Deep Learning & LLMs 1

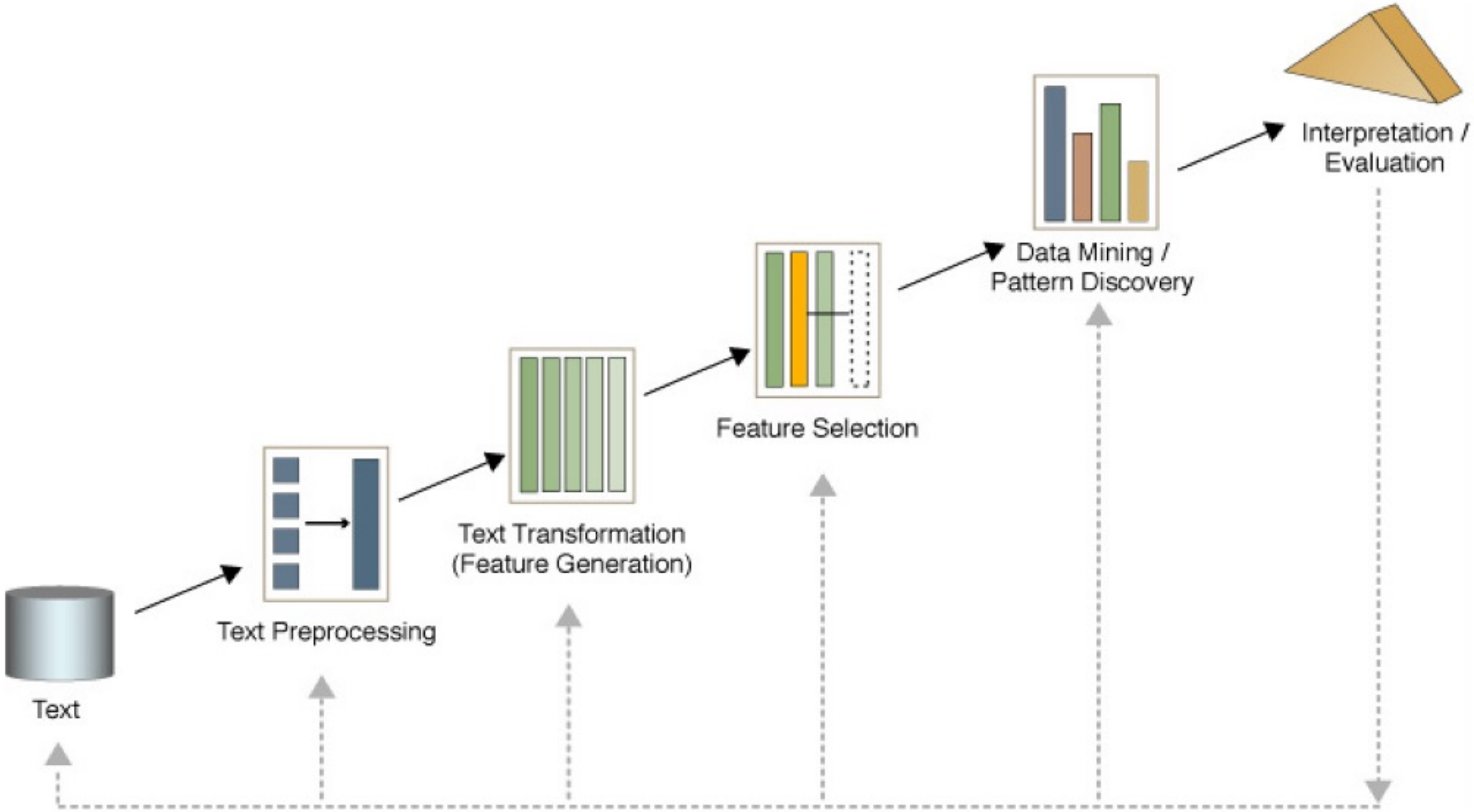
Applied Text Mining, from Foundations to Advanced

Ayoub Bagheri

This lecture

- Introduction to neural networks
- Feed-forward & deep neural networks
- Recurrent neural networks

Text mining process



Introduction

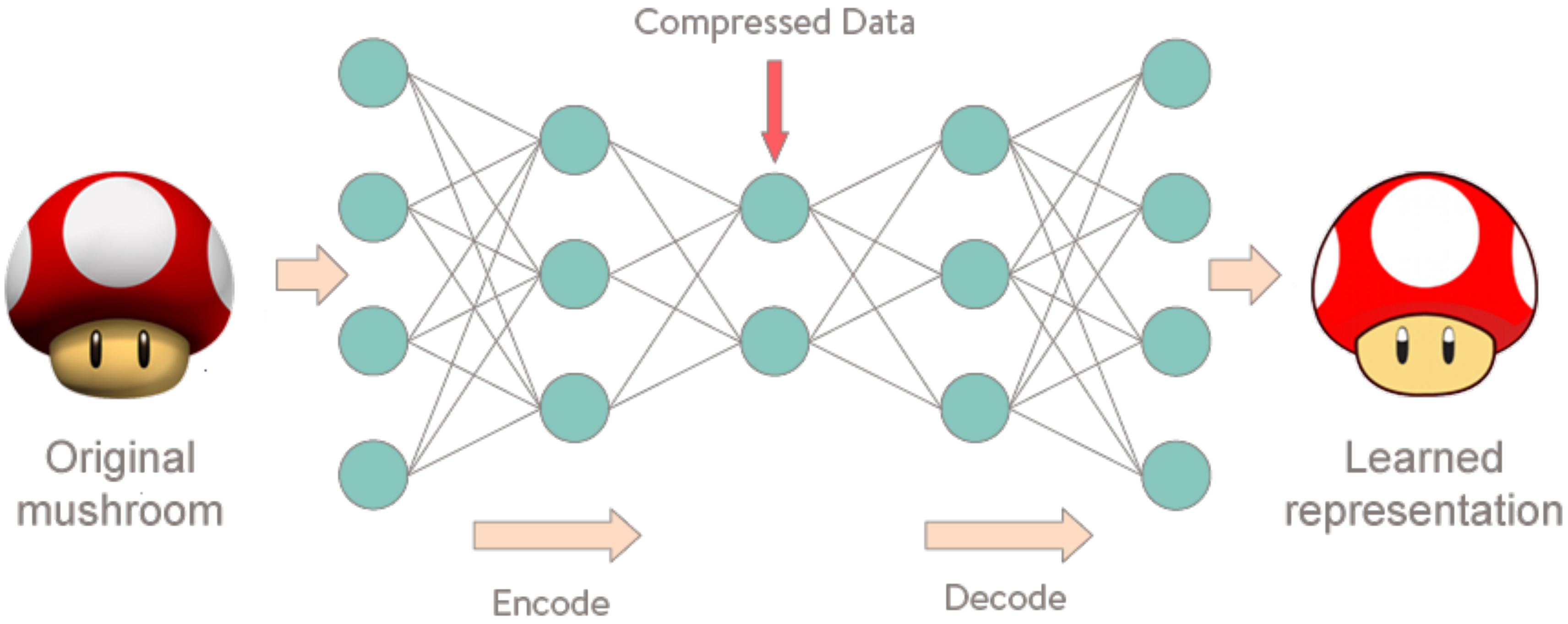
Why should we learn this?

State-of-the-art performance on various tasks

- Text prediction (your phone's keyboard)
- Text mining
- Forecasting
- Spam filtering
- Compression (dimension reduction)
- Text generation
- Translation
- ...

https://thispersondoesnotexist.com/

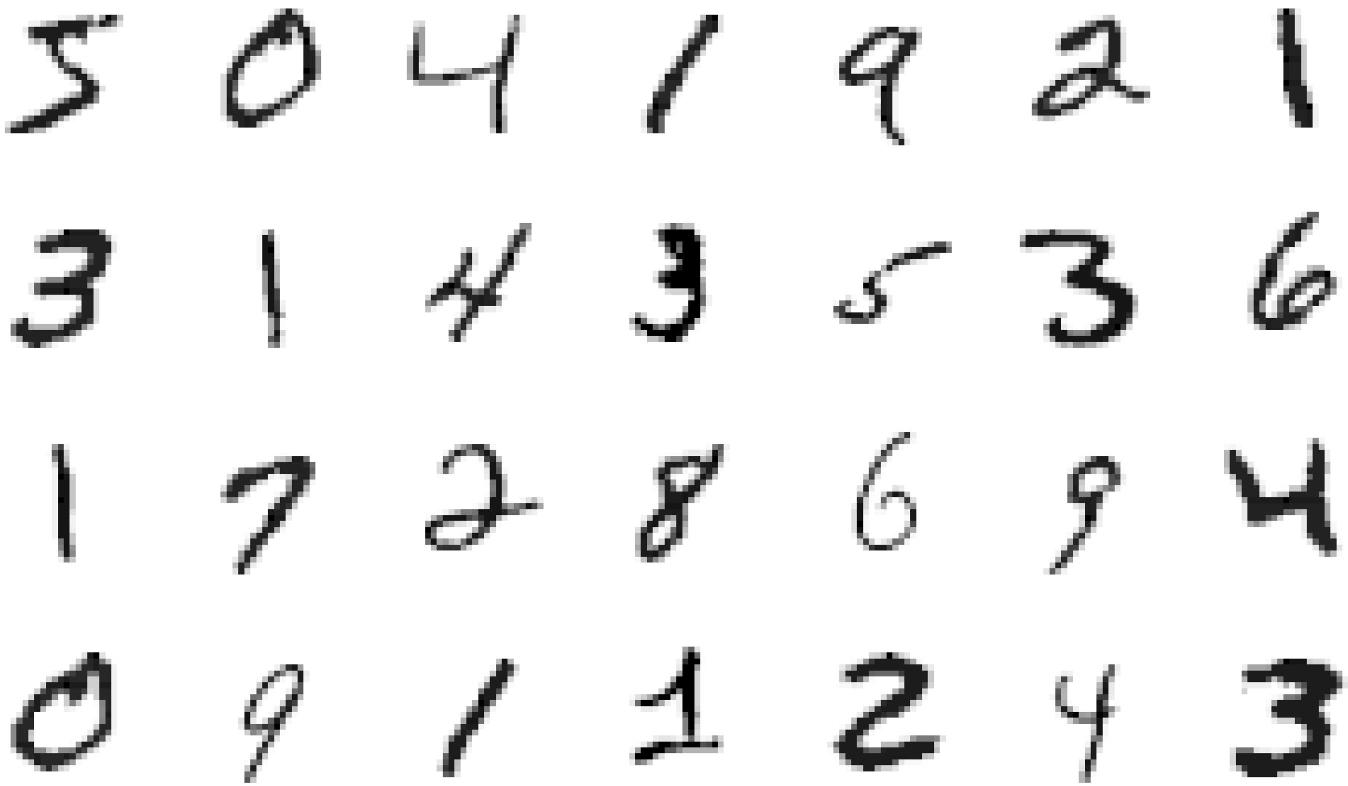
http://bethgelab.org



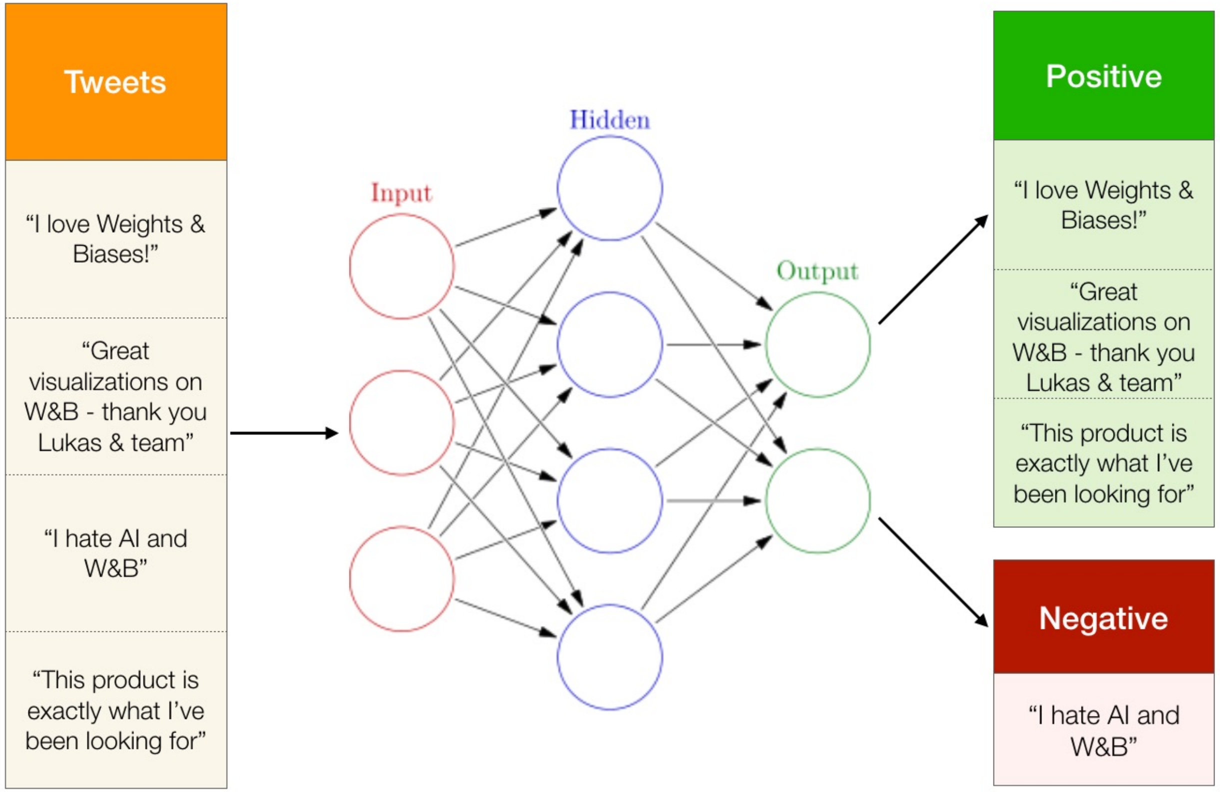
https://community.canvaslms.com/t5/Canvas-Developers-Group/Canvas-LMS-Cheat-Detection-System-In-Python/m-p/118134

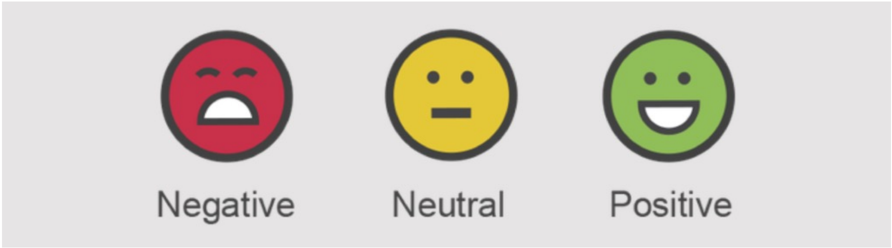
“Hello world” of neural networks

- MNIST (Modified National Institute of Standards and Technology)
- Handwritten digits
- 28 * 28 pixels
- 60 000 training images and 10 000 testing images



"Hello world" of neural networks for text: Sentiment classification with LSTM





So what is a neural network?

Neural networks

$$y = f(X) + \epsilon$$

- Neural networks are a way to specify $f(X)$
- You can display $f(X)$ graphically

- Let's graphically represent linear regression!

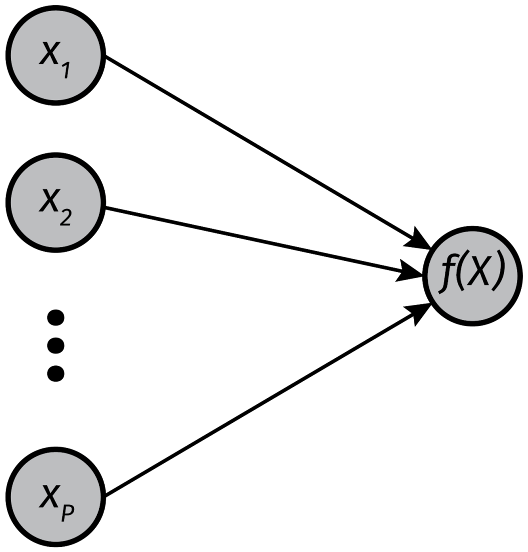
$$f(X_i) = \sum_{p=1}^{P} \beta_p x_{pi}$$

Linear regression as neural net

Graphical representation

- Parameters are arrows
- Arrows ending in a node are summed together
- Intercept is not drawn

$$f(X_i) = \alpha + \sum_{p=1}^{P} \beta_p x_{pi}$$

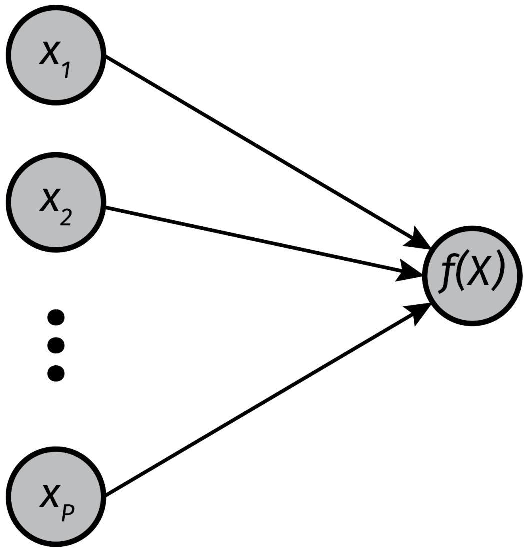


Linear regression as neural net

Neural network jargon

- Parameter = **weight**
- Intercept = **bias**

$$f(X_i) = \boldsymbol{\beta} + \sum_{p=1}^{P} \boldsymbol{w}_p x_{pi}$$



Single layer neural networks

$$y = f(X) + \epsilon$$

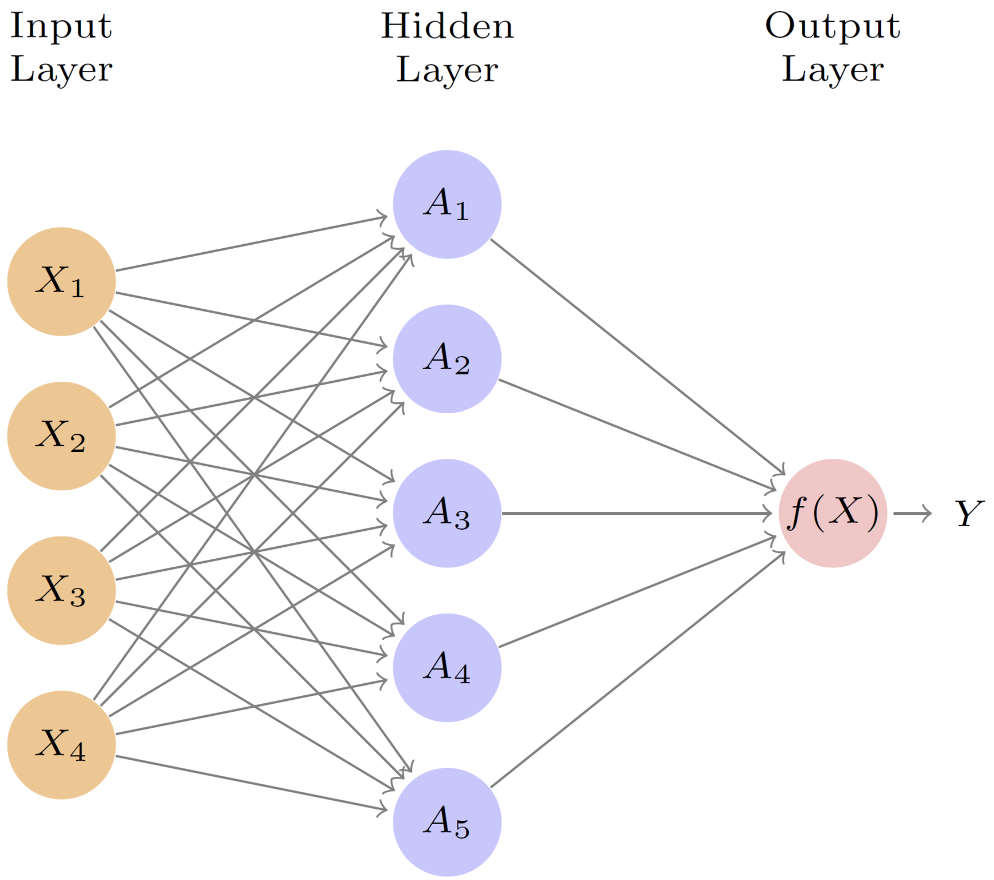
Specify a layer with K *hidden units* called A

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

Where

$$A_k = h_k(X) = g\left(w_{0k} + \sum_{p=1}^{P} w_{pk} x_p\right)$$

Single layer neural networks

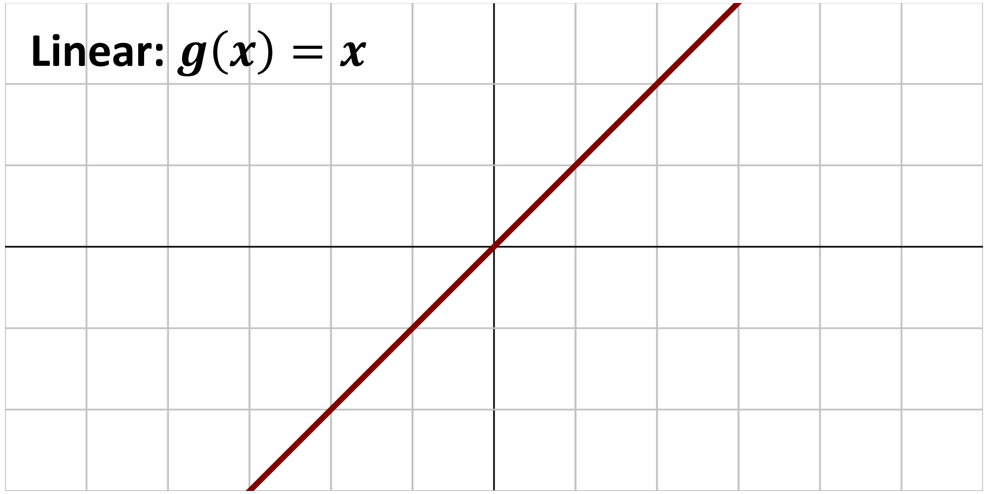


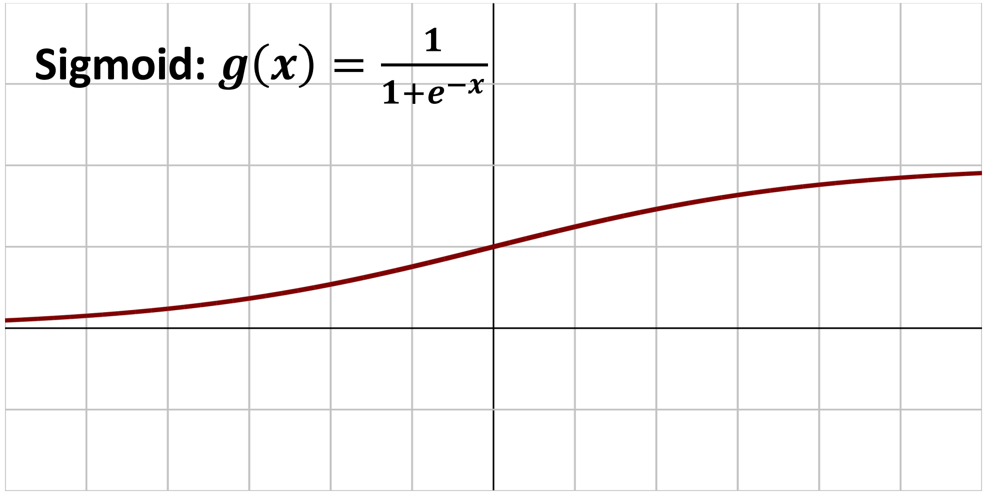
Single layer neural networks

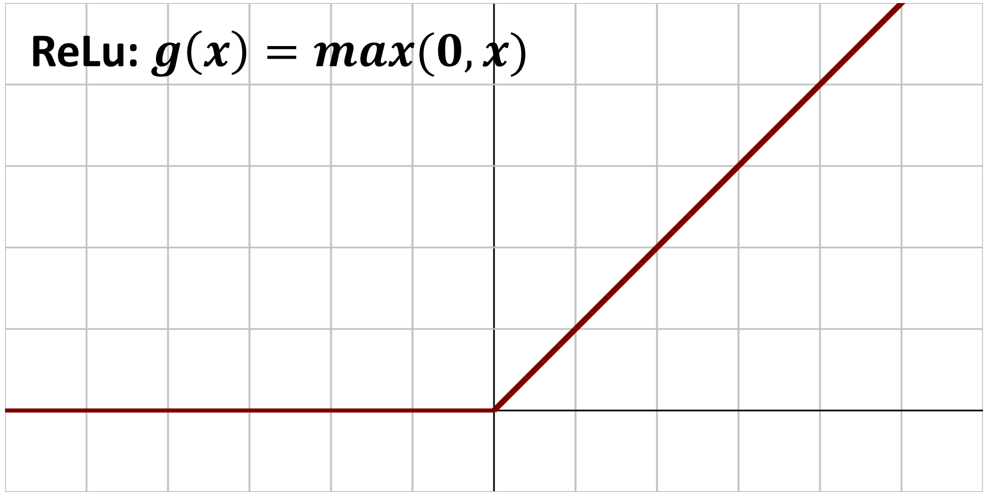
- What about the function $g(\cdot)$?
- This is called the **activation function**
- A transformation of the linear combination of predictors

$$h_k(X) = g\left(w_{0k} + \sum_{p=1}^{P} w_{pk} x_p\right)$$

Activation functions

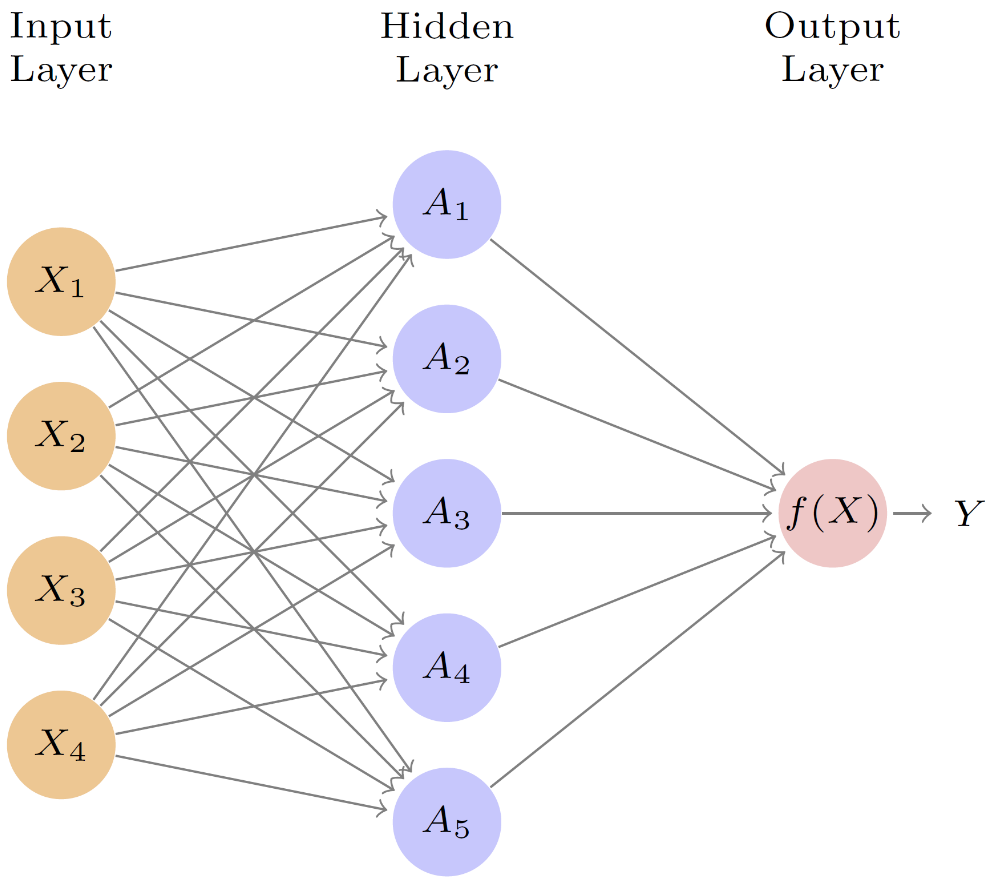






- Rectified linear (ReLu) is most popular nowadays
- Nonlinearity necessary! Otherwise: collapse to linear regression

Single layer neural networks



Feed-forward Neural Networks

Feed-forward neural networks

We can go deeper

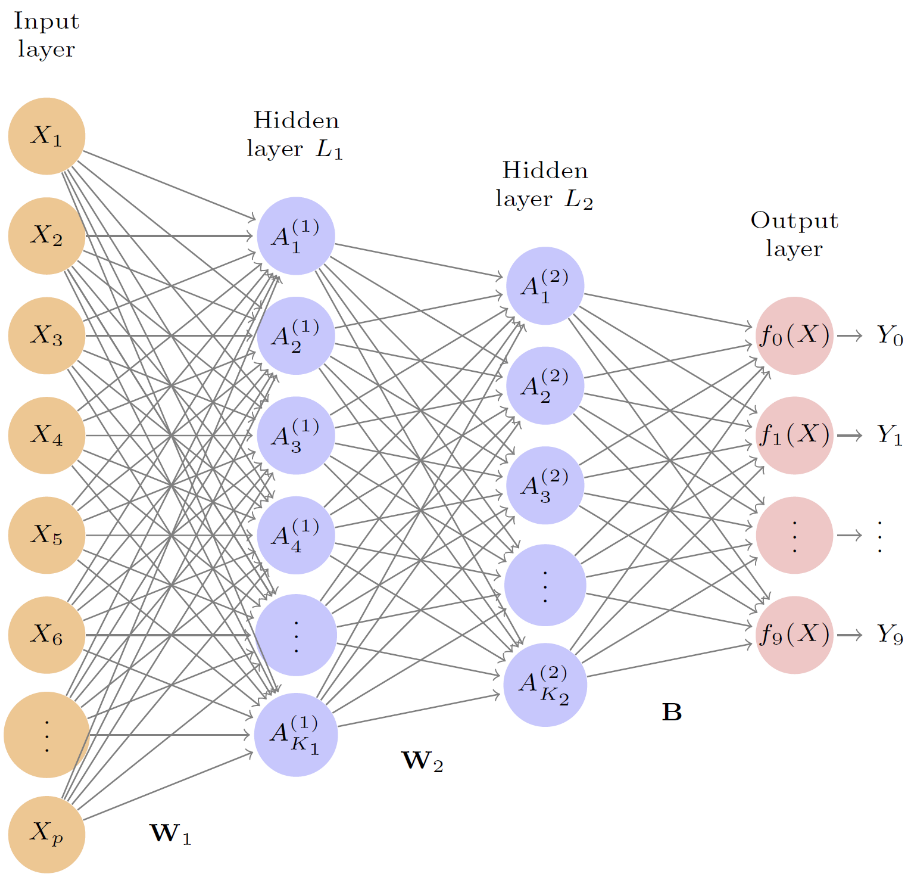
- More hidden layers after one another
- Higher-order features composed of lower-order features

Universal function approximation theorem, version 2

Any “well-behaved” function can be represented by neural net of sufficient *depth* with nonlinear activation function

(deep neural nets may be more tractable than wide)

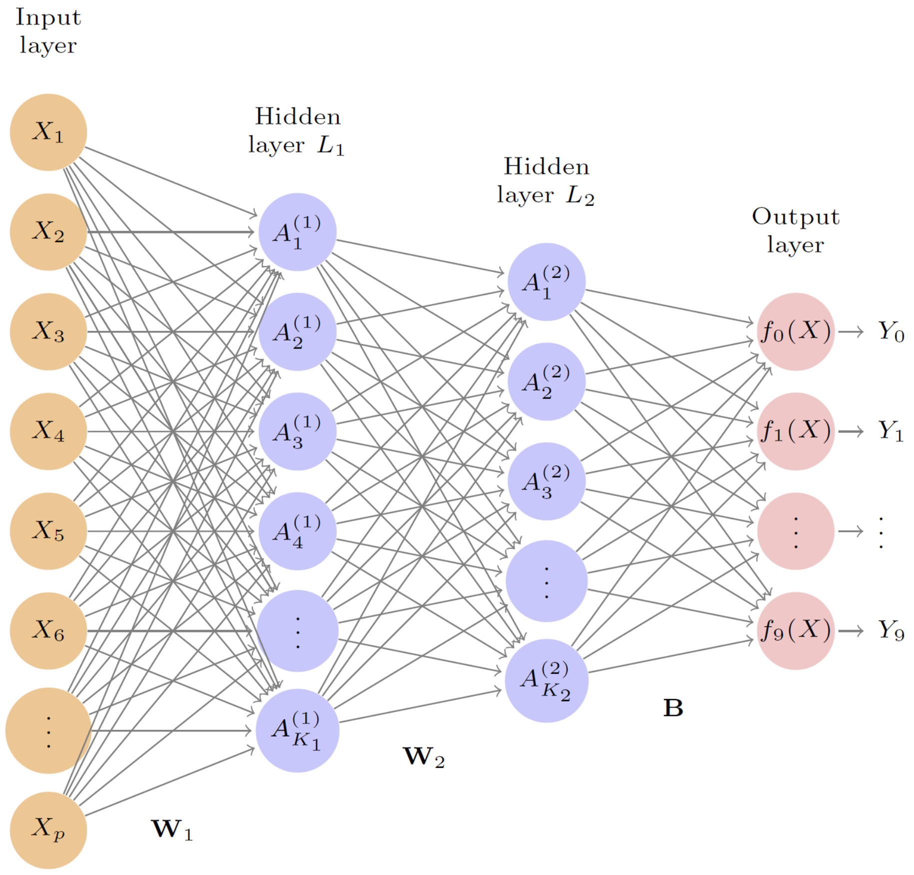
Feed-forward neural networks



Feed-forward neural networks

Feed-forward network architecture defined by:

- Number of layers
- Number of hidden units in each layer
- Activation function for each layer
- Activation function for output layer



Keras!

```
import(keras)

model_dff =
  keras_model_sequential() %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 256, activation = "relu") %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(10, activation = "softmax")
```

Keras!

```
summary(model_dff)

Layer (type)                  Output Shape              Param #
=====================================================================
flatten (Flatten)             (None, 784)               0

_____________________________________________________________________
dense_1 (Dense)               (None, 256)               200960

_____________________________________________________________________
dense_2 (Dense)               (None, 128)               32896

_____________________________________________________________________
dense_3 (Dense)               (None, 10)                1290
=====================================================================
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0

_____________________________________________________________________
```

How to estimate parameters?

Estimating parameters

- We need some way to measure how well the network does
- Parameters that make the network perform well are good!

Loss function

- For continuous outcomes you can use squared error
  (same as linear regression!)

$$L(\theta) = (f(X_i;\theta) - y_i)^2$$

- For binary outcomes you can use binary cross-entropy
  (same as logistic regression!)

$$L(\theta) = -\big(y_i \log\big(f(X_i;\theta)\big) + (1 - y_i)\log\big(f(X_i;\theta)\big)\big)$$

Gradient descent

Iteration: step of size λ in the direction of the negative gradient

$$\theta^{(j+1)} = \theta^{(j)} - \lambda \cdot g\left(\theta^{(j)}\right)$$

- But in neural networks, how do we compute gradients?
- We have functions of functions!
- Software like tensorflow / Keras / torch does this for you!
- **Backpropagation**: smart repeated use of the *chain rule* to compute derivatives

Break

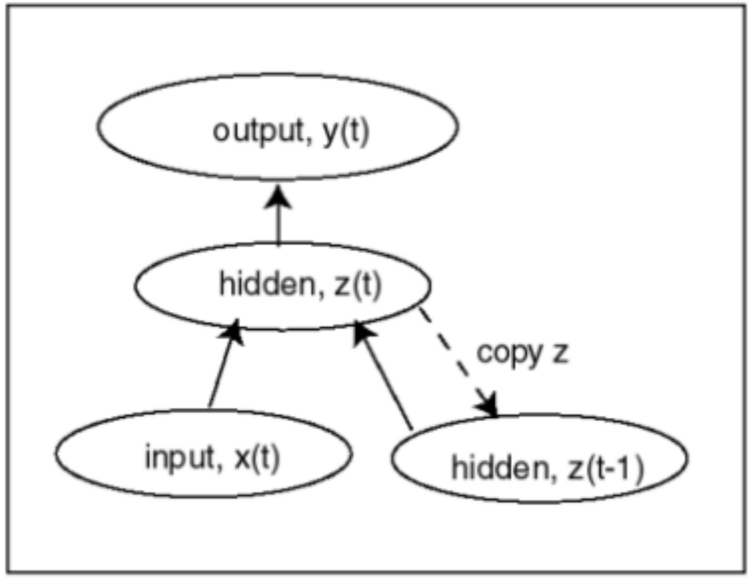
Recurrent Neural Network (RNN)

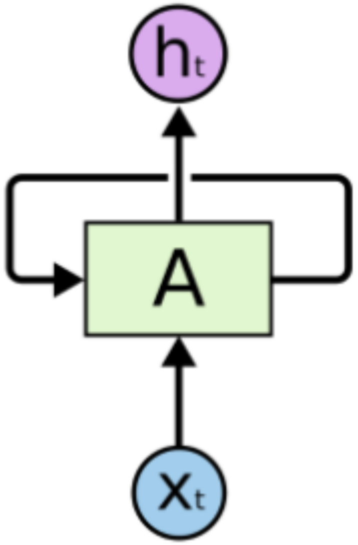
Recurrent Neural Network

- Another famous architecture of **Deep Learning**
- Preferred algorithm for sequential data
  - time series, speech, **text**, financial data, audio, video, weather and much more.
  - **text**: sentiment analysis, sequence labeling, speech tagging, machine translation, etc.

- Maintains **internal memory**, thus can remember its previous inputs

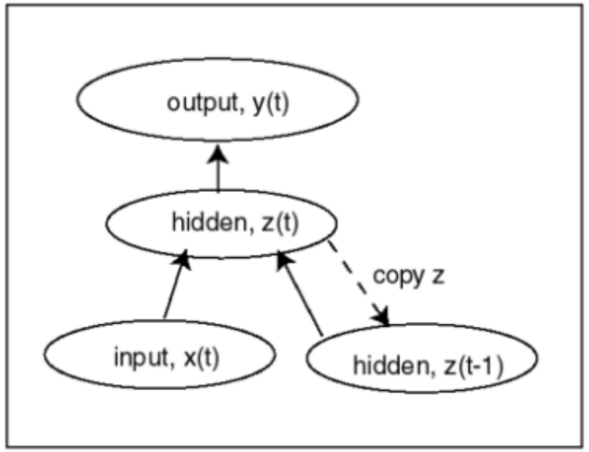
Simple recurrent network

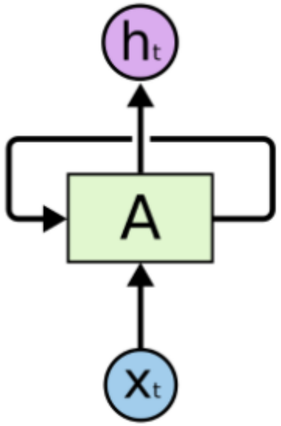




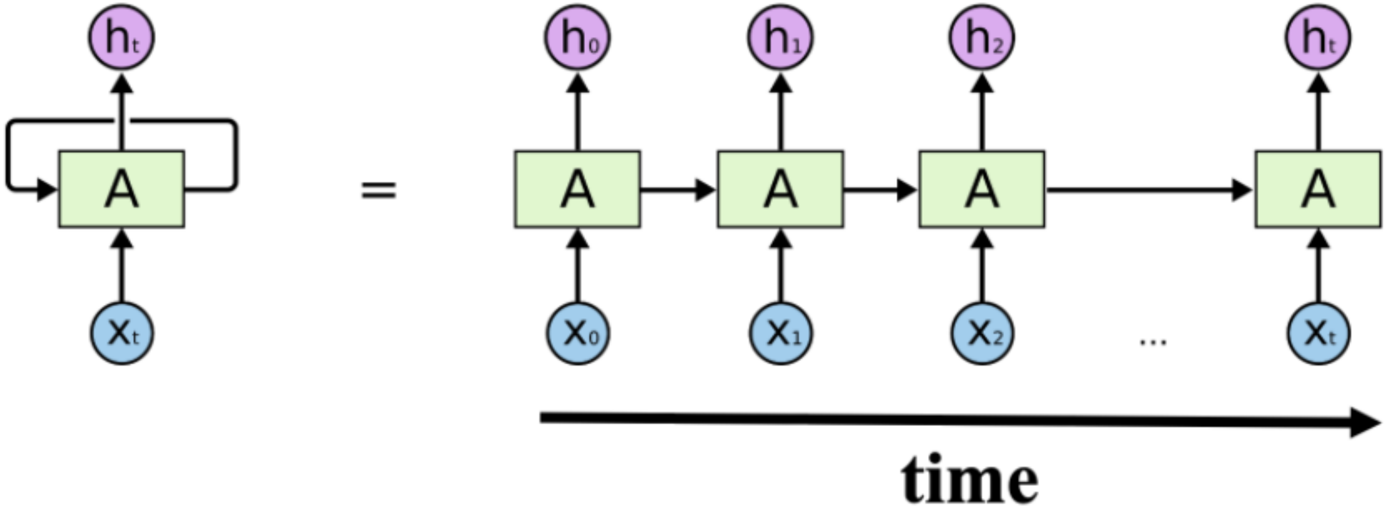
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Simple recurrent network



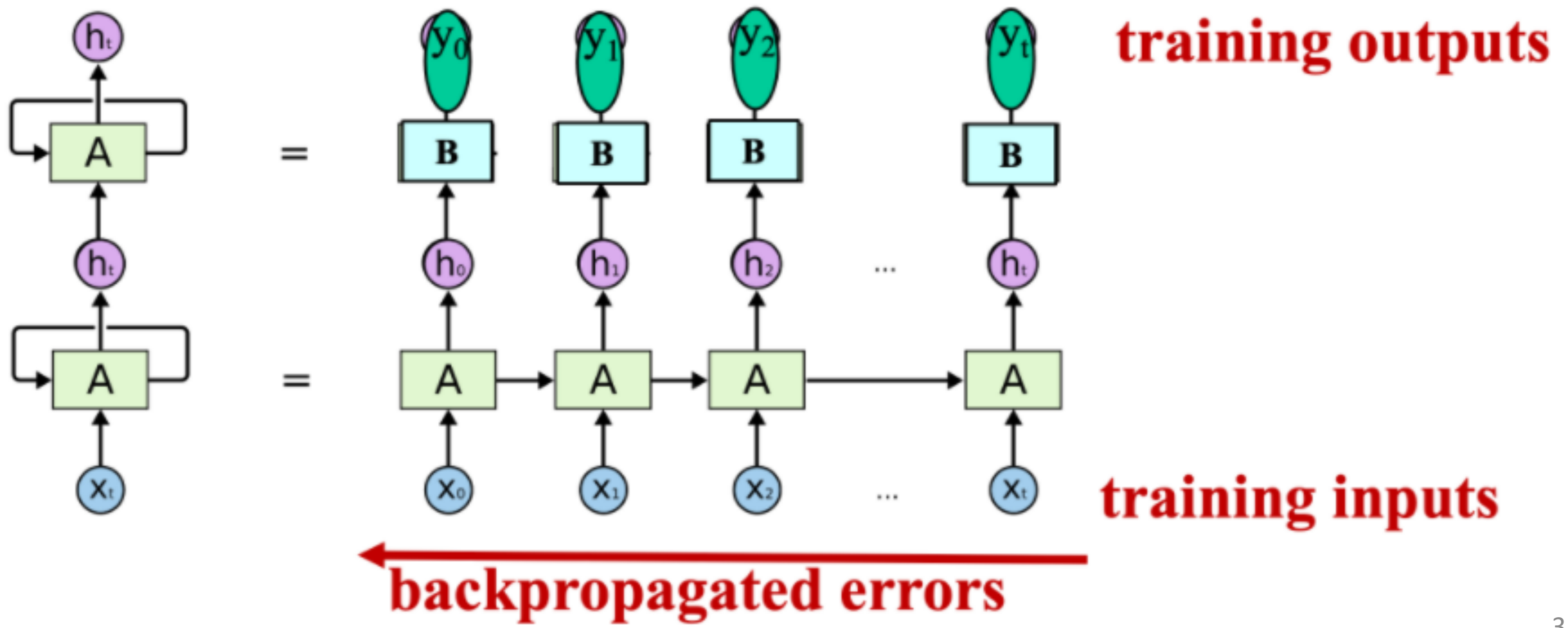


http://colah.github.io/posts/2015-08-Understanding-LSTMs/



Training RNNs

- RNNs can be trained using "backpropagation through time."
- Can viewed as applying normal backprop to the unrolled network.



The problem of Vanishing Gradient

- Consider a **RNN** model for a **machine translation** task from English to Dutch.
- It has to read an English sentence**, store as much information as possible** in its hidden activations, and output a Dutch sentence.
- The information about the first word in the sentence doesn't get used in the predictions until it starts generating Dutch words.
- There's **a long temporal gap** from when it sees an input to when it uses that to make a prediction.
- It can be hard to learn **long-distance dependencies**.
- In order to adjust the input-to-hidden weights based on the first input, the error signal needs to travel backwards through this entire pathway.

Long Short-Term Memory (LSTM)

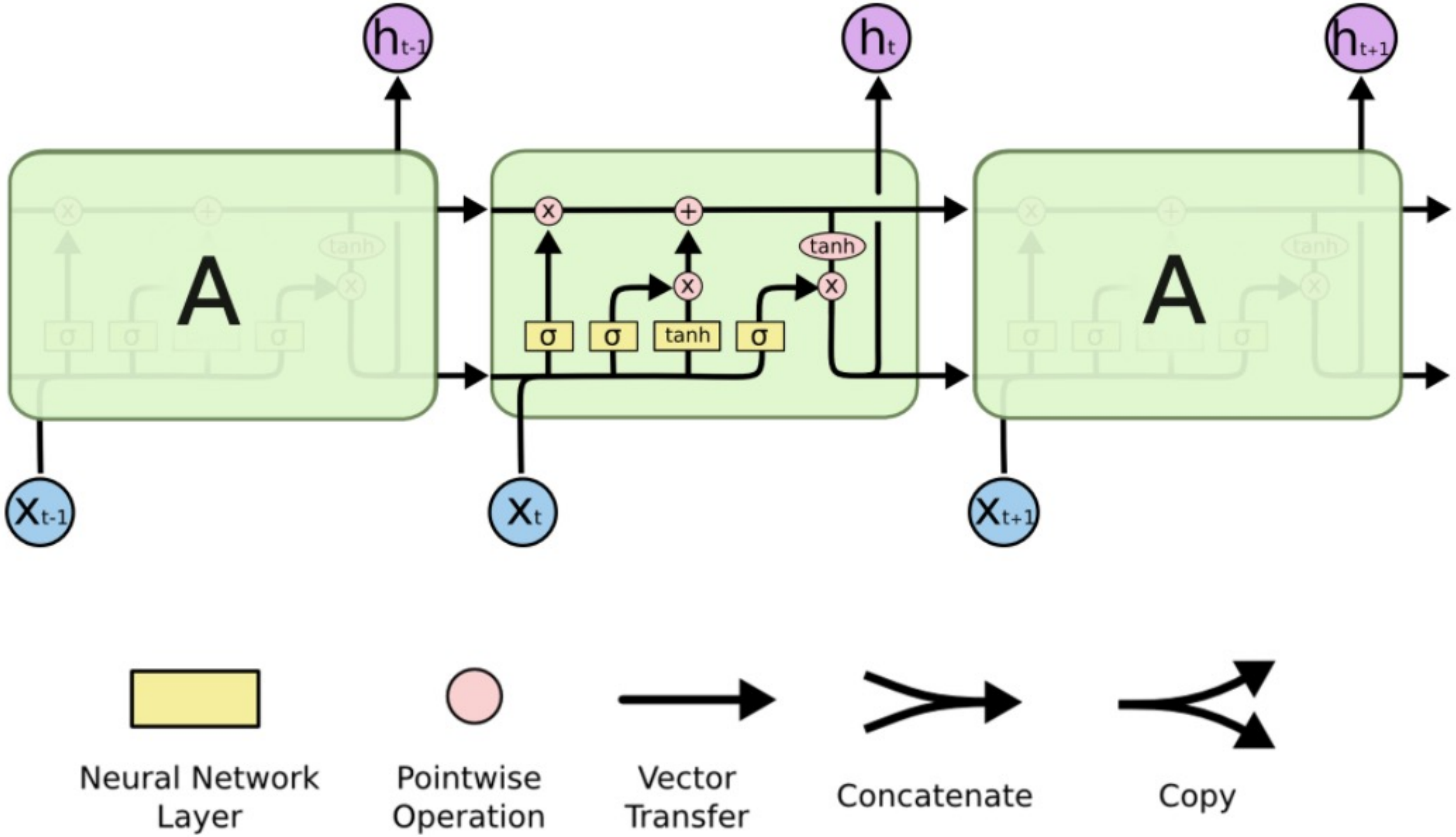
Long Short-Term Memory

- Prevents vanishing/exploding gradient problem by:
  - introducing a gating mechanism
  - turning multiplication into addition

- Designed to make it easy to remember information over long time periods until it's needed.

- The activations of a network correspond to short-term memory, while the weights correspond to long-term memory.

LSTM architecture



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Extensions

- **Bi-directional** network: separate LSTMs process forward and backward sequences, and hidden layers at each time step are concatenated to form the cell output.
- **Gated Recurrent Unit (GRU):** alternative RNN to LSTM that uses fewer gates, combines forget and input gates into “update” gate, eliminates cell state vector.
- **Attention:** Allows network to learn to attend to different parts of the input at different time steps, shifting its attention to focus on different aspects during its processing.

State-of-the-Art

- Recurrent neural networks
  - LSTM
  - GRU
  - Bi-directional network
- Transformers
- Contextual embeddings
- Large Language Models (LLMs) --> ChatGPT

Conclusion

- Neural networks are popular methods especially for text mining
- Feed-forward & RNN & CNN (tomorrow)
- RNN works better for text data

Practical 6

Questions?