The statistical mechanics of constructive algorithms

1998 J. Phys. A: Math. Gen. 31 8977

(http://iopscience.iop.org/0305-4470/31/45/002)

J. Phys. A: Math. Gen. **31** (1998) 8977–9021. Printed in the UK PII: S0305-4470(98)91166-0

The statistical mechanics of constructive algorithms

Ansgar H L West†‡ and David Saad†§

† Neural Computing Research Group, Aston University, Aston Triangle, Birmingham B4 7ET, UK

‡ Department of Physics, University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, UK

Received 29 January 1998, in final form 3 August 1998

Abstract. The storage capacity of multilayer networks with overlapping receptive fields is studied for constructive algorithms using Boolean perceptrons as their basic building block which have been investigated within a replica framework. The assumption of weak coupling between subsequently constructed perceptrons is verified within a replica symmetric (RS) ansatz and shown to be negligible in most cases in comparison with correction due to replica symmetry breaking (RSB) in individual perceptrons. The capacities of a tiling-like and variants of the upstart algorithm are then calculated within RS and one-step RSB with the quenched average taken over the individual units separately for networks with up to $K = 4000$ and $K = 600$ units respectively. Within this treatment, the storage capacity α_c^K seems to exhibit a power-law behaviour in $\log K$ with an exponent n that may depend on the algorithm and the stability. However, due to finite size effects in K reliable estimates of n could not be extracted. Nevertheless, the results strongly indicate that n should be strictly smaller than 1 within one-step RSB, whereas within RS the Mitchison–Durbin bound is violated for finite K and $n > 1$ may hold asymptotically.

1. Introduction

Since the ground breaking work of Gardner [1, 2] on the storage capacity of Boolean perceptrons, the replica [3] and other techniques of statistical mechanics have been successfully employed to investigate many aspects of the performance of simple neural network models. Whereas initially research focused on the storage capacity problem attempting to evaluate the number of examples with random output that can be stored on average, the attention has shifted recently mainly towards the understanding of the supervized learning problem within the student–teacher scenario, which calculates the generalization capability of a neural network model with the number of training examples available.

The capacity problem does, however, remain relevant due to its relation to the Vapnik–Chervonenkis (VC) dimension [4] of computational learning theory and the probably approximately correct (PAC) framework, the difference broadly speaking being that statistical mechanics analyses the average case, whereas the PAC framework analyses the worst case. Within the PAC framework, the VC-dimension enables one to determine an upper bound on the examples needed to achieve a certain generalization error [5, 6], broadly reflecting the view that generalization can only begin once the storage capability has been exceeded. The capacity of a network model therefore influences both its flexibility of

§ Author to whom correspondance should be addressed. E-mail address: D.Saad@aston.ac.uk

0305-4470/98/458977+45$19.50

implementing complicated mappings and its generalization ability for a training set of given size.

Substantial work has therefore been carried out in both communities in order to calculate these storage quantities, however, the problem has proved to be very hard for multilayer perceptrons (MLPs) and most success has been reserved to the estimation of upper and lower bounds [6–11] of two-layer networks, resulting in the well known lower and upper bounds of the storage capacity (per adjustable weight) of 1 and $\log_2 K$ (Mitchison–Durbin (MD) bound [7]) respectively, where K is the number of units in the hidden layer. Attempts for the direct calculation of the capacity limit of MLPs have been hampered by the inherent difficulties of the replica calculation needed to perform the quenched average of the training set†. In the capacity calculations of the parity‡ [13] and committee§ [14, 15], replica symmetric (RS) treatments violate the MD bound derived by information theory or counting arguments similar to [16], whereas a replica symmetry breaking (RSB) calculation [13] saturate the bound in the $K \to \infty$ limit for the tree parity machine. Other efforts [17] suggest breaking the symmetry of the hidden units explicitly prior to the actual calculation, but the resulting equations are approximations and difficult to solve for large networks. Recently, the introduction of a new technique [18], focusing on the number of implementable internal representations instead of on the Gardner volume, has been used to calculate the capacity of the tree [19] and fully connected committee machine [20, 21]. In the $K \to \infty$ limit, the committee machine does not saturate the MD bound but still diverges with $\sqrt{\log K}$.

This paper avoids these problems by addressing the capacity of a class of networks with variable architecture produced by constructive algorithms. In this case, the basic building blocks are simple Boolean perceptrons, which are trained individually and results derived for Boolean perceptrons above their saturation limit [22] can be applied iteratively to yield the storage capacity of two-layer networks.

A multitude of constructive algorithms have been proposed over the years, e.g. [23–29]. They are all loosely based on the idea that in general it is *a priori* unknown how large a network must be to perform a certain regression or classification task. It seems therefore appealing to start off with a simple network, e.g. a Boolean or sigmoidal perceptron, and to increase its complexity by adding further units only when needed, thereby eliminating the cumbersome search for the right network size.

However, the constructive algorithms proposed differ in several aspects. Some of them are applicable to regression [24, 27] others to classification [25, 26, 28, 29] tasks, which are often reflected in the type of units they use (Boolean, sigmoid, RBF (radial basis function), or HON (higher order network)). They also produce several typical architectures, e.g. hierarchical tree type [28], list type [29], cascade type [26, 27], self-organizing cell-type [30], multilayer with either fixed [28, 31, 35] or problem driven deepness [25]. Some algorithms also have several versions, which usually result in different architectures.

Another important difference lies in the training procedure performed once a new unit has been added. Some algorithms require only the training of the newly added unit fixing the weights of previously constructed units, while others require some sort of retraining of connections involving output weights and/or unit weights. The great advantage of the former is that the training time of the whole network is usually relatively short, since training involves only small units, typically single-layer networks, for which fast training algorithms are available even for Boolean units [36–42]‖. These constructive algorithms

† Such difficulties can be avoided in the generalization problem by studying on-line learning [12].

‡ The output of a parity machine is the product of all hidden unit outputs.

§ The output of a committee machine is the majority of all hidden unit outputs.

‖ Some of these algorithms have to be stabilized for nonlinear separable problems by the *pocket* algorithm [23].

can therefore avoid the difficulty of learning internal representation of units, e.g. by back-propagation for sigmoid units [43, 44] or by the CHIR (learning by choice of internal representations) algorithm for Boolean units [45]. This is in contrast to other proposed constructive algorithms [30, 24] and general MLPs with an *a priori* fixed architecture. This training process is especially difficult for Boolean units, where powerful second-order gradient-based techniques [46] are not available.

A further advantage of some constructive algorithms, which train only single-layer units, is the existence of convergence proofs, i.e. one can show that training will converge in finite steps, unlike conventional networks which can be trapped in bad local minima and often have to be restarted many times before an acceptable solution is found. For some algorithms this convergence is to zero training error, a feature which leads to undesirable over-fitting and subsequent poor generalization for noisy data. However, this problem can be addressed by including some kind of penalty term on the creation of new units to the training error and/or by training with negative stability allowing for errors close to the decision boundary. That a constructive algorithm can in principle be a very good generalizer has been shown in [47]. There it has been proven within the PAC framework, that any *weak learner*, a machine which only achieves a generalization error just below random guessing, can be used to constructively build a *strong learner*, a machine which achieves any arbitrary small generalization error. This *boosting* algorithm and its improved variants [48, 49] have shown very promising results in real world applications [50], along with other constructive algorithms which have been tested on noisy problems [51].

Other approaches, which aim at automating the choice of appropriate network size, are based on starting with large networks and then attempt to optimize performance by identifying and removing unnecessary individual weights and/or units according to some predefined rules, e.g. [52–56]. These procedures usually require computationally expensive calculations and further retraining of the pruned network. A conceptionally different but effectively similar approach is to add penalty terms [57–59] to the energy function to be minimized, often also termed weight decay or regularization, which practically eliminate weights which do not significantly contribute to the reduction of the training error. Within a Bayesian framework [60], it is also not necessary to restrict the number of units *a priori*; however, Bayesian methods can be computationally prohibitively expensive in many situations.

Overall, constructive algorithm seem therefore rather appealing, but the abilities of different algorithms have neither been compared heuristically in a systematic way on real world problems nor has any attempt been made to understand their properties within a theoretical framework. The aim of this paper is, therefore, to introduce a framework in which one aspect of the performance, the learning of random dichotomies or capacity problem, of a class of constructive algorithms can by analysed and compared objectively. This should give us some indication of how effective different constructive algorithms use their weights in comparison with each other and to the upper bounds known for unconstrained MLPs.

The class of constructive algorithms susceptible to this framework consists of algorithms which use only Boolean perceptrons as the basic building block and where later generations of units receive no input from previously constructed units, unlike e.g. cascade networks such as [25, 27]. This is due to the fact that our treatment relies on iteratively using results derived for individual Boolean perceptrons above their saturation limit [22]. We therefore rely on the approximation that the quenched average over the training set can be taken separately for each individual perceptron, i.e. correlations between the output of previous hidden units need not be taken into account and the correlations between the errors of the units are small.

In this paper, we will investigate in particular variants of the upstart algorithm [28] and a tiling-like algorithm [62], although our calculations can easily be extended to many other algorithms, such as [32, 34, 29]. For the algorithms studied here, the corrections to the decoupled approximation have been calculated for two consecutive units within an RS ansatz and turn out to be small in most regimes in comparison with correction due to RSB in each individual perceptrons, rendering errors due to the decoupling assumption small.

For these algorithms, we calculate the capacity within the RS and one-step RSB ansatz for networks with up to $K = 4000$ (RS) and $K = 600$ (RSB) units. Within this treatment, the numerical results strongly indicate that the storage capacity α_c^K exhibits a power-law behaviour in $\log K$ with an exponent n, which may be stability and algorithm dependent. The exponent has been measured locally (in contrast to an extrapolated estimate) showing slight systematic shifts with the number of units K, so that reliable upper bounds or estimates of n for $K \to \infty$ could not be extracted. Therefore, recent asymptotic capacity results may be interesting theoretically, although, finite K effects may render them irrelevant for practical considerations. For all constructive algorithms studied, the finite K results further indicate that n is strictly smaller than 1 when accounting for RSB. Within the simpler RS treatment, the Mitchison–Durbin bound is violated for large finite K and $n > 1$ may hold asymptotically.

The paper is structured as follows. In section 2 the capacity problem is introduced and the investigated constructive algorithms described. In section 3, an introduction to the replica framework used for the capacity calculation is given and the mechanism for employing results derived for simple perceptrons to obtain results for the capacity limit of constructive algorithms is explained. This will be complemented by a brief presentation of results for a single and two coupled perceptrons, which give insight into the numerical results of the iterative calculation of the capacity limit of networks built by constructive algorithms. In sections 4 and 5 the numerical capacity data is presented and analysed by calculating the local power-law exponent $n(K)$. The paper finishes with a discussion and some concluding remarks in section 6.

2. The capacity problem and constructive algorithms

In this section, we first define the capacity problem and introduce the simplest neural network model, the Boolean perceptron. This will be used to motivate the introduction of constructive algorithms. The tiling-like algorithm [62], which has previously been analysed numerically within an RS ansatz, is explained and used to describe features underlying generic constructive algorithms. The ideas of the original upstart algorithm [28] are introduced also and compared with the versions, termed upstart II-III, which have been used in the capacity calculations.

2.1. The capacity problem and the Boolean perceptron

Both the capacity and VC-dimension problem consider whether a learner, e.g. a neural network, can implement a set of $p = \alpha N$ random dichotomies given as a (training) set of input–output pairs $(\boldsymbol{\xi}^\mu, \zeta^\mu)$ $(\mu = 1, \ldots, p)$ with $\boldsymbol{\xi}^\mu \in \{-1, 1\}^N$ and $\zeta^\mu \in \{-1, 1\}$, where both the inputs $\boldsymbol{\xi}^\mu$ and the outputs ζ^μ are drawn independently from their respective probability distributions $P(\boldsymbol{\xi}^\mu)$ and $P(\zeta^\mu)$. Note, that one can use a symmetric Boolean $\{-1, 1\}$ output representation without loss of generality (w.l.o.g.) since an asymmetric representation $\{0, 1\}$ can be mapped to a symmetric one by redefining $\zeta^{\mu\prime} = (2\zeta^\mu - 1)$. For simplicity, we will henceforth refer to a Boolean perceptron just as perceptron.

The difference between the capacity and the VC-dimension definition is roughly that the former is probabilistic and distribution dependent, whereas the latter is not. The VC-dimension is formally defined as the maximal set size p of input examples which can be shattered, i.e. mapped to any desired output set. The capacity limit is defined as the set size p for which a random input example set can be correctly mapped to a random output set with probability $\frac{1}{2}$, i.e. when taking the *quenched* average over input and output sets. In the thermodynamic limit of infinite input dimension, $N \to \infty$, this probability can be conveniently redefined as arbitrarily close to 1 as the probability of implementability becomes a step function. Furthermore, the capacity limit α_c is usually defined not in terms of the set size p but as the ratio between p and the number of free parameters in the network, which, for example, for a two-layer network in the thermodynamic limit is NK, where N is the input dimension and K the number of hidden units. For the distributions it is generally assumed that the binary input distribution is independent of the pattern and site indices μ and j

$$P(\xi_j^\mu) = P(\xi) = \tfrac{1}{2}(1+m_i)\delta(1-\xi) + \tfrac{1}{2}(1-m_i)\delta(1+\xi). \tag{1a}$$

The random output distribution is also chosen to be independent of the pattern index

$$P(\zeta^\mu) = P(\zeta) = \tfrac{1}{2}(1+m_o)\delta(1-\zeta) + \tfrac{1}{2}(1-m_o)\delta(1+\zeta) \tag{1b}$$

where m_i and m_o represent the input and output biases respectively.

The simplest neural network, the perceptron, is parametrized by its synaptic weight vector $\boldsymbol{W} \in \mathbb{R}^N$ and threshold $\theta \in \mathbb{R}$, performing the mapping

$$\sigma^\mu = \operatorname{sgn}\left(\frac{1}{\sqrt{N}}\boldsymbol{W}\cdot\boldsymbol{\xi}^\mu - \theta\right) = \operatorname{sgn}(h^\mu) \tag{2}$$

where $\operatorname{sgn}(x)$ is the sign of x and h^μ is termed the activation of the perceptron.

A further property, which has a strong influence on the capacity limit is the error measure used to train the perceptron. Here it is defined as

$$E = \sum_\mu \Theta[\kappa - \zeta^\mu h^\mu] \tag{3}$$

where $\Theta(x)$ is the Heaviside step function, which is 1 for $x > 0$ and 0 otherwise and κ is the stability with which the patterns are required to be stored. The choice of the stability κ has a significant impact on the capacity limit since it fixes the minimal allowed distance between a pattern and the decision hyperplane of the perceptron. This error function, often referred to as the Gardner–Derrida cost function, counts the number of patterns which are implemented with a stability less than κ, i.e. all misclassified patterns but also some correctly classified patterns for $\kappa > 0$. The Gardner–Derrida cost function leads to the least number of errors theoretically achievable by any 'practical' learning algorithm (e.g. [41]).

The capacity of the perceptron has been calculated initially for zero stability using geometric arguments [16] ($\alpha_c = 2$) a result which has been reproduced for arbitrary stability [1] within a replica framework. However, it is obvious that the perceptron can only learn linear decision boundaries. Therefore, if the set of examples is not linearly separable with a minimum distance κ of all patterns to the hyperplane, the perceptron will not be able to classify all patterns without errors. In this case, a learning machine, such as a MLP, that is able to learn nonlinear decision boundaries needs to be trained on the examples. The basic idea of many constructive algorithms is to add new perceptrons in such a way that the combination of all their linear decision boundaries leads to a nonlinear decision boundary that performs the required task. The constructive algorithms vary significantly in the way these extra decision boundaries are trained and how they are combined to yield the overall

output. Below, this will be explained for the constructive algorithms analysed in detail in this paper, a tiling-like algorithm [62] and the upstart algorithm [28].

2.2. The tiling-like algorithm

The basic idea of the tiling-like algorithm [62] and of most other constructive algorithms such as [25, 26, 34] is to constructively build a faithful internal representation in the hidden layer, i.e. all patterns with the same internal representation share the same target output. The remaining problem is then to devise a way to map the internal representation to the desired output, which can be solved in many different ways. The tiling-like algorithm achieves this by constructively building a very specific set of internal representations, which is automatically mapped to the desired output by a hardwired parity function as a fixed hidden-output mapping, where the output is just the product of the individual outputs of all constructed units, leading to a chequered partition of the input-space.

The faithful representation in the hidden layer is achieved in the following way. The first perceptron, $\mathcal{U}_1$, is trained on the original Boolean targets $\zeta^\mu \in \{-1, 1\}$. If this unit makes any errors, ϵ_1, on the training set, a second unit, $\mathcal{U}_2$, is created which is trained on the complete training set but with modified targets exploiting the property of the parity function: whereas the output of the whole network would remain unchanged for an output of $+1$ by $\mathcal{U}_2$, it would be reversed for -1. Hence, the targets of $\mathcal{U}_2$ are $+1$ for previously correctly classified and -1 for misclassified patterns. This procedure is iterated until the current unit $\mathcal{U}_i$ classifies all patterns correctly (according to its targets). It can be shown that this algorithm will eventually converge as $\mathcal{U}_i$ corrects at least one previously incorrectly classified pattern without upsetting any correctly classified ones. Note, that it is sufficient to train each perceptron with stability κ to ensure that all examples are finally implemented with the desired stability, a property which also holds for most other constructive algorithms including the upstart algorithm.

2.3. The upstart algorithm

Although the basic idea of building a faithful internal representation by adding new units holds also for the upstart algorithm [28], the technical details of the algorithm are somewhat different to many other constructive algorithms. First of all, it uses an asymmetric $\zeta^\mu \in \{0, 1\}$ instead of the usual symmetric (Ising) representation $\zeta^\mu \in \{-1, 1\}$ for the outputs. Similar to other algorithms, one starts with a single perceptron, the *mother* unit $\mathcal{M}$, and further units are created only if erroneous patterns exist. However, in this algorithm potentially two *daughter* units, $\mathcal{U}^+$ and $\mathcal{U}^-$, are created to specifically correct one of the two possible type of errors: *wrongly-off* errors, where the target was 1 but the output is 0, and *wrongly-on* errors, where the target was 0 but the actual output is 1. $\mathcal{U}^+$ and $\mathcal{U}^-$ are connected to their mother unit $\mathcal{M}$ by a large enough positive or negative weight, respectively, so that they overrule any decision by $\mathcal{M}$ when they are active ($\sigma = 1$).

Consider, for example, the new training set and targets that would be assigned to $\mathcal{U}^-$, which will be connected with a large negative weight to $\mathcal{M}$, i.e. whose role will be to inhibit $\mathcal{M}$. $\mathcal{U}^-$ should be active ($\sigma = 1$) for patterns where $\mathcal{M}$ is currently wrongly-on and inactive ($\sigma = 0$) for patterns where $\mathcal{M}$ is correctly-on. However, $\mathcal{U}^-$ does not have to be trained on patterns for which $\mathcal{M}$ is correctly-off, since an active $\mathcal{U}^-$ would only reinforce $\mathcal{M}$'s already correct response. The remaining patterns, for which $\mathcal{M}$ is wrongly-off, need special consideration. They have to be included in $\mathcal{U}^-$'s training set with target 0, in order to avoid inhibiting the pattern further which would lead to frustration

Table 1. The targets of the original upstart algorithm and its variants depending on the targets ζ and the output σ of the current mother unit (or master output unit) $\mathcal{M}$. The target '$*$' means that the pattern is not included in the training set of $\mathcal{U}_i^{\pm}$ for all algorithms, whereas a bracket $[\cdot]$ around a target has the same meaning for upstart III where only one type of unit, $\mathcal{U}_i^+$ or $\mathcal{U}_i^-$, is created per generation.

	Target ζ	
Output σ	$\zeta = 1$	$\zeta = 0$
$\sigma = 1$	correctly-on $\mathcal{U}_i^+$ $*$ $\mathcal{U}_i^-$ 0	wrongly-on $\mathcal{U}_i^+$ [0] $\mathcal{U}_i^-$ 1
$\sigma = 0$	wrongly-off $\mathcal{U}_i^+$ 1 $\mathcal{U}_i^-$ [0]	correctly-off $\mathcal{U}_i^+$ 0 $\mathcal{U}_i^-$ $*$

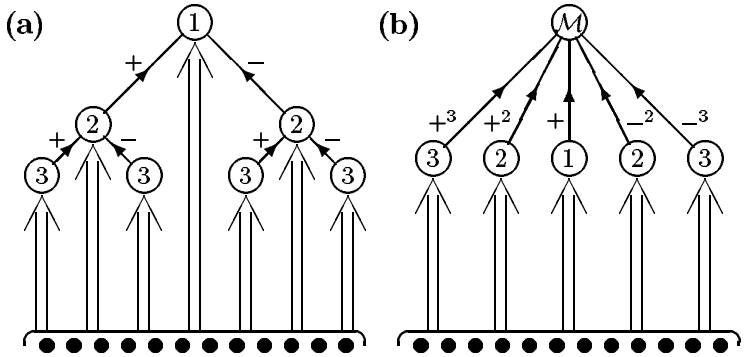

Figure 1. Networks of three generations produced by (*a*) the original upstart algorithm and (*b*) the modified upstart II algorithm. The number of units of the original algorithm grows exponentially with each generation whereas the modified version grows only linearly. The black dots represent the input units. The open circles are hidden and output units created by the two version of the upstart algorithm numbered after their generation. The wide arrows symbolize input weights from all the input units, whereas the normal arrows represent single weights between hidden units and to the output unit $\mathcal{M}$. The plus and minus signs are the sign of the connecting weights and the powers give an indication of their magnitude.

when combined with the output of $\mathcal{U}^+$, which is trying to correct the wrongly-off patterns. Similar arguments can be applied to $\mathcal{U}^+$, and the resulting targets and training sets for both unit types are summarized in table 1.

If $\mathcal{U}^+$ and $\mathcal{U}^-$ can correct all erroneous patterns, the algorithm has achieved its objective and terminates. Otherwise, various possibilities exist for its continuation, of which several have already been reported in the original works [28, 61]. In the original algorithm (termed here upstart I), those daughter units with non-zero training errors in turn become the mothers of the daughters of the next generation, leading to a hierarchical network architecture as shown in figure 1(*a*). Consequently, this allows for a parallelization of the local training procedure, but also tends to lead to an exponentional increase of the number of hidden units (and hence unit specific training sets) with each generation, which may potentially make extremely wasteful use of hidden units. It has already been pointed out in the original publication [28], that this hierarchical tree can be squashed into an equivalent

more conventional two-layer architecture, where all units (including the original unit $\mathcal{M}$, which can then be seen as $\mathcal{U}_1^+$) are connected to a master output unit $\mathcal{M}$, with positive or negative weights whose magnitude increases with each generation in order to guarantee that erroneous decisions are actually corrected by the following generation as shown in figure 1(*b*). These two-layer networks show only a linear increase of the number of units with each generation, which may use each unit more efficiently and which is also easier to analyse for the purposes of this paper.

In this case, one has to decide what criteria to use for the creation of new units. One could obviously create both types of units, $\mathcal{U}_i^+$ and $\mathcal{U}_i^-$, simultaneously with each generation as before if both types of errors are made, but this may be wasteful if, for example, there are much more wrongly-off than wrongly-on errors.

A more efficient variant, here termed upstart II, is therefore to create both types of units if the probability of both error types is identical and only one unit correcting the error of higher frequency otherwise. Two obvious choices exist for evaluating these probabilities. One could either use the marginal probabilities of wrongly-on and wrongly-off errors or one could condition the probabilities on the original target, i.e. compare the probability of a wrongly-on error given a pattern with original target $\zeta = 0$ with that of a wrongly-off error given a pattern with original target $\zeta = 1$. We will denote the former as criterion (a) (i.e. upstart IIa) and the latter as criterion (b) (i.e. upstart IIb). Obviously, these two variants are identical for the case when the initial output distribution is unbiased, i.e. the number of targets for each class is identical.

Note, that in principle it would not be necessary to include the wrongly-off patterns in $\mathcal{U}_i^-$'s training set (and similarly wrongly-on patterns in $\mathcal{U}_i^+$'s training set) if only one unit is constructed per generation, since in this case frustration of the output unit is not an issue that has to be addressed. In order to investigate possible efficiency gains, a further variant, upstart III, is proposed, which always creates only one type of unit and can therefore implement the above mentioned reductions in training sets. As for upstart II, we again consider both criteria (a,b).

The formal definition of these versions of the upstart algorithm is as follows.

Step 0. Create an asymmetric Boolean $\{0, 1\}$ output unit $\mathcal{M}$ with threshold 1, to which all other units, forming the hidden layer, will be attached to. Subsequently, create the initial processing unit $\mathcal{U}_1^+$ train it on the original targets ζ^μ, freeze its weights, and connect it to $\mathcal{M}$ with $a+1$ weight, i.e. $\mathcal{M}$ has initially the same outputs as $\mathcal{U}_1^+$. Initialize the generation index $i = 1$ and the index for $\mathcal{U}^+$ and $\mathcal{U}^-$ units to $p = 1$ and $m = 0$.

Step 1. Evaluate the number of wrongly-off and wrongly-on errors, ϵ^{off} and ϵ^{on}, made by $\mathcal{M}$ in generation i. Terminate if all patterns are correct, otherwise calculate the error probabilities to be applied, i.e. $p^{\text{off}} = P(\epsilon^{\text{off}})$ and $p^{\text{on}} = P(\epsilon^{\text{on}})$ for criterion (a) or $p^{\text{off}} = P(\epsilon^{\text{off}}|\zeta^\mu = 1)$ and $p^{\text{on}} = P(\epsilon^{\text{on}}|\zeta^\mu = 0)$ for criterion (b). Then create unit(s) according to the employed variant, i.e. if $p^{\text{on}} > p^{\text{off}}$ then $\mathcal{U}_{i+1} := \mathcal{U}_{m+1}^-$, if $p^{\text{on}} < p^{\text{off}}$ then $\mathcal{U}_{i+1} := \mathcal{U}_{p+1}^+$, and otherwise ($p^{\text{on}} = p^{\text{off}}$) $\mathcal{U}_{i+1} := \mathcal{U}_{m+1}^- + \mathcal{U}_{p+1}^+$ for upstart II and $\mathcal{U}_{i+1} := \mathcal{U}_{p+1}^+$ for upstart III. The targets and training sets of the new unit(s) are given by table 1.

Step 2. The new unit(s) are trained on their new training set(s) and their weights are frozen.

Step 3. The new unit(s), $\mathcal{U}^{+}_{p+1}$ and/or $\mathcal{U}^{-}_{m+1}$, are connected with positive respectively negative weight of identical magnitude to the output unit $\mathcal{M}$. The magnitude is adjusted so that previous decisions are overruled if one new unit is active. Go back to step 1.

Similarly to the tiling-like algorithm, these versions of the upstart converge eventually, as each new generation reduces the total error by at least one pattern per created unit.

3. Calculation of the capacity in a replica framework

As mentioned in section 2.1, the capacity limit is defined in probabilistic terms, as the property of being able to realize a mapping on 'average' over all possible training sets. In the statistical mechanics community this problem has been addressed in several ways, all of them using the same basic technique. The initial approach [1], calculates the average (logarithm of the) volume in parameter space, which implements the training set perfectly. The reason for calculating the logarithm of the volume (often termed *Gardner volume*) rather than the volume itself, is that the first is assumed to be *self-averaging* in the thermodynamic limit of infinite input dimension ($N \to \infty$), whereas the second is not. The capacity limit is reached, when this volume vanishes. The second approach [2], calculates the average free energy (corresponding to the error for the Gardner–Derrida cost function) the network makes for a given training set size and training temperature. In the limit of zero training temperature, the capacity limit is determined by the largest training set size with zero error. The third approach [18], calculating the (logarithmic) number of implementable faithful internal representation, is similar to the Gardner-volume approach and is especially applicable in MLPs. The average of the logarithmic quantity over the input and output distributions is in all cases performed by using the replica trick, which replaces the average over the logarithm at the expense of introducing replicated network parameters.

These replica calculations are notoriously difficult for MLPs. Furthermore, it is not clear how the above frameworks can be extended to the case of constructive algorithms, where the architecture is not fixed *a priori*, but evolves during training and two instances of the same training set size could, for example, lead to different architectures. However, in the case of constructive algorithms, the optimization of each perceptron is performed individually, severely constraining the interaction between the different perceptrons. The influence of the previous perceptrons is only via the modification of the training set, by redefining the targets and/or selecting a subset of the patterns. Since the original targets are random, it could be argued that the redefined targets are approximately random as well, but coming from a distribution with modified bias. In this case, the quenched average over the patterns decouples, and the capacity of networks built by constructive algorithms can be calculated from results derived for simple perceptrons: the errors made be the perceptron(s) of the current generation determine the example load α and the bias of the output distribution m_{o} for the next generation.

3.1. Assessing the influence of correlations

In order to assess whether this approximation is justifiable, a replica calculation for two perceptrons created in consecutive steps of the considered constructive algorithms has been carried out within a RS ansatz. These correlations should be dominant in comparison with correlations between perceptrons which are more than one generation apart. The details of the calculation and the results are reported in appendix A; here, only the main implications will be reported. For both the upstart and the tiling-like algorithms, the effect of correlations

between consecutive units on the capacity limit or the error rate is usually insignificant in comparison with the effects of one-step RSB in the individual perceptrons. For the tiling-like algorithm, the effects of correlations are usually smaller than for the upstart algorithm and one even finds situations where the correlations are non-existent (zero bias and small stability). Although, one can identify one region—small (but finite) m_o, large κ, and α around the capacity limit—where correlations are substantial, this region should be only relevant for small networks and will have no bearing on the results presented here.

These correlation results may be compared with capacity results for fixed two-layer architectures with unconstrained optimization. For the parity machine with fixed architecture, which is somewhat related to the tiling-like algorithm, one finds that the correlations (in terms of overlaps) between units are zero for any number of hidden units K [15] (for zero stability and unbiased output distribution), leading to the same capacity for tree and fully connected architectures. For the fully connected committee machine, which is in spirit more similar to the upstart algorithm, one finds (anti-) correlations in the capacity calculation for finite K, which vanish proportional to K^{-1} in the limit of large K [20, 21], leading only to a correction in the prefactor when compared with the tree architecture [18, 19].

The two most relevant features found for the correlations hold for all constructive algorithms considered here and should carry over at least qualitatively to a more accurate one-step RSB calculation (which is beyond the scope of this work). First, any correlation between the perceptrons leads to a decreased capacity of the combined network or to an increased error rate above saturation†. The uncoupled approximation should, therefore, constitute at least an upper bound to the true capacity, if this result holds qualitatively when accounting for RSB‡. Furthermore, the correlations vanish in the region where both units are highly saturated, which could be considered the most relevant region, since most units operate in this regime for large networks. Hence, we believe that the upper bound calculated should be relatively tight, especially for the tiling-like algorithm.

3.2. Capacity and error rates for single perceptrons

After having assessed the influence of correlations, and having concluded that their impact is less significant to negligible in comparison with one-step RSB in the single perceptron, the purpose of this section is to briefly review the results for the capacity and the error rates for simple perceptrons insofar they are relevant for the ensuing capacity calculation. We limit ourselves to the case of the spherical perceptron as the Ising perceptron behaves generically similar. For completeness, the equations for the free energies and the error rates are summarized in appendix B.

The capacity limit, α_c, of a simple perceptron is only a function of two parameters§: the output bias m_o and the stability κ. It is evident, that an increase in stability leads to a decrease in the capacity, whereas an increase in output bias m_o leads to an increase in the capacity, with $\alpha_c \to \infty$ for $m_o \to \pm 1$. It is therefore convenient to introduce the 'bias' $\tilde{m}_o \equiv (1 - |m_o|)$ for large magnitude of the bias. In figure 2 $\tilde{m}_o$ is shown as a function of

† This may seem somewhat surprising initially, since the anti-correlations in the committee-machine result in an increase of the capacity of the fully connected in comparison with the tree committee-machine. However, this effect may be explained by the constrained optimization for constructive algorithms and/or the increase in functional flexibility when going from a tree to an overlapping architecture.

‡ The breakdown of one-step RSB in the individual perceptrons does not constitute a problem since further RSB steps increase the error rate [63].

§ The third potential parameter, the input bias m_i, can be absorbed by a suitable rescaling of the stability κ and can therefore be set to zero w.l.o.g.

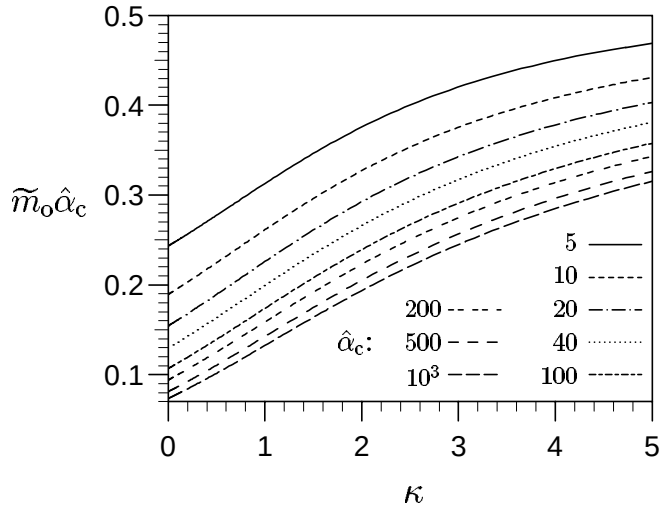

Figure 2. The bias $\tilde{m}_o$ (or rather $\tilde{m}_o\hat{\alpha}_c$ to highlight the scaling of α_c with $\tilde{m}_o$) is shown as a function of the stability κ for several fixed normalized capacities $\hat{\alpha}_c(m_o,\kappa) \equiv \alpha_c(m_o,\kappa)/\alpha_c(0,\kappa)$ (see the legend). The increase of $\tilde{m}_o$ with κ shows that for larger κ a smaller bias m_o is sufficient to realize the same increase in the normalized capacity. The normalization of $\tilde{m}_o$ with $\hat{\alpha}_c$ shows the capacity grows slightly slower than linearly in ${\tilde{m}_o}^{-1}$.

the stability κ for several fixed normalized capacities $\hat{\alpha}_c$, here defined as the ratio between the capacity for non-zero and zero bias $\hat{\alpha}_c(m_o,\kappa) \equiv \alpha_c(m_o,\kappa)/\alpha_c(0,\kappa)$. The increase of $\tilde{m}_o$ with κ shows that for larger κ a smaller bias m_o is sufficient to realize the same normalized capacity. The other important feature, can be seen by the normalization of the curves in $\tilde{m}_o$ with $\hat{\alpha}_c$, which demonstrates that the capacity α_c grows slightly slower than linearly in ${\tilde{m}_o}^{-1}$ for all κ. Extending the asymptotic result for $\tilde{m}_o \to 0$ from [1] to finite stability κ, one finds to leading orders

$$\alpha_c(\tilde{m}_o) = \frac{1}{\tilde{m}_o}\{\log({\tilde{m}_o}^{-1}) + 2\sqrt{2}\sqrt{\log({\tilde{m}_o}^{-1})}\,\kappa + \mathrm{O}(\log[\log({\tilde{m}_o}^{-1})])\}^{-1}. \tag{4}$$

This shows that the sublinearity of α_c is dominated for $\tilde{m}_o \to 0$ by the term $1/\log({\tilde{m}_o}^{-1})$ independent of the stability κ for $\sqrt{\log({\tilde{m}_o}^{-1})} \gg \kappa$. The increase of $\tilde{m}_o$ with κ for constant $\hat{\alpha}_c$ can therefore be explained by the decreasing relevance of κ in determining the capacity for large bias m_o (or large α_c), when compared with $m_o = 0$, where one finds for $\kappa \to \infty$ to leading order $\alpha_c(\kappa) = (1+\kappa^2)^{-1}$. The capacity of the Ising perceptron in the limit $\tilde{m}_o \to 0$ can be calculated along similar lines using self-consistent ansätze for the order parameters (justified by numerical results) yielding the identical result up to leading order in κ as in (4) but for a rescaling of α_c by $2/\pi$.

Above the capacity limit, the perceptron has to misclassify a certain fraction of the example set, expressed in the error rate, ϵ, which depends on the output bias m_o, the stability κ, and on the example load α. It is self-evident that an increase in α beyond the capacity limit is followed by an increase in ϵ. In order to assess how this increase relates to the stability κ, figure 3 shows the normalized load $\hat{\alpha} \equiv \alpha/\alpha_c$ as a function of κ for various fixed error rates ϵ. The dependence on the output bias m_o is illustrated by the choice of $m_o = 0$ and $m_o = 0.9$ in figures 3(*a*) and (*b*), respectively. In order to highlight the scaling behaviour of $\hat{\alpha}$ with the error rate, $\hat{\alpha}$ was adjusted by the normalized remnant error $\bar{\epsilon} \equiv (\epsilon^\infty - \epsilon)/\epsilon^\infty$, where ϵ^∞ is the asymptotic error rate for $\alpha \to \infty$.

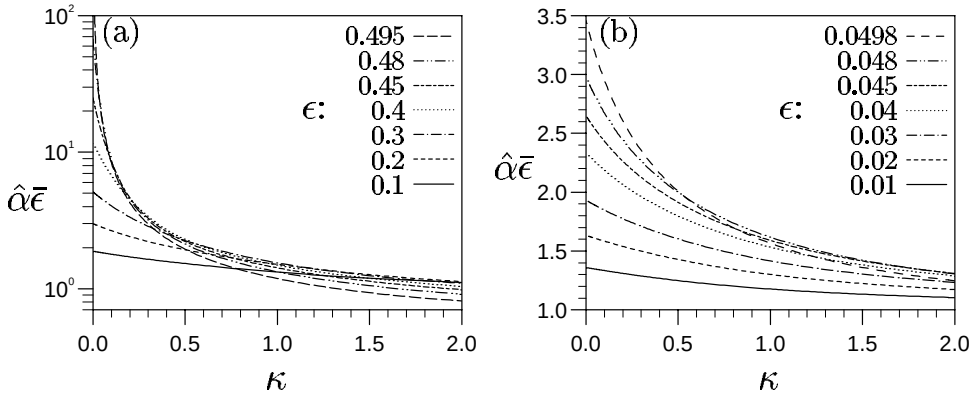

Figure 3. The normalized load $\hat{\alpha} \equiv \alpha/\alpha_c$ (or rather $\hat{\alpha}\bar{\epsilon}$ (see the text) to highlight the scaling of ϵ with α) is shown as a function of the stability κ for several fixed error rates ϵ (see the legend) and two output bias values (*a*) $m_o = 0$ and (*b*) $m_o = 0.9$. The decrease of $\hat{\alpha}$ with κ shows that the error increases more quickly for larger stability even if accounting for the decrease of the capacity. The normalization of $\hat{\alpha}$ with $\bar{\epsilon} \equiv (\epsilon^\infty - \epsilon)/\epsilon^\infty$ helps to highlight the deviation of the scaling of $\bar{\epsilon}$ from α^{-1}.

For both output bias values, one finds that $\hat{\alpha}$ increases (for given $\bar{\epsilon}$) for decreasing stability κ, an effect which is somewhat reverse to the observation made for the capacity limit as described in figure 2. That this effect is more pronounced for $m_o = 0$ than for $m_o \neq 0$ has its root in the changing structure of solution space. For $m_o \neq 0$, the solutions are always characterized by a non-zero threshold θ, reflecting the non-zero output bias. The asymptotic error rate (in α) is given by $\epsilon^\infty = (1 - |m_o|)/2$, which corresponds to deterministically classifying the larger example class correctly and misclassifying the smaller example class by using a threshold of infinite absolute value. The asymptotic error rate is approached by a power law of α^{-1}, corresponding to the rescaling used in figure 3, modified by κ-dependent logarithmic corrections, which explain the κ-dependence found in figure 3. For $m_o = 0$, the behaviour is more complex [22]. Initially, the solutions for all κ are characterized by zero threshold and $\epsilon^{on} = \epsilon^{off} = \epsilon/2$ as expected by the pattern symmetry. However, for any finite stability exists a critical load α_p (with $\alpha_p \to \infty$ for $\kappa \to 0$) at which a phase transition to a solution with non-zero threshold occurs, breaking the symmetry of the patterns ($\epsilon^{on} \neq \epsilon^{off}$). This is caused by the fact that the zero-threshold solution has a κ-dependent asymptotic error rate of $\epsilon^\infty = 1 - H(\kappa) \geqslant \frac{1}{2}$, which is strictly larger than $\frac{1}{2}$ for any finite stability, making it advantageous to adopt the strategy of the non-zero bias case to classify the examples deterministically for $\alpha \to \infty$. The asymptotic error rate is approached by a power laws of $1/\alpha$ (with logarithmic corrections) for the non-zero threshold solution as for the non-zero bias case, but of $1/\sqrt{\alpha}$ (with logarithmic corrections) for the zero threshold solution, applicable for very small κ.

3.3. Employing results for the simple perceptron

In this section it is shown how the results for the capacity limit and the error rates of simple perceptrons demonstrated above can be used to calculate the capacity of the considered constructive algorithms. As an example, consider the capacity limit of a network with K units constructed by the tiling-like algorithm for given initial output bias, m_o, and stability, κ. For convenience, the example load on the whole network and capacity of individual

```
tile_cap := proc(α_c^K)                          % begin procedure
global K, κ, m_o;                                % global variables
local ···;                                       % local variables (here unspecified)
α_1 := Kα_c^K; m̃_1 := 1 − m_o;                   % initialize algorithm
for (i = 1, K − 1) do                            % loop over the erroneous perceptrons
    ε_i := error_calc(α_i, m̃_i, κ);              % error rate calculating procedure
    α_{i+1} := α_i; m̃_{i+1} := 2ε_i;             % calculate new parameters
od;                                              % have reached last perceptron
α_c := perc_cap(m̃_K, κ)                          % calculate capacity limit
RETURN(α_c − α_K);                               % return difference between capacity and load
end;
```

Figure 4. A symbolic capacity calculation procedure of the tiling-like algorithm called by an all-purpose root solving routine.

perceptron units are expressed in terms of $\alpha \equiv p/N$ and α_c, respectively, whereas the capacity of the whole network is defined as $\alpha_c^K \equiv \alpha/K$ (and $\alpha_c^1 = \alpha_c$).

Assume that the current guess of the network capacity at iteration j is α_j^K resulting in an initial example load of $\alpha_1 = K\alpha_j^K$ with output bias $m_1 = m_o$. These parameters together with the desired stability κ determine the error rate ϵ_1 made by the first perceptron $\mathcal{U}_1$. The example load α_2 and bias m_2 of the second perceptron $\mathcal{U}_2$ result by simply applying the rules of the algorithm: the load $\alpha_2 = \alpha_1$, since the complete training set is used, and the bias $m_2 = 1 - 2\epsilon_1$ (or the more natural parametrization $\tilde{m}_2 = 1 - m_2 = 2\epsilon_1$), since the target of all examples but the erroneous ones is $+1$. Obviously, these parameters determine ϵ_2 and this procedure is repeated until the last perceptron $\mathcal{U}_K$ which is supposed to have reached its capacity $\alpha_K = \alpha_1$ for $\tilde{m}_K = 2\epsilon_{K-1}$. Therefore, the actual capacity limit α_c for $\tilde{m}_K$ is calculated and compared with α_K: if the last perceptron is below its capacity limit, the true capacity $\alpha_c^K > \alpha_j^K$ otherwise $\alpha_c^K < \alpha_j^K$ and a root solving routine can be employed to solve

$$[\alpha_c(\tilde{m}_K) - \alpha_K] = 0 \tag{5}$$

as a function of α_c^K. A symbolic program for the procedure called by the root solver can then be written as outlined in figure 4.

It is now more clear, what the relevance of the results for the simple perceptron presented in section 3.2 is. The error rate of the previous perceptron determines the output distribution bias of the current perceptron and the curves of constant error in figure 3 are curves of constant bias for the next generation. With each step of the algorithm, the decreased 'bias' $\tilde{m}_o$ leads to a reduced error rate, until the capacity limit for the current bias is larger than the current load α. The influence of the stability is therefore twofold. The increase in $\tilde{m}_o$ for constant $\hat{\alpha}_c$ with κ (observed in figure 2) should have the effect of increasing the normalized capacity limit, $\hat{\alpha}_c^K \equiv \alpha_c^K/\alpha_c$, with κ of the whole network, whereas the decrease of $\hat{\alpha}$ for fixed error with κ should have qualitatively the opposite effect.

For the upstart algorithm similar consideration are applied to yield a procedure for a root solver that calculates the capacity of the created networks. The resulting procedures are much more complicated and summarized in appendix C.

Note, that the numerical uncertainty in the solution of the order parameter, mainly caused by the numerical integration inaccuracy, results in an inaccurate error rate calculation. Propagating the upper and lower bound of the error rate in each generation separately through

the whole network gives estimated error bars in the capacity†. Although the relative capacity error increases with network size it never exceeded 10^{-4} and could therefore be neglected.

4. Numerical capacity results

Within the uncorrelated approximation, the procedures described within section 3.3 (and appendix C) have been used to calculate the capacity of the considered constructive algorithms. This section will roughly fall into three parts. In section 4.1 numerical capacity results for an unbiased random mapping, usually considered in capacity calculations, will be presented as a function of K using the various algorithms, different stabilities κ, and employing either the RS or one-step RSB ansatz. In section 4.2 biased output distributions will be considered for $\kappa = 0$ and the different algorithms compared. This is followed by an investigation into the capacity when an Ising weight prior is used for the individual perceptrons instead of a spherical weight prior employed in the previous sections. The functional behaviour of these numerical results will then be analysed for finite K and suggestions are made concerning the asymptotic limits of the capacity for $K \to \infty$ in section 5.

4.1. Capacity for unbiased outputs

Previous capacity calculation for MLPs have only investigated zero stability for unbiased output distributions. In this section, one aim is therefore to assess the influence of finite stability on the asymptotic functional form of the capacity limit for unbiased output distributions. Another goal is to compare the capacity‡ between the networks built by the different constructive algorithms considered§.

In figure 5, the capacity limit of networks constructed by the upstart II algorithm is shown as a function of the number of hidden units for various stabilities κ for the uncoupled RS (figure 5(a)) and one-step RSB (figure 5(b)) ansätze. Note, that the capacity curves have been normalized by $\alpha_c^1(\kappa)$ for presentational reasons. For both ansätze, one finds that the capacity grows monotonically for small stabilities. for larger stabilities the capacity decreases initially for small K but increases for larger K. This non-monotonic behaviour can be explained by the initial inefficiency of the upstart algorithm by tackling the two type of errors with two different unit types, such that the network grows directly from one to three units. For even larger stabilities, one finds that the capacity actually decreases in the $K \to \infty$ limit‖.

Notice that for finite stability κ and large enough K, one finds a kink in the capacity curve due to the phase transition in the solution of the first perceptron from zero to finite threshold, which leads to a breaking of the error symmetry and consequent network symmetry for upstart II. The asymmetry in the error also leads to the jags in the capacity as

† This technique was compared with the change in the capacity resulting from relaxing the accuracy requirements for the numercial integration over several order of magnitude and it was found that the error propagation method overestimates the capacity error by about two orders of magnitude.

‡ For brevity, the capacity limit of networks constructed by an algorithm will often be referred to just as the capacity of the algorithm. It is also only fair to emphasize that our numerical solutions have not been compared with simulation results as the latter do not exist in the literature for the cases considered in this paper, and are notoriously difficult to carry out, especially for high values of K which are the focus of our study.

§ Note, that for unbiased output distributions, the two selection criteria considered for the variants of the upstart algorithm are identical, and the two variants will therefore be referred to as upstart II and III in this section.

‖ Note that the decrease is in the capacity per weight of the networks, the capacity of the network still increases linearly in K to leading order.

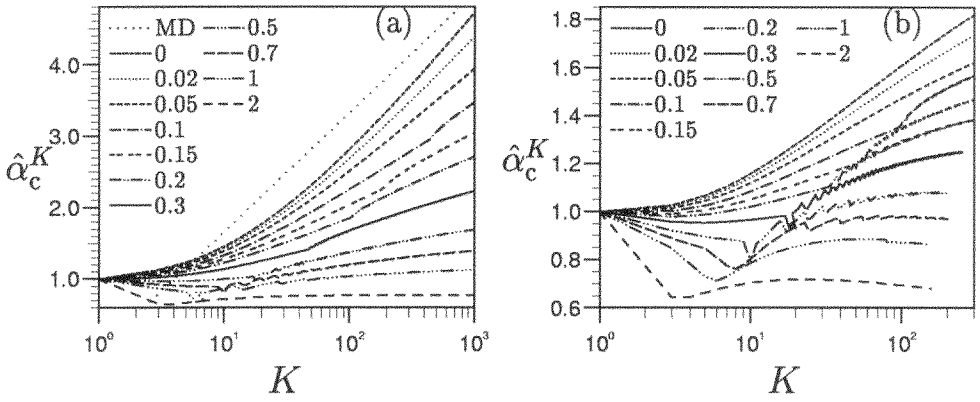

Figure 5. The normalized capacity limit $\hat{\alpha}_c^K \equiv \alpha_c^K/\alpha_c^1$ (where $\alpha_c^1(\kappa)$ is the capacity of a simple perceptron) of networks constructed by upstart II is shown as a function of the number of hidden units K for several stabilities κ (see the legend) for the uncoupled (*a*) RS and (*b*) one-step RSB ansätze. The RS capacity violates the superimposed MD bound for small stabilities and large enough K in the range of hidden units investigated.

the two types of units cease to saturate simultaneously. Furthermore, due to the increasingly deterministic classification of the first perceptron (for an explanation see section 3.2), wrongly-off errors become increasingly more common than wrongly-on errors, resulting in the upstart II algorithm constructing much more $\mathcal{U}^+$ than $\mathcal{U}^-$ nodes†. In fact, for certain stabilities, one finds that a $\mathcal{U}^-$ node can actually vanish before a new $\mathcal{U}^+$ node needs to be created, when increasing the load α on the first perceptron, leading to a decrease in total network size (e.g. $\kappa = 0.3$ for the one-step RSB ansatz, where the decrease in network size at $K = 20$ is apparent).

In comparison with the bounds and capacity limits known for fixed-architecture MLPs, one finds that the capacity curves for networks built by upstart II violate the MD bound within the uncoupled RS ansatz for large enough number of hidden units; in the curve shown the slope of the RS curve is larger than the MD bound for $K \approx 80$‡. However, within the uncoupled one-step RSB ansatz the MD bound is not saturated. Although, the one-step RSB results initially seem to predict a logarithmic increase of the capacity§ $\alpha_c^K \propto \log K$ for small stabilities and network sizes $20 \lessapprox K \lessapprox 150$ as reported in preliminary work [64], this functional description is inadequate for larger stabilities and/or larger network sizes and a detailed analysis will be carried out in section 5.

In figure 6, the capacity resulting from the different constructive algorithms considered is investigated for a few stabilities. For both small and large stabilities, shown in figures 6(*a*) and (*b*) respectively, one finds that the tiling-like algorithm has a larger capacity for small network size and for small stabilities than both variants of the upstart algorithm, which may be mainly attributed to the fact that the tiling-like algorithm attempts to correct both error

† Or vice versa due to random symmetry breaking in the first perceptron.

‡ The MD bound is an asymptotic bound as it relies of Stirling's approximation, we therefore consider the violation of the slope of the MD bound as the significant indication rather than the actual capacity value itself (although we also find an absolute violation for very large K). Within RS the slope of the tiling-like and the upstart III algorithms violates the MD bound above $K \approx 20$ and $K \approx 50$, respectively. Note that this result only holds for the range of $K \leqslant 4000$ investigated since the asymptotic values could not be calculated self-consistently in section 5.

§ The estimated prefactors are significantly smaller than the $(\log 2)^{-1}$ of the MD bound.

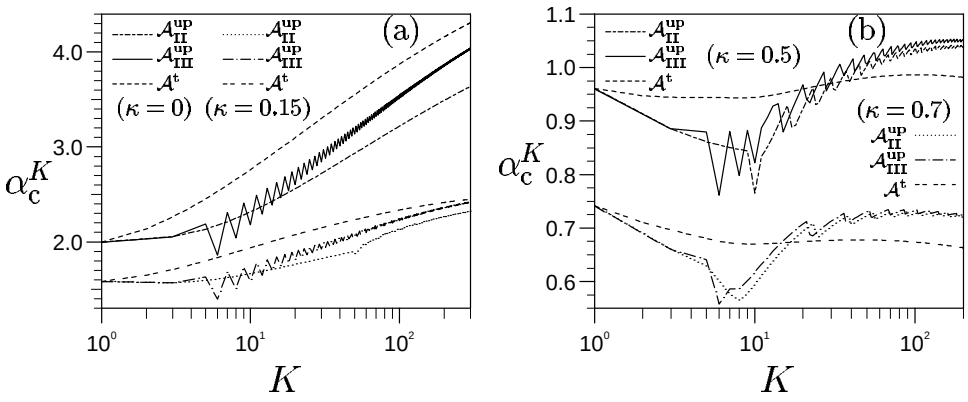

Figure 6. The capacity limit $\alpha_{\rm c}^K$ of networks constructed by the upstart II ($\mathcal{A}_{\rm II}^{\rm up}$), upstart III ($\mathcal{A}_{\rm III}^{\rm up}$), and tiling-like ($\mathcal{A}^{\rm t}$) algorithms is shown as a function of the number of hidden units K for several stabilities: (*a*) $\kappa = 0$ and $\kappa = 0.15$; (*b*) $\kappa = 0.5$ and $\kappa = 0.7$.

types simultaneously.

However, asymptotically the tiling-like algorithm is less efficient than both variants of the upstart algorithm for larger stabilities, but even for small stabilities the capacity curves for upstart networks (in particular for upstart III) approach those of the tiling-like network. This behaviour may be explained by the fact that the upstart algorithm eliminates part of the original training data in the training set of consecutive units.

This argument can also explain the fact that upstart III is more efficient than upstart II for all stabilities as it eliminates in general more patterns from the training set. This advantage, however, becomes less significant for large κ, where almost all errors are wrongly-off and consequently almost all units are of the $\mathcal{U}^+$ type beyond the phase transition of the first perceptron. The fraction of wrongly-on errors eliminated from the training sets of $\mathcal{U}^+$ units is therefore small and the two versions behave similarly.

4.2. Capacity for biased outputs and zero stability

Similarly to exploring finite stabilities as above, it is interesting to address the influence of biased output distributions on the capacity limit of MLPs for which no results are a known in the case of fixed architectures. Due to the symmetry it is sufficient to study $m_{\rm o} < 0$ w.l.o.g. for $m_{\rm o} > 0$ the rôles of of wrongly-on and wrongly-off errors reverse and consequently the rôles of $\mathcal{U}^-$ and $\mathcal{U}^+$ units for the upstart algorithm. Again, we would like to compare the capacity between the different constructive algorithms considered and for the variants of the upstart algorithm also the criteria selecting the next unit type.

In figure 7(*a*), the capacity limit of networks constructed by the upstart IIb algorithm is shown as a function of the number of hidden units for various 'biases' $\tilde{m}_{\rm o} \equiv 1 - |m_{\rm o}|$ for the uncoupled one-step RSB ansatz. Again, the capacity curves have been normalized by $\alpha_{\rm c}^1(\tilde{m}_{\rm o})$ for presentational reasons. Although, the normalized capacity limit for biased output distributions is initially larger than less biased or unbiased output distribution for $K > 6$, one finds asymptotically (in K) that the (normalized) slope† seems to decrease for increasing bias (see below), suggesting that the constructive algorithms are less efficient (when compared with a single perceptron) in exploiting the bias of the output distribution.

† For larger bias the actual slope is still much larger due to the normalization factor $\alpha_{\rm c}^1(\tilde{m}_{\rm o})$ that scales with (4).

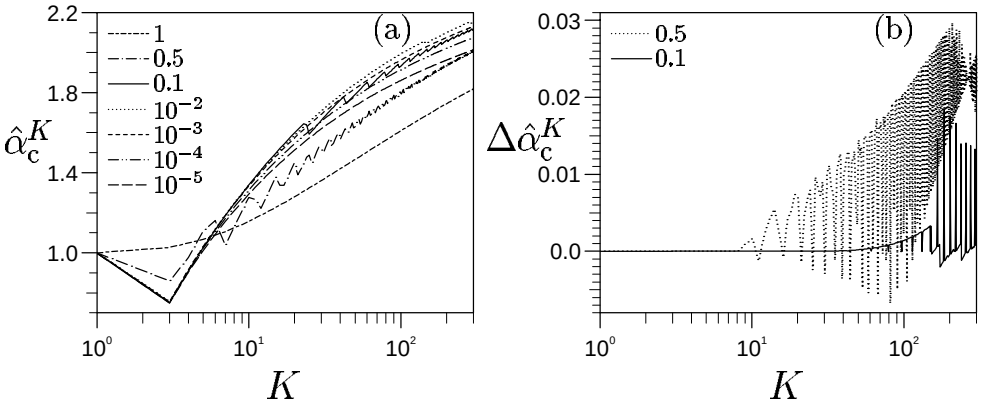

Figure 7. (*a*) The normalized capacity limit $\hat{\alpha}_c^K \equiv \alpha_c^K/\alpha_c^1$ (where $\alpha_c^1(\tilde{m}_o)$ is the capacity of a simple perceptron) of networks constructed by upstart IIb is shown as a function of the number of hidden units K for several 'biases' $\tilde{m}_o$ (see the legend) for the uncoupled one-step RSB ansatz. (*b*) To highlight the influence of the selection criteria, the difference between the normalized capacity limits $\Delta\hat{\alpha}_c^K = \hat{\alpha}_c^K(\mathcal{A}_{\rm IIa}^{\rm up}) - \hat{\alpha}_c^K(\mathcal{A}_{\rm IIb}^{\rm up})$ of the two selection criteria is shown for $\tilde{m}_o = 0.5$ and $\tilde{m}_o = 0.1$, suggesting that on average upstart IIa outperforms upstart IIb.

The curves are jagged for finite but small bias, since both unit types are constructed and the units do not saturate simultaneously. For very large bias, the larger example class is almost deterministically classified correctly and only one type of unit is constructed to correctly classify the smaller class, again leading to smooth capacity curves.

The influence of the unit creation criterion is therefore only important for small bias and its influence on the capacity limit is depicted in figure 7(*b*) and shows that upstart IIa is slightly more efficient than upstart IIb. A more detailed examination of the constructed networks shows that both criteria are not ideal. For unbiased output distribution, one finds that after the breaking of the network symmetry (i.e. beyond the phase transition of the solution of the first perceptron for finite κ), the algorithm initially only creates $\mathcal{U}^+$ units until both error types have the same frequency after which it alters between the unit types. We find this creation scheme the most natural and believe that it is probably also optimal. For finite output bias, criterion (*a*), using the number of errors as the decision criterion, builds $\mathcal{U}^-$ units too early into the network, i.e. instead of alternating between unit types at the end the $\mathcal{U}^-$ units are dispersed less frequently over a wider unit number range. This may be considered a wasteful use of $\mathcal{U}^-$ units.

Criterion (b), basing its selection on the number of errors made normalized by the size of the its target class, alleviates this problem leading to networks with fewer $\mathcal{U}^-$ units for fixed total network size, however, in this case we find that the creation of $\mathcal{U}^-$ units tends to be left too late, leading to extra $\mathcal{U}^+$ units that have to be created at the very end to correct the few wrongly-on errors the $\mathcal{U}^-$ units make. In fact, for both creation criteria and for the last few units, we find sometimes that the algorithm actually decides on building a unit type which is below its capacity limit whereas the other unit type is above its capacity, i.e. criterion (a) selects a $\mathcal{U}^-$ although it should have selected a $\mathcal{U}^+$ and vice versa for criterion (b)†.

Both criteria are therefore not optimal, criterion (a) selects $\mathcal{U}^-$ units too early and

† To cater for those few cases, we have decided to amend both criteria such that the algorithm always selects a unit above its capacity limit first.

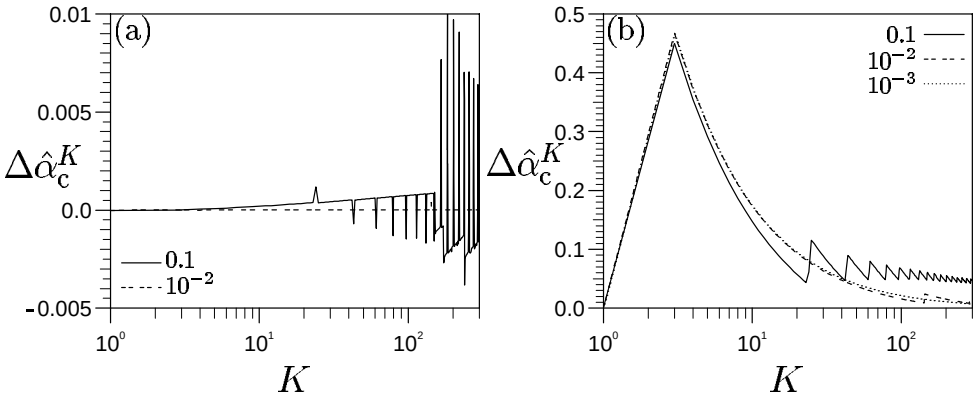

Figure 8. The influence of the constructive algorithm $\mathcal{A}$ is assessed by plotting the difference between the normalized capacity limits $\Delta\hat{\alpha}_c^K = \hat{\alpha}_c^K(\mathcal{A}) - \hat{\alpha}_c^K(\mathcal{A}_{\mathrm{IIb}}^{\mathrm{up}})$ for (*a*) where $\mathcal{A}$ is the upstart IIIb and (*b*) where $\mathcal{A}$ is the tiling-like algorithm and several bias values (see the legend).

criterion (b) selects $\mathcal{U}^-$ units to late. A better criterion should therefore compromise somewhat between these two; however, we were not able to devise a more suitable objective criterion.

In figure 8, upstart IIb is compared with upstart IIIb and the tiling-like algorithm. For the versions of the upstart algorithm (figure 8(*a*)), the difference in capacity is very small and decays rapidly for increasing bias, since the difference in the training sets becomes negligible as the fraction of wrongly-on errors goes to zero and only one $\mathcal{U}^-$ unit is created in all network sizes investigated.

The difference of the upstart IIb to the tiling-like capacity (figure 8(*b*)) is significant for small networks, due to the separate treatment of each error type. For large bias, the tiling-like capacity approaches the upstart IIb capacity rapidly, as almost all errors become wrongly-off and only few $\mathcal{U}^-$ units are created, consequently leading to almost identical training sets and networks besides the extra $\mathcal{U}^-$ unit built by the upstart algorithm.

This should be contrasted to large stabilities κ, where the upstart algorithm also constructs almost entirely a single unit type; however, in this case, the difference in the training sets between the upstart and tiling-like algorithm does not vanish asymptotically in κ. The fraction of correctly-on patterns excluded from the training set of all $\mathcal{U}^+$ units in the upstart algorithm approaches $\frac{1}{2}$ of all patterns for later units in the unbiased output distribution case, whereas this fraction approaches $\tilde{m}_o/2$ in the biased output distribution case—vanishing for $\tilde{m}_o \to 0$.

For small bias, the picture is less clear. The tiling-like capacity seems to decay to a value which has approximately a constant difference to the upstart capacity, although we have found for zero bias, that at least the upstart III capacity curve approaches that of the tiling-like algorithm. This difference may be explained by the suboptimality of both upstart selection criteria for $m_o \neq 0$.

4.3. Capacity for the Ising perceptron

Up to now, we have only considered the constructive algorithms using spherical perceptron with real valued weights of arbitrary accuracy as their basic building block. In realistic implementations weights are only stored up to a certain accuracy and especially for VLSI implementations Ising (binary $\{-1, 1\}$) weights are an often considered alternative. In this

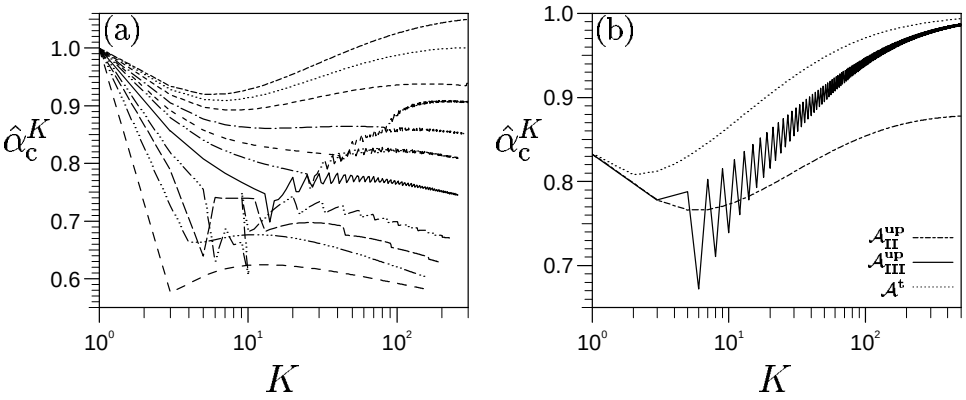

Figure 9. This figure illustrates the differences when using the Ising rather than the spherical perceptron as the basic building block of the constructive algorithms. (*a*) The normalized capacity limit $\hat{\alpha}_c^K \equiv \alpha_c^K/\alpha_c^1$ for networks constructed by upstart II is shown as a function of the number of hidden units K for several stabilities κ (from top to bottom $\kappa = 0$, 0.02, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5, 0.7, 1.0 and 2.0, as in figure 5) for the uncoupled one-step RSB ansatz. (*b*) The capacity limit α_c^K of networks constructed by the upstart II ($\mathcal{A}_{II}^{up}$), upstart III ($\mathcal{A}_{III}^{up}$), and tiling-like ($\mathcal{A}^t$) algorithms is shown as a function of the number of hidden units K for $\kappa = 0$.

section, we therefore investigate the influence of an Ising weight prior for the perceptron, usually referred to as the *Ising perceptron* (in contrast to the *spherical* perceptron with real weights), on the capacity of the resulting networks. For brevity we will only consider unbiased output distributions and mainly networks constructed by the upstart II algorithm.

In figure 9(*a*) the normalized capacity limit of networks constructed by the upstart II algorithm is shown as a function of the number of hidden units for various stabilities κ for the uncoupled one-step RSB ansatz. As for the spherical perceptron, we find that not only the capacity but also the slope of the normalized capacity curves decrease with increasing stability. In comparison with the spherical perceptron, the normalized capacity is, however, much smaller; in fact, for all stabilities the normalized capacity decreases initially for small K. Although the capacity increases for very small κ for larger K, the curves flatten out asymptotically. For larger κ, the decrease of $\hat{\alpha}_c^K$ is only abated briefly due to the phase transition in the solution of the first perceptron (which occurs for much smaller α in the Ising perceptron), leading to the already observed kink in the capacity curves and the possibility of a decrease of the number of units for increasing load α.

In figure 9(*b*) the dependence of the capacity curves on the constructive algorithm is investigated for $\kappa = 0$. As for small stability in the spherical perceptron, the tiling-like algorithm has the largest capacity for the range of hidden units investigated. Whereas, the upstart III capacity closes the gap in the capacity to the tiling-like algorithm, the upstart II algorithm seems to be asymptotically less efficient for this stability.

5. Analysis of the capacity

Although the visual inspection of the capacity curves already reveals some information about the efficiency of the considered constructive algorithms as a function of K and κ, it would be more useful to be able to model the capacity curves at least for large K with a reasonable functional form.

As mentioned above, a functional form of the capacity limit α_c^K linear in log K cannot fit

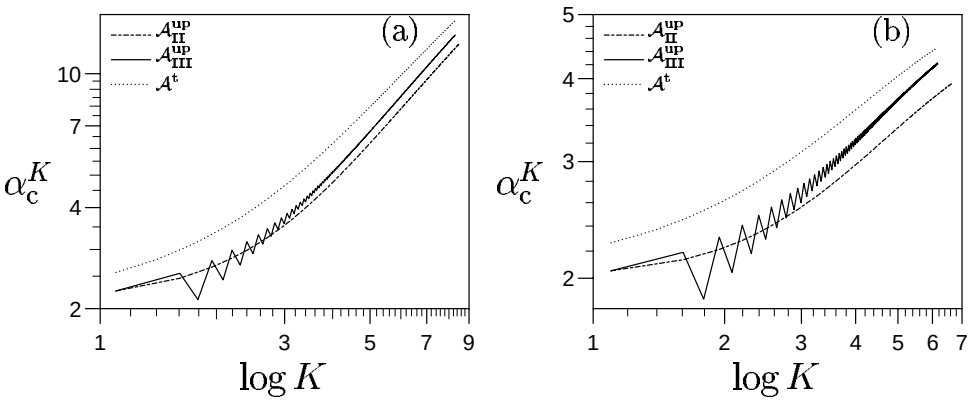

Figure 10. The power-law relationship between $\alpha_{\rm c}^{K}$ and $\log K$ (6) is shown to hold approximately for all considered constructive algorithms [upstart II ($\mathcal{A}_{\rm II}^{\rm up}$), upstart III ($\mathcal{A}_{\rm III}^{\rm up}$), and tiling-like ($\mathcal{A}^{\rm t}$)] for $\kappa = 0$ and (*a*) RS; (*b*) one-step RSB.

the curves adequately for large κ and/or K. For $\kappa = 0$ and unbiased output distributions, the capacity results for committee and parity machines within the replica framework [13, 19–21] yield power laws in $\log K$

$$\alpha_{\rm c}^{K} \simeq c[\log K]^{n} \tag{6}$$

with $n = \frac{1}{2}$ and $n = 1$ respectively. This suggests that power laws may hold also for constructive algorithms at least for $\kappa = 0$ and $m_{\rm o} = 0$ which we will investigate first in this section. Later, we will extend our analysis to finite $m_{\rm o}$ and finite κ.

5.1. Zero stability and unbiased output distributions

In figure 10, the power-law ansatz is scrutinized for the capacity curves of all considered algorithms for both RS and one-step RSB, by plotting $\alpha_{\rm c}^{K}$ versus $\log K$ on a log–log scale. We find that the ansatz is reasonable for both cases, although the slope for one-step RSB drops slightly for larger $\log K$. The determination of accurate exponents n (which are arguably more relevant than the prefactor c), however, is difficult for two reasons. As the power law is in $\log K$, the range for fitting the exponent is relatively short, since the calculation of the capacity is computationally quite expensive†, making it impossible to calculate capacities over several decades of $\log K$. Furthermore, the power-law behaviour in $\log K$ seems impure, as locally (i.e. around a particular K value) calculated exponent values $n(K)$ exhibit small constant shifts. This may, for example, be caused by superimposed corrections decaying in $\log(\log K)$. This shift may be a cumulative effect of the error calculations propagated through the perceptrons as well as of the capacity calculation for the last perceptron(s).

These difficulties will become more apparent in the course of this analysis. The asymptotically derived exponent values should therefore be seen as a local snapshot. Bearing this in mind, finite K measurements $n(K)$ were derived using a moving regression window. The resulting curve of $n(K)$ was then either averaged for the largest K available resulting in a local mean approximation $n_{\rm l}$ or extrapolated to $1/K \to 0$ using linear and quadratic

† For example, the calculation of the capacity limit for the tiling-like algorithm to up to $K = 300$ takes approximately 2 days (10 min) of CPU on a state-of-the-art workstation within the one-step RSB (RS) ansatz.

Table 2. The estimated power-law exponents n_l and n_e for $\kappa = 0$, the considered algorithms, and several replica ansätze for two values of K in order to highlight the occurance of the systematic errors.

		RS			
$\mathcal{A}$	K[a]	n_l	n_e[b]	n_l	n_e
$\mathcal{A}_{II}^{up}$	300	0.554(2)	0.48(2)	1.291(0)	1.279(0)
	750	0.479(1)	0.36(2)	1.264(0)	1.243(0)
$\mathcal{A}_{III}^{up}$	300	0.602(1)	0.54(1)	1.305(0)	1.290(0)
	450	0.557(1)	0.47(1)	1.272(0)	1.247(0)
$\mathcal{A}^{t}$	300	0.468(1)	0.36(3)	1.174(0)	1.167(0)
	450	0.428(1)	0.31(2)	1.158(0)	1.145(0)

[a] The numbers given for K apply to one-step RSB; for RS $K = 1000$ and $K = 4000$ were used instead (lower and upper limit respectively).

[b] The error in the exponents is usually given for the leading digit in brackets only, i.e. 0.49(2) is equivalent to 0.49 ± 0.02. The error is given as (0) when smaller than the last significant digits given.

regression models yielding n_e. This extrapolated estimate n_e should be seen as an indication in which direction n_l is moving rather than a true extrapolation to n for $K \to \infty$.

These calculations still depend on the length of the regression window and the number of resulting $n(K)$ values included in the mean and extrapolated estimates, due to the noisy capacity curves for the upstart algorithms and the small constant shifts in $n(K)$. To minimize these effects, an average was taken over different lengths of the regression windows for local $n(K)$ values as well the number of $n(K)$ values included in the extrapolation. The values of n_l and n_e were then determined by a weighted mean of the individual resulting models by their respective likelihoods.

The final results for n_l and n_e are shown in table 2 for the various constructive algorithms and both RS and one-step RSB ansätze for the spherical perceptron. For this case, the power-law exponents were calculated once for the usual range of hidden units explored ($K = 300$ for one-step RSB and $K = 1000$ for RS) and in most cases re-evaluated for larger K ($450 \leqslant K \leqslant 750$ for one-step RSB and $K = 4000$ for RS) in order to verify the size of the systematic errors.

For all algorithms, we observe that the resulting extrapolated estimate n_e is smaller than the local mean n_l, suggesting that n_l could be an upper bound (practical rather than rigorous) to the true exponent n. As expected, the extrapolated estimates n_e themselves are not very accurate; the re-evaluated local mean n_l for the larger K value shows a strong shift and is in many cases smaller than the extrapolation n_e for smaller K. It may be argued that this is due to the extrapolation carried out in $1/K$ rather than in $1/\log K$, however, such an extrapolation is not advisable within the values of K explored. From table 2, we can therefore conclude that the K values explored are not in the asymptotic regime.

Comparing the exponents between the different algorithm may suggest that both upstart algorithms are asymptotically more efficient than the tiling-like algorithm and upstart III outperforms upstart II, with the reservation that the large finite size effects may undermine this observation. Furthermore, one can speculate upon whether the exponents n for RS are asymptotically larger than 1, i.e. violate the MD bound, as their finite K estimates suggest.

Finally, note that the exponents were calculated under the assumption of subsequent perceptrons being uncorrelated and the effect of correlation (and also higher RSB step) may

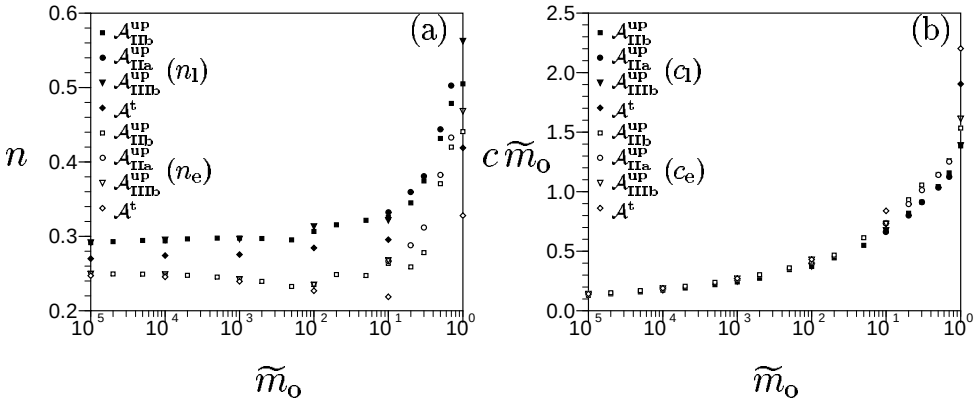

Figure 11. (*a*) The power-law exponent estimates n_l and n_e for the capacity limit $\alpha_c^K \propto [\log K]^n$ are shown as a function of the 'bias' $\tilde{m}_o \equiv 1 - |m_o|$ for the various algorithms (upstart IIa ($\mathcal{A}^{up}_{IIa}$), upstart IIb ($\mathcal{A}^{up}_{IIb}$),upstart IIIb ($\mathcal{A}^{up}_{IIIb}$), and tiling-like ($\mathcal{A}^{t}$)) for the uncoupled one-step RSB ansatz. (*b*) The corresponding prefactor c_l and c_e, where the values were adjusted by the dominant linear scaling in $\tilde{m}_o{}^{-1}$. The local values were determined for $K = 300$ and are denoted by filled symbols, whereas the extrapolation estimates are represented by open symbols (see the legend). The estimation error for all estimates does roughly not exceed more than five times the size of the symbols, and is about their size in many cases, especially for small $\tilde{m}_o$.

cause a shift to smaller (local) n values, similarly to the transition from RS to one-step RSB. Since RSB in the single perceptron is believed to be more relevant than correlations, this correction should, however, be significantly smaller and likely negligible in comparison with finite size effects in K.

One way to probe further into the asymptotic regime for fixed K is by studying large bias. For $\tilde{m}_o \to 0$, the relevant α values are much larger, and the error rates and the capacity of most units are closer to their asymptotic expansions (see [22] and (4)).

5.2. Analysis for biased outputs and zero stability

To assess the influence of the output bias more objectively and also to gain some qualitative insight into the likely behaviour for much larger K in the more interesting case of one-step RSB, we have calculated n_l for $K = 300$ and n_e along the same lines as above and present the results in figure 11(*a*).

For all algorithms, we find that n initially decreases with increasing bias m_o, before both estimates level off. This result is inconsistent since n must either be constant or increase with m_o, otherwise the capacity curve for a smaller bias would eventually cross that for a larger bias. This result could have been anticipated from the decreasing slope for large bias observed in the raw capacity curve (see figure 7(*a*)). Evidently, this contradiction causes no actual violation for any practical range of hidden units†, since the prefactor estimates c scale to leading order‡ with $c \propto \tilde{m}_o{}^{-1}$ (see figure 11(*b*)).

The question remains open whether the asymptotic exponents n will have any functional dependence on m_o. If n were independent of m_o, n would also have to be independent of the

† For example, for the upstart III algorithm the capacity curve for $\tilde{m}_o = 1$ were to cross the one for $\tilde{m}_o = 10^{-5}$ for $K \approx \exp(10^{17})$.

‡ Note that the logarithmic correction to this leading behaviour for c resembles the result for the perceptron in (4).

considered constructive algorithms for any bias since for $\tilde{m}_o \to 0$ their differences vanish as explained in section 4.2. The performance of the constructive algorithms studied would then only vary in the prefactor, in contrast to the case of fixed architectures where n can be architecture dependent (see (6)).

Finally, it is worthwile illustrating how such inconsistency can arise. Consider the asymptotics of the perceptron capacity for $\tilde{m}_o \to 0$, for which an asymptotic expansion can be derived explicitly (4). A numerical determination of the leading asymptotic behaviour in $\log(\tilde{m}_o{}^{-1})$ would require extremely small $\tilde{m}_o$ values. For the κ-corrections to become negligible $\sqrt{\log(\tilde{m}_o{}^{-1})} \gg 2\sqrt{2}\kappa$, i.e. representing the inequality as a small factor δ requires $\tilde{m}_o \approx \exp(-8\delta^2\kappa^2)$. The true κ-independent exponent (-1) is therefore numerically almost impossible to predict. In fact, for any small but finite $\tilde{m}_o$, a numerical evaluation of the exponent for finite κ is always strictly larger than -1 and increasing with κ. At face value, this result would also inconsistently predict that asymptotically the capacity curves for larger κ cross those of smaller κ. Considering, that such numerical difficulties already mar asymptotic results for a simple perceptron, it may be of no surprise that consistent and accurate asymptotic results could not be extracted in the case studied here.

5.3. Analysis for finite stability

Although we have shown in the previous section that consistent power-law exponents cannot be determined for the range of hidden units available, it is nevertheless interesting to study the local exponents for finite stabilities to gain a qualitative insight.

For finite stability, the fit of a power-law model on the capacity curves themselves deteriorates for increasing stability. This is mainly due to the non-negligible corrections to the asymptotic capacity limit of the last perceptron as described above. For example, if a power-law model is fit to the capacity limit of a simple perceptron as a function of $\log(\tilde{m}_o{}^{-1})$ for $\kappa = 2$ and the range of $\tilde{m}_o{}^{-1}$ relevant for the bias of the last perceptron within the range of hidden units explored, the estimated exponent is around $-\frac{1}{2}$ and increasing systematically towards the correct value -1.

In order to be able to extend the analysis to finite stability, it is therefore necessary to separate the cumulative effect of the errors and the capacity of the last perceptron. Consider, for example the tiling-like algorithm. The capacity of the complete network in terms of the capacity of the last perceptron is according to (5)

$$\alpha_c^K = \frac{1}{K}\alpha_c(\widetilde{m}_K). \tag{7}$$

Since $\widetilde{m}_K \to 0$ for $K \to \infty$, one can use the asymptotic expansion (4) of α_c to express the capacity solely in terms of $\widetilde{m}_K$ (to leading order for $\widetilde{m}_K \to 0$)

$$\alpha_c^K \simeq \frac{1}{K}\frac{1}{\widetilde{m}_K \log(\widetilde{m}_K^{-1})}. \tag{8}$$

For $\widetilde{m}_K$ we make the (numerically justified) ansatz

$$\widetilde{m}_K \simeq \frac{b}{K[\log K]^l} \tag{9}$$

with $l = l(\kappa, K)$ and $b = b(\kappa, K)$ being weakly dependent on K. The final result for the capacity then becomes to leading order

$$\alpha_c^K \simeq \frac{1}{b}[\log(K)]^{l-1} \equiv \tilde{c}[\log(K)]^{\tilde{n}} \tag{10}$$

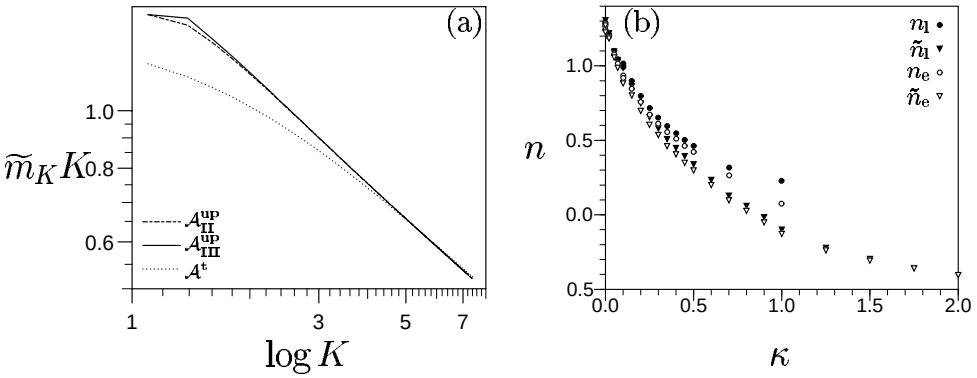

Figure 12. (*a*) The power-law relationship between $\widetilde{m}_K K$ and $\log K$ (9) is shown to hold approximately for $\kappa = 2$ and RS (and all constructive algorithms). Similar results can be obtained for other stabilities and one-step RSB. (*b*) The exponent values for upstart II extracted either from the capacity curve itself (n) or from the bias on the last unit ($\tilde{n}$) are very similar for small stability confirming that the two methods are equally suited in this regime. For larger stability the two approaches differ mainly due to the slowly decaying corrections to the asymptotic capacity limit of the last perceptron for $\tilde{m}_{\rm o} \to 0$ (see the text).

where the exponent $\tilde{n}$ (and the prefactor $\tilde{c}$) are now purely determined from the error calculation avoiding the slowly vanishing terms in the perceptron capacity (4) for finite κ. For the variants of the upstart algorithm similar considerations can be applied, leading asymptotically to the same equation for the exponent n, but with a prefactor which asymptotically depends on the initial output-distribution bias, the exact derivation of which can be found in appendix D.

The adequacy of this approach is scrutinized in figure 12(*a*), where (9) is shown to hold well for $\kappa = 2$ and RS, by plotting $\widetilde{m}_K K$ versus $\log K$ on a log–log scale for all considered algorithms. In figure 12(*b*), the differences between the RS power-law exponent estimates resulting from the capacity curve itself (n) and indirectly via the bias on the last unit ($\tilde{n}$) are depicted for upstart II.

For small stability κ, the two estimates almost coincide. The small deviations can be explained by two factors. First, higher order terms in the expansion of the last perceptron's capacity limit have been neglected in (8). Second, the indirect calculation of the exponent value suffers from some systematic errors in the case of the upstart algorithm. For small finite stabilities the capacity limit can be reached purely due to a change in architecture without either of the units ever being very close to their respective saturation limits. Furthermore, the capacity as a function of $\widetilde{m}_K$ has further corrections which only strictly vanish for $\widetilde{m}_K = 0$ (see appendix D). The re-evaluated exponents are explicitly listed in table 3 for $\kappa = 0$ to allow a comparison with the original values table 2. Studying both tables, the largest systematic differences can be found in the case of the tiling-like algorithm†, whereas the deviations for the upstart algorithm are very small, which may be explained by the systematic corrections cancelling neglected higher order terms in the capacity. Again, these differences show, that we cannot expect quantitatively accurate results.

Returning to figure 12(*b*), for large stability κ, the n estimates differ significantly between the calculation methods. The difference approximately corresponds to the expected

† The difference between n and $\tilde{n}$ of around 0.08–0.1 correspond almost exactly to the expected finite $\tilde{m}_{\rm o}$ corrections to the asymptotic perceptron capacity limit for $\kappa = 0$ (of $O(\log[\log(\tilde{m}_{\rm o}{}^{-1})])$).

Table 3. The estimated power-law exponents n_l and n_e for $\kappa = 0$, the considered algorithms, and several replica ansätze for two values of K in order to highlight the occurance of the systematic errors.

		RSB		RS		IRSB	
$\mathcal{A}$	K[a]	n_l	n_e	n_l	n_e	n_l	n_e
$\mathcal{A}_\mathrm{II}^\mathrm{up}$	300	0.583(1)	0.49(2)	1.305(0)	1.271(5)	−0.014(4)	−0.112(6)
	750	0.495(1)	0.36(2)	1.258(0)	1.225(0)	−0.070(0)	−0.084(1)
$\mathcal{A}_\mathrm{III}^\mathrm{up}$	300	0.622(2)	0.53(2)	1.312(0)	1.276(6)	0.035(2)	−0.035(17)
	450	0.571(1)	0.47(1)	1.259(0)	1.223(1)	−0.004(1)	−0.064(2)
$\mathcal{A}^\mathrm{t}$	300	0.367(1)	0.29(1)	1.079(0)	1.070(0)	−0.128(0)	−0.152(1)
	450	0.330(1)	0.23(2)	1.062(0)	1.049(0)	−0.136(0)	−0.147(1)

[a] The numbers given for K apply to one-step RSB in the case of the spherical perceptron. For RS $K = 1000$ and $K = 4000$ were used as before and for the Ising perceptron the larger network sizes were $K = 1000$ for $\mathcal{A}_\mathrm{II}^\mathrm{up}$ and $K = 500$ for $\mathcal{A}_\mathrm{III}^\mathrm{up}$ and $\mathcal{A}^\mathrm{t}$.

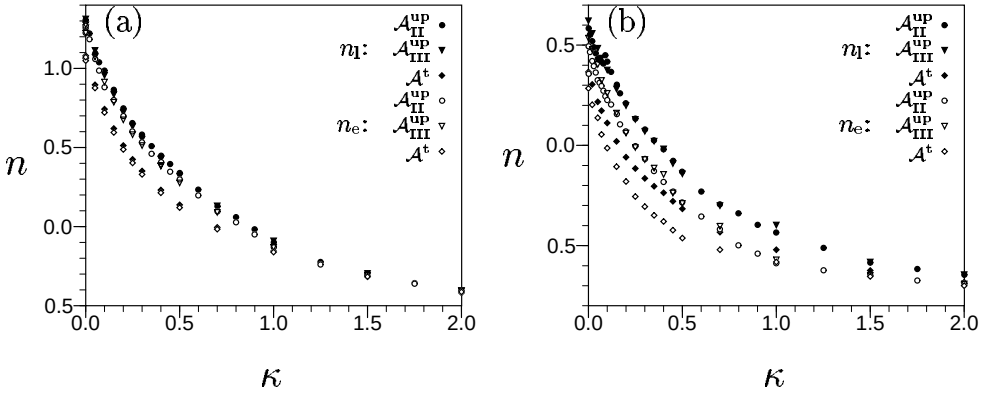

Figure 13. The power-law exponent estimates $\tilde{n}_\mathrm{l}$ and $\tilde{n}_\mathrm{e}$ for the capacity limit $\alpha_\mathrm{c}^K \propto [\log K]^n$ is shown as a function of the stability κ for (a) RS and (b) one-step RSB for the various algorithms (upstart II ($\mathcal{A}_\mathrm{II}^\mathrm{up}$), upstart III ($\mathcal{A}_\mathrm{III}^\mathrm{up}$), and tiling-like ($\mathcal{A}^\mathrm{t}$)). The local values $\tilde{n}_\mathrm{l}$ were determined for $K = 1000$ for RS and for $150 \leqslant K \leqslant 300$ for one-step RSB and are denoted by filled symbols, whereas the extrapolation estimate $\tilde{n}_\mathrm{e}$ are represented by open symbols (see the legend). The estimation error for $\tilde{n}_\mathrm{l}$ and $\tilde{n}_\mathrm{e}$ does roughly not exceed more than three times the size of the symbols, and is about their size in many cases especially for RS or one-step RSB and very large κ.

correction from neglecting the slowly decaying systematic shifts of the last perceptrons asymptotic ($\widetilde{m}_K \to 0$) capacity for finite κ. For large stabilities, where the slope of the raw capacity curves becomes very small or even changes sign, any reliable exponent cannot be determined from the capacity curve itself, which can be seen by the diverging n_e and n_l estimates for $\kappa = 1$. Below, we will therefore use the indirect method of determining estimates for n. Unfortunately, $\tilde{n}_\mathrm{l}$ is not a local estimate of n and therefore cannot be compared with the raw capacity curve in section 4.

Keeping these restrictions in mind, the behaviour of the power-law exponent estimates $\tilde{n}_\mathrm{l}$ and $\tilde{n}_\mathrm{e}$ as a function of the stability κ is shown in figure 13 for all considered constructive algorithms within the RS (figure 13(*a*)) and one-step RSB (figure 13(*b*)) ansatz. In all cases the behaviour of $\tilde{n}$ mirrors the qualitative observations made from figures 5 and 6;

the exponent estimates of n decrease monotonically for increasing stability κ and above a critical stability κ_c, the slope becomes negative. For large stabilities, the power-law exponent estimates seem to converge to a finite limit $\tilde{n}(\kappa \to \infty)$. It is a very interesting question, whether the shape of the functional dependence of n on κ is preserved for $K \to \infty$.

Within the RS ansatz, depicted in figure 13(a), the estimated exponent $\tilde{n} > 1$ for small stabilities and all algorithms suggesting that the MD bound is violated. This is in contrast to the original work for the tiling-like algorithm [62], where $n = 1$ was reported within the RS ansatz based on smaller network sizes. However, it is not entirely clear whether $n > 1$ actually holds asymptotically. It is worth mentioning that the simple RS treatment in the Gardner-volume calculation for the committee machine [14, 15] predicts an asymptotic capacity limit proportional to $\sqrt{K}$ instead of the correct $\sqrt{\log K}$ [19–21], i.e. predicts K rather than $\log K$ as the relevant quantity. The RS ansatz in the present case seems to lead at least to the correct scaling with $[\log K]^n$.

For the one-step RSB ansatz shown in figure 13(b), the picture is similar, with the noticeable difference of a shift of n such that $n < 1$ for $\kappa = 0$. Note that for one-step RSB the errors for the individual exponent estimates are much larger since $150 \leqslant K \leqslant 300$ instead of $K > 1000$ for RS. Especially large error bars are obtained for $\kappa \approx 0.1$ and upstart II, where the kink in the curve can be explained by the phase transition at $100 \leqslant K \leqslant 300$. Note that for upstart III the estimates of n (see also both tables 2 and 3) are close to $\frac{1}{2}$, which would suggest that networks constructed by the upstart III algorithm have a similar performance as the committee machine. Since the committee machine uses unconstrained optimization of the internal representations, this may be seen as a further indication that the available n estimates are significantly too large.

It would further be interesting to compare the performance of the three algorithms. Although, we find that the $\tilde{n}$ estimates somewhat mirror the observations made from the visual inspection of the capacity curves in section 4.1, a significant difference is that the tiling-like algorithm is estimated to perform asymptotically worse than both upstart versions. However, due to the larger finite size effects and the systematic errors for the exponents of the upstart algorithm, this may not hold asymptotically. Furthermore, this calculation does not account for the influence of correlations between the perceptrons on n. We expect these corrections to be larger for the upstart algorithm.

Note that the difference between $\tilde{n}_l$ and $\tilde{n}_e$ approximately decreases monotonically with κ, although K was at least decreased for one-step RSB from $K = 300$ for $\kappa = 0$ to $K = 150$ for $\kappa = 2$. We have identified two possible causes. First, the n itself may have a finite limit $n(\kappa \to \infty)$. Second, for large stability the asymptotic error regime of the individual perceptrons (where the error approaches the asymptotic error in a simple power law with logarithmic corrections) is reached faster, which may dampen higher order corrections to the measured power law in $\tilde{m}_K$.

In conclusion, we believe that it is plausible to assume that $\tilde{n}_l$ as well as $\tilde{n}_e$ constitute practical upper bounds for $n(\kappa)$. For smaller stabilities significant corrections are to be expected as has already been highlighted by the inconsistency found for non-zero bias. For increasing stabilities, these bounds become arguably tighter.

5.4. Analysis for the Ising perceptron

In this section, we briefly assess whether the observations made for the spherical perceptron also hold for the Ising perceptron. In figure 14 both estimates of the power-law exponent, $\tilde{n}_l$ and $\tilde{n}_e$ for upstart II, show the same characteristic decay to a finite limit for large stability. The main difference is that the curves are shifted to much smaller n values. In fact for all

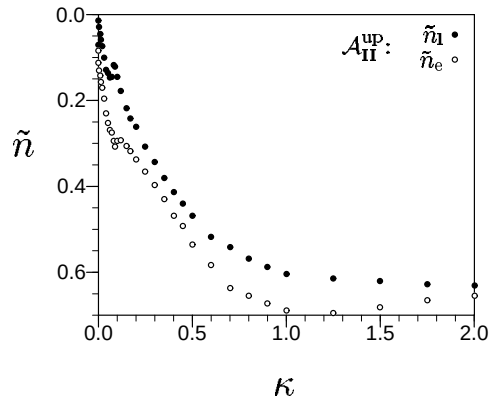

Figure 14. For upstart II and the uncoupled one-step RSB ansatz, employing the Ising perceptron as the basic building block results in $\tilde{n}_{\mathrm{l}}$ and $\tilde{n}_{\mathrm{e}}$ being shifted to smaller values for all stabilities κ. The local values $\tilde{n}_{\mathrm{l}}$ were determined for $150 \leqslant K \leqslant 300$ and are denoted by filled circles, whereas their extrapolation $\tilde{n}_{\mathrm{e}}$ are represented by open circles (see the legend).

algorithms (see also table 3) both $\tilde{n}_{\mathrm{l}}$ and $\tilde{n}_{\mathrm{e}}$ are predicted to be slightly smaller than 0 for large enough K, i.e. asymptotically the capacity is expected to decrease for all stabilities.

For the Ising perceptron, the error in the $\tilde{n}(\kappa)$ estimates is especially large for $\kappa \approx 0.08$, where the kink in the capacity for relatively large K makes the measurements of $\tilde{n}$ more difficult. For large κ, the non-monotonic behaviour may be an artefact caused by the decrease of K with κ.

6. Summary and conclusions

The main thrust of this paper has been the study of the capacity limit of MLPs built by constructive algorithms. Here we have concentrated on two algorithms, a tiling-like algorithm ($\mathcal{A}^{\mathrm{t}}$) for a parity machine [62], inspired by the tiling algorithm [25], and variants of the upstart algorithm [28] ($\mathcal{A}_{\mathrm{IIa}}^{\mathrm{up}}$, $\mathcal{A}_{\mathrm{IIb}}^{\mathrm{up}}$, $\mathcal{A}_{\mathrm{IIIa}}^{\mathrm{up}}$, and $\mathcal{A}_{\mathrm{IIIb}}^{\mathrm{up}}$) which are accessible to a statistical mechanics framework. The variants of the upstart algorithm differ in the make-up of the training set (II,III) and in the selection criteria (a,b) used for the creation of new units, allowing us to assess the impact of small changes in the algorithm to the resulting capacity.

In order to calculate their capacity explicitly, the approximation has been made that the quenched average over the training sets can be taken separately for each perceptron generated by the algorithms, effectively assuming that the perceptrons are uncorrelated. This approximation allows the capacity being calculated employing results for simple perceptrons derived within a replica framework. The validity of this 'uncoupled' approximation has been assessed within a RS ansatz for the case of two perceptrons being generated in successive steps of the algorithms. The corrections to the capacity (and the errors made above saturation) due to the correlations turn out to be in most cases negligible in comparison with the effect of RSB in the individual perceptrons, suggesting that the results derived are a good approximation.

For the case of zero stability and unbiased output distributions for which exact asymptotic capacity results are known for the parity ($\alpha_{\mathrm{c}}^{K} \propto \log K$) [13] and the committee machine ($\alpha_{\mathrm{c}}^{K} \propto \sqrt{\log K}$) [19–21], we find that all constructive algorithm considered are

likely to exhibit a power-law behaviour† in $\log K$, $\alpha_c^K \simeq c[\log K]^n$. The power-law ansatz modifies preliminary results [64] and earlier work [62] which proposed a simpler linear law in $\log K$.

Visual inspection of the capacity curves suggest that the prefactor c is dependent on the constructive algorithm employed. For a more objective analysis, local estimates of the exponent $n(K)$ for finite K have been obtained, providing a reasonable fit for the capacity curves and then extrapolated to $1/K \to 0$. However, two sources for systematic errors were identified. First and probably most important, due to the small range of $\log K$ values accessible to such an approach, significant finite K effects were encountered. A re-estimation of n using larger networks shows that the n estimates systematically shift to smaller values. Furthermore, the K values explored are too small to make the extrapolation accurate—the local estimate of n at a larger K value were often found to be significantly smaller than the extrapolation from smaller K values to $1/K \to 0$. Second, further RSB and the ignored correlation between perceptrons should lead to systematic shifts of n to smaller values—although the magnitude of these corrections is unknown, they should be significantly smaller than the corrections going from the uncoupled RS to one-step RSB. In summary, the exponent values cannot be estimated reliably enough to decide whether n is algorithm dependent. Nevertheless, the extracted exponents may still provide practical upper bounds for the true exponents since the extrapolated values were in all cases smaller than their corresponding local estimates.

Within these restrictions, the exponent n for the RS ansatz has been estimated to be greater than 1 for all constructive algorithms within the K values studied, which would violate the MD bound [7]. If this violation holds asymptotically, it should be compared with the failure of the RS ansatz in fixed architecture cases [14, 15], where power laws in K instead of $\log K$ are predicted.

For the uncoupled one-step RSB ansatz, we find $0.23 \lessapprox n \lessapprox 0.47$ for various estimation methods and constructive algorithms. Especially the result for the upstart III algorithm predicting values close to $\frac{1}{2}$ (the exponent for the committee machine using unconstrained optimization) seems to confirm that these finite K predictions are too optimistic.

This work has furthermore extended the study of the capacity limit to finite stabilities and biased output distributions, issues which, to our knowledge, have not been addressed previously for multilayer networks. In both cases, the most reasonable functional form of the capacity limit remains $\alpha_c^K \simeq c[\log K]^n$ for large K and the constructive algorithms studied.

In the case of biased output distributions (but zero stability), the limitations of the validity of the extracted exponents have been made more apparent as no consistent n values could be determined. Theoretically n must either increase or remain constant for $m_o \to 1$; our numerical results, however, suggest otherwise. This contradiction may be explained by the fact that networks are pushed further into the asymptotic error and capacity regimes for increasing bias, increasing the 'effective' K value. The estimated exponents for large m_o may therefore provide a tighter practical upper bound for the true exponents than could be achieved for zero output bias. It remains an interesting open question whether the true exponent n is a function of m_o or a constant. Constant n implies that the exponent must also be independent of the constructive algorithms studied since it can be shown that they become equivalent in the $m_o \to 1$ limit. A visual inspection for finite K values relevant in practice reveals some performance difference between the algorithms for small but finite

† Although it may be argued that the size of K explored does not allow us to rule out any other functional ansätze, a power-law was the only simple functional form which held reasonably across all cases studied.

bias. The tiling-like algorithm outperforms both upstart variants, which is partly due to the fact that only suboptimal unit creation selection criteria could be identified for the upstart algorithm.

For finite stability κ but unbiased output distributions, we find that the $n(\kappa)$ estimates decay monotonically in κ for all algorithms to finite limits $n(\kappa \to \infty)$. For both RS and one-step RSB, we find that for stabilities beyond a 'critical' stability κ_c (defined by a K independent constant capacity for large K) the capacity (per network weight) decreases asymptotically (as a function of K). This effect has also been observed from a visual inspection of the capacity curves, however, the analysis suggests that the critical stability may asymptotically be much smaller than anticipated from the curves themselves. For the Ising perceptron (and one-step RSB), the n estimates are smaller than 0 for all stabilities, whereas numerically we find this transition for small but finite stability for the K values explored.

In all cases, it is of considerable interest whether the true exponent n is dependent on the stability κ. Within the limitations of our analysis this question cannot be answered with certainty. However, it may be argued that, although the n estimates themselves are not accurate, the generic shape of the dependence between n estimates and κ has consistently carried over across different perceptron weight models and replica ansätze, making a stability dependent exponent a reasonable conjecture.

Of further interest is whether the exponent n is dependent on the constructive algorithm employed analogous to the functional dependence of n on the architecture type found for conventional networks. Comparing the various constructive algorithms for all κ, the estimates predict consistently $n(\mathcal{A}_{\mathrm{III}}^{\mathrm{up}}) > n(\mathcal{A}_{\mathrm{II}}^{\mathrm{up}}) > n(\mathcal{A}^{\mathrm{t}})$, suggesting that the upstart algorithm is asymptotically more efficient. However, this result completely neglects the issue of the size of systematic errors in the n estimates, which are considerably larger for the upstart algorithm. Especially for small stability, the difference between the n estimates for the tiling-like and upstart algorithms seem grossly exaggerated considering the numerical results. The above discussion of this issue for large bias gives also some credit to the conjecture that the performance difference between (the considered) constructive algorithms may lead asymptotically only to algorithm (and stability) dependent prefactors.

For the prefactors, the numerical capacity results predict, that upstart III always outperforms upstart II. This performance difference is caused by the fact that the design of upstart III allows for the elimination of more training patterns from the training set of units constructed consecutively (although this difference vanishes for $\kappa \to \infty$). The comparison between upstart and tiling-like algorithm is less straightforward. For small stability the tiling-like capacity remains above both upstart capacities for all K values and calculation ansätze (RS, RSB, IRSB) studied, although the upstart III capacity closes the gap for increasing K. For larger stability, both upstart algorithms exhibit a higher capacity than the tiling-like algorithm for large enough K. This behaviour reflects two competing effects. The design of the upstart algorithm includes pattern elimination for latter units (increasing its efficiency) but also features the creation of specialized units correcting only one error type (decreasing its efficiency).

In conclusion, we believe it is reasonable to assume that all considered constructive algorithm use their hidden units less efficiently than a fixed architecture multilayer network with unconstrained optimization. This is due to the fact that the constructive algorithms studied use their hidden units to overrule previous decision and can therefore explore only a much smaller space of internal representation than a general two-layer network.

One can identify several future research directions as a natural extension to the current work. First, the framework presented here may be extended straightforwardly to include

other commonly used cost functions, e.g. of the form $V = (\kappa - \lambda)^n \Theta(\kappa - \lambda)$ with $n = 1, 2$, which may result in a different behaviour. Second, it would be highly desirable to investigate the effect of finite stability and non-zero output distribution bias for fixed architectures where the $K \to \infty$ limit can be taken analytically. Of special interest would be whether the exponent of the power-law is actually dependent on the stability κ and/or the bias m_o of the output distribution—two issues which have been addressed but could not be answered with sufficient certainty in the framework employed in this paper.

Acknowledgments

AHLW is grateful for financial support by the EPSRC, a research scholarship of the Department of Physics at the University of Edinburgh, and the financial support and hospitality of the Neural Computing Research Group at Aston University, where part of this research was carried out. DS acknowledges support by EPSRC grant no GR/L19232.

Appendix A. Replica calculation for two coupled perceptrons

In this appendix, we will briefly outline the replica calculation for two Boolean perceptrons which have been successively constructed by variants of the upstart algorithm or the tiling-like algorithms introduced in section 2 for the case of learning a set of random dichotomies. The calculation is in spirit similar to the single perceptron [2, 22]. We have restricted ourselves to the case of real valued weights and a spherical constraint and the RS ansatz for simplicity, although the calculation is in principle also extendible to one-step RSB or/and to binary $\{-1, 1\}$ weights (Ising constraint).

A.1. Free energies of the coupled perceptrons

As mentioned in section 2.1, the task of the learner in the capacity problem is to implement a given set of $p = \alpha N$ random dichotomies $(\boldsymbol{\xi}^\mu, \zeta^\mu)$, where the inputs $\boldsymbol{\xi}^\mu$ and outputs ζ^μ are taken from the distributions (1). In general, it is useful to extend this capacity problem beyond saturation, where the learner cannot implement all examples but has to misclassify some of them. The aim of training is then to minimize the training error, which is given by summing over a suitable *cost function* for each example. Below, we will investigate the minimal error and the capacity achievable for a learner consisting of two perceptrons created consecutively by the constructive algorithms in question and for cost functions which penalize the number of misclassifications.

The first perceptron with parameters $\boldsymbol{\Omega}_1 = \{\boldsymbol{W}_1, \theta_1\}$, performing the mapping in (2), aims to minimize the number of misclassifications irrespective of the constructive algorithm and is therefore trained on the error function equivalent to the one introduced in (3)

$$E_1 = \sum_\mu \boldsymbol{\Theta}(\kappa - \lambda_1^\mu) = \sum_\mu V_1(\lambda_1^\mu, \kappa) \tag{A.1}$$

where $\lambda_1^\mu = \zeta^\mu h_1^\mu$ and V_k is the error measure for a single example and has been introduced for convenience. It enables one to carry out most of the calculation without specifying a particular error function. It also allows the introduction of the auxiliary term $(\epsilon^- \boldsymbol{\Theta}(\zeta^\mu) + \epsilon^+ \boldsymbol{\Theta}(-\zeta^\mu))$ picking out the wrongly-on and wrongly-off errors when taking derivatives with respect to ϵ^+ or ϵ^-.

The inputs $\boldsymbol{\xi}^\nu$ for the training set of the second perceptron, with parameters $\boldsymbol{\Omega}_2 = \{\boldsymbol{W}_2, \theta_2\}$, are a subset of the original inputs while the targets ζ^ν are defined according to

the rules of the specific constructive algorithm and depend on the original target and the output of the first perceptron.

For the tiling-like algorithm, the training set consists of all previous inputs and the target is $+1$ for correctly and -1 for erroneously classified patterns. The error function to be minimized is therefore

$$E_2^{\mathrm{t}} = \sum_\mu \Theta(\kappa - \lambda_1^\mu)\Theta(\kappa + \lambda_2^\mu) + [1 - \Theta(\kappa - \lambda_1^\mu)]\Theta(\kappa - \lambda_2^\mu) \tag{A.2a}$$

$$= \sum_\mu V_2^{\mathrm{t}}(\lambda_1^\mu, \lambda_2^\mu, \kappa) \tag{A.2b}$$

where $\lambda_2^\mu = h_2^\mu$ and V_2^{t} is the generic error measure.

The error function for the upstart algorithm depends on the variant used and whether the second unit is constructed to correct wrongly-off or wrongly-on errors. However, it is self-evident by symmetry arguments that one has only to investigate the case of wrongly-off error correction: the result for wrongly-on error correction can be obtained by flipping the random output target $\zeta^\mu \to -\zeta^\mu$, which corresponds to changing the sign of the output distribution bias $m_{\mathrm{o}} \to -m_{\mathrm{o}}$. Following similar considerations as above concerning the training set and targets of the daughter unit (see table 1), one finds

$$E_2^{\mathrm{up},\gamma} = \sum_\mu \Theta(\zeta^\mu)\Theta(\kappa - \lambda_1^\mu)\Theta(\kappa - \lambda_2^\mu)$$

$$+\Theta(-\zeta^\mu)[1 + (\gamma - 1)\Theta(\kappa - \lambda_1^\mu)]\Theta(\kappa + \lambda_2^\mu) \tag{A.3a}$$

$$= \sum_\mu V_2^{\mathrm{up},\gamma}(\lambda_1^\mu, \lambda_2^\mu, \kappa) \tag{A.3b}$$

where γ switches between upstart II ($\gamma = 1$) and upstart III ($\gamma = 0$).

The total training error function therefore becomes for either algorithm using the generic E_2 for the second perceptron

$$E = E_1 + \delta E_2 \tag{A.4}$$

where we have introduced a weighting factor $\delta = \beta_2/\beta$ in the total energy where β_2 acts as a 'quasi'-temperature for the second perceptron in the total energy function. This is necessary because the minimization of the error is not unconstrained, i.e. the weights of the first perceptron are trained first and subsequently frozen during the training of the second perceptron.

The calculation will be performed in the thermodynamic limit $N \to \infty$ with finite example load $\alpha = p/N$, where the free energy per input $N\beta f = \log Z$ is assumed to be self-averaging. In the following, we will be interested only in the minimum error possible and will therefore consider zero-temperature Gibbs learning for both perceptrons. The separation of the training is achieved by first taking the $\delta \to 0$ limit, keeping terms up to $\mathrm{O}(\delta)$, and subsequently taking the $\beta \to \infty$ limit†. The free energy splits into two parts as a consequence: f_1, which is of O(1), and represents the free energy of the first perceptron and f_2, which is of $\mathrm{O}(\delta) = \mathrm{O}(\beta_2/\beta)$, and is the free energy of the second perceptron. Hence

$$\langle\langle f \rangle\rangle = \langle\langle f_1 + \delta f_2 \rangle\rangle = -\lim_{\beta\to\infty}\lim_{\delta\to 0}\lim_{N\to\infty}\frac{1}{N\beta}\langle\langle \log Z \rangle\rangle$$

$$= -\lim_{\beta\to\infty}\lim_{\delta\to 0}\lim_{N\to\infty}\frac{1}{N\beta}\left\langle\!\!\left\langle \log \int \mathrm{d}\mu\,(\boldsymbol{W}_2)\mathrm{d}\mu\,(\boldsymbol{W}_1)\exp[-\beta(E_1 + \delta E_2)] \right\rangle\!\!\right\rangle \tag{A.5}$$

† Alternatively, one can see this procedure as taking the $\beta \to \infty$ with β_2 constant and then taking the $\beta_2 \to \infty$ limit.

where $\langle\langle\cdot\rangle\rangle$ is the quenched average over the distribution of patterns, consisting of integrations over the biased input and output distributions (2.1).

To avoid the invariance $(\boldsymbol{W}_i, \kappa) \to (\lambda \boldsymbol{W}_i, \lambda\kappa)$, one enforces spherical constraints on the weight vectors

$$\mathrm{d}\mu\,(\boldsymbol{W}_i) = \delta(\boldsymbol{W}_i \cdot \boldsymbol{W}_i - N) \prod_{k=1}^{N} \mathrm{d}W_k. \tag{A.6}$$

To be able to perform the quenched average we make use of the replica trick $\langle\langle \log Z\rangle\rangle = \lim_{n\to 0}(\langle\langle Z^n\rangle\rangle - 1)/n$. Note, that we treat the two perceptrons as one physical system with parameters $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2\}$ when replicating the partition function. We apply the same standard techniques including the introduction of order parameters for the single perceptrons and order parameters describing the cross-overlaps between the two perceptrons

$$Q_i^{\sigma\rho} = \frac{1}{N}\boldsymbol{W}_i^{\sigma} \cdot \boldsymbol{W}_i^{\rho} \qquad \text{for } \forall i \text{ and } \forall \sigma < \rho \tag{A.7a}$$

$$M_i^{\sigma} = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} W_{i_k}^{\sigma} \qquad \text{for } \forall i \text{ and } \forall\, \sigma \tag{A.7b}$$

$$R^{\sigma\rho} = \frac{1}{N}\boldsymbol{W}_1^{\sigma} \cdot \boldsymbol{W}_2^{\rho} \qquad \text{for } \forall\, \sigma, \rho \tag{A.7c}$$

with their Lagrange multipliers $\hat{Q}_i^{\sigma\rho}$, $\hat{M}_i^{\sigma}$, $\hat{R}^{\sigma\rho}$ and the Lagrange multiplier $\hat{E}_i^{\sigma}$ associated with the spherical constraints†. After some algebra, the replicated partition function becomes

$$\begin{aligned}\langle\langle Z^n\rangle\rangle = \int_{-\infty}^{\infty}\int_{-\mathrm{i}\infty}^{\mathrm{i}\infty} &\left(\prod_{i=1}^{2}\prod_{\sigma}\frac{\mathrm{d}M_i^{\sigma}\mathrm{d}\hat{E}_i^{\sigma}}{2\pi}\right)\left(\prod_{i=1}^{2}\prod_{\sigma<\rho}\frac{\mathrm{d}Q_i^{\sigma\rho}\mathrm{d}\hat{Q}_i^{\sigma\rho}}{2\pi}\right)\left(\prod_{\sigma,\rho}\frac{\mathrm{d}R^{\sigma\rho}\mathrm{d}\hat{R}^{\sigma\rho}}{2\pi}\right)\\ &\times \exp\left\{N\left[G_0(\hat{Q}_i^{\sigma\rho}, \hat{E}_i^{\sigma}, \hat{R}^{\sigma\rho}) + \alpha G_{\mathrm{r}}(Q_i^{\sigma\rho}, \theta_{i\,i}^{\sigma}, M_i^{\sigma}, R^{\sigma\rho})\right.\right.\\ &\left.\left. +\tfrac{1}{2}\sum_{\sigma,i}\hat{E}_i^{\sigma} - \sum_{\sigma<\rho}\sum_{i} Q_i^{\sigma\rho}\hat{Q}_i^{\sigma\rho} - \sum_{\sigma,\rho} R^{\sigma\rho}\hat{R}^{\sigma\rho}\right]\right\}.\end{aligned} \tag{A.8}$$

The two terms in the integral are the prior constraint Hamiltonian,

$$\begin{aligned}G_0(\hat{Q}_i^{\sigma\rho}, \hat{E}_i^{\sigma}, \hat{R}^{\sigma\rho}) = \log\left\{\int_{-\infty}^{\infty}\prod_{\sigma}\mathrm{d}W^{\sigma}{}_1\,\mathrm{d}W^{\sigma}{}_2 \exp\left[-\tfrac{1}{2}\sum_{\sigma,i}\hat{E}_i^{\sigma}W^{\sigma}{}_i W^{\sigma}{}_i\right.\right.\\ \left.\left. + \sum_{\sigma<\rho,i}\hat{Q}_i^{\sigma\rho}W^{\sigma}{}_i W^{\rho}{}_i + \sum_{\sigma,\rho}\hat{R}^{\sigma\rho}W^{\sigma}{}_1 W^{\rho}{}_2\right]\right\}\end{aligned} \tag{A.9a}$$

which depends on the weight prior and the replicated Hamiltonian

$$\begin{aligned}G_{\mathrm{r}}(Q_i^{\sigma\rho}, \theta_{i\,i}^{\sigma}, M_i^{\sigma}, R^{\sigma\rho}) = \log\left\langle\int_{-\infty}^{\infty}\left(\prod_{i=1}^{2}\prod_{\sigma}\frac{\mathrm{d}\lambda_i^{\sigma}\mathrm{d}\hat{\lambda}_i^{\sigma}}{2\pi}\right)\right.\\ \times\exp\left\{-\beta[V_1(\lambda_1^{\sigma}, \kappa) + \delta V_2(\lambda_1^{\sigma}, \lambda_2^{\sigma}, \kappa)] - \mathrm{i}\sum_{\sigma,i}\hat{\lambda}_i^{\sigma}\lambda_i^{\sigma}\right.\\ \left.\left.-\mathrm{i}\sum_{\sigma}\left[\zeta\hat{\lambda}_1^{\sigma}(\theta_1^{\sigma} - m_{\mathrm{i}}M_1^{\sigma}) + \hat{\lambda}_2^{\sigma}(\theta_2^{\sigma} - m_{\mathrm{i}}M_2^{\sigma})\right]\right.\end{aligned}$$

† The contribution of $\hat{M}_i^{\sigma}$ vanish in the thermodynamic limit similar to single perceptron calculations.

$$-\tfrac{1}{2}(1-m_{\mathrm{i}}^2)\Bigg[\sum_{\sigma,i}\hat{\lambda}_i^\sigma\hat{\lambda}_i^\sigma+2\sum_{\sigma<\rho,i}\hat{\lambda}_i^\sigma\hat{\lambda}_i^\rho Q_i^{\sigma\rho}+2\zeta\sum_{\sigma,\rho}\hat{\lambda}_1^\sigma\hat{\lambda}_2^\rho R^{\sigma\rho}\Bigg]\Bigg\}\Bigg\rangle_\zeta \qquad \text{(A.9}b\text{)}$$

where $\langle\cdot\rangle_\zeta$ denotes an average over the output distribution.

A.2. The RS ansatz

To make further progress one has to make an assumption for the structure of the replica space. Here, we will only pursue the simplest RS ansatz, which assumes

$$M_i^\sigma = M_i \qquad \begin{array}{llll} Q_i^{\sigma\rho}=q_i & \text{and} & \hat{Q}_i^{\sigma\rho}=\hat{q}_i & \text{for } \forall i \text{ and } \sigma<\rho \\ \theta_i{}_i^\sigma=\theta_i & \text{and} & \hat{E}_i^\sigma=\hat{E}_i & \text{for } \forall i \text{ and } \forall\sigma \\ R^{\sigma\sigma}=r & \text{and} & \hat{R}^{\sigma\sigma}=\hat{r} & \text{for } \forall\sigma \\ R^{\sigma\rho}=s & \text{and} & \hat{R}^{\sigma\rho}=\hat{s} & \text{for } \forall\sigma\neq\rho. \end{array} \qquad \text{(A.10)}$$

The physical interpretation of q_i is the same as for a single perceptron calculation: the typical overlap of two solutions within the version space of the individual perceptrons. The overlap s and r both describe the overlap between the two perceptrons, but r describes the overlap of the second perceptron with the first perceptron on whose errors it has been trained, whereas s describes the overlap of the second perceptron with any other first perceptron from the version space.

We note that RS is broken in this scenario. However, the aim of this calculation is to assess whether the effect of coupling two perceptrons in a capacity calculation is stronger than that of RSB in the individual perceptrons. A one-step RSB calculation would result in four-dimensional integrals, which are difficult to evaluate numerically accurate enough to find solutions to the saddle-point equations.

Inserting the above ansätze into (A.9a) and (A.9b) and taking the $n\to 0$ limit yields

$$G_0^{\mathrm{RS}}=\frac{1}{2}\frac{(\hat{E}_1+\hat{q}^1)\hat{q}^2+(\hat{E}_2+\hat{q}^2)\hat{q}^1+2\hat{s}(\hat{r}-\hat{s})}{(\hat{E}_1+\hat{q}^1)(\hat{E}_2+\hat{q}^2)-(\hat{r}-\hat{s})^2} -\frac{1}{2}\log[(\hat{E}_1+\hat{q}^1)(\hat{E}_2+\hat{q}^2)-(\hat{r}-\hat{s})^2] \qquad \text{(A.11}a\text{)}$$

$$G_{\mathrm{r}}^{\mathrm{RS}}=\left\langle\int D\mu(t_1,t_2)\ \log[\mathcal{F}_{\mathrm{RS}}(\boldsymbol{t},\beta,\kappa,\zeta,q_i,\theta_i,r,s)]\right\rangle_\zeta \qquad \text{(A.11}b\text{)}$$

where all integrals without explicit limits are from $-\infty$ to $+\infty$. The measure $D\mu(t_1,t_2)$ is given by

$$D\mu(t_1,t_2)=\frac{\mathrm{d}t_1\mathrm{d}t_2}{2\pi\sqrt{q_1q_2-s^2}}\exp\left[-\frac{1}{2(q_1q_2-s^2)}\begin{pmatrix}t_1\\t_2\end{pmatrix}^T\begin{pmatrix}q_2 & -\zeta s\\ -\zeta s & q_1\end{pmatrix}\begin{pmatrix}t_1\\t_2\end{pmatrix}\right] \qquad \text{(A.12)}$$

and the function $\mathcal{F}_{\mathrm{RS}}$ is given by

$$\mathcal{F}_{\mathrm{RS}}(\boldsymbol{t},\beta,\kappa,\zeta,q_i,\theta_i,r,s)=\int\frac{\mathrm{d}\lambda_1\mathrm{d}\lambda_2}{2\pi(1-m_{\mathrm{i}}^2)\sqrt{(1-q_1)(1-q_2)-(r-s)^2}} \times\exp\Bigg\{-\beta\Bigg[V_1(\lambda_1,\kappa)+\delta V_2(\lambda_1,\lambda_2,\kappa) +\frac{1}{2}\frac{x_2(\psi_1+t_1)^2+\delta x_1(\psi_2^+ +t_2)^2-2\delta\zeta z(\psi_1+t_1)(\psi_2^+ +t_2)}{x_1x_2-\delta z^2}\Bigg]\Bigg\} \qquad \text{(A.13)}$$

where $x_1 = \beta(1-q_1)$, $x_2 = \beta_2(1-q_2)$, $z = \beta(r-s)$, and

$$\psi_1(\lambda_1) = \frac{\lambda_1 + \zeta(\theta_1 - m_\mathrm{i} M_1)}{\sqrt{1-m_\mathrm{i}^2}} \qquad \text{and} \qquad \psi_2^{\pm}(\lambda_2) = \frac{\lambda_2 \pm (\theta_2 - m_\mathrm{i} M_2)}{\sqrt{1-m_\mathrm{i}^2}}. \tag{A.14}$$

Here, we have introduced the self-consistent scaling ansätze for the order parameters of the two perceptrons when taking the $\delta \to 0$ and $\beta \to \infty$ limits with $\delta\beta = \beta_1 \to \infty$ in order to access the ground states with least errors only: the volume of the individual solution spaces of the two perceptrons above their capacity limits shrink proportional to the applied 'temperature', which is β for the first and β_2 for the second perceptron. Since the version space of the first perceptron induces the difference between r and s, $r-s$ should scale with $1/\beta$.

For $\beta \to \infty$, the integrals over λ_1 and λ_2 in (A.13) can be calculated by the saddle-point method; the exponentional is evaluated at $\lambda_1 = \lambda_1^{\mathrm{opt}}$ and $\lambda_2 = \lambda_2 = \lambda_2^{\mathrm{opt}}$, where λ_1^{opt} and $\lambda_2 = \lambda_2^{\mathrm{opt}}$ jointly minimize the square bracket for given t_1 and t_2. The $\delta \to 0$ limit effectively constraints this minimization as required†: the dominant term of O(1) in (A.13) is independent of λ_2 and can therefore only determine λ_1^{opt}, which optimizes the first perceptron. The inclusion of O(δ) terms determines $\lambda_2 = \lambda_2^{\mathrm{opt}}$, which corresponds with the optimization of the second under the constraint of the first perceptron. Whereas the calculation of $\lambda_1^{\mathrm{opt}}(t_1)$ is identical for both upstart and tiling-like, $\lambda_2 = \lambda_2^{\mathrm{opt}}(t_1, t_2)$ is determined algorithm dependent. We furthermore eliminate the conjugate order parameters $\hat{q}_i$, $\hat{E}_i$, $\hat{r}$, and $\hat{s}$, keeping only the terms up to O(δ).

The free energy f_1 of O(1), i.e. for the first perceptron, then simplifies to the already known result

$$\langle\!\langle f_1 \rangle\!\rangle = \alpha \left\langle \int_{-\tau_1}^{\sqrt{2x_1}-\tau_1} \mathrm{D}t \, \frac{(t+\tau_1)^2}{2x_1} + H\left(\sqrt{2x_1} - \tau_1\right) \right\rangle_\zeta - \frac{1}{2x_1} \tag{A.15}$$

where

$$\tau_1 = \psi_1(\kappa) = \frac{\kappa + \zeta(\theta_1 - m_\mathrm{i} M_1)}{\sqrt{1-m_\mathrm{i}^2}} \qquad \mathrm{D}t = \frac{\mathrm{d}t}{\sqrt{2\pi}} \mathrm{e}^{-t^2/2}$$

$$\text{and} \qquad H(u) = \int_u^\infty \mathrm{D}t \, . \tag{A.16}$$

The free energy f_1 is be evaluated at the saddle points with respect to the variables x_1 and θ_1.

The free energy of the second perceptron for the tiling-like algorithm f_2^{t} simplifies to

$$\begin{aligned}
\langle\!\langle f_2^{\mathrm{t}} \rangle\!\rangle = \alpha \Bigg\langle & \int_{\sqrt{2x_1}-\tau_1}^{\infty} \mathrm{d}t_1 \left[\int_{\tau_2^- - \sqrt{2x_2}}^{\tau_2^-} \mathrm{d}t_2 \mu(t_1, t_2) \frac{(t_2 - \tau_2^-)^2}{2x_2} + \int_{-\infty}^{\tau_2^- - \sqrt{2x_2}} \mathrm{d}t_2 \mu(t_1, t_2) \right] \\
& + \int_{-\tau_1}^{-\sqrt{2x_1}-\tau_1} \mathrm{d}t_1 \left[\int_{-\tilde{\tau}_2^+}^{\sqrt{2x_2}-\tilde{\tau}_2^+} \mathrm{d}t_2 \mu(t_1, t_2) \frac{(t_2 + \tilde{\tau}_2^+)^2}{2x_2} + \int_{\sqrt{2x_2}-\tilde{\tau}_2^+}^{\infty} \mathrm{d}t_2 \mu(t_1, t_2) \right] \\
& + \int_{-\infty}^{-\tau_1} \mathrm{d}t_1 \left[\int_{-\tau_2^+}^{\sqrt{2x_2}-\tau_2^+} \mathrm{d}t_2 \mu(t_1, t_2) \frac{(t_2 + \tau_2^+)^2}{2x_2} + \int_{\sqrt{2x_2}-\tau_2^+}^{\infty} \mathrm{d}t_2 \mu(t_1, t_2) \right] \Bigg\rangle_\zeta \\
& - \frac{1 - 2\hat{z}r}{2x_2}
\end{aligned} \tag{A.17}$$

† It is fairly straightforward but cumbersome to calculate the free energy for $\delta = 1$, i.e. unconstrained minimization, in order to assess the performance degradation due to constraining the optimization in the constructive algorithms. Such a study is, however, beyond the scope of this work.

where the measure $\mu(t_1, t_2)$ is derived from (A.12) by taking the appropriate limits

$$\mu(t_1, t_2) = \frac{1}{2\pi\sqrt{1-r^2}} \exp\left[-\frac{t_1^2 - 2\zeta r t_1 t_2 + t_2^2}{2(1-r^2)}\right] \tag{A.18}$$

and the following variables were introduced for convenience

$$\tau_2^{\pm} = \psi_2^{\pm}(\kappa) = \frac{\kappa \pm (\theta_2 - m_i M_2)}{\sqrt{1-m_i^2}} \qquad \tilde{\tau}_2^{\pm} = \tau_2^{\pm} \mp \zeta\hat{z}(t_1 + \tau_1)$$
$$\text{and} \qquad \hat{z} = \frac{z}{x_1}. \tag{A.19}$$

The free energy of the second perceptron for the versions of the upstart algorithm $f_2^{\mathrm{up},\gamma}$ can be simplified similarly

$$\begin{aligned}\langle\langle f_2^{\mathrm{up},\gamma}\rangle\rangle = \alpha\Bigg\langle \int_{\sqrt{2x_1}-\tau_1}^{\infty} \mathrm{d}t_1 \Bigg\{ &\Theta(\zeta)\Bigg[\int_{-\tau_2^+}^{\sqrt{2x_2}-\tau_2^+} \mathrm{d}t_2\mu(t_1,t_2)\frac{(t_2+\tau_2^+)^2}{2x_2} + \int_{\sqrt{2x_2}-\tau_2^+}^{\infty}\mathrm{d}t_2\mu(t_1,t_2)\Bigg] \\ &+\gamma\Theta(-\zeta)\Bigg[\int_{\tau_2^- - \sqrt{2x_2}}^{\tau_2^-}\mathrm{d}t_2\mu(t_1,t_2)\,\frac{(t_2-\tau_2^-)^2}{2x_2} + \int_{-\infty}^{\tau_2^- - \sqrt{2x_2}}\mathrm{d}t_2\mu(t_1,t_2)\Bigg]\Bigg\} \\ &+\Theta(-\zeta)\Bigg\{\int_{-\tau_1}^{-\sqrt{2x_1}-\tau_1}\mathrm{d}t_1\Bigg[\int_{\tilde{\tau}_2^- - \sqrt{2x_2}}^{\tilde{\tau}_2^+}\mathrm{d}t_2\mu(t_1,t_2)\frac{(t_2-\tilde{\tau}_2^+)^2}{2x_2} \\ &+\int_{-\infty}^{\tilde{\tau}_2^- - \sqrt{2x_2}}\mathrm{d}t_2\mu(t_1,t_2)\Bigg] \\ &+\int_{-\infty}^{-\tau_1}\mathrm{d}t_1\Bigg[\int_{\tau_2^- - \sqrt{2x_2}}^{\tau_2^-}\mathrm{d}t_2\mu(t_1,t_2)\frac{(t_2-\tau_2^-)^2}{2x_2} + \int_{-\infty}^{\tau_2^- - \sqrt{2x_2}}\mathrm{d}t_2\mu(t_1,t_2)\Bigg]\Bigg\}\Bigg\rangle_{\zeta} \\ &-\frac{1-2\hat{z}r}{2x_2}.\end{aligned} \tag{A.20}$$

Since the free energy f_2 for either algorithm is only of $\mathrm{O}(\delta)$, it is evaluated with respect to the variables x_2, θ_2, r, and $\hat{z}$, with x_1 and θ_1 fixed by (A.15). The capacity limit α_c (here not normalized with respect to the number of units) of the combination of the two perceptrons can be calculated from the saddle-point equations by taking the $x_2 \to \infty$ limit, i.e. the second perceptron does not make any errors. Note, that for the upstart calculation this is only a formal capacity limit, since the wrongly-on errors still need to be corrected.

A.3. Solutions of the saddle-point equations

The saddle-point solutions for the order parameters and the error rates as a function of the normalized example number α were evaluated for the different constructive algorithms and a range of stabilities κ and output biases m_o. For brevity, only the most relevant effects for the purpose of this paper will be reported graphically, especially the size of the correlations and their impact on the capacity limit (and the error rate) in comparison with the impact of one-step RSB in the individual perceptrons.

For the tiling-like algorithm the order parameters r and $\hat{z}$, describing the correlations between the perceptrons, are shown in figure A1 as a function of α for $\kappa = 1$ and various m_o values. For zero output-distribution bias, the correlations are initially identical to zero above the capacity limit, before their magnitude rises abruptly corresponding to $\alpha = \alpha_p$, i.e. at the phase transition of the first perceptron from the zero to the non-zero threshold solution. Consequently for zero stability and zero bias, the perceptrons remain uncorrelated for all α.

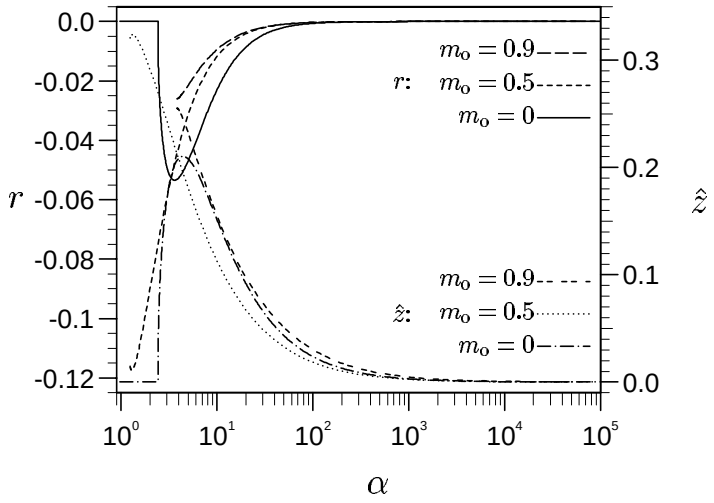

Figure A1. The correlations of two consecutive perceptrons created by the tiling-like algorithm are shown via the two order parameters r and $\hat{z}$ as a function of α for $\kappa = 1$ and three bias values (see the legend).

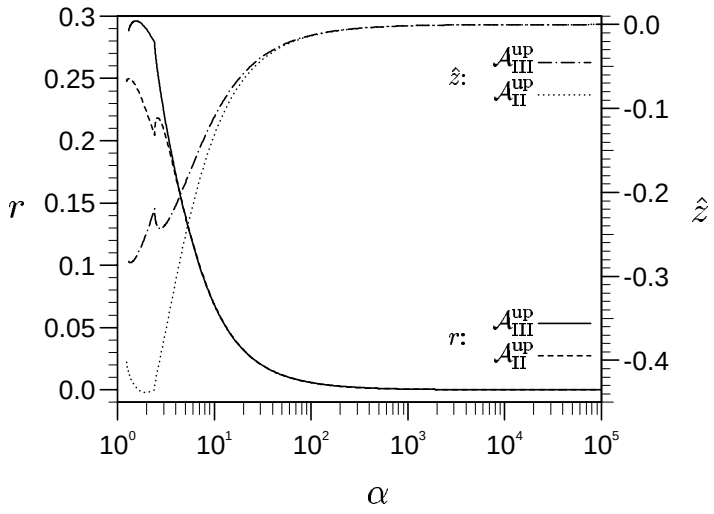

Figure A2. The correlations of two consecutive perceptrons created by two variants of the upstart algorithm (see the legend) are shown via the two order parameters r and $\hat{z}$ as a function of α for $\kappa = 1$ and $m_o = 0$.

After the magnitude of the correlations has passed through a maximum, the order parameters decay to zero asymptotically with power laws whose exponents are approximately

$$r \propto \alpha^{-1.9\pm1} \qquad \text{and} \qquad \hat{z} \propto \alpha^{-0.95\pm5} \tag{A.21}$$

where the error indicates the uncertainty in the last significant digits only. The uncertainty is most likely caused by logarithmic corrections to power laws with theoretical exponents of -2 and -1, respectively.

For non-zero bias, the correlations are largest at the capacity limit and the order parameters decay with the same power laws for $\alpha \to \infty$. Note, that a non-zero overlap r

between the perceptrons is always negative, i.e. the perceptrons are anti-correlated. The physical interpretation of the order parameter $\hat{z}$ is less clear. In figure A2 the order parameters r and $\hat{z}$ are shown for the upstart algorithm with either $\gamma = 1$ ($\mathcal{A}_{\mathrm{II}}^{\mathrm{up}}$) or $\gamma = 0$ ($\mathcal{A}_{\mathrm{III}}^{\mathrm{up}}$) and $\kappa = 1$, $m_{\mathrm{o}} = 0$. In comparison with the tiling-like algorithm, several differences and similarities are remarkable. First, the correlations for the upstart algorithm are always finite for zero bias. Second, one finds $r\hat{z} < 0$ as previously, however, the sign of the order parameters is reversed, i.e. $r > 0$ and the perceptrons are positively correlated. Third, the magnitude of the correlations for both variants of the upstart algorithm are always significantly larger than for the tiling-like algorithm. Note that the overlap r of the perceptrons for upstart III is larger than for upstart II, but the reverse is true for the magnitude of $\hat{z}$. The correlations for both variants and their differences are largest around the capacity limit. For $\alpha \to \infty$ the differences between the two variants vanish, since the difference in the training sets becomes negligible, and the correlations decay to zero with identical power-law exponents

$$r \propto \alpha^{-0.995\pm10} \qquad \text{and} \qquad \hat{z} \propto \alpha^{-0.96\pm5} \tag{A.22}$$

where the deviation from -1 may be caused by logarithmic corrections. Note furthermore that at the phase transition point α_{p}, the order parameters are non-differentiable as for the tiling-like algorithm with the possibility of local maxima for the magnitude of the correlation order parameters.

In general, one finds that any correlations decrease the capacity limit of the combined perceptrons or increase the error rate above saturation. The correlations are usually the largest around the capacity limit (besides for the tiling-like algorithm, where the maximum can be found around α_{p} for finite κ and zero or very small bias) and decay algebraically above it. In general, one also finds that the correlations grow with increasing stability and diminish for very large bias after passing through a maximum for non-zero bias.

In order to assess the impact of the correlations in comparison with RSB in the individual perceptrons above saturation more quantitatively, we concentrate on the capacity, the region where the correlations have generally the largest impact. The most meaningful comparison is found by calculating and comparing $\tilde{m}_{\mathrm{o}}$ as a function of κ for various fixed normalized capacities $\hat{\alpha}_{\mathrm{c}}$ (as in figure 2 for the single perceptron) for the tiling-like and the variants of the upstart algorithm and for correlated RS as well as uncorrelated RS and one-step RSB ansätze. Note, that $\tilde{m}_{\mathrm{o}}$ (defined as $\tilde{m}_{\mathrm{o}} \equiv (1 - |m_{\mathrm{o}}|)$) is ambiguous for the upstart algorithm as the symmetry of the capacity under $m_{\mathrm{o}} \leftrightarrow -m_{\mathrm{o}}$ is broken in the above calculation and a true capacity limit would also need three perceptrons. This ambiguity is resolved by postulating that the 'capacity limit' for bias m_{o} is given by the unit type that saturates first. This leads to the constraint $m_{\mathrm{o}} < 0$ for the upstart calculation w.l.o.g. Note furthermore that for decreasing $\tilde{m}_{\mathrm{o}}$ the portion of wrongly-on and correctly-off patterns, which cause the difference between the versions of the upstart algorithm and between the upstart and the tiling-like algorithm, become smaller, leading to similar results for large $\hat{\alpha}_{\mathrm{c}}$.

In figure A3, $\tilde{m}_{\mathrm{o}}$ is therefore only shown for two perceptrons coupled via the tiling-like algorithm in the uncorrelated one-step RSB ansatz (figure A3(*a*)), whereas $\overline{\Delta\tilde{m}_{\mathrm{o}}} = [\tilde{m}_{\mathrm{o}}(\mathcal{A}_{\mathrm{III}}^{\mathrm{up}}) - \tilde{m}_{\mathrm{o}}(\mathcal{A}^{\mathrm{t}})]/\tilde{m}_{\mathrm{o}}(\mathcal{A}^{\mathrm{t}})$ is shown for the upstart III algorithm to magnify the differences (figure A3(*b*)). Both quantities are multiplied by $\hat{\alpha}_{\mathrm{c}}$ in order to eliminate the most dominant scaling. In comparison with the single perceptron (figure 2), $\tilde{m}_{\mathrm{o}}$ is larger due to the addition of the second perceptron and grows slower with κ due to the error rate of the first perceptron increasing more quickly for larger stability. This already indicates that the tendency of $\tilde{m}_{\mathrm{o}}$ growing with κ reverses eventually for larger networks. The comparison of $\tilde{m}_{\mathrm{o}}$ in figure A3(*b*) for the two constructive algorithms shows that $\tilde{m}_{\mathrm{o}}$ for the upstart algorithm is

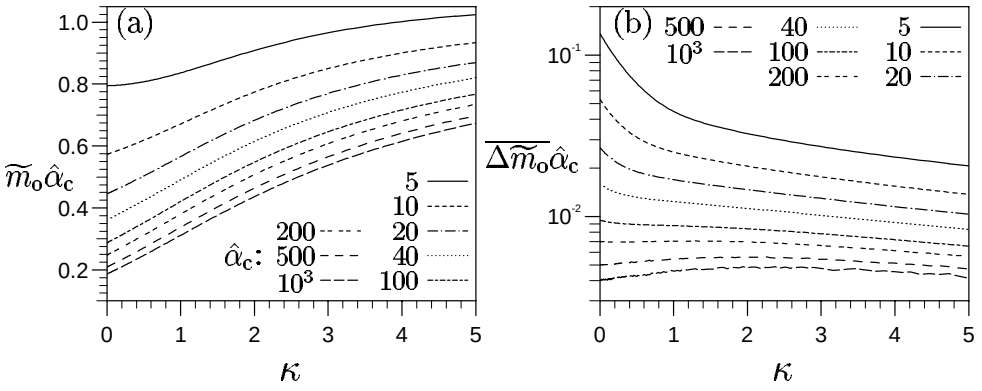

Figure A3. (*a*) The bias $\tilde{m}_o$ (or rather $\tilde{m}_o\hat{\alpha}_c$ to highlight the scaling of α_c with $\tilde{m}_o$) is shown as a function of the stability κ for several fixed normalized capacities $\hat{\alpha}_c(m_o) \equiv \alpha_c(m_o)/\alpha_c(0)$ (see the legends) for two perceptrons built by the tiling-like algorithm. (*b*) For the upstart III algorithm, the results are quite similar for large $\hat{\alpha}_c$ and to highlight the differences, $\overline{\Delta\tilde{m}_o} \equiv [\tilde{m}_o(\mathcal{A}_{\rm III}^{\rm up}) - \tilde{m}_o(\mathcal{A}^t)]/\tilde{m}_o(\mathcal{A}^t)$ is shown normalized by $\hat{\alpha}_c$.

always larger, although the difference in $\tilde{m}_o$ vanishes very quickly $\hat{\alpha}_c \to \infty$ (or $\tilde{m}_o \to 0$), especially considering the fact that the difference in $\tilde{m}_o$ has been magnified additionally by $1/\tilde{m}_o(\mathcal{A}^t)$.

In figure A4, the impact of one-step RSB and correlations (within an RS) ansatz (RS_c) is compared by plotting $\Delta\tilde{m}_{o\,RS_c}^{RSB} \equiv (\tilde{m}_o{}^{RSB} - \tilde{m}_o{}^{RS})/(\tilde{m}_o{}^{RS_c} - \tilde{m}_o{}^{RS})$ in the same scenario as figure A3 for the tiling-like (figure A4(*a*)) and the upstart III algorithm (figure A4(*b*)) (the differences to upstart II are insignificant). For both tiling-like and upstart algorithms, one finds that the corrections due to correlations are usually smaller than those due to RSB and become insignificant for large $\hat{\alpha}_c$ ($\Delta\tilde{m}_{o\,RS_c}^{RSB} \gg 1$), corresponding to large bias m_o (as found for the last perceptrons of a large network). Furthermore, one finds that the impact of the correlations increases in general with the stability κ. For small $\hat{\alpha}_c$ and large stability κ, one actually finds a region where correlations are more significant than RSB ($\Delta\tilde{m}_{o\,RS_c}^{RSB} < 1$), suggesting corrections of the capacity results for small networks and slightly biased output distributions. Note, that the correlation corrections of the capacity are usually at least twice as large for the upstart than for the tiling-like algorithm, which has already been suggested by the larger magnitude of the upstart than the tiling-like correlation order parameters shown in figures A2 and A1, respectively.

Although, the correlations can therefore be significant in some regions of m_o, κ, and α space, the area is confined to small m_o, large κ, and α around the capacity limit, which is not extremely relevant for large networks, where most units are highly saturated, and small stabilities, which are of most interest. An open question is the effect of correlation over several generations of the constructive algorithm. It seems, however, natural that the correlations between consecutive perceptrons should be dominant.

Appendix B. Replica calculation for a single perceptron

Below, we briefly summarize the results for the replica calculation for the Boolean perceptron learning random dichotomies above its saturation limit in order to make this work self-contained, as they are extensively used in the capacity calculation for the constructive algorithms. For a more detailed treatment the reader is referred to [2, 22].

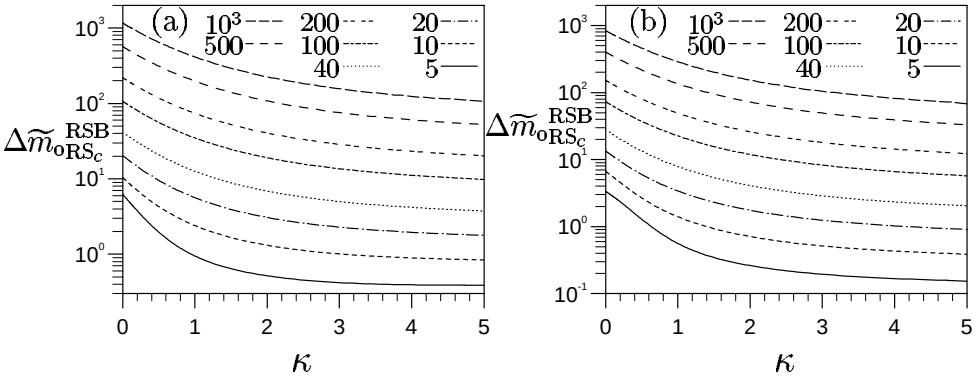

Figure A4. The size of corrections to $\tilde{m}_{\mathrm{o}}$ from the RS ansatz due to RSB and correlations within the RS ansatz (RS_c) is compared by plotting $\Delta\tilde{m}_{\mathrm{o}}{}^{\mathrm{RSB}}_{\mathrm{RS}_c} \equiv (\tilde{m}_{\mathrm{o}}{}^{\mathrm{RSB}} - \tilde{m}_{\mathrm{o}}{}^{\mathrm{RS}})/(\tilde{m}_{\mathrm{o}}{}^{\mathrm{RS}_c} - \tilde{m}_{\mathrm{o}}{}^{\mathrm{RS}})$ in the same scenario as figure A3 for (*a*) the tiling-like and (*b*) the upstart III algorithm with the fixed normalized capacities $\hat{\alpha}_{\mathrm{c}}(m_{\mathrm{o}})$ given in the legends.

The basic ideas for the replica calculation are similar to appendix A and the free energy of the RS ansatz can be directly taken from (A.15), by dropping the perceptron index. The capacity limit, α_{c}, can then be calculated by taking the limit $x \to \infty$ for the saddlepoint equation of (A.15). As mentioned previously, RS is broken beyond the capacity limit and the Parisi scheme of successive steps of RSB [3] must be employed, whose first approximation is the one-step RSB ansatz, for which $Q^{\sigma\rho}$ is a $n \times n$ matrix

$$(Q^{\sigma\rho})_{n\times n} = \begin{pmatrix} Q_1 & Q_0 & \cdots & Q_0 \\ Q_0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & Q_0 \\ Q_0 & \cdots & Q_0 & Q_1 \end{pmatrix}_{n\times n} \tag{B.1}$$

where Q_0 is a $m \times m$ matrix with elements q_0 and Q_1 is a $m \times m$ matrix with 0 on the diagonal and q_1 elsewhere. The ansatz for $\hat{Q}^{\sigma\rho}$ has the same block structure as for $Q^{\sigma\rho}$ with matrices $\hat{Q}_0$ and $\hat{Q}_1$. The ansatz for the remaining order parameters M^{σ}, $\theta_i{}^{\sigma}$, and $\hat{E}^{\sigma}$ is as for RS (see (A.10)). Note that q_1 and q_0 are now overlap parameters which describe different aspects of the solution space of the single perceptron and can be interpreted as the typical overlap between pairs of weight vectors in the same and different solution spaces, respectively.

After some algebra, including the elimination of the conjugate order parameters ($\hat{q}_1$, $\hat{q}_0$, and $\hat{E}$) and the introduction of the self-consistent ansatz for m that $w = m/(1 - q_1)$ remains finite in the $\beta \to \infty$ limit, the one-step RSB free energy can be simplified to

$$\langle\langle -f_{\mathrm{RSB}} \rangle\rangle = \frac{\alpha}{wx} \left\langle \int \mathrm{D}t \, \log[\mathcal{F}_{\mathrm{RSB}}(t, w, x, q_0, \kappa, \zeta\theta)] \right\rangle_{\zeta} + \frac{q_0}{2x(1 + w\Delta q)} + \frac{\log(1 + w\Delta q)}{2wx} \tag{B.2a}$$

where τ and $\mathrm{D}t$ are taken from (A.16), $\Delta q = 1 - q_0$, and the function $\mathcal{F}_{\mathrm{RSB}}$ is given by

$$\mathcal{F}_{\mathrm{RSB}}(t, w, x, q_0, \kappa, \zeta\theta) = \int_{-\frac{\mu}{\sqrt{\Delta q}}}^{\frac{\sqrt{2x}-\mu}{\sqrt{\Delta q}}} \mathrm{D}z \, \exp\left[-\frac{w}{2} \left(\sqrt{\Delta q}\, z + \mu \right)^2 \right]$$

$$+H\left(\frac{\mu}{\sqrt{\Delta q}}\right)+\mathrm{e}^{-wx}H\left(\frac{\sqrt{2x}-\mu}{\sqrt{\Delta q}}\right) \tag{B.2b}$$

with $\mu=\tau+\sqrt{q_0}\,t$. The free energy has to be evaluated at the saddle points with respect to the variables w, x, q_0, and θ.

B.1. The Ising perceptron

For the Ising perceptron the replica calculation is identical to the spherical perceptron but for a change in the prior constraint Hamiltonian, where the integration over weight space is performed. However, this leads to significant changes when calculating the capacity limit or the $\beta\to\infty$ limit above saturation, due to the discreteness of the weight vector. Again, the RS ansatz breaks down above the capacity limit α_{c}, which, however, coincides with the entropy of the RS solution becoming zero and is not given in the $x\to\infty$ limit†. The entropy is given by

$$s=-\frac{1}{2}(1-q_1)\hat{q}_1+\int \mathrm{D}t\ \log\left[2\cosh\left(t\sqrt{\hat{q}_1}\right)\right]+\alpha\left\langle\int \mathrm{D}t\ \log H\left(\frac{\tau+\sqrt{q_1}\,t}{\sqrt{1-q_1}}\right)\right\rangle_{\zeta} \tag{B.3}$$

and has to be evaluated at its saddle point with respect to q_1, $\hat{q}_1$, and θ.

Above the capacity limit, a one-step RSB solution is found characterized by $q_1=1$, $\hat{q}_1=\infty$ for finite β, and the self-consistent ansätze that $v=wx=m\beta$ and $y=m\sqrt{\hat{q}_0}$ remain finite for $\beta\to\infty$. The one-step RSB free energy of the Ising perceptron is then given by

$$\langle\!\langle -f_{\mathrm{IRSB}}\rangle\!\rangle=\frac{\alpha}{v}\left\langle\int \mathrm{D}t\ \log\left[\mathcal{F}_{\mathrm{IRSB}}(t,v,y,q_0,\kappa,\zeta\theta)\right]\right\rangle_{\zeta}$$
$$+\frac{1}{v}\int \mathrm{D}t\ \log[2\cosh(yt)]-\frac{\Delta q\,y^2}{2v} \tag{B.4a}$$

where the function $\mathcal{F}_{\mathrm{IRSB}}$ is

$$\mathcal{F}_{\mathrm{IRSB}}(t,v,y,q_0,\kappa,\zeta\theta)=\mathrm{e}^{-v}+(1-\mathrm{e}^{-v})H\left(\frac{\mu}{\sqrt{\Delta q}}\right) \tag{B.4b}$$

with μ as before. The free energy has to be evaluated at its saddle point with respect to the variables v, y, q_0, and θ.

B.2. The error rates

The error rates of wrongly-off and wrongly-on patterns for the spherical perceptron (within both a RS and one-step RSB ansatz) and for the Ising perceptron (within a one-step RSB ansatz) are given by

$$\epsilon_{\mathrm{RS}}^{\mathrm{off/on}}=\tfrac{1}{2}(1\pm m_{\mathrm{o}})H\left(\sqrt{2x}-\kappa\mp\theta\right) \tag{B.5a}$$

$$\epsilon_{\mathrm{RSB}}^{\mathrm{off/on}}=\tfrac{1}{2}(1\pm m_{\mathrm{o}})\int \mathrm{D}t\ \frac{\mathrm{e}^{-wx}H\left(\sqrt{2x}-\sqrt{q_0}\,t-\kappa\mp\theta\right)}{\mathcal{F}_{\mathrm{RSB}}(t,w,x,q_0,\kappa,\pm\theta)} \tag{B.5b}$$

$$\epsilon_{\mathrm{IRSB}}^{\mathrm{off/on}}=\tfrac{1}{2}(1\pm m_{\mathrm{o}})\int \mathrm{D}t\ \frac{\mathrm{e}^{-v}H\left(\sqrt{2x}-\sqrt{q_0}\,t-\kappa\mp\theta\right)}{\mathcal{F}_{\mathrm{IRSB}}(t,v,y,q_0,\kappa,\pm\theta)} \tag{B.5c}$$

respectively, where the order parameters are determined by the respective saddle points.

† In this region the ansatz for x breaks down, since $q_1<1$ even for $\beta\to\infty$.

Appendix C. Propagation of errors for the upstart algorithm

Similarly to the tiling-like algorithm, the capacity of networks built by variants of the upstart algorithm can be calculated by propagating the errors of the individual perceptrons. Consider for example the procedure for upstart IIIa. Again the rules of the algorithm as described in section 2 are followed and the similar considerations to section 3.3 are applied. The following notable differences are caused by the upstart algorithm creating two different types of units, $\mathcal{U}^+$ and $\mathcal{U}^-$. First, the procedure becomes dependent on the type of error made, i.e. is written in terms of $\epsilon^{\rm on}$ and $\epsilon^{\rm off}$ instead of $\epsilon = \epsilon^{\rm on} + \epsilon^{\rm off}$. Furthermore, since $\mathcal{U}^-$ is connected to the output with a negative weight, the rôles of *off* and *on* are reversed, e.g. wrongly-on patterns of $\mathcal{U}^-$ are actually wrongly-off patterns for $\mathcal{M}$. Second, the example load α_i decreases from the initial load α_0 over subsequent generations, since patterns can be omitted from the training sets and it is useful to introduce the quantities $\bar{\alpha}_i \equiv \alpha_i/\alpha_0$, which is the probability of an example being in the training set and will be referred to as load fractions. Third, $\widetilde{m}_i$ is in general negative since the target of the majority of the patterns is 0 (or -1) and not $+1$ as in the tiling-like algorithm†, and $\widetilde{m}_i \equiv 1 + m_i$. It is furthermore useful to restrict the bias of the original output distribution to $m_{\rm o} < 0$ w.l.o.g. and to introduce the fraction of training patterns $\bar{\alpha}_0^{\zeta^\pm} \equiv (1 \pm m_{\rm o})/2$ with $\zeta^\pm = \pm 1$. Fourth, it becomes necessary to propagate errors over several generations, since only one type of unit is created with each generation and subsequently only one type of error is dealt with. A symbolic program for the procedure called by the capacity root solver is outlined in figure C1. The procedure for upstart IIIb is identical to upstart IIIa, but for a change in the creation criterion, which is changed to $(\epsilon_i^{\rm on}\bar{\alpha}_0^{\zeta^-} > \epsilon_i^{\rm off}\bar{\alpha}_0^{\zeta^+})$ to account for conditioning the error type probability on the initial target probability.

For the procedure of upstart II, two major changes have to be made. The first is induced by the different training set criteria, i.e. the inclusion of wrongly-off patterns into $\mathcal{U}_i^-$'s (and wrongly-on patterns from $\mathcal{U}_i^+$'s) training set. The load fractions for $\mathcal{U}_{i+1}^\pm$ are therefore changed to $\bar{\alpha}_{i+1} := \epsilon_i^{\rm off/on} + \bar{\alpha}_0^{\zeta^\mp}$. Since these incorrect pattern are included in the training set, although no attempt is been made to correct them, it is self-evident, that it is possible to make an error on an example which is already labelled as incorrect. Therefore, the calculated error rates have to be corrected in order to avoid such multiple counting of errors. Employing the assumption that the perceptrons are uncorrelated, the overall errors after the creation of a $\mathcal{U}_i^\pm$ unit in the current generation become

$$\epsilon_i^{\rm off/on} = \bar{\alpha}_i \epsilon^{\rm off} \qquad \text{and} \qquad \epsilon_i^{\rm on/off} = \epsilon_{i-1}^{\rm on/off} + \epsilon^{\rm on}\left[1 - \frac{\epsilon_{i-1}^{\rm on/off}}{\bar{\alpha}_0^{\zeta^\mp}}\right]\bar{\alpha}_i . \tag{C.1}$$

The second major changes caters for cases where both types of perceptrons are created simultaneously, e.g. allowing for the possibility of two error rate calculations in the inner loop. This case necessitates the introduction of unit specific quantities, such as $\widetilde{m}_i^\pm$ and $\bar{\alpha}_i^\pm$, where i is now purely a generation index‡. Note, that in this case the errors are not propagated over generations, since both type of errors are corrected simultaneously, but the simultaneous training combined with the non-zero overlap of patterns between the two training sets again leads to an overlap of the erroneous patterns. Following similar

† Due to the symmetry of the equations for $m_{\rm o} \to -m_{\rm o}$, $\zeta^\mu \to -\zeta^\mu$, and $\theta \to -\theta$, this effectively only changes $\epsilon^{\rm on} \leftrightarrow \epsilon^{\rm off}$ and the sign of the threshold.

‡ The number of units created have therefore to be counted by a separate label.

upstartIIIa_cap := proc(α_c^K) % begin procedure
global K, κ, m_o; % global variables
local $\cdots$; % local variables (here unspecified)
$\alpha_0 := K\alpha_c^K$; $\tilde{m}_o := 1 + m_o$; $\bar{\alpha}_0^{\zeta^+} := \tilde{m}_o/2$; $\bar{\alpha}_0^{\zeta^-} := 1 - \bar{\alpha}_0^{\zeta^+}$; % more global
m_flag:= FALSE; % flag for unit type
$\bar{\alpha}_1 := 1$; $\tilde{m}_1 := \tilde{m}_o$; % initialize first perceptron $\mathcal{U}_1^+$
ϵ_0^{on}:=0;$\epsilon_0^{off} := 0$; % initialize errors
for $(i = 1, K-2)$ do % loop over the erroneous perceptrons
  $\alpha_i := \alpha_0\bar{\alpha}_i$; % calculate load
  $(\epsilon^{on}, \epsilon^{off}) :=$ error_calc$(\alpha_i, \tilde{m}_i, \kappa)$; % calculate error rates
  if (m_flag) then $\epsilon^{on} \leftrightarrow \epsilon^{off}$; fi; % swap error types for $\mathcal{U}_i^-$
  $\epsilon_i^{on} := \epsilon_{i-1}^{on} + \bar{\alpha}_i\epsilon^{on}$; % accumulate error rates
  $\epsilon_i^{off} := \epsilon_{i-1}^{off} + \bar{\alpha}_i\epsilon^{off}$; % (with respect to α_0)
  if $(\epsilon_i^{on} > \epsilon_i^{off})$ then % apply creation criterion
    $\bar{\alpha}_{i+1} := \epsilon_i^{on} + \bar{\alpha}_0^{\zeta^+} - \epsilon_i^{off}$; % wrongly-on and correctly-on patterns
    $\tilde{m}_{i+1} := 2\epsilon_i^{on}/\bar{\alpha}_{i+1}$; % calculate new bias
    m_flag:= TRUE; $\epsilon_i^{on} := 0$; % create $\mathcal{U}_{m+1}^-$ and reset ϵ^{on}
  else
    $\bar{\alpha}_{i+1} := \epsilon_i^{off} + \bar{\alpha}_0^{\zeta^-} - \epsilon_i^{on}$; % wrongly-off and correctly-off patterns
    $\tilde{m}_{i+1} := 2\epsilon_i^{off}/\bar{\alpha}_{i+1}$; % calculate new bias
    m_flag:= FALSE; $\epsilon_i^{off} := 0$; % create $\mathcal{U}_{p+1}^+$ and reset ϵ^{off}
  fi;
od; % have reached last two perceptrons
if (m_flag) then % have already created $\mathcal{U}_{m+1}^-$
  $p := p + 1$; % additionally create $\mathcal{U}_{p+1}^+$
  $\bar{\alpha}_K := \epsilon_{K-1}^{off} + \bar{\alpha}_0^{\zeta^-}$; $\tilde{m}_K := 2\epsilon_{K-1}^{off}/\bar{\alpha}_K$;
else % have already created $\mathcal{U}_{p+1}^+$
  $m := m + 1$; % additionally create $\mathcal{U}_{p+1}^-$
  $\bar{\alpha}_K := \epsilon_{K-1}^{on} + \bar{\alpha}_0^{\zeta^+}$; $\tilde{m}_K := 2\epsilon_{K-1}^{on}/\bar{\alpha}_K$;
fi;
$\alpha_{K-1} := \alpha_0\bar{\alpha}_{K-1}$; $\alpha_K := \alpha_0\bar{\alpha}_K$; % calculate loads
$\alpha_c^a :=$ perc_cap$(\tilde{m}_{K-1}, \kappa)$ % calculate the two capacity limit
$\alpha_c^b :=$ perc_cap$(\tilde{m}_K, \kappa)$
$\Delta\alpha^a := \alpha_c^a - \alpha_{K-1}$; $\Delta\alpha^b := \alpha_c^b - \alpha_K$; % difference of capacity and load
RETURN(min $(\Delta\alpha^a, \Delta\alpha^b)$); % return more saturated perceptron
end;

Figure C1. A symbolic capacity calculation procedure of the upstart IIIa algorithm called by an all-purpose root solving routine.

considerations as above, one finds

$$\epsilon_i^{off/on} = \epsilon_{\pm}^{off}\bar{\alpha}_i^{\pm} + \epsilon_{\mp}^{on}\left[1 - \epsilon_{\pm}^{off}\frac{\bar{\alpha}_i^{\pm}}{\bar{\alpha}_0^{\zeta^{\mp}}}\right]\bar{\alpha}_i^{\mp}. \tag{C.2}$$

A closer inspection of (C.2) reveals that its symmetry leads to identical wrongly-on and wrongly-off errors for all generations of the algorithm if the initial output distribution bias is zero ($\bar{\alpha}_0^{\zeta^{\pm}} = \frac{1}{2}$) and the first perceptron makes the identical fraction of error types, i.e. the solution with zero threshold (see discussion in section 3.2) applies. In this

case, the computational load for the capacity calculation is reduced by a factor of two and the symmetry leads to much smoother capacity curves as the two last units saturate simultaneously.

Appendix D. Derivation of the asympotitic capacity of the upstart algorithm

In the case of upstart algorithm variants, the capacity of the complete network is a function of the capacity α_c^K of the last perceptron $\mathcal{U}_K^{\pm}$ and its load fraction $\bar{\alpha}_K^{\pm}$, due to some patterns being eliminated from its training set. For upstart II, $\bar{\alpha}_K^{\pm}$ can be expressed in terms of the applied bias $\widetilde{m}_K^{\pm}$, and the initial output-distribution bias m_o [which is for convenience again written in terms of the initial load fractions $\bar{\alpha}_0^{\zeta^{\pm}} \equiv (1 \pm m_o)/2$] to yield

$$\alpha_c^K = \frac{\alpha_c(\widetilde{m}_K^{\pm})}{K} \frac{(2 - \widetilde{m}_K^{\pm})}{2\bar{\alpha}_0^{\zeta^{\mp}}}. \tag{D.1}$$

For $K \to \infty$ and consequently $\widetilde{m}_K^{\pm} \to 0$, the capacity becomes asymptotically

$$\alpha_c^K \simeq \frac{1}{b\bar{\alpha}_0^{\zeta^{\mp}}} [\log(K)]^{m-1} \tag{D.2}$$

following the derivation for the tiling-like algorithm in section 5.

For upstart III, the derivation becomes slightly more complicated due to the additional exclusion of one class of incorrect patterns from the training set of the saturated unit. In this case, $\bar{\alpha}_K^{\pm}$ can only be bounded in terms of $\widetilde{m}_K^{\pm}$ and $\bar{\alpha}_0^{\zeta^{\pm}}$, leading to bounds on the capacity which are specific on the unit creation selection criterion applied. For upstart IIIa, the capacity bounds are

$$\frac{\alpha_c(\widetilde{m}_K^{\pm})}{K} \frac{(2 - \widetilde{m}_K^{\pm})}{2\bar{\alpha}_0^{\zeta^{\mp}}} \leqslant \alpha_c^K \leqslant \frac{\alpha_c(\widetilde{m}_K^{\pm})}{K} \frac{1}{\bar{\alpha}_0^{\zeta^{\mp}}} \tag{D.3}$$

and similarly for upstart IIIb, one finds

$$\frac{\alpha_c(\widetilde{m}_K^{\pm})}{K} \frac{(2 - \widetilde{m}_K^{\pm})}{2\bar{\alpha}_0^{\zeta^{\mp}}} \leqslant \alpha_c^K \leqslant \frac{\alpha_c(\widetilde{m}_K^{\pm})}{K} \frac{[2 - (2 - 1/\bar{\alpha}_0^{\zeta^{\mp}})\widetilde{m}_K^{\pm}]}{2\bar{\alpha}_0^{\zeta^{\mp}}}. \tag{D.4}$$

However, for $K \to \infty$ ($\widetilde{m}_K^{\pm} \to 0$), these bounds become tight and the asymptotic capacity again reduces to (D.2).

References

[1] Gardner E 1988 *J. Phys. A: Math. Gen.* **21** 257–70
[2] Gardner E and Derrida B 1988 *J. Phys. A: Math. Gen.* **21** 271–84
Bouten M 1994 *J. Phys. A: Math. Gen.* **27** 6021–23
Derrida B 1994 *J. Phys. A: Math. Gen.* **27** 6025
[3] Mézard M, Parisi G and Virasoro M G 1987 *Spin Glass Theory and Beyond* (Singapore: World Scientific)
[4] Blumer A, Ehrenfeucht A, Haussler D and Warmuth M K 1989 *J. Ass. Comp. Mach.* **36** 929–65
[5] Vapnik V N and Chervonenkis A Y 1971 *Theory Prob. Appl.* **16** 264–80
[6] Baum E B and Haussler D 1989 *Neural Comput.* **1** 151–60
[7] Mitchison G J and Durbin R M 1989 *Biol. Cybern.* **60** 345–56
[8] Bartlett P L 1993 *Neural Comput.* **5** 371–3
[9] Wendemuth A 1994 *Int. J. Neural Syst.* **5** 217–28
[10] Maass W 1994 *Neural Comput.* **6** 877–84
[11] Sakurai A 1995 *Theor. Comp. Sc.* **137** 109–27
[12] Saad D and Solla S A 1995 *Phys. Rev.* E **52** 4225–43

[13] Barkai E, Hansel D and Kanter I 1990 *Phys. Rev. Lett.* **65** 2312–5
Barkai E 1990 *Phys. Rev. Lett.* **65** 3210
[14] Barkai E, Hansel D and Sompolinsky H 1992 *Phys. Rev.* A **45** 4146–61
[15] Engel A, Köhler H M, Tschepke F, Vollmayr H and Zippelius A 1992 *Phys. Rev.* A **45** 7590–609
[16] Cover T M 1965 *Trans. Elec. Comp.* **14** 326–34
[17] Saad D 1994 *J. Phys. A: Math. Gen.* **27** 2719–34
[18] Monasson R and Zecchina R 1995 *Phys. Rev. Lett.* **75** 2432–5
[19] Monasson R and Zecchina R 1996 *Mod. Phys. Lett. B* **9** 1887–97
[20] Urbanczik R 1997 *J. Phys. A: Math. Gen.* **30** L387–92
[21] Xiong Y, Oh J-H and Kwon C 1997 *Phys. Rev.* E **56** 4540–4
[22] West A H L and Saad D 1997 *J. Phys. A: Math. Gen.* **30** 3471–96
[23] Gallant S I 1990 *IEEE Trans. Neural Netw.* **1** 179–91
Gallant S I 1986 *Proc. 8th Ann. Conf. Cognitive Science Society* (Hillsdale, NJ: Lawrence Erlbaum) pp 652–60
Gallant S I 1986 *Proc. 8th IEEE Int. Conf. Pattern Recognition* (Washington DC: IEEE Computer Society) pp 849–52
[24] Ash T 1989 *Connect. Sci.* **1** 365–75
[25] Mézard M and Nadal J-P 1989 *J. Phys. A: Math. Gen.* **22** 2191–203
[26] Nadal J-P 1989 *Int. J. Neural Syst.* **1** 55–9
[27] Fahlman S E and Lebiere C 1990 *Advances in Neural Information Processing Systems 2* ed D S Touretzky (San Mateo, CA: Morgan Kaufmann) pp 524–32
[28] Frean M 1990 *Neural Comput.* **2** 198–209
[29] Campbell C and Perez Vicente C J 1995 *Neural Comput.* **7** 1245–64
[30] Fritzke B 1994 *Neural Netw.* **7** 1441–60
[31] Marchand M, Golea M and Rujan P 1990 *Europhys. Lett.* **11** 487–92
[32] Marchand M and Golea M 1993 *Network: Comput. Neural Syst.* **4** 67–85
[33] Golea M and Marchand M 1990 *Europhys. Lett.* **12** 205–10
[34] Zollner R, Schmitz H J, Wunsch F and Krey U 1992 *Neural Netw.* **5** 771–7
[35] Martinez D and Estève D 1992 *Europhys. Lett.* **18** 95–100
[36] Rosenblatt F 1962 *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* (Washington DC: Spartan)
[37] Minsky M L and Papert S A 1969 *Perceptrons* (Cambridge, MA: MIT Press)
[38] Diederich S and Opper M 1987 *Phys. Rev. Lett.* **58** 949–52
[39] Krauth W and Mézard M 1987 *J. Phys. A: Math. Gen.* **20** L745–52
[40] Anlauf J K and Biehl M 1989 *Europhys. Lett.* **10** 687–92
[41] Frean M 1992 *Neural Comput.* **4** 946–57
[42] Wendemuth A 1995 *J. Phys. A: Math. Gen.* **28** 5423–36
Wendemuth A 1995 *J. Phys. A: Math. Gen.* **28** 5485–93
[43] Werbos P J 1974 *PhD Thesis* Harvard University
[44] Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* **323** 533–6
Rumelhart D E, Hinton G E and Williams R J 1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* vol 1, ed D E Rumelhart *et al* (Cambridge MA: MIT Press) p 318
[45] Grossman T, Meir R and Domany E 1989 *Complex Syst.* **2** 555–75
[46] Bishop C M 1995 *Neural Networks for Pattern Recognition* (Oxford: Clarendon Press)
[47] Schapire R E 1990 *Mach. Learn.* **5** 197–227
[48] Freund Y 1995 *Inf. Comput.* **121** 256–85
[49] Freund Y and Schapire R E 1995 *Computational Learning Theory: 2nd Eur. Conf., EuroCOLT '95 (Barcelona)* (Heidelberg: Springer) pp 23–27
Freund Y and Schapire R E 1996 (unpublished, see http://www.research.att.com/∼yoav)
[50] Drucker H, Cortes C, Jackel L D, Le Cun Y and Vapnik V 1994 *Neural Comput.* **6** 1289–301
[51] Littmann E and Ritter H 1996 *Neural Comput.* **8** 1521–39
[52] Mozer M C and Smolensky P 1989 *Advances in Neural Information Processing Systems 1* ed D S Touretzky (San Mateo, CA: Morgan Kaufmann) pp 107–15
[53] Chauvin Y 1989 *Advances in Neural Information Processing Systems 1* ed Touretzky D S (San Mateo, CA: Morgan Kaufmann) pp 519–26
[54] Le Cun Y, Denker J and Solla S A 1990 *Advances in Neural Information Processing Systems 2* ed D S Touretzky (San Mateo, CA: Morgan Kaufmann) pp 598–605
[55] Hassibi B and Stork D G 1993 *Advances in Neural Information Processing Systems 5* ed S J Hanson *et al* (San Mateo, CA: Morgan Kaufmann) pp 164–71

[56] Levin A U, Leen T K and Moody J E 1994 *Advances in Neural Information Processing Systems 6* ed J D Cowan *et al* (San Mateo, CA: Morgan Kaufmann) pp 35–42
[57] Weigend A S, Rumelhart D E, and Huberman B A 1990 *Proceedings of the 1990 Connectionist Models Summer School* ed D S Touretzky *et al* (San Mateo, CA: Morgan Kaufmann) pp 105–16
[58] Nowlan S J and Hinton G E 1992 *Neural Comput.* **4** 473–93
[59] Setiono R 1997 *Neural Comput.* **9** 185–204
[60] Neal R M 1996 *Bayesian Learning for Neural Networks* (Heidelberg: Springer)
[61] Frean M 1990 *PhD Thesis* Edinburgh University
[62] Biehl M and Opper M 1991 *Phys. Rev.* A **44** 6888–94
[63] Whyte W and Sherrington D 1996 *J. Phys. A: Math. Gen.* **29** 3063–73
[64] West A H L and Saad D 1997 *Mathematics of Neural Networks: Models, Algorithms and Applications* ed S W Ellacott *et al* (Boston, MA: Kluwer) pp 372–7