up HMM pruning, the unit-OBS was conducted by removing the most un-important states in an HMM. According to the results in Section 5.2.4, the HMM may reduce 5 out of 46 states in 5 iterations, which is nearly 11% of the total number of states in each HMM. Besides that, the performance of recognition on testing set was improved from 89.28% to 89.35%.

It should be emphasized that the improvement in the generalization performance is not guaranteed for all HMM topologies. In the over-fitted HMM topologies, the OBS / unit-OBS method may improve the generalization performance by reducing the less important parameters. but, in a under-fitted case, the pruning of HMM parameters may harm the performance.

Chapter 6

Conclusion and future directions

In this chapter, the work and the main contributions in this thesis will be summarized. The future directions on the OBS method are also discussed.

6.1 Contributions

In this thesis, the main contributions can be summarized as the following:

I. Adopting OBS in pruning neural network weights to pruning HMM transitions

The pruning method, OBS, is derived for reducing HMM complexity successfully. The main objective of the OBS method is to reduce the HMM complexity and to refine the HMM topology optimally. It makes use of the solution of the quadratic programming problem to perform transition deletion and re-weights the remaining transition parameters. Despite of reducing model complexity, it also brings two advantages to the pruned HMM: it reduces the memory and computation costs, and possibly improves the generalization performance.

II. Adopting unit-OBS in pruning neural network units to pruning

HMM states

The OBS method for pruning HMM transitions is also extended to unit-OBS method for pruning HMM states. State deletion is achieved by allowing deletion of multiple transitions around a state. This method is more suitable for pruning a complex and large HMM topology since it requires fewer iterations than the OBS method. The experimental results showed that unit-OBS can reduce the topology of a composite HMM framework and, at the same time, save the computation and memory costs and the improves its generalization performance.

6.2 Future directions

In the future, the OBS method maybe extended to several directions.

I. Combine OBS and unit-OBS into a robust method of topology selection

It is possible to integrate OBS and unit-OBS method together as a single HMM topology pruning method. Practically, this new method can optimally choose which transition(s) or state(s) to be removed. Since the unit-OBS is restricted to delete all transitions around one particular state, it is not considered as flexible as the OBS method. If a large and complex HMM topology is pruned by unit-OBS method, it is reasonable to remove more transitions individually in order to fine-tune the HMM. This will leads to the investigation of when to start pruning with the OBS method and when to stop pruning with the unit-OBS method

II. Extend the OBS theory to global optimization of the whole HMM set

Instead of refining each HMM topology independently within the HMM set, the global optimization of the whole HMM set is worthy to be considered. In the global optimization of the HMM set, the inter-relationship and the discrimination between models is taken into account. In this thesis, OBS / unit-OBS method is applied on each HMM individually and optimizations are made independently. However, this pruning algorithm has not considered to improve their discrimination abilities. Other than that, deletion of model elements need not to be distributed evenly in each model. If one model is over-sized more than the others, more deletions can be made in this model in such a way that all deletions of parameters in HMMs are distributed according to their needs.

Bibliography

[1] S. Renals, H. A. Bourlard, N. Morgan, M. Cohen, and H. Franco, "Connectionist probability estimators in hmm speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. II, pp. 161–174, 1994.

[2] Z. Ghahramani and M. I. Jordan, "Factorial hidden Markov models," in *Proc. Conf. Advances in Neural Information Processing Systems, NIPS* (D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds.), vol. 8, pp. 472–478, MIT Press, 1995.

[3] B. L. Pedro, "Factorial hidden markov models for speech recognition:."

[4] H. Bourlard and S. Dupont, "A new ASR approach based on independent processing and recombination of partial frequency bands," in *Proc. ICSLP '96*, vol. 1, (Philadelphia, PA), pp. 426–429, 1996.

[5] H. Hermansky, S. Timberwala, and M. Pavel, "Towards ASR on partially corrupted speech," in *Proc. ICSLP '96*, vol. 1, (Philadelphia, PA), pp. 462–465, Oct. 1996.

[6] S. Tibrewala and H. Hermansky, "Sub-band based recognition of noisy speech," in *Proc. ICASSP '97*, (Munich, Germany), pp. 1255–1258, Apr. 1997.

[7] Y.-C. Tam, "Development of an asynchronous multi-band system for continuous speech recognition," master of philosophy thesis, Hong Kong University of Science and Technology, Computer Science Department, May 2001.

[8] R. O. Duda, P. E. Hart, and D. G. Stock, *Pattern Classification*, ch. 6.11. A Wiley-Interscience Publication, 2001.

[9] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[10] P. G. Bryant and O. I. Cordero-Brana, "Model selection using the minimum description length principle," *The American Statistician*, vol. 54, pp. 257–??, Nov. 2000.

[11] A. Biem, J. Y. Ha, and J. Subrahmonia, "A bayesian model selection criterion for hmm toplogy optimization," in *Proc. ICASSP 2002*, pp. 989–992, 2002.

[12] A. Lanterman, "Schwarz, wallace, and rissanen: Intertwining themes in theories of model order estimation," Aug. 2001.

[13] J.Takami and S. Sagayama, "A successive state splitting algorithm for efficient allophone modeling," in *Proc. of the ICASSP 92*, pp. 573–576, IEEE, 1992.

[14] J. Takami and S. Sagayama, "Automatic generation of hidden markov networks by a successive state splitting algorithm," *SYSTEMS AND COMPUTERS IN JAPAN*, vol. 25, no. 12, pp. 42–53?, 1994.

[15] H. Singer and M. Ostendorf, "Maximum likelihood successive state splitting," in *Proc. ICASSP '96*, (Atlanta, GA), pp. 601–604, May 1996.

[16] 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.), vol. 5, pp. 164–171, Morgan Kaufmann, San Mateo, CA, 1993.

[17] A. Girardi, H. Singer, K. Shikano, and S. Nakamura, "Maximum likelihood successive state splitting algorithm for tied-mixture HMNET," in *Proc. Eurospeech '97*, (Rhodes, Greece), pp. 119–122, Sept. 1997.

[18] Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage," in *Advances in Neural Information Processing Systems II* (D. S. Touretzky, ed.), (San Mateo, CA), Morgan Kauffman, 1990.

[19] R. Reed, "Pruning algorithms — A survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–746, 1993.

[20] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Neural networks Theory, Technology, and Applications* (P. K. Simpson, ed.), pp. 56–62, -, 1996.

[21] 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.), vol. 9, p. 669, The MIT Press, 1997.

[22] A. Stahlberger and M. Riedmiller, "Fast network pruning and feature extraction by using the unit-OBS algorithm," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, p. 655, The MIT Press, 1997.

[23] T. Ragg, H. Braun, and H. Landsberg, "A comparative study of neural network optimization techniques," in *Proc. of the ICANNGA 97*, Springer-Verlag, 1997.

[24] B. Hassibi, D. G. Stork, and G. Wolff, "Optimal brain surgeon: Extensions and performance comparison," in *Advances in Neural Information Processing*

Systems (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), vol. 6, pp. 263–270, Morgan Kaufmann Publishers, Inc., 1994.

[25] M. W. P. . D. G. Stork, "Pruning boltzmann networks and hidden markov models."

[26] R. Fletcher, *Practical Methods of Optimization*, ch. 12, pp. 277–330. The Art of Computer Programming, Reading, Massachusetts: John Wiley & Sons, second ed., March 1990.

[27] T. F. Coleman and L. Hulbert, "A direct active set algorithm for large sparse quadratic programs with simple bounds," Technical Report TR88-926, Cornell University, Computer Science Department, July 1988.

[28] E. Panier, "An active set method for solving linearly constrained nonsmooth optimization problems," Technical Report RR-0475, Inria, Institut National de Recherche en Informatique et en Automatique, 1986.

[29] P. C. Woodland, C. J. Leggetter, J. J. Odell, V. Valtchev, and S. Young, "The 1994 HTK large vocabulary speech recognition system," in *Proc. ICASSP '95*, (Detroit, MI), pp. 73–76, May 1995.

[30] B.-H. J. Lawrence Rabiner, *Fundamentals of Speech Recognition*, ch. 6. Pearson Education, 1993.

[31] R. Leonard, "A Database for Speaker-Independent Digit Recognition," in *icassp*, 1984.

[32] M. S. Ryan and G. R. Nudd, "The viterbi algorithm," Research Report CS-RR-238, Department of Computer Science, University of Warwick, Coventry, UK, Feb. 1993.

[33] H. G. Hirsch and D. Pearce, "The aurora experimental framework for the performance evaluations of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000 'automatic Speech Recognition: Challenges for the Next Millennium'*, (Paris, France), Sept. 2000.