STATISTICAL MACHINE LEARNING

COMP4670/8600
2025 Semester 1, Week 7, Lecture 1

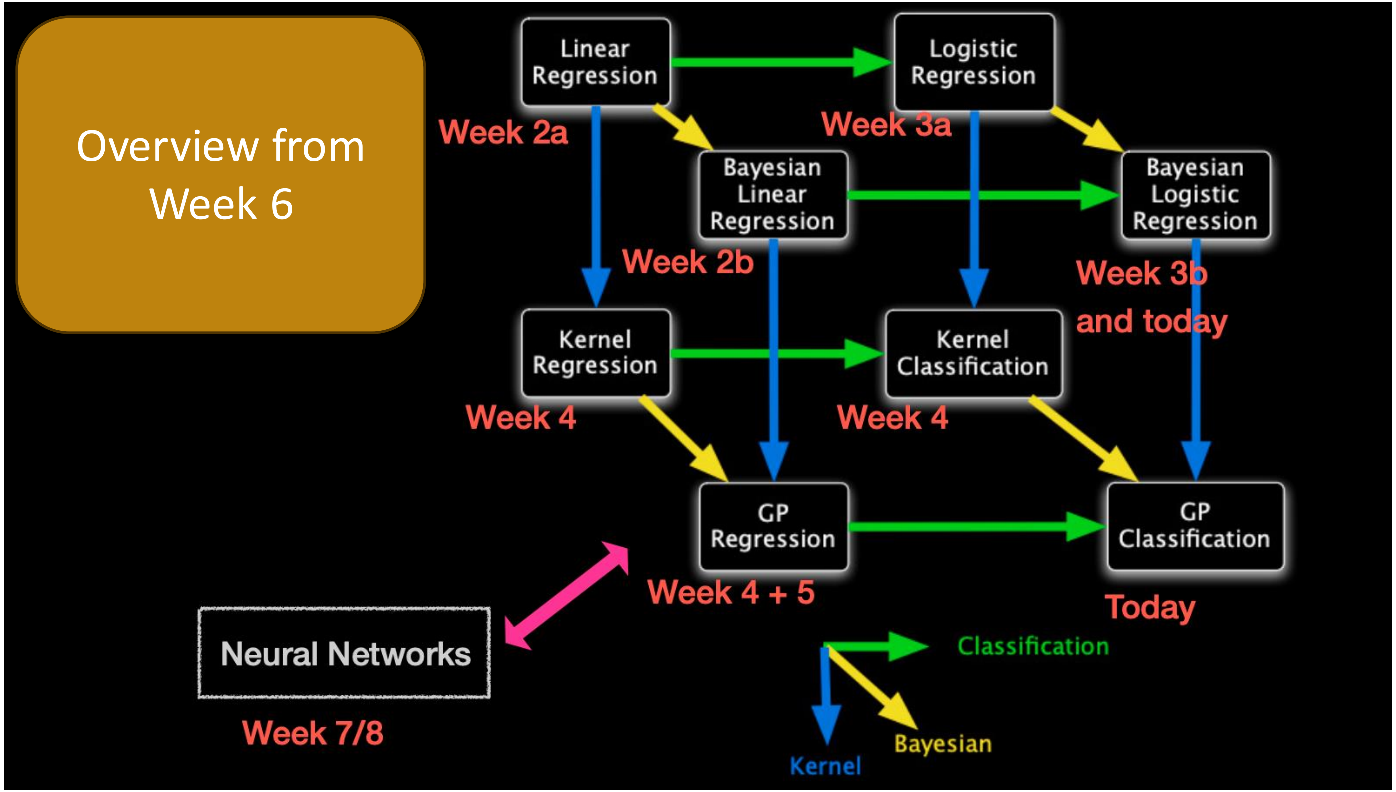
Jing Jiang
School of Computing

Australian National University

Administrative Matters

- Assignment 2 will be released at the end of Week 7
  - Due at the end of Week 12

First Half of the Semester



Overview of the Second Half

- Topics to be covered from Week 7 to Week 11
  - Neural networks (Week 7, Week 8a)
  - Mixture models (Week 8b)
  - Sampling (Week 9)
  - Graphical models (Week 10, Week 11)
- Course review in Week 12

Agenda

- Motivation
- Elements of a neural network
- Why deep neural networks?
- Forward pass → making predictions
- Backward pass or backpropagation → update model parameters

Success Stories of Neural Networks

Image Recognition

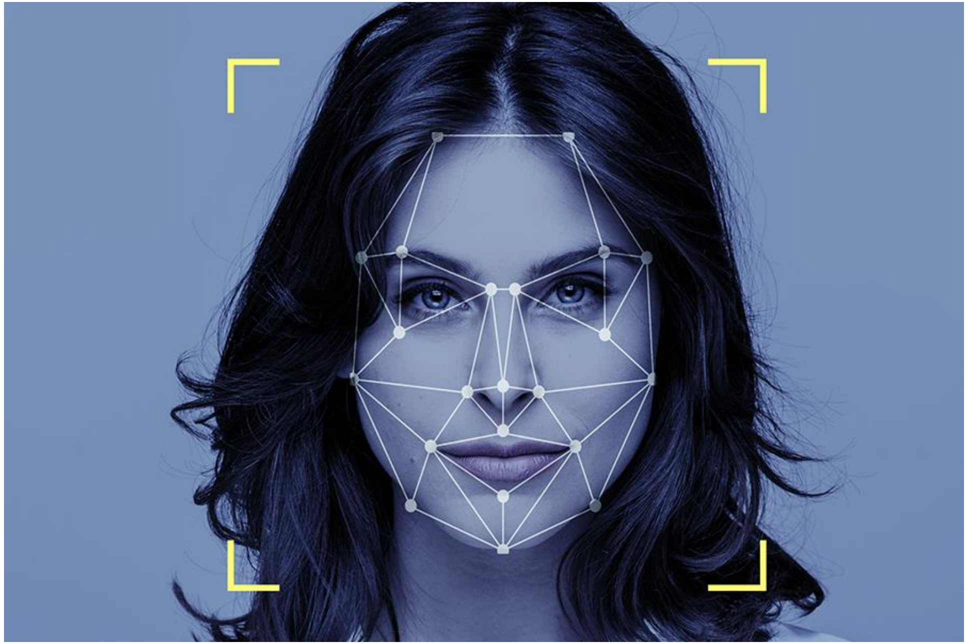
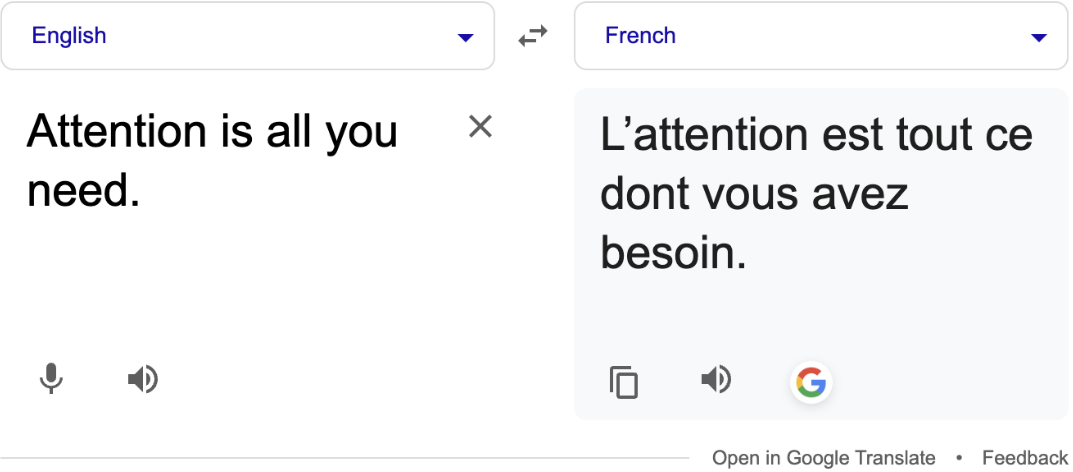


Image from https://news.mit.edu/2022/optimized-solution-face-recognition-0406

Image Generation

Image from https://www.tomorrowsworldtoday.com/art/worlds-first-ai-generated-art-gallery-opens/

Machine Translation



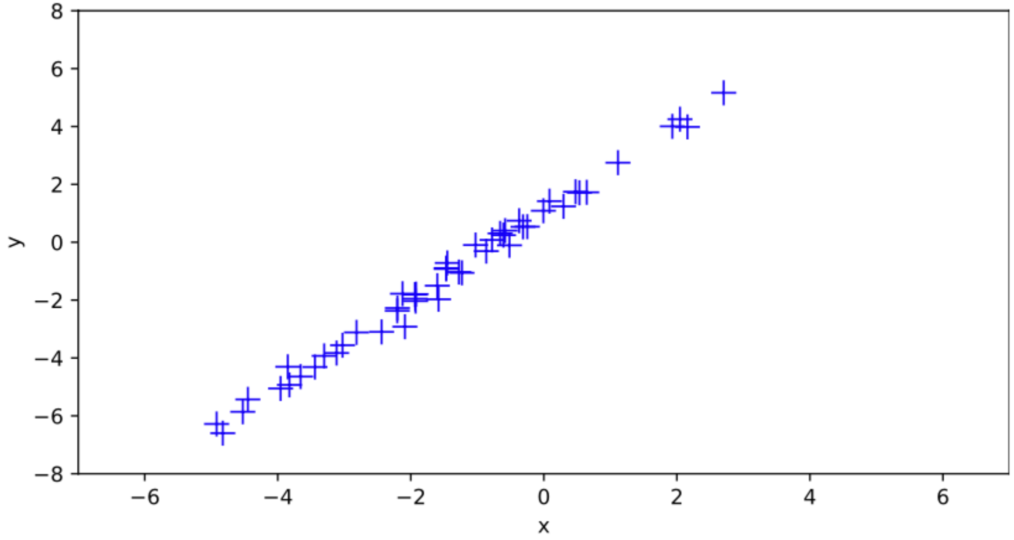
Chatbots

Why Neural Networks?

A Motivating Example

- Recall regression problems

Given a set of data points $\{x_n, y_n\}_{n=1}^{N}$:



Linear regression assumes:

$$y_n = f(x_n) + \epsilon_n = w x_n + b + \epsilon_n$$

A Motivating Example

Linear regression assumes:

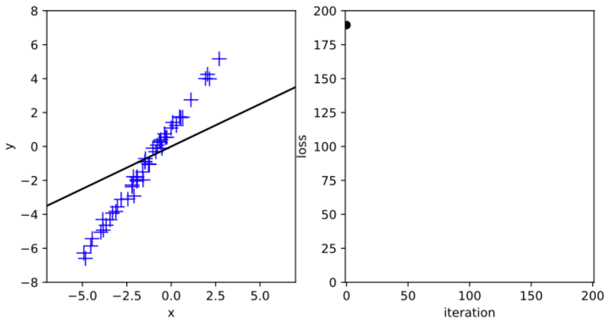
$$y_n = f(x_n) + \epsilon_n = wx_n + b + \epsilon_n$$

- Standard MSE loss

$$\text{loss}(w, b) = \sum_{n=1}^{N} (y_n - wx_n - b)^2$$

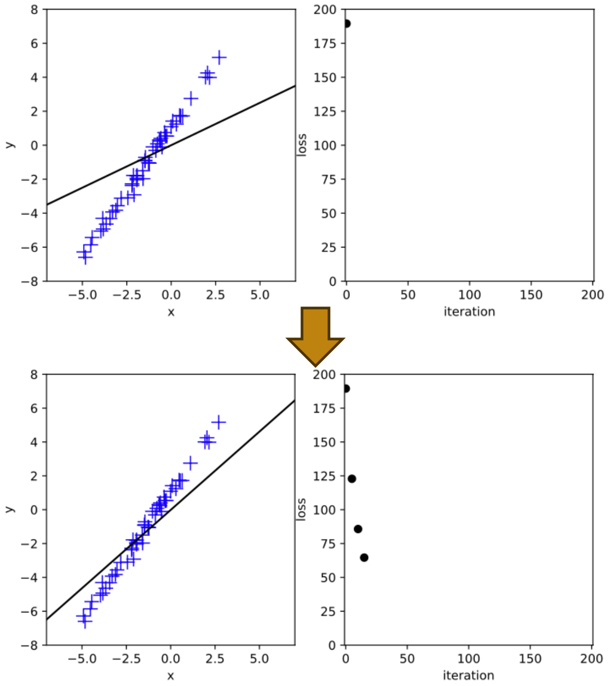
A Motivating Example

- Optimise the loss with gradient descent, we get



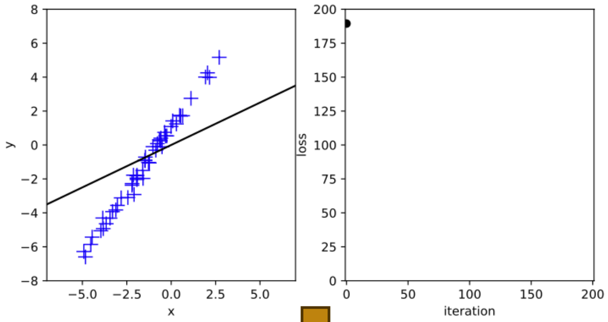
A Motivating Example

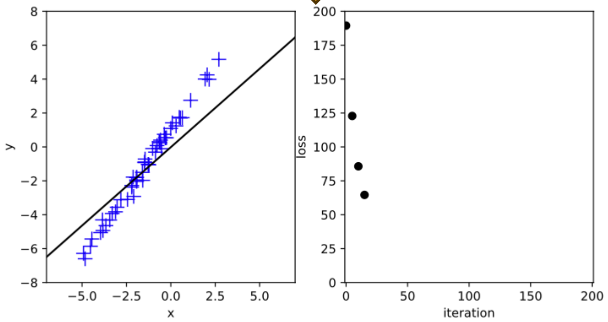
- Optimise the loss with gradient descent, we get

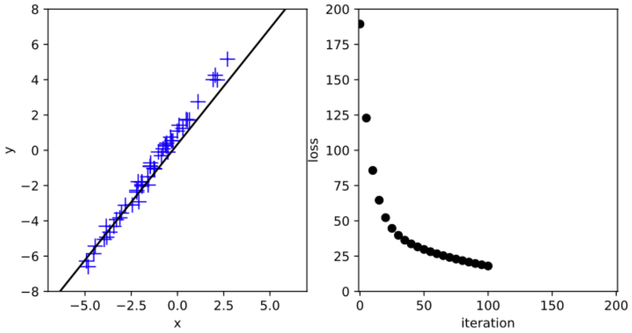


A Motivating Example

- Optimise the loss with gradient descent, we get

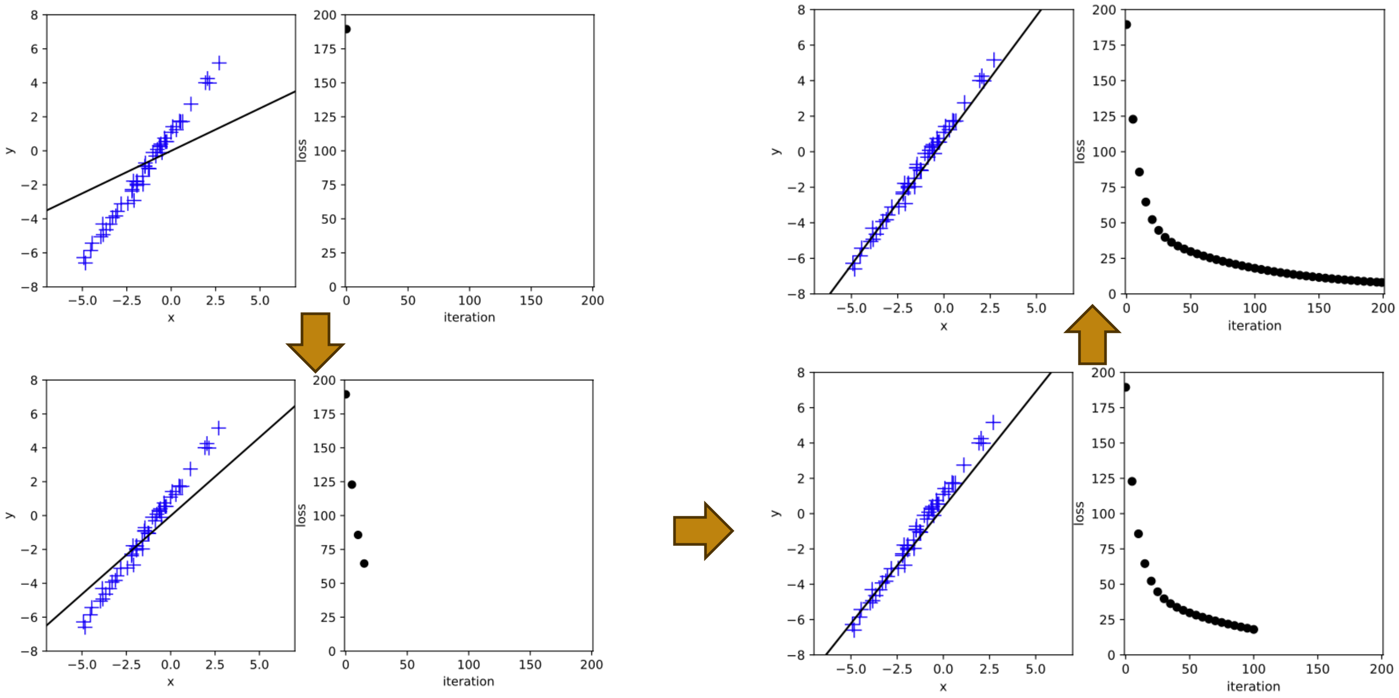






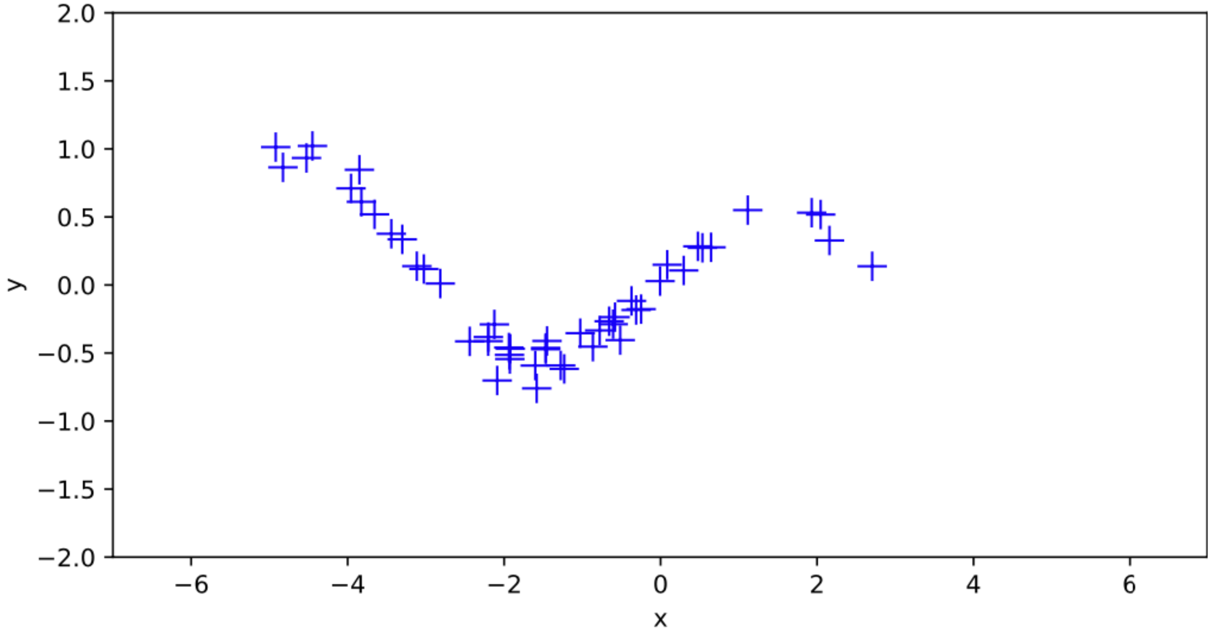
A Motivating Example

- Optimise the loss with gradient descent, we get



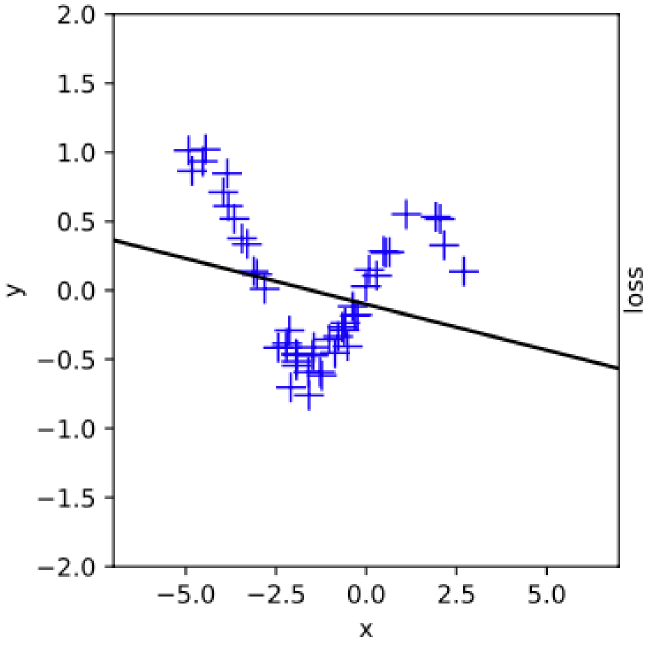
A Motivating Example

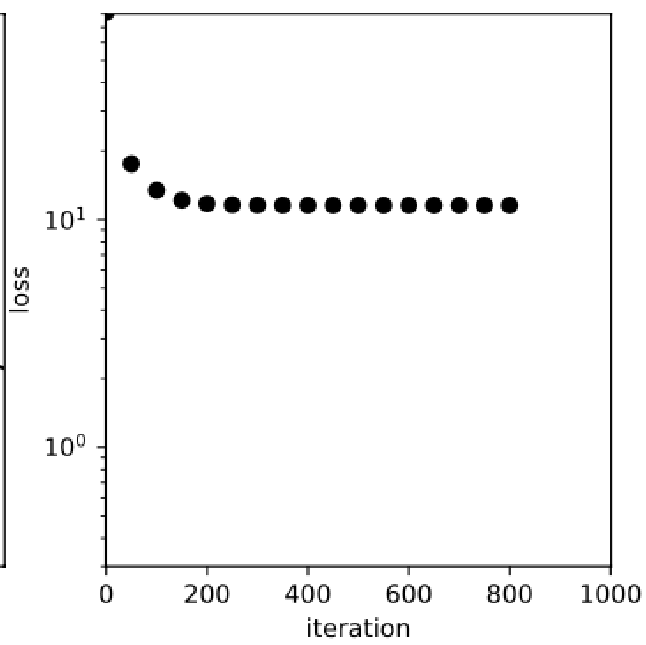
- What if we are trying to fit the following curve?



A Motivating Example

- Linear regression clearly would not work





A Motivating Example

- Let's try to add non-linear features

Let's add x^2:

$$y_n = f(x_n) + \epsilon_n = \beta_1 x_n + \beta_2 x_n^2 + \beta_0 + \epsilon_n,$$

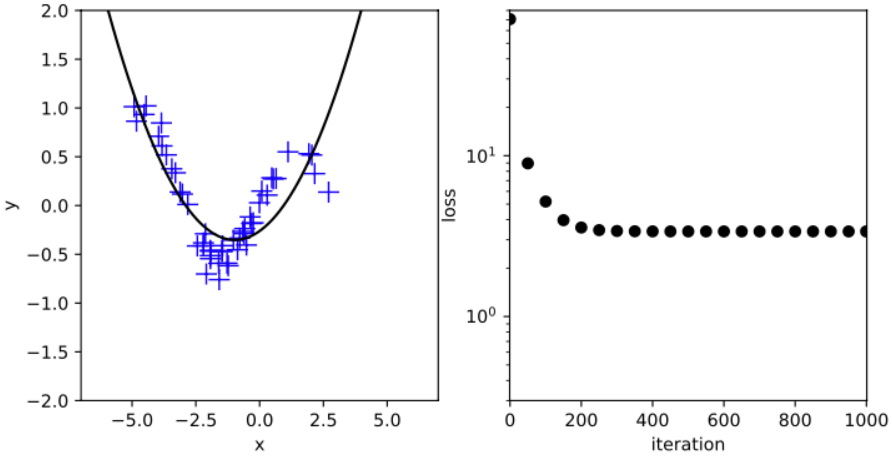
$$\text{loss}(\beta_1, \beta_2, \beta_0) = \sum_{n=1}^{N} (y_n - \beta_1 x_n - \beta_2 x_n^2 - \beta_0)^2,$$

A Motivating Example

- Better than linear regression but not ideal

$$y_n = f(x_n) + \epsilon_n = \beta_1 x_n + \beta_2 x_n^2 + \beta_0 + \epsilon_n,$$

$$\text{loss}(\beta_1, \beta_2, \beta_0) = \sum_{n=1}^{N} (y_n - \beta_1 x_n - \beta_2 x_n^2 - \beta_0)^2,$$



A Motivating Example

- Let's try to add more non-linear features

Let's add x^2 and $\sin(x)$:

$$y_n = f(x_n) + \epsilon_n = \beta_1 x_n + \beta_2 x_n^2 + \beta_3 \sin(x_n) + \beta_0 + \epsilon_n,$$

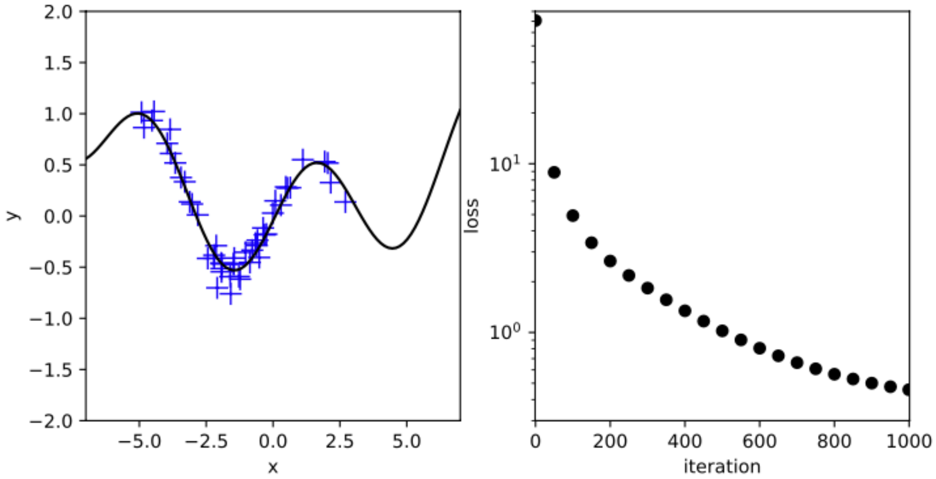
$$\text{loss}(\beta_1, \beta_2, \beta_3, \beta_0) = \sum_{n=1}^{N} (y_n - \beta_1 x_n - \beta_2 x_n^2 - \beta_3 \sin(x_n) - \beta_0)^2,$$

A Motivating Example

- Much better

$$y_n = f(x_n) + \epsilon_n = \beta_1 x_n + \beta_2 x_n^2 + \beta_3 \sin(x_n) + \beta_0 + \epsilon_n,$$

$$\text{loss}(\beta_1, \beta_2, \beta_3, \beta_0) = \sum_{n=1}^{N} (y_n - \beta_1 x_n - \beta_2 x_n^2 - \beta_3 \sin(x_n) - \beta_0)^2,$$



What does this example show?

Recipe for what we have done:

- try a linear regression model
- if doesn't work well, **manually** select and add more features (x^2, $\sin(x)$).
- try again until getting a satisfactory prediction accuracy!

What does this example show?

Recipe for what we have done:

- try a linear regression model
- if doesn't work well, **manually** select and add more features (x^2, $\sin(x)$).
- try again until getting a satisfactory prediction accuracy!

But:

- How to select these non-linear terms? What about $\sqrt{x}$, x^3, $\exp(x)$...?
- When should they be added?

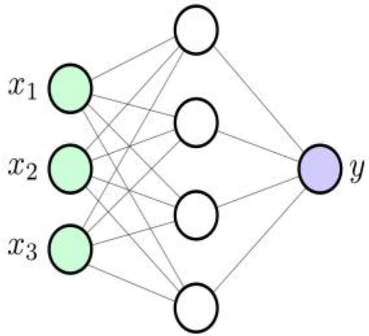
Requires domain-knowledge and **time-consuming** and **error-prone** trial-and-errors!

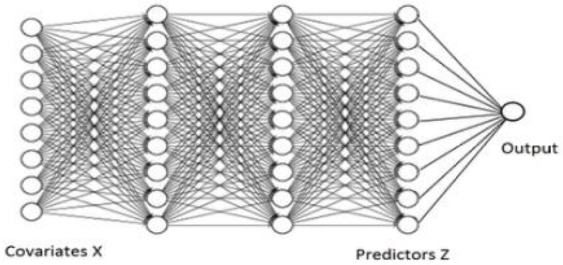
What does this example show?

- Ideally, we would like to be able to ***automatically*** learn a set of features $\phi_d(x)$

$$y_n = f(x_n) + \epsilon_n = \sum_{d=1}^{D} \beta_d \phi_d(x_n) + \beta_0 + \epsilon_n$$

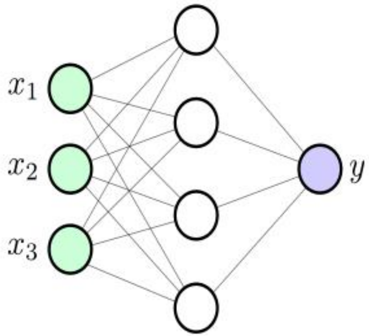
Neural Networks (NNs) provide the solutions

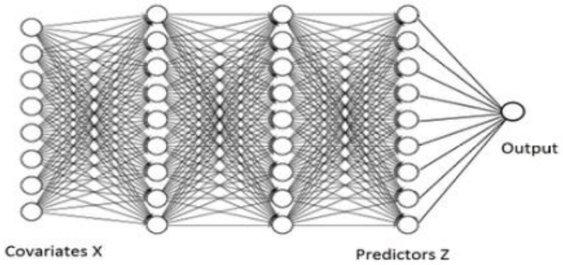




- A neural network is an interconnected assembly of simple processing units (a.k.a. **neurons**), which communicate by sending signals to each other through **weighted connections**.

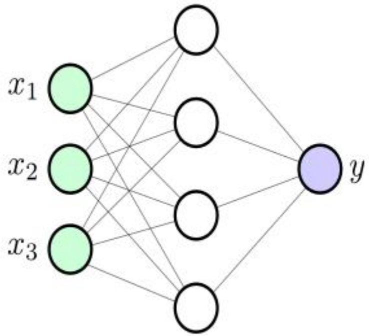
Neural Networks (NNs) provide the solutions

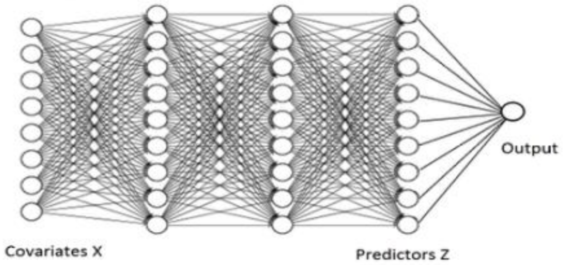




- A neural network is an interconnected assembly of simple processing units (a.k.a. **neurons**), which communicate by sending signals to each other through **weighted connections**.
- A neural network is made of **layers** of neurons: an input layer, (one or many) hidden layers, and an output layer.

Neural Networks (NNs) provide the solutions





- A neural network is an interconnected assembly of simple processing units (a.k.a. **neurons**), which communicate by sending signals to each other through **weighted connections**.
- A neural network is made of **layers** of neurons: an input layer, (one or many) hidden layers, and an output layer.
- A neural network is said to be **deep** if it has many hidden layers. → **Deep Learning**

A few more notes about neural networks

- Neural networks are sometimes called artificial neural networks because they are inspired by the network of neurons in the human brain.
  - However, much of the computation done by modern neural networks is not biologically plausible.

A few more notes about neural networks

- Neural networks are sometimes called artificial neural networks because they are inspired by the network of neurons in the human brain.
  - However, much of the computation done by modern neural networks is not biologically plausible.
- Variants of neural networks:

A few more notes about neural networks

- Neural networks are sometimes called artificial neural networks because they are inspired by the network of neurons in the human brain.
  - However, much of the computation done by modern neural networks is not biologically plausible.
- Variants of neural networks:
  - The network you have seen just now is called a **feedforward** or **fully-connected** neural network, which are most suitable for cross-sectional data or tabular data. (E.g., given a person's age, gender, income, etc., predict the person's monthly expenses.)

A few more notes about neural networks

- Neural networks are sometimes called artificial neural networks because they are inspired by the network of neurons in the human brain.
  - However, much of the computation done by modern neural networks is not biologically plausible.
- Variants of neural networks:
  - The network you have seen just now is called a **feedforward** or **fully-connected** neural network, which are most suitable for cross-sectional data or tabular data. (E.g., given a person’s age, gender, income, etc., predict the person’s monthly expenses.)
  - **Recurrent** neural networks and attention-based networks such as transformers are most suitable for time series and sequences.

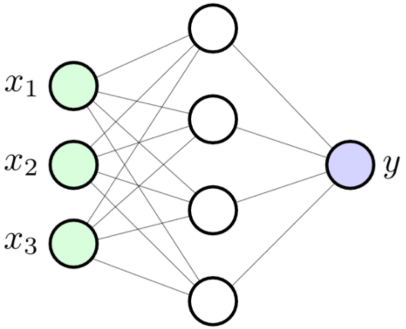
A few more notes about neural networks

- Neural networks are sometimes called artificial neural networks because they are inspired by the network of neurons in the human brain.
  - However, much of the computation done by modern neural networks is not biologically plausible.
- Variants of neural networks:
  - The network you have seen just now is called a **feedforward** or **fully-connected** neural network, which are most suitable for cross-sectional data or tabular data. (E.g., given a person’s age, gender, income, etc., predict the person’s monthly expenses.)
  - **Recurrent** neural networks and attention-based networks such as transformers are most suitable for time series and sequences.
  - **Convolutional** neural networks are most suitable for images and videos.

Agenda

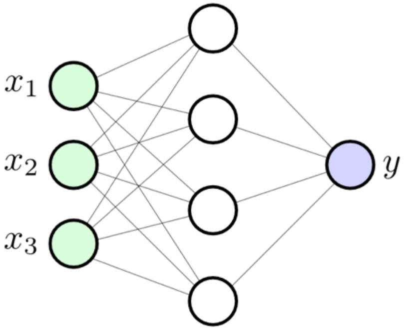
- Motivation
- **Elements of a neural network**
- Why deep neural networks?
- Forward pass → making predictions
- Backward pass or backpropagation → update model parameters

Terminologies



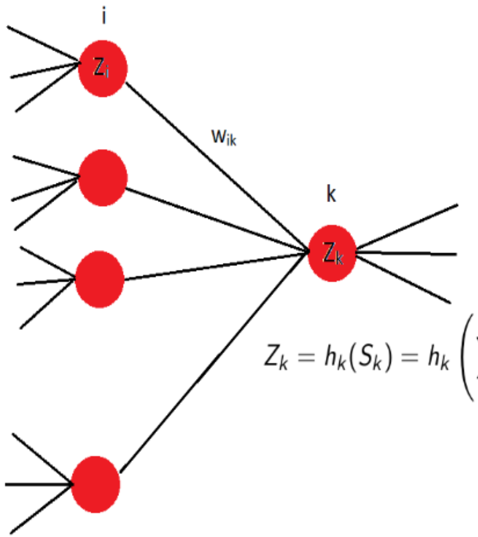
- It is useful to distinguish three types of units:
  - Input units (often denoted by X): input to the network
  - Hidden units (often denoted by Z): receive data from and send data to units within the network
  - Output units: the type of the output depends on the task (e.g., regression, binary or multiclass classification). In many cases, there is only one scalar output unit.
- Given inputs X, a neural network produces an output y

Terminologies



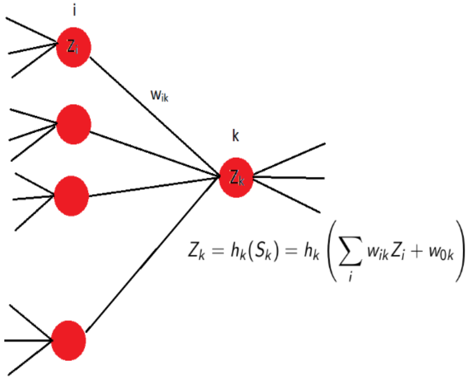
- A feedforward or fully-connected neural net includes the following:
  - A set of processing units (also called neurons or nodes)
  - L unit layers (one input layer, one output layer, and $L-2$ hidden layers), and $L-1$ weight layers
  - Weights $w_{i,k}^{l}$, which are connection strengths from unit i of the l-th layer to unit k of the $(l+1)$-th layer
  - A propagation rule that determines the total input or activation S_k of unit k, from the outputs in the previous layer that are connected to unit k
  - The output Z_k for each unit k, which is a function of the activation S_k, $Z_k = h_k(S_k)$, where h_k is an activation function

Computation at a single neuron



$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Computation at a single neuron



The total input sent to unit k is

$$S_k = \sum_i w_{ik} Z_i + w_{0k}$$

which is a weighted sum of the outputs from all units i that are connected to unit k, plus a bias/intercept term w_{0k} [or b_k].

Computation at a single neuron

The total input sent to unit k is

$$S_k = \sum_i w_{ik} Z_i + w_{0k}$$

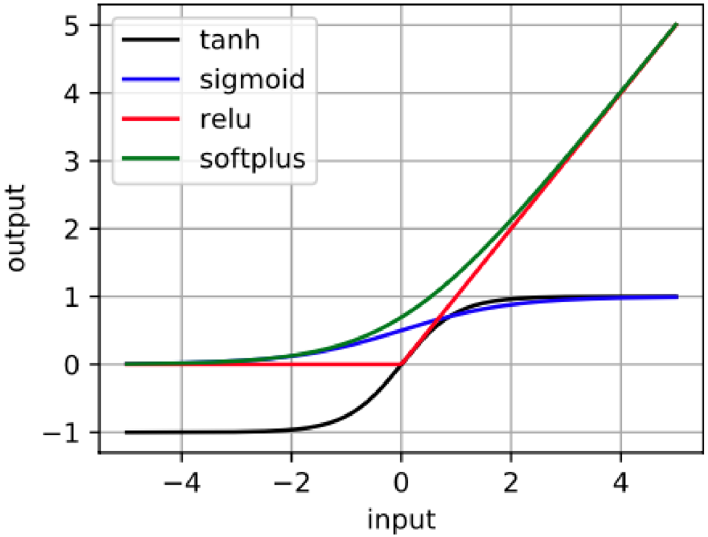
which is a weighted sum of the outputs from all units i that are connected to unit k, plus a bias/intercept term w_{0k} [or b_k].
Then, the output of unit k is

$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Usually, we use the same activation function $h_k = h$ for all units.

Activation Functions

- Provide nonlinearity
- Popular choices:



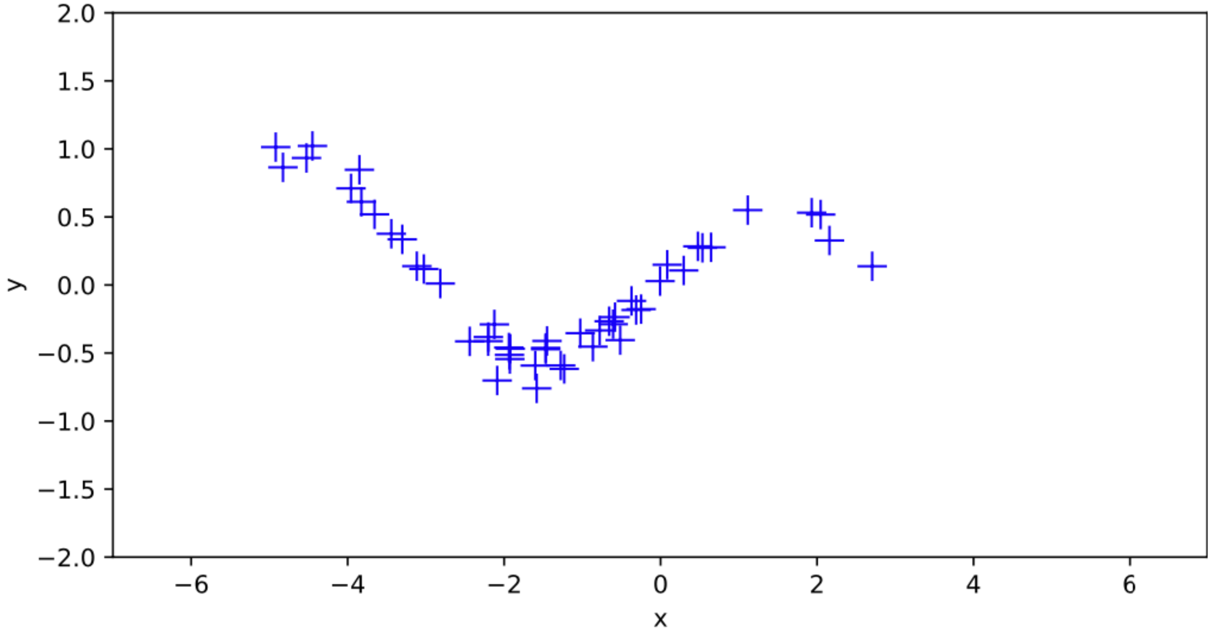
Tanh: $h(S) = \frac{e^S - e^{-S}}{e^S + e^{-S}}$

Sigmoid: $h(S) = \frac{1}{1+e^{-S}}$

Rectified linear: $h(S) = \max(0, S) = \begin{cases} S, & S > 0 \\ 0, & S \leq 0 \end{cases}$

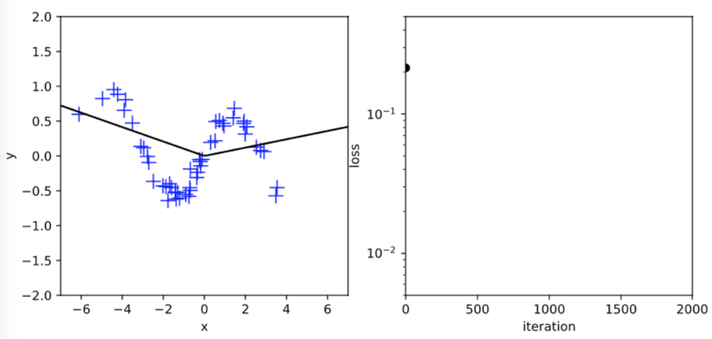
softplus: $h(S) = \ln(1 + e^S)$

Example: using a feedforward NN for non-linear regression



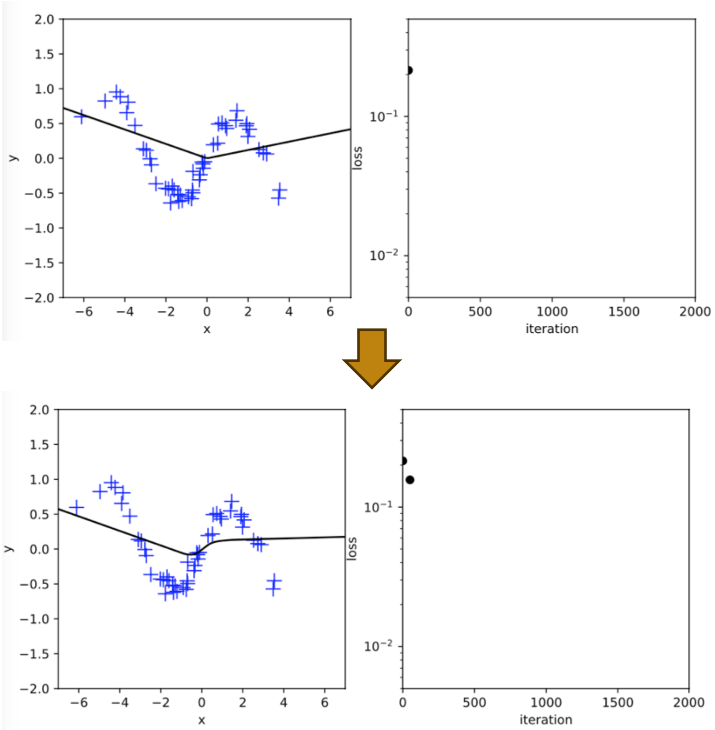
Example: using a feedforward NN for non-linear regression

- One hidden layer with 50 ReLU units



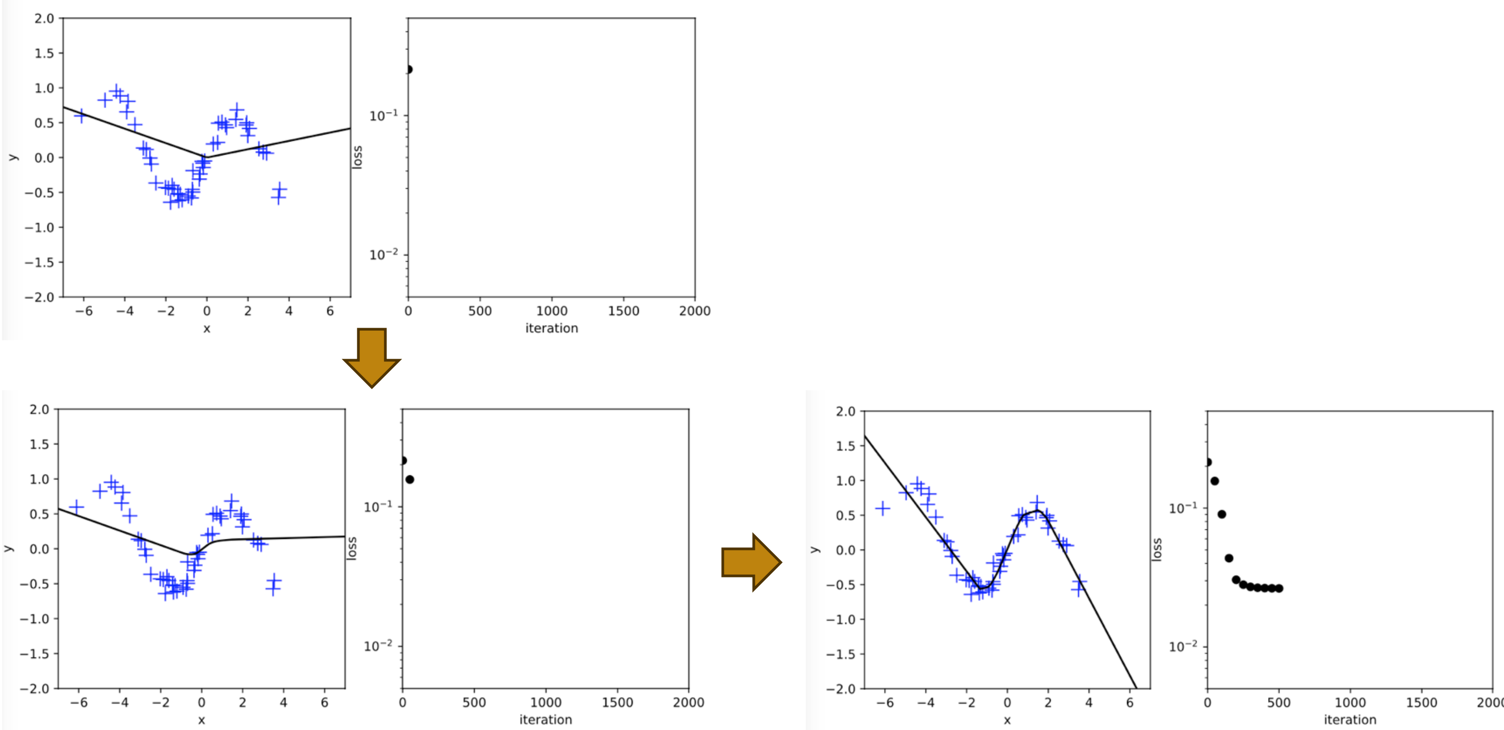
Example: using a feedforward NN for non-linear regression

- One hidden layer with 50 ReLU units



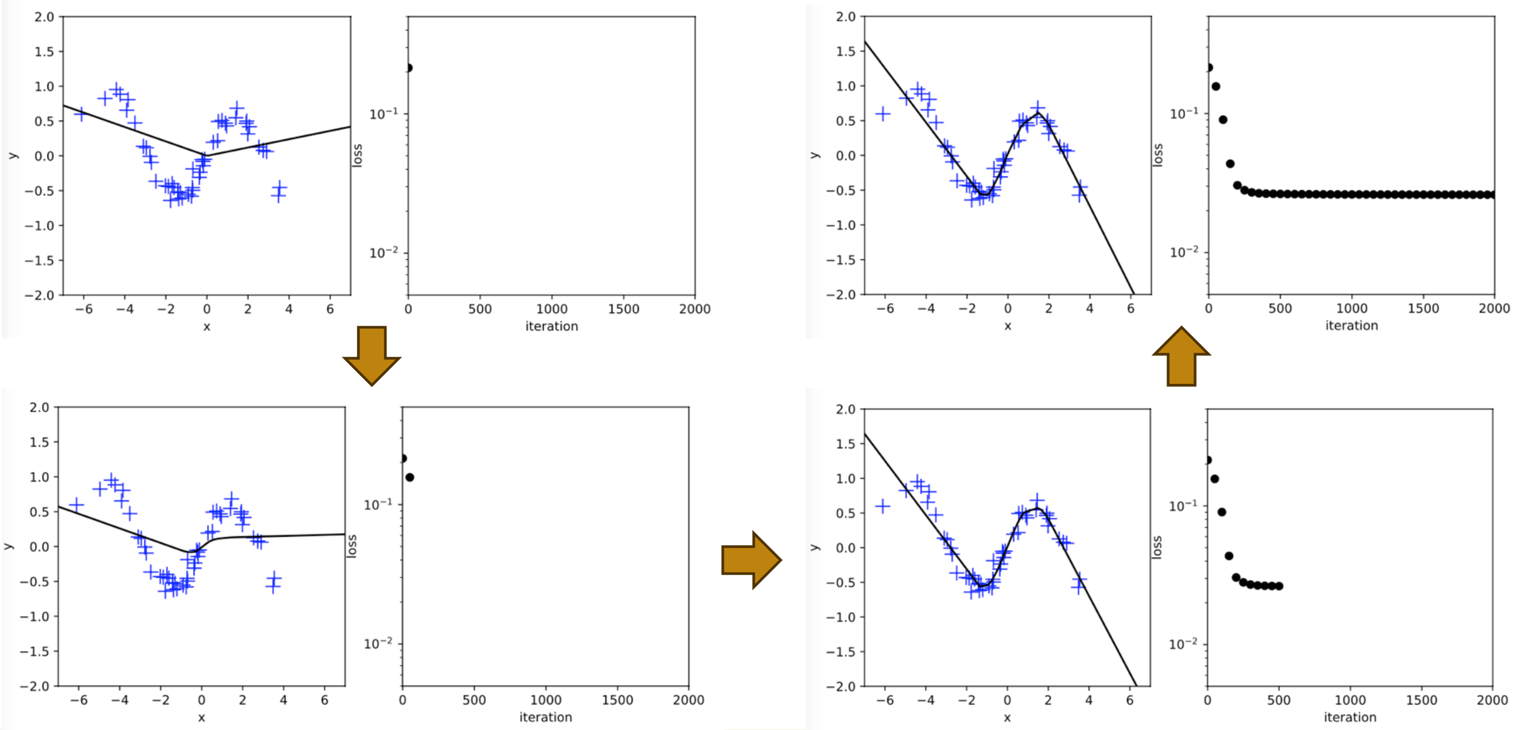
Example: using a feedforward NN for non-linear regression

- One hidden layer with 50 ReLU units



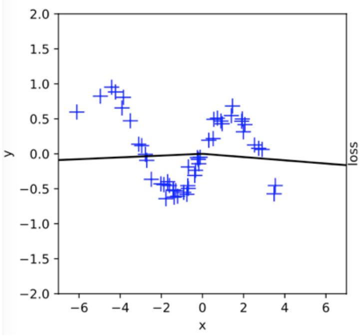
Example: using a feedforward NN for non-linear regression

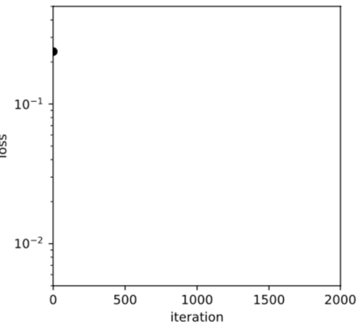
- One hidden layer with 50 ReLU units



Example: using a feedforward NN for non-linear regression

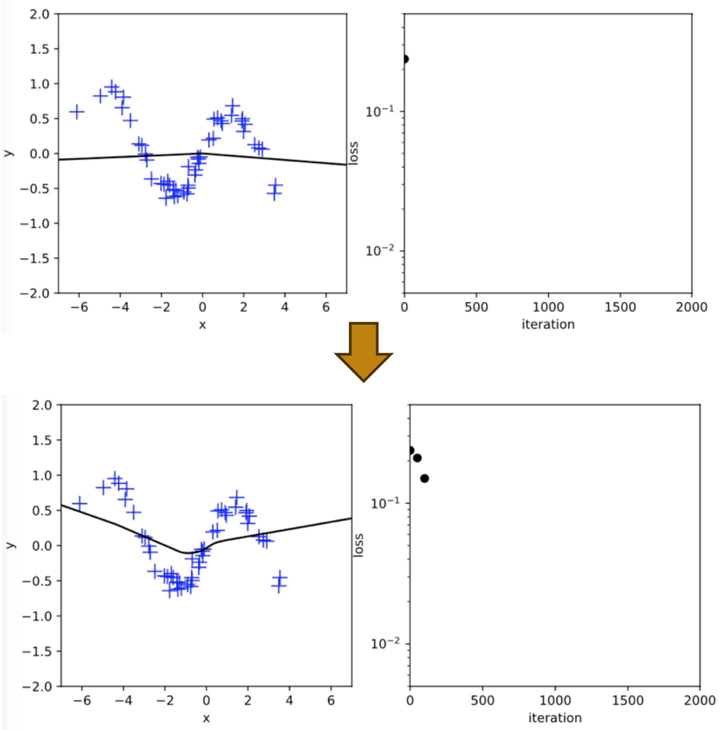
- Two hidden layers with 20 ReLU units





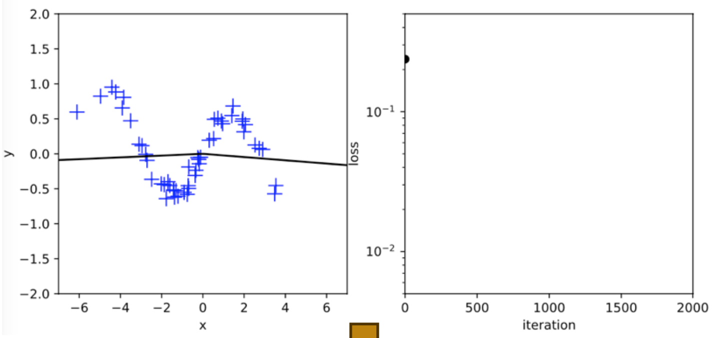
Example: using a feedforward NN for non-linear regression

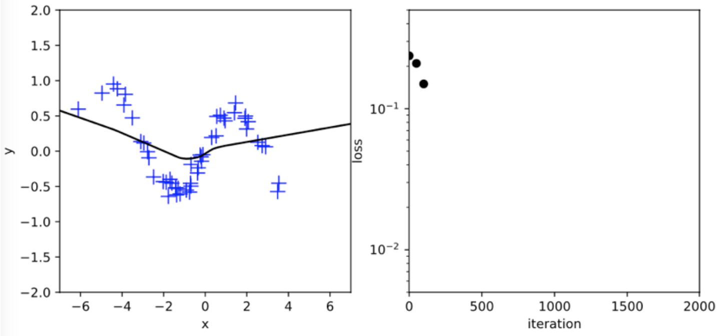
- Two hidden layers with 20 ReLU units

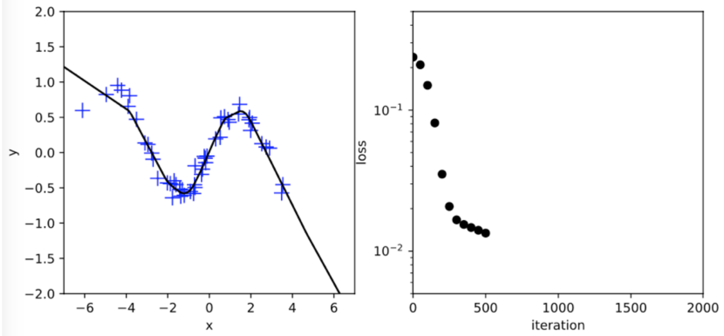


Example: using a feedforward NN for non-linear regression

- Two hidden layers with 20 ReLU units

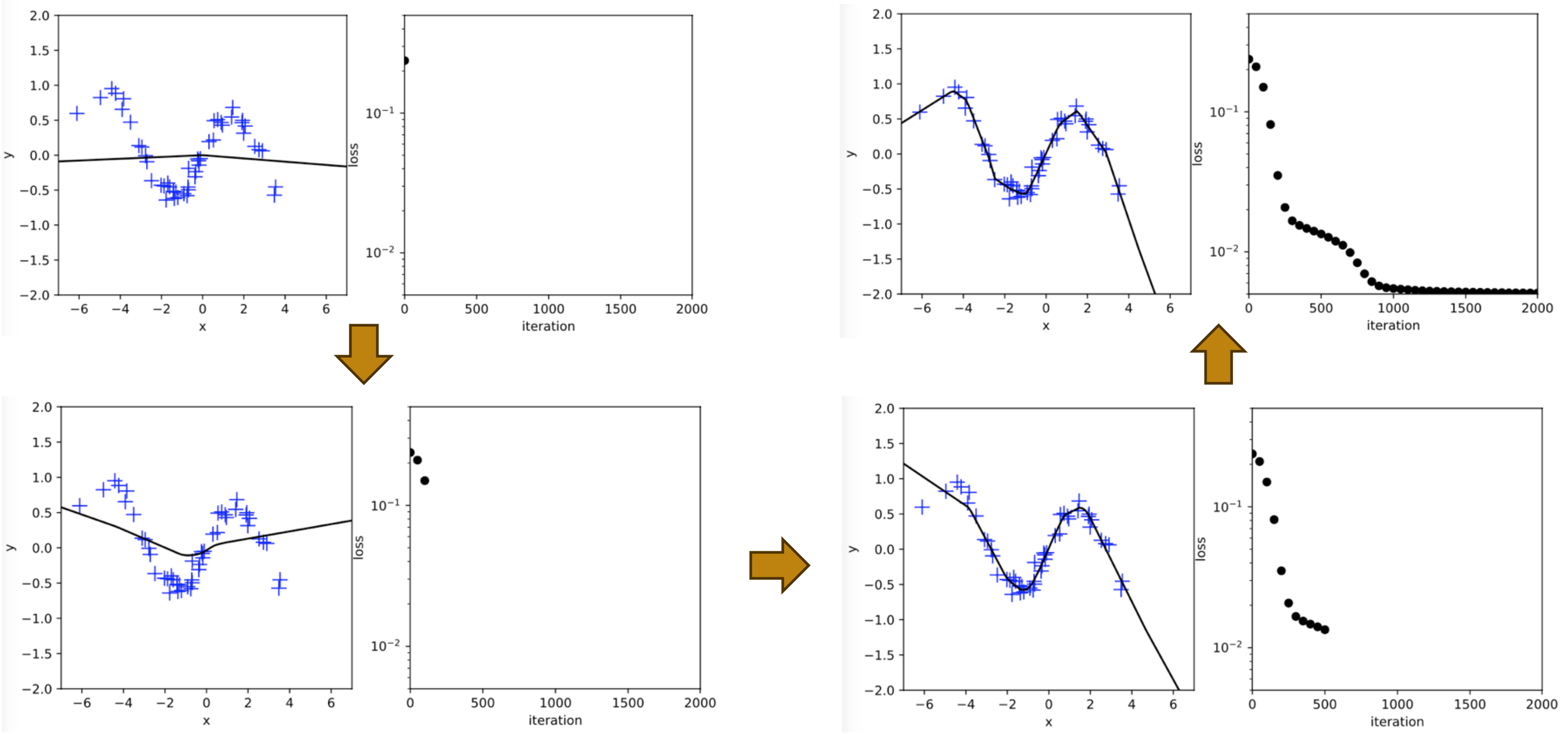






Example: using a feedforward NN for non-linear regression

- Two hidden layers with 20 ReLU units



Agenda

- Motivation
- Elements of a neural network
- **Why deep neural networks?**
- Forward pass → making predictions
- Backward pass or backpropagation → update model parameters

Why do we use hierarchical multi-layered models (deep neural networks)?

Theoretically:

1. visual scenes are hierarchical organised: input image - primitive features - object parts - object, e.g. dining room image - blobs and edges - chair/table legs/tops - chair/table
2. biological vision system is hierarchically organised
3. shallow architectures are inefficient at representing deep functions

Why do we use hierarchical multi-layered models (deep neural networks)?

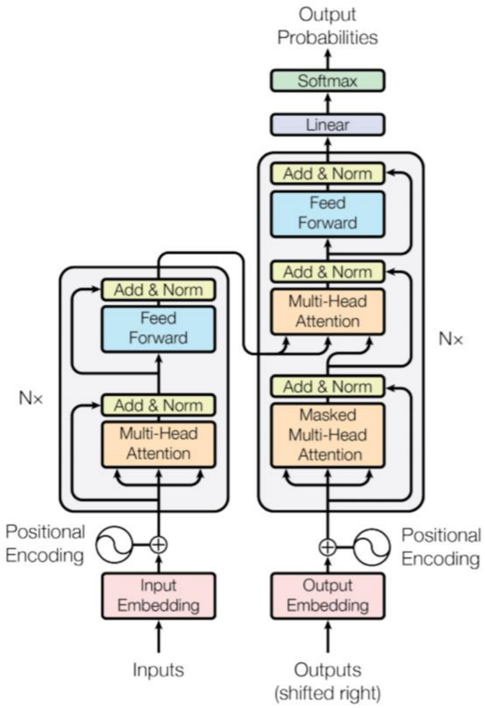
Theoretically:

1. visual scenes are hierarchical organised: input image - primitive features - object parts - object, e.g. dining room image - blobs and edges - chair/table legs/tops - chair/table
2. biological vision system is hierarchically organised
3. shallow architectures are inefficient at representing deep functions

Practically:

1. resurgence of neural networks/deep learning due to algorithmic improvements (new activation functions, new initialisation schemes, batch normalisation), data and compute
2. modular implementation: just need the (stochastic) gradients
3. many modern computational frameworks for prototyping, testing and deploying neural networks at scale
4. good performance on many modern perception tasks in computer vision and natural language processing/understanding.

An example deep neural network: Transformers



When do people avoid deep neural networks?

Many areas of science and safety systems still prefer simpler models, such as linear/logistic regression.
Compared to linear/logistic regression, the drawbacks are:

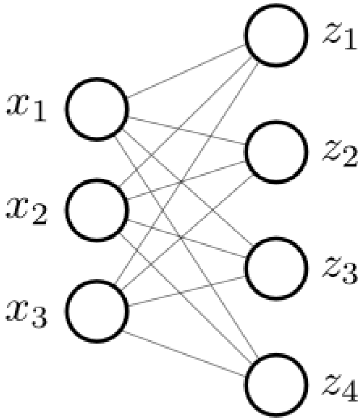
- potentially local optima yielding poor predictions
- less interpretable
- no easy way to deal with sparse inputs (could use embedding but expensive)

Agenda

- Motivation
- Elements of a neural network
- Why deep neural networks?
- **Forward pass → making predictions**
- Backward pass or backpropagation → update model parameters

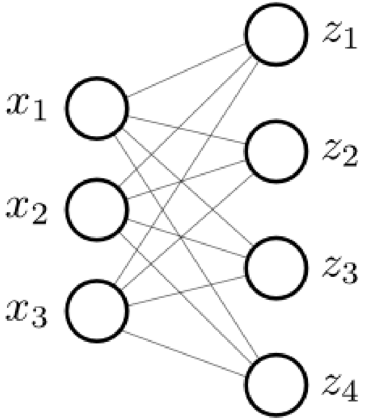
Computation in a single layer: an example

Consider a single weight layer connecting two hidden layers in a network.



Computation in a single layer: an example

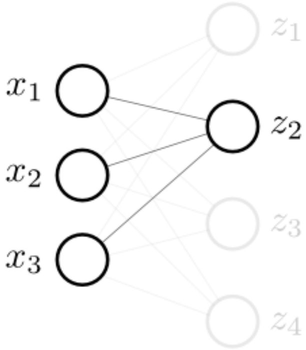
Consider a single weight layer connecting two hidden layers in a network.



Here $\{x_i\}$ are either the input dimensions or outputs of the previous hidden layer.

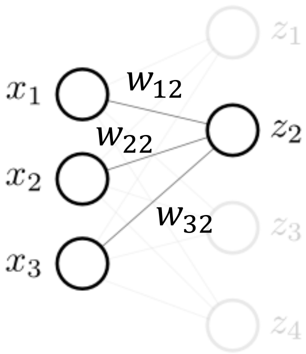
Computation in a single layer: an example

Let's consider a single hidden unit:



Computation in a single layer: an example

Let's consider a single hidden unit:



As discussed earlier

$$S_2 = \sum_{i=1}^{3} w_{i2} x_i + b_2 = w_{12} x_1 + w_{22} x_2 + w_{32} x_3 + b_2$$

$$z_2 = h(S_2)$$

Computation in a single layer: an example

From the last slide:

$$S_2 = \sum_{i=1}^{3} w_{i2} x_i + b_2 = w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + b_2$$

We can rewrite this as:

$$S_2 = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} w_{12} \\ w_{22} \\ w_{32} \end{bmatrix} + b_2$$

Computation in a single layer: an example

If we consider the other three hidden units:

$$S_1 = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix} + b_1,$$

$$S_3 = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} w_{13} \\ w_{23} \\ w_{33} \end{bmatrix} + b_3,$$

$$S_4 = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} w_{14} \\ w_{24} \\ w_{34} \end{bmatrix} + b_4.$$

Computation in a single layer: an example

We can put things together:

$$[S_1 \quad S_2 \quad S_3 \quad S_4] = [x_1 \quad x_2 \quad x_3] \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} + [b_1 \quad b_2 \quad b_3 \quad b_4] .$$

Computation in a single layer: an example

We can put things together:

$$[S_1 \quad S_2 \quad S_3 \quad S_4] = [x_1 \quad x_2 \quad x_3] \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} + [b_1 \quad b_2 \quad b_3 \quad b_4] \,.$$

or

$$\mathbf{S} = \mathbf{x}\mathbf{W} + \mathbf{b}$$

where $\mathbf{S}$, $\mathbf{b}$ are row vectors with 4 elements, $\mathbf{x}$ is a row vector with 3 elements, and W is a matrix of size 3×4.

Computation in a single layer: an example

We can put things together:

$$\begin{bmatrix} S_1 & S_2 & S_3 & S_4 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix}.$$

or

$$\mathbf{S} = \mathbf{x}\mathbf{W} + \mathbf{b}$$

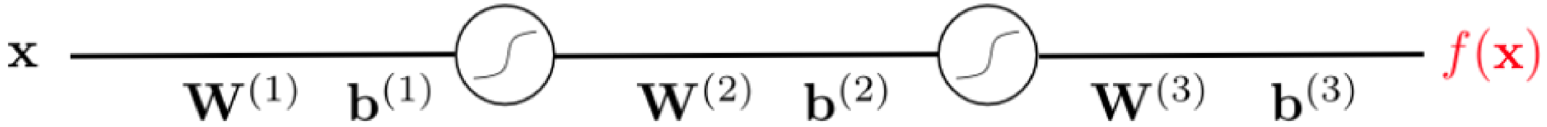
where $\mathbf{S}$, $\mathbf{b}$ are row vectors with 4 elements, $\mathbf{x}$ is a row vector with 3 elements, and W is a matrix of size 3×4.

We can then apply the activation function:

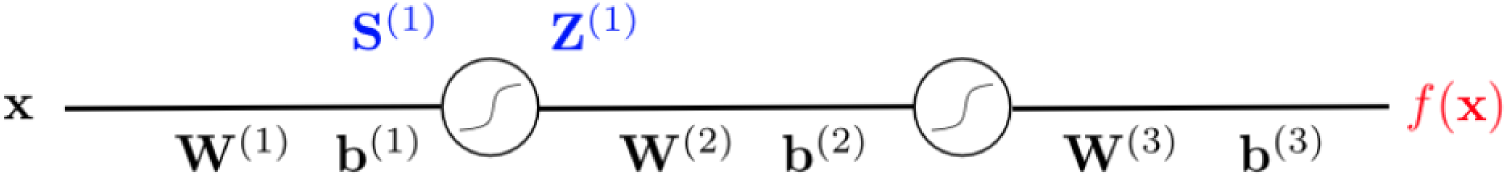
$$\mathbf{Z} = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 \end{bmatrix} = \begin{bmatrix} h(S_1) & h(S_2) & h(S_3) & h(S_4) \end{bmatrix} := h(\mathbf{S})$$

We can then use $\mathbf{Z} = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 \end{bmatrix}$ as the input to the next layer.

Computation with multiple layers: forward pass



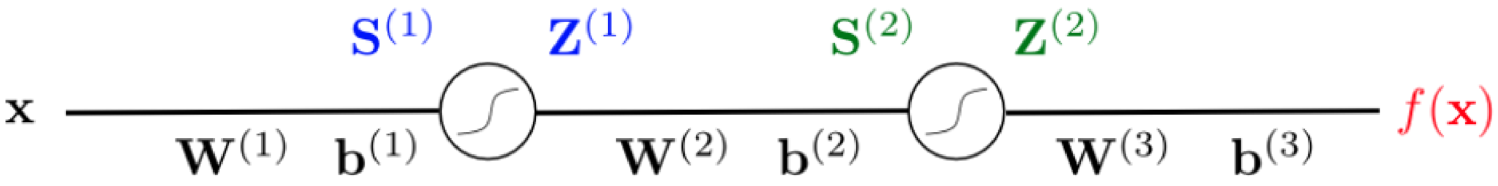
Computation with multiple layers: forward pass



$$\mathbf{S}^{(1)} = \mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}$$

$$\mathbf{Z}^{(1)} = h(\mathbf{S}^{(1)})$$

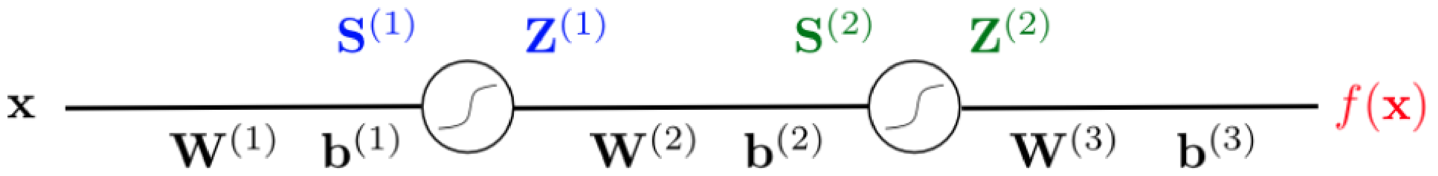
Computation with multiple layers: forward pass



$$\mathbf{S}^{(1)} = \mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \qquad \mathbf{S}^{(2)} = \mathbf{Z}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

$$\mathbf{Z}^{(1)} = h(\mathbf{S}^{(1)}) \qquad \mathbf{Z}^{(2)} = h(\mathbf{S}^{(2)})$$

Computation with multiple layers: forward pass



$$\mathbf{S}^{(1)} = \mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \quad \mathbf{S}^{(2)} = \mathbf{Z}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \quad f(\mathbf{x}) = \mathbf{Z}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)}$$

$$\mathbf{Z}^{(1)} = h(\mathbf{S}^{(1)}) \quad \mathbf{Z}^{(2)} = h(\mathbf{S}^{(2)})$$

Note: We do not apply the activation function at the output layer.

Agenda

- Motivation
- Elements of a neural network
- Why deep neural networks?
- Forward pass → making predictions
- Backward pass or backpropagation → update model parameters

Reading

- Bishop 5.1

Acknowledgments

- The slides are largely adopted from COMP4670/8600, 2024 Semester 1.