Training Kanter's bit generator by natural gradient descent

Daniel Wagenaar*

10th September 1998

Natural gradient descent (NGD) learning is compared with ordinary gradient descent (OGD) for Kanter's bit generator. Analytic results for one or two input bits and one output bit show that the generalization error decreases exponentially with time for NGD learning, while OGD atudents attain an error that only decreases inversely proportional to learning time. Similar results are found numerically for two input bits and more output bits. In some cases students end up in local minima. In this study NGD suffered slightly more from this problem than OGD, but we suspect that in other models it could just as easily work out the other way round. For more than two input bits no analytic results are obtained, and various options for future research with numerical methods are suggested.

* wagenaar@mth.kcl.ac.uk

Daniel Wagenaar, *Training Kanter's bit generator by natural gradient descent — MSc project for Information Processing and Neural Networks*

Centre for Neural Networks, King's College London, September 1998

Supervisor: Dr A. C. C. Coolen

CONTENTS

Introduction **4**

Conventions **6**

1 Natural gradient descent and Kanter's bit generator **7**
1.1 Ordinary and natural gradient descent 7
1.2 Bit generation 8
1.3 The metric for the bit generator 9
1.4 The $N = 1$ bit generator 9

2 The $N = 2$ bit generator **12**
2.1 Input distribution 12
2.2 Metric 13
2.3 Natural gradient descent learning 14
2.4 $N = 2$ bit generator with ordinary gradient descent 14
2.A Calculations 16

3 Predicting further into the future for $N = 2$ **18**
3.1 Generalized definition of training error 18
3.2 Metric for $n > 1$ 19
3.3 Simulations 19
3.A Calculations 23

4 The road to larger N **25**
4.1 The metric in $\boldsymbol{w}$-space 25
4.2 Alternative approaches 26

Conclusions and outlook **29**

Bibliography **30**

INTRODUCTION

Gradient descent has since long been an important training technique for neural networks. Its aim is to find a minimum of the generalization error by repeatedly making small changes to the weights of a student neural networks each of which reduces the error. In its most basic form, the weight changes prescribed by gradient descent are

$$\frac{\mathrm{d}w^i}{\mathrm{d}t} = -\alpha \frac{\partial \mathcal{E}_g[\boldsymbol{w}]}{\partial w^i},$$

where w^i are N weights, $\mathcal{E}_g$ is the error function and α is a constant which is called the learning rate.

It was soon found however, that this form of gradient descent was far from optimal in the sense of achieved learning times. Various schemes were investigated, for example time dependent learning rates $\alpha(t)$ or choosing the learning rate independently for each layer of multi-layered networks.

In 1985 Amari proposed a novel technique for choosing the learning rate using concepts from Riemannian geometry [1, 2]. It involves independently scaling each weight change inversely with the impact such a change would make on the probability distribution of network outputs[1)]. This method of 'natural' gradient descent has been found to offer significant improvement over ordinary gradient descent in terms of learning time for feed forward networks [3, 4, 5].

In this thesis I shall investigate the merits of using natural gradient descent as a learning rule for Kanter's 'bit generator' [6], a device that produces an infinite sequence of bits from a small number of initial values. Each bit is computed by feeding the previous N bits through a stochastic binary neuron. For example, with $N = 3$, the sixth bit is computed by feeding the third, fourth and fifth bits through the neuron.

This is perhaps the simplest model that can be used for time series prediction, an art which has applications ranging from forecasting stock values to controlling the equipment on intensive care units. In any practical application, one would clearly use a somewhat higher dimensional system: instead of single bits, one would typically generate (vectors of) real numbers, and one would probably feed them through slightly more complex networks as well. However, in this pilot study we have opted to concentrate on the simplest model. We hope that our results may be a first step towards insight into the more realistic systems.

Still, although it looks very simple, the bit generator has quite non-trivial dynamics. Some very interesting research into this subject has been done by Kanter et al. For example, it can produce pseudo-periodic cycles of various lengths [7].

[1)] In chapter one I shall be more precise.

The present work will not focus on the sequence itself but on the dynamics of learning: given a part of a 'teacher' bit sequence, we wish to train a 'student' bit generator to correctly predict the next few bits when presented with any piece of N bits from the sequence. This is done by minimizing the training error — that is, the probability that the student incorrectly predicts a small number of bits from the training sample given the previous N bits. For noise free bit generators, Kanter et al show that for any N there is a p less than N such that a student which is trained to perfection on a training sample of length $N + p$ (i.e. a sample containing p predictable bits) can faultlessly predict the rest of the sequence, but an analogous result has not been found for noisy bit generators.

The structure of the rest of this text is as follows: chapter one will begin with a short review of the mathematics of natural gradient descent, after which the learning process of the $N = 1$ bit generator will be analyzed as a simple example. In chapter two we shall calculate the geometry and error dynamics of the $N = 2$ bit generator when a single bit is to be predicted. Chapter three looks at prediction of more than a single bit for $N = 2$. In Chapter four I briefly discuss some of the possible methods to proceed beyond $N = 2$. Conclusions are presented in the final chapter.

CONVENTIONS

The following notational conventions are used in this text:

- The number of input bits will always be denoted by N.
- The number of bits to be predicted will be denoted by n.
- Vectors in weight space shall have upper indices, so $\boldsymbol{w} \equiv (w^i)_{i=1\ldots N}$. Normally, $\boldsymbol{w}$ will be a student weight vector while $\tilde{\boldsymbol{w}}$ will be a teacher weight vector.
- Learning time will be denoted by t and should not be confused with indices into the bit sequence, for which we shall use j, k, ….
- Spin[1)] indices will be counted from the beginning of the sequence *or* from the beginning of the student's input.
- No sums over indices will be left implicit, but the ranges of summation will not normally be mentioned, so $\sum_i (w^i)^2 \equiv \sum_{i=1}^N (w^i)^2$.
- Except where mentioned, σ and τ will be spin variables, so $\sum_\tau \equiv \sum_{\tau\in\{-1,1\}}$.
- We shall use standard Riemannian geometry conventions in that (g^{ij}) is the matrix inverse of (g_{ij}). Where appropriate metrics will be adorned with an additional label specifying in which parametrization it is supposed to be computed, so $g^{(\boldsymbol{x})}_{\tau\tau'} \equiv \sum_{\boldsymbol{\sigma}} p \frac{\partial \log p}{\partial x^\tau} \frac{\partial \log p}{\partial x^{\tau'}}$, while $g^{(\boldsymbol{w})}_{ij} \equiv \sum_{\boldsymbol{\sigma}} p \frac{\partial \log p}{\partial w^i} \frac{\partial \log p}{\partial w^j}$.
- The notation $\mathrm{E}[\cdots]$ will be used to denote expectation values. The probability distribution with which this expectation value is to be taken will be mentioned in the surrounding text.
- To reduce notational clutter we often write $\langle \mathcal{F}(\sigma_1,\ldots,\sigma_m)\rangle_{\sigma_3,\ldots,\sigma_m}$, instead of the more explicit $\frac{1}{2^{m-2}} \sum_{\sigma_3,\ldots,\sigma_m} \mathcal{F}(\sigma_1,\ldots,\sigma_m)$. Note that in this particular example σ_1 and σ_2 are *not* averaged out.

[1)] The terms 'bit' and 'spin' will be used without distinction. Thus, our 'bits' take values ± 1.

CHAPTER 1

NATURAL GRADIENT DESCENT AND KANTER'S BIT GENERATOR

After a brief overview of the mathematics of natural gradient descent, we shall obtain a general expression for the metric of Kanter's bit generator. In the final section we shall compute the metric for just one input bit, and find the generalization error as a function of learning time.

1.1 Ordinary and natural gradient descent

As mentioned in the introduction,

$$\frac{\mathrm{d}w^i}{\mathrm{d}t} = -\alpha \frac{\partial \mathcal{E}_g[\boldsymbol{w}]}{\partial w^i},$$

or its discretized form

$$w^i \mapsto w^i - \alpha \frac{\partial \mathcal{E}_g[\boldsymbol{w}]}{\partial w^i}. \tag{1.1}$$

is by no means the optimal variant of gradient descent learning. One theoretical argument to prove this statement is the following: the changes prescribed by (1.1) strongly depend on the way the network is parametrized. This immediately proves that it cannot be the optimal way to update the weights: since a reparametrization clearly does not really alter the learning task, the quickest training path (the one that optimally uses the available information) must be independent of the way the student is parametrized.

A way to create a parametrization invariant version of gradient descent was suggested by Amari in [1] using concepts from Riemannian geometry[1)]: technically, the problem with (1.1) is that a covariant vector is subtracted from a contravariant vector. A much better behaved learning rule would be

$$w^i \mapsto w^i - \alpha \sum_j g^{ij} \frac{\partial \mathcal{E}_g[\boldsymbol{w}]}{\partial w^j}, \tag{1.2}$$

where the matrix (g^{ij}) — the *inverse metric* — takes care of the parametrization dependence. With the proper choice of metric, (1.2) is called *natural gradient descent* (NGD). For contrast, I shall refer to (1.1) as *ordinary* gradient descent (OGD).

It has recently been proven [10] that the only metric which makes the learning rule independent of both the parametrization of the weight space and of the input-output space is the Fisher metric:

$$g_{ij}(\boldsymbol{w}) = \sum_{\boldsymbol{\sigma}} p_{\boldsymbol{w}}(\boldsymbol{\sigma}) \frac{\partial \log(p_{\boldsymbol{w}}(\boldsymbol{\sigma}))}{\partial w^i} \frac{\partial \log(p_{\boldsymbol{w}}(\boldsymbol{\sigma}))}{\partial w^j}. \tag{1.3}$$

[1)] For a gentle introduction to Riemannian geometry, textbooks on general relativity, such as [8] are a good starting point. My short thesis on the applications of Riemannian geometry in neural networks [9] may also be of use.

Thus (1.2) — with g^{ij} the matrix inverse of (1.3) — is shown to be the optimal gradient descent rule. Note in particular that time dependent learning rates cannot offer any further improvements for NGD learning[1)].

Before we go on to investigate the metric for the bit generator, we shall give a more explicit definition of the way the bit sequences are generated.

1.2 Bit generation

Kanter's bit generator produces a sequence of bits as a function of some binary inputs and a number of weights: the inputs are N spins, $\sigma_1,\ldots,\sigma_N$ taking values ± 1, the output consists of an infinite sequence of spins, $\sigma_{N+1},\sigma_{N+2},\ldots$. Each spin is calculated by feeding the previous N bits through a neuron:

$$\sigma_k = \mathrm{sgn}\left(\tanh\left(\beta\sum_{i=1}^{N} w^i\sigma_{k-i}\right)+\eta_k\right), \tag{1.4}$$

where w^i, $i=1\ldots N$ are real valued connection weights, β can be viewed as an inverse temperature and η_k are random variables, independently drawn from the uniform $[-1,1]$ distribution.

The probability to find a given subsequence of length n may now be computed as

$$p^n_{\boldsymbol{w}}(\sigma_{N+1},\ldots,\sigma_{N+n}|\sigma_1,\ldots,\sigma_N) = \prod_{m=N+1}^{N+n} p_{\boldsymbol{w}}(\sigma_m|\sigma_{m-N},\ldots\sigma_{m-1}), \tag{1.5}$$

where

$$p_{\boldsymbol{w}}(\sigma_m|\sigma_{m-N},\ldots,\sigma_{m-1}) = \tfrac{1}{2}+\tfrac{1}{2}\sigma_m\tanh(\beta\sum_{i=1}^{N} w^i\sigma_{m-i}). \tag{1.6}$$

In its simplest form[2)] the student's training error on a sample $(\tilde\sigma_1,\ldots,\tilde\sigma_{N+p})$ is defined as the probability that the student incorrectly predicts a randomly picked spin from the sequence given full information of the previous spins:

$$\mathcal{E}_t = \tfrac{1}{2}-\tfrac{1}{2}\frac{1}{p}\sum_{k=N+1}^{N+p}\tilde\sigma_k\mathrm{E}_{\tilde{\boldsymbol{\sigma}}}[\sigma_k], \tag{1.7}$$

where $\mathrm{E}_{\tilde{\boldsymbol{\sigma}}}[\sigma_k]$ is the expectation value of the student's k-th spin given that its previous N spins are taken from the teacher sequence:

$$\mathrm{E}_{\tilde{\boldsymbol{\sigma}}}[\sigma_k] = \sum_{\sigma_k}\sigma_k p_{\boldsymbol{w}}(\sigma_k|\sigma_{k-N}=\tilde\sigma_{k-N},\ldots,\sigma_{k-1}=\tilde\sigma_{k-1}).$$

In the limit $p\to\infty$ the training error reduces to a generalization error.

[1)] This is only true in cases where the full generalization error can be known. For on-line learning — where the generalization error is estimated on the basis of a single measurement — there is as yet no proof that natural gradient descent is optimal.

[2)] That is, for a student that predicts a single bit. In chapter three a more general case will be considered.

1.3 The metric for the bit generator

An important difference between our learning task and most others, is that the probability distribution of inputs is explicitly dependent on the teacher bit sequence. This in turn makes the metric dependent on the teacher. Fortunately, the full distribution of pairs of input and student output can be decomposed as

$$p_{\boldsymbol{w},\tilde{\boldsymbol{w}}}(\sigma_1,\ldots,\sigma_{N+1}) = p^{(\mathrm{input})}_{\tilde{\boldsymbol{w}}}(\sigma_1,\ldots,\sigma_N)\, p_{\boldsymbol{w}}(\sigma_{N+1}|\sigma_1,\ldots,\sigma_N),$$

where the input distribution $p_{\tilde{\boldsymbol{w}}}(\sigma_1,\ldots,\sigma_N)$ equals the frequency with which the subsequence $(\sigma_1,\ldots,\sigma_N)$ occurs within the (infinite length) teacher bit sequence, and thus clearly depends on the teacher weights $\tilde{\boldsymbol{w}}$ (only). The output conditional distribution $p_{\boldsymbol{w}}(\sigma_{N+1}|\sigma_1,\ldots,\sigma_N)$ on the other hand, depends only on the student weights $\boldsymbol{w}$.

We thus get the following general expression for the metric:

$$g_{ij}(\boldsymbol{w};\tilde{\boldsymbol{w}}) = \sum_{\sigma_1,\ldots,\sigma_N} p^{(\mathrm{input})}_{\tilde{\boldsymbol{w}}}(\sigma_1,\ldots,\sigma_N) \sum_{\sigma_{N+1}} p_{\boldsymbol{w}}(\sigma_{N+1}|\sigma_1,\ldots,\sigma_N) \times \\ \times \frac{\partial}{\partial w^i} \log p_{\boldsymbol{w}}(\sigma_{N+1}|\sigma_1,\ldots,\sigma_N) \frac{\partial}{\partial w^j} \log p_{\boldsymbol{w}}(\sigma_{N+1}|\sigma_1,\ldots,\sigma_N). \qquad (1.8)$$

To obtain a more explicit expression, we have to insert

$$p^{(\mathrm{input})}_{\tilde{\boldsymbol{w}}}(\sigma_1,\ldots,\sigma_N) = \\ = \lim_{n\to\infty} \sum_{\tilde{\sigma}_1,\ldots,\tilde{\sigma}_{N+n-1}} p_{\tilde{\boldsymbol{w}}}(\tilde{\sigma}_{N+1},\ldots,\tilde{\sigma}_{N+n-1}|\tilde{\sigma}_1,\ldots,\tilde{\sigma}_N) \frac{1}{n} \sum_{k=0}^{n-1} \delta_{\sigma_1,\tilde{\sigma}_{k+1}} \cdots \delta_{\sigma_N,\tilde{\sigma}_{k+N}} \qquad (1.9)$$

and (1.5) with $\boldsymbol{w}$ replaced by $\tilde{\boldsymbol{w}}$ into (1.8).

Strictly speaking, (1.9) also depends on the initial values $(\tilde{\sigma}_1,\ldots,\tilde{\sigma}_N)$ of the teacher sequence, but for finite temperature and teacher weights we can expect the effects of these initial values to have finite range, leaving the input distribution independent of them. We shall come back to this point later on.

1.4 The $N = 1$ bit generator

As promised, we shall now calculate the metric and error evolution for the $N = 1$ bit generator. We could work with w^1 and $\tilde{w}^1$, but the calculations turn out to be much easier in terms of the variables

$$x = \tanh \beta w^1 \qquad \text{and} \qquad \tilde{x} = \tanh \beta \tilde{w}^1.$$

We then have

$$p_x(\sigma_k|\sigma_{k-1}) = \tfrac{1}{2} + \tfrac{1}{2}\sigma_k\sigma_{k-1}x.$$

This can straightforwardly by used to find

$$p_x(\sigma_k|\sigma_1) = \sum_{\sigma_2,\ldots,\sigma_{k-1}} \prod_{l=2}^{k} p_x(\sigma_l|\sigma_l - 1) = \tfrac{1}{2} + \tfrac{1}{2}\sigma_k\sigma_1 x^{k-1}.$$

For generic x, (that is $|x| < 1$) this reduces to a half in the large k limit, so the probability (1.9) of finding any given input value also reduces to a half: not only is the student's input distribution independent of the initial values of the teacher sequence (as expected), but — in this simple case — it is independent of the teacher's weights as well.

Inserting these results into (1.8) yields

$$g_{11}^{(x)} = \frac{1}{1-x^2}.$$

The metric in w-space can subsequently be computed using[1)]

$$g_{11}^{(w)} = \left(\frac{\mathrm{d}x}{\mathrm{d}w^1}\right)^2 g_{11}^{(x)},$$

yielding

$$g_{11}^{(w)} = \frac{\beta^2}{\cosh^2 \beta w^1}.$$

1.4.1 Learning

For simplicity, we shall consider the limit of infinite length training samples only. Given that $\tilde{\sigma}_k$ becomes uniformly distributed for large k, we find that the training error (1.7) becomes equal to the generalization error

$$\mathcal{E}_g = \tfrac{1}{2} - \tfrac{1}{2} \lim_{k\to\infty} \tfrac{1}{2} \sum_{\tilde{\sigma}_{k-1}} \sum_{\tilde{\sigma}_k} \tilde{\sigma}_k p_{\tilde{w}}(\tilde{\sigma}_k|\tilde{\sigma}_{k-1}) \sum_{\sigma_k} \sigma_k p_w(\sigma_k|\sigma_{k-1} = \tilde{\sigma}_{k-1}) = \tfrac{1}{2} - \tfrac{1}{2}\tilde{x}x \tag{1.10}$$

in this limit.

The natural gradient descent learning rule updates x according to

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\alpha g_{(x)}^{11} \frac{\mathrm{d}\mathcal{E}}{\mathrm{d}x} = \tfrac{1}{2}\alpha \frac{\tilde{x}}{1-x^2}, \tag{1.11}$$

where α is an arbitrary (but constant) learning rate.

(1.11) is easily solved, yielding $x = \tanh\left(\frac{1}{2}\alpha\tilde{x}(t-t_0)\right)$, where the integration constant t_0 is determined by the value of x at $t = 0$. Inserting this result into (1.10) yields

$$\mathcal{E}_g(t) = \tfrac{1}{2} - \tfrac{1}{2}|\tilde{x}| \tanh\left(\tfrac{1}{2}\alpha|\tilde{x}|(t-t_0)\right)$$

for the $N = 1$ bit generator with natural gradient descent learning. Asymptotically the optimal error $\mathcal{E}_\infty = \frac{1}{2} - \frac{1}{2}|\tilde{x}|$ is approached as

$$\mathcal{E}_g(t) \approx \mathcal{E}_\infty + |\tilde{x}|\mathrm{e}^{-|\tilde{x}|\alpha t}. \tag{1.12}$$

[1)] The general rule (for any number of dimensions) is:

$$g_{ij}^{(\boldsymbol{w})} = \sum_{k,l} \frac{\partial x^k}{\partial w^i} \frac{\partial x^l}{\partial w^j} g_{kl}^{(\boldsymbol{x})}.$$

This must be compared to natural gradient descent learning, which states that the weights should be updated according to

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\alpha \frac{\mathrm{d}\mathcal{E}_g}{\mathrm{d}w}.$$

As before, the calculation is most easily performed in x-space: using $\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{\mathrm{d}w}{\mathrm{d}x}\frac{\mathrm{d}x}{\mathrm{d}t}$ we obtain $\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{2}\alpha\beta^2\tilde{x}\left(1-x^2\right)^2$. Since (1.10) tells us that $x = \frac{1-2\mathcal{E}}{\tilde{x}}$, this is equivalent to

$$\frac{\mathrm{d}\mathcal{E}_g}{\mathrm{d}t} = -\frac{\alpha\beta^2}{4}\tilde{x}^2\left[1-\left(\frac{1-2\mathcal{E}_g}{\tilde{x}}\right)^2\right]^2. \tag{1.13}$$

To find the asymptotics, we make the *Ansatz* that in the limit $t \to \infty$ the optimal error $\mathcal{E}_\infty$ is again reached, and expand (1.13) in terms of $u \equiv \mathcal{E}_g - \mathcal{E}_\infty$:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{\alpha\beta^2}{4}\tilde{x}^2\left[4u/|\tilde{x}| - 4u^2/\tilde{x}^2\right]^2 = -4\alpha\beta^2u^2 + O(u^3).$$

Thus for large t:

$$\mathcal{E}_g(t) \approx \mathcal{E}_\infty + \frac{1}{4\alpha\beta^2 t} \tag{1.14}$$

for ordinary gradient descent learning.

Thus natural gradient descent learning is asymptotically dramatically faster in this case. While it is impossible to directly compare (1.12) and (1.14) for finite times because α can be chosen independently for NGD and OGD learning, a reasonable choice for fair comparison seems to set α_{NGD} equal to $\alpha\beta^2\big|_{\mathrm{OGD}}$, since this ensures that the first few NGD steps are of comparable size to those of OGD. With this choice NGD is seen to outperform OGD even for relatively short learning times.

THE $N = 2$ BIT GENERATOR

Explicit calculation of the error evolution becomes very difficult for generic N. In fact, even finding a simple expression[1)] for the metric is not easy. For this reason we concentrate on $N = 2$ in this chapter and the next. Some expeditions into $N > 2$ territory are described in chapter four.

As before a change of variables makes the computation much easier. In this case we pick

$$x^{\pm} = \tanh\beta(w^1 \pm w^2).$$

In terms of these (1.6) becomes

$$p_{\boldsymbol{x}}(\sigma_m|\sigma_{m-2},\sigma_{m-1}) = \tfrac{1}{2} + \tfrac{1}{2}\sigma_m\sigma_{m-1}x^{\sigma_{m-1}\sigma_{m-2}}, \tag{2.1}$$

where we write $x^{\tau} \equiv x^{+}$ if $\tau = 1$, or x^{-} if $\tau = -1$.

Before we can calculate the error dynamics, or even the metric, we must know the student's input distribution, which is determined by the teacher bit sequence.

2.1 Input distribution

The input distribution is derived from (1.9) for $N = 2$:

$$p^{(\mathrm{input})}(\sigma_1,\sigma_2) = \lim_{n\to\infty} \sum_{\tilde{\sigma}_1,\ldots,\tilde{\sigma}_n+1} p_{\tilde{\boldsymbol{x}}}(\tilde{\sigma}_3,\ldots,\tilde{\sigma}_{n+1}|\tilde{\sigma}_1,\tilde{\sigma}_2)\frac{1}{n}\sum_{m=0}^{n-1}\delta_{\sigma_1,\tilde{\sigma}_{m+1}}\delta_{\sigma_2,\tilde{\sigma}_{m+2}},$$

which — using $\delta_{\sigma,\sigma'} = \frac{1}{2} + \frac{1}{2}\sigma\sigma'$ — may be written as

$$\begin{aligned} p^{(\mathrm{input})}(\sigma_1,\sigma_2) &= \lim_{n\to\infty}\frac{1}{n}\sum_{m=0}^{n-1}\mathrm{E}[(\tfrac{1}{2}+\tfrac{1}{2}\sigma_1\tilde{\sigma}_{m+1})(\tfrac{1}{2}+\tfrac{1}{2}\sigma_2\tilde{\sigma}_{m+2})] \\ &= \lim_{n\to\infty}\frac{1}{4n}\sum_{m=0}^{n-1}\Big(1+\sigma_1\mathrm{E}[\tilde{\sigma}_{m+1}]+\sigma_2\mathrm{E}[\tilde{\sigma}_{m+2}]+\sigma_1\sigma_2\mathrm{E}[\tilde{\sigma}_{m+1}\tilde{\sigma}_{m+2}]\Big). \end{aligned} \tag{2.2}$$

The (somewhat technical) calculation of $\mathrm{E}[\tilde{\sigma}_m]$ and $\mathrm{E}[\tilde{\sigma}_m\tilde{\sigma}_{m+1}]$ can be found in the appendix to this chapter. It results in:

$$p^{(\mathrm{input})}(\sigma_1,\sigma_2) = \frac{1}{4} + \frac{1}{4}\sigma_1\sigma_2\frac{\tilde{y}^{+}}{1-\tilde{y}^{-}}, \tag{2.3}$$

where $\tilde{y}^{\pm} = \frac{1}{2}(\tilde{x}^{+} \pm \tilde{x}^{-})$, $\tilde{x}^{\pm} = \tanh\beta(\tilde{w}^1 \pm \tilde{w}^2)$, and $\tilde{w}^i$ are the teacher weights.

Note that (2.3) does not depend on the teacher's initial values, but *does* depend on the teacher's weights, unlike before.

[1)] That is, an expression without any explicit summations over spin variables left in it.

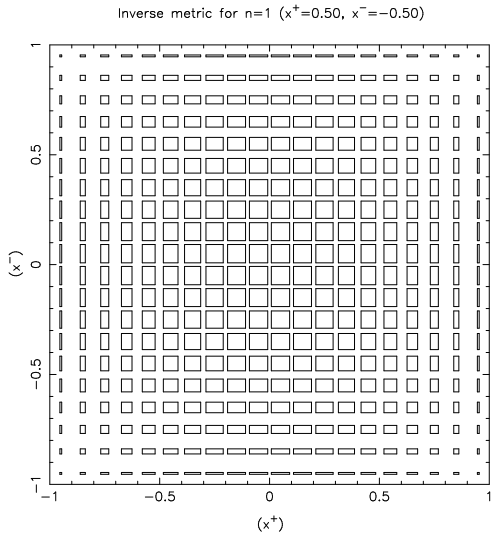

Figure 2.1: A graphical representation of the metric in $x^{\pm}$-space: the width and height of the boxes are proportional to the components g^{++} and g^{--} of the inverse metric respectively. Thus the width (height) of the boxes equals the change in x^+ (x^-) that corresponds to a step of fixed Riemannian length. Note that this representation is possible due to the fact that $g^{\tau\tau'}$ is diagonal, which ensures that the remaining components are positive.

2.2 Metric

With the input distribution done, calculating the metric is straightforward:

$$\begin{aligned} g^{(\boldsymbol{x})}_{\tau\tau'} &= \sum_{\sigma_1,\sigma_2} \left(\frac{1}{4} + \frac{\sigma_1\sigma_2}{4}\frac{\tilde{y}^+}{1-\tilde{y}^-}\right) \sum_{\sigma_3} p(\sigma_3|\sigma_1,\sigma_2) \frac{\partial \log p(\sigma_3|\sigma_1,\sigma_2)}{\partial x^\tau} \frac{\partial \log p(\sigma_3|\sigma_1,\sigma_2)}{\partial x^{\tau'}} \\ &= \left\langle \left(1+\sigma_1\sigma_2\frac{\tilde{y}^+}{1-\tilde{y}^-}\right)(1+\sigma_2\sigma_3 x^{\sigma_1\sigma_2}) \frac{\sigma_2\sigma_3\delta_{\sigma_1\sigma_2,\tau}}{1+\sigma_2\sigma_3 x^\tau} \frac{\sigma_2\sigma_3\delta_{\sigma_1\sigma_2,\tau'}}{1+\sigma_2\sigma_3 x^{\tau'}} \right\rangle_{\sigma_1,\sigma_2,\sigma_3} \\ &= \frac{\delta_{\tau\tau'}}{1-(x^\tau)^2}\left[\tfrac{1}{2} + \tfrac{1}{2}\tau\frac{\tilde{y}^+}{1-\tilde{y}^-}\right]. \end{aligned} \tag{2.4}$$

A graphical representation of this metric is drawn in figure 2.1. It is clearly seen that near the boundaries $x^+ = \pm 1$ and $x^- = \pm 1$ an infinitesimal step in coordinate space corresponds to a much larger Riemannian distance than near the centre. This reflects the fact that it is impossible for $|x^\pm|$ to be larger than one: the latter would correspond to non-physical values for w^i. Another clearly visible aspect of this metric is the fact that g^{++} depends only on x^+ while g^{--} depends only on x^-.

Because of the way the teacher weights occur in (2.4), the metric for different teachers looks like the one depicted in figure 2.1, but with the widths and heights of the boxes globally scaled by $1 \pm \frac{\tilde{y}^+}{1-\tilde{y}^-}$ respectively.

In textbooks on general relativity, one sometimes finds the curvature of space depicted as a potential plot: with the potential of a test particle plotted on an extra axis, the bending of trajectories is beautifully illustrated. Unfortunately, this option is not available here, since there exists no potential field that corresponds to the metric (2.4).

2.3 Natural gradient descent learning

In the limit of infinite training sample size, the training error (which then equals the generalization error) may be computed as the probability that the student incorrectly predicts a spin from the teacher sequence, as in the case $N = 1$ (cf (1.10)):

$$\begin{aligned}\mathcal{E}_g &= \tfrac{1}{2} - \tfrac{1}{2}\lim_{n\to\infty} \sum_{\tilde\sigma_{n-2},\tilde\sigma_{n-1}} p^{(\mathrm{input})}(\tilde\sigma_{n-2},\tilde\sigma_{n-1}) \times \\ &\qquad \times \sum_{\tilde\sigma_n} p_{\tilde x}(\tilde\sigma_n|\tilde\sigma_{n-2},\tilde\sigma_{n-1}) \sum_{\sigma_n} p_x(\sigma_n|\sigma_{n-2}=\tilde\sigma_{n-2},\sigma_{n-1}=\tilde\sigma_{n-1}) \\ &= \frac{1}{2} - \frac{1}{4}\left[(\tilde x^+ x^+ + \tilde x^- x^-) + \frac{\tilde y^+}{1-\tilde y^-}(\tilde x^+ x^+ - \tilde x^- x^-)\right]. \end{aligned} \tag{2.5}$$

Taking the derivatives with respect to $x^\pm$ is trivial, so the evolution of $x^\pm$ can now be computed:

$$\frac{\mathrm{d}x^\tau}{\mathrm{d}t} \equiv -\alpha \sum_{\tau'} g^{\tau\tau'}_{(x)} \frac{\partial \mathcal{E}_t}{\partial_{x^{\tau'}}} = \tfrac{1}{2}\alpha\tilde x^\tau \left(1-(x^\tau)^2\right).$$

This is easily solved:

$$x^\tau(t) = \tanh(\tfrac{1}{2}\alpha\tilde x^\tau(t-t_0^\tau)), \tag{2.6}$$

where the integration constants $t_0^\pm$ incorporate the initial values of $x^\pm$. Note, by the way, that the evolution of x^+ and x^- are completely independent.

Combining (2.6) with (2.5) gives the error as a function of learning time. For large t this function approaches

$$\mathcal{E}_g(t) \approx \mathcal{E}_\infty + \mathcal{A}_+ \mathrm{e}^{-\alpha|\tilde x^+|t} + \mathcal{A}_- \mathrm{e}^{-\alpha|\tilde x^-|t}, \tag{2.7}$$

where

$$\begin{aligned}\mathcal{E}_\infty &= \frac{1}{2} - \frac{1}{4}\left(1+\frac{\tilde y^+}{1-\tilde y^-}\right)|\tilde x^+| - \frac{1}{4}\left(1-\frac{\tilde y^+}{1-\tilde y^-}\right)|\tilde x^-| = \\ &= \frac{1}{2}\left(1 - \frac{1+\tilde x^-}{2-(\tilde x^+ + \tilde x^-)}|\tilde x^+| - \frac{1-\tilde x^+}{2-(\tilde x^+ + \tilde x^-)}|\tilde x^-|\right)\end{aligned}$$

is the global minimum of (2.5) for fixed teacher weights, and $\mathcal{A}_\pm = \frac{1\pm\tilde x^\mp}{2-(\tilde x^+ + \tilde x^-)}|\tilde x^\pm|\mathrm{e}^{\alpha|\tilde x^\pm|t_0^\pm}$. The error approaches its minimum value exponentially fast.

The asymptotic behaviour of $\mathcal{E}_g$ is depicted in figure 2.2. The general shape of the left hand side picture is due to the fact that a very noisy teacher (small weights, so both $\tilde x^+$ and $\tilde x^-$ small) cannot be emulated very well. On the right it can be seen that the learning time is governed by the smaller of $\tilde x^+$ and $\tilde x^-$.

NGD learning times should be compared with the result of ordinary gradient descent, which we compute next.

2.4 $N = 2$ bit generator with ordinary gradient descent

Even though we have to find the evolution of the error governed by

$$\frac{\mathrm{d}w^i}{\mathrm{d}t} \equiv -\alpha\frac{\partial \mathcal{E}_t}{\partial w^i}, \tag{2.8}$$

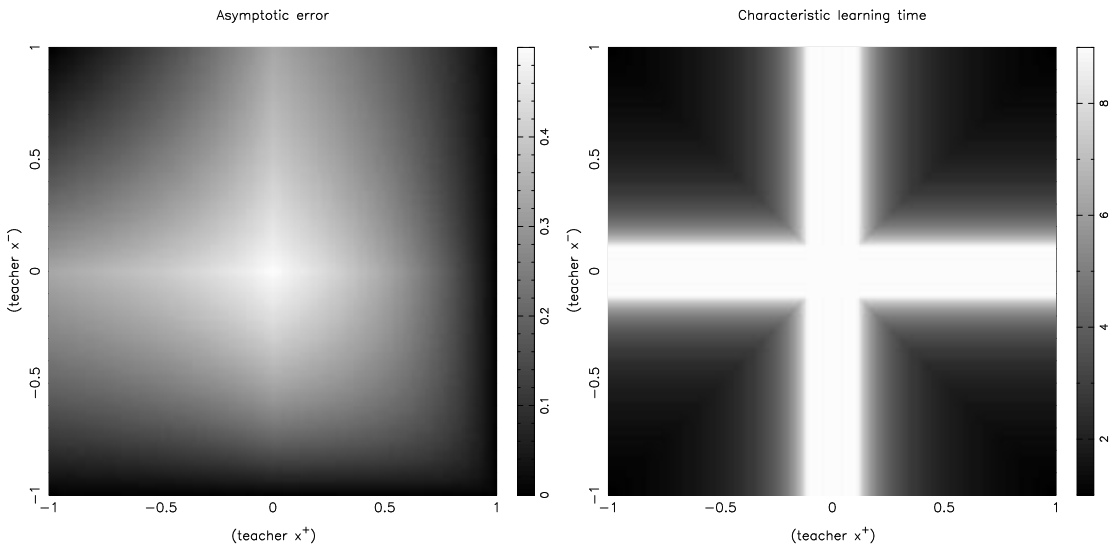

Figure 2.2: The asymptotic generalization error (left) and characteristic learning time (right) for NGD learning of the $N = 2$ bit generator. (The characteristic time is the constant τ in $\mathcal{E}_g - \mathcal{E}_\infty \sim \mathrm{e}^{-t/\tau}$.)

we still choose to work in terms of $x^\pm$, since (2.5) provides such a simple expression for the generalization error in terms of the parameters. The resulting differential equations

$$\frac{\mathrm{d}x^\tau}{\mathrm{d}t} = -\alpha \sum_{\tau'} \sum_i \frac{\partial x^\tau}{\partial w^i} \frac{\partial x^{\tau'}}{\partial w^i} \frac{\partial \mathcal{E}_t}{\partial x^{\tau'}} = \tfrac{1}{2}\alpha\beta^2 \left[1 - (x^\tau)^2\right]^2 \tilde{x}^\tau \left(1 + \tau \frac{\tilde{y}^+}{1-\tilde{y}^-}\right)$$

can be integrated:

$$\alpha\beta^2 \tilde{x}^\tau \left(1 + \tau \frac{\tilde{y}^+}{1-\tilde{y}^-}\right)(t - t_0^\tau) = \frac{x^\tau}{1-(x^\tau)^2} + \operatorname{atanh} x^\tau,$$

but this does not yield closed expressions for x^τ in terms of t. However, we may proceed by expanding in $u^\tau \equiv 1 - \operatorname{sgn}(\tilde{x}^\tau) x^\tau$. In terms of $u^\pm$, we may write the error (2.5) as

$$\mathcal{E}_t = \mathcal{E}_\infty + \frac{1}{4} \sum_\tau \left(1 + \tau \frac{\tilde{y}^+}{1-\tilde{y}^-}\right) |\tilde{x}^\tau| u^\tau. \tag{2.9}$$

With $\left[1 - (x^\tau)^2\right]^2 = 4(u^\tau)^2 + O((u^\tau)^3)$, we find that $u^\pm$ evolve according to

$$\frac{\mathrm{d}u^\tau}{\mathrm{d}t} = -2\alpha\beta^2 |\tilde{x}^\tau| \left(1 + \tau \frac{\tilde{y}^+}{1-\tilde{y}^-}\right) (u^\tau)^2 + O((u^\tau)^3), \tag{2.10}$$

which is easily solved. Inserting the solution into (2.9) yields

$$\mathcal{E}_t(t) \approx \mathcal{E}_\infty + \frac{1}{4\alpha\beta^2 t} \tag{2.11}$$

to lowest order.

As in the case $N = 1$, natural gradient descent dramatically outperforms ordinary gradient descent when the learning rates are chosen in such a way that the initial steps are of comparable size. (i.e. setting $\alpha\beta^2\big|_{\mathrm{OGD}} \sim \alpha_{\mathrm{NGD}}$.)

2.A Calculations

The calculations that were skipped in §2.1 are presented here.

2.A.1 The expectation value of $\tilde{\sigma}_m$

Dropping the tildes for the moment, we have

$$\mathrm{E}[\sigma_m] = \left\langle \left[\prod_{j=3}^{n+1} (1 + \sigma_j \sigma_{j-1} x^{\sigma_{j-1}\sigma_{j-2}}) \right] \sigma_m \right\rangle_{\sigma_3,\ldots,\sigma_m},$$

where $x^{\pm}$ and σ_1, σ_2 are free parameters.

It is clear that the averages over σ_j's with j larger than m can be performed straight away. After explicit summation of σ_m we are left with

$$\mathrm{E}[\sigma_m] = \left\langle \left[\prod_{j=3}^{m-1} (1 + \sigma_j \sigma_{j-1} x^{\sigma_{j-1}\sigma_{j-2}}) \right] \sigma_{m-1} x^{\sigma_{m-1}\sigma_{m-2}} \right\rangle_{\sigma_3,\ldots,\sigma_{m-1}}.$$

Summing out σ_{m-1} as well yields

$$\begin{aligned} \mathrm{E}[\sigma_m] = {} & y^+ \left\langle x^{\sigma_{m-2}\sigma_{m-3}} \sigma_{m-2} \prod_{k=3}^{m-2} (1 + \sigma_k \sigma_{k-1} x^{\sigma_{k-1}\sigma_{k-2}}) \right\rangle_{\sigma_3,\ldots\sigma_{m-2}} + \\ & + y^- \left\langle \sigma_{m-2} \prod_{k=3}^{m-2} (1 + \sigma_k \sigma_{k-1} x^{\sigma_{k-1}\sigma_{k-2}}) \right\rangle_{\sigma_3,\ldots\sigma_{m-2}}, \end{aligned}$$

where $y^{\pm} = \frac{1}{2}(x^+ \pm x^-)$ was introduced for convenience. We note that the second term contains $\mathrm{E}[\sigma_{m-2}]$, while the first can be expanded recursively. Thus we arrive at

$$\mathrm{E}[\sigma_m] = y^- \sum_{k=2}^{m-2} (y^+)^{m-2-k} \mathrm{E}[\sigma_k] + \sigma_2 x^{\sigma_1\sigma_2} (y^+)^{m-3}. \tag{2.12}$$

To solve this recursion relation, we first introduce $f_m = (y^+)^{-m}\mathrm{E}[\sigma_m]$ so we are left to solve

$$f_m = a_1 \sum_{k=2}^{m-2} f_k + a_2,$$

where $a_1 = \frac{y^-}{(y^+)^2}$ and $a_2 = \frac{\sigma_2 x^{\sigma_1\sigma_2}}{(y^+)^3}$. Then we introduce $g_m = \sum_{k=2}^{m-2} f_m$ (so $f_m = g_{m+2} - g_{m+1}$), to obtain the much easier recursion relation

$$g_{m+2} = g_{m+1} + a_1 g_m + a_2. \tag{2.13}$$

The annoying constant a_2 can be absorbed by introducing $h_m = g_m + \frac{a_2}{a_1}$. The resulting $h_{m+2} = h_{m+1} + a_1 h_m$ can be solved using the *Ansatz* $h_m = c^m$ (that's c to the power m): this gives $c^2 = c + a_1$, so $c = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 + 4a_1}$, where we define $\sqrt{a} = i\sqrt{-a}$ for negative real a. The full solution must be a linear combination of these two. Therefore we have

$$h_m = A\left[\tfrac{1}{2} + \tfrac{1}{2}\sqrt{1 + 4a_1}\right]^m + B\left[\tfrac{1}{2} - \tfrac{1}{2}\sqrt{1 + 4a_1}\right]^m,$$

where A and B are to be computed from the initial conditions, which can be written as $g_3 = 0$ and $g_4 = \frac{\sigma_2}{(y^+)^2}$.

If one does the algebra carefully, one finds

$$\begin{aligned}\mathrm{E}[\sigma_m] = {} & \tfrac{1}{2}\sigma_2 x^{\sigma_1\sigma_2}(y^+)^{m-3}\left[\left(\tfrac{1}{2}+\tfrac{1}{2}\eta\right)^{m-3}+\left(\tfrac{1}{2}-\tfrac{1}{2}\eta\right)^{m-3}\right] \\ & +\tfrac{1}{2}\sigma_2\left(x^{\sigma_1\sigma_2}+2\frac{y^-}{y^+}\right)\frac{(y^+)^{m-3}}{\eta}\left[\left(\tfrac{1}{2}+\tfrac{1}{2}\eta\right)^{m-3}-\left(\tfrac{1}{2}-\tfrac{1}{2}\eta\right)^{m-3}\right], \qquad (2.14)\end{aligned}$$

where $\eta \equiv \sqrt{1+4\frac{y^-}{(y^+)^2}}$, in which we still use $\sqrt{a} \equiv i\sqrt{-a}$ for negative real a. It should be noted that although η may be imaginary if $4y^- < -(y^+)^2$, the resulting value for $\mathrm{E}[\sigma_m]$ is always real, since the first 'square bracket'-term yields only even powers of η, while the second one yields only odd powers, which cancel the η in the denominator.

Another property of (2.14) which may be slightly less obvious, is that for large $\mathrm{E}[\sigma_m]$ vanishes for large m: this is seen from the fact that

$$\left|y^+\left(\tfrac{1}{2}\pm\tfrac{1}{2}\eta\right)\right| \leq 1,$$

for any possible value of $(x^\pm) \in [-1,+1]^2$, with equality only if $x^+ = 1$ or $x^- = -1$. Thus $\lim_{n\to\infty}\frac{1}{n}\sum_{m=0}^{n-1}\mathrm{E}[\tilde{\sigma}_m]$ vanishes irrespective of σ_1 and σ_2 for any finite $\boldsymbol{w}$.

2.A.2 The expectation value of $\tilde{\sigma}_m\tilde{\sigma}_{m-1}$

With the same notation as above we may write

$$\mathrm{E}[\sigma_m\sigma_{m-1}] = \left\langle\left[\prod_{j=3}^{m}(1+\sigma_j\sigma_{j-1}x^{\sigma_{j-1}\sigma_{j-2}})\right]\sigma_m\sigma_{m-1}\right\rangle_{\sigma_3,\ldots,\sigma_m}.$$

Again, averages over σ_j's with j larger than m can be performed straight away. This time explicit summation of σ_m leaves us with

$$\mathrm{E}[\sigma_m\sigma_{m-1}] = \left\langle\left[\prod_{j=3}^{m-1}(1+\sigma_j\sigma_{j-1}x^{\sigma_{j-1}\sigma_{j-2}})\right]x^{\sigma_{m-1}\sigma_{m-2}}\right\rangle_{\sigma_3,\ldots,\sigma_{m-1}}.$$

The rest is easy: starting at $j = m-1$ each σ_j may be summed out in turn, and we end up with

$$\mathrm{E}[\sigma_m\sigma_{m-1}] = y^+\frac{1-(y^-)^{m-3}}{1-y^-} + (y^-)^{m-3}x^{\sigma_1\sigma_2}.$$

For finite weights the second term falls of to zero as m grows, so

$$\lim_{n\to\infty}\frac{1}{n}\sum_{m=0}^{n-1}\mathrm{E}[\tilde{\sigma}_m\tilde{\sigma}_{m+1}] = \frac{y^+}{1-y^-},$$

independent of initial conditions.

Finally, reinstating tildes and inserting the results in (2.2) yields

$$p^{(\mathrm{input})}(\sigma_1,\sigma_2) = \frac{1}{4}+\frac{1}{4}\sigma_1\sigma_2\frac{\tilde{y}^+}{1-\tilde{y}^-}.$$

(Recall that $\tilde{y}^\pm = \frac{1}{2}(\tilde{x}^+ \pm \tilde{x}^-)$, where $x^\pm = \tanh\beta(\tilde{w}^1 \pm \tilde{w}^2)$ and $\tilde{w}^i$ are the teacher weights.)

As expected, the result is independent of the teacher's initial values.

PREDICTING FURTHER INTO THE FUTURE FOR $N = 2$

3.1 Generalized definition of training error

Until now the student's task was to learn how to predict a single future bit from knowledge about the past. More generally, we may be interested in a system that can give predictions for a larger number of future bits, where we may attach a different weighting to each of those predictions. Training such a system can be done by gradient descent on the following error measure:

$$\mathcal{E}_g[\boldsymbol{w};\tilde{\boldsymbol{w}}] = \tfrac{1}{2} - \tfrac{1}{2} \sum_{\sigma_1,\ldots,\sigma_N} p^{(\mathrm{input})}(\sigma_1,\ldots,\sigma_N) \sum_{\sigma_{N+1},\ldots,\sigma_{N+n}} p_{\boldsymbol{w}}(\sigma_{N+1},\ldots,\sigma_{N+n}|\sigma_1,\ldots,\sigma_N) \times$$
$$\times \sum_{\tilde{\sigma}_{N+1},\ldots,\sigma_{N+n}} p_{\tilde{\boldsymbol{w}}}(\tilde{\sigma}_{N+1},\ldots,\tilde{\sigma}_{N+n}|\sigma_1,\ldots,\sigma_N) \sum_{k=1}^{n} a_k \sigma_{N+k}\tilde{\sigma}_{N+k}, \qquad (3.1)$$

where $\sum a_k$ must equal one for proper normalization. We shall only consider $a_k = 1/n$, and stick to $N = 2$. Even in this regime the error function is a lot more complex than for $n = 1$ as studied in the previous chapter.

Whereas the generalization error for $n = 1$ (2.5) only has one minimum, which is located at the corner of $x^{\pm}$-space closest to the teacher parameters, (3.1) can have several local minima depending on the value of the teacher weights. Additionally, the global minimum may not lie at a corner for certain combinations of teacher weights. Figure 3.1 shows in which regions of teacher weight space these phenomena occur for $n = 2$.

The presence of non-global minima complicate the learning process considerably, as we shall see below. It is worth noting that the global minimum always lies at a corner of (x^+, x^-)-space, that is both student weights grow without bounds, totally suppressing any noise — except when one of $\tilde{x}^{\pm}$ is near zero. The latter case corresponds to a teacher with $\tilde{w}^1 \approx \mp\tilde{w}^2$. ($\tilde{x}^{\pm}$ remains zero even for low temperature if $\tilde{w}^1 = \mp\tilde{w}^2$.)

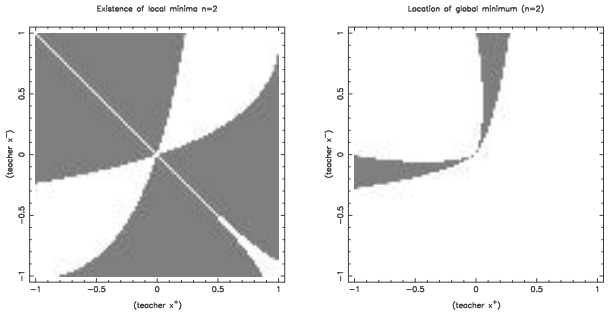

Figure 3.1: The generalization error for $N = 2$, $n = 2$ has local minima for most combinations of teacher weights (grey areas in LHS picture). The grey patches in the RHS picture indicate areas where the global minimum is found at an edge rather than at a corner of $x^{\pm}$-space.

3.2 Metric for $n > 1$

When n future bits are being predicted, we must clearly define the metric as the Fisher information on the probability distribution $p_{w;\tilde{w}}(\sigma_1, \sigma_2; \sigma_3, \ldots, \sigma_{2+n})$:

$$g^{(x;n)}_{\tau\tau'} = \sum_{\sigma_1,\sigma_2} p^{(\text{input})}_{\tilde{w}}(\sigma_1,\sigma_2) \sum_{\sigma_3,\cdots,\sigma_{2+n}} p_w(\sigma_3,\ldots,\sigma_{2+n}|\sigma_1,\sigma_2) \times \\ \times \frac{\partial \log p_w(\sigma_3,\ldots,\sigma_{2+n}|\sigma_1,\sigma_2)}{\partial x^\tau} \frac{\partial \log p_w(\sigma_3,\ldots,\sigma_{2+n}|\sigma_1,\sigma_2)}{\partial x^{\tau'}}. \quad (3.2)$$

After a tedious calculation which is included as an appendix at the end of this chapter this leads to

$$g^{(x;n)}_{\tau\tau'} = \frac{\delta_{\tau\tau'}}{1-(x^\tau)^2}\left[\frac{n}{2}\left(1+\tau\frac{y^+}{1-y^-}\right)+\frac{\tau}{2}\left(\frac{\tilde{y}^+}{1-\tilde{y}^-}-\frac{y^+}{1-y^-}\right)\frac{1-(y^-)^n}{1-y^-}\right]. \quad (3.3)$$

(Recall that $y^\pm$ and $\tilde{y}^\pm$ are just shorthands: $y^\pm \equiv \frac{1}{2}(x^+ \pm x^-)$; $\tilde{y}^\pm \equiv \frac{1}{2}(\tilde{x}^+ \pm \tilde{x}^-)$.)

In figure 3.2 I attempt to visualize this metric for several values of n and teacher weights. The projection technique is the same as for figure 2.1.

3.3 Simulations

I simulated NGD and OGD learning for the $N = 2$ bit generator for a wide range of teacher weights. In each case the student was initialized with very small random weights. The results

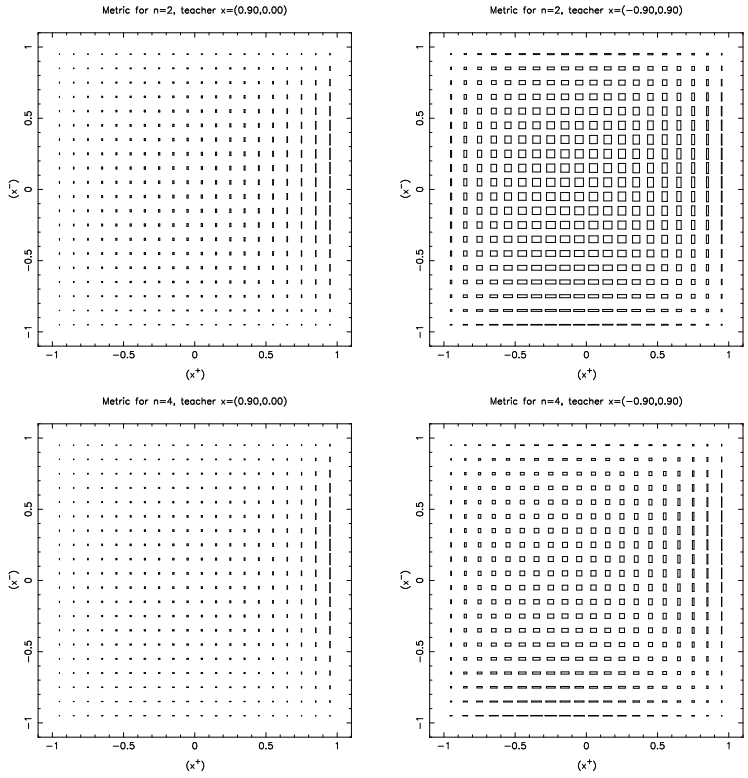

Figure 3.2: Metrics for bit generators with $N = 2$. Top row: predicting two spins. Bottom row: predicting four spins. The left hand side pictures show the metric for teachers ($\tilde{x}^+ = .9$, $\tilde{x}^- = 0$). On the right the teachers are ($\tilde{x}^+ = -.9, \tilde{x}^- = .9$).

are shown in figures 3.3 through 3.6. For NGD learning, I fitted the achieved error as a function of learning time to $\mathcal{E}_g(t) = \mathcal{E}_\infty + c_0 \mathrm{e}^{-t/\tau}$. The characteristic time τ is shown in the figures. OGD learning achievements were fitted to $\mathcal{E}_g(t) = \mathcal{E}_\infty + (t/T)^{-a}$. Both the exponent a and the characteristic time T are shown in the figures. $\mathcal{E}_\infty$ is the the same as for NGD, except in areas where either of the students gets stuck in a local minimum (see below).

As for $n = 1$ it is clear that NGD learning of noisy teachers (centre of the pictures) is more difficult than when the teacher is almost noise free (near the corners). However, the characteristic time is no longer simply inversely proportional to the smaller of $\tilde{x}^\pm$.

Perhaps more interesting is the fact that for some values of the teacher eights, a local minimum occurs in the generalization error at $\tilde{x}^- = -1$, $\tilde{x}^+ = +1$. Depending on the extent of the domain of attraction this minimum, there can be two effects: either the student may get stuck in it, or it may take a long time hovering on the boundaries. For example, figure 3.7 shows an NGD student poised on the edge of the local minimum while trying to mimic a teacher with $\tilde{x}^+ = -.85$ and $\tilde{x}^- = -.99$. It is worth noting that both before and after the 'plateau', the generalization error decreases exponentially with time.

For slightly larger values of $\tilde{x}^+$, the NGD student actually gets stuck in the local minimum (figure 3.8). In this area, the OGD student does find the global minimum, and thus outperforms the NGD student in the limit of infinite learning time, although the small difference between

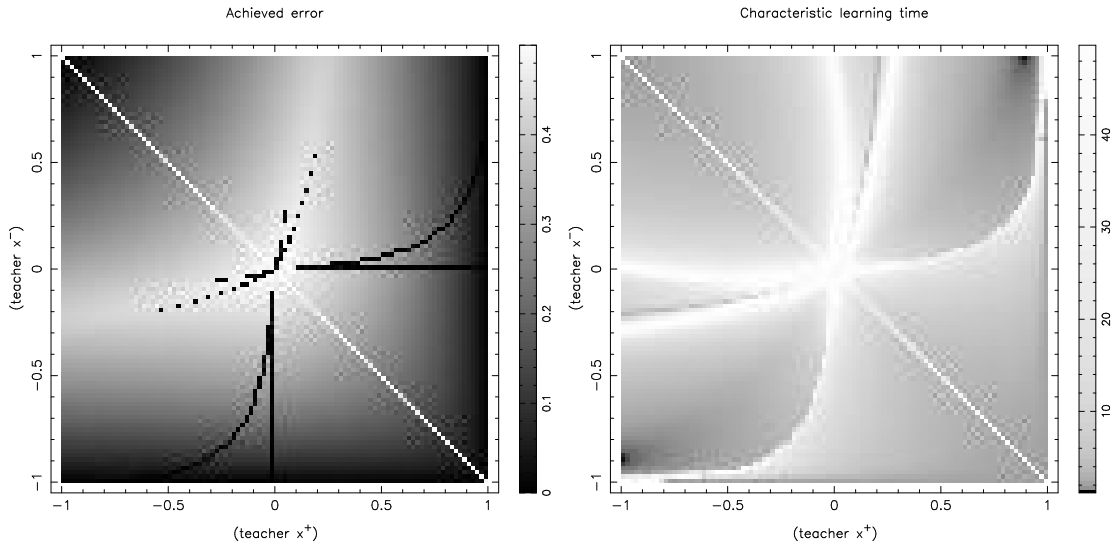

Figure 3.3: Asymptotic error and characteristic time for $N = 2$ bit generator predicting 2 future spins using NGD learning. The white line through the centre is due to numerical instability at $\tilde{x}^+ = -\tilde{x}^-$. The areas enclosed by black curves are those where the student does not find the global minimum. The near invisibility of the jump on the edges of this area indicates that the difference between the true minimum and the achieved error is quite small. In areas where local minima exist, but the student did find the global minimum, the plotted characteristic learning time (right) may not be very meaningful: the fit is rather poor due to the plateau phase (see figure 3.7). This is especially true in regions where the local minimum was only barely avoided (eg. along the white curves near the bottom left and near the top right of the RHS picture). Note that the grey scale on the right hand side is logarithmic, unlike the one in figure 2.2.

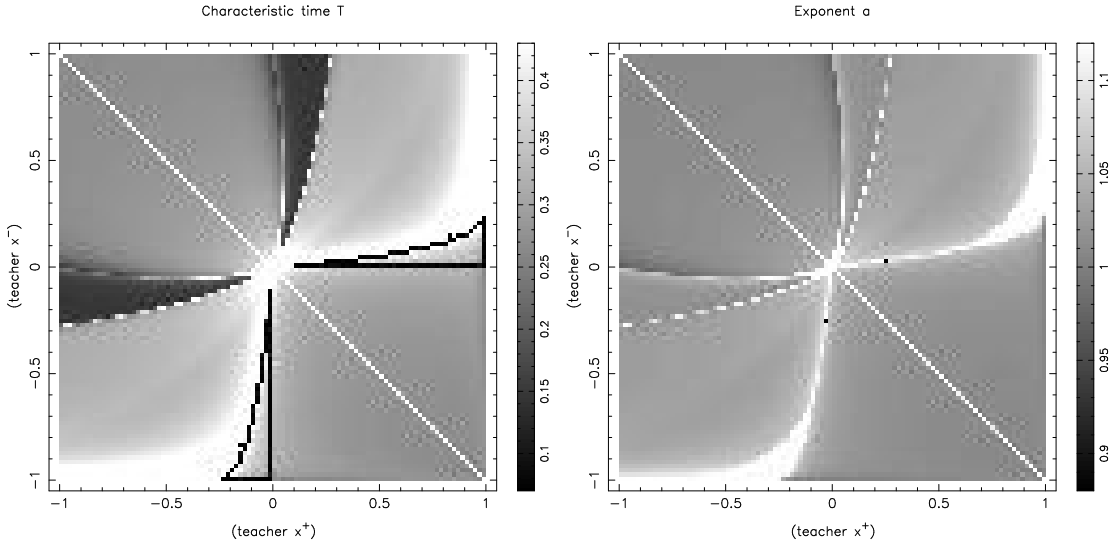

Figure 3.4: Typical learning time (left, see text) and exponent for $N = 2$ bit generator predicting 2 future spins using OGD learning. Again, the white lines through the centre are not physical. The large white patches near the edges are caused by the student needing a lot of time to avoid the local minimum — see figure 3.8. The precision of the simulations is insufficient to establish that the asymptotic exponent is different from one. The black lines indicate areas of teacher weight space for which the student fails to find the true minimum.

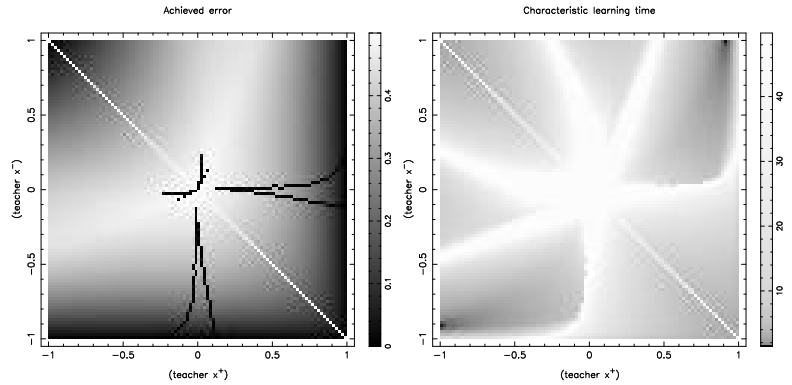

Figure 3.5: Asymptotic error and critical time for $N = 2$ bit generator predicting 4 future spins using NGD learning. Comments below figure 3.3 apply.

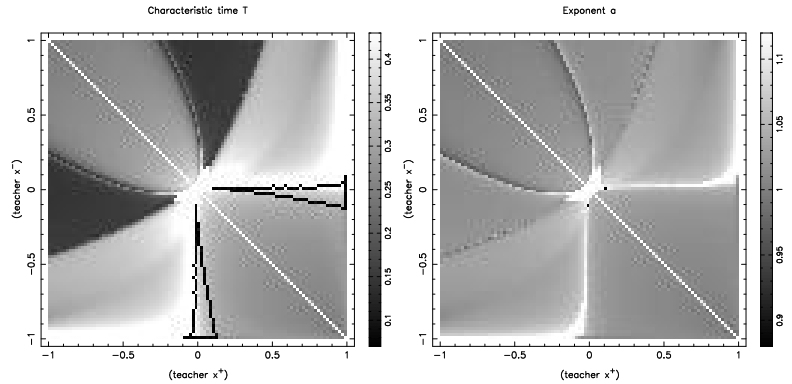

Figure 3.6: Typical learning time (far left, see text) and exponent (right) for $N = 2$ bit generator predicting 4 future spins using OGD learning.

the local and the global maximum means the for intermediate values of learning time the NGD student still does better. This can be seen from the graph in figure 3.8: the generalization error of the NGD student relative to the optimal error $\mathcal{E}_\infty$ is shown in grey, with the time scales

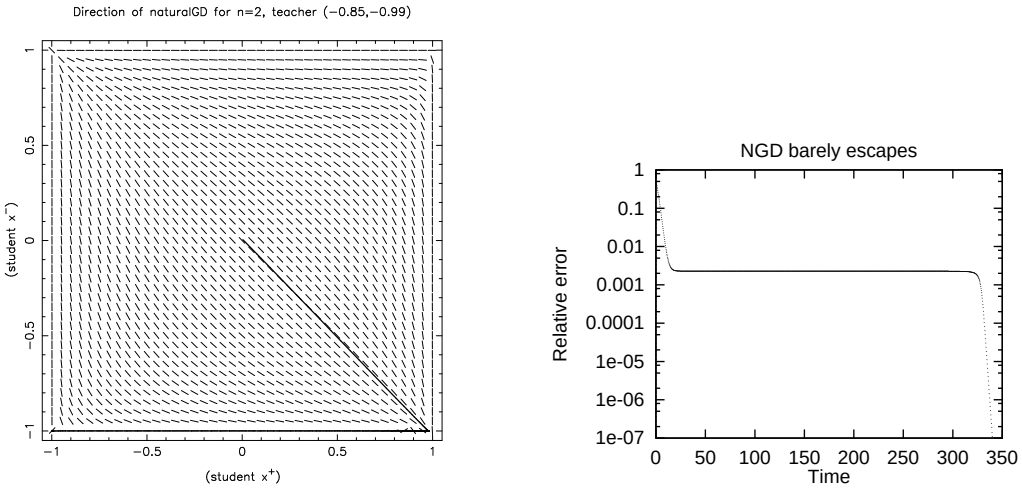

Figure 3.7: The path of an NGD student may pass close by a local minimum (left). In the plot of $\log(\mathcal{E}_g - \mathcal{E}_\infty)$ versus learning time (right) this results in a plateau. Once the plateau has been left, exponentially fast learning resumes. The arrows in the LHS picture show the direction of NGD only. Their lengths have been normalized to one.

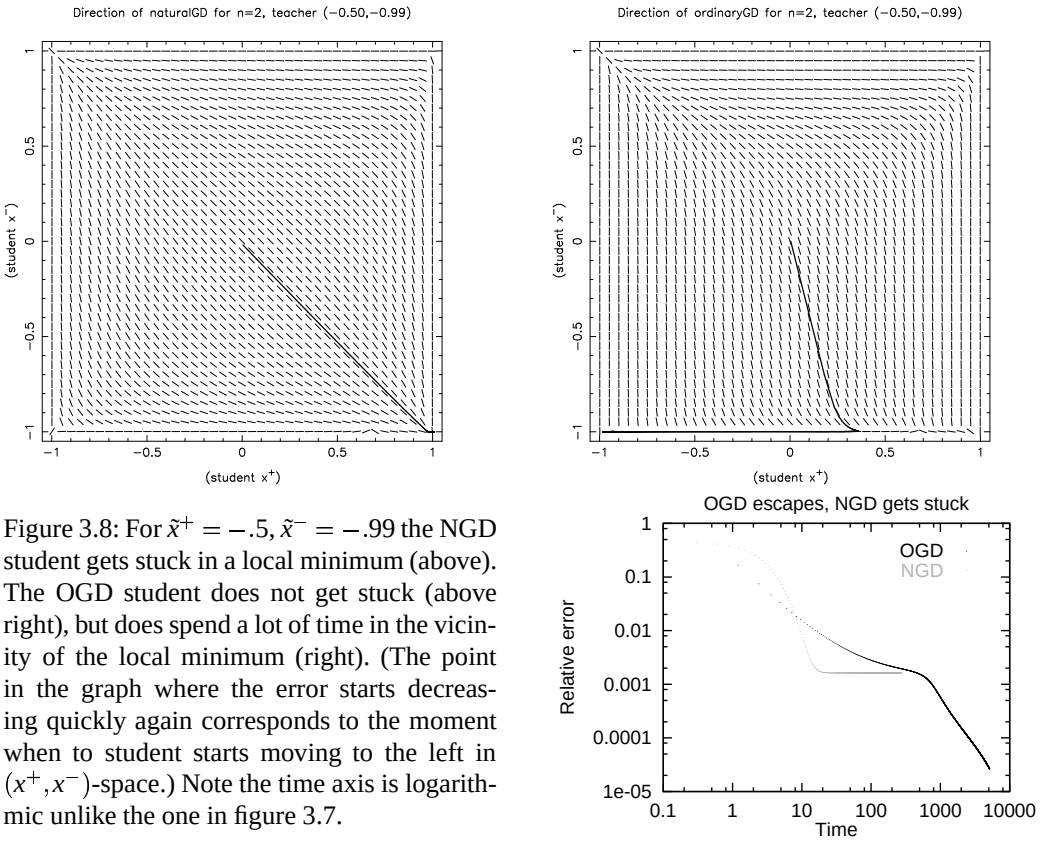

Figure 3.8: For $\tilde{x}^+ = -.5, \tilde{x}^- = -.99$ the NGD student gets stuck in a local minimum (above). The OGD student does not get stuck (above right), but does spend a lot of time in the vicinity of the local minimum (right). (The point in the graph where the error starts decreasing quickly again corresponds to the moment when to student starts moving to the left in (x^+, x^-)-space.) Note the time axis is logarithmic unlike the one in figure 3.7.

set according to $\alpha_{\mathrm{NGD}} = \alpha\beta^2\big|_{\mathrm{OGD}}$. It is important to note that the time scales can be varied arbitrarily by changing the learning rates: setting α_{NGD} to a lower value reduces the time during which NGD has a lower error. Conversely, it is possible to increase α_{NGD} up to a point where NGD does better for small values of time as well.

For larger values of $\tilde{x}^+$ the OGD student does not find the global minimum at $\tilde{x}^+ = \tilde{x}^- = -1$ either. In the graphs 3.3 through 3.6 black curves are drawn around areas in teacher weight space in which students get stuck in the local minimum.

The existence of teacher weight values for which OGD finds the global minimum while NGD does not, may seem somewhat surprising, but can be understood as follows: the boundary between two domains of attraction is the curve of steepest ascent from the saddle point that separates them. (See figure 3.9.) With a different choice of metric, local concepts of distance change, and therefore so does the notion of steepness. This means that boundaries between domains of attraction shift under such a change. In our case, these changes work out such that NGD students are more easily trapped in the local minimum, but we have no evidence as to whether this is a more general property of natural gradient descent: in other cases it could just as well be the other way round. The important lesson is that NGD and OGD do *not* necessarily prescribe the same trajectories through weight space.

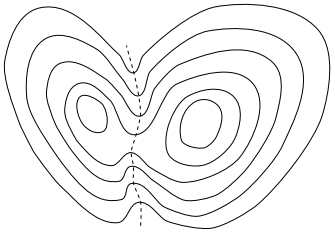

Figure 3.9: Two minima with the boundary between their domains of attraction. Deforming the space (changing the metric) moves this boundary.

3.A Calculations

The calculations that were skipped in the main text are presented here.

3.A.1 The metric for $N = 2$ and general n

The following calculation leads to (3.3).

Insert (2.3) and (1.5) with (2.1) into (3.2). This yields:

$$
\begin{aligned}
g_{\tau\tau'}^{(x;n)} = \Bigg\langle & \left(1+\sigma_1\sigma_2\frac{\tilde{y}^+}{1-\tilde{y}^-}\right)\times \\
& \times\Bigg\{\sum_{m=3}^{n+2}\left[\prod_{k=3}^{n+2}\left(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}}\right)\right]\left(\frac{\sigma_m\sigma_{m-1}\delta_{\sigma_{m-1},\tau\sigma_{m-2}}}{1+\sigma_m\sigma_{m-1}x^{\tau}}\right)^2\delta_{\tau,\tau'}+ \\
& \quad+\Bigg[\sum_{3\le l<m\le n+2}\left[\prod_{k=3}^{n+2}\left(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}}\right)\right]\frac{\sigma_l\sigma_{l-1}\delta_{\sigma_{l-1}\sigma_{l-2},\tau}}{1+\sigma_l\sigma_{l-1}x^{\tau}}\frac{\sigma_m\sigma_{m-1}\delta_{\sigma_{m-1}\sigma_{m-2},\tau'}}{1+\sigma_m\sigma_{m-1}x^{\tau'}}+ \\
& \qquad+\left(\tau\leftrightarrow\tau'\right)\Bigg]\Bigg\}\Bigg\rangle_{\sigma_1,\ldots,\sigma_{2+n}}.
\end{aligned}
$$

As before, any factors with $k > m$ can be summed out directly. Thus the second and third terms vanish all together, since the factor $\frac{1}{1+\sigma_m\sigma_{m-1}x^{\tau'}}$ is cancelled by the $k = m$ factor of the product, after which only one occurrence of σ_m remains, and obviously $\langle\sigma_m\rangle = 0$. Thus we have established:

$$g_{\tau\tau'} = \sum_{m=3}^{n+2}\left\langle\left(1+\sigma_1\sigma_2\frac{\tilde{y}^+}{1-\tilde{y}^-}\right)\left[\prod_{k=3}^{m-1}(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}})\right]\frac{\delta_{\sigma_{m-1}\sigma_{m-2},\tau}}{1+\sigma_m\sigma_{m-1}x^{\tau}}\right\rangle_{\sigma_1,\ldots,\sigma_m}\delta_{\tau,\tau'}. \quad (3.4)$$

The averaging over σ_m can be performed:

$$\left\langle\frac{1}{1+\sigma_m\sigma_{m-1}x^{\tau}}\right\rangle_{\sigma_m} = \frac{1}{1-(x^{\tau})^2}.$$

Next we compute for $m \geq 4$:

$$\left\langle\left[\prod_{k=3}^{m-1}(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}})\right]\delta_{\sigma_{m-1}\sigma_{m-2},\tau}\right\rangle_{\sigma_3,\ldots,\sigma_{m-1}} =$$
$$= \frac{1}{2}\left\langle\prod_{k=3}^{m-2}(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}})\right\rangle_{\sigma_3,\ldots,\sigma_{m-2}} + \frac{\tau}{2}\left\langle x^{\sigma_{m-2}\sigma_{m-3}}\prod_{k=3}^{m-2}(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}})\right\rangle_{\sigma_3,\ldots,\sigma_{m-2}}.$$

The first term can be summed out, leaving just $\frac{1}{2}$. The second term can be computed iteratively:

$$\left\langle x^{\sigma_m\sigma_{m-1}}\prod_{k=3}^{m}(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}})\right\rangle = y^+ + y^-\left\langle x^{\sigma_{m-1}\sigma_{m-2}}\prod_{k=3}^{m-1}(1+\sigma_k\sigma_{k-1}x^{\sigma_{k-1}\sigma_{k-2}})\right\rangle$$
$$= \cdots = y^+\sum_{k=0}^{m-3}\left[y^-\right]^k + \left[y^-\right]^{m-2}x^{\sigma_1\sigma_2},$$

where we have defined $\sum_{k=m}^{m-1}\cdots \equiv 0$.

Inserting these results into (3.4) we find

$$g_{\tau\tau'} = \frac{\delta_{\tau\tau'}}{1-(x^{\tau})^2}\left\langle\left(1+\sigma_1\sigma_2\frac{\tilde{y}^+}{1-\tilde{y}^-}\right)\times\right.$$
$$\left.\times\left(\delta_{\sigma_1\sigma_2,\tau}+\frac{n-1}{2}+\frac{\tau}{2}\sum_{m=4}^{2+n}\left\{y^+\sum_{k=0}^{m-5}(y^-)^k+(y^-)^{m-4}x^{\sigma_1\sigma_2}\right\}\right)\right\rangle_{\sigma_1,\sigma_2}.$$

Using

$$\sum_{k=0}^{n}x^k = \frac{1-x^{n+1}}{1-x} \qquad \text{and} \qquad \sum_{k=0}^{n}\sum_{i=0}^{k}x^i = \frac{n+1}{1-x} - \frac{x(1-x^{n+1})}{(1-x)^2},$$

this leads to (3.3).

THE ROAD TO LARGER N

For N larger than two, the method used to explicitly calculate the metric and generalization error no longer works. One could introduce new parameters $\boldsymbol{x} = (\tanh\beta(w^1 \pm w^2 \pm \cdots \pm w^N))$, but there will be more x^τ's than w^i's, which will lead to a metric that has less than full rank. The inverse metric will then be ill-defined.

We shall perform the first few steps of the calculation of the metric in $\boldsymbol{w}$-space to find out where the trouble starts, and then consider various possible ways to proceed and the problems connected with each. No concrete results have so far been obtained for N larger than two.

4.1 The metric in w-space

From the general expression (1.3) with

$$p_{\boldsymbol{w}}(\boldsymbol{\sigma}) = p^{(\mathrm{input})}(\sigma_1,\ldots,\sigma_N)\, p_{\boldsymbol{w}}(\sigma_{N+1},\ldots,\sigma_{N+n}|\sigma_1,\ldots,\sigma_N)$$

and (1.5) with (1.6) we obtain

$$g_{ij}^{(w;n)} = \beta^2 \sum_{\sigma_1,\ldots,\sigma_N} p^{(\mathrm{input})}(\sigma_1,\ldots,\sigma_N) \times \\ \times \left\langle \sum_{k=N+1}^{N+n} \sum_{l=N+1}^{N+n} \left[\prod_{m=N+1}^{N+n} \frac{\mathrm{e}^{\beta\sigma_m h_m}}{\cosh\beta h_m} \right] \sigma_k\sigma_l\sigma_{k-i}\sigma_{l-j} \frac{\mathrm{e}^{-\beta\sigma_k h_k}}{\cosh\beta h_k} \frac{\mathrm{e}^{-\beta\sigma_l h_l}}{\cosh\beta h_l} \right\rangle_{\sigma_{N+1},\ldots,\sigma_{N+n}},$$

where we used the relation $1+\tanh x = \frac{\mathrm{e}^x}{\cosh x}$ and introduced $h_m \equiv \sum_{i=1}^N w^i \sigma_{m-i}$.

Any factors with $m > k$ and $m > l$ can be averaged out to unity straightaway. Splitting the remaining sum into terms with $k = l$ and $k \neq l$, we see that the latter vanish: suppose that $k > l$. No factors with $m > k$ remain. The factor $\mathrm{e}^{-\beta\sigma_k h_k}$ cancels against the factor with $m = k$ in the product. This leaves just one occurrence of σ_k in the entire expression. Thus the average over σ_k can be taken, annihilating the entire expression.

Exchanging the roles of k and l in the above argument, it becomes clear that the only surviving terms are those for which $k = l$. After averaging σ_k out as well, we are left with:

$$g_{ij}^{(w;n)} = \beta^2 \sum_{\sigma_1,\ldots,\sigma_N} p^{(\mathrm{input})}(\sigma_1,\ldots,\sigma_N) \times \\ \times \sum_{m=N+1}^{N+n} \left\langle \left[\prod_{k=N+1}^{m-1} \frac{\mathrm{e}^{\beta\sigma_k h_k}}{\cosh\beta h_k} \right] \sigma_{m-i}\sigma_{m-j} \frac{1}{\cosh^2\beta h_m} \right\rangle_{\sigma_{N+1},\ldots,\sigma_{m-1}}. \qquad (4.1)$$

The next step would be to calculate $p^{(\text{input})}(\sigma_1, \ldots, \sigma_N)$, but the analogue of (2.2) for general N contains expectation values of products of up to N spins, and the recursion relations encountered while computing these are far more tricky than the ones in appendix 2.A: not only are the polynomials of higher order, but the scalar equation (2.13) is replaced by a matrix equation as well.

The averaging over $\sigma_{N+1} \ldots \sigma_{N+n}$ poses additional problems, and the calculation of the generalization error again involves computing the expectation value of various spins. Straightforwardly performing all these calculations without approximations seems to be out of the question.

4.2 Alternative approaches

Several alternatives are worth considering, and I shall briefly list some of them here. I have scarcely given any of them the amount of attention they deserve, and there is plenty of scope for future work here.

4.2.1 High temperature approximation

It is possible to compute the metric in a high temperature approximation, but this is of limited value, since the learning process will inevitably lead to large values of the parameters w^i, as zero error is attained only in the limit where some of the w^i's become infinite. The high temperature limit becomes more relevant if the learning task is altered by imposing $\sum (w^i)^2 = 1$.

4.2.2 Only one output bit

If we limit ourselves to $n = 1$, (4.1) reduces to the metric for a single stochastic binary neuron (with a somewhat unusual input distribution). Although it is not easy to obtain analytic results because of the difficulties in computing the input distribution, various approximation schemes could be tried. It is a considerable advantage that the output conditional distribution can be calculated exactly, because this makes it possible to guarantee that the approximation of the metric is invertible.

4.2.3 Large n limit

Another possibility would be to consider the large n limit for finite temperature. However, for both $N = 1$ and $N = 2$ the generalization error trivializes in this limit, making learning irrelevant, and there does not seem to be much reason to assume that for larger (but finite) N this will be any different.

4.2.4 Large N limit

The large N limit is certainly interesting, but highly non-trivial, since it involves infinite dimensional Riemann spaces.

4.2.5 Calculation of $\mathcal{E}$ in terms of other variables

One could attempt to generalize the method used for $N = 2$: although the space spanned by $x^{(\tau)} = \tanh\beta(w^1 \pm w^2 \pm \cdots \pm w^N)$ has dimension 2^{N-1} and is therefore degenerate for N larger than two, this does not stop us from trying to express the error in terms of these variables. For $N = 3$ I was able to obtain a recursion relation similar to (2.12), but it resulted in a third order polynomial equation with matrix valued coefficients instead of the quadratic scalar equation derived from (2.13). These equations will thus have to be solved numerically, but this might be feasible.

4.2.6 Approximation of error and metric by simulation

One could try to approximate the generalization error simply by letting the student (and teacher) generate a large number of bit sequences according to their respective probability distributions, and compute an expectation value for the generalization error from the results. Some preliminary tests of this approach showed that a very large number of measurements must be taken to obtain a reasonably accurate estimate.

It is not clear whether it is worthwhile spending a lot of effort at each time step just to get a rather poor estimate of the error.

4.2.7 Online learning

This leads us to the final possibility: not to attempt to get a good estimate of the generalization error at all, but just use the error obtained from a single example and a single run of the student network, in other words, do *online* gradient descent. It will be interesting to see if even in this case using a rough approximation of the metric — possibly based on the same online data as the estimate of the error — offers improvement of learning times over OGD online learning. A good starting point might be the $n = 1$ regime since this allows a relatively safe approximation for the metric.

CONCLUSIONS AND OUTLOOK

We have investigated whether natural gradient descent (NGD) learning as proposed by Amari [2] offers a substantial improvement in terms of learning times over ordinary gradient descent (OGD) learning for Kanter's bit generator, a time series generator that constructs an infinite stream of bits from a small number of initial values, each successive bit being the output of a stochastic binary neuron presented with the previous N bits as input. For one output bit and one or two input bits, we have shown analytically that the achieved generalization error decreases exponentially with time for NGD learning, while OGD learning yields a generalization error that decreases linearly with one over the learning time. These results hold for any value of the teacher weights, independently of noise levels.

For two input bits and more than one output bit, we were able to calculate the metric exactly, and we used computer simulations to find the evolution of the training error with time. For generic values of the teacher weights, NGD was again found to yield exponential decay of training error, while OGD gave power law decay. The simulations indicate that the exponent of the latter varies between 0.95 and 1.05 in the bulk of teacher weight space, but we do not have enough evidence to conclude that this exponent actually differs from one (the value found analytically when predicting only one future bit). In some areas of teacher weight space the generalization error function was found to contain local minima. In part of these areas learning was slowed down considerably, as the student spent time hovering on the edge of the domain of attraction of that minimum. In other parts the students actually ended up in the local minimum. For a small region of teacher weight space the OGD student escaped to the global minimum, while the NGD student did not. This perhaps unexpected result is possible because a change of metric induces a shift in the boundaries between domains of attraction, which can theoretically put either OGD or NGD at an advantage.

Considering more than two input bits makes the calculations substantially more difficult, and in fact we could not find a simple expression for the metric in this case. Several options for proceeding beyond two input bits are described in the last chapter. It would be most interesting to compare NGD with OGD in an online learning scenario. The first regime to be considered should be a student that predicts only one output bit, since this would allow for comparison with results obtained for stochastic binary neurons [4]. An important question would be how much effort should be invested in obtaining an estimate for the metric relative to the amount of effort invested in the estimate of the generalization error. A priori it is uncertain whether a poor guess of the metric is better than no guess at all, but our findings indicate that NGD learning could offer huge time savings.

BIBLIOGRAPHY

[1] S. Amari, *Differential-geometrical methods in statistics*, Lecture notes in statistics, Springer-Verlag, Berlin, 1985.

[2] A. Fujiwara and S. Amari, *Gradient systems in view of information geometry*, Physica **D 80** (1995) 317.

[3] S. Amari, *Natural gradient works efficiently in learning*, Neural Computation **10** (1998) 251.

[4] M. Rattray and D. Saad, *Transients and asymptotics of natural gradient learning*, Submitted to ICANN 98, 1998.

[5] A. C. C. Coolen, *Information theory in neural networks, lecture notes of course G31/NN4*, available from KCL Math. dept., July 1998.

[6] E. Eisenstein, I. Kanter, D. A. Kessler, and W. Kinzel, *The bit generator and time-series prediction*, cond-mat/9502102.

[7] A. Priel, I. Kanter, and D. A. Kessler, *Noisy time series generation by feed-forward networks*, cond-mat/9803267.

[8] R. d'Inverno, *Introducing Einstein's relativity*, Clarendon Press, Oxford, 1992.

[9] D. A. Wagenaar, *Information geometry for neural networks*, Thesis for KCL Math. dept., available from author, April 1998.

[10] J. M. Corcuera and F. Giummolè, *A characterization of monotone and regular divergences*, Accepted by Annals of the Institute of Statistical Mathematics, 1998.