Linear Regression

CMPUT 328

Nilanjan Ray

Computing Science, University of Alberta, Canada

Material source: “Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems,” by Géron, Aurélien.

Linear regression with PyTorch

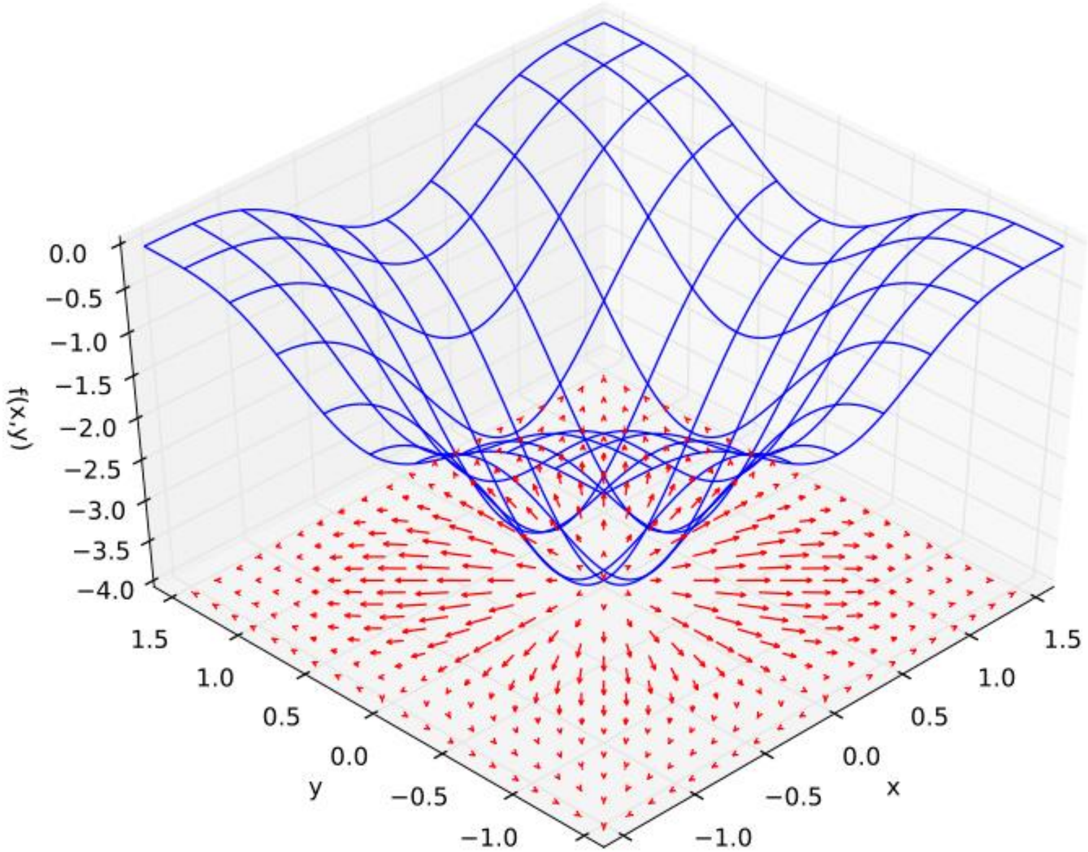
- We will start with a linear regression “model”
- Next, we need to understand “loss” function for regression task
- Next we will estimate the model by minimizing the loss function
- We will use PyTorch

Quick review: Gradient of a function

Example:

$$f(x,y) = -(cos^2 x + cos^2 y)^2$$

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 4\left(cos^2(x) + cos^2(y)\right)\cos(x)\sin(x) \\ 4\left(cos^2(x) + cos^2(y)\right)\cos(y)\sin(y) \end{bmatrix}$$



Note 1: *f* is a function of two variables,
so gradient of *f* is a two dimensional vector

Note 2: Gradient (vector) of *f* points toward the
steepest ascent for *f*

Note 3: At a (local) minimum of *f* its gradient
becomes a zero vector

Example source: Wikipedia

Quick review: Gradient descent optimization

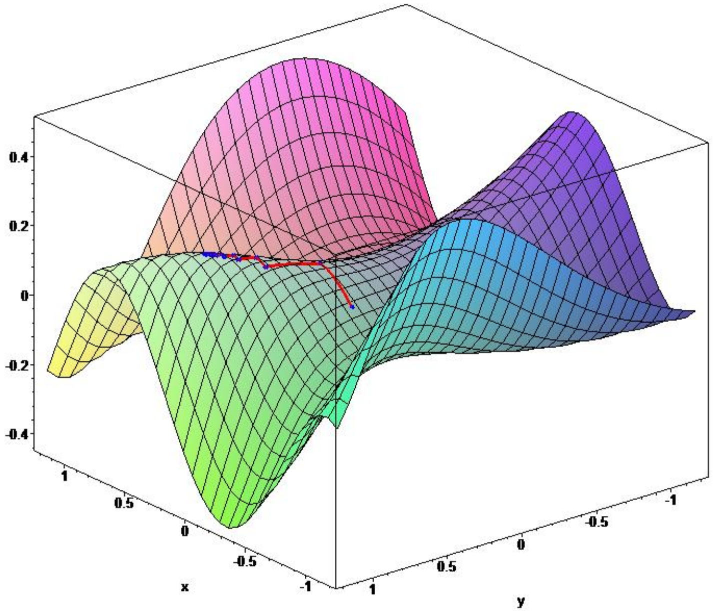
Start at an initial guess for the optimization variable: $\mathbf{x}_0$

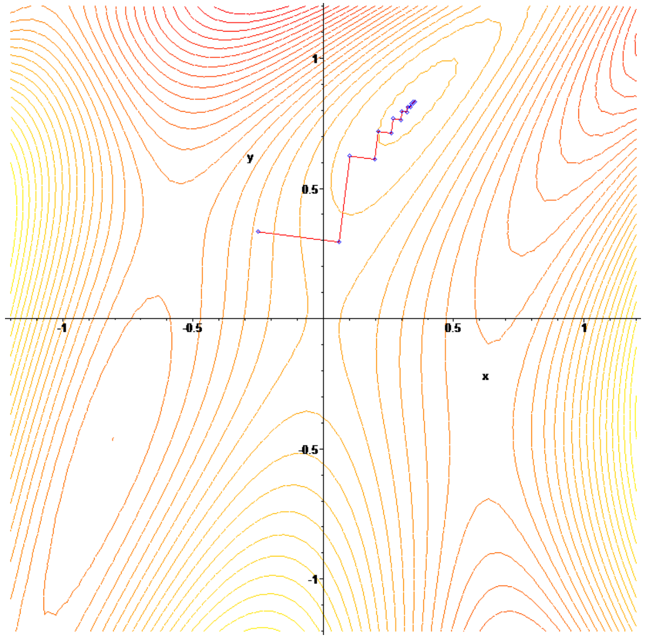
Iterate until gradient magnitude becomes too small: $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \nabla f(\mathbf{x}^t)$

Gradient descent algorithm

α is called the step-length.

f(x, y)





Gradient descent creates a zig-zag path leading to a local minimum of f

Picture source: Wikipedia

Supervised machine learning: the tabular view

Independent variable (aka feature vector)

Prediction / dependent variable

x_1	x_2	x_3	x_4	y
1.2	-3.9	4.0	0	1.6
2.1	2.4	-0.7	-0.2	1.2
…	…	…	…	…
…	…	…	…	…
3.2	…	…	1.9	0.3
1.4	…	…	1.5	?
3.1	…	…	2.1	?

Training data: complete table

Test data: incomplete table

ML learns to map **x** to y

In other words, ML learns a function, f so that
$y = f(\mathbf{x})$

The function f is called prediction function

Linear prediction: formal setup

Linear prediction function: $y^p = \mathbf{x}\boldsymbol{\theta} + b$ or, $y^p = \sum_{j=1}^{m} \theta_j x_j + b$

vector equation form

scalar equation form

A training set consists of ($\mathbf{x}$, y) pairs: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$

Linear prediction on the training data point *i*: $y_i^p = \mathbf{x}_i\boldsymbol{\theta} + b$ or, $y_i^p = \sum_{j=1}^{m} \theta_j x_{i,j} + b$

Loss or cost function (on training data): $L = \frac{1}{2}\sum_{i=1}^{n}(y_i^p - y_i)^2$

Learning the linear model: find out $\boldsymbol{\theta}$ and b to minimize loss function

Linear regression: A toy example

Let's take a toy example:

x_1	x_2	y
1	2	-1
3	-4	7
6	2	3
-3	5	-4
7	-3	5
4	3	?

This equation $y_i^p = \sum_{j=1}^{m} \theta_j x_{i,j} + b$

can be written for the toy training set as

$$y_1^p = \theta_1(1) + \theta_2(2) + b$$
$$y_2^p = \theta_1(3) + \theta_2(-4) + b$$
$$y_3^p = \theta_1(6) + \theta_2(2) + b$$
$$y_4^p = \theta_1(-3) + \theta_2(5) + b$$
$$y_5^p = \theta_1(7) + \theta_2(-3) + b$$

We also have responses:

$$y_1 = -1, y_2 = 7, y_3 = 3, y_4 = -4, y_5 = 5$$

So, the loss is $L = \frac{1}{2}\sum_{i=1}^{n} (y_i^p - y_i)^2 = \frac{1}{2}[(y_1^p + 1)^2 + (y_2^p - 7)^2 + (y_3^p - 3)^2 + (y_4^p + 4)^2 + (y_5^p - 5)^2]$

Learning a linear model

For the convenience of math, let us change our linear model a bit:

$$y_i^p = \sum_{j=1}^{m} \theta_j (x_{i,j} - \bar{x}_j) + b \qquad \text{where,} \quad \bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{i,j}$$

And a slightly modified loss function:

$$L = \underbrace{\frac{1}{2} \sum_{i=1}^{n} (y_i^p - y_i)^2}_{\text{Data fidelity}} + \underbrace{\frac{\gamma}{2} \sum_{j=1}^{m} \theta_j^2}_{\text{Regularization}}$$

γ is a hyper parameter

Why do we need regularization?

Minimization of linear regression loss function

Regularized loss function: $L = \frac{1}{2}\sum_{i=1}^{n}(y_i^p - y_i)^2 + \frac{\gamma}{2}\sum_{j=1}^{m}\theta_j^2$

Taking partial derivative using chain rule: $\frac{\partial L}{\partial b} = \sum_{i=1}^{n}(y_i^p - y_i)\frac{\partial y_i^p}{\partial b} = \sum_{i=1}^{n}(y_i^p - y_i)$ because, $\frac{\partial y_i^p}{\partial b} = 1$

Using $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}$ and $y_i^p = \sum_{j=1}^{m}\theta_j(x_{i,j} - \bar{x}_j) + b$ we get: $\frac{\partial L}{\partial b} = nb - \sum_{i=1}^{n} y_i$

At the minimum of *L*, $\frac{\partial L}{\partial b} = 0$ So, $\boxed{b = \frac{1}{n}\sum_{i=1}^{n} y_i = \bar{y}}$

Linear regression: A toy example...continued

Let's take a toy example:

x_1	x_2	y
1	2	-1
3	-4	7
6	2	3
-3	5	-4
7	-3	5
4	3	?

$$b = \frac{1}{n}\sum_{i=1}^{n} y_i = \bar{y} = \frac{1}{5}(-1+7+3-4+5) = 2$$

$$\bar{x}_1 = \frac{1}{n}\sum_{i=1}^{n} x_{i,1} = \frac{1}{5}(1+3+6-3+7) = 2.8$$

$$\bar{x}_2 = \frac{1}{n}\sum_{i=1}^{n} x_{i,2} = \frac{1}{5}(2-4+2+5-3) = 0.4$$

So, using centered data, the prediction equation becomes:

$$y_i^p = \sum_{j=1}^{m} \theta_j(x_{i,j} - \bar{x}_j) + b = \theta_1\left(x_{i,1} - 2.8\right) + \theta_2\left(x_{i,2} - 0.4\right) + 2$$

So, the loss is

$$L = \frac{1}{2}\sum_{i=1}^{n}(y_i^p - y_i)^2 = \frac{1}{2}\left[\left(y_1^p + 1\right)^2 + \left(y_2^p - 7\right)^2 + \left(y_3^p - 3\right)^2 + \left(y_4^p + 4\right)^2 + \left(y_5^p - 5\right)^2\right]$$

$$= \frac{1}{2}[(\theta_1(1-2.8) + \theta_2(2-0.4) + 2 + 1)^2 + (\theta_1(3-2.8) + \theta_2(-4-0.4) + 2 - 7)^2$$

$$+(\theta_1(6-2.8) + \theta_2(2-0.4) + 2 - 3)^2 + (\theta_1(-3-2.8) + \theta_2(5-0.4) + 2 + 4)^2 + (\theta_1(7-2.8) + \theta_2(-3-0.4) + 2 - 5)^2]$$

Minimization of linear regression loss function…

Regularized loss function:

$$L = \frac{1}{2}\sum_{i=1}^{n}(y_i^p - y_i)^2 + \frac{\gamma}{2}\sum_{j=1}^{m}\theta_j^2$$

Taking partial derivative of L using chain rule:

$$\frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{n}(y_i^p - y_i)\frac{\partial y_i^p}{\partial \theta_j} + \gamma\theta_j$$

Using $\boxed{y_i^p = \sum_{k=1}^{m}\theta_k(x_{i,k} - \bar{x}_k) + b,}$ $\boxed{b = \bar{y}}$ and $\boxed{\frac{\partial y_i^p}{\partial \theta_j} = x_{i,j} - \bar{x}_j}$

We get: $$\frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{n}\left(\sum_{k=1}^{m}\theta_k(x_{i,k} - \bar{x}_k) + \bar{y} - y_i\right)(x_{i,j} - \bar{x}_j) + \gamma\theta_j$$

Linear regression: A toy example…continued

Let's take a toy example:

x_1	x_2	y
1	2	-1
3	-4	7
6	2	3
-3	5	-4
7	-3	5
4	3	?

Note: For this problem I did not assume any regularization

$$\frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{n} \left(\sum_{k=1}^{m} \theta_k (x_{i,k} - \bar{x}_k) + \bar{y} - y_i \right) (x_{i,j} - \bar{x}_j) + \gamma \theta_j$$

$$\begin{aligned}\frac{\partial L}{\partial \theta_1} &\\ = (\theta_1(1-2.8) + \theta_2(2-0.4) + 2 + 1)(1-2.8) &\\ + (\theta_1(3-2.8) + \theta_2(-4-0.4) + 2 - 7)(3-2.8) &\\ + (\theta_1(6-2.8) + \theta_2(2-0.4) + 2 - 3)(6-2.8) &\\ + (\theta_1(-3-2.8) + \theta_2(5-0.4) + 2 + 4)(-3-2.8) &\\ + (\theta_1(7-2.8) + \theta_2(-3-0.4) + 2 - 5)(7-2.8) &\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial \theta_2} &\\ = (\theta_1(1-2.8) + \theta_2(2-0.4) + 2 + 1)(2-0.4) &\\ + (\theta_1(3-2.8) + \theta_2(-4-0.4) + 2 - 7)(-4-0.4) &\\ + (\theta_1(6-2.8) + \theta_2(2-0.4) + 2 - 3)(2-0.4) &\\ + (\theta_1(-3-2.8) + \theta_2(5-0.4) + 2 + 4)(5-0.4) &\\ + (\theta_1(7-2.8) + \theta_2(-3-0.4) + 2 - 5)(-3-0.4) &\end{aligned}$$

Minimization of linear regression loss function…

$$\frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{n} \left(\sum_{k=1}^{m} \theta_k (x_{i,k} - \bar{x}_k) + \bar{y} - y_i \right) (x_{i,j} - \bar{x}_j) + \gamma \theta_j$$

simplification

$\Downarrow$

Gradient of L: $$\boxed{\nabla L = \left[\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})^T (\mathbf{x}_i - \bar{\mathbf{x}}) \right] \boldsymbol{\theta} + \gamma \boldsymbol{\theta} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (y_i - \bar{y}) (\mathbf{x}_i - \bar{\mathbf{x}})^T}$$

where $\mathbf{x}_i = [x_{i,1} \quad \cdots \quad x_{i,m}]$, $\bar{\mathbf{x}} = [\bar{x}_1 \quad \cdots \quad \bar{x}_m]$ and $\boldsymbol{\theta} = [\theta_1 \quad \cdots \quad \theta_m]^T$

More simplified form: $\boxed{\nabla L = (X^T X + \gamma I) \boldsymbol{\theta} - X^T \mathbf{y}}$

where matrix X is defined as: $X = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \vdots \\ \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}$ and vector $\mathbf{y}$ is defined as: $\mathbf{y} = \begin{bmatrix} y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{bmatrix}$

and I is an identity matrix of size m-by-m

Equating gradient of L to zero vector and solving gives us: $\boldsymbol{\theta} = (X^T X + \gamma I)^{-1} X^T \mathbf{y}$

Linear regression: A toy example…finally!

Let's take a toy example:

x_1	x_2	y
1	2	-1
3	-4	7
6	2	3
-3	5	-4
7	-3	5
4	3	?

$$X = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \vdots \\ \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} 1-2.8 & 2-0.4 \\ 3-2.8 & -4-0.4 \\ 6-2.8 & 2-0.4 \\ -3-2.8 & 5-0.4 \\ 7-2.8 & -3-0.4 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{bmatrix} = \begin{bmatrix} -1-2 \\ 7-2 \\ 3-2 \\ -4-2 \\ 5-2 \end{bmatrix}$$

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T \mathbf{y} = \begin{bmatrix} 0.3580 \\ -0.8535 \end{bmatrix}$$

So, finally the prediction for the test data point

$$? = \sum_{j=1}^{m} \theta_j (x_j - \bar{x}_j) + b = 0.3580(4-2.8) - 0.8535(3-0.4) + 2 = 0.2105$$

Note: For this problem I did not assume any regularization

Linear regression by gradient descent

$$\boldsymbol{\theta} = (X^T X + \gamma I)^{-1} X^T \mathbf{y}$$

If the data does not fit into the memory, you cannot compute $\boldsymbol{\theta}$ directly with the above formula; you apply gradient descent to compute it (approximately).

$$\boxed{\nabla L(\boldsymbol{\theta}; X, \mathbf{y}) = X^T X\boldsymbol{\theta} + \gamma\boldsymbol{\theta} - X^T \mathbf{y}}$$

Guess a starting value for $\boldsymbol{\theta} = \boldsymbol{\theta}_0$

Initialize learning rate and regularization parameter: α, γ

Iterate for t = 0, 1,…

Consider a subset of data (X_t, $\mathbf{y}_t$)

Update: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla L(\boldsymbol{\theta}_t; X_t, \mathbf{y}_t)$

Gradient descent algorithm

Also known as batch update or batch method

Derivation using vector calculus

Regularized loss function: $L = \frac{1}{2}\sum_{i=1}^{n}(y_i^p - y_i)^2 + \frac{\gamma}{2}\sum_{j=1}^{m}\theta_j^2$ or, $L = \frac{1}{2}\sum_{i=1}^{n}(y_i^p - y_i)^2 + \frac{\gamma}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$

Using vector calculus: $\nabla L = \sum_{i=1}^{n}(y_i^p - y_i)\nabla y_i^p + \frac{\gamma}{2}\nabla(\boldsymbol{\theta}^T\boldsymbol{\theta})$

Use vector differentiation to: $y_i^p = (\mathbf{x}_i - \bar{\mathbf{x}})\boldsymbol{\theta} + \bar{y}$ and get: $\nabla y_i^p = (\mathbf{x}_i - \bar{\mathbf{x}})^T$

Also note, using vector differentiation rule: $\nabla(\boldsymbol{\theta}^T\boldsymbol{\theta}) = 2\boldsymbol{\theta}$

$$\nabla L = \underbrace{\begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \vdots \\ \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}^T}_{\text{“centered data”}} \underbrace{\begin{bmatrix} y_1^p - y_1 \\ \vdots \\ y_n^p - y_n \end{bmatrix}}_{\text{“error”}} + \gamma\boldsymbol{\theta}$$

https://en.wikipedia.org/wiki/Matrix_calculus

MNIST Dataset

Classify images into digits

Each image is **28x28**

10 labels

55,000 training images

5,000 validation images

10,000 test images.

Linear regression on MNIST dataset

Small 28 pixels-by-28 pixels images of hand written digits

The visual recognition problem definition:
to recognize the digit from an image

Our very first line of attack would be to use linear regression.

Feature dimension, m = 28 * 28 = 784

Let's look at our PyTorch implementation

Pixel values (feature) Digit

x_1	x_2	…	x_{784}	y
0.1	0.3	…	0.0	0
0.2	0.1	…	0.5	1
…	…	…	…	…
…	…	…	…	…
0.0	0.98	…	0.8	9
0.5	0.25	…	0.36	?
0.1	0.95	…	0.1	?