CIS 5150
Example of a Learning Problem

Jean Gallier

August 28, 2024

Chapter 1

Learning a Function

Suppose we are interested in predicting the price of a wine from various regions of France.

A given wine has the following features:

(1) Wine "color": red, rosé, white.

(2) Denomination (region): Pommard, Volnay, Clos Vougeot, Chablis, Sancerre.

(3) Year of production.

Suppose we record the prices (in Euros) of some bottles of wines purchased in 2013–2022 (over 10 years).

Wines $x_1, \ldots, x_{10}$ purchased in 2013-2022:

2013	2014	2015	2016	2017
red	red	white	rosé	red
Pommard	Volnay	Chablis	Sancerre	Clos Vougeot
1985	1995	2010	2016	2003

2018	2019	2020	2021	2022
red	red	white	red	red
Pommard	Volnay	Chablis	Sancerre	Clos Vougeot
1980	2000	2016	2017	2005

Prices $y_1, \ldots, y_{10}$ of the wines listed in the above table in Euros:

200
100
40
25
150
250
135
40
20
300

Question: given a bottle of wine x specified by three attributes

color
denomination
year of production

predict its price y.

To solve the problem, first we need to encode the features as numbers:

Say

red $= 1$, rosé $= 2$, white $= 3$;

Pommard $= 1$, Volnay $= 2$, Clos Vougeot $= 3$,
Chablis $= 4$, Sancerre $= 5$.

Our data set of 10 wines becomes a *matrix*.

As we will later, it is more convenient to use its transpose:

$$X = \begin{pmatrix} 1 & 1 & 1985 \\ 1 & 2 & 1995 \\ 3 & 4 & 2010 \\ 2 & 5 & 2016 \\ 1 & 3 & 2003 \\ 1 & 1 & 1980 \\ 1 & 2 & 2000 \\ 3 & 4 & 2016 \\ 1 & 5 & 2017 \\ 1 & 3 & 2005 \end{pmatrix}.$$

We view our set of data, (10 wines), and their prices, as defining a partial function specified by the sequence of input/output pairs

$$((x_1, y_1), (x_2, y_2), \ldots, (x_{10}, y_{10})),$$

where $x_1, \ldots, x_{10}$ are encoded as the rows of the matrix X (technically, each row of X is the transpose of the column vectors x_i).

We would like to find a *real-valued function* f such that

$$f(x_i) = y_i, \quad i = 1, \ldots, 10,$$

to *predict* the price $y = f(x)$ of a new wine x.

For example, what is an estimate for the price of the wine

red
Clos Vougeot
2000

that is

$$x = \begin{pmatrix} 1 \\ 3 \\ 2000 \end{pmatrix}.$$

The *big* question: what kind of function is f?

Before deep learning, an *affine function*.

After deep learning,
a composition of (vector-valued) affine functions interleaved with some non-linear function such as RELU.

Such compositions can be represented as certain kinds of nets.

Deep learning provides a much larger supply of functions to be learned.

We still have the problem that it usually impossible to find a function f that fits exactly the data, in the sense that $f(x_i) = y_i$ for $i = 1, \ldots, 10$, so we do the best we can, which means that we introduce an *error function*, also known as a *loss function*, and we try to *minimize* this error function.

A pretty good error function is

$$\sum_{i=1}^{10}(f(x_i) - y_i)^2.$$

The function f is defined by some parameters that need be inferred from the data set

$$((x_1, y_1), (x_2, y_2), \ldots, (x_{10}, y_{10})),$$

known as *training data*.

Typically to minimize the error function we need to find its *gradient* and set it to zero. This process will (hopefully!) determine the parameters defining the function f.

The simplest case is to find an *affine function*, of the form

$$f(z_1, z_2, z_3) = w_1 z_1 + w_2 z_2 + w_3 z_3 + b,$$

where $z_1, z_2, z_3, b \in \mathbb{R}$. The number w_1, w_2, w_3 are called *weights*, and they constitute the weight vector w.

We need to "solve" the system

$$\begin{pmatrix} 1 & 1 & 1985 \\ 1 & 2 & 1995 \\ 3 & 4 & 2010 \\ 2 & 5 & 2016 \\ 1 & 3 & 2003 \\ 1 & 1 & 1980 \\ 1 & 2 & 2000 \\ 3 & 4 & 2016 \\ 1 & 5 & 2017 \\ 1 & 3 & 2005 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + b \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 200 \\ 100 \\ 40 \\ 25 \\ 150 \\ 250 \\ 135 \\ 40 \\ 20 \\ 300 \end{pmatrix}$$

with respect to the unknown w_1, w_2, w_3, b.

For example, the second equation is

$$w_1 + 2w_2 + 1995w_3 + b = 100.$$

This is generally impossible so instead we try to minimize an error function. In the case of least squares, we wish to minimize

$$\|Xw + b\mathbf{1} - y\|_2^2$$

with respect to w.

Here we use the 2-norm given by

$$\|(z_1, \ldots, z_n)\|_2^2 = \sum_{i=1}^{n} z_i^2.$$

Typically it is preferable to *penalize (regularize)* w so instead we minimize

$$\|Xw + b\mathbf{1} - y\|_2^2 + K \|w\|_2^2,$$

where $K > 0$ controls the influence of the penalty.

This is *ridge regression*.

The smaller K is, the smaller is the 2-norm $\|Xw + b\mathbf{1} - y\|_2$ of the error, and the larger is $\|w\|_2$.

Here are the results for $x = (1, 3, 2000)$ (red Clos Vougeot 2000).

For $K = 0.01$, we get

$$\begin{aligned} w &= (-33.27, -40.29, 0.24), \quad b = -192.90, \\ \|Xw + b\mathbf{1} - y\|_2 &= 192.29, \quad \|w\|_2 = 52.25 \\ y &= 141.97. \end{aligned}$$

For $K = 10$, we get

$$\begin{aligned} w &= (-10.47, -5.84, -4.35), \quad b = 8878 \\ \|Xw + b\mathbf{1} - y\|_2 &= 202.97, \quad \|w\|_2 = 12.75 \\ y &= 142.99. \end{aligned}$$

The price of the red Clos Vougeot 2000 is predicted to be approximately 142 Euros.

My colleagues Kostas Daniilidis and Jianbo Shi pointed out that the conversion of strings (red, rosé, *etc.*) as vectors that I used yields weight vectors of very small dimension (3), so it is hard for an affine function to fit the data well.

It might be preferable to use the following encoding:

red $= (1, 0, 0)$, rosé $= (0, 1, 0)$, white $= (0, 0, 1)$,

Pommard $= (0, 0, 0)$, Volnay $= (1, 0, 0)$, Clos Vougeot $= (0, 1, 0)$, Chablis $= (0, 0, 1)$, Sancerre $= (1, 0, 1)$.

For example, red Pommard 1985 is encoded by the vector

$$(1, 0, 0, 0, 0, 0, 1985).$$

The new matrix corresponding to the data is

$$X_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1985 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1995 \\ 0 & 0 & 1 & 0 & 0 & 1 & 2010 \\ 0 & 1 & 0 & 1 & 0 & 1 & 2016 \\ 1 & 0 & 0 & 0 & 1 & 0 & 2003 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1980 \\ 1 & 0 & 0 & 1 & 0 & 0 & 2003 \\ 0 & 0 & 1 & 0 & 0 & 1 & 2016 \\ 1 & 0 & 0 & 1 & 0 & 1 & 2017 \\ 1 & 0 & 0 & 0 & 1 & 0 & 2005 \end{pmatrix}.$$

This matrix has rank 7, which means that its columns are linearly independent. So $(X_2^\top)X_2$ is invertible.

Here are the results for $x = (1, 0, 0, 0, 1, 0, 2000)$ (red Clos Vougeot 2000).

For $K = 0.01$, we get

$$\begin{aligned} w &= (33.76, 39.06, -72.82, -132.47, \\ &\quad -33.25, -125.95, 1.61), \\ b &= -3000.9, \\ \|X_2 w + b\mathbf{1} - y\|_2 &= 112.93, \quad \|w\|_2 = 206.12 \\ y &= 218.73. \end{aligned}$$

For $K = 10$, we get

$$\begin{aligned} w &= (6.86, -1.89, -4.97, -9.79, \\ &\quad 15.99, -9.66, -4.75), \\ b &= 9634.7, \\ \|X_2 w + b\mathbf{1} - y\|_2 &= 180.73, \quad \|w\|_2 = 23.29 \\ y &= 164.05. \end{aligned}$$

For $K = 1$, we get $\|X_2 w + b\mathbf{1} - y\|_2 = 130.08$ and

$$y = 213.32.$$

This time the price of the red Clos Vougeot 2000 is predicted in a wide range.

The best fit of the data for the three cases is achieved when $K = 0.01$.

One problem is that our data set is quite small. The other problem is that our choice of attributes is rather crude.

There are other strategies: lasso, elastic net.

In *lasso* we *penalize the 1-norm* $\|w\|_1$ *of* w, so we minimize

$$\|X_2 w + b\mathbf{1} - y\|_2^2 + \tau \|w\|_1 ,$$

where $\tau > 0$ and

$$\|(z_1, \ldots, z_n)\|_1 = |z_1| + \cdots + |z_n|.$$

For $\tau = 0.01$, we get

$$\|X_2 w + b\mathbf{1} - y\|_2 = 112.652, \quad y = 214.81.$$

For $\tau = 0.1$, we get

$$\|X_2 w + b\mathbf{1} - y\|_2 = 112.655, \quad y = 215.25.$$

For $\tau = 1$, we get

$$\|X_2 w + b\mathbf{1} - y\|_2 = 113.03, \quad y = 219.64.$$

For $\tau = 10$, we get

$$\|X_2 w + b\mathbf{1} - y\|_2 = 117.65, \quad y = 225.08.$$

When $\tau = 10$, the first two components of w are basically zero.

For $\tau = 50$, we get

$$\|X_2 w + b\mathbf{1} - y\|_2 = 137.29, \quad y = 222.06.$$

When $\tau = 50$, five components of w are basically zero.

In the case of deep learning, we have several affine functions (typically vector-valued) interleaved with RELU, so gradients are computed using a *back-propagation process* (based on the chain rule), and because the dimension of the data is very large, we use *stochastic gradient descent* methods.

Note that our data set is very crude, because how dry or rainy a year is has great influence on the quality and quantity of wine produced. So our prediction function will probably not be very good! We should also include the date of purchase to the data base.

This is an important modelling issue, but not so much a mathematical issue.