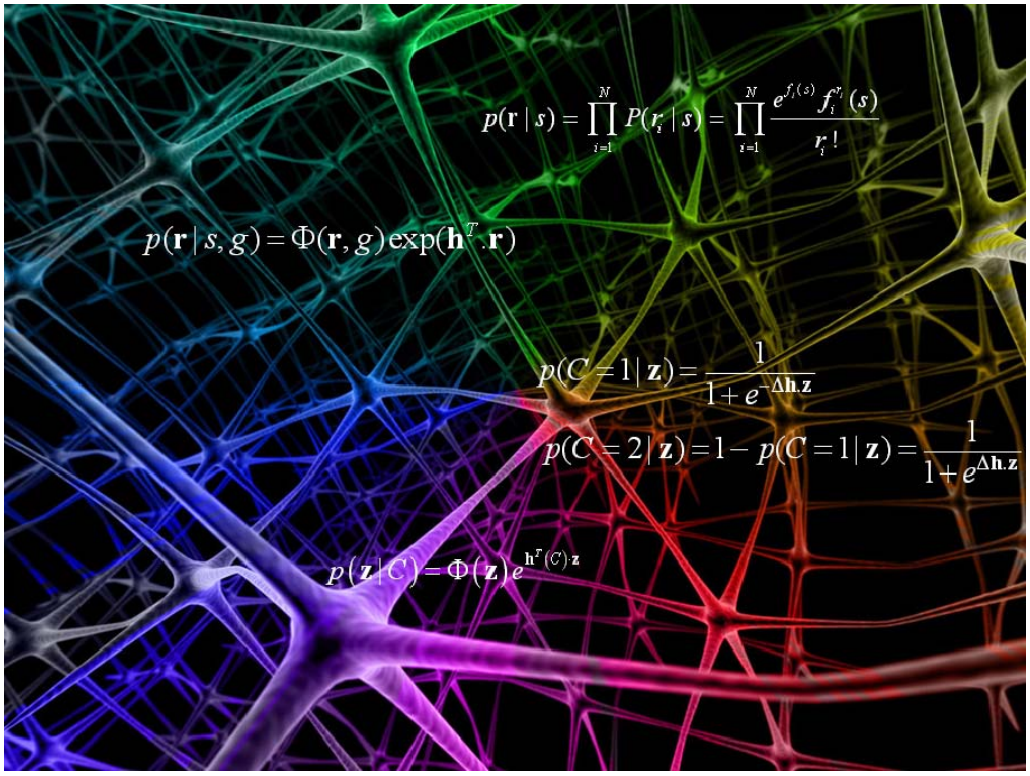
CHALMERS



Neural Implementation of Causal Inference

Master of Science Thesis in Biomedical Engineering

MASIH RAHMATI

Department of Signals and Systems
Division of Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2011
Report No. EX096/2011

REPORT NO. EX096/2011

Neural Implementation of Causal Inference

A neural model of the Causal Inference task using Probabilistic Population Codes(PPC)

Masih Rahmati

Supervisor: Dr. Wei Ji Ma

Department of Neuroscience

Baylor College of Medicine

Houston, United States, 2011

Examiner: Dr. Yngve Hamnerius

Department of Signals and Systems

Chalmers University of Technology

Gothenburg, Sweden 2011

Acknowledgment

I owe my deepest gratitude to Dr. Wei Ji Ma[1] as my supervisor. This thesis would not have been possible without his kind guidance, patience and support from the initial to the final level.

I also should show my appreciation to Dr. Jeff Beck[2] for his great suggestions during this work.

Finally I should thank Dr. Yngve Hamnerius[3] at Chalmers University of Technology as my examiner and advisor at Chalmers.

[1] Department of Neuroscience, Baylor College of Medicine, Houston, Texas, USA
[2] Gatsby Computational Neuroscience Unit, University College London, London, UK
[3] Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden

Contents

Introduction

Imagine that you are walking in the jungle. Suddenly you see a movement in the bushes and simultaneously hear a sound. At that moment you should take different possibilities into account. You may think that the sound you heard and the movement had different sources (for example the sound came from your stepping on a piece of wood and the wind just moved the bushes). On the other hand, you may conclude from the two clues that there is one source for the sound and the movement (for example an animal is behind the bushes and about to attack).

There are many different tasks in which the brain has to infer whether a set of stimuli originate from a common source or from different sources. The task of inferring the number of sources of a set of stimuli is a form of causal inference.

In this project we are trying to address how the brain performs the causal inference task using neurons. More precisely we want to investigate how the brain combines two neural population activities; in our case we consider visual and audio neural populations, for example but not necessarily located in primary visual and auditory cortices (V1 and A1). Our approach is based on the theory that neural populations encode probability distributions over stimuli. Specifically, we use probabilistic population codes [1] to implement the causal inference model proposed by [2]. In [1], Ma et al. proposed a method in which the brain combines different pieces of information encoded as probability distributions using a Bayesian approach; this is called Probabilistic Population Coding (PPC). In [2] Kording et al. suggest a Bayesian behavioral model for causal inference and stimulus estimation. We use the PPC framework to design and implement a neural structure capable of doing causal inference according to the model presented by Kording et al.

Briefly, we suppose that there is a parameter C in the outside world which determines if there is one source, $C = 1$, or two different sources, $C = 2$, for the visual and auditory stimuli. Then we expect that our neural structure is capable of calculating the probability of each of these cases given visual and auditory neural population activities, $P(C=1|\ \mathbf{r_A}, \mathbf{r_V})$ and $P(C=2|\ \mathbf{r_A}, \mathbf{r_V})$, where $\mathbf{r_A}$ and $\mathbf{r_V}$ are the auditory and visual neural population activities, respectively.

There are several challenges we have to deal with in implementing this model. One is that the network should be able to do the task regardless of the reliabilities of the two inputs. Going back to our jungle example, different environmental situations can result in different input reliabilities. Visual reliability could be high or low depending on lighting conditions, and auditory reliability could be high or low depending on the level of background noise.

To measure the performance of our network, we compare its results with the results of the behavioral model proposed by Kording et al, when fed by the same test input data. Chapter 1 contains a detailed statement of the problem. In chapter 2, the previous work on this problem is discussed. Chapter 3 contains details of the methods we used, and the simulation results are presented in Chapter 4.

Chapter 1: Background

In many ecologically important situations, the brain combines multiple pieces of information (cues) from one or multiple sensory modalities. For example, when trying to understand what someone is saying, one combines auditory information with visual information about the movement of the lips. To judge the depth of a surface, one might combine visual and tactile information. In such cue combination, the way in which the brain combines these pieces of information is of interest. It is not always a good idea to combine, since the cues might have been produced by different sources. Evaluating the probability of two cues having the same source and using this information to judge the underlying stimulus or stimuli is known as causal inference. The goal of this thesis is to investigate the neural basis of an optimal model of cue combination in this general setting.

We will study cue combination at a behavioral and at a neural level. At the behavioral level, we abstract the internal representation of a stimulus without referring to neurons. Yet, it is possible to write down a model that describes how a probabilistically optimal observer would solve the task. The predictions of this model can then be compared with behavioral experiments in humans, as has been done in previous studies. At the neural level, the goal is to implement the computations of the optimal behavioral model using a biologically plausible model of neural populations. In this chapter we will explain previous studies in cue combination at both the behavioral and neural levels.

1.1. Behavioral models of cue combination

During the last decades, many researchers have focused on probabilistic interpretation of cue combination. For example in one study, the integration of visual and proprioceptive position-information is investigated[3].In this study they concluded that hand position is better predicted by a model which integrates visual and proprioceptive information than by a model that models depends only on one cue. They also showed that the weight of each cue in the integration depends on the precision of the corresponding unimodal information. In another study, the combination of texture and motion in depth perception is considered [4]. In this study, two optimal cue integration models are presented to analyze subject's behavioral data from a task in which subjects were asked to adjust the height of an ellipse until it matched the depth of a simulated cy-

linder defined by texture and motion cues. The generative model of cue integration task in general is shown in figure 1.1, where x_i is the internal presentation of *i* th cue in the brain.

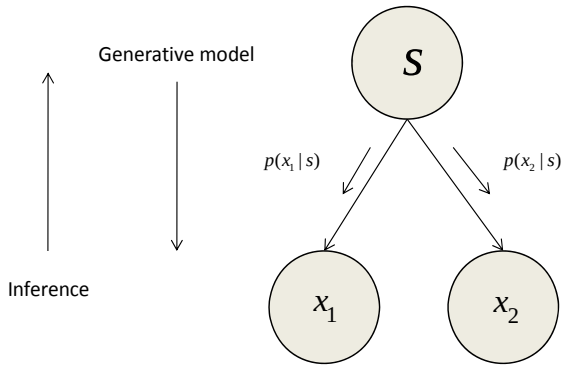


Figure 1.1 Generative model of cue integration for multiple cues from one source.

A generative model is a useful tool to visualize the structure of perceptual tasks. Using generative models, we can show the links between different variables of the model and the probability distribution over each parameter. It also helps us to calculate joint and conditional probability of different variables of the network.

In our generative model there are two kinds of parameters. First, a "state-of-the-world variable", which can be location, orientation, etc. In cue combination, it is the variable in the outside world that generates the cues - *s* in figure 1.1. Second, an "observation" is a noisy representation of the state-of-the-world variable. In cue combination, an *observation* is the internal presentation of a cue inside the brain - x_i in figure 1.1. Then the generative model shows how the observations arise stochastically from the state-of-the-world variable.

In figure 1.1, each node represents a random variable and each arrow represents a conditional distribution. When an observer knows the functional forms of $P(x_1 \mid s)$ and $p(x_2 \mid s)$, he can use Bayes' rule to calculate $p(s \mid x_1, x_2)$, this can be written mathematically as:

$$P(s \mid x_1, x_2) = \frac{P(x_1, x_2 \mid s)P(s)}{P(x_1, x_2)} \qquad (1.1)$$

Here, $P(x_1, x_2)$ is the normalization factor which ensures that $P(s \mid x_1, x_2)$ is a probability distribution and integrates to one. Therefore:

$$P(s \mid x_1, x_2) \propto P(x_1, x_2 \mid s)P(s) \qquad (1.2)$$

Where $P(s \mid x_1, x_2)$ is the posterior. $P(s)$ is the prior which is the probability the observer assigns to *s* without any information about current trial, $P(x_1, x_2 \mid s)$ is the likelihood function over *s*, which modifies the prior using information from current trial . Intuitively, Bayes' rule involves moving "against the direction of the arrows" in the generative model.

In cue integration, we consider a uniform prior over *s*. As x_1 and x_2 are internal representations of two cues originating from a common stimulus, they are not independent. However, for a given *s* they are conditionally independent, which means:

$$P(x_1, x_2 \mid s) = P(x_1 \mid s)P(x_2 \mid s) \qquad (1.3)$$

Therefore we can write equation.(1.2) as:

$$P(s \mid x_1, x_2) \propto P(x_1 \mid s)P(x_2 \mid s) \qquad (1.4)$$

We assume that $P(x_i \mid s)$ has a Gaussian form which is $P(x_i \mid s) \sim \mathcal{N}(\mu, \sigma^2)$, a normal distribution with mean μ and variance σ^2. For a given *s*:

$$P(x_i \mid s) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}} \qquad (1.5)$$

From equation (1.4) we can write:

$$\begin{aligned} P(s \mid x_1, x_2) &\propto \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_1-s)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_2-s)^2}{2\sigma_2^2}} \\ &= \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(s-x_1)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(s-x_2)^2}{2\sigma_2^2}} \qquad (1.6) \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(s-\mu)^2}{2\sigma^2}} \end{aligned}$$

with:

$$\mu = \frac{x_1\sigma_2^2 + x_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \text{ , } \sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \qquad (1.7)$$

Therefore, $P(s \mid x_1, x_2)$ is a Gaussian with mean and variance calculated in equation(1.7). So there is a specific *s,* which equals μ , that maximizes $P(s \mid x_1, x_2)$. As a Gaussian form function, there is an *s* - μ in equation(1.7)- which maximizes $P(x_1, x_2 \mid s)$. Then proportionality of posterior and likelihood in Equation(1.1) implies that this *s* maximizes the posterior too.

From equation(1.7) we can calculate the value of *s* which maximizes the probability of *s* given the data on a single trial - x_1 and x_2. It is called the maximum-a-posteriori (MAP) estimate of *s*. We can write the best estimate as:

$$\hat{s} = w_1 x_1 + w_2 x_2 \qquad (1.8)$$

Where:

$$w_1 = \frac{\frac{1}{\sigma_1^2}}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}} \quad , w_2 = \frac{\frac{1}{\sigma_2^2}}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}} \qquad (1.9)$$

In summary, we supposed that there is one stimulus *s* which have two internal representations x_1 and x_2 in the brain. In each trial for given x_1 and x_2 maximum likelihood calculator calculates the best estimate for the stimulus - $\hat{s}$ in equation(1.8).

This is the idea which many researchers [5-7] have used to study cue integration task. In [4] Ernst and Banks have studied an experiment to investigate integration of visual and haptic information. In this study they measured the variance of visual and haptic estimations of height and used these variances to use in Maximum Likelihood Estimation(MLE). In .such experiments, the experimenter uses two stimuli s_1 and s_2 - as visual and haptic stimulus respectively - which are slightly different, but in a way that the observer still believes that there is a single underlying stimulus.

If the experimenter presents two stimuli s_1 and s_2 repeating over many trials, the MLE will follow a distribution $P(\hat{s} \mid s_1, s_2)$ Two random variables x_1 and x_2 are the internal representations of s_1 and s_2 . Therefore the means of x_1 and x_2 are s_1 and s_2 . Then we can calculate the mean and variance of distribution of the $\hat{s}$ from equation (1.8) :

$$\begin{aligned} \left\langle \hat{s} \right\rangle &= w_1 s_1 + w_2 s_2 \\ \mathrm{var}(\hat{s}) &= \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \end{aligned} \qquad (1.10)$$

Then if there is just one stimulus *s*, $s_1 = s_2 = s$, equation (1.10) reduces to $\left\langle \hat{s} \right\rangle = s$. This is different from equation(1.8) since that equation calculates the best estimation on one trial while equation (1.10) calculates the mean of best guess over many trials.

Equation(1.10) says that the weight of each stimulus is inversely proportional to the variance of the likelihood of each cues in the In other words more reliable the cue is, larger weight the brain gives to it for cue integration. In their study Ernst and Banks showed that this optimal model behaves very similarly to humans in a visual-haptic task.

In another study [5] Alais and Burr investigated how the brain integrates visual and auditory cues when watching a ventriloquist playing a puppet. To do this, they have designed an experiment in which subjects were asked to localize short light blobs or sound clicks. They showed that in this case the brain combines auditory and visual stimuli using weighted some of the stimuli where weight of each stimulus is proportional to it's reliability. Their model also follows equations (1.1) to (1.10) with visual and auditory cues as x_1 and x_2.

1.2. Causal inference

In cue combination studies we discussed above, it is assumed that there is a common source for two perceived stimuli. But in real world this is not always the case. In many situations there are two different sources for two stimuli and the brain should first infer if there is one source or different sources and then integrate cues if they have a common cause. For example in the case of ventriloquist if the performer is not professional enough, the brain can infer that the source of sound and movement are not the same, so it doesn't integrate visual and auditory stimuli. Vice versa if the ventriloquist performs well enough, the brain infers one cause for visual and auditory stimuli and integrates them. In psychophysics experiments, it has been found that there is a considerable interaction between perceptual unification, localization bias and uncertainty of subjects in localization estimation [8].

Wallace et al. showed in their study [8] that when the disparity between auditory and visual stimuli increases, then the localization bias increases. Their results also show that increasing spatial disparity results in lower percentage of unifying reports. In all trials which subjects reported spatially unified stimuli, variability was significantly lower. This means the subject was more certain than cases where subjects reported disjointed stimuli. In unified stimuli cases, higher disparity results in higher variability while in non-unity judgments, higher disparity results in lower variability. Therefore, a complete model of cue combination / multisensory perception should be able to deal with situations in which it is not certain whether two cues have a single source or different sources.

In [2] Kording et al. proposed a causal inference model in multisensory perception. In this paper they have suggested a model of an optimal Bayesian observer for causal inference as well as

source localization. Their model is based on this fact that the brain has access to a noisy presentation of the source locations and should infer the best estimate of those locations while it is considerably uncertain about the presence of a common cause. The optimal observer model defines how two cues should be combined depending on the observer's inference about the presence of a common cause. If the observer believes strongly that the cues have a common cause, they are fused. If the observer believes strongly that there are two different causes, they will be segregated. In practice, the optimal observer always has uncertainty about the number of causes and adjusts its cue combination depending on his degree of belief about the causal structure. Causal inference models tries to predict conditions in which the brain perceives a common cause or different causes as well as the way in which two cues should be combined.

In their study Koridng et al. tried to model cases where subjects are presented by both visual and auditory stimuli at the same time. Their task is to estimate the location of the light and sound sources. Where there is a common cause for two stimuli, one stimulus can be used to improve the estimation of the other stimulus source. One point to be considered is that in some cases there are two sources for two stimuli but there are so close that the observer reports one cause for both of them. Figure 1.2 shows the task.

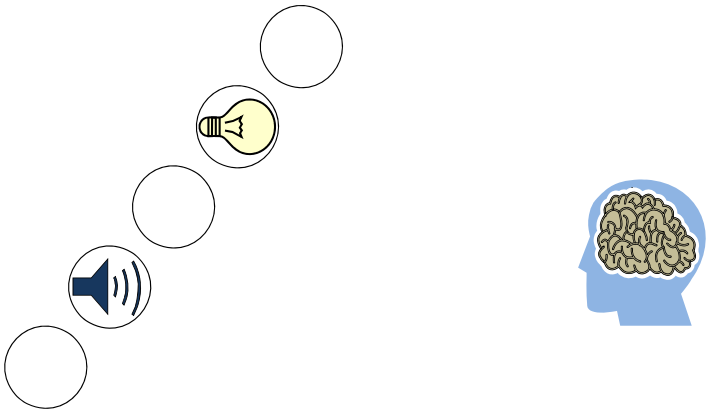


Figure 1.2 Causal inference task.

To perform causal inference task and estimate the position of causes, this model uses Bayesian statistics for two pieces of information. The first piece is the likelihood, which is provided by the observed visual and auditory stimuli. Because of noise in our sensory system, each single stimulus does not provide exact information about its cause but it presents a probability distribution of where the stimulus could be. The second piece of information is priors about the unity of causes- for causal inference- and the positions of sources- for position estimation. Combining these two pieces of information, causal inference model infers if there is a common cause or two different causes and estimates their positions.

The task consists of calculating the probability of a common cause, given x_A and x_V which are the noisy internal representations of auditory and visual stimuli s_A and s_V in the outside world.. Considering inferring number of causes as a binary decision – common cause or different causes – the brain can use a threshold. If probability of a common cause is higher than determined threshold then the brain reports a common cause. If probability of a common cause is smaller than the threshold, the brain infers two different causes for two perceived stimuli. Figure 1.3 shows the generative model of the task.

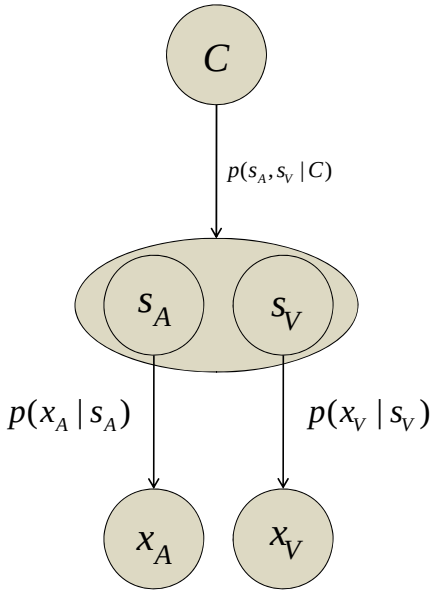


Figure 1.3 Generative model of causal inference.

As can be seen in the generative model, the binary variable *C* determines if there is one cause or two different causes. If *C*=1 it means that auditory and visual stimuli have a common cause and if *C*=2 it means that there are two different causes for s_A and s_V. On each trial, *C* is drawn from a binomial distribution with $P(C=1)=P_{common}$ and $P(C=2)=1-P_{common}$. If there is a common source (*C*=1) a stimulus - here we can say the position of the source - is drawn from a normal prior distribution $N(0,\sigma_p)$ which means a normal distribution with mean 0 and standard deviation σ_p. This means we expect that in most trials the stimulus is close to the center. If *C*=1 we have $s_A = s$ and $s_V = s$. Vice versa if there are two different causes (*C*=2), s_A and s_V are drawn independently from a normal prior distribution $N(0,\sigma_p)$. The noise corrupting auditory and visual signals is modeled as unbiased Gaussian noise with standard deviations σ_A and σ_V. Thus we can say that the internal representation of auditory and visual stimuli are drawn from $N(s_A,\sigma_A)$ and $N(s_V,\sigma_V)$ respectively. Now that we have our generative model the goal is to calculate $P(C \mid x_A, x_V)$.

According to Bayes' rule:

$$P(C \mid x_A, x_V) = \frac{P(x_A, x_V \mid C)P(C)}{P(x_A, x_V)} \qquad (1.11)$$

Also we know that $P(C=1 \mid x_A, x_V) + P(C=2 \mid x_A, x_V) = 1$. We consider a prior probability P_{common} for $P(C=1)$ and $1-P_{common}$ for $P(C=2)$, so we can rewrite equation(1.11) as:

$$P(C=1 \mid x_A, x_V) = \frac{P(x_A, x_V \mid C=1)P_{common}}{P(x_A, x_V \mid C=1)P_{common} + P(x_A, x_V \mid C=2)(1-P_{common})} \qquad (1.12)$$

In view of the conditionally independence of x_A and x_V we can write from the generative model in figure 1.3:

$$\begin{aligned} P(x_A, x_V \mid C=1) &= \iint P(x_A, x_V \mid s_A, s_V, C=1)P(s_A, s_V \mid C=1)ds_A ds_V \\ &= \iint P(x_A \mid s_A)P(x_V \mid s_V = s_A)P(s_A)\delta(s_V - s_A)ds_A ds_V \qquad (1.13) \\ &= \int P(x_A \mid s_A)P(x_V \mid s_A)P(s_A)ds_A \end{aligned}$$

All three factors inside the integral have Gaussian distribution - for *C*=1 are $N(s, \sigma_A)$, $N(s, \sigma_V)$ and $N(\mu_p, \sigma_p)$ respectively. We can calculate $P(x_A, x_V \mid C=1)$ analytically from equation(1.14).

$$\begin{aligned} P(x_A, x_V \mid C=1) &= \int \frac{1}{\sqrt{2\pi\sigma_A^2}} \mathrm{e}^{-\frac{(x_A - s_A)^2}{2\sigma_A^2}} \frac{1}{\sqrt{2\pi\sigma_V^2}} \mathrm{e}^{-\frac{(x_V - s_A)^2}{2\sigma_V^2}} \frac{1}{\sqrt{2\pi\sigma_p^2}} \mathrm{e}^{-\frac{(s_A - \mu_p)^2}{2\sigma_p^2}} ds_A \\ &= \frac{1}{2\pi\sqrt{\sigma_A^2\sigma_V^2 + \sigma_V^2\sigma_p^2 + \sigma_A^2\sigma_p^2}} \exp\left[-\frac{1}{2} \frac{(x_V - x_A)^2\sigma_p^2 + (x_V - \mu_p)^2\sigma_A^2 + (x_A - \mu_p)^2\sigma_V^2}{\sigma_A^2\sigma_V^2 + \sigma_V^2\sigma_p^2 + \sigma_A^2\sigma_p^2} \right] \end{aligned} \qquad (1.14)$$

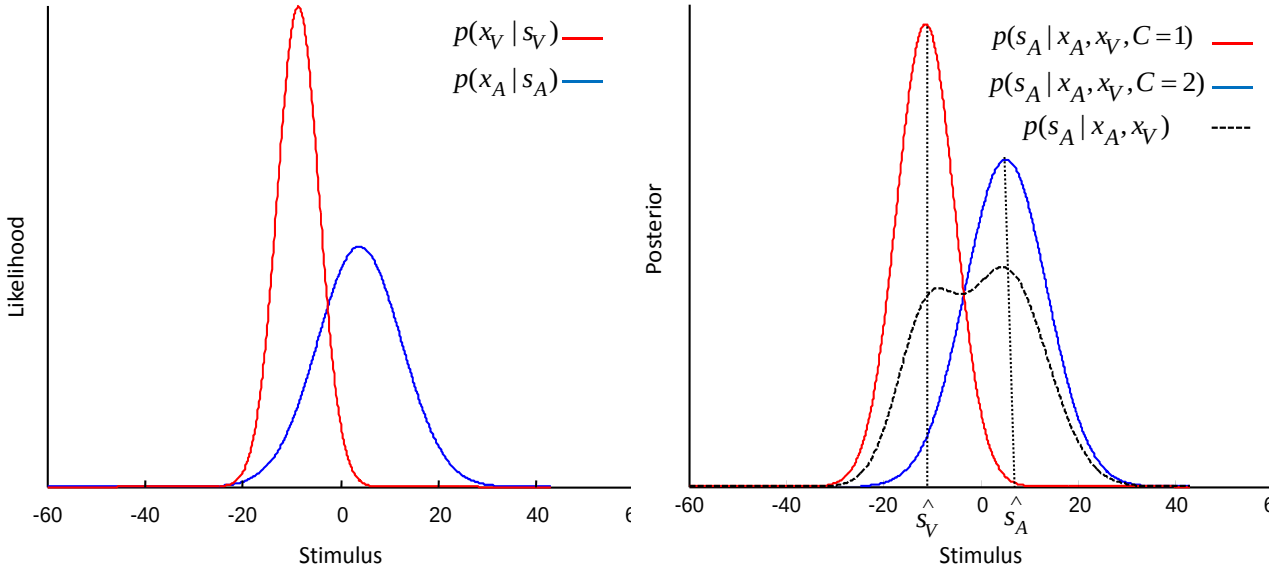
For *C*=2 we can write:

$$
\begin{aligned}
P(x_A, x_V \mid C=2) &= \iint P(x_A, x_V \mid s_A, s_V, C=2) P(s_A, s_V \mid C=2) ds_A ds_V \\
&= \iint P(x_A \mid s_A) P(x_V \mid s_V) P(s_A) P(s_V) ds_A ds_V \qquad (1.15) \\
&= \left(\int P(x_A \mid s_A) p(s_A) ds_A \right) \left(\int P(x_V \mid s_V) p(s_V) ds_V \right)
\end{aligned}
$$

Like for C=1, here all distributions in these integrals are Gaussian. As a result, we can find an analytical answer for $P(x_A, x_V \mid C=2)$ which is:

$$
P(x_A, x_V \mid C=2) = \frac{1}{2\pi\sqrt{(\sigma_A^2+\sigma_p^2)(\sigma_V^2+\sigma_p^2)}} \exp\left[-\frac{1}{2}\left(\frac{(x_A-\mu_p)^2}{\sigma_A^2+\sigma_p^2} + \frac{(x_V-\mu_p)^2}{\sigma_V^2+\sigma_p^2} \right) \right] \qquad (1.16)
$$

After calculating the probabilities of C=1 and C=2, the observer has to make a decision. We use MAP estimation to decide if C equals 1 or 2. We know that $P(C=1 \mid x_A, x_V) + P(C=2 \mid x_A, x_V) = 1$. Thus, the observer uses 0.5 as a threshold for his decision i.e. if $P(C=1 \mid x_A, x_V) > 0.5$ the observer reports C=1 and if $P(C=1 \mid x_A, x_V) < 0.5$, the observer reports C=2. Figure 1.4 shows the performance of the causal inference model.



a b

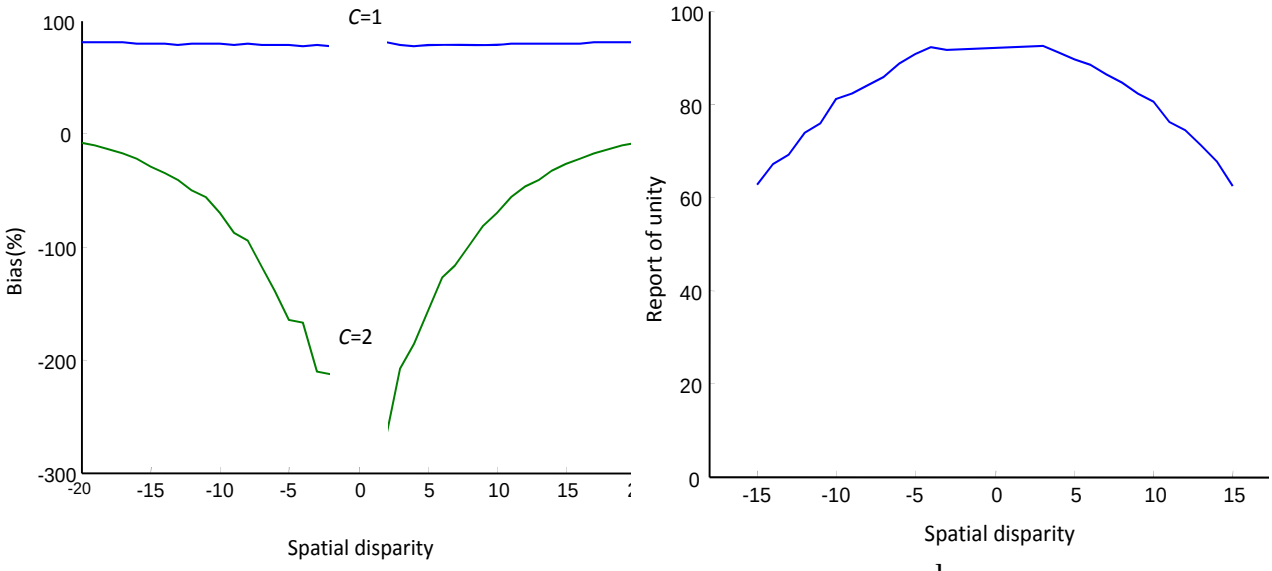


Figure 1.4 (a) Shows the likelihood of S_A and S_V as a function of stimulus. (b) Shows the best guesses for stimulus knowing x_A and x_V respectively. In (c) the Bias - which shows the influence of vision on the perceived auditory stimulus - is plotted as a function of difference between two stimuli. Trials for which reported *C* equals 1 are plotted in blue and trials for which reported *C* equals 2 are plotted in green (d) shows the relative number of trials on which the subject reports C = 1.

The causal inference model gives us an analytical method to calculate probability of common or different causes for given internal representation of two stimuli. But as we know the brain performs any function via its neural populations and networks. So to check the neurally plausibility of the causal inference model, we need a neural structure capable of implementing this model.

1.3. Neural Models

Probabilistic Population Coding

In most of our brain activities the brain receives one or multiple stimuli from the outside world in form of physical signals like light, sound, tactile, etc. These stimuli contain different amounts of information. Depending on their nature- visual, auditory, etc- each stimulus goes to its' corresponding region of the brain to be processed. Inside the brain these stimuli are presented as neural activities. In fact the information carried by the stimuli is encoded in neural activities of the corresponding part of the brain. Then the brain decodes and processes this information to be prepared for different operations such as decision making, object recognition, etc. The general schematic of this procedure is shown in figure 1.5

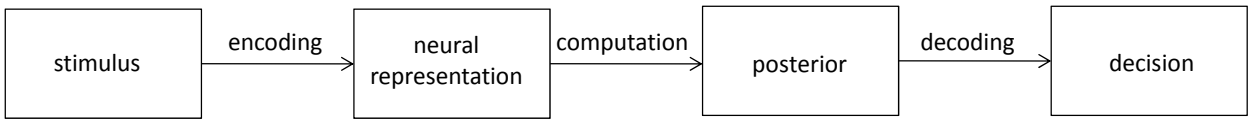


Figure 1.5 Schematic of what the brain does with stimulus.

Before describing how neural populations encode information, some basic concepts need to be described.

Stimulus, preferred stimulus, tuning curve

Stimulus can be defined as any physical feature of the world. It can be distance between two objects, number of an event in a specific period of time, color of an object, etc. Each stimulus is processed in a specific part of the brain consists of a population of neurons. For simplicity we can suppose that there are a line of neurons in each part. When the corresponding part receives the stimulus its' neurons start firing neural spikes. In fact the activity of a neural population in response to a received stimulus encodes information content of that stimulus.

Each neuron in a neural population is most sensitive to a specific value of the stimulus. In other words, different neurons in a neural population are tuned to different values of stimulus. The value to which each neuron is tuned is called preferred stimulus of that neuron. Figure 1.6 shows the preferred stimulus for a single neuron. This curve is called tuning curve of a neuron which shows the mean response of a single neuron in neural population to all possible stimuli. The neuron fires most intensely in response to the stimulus to which it is tuned. Tuning curve and preferred stimulus are shown in figure 1.6.

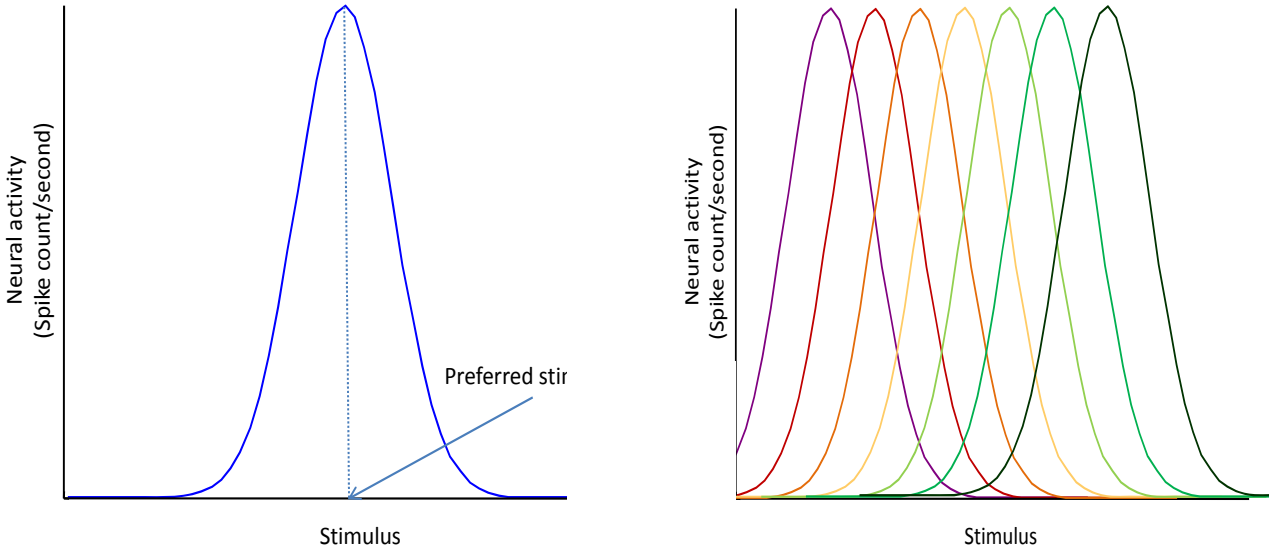


Figure 1.6 (a) Tuning curve and preferred stimulus of one neuron; (b) Tuning curve of a neural population

Variability

When a neuron is stimulated with a stimulus it fires neural spikes in a rate proportional to the value of the stimulus. But it has been shown that if we stimulate a neuron with a specific stimulus on different trials, the firing rate of the neuron differs trial to trial. In neural population this variability in neural activities is proportional to the mean of activities i.e. larger mean of neural activity results in larger variance in distribution. Figure 1.7 shows population activities for a single trial and for multi trials case.

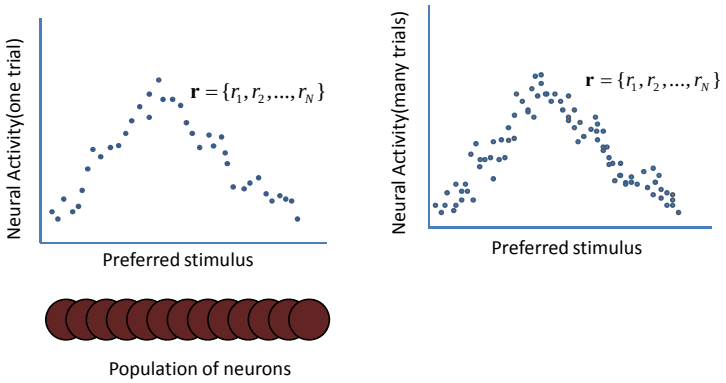


Figure 1.7 Population activity and neural variability

Encoding

When there is variability it means that on each trial one cannot specifically say that what the value of the stimulus is. This leads us to this conclusion that neural populations encode probability distribution over stimuli not their exact value [9, 10]. One of the most important reasons for encoding probability distributions instead of deterministic values of stimuli is the role of uncertainty in neural computations. When there is variability it means that on each trial one cannot specifically say that what the value of the stimulus is. This leads us to this conclusion that neural populations encode probability distribution over stimuli not their exact value. One of the most important reasons for encoding probability distributions instead of deterministic values of stimuli is the role of uncertainty in neural computations. The generative model in figure 1.8 shows the relation between stimulus and population activity.

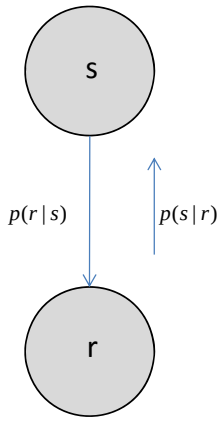


Figure 1.8 Relationship between stimulus and population activity.

One of the distributions that satisfy this mean-variance relation in neural activities is Poisson distribution[11]. This distribution expresses the probability of a number of events in a specific period of time, physical distance, etc with a known average rate and independent of the beginning point. For a Poisson distribution, if the expected number of occurrences is λ, then the probability of exactly k occurrences during the given interval is:

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \qquad (1.17)$$

Now if we show the response distribution of the *i*-th neuron to a given stimulus as $P(r_i \mid s)$ then the variance of the spike count is proportional to the mean of it. Then substituting $f_i(s)$ for λ and r_i for k in equation (1.17) we can write:

$$p(r_i \mid s) = \frac{e^{-f_i(s)} f_i^{r_i}(s)}{r_i!} \qquad (1.18)$$

Where r_i is an integer number which represents the firing rate of the *i*-th neuron in the neural population and $f_i(s)$ is its' tuning curve. Then for the neural population $\mathbf{r} = \{r_1, r_2, ..., r_N\}$ we have:

$$p(\mathbf{r} \mid s) = \prod_{i=1}^{N} P(r_i \mid s) = \prod_{i=1}^{N} \frac{e^{f_i(s)} f_i^{r_i}(s)}{r_i!} \quad (1.19)$$

Here we assume that the firing rate of each of *N* neurons in the population is conditionally independent of other neurons.

According to Bayes rule $p(s \mid \mathbf{r}) \propto p(\mathbf{r} \mid s) p(s)$. In this case, x - the internal representation of *s* - is the *s* which maximizes this likelihood:

$$x = \arg\max_{s} L_{\mathbf{r}}(s) = \arg\max_{s} p(\mathbf{r} \mid s) \quad (1.20)$$

Figure 1.9 shows $L_{\mathbf{r}}(s)$ and *x*.

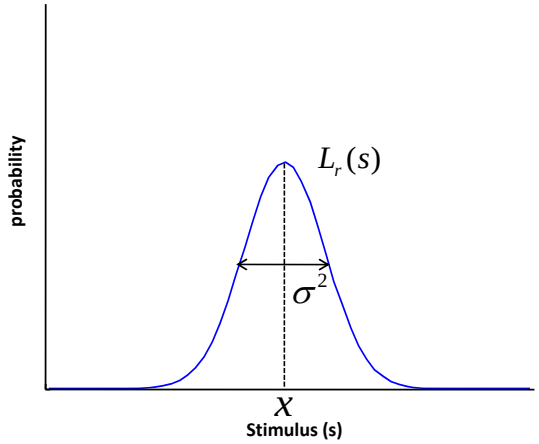


Figure 1.9 The interpretation of *x*.

Decoding

Information encoded in population activities needs to be decoded for being used by other parts of the brain for different tasks such as judgment and decision making. This means knowing the population activity, the brain should estimate the value of perceived stimulus – inference in generative model. To do this the observer can calculate the probability distribution over stimulus - $p(s \mid \mathbf{r})$ - and then read out the best value of *s* from it's distribution. Using Bayes rule, posterior about the stimulus can be written as:

$$p(s \mid \mathbf{r}) = \frac{p(\mathbf{r} \mid s)\, p(s)}{p(\mathbf{r})} \quad (1.21)$$

Where $p(s \mid \mathbf{r})$ is the likelihood, $p(s)$ is the prior probability of the stimulus and $p(\mathbf{r})$ is the normalization factor which is:

$$p(\mathbf{r}) = \sum p(\mathbf{r} \mid s_i)\, p(s_i) \quad (1.22)$$

Equation (1.21) means that we can calculate the probability of a stimulus given the population activity - $\mathbf{r}$ - and the prior probability over the stimulus. This is shown figure 1.10:

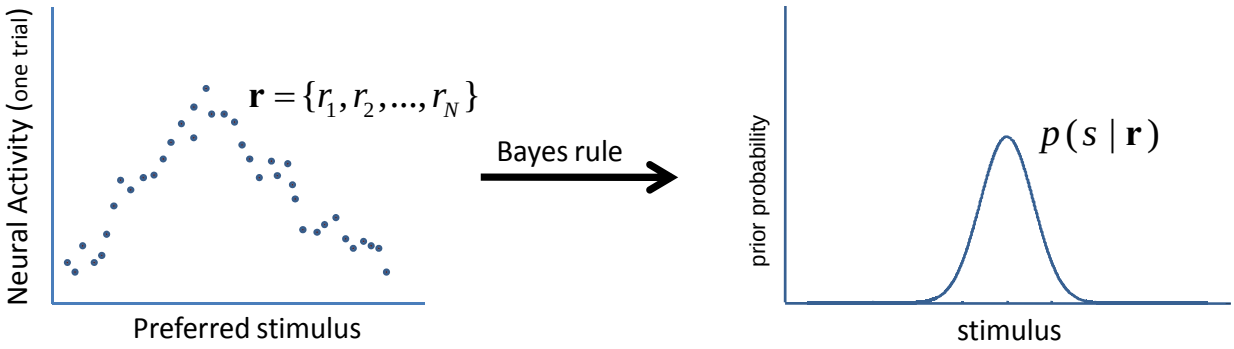


Figure 1.10 Posterior distribution given population activity.

From equations (1.19) and (1.21) we can write:

$$p(s \mid \mathbf{r}) \propto \prod_i \frac{e^{-f_i(s)} f_i^{r_i}(s)}{r_i!} p(s) \qquad (1.23)$$

Using a flat distribution for $p(s)$, it can be shown from equation (1.23) that the posterior distribution $p(s \mid \mathbf{r})$ converges to a Gaussian distribution with mean μ and variance σ^2. Using maximum aposteriori (MAP) to read out the best guess, the observer chooses the mean of the posterior as the stimulus. Then the activity of each neuron in the population is proportional to the distance between its preferred stimulus and the input stimulus. So the peak of the posterior - mean of the distribution - happens very close to the preferred stimulus of the neuron with highest activity in the population. It is shown in [12-14] that the variance of the posterior is proportional to the hill of the activity i.e. $g \propto \frac{1}{\sigma^2}$.

One of the most important reasons for decoding probability distributions instead of deterministic values of stimuli is the role of uncertainty in neural computations. In fact to use sensory information for both judgment and motor control, the brain needs to take into account the uncertainty in information it receives or how reliable are the received information.

1.4. Cue Combination using PPC

To show the usefulness of encoding the probability distributions, here we describe an example of multisensory integration using PPC. In cue combination, the brain integrates two cues such as auditory and visual cues, each of which contains information about the stimulus with some uncertainty. Figure11 shows this cue integration task.

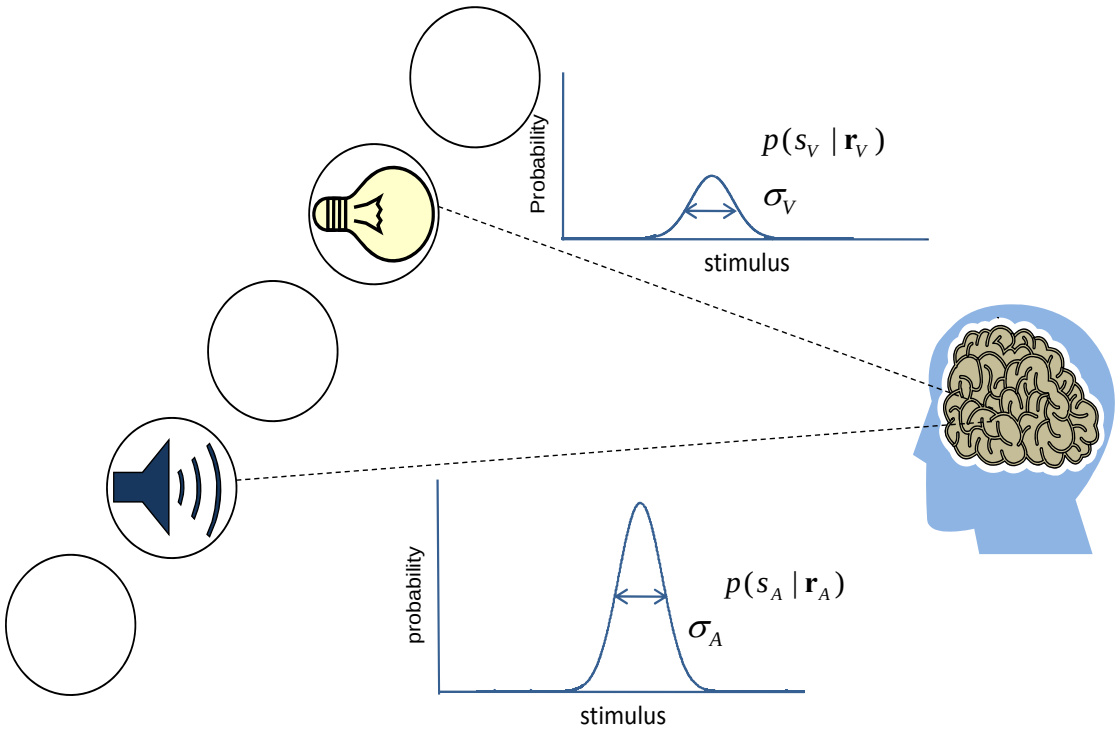


Figure 1.11 An example of cue integration.

The relation between cues the mean and variance of the cues and the inferred stimuli follows equation(1.7), which describes the Bayes-optimal solution. . As mentioned before, the human brain performs very close to equation(1.7). Now the question is how neurons can achieve this relationship. Figure 1.11 shows that two cues - c_1 and c_2 - are encoded in two population activities $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ with gains g_1 and g_2 respectively. This figure also shows the relation between $\boldsymbol{r}_3$ with $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$.

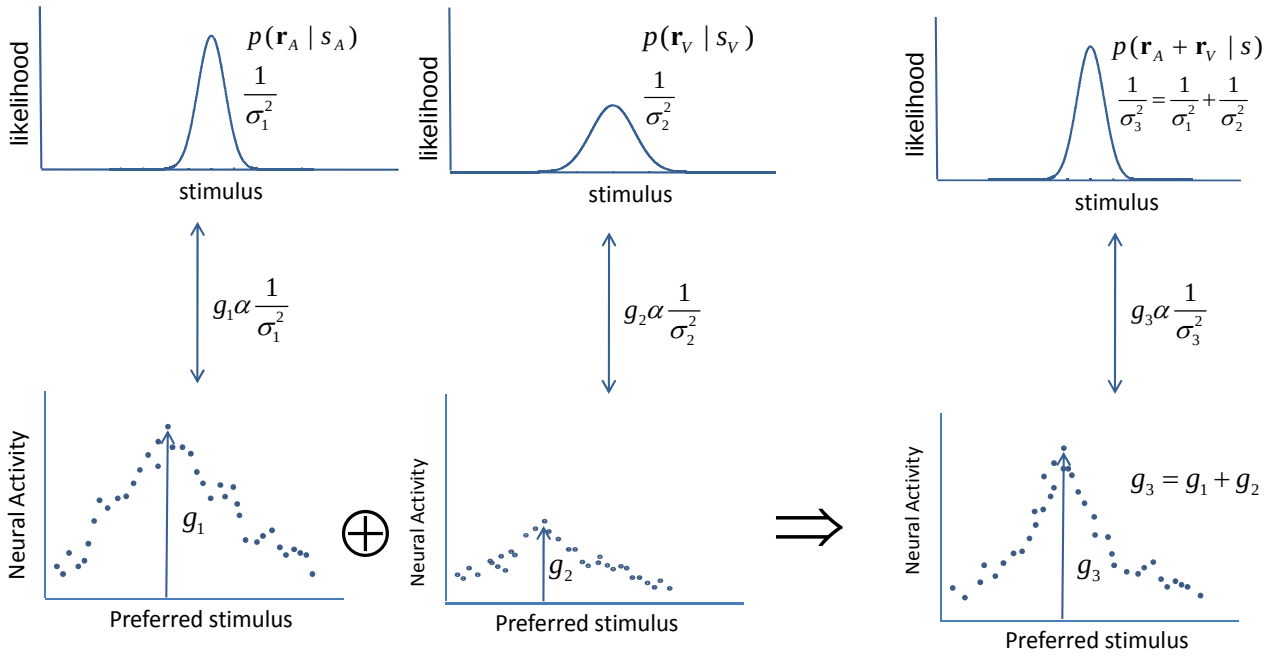


Figure 1.12 The relationship between population activity of two cues and a third population which can encode the distribution of the stimuli.

In fact these PPCs - $\mathbf{r}_1$ and $\mathbf{r}_2$ – encode likelihood functions $p(\mathbf{r}_1 \mid s)$ and $p(\mathbf{r}_2 \mid s)$. Using the same number of neurons in both populations and same tuning curves for corresponding neurons in two populations, it can be shown that optimal Bayesian inference is equivalent to summing up the activity of two populations $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ on a neuron by neuron basis. In other words, we can construct a new population $\boldsymbol{r}_3$ for which we have:

$$\mathbf{r}_3 = \mathbf{r}_1 + \mathbf{r}_2 \qquad (1.24)$$

If $\mathbf{r}_1$ and $\boldsymbol{r}_2$ have a Poisson distribution $\mathbf{r}_3$ will have too. As another result we can see that r_3 encodes the likelihood function with variance σ_3^2 which is inversely proportional to the gain of r_3. Figure 1.12 shows the relation between $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and r_3.

As can be seen in this figure, the gain of the final population, g_3, equals the sum of the first two populations - $g_3 = g_1 + g_2$. Since the gains are inversely proportional to the variance of the corresponding populations with the same proportionality, we conclude $\frac{1}{\sigma_3^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$ which is exactly equation (1.7). As a consequence, the variance of the distribution encoded by r_3 is the variance of the posterior probability $p(s \mid \mathbf{r}_1, \mathbf{r}_2)$.

To prove that summing up population activities results in Bayes-optimal inference we assume that neural variability follows the so-called *Poisson-like* distribution, defined as:

$$p(\mathbf{r}_k \mid s, g_k) = \Phi_k(\mathbf{r}_k, g_k)\exp(\mathbf{h}^T(s)\mathbf{r}_k) \qquad (1.25)$$

Then we want to prove that $p(\mathbf{r}_3 \mid s)$ - with $r_3 = r_1 + r_2$ - is a Poisson-like distribution. To do this we write:

$$\begin{aligned} p(\mathbf{r}_3 = \mathbf{r}_1 + \mathbf{r}_2 \mid s) &= \iint p(\mathbf{r}_1 \mid s)\, p(\mathbf{r}_2 \mid s)\delta(\mathbf{r}_3 - \mathbf{r}_1 + \mathbf{r}_2)d\mathbf{r}_1 d\mathbf{r}_2 \\ &= \int \Phi(\mathbf{r}_1, g_1)e^{\mathbf{h}^T(s).\mathbf{r}_1}\Phi(\mathbf{r}_3 - \mathbf{r}_1, g_2)e^{\mathbf{h}^T(s).(\mathbf{r}_3 - \mathbf{r}_1)}d\mathbf{r}_1 \\ &= e^{\mathbf{h}^T(s).\mathbf{r}_1}\int \Phi(\mathbf{r}_1, g_1)\Phi(\mathbf{r}_3 - \mathbf{r}_1, g_2)d\mathbf{r}_1 \\ &= \Phi(\mathbf{r}_3, g_3)e^{\mathbf{h}^T(s).\mathbf{r}_1} \end{aligned} \qquad (1.26)$$

With $\Phi(\mathbf{r}_3, g_3) = \int \Phi(\mathbf{r}_1, g_1)\Phi(\mathbf{r}_3 - \mathbf{r}_1, g_2)d\mathbf{r}_1$. So we can decode the information content of the population generated from summing up two former populations. The information encoded by the new population is a combination of the information from the two input populations.

In equation(1.25), *k* can be 1, 2 or 3 and **h**(s) is the kernel defined as:

$$\mathbf{h}'(s) = \Sigma_k^{-1}(s, g_k)\mathbf{f}'(s, g_k) \qquad (1.27)$$

Where Σ_k^{-1} is the covariance matrix of $\mathbf{r}_k$, and $\mathbf{f}_k^{'}$ is the derivative of the tuning curves. This is the general form. Then if the noise is independent Poisson and tuning curves for different gains and in two populations are identically shaped, then we have:

$$\mathbf{h}(s) = \log \mathbf{f}(s) \qquad (1.28)$$

$$\Phi_k(r_k, g_k) = \exp(-cg_k)\prod\nolimits_i \frac{\exp(r_{ki}\log g_k)}{r_{ki}!} \qquad (1.29)$$

Where *c* is a constant.

So far, we have argued that probabilistic population coding can be a very convenient way for neural populations to encode probability distributions over outside world parameters.

In this chapter we reviewed some literatures on causal inference and cue combination both in behavioral and neural level. We saw that using causal inference model, the observer can infer if there is a common cause for two cues as well as he can estimate the value of the stimulus. In next

chapter we will discuss neural implementation of causal inference model of inferring the number of cause using probabilistic population codes (PPC).

Chapter 2: Problem Statement

In the causal inference model presented in the previous chapter, the observer uses the noisy internal representations of stimuli to calculate the probability that those stimuli have a common cause. So far, we have chosen the internal representations to be rather abstract representations of the stimuli. In reality, a stimulus is internally represented through a pattern of firing rates in a population of neurons. The central problem in this work is to find a neural network that takes these patterns of activity as input and uses them to output the probability that the originating stimuli have a common cause.

2.1.Neural implementation of Causal Inference

In Section 1.2, we presented an analytical calculation of the probability of a common cause for a given pair of internal representations. Now we reformulate that calculation using neural activities instead of internal representations of stimuli. For concreteness, we will refer to auditory and visual stimuli, but the framework can be applied to any set of stimuli.

Information about auditory and visual stimuli, s_A and s_V , is encoded in auditory and visual neural populations $\mathbf{r_A}$ and $\mathbf{r}_V$. As mentioned in Section 1.3, this information is encoded as likelihoods $p(\mathbf{r_A} \mid s_A)$ and $p(\mathbf{r_V} \mid s_V)$ respectively. In the previous chapter, we calculated the probability of C=1 and C=2 given internal representations of s_A and s_V - x_A and x_V . However, we need a method to calculate which is the same as $P(C \mid \mathbf{r}_A, \mathbf{r}_V)$ except that here we know $\mathbf{r}_A$ and $\mathbf{r}_V$ instead of x_A and x_V .

We can rewrite(1.12), (1.13) and (1.15) for $\mathbf{r}_A$ and $\mathbf{r}_V$ instead of x_A and x_V so we have:

$$P(C=1 \mid \mathbf{r}_A, \mathbf{r}_V) = \frac{P(\mathbf{r}_A, \mathbf{r}_V \mid C=1)\, p_{common}}{P(\mathbf{r}_A, \mathbf{r}_V)} \qquad (2.1)$$

where

$$\begin{aligned} P(\mathbf{r}_A, \mathbf{r}_V \mid C=1) &= \int P(\mathbf{r}_A, \mathbf{r}_V \mid s) P(s) ds \\ &= \int P(\mathbf{r}_A \mid s) P(\mathbf{r}_V \mid s) P(s) ds \end{aligned} \quad (2.2)$$

$$\begin{aligned} P(\mathbf{r}_A, \mathbf{r}_V \mid C=2) &= \iint P(\mathbf{r}_A, \mathbf{r}_V \mid s_A, s_V) P(s_A, s_V) ds_A ds_V \\ &= \left(\int P(\mathbf{r}_A \mid s_A) p(s_A) ds_A \right) \left(\int P(\mathbf{r}_V \mid s_V) p(s_V) ds_V \right) \end{aligned} \quad (2.3)$$

Unlike the distribution of x_A and x_V over *s*, which is Gaussian, the distributions of $\mathbf{r}_A$ and $\mathbf{r}_V$ over *s* are Poisson-like, as given by (1.25). Then substituting in (1.13) gives:

$$P(\mathbf{r}_A, \mathbf{r}_V \mid C=1) = \Phi(\mathbf{r}_A, g_A) \Phi(\mathbf{r}_V, g_V) \int e^{\mathbf{h}^T(s)\mathbf{r}_A} e^{\mathbf{h}^T(s)\mathbf{r}_V} p(s) ds \quad (2.4)$$

and (1.15) turns into:

$$P(\mathbf{r}_A, \mathbf{r}_V \mid C=2) = \Phi(\mathbf{r}_A, g_A) \Phi(\mathbf{r}_V, g_V) \int e^{\mathbf{h}^T(s)\mathbf{r}_A} p(s) ds \int e^{\mathbf{h}^T(s)\mathbf{r}_V} p(s) ds \quad (2.5)$$

with $p(s)$ a Gaussian distribution as before.

In the case of independent Poisson variability and Gaussian tuning curves, we can write from (2.4) and (2.5): :

$$\begin{aligned} d = \log\left(\frac{P(C=1 \mid \mathbf{r}_A, \mathbf{r}_V)}{P(C=2 \mid \mathbf{r}_A, \mathbf{r}_V)} \right) &= \log\left(\frac{\Phi(\mathbf{r}_A, g_A) \Phi(\mathbf{r}_V, g_V) \int e^{\log(f(s)) \cdot \mathbf{r}_A} e^{\log(f(s)) \cdot \mathbf{r}_V} p(s) ds}{\Phi(\mathbf{r}_A, g_A) \Phi(\mathbf{r}_V, g_V) \int e^{\log(f(s)) \cdot \mathbf{r}_A} p(s) ds \int e^{\log(f(s)) \cdot \mathbf{r}_V} p(s) ds} \right) \\ &= \log\left(\frac{\int e^{\log(f(s)) \cdot \mathbf{r}_A} e^{\log(f(s)) \cdot \mathbf{r}_V} p(s) ds}{\int e^{\log(f(s)) \cdot \mathbf{r}_A} p(s_A) ds_A \int e^{\log(f(s)) \cdot \mathbf{r}_V} p(s_V) ds_V} \right) \end{aligned} \quad (2.6)$$

Equation (2.6) gives us the log-posterior ratio which we can use as a threshold for binary variable *C* to decide if *C*=1 or *C*=2.

Now we are looking for a population of neurons – called $\mathbf{z}$ - which encodes the posterior probability of the number of causes - $p(C \mid \mathbf{r}_A, \mathbf{r}_V)$. Having such neural population allows us to use the same encoding and decoding techniques for information about *C*. As mentioned before, we suppose that neural populations have Poisson-like variability. The same assumption is needed for $\mathbf{z}$ to be able to follow the same logic in inferring *C*.

$$p(\mathbf{z}\,|\,C) = \Phi(\mathbf{z})\, e^{\mathbf{h}^T(C)\cdot\mathbf{z}} \qquad (2.7)$$

This allows for recursive computation and facilitates decoding. Assuming $\mathbf{z}$ encodes probability of a common cause, we can define a variable *d* as below:

$$d = \log\frac{p(C=1\,|\,\mathbf{z})}{p(C=2\,|\,\mathbf{z})} = \log\left(\frac{p(\mathbf{z}\,|\,C=1)}{p(\mathbf{z}\,|\,C=2)}\right) \qquad (2.8)$$

Having $p(C\,|\,\mathbf{z})$, we need to readout the value of *C* from its distribution. If $d>0$ them $p(C=1\,|\,\mathbf{z}) > p(C=2\,|\,\mathbf{z})$ and if $d<0$,then $p(C=2\,|\,\mathbf{z}) > p(C=1\,|\,\mathbf{z})$ This is using MAP to readout the value of *s* from its distribution. From equation (2.8) we can write:

$$d = \log\left(\frac{\Phi(\mathbf{z})e^{\mathbf{h}(C=1)\cdot\mathbf{z}}}{\Phi(\mathbf{z})e^{\mathbf{h}(C=2)\cdot\mathbf{z}}}\right) = \log\left(e^{(\mathbf{h}(C=1)-\mathbf{h}(C=2))\cdot\mathbf{z}}\right) = \log\left(e^{\Delta\mathbf{h}\cdot\mathbf{z}}\right) = \Delta\mathbf{h}\cdot\mathbf{z} \qquad (2.9)$$

$$\Rightarrow \begin{cases} p(C=1\,|\,\mathbf{z}) = \dfrac{1}{1+e^{-\Delta\mathbf{h}.\mathbf{z}}} \\ p(C=2\,|\,\mathbf{z}) = 1 - p(C=1\,|\,\mathbf{z}) = \dfrac{1}{1+e^{\Delta\mathbf{h}.\mathbf{z}}} \end{cases} \qquad (2.10)$$

So here it is shown how we can calculate probability of common cause using the activity of one neural population.

To find a neural network that can perform near-optimal causal inference, we now need to determine the relation between the input population activities, $\mathbf{r}_A$ and $\mathbf{r}_V$, and the population which encodes the probability of a common cause, z. In other words, we need to find the neural operations $-\mathbf{R}$ which can be used in the network structure to implement the causal inference model. $\mathbf{R}$ is in fact a hidden layer in the network that determines which neurons from each population and in what form are combined to improve networks discrimination ability. Figure 2.1 shows such a structure.

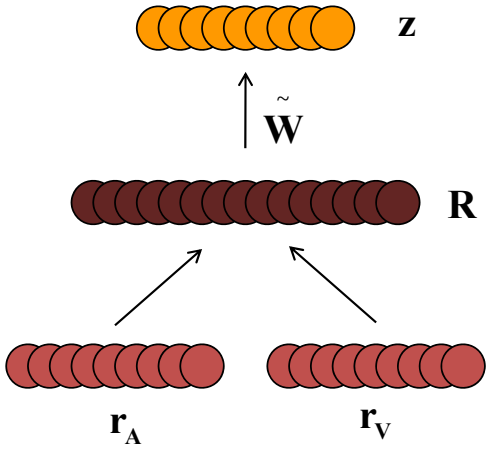


Figure 2.1 Neural network structure for implementing the causal inference model.

We can construct the neurons in the hypothetical **R** layer to perform products of activities of neurons in the input populations $\mathbf{r}_A$ and $\mathbf{r}_V$; this effectively computes a kind of correlation between neurons. For example, if there is no correlation between the neurons in the two input populations, then the layer **R** consists of neurons of two populations in a row beside each other. In another case, if we suppose that there are some correlations between each neuron with its corresponding neuron in the other population, we can make layer **R** consists of nodes which are the product of each two corresponding neuron from two populations. Figure 2.2 shows these two examples.

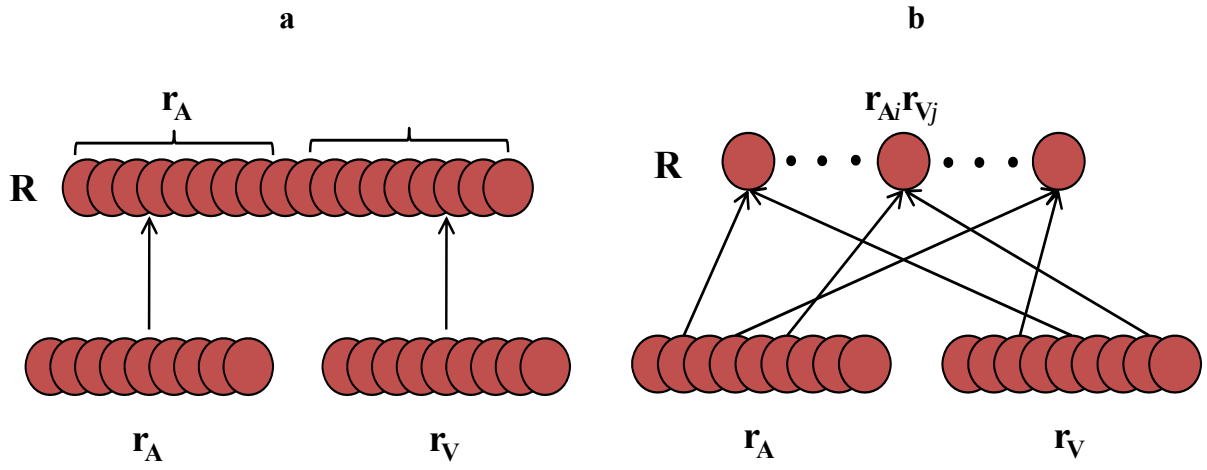


Figure 2.2 Two examples of possible structures of R.

The last step in completing the network is to find a set if weights which allow the network calculating the same probability as equation(2.6). In other words, we need a set of weights such that

$$\mathbf{z} = \tilde{\mathbf{W}} \cdot \mathbf{R} \qquad (2.11)$$

One important point here is that, as can be seen in equation (2.11), $\tilde{\mathbf{W}}$ is a matrix which maps $\mathbf{R}$ to $\mathbf{z}$. Substituting this $\mathbf{z}$ into equation (2.10) we have:

$$\Rightarrow \begin{cases} p(C=1|\mathbf{z}) = \dfrac{1}{1+e^{-\Delta\mathbf{h}\cdot(\tilde{\mathbf{W}}\cdot\mathbf{R})}} \\ p(C=2|\mathbf{z}) = 1 - p(C=1|\mathbf{z}) = \dfrac{1}{1+e^{\Delta\mathbf{h}\cdot(\tilde{\mathbf{W}}\cdot\mathbf{R})}} \end{cases} \qquad (2.12)$$

As will be shown in next chapter, because we do not know $\Delta\mathbf{h}$, what we try to find is $\mathbf{W} = \Delta\mathbf{h}\cdot\tilde{\mathbf{W}}$ so the final result is:

$$\Rightarrow \begin{cases} p(C=1|\mathbf{z}) = \dfrac{1}{1+e^{-\mathbf{W}\cdot\mathbf{R}}} \\ p(C=2|\mathbf{z}) = 1 - p(C=1|\mathbf{z}) = \dfrac{1}{1+e^{\mathbf{W}\cdot\mathbf{R}}} \end{cases} \qquad (2.13)$$

To calculate the probability of a common cause from equation(2.13), we need to find the $\mathbf{W}$ using which gives us a result as close as possible to the probability calculated from equation 15. But before that, we need to determine the elements of $\mathbf{R}$ layer. The simplest case is to use $\mathbf{R} = [\mathbf{r}_A \ \ \mathbf{r}_V]$ which contains no correlation between different neurons.

We can consider the causal inference task over many trials as a classification process. Data points are patterns of neural activity ($\mathbf{r}_A$, $\mathbf{r}_V$) and they have to be classified as originating from same source (C=1) or different sources (C=2). Figure 2.3 shows the scatter plot of the activity of two corresponding neurons in the auditory and visual populations for both *C*=1 and *C*=2 cases. In reality, the data points are not one-dimensional, but each data point is has *N* dimensions, where *N* is the number of neuron in the population. So each point is a population activity in an *n* dimensional space.

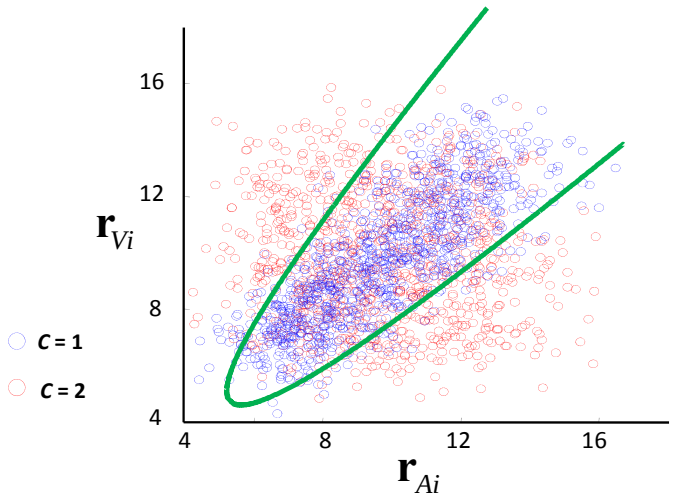


Figure 2.3 Demonstration of discrimination boundary.

In the simplest case we can use $\mathbf{R}_{Lin} = [\mathbf{r}_A \ \ \mathbf{r}_V]$ which corresponds to a linear boundary. But as can be seen in figure 2.3, a linear boundary cannot discriminate the data points well. One of the simplest boundaries that might do a better job is a quadratic boundary, shown as a green curve in figure 2.3. To have such a boundary we need quadratic elements in $\mathbf{R}$. So we set $\mathbf{R}$ as $\mathbf{R}_{Quad} = [\mathbf{r}_A \ \ \mathbf{r}_V \ \ \mathbf{r}_A\mathbf{r}_A \ \ \mathbf{r}_V\mathbf{r}_V \ \ \mathbf{r}_A\mathbf{r}_V]$. This $\mathbf{R}$ contains quadratic elements, $\mathbf{r}_{Ai}\mathbf{r}_{Aj}$, $\mathbf{r}_{Vi}\mathbf{r}_{Vj}$ and $\mathbf{r}_{Ai}\mathbf{r}_{Vj}$ which actually represent correlation between not only different neurons in the same population, but also neurons in different populations.

Calculating d in equation (2.9), it can be shown that in analytical formula of d , $\mathbf{R}$ contains some non-linear function of $\mathbf{r}_A$ and $\mathbf{r}_V$ as well as terms containing $\mathbf{r}_A$ and $\mathbf{r}_V$ in the denominator. This denominator term containing $\mathbf{r}_V$ and $\mathbf{r}_V$ is called divisive normalization. Then $\mathbf{R}$ will be $\mathbf{R}_{LDN} = [\mathbf{R}_{Lin} \ \ \frac{\mathbf{R}_{Lin}}{\mathbf{a.r}}]$ as linear with divisive normalization and $\mathbf{R}_{QDN} = \left[\mathbf{R}_{Quad} \ \ \frac{\mathbf{R}_{Quad}}{\mathbf{a.r}}\right]$ as quadratic with divisive normalization. $\boldsymbol{a}$ is a constant vector which should be learned through an iterative algorithm. Considering $\mathbf{a}$ in our calculations, formula (2.13) can be rewritten as:

$$\begin{cases} p(C=1|\mathbf{z}) = \dfrac{1}{1+e^{-\frac{\mathbf{W}\cdot\mathbf{R}}{\mathbf{a}\cdot\mathbf{r}}}} \\ p(C=2|\mathbf{z}) = 1 - p(C=1|\mathbf{z}) = \dfrac{1}{1+e^{\frac{\mathbf{W}\cdot\mathbf{R}}{\mathbf{a}\cdot\mathbf{r}}}} \end{cases} \qquad (2.14)$$

Therefore we tested four networks, characterized by the following operations:

$\mathbf{R}_{Lin} = [1 \ \mathbf{r}_{\mathbf{A}} \ \mathbf{r}_{\mathbf{V}}]$	Linear (Lin)
$\mathbf{R}_{LDN} = [\mathbf{R}_{Lin} \ \frac{\mathbf{R}_{Lin}}{\mathbf{a.r}}]$	Linear with divisive normalization (LDN)
$\mathbf{R}_{Quad} = [1 \ \mathbf{r}_{\mathbf{A}} \ \mathbf{r}_{\mathbf{V}} \ \mathbf{r}_{\mathbf{A}}\mathbf{r}_{\mathbf{A}} \ \mathbf{r}_{\mathbf{V}}\mathbf{r}_{\mathbf{V}} \ \mathbf{r}_{\mathbf{A}}\mathbf{r}_{\mathbf{V}}]$	Quadratic (Quad)
$\mathbf{R}_{QDN} = [\mathbf{R}_{Quad} \ \frac{\mathbf{R}_{Quad}}{\mathbf{a.r}}]$	Quadratic with divisive normalization (QDN)

We also added element 1 to $\mathbf{R}$ in all networks as the bias. As can be seen in Eq. (2.13), this bias lets network to shift the value of $\mathbf{W} \cdot \mathbf{R}$ independent of the input. This in fact, helps network to learn any bias in the input data.

So far we have suggested a neural network structure that can implement causal inference. For linear and quadratic networks, $\mathbf{R}$ is calculated from $\mathbf{r}_A$ and $\mathbf{r}_V$. Then we need a set of weights – $\mathbf{W}$ – to calculate $p(C \mid \mathbf{z})$. For the LDN and QDN networks, we also need a vector $\boldsymbol{a}$ to construct $\boldsymbol{R}$. S. Therefore, for LDN and QDN networks, there are two set of weights – $\boldsymbol{W}$ and $\boldsymbol{a}$ - to be learnt. In the next chapter, we will discuss learning algorithms for the network and present simulation results of the networks we trained.

Chapter 3: Methods

In the previous chapter, we proposed a neural network structure for implementing the causal inference model. The goal of the network is to calculate $p(C \mid \mathbf{z})$. In this chapter, we explain two learning algorithms we used to train the network.

3.1.Stochastic Gradient Descent

In chapter 2 we presented two ways to calculate the probability distribution over C, $p(C \mid \mathbf{z})$. One from the analytical optimal method, equations (2.1) to (2.5), and one using neural network, equation (2.13). The second method uses a set of weights – **W** – for linear(Lin) and quadratic (Quad) networks and a set of weights plus divisive normalization factor – **W** and **a** respectively – for linear with divisive normalization (LDN) and quadratic with divisive normalization (QDN) networks. Then the goal is to find **W** and **a** in a way that two probability distributions calculated from analytical and network methods, have the smallest possible distance from each other. For this purpose we need a method to calculate the distance between two probability distributions. One of the most common measures of the distance between two probability distributions is the Kullback-Leibler divergence[15]. If $p(x)$ and $q(x)$ are two distributions over discrete random variable X, then the KL divergence (D_{KL}) is defined as:

$$D_{KL}(p \| q) = \sum_i p(x_i) \log\left(\frac{p(x_i)}{q(x_i)}\right) \qquad (3.1)$$

In the case of a continuous variable, the sum turns into an integral.

In our problem, we can substitute the distribution calculated from the analytical method for p and distribution calculated from neural network for q. Therefore $D_{KL}(p \| q)$ shows the distance between these two distributions which is:

$$D_{KL}(p \| q) = \sum_{C,\mathbf{R}} p(C \mid \mathbf{R}) \log\left(\frac{p(C \mid \mathbf{R})}{q(C \mid \mathbf{R})}\right) \qquad (3.2)$$

With

$$q(C \mid \mathbf{R}) = \frac{1}{1+e^{-C\mathbf{W}.\mathbf{R}}} \qquad (3.3)$$

From equation(3.2), it can be seen that $D_{KL}(p \,\|\, q)$ is a function of **W** (and **a** for networks with divisive normalization). Then we have to find **W** – for networks without divisive normalization – and **W** plus ***a*** – for networks with divisive normalization which minimize $D_{KL}(p \,\|\, q)$. To find the minimum value of $D_{KL}(p \,\|\, q)$ as a function of **W**, we use stochastic gradient descent algorithm (SGDS). This algorithm finds the minimum of a function iteratively after choosing an initial point. In each trial the algorithm finds the gradient of $D_{KL}(p \,\|\, q)$ at that point and moves in the opposite direction of the gradient until gets to its minimum. This can be formulated as:

$$\mathbf{W}^{n+1} = \mathbf{W}^{n} + \alpha \Delta \mathbf{W} \qquad (3.4)$$

Where $\mathbf{W}^{n}$ and $\mathbf{W}^{n+1}$ are the weight vectors in nth and n+1th trial respectively. α is defined as the learning rate and $\Delta\mathbf{W}$ as below:

$$\Delta \mathbf{W} = -\frac{\partial D_{KL}}{\partial \mathbf{W}} \qquad (3.5)$$

Then from equation (3.2) and (3.3), we can write:

$$\frac{\partial D_{KL}(p \,\|\, q)}{\partial \mathbf{W}} = -\sum_{C,\mathbf{R}} p(C \mid \mathbf{R})[C\mathbf{R}(1 - q(C \mid \mathbf{R}, \mathbf{W}))] \qquad (3.6)$$

Equation (3.6) calculates $\Delta\mathbf{W}$ for one trial in each iteration. We choose **R** from the generative model in figure 1.3 stochastically – and that's why it's called stochastic gradient descent method. Therefore to find a $\Delta\mathbf{W}$ and finally **W** independent of a specific **R,** we use the average of $\frac{\partial D_{KL}(p \,\|\, q)}{\partial \mathbf{W}}$ over many trials. So equations (3.6) can be rewritten as:

$$\begin{aligned} \frac{\partial D_{KL}(p \,\|\, q)}{\partial \mathbf{W}} &= -\sum_{C,\mathbf{R}} p(C \mid \mathbf{R})\, p(\mathbf{R})[C\mathbf{R}(1 - q(C \mid \mathbf{R}, \mathbf{W}))] \\ &= -\sum_{C,\mathbf{R}} p(C, \mathbf{R})[C\mathbf{R}(1 - q(C \mid \mathbf{R}, \mathbf{W}))] \end{aligned} \qquad (3.7)$$

Equation (3.7) calculates the weighted sum of the function $[C\mathbf{R}(1-q(C\mid\mathbf{R},\mathbf{W}))]$ as its average over C and $\mathbf{R}$. Instead of averaging in this way, we can use sampling. In this case we sample C_i and $\mathbf{R}_i$ from $p(C,\mathbf{R})$ according to the generative model. Therefore we can write the following equation instead of equation (3.7):

$$\frac{\partial D_{KL}(p\,\|\,q)}{\partial \mathbf{W}} = -\sum_i [C_i \mathbf{R}_i (1-q(C_i \mid \mathbf{R}_i, \mathbf{W}))] \tag{3.8}$$

From equations (3.4), (3.5) and (3.8) we can write:

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \alpha \sum_i [C_i \mathbf{R}_i (1-q(C_i \mid \mathbf{R}_i, \mathbf{W}))] \tag{3.9}$$

Equation (3.8) calculates $\Delta\mathbf{W}$ for networks both with and without divisive normalization. In Networks with divisive normalization we also need to find the best $\boldsymbol{a}$ which minimizes $D_{KL}(p\,\|\,q)$. Therefore we again use stochastic gradient descent algorithm to minimize $D_{KL}(p\,\|\,q)$ regarding $\boldsymbol{a}$. Then we have:

$$\frac{\partial D_{KL}(p\,\|\,q)}{\partial \mathbf{a}} = -\sum C(1-q(C\mid\mathbf{r},\mathbf{W},\mathbf{a}))\left(\frac{-\mathbf{r}\mathbf{W}_2\cdot\mathbf{r}_{comb}}{(\mathbf{a}\cdot\mathbf{r})^2}\right) \tag{3.10}$$

$\mathbf{W}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{r}$ and $\mathbf{r}_{comb}$ are defined as below:

$\mathbf{r} = [1 \;\; \mathbf{r}_A \;\; \mathbf{r}_V]$

$\mathbf{r}_{comb} = \mathbf{r}$ for LDN

$\mathbf{r}_{comb} = [1 \;\; \mathbf{r}_A \;\; \mathbf{r}_V \;\; \mathbf{r}_A\mathbf{r}_A \;\; \mathbf{r}_V\mathbf{r}_V \;\; \mathbf{r}_A\mathbf{r}_V]$ for QDN

$\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2)$ where $\mathbf{W}_1$ corresponds to elements of $\mathbf{R}$ without divisive normalization and $\mathbf{W}_2$ corresponds to elements of $\mathbf{R}$ with divisive normalization.

One important factor is the time when we stop updating $\mathbf{W}$ and $\mathbf{a}$. We continue updating $\mathbf{W}$ and $\mathbf{a}$ until a convergence criterion is satisfied. This criterion can be a threshold for $\Delta\mathbf{W}$, a threshold for information loss rate – which is defined in testing criteria- or any measure for convergence. In our simulations, we used information loss rate as convergence measure.

3.2. VBLR

The second learning algorithm we used is Variational Bayesian Logistic Regression (VBLR). VBLR tries to find a simpler approximation for the sigmoid function (3.3). To have the best approximation, it should find the **W** which maximizes a lower bound for the sigmoid function. In this algorithm some latent parameters are defined and then the approximation is optimized regarding those latent parameters as well as **W**. In our simulations we used VBLR code written by Jan Drugowitsch [16]

One important point about using both SGDS and VBLR algorithm is that in LDN and QDN networks we have two set of weights to be learnt, **W** and **a**. To do this we in each iteration, we first update **W** and then use this **W** to do the calculations for updating **a.** This idea comes from the Expected Maximization (EM) algorithm in which in *E* step the expectation of the log-likelihood of the model parameters is calculated and in *M* step we find the values of some hidden parameters which maximize the calculated expectation.

3.3. Test criteria

After finding optimized **W** and **a**, we calculate the information loss rate to measure the performance of the network. The information loss rate is defined as the normalized KL divergence to the mutual information between **R** and *C*:

$$\frac{\delta I}{I} = \frac{D_{KL}\left(p(C_i \mid \mathbf{R}_i), q(C_i \mid \mathbf{R}_i, \mathbf{W})\right)}{I(C_i, \mathbf{R})} = \frac{\left\langle \sum_{C_i} \log p(C_i \mid \mathbf{R}_i) - \log q(C_i \mid \mathbf{R}_i, \mathbf{W}) \right\rangle_{\mathbf{R}_i}}{\left\langle \sum_{C_i} \log p(C_i \mid \mathbf{R}_i) - \log p(C_i) \right\rangle_{\mathbf{R}_i}} \qquad (3.11)$$

In (3.11) to calculate the information loss rate, we approximate $D_{KL}(p \| q)$ and $I(C_i, \mathbf{R})$ by averaging over different trials. It can be shown that the larger the number of trials we use, the better the approximation is.

In real world, different stimuli have different reliabilities or gains. So the network should be able to discriminate C=1 and C=2 trials for different gains. Therefore in both training and testing we used different gains for different trials.

To check the performance of the network we also create the scatter plot of $p(C=1|\mathbf{R})$ and $q(C=1|\mathbf{R},\mathbf{W})$ over many trials.

Chapter 4: Simulation Results

As mentioned in chapter 2, we can categorize our networks based on their neural operations. This gives us four categories – Lin, Quad, LDN and QDN. Here we explain different training and testing parameters. We also present the performance of each of the networks with different training and testing parameters.

4.1. Training

There are three important training parameters. The first parameter is the learning rate - α in (3.4). We can use the same learning rate for all elements of **W** and **a** or different rates for different elements. For example for LDN and QDN, we can use two learning rates for $\mathbf{W}_1$ and $\mathbf{W}_2$ in $\mathbf{W} = [\mathbf{W}_1 \ \mathbf{W}_2]$ where $\mathbf{W}_1$ corresponds to elements of **R** without divisive normalization and $\mathbf{W}_2$ corresponds to elements of **R** with divisive normalization. Suitable values of learning rates are obtained by trial and error.

The second training parameter is the initial value of **W**- and in addition, of **a** for the LDN and QDN networks. We set the initial value of **W** as the zero vector - $\mathbf{W}_0 = \mathbf{0}$. This makes sense because we expect that without any input information the network results in equal probabilities for *C*=1 and *C*=2 - $p(C=1|\mathbf{R},\mathbf{W}_0) = p(C=2|\mathbf{R},\mathbf{W}_0) = \frac{1}{2}$. This can be achieved by using $\mathbf{W} = \mathbf{0}$ in (3.3).

The third parameter is the number of trials in each iteration. As mentioned in chapter 3 both learning algorithms use stochastic data generated from the generative model of the task. Therefore, the number of trials should be large enough to achieve a reasonable approximation.

In the learning procedure, we also need a criterion for network convergence. We can say the network is trained when the square root of $\sum(\Delta\mathbf{W}_i)^2$ is smaller than a threshold. This means that we are as close as we want as to the minimum of $D_{KL}(p\,||\,q)$. This is a somewhat arbitrary criterion. The more principled convergence criterion is the rate of change in information loss defined by (3.11). We used information loss as a convergence criterion because it is already one of the performance criteria of the network.

Regarding gain or reliability, we first trained and tested networks with fixed-equal visual and auditory gains. This means that all trials in all iterations have the same auditory and visual gains. Then we trained networks for multiple gains or different reliabilities. Using multiple-unequal

visual and auditory gains is much closer to real circumstances. However, it is easier for the network to learn the weights for fixed-equal gains because in this case, there is less variance in training data which is equivalent to the data being more classifiable. Therefore, we can find the proper order of the parameters we need for the more complicated case.

4.2. Test

We have two main criteria to measure the network's performance. The first one is the information loss defined by(3.11). The less the information loss is, the better the network performance. The second one is the scatter plot of $p(C=1|\mathbf{R})$, the probability calculated using analytical method, and $q(C=1|\mathbf{R},\mathbf{W})$, the probability calculated using network weights, over many trials. In a scatter plot, the performance of the network is better when the points lie closer to the diagonal. A sample of the scatter plot with error bars, which show the standard deviation of data, is shown in figure 4.1.

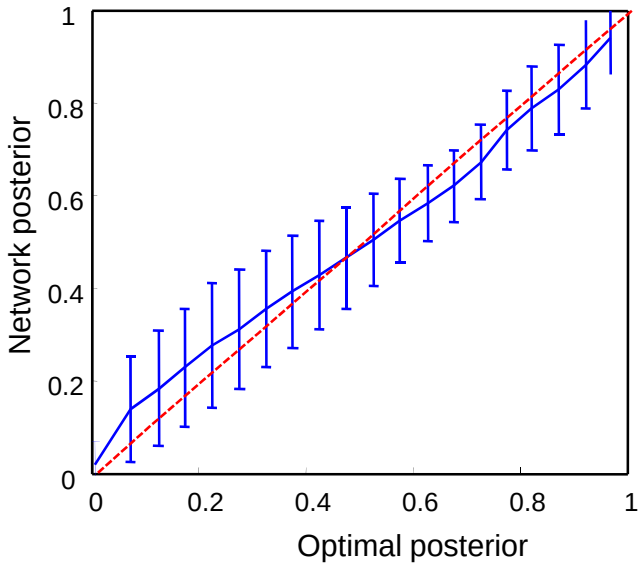


Figure 4.1 Scatter plot of $p(C=1|\mathbf{R})$ and $p(C=1|\mathbf{R},\mathbf{W})$.

4.3. Results

First of all we present the information loss of all four networks for both fixed and multiple gains. Then we present the scatter plot and information loss diagram of all four networks for both sixed and multiple gains.

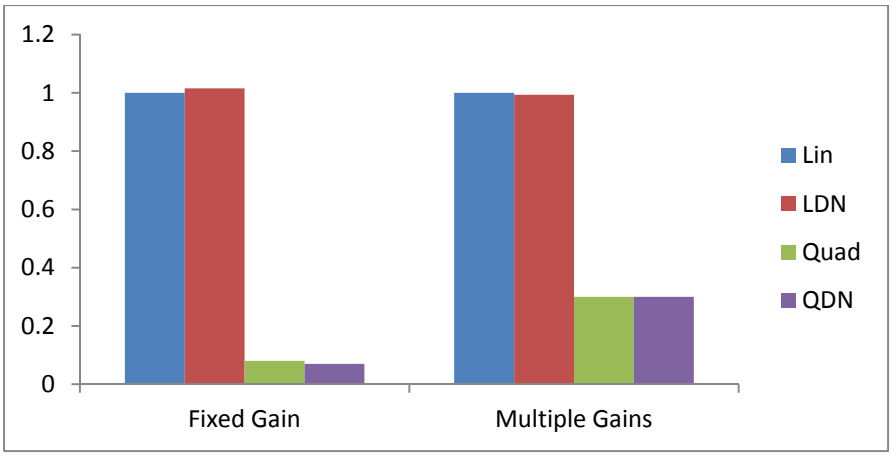


Figure 4.2 The information loss of all networks for both fixed and multiple gains.

As can be seen in figure 4.2, Lin and LDN networks have a large information loss. This was predicted in view of the discrimination boundary in figure.2.3.

4.3.1. Linear and LDN

In figures 4.3-a and 4.3-b it can be seen that linear network performs completely at chance for both fixed gain and multiple gains. The probability of C=1 estimated by the linear network is close to 0.5 for any input pattern of activity. This shows that the network performance is very poor. This is the result of using a too limited set of operations for this network.

Figures 4.3-c and 4.3-d show that the LDN network does not do well in discriminating C=1 from C=2. The scatter plot is still completely off-diagonal and like LIN network very close to 0.5.

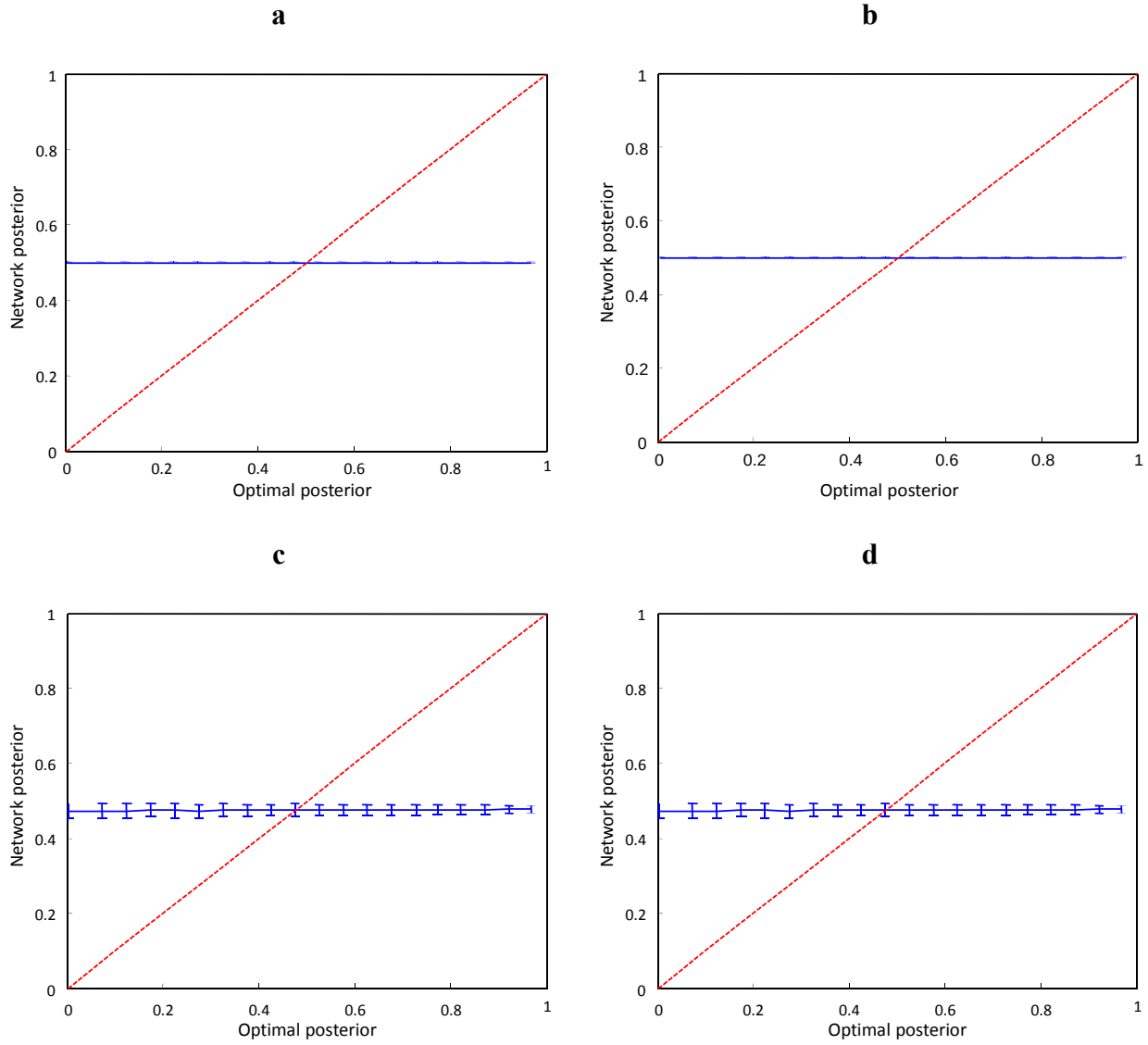


Figure 13 scatter plot of network and analytical results for P(C=1) (a) linear network with fixed-equal gains, (b) linear network with multiple-unequal gains (c) LDN network with fixed-equal gains (d) LDN network multiple-unequal gains.

4.3.2. Quadratic

Quadratic with fixed gain

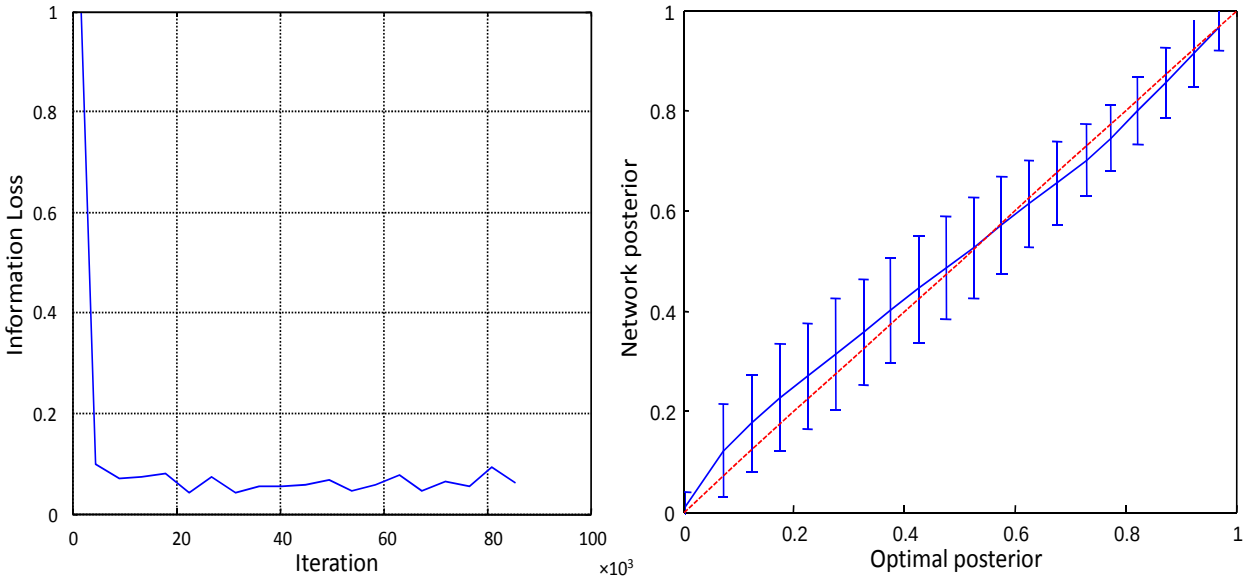


Figure 4.4 Information loss diagram and scatter plot of Quadratic SGD (Stochastic Gradient Descent) network and analytical results for P(C=1) with fixed train gain and fixed test gain.

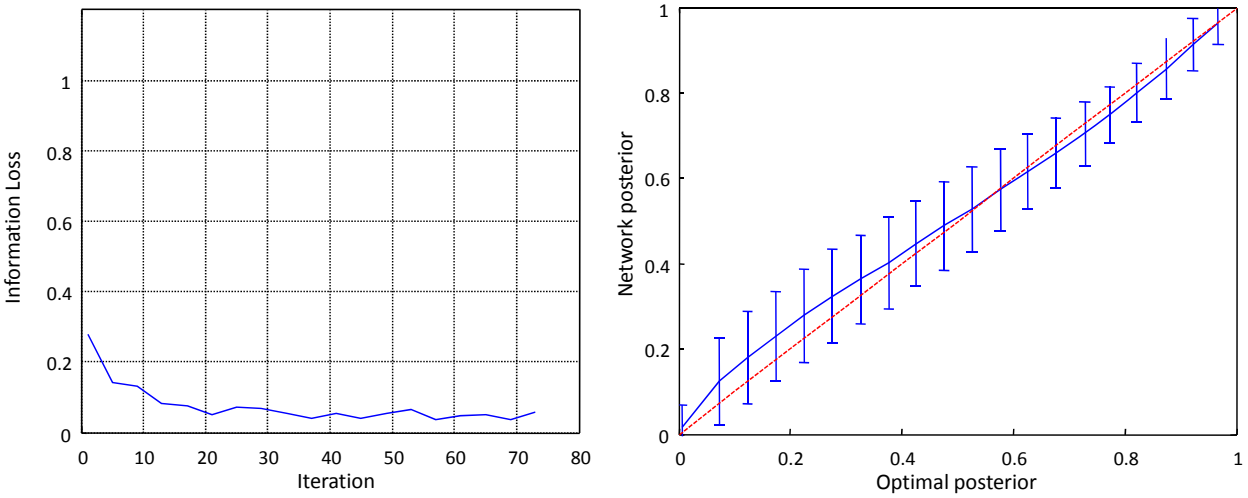


Figure 4.5 Information loss diagram and scatter plot of Quadratic VBLR network and analytical results for P(C=1) with fixed train gain and fixed test gain.

Figures 4.4 and 4.5 show the $P(C=1)$ calculated with a Quad network. The network in figure 4.4 has been trained using the Stochastic Gradient Descent (SGD) algorithm and the network in figure 4.5 has been trained with the Variational Bayesian Logistic Regression (VBLR) algorithm. All training trials had fixed equal visual and auditory gains. This is not compatible with real con-

ditions where the input gains can be different and unequal. Equal fixed auditory and visual gains result in less variance in training data, so the network can learn the weights better and faster. As can be seen in these figures, the scatter plot is almost diagonal but the error bars are large in comparison with the mean. The results of VBLR network are almost identical to those of the SGD network. This shows that the large error bars are not related to the learning algorithm. The information loss is under 10%. But to have an acceptable network, we need information loss smaller than 2%. This is why the quadratic network is not suitable for this task.

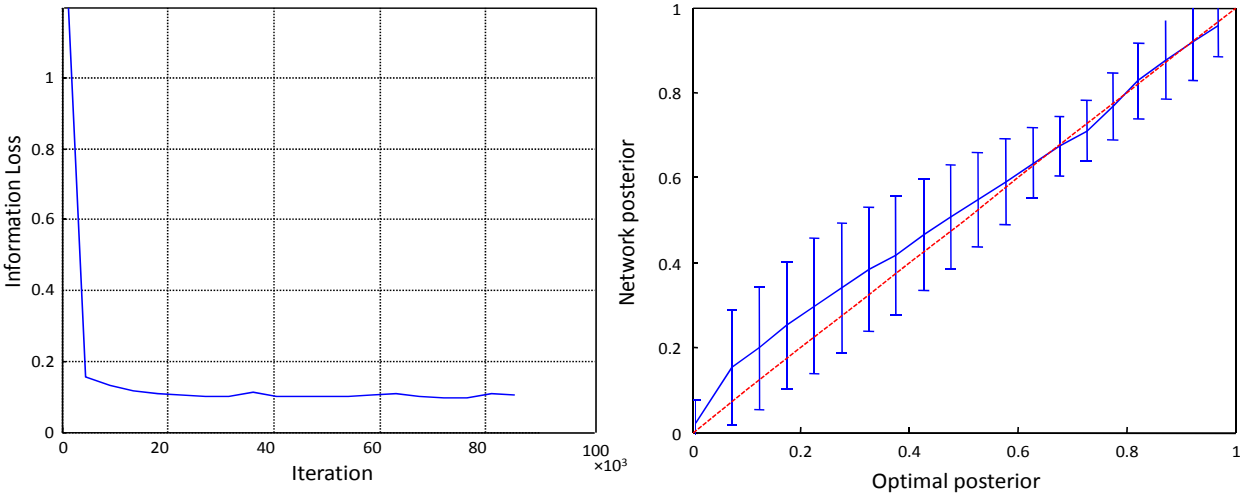


Figure 4.6 Information loss diagram and scatter plot of Quadratic SGD network and analytical results for P(C=1) with fixed training gain and multiple test gains.

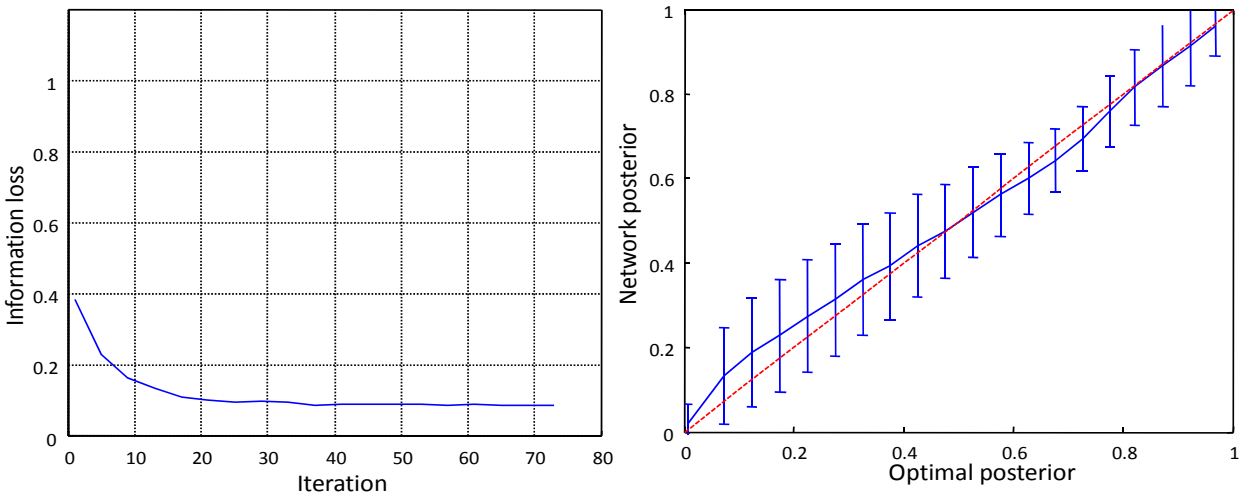


Figure 4.7 Information loss diagram and scatter plot of Quadratic VBLR network and analytical results for P(C=1) with fixed train gain and multiple test gains.

In addition to testing the Quad networks trained with equal and fixed gain with same gain, we tested these networks-SGD and VBLR- with multiple and unequal auditory and visual gains. As

can be seen in figures 4.6 and 4.7 the network trained with data with fixed and equal gains perform also well for test data with multiple and unequal gains. A test over all possible combination of auditory and visual gains is done. Figures 4.8 to 4.11 show the scatter plot and information loss of these combinations for SGD and VBLR networks respectively.

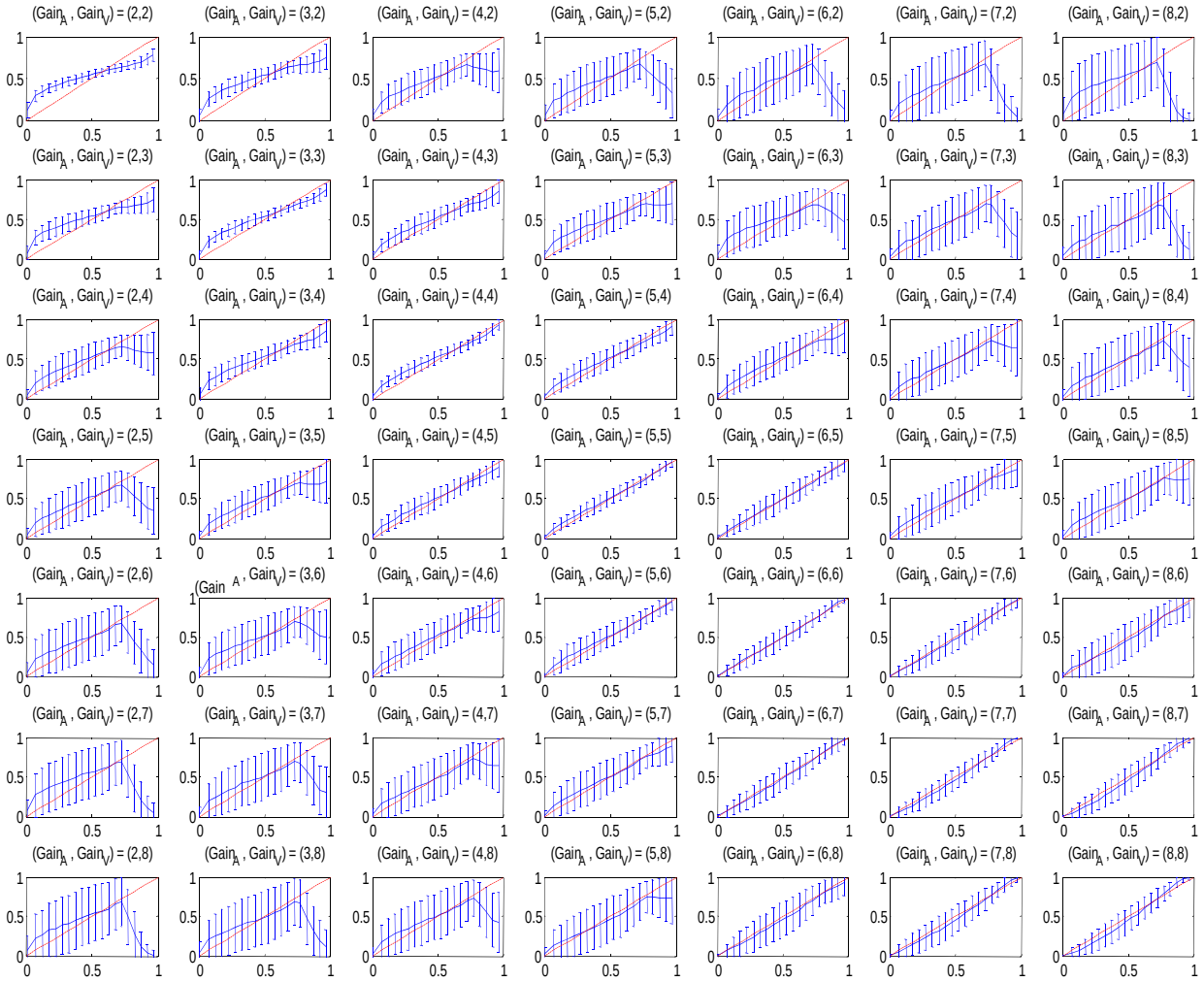


Figure 4.8 Scatter plot of different combinations of auditory and visual gains for a Quad network trained with SGD algorithm using fixed and equal auditory and visual gains.

It can be seen in figure 4.8 and 4.9 for SGD and VBLR networks that when the gains are equal, the network performance is much better. As well as the difference between gains gets larger, both deviation and error bars increase. We couldn't find an intuitive explanation for the part where the line start folding down when the difference between gains increase. This shows that this network can't discriminate C=1 and C=2 properly, when auditory and visual stimuli have different gains. It is also shown that the scatter plot has larger deviation for cases with smaller gain.

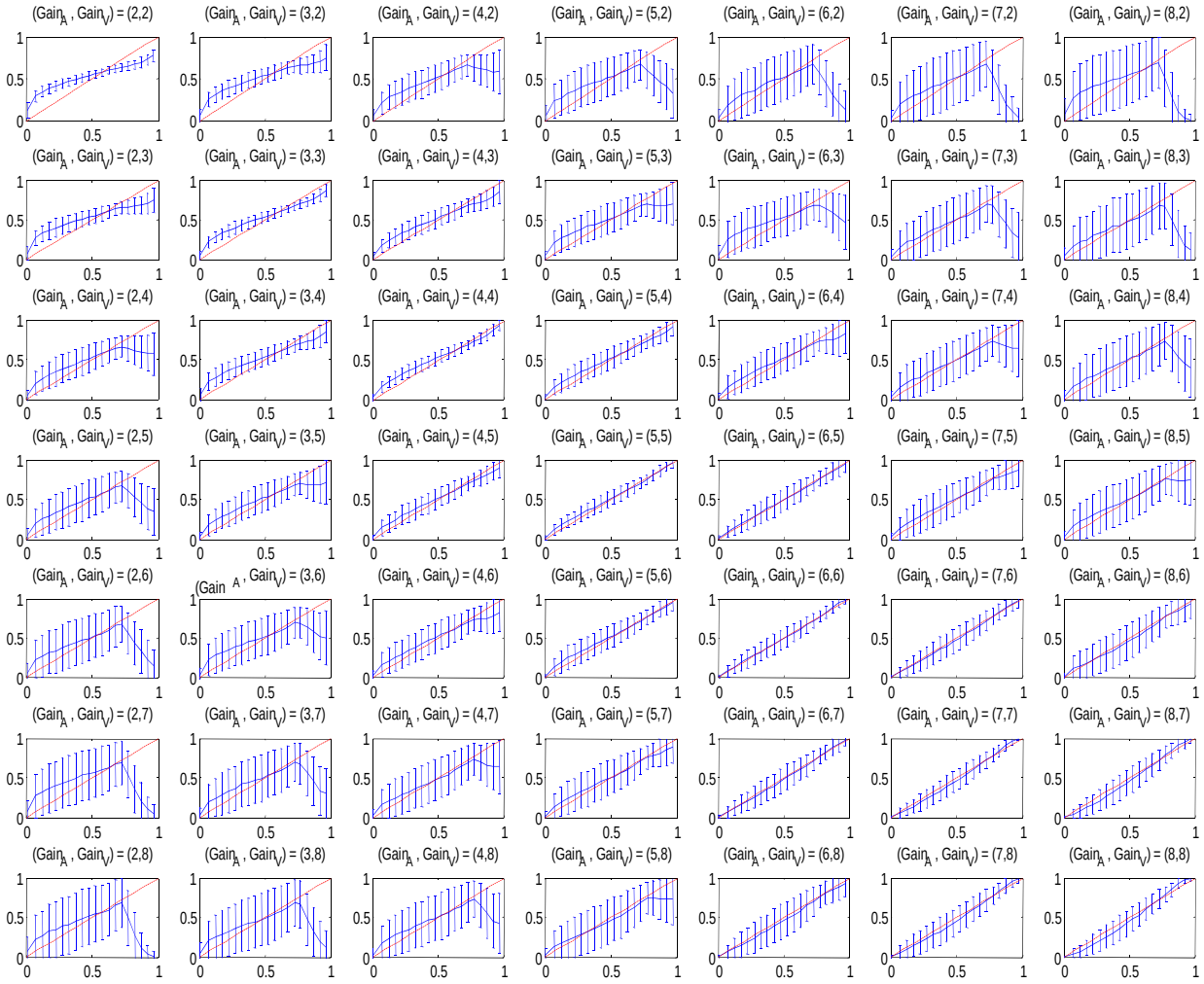


Figure 4.9 Scatter plot of different combinations of auditory and visual gains for a Quad network trained with VBLR algorithm using fixed and equal auditory and visual gains.

The information loss of these combinations is shown in figures 4.10 for SGD and VBLR networks respectively. Here also we can see that there is less information loss for combinations with smaller difference between auditory and visual gains.

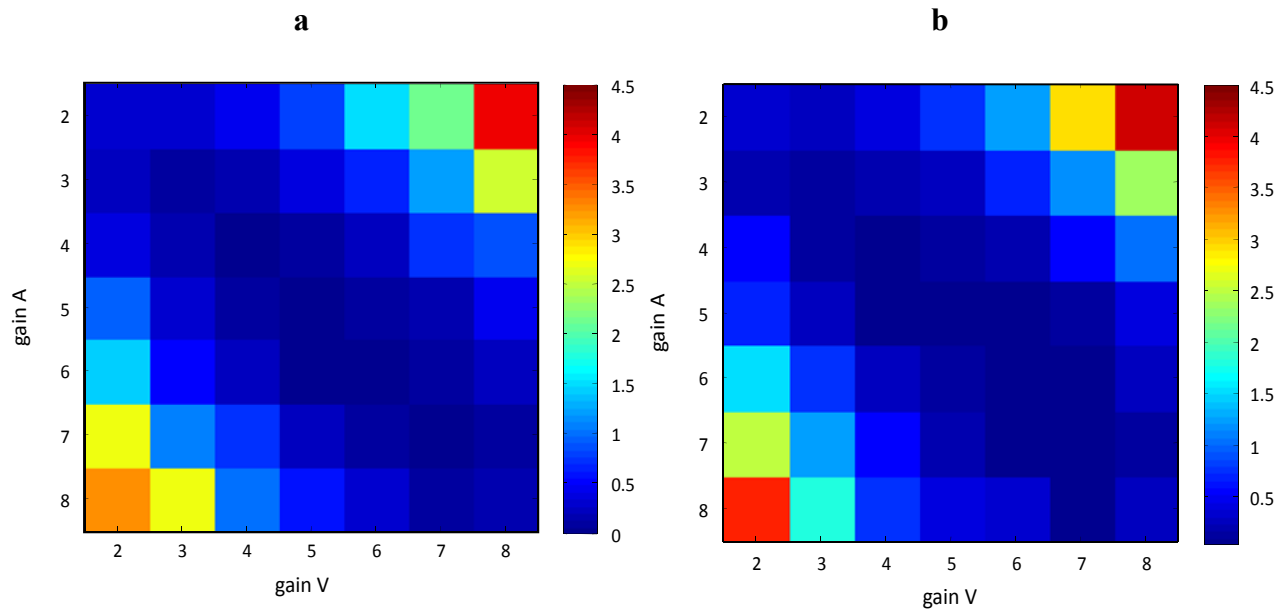


Figure 4.10 Information loss for different combinations auditory and visual gains; a) VBLR network trained with a fixed gain b) SGD network trained with a fixed gain.

Quadratic with multiple gains

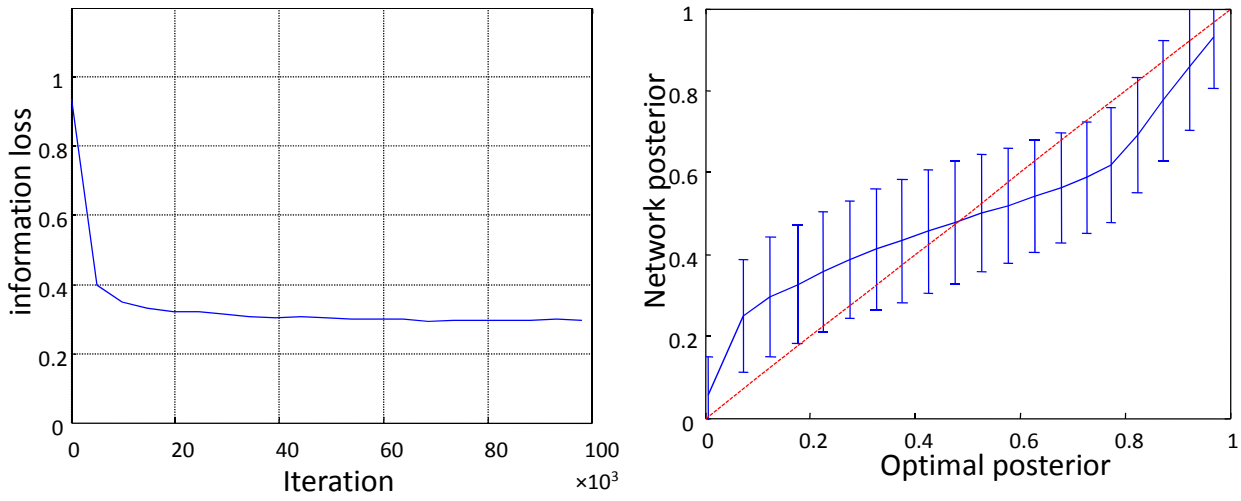


Figure 4.11 Information loss diagram and scatter plot of Quadratic SGD network and analytical results for P(C=1) with multi train gains and multiple test gains.

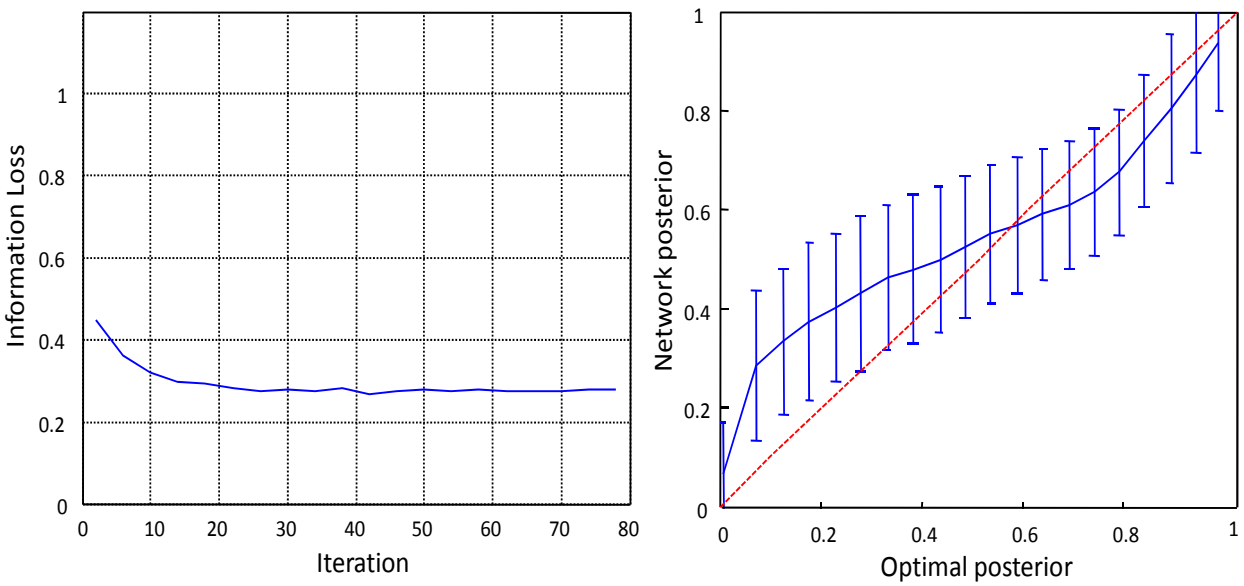


Figure 4.12 Information loss diagram and scatter plot of Quadratic VBLR network and analytical results for P(C=1) with multi train gains and multiple test gains.

Figures 4.11 and 4.12 shows the results of Quad networks trained with multiple and unequal gains using SGD and VBLR respectively. As we predicted multiple and unequal gains result in more complex training data which makes it harder for the network to learn proper weights. This can be seen as more deviation from diagonal line in the scatter plot plus larger error bars as well as higher information loss.

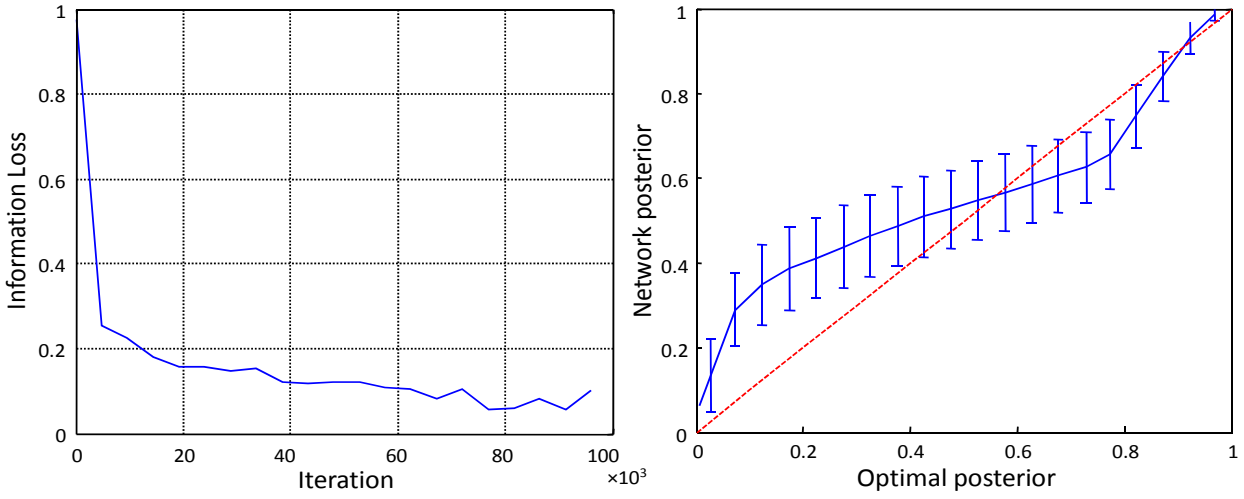


Figure 4.13 Information loss diagram and scatter plot of Quadratic SGD network and analytical results for P(C=1) with multi train gains and fixed test gain.

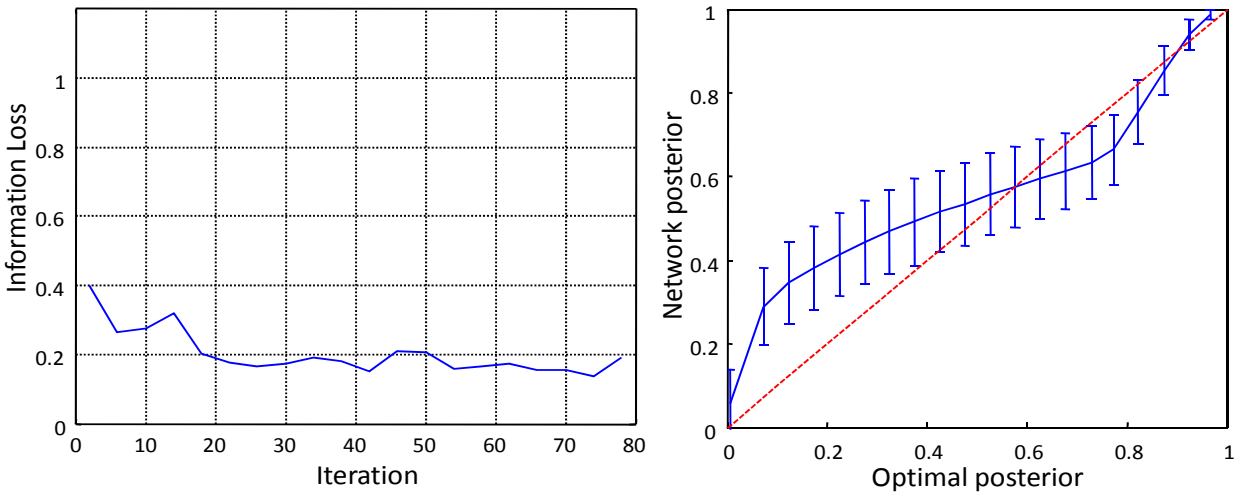


Figure 4.14 Information loss diagram and scatter plot of Quadratic VBLR network and analytical results for P(C=1) with multi train gains and fixed test gain.

We also used data with fixed and equal gains to test the networks trained with multiple and un-equal gains. It can be seen that in scatter plot, the error bars are smaller. The information loss is also smaller in this case. In fact figures 4.13 and 4.14 are the best results among different pairs of equal auditory and visual gains. So we can conclude that the network performs differently for different combination of weights. This is shown more clearly in figures 3 and 4 in appendix. A as scatter plots of different gain combinations and figures 4.15 as information loss of different gain combinations for networks trained with SGD and VBLR algorithms respectively.

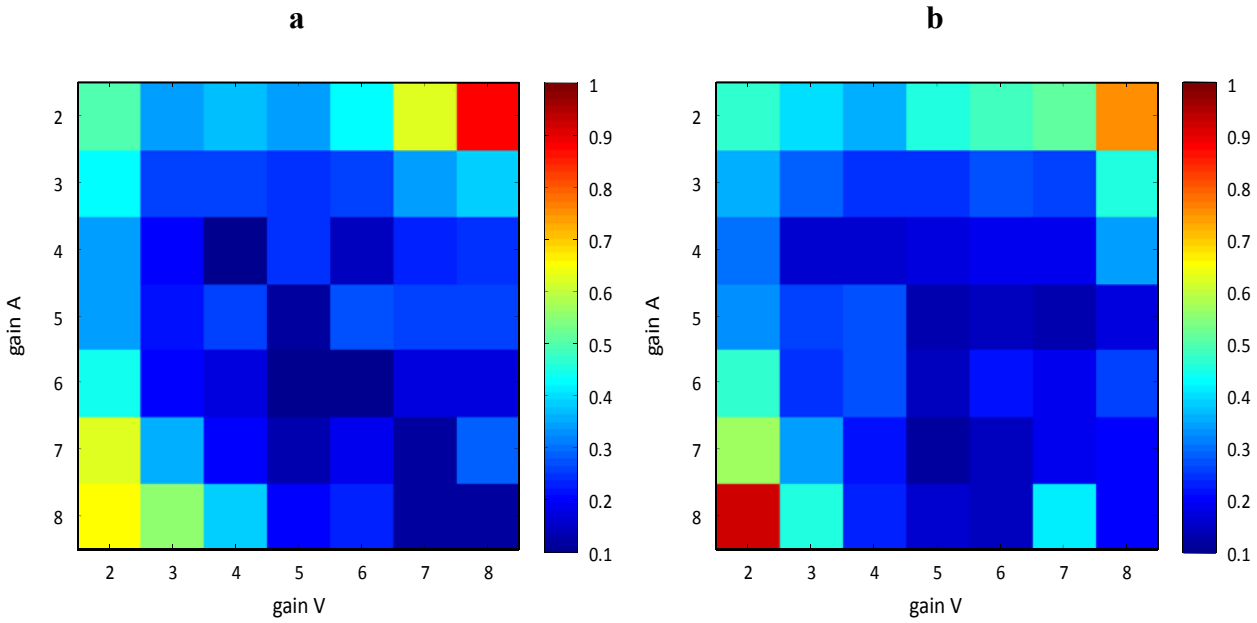


Figure 4.15 Information loss for different combinations auditory and visual gains; a) SGD network trained with multiple gains b) network trained with multiple gains.

4.3.3. QDN

QDN with fixed gain

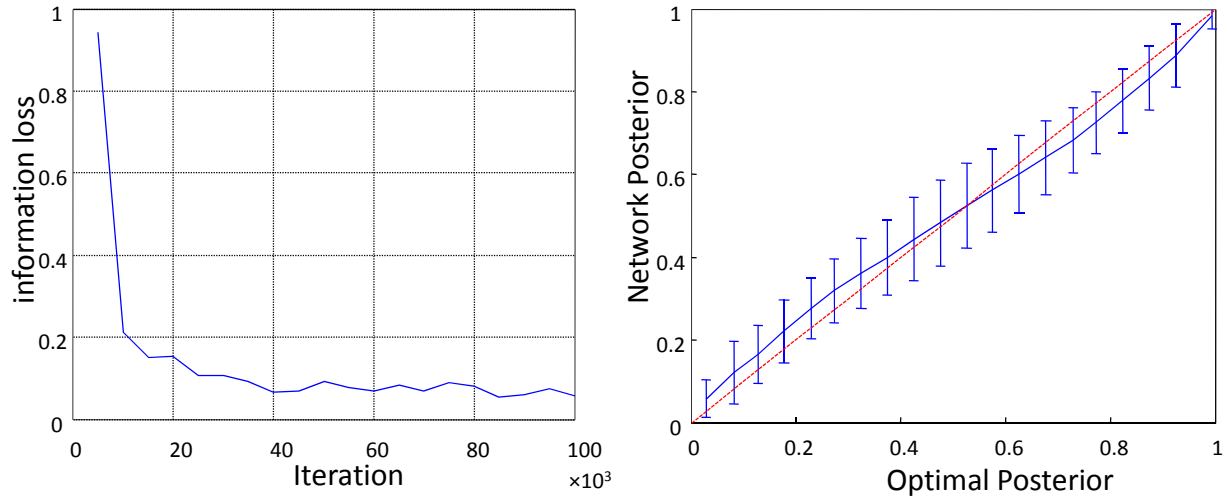


Figure 4.16 Information loss diagram and scatter plot of QDN SGD network and analytical results for P(C=1) with fixed train gain and fixed test gain.

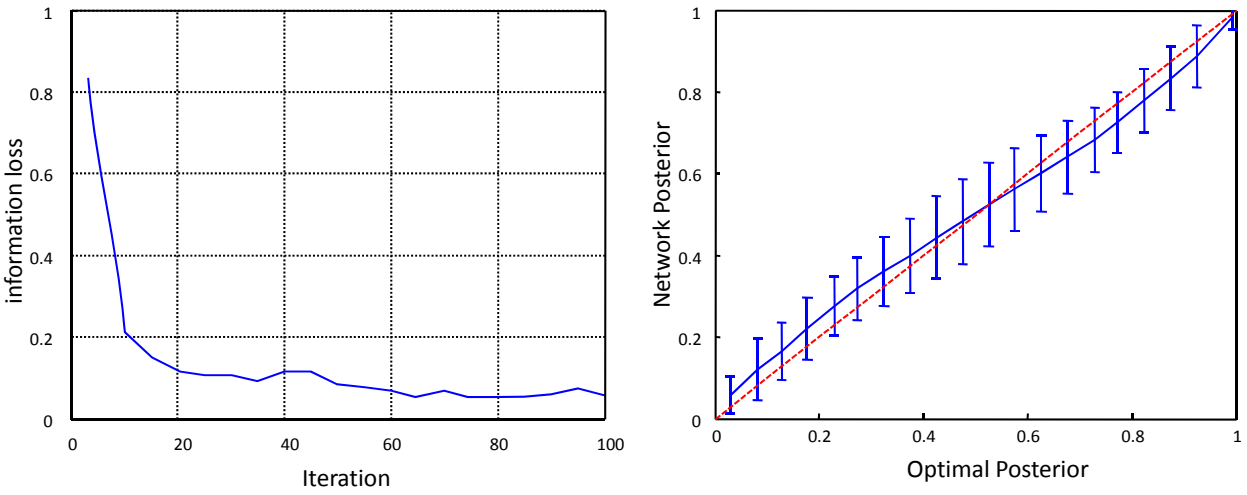


Figure 4.17 Information loss diagram and scatter plot of QDN VBLR network and analytical results for P(C=1) with fixed train gain and fixed test gain.

Figure 4.16 and 4.17 show the $P(C=1)$ calculated with a QDN network. The network in figure 4.18 has been trained using the (SGDS) algorithm and the network in figure 4.19 has been trained with the Variational Bayesian Logistic Regression (VBLR) algorithm. All training trials had fixed and equal visual and auditory gains. This is not compatible with real conditions where the input gains can be different and unequal. Fixed equal auditory and visual gains result in less variance in training data, so the network can learn the weights better and faster. As can be seen in these figures, the scatter plot is almost diagonal but with large error bars in comparison with the mean. The results of the VBLR network are almost identical to those of the SGD network. This shows that the large error bars are not related to the learning algorithm. The information loss is around 8%. This is far from what we expected from a QDN network. Depending on the two classes' data boundaries and the divisive normalization effect, our expectation for the QDN network structure was much higher performance to less than 3% information loss. But the results show that the combination of quadratic terms and first order normalization does not discriminate two classes well enough.

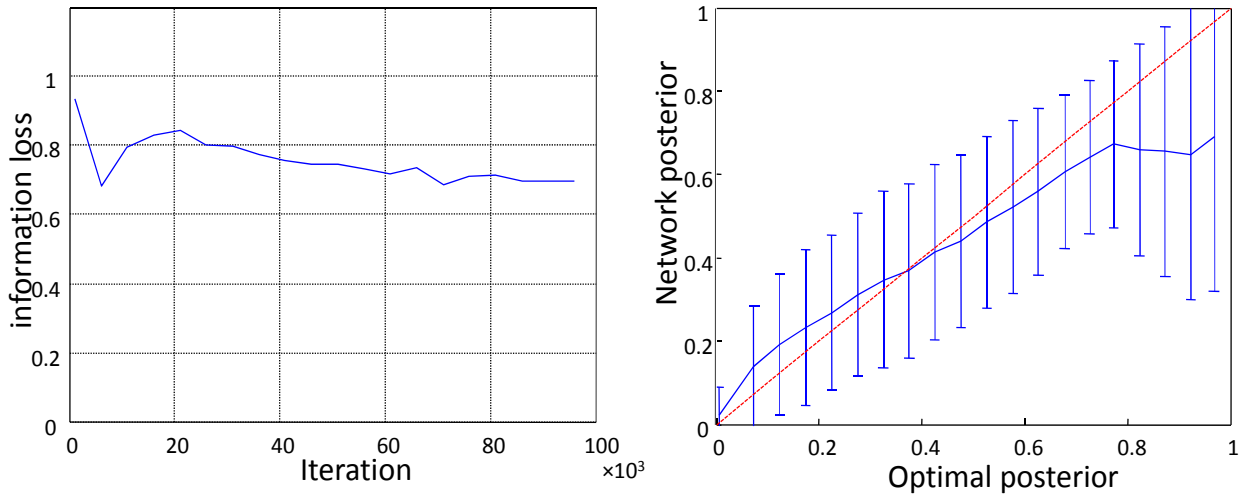


Figure 4.18 Information loss diagram and scatter plot of QDN SGD network and analytical results for P(C=1) with fixed train gain and multiple test gains.

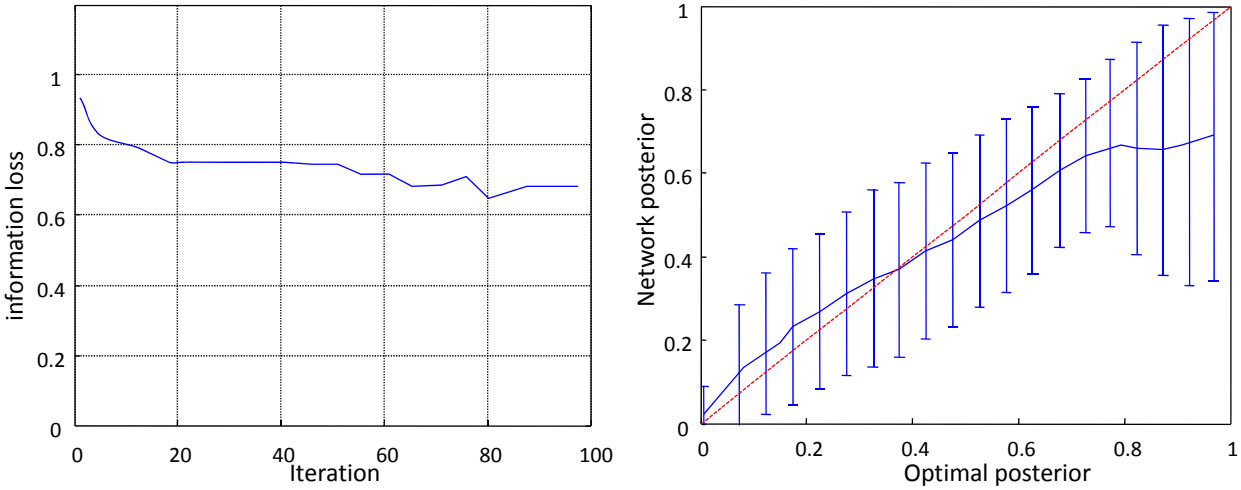


Figure 4.19 Information loss diagram and scatter plot of QDN VBLR network and analytical results for P(C=1) with fixed train gain and multiple test gains.

In addition to the QDN networks trained with equal and fixed gain with same gain, we tested these networks-SGD and VBLR- with multiple and unequal auditory and visual gains. As can be seen in figures 4.18 and 4.19 the network trained with data with fixed and equal gains performs poorly on test data with multiple and unequal gains. The reason for this can be the bigger role of the gains in training. In the QDN network there is an additional set of weights, divisive normalization factors, to be learnt. This increases the effect of variance in the data and as a result makes it more difficult for the network to learn the structure of training data. A test over all possible combination of auditory and visual gains is done. Here again for both SGD and VBLR trainings when the gains are equal, the network performance is much better. As well as the difference be-

tween gains gets larger, both deviation and error bars increase. The scatter plots are shown in figures 5 and 6 of appendix. A.

The information loss of these combinations is shown in figure 4.20 for SGD and VBLR networks respectively. Here also we can see that there is less information loss for combinations with smaller difference between auditory and visual gains.

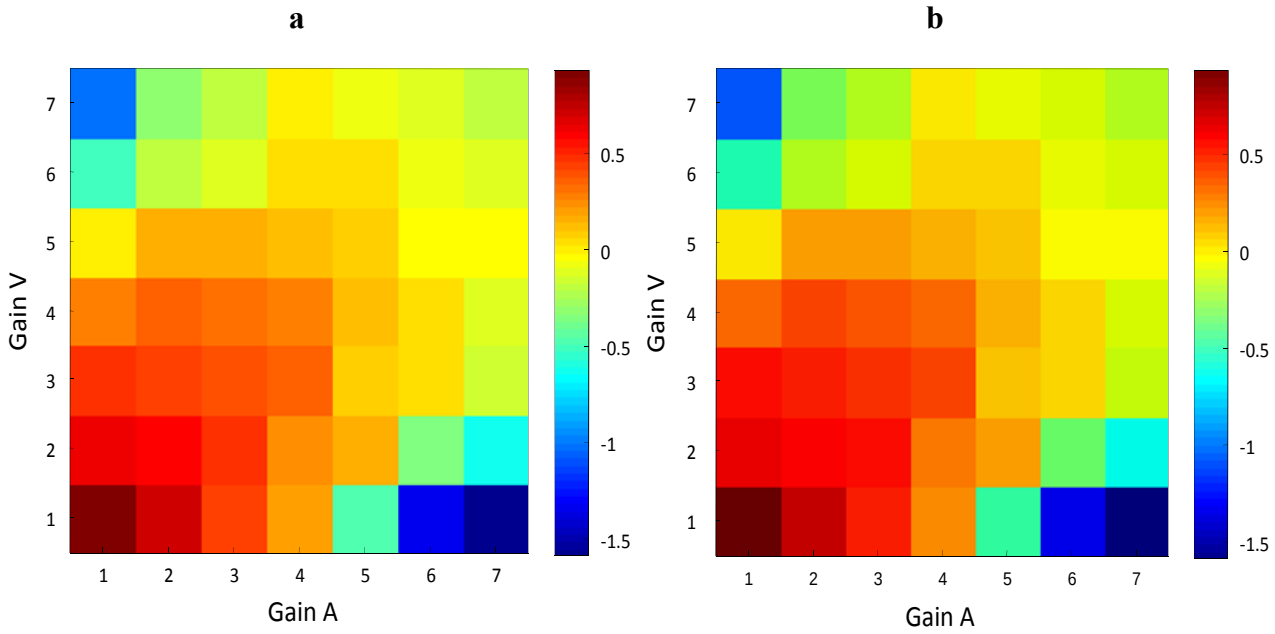


Figure 4.20 Information loss for different combinations auditory and visual gains for QDN; a) VBLR network trained with a fixed gain b) SGD network trained with a fixed gain.

QDN with multiple gains

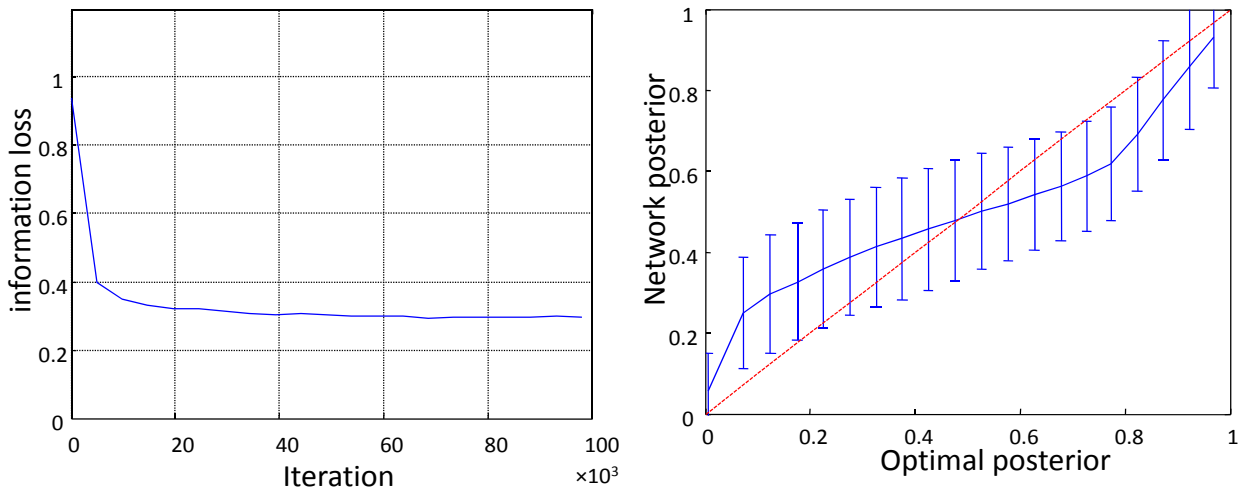


Figure 4.21 Information loss diagram and scatter plot of QDN SGD network and analytical results for P(C=1) with multi train gains and multiple test gains.

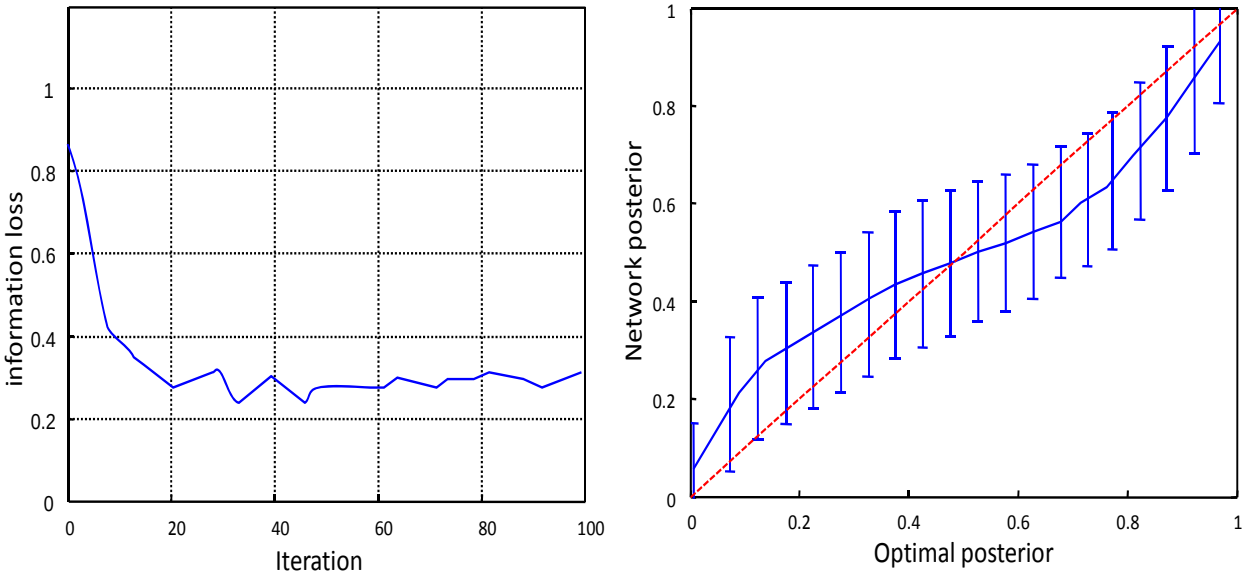


Figure 4.22 Information loss diagram and scatter plot of QDN VBLR network and analytical results for P(C=1) with multi train gains and multiple test gains.

Figures 4.21 and 4.22 show the results of QDN networks trained with multiple and unequal gains using SGD and VBLR respectively. As we predicted multiple and unequal gains result in more complex training data which makes it harder for the network to learn proper weights. This can be seen as more deviation from diagonal line in the scatter plot plus larger error bars as well as higher information loss.

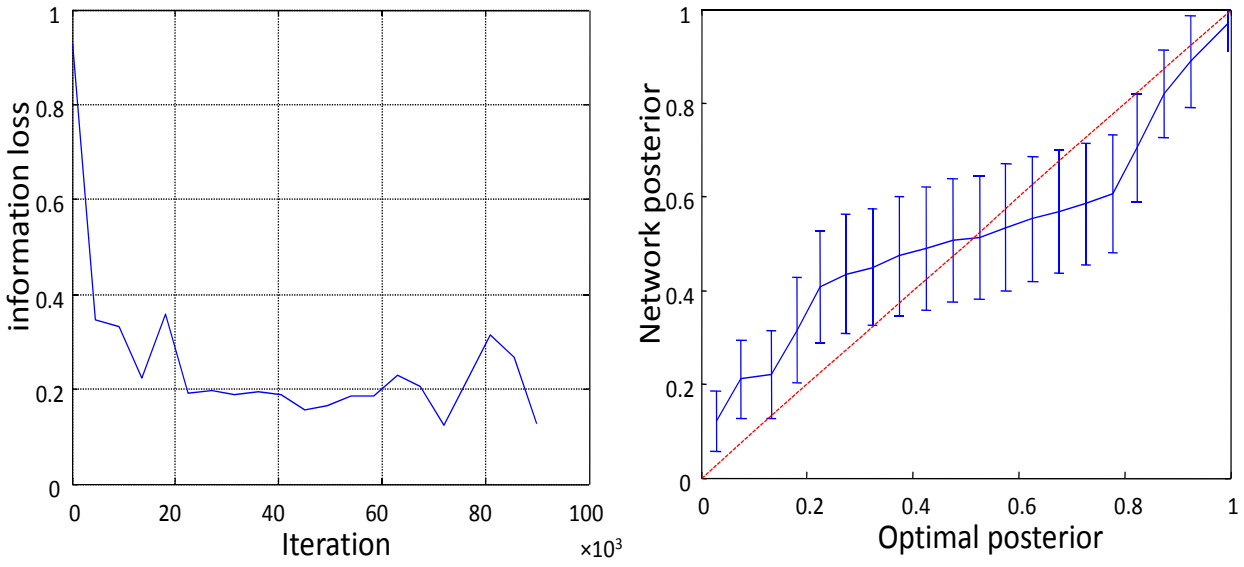


Figure 4.23 Information loss diagram and scatter plot of QDN SGD network and analytical results for P(C=1) with multi train gains and fixed test gain.

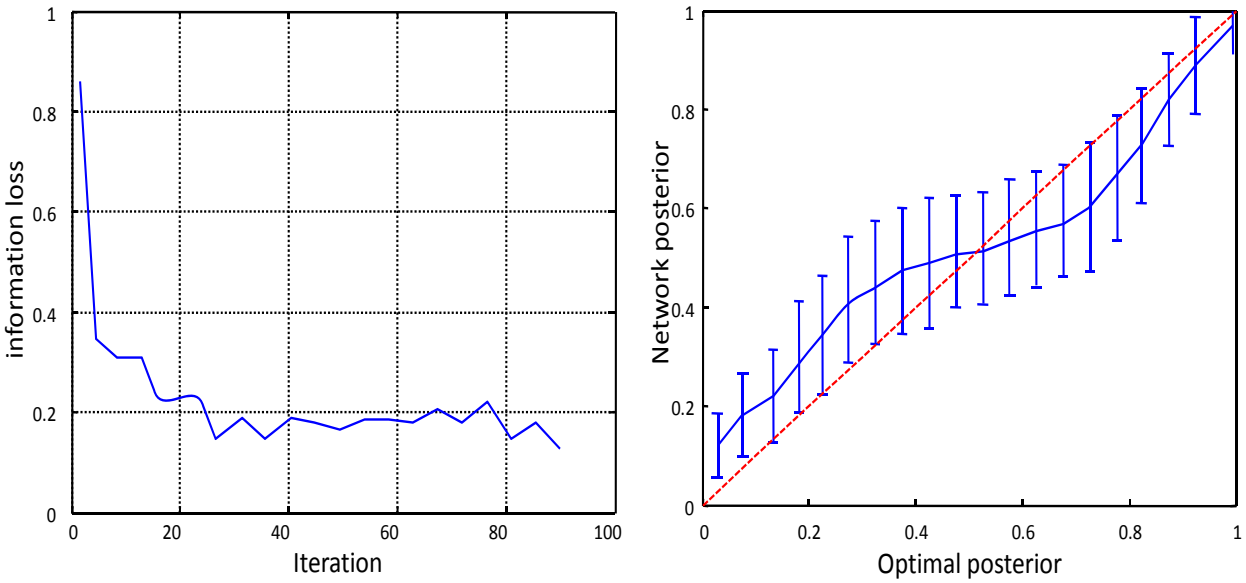


Figure 4.24 Information loss diagram and scatter plot of QDN VBLR network and analytical results for P(C=1) with multi train gains and fixed test gain.

We also used data with fixed and equal gains to test the networks trained with multiple and un-equal gains. It can be seen that in scatter plot, the error bars are smaller. The information loss is also smaller in this case. In fact figures 4.23 and 4.24 are the best results among different pairs of equal auditory and visual gains. So we can conclude that the network performs differently for different combination of weights. This is shown more clearly in figures 7 and 8 of appendix. A.

Figures 4.25 as information loss of different gain combinations for networks trained with SGD and VBLR algorithms respectively.

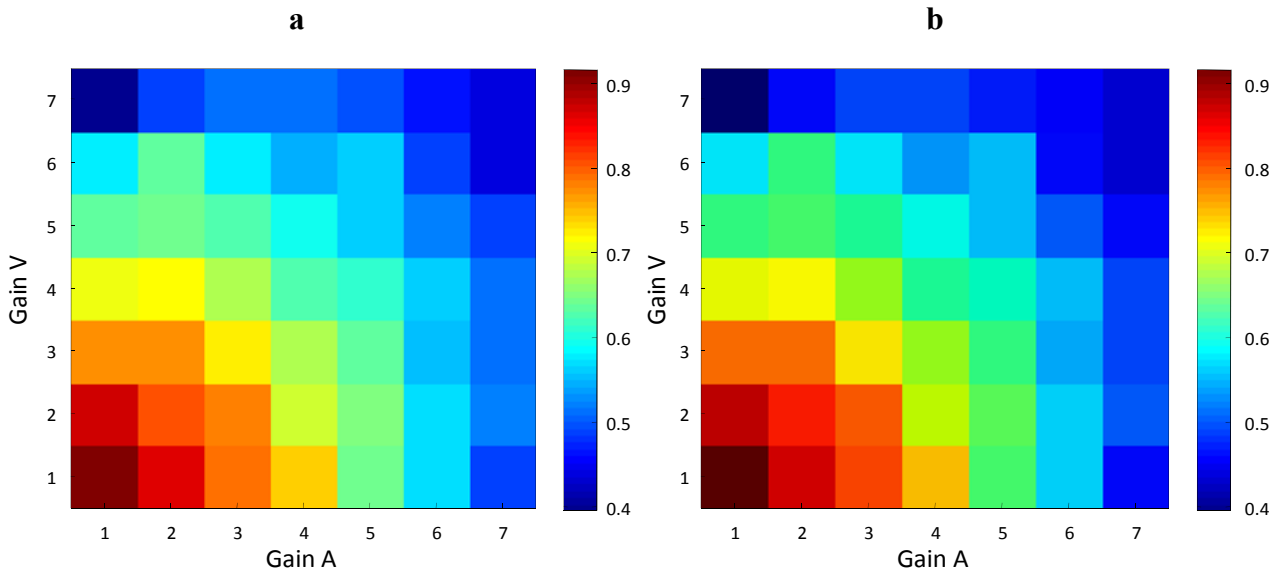


Figure 4.25 Information loss for different combinations auditory and visual gains for QDN; a)SGD network trained with multiple gains b) VBLR network trained with multiple gains.

4.4.Discussion and conclusion

Our simulation results show that to have a neural structure which is capable of doing causal inference task a linear discriminator performs very poorly. Although the quadratic basis function performs better than the linear one, it results in large information loss. This large information loss could be explained the need to divisive normalization for implementing causal inference using probabilistic population codes. However, we found that even a network that uses a first-order divisive normalization as in (2.14) does not decrease the information loss to an acceptable amount. We used two different learning algorithms, SGD and VBLR. These two methods train the network from two different approaches. Almost identical results of these two learning methods show that it is unlikely that the poor performance of the networks is due to the learning method.

One solution to improve the network performance is to construct a more complicated basis function, which could be a higher order polynomial and/or higher-order divisive normalization. Then the question is to what order we should increase the basis function and divisive normalization factors. One possible way is to manipulate formula (2.6) in a way that we can extract higher-order terms.

References

1. Ma, W.J., et al., *Bayesian inference with probabilistic population codes.* Nat Neurosci, 2006. **9**(11): p. 1432-8.
2. Kording, K.P., et al., *Causal Inference in Multisensory Perception.* Plos One, 2007. **2**(9): p. -.
3. van Beers, R.J., A.C. Sittig, and J.J. Gon, *Integration of proprioceptive and visual position-information: An experimentally supported model.* J Neurophysiol, 1999. **81**(3): p. 1355-64.
4. Jacobs, R.A., *Optimal integration of texture and motion cues to depth.* Vision Res, 1999. **39**(21): p. 3621-9.
5. Yuille, A.L. and H.H. Bülthoff, *Bayesian decision theory and psychophysics*, in *Perception as Bayesian inference*, D.C. Knill and W. Richards, Editors. 1996, Cambridge University Press New York, NY, USA p. 123 - 161
6. Ernst, M.O. and M.S. Banks, *Humans integrate visual and haptic information in a statistically optimal fashion.* Nature, 2002. **415**(6870): p. 429-33.
7. Alais, D. and D. Burr, *The ventriloquist effect results from near-optimal bimodal integration.* Curr Biol, 2004. **14**(3): p. 257-62.
8. Wallace, M.T., et al., *Unifying multisensory signals across time and space.* Exp Brain Res, 2004. **158**(2): p. 252-8.
9. Sanger, T.D., *Probability density estimation for the interpretation of neural population codes.* J Neurophysiol, 1996. **76**(4): p. 2790-3.
10. Zemel, R.S., P. Dayan, and A. Pouget, *Probabilistic interpretation of population codes.* Neural Comput, 1998. **10**(2): p. 403-30.
11. Dayan, P. and L.F. Abbott, *Theoretical neuroscience : computational and mathematical modeling of neural systems*. Computational neuroscience. 2001, Cambridge, Mass.: Massachusetts Institute of Technology Press. xv, 460 p.
12. Seung, H.S. and H. Sompolinsky, *Simple models for reading neuronal population codes.* Proc Natl Acad Sci U S A, 1993. **90**(22): p. 10749-53.
13. Hinton, G.E. *Products of Experts*. in *International Confernce on Artificial Neural Networks(ICANN)*. 1999. London, England: IEEE.
14. Snippe, H.P., *Parameter extraction from population codes: a critical assessment.* Neural Comput, 1996. **8**(3): p. 511-29.
15. Bishop, C.M., *Pattern recognition and machine learning*, in *Information science and statistics*. 2006, Springer: New York. p. p. 498-505 .
16. http://www.lnc.ens.fr/~jdrugowi/code_vb.html

Appendix A

The following figures show the scatter plot of the posterior probability of C=1 calculating from the optimal method and using the trained neural network for both SGD and VBLR training methods. It can be that for all the basis function we used, when the gains are equal, the network performance is much better. As well as the difference between gains gets larger, both deviation and error bars increase. We couldn't find an intuitive explanation for the part where the line start folding down when the difference between gains increase. This shows that this network can't discriminate C=1 and C=2 properly, when auditory and visual stimuli have different gains. It is also shown that the scatter plot has larger deviation for cases with smaller gain.

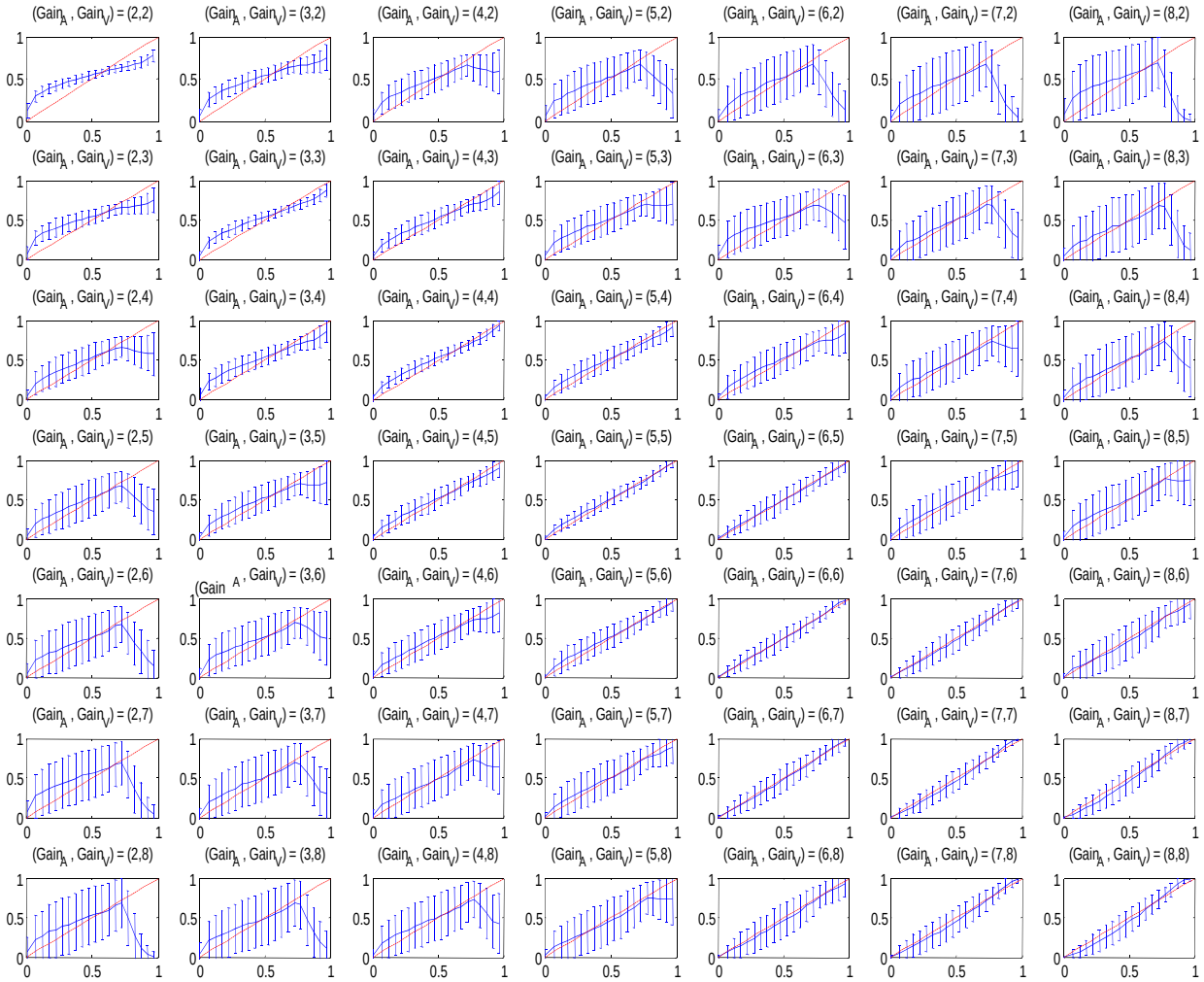


Figure 1 Scatterplot of different combinations of auditory and visual gains for a Quad network trained with SGD algorithm using fixed and equal auditory and visual gains.

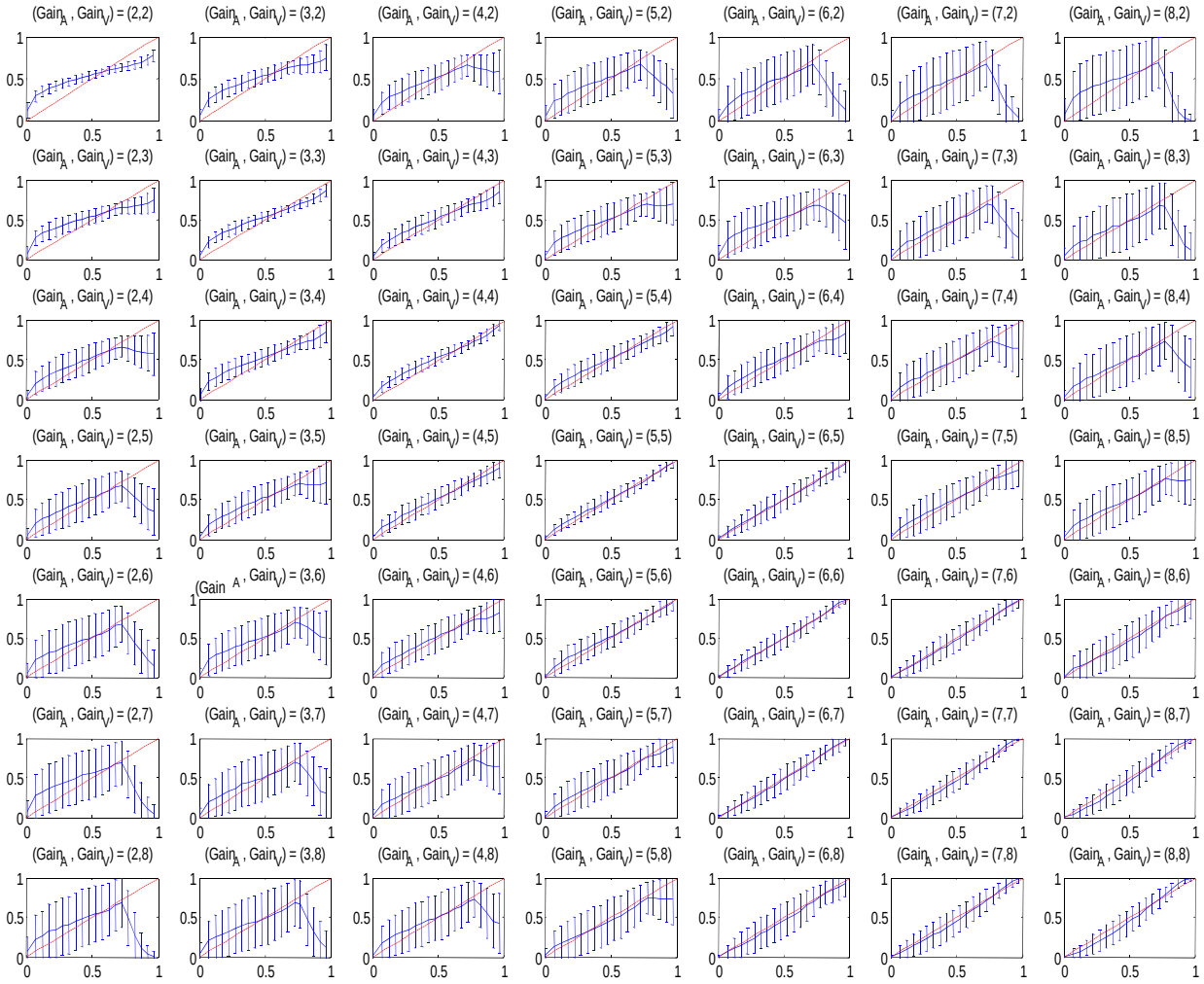


Figure 2 Scatterplot of different combinations of auditory and visual gains for a Quad network trained with VBLR algorithm using fixed and equal auditory and visual gains.

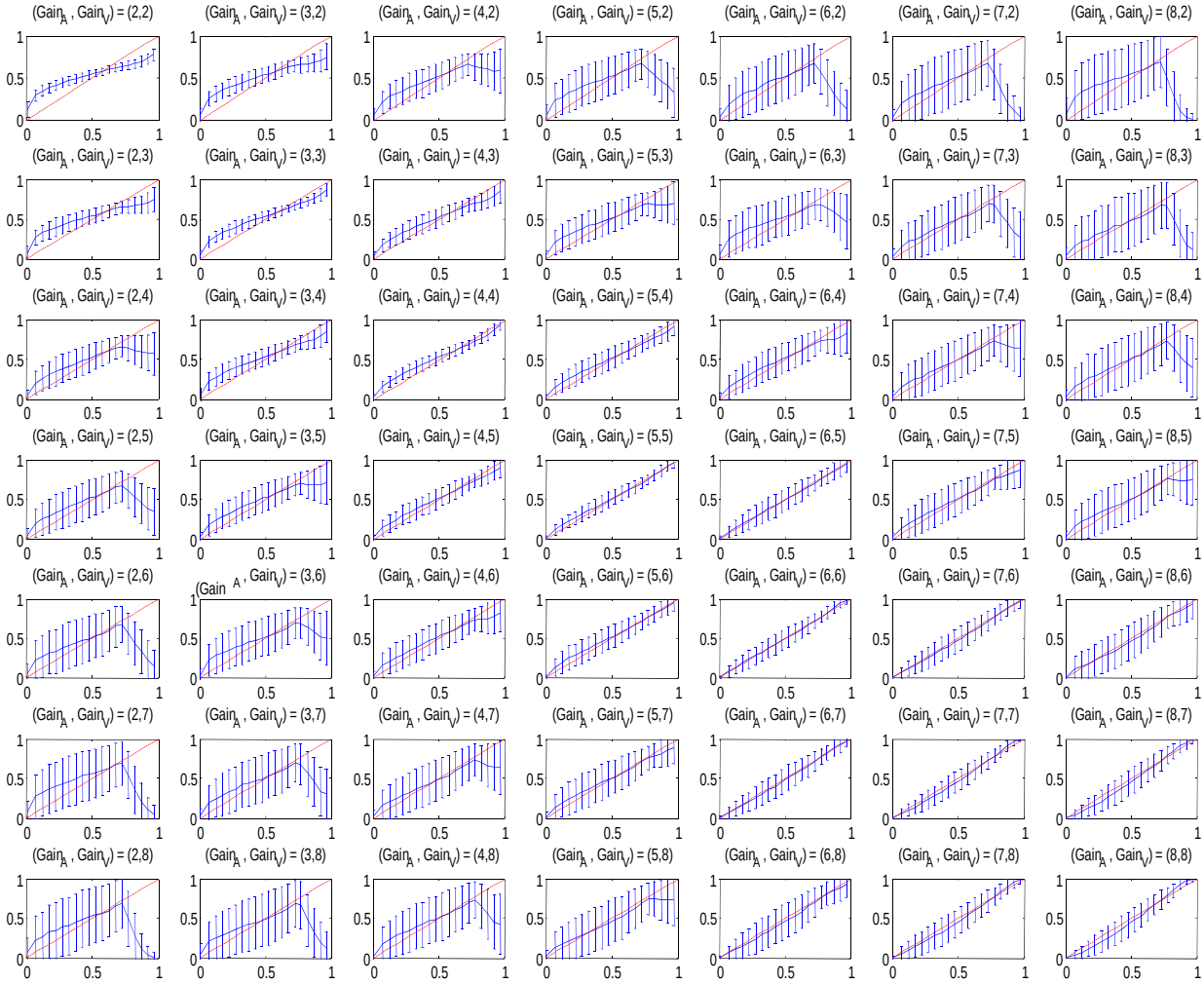


Figure 3 Scatterplot of different combinations of auditory and visual gains for a Quad network trained with SGD algorithm using multiple and unequal auditory and visual gains.

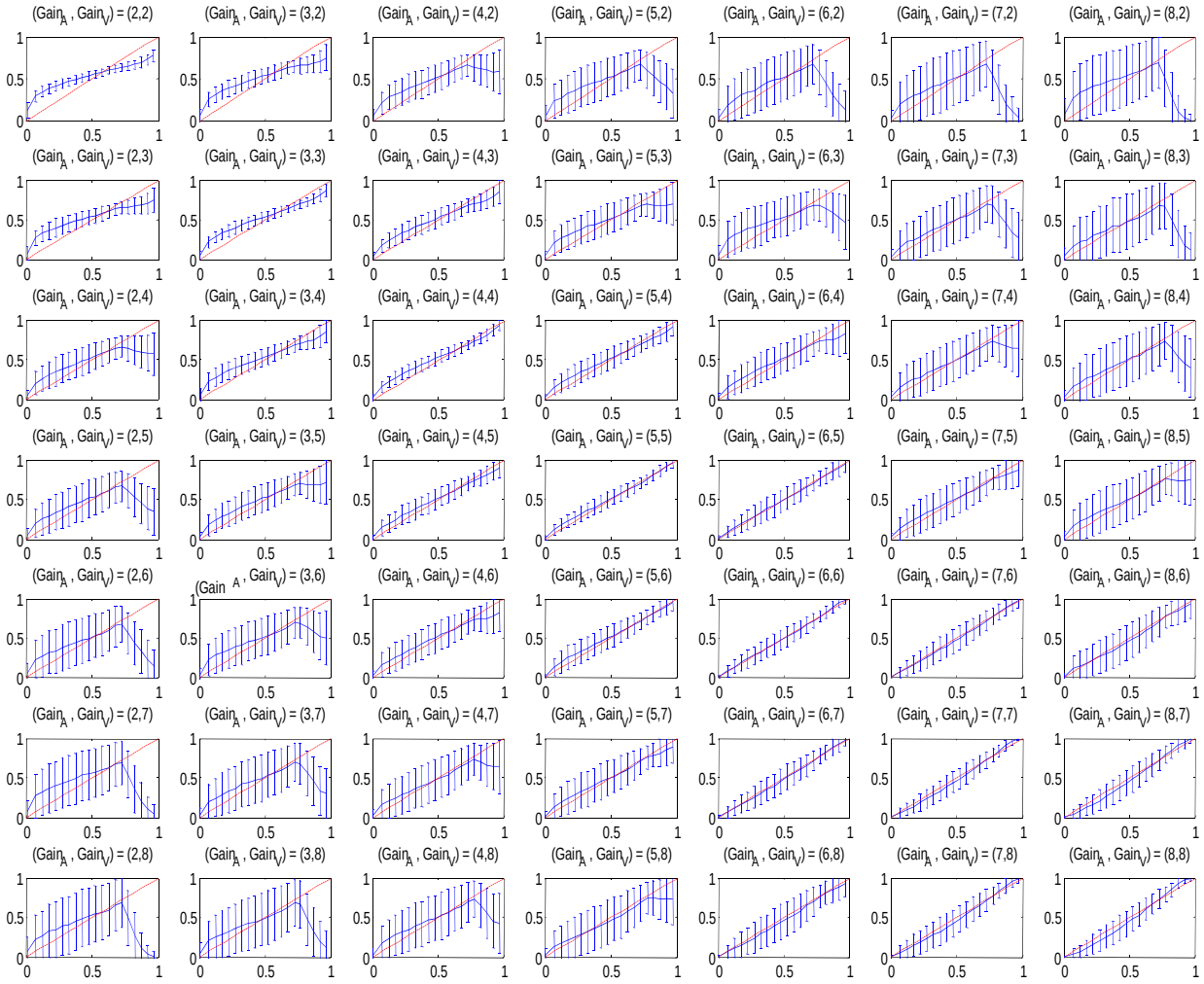


Figure 4 Scatterplot of different combinations of auditory and visual gains for a Quad network trained with VBLR algorithm using multiple and unequal auditory and visual gains.

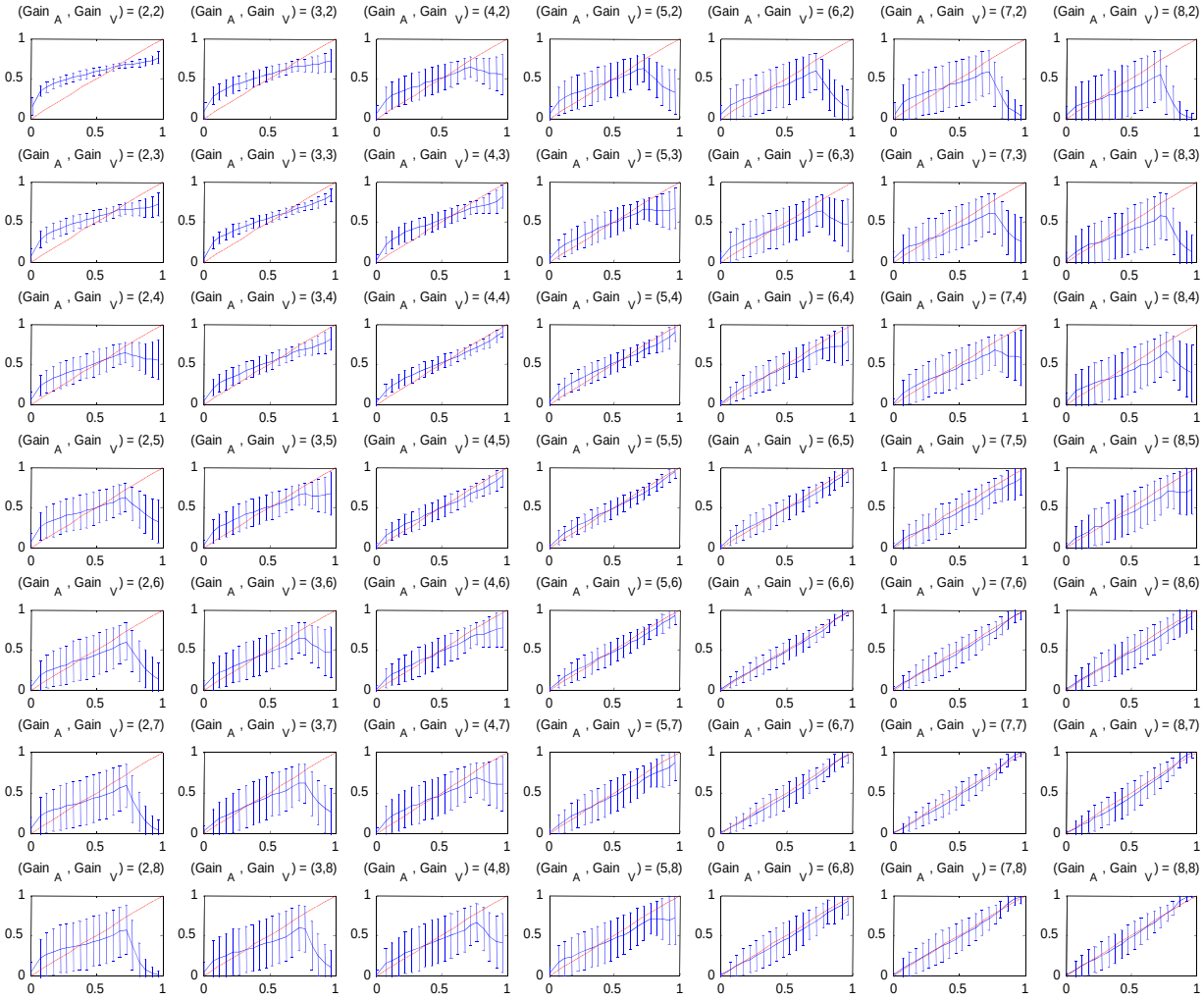


Figure 5 Scatterplot of different combinations of auditory and visual gains for a QDN network trained with SGD algorithm using fixed and equal auditory and visual gains.

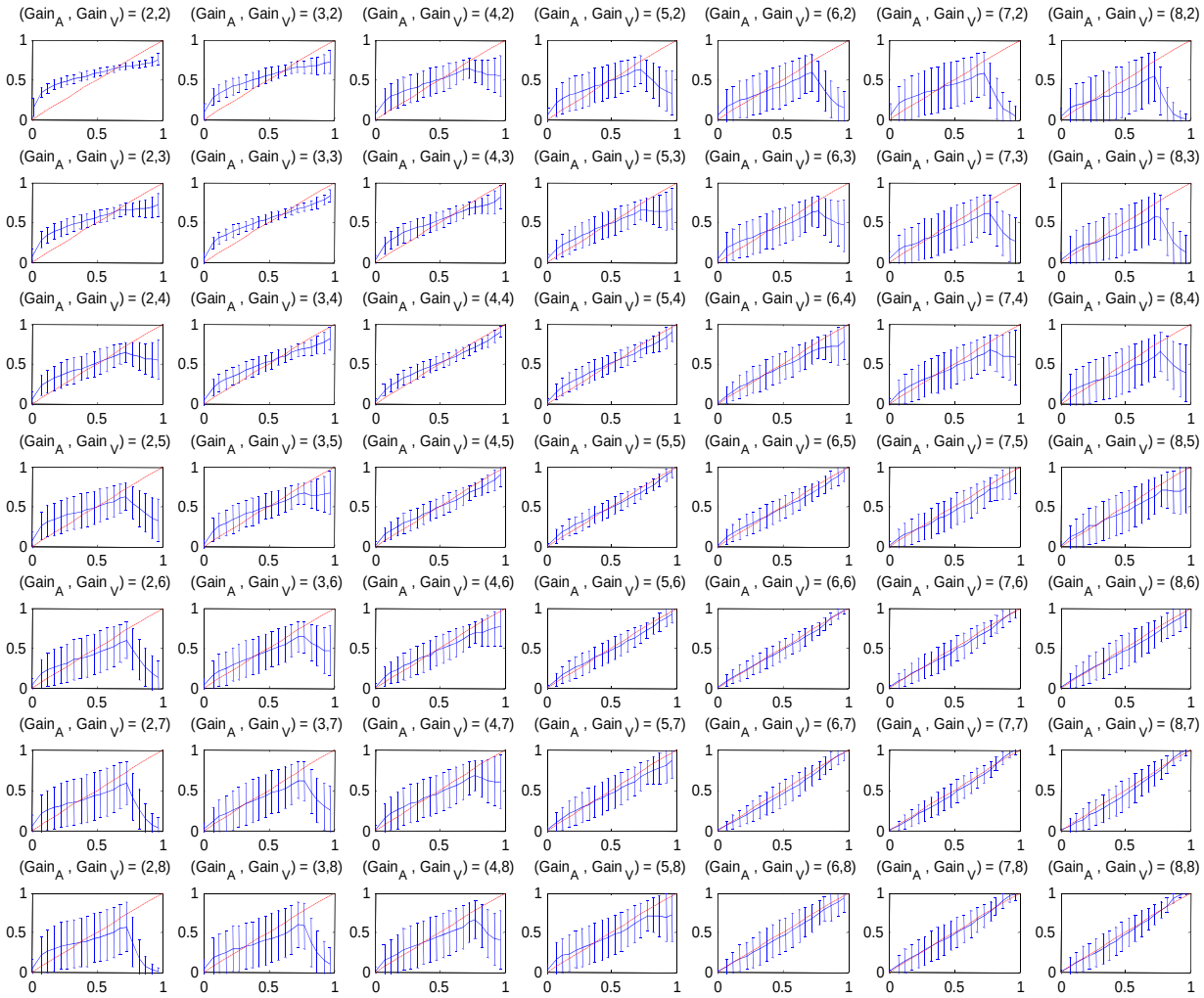


Figure 6 Scatterplot of different combinations of auditory and visual gains for a QDN network trained with SGD algorithm using fixed and equal auditory and visual gains.

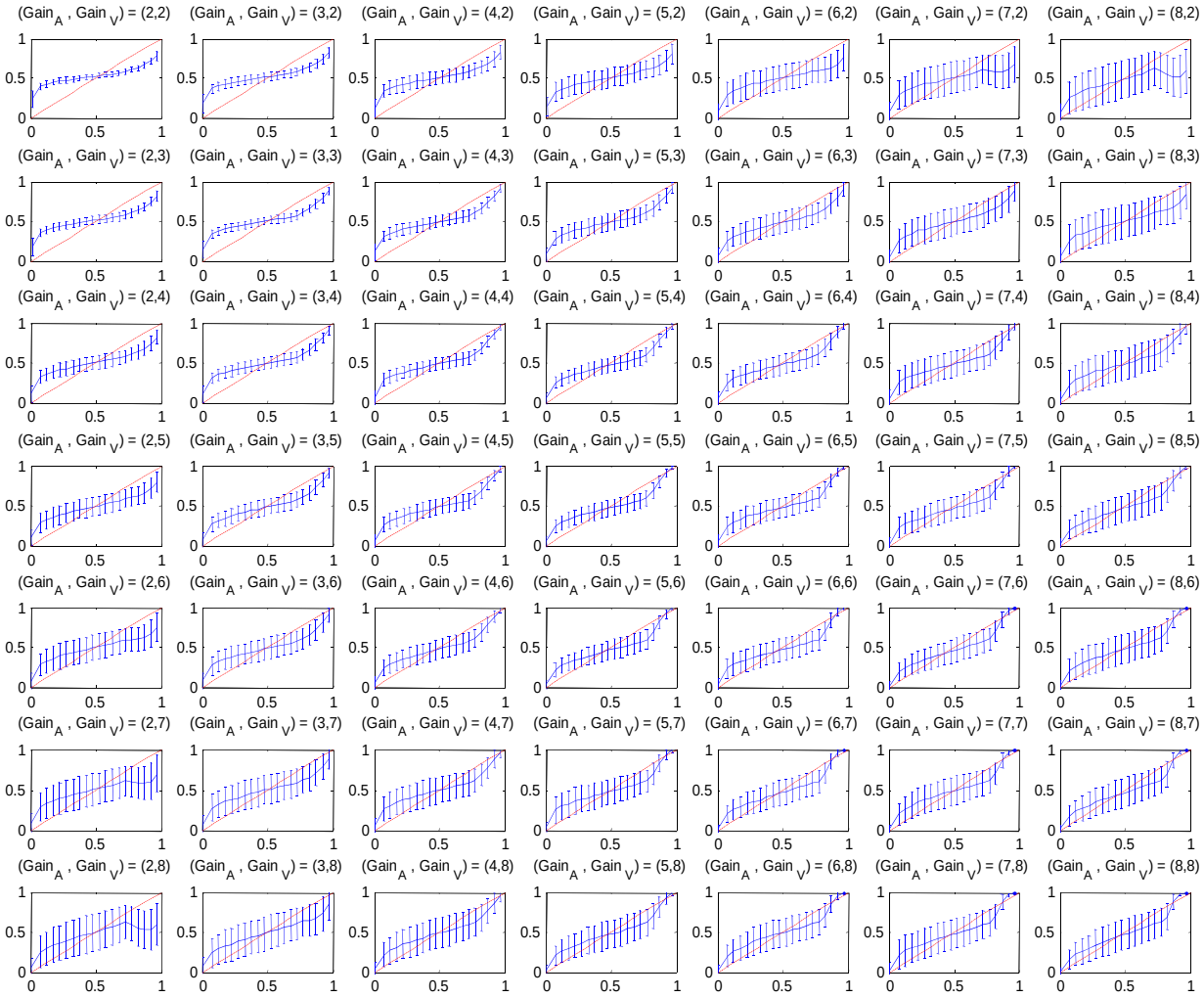


Figure 7 Scatterplot of different combinations of auditory and visual gains for a QDN network trained with SGD algorithm using multiple and unequal auditory and visual gains.

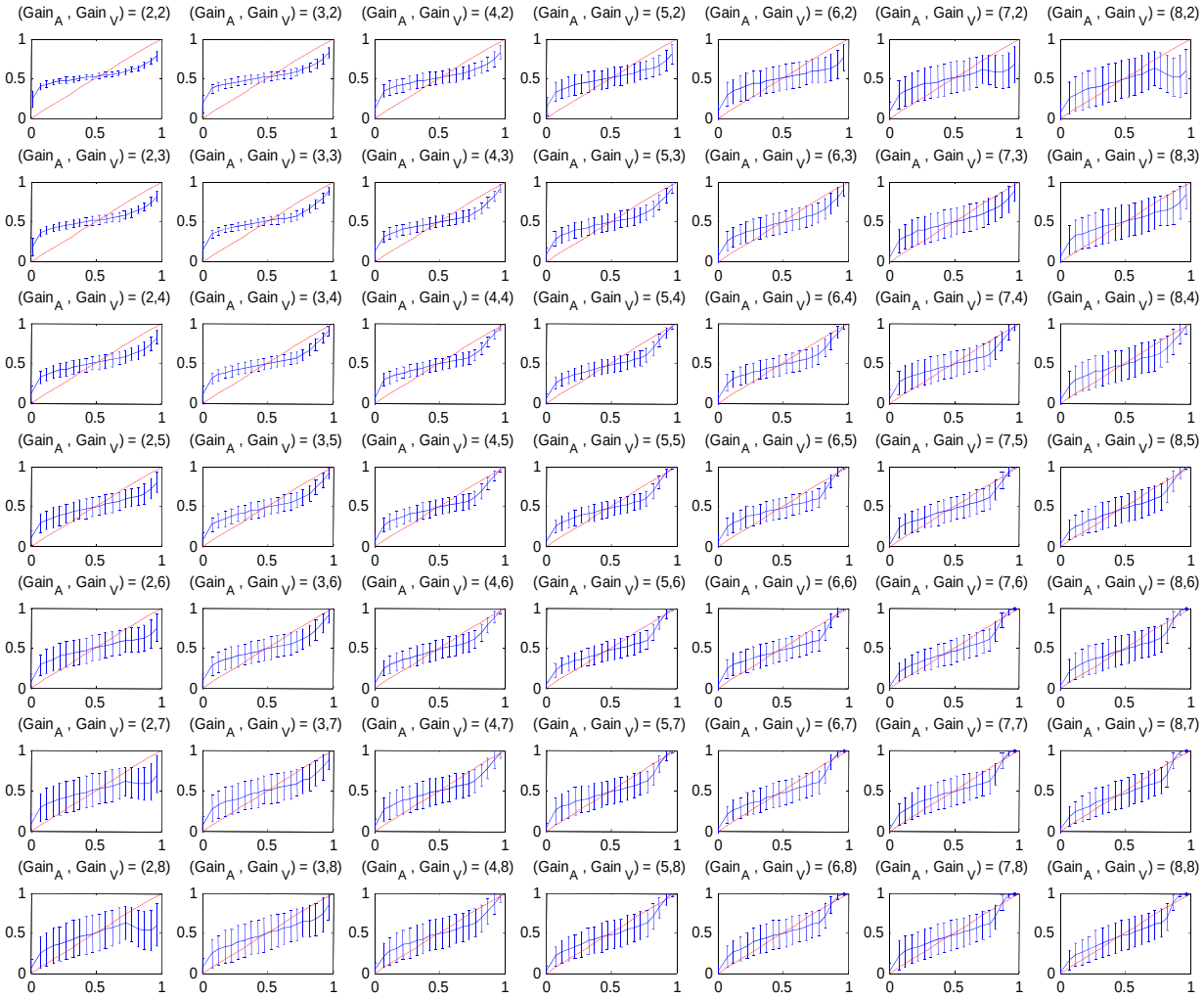


Figure 8 Scatterplot of different combinations of auditory and visual gains for a QDN network trained with VBLR algorithm using multiple and unequal auditory and visual gains.