Understanding the *exploding gradient* problem

Razvan Pascanu, Tomas Mikolov, Yoshua Bengio
r.pascanu@gmail.com
imikolov@fit.vutbr.cz
bengioy@iro.umontreal.ca
Université de Montréal
Brno University of Technology

Abstract

Training Recurrent Neural Networks is more troublesome than feedforward ones because of the *vanishing and exploding* gradient problems detailed in Bengio *et al.* (1994). In this paper we attempt to understand the fundamental issues underlying the *exploding gradient problem* by exploring it from an analytical, a geometric and a dynamical system perspective. Our analysis is used to justify the simple yet effective solution of norm clipping the exploded gradient. In the experimental section, the comparison between this heuristic solution and standard SGD provides empirical evidence towards our hypothesis as well as it shows that such a heuristic is required to reach state of the art results on a character prediction task and a polyphonic music prediction one.

1 Introduction

A recurrent neural network (RNN), see Figure 1, is a neural network model proposed in the 80's (Rumelhart *et al.*, 1986; Elman, 1990; Werbos, 1988) for modeling time series. The structure of the network is similar to that of a standard multilayer perceptron, with the distinction that we allow connections among hidden units associated with a time delay. Through these connections the model can retain information about the past inputs, enabling it to discover temporal correlations between events that are possibly far away from each other in the data (a crucial property for proper learning of time series).

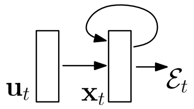

Figure 1: Schematic of a recurrent neural network. The recurrent connections in the hidden layer allow information to persist from one input to another.

While in principle the recurrent network is a simple and powerful model, in practice, it is unfortunately hard to train properly. A plethora of training algorithms have been proposed in the literature, like Backpropagation Through Time (BPTT) (Rumelhart *et al.*, 1986; Werbos, 1988), Real Time Recurrent Learning (Williams and Zipser, 1989), Atiya-Parlos learning rule (Atiya and Parlos, 2000), etc. Most of these are gradient based, though alternative approaches are also available (for example see Lukoševičius and Jaeger (2009)), and as shown in Atiya and Parlos (2000) most gradient-based methods behave qualitatively the same, providing little success in properly addressing complex tasks. Among the main reasons why this model is so unwieldy are the *vanishing gradient* and *exploding gradient* problems described in Bengio *et al.* (1994).

In this paper we will only address the exploding gradient problem and provide an understanding of the issues stochastic gradient descent (SGD) faces when training a recurrent network. We should note that there are a few solutions proposed for this problem in the literature, amongst which the Hessian-Free learning (Sutskever *et al.*, 2011), a second order method that seems promising, though more needs to be done to analyze its success, especially compared to other second order methods that seem to do less well in general. Long Short Term Memory networks (Hochreiter and Schmidhuber, 1997) is another approach, relying on a change in the structure of the model, and designed to help with the vanishing gradient problem. The extend to which this alteration limits the modeling power of the model is not clear, nor does the approach address the exploding gradient problem. We do not intend to directly compete with these solutions, but rather improve the limited understanding of why such approaches are required. To validate some of our hypotheses we devise a simple alteration of SGD which, given its simplicity, can be seen as a reliable alternative to SGD learning.

The structure of the paper is as follows. In subsection 1.1 we briefly formalise the concept of training recurrent networks. In section 2 we introduce and analyze the exploding gradient problem, while in section 3 we describe our proposed solution. Section 4 provides empirical experimentation on different datasets and finally in section 5 we have some final remarks.

1.1 Training recurrent networks

A generic recurrent neural network is given by equation (1). In the theoretical section of this paper we will sometimes make use of the specific parametrization given by equation (2) [1] in order to provide more precise conditions and intuitions about the everyday use-case.

$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{u}_t, \theta) \tag{1}$$

$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b} \tag{2}$$

The parameters of the model are given by the recurrent weight matrix $\mathbf{W}_{rec}$, the biases $\mathbf{b}$ and input weight matrix $\mathbf{W}_{in}$, collected in θ for the general case. $\mathbf{x}_t$ and $\mathbf{u}_t$ represent the state and the input at time t, where $\mathbf{x}_0$ is provided by the user or set to zero (or learned), and σ is an element-wise function (usually the *sigmoid* or *tanh*). A cost $\mathcal{E}$ measures the performance of the network on some given task and it can be broken apart into individual costs for each step $\mathcal{E} = \sum_{1 \leq t \leq T} \mathcal{E}_t$, where $\mathcal{E}_t = \mathcal{L}(\mathbf{x}_t)$.

One approach that can be used to compute the necessary gradients for learning is to represent the recurrent model as a deep multi-layer one (with an unbounded number of layers) and apply backpropagation on the unrolled model (see Figure 2).

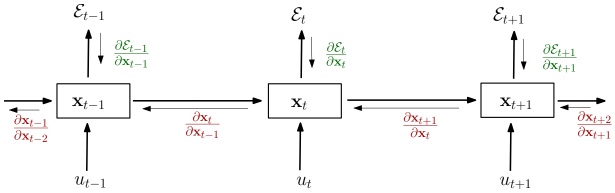
$\mathcal{E}_{t-1}$
$\frac{\partial \mathcal{E}_{t-1}}{\partial \mathbf{x}_{t-1}}$
$\mathcal{E}_t$
$\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}$
$\mathcal{E}_{t+1}$
$\frac{\partial \mathcal{E}_{t+1}}{\partial \mathbf{x}_{t+1}}$
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
$\mathbf{x}_{t+1}$
$\frac{\partial \mathbf{x}_{t-1}}{\partial \mathbf{x}_{t-2}}$
$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}$
$\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}$
$\frac{\partial \mathbf{x}_{t+2}}{\partial \mathbf{x}_{t+1}}$
u_{t-1}
u_t
u_{t+1}

Figure 2: Unrolling recurrent neural networks in time by creating a copy of the model for each time step. We denote by $\mathbf{x}_t$ the hidden state of the network at time t, by $\mathbf{u}_t$ the input of the network at time t and by $\mathcal{E}_t$ the error obtained from the output at time t.

We will diverge from the classical BPTT equations at this point and re-write the gradients (see equations (3), (4) and (5)) in order to better highlight the exploding gradient problem.

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta} \tag{3}$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \tag{4}$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \tag{5}$$

where ∂^+ represents the partial derivative of the state at some point in time with respect to some parameter *as a direct argument*, i.e., without considering the recurrent path (i.e. when computing the derivative of $\mathbf{x}_k$ with respect to θ we ignore the fact that $\mathbf{x}_k$ is a function of $\mathbf{x}_{k-1}$ which in turn is a function of θ). Equation (5) also provides the form of $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}$ for the

[1] This formulation is equivalent to the more widely known equation $\mathbf{x}_t = \sigma(\mathbf{W}_{rec}\mathbf{x}_{t-1} + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b})$, and it was chosen for convenience.

specific parametrization given in equation (2), where *diag* converts a vector into a diagonal matrix, and σ' computes the derivative of σ in an element-wise fashion.

Note that each term $\frac{\partial \mathcal{E}_t}{\partial \theta}$ from equation (3) has the same form and the behaviour of these individual terms determine the behaviour of the sum. Henceforth we will focus on one such generic term, calling it simply the gradient when there is no confusion.

Any gradient component $\frac{\partial \mathcal{E}_t}{\partial \theta}$ is also a sum (see equation (4)), whose terms we refer to as *temporal* contributions or *temporal* components. One can see that each such temporal contribution $\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}\frac{\partial^+ \mathbf{x}_k}{\partial \theta}$ measures how θ at step k affects the cost at step t. The factors $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}$ (equation (5)) transport the error “in time“ from step k to step t. We would further loosely distinguish between *long term* and *short term* contributions, where long term refers to components for which $k \ll t$ and short term to everything else.

2 Exploding Gradients

As introduced in Bengio *et al.* (1994), the *exploding gradient* problems refers to the large increase in the norm of the gradient during training. Such events are caused by the explosion of the long term components, which can grow exponentially more then short term ones.

2.1 The mechanics

To understand this phenomenon we need to look at the form of each temporal component, and in particular at the factors $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}$ (see equation (5)) that take the form of a product of l Jacobian matrices, with $l = t - k$. Intuitively, these products can grow exponentially fast with l (in some direction $\mathbf{v}$), leading to the explosion of long term components when l is large. Because the gradient is just a sum of these components, it follows that it should also grow exponentially fast following the long term component with $k = 0$ (for which $l = t$). In what follows we will try to formalize these intuitions (extending a similar derivation done in Bengio *et al.* (1994) where only a single hidden unit case was considered).

2.2 Linear model

Let us consider the term $\mathbf{g}_k^T = \frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}\frac{\partial^+ \mathbf{x}_k}{\partial \theta}$ for the linear version of the parametrization in equation (2) (i.e. set σ to the identity function) and assume t goes to infinity. We have that:

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \left(\mathbf{W}_{rec}^T\right)^l \qquad (6)$$

By employing the same approach as the *power iteration method* we can show that, given certain conditions, $\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}\left(\mathbf{W}_{rec}^T\right)^l$ grows exponentially.

Proof Let $\mathbf{W}_{rec}$ have the eigenvalues $\lambda_1, .., \lambda_n$ with $|\lambda_1| > |\lambda_2| > .. > |\lambda_n|$ and the corresponding eigenvectors $\mathbf{q}_1, \mathbf{q}_2, .., \mathbf{q}_n$ which form a vector basis. We can now write the row vector $\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}$ into this basis:
$\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} = \sum_{i=1}^N c_i \mathbf{q}_i^T$

If j is such that $c_j \neq 0$ and any $j' < j, c_{j'} = 0$, using the fact that $\mathbf{q}_i^T\left(\mathbf{W}_{rec}^T\right)^l = \lambda_i^l \mathbf{q}_i^T$ we have that

$$\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = c_j \lambda_j^l \mathbf{q}_j^T + \lambda_j^l \sum_{i=j+1}^{n} c_i \frac{\lambda_i^l}{\lambda_j^l}\mathbf{q}_i^T \approx c_j \lambda_j^l \mathbf{q}_j^T \quad (7)$$

We used the fact that $\left|{}^{\lambda_i}/_{\lambda_j}\right| < 1$ for $i > j$, which means that $\lim_{l \to \infty}\left|{}^{\lambda_i}/_{\lambda_j}\right|^l = 0$. If $|\lambda_j| > 1$, it follows that $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}$ grows exponentially fast with l, and it does so along the direction $\mathbf{q}_j$. □

The proof assumes $\mathbf{W}_{rec}$ is diagonalizable for simplicity, though using the Jordan normal form of $\mathbf{W}_{rec}$ one can extend this proof by considering not just the eigenvector of largest eigenvalue but the whole subspace spanned by the the eigenvectors sharing the same (largest) eigenvalue.

This result provides a necessary condition for gradients to grow, namely that the spectral radius (the absolute value of the largest eigenvalue) of $\mathbf{W}_{rec}$ must be larger than 1.

If $\mathbf{q}_j$ is not in the null space of $\frac{\partial^+ \mathbf{x}_k}{\partial \theta}$ the entire temporal component grows exponentially with l.

This approach extends easily to the entire gradient. If we re-write it in terms of the eigen-decomposition of $\mathbf{W}$, we get:

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{j=1}^{n}\left(\sum_{i=k}^{t} c_j \lambda_j^{t-k} \mathbf{q}_j^T \frac{\partial^+ \mathbf{x}_k}{\partial \theta}\right) \qquad (8)$$

We can now pick j and k such that $c_j \mathbf{q}_j^T \frac{\partial^+ \mathbf{x}_k}{\partial \theta}$ does not have 0 norm, while maximizing $|\lambda_j|$. If for the chosen j it holds that $|\lambda_j| > 1$ then $\lambda_j^{t-k} c_j \mathbf{q}_j^T \frac{\partial^+ \mathbf{x}_k}{\partial \theta}$ will dominate the sum and because this term grows exponentially fast to infinity with t, the same will happen to the sum.

2.3 Nonlinear model

To generalize this proof to the nonlinear case, we define the concept of expanding and non-expanding matrices for some direction $\mathbf{v}$. We say that the Jacobian matrix $\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}}$ expands along a vector $\mathbf{v}$ by $\alpha > 1$ if equation (9) holds.

$$\forall \mathbf{u}, \left| \mathbf{u}^T \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \mathbf{v} \right| > \alpha |\mathbf{u}^T \mathbf{v}| \tag{9}$$

Intuitively, to achieve exponential growth, it is sufficient that most of the Jacobians $\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}}$ are expanding, such that their product expands exponentially with their number and also that there is no Jacobian matrix that kills off these exponentially large increases.

We formalize this by constructing the set P of matrices that are non-expanding, and considering a lower bound $\beta > 0$ on how much these matrices shrink a vector in the direction $\mathbf{v}$ (see equation (10)).

$$\forall \mathbf{u}, \left| \mathbf{u}^T \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \mathbf{v} \right| > \beta |\mathbf{u}^T \mathbf{v}|, \text{ if } \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \in P \tag{10}$$

This means that if α is the least amount by which any matrix $\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \notin P$ expands, $\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k}$ should expand roughly by $\beta^{|P|} \alpha^{t-|P|}$. If the cardinality of P is bounded as t grows, it means this product grows exponentially fast with $t - |P|$.

It is worth mentioning that α is bounded by the spectral radius of each matrix $\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}}$ (which is easy to see as $\left\| \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \mathbf{v} \right\| \leq \left\| \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \right\| \|\mathbf{v}\|$). If we consider the parametrization in equation (2), this spectral radius is in its turn bounded by the product of the spectral radii $\rho_{\mathbf{W}_{rec}}$ of $\mathbf{W}_{rec}$ and $\rho_{\sigma'}$ of $diag(\sigma'(\mathbf{x}_{i-1}))$. We know that $\rho_{\sigma'} < 1$ for tanh and $\rho_{\sigma'} < 1/4$ for sigmoid, and hence we recover the necessary condition for the gradients to explode, namely that $\rho_{\mathbf{W}_{rec}} > 1$ (with the tighter version for the sigmoid, $\rho_{\mathbf{W}_{rec}} > 4$).

We also need for equations (11) and (12) to hold, where the first equation implies that our chosen direction $\mathbf{v}$ is not in the null space of $\frac{\partial^+ \mathbf{x}_k}{\partial \theta}$, while the second equation writes the vector $\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}$ in a orthonormal vector basis $\mathbf{v}_1, .., \mathbf{v}_N$, where $\mathbf{v}_1 = \mathbf{v}$ and $c_i^{(loss)} \in \mathbb{R}$.

$$\forall \mathbf{u} \in \mathbb{R}^N, |\mathbf{u}^T \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \mathbf{v}| \geq \gamma_k |\mathbf{u}^T \mathbf{v}|, \gamma_k > 0 \tag{11}$$

$$\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} = \sum_{j=1}^{N} c_j^{(loss)} \mathbf{v}_j^T, c_1^{(loss)} \neq 0 \tag{12}$$

Using these relations we can find a lower bound for $|\mathbf{g}_k^T \mathbf{v}|$, where $\mathbf{g}_k$ is the temporal component corresponding to time step k. Equation (13) shows a few steps of this derivation, where without loss of generality we assigned the first element to P, but not the second one.

$$\begin{aligned}
|\mathbf{g}_k^T \mathbf{v}_1| &\geq \left| \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \right) \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \mathbf{v}_1 \right| \\
&\geq \gamma_k \left| \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{k+1}} \right) \overbrace{\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k}}^{\in P} \mathbf{v}_1 \right| \\
&\geq \gamma_k \beta \left| \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{k+2}} \right) \overbrace{\frac{\partial \mathbf{x}_{k+2}}{\partial \mathbf{x}_{k+1}}}^{\notin P} \mathbf{v}_1 \right| \\
&\geq \gamma_k \beta \alpha \left| \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{k+3}} \right) \frac{\partial \mathbf{x}_{k+3}}{\partial \mathbf{x}_{k+2}} \mathbf{v}_1 \right| \\
&.. \\
&\geq \gamma_k \beta^{|P|} \alpha^{l-|P|} \left| \frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \mathbf{v}_1 \right| \\
&\geq \gamma_k \beta^{|P|} \alpha^{l-|P|} |c_1^{(loss)}| \geq C_k \alpha^{l-|P|}
\end{aligned} \tag{13}$$

Equation (13) ensures that long term components explode along $\mathbf{v}$ as long as the coefficient C_k (where $C_k = \gamma_k \beta^{|P|} |c_1^{(loss)}|$) does not go exponentially fast to 0. This is mostly a constraint on γ_k (since $|P|$ is bounded from our initial assumption), which is determined by $\frac{\partial^+ \mathbf{x}_k}{\partial \theta}$. For a classical parametrization of the model the norm of the partial derivative $\frac{\partial^+ \mathbf{x}_k}{\partial \theta}$ is determined by the norm of the state and input at time k, where the constraint on the state roughly translates into not having the state going towards its saturated state faster than $\alpha^{l-|P|}$ (which can be satisfied for tanh and sigmoid).

We summarize this derivation by saying that what we achieve is to provide a lower bound of the gradient when it explodes, bound that grows exponentially with t. Further more, if we consider $\mathbf{v}$ such that we maximize α, given that $|P|$ is fixed, in the limit of $t \to \infty$ the bound becomes tight and we can approximate the gradient by $C_k \alpha^{t-|P|}\mathbf{v}$.

2.4 The geometrical interpretation

Let us consider a simple one hidden unit model (equation (14)) where we provide an initial state x_0 and train the model to have a specific target value after 50 steps. Note that for simplicity we assume no input.

$$x_t = w\sigma(x_{t-1}) + b \tag{14}$$

Figure 3 shows the error surface $\mathcal{E}_{50} = (\sigma(x_{50}) - 0.7)^2$, where we use the initial state $x_0 = .5$.

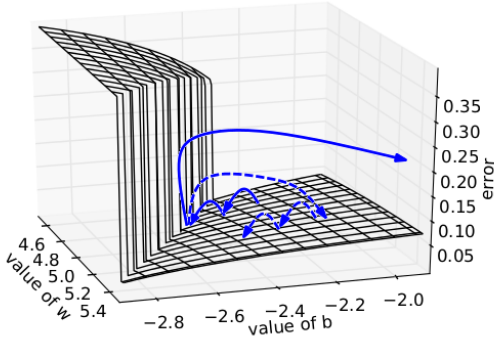

Figure 3: We plot the error surface of a one hidden unit recurrent network, highlighting the existence of high curvature walls. The solid lines depicts standard trajectories that gradient descent might follow. Using dashed arrow the diagram shows what would happen if the gradients is rescaled to a fixed size when its norm is above a threshold.

Section 2.1 shows that when the gradient explodes, it is bounded by $C\alpha^k\mathbf{v}$ (see equation (13)). If we assume that this bound is tight, we can derive that when gradients explode so does the curvature along $\mathbf{v}$, leading to a wall in the error surface as the one seen in 3.

This provides us with a hypothesis, which if it holds, gives us a simple solution to the exploding gradient problem depicted in 3.

If both the gradient and the leading eigenvector of the curvature are aligned with the exploding direction $\mathbf{v}$, it follows that the error surface has a steep wall perpendicular to $\mathbf{v}$ (and consequently to the gradient). This means that when SGD reaches the wall and does a gradient descent step, it will be forced to jump across the valley moving perpendicular to the steep walls, possibly leaving the valley and disrupting the learning process.

The dashed arrows in Figure 3 correspond to *ignoring the norm of this large step*, ensuring that the model stays close to the wall. The key insight is that all the steps taken when the gradient explodes are aligned with $\mathbf{v}$ and ignore other descent direction (i.e. the model moves perpendicular to the wall). At the wall, small-norm step in the direction of the gradient therefore only pushes us back inside the smoother low-curvature region besides the wall. In that region, SGD is free to explore other descent directions.

The important addition in this scenario to the classical high curvature valley, is that we assume that the valley is wide, as we have a large region around the wall where if we land we can rely on first order methods to move towards the local minima. This justifies why just clipping the gradient might be sufficient and we are not necessarily constraint to use a second order method.

Our hypothesis could also help to understand the recent success of the Hessian-Free approach compared to other second order methods. There are two key differences between Hessian-Free and most other second-order algorithms. First, it uses the full Hessian matrix and hence can deal with exploding directions that are not necessarily axis aligned. Second, it computes a new estimate of the Hessian matrix before each update step, which can take into account abrupt changes in curvature (such as the ones suggested by our hypothesis) while most other approaches rely on a smoothness assumption, averaging 2nd order signals over many steps.

2.5 Drawing similarities with Dynamical Systems

One can consider yet another perspective, namely that of dynamical systems. Recurrent networks are universal approximators of dynamical systems (see for e.g. Siegelmann and Sontag (1995)) and as such it is sometimes useful to analyze them using dynamical systems tools which allow for better abstractions that we can use for reasoning about the behaviour of the model. Looking at dynamical systems theory for explaining the *exploding gradient* problem has been done before in Doya (1993); Bengio *et al.* (1993). In this paper we will extend and improve these previous observations.

For any parameter assignment θ, depending on the initial state $\mathbf{x}_0$, the $\mathbf{x}_t$ of an autonomous dynamical system converges, under the repeated application of the map F, to one of several possible different attractor states (e.g. point attractors). They describe the asymptotic behaviour of the model. The state space is divided into basins of attraction, one for each attractor. If the model is started in one basin of attraction it will converge to the corresponding attractor.

The next step is to consider how θ affects the asymptotic behaviour. Dynamical systems theory tells us that as θ changes slowly, the asymptotic behaviour changes smoothly almost everywhere except for certain crucial points where drastic changes occur (the new asymptotic behaviour is no more topologically equivalent with the old one). These points are called bifurcation boundaries and are caused by attractors that appear, disappear or change shape.

Specifically, if we re-consider the simple model defined by equation (14) from section 2.4, where we fix w to 5.0 and allow b to change we get its bifurcation diagram (with respect to b) of Figure 4. Such diagrams convey an abstract but complete picture of how the system can behave.

The x-axis covers the parameter b and the y-axis the asymptotic state $\mathbf{x}_\infty$. The bold line follows the movement of the final point attractor, $\mathbf{x}_\infty$, as b changes. At b_1 we have a bifurcation boundary where a new attractor emerges (when b decreases from ∞), while at b_2 we have another that results in the disappearance of one of the two attractors. In the interval (b_1, b_2) we are in a rich regime, where there are two attractors and the change in position of boundary between them, as we change b, is traced out by a dashed line. The vector field (gray dashed arrows) describe the evolution of the state $\mathbf{x}$ if the network is initialized in that region.

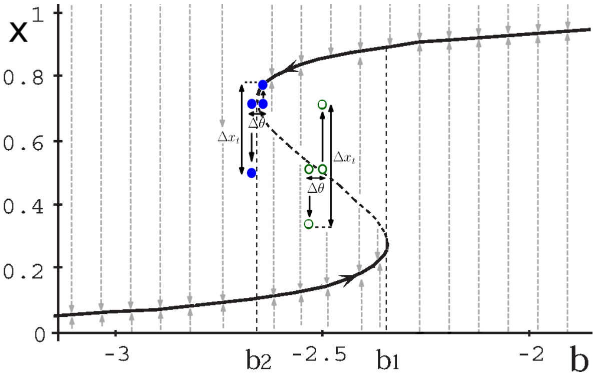

Figure 4: Bifurcation diagram of a single hidden unit RNN (with fixed recurrent weight of 5.0 and adjustable bias). See text.

We show that there are two types of events that could lead to a large change in $\mathbf{x}_t$, with $t \to \infty$. One is crossing a boundary between basin of attraction (depicted with a unfilled circles), while the other is crossing a bifurcation boundary (filled circles). For large t, the Δx_t resulting from a change in b will be large even for very small changes in b (as the system is attracted towards different attractors) which leads to a large gradient. In practice it is sufficient to be close to such a boundary for the gradient to be quite large (as F is smooth and hence it needs to gradually change direction, with the norm $||F'||$ of the Jacobian of F being 0 on the boundary).

Using these notions we can define a necessary and sufficient condition for the gradients to explode. The condition is for a boundary between basins of attraction to be crossed either by a change in $\mathbf{x}_0$ or *a change in* θ. Not that we take the non-conventional approach of considering a change in θ to cause switching between basins of attraction by changing the position of the border. When crossing a bifurcation boundary that leads to large change in $\mathbf{x}_t$, by an abuse of language, we say a boundary between basins of attractions was also crossed implicitly (as for e.g. the boundary between the old attractor and the emerging one), allowing us to unify the two different scenarios.

In Doya (1993) only crossing bifurcation boundaries are considered ignoring changes in the state or the position of the state relative a borders between basin of attraction. We argue that this is an incomplete view. Crossing a bifurcation implies a global change, but locally things could stay the same (i.e., after the bifurcation we can find ourselves in the same basin of attraction). Also a change in θ means a change in the position of the boundary between basins of attractions which could lead to crossing such a boundary, a scenario that is not considered by analyzing only bifurcations. Therefore we consider our proposed view more lucrative.

Another limitation of previous analysis is that they only consider autonomous systems. We propose to extend our analysis to input driven model by folding the input into the map. We consider the family of maps F_t, where we apply a different F_t at each step. Intuitively, for the gradients to explode we require the same behaviour as before, where (at least in some direction) the maps $F_1, .., F_t$ agree and change direction, for a small change in θ or $\mathbf{x}_0$ (even for the same input sequence). Figure 5 describes this behaviour.

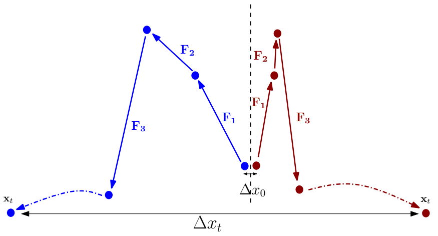

Figure 5: This diagram illustrates how the change in $\mathbf{x}_t$, $\Delta\mathbf{x}_t$, under the successive maps F_t can be large for a small $\Delta\mathbf{x}_0$. The blue vs red (left vs right) trajectories are generated by the same maps $F_1, F_2, ..$ for two different initial states.

For the specific parametrization provided by equation (2) we can take the analogy on step further by decomposing the maps F_t into a fixed map $\tilde{F}$ and a time-varying one U_t. $F(\mathbf{x}) = \mathbf{W}_{rec}\sigma(\mathbf{x}) + \mathbf{b}$ corresponds to an input-less recurrent network, while $U_t(\mathbf{x}) = \mathbf{x} + \mathbf{W}_{in}\mathbf{u}_t$ describes the effect of the input. This is depicted in in Figure 6. Since U_t changes with time, it can not be analyzed using standard dynamical systems tools, but $\tilde{F}$ can. This means that when a boundary between basins of attractions is crossed for $\tilde{F}$, the state will move towards a different attractor, which for large t can lead to a large discrepancy in $\mathbf{x}_t$. If U_t is bounded, that it can interfere with this behaviour only when the state is close to the boundary, but not far away. Therefore studying the asymptotic behaviour of $\tilde{F}$ can provide some information about where such events are likely to happen.

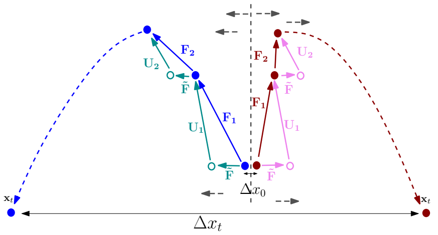

Figure 6: Illustrates how one can break apart the maps $F_1, ..F_t$ into a constant map $\tilde{F}$ and the maps $U_1, .., U_t$. The dotted vertical line represents the boundary between basins of attraction, and the straight dashed arrow the direction of the map $\tilde{F}$ on each side of the boundary. This diagram is an extension of Figure 5.

These derivations (specifically figure 6) provide an intuitive way of understanding equation 13, and the surrounding constraints (the set P for e.g. regards the behaviour of the maps U_i which could move against the exploding direction ensuring that gradients do not explodes). Another interesting connection is with our geometrical interpretation. If there is indeed such a boundary, that means not only that when crossed the norm of the gradient grows considerably, but it can do so quickly, i.e. the curvature has to be high as well. This speaks towards our hypothesis that both gradients and curvature tend to explode together.

3 Dealing with the exploding gradient

3.1 Previous solutions

One approach to avoid exploding gradients is to use L1 or L2 penalty on the recurrent weights. Given that the model is initialized with small numbers, the spectral radius of $\mathbf{W}_{rec}$ is probably smaller than 1, from which it follows that the gradient can not explode (see necessary condition found in section 2.1). The regularization term can ensure that during training the spectral radius never exceeds 1.

This approach limits the model to a simple regime (with a single point attractor at the origin), where any information inserted in the model has to die out exponentially fast in time. In such a regime we can not train a generator network, nor can we exhibit long term memory traces.

This suggest that solutions that exploit changes in the architecture to avoid vanishing gradients, such as LSTMs (Hochreiter and Schmidhuber, 1997) can deal with the exploding gradient by operating the recurrent model in a damping regime and relying exclusively on the highly specialized LSTM units to exhibit memory, thus justifying why the exploding gradient does not seem to be an issue in practice.

Doya (1993) proposes to pre-program the model (to initialize the model in the right regime) or to use *teacher forcing*. The first proposal assumes that if the model exhibits from the beginning the same kind of asymptotic behaviour as the one required by the target, then there is no need to cross a bifurcation boundary. The downside is that one can not always know the required asymptotic behaviour, and, even if such information is known, it is not trivial to initialize a model in this specific regime. We should also note that such initialization does not prevent crossing the boundary between basins of attraction, which, as showed, could happen even though no bifurcation boundary is crossed.

Teacher forcing is a more interesting and not very well understood solution. It can be seen as a way of initializing the model in the right regime and the right region of space. It has been showed that in practice it can reduce the chance that gradients explode, and even allow training generator models or models that work with unbounded amounts of memory(Pascanu and Jaeger, 2011; Doya and Yoshizawa, 1991). One important downside is that it requires a target to be defined at every time step. It also requires expertise, as models trained with it have a tendency to be unstable.

Another approach was proposed by Tomas Mikolov recently described in his PhD thesis (Mikolov, 2012) involves clipping the gradient element-wise if the value exceeds in absolute value a fix threshold. This approach has been showed to do well in practice and it forms the backbone of our approach, which tries to justify this implementation trick.

3.2 Scaling down the gradients

As suggested in section 2.4, one simple mechanism to deal with a sudden increase in the norm of the gradients is to rescale them whenever they go over a threshold (see algorithm 1).

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

```
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if
```

This algorithm is very similar to the one proposed by Tomas Mikolov and the only reason we diverged from his original proposal in an attempt to provide a better theoretical foundation (for e.g. we ensure that we always move in a descent direction), though in practice they behave similarly.

The proposed clipping is simple to implement and computationally efficient, but it does however introduce an additional hyper-parameter, namely the threshold. One good heuristic for setting this threshold is to look at statistics on the average norm over a sufficiently large number of updates. In our experiments we have noticed that for a given task and model size training is not very sensitive to this hyper-parameter and the algorithm behaves well even for rather small thresholds.

The algorithm can also be thought of as adapting the learning rate based on the norm of the gradient.

Table 1: Results on polyphonic music prediction in negative log likelihood per time step. Lower is better.

Data set	Data fold	BPTT	Rescaling gradients
Piano-midi.de	train	**7.06±0.052**	7.12±0.027
	test	7.79±0.013	**7.77±0.019**
Nottingham	train	4.12±0.330	**3.71±0.059**
	test	4.57±0.718	**4.05±0.052**
MuseData	train	7.99±0.768	**7.02±0.031**
	test	8.00±0.426	**7.24±0.037**

Table 2: Results on the next character prediction task in entropy (bits/character)

Data set	Data fold	BPTT	Rescaling gradients
text8	train	**1.66±0.010**	1.67±0.006
	test	1.82±0.067	**1.80±0.021**
Penn Treebank	train	2.25±0.960	**1.41±0.120**
	test	2.24±0.942	**1.44±0.006**

Compared to other learning rate adaptation strategies, which focus on improving convergence by collecting statistics on the gradient (as for example in Duchi *et al.* (2011), or Moreira and Fiesler (1995) for an overview), we rely on the *instantaneous* gradient. This means that we can handle very abrupt changes in norm, while the other methods would not be able to do so. We do not ensure faster convergence though.

4 Experiments and Results

4.1 Polyphonic music prediction

The first task we consider is polyphonic music prediction, using the datasets Piano-midi.de, Nottingham and MuseData described in Boulanger-Lewandowski *et al.* (2012). In this case the vocabulary size is 88 different notes, with the distinction that multiple notes can be played at the same time.

We use a 100 tanh units model with biases in order to stay close to the original setup (Boulanger-Lewandowski *et al.*, 2012). Each song is divided into non-overlapping sequences of 100 steps, and the hidden state is carried over only along the same song (and set to 0 for the first sequence of each song). We use a learning rate of .001 and a threshold of 120. The training and test scores reported in table 1 are average negative log likelihood per time step. We use 5 different runs to estimate these values.

These results are an improvement on the state of the art obtained using RNNs models(see Boulanger-Lewandowski *et al.* (2012)).

4.2 Next character prediction

The second task is next character prediction on three different datasets: Penn Treebank Corpus, Wikipedia ‘text8’. The same datasets had been considered in Mikolov *et al.* (2012).

The model used is a 500 sigmoid units RNN with no biases in order to have a similar setup as the one used in Mikolov *et al.* (2012). Each gradient is computed over non-overlapping sequences of 180 characters, where the hidden state is carried over from one sequence to the next one. Table 2 provides the training and test error for best validation score. These values are computed over 5 different random initializations and we report entropy (bits per character) as a measure of error. We used a threshold of 20 for ‘text8‘ and 45 for Penn Treebank dataset.

These results (together with the one on polyphonic music prediction) suggest that clipping the gradients solves an optimization issue and does not act as a regularizer, as both the training and test error improve in general. Also results on Penn Treebank reach the state of the art (Mikolov *et al.*, 2012), where those results were obtained using a different clipping algorithm similar to ours providing evidence that both behave similarly. For the ‘text8‘ experiment, our model is much smaller than the one used for state of the art results (for computational reasons) and hence the numbers are higher.

4.3 Temporal order task

In our final experiments we consider the synthetic problem proposed as task 6a in Hochreiter and Schmidhuber (1997) (see that paper for full details).

We use a 50 sigmoid units model with a learning rate of .001 and limit the number of training steps to 100k. We use mini-batch gradient descent with 1000 examples per batch. The task is considered solved if for 1000 consecutive inputs the model returns the correct answer. Figure 7 shows the success rate of standard BPTT and gradient rescaling for 50 different runs. Note that for sequences longer than 20 the vanishing gradient problem ensures that neither BPTT nor our algorithm can solve the task.

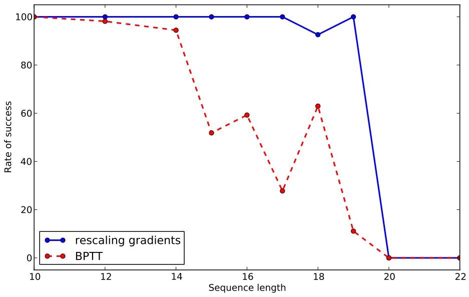

Figure 7: Rate of success for solving the temporal order problem versus sequence length. See text for more details.

This task provides empirical evidence that the exploding gradient is linked with tasks that require long memory traces. We know that initially the model operates in the one-attractor regime (i.e. $\rho_{\mathbf{W}_{rec}} < 1$), in which the amount of memory is controlled by $\rho_{\mathbf{W}_{rec}}$. More memory means larger spectral radius, and, when this value crosses a certain threshold the model enters rich regimes where gradients are likely to explode. We see in Figure 7 that as long as the vanishing gradient problem does not become an issue, addressing the exploding gradient ensures a much larger success rate.

5 Summary and Conclusions

The *exploding gradient* problem is just one facet of the difficulty of training recurrent networks. We provided different perspectives through which one can gain more insight into this issue, though these descriptions can easily be useful for also understanding the *vanishing gradient* problem. We propose a solution that involves clipping the norm of the exploded gradients when it is too large. The algorithm is motivated by the assumption that when gradients explode, the curvature explodes as well, and we are faced with a specific pattern in the error surface, namely a valley with a single steep wall. In practice we show that this approach improves performance on all of the 6 tested tasks.

References

Atiya, A. F. and Parlos, A. G. (2000). New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Trans. Neural Networks*, **11**, 697–709.

Bengio, Y., Frasconi, P., and Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. pages 1183–1195, San Francisco. IEEE Press. (invited paper).

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**(2), 157–166.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the Twenty-nine International Conference on Machine Learning (ICML'12)*. ACM.

Doya, K. (1993). Bifurcations of recurrent neural networks in gradient descent learning. *IEEE Transactions on Neural Networks*, **1**, 75–80.

Doya, K. and Yoshizawa, S. (1991). Adaptive synchronization of neural and physical oscillators. In J. E. Moody, S. J. Hanson, and R. Lippmann, editors, *NIPS*, pages 109–116. Morgan Kaufmann.

Duchi, J. C., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**, 2121–2159.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, **14**(2), 179–211.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.

Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, **3**(3), 127–149.

Mikolov, T. (2012). *Statistical Language Models based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Mikolov, T., Sutskever, I., Deoras, A., Le, H.-S., Kombrink, S., and Cernocky, J. (2012). Subword language modeling with neural networks. preprint (http://www.fit.vutbr.cz/ imikolov/rnnlm/char.pdf).

Moreira, M. and Fiesler, E. (1995). Neural networks with adaptive learning rate and momentum terms. Idiap-RR Idiap-RR-04-1995, IDIAP, Martigny, Switzerland.

Pascanu, R. and Jaeger, H. (2011). A neurodynamical model for working memory. *Neural Netw.*, **24**, 199–207.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**(6088), 533–536.

Siegelmann, H. T. and Sontag, E. D. (1995). On the computational power of neural nets. *JOURNAL OF COMPUTER AND SYSTEM SCIENCES*, **50**(1), 132–150.

Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1017–1024, New York, NY, USA. ACM.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, **1**(4), 339–356.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, **1**, 270–280.