

Section 5: Constructing A First AI System

DO NOT POLLUTE! AVOID PRINTING, OR PRINT 2-SIDED MULTIPAGE.

Introduction

Here we will construct our first AI system from scratch. The goal is to make concrete and concise the abstract steps we have discussed so far. By doing so, we aim to demystify AI systems, showing that, at their core, AI systems are far simpler than they may initially appear.

In order to construct our model, we will follow the general steps outlined before:

1. Decide our goal.
2. Obtain the data.
3. Specify the model, that is, the function f.
4. Specify the loss, that is the function ℓ.
5. Learn the optimal parameter $\mathbf{w}^\star$.
6. Use our AI system on new data.

5.1 Deciding our goal

This is arguably the simplest step. In this example we will explore a very concrete and tangible goal: predicting sales as a function of a diversified advertisement budget.

5.2 Obtaining the data

Here we will use a common advertising dataset containing information about the sales of one product in relation to the advertising budgets spent on TV, radio and newspaper.

This dataset contains 200 samples, each with 4 attributes representing:

- Budget allocated to TV advertising.
- Budget allocated to radio advertising.
- Budget allocated to newspaper advertising.

- Sales of the product (in thousands of units).

Here is a snapshot of how this dataset looks like:

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12.0
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9
8.7	48.9	75.0	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1.0	4.8
199.8	2.6	21.2	15.6

This data can be loaded to our computer with a few simple lines of code:

```
import numpy as np
import pandas as pd

# Load data
df = pd.read_csv("advertising.csv")

# Data matrix and response
X = df.iloc[:, :3].values
y = df.iloc[:, 3].values.reshape(-1, 1)
```

5.3 Specifying the Model

For our first AI system we will use one of the simplest models imaginable: a linear model of the form:

$$f(\mathbf{x}) \;=\; \mathbf{a}^{\mathsf{T}}\mathbf{x} + \mathrm{b}$$

Notice that by appending a 1 on top of $\mathbf{x}$, this model can be simplified to

$$f(\mathbf{x}) \;=\; \mathbf{w}^{\mathsf{T}}\mathbf{x},$$

where $\mathbf{w}$ is a vector with one more entry than $\mathbf{a}$, corresponding to b.

This data transformation can be trivially achieved with the following code:

```
# Add bias term (intercept)
X = np.hstack([np.ones((X.shape[0], 1)), X])
```

5.4 Specifying the Loss

For this first AI model we will consider one of the most prevalent losses, known as the mean squared error (MSE):

$$\ell(\mathbf{w}) \;=\; \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

5.5 Learning the Optimal Parameter $\mathbf{w}^\star$

Recall that our goal is to find:

$$\mathbf{w}^\star \;=\; \arg\min_{\mathbf{w}} \; \ell(\mathbf{w}) \;=\; \arg\min_{\mathbf{w}} \; \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

Luckily for us, this is such a simple loss that we can minimize it using our 3-step optimization 101 procedure: take the derivative, set to zero, and solver for $\mathbf{w}$. To see this, we can first rewrite the loss in a way that is easier to obtain its derivative:

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \;=\; (\mathbf{y} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}) \;=\; \mathbf{y}^\mathsf{T}\mathbf{y} - 2\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y} + \mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w}.$$

At this point, it is straightforward to take the derivative with respect to $\mathbf{w}$. To learn more about how to take derivatives w.r.t. vectors and matrices see *Old and new matrix algebra useful for statistics* by Thomas P. Minka. In this case, the derivative is:

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}} \;=\; -2\mathbf{X}^\mathsf{T}\mathbf{y} + 2\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w}.$$

Setting this to zero we have:

$$\begin{aligned} -2\mathbf{X}^\mathsf{T}\mathbf{y} + 2\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &= 0 \\ 2\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &= 2\mathbf{X}^\mathsf{T}\mathbf{y} \\ \mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &= \mathbf{X}^\mathsf{T}\mathbf{y} \\ \mathbf{w} &= (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}. \end{aligned}$$

We thus conclude that

$$\mathbf{w}^\star = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}.$$

This can be easily computed in python:

```
# Closed-form solution
w_star = np.linalg.inv(X.T @ X) @ X.T @ y
print(w_star)
```

5.6 Using our AI System on New Data

At this point we are ready to use our trained AI system on new data. Given a new sample, for example

$$\mathbf{x}_{new} \;=\; \begin{bmatrix} 1 \\ 400 \\ 10 \\ 60 \end{bmatrix},$$

all we have to do is compute its predicted response, given by

$$f(\mathbf{x}_{new}) \;=\; (\mathbf{w}^{\star})^{\mathsf{T}}\mathbf{x}_{new}.$$

This can be trivially done with the following code:

```
# Compute response of new sample
x_new = [ 1,400,10,60 ]
y_new = x_new @ w_star
print(y_new)
```