Machine Learning with R

Implementing Neural Networks using the Torch Library

Presenter:

Ezhilarasi Muthukumar

Introduction to PyTorch/Torch

What is Torch?

Torch for R is the R interface to PyTorch, bringing deep learning capabilities to the R ecosystem.

Brief History

PyTorch: Developed by Facebook's AI Research lab (2016)
Torch for R: Released in 2020, bringing PyTorch's power to R programmers
Built on the same C++ backend as PyTorch

Why Torch is Popular

Feature	Description
Dynamic Computation Graphs	Build models that change on-the-fly during runtime
Pythonic/R-native Interface	Intuitive syntax that feels natural to R users
Strong GPU Support	Seamless acceleration with CUDA
Active Community	Extensive documentation and community support
Research-Friendly	Flexibility for experimentation and prototyping

Key Features for R Users

- Familiar R syntax with %>% pipe operators
- Integration with tidyverse ecosystem (dplyr, ggplot2)
- Automatic differentiation for gradient computation
- Comprehensive neural network modules (nn_module)
- Multiple optimization algorithms (Adam, SGD, RMSprop)

Overview of the Project

Project Objectives

Primary Goal:

Demonstrate that R programming is a viable platform for implementing Machine Learning models, specifically Neural Networks, with comparable capabilities to Python.

Secondary Goals:

1. Compare R and Python approaches to deep learning
2. Explore different neural network architectures.
3. Implement hyperparameter optimization (Grid Search).
4. Evaluate model performance with multiple metrics

Note on Customer Churn:

This project produced lower accuracy but successfully demonstrates ML feasibility in R.
Further optimization would require significant computational resources and is recommended as future work for interested researchers.

Two Case Studies

Project	Dataset	Problem Type	Primary Goal
Customer Churn	Customer behavior data	Binary Classification	Proof of concept
Wheat Seed Classification	Wheat kernel measurements	Multi-class Classification	Optimal model discovery

Dataset Details

Customer Churn Dataset

Source & Description

Domain: Telecommunications/Subscription Service
Objective: Predict whether a customer will churn (leave the service)
Features: Customer demographics, usage patterns, service history

Dataset Characteristics

Attribute	Training Set	Testing Set
Samples	~7,000 rows	~3,000 rows
Features	11 columns	11 columns
Target	Churn (0/1)	Churn (0/1)

Key Features

- **Numeric:** Age, Tenure, Usage Frequency, Support Calls, Payment Delay, Total Spend, Last Interaction
- **Categorical:** Gender, Subscription Type, Contract Length
- **Target:** Churn (Binary: 0 = Stay, 1 = Leave)

Preprocessing Steps

1. Remove identifier columns (CustomerID)
2. Handle missing values (na.omit)
3. Encode categorical variables (factor to integer)
4. Normalize features (z-score standardization)
5. Convert to torch tensors

Dataset Details (Continued)

Wheat Seeds Dataset

Source & Description

Domain: Agricultural Science / Grain Quality Assessment
Objective: Classify wheat kernels into 3 varieties
Features: Geometric measurements of wheat kernels

Dataset Characteristics

Attribute	Value
Total Samples	210 observations
Training Set	168 samples (80%)
Testing Set	42 samples (20%)
Features	7 measurements
Classes	3 wheat varieties

Features (Kernel Measurements)

1. AREA - Kernel area
2. PERIMETER - Kernel perimeter
3. COMPACTNESS - Compactness measure
4. LENGTH - Kernel length
5. WIDTH - Kernel width
6. ASYMMETRY_COEFFICIENT - Asymmetry coefficient
7. GROOVE_LENGTH - Length of kernel groove

Preprocessing

1. 80-20 train-test split (with seed for reproducibility)
2. Matrix conversion for torch compatibility
3. Tensor conversion (float for features, long for labels)

Key Insights

R Torch Advantages:

- Seamless integration with existing R data science workflows.
- Superior statistical analysis and visualization capabilities
- Familiar syntax for R programmers
- Great for research and exploratory analysis

Python PyTorch Advantages:

- Larger community and more learning resources.
- Better production deployment options.
- More pre-trained models and libraries.
- Industry standard for deep learning

Conclusion:

Both are powerful. Choose based on your existing ecosystem and team expertise!

Grid Search

Hyperparameter Tuning

What is Grid Search?

Grid Search is an exhaustive search over specified parameter combinations to find the optimal model configuration.

Parameters Tested:

Parameter	Options	Count
Model Architecture	Simple, Medium, Complex	3
Optimizer	Adam, SGD, RMSprop	3
Learning Rate	0.1, 0.02, 0.0001	3
Total Combinations	3 × 3 × 3	**27**

Performance Comparison(Top 10)

Rank	Model	Optimizer	Learning Rate	Accuracy
1	Medium	Adam	0.02	95.2%
2	Complex	Adam	0.02	94.8%
3	Medium	RMSprop	0.02	94.1%
4	Complex	RMSprop	0.02	93.7%
5	Medium	SGD	0.02	92.9%
6	Simple	Adam	0.02	91.5%
7	Complex	Adam	0.0001	90.8%
8	Medium	Adam	0.0001	89.4%
9	Simple	RMSprop	0.02	88.6%
10	Complex	SGD	0.02	87.3%

Visualization of Results

Loss and Accuracy Curves

Interpretation:

- Rapid initial decrease (epochs 1-20)
- Gradual convergence (epochs 20-60)
- Stable plateau (epochs 60-100)

Accuracy Progression

Interpretation:

- Quick learning in early epochs
- Accuracy stabilizes around epoch 40
- Final accuracy: 95.2%

Confusion Matrix Heat

Interpretation:

- Strong diagonal (correct predictions)
- Minimal off-diagonal values (few errors)
- Balanced performance across all 3 classes

Insights & Interpretation

For Wheat Seeds Classification

Technical Achievement:

- **95.2% accuracy** demonstrates R's capability for ML tasks
- **Successful multi-class classification** with minimal data (210 samples)
- **Efficient training** converges in <100 epochs

Real-World Impact:

- Grain processing plants can scan kernels automatically.
- Reduces human error in variety identification
- Enables real-time quality assurance
- Scalable to millions of kernel classifications per day

For Customer Churn (Proof of Concept)

Technical Achievement:

- Demonstrated feasibility of neural networks in R
- Complete pipeline:
  - data preprocessing → model training → evaluation
- Foundation for further optimization

Important Note:

- Current accuracy (68%) is insufficient for production use
- This serves as a **proof of concept** that R can handle ML workflows.
- Production models would require:
- More data
- Feature engineering
- Hyperparameter tuning
- Ensemble methods
- GPU resources for extensive training

Conclusion

R Torch is Ready for Production Use When:

- Your team is already proficient in R
- You need tight integration with statistical analysis
- Visualization quality is critical for stakeholder communication
- You're building internal tools and analytics dashboards
- Research and experimentation are primary goals

Consider Python PyTorch When:

- You need extensive pre-trained models
- Production deployment at scale is immediate priority
- Your team is already in Python ecosystem
- You require maximum community support and resources

Key Takeaway

Machine Learning in R is not just feasible, it's excellent for the right use cases.

The Power of Choice in Machine Learning

"The best tool is the one you know how to use effectively."

Final words:

- This presentation demonstrates that R programmers don't need to switch to Python to build powerful neural networks.
- The Torch library brings world-class deep learning to the R ecosystem while preserving R's strengths in statistics and visualization.

Thank You

Appendix: References

1.Torch for R Documentation: https://torch.mlverse.org/
2.PyTorch Official Documentation: https://pytorch.org/
3.UCI Machine Learning Repository: Wheat Seeds Dataset
4.Deep Learning with R (Book): François Chollet & J.J. Allaire
5.R for Data Science: Hadley Wickham & Garrett Grolemund