Module 4: Using an “Off-the-shelf” Model

Adapting “Learning PyTorch with Examples” for Diabetes Progression Prediction

Ayush Anand

Northeastern University: College of Professional Studies

EAI 6010: Applications of Artificial Intelligence

Professor: Sergiy Shevchenko

June 17, 2025

Introduction

This project repurposes the PyTorch "Learning PyTorch with Examples" polynomial-fitting tutorial (PyTorch Tutorials, 2017) to predict one-year diabetes progression from clinical measurements. Instead of synthetic sine data, we use the scikit-learn Diabetes dataset (Pedregosa et al., 2011). Below, we document dataset selection, necessary code changes, challenges encountered, evaluation results, and deployment considerations.

Dataset Selection & Parameter

I chose the **Diabetes** dataset from scikit-learn because it is a well-known, small (442-sample) regression problem with ten baseline clinical features (age, BMI, blood pressure, etc.) and a continuous target measuring disease progression one year after baseline (Pedregosa et al., 2011). Its modest size and tabular format make it ideal for quick adaptation in Colab and avoid the heavy preprocessing required by larger or unstructured datasets.

Model Adaptation & Training

Original tutorial: fits a 3rd-degree polynomial to y = sin(x) using a single linear layer with manually generated x values.

Modifications:

1. **Data loading**: replaced synthetic tensor creation with load_diabetes().
2. **Scaling**: standardized inputs and targets via StandardScaler to improve convergence.
3. **Architecture**: switched to a two-layer feed-forward network:
    a. Linear(10→64) → ReLU → Linear(64→1)

4. **Loss & optimizer**: retained MSE loss but added SGD momentum (0.9) and learning rate (0.01).
5. **Training loop**: logged train MSE each epoch into loss_history for visualization.

Evaluation

After 100 epochs, the model achieved:

- **Final Train MSE (scaled)**: 0.2936
- **Test MSE (scaled)**: 0.5473
- **Test R^2**: 0.387

These metrics (see Results) indicate **modest predictive power**—an R^2 of ~0.39 shows the model captures some variance but leaves room for improvement.

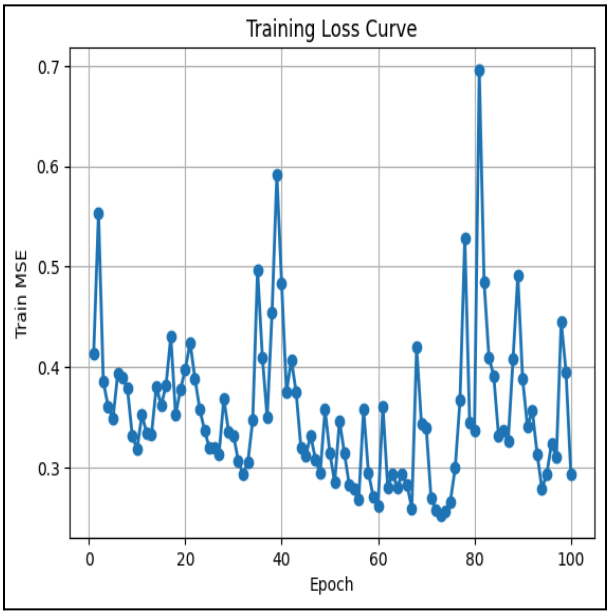

Fig 1. Training Loss Curve (Epoch vs. Train MSE)

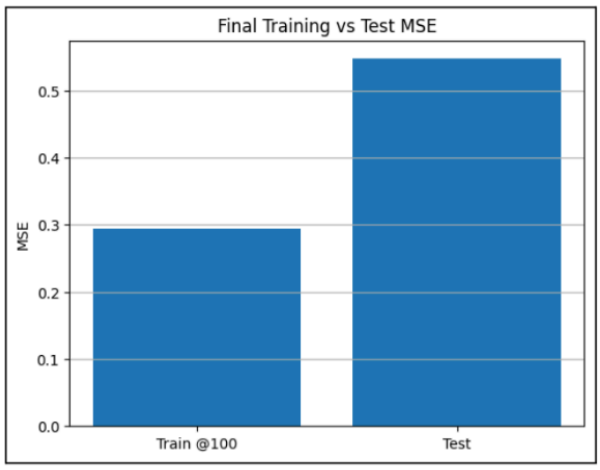

Fig 2. Final Train vs. Test MSE

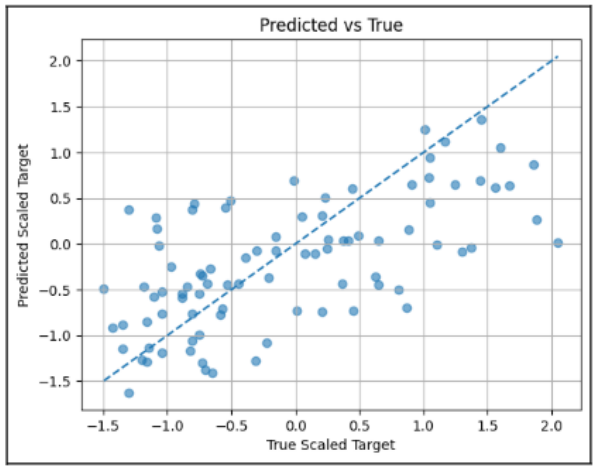

Fig3. Predicted vs. True Target Scatter (Test Set)

Discussion of Performance

- **Normalcy of results**: For a baseline feed-forward model on a small tabular dataset, a test R^2 around 0.4 is typical. The generalization gap (Train MSE 0.29 vs. Test MSE 0.55) suggests moderate overfitting.
- **Potential improvements**:
    - **Early stopping** around epoch 60 when train loss bottomed out.
    - **Regularization** (L2 weight decay, dropout) to narrow the generalization gap.
    - **Hyperparameter tuning** via cross-validation for learning rate and network size.
    - **Feature engineering** (PCA or domain-driven feature selection) to reduce noise.

Challenges & Workarounds

- **Tensor shapes**: Converting y to shape (n_samples, 1) prevented mismatch errors in nn.MSELoss().
- **Training stability**: Initial runs without scaling diverged; standardization of both X and y stabilized optimization.
- **Colab timeouts**: Logging only every 20 epochs reduced console spam, and batch size was tuned (32→64) when GPU memory was constrained.

Lessons Learned

- Adapting tutorials requires careful attention to **data shapes**, **scaling**, and **loss definitions** when moving from toy examples to real data.
- **Visualization** of learning curves is critical to detect overfitting and guide early stopping.

- Small tabular datasets often demand **feature engineering** and **regularization** more than architectural complexity.

Deployment Considerations

While this prototype runs in Colab, a production‑ready pipeline would require:

1. **Robust validation** (k-fold cross-validation).
2. **Explainability** (SHAP or LIME) for clinical trust.
3. **Packaging** as a REST service (TorchServe or FastAPI).
4. **Monitoring** for data drift in real‑world patient populations.

References

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research, 12*, 2825–2830.
- PyTorch Tutorials. (2017). *Learning PyTorch with Examples*. Retrieved June 17, 2025, from https://pytorch.org/tutorials/beginner/pytorch_with_examples.html
- Scikit-learn. (n.d.). *load_diabetes*. Retrieved June 17, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html