Deep learning

Regularisation & Sequential modelling

Announcements

- Environment
  - It is slower than we want, trying to resolve.
  - If you want, we have created instructions for local development.
    - Docker image updated to 1.1 for today.
- Assignment back on track
  - Deadline 9th of April.
  - Grading will be 0, 1, 2
- Mid-course evaluation.

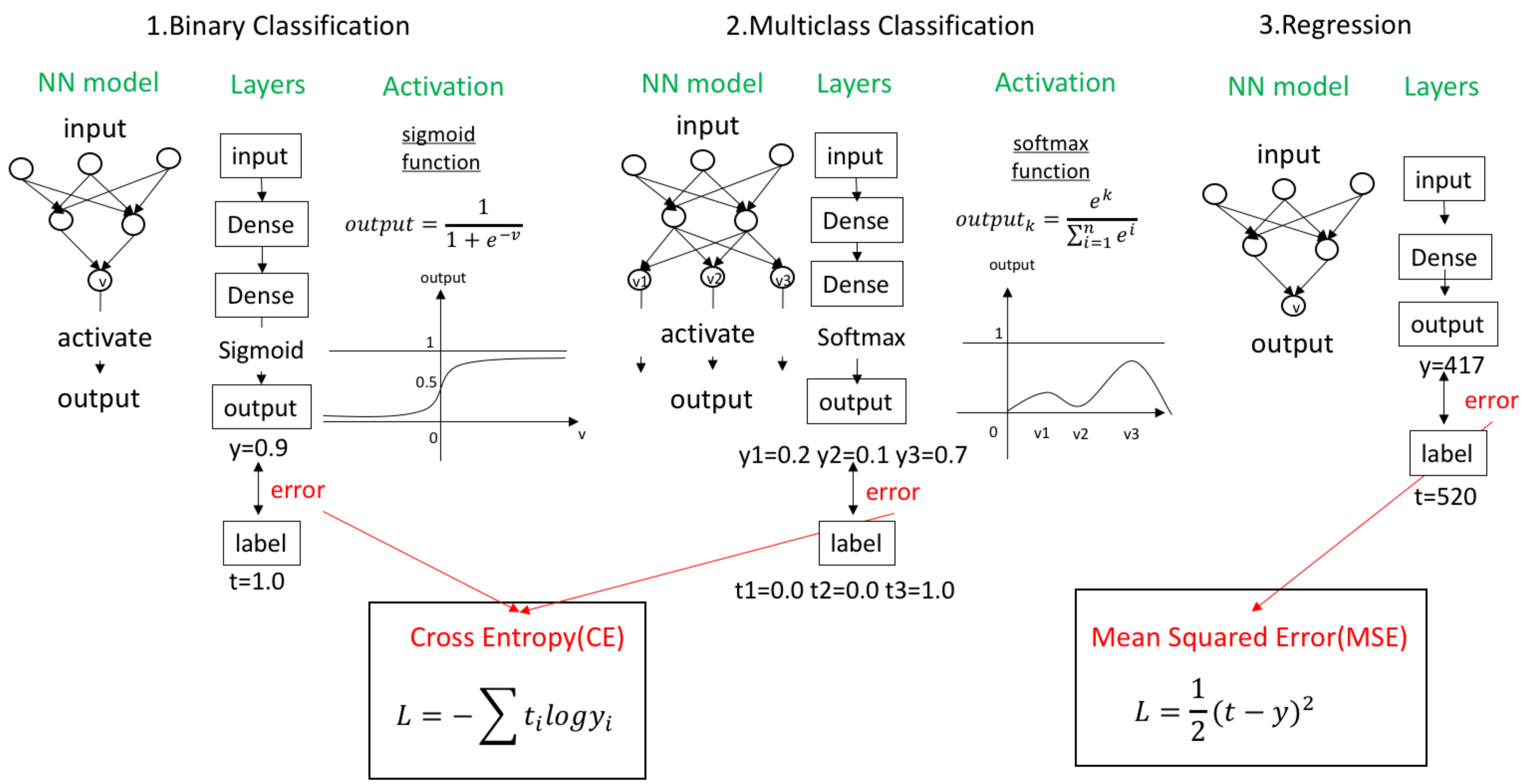
Recap / Questions?

- (supervised) Machine learning tasks
  - Regression
  - Classification
  - Multi-class classification
    - Output many things at once!

layer_dense(unit = 1, activation = "sigmoid") + loss = "binary_crossentropy"

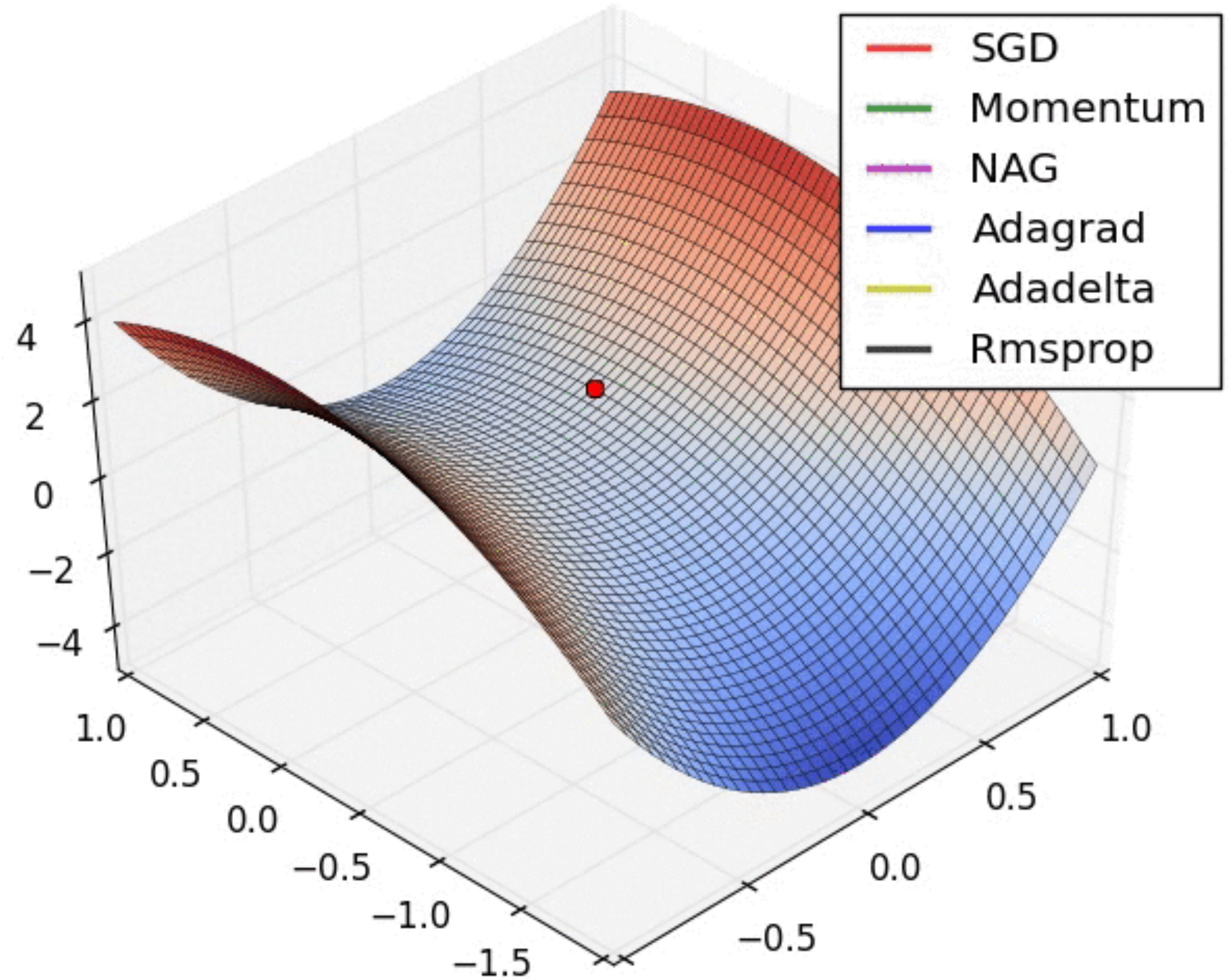
layer_dense(unit = 10, activation = "softmax") + loss = "categorical_crossentropy"

layer_dense(unit = 1) + loss = "mse"



1.Binary Classification
2.Multiclass Classification
3.Regression
NN model
Layers
Activation
input
activate
output
input
Dense
Dense
Sigmoid
output
y=0.9
error
label
t=1.0
sigmoid function
$output = \frac{1}{1+e^{-v}}$
output
1
0.5
0
v
NN model
Layers
Activation
input
activate
output
input
Dense
Dense
Softmax
output
y1=0.2 y2=0.1 y3=0.7
error
label
t1=0.0 t2=0.0 t3=1.0
softmax function
$output_k = \frac{e^k}{\sum_{i=1}^{n} e^i}$
output
1
0
v1
v2
v3
NN model
Layers
input
output
input
Dense
output
y=417
error
label
t=520
Cross Entropy(CE)
$L = -\sum t_i log y_i$
Mean Squared Error(MSE)
$L = \frac{1}{2}(t-y)^2$

Recap / Questions?



- Improving networks
  - Optimisers

Recap / Questions?

- Improving networks
    - Batch normalisation

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

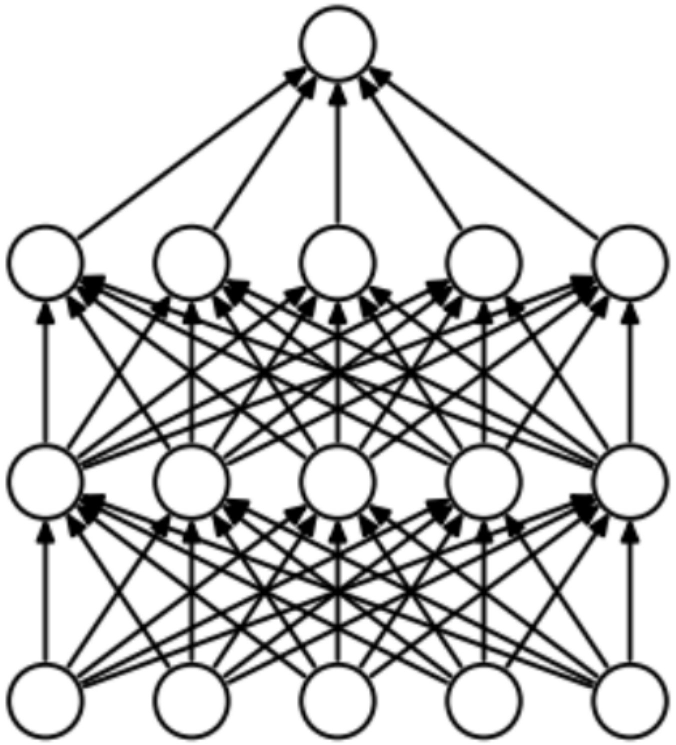
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

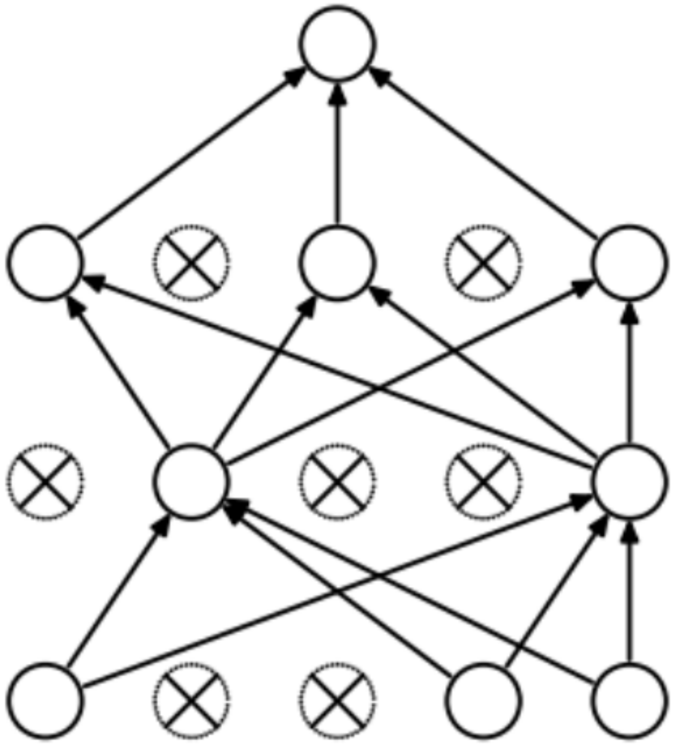
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Recap / Questions?

- Reducing overfitting
  - Dropout



(a) Standard Neural Net



(b) After applying dropout.

Overview

Today we will cover

- Regularisation
  - L2 regularisation
- Practice BN, Dropout and L2
- Sequential modelling
  - Understanding sequential data
  - Basic sequential model (Recurrent Neural Network, RNN)
- Practice working with sequential data

Regularisation

Reducing overfitting

- What is regularisation?
- Any kind of technique which helps you select one model over another using a structured approach.
- We will add extra terms to the loss function (L2)
- We will add intermediary layers to the network (Dropout)

L2 Regularisation

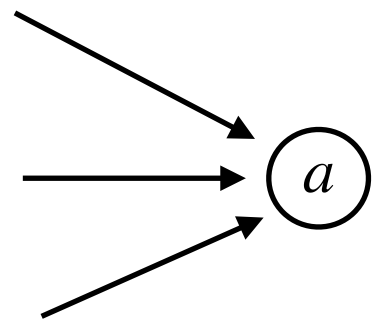
Reducing overfitting

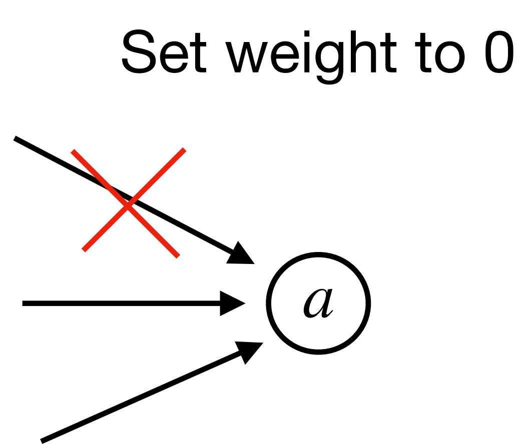
- We add a new term to the total loss function.
- This term adds additional loss to the function which takes the value of the weights into account.
- We then optimise this new loss function instead.
- A new **hyperparameter**, λ is added. This is usually a small value and we will need trial and error to find an acceptable value. It can be considered as a discount factor.

L2 Regularisation

Reducing overfitting

- Why does L2 regularisation work?
- We some cost to the weights, thus making a "**more complex**" model "**more expensive**".
- If some learnt weight is high (say, 10) it "costs more" than a weight with value 1.
- Thus, our model becomes "simpler" by **forcing the weights down**.
- When some weights are forced to 0, we are effectively "**removing connections**".

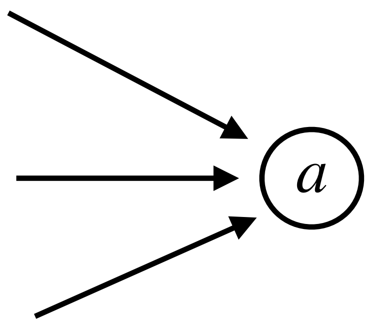




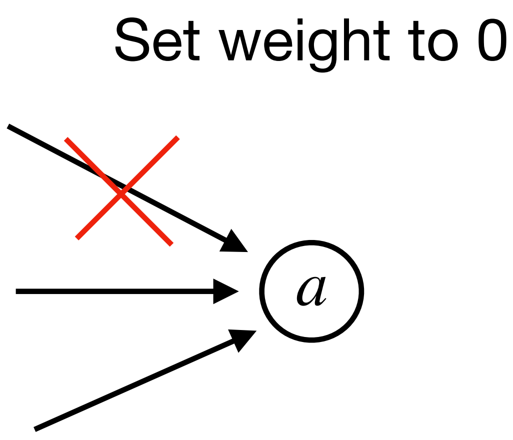
L2 Regularisation

Reducing overfitting

- We use L2 regularisation to fight overfitting, because it makes our model less expressive.



- We use it **after we have fitted the data**.
- It will **increase** the **training loss** during training and **hopefully reduce** the **test loss**.



- Also known as **weight decay**.

```
layer_dense(unit = 256, activation = "relu", kernel_regularizer = regularizer_l2(l = 0.000001))
```

L2 Regularisation

Reducing overfitting

$$
\begin{tikzcd}
& \boldsymbol{w} \arrow[d] & & \\
& f(x_i, \boldsymbol{w}) \arrow[r] & \hat{y}_i \arrow[r] & l(\hat{y}_i, y_i) \\
x_i \arrow[ur] & & y_i \arrow[ur] &
\end{tikzcd}
$$

Total loss $= J(\boldsymbol{w}) = \frac{1}{n}\sum_i^n (l(f(x_i, \boldsymbol{w}), y_i)$

Now becomes

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_i^n (l(f(x_i, \boldsymbol{w}), y_i) + \boxed{\frac{\lambda}{2n}\sum_j w_j^2}$$

Hands-on

Go to https://dba.projects.sda.surfsara.nl/

Notebook: `04a-regularisation.ipynb`

Break at 11:00 / 15:00

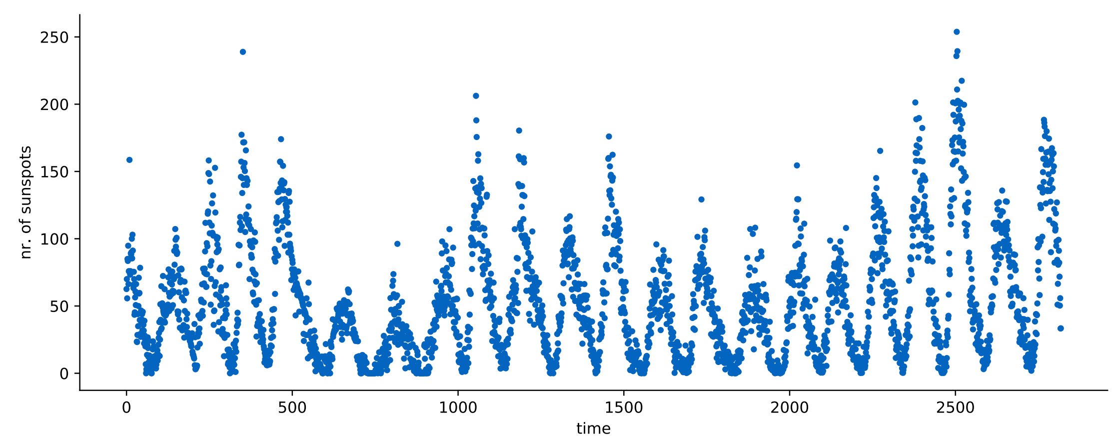
Second part at 11:10 / 15:10

Notebook recap

- We were not really able to improve the baseline much, but made it converge faster.
- We saw that we really need to test if the regularisation technique is helping us.
  - L2 regularisation was not very stable. Dropout was better.
- It depends on the task, architecture, ..., trial and error.

Sequential data

- Data is sequential when the data has some order.
- The whole dataset can consist of a single order (sunspots) or many individual orders (sentences).



the cat sat on the mat .
the book is open .

Sequential data

in deep learning

- Machine translation

"Hoi, hoe gaat het?" → "Hi, how are you?"

- Speech recognition

→ "Hi, how are you?"

- Music generation

Some style / nothing →

- Sentiment classification

"Hoi, hoe gaat het?" → ★★★★★

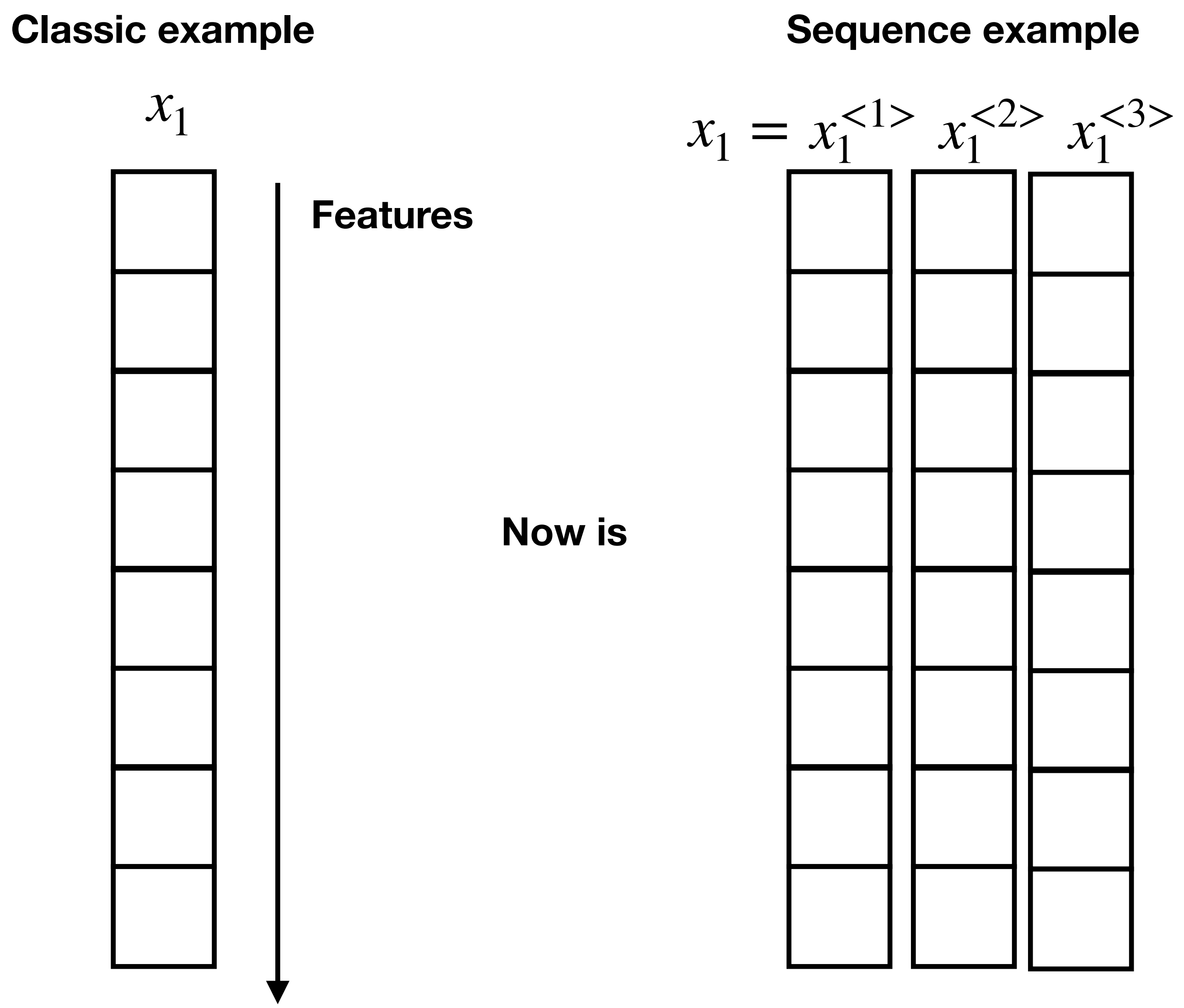
- Video activity recognition

→ Running

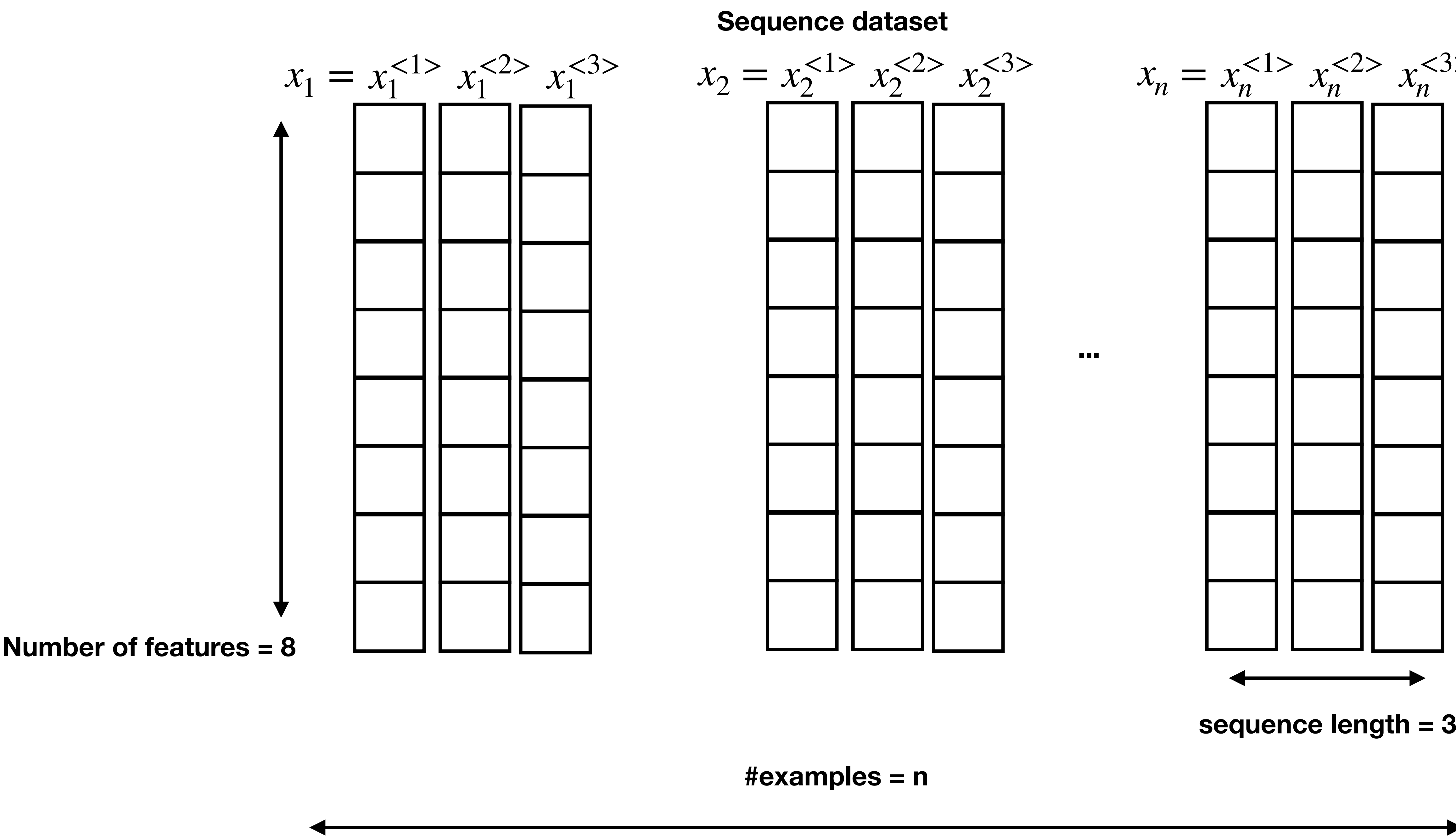
- ...

Sequential data

- In previous lectures our data have been made up from a single example.
- A single example can have many features.
  - Temperature, air pressure, etc.
- Now each example is made from a single sequence.
  - "Hi, hoe gaat het?"
- Each sequence has many examples.
  - "Hi", "hoe", "gaat", "het"
- That is, in each iteration we process a single sequence, many examples.
  -

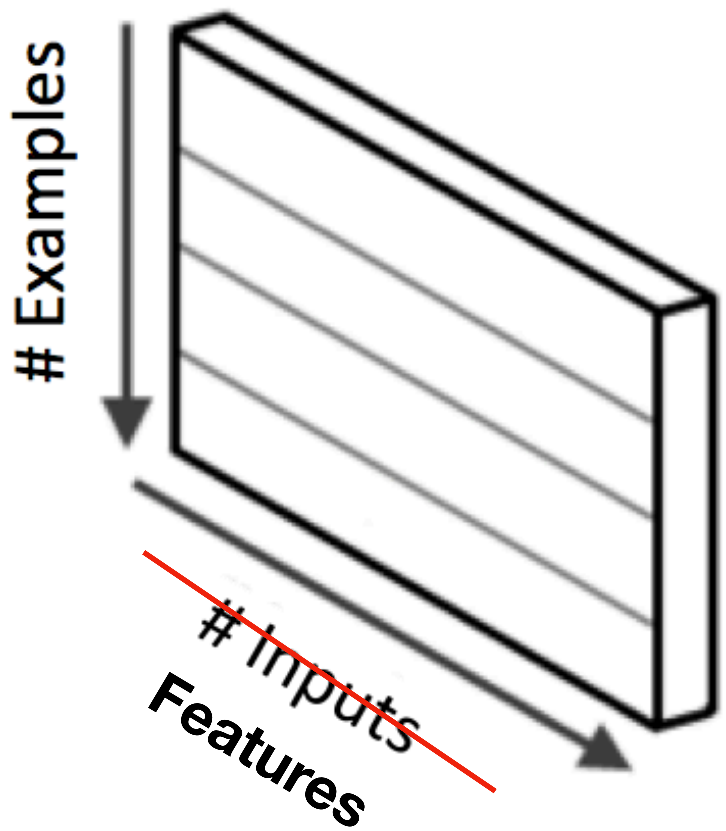


Sequential data



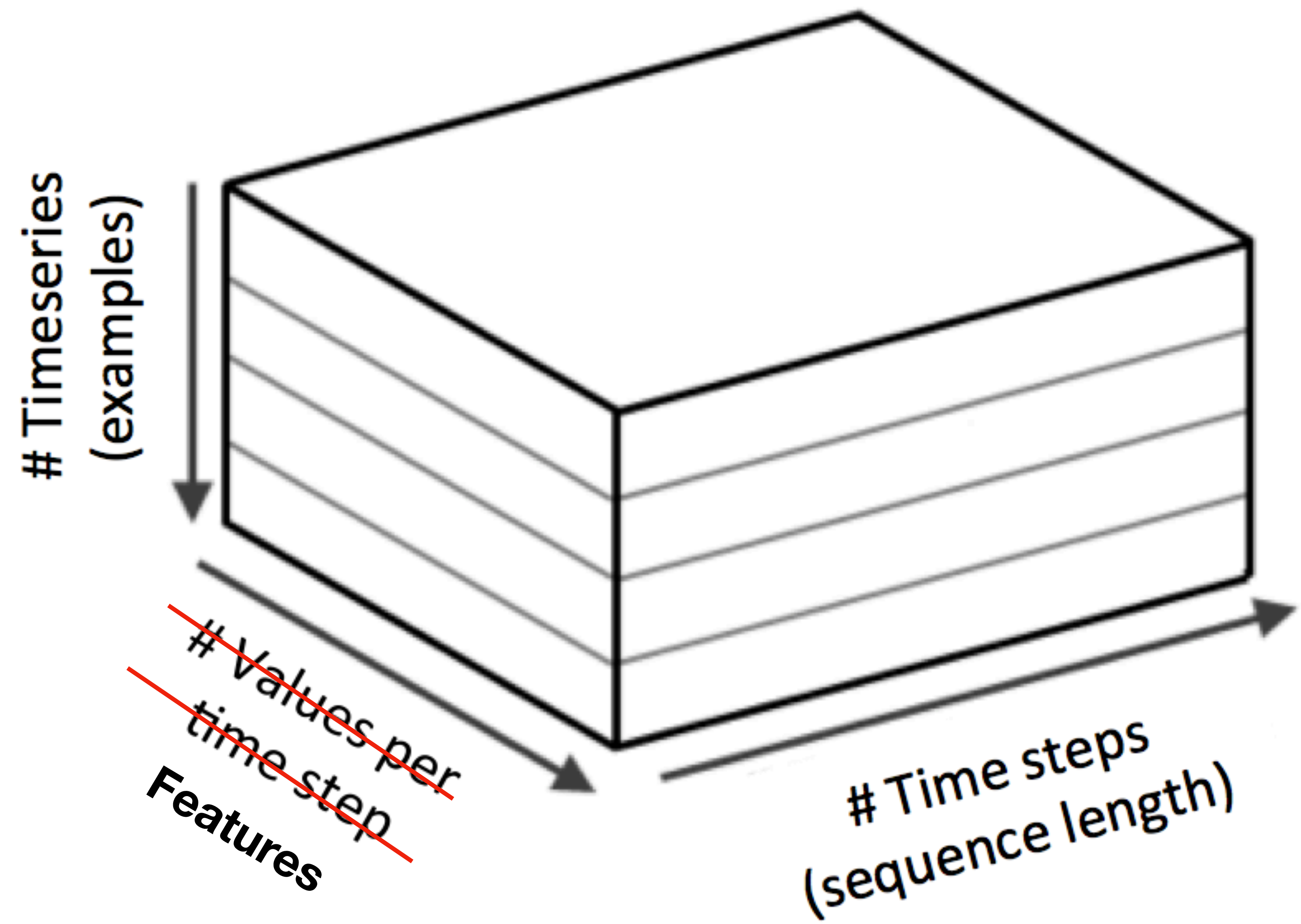
Data = (examples, sequence_length, features)

Feed-Forward Network Data



Data = (examples, features)

Recurrent Network Data

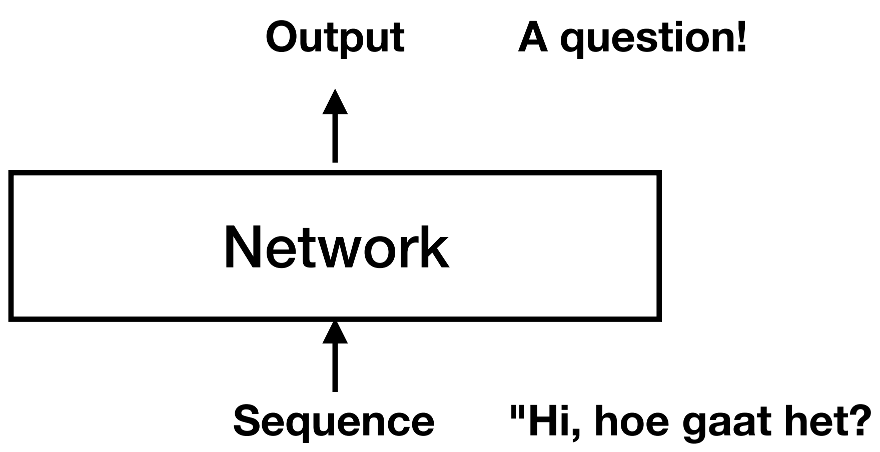


Data = (examples, sequence_length, features)

Source: https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html

Which task?

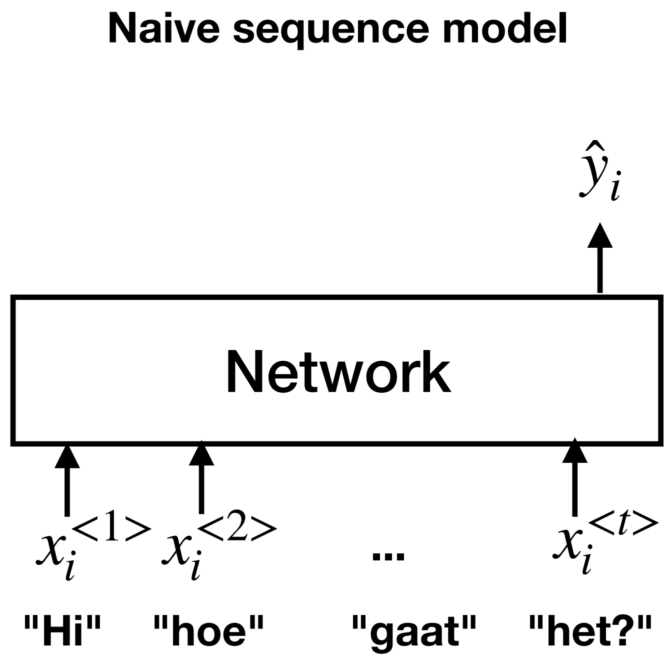
- Given a sequence we can try to solve many supervised learning tasks.
  - Regression
    - Predict temperature tomorrow given the last few days.
  - Classification
    - Is this a question?
    - Is the person yelling?
  - Just add the output layer required along with the loss.



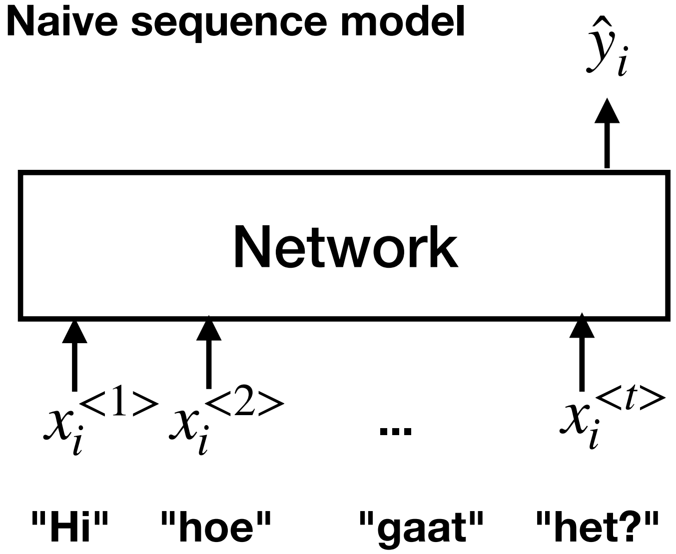
How to model?

Naive model

- Why not create a network which accepts multiple inputs at once?
  - For this we will need VERY many parameters.
  - The sequence length might vary between examples.
  - We can often process each element independently and equally.
    - "gaat" is the same word regardless of position in a sentence.



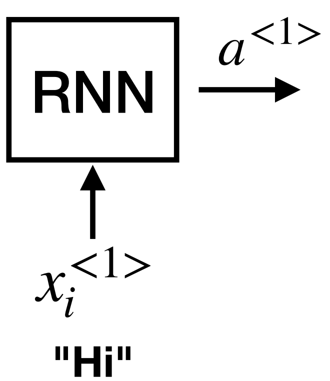
How to model?



- A more clever approach is to use the same, smaller, network for each element in the sequence.

- Then we pass information to the next time step.

- An RNN cell.



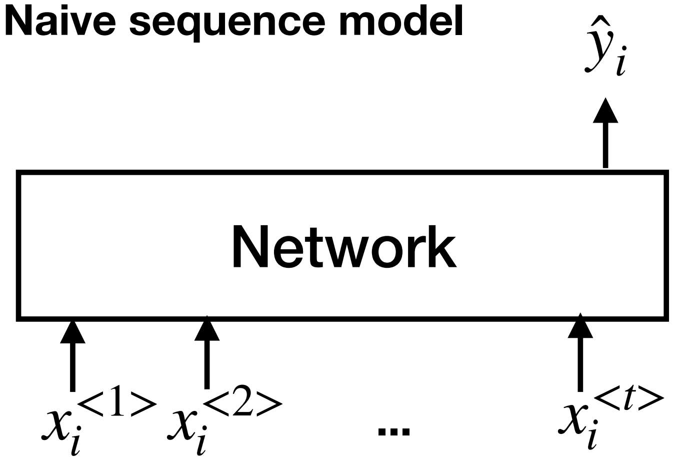
Basic RNN

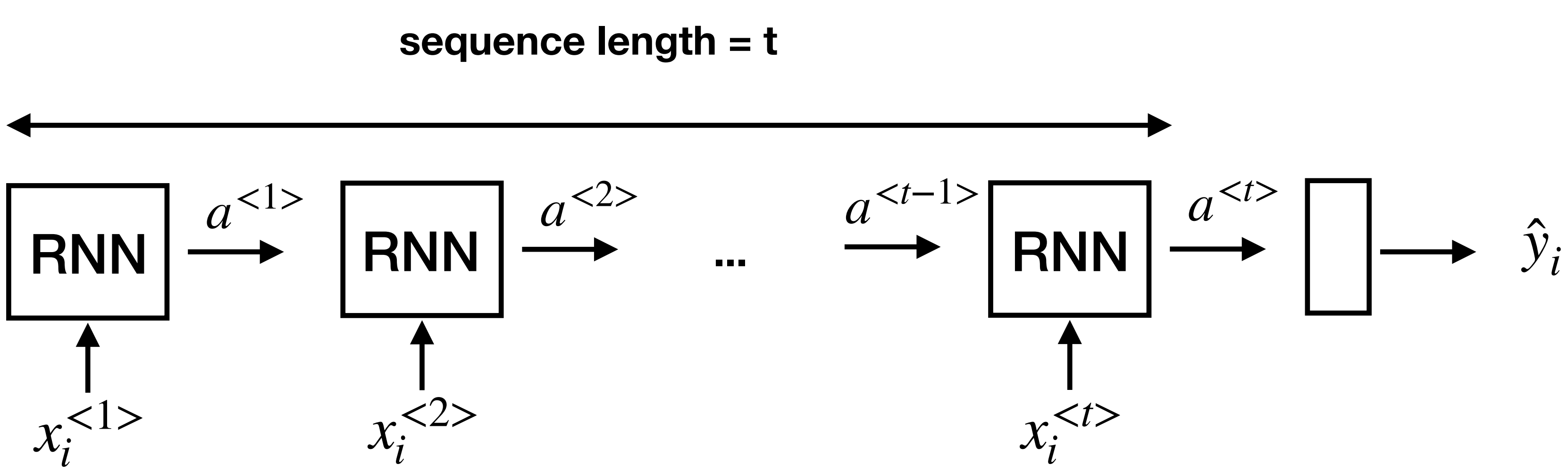
passing information

- Read the sequence, step by step.

- Until the sequence has been read.

Naive sequence model



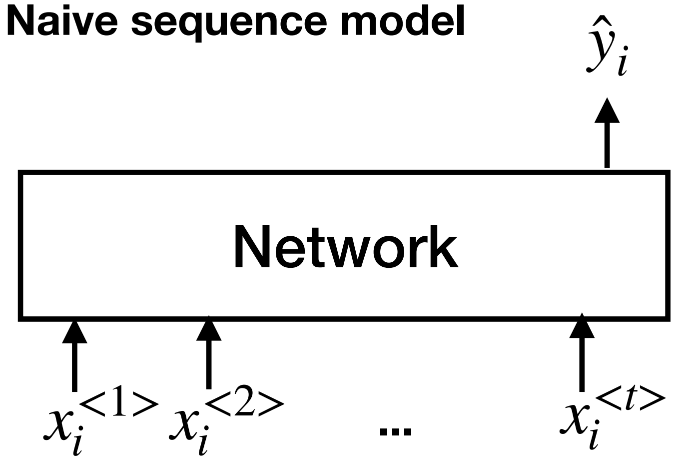


Basic RNN

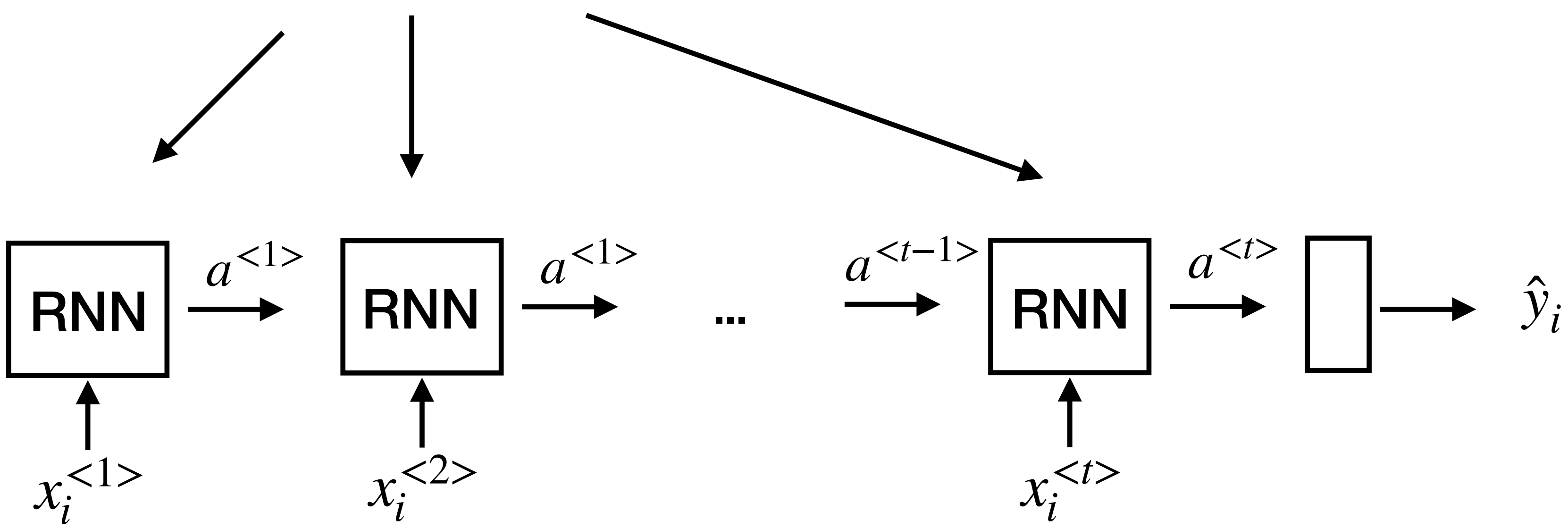
passing information

- Use the same network for each time-step.
- RNN cell contains the parameters

Naive sequence model



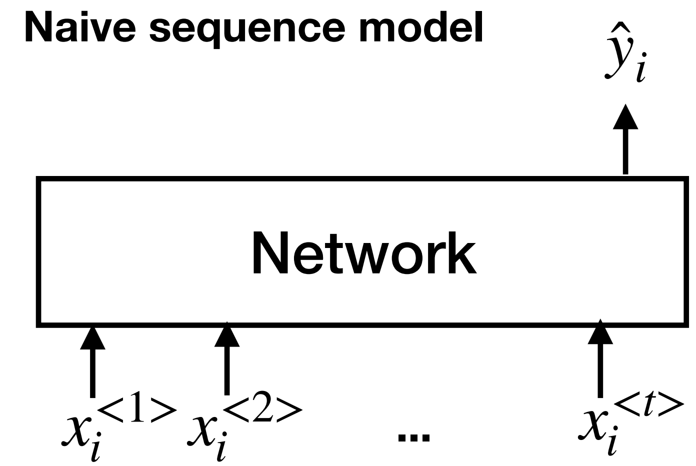
The same network, with the same weights, replicated

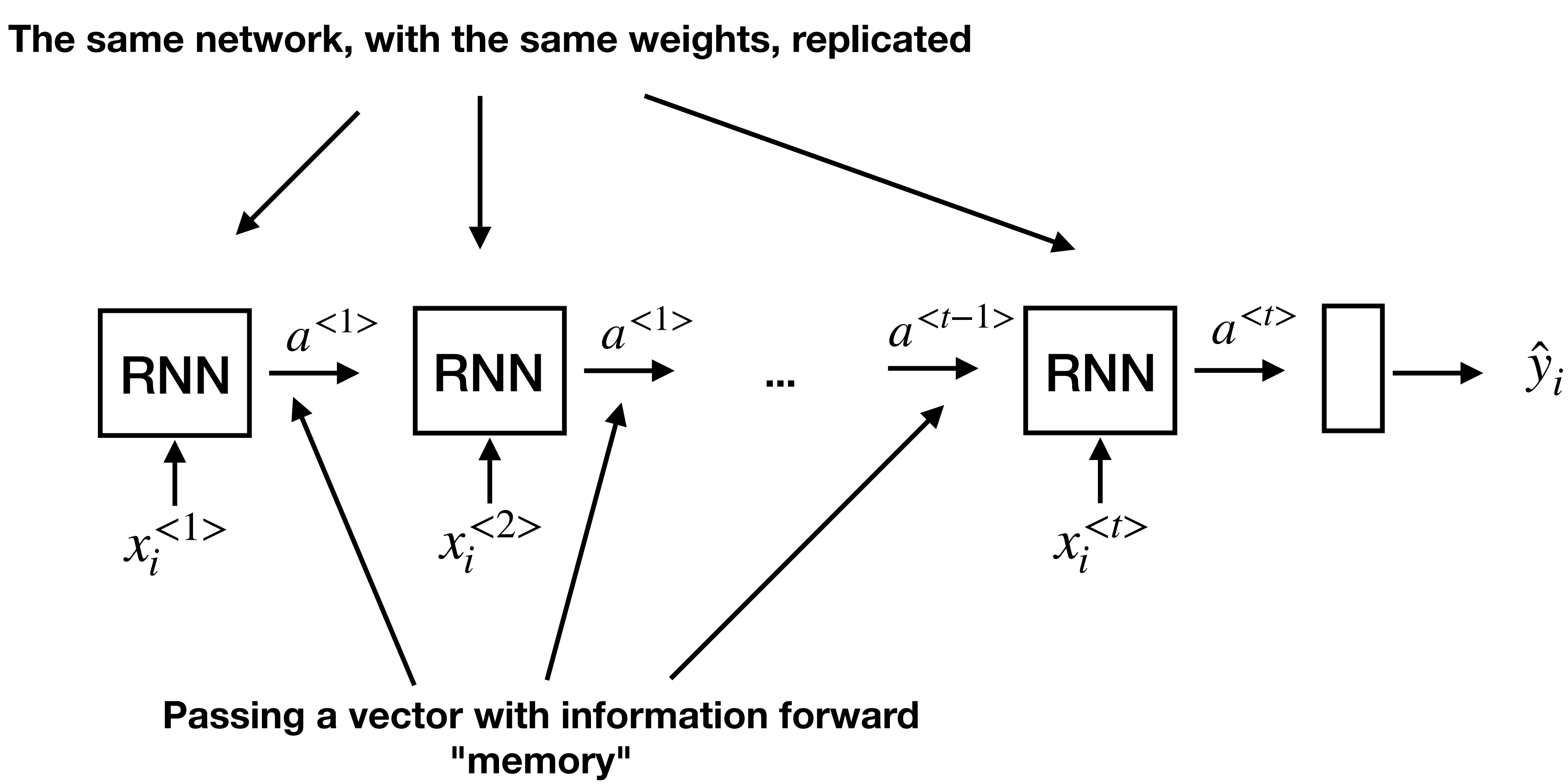


Basic RNN

passing information

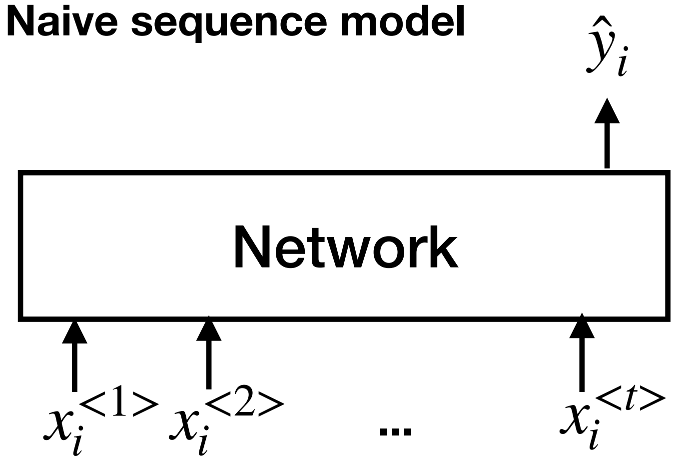
- Pass information forward in a "memory".
- "Memory" is updated each step.



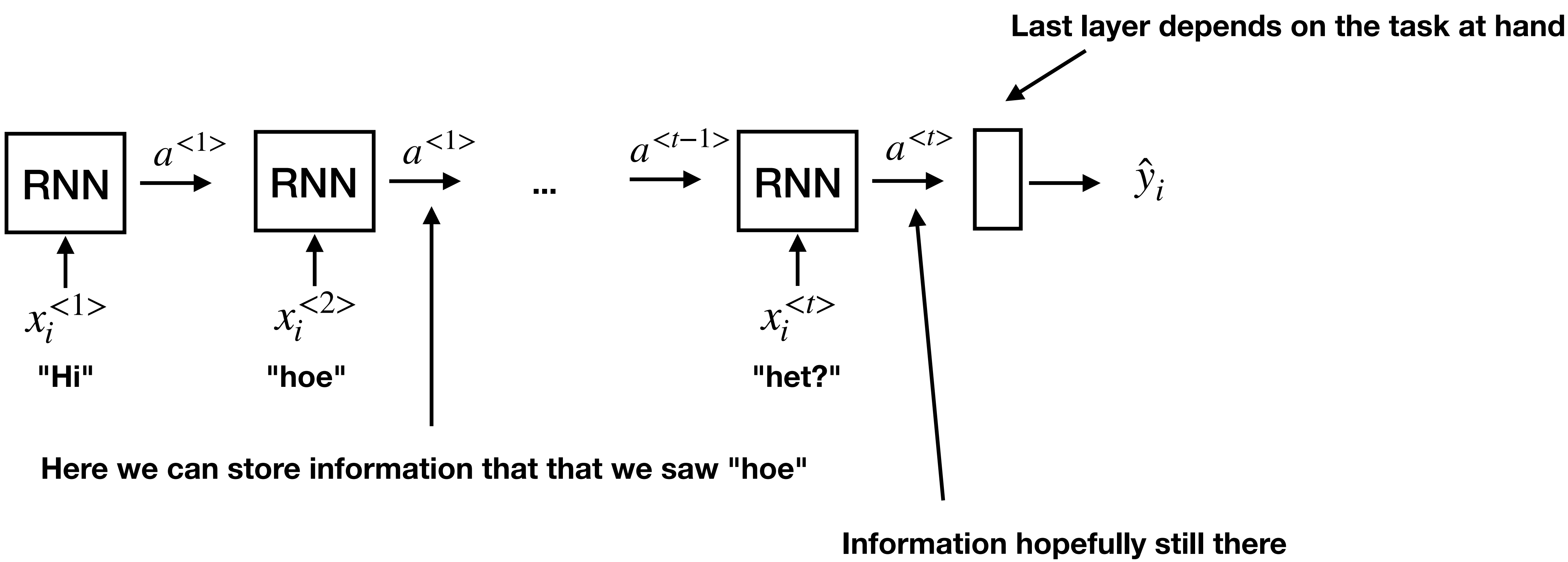


Basic RNN

passing information



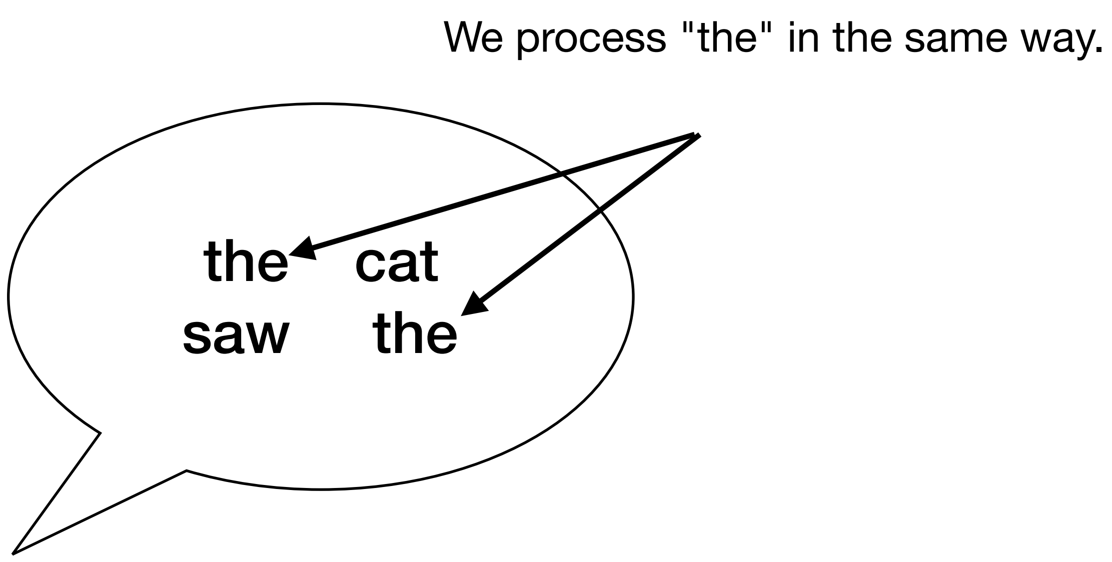
- Hopefully good information is stored.
- Task is just the last layer.



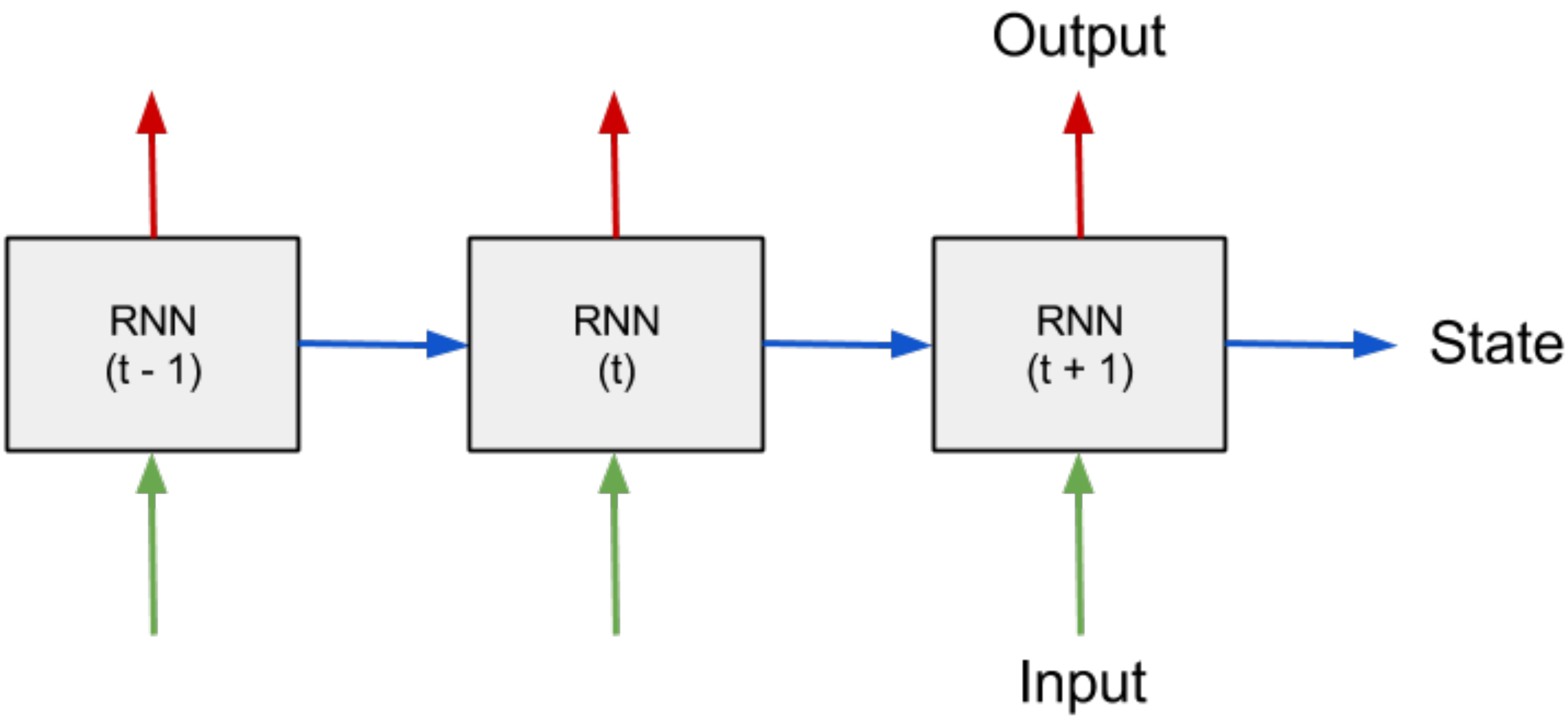
Basic RNN

sharing parameters

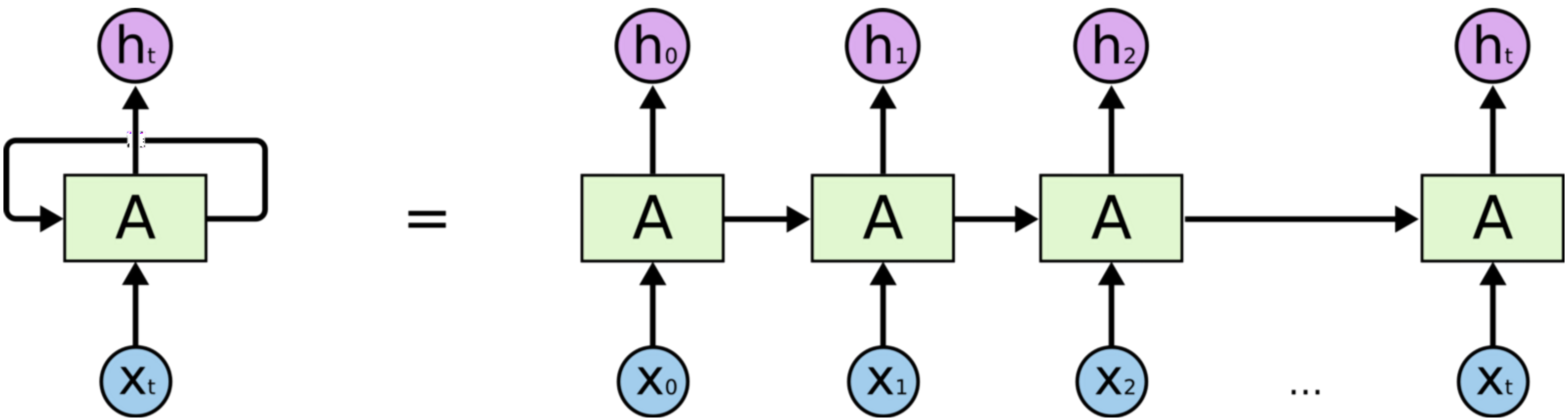
- For each time-step we use the **same network**, it just gets **different "memory"** passed to it.
- This allows us to **share parameters** in different locations of the model.
- Sharing parameters **reduces** the number of **parameters** in the model.
- More intuitive and works better.
- The idea behind an RNN.




```
layer_simple_rnn(units = 12, input_shape = c(sequence_length, features))
```



Source: https://medium.com/@erikhallstrm/hello-world-rnn-83cd7105b767



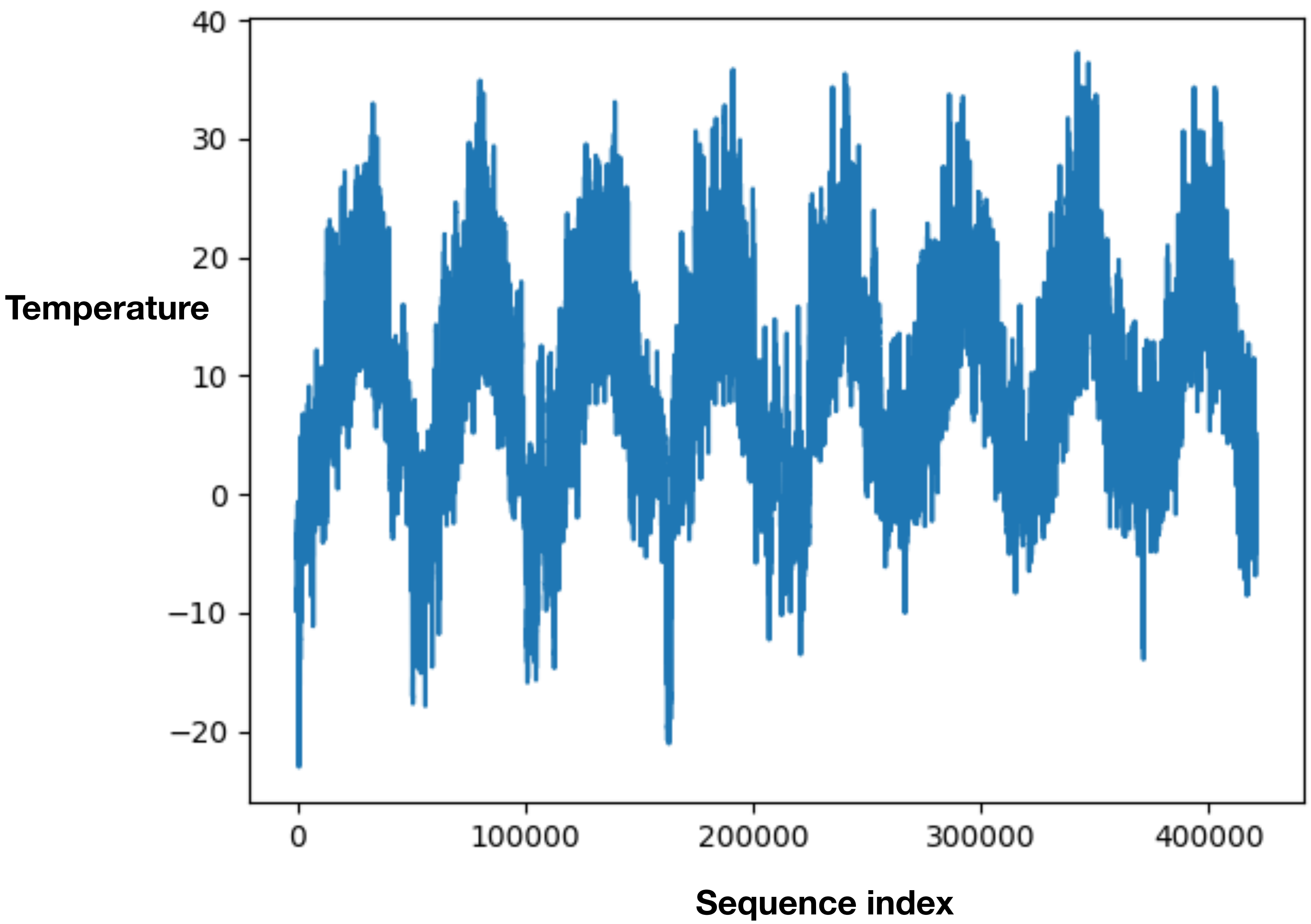
Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Summary

- Sequential data is **ordered** to it.
- Sequential data needs to be processed using dimensions, **(examples, sequence length, features)**.
- We feed each time-step of the sequence into the **same RNN cell** and **remember** what we have seen.
- The **last layer** still takes care of the **task**.
- Allows us to **share parameters**.

Notebook

- The Jena weather dataset.
- A long sequence of weather measurements.
- Each measurement consists of 15 variables.
- We see a time-dependent pattern in the data.



Hands-on

Go to https://dba.projects.sda.surfsara.nl/

Notebook: `04b-time-series-prediction.ipynb`

Wrap-up at 12:20 / 16:20

Summary

- Regularisation
    - L2 regularisation
- Practice BN, Dropout and L2
- Sequential modelling
    - Understanding sequential data
    - Basic sequential model (Recurrent Neural Network, RNN)
- Practice working with sequential data