Conda Environment Management

Chapter 2.2: Setting Up Your ML Environment

What is Conda?

Conda is a package and environment manager that simplifies Python development

Key Benefits:

- **Package Management:** Install, update, and remove packages easily
- **Environment Isolation:** Different projects, different dependencies
- **Cross-Platform:** Works on Windows, macOS, and Linux
- **Language Agnostic:** Not just Python (R, Scala, etc.)

Two Main Distributions:

- **Anaconda:** Full distribution (~3GB) with 250+ packages
- **Miniconda:** Minimal installer (~400MB) - **Recommended for beginners**

Why Use Conda for Machine Learning?

The Problem Without Conda:

```
# This can break your system Python!
sudo pip install tensorflow  # ❌ DON'T DO THIS
```

The Solution With Conda:

```
# Safe, isolated environment
conda create -n ml-project python=3.10
conda activate ml-project
conda install tensorflow  # ✅ Safe and clean
```

Why Use Conda for Machine Learning?

Benefits:

- No conflicts between projects
- Easy to share environments
- Simple dependency management
- Reproducible research

Installing Miniconda

Step 1: Download Miniconda

macOS (Intel Macs):

```
curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
```

macOS (Apple Silicon M1/M2/M3):

```
curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh
```

Linux:

```
curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

Windows: Download from https://docs.conda.io/en/latest/miniconda.html

Installing Miniconda

Step 2: Install and Configure

macOS/Linux:

```
# Make installer executable and run
chmod +x Miniconda3-latest-*.sh
bash Miniconda3-latest-*.sh

# Follow the prompts:
# - Accept license agreement
# - Choose installation location (default is fine)
# - Say "yes" to initialize conda
```

After Installation:

```
# Restart terminal or run:
source ~/.bashrc  # Linux
source ~/.zshrc   # macOS with zsh
```

First Steps with Conda

Verify Your Installation

```
# Check conda version
conda --version

# Get detailed information
conda info

# List installed packages
conda list

# Check which environment you're in
conda info --envs
```

Expected Output:

```
conda 23.7.4
# conda environments:
```

Essential Conda Commands

Package Management

```
# Search for packages
conda search numpy
conda search "scikit*"

# Install packages
conda install numpy
conda install numpy scipy matplotlib

# Install specific versions
conda install numpy=1.21.0
conda install "numpy>=1.20,<1.25"

# Update packages
conda update numpy
conda update --all

# Remove packages
```

Working with Environments

Why Use Virtual Environments?

The Problem:

- Project A needs TensorFlow 2.8
- Project B needs TensorFlow 2.12
- Your system can only have one version installed

The Solution: Virtual Environments

- Each project gets its own isolated Python installation
- Different package versions per project
- No conflicts, clean development

Creating and Managing Environments

Basic Environment Operations

```
# Create new environment
conda create -n myproject python=3.10

# Create with specific packages
conda create -n ml-course python=3.10 numpy scipy matplotlib

# Activate environment
conda activate ml-course

# Deactivate environment
conda deactivate

# List all environments
conda env list
conda info --envs

# Remove environment
```

Example: ML Course Environment

Setting Up a Complete ML Environment

```
# Create environment for this course
conda create -n ml-course python=3.10 \
    numpy scipy matplotlib pandas \
    scikit-learn jupyter notebook \
    seaborn plotly

# Activate the environment
conda activate ml-course

# Verify installation
python -c "import numpy; print('NumPy version:', numpy.__version__)"
python -c "import sklearn; print('Scikit-learn version:', sklearn.__version__)"

# Start Jupyter notebook
jupyter notebook
```

Environment Files: Sharing Your Setup

Creating an Environment File

```
# Export current environment
conda env export > environment.yml

# Create from environment file
conda env create -f environment.yml
```

Environment Files: Sharing Your Setup

Sample environment.yml

```yaml
name: ml-course
channels:
  - conda-forge
  - defaults
dependencies:
  - python=3.10
  - numpy>=1.21.0
  - scipy>=1.7.0
  - matplotlib>=3.5.0
  - pandas>=1.3.0
  - scikit-learn>=1.0.0
  - jupyter>=1.0.0
  - seaborn>=0.11.0
```

Package Channels

Understanding Conda Channels

Default Channels: Managed by Anaconda Inc.

```
conda install numpy  # From default channel
```

Conda-Forge: Community-maintained, more packages

```
conda install -c conda-forge xgboost
```

Setting Channel Priority:

```
# Add conda-forge as highest priority
conda config --add channels conda-forge

# Set strict channel priority
conda config --set channel_priority strict
```

Common ML Packages

Essential Libraries for Machine Learning

```
# Core scientific computing
conda install numpy scipy matplotlib

# Data manipulation and analysis
conda install pandas

# Machine learning
conda install scikit-learn

# Deep learning
conda install tensorflow  # or pytorch

# Jupyter environment
conda install jupyter notebook ipykernel

# Visualization
conda install seaborn plotly
```

Troubleshooting Common Issues

Package Conflicts

Problem: Conda can't resolve package versions

```
# This might fail due to conflicts
conda install package-a package-b
```

Solutions:

```
# Try conda-forge channel
conda install -c conda-forge package-a package-b

# Install one at a time
conda install package-a
conda install package-b

---
```

Conda vs. Pip

When to Use What?

Use Conda When:

- Setting up new environments
- Installing scientific/ML packages
- Need non-Python dependencies (C libraries, etc.)
- Want easy environment management

Use Pip When:

- Package not available in conda
- Installing from GitHub
- Working with specific PyPI packages

Managing Jupyter Kernels

Using Different Environments in Jupyter

```
# Install ipykernel in your environment
conda activate ml-course
conda install ipykernel

# Register environment as Jupyter kernel
python -m ipykernel install --user --name ml-course --display-name "ML Course"

# Now available in Jupyter notebook kernel menu
jupyter notebook
```

List Available Kernels:

```
jupyter kernelspec list
```

Remove Kernel:

Performance Tips

Making Conda Faster

Use Mamba (Faster Solver):

```
# Install mamba
conda install mamba -c conda-forge

# Use mamba instead of conda for installations
mamba install numpy scipy matplotlib  # Much faster!
```

Configure Conda:

```
# Disable automatic base activation
conda config --set auto_activate_base false

# Show channel URLs
conda config --set show_channel_urls true
```

Common Commands Reference

Quick Reference Card

```
# Environment Management
conda create -n myenv python=3.10    # Create environment
conda activate myenv                 # Activate environment
conda deactivate                     # Deactivate environment
conda env list                       # List environments
conda env remove -n myenv            # Remove environment

# Package Management
conda install package                # Install package
conda install package=1.0            # Install specific version
conda update package                 # Update package
conda remove package                 # Remove package
conda list                           # List installed packages

# Information
conda info                           # System information
conda search package                 # Search for package
```

Backup and Restore

Saving Your Work

Export Exact Environment:

```
# Includes exact versions and build numbers
conda list --explicit > spec-file.txt

# Recreate exact environment
conda create -n myenv-copy --file spec-file.txt
```

Export Cross-Platform Environment:

```
# Works across different operating systems
conda env export --no-builds > environment.yml
conda env create -f environment.yml
```

Next Steps

You're Ready When You Can:

✅ Create and activate a new conda environment
✅ Install packages using conda
✅ Export and share environment files
✅ Switch between different project environments
✅ Use Jupyter with different kernels

For This Course:

```
# Create your ML course environment
conda create -n ml-course python=3.10 numpy pandas matplotlib scikit-learn jupyter

# Activate it
conda activate ml-course

# You're ready for machine learning!
```

Thank You!

Next: Fundamental Algorithms

Resources:

- **Conda Documentation:** https://docs.conda.io/
- **Conda Cheat Sheet:** https://conda.io/projects/conda/en/latest/user-guide/cheatsheet.html
- **Environment Files:** Best practice for reproducible research