ICT303 – Advanced Machine Learning and Artificial Intelligence

Topic 2: Linear Neural Networks

Hamid Laga
H.Laga@murdoch.edu.au
Office: 245.1.020

How to Get in Touch with the Teaching Team

- Internal and External Students
  - Email: H.Laga@murdoch.edu.au.
- Important
  - In any communication, please make sure that you
    - Start the subject of your email with ICT303
    - Include your student ID, name, and the lab slot in which you are enrolled.
  - We will do all our best to answer your queries within 24 hrs.

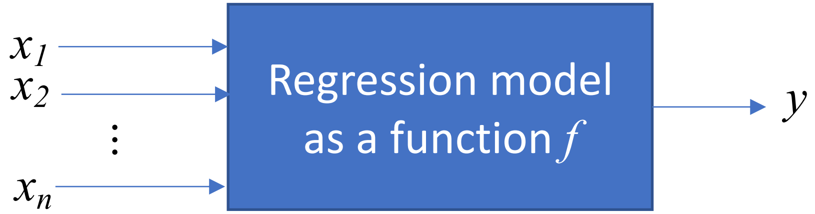
In this Lecture

- Linear Neural Networks
  - Linear Regression
  - Motivating examples
  - What is a LNN
  - Loss functions
  - Training as optimization
    - Closed-form formula
    - Gradient descent algorithm
    - Mini-batch stochastic gradient descent (SGD)
  - Summary

- Learning objectives
  - Understand the basic components of a machine learning model
  - Understand Linear regression and formulate it using artificial neurons
  - Implement linear regression using Python, NumPy and PyTorch

- Additional readings
  - Chapter 3 of the textbook, available at: https://d2l.ai/chapter_linear-regression/linear-regression.html

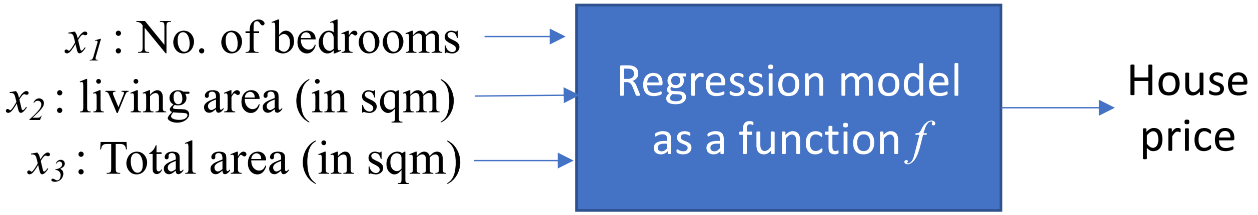
Regression Problems



- Regression is the task of predicting a numerical value from an input
- Common Examples
  - Predicting future (home, stock) prices
  - Predicting length of stay of a patient in hospital following an intervention
  - Forecasting demand (for a specific product)
- Regression model
  - A function that has input (parameters) and returns the predicted value based on the observed parameters

Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3



Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3

This makes 3 variables (x_1, x_2, x_3), which we can write as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3

This makes 3 variables (x_1, x_2, x_3), which we can write as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

- Assumption 2 - The price is a weighted sum of these three factors (the model)

$$y = \underbrace{\boxed{w_1}x_1 + \boxed{w_2}x_2 + \boxed{w_3}x_3}_{\text{Weights}} + \underbrace{\boxed{b}}_{\text{Bias}}$$

Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3

This makes 3 variables (x_1, x_2, x_3), which we can write as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

- Assumption 2 - The price is a weighted sum of these three factors (the model)

$$y = \boxed{w_1}x_1 + \boxed{w_2}x_2 + \boxed{w_3}x_3 + \boxed{b}$$

Weights

Bias

$$\underbrace{y = \begin{bmatrix} w_1 & w_2 & w_3 & b \end{bmatrix} \text{X} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}}_{\text{Vector representation}}$$

Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3

This makes 3 variables (x_1, x_2, x_3), which we can write as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

- Assumption 2 - The price is a weighted sum of these three factors (the model)

$$y = \boxed{w_1}x_1 + \boxed{w_2}x_2 + \boxed{w_3}x_3 + \boxed{b}$$

Weights

Bias

$$\Rightarrow \quad y = \begin{bmatrix} w_1 & w_2 & w_3 & b \end{bmatrix} \text{X} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \quad \Rightarrow \quad y = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b \end{bmatrix}^T \text{X} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}$$

Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3

This makes 3 variables (x_1, x_2, x_3), which we can write as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

- Assumption 2 - The price is a weighted sum of these three factors (the model)

$$y = \boxed{w_1}x_1 + \boxed{w_2}x_2 + \boxed{w_3}x_3 + \boxed{b}$$

Weights

Bias

$$\Rightarrow \quad y = \begin{bmatrix} w_1 & w_2 & w_3 & b \end{bmatrix} \text{X} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \quad \Rightarrow \quad y = \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b \end{bmatrix}^{T}}_{\text{W}} \text{X} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}}_{\text{X}}$$

Example – Predicting the Sale Price of a House

- Assumption 1 – Factors impacting the prices (Input)
  - No. of bedrooms → variable x_1
  - Living area (in sqm) → variable x_2
  - Total area (in sqm) → variable x_3

This makes 3 variables (x_1, x_2, x_3), which we can write as a column vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

- Assumption 2 - The price is a weighted sum of these three factors (the model)

$$y = \boxed{w_1}x_1 + \boxed{w_2}x_2 + \boxed{w_3}x_3 + \boxed{b}$$

Weights

Bias

$$y = \begin{bmatrix} w_1 & w_2 & w_3 & b \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \Rightarrow y = \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b \end{bmatrix}^T}_{W} \times \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}}_{X}$$

$$\boxed{y = W^T X = X^T W = \langle W, X \rangle = \langle X, W \rangle}$$

Linear Regression Model – General Formulation

- Given n-dimensional input

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

- The linear regression model has n-dimensional weights and a bias

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$$

- Its output is a weighted sum of the inputs

$$y = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

- Vectorized version

$$\boxed{y = W^TX = X^TW = \langle W, X \rangle = \langle X, W \rangle}$$

Linear Regression Model – General Formulation

- Given n-dimensional input

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

- The linear regression model has n-dimensional weights and a bias

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$$

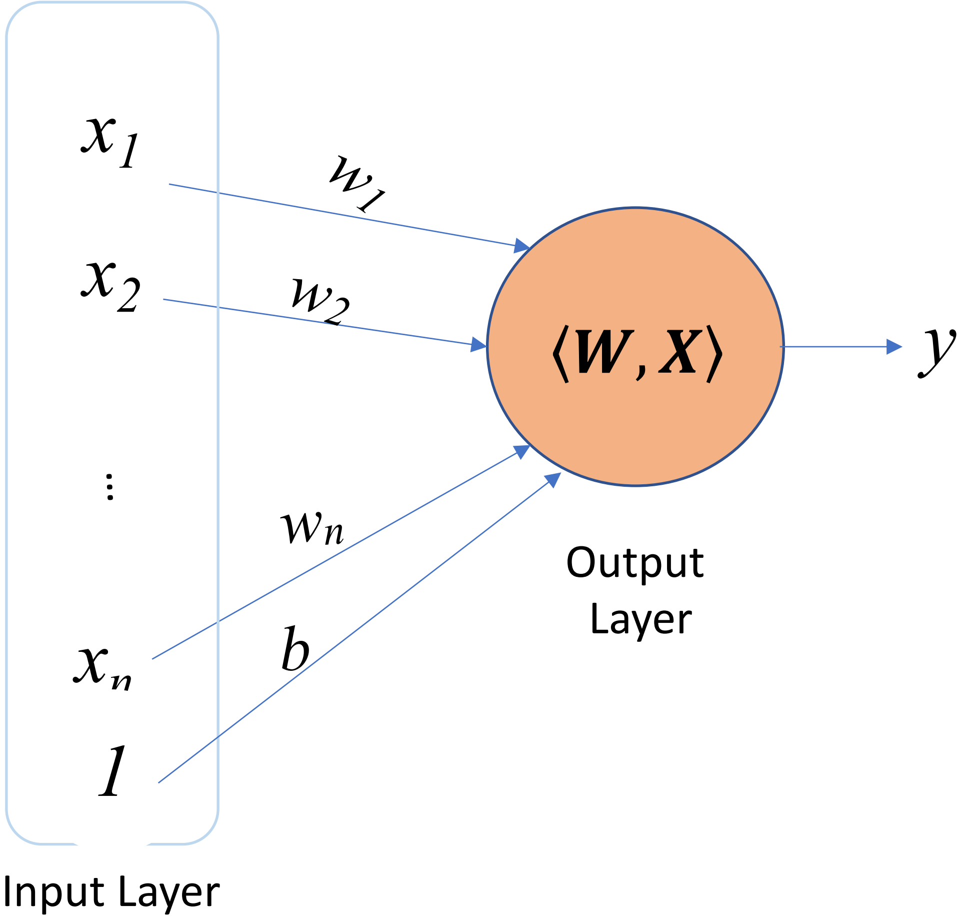
- Its output is a weighted sum of the inputs

$$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

- Vectorized version

$$\boxed{y = W^T X = X^T W = \langle W, X \rangle = \langle X, W \rangle}$$

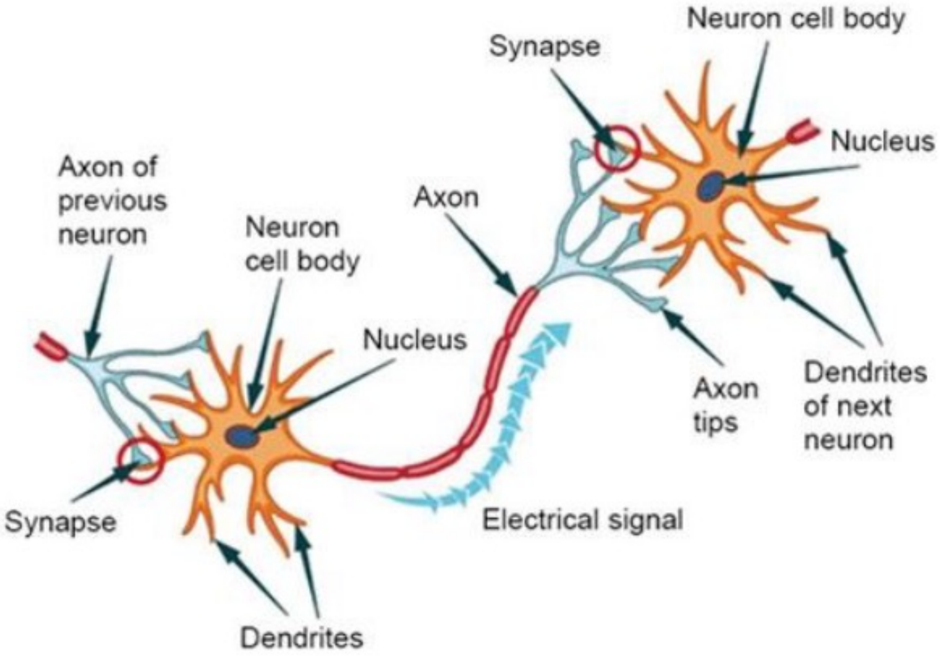
- Single layer Neural Network



We can stack multiple layers to get deep neural networks

Neural Networks Derived from Neuroscience

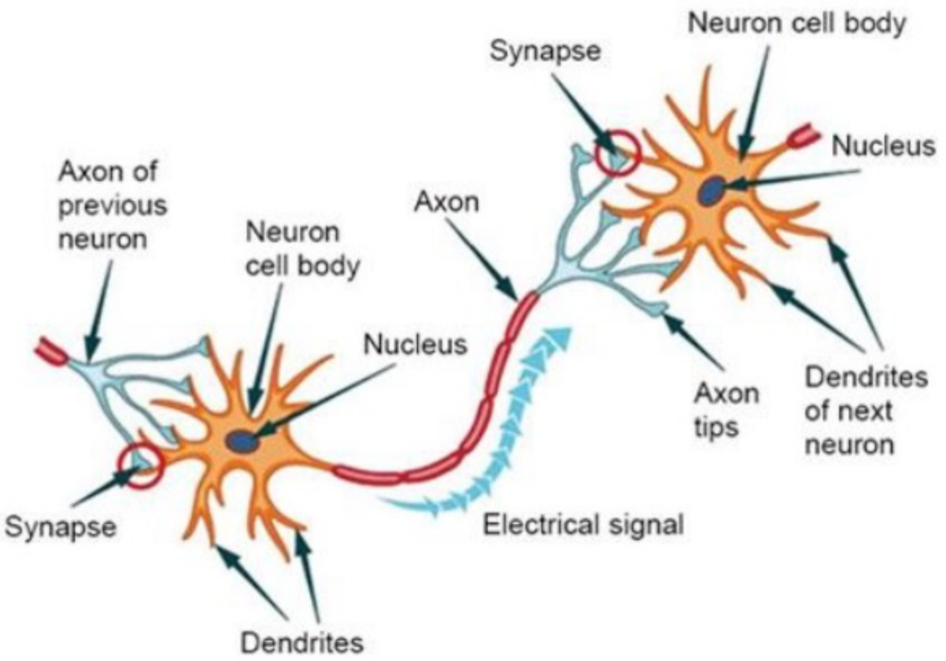
- The real neuron



https://deeplearning.cs.cmu.edu/S22/index.html

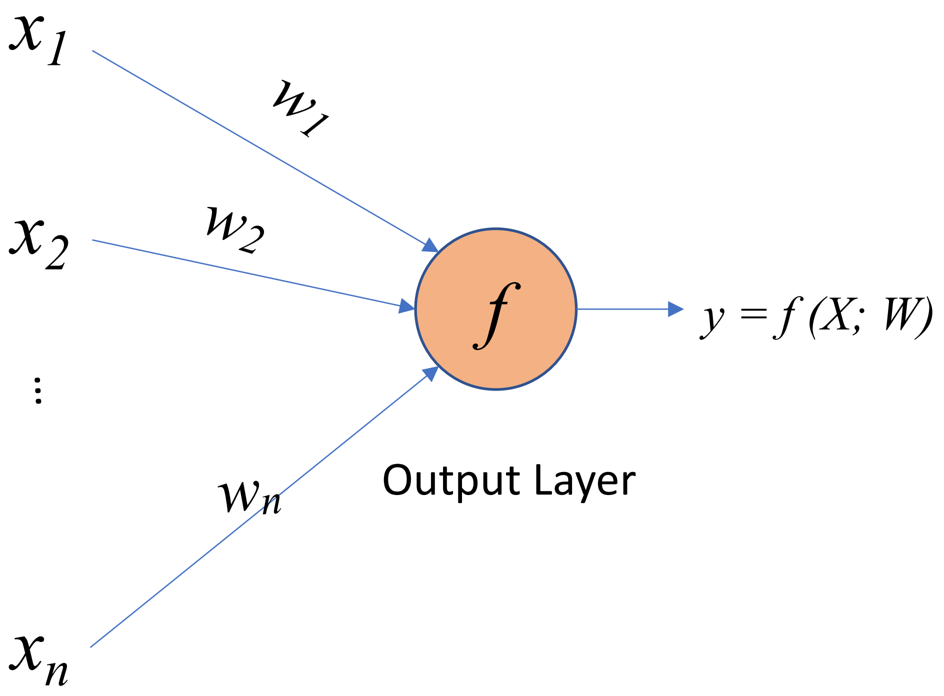
Neural Networks Derived from Neuroscience

- The real neuron



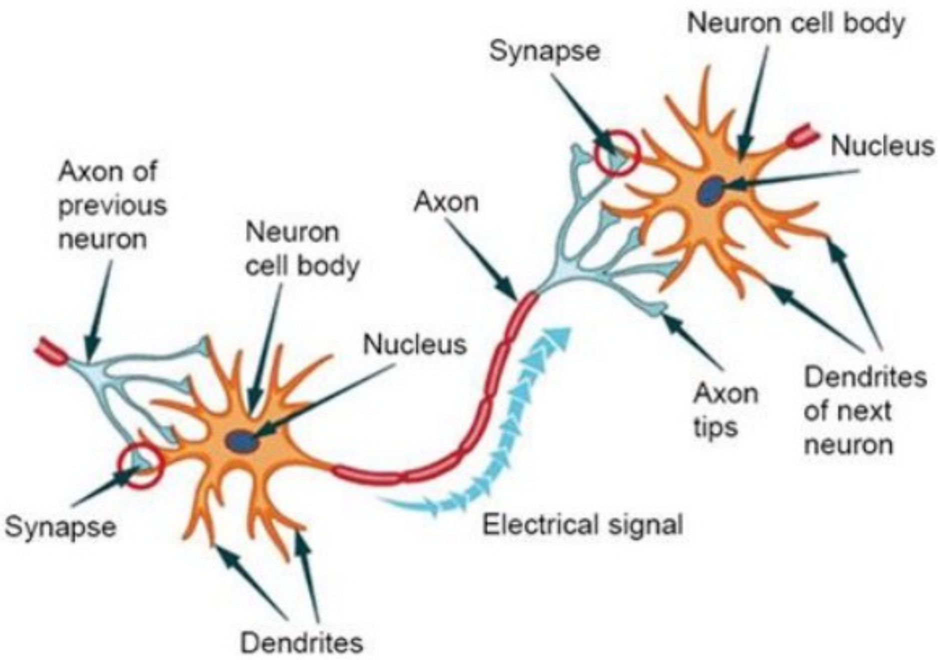
https://deeplearning.cs.cmu.edu/S22/index.html

- The artificial neuron



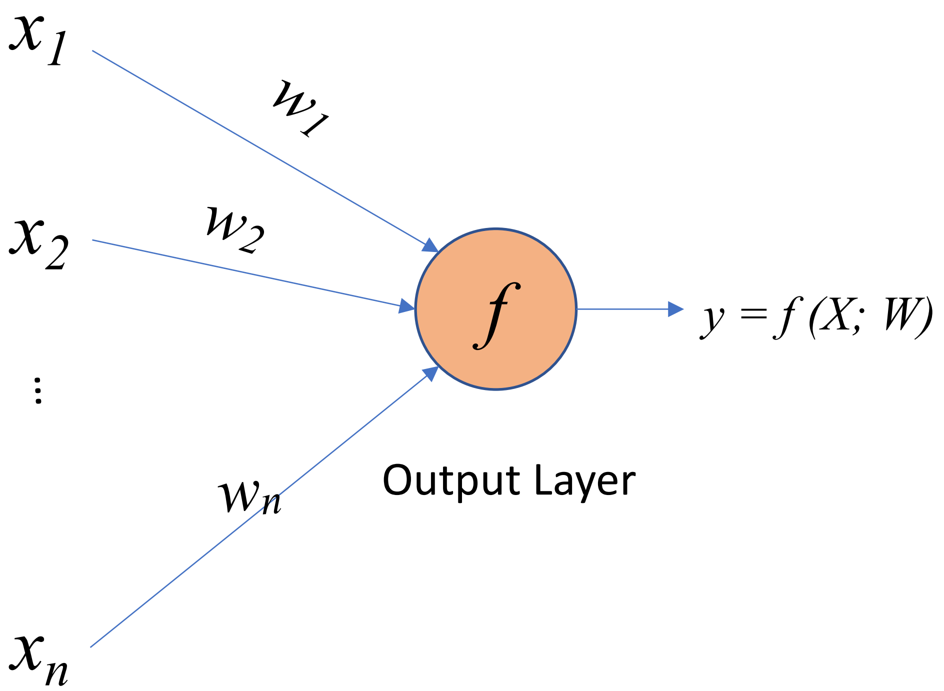
Neural Networks Derived from Neuroscience

- The real neuron



https://deeplearning.cs.cmu.edu/S22/index.html

- The artificial neuron



- For linear neurons

$$\boxed{y = f(X; W) = \langle W, X \rangle}$$

Linear Regression Model – Training (Learning)

- Problem
  - We know the model, which defines the relation between the input and output

$$y = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

$$y = \langle W, X \rangle$$

  - We, however, do not know the parameters or weights of the model

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$$

Linear Regression Model – Training (Learning)

- Problem
  - We know the model, which defines the relation between the input and output

$$y = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

$$y = \langle W, X \rangle$$

  - We, however, do not know the parameters or weights of the model

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$$

- Training (or learning)
  - The process of finding the best values for weights $w_1, w_2, \ldots, w_n$ and bias b from examples of input and their corresponding correct output

Linear Regression Model – Training (Learning) Process

- Step 1 – Collect training data, each sample is composed of

	No. Bedrooms x1	Land area x2	Living area x3	Sale price y
House 1	3	500	110	55000
House 2	2	350	80	45000
House 3	4	650	120	75000
…				
…				
…				
House n	3	400	102	52000

Input

$$\mathbf{X}_0 = [3, 500, 110, 1]^T,$$

$$\mathbf{X}_1 = [2, 350, 80, 1]^T,$$

$$\mathbf{X}_2 = [4, 650, 120, 1]^T,$$

…

$$\mathbf{X}_n = [3, 400, 102, 1]^T,$$

Desired output

$$y_0 = 55{,}000$$

$$y_1 = 45{,}000$$

$$y_2 = 75{,}000$$

$$y_n = 52{,}000$$

Linear Regression Model – Training (Learning) Process

- Step 2 – Training
  - Find **W** (which includes the weights and the bias **b**) so that when any of the training samples is fed to the model, it will produce a solution that is as close as possible to the ground-truth (desired) output
  - We need a measure of closeness – it is called the loss function

Linear Regression Model – Training (Learning)

- We need a measure of closeness
  - It is called training loss function, e.g., the difference between the output produced by the model and the desired output
  - Examples of loss functions: the L2 loss

$$L(X, y, W) = \frac{1}{n}\sum_{i=1}^{n}(\underbrace{y_i}_{\text{Desired output}} - \underbrace{\langle X_i, W\rangle}_{\text{Output of the model}})^2$$

Linear Regression Model – Training (Learning)

- We need a measure of closeness
  - It is called training loss function, e.g., the difference between the output produced by the model and the desired output
  - Examples of loss functions: the L2 loss

$$L(X, y, W) = \frac{1}{n}\sum_{i=1}^{n}(\underbrace{y_i}_{\text{Desired output}} - \underbrace{\langle X_i, W\rangle}_{\text{Output of the model}})^2$$

  - The optimal weights

$$W^* = \underset{W}{\operatorname{argmin}}\{L(X, y, W)\} = \underset{W}{\operatorname{argmin}}\{\tfrac{1}{n}\textstyle\sum_{i=1}^{n}(y_i - \langle X_i, W\rangle)^2\}$$

Linear Regression Model – Training (Learning) Process

- In the case, of linear regression

Input	Desired output
$\mathbf{X}_0 = [3, 500, 110, 1]^T,$	$y_0 = 55{,}000$
$\mathbf{X}_1 = [2, 350, 80, 1]^T,$	$y_1 = 45{,}000$
$\mathbf{X}_2 = [4, 650, 120, 1]^T,$	$y_2 = 75{,}000$
…	
$\mathbf{X}_n = [3, 400, 102, 1]^T,$	$y_n = 52{,}000$

$$\Rightarrow \begin{cases} \langle W, X_0 \rangle = y_0 \\ \langle W, X_1 \rangle = y_1 \\ \langle W, X_2 \rangle = y_2 \\ \quad \dots \\ \langle W, X_n \rangle = y_n \end{cases} \Rightarrow \begin{cases} X_0^T W = y_0 \\ X_1^T W = y_1 \\ X_2^T W = y_2 \\ \quad \dots \\ X_n^T W = y_n \end{cases} \Rightarrow \overbrace{(X_0 | X_1 | X_2 | \dots | X_n | 1)}^{X}{}^T W = Y$$

$$\Downarrow$$

$$X^T W = Y$$

Linear Regression Model – Training (Learning) Process

- In the case, of linear regression
  - It is the process of estimating W such that

$$X^T W = Y$$

  - X is a matrix where each column corresponds to an input sample X_i
  - Y is a column vector where the i-th element y_i is the desired (groundtruth) output for X_i

- The solution has a closed-form (see proof in the supplementary slides as well as the textbook)

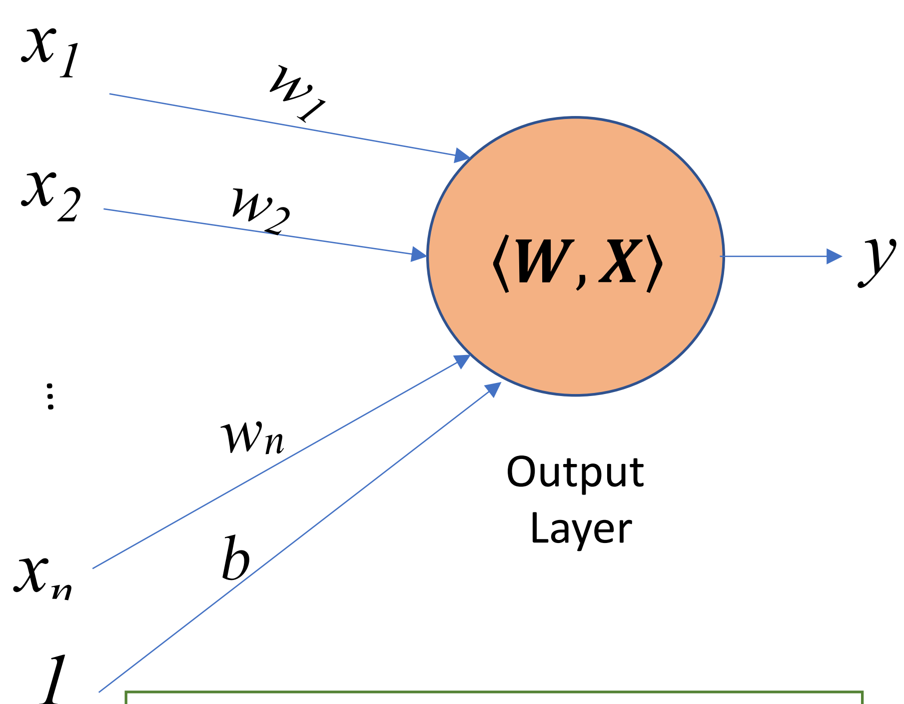
Linear Regression Model – Training (Learning) Process

- Algorithm
  - Collect the input training data (as column vectors) and stack them into a X
    - Each column of X is one input vector. The last element of each column is value 1
  - Collect the desired output for each of the input and put them all into a vector Y
  - The optimal weights, i.e., the weights that minimize the loss function are given by the following equation:

$$\boxed{\mathbf{W}^* = \left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{Y}}$$

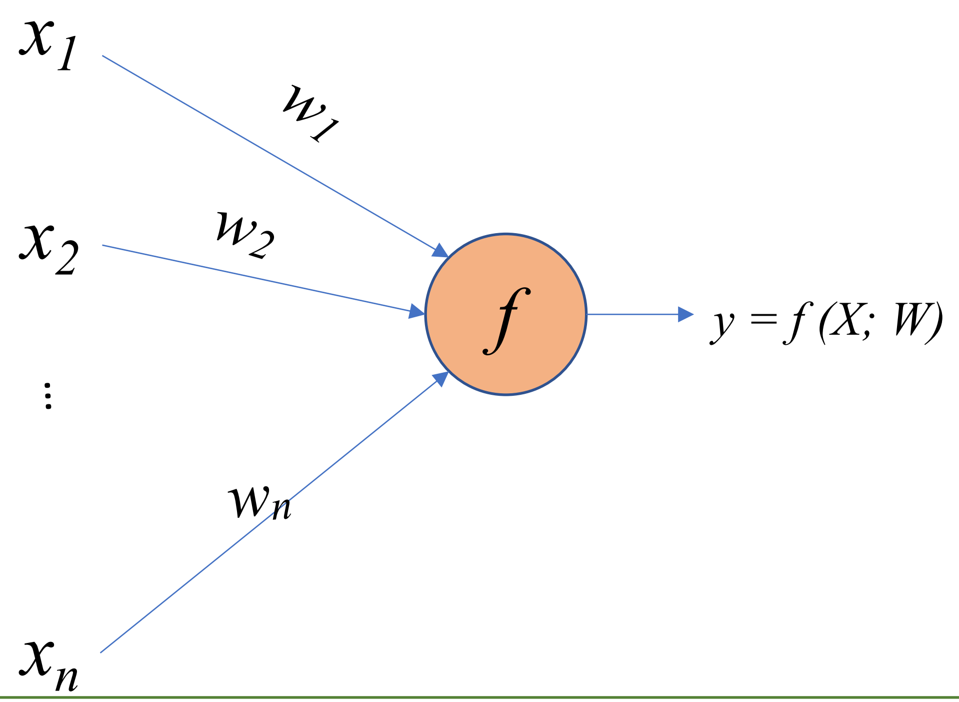
Training in general

- Linear regression



$$\mathbf{W}^* = \left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{Y}$$

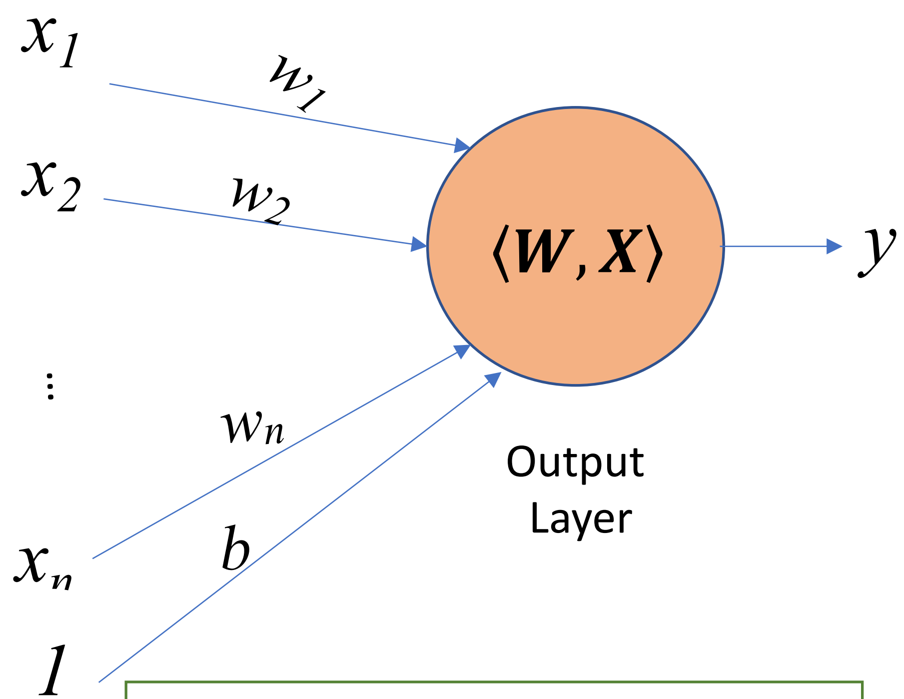
- The general case



$$W^* = \underset{W}{\operatorname{argmin}}\left\{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f\,(X_i; W)\,\right)^2\right\}$$

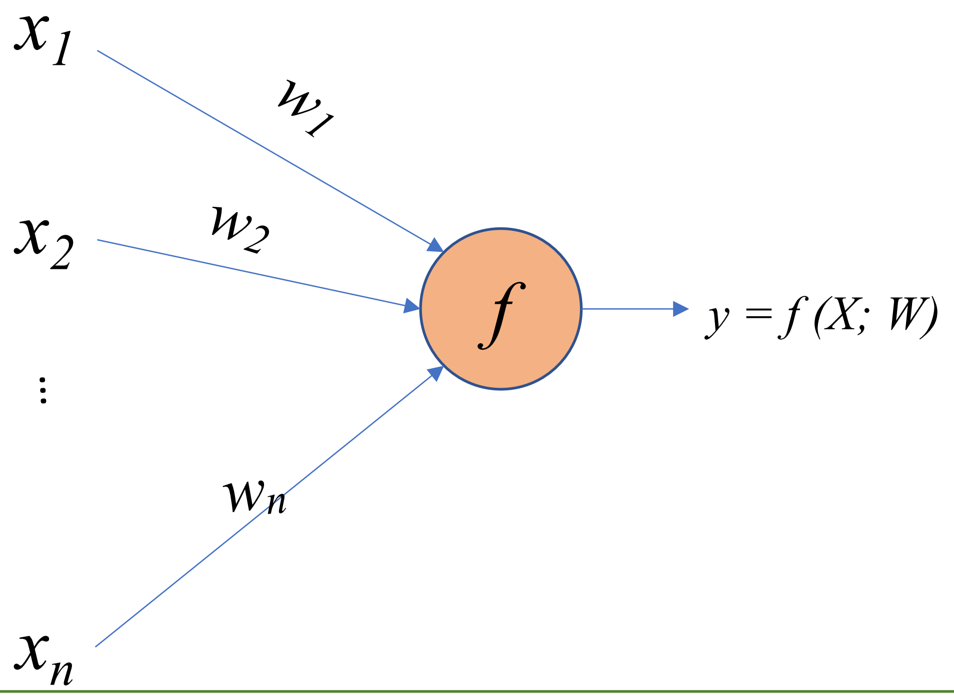
Training in general

- Linear regression



$$\mathbf{W}^* = \left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{Y}$$

- The general case

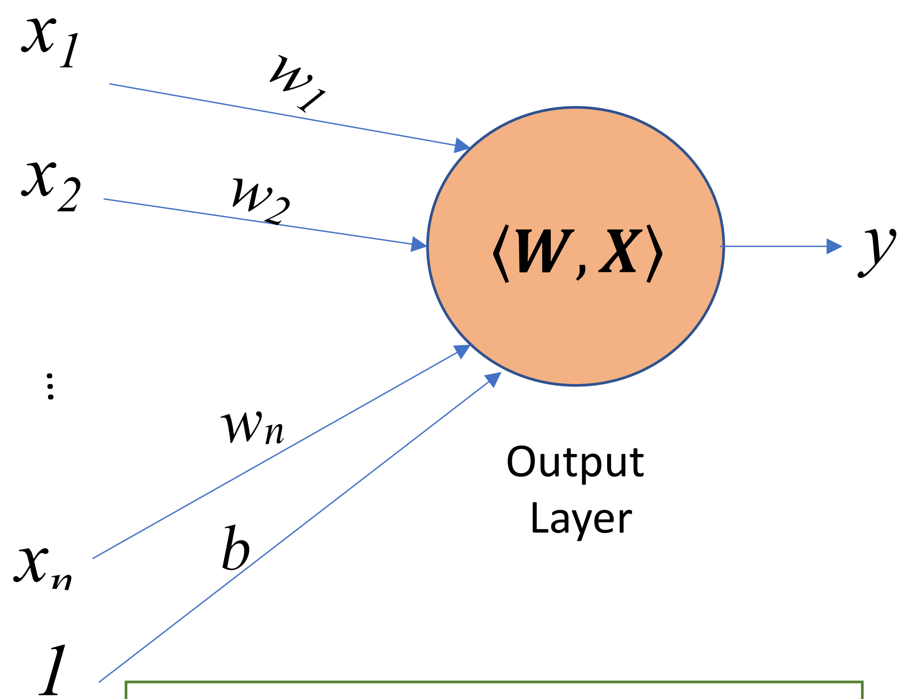


$$W^* = \underset{W}{\operatorname{argmin}}\{\tfrac{1}{n}\sum_{i=1}^{n}(y_i - f(X_i; W))^2\}$$

The solution has no analytical form

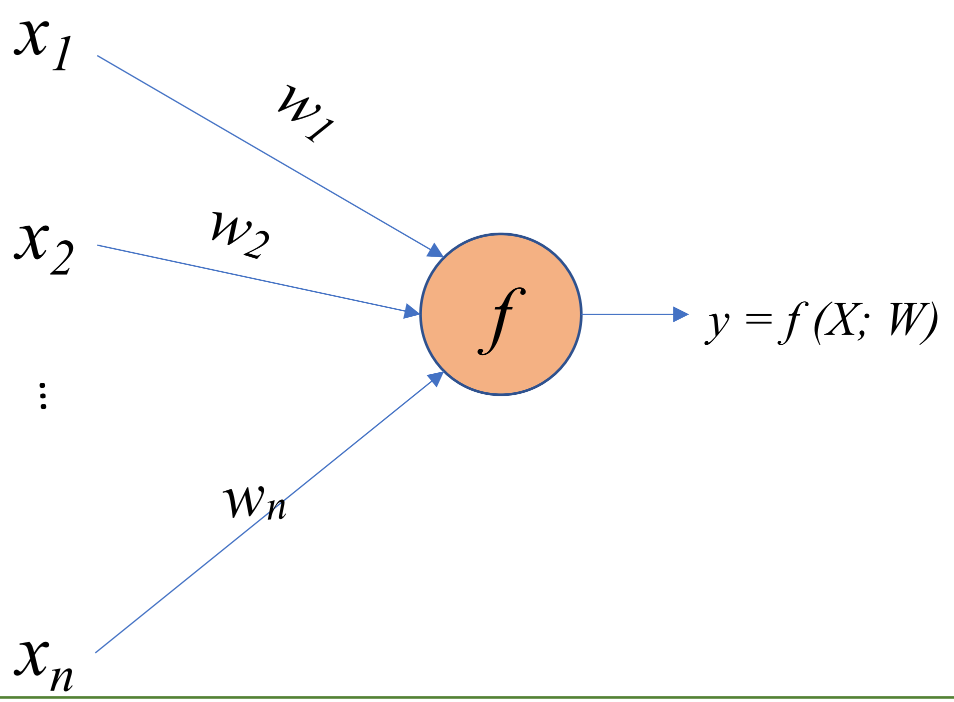
Training in general

- Linear regression



$$\mathbf{W}^* = \left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{Y}$$

- The general case



$$W^* = \underset{W}{\operatorname{argmin}}\{\tfrac{1}{n}\textstyle\sum_{i=1}^{n} L(y_i, f(X_i; W)) \;\}$$

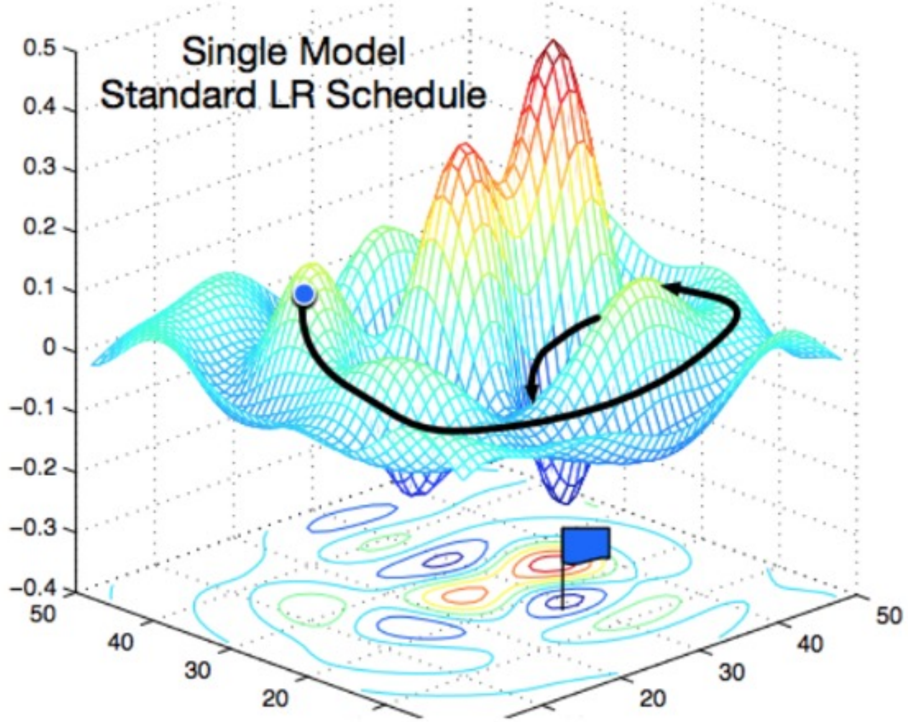
The solution has no analytical form

In general, Training is an Optimization Process

- You are given a loss function that depends on many parameters (variables)
  - Find the values of the variables for which the function takes the minimal value

- When the function is simple, e.g., linear regression model
  - The solution has a closed-form formula

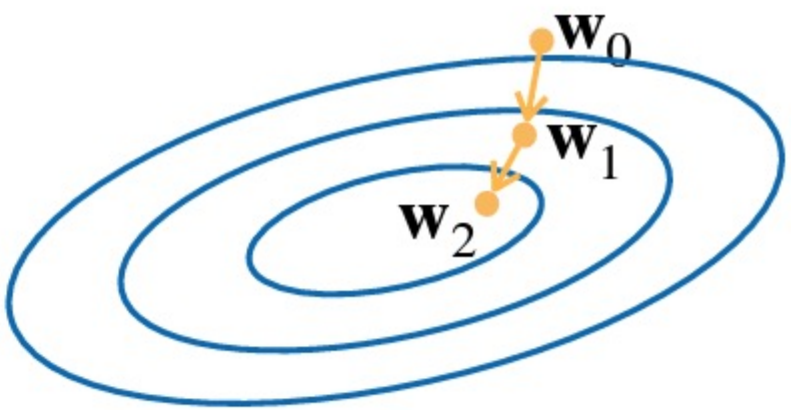
- Often, we are dealing with complex functions
  - Need to find the optimal solution using a numerical algorithm



https://c.d2l.ai/berkeley-stat-157/slides/1_31/4-Linear-Methods.pdf

Optimization – The Gradient Descent Algorithm

- Iteratively reduce the error by updating the parameters in the direction that lowers the loss function
  - Choose initial values w0 for the weights (e.g., at random)
  - Repeat (for t = 1, 2, …)
    - Evaluate the Gradient of the loss function at w_{t-1} with respect to the weights $\frac{\partial \ell}{\partial \mathbf{w}_{t-1}}$
      - This gives a direction the reduces the value of the loss function
    - Update the weights with a fraction of the computed gradient



$$\underbrace{\mathbf{w}_t}_{\text{New values}} = \underbrace{\mathbf{w}_{t-1}}_{\text{Old values}} - \eta \boxed{\frac{\partial \ell}{\partial \mathbf{w}_{t-1}}}$$

The gradient

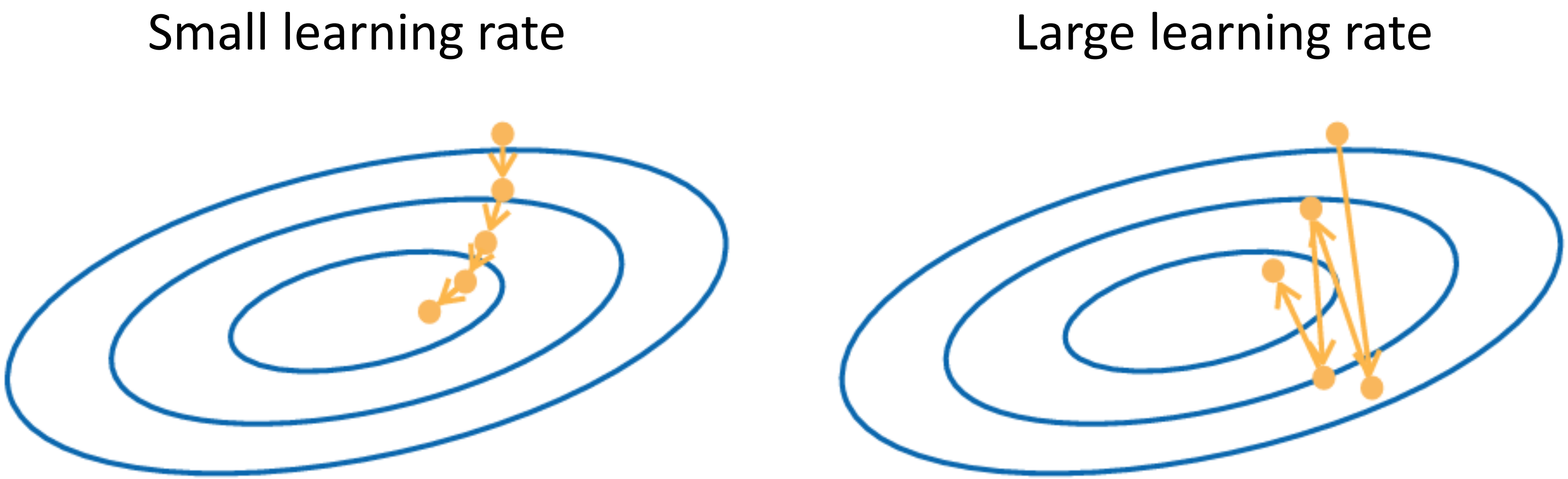
Learning rate (a positive scalar)

  - Until the change in the weights is insignificant

https://c.d2l.ai/berkeley-stat-157/slides/1_31/4-Linear-Methods.pdf

Optimization – Gradient Descent Algorithm

- Choosing the Learning Rate



Accurate but can be very slow

Fast, but can lead to incorrect solution

https://c.d2l.ai/berkeley-stat-157/slides/1_31/4-Linear-Methods.pdf

Optimization – Mini-batch Stochastic Gradient Descent (SGD)

- Computing the gradient over the whole training data is too expensive
  - Takes minutes to hours for Deep Neural Network (DNN) models
- Solution: Use Mini-batch Stochastic Gradient Descent (SGD)
  - Partition the training data set into batches, each batch is of size b
  - For i=1 to max number of epochs
    - For every batch
      - Evaluate the Gradient of the loss function with respect to the weights
      - Update the weights with a fraction of the computed gradient
  - At each epoch, the algorithm goes through all the batches, i.e., every training data point is visited once

https://c.d2l.ai/berkeley-stat-157/slides/1_31/4-Linear-Methods.pdf

Optimization – Mini-batch Stochastic Gradient Descent (SGD)

- Computing the gradient over the whole training data is too expensive
  - Takes minutes to hours for Deep Neural Network (DNN) models
- Solution: Use Mini-batch Stochastic Gradient Descent (SGD)
  - Partition the training data set into batches, each batch is of size b
  - For i=1 to max number of epochs
    - For every batch
      - Evaluate the Gradient of the loss function at with respect to the weights
      - Update the weights with a fraction of the computed gradient
  - At each epoch, the algorithm goes through all the batches, i.e., every training data point is visited once
- Choosing the batch size
  - If too small
    - workload is too small, hard to fully utilize the computation resources
  - If too big
    - Memory issue, waste computation (e.g., when all xi are identical)

https://c.d2l.ai/berkeley-stat-157/slides/1_31/4-Linear-Methods.pdf

Summary – Linear Regression

- Problem
  - Estimate a real value from some observations (e.g., how much)
- Model
  - The relation between the output and the input is linear $y = \langle W, X \rangle$
- Training data
  - Examples of input and their corresponding output
- Loss function
  - A measure of discrepancy between the actual output and the desired output

$$L(X, y, W) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \langle X_i, W \rangle)^2$$

- Learning (optimization) algorithm – mini-batch Stochastic Gradient Descent
  - Choose a starting point (initialize the values of the weights w and b
  - Repeat
    - Compute gradient
    - Update the parameters
  - Until convergence (the change is insignificant)

This Week's Lab

- Implementation
  - Naïve implementation
  - Concise implementation as a neuron that requires training …

Next Week

- Classification problems
  - Answer questions such as “which category …”, e.g.,
    - Should I put this email to the inbox or to the “spam folder”?
    - Does the image depict a “dog”, a “horse”, a “car”, ….
- Non-linear model and Multilayer Perceptron (MLP)

Questions