ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN

Môn học: CS115 – Toán cho Khoa học máy tính

BÁO CÁO ĐỒ ÁN

Recurrent Neural Networks (RNNs), and the Exploding and Vanishing Gradient Problems

GVHD: [Lương Ngọc Hoàng]
Nhóm thực hiện: Nhóm 6
Lê Quang Trung
Nguyễn Ngọc Hưng

TP. Hồ Chí Minh, Ngày 8 tháng 1 năm 2026

Mục lục

Tóm tắt nội dung

Báo cáo này trình bày cơ sở toán học của Recurrent Neural Networks, tập trung vào cơ chế lan truyền ngược qua thời gian (BPTT) và các vấn đề bùng nổ (exploding) và triệt tiêu (vanishing) gradient. Tính hiệu quả của mô hình được minh họa qua bài toán dự báo chuỗi thời gian giá cổ phiếu (dữ liệu crawl), trong khi hiện tượng bất ổn của gradient được phân tích và đề xuất cách khắc phục bằng Gradient Clipping và kiến trúc LSTM.

1 Tổng quan về Recurrent Neural Networks

1.1 Dữ liệu chuỗi là gì?

Trong nhiều bài toán học máy, dữ liệu được giả định độc lập và phân phối như nhau (i.i.d). Với dữ liệu chuỗi (chuỗi thời gian, ngôn ngữ tự nhiên, gen), thứ tự chứa thông tin quan trọng. RNN giải quyết bài toán này bằng cách duy trì một trạng thái ẩn $\boldsymbol{h}_t$ đóng vai trò bộ nhớ theo thời gian.

1.2 Kiến trúc mô hình

RNN chia sẻ tham số trọng số qua mọi bước thời gian. Các công thức sau mô tả kiến trúc *many-to-many*, nơi mỗi bước vào sinh một bước ra:

$$\begin{aligned}\boldsymbol{h}_t &= \phi_h(\boldsymbol{x}_t W_{xh} + \boldsymbol{h}_{t-1} W_{hh} + \boldsymbol{b}_h)\,,\\ \boldsymbol{o}_t &= \phi_o(\boldsymbol{h}_t W_{ho} + \boldsymbol{b}_o)\,,\end{aligned}$$

với ϕ_h và ϕ_o là hàm kích hoạt (thường dùng tanh hoặc sigmoid).

2 Huấn luyện RNN

2.1 Thuật toán Lan truyền ngược qua thời gian (BPTT)

Trước hết, xác định hàm mất mát tổng trên chuỗi many-to-many dài T bước, với một *mất mát ở mỗi thời điểm*:

$$\mathcal{L} = \sum_{t=1}^{T} \ell_t(\boldsymbol{y}_t, \boldsymbol{z}_t).$$

Do chia sẻ trọng số, đạo hàm của $\mathcal{L}$ theo một tham số là tổng đóng góp từ mọi bước thời gian. Với RNN many-to-many, ℓ_t được tính tại *mọi* bước, nên gradient tích lũy toàn bộ chuỗi. BPTT dựng đồ thị tính toán theo trục thời gian, áp dụng quy tắc chuỗi để lan truyền gradient từ tương lai về quá khứ.

2.2 Các công thức gradient đã tìm ra

Các công thức đạo hàm chính cho RNN many-to-many:

- **Loss chuỗi**: $\mathcal{L} = \sum_{t=1}^{T} \ell_t(\boldsymbol{y}_t, \boldsymbol{z}_t)$.
- **Sai số tầng hidden**: $\boldsymbol{\delta}_t = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t)\right)$, với $\boldsymbol{a}_t = \boldsymbol{x}_t W_{xh} + \boldsymbol{h}_{t-1} W_{hh} + \boldsymbol{b}_h$.
- **Gradient** W_{xh}: $\dfrac{\partial \mathcal{L}}{\partial W_{xh}} = \sum_{t=1}^{T} \boldsymbol{\delta}_t \boldsymbol{x}_t^\top$.
- **Gradient** W_{hh}: $\dfrac{\partial \mathcal{L}}{\partial W_{hh}} = \sum_{t=1}^{T} \boldsymbol{\delta}_t \boldsymbol{h}_{t-1}^\top$ (*nguồn gốc vanishing/exploding do tích lũy thời gian*).
- **Gradient** W_{ho}: $\dfrac{\partial \mathcal{L}}{\partial W_{ho}} = \sum_{t=1}^{T} \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t} \boldsymbol{h}_t^\top$.

3 Thực nghiệm 1: Dự báo giá chứng khoán

Bài toán: Many-to-One RNN dự báo giá dựa trên chuỗi quá khứ.

- **Dữ liệu**: Giá đóng cửa theo ngày crawl từ Stooq, lưu CSV và tách train/test theo thời gian với tỉ lệ 80/20.
- **Tiền xử lý**: Chuẩn hóa Min–Max fit trên tập train, tạo cửa sổ trượt lookback=20. Mỗi mẫu $X \in \mathbb{R}^{20\times 1}$, nhãn y là giá ở bước kế tiếp (one-step ahead).
- **Mô hình**: RNN thuần 1 lớp, hidden size 16, tanh; output tuyến tính; loss $0.5(o_T - y)^2$ chỉ trên bước cuối; BPTT thủ công.
- **Huấn luyện**: Gradient Descent theo kiểu full-batch (gộp trung bình gradient toàn bộ chuỗi), learning rate 0.01, 300 epoch, áp dụng Gradient Clipping với ngưỡng $\tau = 5$.
- **Đánh giá**: Dự báo trên test, đảo chuẩn hóa bằng scaler của train và tính RMSE/MAE.

Chỉ số	Giá trị
RMSE (test)	≈ 136.39
MAE (test)	≈ 135.67

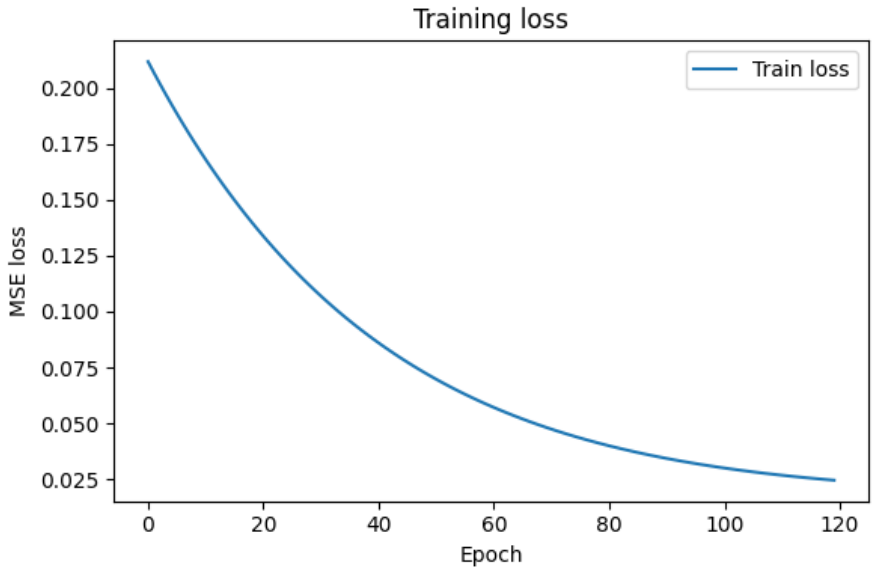
Bảng 1: Hiệu năng dự báo trên tập kiểm tra.

Kết luận:

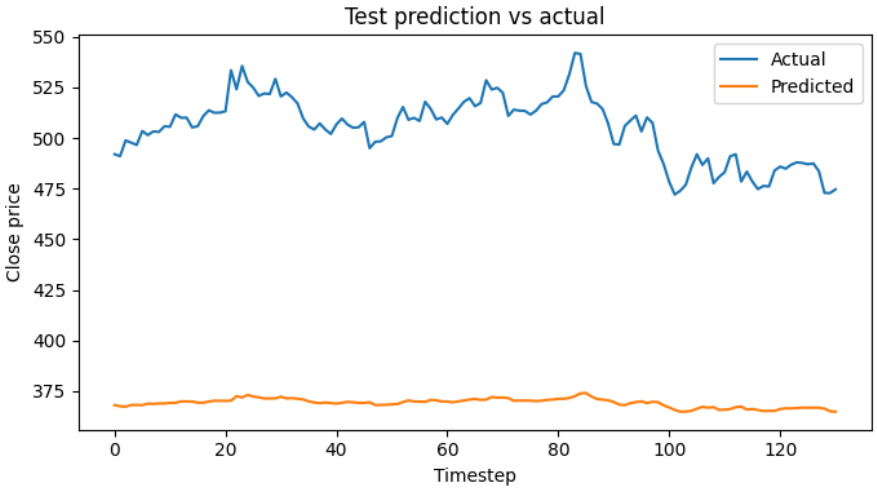
- Kết quả RMSE ≈ 136.39 và MAE ≈ 135.67 vẫn còn lớn so với biên độ giá trong demo (khoảng 220–540), cho thấy sai số trung bình ở mức hàng trăm đơn vị nên giá trị dự báo thực tế còn hạn chế. Đường dự báo thường bám quanh mức trung bình và bị trễ so với biến động thực; clipping chủ yếu giúp tránh exploding gradient chứ chưa cải thiện đáng kể năng lực mô hình.

Lý do chính:

- **Kiến trúc quá cơ bản**: RNN thuần không có cơ chế cổng (LSTM/GRU), nên khả năng giữ thông tin dài hạn yếu và nhạy với vanishing/exploding.
- **Dung lượng mô hình nhỏ**: Chỉ 1 lớp với hidden 16 khiến mô hình dễ underfit trước động lực giá phức tạp.
- **Tín hiệu học tập mỏng**: Loss chỉ lấy ở bước cuối làm gradient học được từ chuỗi bị hạn chế, khó khai thác cấu trúc theo thời gian.
- **Tối ưu hóa đơn giản**: Full-batch GD với learning rate cố định và chỉ dùng clipping giúp ổn định nhưng không tăng năng lực dự báo.
- **Đầu vào tối giản**: Chỉ dùng chuỗi Close và dự báo giá tuyệt đối làm bài toán khó hơn, mô hình nhỏ dễ học quanh mức trung bình.



Hình 1: Quá trình huấn luyện với Gradient Clipping ($\tau = 5$).



Hình 2: Dự báo giá chứng khoán (dữ liệu crawl từ Stooq).

4 Vấn đề Gradient và Thiết kế Thực nghiệm

4.1 Phân tích nguyên nhân Vanishing/Exploding Gradient

Nguồn gốc toán học. Khi lan truyền ngược BPTT từ bước k về t, gradient đi qua tích của các Jacobian:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} = \sum_{k=t}^{T} \left(\prod_{i=t+1}^{k} \underbrace{\operatorname{diag}\left(\phi_h'(\boldsymbol{a}_i)\right) W_{hh}^{\top}}_{J_i} \right) \frac{\partial \mathcal{L}_k}{\partial \boldsymbol{h}_k}.$$

Do $\|\operatorname{diag}(\phi_h')\| \le \gamma \le 1$ (tanh/sigmoid bão hòa), ta có

$$\left\| \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \right\| \le \sum_{k=t}^{T} \gamma^{k-t} \left\| W_{hh} \right\|^{k-t} \left\| \frac{\partial \mathcal{L}_k}{\partial \boldsymbol{h}_k} \right\|.$$

Như vậy độ lớn gradient phụ thuộc lũy thừa của $\gamma\|W_{hh}\|$ (xấp xỉ bán kính phổ của W_{hh} khi $\gamma \approx 1$):

- $\gamma\|W_{hh}\| < 1 \;\Rightarrow$ tích Jacobian suy giảm theo hàm mũ $\rightarrow$ vanishing.
- $\gamma\|W_{hh}\| > 1 \;\Rightarrow$ tích Jacobian phình to theo hàm mũ $\rightarrow$ exploding.
- Kích hoạt bão hòa (tanh/sigmoid) làm $\gamma < 1$ càng kéo gradient về 0 khi chuỗi dài.

Ngoài ra, khởi tạo tham số xấu hoặc chuỗi quá dài khiến tích ma trận lặp lại nhiều lần, càng khuếch đại hai hiệu ứng trên.

4.2 Các giải pháp đề xuất

- **Gradient Clipping**: Cắt chuẩn gradient khi vượt ngưỡng τ, giảm nguy cơ exploding.
- **LSTM/GRU**: Thay đổi kiến trúc với các cổng để luồng gradient truyền tuyến tính hơn, khắc phục vanishing.

5 Thực nghiệm 2: Khảo sát sự ổn định của Gradient

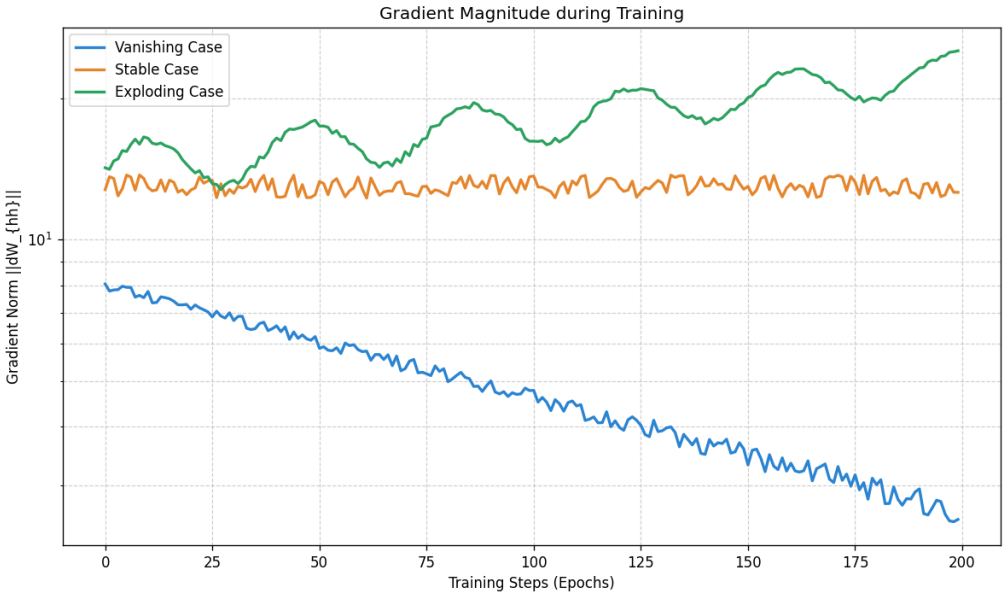
Mục tiêu: Chứng minh thực nghiệm nhận định lý thuyết về vanishing/exploding bằng cách điều khiển bán kính phổ của W_{hh}.

Thiết lập:

- **Dữ liệu**: Chuỗi ký tự "the quick brown fox jumps over the lazy dog" nhân 200 lần, mã hóa one-hot.
- **Mô hình**: RNN 1 lớp, hidden size 50, ReLU; khởi tạo $\boldsymbol{W}_{hh}$ theo $\mathcal{N}(0, \frac{1}{\sqrt{h}})$ rồi nhân hệ số ρ để thay đổi bán kính phổ.
- **Huấn luyện**: Chuỗi cắt thành đoạn dài 50, 200 bước update, learning rate 10^{-4}; theo dõi $\|\nabla W_{hh}\|_2$ mỗi bước (log-scale).
- **Kịch bản**: $\rho = 0.90$ (vanishing), $\rho = 0.99$ (gần ổn định), $\rho = 1.20$ (exploding).

Kết quả:

- *Vanishing* ($\rho = 0.90$): Chuẩn gradient giảm dần dưới 1 và tiếp tục suy giảm khi chuỗi dài hơn, phù hợp tích Jacobian suy giảm.
- *Gần ổn định* ($\rho = 0.99$): Chuẩn gradient dao động quanh 10^1, không tăng/giảm đáng kể sau 200 bước.
- *Exploding* ($\rho = 1.20$): Chuẩn gradient tăng dần, vượt mốc $\approx 2 \times 10^1$ về cuối, biểu hiện bùng nổ theo lũy thừa.



Hình 3: Độ lớn $\|\nabla W_{hh}\|$ theo bước train ở ba kịch bản ρ.

6 Kết luận

Chúng tôi đã trình bày cơ sở toán học của RNN và BPTT, minh họa bằng bài toán dự báo giá cổ phiếu (Thực nghiệm 1) và kiểm chứng hiện tượng vanishing/exploding qua việc điều khiển bán kính phổ của W_{hh} (Thực nghiệm 2). Kết quả cho thấy RNN thuần dễ underfit và nhạy với bất ổn gradient; gradient clipping chỉ giúp ổn định chứ không tăng năng lực biểu diễn, trong khi lựa chọn kiến trúc và khởi tạo là then chốt quyết định việc gradient suy giảm hay bùng nổ. Về thực hành, cần chú trọng khởi tạo/kiến trúc phù hợp và theo dõi chuẩn gradient trong huấn luyện, đặc biệt khi chuỗi dài, để tránh mất ổn định.

Tài liệu tham khảo

- I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.
- R. Pascanu, T. Mikolov, Y. Bengio, "On the Difficulty of Training Recurrent Neural Networks," ICML, 2013.
- S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory," Neural Computation, 1997.