Recap: Neural Networks II — Making Training Work

DS701 Session 20 — Wed Nov 11, 2026

Today’s plan

Knowledge-Check Review

KC 1: The knobs

The lecture maps the ideas from Neural Networks I onto `MLPClassifier` arguments. Name the scikit-learn argument that sets each of the following, and say in one phrase what a larger value does: (a) the number of layers and neurons per layer, (b) the learning rate, (c) the mini-batch size, (d) the maximum number of epochs, (e) the strength of L2 regularization. Which `solver` does the lecture recommend as the default?

KC 2: What the loss curve does — and does not — tell you

After training the MNIST model, the lecture plots `mlp.loss_curve_` *and remarks that "the loss decreases smoothly — our model is learning." What exactly is being plotted (which data, which quantity), and why does a smoothly decreasing curve of that kind* not *by itself guarantee good performance on the test set? Which two* `MLPRegressor` *arguments in the California-housing example address that concern, and how?*

KC 3: “Because neural networks”

A colleague has 8,000 rows of tabular customer data with 25 numeric features and wants to build the classifier in PyTorch “because neural networks.” Using the lecture’s scikit-learn-vs-PyTorch/TensorFlow criteria, what would you recommend and why? Name two things about the problem that would change your recommendation, and one preprocessing step you would insist on either way.

Highlights

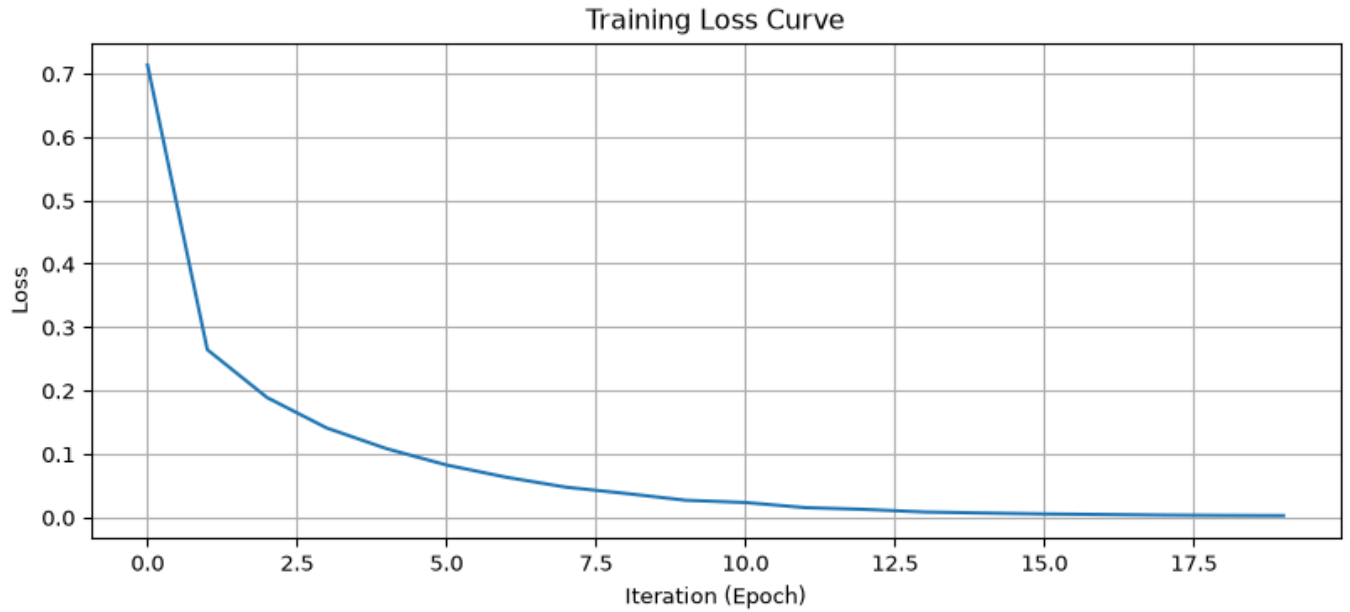
Every idea from last time is now an argument

Concept (NN I)	scikit-learn
layers and neurons	`hidden_layer_sizes`
activation f	`activation`
learning rate η	`learning_rate_init`
mini-batch size	`batch_size`
optimizer	`solver` — `adam` / `sgd` / `lbfgs`
epochs / stopping	`max_iter`, `early_stopping`, `n_iter_no_change`

Standardize inputs. Always.

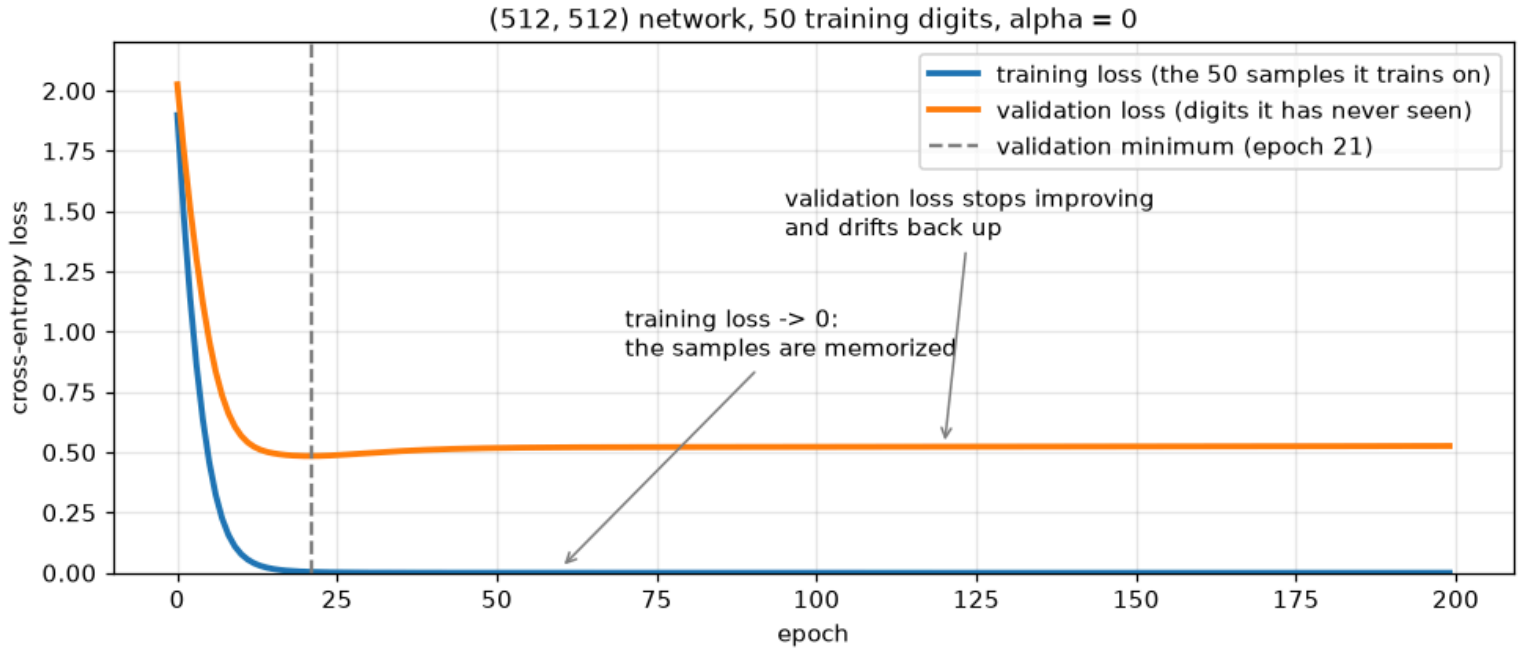
Gradient descent takes one step size for *every* direction. If one feature spans 0–4000 and another 0–0.2, that feature dominates every weighted sum and every gradient, the loss surface is a long narrow valley, and no single learning rate suits both.

Read the loss curve before you read the accuracy



One point = training loss after one epoch. **Healthy**: fast initial drop, smooth flattening. That is what “the model is learning” looks like.

Overfitting: what the loss curve cannot show you



The three repairs — and what each one changes

repair	scikit-learn	what it changes
L2 regularization	`alpha=0.01` (default `1e-4`)	the **objective**: penalizes large weights
early stopping	`early_stopping=True, validation_fraction=0.1, n_iter_no_change=10`	the **training loop**: stop on validation evidence, not training loss
cross-validation	`cross_val_score(mlp, X, y, cv=5)`	the **evaluation**: an honest estimate, so you *notice* — a measurement, not a repair

Tuning honestly

When the toy tool is enough

scikit-learn `MLPClassifier` / `MLPRegressor`

PyTorch / TensorFlow

In-Class Activity

Activity: train it properly, then break it and fix it

Goal: run the lecture's recipe on a real dataset — standardize, fit, read the loss curve, evaluate, compare solvers and learning rates on a fixed budget — then **overfit a huge network on purpose**, watch training and validation loss diverge, repair it three ways (`alpha`, `early_stopping`, more data) and measure which lever actually helped, and finally diagnose three mystery loss curves.

- Work in groups of 2–3. Open **your section's** notebook.
- Parts marked **(autograded)** are submitted to Gradescope; the rest is participation.
- Staff will circulate — be ready to explain any part of your work.
- Dependencies: `numpy`, `matplotlib`, `scikit-learn`. Every fit takes well under a second.

Section A1 

Section B1 

Going deeper

Full lecture notes: Neural Networks II: Making Training Work

- Today: The Knobs — the concept-to-argument table
- Prepare the Data and Data Preprocessing — scaling, and the `Pipeline` pattern
- Training Loss Curve — the MNIST curve; Common Issues — convergence warnings and poor performance checklists
- Prepare Data and Define Model — early stopping in the housing regressor
- Preventing Overfitting — `alpha`, early stopping, cross-validation
- Grid Search for Optimal Architecture and Solver Selection
- Scikit-Learn vs PyTorch/TensorFlow
- Last session's theory: Neural Networks I: How Learning Works; deeper dives: Gradient Descent, Backpropagation, SGD, batches and CNNs, RNNs, NNs with Scikit-Learn