ML4P: Problem Set #0

Getting Your Machine Learning Toolkit Ready

Spring 2026

Welcome!

Hey everyone! Before getting more into the course, let's make sure we all have the tools we need installed and working. This worksheet is **completely optional**. It's just here to help you get everything set up on your own computer so you're ready to hit the ground running when we start the real problem sets. Also, don't worry if you run into problems along the way. Setting up a development environment is tricky and sometimes frustrating. Feel free to ask question in Discord or in office hours.

1 Why These Tools?

To do machine learning you'll need access to a scientific computing backend. We won't build one ourselves, but we'll make use of some open-source libraries.

For demos in class, we'll be using **Python** with the **PyTorch** library. Why PyTorch?

- It was created by researchers at NYU.
- It's the go-to library for machine learning research; almost every ML paper you read will use it.
- It's beginner-friendly while still being powerful enough for cutting-edge research.

You might also hear about **JAX**, another popular library. It's fantastic for scientific computing, but it has a steeper learning curve, so we'll stick with PyTorch for this course.

2 Step 1: Version Control with Git

You'll be submitting your work as code repositories, so let's get version control set up first.

What is Git? It's a tool that keeps track of all the changes you make to your code. Think of it like "undo history" for your entire project, but way more powerful. It also lets you sync your work to the cloud so you never lose it.

To Do:

1. Install `git`: Installation instructions
2. Create a free account on GitHub if you don't have one already
3. Create a repository for this problem set.

3 Step 2: Managing Python Environments with Conda

Here's a common problem: you're working on Project A which needs `numpy` version 2.0, but Project B breaks if you use anything newer than version 1.26. How do you work on both? The answer is **virtual environments**. These are isolated spaces where each project can have its own set of packages without interfering with each other.

To do:

1. Install Miniconda: Installation instructions
2. Create a new environment for this course:

```
conda create -n ml4p python=3.11
conda activate ml4p
```

Tip: Always activate your environment before installing packages or running code for this course!

4 Step 3: Installing PyTorch

Let's install PyTorch.

To do:

1. Head to the PyTorch installation page
2. Select your operating system and package manager (pip or conda).
3. Run the installation command they give you

Quick sanity check: Make sure packages are installing in the right place! Run:

`which pip`

You should see something like:

`/Users/YOUR_NAME/miniconda3/envs/ml4p/bin/pip`

If it points to a different location, you might need to activate your conda environment first. You can also access pip via your miniconda's python

```
python -m pip
```

5 Step 4: Test It Out!

Let's make sure everything is working. Create a file called `test_setup.py` and paste this in:

```
import torch
print(f"PyTorch version: {torch.__version__}")
print(f"CUDA available: {torch.cuda.is_available()}")
print("You're all set! Nice work!")
```

Run it by writing `python test_setup.py` in your terminal w/ the conda environment activated. If you see the version number and a friendly message, you're good to go!

6 Automatic Differentiation

In machine learning, we take a lot of derivatives. Doing this by hand would be tedious and error-prone. Luckily, PyTorch can compute derivatives automatically using a technique called **automatic differentiation** (autodiff).

For example, consider the function:

$$f(x) = \cos^2(7x) \cdot e^{-x} \tag{1}$$

The derivative is (if you work it out by hand):

$$\frac{d}{dx} f(x) = -\left(14 \cos(7x) \sin(7x) + \cos^2(7x)\right) e^{-x} \tag{2}$$

Instead of implementing the derivative manually or using a numerical approximation like:

$$\frac{d}{dx} f(x) \approx \frac{f(x+h) - f(x)}{h}$$

. . . you can use autodiff to get the exact derivative with just a few lines of code!

Your Challenge (Optional)

Try implementing this function (or a function of your choice) in PyTorch, use autodiff to compute its derivative, and compare with the analytical formula above. Plot both to convince yourself they match!

Helpful resources:

- PyTorch Autograd Tutorial—a great place to start
- How Autodiff Works Under the Hood—only if you're curious about the magic behind the scenes