

Title: Machine Learning Lecture - 230228 pt 2

Speakers: Lauren Hayward

Collection: Machine Learning for Many-Body Physics (2022/2023)

Date: February 28, 2023 - 11:30 AM

URL: <https://pirsa.org/23030033>

Outline for Lecture #2

- Our first SL algorithm: linear regression
- Parameter optimization using gradient descent
- Another SL algorithm: k-Nearest Neighbours

Useful references on linear regression:

- arXiv: 1803.08823, Section VI
- U of Toronto course CSC 411, Lecture 6

Recall the set-up

Dataset $\mathcal{D} = \{$

datapoints
 $\vec{x} = (x_1, x_2, \dots)$

Task: fit some

$\vec{f}: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$

Recall the set-up of a SL problem:

$$\text{Dataset } \mathcal{D} = \{(\vec{x}, \vec{y})\}$$

datapoints

$$\vec{x} = (x_1, x_2, \dots, x_{d_x})^T$$

labels

$$\vec{y} = (y_1, y_2, \dots, y_{d_y})^T$$

Task: fit some $\vec{f}(\vec{x})$ to \vec{y}

$$\vec{f}: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$$

Recall the set-up of a SL problem:

$$\text{Dataset } \mathcal{D} = \{(\vec{x}, \vec{y})\}$$

datapoints

$$\vec{x} = (x_1, x_2, \dots, x_{d_x})^T$$

labels

$$\vec{y} = (y_1, y_2, \dots, y_{d_y})^T$$

Task: fit some $\vec{f}(\vec{x})$ to \vec{y}

$$\vec{f}: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$$

Linear Regression (LR)

For \mathcal{D} , f

Linear Regression (LR)

For \mathcal{D} , fit $f(\vec{x})$ to y , with

$$f(\vec{x}) = \sum_{j=1}^{d_x} w_j x_j = \vec{w}^T \vec{x}$$

$\vec{w} = (w_1, w_2, \dots, w_{d_x})^T$ is a vector of fitting parameters

Let's introduce notation

$M = \#$ of datapoints in \mathcal{D}

$x_j^{(i)}$ is the j^{th} element of sample i

$1 \leq j \leq d_x, 1 \leq i \leq M$

matrix $X \in \mathbb{R}^{M \times d_x}$, with $X_{ij} = x_j^{(i)}$

Let's introduce notation

$M = \#$ of datapoints in \mathcal{D}

$x_j^{(i)}$ is the j^{th} element of sample i

$1 \leq j \leq d_x, 1 \leq i \leq M$

matrix $X \in \mathbb{R}^{M \times d_x}$, with $X_{ij} = x_j^{(i)}$

label vector $\vec{y} = (y^{(1)}, y^{(2)}, \dots, y^{(M)})$

The values

Fitting parameters

The values of the parameters are determined by minimizing \mathcal{L} :

$$\mathcal{L} \equiv \sum_{i=1}^M \left(\sum_{j=1}^{d_x+1} w_j x_j^{(i)} - y^{(i)} \right)^2 = \|X\vec{w} - \vec{y}\|_2^2$$

L_2 norm

The values of the parameters are determined by minimizing \mathcal{L} :

$$\mathcal{L} \equiv \sum_{i=1}^M \left(\sum_{j=1}^{d_x} w_j x_j^{(i)} - y^{(i)} \right)^2 = \|X\vec{w} - \vec{y}\|_2^2$$

← L_2 norm

Solving $\vec{w}_{LR} = \underset{\vec{w} \in \mathbb{R}^{d_x}}{\operatorname{argmin}} \mathcal{L}$:

$$\vec{w}_{LR} = (X^T X)^{-1} X^T \vec{y}$$

$$f_{LR}(\vec{x}) = \vec{w}_{LR}^T \vec{x}$$

LR

Note:

$$\|X\vec{w} - \vec{y}\|_2^2$$

L₂ norm

$$X^T X \vec{w} = X^T \vec{y}$$

LR

Note: solution
 assumes $X^T X$ is
 invertible. Infinitely
 many solutions
 for $\text{rank}(X) < d_x$

The values of the parameters are determined by minimizing \mathcal{L} :

$$\mathcal{L} \equiv \sum_{i=1}^M \left(\sum_{j=1}^{d_x} w_j x_j^{(i)} - y^{(i)} \right)^2 = \|X\vec{w} - \vec{y}\|_2^2$$

\swarrow L_2 norm

Solving $\vec{w}_{LR} = \underset{\vec{w} \in \mathbb{R}^{d_x}}{\operatorname{argmin}} \mathcal{L}$:

$$\vec{w}_{LR} = (X^T X)^{-1} X^T \vec{y}$$
$$f_{LR}(\vec{x}) = \vec{w}_{LR}^T \vec{x}$$

LR

label vector $\vec{y} = (y^{(1)}, y^{(2)}, \dots, y^{(M)})$, with $X_{ij} = X_j$

Alternatively, we can use an algorithm such as gradient descent (GD) to search for the params w_i

Why use GD when we already have an exact solution?

label vector $y = (y_1, y_2, \dots, y_n)$

Alternatively, we can use an algorithm such as gradient descent (GD) to search for the params w_i

Why use GD when we already have an exact solution?

- can be more efficient when the matrix $X^T X$ is large
- broadly useful within many ML algorithms

$$\vec{w} \in \mathbb{R}^{d_x}$$

$$f_{LR}(\vec{x}) = \vec{w}_{LR}^T \vec{x}$$

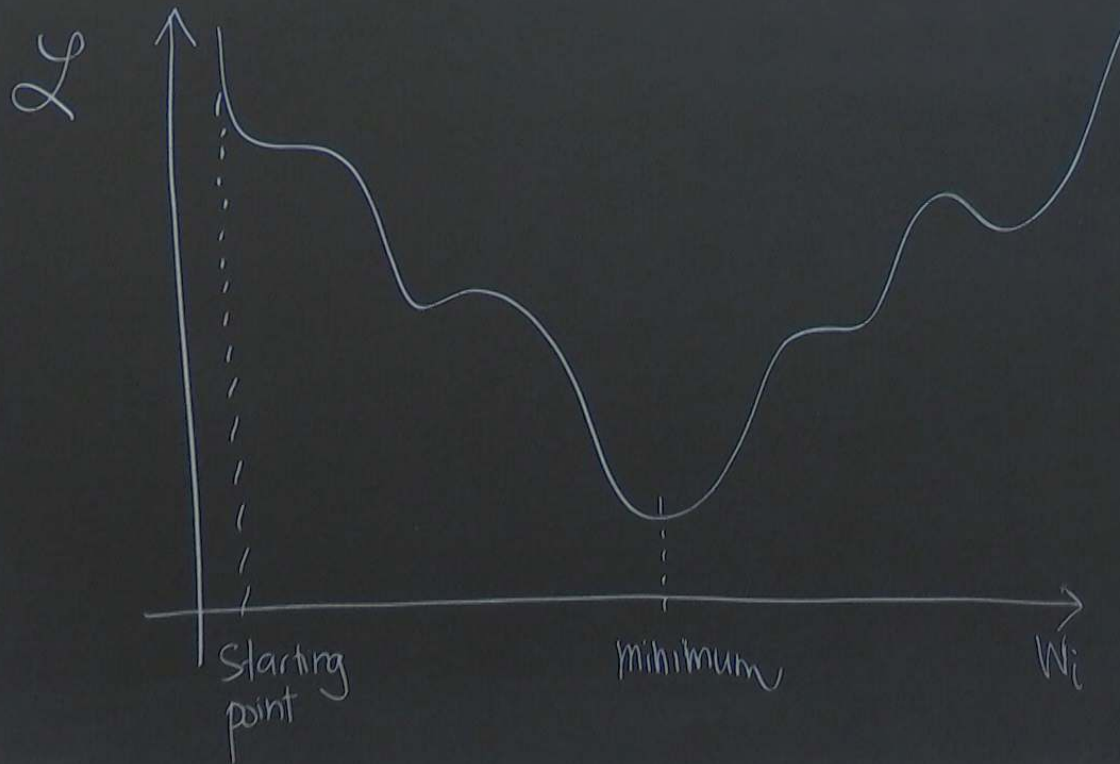
Gradient Descent (GD)

Used when we want to minimize some \mathcal{L} w.r.t. parameters

$$w_1, w_2, \dots, w_n$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0 \quad \text{for } 1 \leq i \leq n$$

For LR: $n = d_x$



Approximate a local

$$\Delta \mathcal{L} \approx \sum_{i=1}^n$$

where:

$$\vec{\nabla}_{\vec{w}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_i} \right)$$

Approximate a local change in \mathcal{L} as:

$$\Delta \mathcal{L} \approx \sum_{i=1}^n \frac{\partial \mathcal{L}}{\partial w_i} \Delta w_i = \vec{\nabla}_{\vec{w}} \mathcal{L} \cdot \vec{\Delta w}$$

where:

$$\vec{\nabla}_{\vec{w}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \dots, \frac{\partial \mathcal{L}}{\partial w_n} \right)$$

$$\vec{\Delta w} = (\Delta w_1, \Delta w_2, \dots, \Delta w_n)$$

We want $\Delta \mathcal{L} < 0$ and we are free to choose $\vec{\Delta w}$.

Let us take:

$$\vec{\Delta w} = -\eta \vec{\nabla}_{\vec{w}} \mathcal{L}$$

GD

\Rightarrow update $w_i \rightarrow w_i - \eta \frac{\partial \mathcal{L}}{\partial w_i}$ each step

We want $\Delta \mathcal{L} < 0$ and we are free to choose $\vec{\Delta w}$.

Let us take

$$\vec{\Delta w} = -\eta \vec{\nabla}_{\vec{w}} \mathcal{L} \quad \text{GD}$$

\Rightarrow update $w_i \rightarrow w_i - \eta \frac{\partial \mathcal{L}}{\partial w_i}$ each step

$\eta > 0$ is a parameter called the "learning rate" that we can choose

We want $\Delta \mathcal{L} < 0$ and we are free to choose $\vec{\Delta w}$.

Let us take:

$$\vec{\Delta w} = -\eta \vec{\nabla}_{\vec{w}} \mathcal{L} \quad \text{GD}$$

\Rightarrow update $w_i \rightarrow w_i - \eta \frac{\partial \mathcal{L}}{\partial w_i}$ each step

$\eta > 0$ is a parameter called the "learning rate" that we can choose (a hyperparameter)

With this choice of $\vec{\Delta w}$:

$$\Delta \mathcal{L} \approx -\eta \left\| \vec{\nabla}_{\vec{w}} \mathcal{L} \right\|_2^2 < 0$$

Notes about η :

With this choice of $\vec{\Delta w}$:

$$\Delta \mathcal{L} \approx -\eta \left\| \vec{\nabla}_{\vec{w}} \mathcal{L} \right\|_2^2 < 0$$

Notes about η :

- if η is too small, it will take a long time to find the solution, can get "stuck" in local minima
- η must be small enough for the first-order approx. to be valid

Another SL algorithm, k-Nearest Neighbours (kNN)

$$\mathcal{D} = \left\{ \left(\vec{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^M$$

Steps of determining $f(\vec{x})$ for kNN:

Another SL algorithm, k-Nearest Neighbours (kNN)

$$\mathcal{D} = \left\{ (\vec{x}^{(i)}, y^{(i)}) \right\}_{i=1}^M$$

Steps of determining $f(\vec{x})$ for kNN:

- Step #1: Calculate the distance $d(\vec{x}, \vec{x}^{(i)}) = \|\vec{x} - \vec{x}^{(i)}\|_2$ for all $\vec{x}^{(i)}$ in \mathcal{D}

◦ Step

- Step #2: Find the k closest points to \vec{x}
Store as $\vec{x}^{*(j)}$, and the corresponding
labels as $y^{*(j)}$ ($1 \leq j \leq k$)
- if classification:



◦ Step #2: Find the k closest points to \vec{x}
Store as $\vec{x}^{*(j)}$, and the corresponding
labels as $y^{*(j)}$ ($1 \leq j \leq k$)

◦ if classification: $f(\vec{x})$ is the majority member of $\{y^{*(j)}\}_{j=1}^M$
if regression: $f(\vec{x})$ is the average of $\{y^{*(j)}\}_{j=1}^M$

Let us take

$$\vec{\Delta W} = -\eta \vec{\nabla}_{\vec{W}} \mathcal{L}$$

⇒ update $W_i \rightarrow W_i - \eta \frac{\partial \mathcal{L}}{\partial W_i}$ each step

GD

$\eta > 0$ is a parameter called the "learning rate" that we can choose (a hyperparameter)

$$\Delta \mathcal{L} \approx -\eta \|\vec{\nabla}_{\vec{W}} \mathcal{L}\|_2 < 0$$

Notes about η

- if η is too small, it will take a long time to find the solution, can get "stuck" in local minima
- η must be small enough for the first-order approx. to be valid

Another SL algorithm, k-Nearest Neighbours (kNN)

$$\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^M$$



Steps of determining $f(\vec{x})$ for kNN

- Step #1: Calculate distance $d(\vec{x}, \vec{x}^{(i)}) = \|\vec{x} - \vec{x}^{(i)}\|_2$ for all $\vec{x}^{(i)} \in \mathcal{D}$

- Step #2: Find the k closest points to \vec{x}

Store as $\vec{x}^{*(j)}$, and the corresponding labels as $y^{*(j)}$ ($1 \leq j \leq k$)

- if classification: $f(\vec{x})$ is the majority member of $\{y^{*(j)}\}_{j=1}^k$
- if regression: $f(\vec{x})$ is the average of $\{y^{*(j)}\}_{j=1}^k$

y_1, y_2, \dots, y_n

- k is a hyperparameter
- no fitting parameters