

A Factor Graph Approach to Hearing Loss Compensation

René Duijkers

Eindhoven University of Technology
Department of Electrical Engineering

Email: r.duijkers@student.tue.nl

Abstract—In modern hearing aids, Hearing Loss Compensation (HLC) is implemented by dynamic range compression circuits. Although this approach provides reasonable results, it is unable to computationally link the compensation algorithm to the hearing loss problem. In this paper, we propose a data-driven approach that describes HLC as a probabilistic inference problem and automatically infers the solution using message passing on a factor graph. We show that using this approach we can not only derive a unique solution for a specified hearing loss model, but we can also solve the parameter estimation (a.k.a. fitting) as demonstrated by a simulation example. The proposed approach allows straightforward substitution by alternative hearing loss models, and can be applied to other signal processing applications.

Index Terms—Hearing Loss Compensation, Factor Graphs, Sum-product Algorithm, Machine Learning.

I. INTRODUCTION

THE audible area of acoustic waves for humans is limited both in frequency and in sound pressure level. We can display this area as shown in Fig. 1. Here, the lower dashed curve is called the *hearing threshold* (HT) and indicates the minimal intensity sounds need to have to be audible. The upper dashed curve is associated with the *uncomfortable loudness level* (UCL) which is the maximum audible intensity above which sounds become physically painful and may cause damage to the hearing organ. When people have a hearing loss disorder their audible range is reduced. In most cases of hearing loss this is caused by an increased hearing threshold (indicated with L in Fig. 1) while the UCL stays about the same. As a result of this reduced range the impaired person is unable to hear sounds a ‘normal’ listener can hear, and his perception of changing sound levels is distorted.

Hearing loss compensation (HLC) is an algorithm designed for compensating this reduced audible range. The goal of a HLC algorithm is to process sounds, so the impaired user perceives the processed sound close to the way a normal listener would perceive the original sound. In other words, the algorithm performs a mapping from the normal audible range to the reduced range.

Performing this mapping is a challenging task. Not only because the mapping depends on both frequency and sound pressure level, but also because it is unknown how actual hearing loss of the user is described. In modern hearing aids the mapping is realized by a Dynamic Range Compressor (DRC) [1].

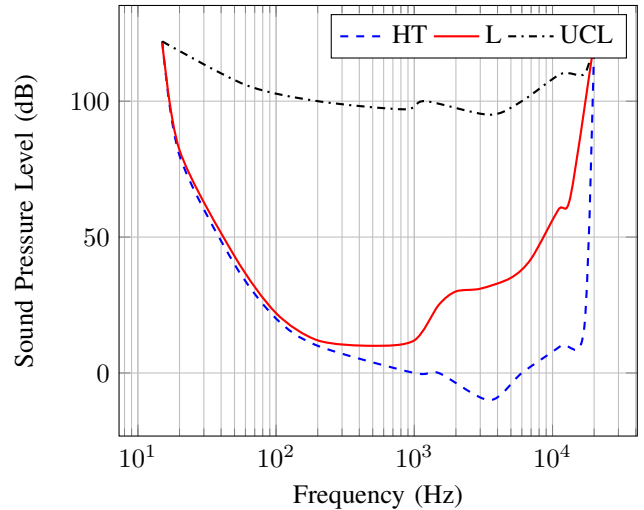


Fig. 1: Hearing threshold (HT) and uncomfortable loudness level (UCL) as a function of frequency. L indicates an increased hearing threshold caused by having a hearing disorder. The area between the UCL and HT corresponds the normal audible area. The area between the UCL and L is the reduced audible area for impaired listeners.

A. Dynamic Range Compressor

A DRC is a dynamic system which reduces the intensity of loud sounds or amplifies quiet sounds by ‘compressing’ a signal’s dynamic range. A typical DRC consists of a variable gain module to compress the signal and a low-pass filter to smooth the gain changes. The amount of compression and smoothing is controlled by a set of parameters. To address the frequency dependency of HLC, most hearing aids use multi-channel compression. In multi-channel compression systems, multiple frequency bands are processed independently, by first splitting the signal into multiple bands using a filter bank and then passing each band through its own independently adjustable DRC circuit [2]. By setting the right parameter values to each individual compressor, multi-channel compression can be used to perform HLC.

The general architecture of a multi-channel compression system is shown in Fig. 2. Here a signal x_n at time-step n is divided into the signals $s_n^{(1)} \dots s_n^{(i)}$, where i indicates the frequency band. In each band, a DRC circuit generates a gain

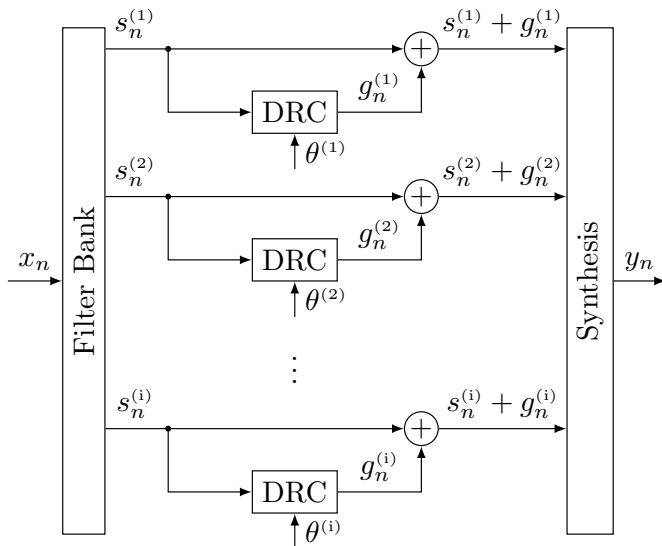


Fig. 2: General architecture of multi-channel compression.

$g_n^{(i)}$ using the parameters $\theta^{(i)}$. Both $s_n^{(i)}$ and $g_n^{(i)}$ are considered to refer to logarithmic power levels. The output y_n is computed by recombining the amplified sounds using a synthesis circuit.

B. Problem statement and our approach

Although using multi-channel compression provides reasonable results, the DRC circuits are designed without reference to an explicit model for hearing loss. As a result, it is not clear what is exactly compensated for. Furthermore, it is not clear how to estimate the parameters of the DRC circuits for the hearing aid patients.

In this paper we address these problems by proposing a new approach to hearing loss compensation based on probabilistic modeling. In this approach, we provide a model that describes the hearing loss and derive a solution which compensates according to this specific model. The proposed probabilistic modelling approach consists of three steps:

- 1) Describe the problem as a data-generating probabilistic model (the "generative" model). For instance, we specify $p(X, Y, Z)$, where Z are the variables of interest, Y relates to observations in the real world and X refers to unobserved (latent) variables such as model parameters.
- 2) Formulate the desired solution as a conditional probability distribution, that in principle can be computed by probabilistic inference on the generative model. E.g. we write $p(Z|Y) = p(Z, Y)/p(Y)$.
- 3) Substitute observed values into the generative model and infer the solution. As we will see in this paper, a factor graph is a formal framework that automates this step using message passing on a graph.

This approach is unusual to signal processing. Traditionally, we formulate a problem and then select the best solution from a set of known solutions. In the probabilistic approach, we always use the same steps to derive a unique solution optimized for the problem.

The goal of this paper is to use the probabilistic modelling approach to derive a solution for HLC using the foregoing steps. Additionally, we show how the same steps can be used to derive an algorithm that estimates the system parameters using user's feedback.

C. Contributions

In short, our contributions are two-fold:

- Using the Sum-product algorithm we propose an update schedule which infers a solution for the HLC (Section III). The solution is based on a probabilistic model given by [3] (Section II). We show that, for a simple hearing loss model, the proposed algorithm corresponds to an optimal Kalman filter.
- In Section IV we show how we can estimate the parameters of this HLC algorithm using a message passing based realization of the Expectation-Maximization (EM) algorithm. The performance of this parameter estimation algorithm is demonstrated by a simulation example in Section V.

II. MODEL SPECIFICATION OF HEARING LOSS COMPENSATION

In this section, we describe HLC as a data-generating probabilistic model, which is the first step of the proposed probabilistic modelling approach. As with multi-channel compression, we implement HLC by using independently operating circuits in the frequency channels of a filter bank. For this we use the same architecture as shown in Fig. 2. Because the filter bank structure is not essential for this paper, we will ignore the specifics of the filter bank and restrict our analysis to only one independent circuit. Like in Fig. 2 we use s_n and g_n to indicate input and gain in dB, now omitting the index i .

A. State-space representation

We describe HLC with a mathematical model following the work of Farmani and De Vries [3]. In their work, they show how HLC can be represented as a state-space model. This model is specified by two equations. The first equation describes how the input level s_n should be mapped from the normal hearing range to the reduced range by amplifying with a gain g_n . This is written as

$$s_n \approx f_{\text{HL}}(s_n + g_n) \quad (1)$$

Here $f_{\text{HL}}(\cdot)$ is a function describing the hearing loss of the user. The equation shows how the amplified sound $g_n + s_n$ should be perceived close to the original sound s_n after being 'filtered' by the hearing loss. We model the approximation in (1) by adding a Gaussian noise term $v_n \sim \mathcal{N}(0, \sigma_v^2)$ leading to

$$s_n = f_{\text{HL}}(s_n + g_n) + v_n \quad (2)$$

The second equation describes how the gain should be restricted from changing too much between two consecutive time steps. This restriction is necessary because rapid changes

in the gain introduces unfavorable effects on the sound quality [1]. We can write this as

$$g_n = g_{n-1} + w_n \quad (3)$$

where $w_n \sim \mathcal{N}(0, \sigma_w^2)$ is a Gaussian noise term representing the amount the gain is allowed to change.

When we combine (2) and (3), we find the complete state-space model describing the HLC:

$$g_n = g_{n-1} + w_n \quad (4a)$$

$$s_n = f_{\text{HL}}(s_n + g_n) + v_n \quad (4b)$$

$$w_n \sim \mathcal{N}(0, \sigma_w^2) \quad (4c)$$

$$v_n \sim \mathcal{N}(0, \sigma_v^2) \quad (4d)$$

In this model, s_n is observed, g_n is a hidden state, the variances σ_w^2 and σ_v^2 are parameters and $f_{\text{HL}}(\cdot)$ is the hearing loss model.

B. Hearing loss filter

The hearing loss function $f_{\text{HL}}(\cdot)$ used in (4) models how the user perceives sounds. Choosing the right hearing loss model is beyond the scope of this research. For illustrative purposes, we use a simplified version of the hearing loss model proposed by Zurek et al. [4]

$$f_{\text{HL}}(x) = \alpha x + \beta \quad (5)$$

This model parameterizes the user's hearing loss problem by two parameters: a scaling factor $\alpha \in \mathbb{R}$ and an offset $\beta \in \mathbb{R}$. Note that using this model, we can rewrite (2) as a simple linear observation model which may not hold in a general case.

C. Probabilistic model

The state-space model (4) can be described by the following joint probability density function (PDF) [5]

$$p(g_0, g_1, \dots, g_n, s_1, \dots, s_n) = p(g_0) \prod_{k=1}^n p(g_k | g_{k-1}) \prod_{k=1}^n p(s_k | g_k) \quad (6)$$

Here, the conditional probability $p(g_k | g_{k-1})$ is given by the state transition (3) and $p(s_k | g_k)$ is given by the observation model (2). With this PDF, we accomplish the first step of the probabilistic modelling approach.

It is important to note that the joint PDF decomposes the model into a product of factors. These factors, which can be even further decomposed using (3) and (2), will play an important role when performing inference as it allows the use of factor graphs and the Sum-product algorithm.

III. GAIN ESTIMATION USING MESSAGE PASSING

The goal of the HLC circuit is to find the value of g_n that compensates s_n based on the model (4). Unfortunately, due to the presence of the Gaussian noise sources v_n and w_n we cannot compute this value directly. Therefore, we want to use a series of past observations $s^n = \{s_1, s_2, \dots, s_n\}$ to 'filter'

this noise providing an estimate of g_n . Traditionally, estimating states in linear Gaussian state-space models based on a series of observations is done by using Kalman filters or related algorithms [5]–[7]. It may be argued that these techniques can be applied to solve the gain estimation problem. However, the presence of s_n as an argument of f_{HL} in the observation process makes this circuit a non-standard state-space model. Rather than deriving the proper Kalman filter for this particular circuit, we follow the more general probabilistic modelling approach and formulate our desired solution as a conditional probability distribution. For the gain estimation problem we write this as

$$p(g_n | s^n) = \frac{p(g_n, s^n)}{p(s^n)} \quad (7)$$

Equation (7) corresponds to the second step of the probabilistic modelling approach. We can write (7) in terms of the distribution $p(g^n, s^n)$ by integrating out the variables we are not interested in

$$p(g_n | s^n) = \frac{\int \dots \int p(g^n, s^n) dg_0 \dots dg_{n-1}}{\int \dots \int p(g^n, s^n) dg_0 \dots dg_n} \quad (8)$$

The distribution $p(g^n, s^n)$ is given by (6), so we can solve this conditional probability distribution when inserting the observed values of s^n . This is the third and final step of our approach as it provides a solution for the HLC problem.

Even though we can compute $p(g_n | s^n)$, the computations are challenging due to the high-dimensional integrals. Because we want run this algorithm online on a hearing aid, we are going to use the recent framework of factor graphs which simplifies and automates this computation by using a message passing algorithm. In the following, we first provide a general introduction about the factor graph notation and the Sum-product algorithm. Then, we provide a standard set of message passing rules which are used to solve (8). At the end of the section we will provide a full analysis of the proposed solution.

A. Factor Graphs and the Sum-product algorithm

A factor graph is a graphical model presenting the factorization of a multivariate function into a product of functions [8]. An example of a factor graph is given in Fig. 3. Here, we see a factor graph representing the factorization

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2) f_2(x_2, x_3, x_4) f_3(x_4, x_5, x_6) \quad (9)$$

We see that square nodes are representing the factors f_1 , f_2 and f_3 and the edges (i.e. connections) are used to represent the variables x_1, x_2, x_3, x_4, x_5 and x_6 . This way of notation was introduced by Forney [9] and is called a *Forney-style factor graph*.

In this paper we use factor graphs to represent probabilistic graphical models, in which case they represent a joint PDF of random variables: each edge represents a random variable and each node represents either the joint or joint conditional PDF of a subset of random variables. This use of factor graphs enables efficient computation of marginal distributions through the Sum-product algorithm as shown in the following example.

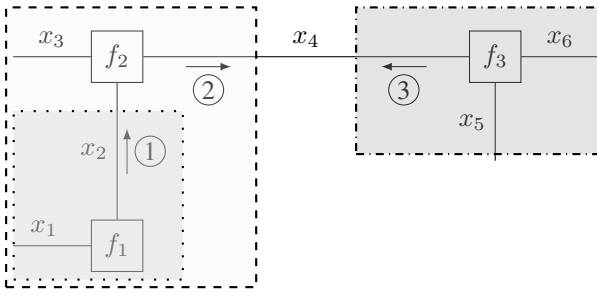


Fig. 3: The factor graph representing the factorization $f_1(x_1, x_2)f_2(x_2, x_3, x_4)f_3(x_4, x_5, x_6)$. The arrows and boxes are used to illustrate the Sum-product algorithm when calculating the marginal $p(x_4)$.

In this example, we have a joint PDF $f(x_1, x_2, x_3, x_4, x_5, x_6)$ and we are interested in finding the marginal of x_4 . We can write this as

$$p(x_4) = \int \cdots \int f(x_1, x_2, x_3, x_4, x_5, x_6) dx_1 dx_2 dx_3 dx_5 dx_6 \quad (10)$$

Assume this joint PDF $f(\cdot)$ can be represented as the factor graph shown in Fig. 3, i.e. $f(\cdot)$ can be written as the factorization (9). Now, we can insert (9) into (10) and write the marginal distribution of x_4 as

$$p(x_4) = \underbrace{\int \int f_2(x_2, x_3, x_4) \underbrace{\int f_1(x_1, x_2) dx_1 dx_2 dx_3}_{\textcircled{1}}}_{\textcircled{2}} \cdot \underbrace{\int \int f_3(x_4, x_5, x_6) dx_5 dx_6}_{\textcircled{3}} \quad (11)$$

By making use of the factorization we have split the four-dimensional integral into smaller ones, simplifying the problem. When looking at the factor graph in Fig. 3, we can display these smaller integrals as boxes presenting summaries of parts of the graph. The factor $\textcircled{1}$ is the summary of everything inside the small dotted box on the left, $\textcircled{2}$ is the summary of the big dashed box on the left and $\textcircled{3}$ is the summary of dashed box on the right. A more common notation of these summaries is by placing arrows on the graph. These arrows are interpreted as messages flowing along the edges of the graph, carrying the complete summary of parts of the graph. Using this interpretation, the final result of (11) can be found by computing the product of the two messages $\textcircled{2}$ and $\textcircled{3}$ arriving at the corresponding edge.

Based on the example we observe the following:

- Marginal distributions can be obtained by computing the product of two opposed messages on the same edge.
- Unlike classic signal processing graphs, messages on a factor graph can flow in two directions and represent distributions.

- Open half-edges such as x_3 in Fig. 3 do not carry a message towards the node. Instead, they carry the constant function ‘1’.
- Messages can be computed using other messages as an intermediate step (e.g. $\textcircled{2}$ is calculated using $\textcircled{1}$)

A general way of computing messages is given by the *Sum-product rule* [10]

$$\mu_{f_{\text{node}} \rightarrow y}(y) = \int \cdots \int f_{\text{node}}(y, x_1, \cdots, x_m) \cdot \mu_{x_1 \rightarrow f_{\text{node}}}(x_1) \cdots \mu_{x_m \rightarrow f_{\text{node}}}(x_m) dx_1 \cdots dx_m \quad (12)$$

Here, $\mu_{U \rightarrow V}$ is the message from U towards V , y is the edge of interest and x_1, \cdots, x_m are all edges except y . The rule states that the message out of a node $f_{\text{node}}(\cdot)$ along edge y is the product of the function and all messages towards $f_{\text{node}}(\cdot)$ along all other edges, integrated over all variables except y .

Calculating messages on an edge by using messages on connected edges and applying the Sum-product rule is the key idea of the Sum-product algorithm. By creating a chain of messages we can condition a variable on other variables on the same graph. This is useful, because when we substitute variables with their observations, we can automatically solve the inference problem using message passing.

Before we show how the Sum-product algorithm provides a solution for our inference problem, we first derive the messages update rules for a set of nodes. We can then use these nodes as ‘building blocks’ to build our graph.

B. Gaussian message passing in linear models

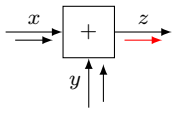
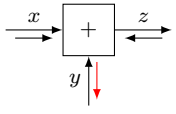
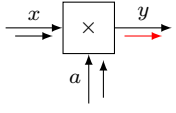
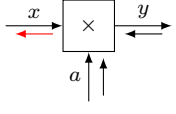
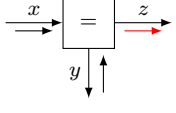
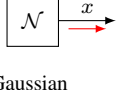
When dealing with linear models we consider nodes (i.e. factorizations) like addition, multiplication, equality and Gaussian noise. When applying the Sum-product rule on these nodes, they all show a useful property: if the messages towards the node are Gaussian distributed the distribution of the result is again Gaussian. This property has two large benefits. First, we can describe all messages by the parameters of the same distribution (e.g. mean and variance). Second, we can derive a set of ‘standard building blocks’ with update equations for these parameters. Since the derivations for these Gaussian message update rules are straightforward, we only provide a summary in Table I. Here we use the mean m and variance V to parameterize the Gaussian distribution¹. For the complete derivations we refer the reader to [11].

Although factor graphs are undirected graphs by definition, the nodes in Table I have edges with arrowheads. We do this for notational ease; the directions of the edges show if the message through the nodes are passed forward or backward.

¹Using this parametrization the one-dimensional Gaussian distribution is written as

$$\begin{aligned} \mu(x) &= \frac{1}{\sqrt{2\pi V_x}} \exp\left(-\frac{(x - m_x)^2}{2V_x}\right) \\ &= \mathcal{N}(m_x, V_x) \end{aligned}$$

TABLE I: Message update equations for Gaussian linear nodes

#	Node	Factor & update equation
1	 Addition	$f(x, y, z) = \delta(x + y - z)$ $m_z = m_x + m_y$ $V_z = V_x + V_y$
2	 Subtraction	$f(x, y, z) = \delta(x + y - z)$ $m_y = m_z - m_x$ $V_y = V_z + V_x$
3	 Multiplication (forward)	$f(x, y, a) = \delta(y - ax)$ $m_y = m_a m_x$ $V_y = m_a^2 V_x$
4	 Multiplication (backward)	$f(x, y, a) = \delta(y - ax)$ $m_x = m_a^{-1} m_y$ $V_x = m_a^{-2} V_y$ With $m_a \neq 0$
5	 Equality	$f(x, y, z) = \delta(x - y)\delta(x - z)$ $m_z = V_z(V_x^{-1}m_x + V_y^{-1}m_y)$ $V_z = V_x V_y (V_x + V_y)^{-1}$
6	 Gaussian	$f(x) = \mathcal{N}(m, V)$ $m_x = m$ $V_x = V$

C. Solving the inference problem

To solve our inference problem (8) using the Sum-product algorithm, we describe the probabilistic HLC model as a factor graph. We do this by making use of the factorization made explicit in (6). This results in the factor graph shown in Fig. 4, which only shows one section containing a single time-step. By concatenating these sections at the dotted lines, the complete graph can be built. Since we express every variable as a Gaussian distribution, we built this graph using the nodes given in Table I.

Now we can estimate $p(g_n | s^n)$ by substituting the observations s^n into the graph and propagating messages from these observed nodes towards the edge representing g_n . In Fig. 4, observed (or known) variables are illustrated with a small black node. We propagate messages by making use of the tree structure: we start from the leaves and proceed with nodes whose input messages become available. A possible way of doing this is by the message schedule illustrated in Fig. 4. Here, the circled numbers ① to ⑭ indicate the order in which these messages are computed for each time-step. The final result of the inference problem is given by the message towards the edge of g_n .

When performing the algorithm every new time-step, the factor graph grows by adding new sections. By expanding the graph, we do not change the structure of the original

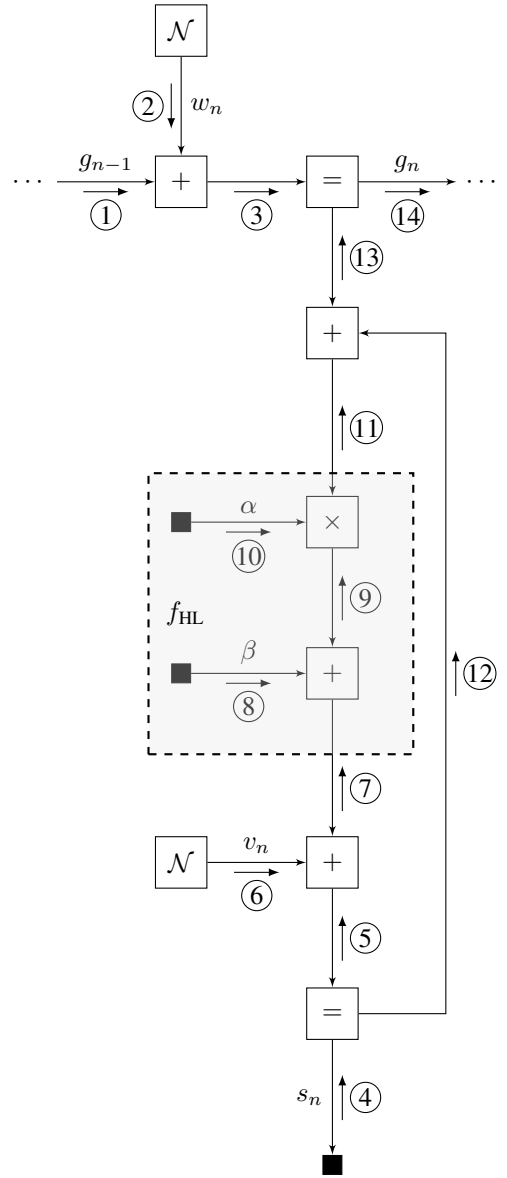


Fig. 4: Factor graph of the HLC. The circled numbers indicate the update schedule solving $p(g_n | g_{n-1}, s_n)$

graph, therefore we can re-use the message already computed in a previous time-step. In other words, when performing the algorithm recursively we already know the distribution $p(g_{n-1} | s^{n-1})$ and only need to solve $p(g_n | g_{n-1}, s_n)$. This corresponds to sending messages on a single section, as shown in Fig. 4.

We analyse this recursive message passing algorithm by writing out all messages. For each message, we indicate which entry of Table I is used to derive the message.

- ① Gain estimate at time $n - 1$:

$$\mu_1(g_{n-1}) = \mathcal{N}(g_{n-1} | \hat{g}_{n-1}, \sigma_{g_{n-1}}^2)$$

- ② Uncertainty w_n (Table I-6):

$$\mu_2(w_n) = \mathcal{N}(w_n | 0, \sigma_w^2)$$

- ③ Adding gain and its uncertainty (Table I-1):

$$\mu_3(\cdot) = \mathcal{N}\left(\cdot \mid \hat{g}_{n-1}, \sigma_{g_{n-1}}^2 + \sigma_w^2\right)$$

- ④ Observed input level:

$$\mu_4(s_n) = \mathcal{N}(s_n \mid s_n, 0)$$

- ⑤ The edge on the right hand side of the bottom equality node does not contain information, so it is given a non-informative message (=1). Thus, the message of 5 equals the message of 4.

$$\mu_5(\cdot) = \mathcal{N}(\cdot \mid s_n, 0)$$

- ⑥ Observation noise v_n (Table I-6):

$$\mu_6(v_n) = \mathcal{N}(v_n \mid 0, \sigma_v^2)$$

- ⑦ Adding observation and observation noise (Table I-1):

$$\mu_7(\cdot) = \mathcal{N}(\cdot \mid s_n, \sigma_v^2)$$

- ⑧ Setting known β :

$$\mu_8(\beta) = \mathcal{N}(\beta \mid \beta, 0)$$

- ⑨ Shifting the level with β (Table I-2):

$$\mu_9(\cdot) = \mathcal{N}(\cdot \mid s_n - \beta, \sigma_v^2)$$

- ⑩ Setting known α :

$$\mu_{10}(\alpha) = \mathcal{N}(\alpha \mid \alpha, 0)$$

- ⑪ Scaling with α (Table I-4):

$$\mu_{11}(\cdot) = \mathcal{N}(\cdot \mid \alpha^{-1}(s_n - \beta), \alpha^{-2}\sigma_v^2)$$

- ⑫ Propagating the input level:

$$\mu_{12}(\cdot) = \mathcal{N}(\cdot \mid s_n, 0)$$

- ⑬ Substraction of the input level (Table I-2):

$$\mu_{13}(\cdot) = \mathcal{N}(\cdot \mid \alpha^{-1}(s_n - \beta) - s_n, \alpha^{-2}\sigma_v^2)$$

- ⑭ Update the gain with information from the new input level (Table I-5):

$$\mu_{14}(g_n) = \mathcal{N}(g_n \mid \hat{g}_n, \sigma_{g_n}^2) \quad (13)$$

$$\hat{g}_n = \hat{g}_{n-1} + \frac{\alpha(\sigma_{g_{n-1}}^2 + \sigma_w^2)(s_n - \alpha(s_n + \hat{g}_{n-1} - \beta))}{\sigma_v^2 + \alpha^2(\sigma_{g_{n-1}}^2 + \sigma_w^2)} \quad (14)$$

$$\sigma_{g_n}^2 = \frac{\sigma_v^2(\sigma_{g_{n-1}}^2 + \sigma_w^2)}{\sigma_v^2 + \alpha^2(\sigma_{g_{n-1}}^2 + \sigma_w^2)} \quad (15)$$

Because g_n is on a half-edge, the message $\mu_{14}(g_n)$ solves the inference problem $p(g_n \mid g_{n-1}, s_n)$. By moving some terms around, we can rewrite (14) and (15) in a more familiar 'Kalman Filter' form with a prediction error e_n and Kalman gain K_n as

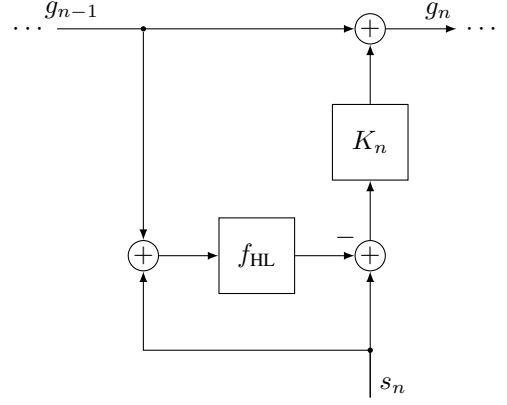


Fig. 5: Block diagram of the recursive estimator that corresponds to the proposed gain estimation algorithm

$$e_n = s_n - \alpha(s_n + \hat{g}_{n-1}) - \beta \quad (16a)$$

$$= s_n - f_{HL}(s_n + \hat{g}_{n-1}) \quad (16a)$$

$$K_n = \frac{\alpha(\sigma_{g_{n-1}}^2 + \sigma_w^2)}{\sigma_v^2 + \alpha^2(\sigma_{g_{n-1}}^2 + \sigma_w^2)} \quad (16b)$$

$$\hat{g}_n = \hat{g}_{n-1} + K_n \cdot e_n \quad (16c)$$

$$\sigma_{g_n}^2 = \frac{\sigma_v^2(\sigma_{g_{n-1}}^2 + \sigma_w^2)}{\sigma_v^2 + \alpha^2(\sigma_{g_{n-1}}^2 + \sigma_w^2)} \quad (16d)$$

The equations of (16) implement a recursive estimator that is depicted in Fig. 5. This is an interesting result. Although we found a solution that looks like a Kalman filter, we did not explicitly derive one. We specified a generative probabilistic model for the observations s_n and used message passing to automatically derive the desired solution $p(g_n \mid g_{n-1}, s_n)$. Note that this form only appears because we have written down the full derivation. If we pass the messages, we are not computing (16) explicitly, but we use a distributed computation scheme with standard update equations.

IV. PARAMETER ESTIMATION

In section III, we derived an algorithm for estimating g_n by passing messages on a factor graph. In this algorithm, the parameters α , β , σ_v^2 and σ_w^2 are assumed to be known. In this section we consider the situation where we have no, or limited knowledge about (some of) these parameters and we want to estimate them using feedback given by the user. Again, we aim to find a solution using the probabilistic modelling approach.

This is really where the approach comes into its own, as the parameter estimation problem is addressed in exactly the same way as the gain estimation problem using the same probabilistic model. We just need to formulate the desired parameter estimation solution as an inference problem and find a message passing schedule which solves this problem.

In this paper, we want to estimate the parameters $\theta = \{\alpha, \beta, \sigma_v^2\}$ by observing both $g^n = \{g_1, g_2, \dots, g_n\}$ and

$s^n = \{s_1, s_2, \dots, s_n\}$. We write this as the conditional distribution

$$p(\theta | g^n, s^n) \quad (17)$$

Here, g^n is a gain signal that satisfies the user, which either comes from a learning database or is generated by a (different) HLC algorithm.

The factor graph we use to solve this inference problem is built by concatenating section given by Fig. 6. Compared to the graph of Fig. 4, we have added additional edges to incorporate the unknown parameters (shown as dashed edges). Since we assume the parameters to be constant from one time-step to the next, the parameter edges are connected using equality nodes. Because we observe g_n , we removed its connection to g_{n-1} and draw a black node. Before we propose an update schedule which solves the inference problem, we first derive additional update rules for the messages towards the parameters.

A. Approximating non-linear nodes using Expectation Maximization

To send messages towards the parameter edges, we first need to derive additional update rules. When we would use the Sum-product rule to calculate these update equations, we get solutions that are non-Gaussian or contain unsolvable integrals (an example is given by [10] p54). To overcome this problem, we have to resort to approximations for these messages. There are different approximation techniques, such as iterative conditional modes, gradient methods or particle methods which can be interpreted as message passing on a factor graph [10]. In this paper, we choose to use a message passing based realization of the *Expectation-Maximisation* (EM) algorithm.

In the general EM algorithm, we are interested in finding the value of θ for which holds

$$\hat{\theta}_{max} = \underset{\theta}{\operatorname{argmax}} f(\theta) \quad (18)$$

Here $f(\theta)$ is assumed to be the marginal distribution of some real-valued function $f(x_1, \dots, x_m, \theta)$, i.e.,

$$f(\theta) = \int \dots \int f(x_1, \dots, x_m, \theta) dx_1 \dots dx_m \quad (19)$$

The EM algorithm attempts to compute (18) by taking four steps [12]:

- 1) Make some initial guess $\hat{\theta}$
- 2) E-Step: evaluate

$$f(\theta) \equiv \int \dots \int f(x_1, \dots, x_m, \hat{\theta}) \cdot \log f(x_1, \dots, x_m, \theta) dx_1 \dots dx_m \quad (20)$$

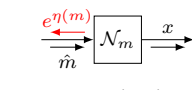
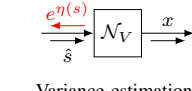
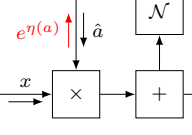
- 3) M-Step: compute

$$\hat{\theta} \equiv \underset{\theta}{\operatorname{argmax}} f(\theta) \quad (21)$$

- 4) Repeat steps 2 and 3 until convergence or until the available time is over.

For a detailed derivation and convergence analysis, we refer the reader to [10], [12].

TABLE II: EM Message update equations

#	Node	Update equation
1	 Mean estimation	$e^{\eta(m)} \propto \mathcal{N}\left(m \mid \frac{s_x \hat{m} + s m_x}{s + s_x}, s\right)$
2	 Variance estimation	$e^{\eta(s)} \propto \text{IG}(s \alpha, \beta)$ $\alpha = -\frac{1}{2}$ $\beta = \frac{1}{2} \left(\frac{\hat{s}(m_x - m)}{\hat{s} + s_x} \right)^2 + \frac{\hat{s}s_x}{2(\hat{s} + s_x)}$
3	 Coefficient Estimation	$e^{\eta(a)} \propto \mathcal{N}\left(a \mid \frac{\mathbb{E}[x, y]}{\mathbb{E}[x^2]}, \frac{s}{\mathbb{E}[x^2]}\right)$

The EM algorithm can be implemented in a message passing framework by expressing both the E-step and the M-step as messages as shown in Fig. 7. Here, the E-step is denoted as the message $e^{\eta(\theta)}$ [12] and is computed by

$$\begin{aligned} \eta(\theta) &= \mathbb{E}_p[\log f(x_1, \dots, x_m, \theta)] \\ &= \int \dots \int p(x | \hat{\theta}) \log f(x_1, \dots, x_m, \theta) dx_1 \dots dx_m \end{aligned} \quad (22)$$

with

$$p(x_1, \dots, x_m | \hat{\theta}) \propto f(x_1, \dots, x_m, \hat{\theta}) \mu_{x_1}(x_1) \dots \mu_{x_m}(x_m) \quad (23)$$

The M-step is computed by taking the mode of the returning message. The complete EM algorithm is performed by iterating the E-step and M-step until convergence or the available time is over.

Table II gives the E-message for the nodes used in Fig. 6. In this table we use $\text{IG}(\cdot)$ to denote an inverse-Gamma distribution. The derivations of the messages and expectations can be found in appendices A to C.

B. Solving the inference problem

Using the messages of Table II, we can solve the inference problem $p(\theta | g^n, s^n)$. We substitute the known values of g^n and s^n into the graph given by Fig. 6 and send messages towards the edges of interest.

Because we need to perform EM iterations we now have to update the messages recursively: a new output message at one node can have influence on the inputs of the same node over another path through the graph. So designing an update schedule becomes a compromise between the accuracy of the approximations and computation time. In this paper, we propose the update schedule given by ① to ②② as shown in Fig 6. This schedule has to be computed for every time-step.

In the proposed update schedule, we divided the algorithm into an E-step and an M-step. The E-step (Fig. 6a) is performed by the messages ① to ①⑥. The goal of this step is to use the best estimate of the parameters to find a new distribution over these parameters. The M-step (Fig. 6b) consists of the

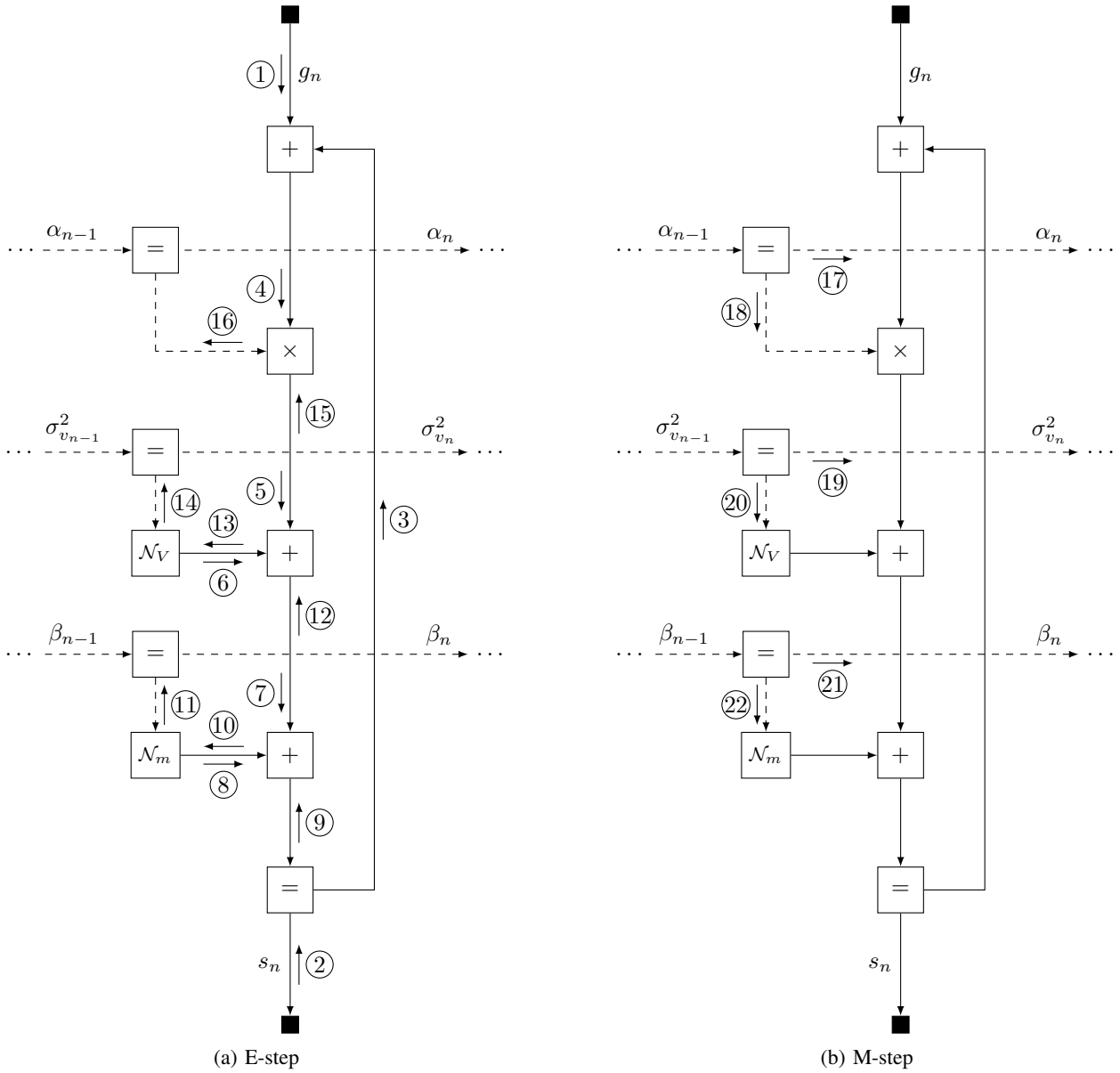


Fig. 6: Factor graph describing parameter estimation for HLC. The left side (a) shows the messages associated with E-step of the parameter estimation algorithm. The right side (b) shows the M-step.

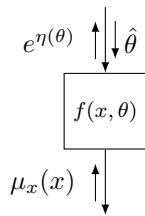


Fig. 7: Factor graph node for the generic EM update rule. The message $e^{\eta(\theta)}$ represents the E-step. $\hat{\theta}$ represents the M-step.

messages (17) to (22) and computes new estimates by combining the newly found distributions with the estimates from the previous time-step. The complete parameter estimation algorithm

is performed by alternating between the E-step and the M-step with a fixed number of iterations, or until convergence.

Providing a full analysis of the parameter estimation algorithm would result in an extensive derivation that would not fit a twelve page paper. Therefore, we demonstrate performance of the parameter estimation algorithm in terms of recovering the model parameters by means of a simulation example in the next section.

V. SIMULATION EXAMPLE

In this section, we provide a simulation example to demonstrate the parameter estimation algorithm. We do not provide a full analysis about the convergence of the algorithm or the identifiability of the model. Therefore the simulation example acts mainly as a proof of concept.

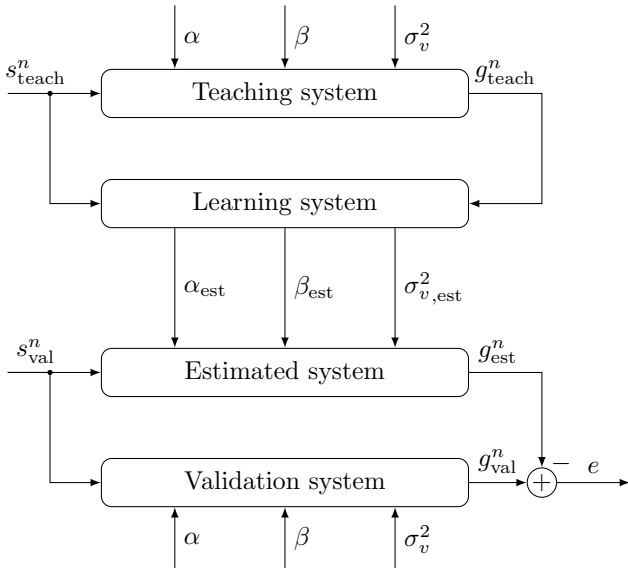


Fig. 8: High level block diagram of the simulation setup. The upper part shows the parameter estimation (learning system) which uses data generated by a HLC algorithm (teaching system). The lower part shows how the estimates are validated by comparing the gain tracks of the original algorithm (validation system) and algorithm using the parameter estimation (validation system).

We want to evaluate if the parameter estimation algorithm can recover the set of parameters $\{\alpha, \beta, \sigma_v^2\}$. Instead of looking at individual parameters, we evaluate the complete system by using a cross-validation approach. First, we estimate the HLC parameters given a dataset that is generated by a HLC algorithm (Section III). This data-generating HLC algorithm is called the teaching system and the parameter estimation algorithm is called the learning system. After learning the parameters, the performance of the original HLC algorithm is compared to a HLC algorithm that uses the parameter estimates. Now we call the original system the validation system and the algorithm that uses the estimates is called the estimated system. In order to limit problems like overfitting, the comparison is done using a different input signal. A block diagram of this experimental setup is shown in Fig. 8.

The teaching system uses fixed parameters and the input signal s_{teach}^n to generate the gain track g_{teach}^n . These parameters are listed in Table III. s_{teach}^n consists of random values drawn from the Gaussian distribution $\mathcal{N}(-70, 1)$. The learning system is performed by the parameter estimation algorithm proposed in Section IV. The initial values that are used for the parameter estimation are listed in Table III. The initial values are randomly picked close to the known parameters. In this paper we do not present any strategy for choosing the initial values.

After estimating the parameters, the final estimates α_{est} , β_{est} and $\sigma_{v,\text{est}}^2$ (listed in Table III) are used as the parameters of estimated system. The performance of this system is compared to the validation system which uses the same parameters as

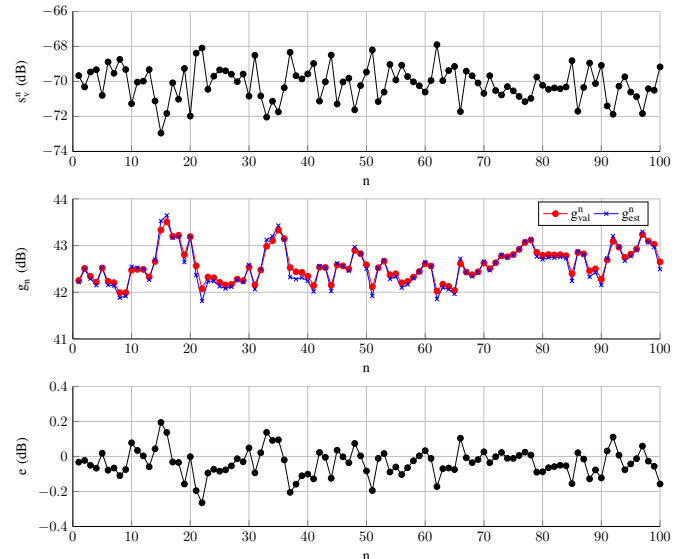


Fig. 9: Results of the simulation example. The upper plot shows the input signal s_{val}^n , the middle plot shows the gain tracks $\{g_{\text{val}}^n, g_{\text{est}}^n\}$ and the bottom plot gives the difference between both gain tracks.

the teaching system. Both use the validation input signal s_{val}^n (shown in the upper plot of Fig. 9) drawn from the same distribution as s_{teach}^n .

The gain tracks of both the estimated system and validation system are shown in the middle plot of Fig. 9. Here, g_{val}^n is the track generated by the validation system and g_{est}^n is generated by the ‘estimated system’. The difference between both tracks is given in the bottom plot of Fig. 9. The relative small differences show that the estimated system comes close to the validation system.

The parameter estimates show a significant difference with the original parameter values, which may indicate identifiability problems of the system. Despite these differences, the similarity of the estimated system and validation system shows that it is feasible to estimate model parameters using message passing, and also provides an implicit validation of the HLC solution.

VI. CONCLUSION

In this paper, we proposed a new approach to hearing loss compensation in hearing aids based on probabilistic modelling. This approach is unusual to signal processing, but it proves to be very powerful in finding solutions.

By describing HLC as a probabilistic model, we showed that the HLC problem can be written as a solvable inference problem. This inference can be solved efficiently using the Sum-product algorithm, which makes use of the factorization of the model. We showed that, in the Gaussian case, the solution can be automatically derived by applying a standard set of message update rules. Using a simple hearing loss model, this solution corresponds to an optimal Kalman filter.

To show how powerful this approach is, we solved the traditional ‘fitting problem’ by describing parameter estimation

TABLE III: Parameters using in the simulation example

Parameters of the teaching and validation system	Value / prior
Time-steps	100
Multiplication factor α	4
Offset β	40
Process noise variance σ_w^2	0.1
Observation noise variance σ_v^2	10
Initial state estimate g_0	$\mathcal{N}(0, \infty)$
Parameters of the learning system	Value / prior
Number of EM iterations	10
Initial multiplication factor α_0	$\mathcal{N}(4.5, \infty)$
Initial offset β_0	$\mathcal{N}(35, \infty)$
Initial observation noise variance $\sigma_{v_0}^2$	IG(2, 7)
Parameters of the estimated system	Value / prior
Estimated multiplication factor α_{est}	3.78
Estimated offset β_{est}	34.02
Estimated observation noise variance $\sigma_{v,\text{est}}^2$	4.48
Initial state estimate g_0	$\mathcal{N}(0, \infty)$

as a different inference problem on the same HLC model. The latter is demonstrated by a simulation example which showed that it is feasible to recover the data-generating system using message passing.

In contrast to classic dynamic range compression solutions, the proposed approach provides a compensation algorithm for a specific hearing loss model given by an engineer. Because the estimation algorithm is derived automatically using the Sum-product algorithm, engineers no longer need to work on this solution, but only need to provide a good hearing loss model. This allows us to design HLC algorithms using the large literature on hearing loss models (e.g. [13], [14]). These models may contain several hierarchical layers or complex dependencies, which would be difficult to compensate for using traditional signal processing approaches.

A promising subject for future work is the implementation of other approximation techniques. Techniques like Variational Message Passing make it possible to use distributions over parameters rather than using maximum likelihood estimates, allowing full Bayesian inference [15].

APPENDIX A

MEAN ESTIMATION USING EM MESSAGE PASSING

In this section we are interested in finding the mean of a Gaussian distribution using the EM algorithm. This Gaussian distribution is described as

$$f(x, m) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s}\right) \quad (24)$$

where $m \in \mathbb{R}$ is the unknown mean and $s \in \mathbb{R}^+$ is the known variance of the Gaussian distribution. We estimate the mean

by computing the E-message $e^{\eta(m)}$ using

$$\eta(m) = \int p(x | \hat{m}) \cdot \log f(x, m) dx \quad (25)$$

$$= \int p(x | \hat{m}) \left(-\frac{1}{2} \log(2\pi s) - \frac{(x-m)^2}{2s} \right) dx \quad (26)$$

$$= C - \frac{1}{2s} \left(m^2 - 2m \int_x p(x | \hat{m}) x dx \right) \quad (27)$$

Here \hat{m} is our initial guess of the mean m and C is a scaling constant containing all terms not depending on m . Due to the quadratic form, we can now write $e^{\eta(m)}$ as a Gaussian distribution

$$e^{\eta(m)} \propto \mathcal{N}\left(m \left| \int p(x | \hat{m}) x dx, s \right.\right) \quad (28)$$

Using (23) we calculate the mean of this distributions as

$$\int p(x | \hat{m}) x dx \propto \int f(x, \hat{m}) \mu_x(x) x dx \quad (29)$$

If $\mu_x(x) \propto \mathcal{N}(x | m_x, s_x)$ is an incoming Gaussian message, this results in

$$\int f(x, \hat{m}) \mu_x(x) x dx \quad (30)$$

$$\propto \int x \exp\left[-\frac{1}{2} \left(\frac{(x-\hat{m})^2}{s} + \frac{(x-m_x)^2}{s_x} \right)\right] dx \quad (31)$$

$$\propto \int x \exp\left[-\frac{1}{2} \left(x^2 \left(\frac{1}{s} + \frac{1}{s_x} \right) - 2x \left(\frac{\hat{m}}{s} + \frac{m_x}{s_x} \right) \right)\right] dx \quad (32)$$

$$= \int x \exp\left[-\frac{1}{2} \left(x^2 \left(\frac{s+s_x}{s s_x} \right) - 2x \left(\frac{s_x \hat{m} + s m_x}{s s_x} \right) \right)\right] dx \quad (33)$$

$$= \int x \exp\left[-\frac{s+s_x}{2s s_x} \left(x^2 - 2x \left(\frac{s_x \hat{m} + s m_x}{s+s_x} \right) \right)\right] dx \quad (34)$$

We can see (34) as the expectation of the Gaussian distribution

$$\mathcal{N}\left(x \left| \frac{s_x \hat{m} + s m_x}{s+s_x}, \frac{s s_x}{s+s_x} \right.\right) \quad (35)$$

which is its mean value. So, the complete E-message is

$$e^{\eta(m)} \propto \mathcal{N}\left(m \left| \frac{s_x \hat{m} + s m_x}{s+s_x}, s \right.\right) \quad (36)$$

This result is equal to equation E.6 in [10].

APPENDIX B

VARIANCE ESTIMATION USING EM MESSAGE PASSING

In this section we are interested in finding the variance of a Gaussian distribution using the EM algorithm. This Gaussian distribution is described as

$$f(x, s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s}\right) \quad (37)$$

where $s \in \mathbb{R}^+$ is the unknown variance and $m \in \mathbb{R}$ is the known mean of the Gaussian distribution. We estimate the

mean by computing the E-message $e^{\eta(s)}$ using

$$\eta(s) = \int p(x | \hat{s}) \cdot \log f(x, s) dx \quad (38)$$

$$= \int p(x | \hat{s}) \left(-\frac{1}{2} \log(2\pi s) - \frac{(x-m)^2}{2s} \right) dx \quad (39)$$

$$= C - \frac{1}{2} \log s - \frac{\int_x p(x | \hat{s}) (x-m)^2 dx}{2s} \quad (40)$$

Here \hat{s} is our initial guess of the variance s and C is a scaling constant containing all terms not depending on s . We can write $e^{\eta(s)}$ as a scaled inverse-Gamma distribution

$$e^{\eta(s)} \propto \text{IG} \left(s \mid \alpha, \beta \right) \quad (41)$$

where

$$\alpha = -\frac{1}{2} \quad (42)$$

$$\beta = \frac{1}{2} \int p(x | \hat{s}) (x-m)^2 dx \quad (43)$$

Next, we want to calculate (43) which is the scale parameter of the inverse-Gamma message we want to send. Because of (23) we can write $p(x | \hat{s})$ as a multiplication of the two Gaussian distributions $f(x, \hat{s}) \sim \mathcal{N}(x | m, \hat{s})$ and $\mu_x(x) \sim \mathcal{N}(x | m_x, s_x)$. Using eq 371 of [16] we write this distribution as

$$p(x | \hat{s}) = \mathcal{N}(x | m_c, s_c) \quad (44)$$

where

$$s_c = \left(\frac{1}{\hat{s}} + \frac{1}{s_x} \right)^{-1} = \left(\frac{\hat{s}s_x}{\hat{s} + s_x} \right) \quad (45)$$

$$m_c = s_c \left(\frac{m}{\hat{s}} + \frac{m_x}{s_x} \right) = \frac{ms_x + m_x \hat{s}}{\hat{s} + s_x} \quad (46)$$

When we insert (46) into (43) we can interpret this as the second moment of the Gaussian distribution $\mathcal{N}(x | m_c, s_c)$. Using eq 380 of [16] we can write this as

$$\frac{1}{2} \int p(x | \hat{s}) (x-m)^2 dx \quad (47)$$

$$= \frac{1}{2} \mathbb{E} [(x-m)^2] \quad (48)$$

$$= \frac{1}{2} ((m_c - m)^2 + s_c) \quad (49)$$

$$= \frac{1}{2} \left(\frac{\hat{s}(m_x - m)}{\hat{s} + s_x} \right)^2 + \frac{\hat{s}s_x}{2(\hat{s} + s_x)} \quad (50)$$

When we insert (50) into (41) we get the E-message for the variance estimation node.

$$e^{\eta(s)} \propto \text{IG} \left(s \mid -\frac{1}{2}, \frac{1}{2} \left(\frac{\hat{s}(m_x - m)}{\hat{s} + s_x} \right)^2 + \frac{\hat{s}s_x}{2(\hat{s} + s_x)} \right) \quad (51)$$

If we would take the case $m = 0$ this result corresponds to eq 4.74 of [17].

APPENDIX C

COEFFICIENT ESTIMATION USING EM MESSAGE PASSING

We are interested in finding the gain coefficient of a multiplication. The factorization of this node is

$$f(x_1, x_2, a) = \delta(x_2 - ax_1) \quad (52)$$

where $a \in \mathbb{R}$ is the coefficient we want to estimate. Computing the E-message would lead to a singularity problem because then we need to calculate $\log \delta(\cdot)$. Therefore we combine the multiplication node with a Gaussian distribution node to "soften" the constraint. The factorization of this combined node is

$$f(x_1, x_2, a) = \frac{1}{\sqrt{2\pi s}} \exp \left(-\frac{(x_2 - ax_1)^2}{2s} \right) \quad (53)$$

where $s \in \mathbb{R}^+$ is the known variance of a zero-mean Gaussian distribution. After choosing an initial guess \hat{a} , we compute the E-message $\eta(a)$ as follows

$$\eta(a) \quad (54)$$

$$= \iint p(x_1, x_2 | \hat{a}) \cdot \log f(x_1, x_2, a) dx_1 dx_2 \quad (55)$$

$$= \iint p(x_1, x_2 | \hat{a}) \left(-\frac{1}{2} \log(2\pi s) - \frac{(x_2 - ax_1)^2}{2s} \right) dx_1 dx_2 \quad (56)$$

$$= C - \frac{1}{2} \log s - \frac{1}{2s} \iint p(x_1, x_2 | \hat{a}) (x_2 - ax_1)^2 dx_1 dx_2 \quad (57)$$

$$= C - \frac{1}{2} \log s - \frac{1}{2s} \iint p(x_1, x_2 | \hat{a}) a^2 x_1^2 dx_1 dx_2 \quad (58)$$

$$+ \frac{1}{2s} \iint p(x_1, x_2 | \hat{a}) 2ax_1 x_2 dx_1 dx_2$$

$$= C - \frac{1}{2} \log s - \frac{\iint p(x_1, x_2 | \hat{a}) x_1^2 dx_1 dx_2}{2s} \quad (59)$$

$$\left[a^2 - 2a \frac{\iint p(x_1, x_2 | \hat{a}) x_1 x_2 dx_1 dx_2}{\iint p(x_1, x_2 | \hat{a}) x_1^2 dx_1 dx_2} \right]$$

where C is a scaling constant containing all terms not depending on a . We can write $e^{\eta(a)}$ as a scaled Gaussian distribution

$$e^{\eta(a)} \propto \mathcal{N}(a | m_a, V_a) \quad (60)$$

with mean

$$m_a = \frac{\iint p(x_1, x_2 | \hat{a}) x_1 x_2 dx_1 dx_2}{\iint p(x_1, x_2 | \hat{a}) x_1^2 dx_1 dx_2} \quad (61)$$

and variance

$$V_a = \frac{s}{\iint p(x_1, x_2 | \hat{a}) x_1^2 dx_1 dx_2} \quad (62)$$

The integrals in (61) and (62) can be seen as the expectations of $x_1 x_2$ and x_1^2 with respect to the distribution $p(x_1, x_2 | \hat{a})$. We compute these expectations by introducing the two-dimensional vector $\mathbf{x} = [x_1 \ x_2]^T$. Using this vector, we can write

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T] = \begin{bmatrix} \mathbb{E}[x_1^2] & \mathbb{E}[x_1 x_2] \\ \mathbb{E}[x_1 x_2] & \mathbb{E}[x_2^2] \end{bmatrix} \quad (63)$$

Which can be rewritten by using the definition of covariance

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T] = \text{var}(\mathbf{x}) + \mathbb{E}[\mathbf{x}]^2 \quad (64)$$

So the expectancies using to compute the E-message can be found by finding the variance and mean of the joint distribution of \mathbf{x} . In order to find this mean and variance we use $p(x_1, x_2 | \hat{a})$ which we can write as

$$p(x_1, x_2 | \hat{a}) \propto f(x_1, x_2, \hat{a}) \mu_{x_1}(x_1) \mu_{x_2}(x_2) \quad (65)$$

When we assume $\mu_{x_1}(x_1)$ and $\mu_{x_2}(x_2)$ are Gaussian distributed we can we augment all three elements to be two-dimensional. For convenience, we do this in the mean-weight matrix representation².

$$\mu_{x_1}(x_1, x_2) = \mathcal{N}_W \left(\begin{bmatrix} m_{x_1} \\ 0 \end{bmatrix}, \begin{bmatrix} 1/s_{x_1} & 0 \\ 0 & 0 \end{bmatrix} \right) \quad (66)$$

$$\mu_{x_2}(x_1, x_2) = \mathcal{N}_W \left(\begin{bmatrix} 0 \\ m_{x_2} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1/s_{x_2} \end{bmatrix} \right) \quad (67)$$

$$f(x_1, x_2, \hat{a}) = \mathcal{N}_W \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \frac{1}{s} \begin{bmatrix} \hat{a}^2 & -\hat{a} \\ -\hat{a} & 1 \end{bmatrix} \right) \quad (68)$$

Using these representations, we can interpreted (65) as propagating three Gaussian messages through an equality node (Korl Table 4.1-1). The result is a joint density which again is a Gaussian distribution

$$p(\mathbf{x}, \hat{a}) = \mathcal{N}_W(\mathbf{x} | m_{\mathbf{x}}, s_{\mathbf{x}}) \quad (69)$$

with mean and weight matrix

$$W_{\mathbf{x}} = (W_1 + W_2 + W_f) \quad (70)$$

$$= \begin{bmatrix} \hat{a}^2/s + 1/s_{x_1} & -\hat{a}/s \\ -\hat{a}/s & 1/s + 1/s_{x_2} \end{bmatrix} \quad (71)$$

$$m_{\mathbf{x}} = \frac{W_1^{-1} m_{x_1} + W_2^{-1} m_{x_2} + W_f^{-1} m_f}{W_1 + W_2 + W_f} \quad (72)$$

$$= W_{\mathbf{x}}^{-1} \begin{bmatrix} m_{x_1} \\ s_{x_1} \\ m_{x_2} \\ s_{x_2} \end{bmatrix} \quad (73)$$

When we fill in this result into (64) we get

$$\mathbb{E}[\mathbf{xx}^T] = W_{\mathbf{x}}^{-1} + m_{\mathbf{x}} m_{\mathbf{x}}^T \quad (74)$$

we can now compute the expectancies $\mathbb{E}[x_1, x_2]$ and $\mathbb{E}[x_1^2]$ as the top-left and top-right element of (74) using (63). This enables the computation of the E-message for estimating the gain coefficient.

$$e^{\eta(a)} \propto \mathcal{N} \left(a \left| \frac{\mathbb{E}[x_1, x_2]}{\mathbb{E}[x_1^2]}, \frac{s}{\mathbb{E}[x_1^2]} \right. \right) \quad (75)$$

ACKNOWLEDGEMENTS

I wish to thank my graduation panel for their time and feedback. I would also like to thank Marco Cox, Thijs van de Laar, Marija Milenkovic and Max Schoonderbeek of the ‘SPS Brats’ team for the advice and assistance I received at the team meetings.

² Using this parametrization the two-dimensional Gaussian distribution is written as

$$\begin{aligned} \mu(\mathbf{x}) &= \frac{1}{\sqrt{(2\pi)^2 |\mathbf{W}_{\mathbf{x}}|^{-1}}} \exp \left(-\frac{1}{2} (\mathbf{x} - \mathbf{m})^T \mathbf{W}_{\mathbf{x}} (\mathbf{x} - \mathbf{m}) \right) \\ &= \mathcal{N}_W(\mathbf{m}_{\mathbf{x}}, \mathbf{W}_{\mathbf{x}}) \end{aligned}$$

Special thanks should be given to Prof. Bert de Vries, my graduation supervisor for his professional guidance and valuable support.

REFERENCES

- [1] H. Dillon, *Hearing Aids*. Boomerang Press, 2012.
- [2] J. M. Kates and K. H. Arehart, “Multichannel dynamic-range compression using digital frequency warping,” *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 18, pp. 3003–3014, 2005.
- [3] M. Farmani and B. de Vries, “A probabilistic approach to hearing loss compensation,” Reims (Fr), Sep. 2014.
- [4] P. M. Zurek and J. G. Desloge, “Hearing loss and prosthesis simulation in audiology,” *The Hearing Journal*, vol. 60, no. 7, pp. 32–33, 2007. [Online]. Available: http://journals.lww.com/thehearingjournal/Abstract/2007/07000/Hearing_loss_and_prosthesis_simulation_in.8.aspx
- [5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. [Online]. Available: <http://www.springer.com/computer/image+processing/book/978-0-387-31073-2>
- [6] S. Sarkka and A. Nummenmaa, “Recursive noise adaptive kalman filtering by variational bayesian approximations,” *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 596–600, 2009.
- [7] N. de Freitas, M. Niranjan, and A. Gee, “Hierarchical bayesian-kalman models for regularisation and ARD in sequential learning,” Department of Engineering, Cambridge University, Tech. Rep., 1998.
- [8] H.-A. Loeliger, “An introduction to factor graphs,” *Signal Processing Magazine, IEEE*, vol. 21, no. 1, pp. 28–41, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1267047
- [9] G. Forney, “Codes on graphs: normal realizations,” *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 520–548, Feb. 2001.
- [10] S. Korl, “A factor graph approach to signal modelling, system identification and filtering,” Ph.D. dissertation, Hartung-Gorre, Konstanz, 2005.
- [11] H.-A. Loeliger, “Least squares and kalman filtering on forney graphs,” in *Codes, Graphs, and Systems, (festschrift in honour of David Forney, 2002)*.
- [12] J. Dauwels, A. Eckford, S. Korl, and H.-A. Loeliger, “Expectation maximization as message passing-part i: Principles and gaussian messages,” *arXiv preprint arXiv:0910.2832*, 2009. [Online]. Available: <http://arxiv.org/abs/0910.2832>
- [13] B. C. Moore and B. R. Glasberg, “Simulation of the effects of loudness recruitment and threshold elevation on the intelligibility of speech in quiet and in a background of speech,” *The Journal of the Acoustical Society of America*, vol. 94, no. 4, pp. 2050–2062, Oct. 1993.
- [14] D.-W. Kim, Y.-c. Park, W.-K. Kim, S. J. Park, W. Doh, S. W. Shin, and D.-H. Youn, “Simulation of hearing impairment with sensorineural hearing loss,” in *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, vol. 5, 1997, pp. 1986–1989 vol.5.
- [15] J. Dauwels, “On variational message passing on factor graphs,” in *IEEE International Symposium on Information Theory, 2007. ISIT 2007*, Jun. 2007, pp. 2546–2550.
- [16] K. B. Petersen and M. S. Pedersen, “The matrix cookbook,” Tech. Rep., 2012. [Online]. Available: http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf
- [17] C. Reller, “State-space methods in statistical signal processing: New ideas and applications,” Ph.D. dissertation, Diss., Eidgenössische Technische Hochschule ETH Zrich, Nr. 20584, 2012, 2012. [Online]. Available: <http://e-collection.library.ethz.ch/view/eth:6740>