Hands-on ML 01: From NumPy to Linear Regression

Navid Seidi

Programmers House

September 2025

Today's Roadmap

1. **NumPy**: arrays, axes, broadcasting, dot products
2. **Pandas**: DataFrames, inspecting datasets
3. **Matplotlib**: visualization (scatter + line)
4. **Linear Regression**: theory, coding, evaluation

What is NumPy?

Definition

NumPy is Python's library for **numerical computing**. It gives us:

- **ndarray**: a fast, multidimensional array (like a super-powered list).
- **Vectorization**: perform math on entire arrays without loops.
- **Linear algebra tools**: dot products, matrix multiplication, inverses.

Why Important for ML?

Machine learning is mostly linear algebra (vectors + matrices). NumPy is the engine behind it.

Creating Arrays

```python
import numpy as np

# 1D array (vector)
v = np.array([1, 2, 3])

# 2D array (matrix)
M = np.array([[1, 2, 3],
              [4, 5, 6]])

print("Vector:", v, "Shape:", v.shape)
print("Matrix:\n", M, "Shape:", M.shape)
```

Key Idea

- 1D array: shape $(n,)$
- 2D array: shape $(rows, cols)$

Shape tells NumPy the structure of your data.

Axes Explained

Axis tells NumPy which direction to operate along.

```
M = np.array([[1, 2, 3],
              [4, 5, 6]])

print(M.sum(axis=0))  # [5, 7, 9] (sum down columns)
print(M.sum(axis=1))  # [6, 15]   (sum across rows)
```

Rule of Thumb

- `axis=0`: operate **down columns**.
- `axis=1`: operate **across rows**.

Broadcasting

Definition

Broadcasting allows NumPy to **automatically expand arrays** so operations with different shapes can still work.

```
X = np.array([[1, 2, 3],
              [4, 5, 6]])
v = np.array([1, 0, -1])

print(X + v)
```

Rules

- NumPy compares shapes from right to left.
- If dimensions match (or one is 1), they are compatible.
- Missing dimensions are filled with 1s.

Reshaping Arrays

Definition

`reshape` changes the shape of an array without changing its data.

```
a = np.arange(6)          # [0 1 2 3 4 5]
b = a.reshape(2, 3)       # 2 rows, 3 columns
print(b)
```

Use Case

When preparing data for ML, we often need shapes like (n, d): $n =$ number of samples, $d =$ features.

Dot Product and Matrix Multiplication

Dot Product

$\mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i$ Predicts values like $\hat{y} = \mathbf{x} \cdot \mathbf{w}$.

```
a = np.array([2, 3, 4])
b = np.array([5, 6, 7])

print(a @ b)  # dot product
```

Matrix Multiplication

If A is $(m \times n)$ and B is $(n \times p)$, then AB is $(m \times p)$.

Broadcasting (Your New Superpower)

```
import numpy as np
X = np.array([[1, 2, 3],
              [4, 5, 6]], dtype=float)
mu = X.mean(axis=0)        # mean per column
X_centered = X - mu        # broadcasts mu to each row
print(mu)                  # [2.5 3.5 4.5]
print(X_centered)
```

Idea

Shapes that "fit" can auto-expand to match. Avoids slow Python loops.

Quick Quiz: NumPy

Which line computes $\mathbf{y} = \mathbf{X}\mathbf{w}$ for $X \in \mathbb{R}^{n \times d}$ and $w \in \mathbb{R}^d$?

1. `y = X * w`
2. `y = X @ w`
3. `y = np.dot(w, X)`

Quick Quiz: NumPy

Which line computes $\mathbf{y} = \mathbf{X}\mathbf{w}$ for $X \in \mathbb{R}^{n \times d}$ and $w \in \mathbb{R}^d$?

1. `y = X * w`
2. `y = X @ w`
3. `y = np.dot(w, X)`

Answer: (2).

What is Pandas?

Definition

Pandas is Python's library for **tabular data**. It provides:

- **Series**: 1D labeled data (like a spreadsheet column).
- **DataFrame**: 2D labeled data (like a spreadsheet).

Why Important?

Most ML datasets are tables (CSV). Pandas makes them easy to load, explore, and clean.

Mini Dataset (House Prices)

```
import pandas as pd
from io import StringIO

csv = StringIO("""size,bedrooms
    ,price
650,1,120000
800,2,155000
900,2,180000
1200,3,240000
1500,3,310000
1800,4,360000
""")
df = pd.read_csv(csv)
df.head()
```

size	bedrooms	price
650	1	120000
800	2	155000
...	...	...

Loading and Inspecting Data

```python
import pandas as pd

df = pd.read_csv("houses.csv")  # load CSV
print(df.head())           # first rows
print(df.describe())       # summary stats
print(df.info())           # datatypes
print(df.isna().sum())     # missing values
```

Tip

Always explore before modeling. Missing values and datatypes matter.

Indexing and Selecting Data

```
# Select a column
sizes = df["size"]

# Select multiple columns
subset = df[["size","price"]]

# Row selection
print(df.iloc[0])          # by index
print(df.loc[0])           # by label
```

Remember

`iloc` = by index position. `loc` = by label.

Correlation

```
print(df.corr())
```

Definition

Correlation measures how strongly two variables move together.

- Range: -1 to 1
- $+1$: perfect positive relation
- -1: perfect negative relation
- 0: no linear relation

Basic EDA

```
print(df.describe())              # quick stats
print(df.isna().sum())            # missing values per column
print(df.corr(numeric_only=True))    # correlations
```

Tip

Keep features numeric for now; we'll add encoding later in the course.

Quick Quiz: Pandas

To select `size` and `price` as a new DataFrame:

1. `df["size","price"]`
2. `df[["size","price"]]`
3. `df.loc["size","price"]`

Quick Quiz: Pandas

To select `size` and `price` as a new DataFrame:

1. `df["size","price"]`
2. `df[["size","price"]]`
3. `df.loc["size","price"]`

Answer: (2).

What is Matplotlib?

Definition

Matplotlib is Python's **visualization library**. It helps create:

- Scatter plots (relationships)
- Line plots (trends)
- Histograms, bar charts

Why Important?

Seeing patterns helps us understand data and model results.

Scatter Plot Example

```
import matplotlib.pyplot as plt

plt.scatter(df["size"], df["price"])
plt.xlabel("Size (sq ft)")
plt.ylabel("Price ($)")
plt.title("Size vs Price")
plt.show()
```

Interpretation

A positive trend: larger houses usually cost more.

Plotting: Scatter + Line

```
import matplotlib.pyplot as plt

plt.scatter(df["size"], df["price"])
plt.xlabel("Size (sq ft)")
plt.ylabel("Price ($)")
plt.title("Size vs. Price")
plt.show()
```

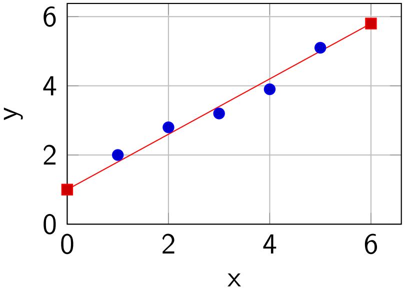
Goal

Visualize patterns before modeling. Does a line make sense here?

A Visual Thought Experiment

Idea: Fit a line $\hat{y} = w_0 + w_1 x$ that best follows the scatter.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$



The Model

Hypothesis (one feature)

$$\hat{y} = w_0 + w_1 x$$

Loss (Mean Squared Error)

$$J(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Goal

Find w_0, w_1 that minimize J.

Two Ways to Fit

Normal Equation (closed-form)

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Fast for small d, no learning rate.

Gradient Descent (iterative)

$$w_j \leftarrow w_j - \alpha \frac{\partial J}{\partial w_j}$$

Scales to big data, teaches optimization.

Build the Design Matrix

```
import numpy as np

x = df["size"].to_numpy().reshape(-1, 1)   # n x 1
y = df["price"].to_numpy().reshape(-1, 1)  # n x 1

# Add bias term (column of ones)
X = np.hstack([np.ones((x.shape[0], 1)), x])  # n x 2
print(X[:3], y[:3], sep="\n")
```

Fit via Normal Equation (NumPy Only)

```
# w = (X^T X)^{-1} X^T y
XtX = X.T @ X
Xty = X.T @ y
w = np.linalg.inv(XtX) @ Xty  # shape (2,1)
w0, w1 = float(w[0]), float(w[1])
print(f"Intercept={w0:.2f}, Slope={w1:.2f}")
```

Interpretation

Every extra square foot adds roughly w_1 dollars (in this toy data).

Visualize the Fit

```
y_hat = X @ w    # predictions

plt.scatter(df["size"], df["price"], label="data")
xs = np.linspace(df["size"].min(), df["size"].max(),
    100)
ys = w0 + w1 * xs
plt.plot(xs, ys, label="fitted line")
plt.xlabel("Size (sq ft)")
plt.ylabel("Price ($)")
plt.title("Linear Regression Fit")
plt.legend()
plt.show()
```

Train/Test Split & Metrics

```
# simple manual split (last 2 rows as test)
train = df.iloc[:-2]
test  = df.iloc[-2:]

Xtr = np.hstack([np.ones((len(train),1)),
                 train["size"].to_numpy().reshape
                    (-1,1)])
ytr = train["price"].to_numpy().reshape(-1,1)

Xte = np.hstack([np.ones((len(test),1)),
                 test["size"].to_numpy().reshape(-1,1)
                    ])
yte = test["price"].to_numpy().reshape(-1,1)

# fit on train
w = np.linalg.inv(Xtr.T @ Xtr) @ (Xtr.T @ ytr)

# predict on test
yhat_te = Xte @ w
```

What Does R^2 Mean?

$$R^2 = 1 - \frac{\sum(y - \hat{y})^2}{\sum(y - \bar{y})^2}$$

- $R^2 = 1$: perfect fit; $R^2 = 0$: predicts the mean; can be negative
- "How much variance did our line explain?"

Quick Quiz: Linear Regression

Which change *definitely* reduces MSE (on train) for a **linear** model?

1. Adding a constant to all targets y
2. Scaling x by 1000
3. Adding a strongly relevant feature (e.g., bedrooms)

Quick Quiz: Linear Regression

Which change *definitely* reduces MSE (on train) for a **linear** model?

1. Adding a constant to all targets y
2. Scaling x by 1000
3. Adding a strongly relevant feature (e.g., bedrooms)

Answer: (3).

Implement GD in 12 Lines

```
X = np.hstack([np.ones((len(df),1)),
               df["size"].to_numpy().reshape(-1,1)])
y = df["price"].to_numpy().reshape(-1,1)

w = np.zeros((2,1))  # init
alpha = 1e-7         # learning rate (tiny units:
    dollars)
for step in range(20000):
    y_hat = X @ w
    grad = (2/len(X)) * (X.T @ (y_hat - y))
    w -= alpha * grad

print(w.ravel())  # close to normal equation solution
```

Tip

Tune α. If loss explodes, reduce it; if learning is too slow, increase it.

Add Bedrooms (2 Features)

```
X2 = np.hstack([np.ones((len(df),1)),
                df[["size","bedrooms"]].to_numpy()])
y  = df["price"].to_numpy().reshape(-1,1)

w2 = np.linalg.inv(X2.T @ X2) @ (X2.T @ y)
print("w0, w_size, w_bedrooms =", w2.ravel())
```

Interpretation

Each coefficient tells the marginal effect holding the other feature fixed.

Today You Learned

- **NumPy**: arrays, shape, axis, broadcasting, reshaping, dot product
- **Pandas**: Series, DataFrames, indexing, correlation
- **Matplotlib**: scatter plots, line plots, labels and legends
- **Linear Regression**: model equation, loss function, fitting with normal equation and gradient descent, evaluation with MSE and R^2, and extension to multiple features

Key Takeaways

Skills You Now Have

- Load and inspect datasets with Pandas
- Perform vectorized math with NumPy
- Visualize data and models with Matplotlib
- Build and evaluate your first ML model: **Linear Regression**

Big Idea

Machine Learning = **Math + Code + Data**. Today you saw how these three come together.

Quiz 1: NumPy Basics

1. What does `axis=0` mean in NumPy?
2. What is broadcasting in your own words?
3. What is the difference between `reshape(6,1)` and `reshape(1,6)`?

Quiz 1: NumPy Basics

1. What does `axis=0` mean in NumPy?
2. What is broadcasting in your own words?
3. What is the difference between `reshape(6,1)` and `reshape(1,6)`?

Answers: 1. Operate down columns. 2. Automatic expansion of arrays to match shapes. 3. One is a column vector, the other is a row vector.

Quiz 2: Pandas & Visualization

1. How do you select the first 5 rows of a DataFrame?
2. What does `corr()` compute?
3. In a scatter plot, which variable usually goes on the x-axis?

Quiz 2: Pandas & Visualization

1. How do you select the first 5 rows of a DataFrame?
2. What does `corr()` compute?
3. In a scatter plot, which variable usually goes on the x-axis?

Answers: 1. `df.head()`. 2. Correlation between numeric columns. 3. The independent variable (input feature).

Quiz 3: Linear Regression

1. In regression, what does the slope (w_1) represent?
2. What does the intercept (w_0) represent?
3. If $R^2 = 0.9$, what does that tell us?

Quiz 3: Linear Regression

1. In regression, what does the slope (w_1) represent?
2. What does the intercept (w_0) represent?
3. If $R^2 = 0.9$, what does that tell us?

Answers: 1. How much the target changes when the input increases by 1. 2. The baseline prediction when $x = 0$. 3. The model explains 90% of the variance in the data.

Where to Go Next?

- Try Linear Regression with a bigger dataset (e.g., Boston Housing, Kaggle datasets).
- Experiment with adding more features.
- Compare manual NumPy implementation with scikit-learn's `LinearRegression`.

Preview of Next Session

We will explore **classification**, starting with Logistic Regression and Spam Filtering.

Thank You!

Questions?