Recurrent Neural Networks

Machine Learning for Behavioral Data

April 8, 2025

Today’s Topic

Week	Lecture/Lab
1	Introduction
2	Data Exploration
3	Regression
4	Classification
5	Model Evaluation
6	Time Series Prediction
7	Time Series Prediction
8	**Time Series Prediction**

Supervised learning on time series:

- **Probabilistic graphical models**
- **GLMMs**
- **Neural networks: LSTM, GRU, etc.**

Getting ready for today's lecture…

- **If not done yet**: clone the repository containing the Jupyter notebook and data for today's lecture into your Noto workspace
- SpeakUp room for today's lecture:

https://go.epfl.ch/speakup-mlbd2025

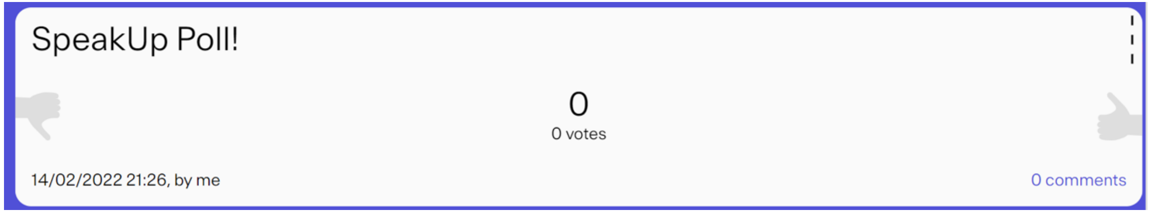
Short quiz about the past…

This KT model uses the # of opportunities the student had per skill and treats prior successes and failures the same.

a) Additive Factors Model (AFM)

b) Performance Factors Analysis (PFA)

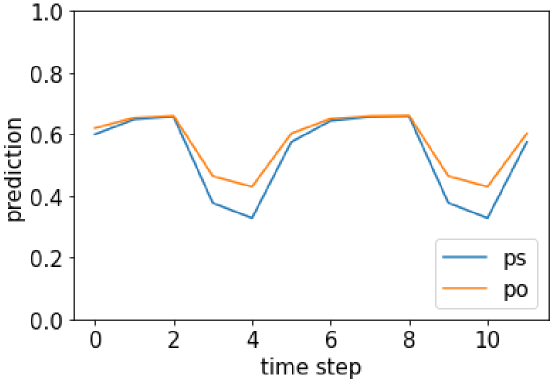
c) Bayesian Knowledge Tracing (BKT)



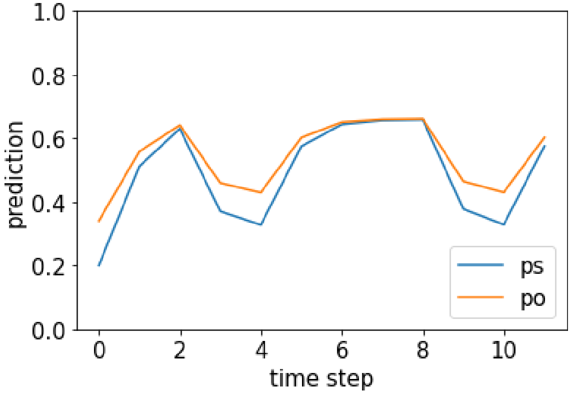
https://go.epfl.ch/speakup-mlbd2025

Short quiz about the past…

Which BKT parameter has been changed between the left and right plot (exactly one)?
The corresponding observations are: [1,1,0,0,1,1,1,0,0,1,1]



$$p_0 = 0.6,\quad p_s = 0.1,\quad p_g = 0.2,\quad p_l = 0.3,\quad p_f = 0.3$$



a) p_g (guess probability)

b) p_l (probability of learning)

c) p_0 (initial probability)

d) p_f (forget probability)

https://go.epfl.ch/speakup-mlbd2025

Knowledge Tracing – Predicting Future Performance

Subtraction 0-100

1	2		•••		n	n+1
0	0	1	0	1	1	?

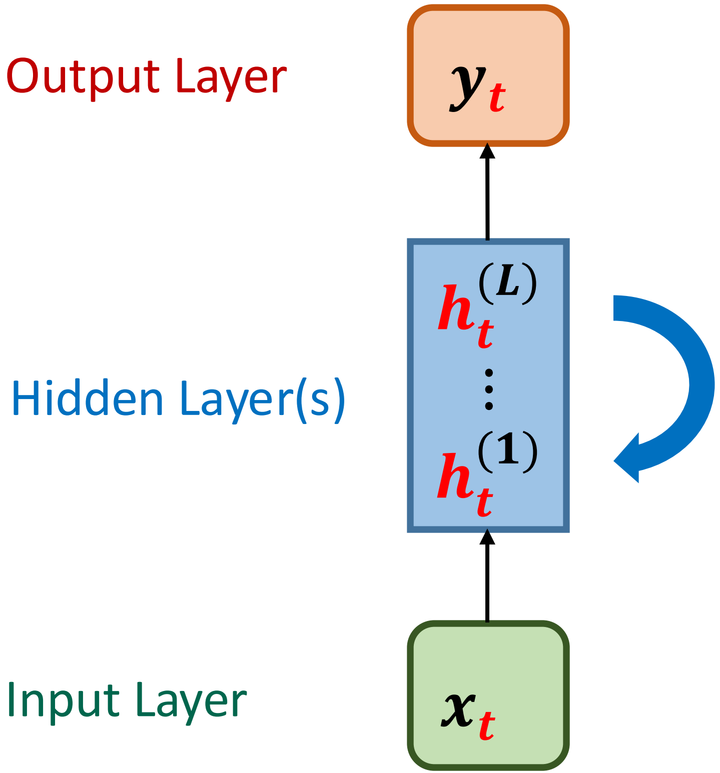
Today – Recurrent Neural Networks

- **Deep Knowledge Tracing**
- Parameters and hyperparameter tuning
- Different architectures
- Different tasks:
  - "Many-to-many" versus "Many-to-one"
  - Classification versus Regression

Neural Networks

- Neural networks are able to represent non-linear functions, i.e. $y_n \approx f(\boldsymbol{x}_n)$ can be non-linear
- Neural networks are able to *learn* the features and the weights (parameters) from the data
- Tutorial: https://go.epfl.ch/tutorial-nn

Recurrent Neural Network



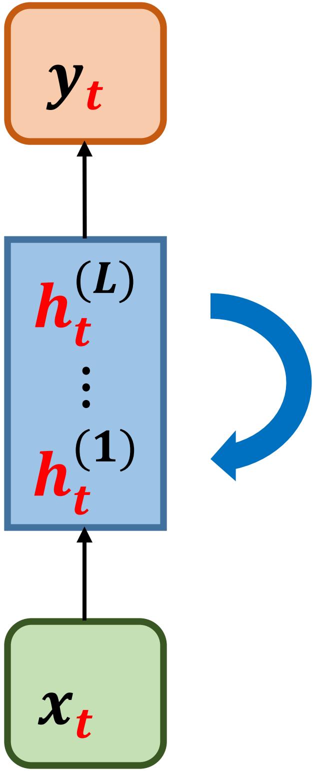
$$\boldsymbol{h}_t^{(1)} = \phi_h \left(W_{hx}\boldsymbol{x}_t + W_{h^{(1)}h^{(1)}}^{(1)} \boldsymbol{h}_{t-1}^{(1)} + \boldsymbol{b}^{(1)}\right)$$

Recurrent Neural Network

Output Layer

Hidden Layer(s)

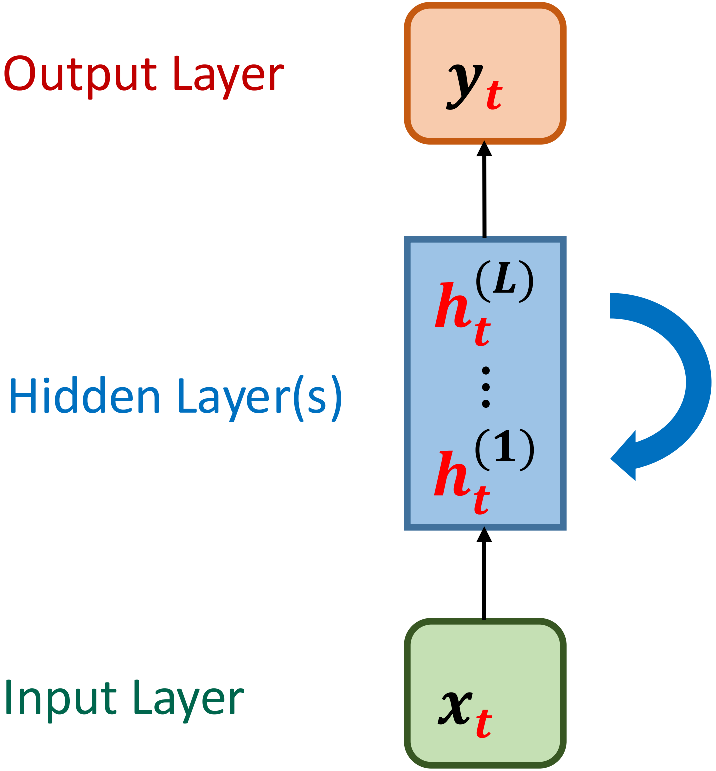
Input Layer



$$\boldsymbol{h}_t^{(l)} = \phi_h(W_{h^{(l)}h^{(l-1)}}^{(l)}\boldsymbol{h}_t^{(l-1)} + W_{h^{(l)}h^{(l)}}^{(l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}^{(l)})$$

$$\boldsymbol{h}_t^{(1)} = \phi_h\left(W_{hx}\boldsymbol{x}_t + W_{h^{(1)}h^{(1)}}^{(1)}\boldsymbol{h}_{t-1}^{(1)} + \boldsymbol{b}^{(1)}\right)$$

Recurrent Neural Network

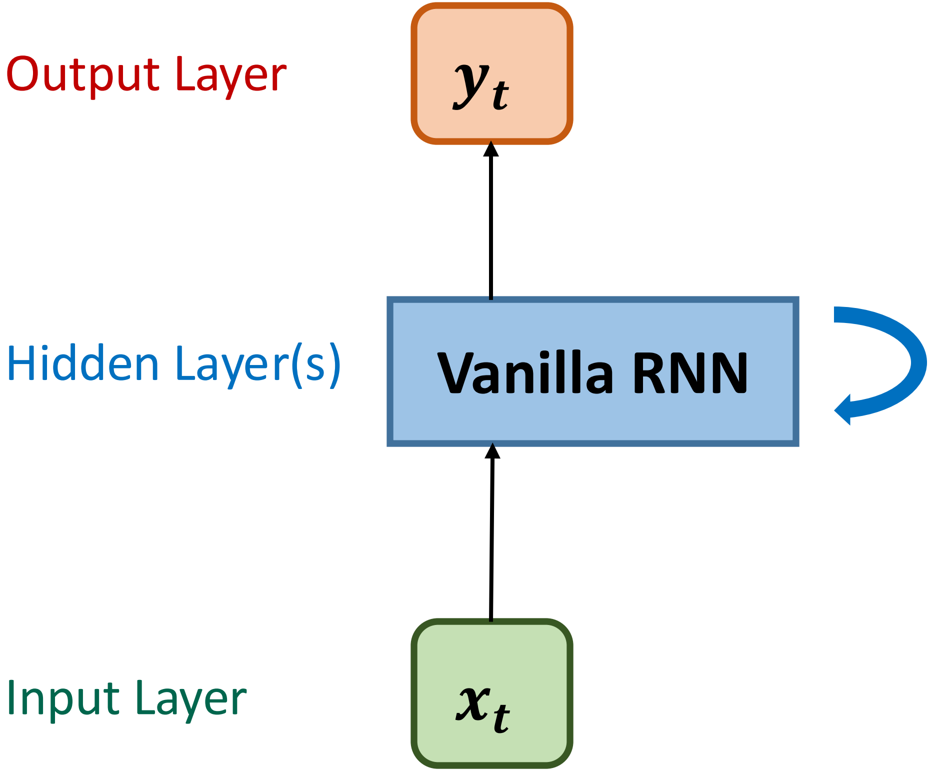


$$\boldsymbol{y_t} = \phi_y(W_{yh}\boldsymbol{h}_t^L + \boldsymbol{b}^{(y)})$$

$$\boldsymbol{h}_t^{(l)} = \phi_h(W^{(l)}_{h^{(l)}h^{(l-1)}}\boldsymbol{h}_t^{(l-1)} + W^{(l)}_{h^{(l)}h^{(l)}}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}^{(l)})$$

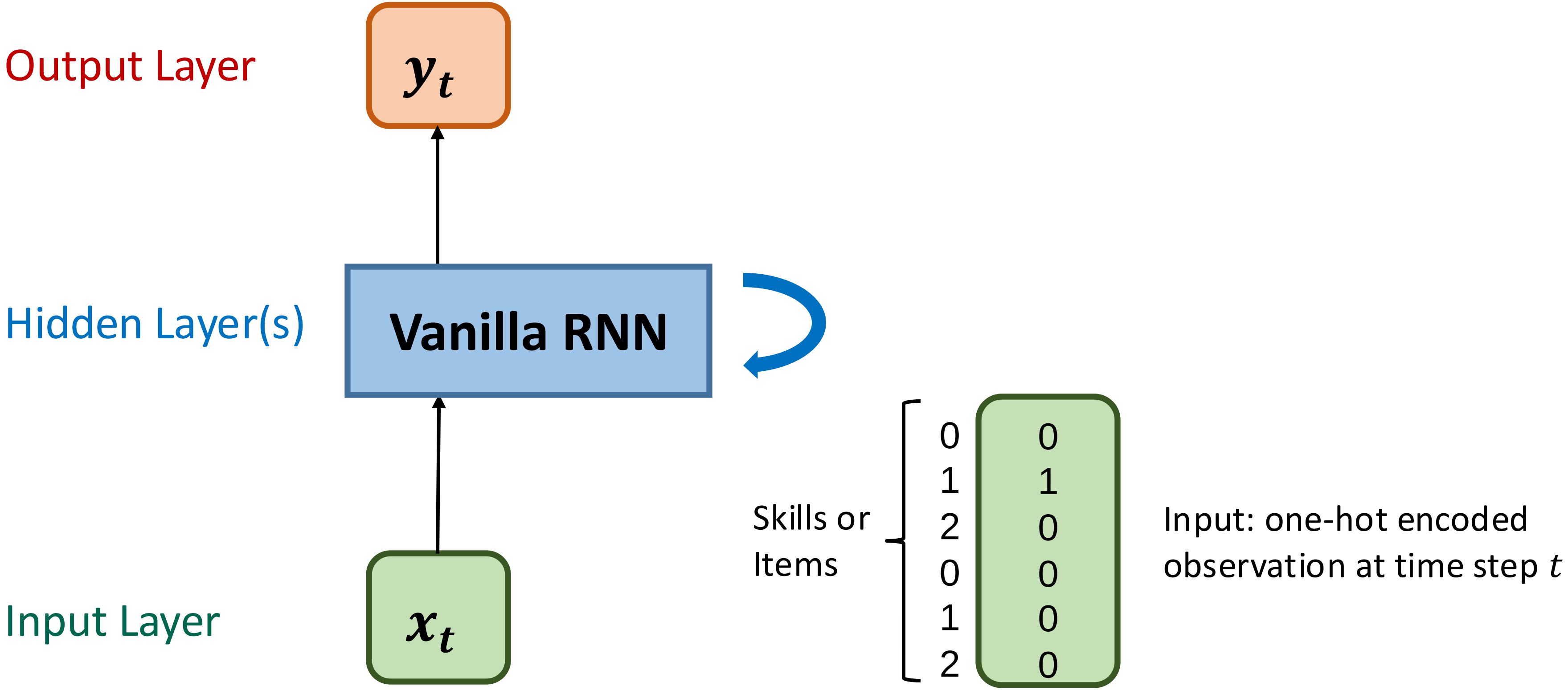
$$\boldsymbol{h}_t^{(1)} = \phi_h\,(W_{hx}\boldsymbol{x}_t + W^{(1)}_{h^{(1)}h^{(1)}}\boldsymbol{h}_{t-1}^{(1)} + \boldsymbol{b}^{(1)})$$

Deep Knowledge Tracing



[*Deep Knowledge Tracing*, Piech et al., NIPS 2015]

Deep Knowledge Tracing



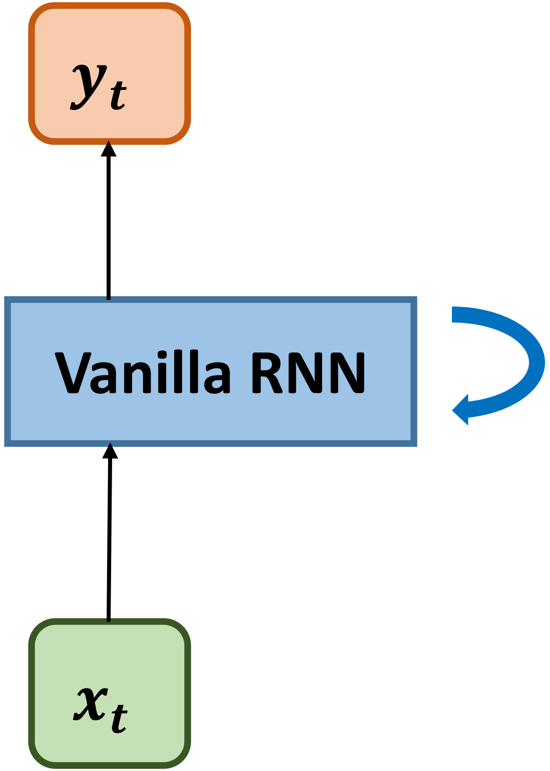
[*Deep Knowledge Tracing*, Piech et al., NIPS 2015]

Deep Knowledge Tracing

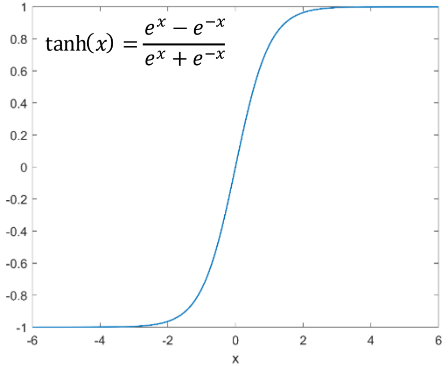
Output Layer

Hidden Layer(s)

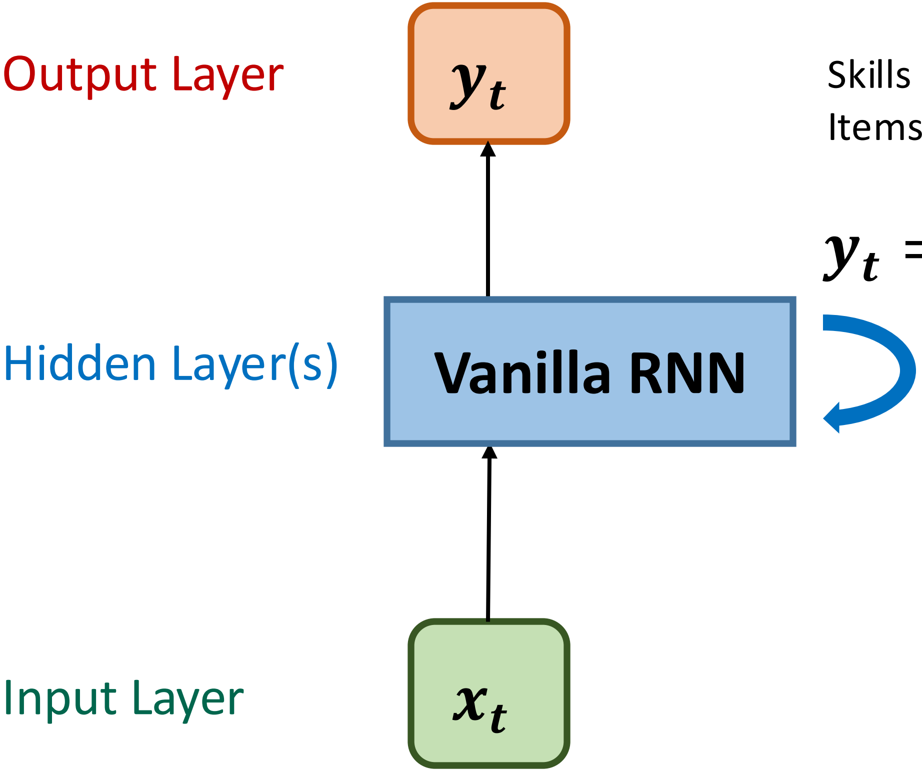
Input Layer



$$\boldsymbol{h_t} = \tanh(W_{hx}\boldsymbol{x_t} + W_{hh}\boldsymbol{h_{t-1}} + \boldsymbol{b_h})$$



Deep Knowledge Tracing

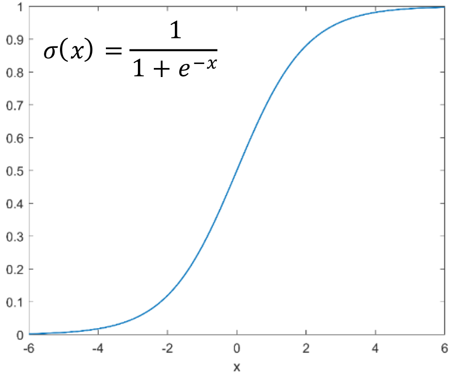


Skills or Items

0	0.3
1	0.5
2	0.7

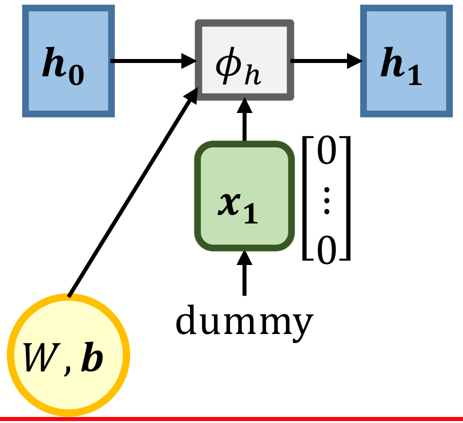
Output: probability for answering skill (or item) correct at time step t

$$\boldsymbol{y}_t = \sigma(W_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

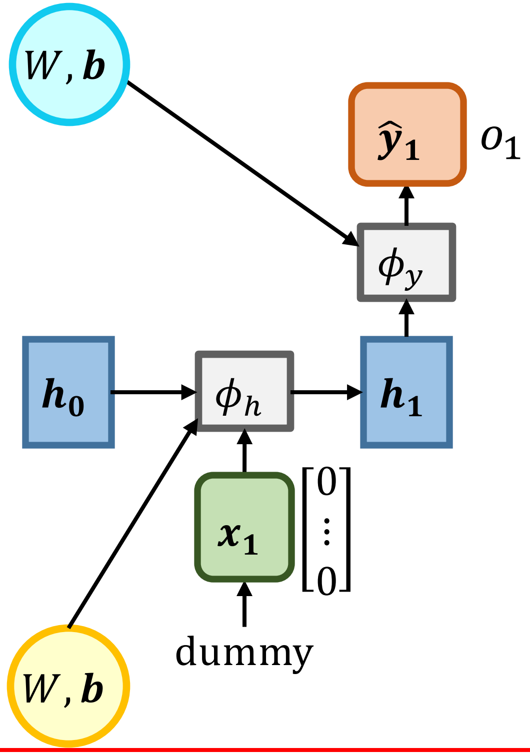


[*Deep Knowledge Tracing*, Piech et al., NIPS 2015]

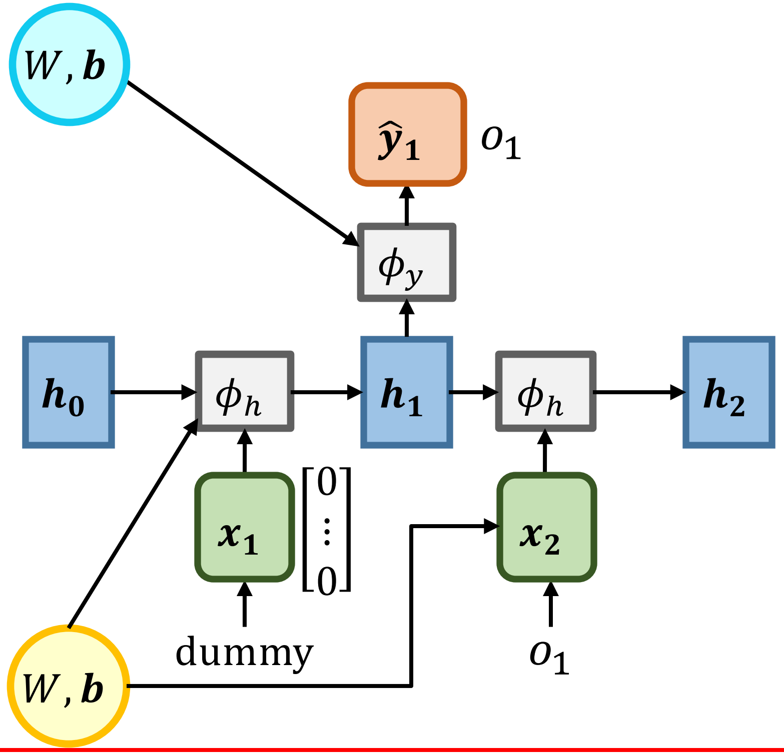
Deep Knowledge Tracing – Computational Graph



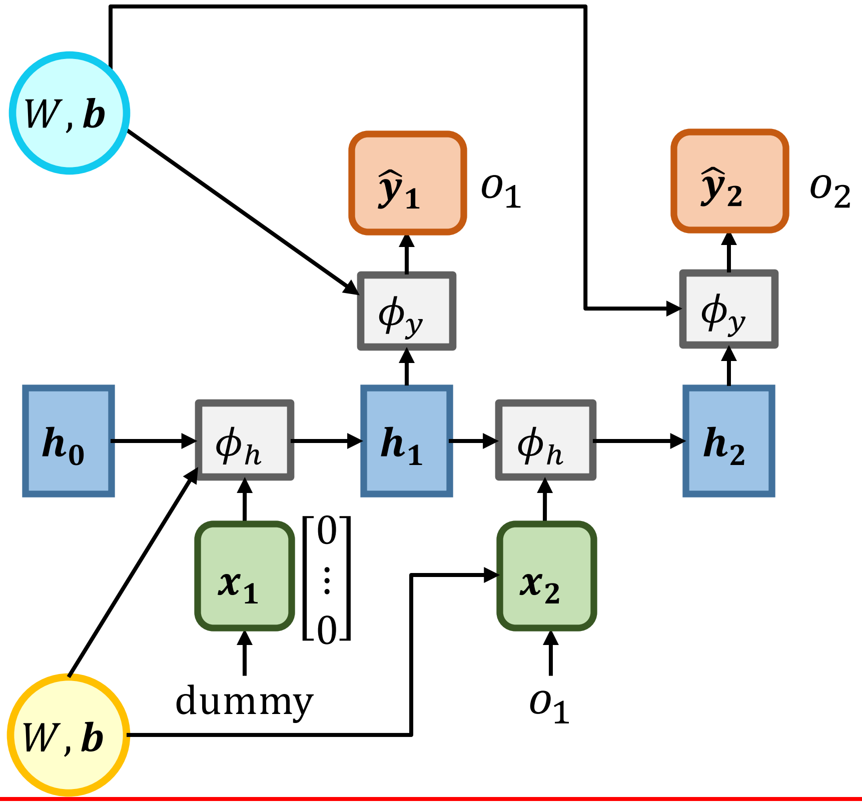
Deep Knowledge Tracing – Computational Graph



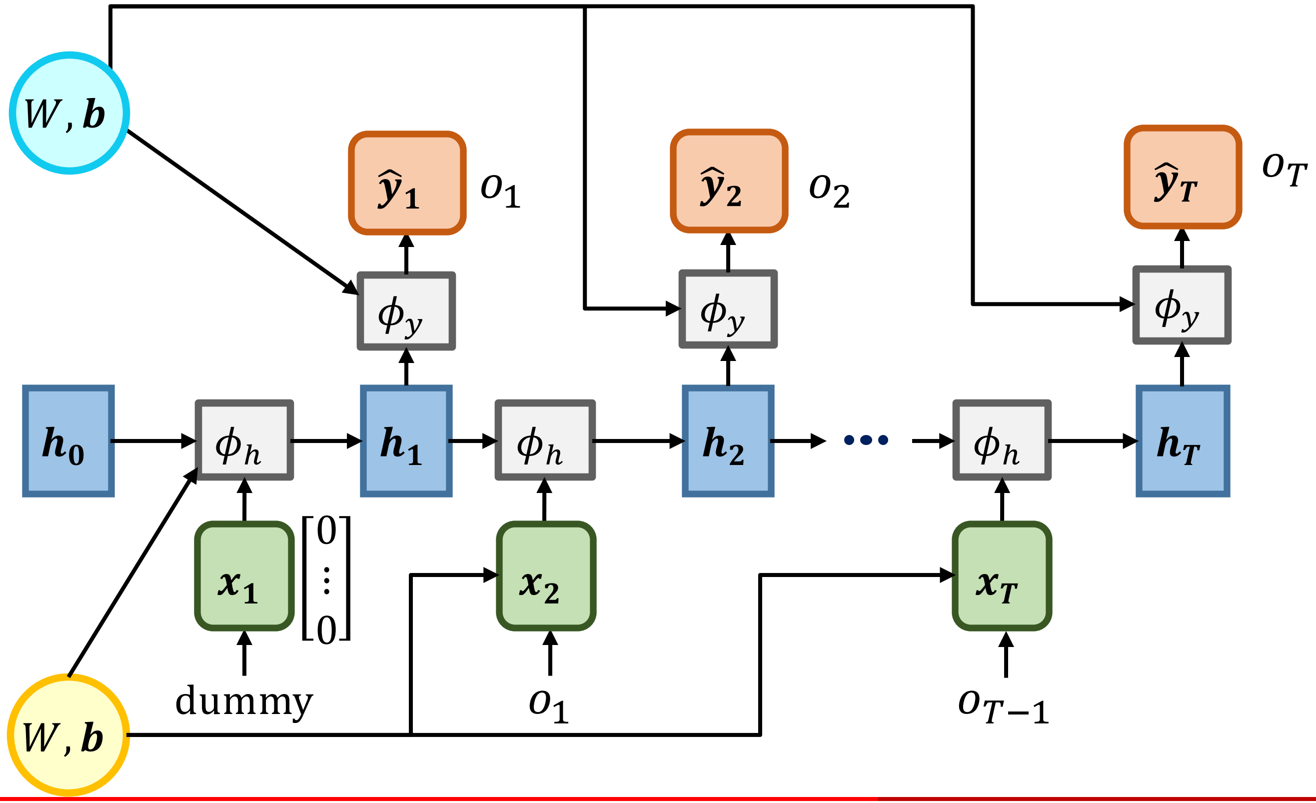
Deep Knowledge Tracing – Computational Graph



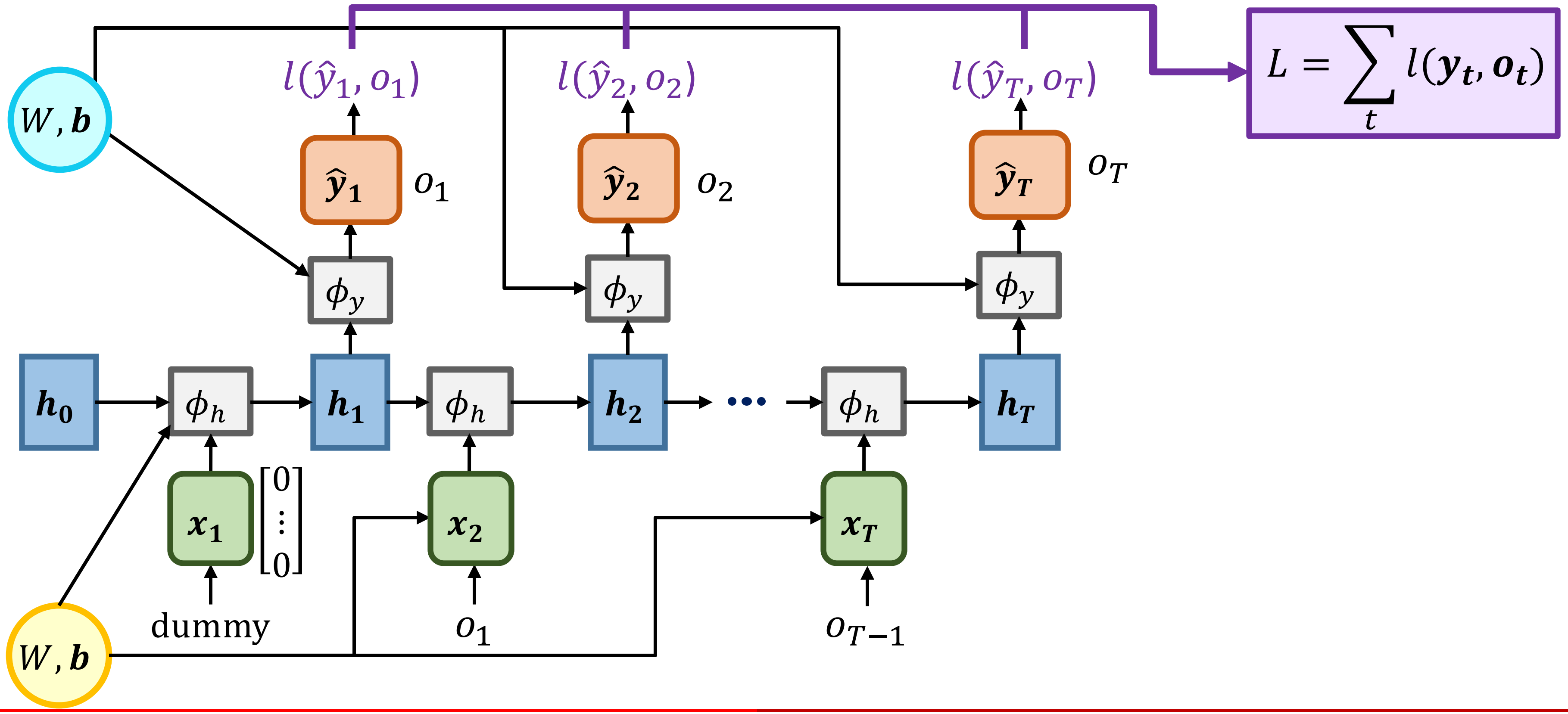
Deep Knowledge Tracing – Computational Graph



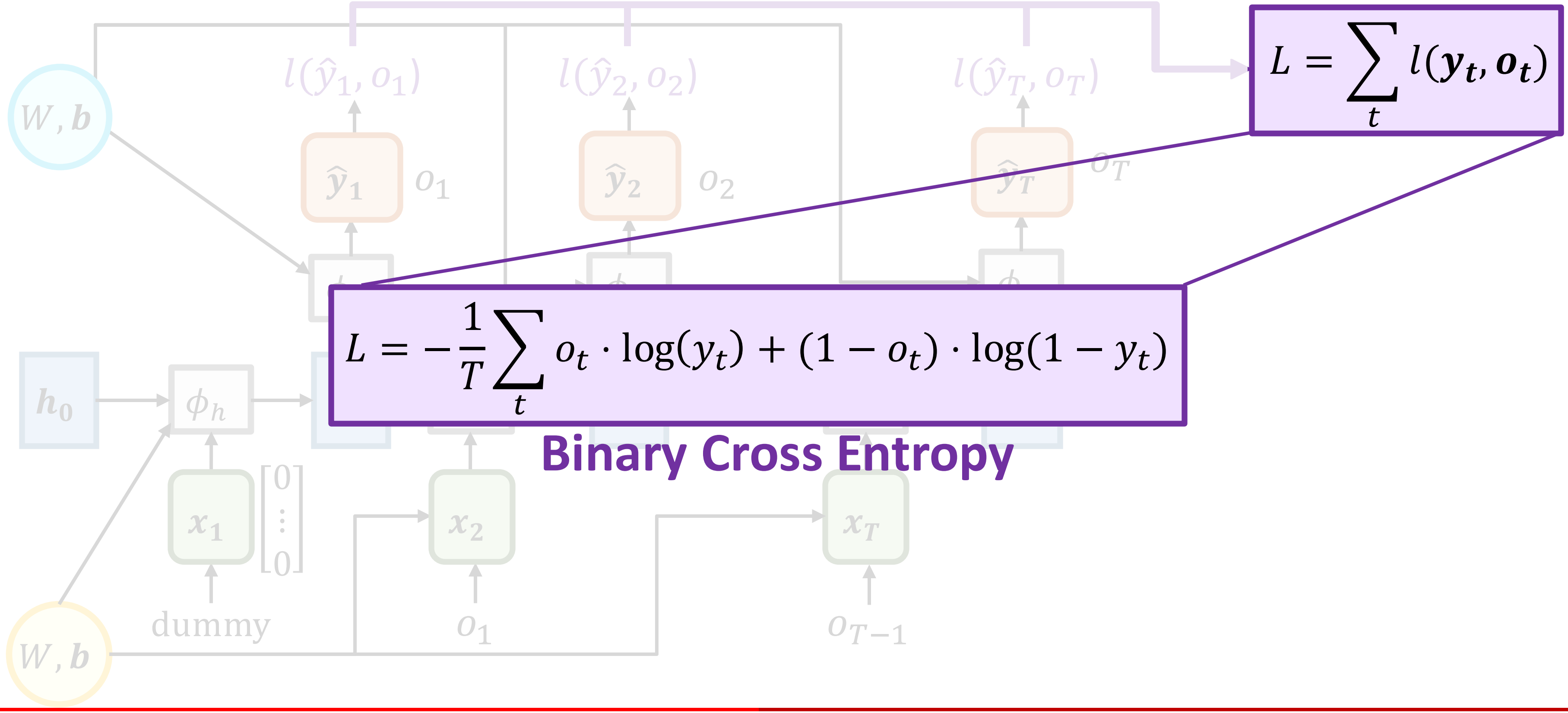
Deep Knowledge Tracing – Computational Graph



Deep Knowledge Tracing – Computational Graph



Training a DKT model: Binary Crossentropy Loss



$$L = \sum_t l(\boldsymbol{y_t}, \boldsymbol{o_t})$$

$$L = -\frac{1}{T}\sum_t o_t \cdot \log(y_t) + (1 - o_t) \cdot \log(1 - y_t)$$

Binary Cross Entropy

Training and Prediction using DKT

- Training: gradient descent
- Prediction: compute inference in the network (see computational graph)

Today – Recurrent Neural Networks

- Deep Knowledge Tracing
- **Parameters and hyperparameter tuning**
- Different architectures
- Different tasks:
  - “Many-to-many” versus “Many-to-one”
  - Classification versus Regression

RNNs – Specifying Parameters

```
[ ]  # Specify the model hyperparameters. Full descriptions included in the demo notebook!
     params = {}

     params['batch_size'] = 32
     params['mask_value'] = -1.0
     params['verbose'] = 1
     params['best_model_weights'] = 'weights/bestmodel'
     params['optimizer'] = 'adam'
     params['recurrent_units'] = 16
     params['epochs'] = 20
     params['dropout_rate'] = 0.1
```

RNNs – Tuning hyperparameters

```
[ ]  # Specify the model hyperparameters. Full descriptions included in the demo notebook!
     params = {}

     params['batch_size'] = 32
     params['mask_value'] = -1.0
     params['verbose'] = 1
     params['best_model_weights'] = 'weights/bestmodel'
     params['optimizer'] = 'adam'
     params['recurrent_units'] = 16
     params['epochs'] = 20
     params['dropout_rate'] = 0.1
```

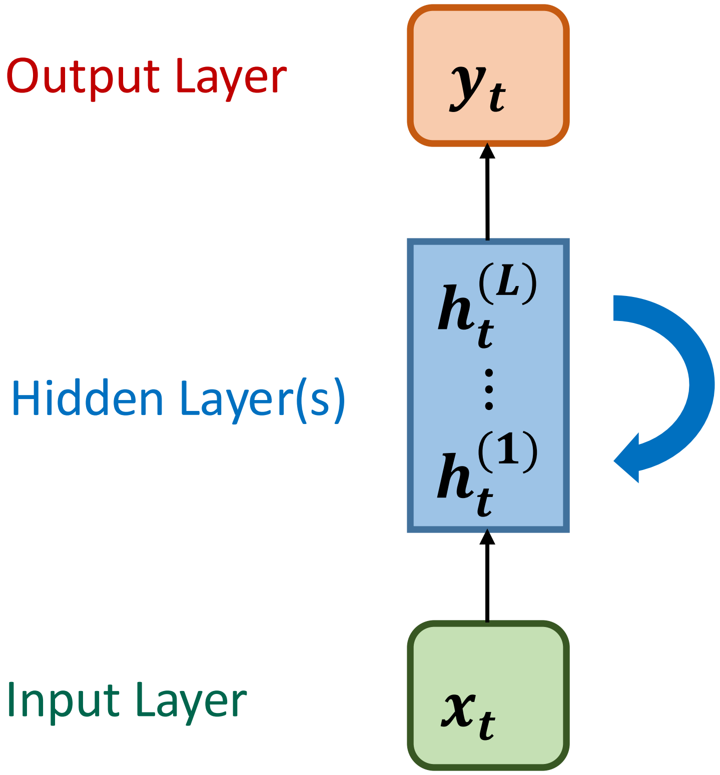
RNNs – Tuning hyperparameters

- Optimal number of epochs can be found using callbacks
- Other parameters can be tuned using for example:
  a) Train-Validation-Test split
  b) Train-Test split, using a k-fold cross validation on the training data to determine the optimal parameters

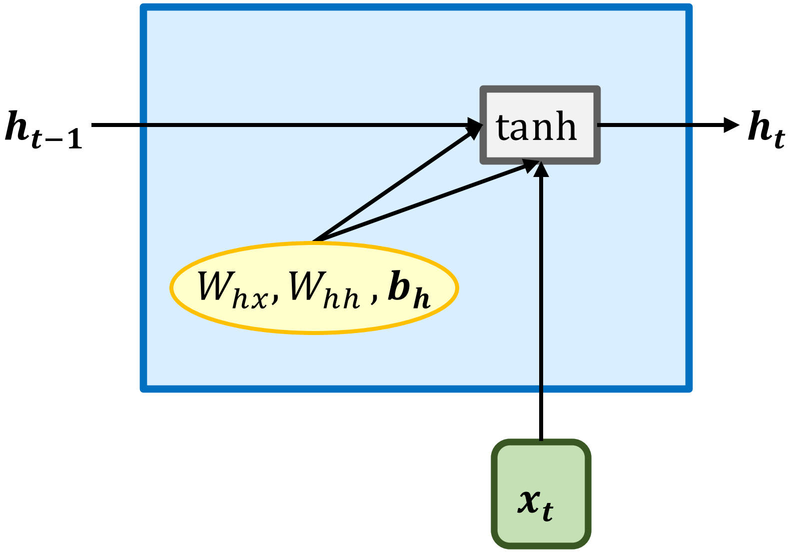
Today – Recurrent Neural Networks

- Parameters and hyperparameter tuning
- **Different architectures**
- Different tasks:
  - “Many-to-many” versus “Many-to-one”
  - Classification versus Regression

Recurrent Neural Network

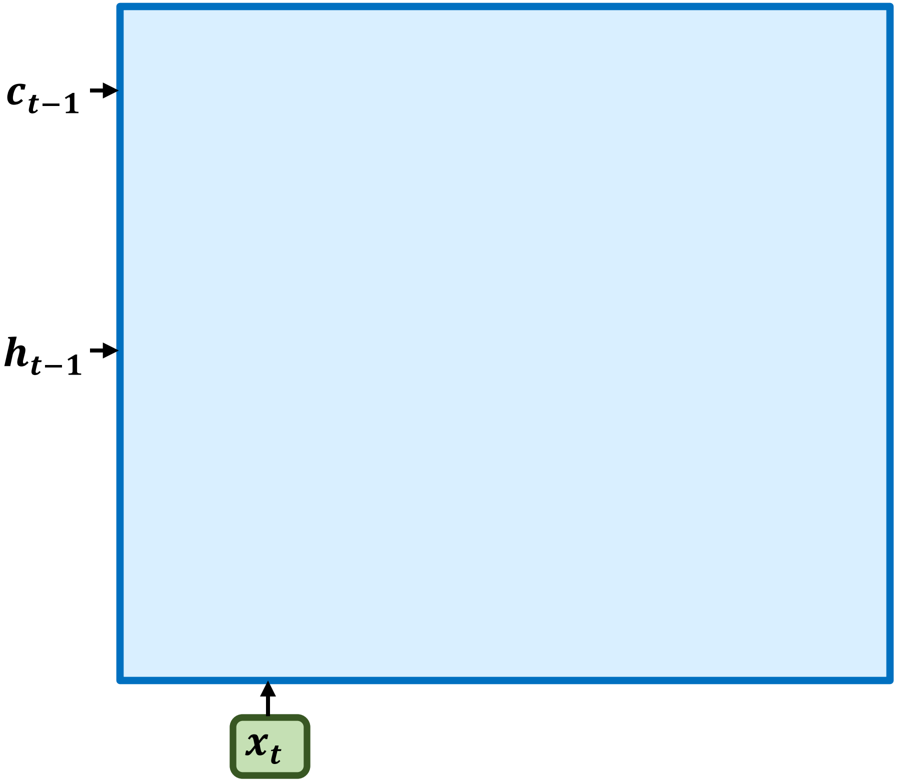


Vanilla RNN - revisited



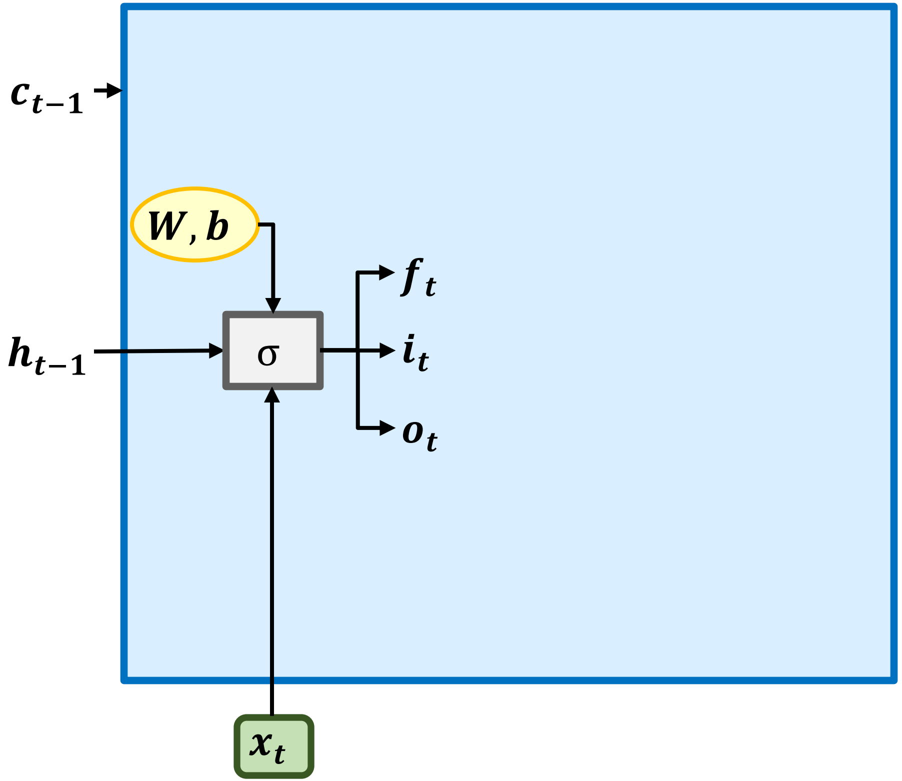
$$\boldsymbol{h}_t = \tanh(W_{hx}\boldsymbol{x}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

Long-Short Term Memory Network (LSTM)



- Two states:
  - Hidden state h_{t-1}
  - Cell state c_{t-1}

Long-Short Term Memory Network (LSTM)



1 Updating the gates:

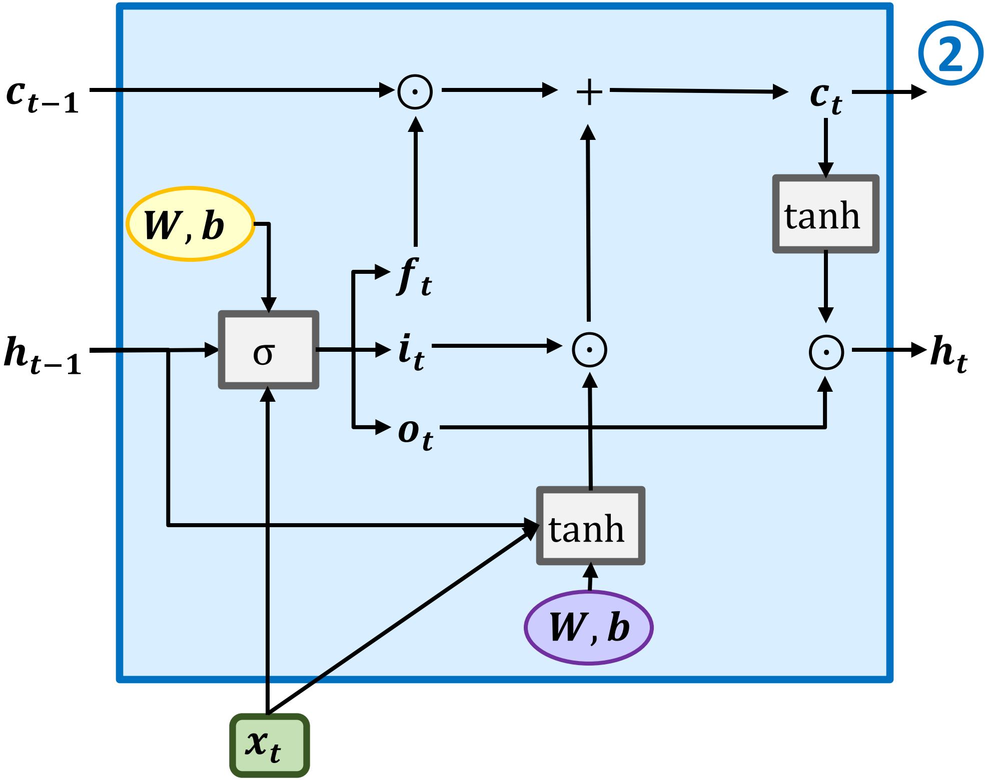
- f forget gate: whether to erase cell
- i input gate: whether to write to cell
- o output gate: how much to reveal cell

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f)$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i)$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o)$$

Long-Short Term Memory Network (LSTM)

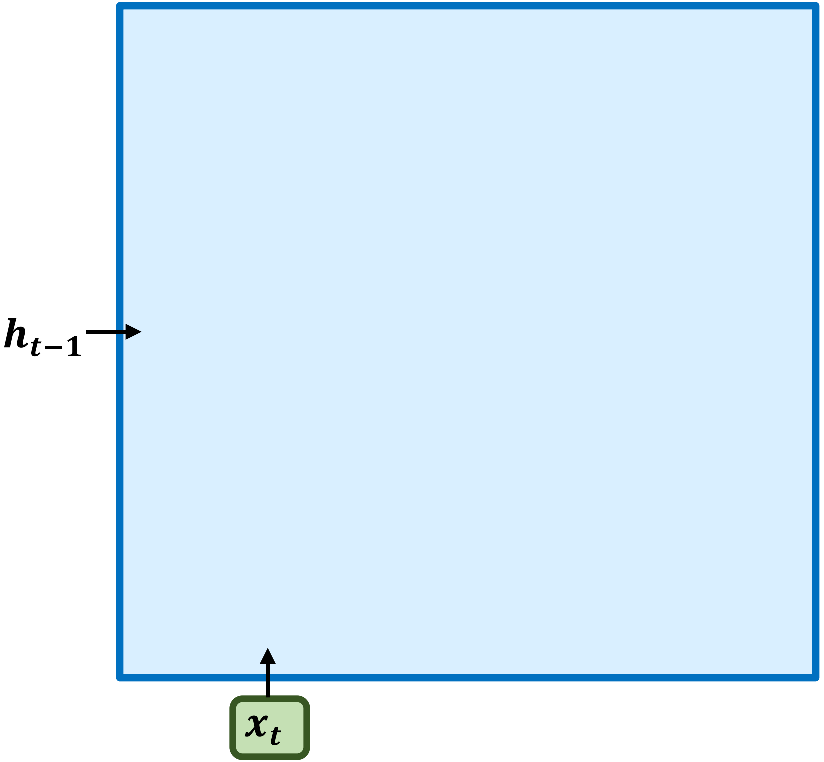


② Updating the states:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx} x_t + W_{ch} h_{t-1} + b_c)$$

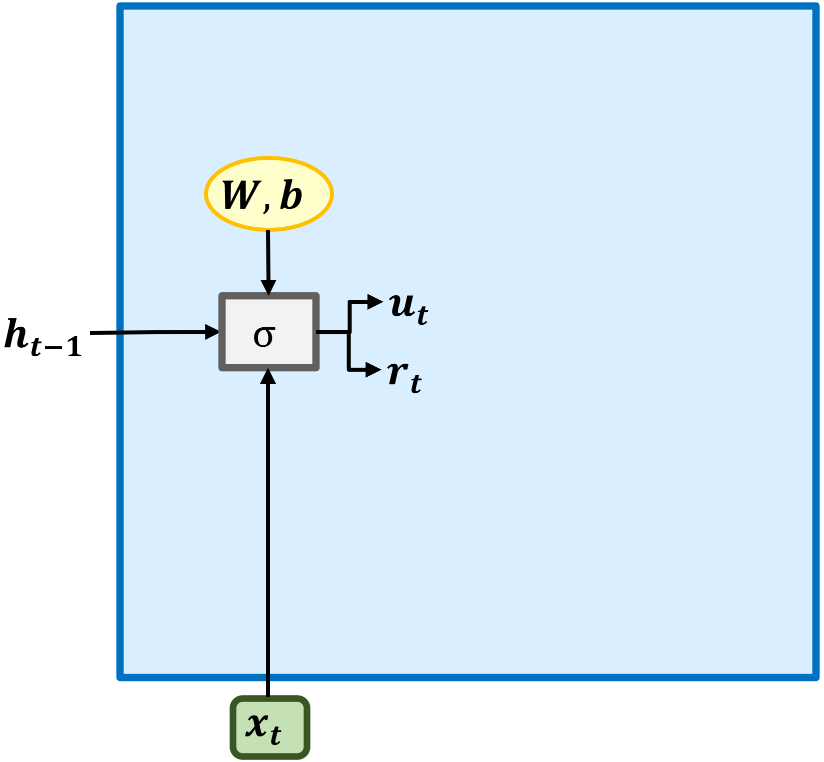
$$h_t = o_t \odot \tanh(C_t)$$

Gated Recurrent Units (GRU)



- Only one state (got rid of cell):
  - Hidden state h_{t-1}

Gated Recurrent Units (GRU)



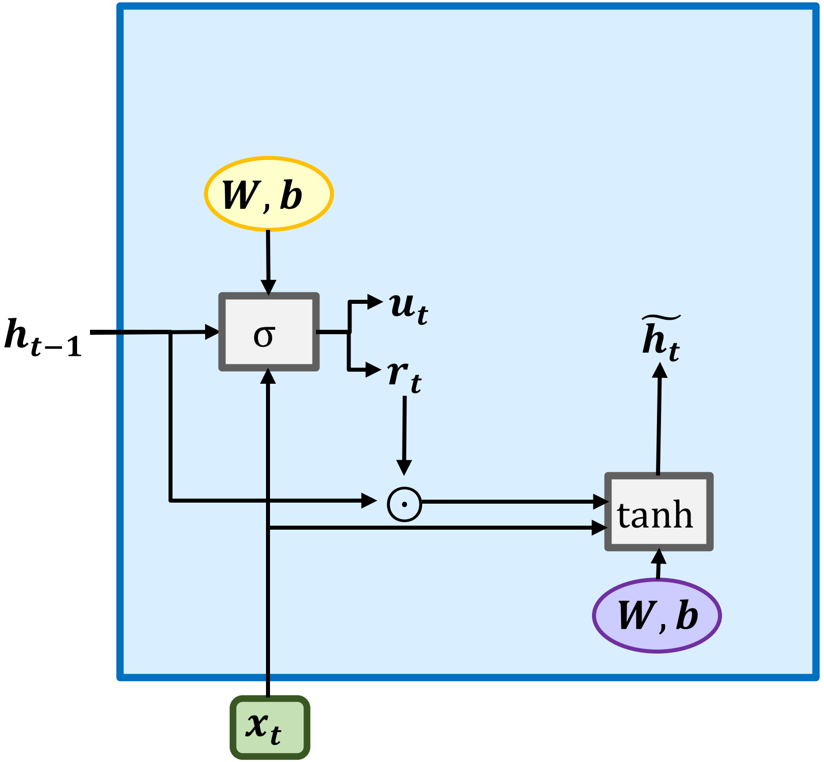
① Updating the gates:

- $\boldsymbol{r}$ reset gate: how much of the previous state to remember
- $\boldsymbol{u}$ update gate: how much of the new state is just a copy of the old state

$$\boldsymbol{r}_t = \sigma(W_{rx}\boldsymbol{x}_t + W_{th}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r)$$

$$\boldsymbol{u}_t = \sigma(W_{ux}\boldsymbol{x}_t + W_{uh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_u)$$

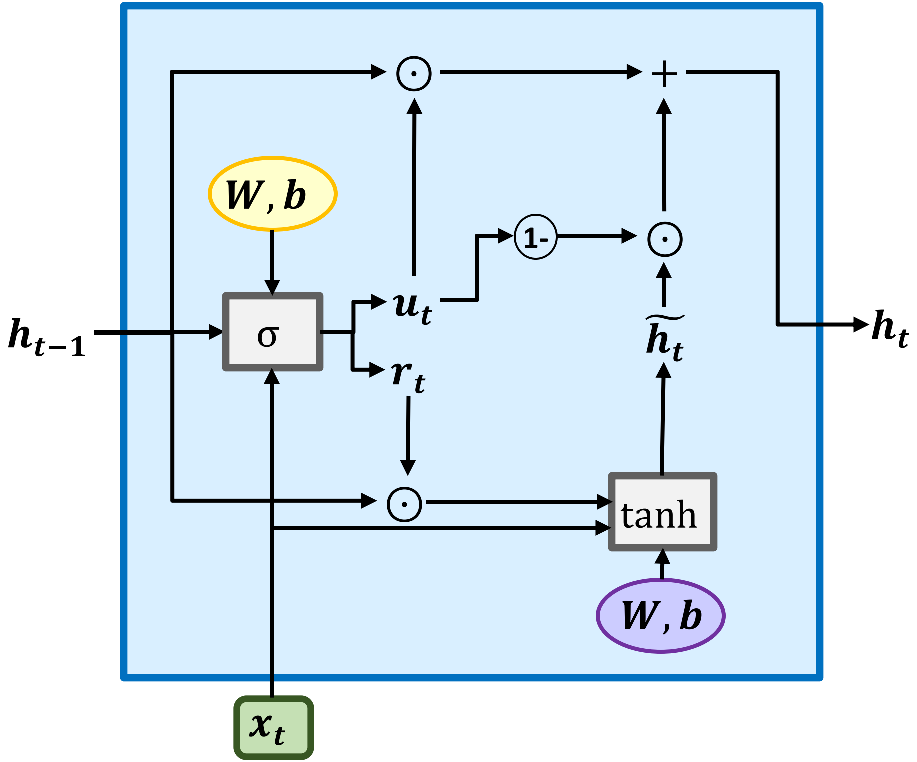
Gated Recurrent Units (GRU)



② Get candidate hidden state:

$$\widetilde{\boldsymbol{h}_t} = \tanh(W_{hx}\boldsymbol{x}_t + W_{ht}(r_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h)$$

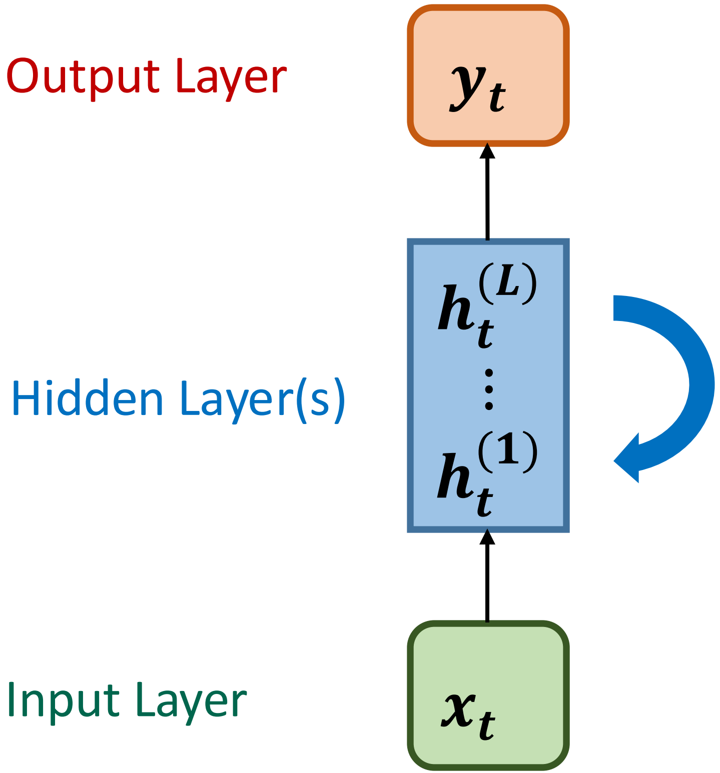
Gated Recurrent Units (GRU)



③ Updating the state:

$$h_t = u_t \odot h_{t-1} + (1 - u_t) \odot \widetilde{h_t}$$

Same input/output – different architectures



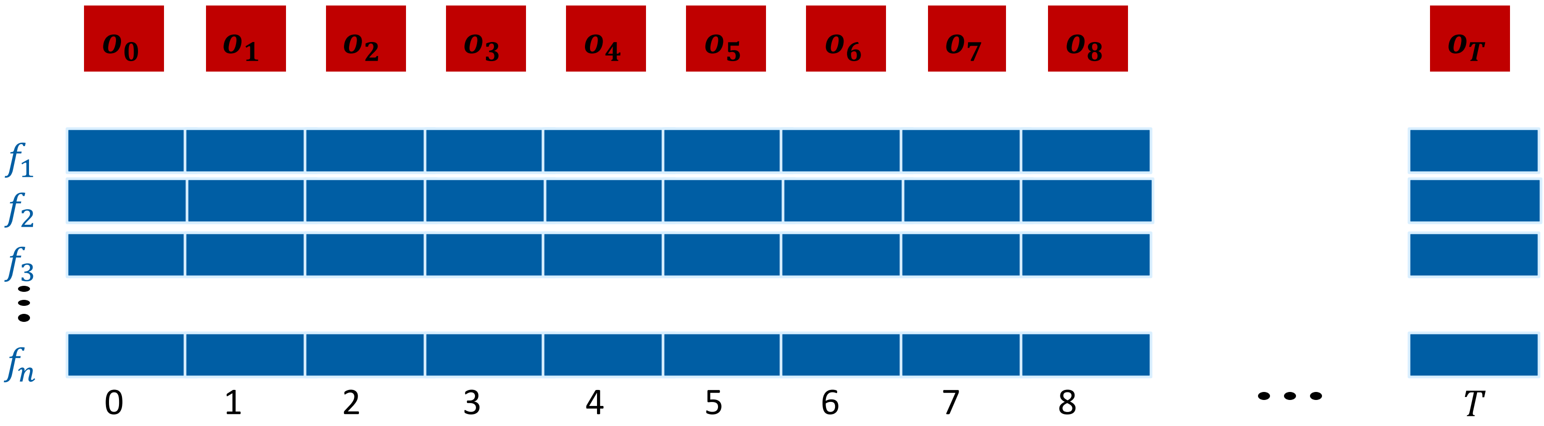
- Vanilla RNN
- LSTM
- GRU

Today – Recurrent Neural Networks

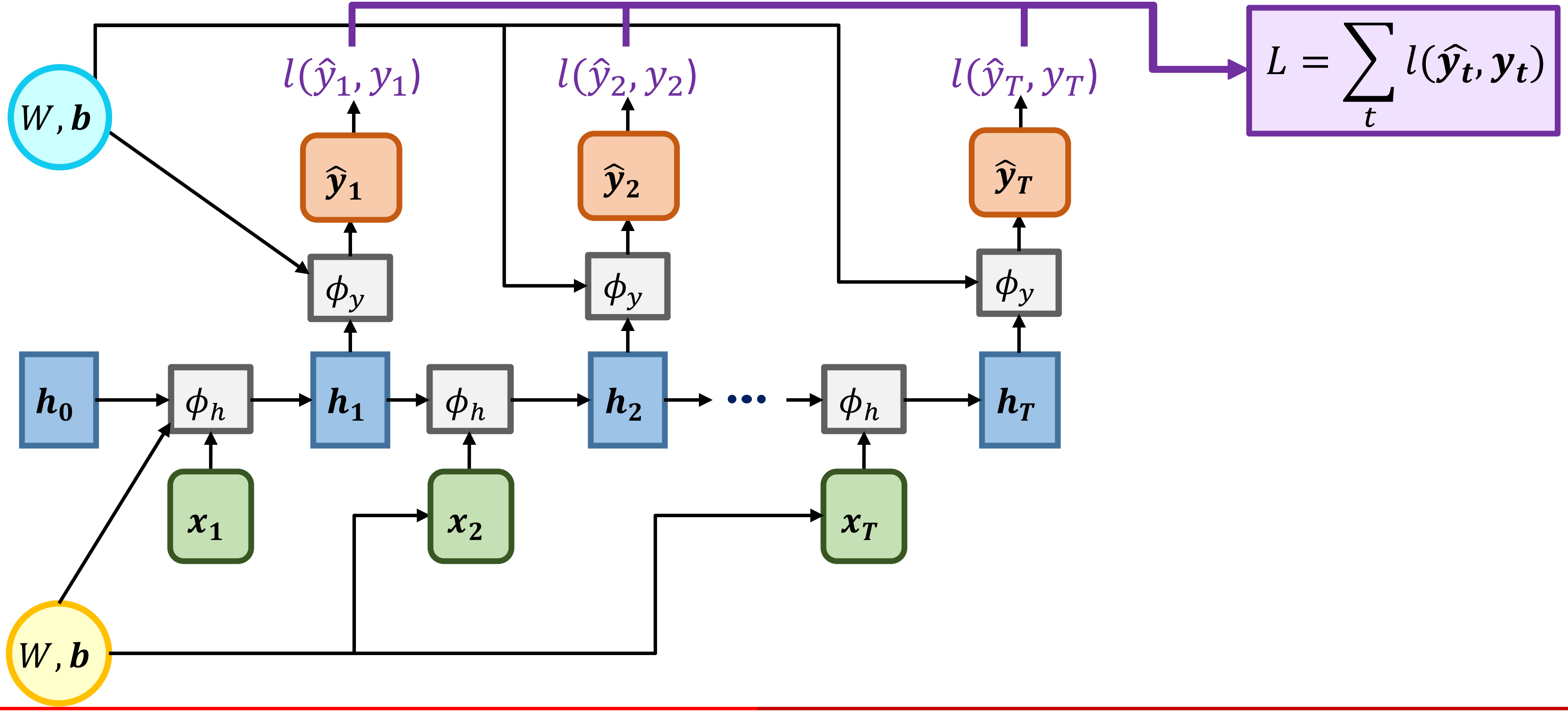
- Deep Knowledge Tracing
- Parameters and hyperparameter tuning
- Different architectures
- **Different tasks**:
  - **"Many-to-many" versus "Many-to-one"**
  - Classification versus Regression

Many-to-many aka the Tracing Task

- Prediction of a target variable o_t at each time step t

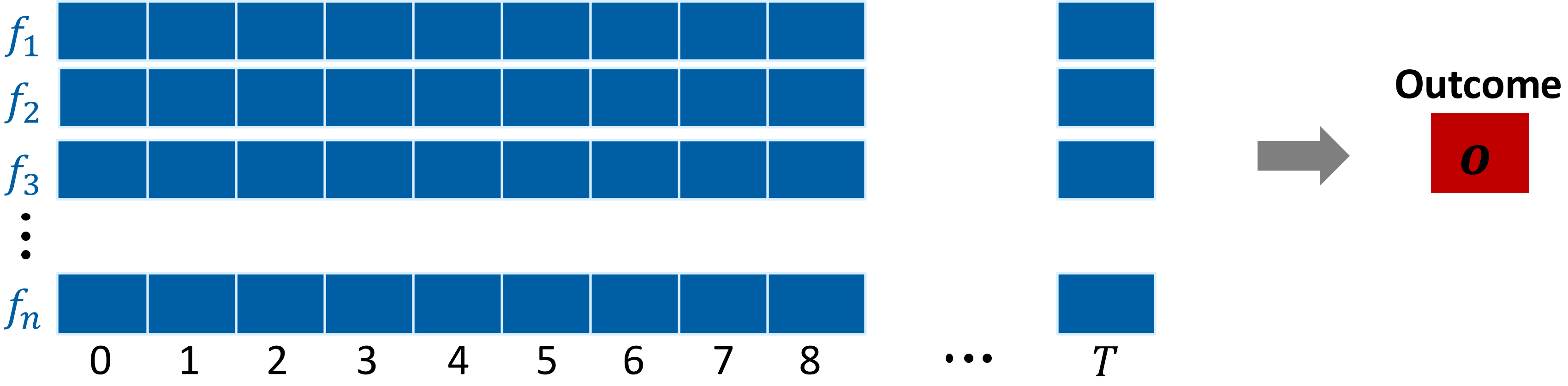


Computational Graph – Many-to-many

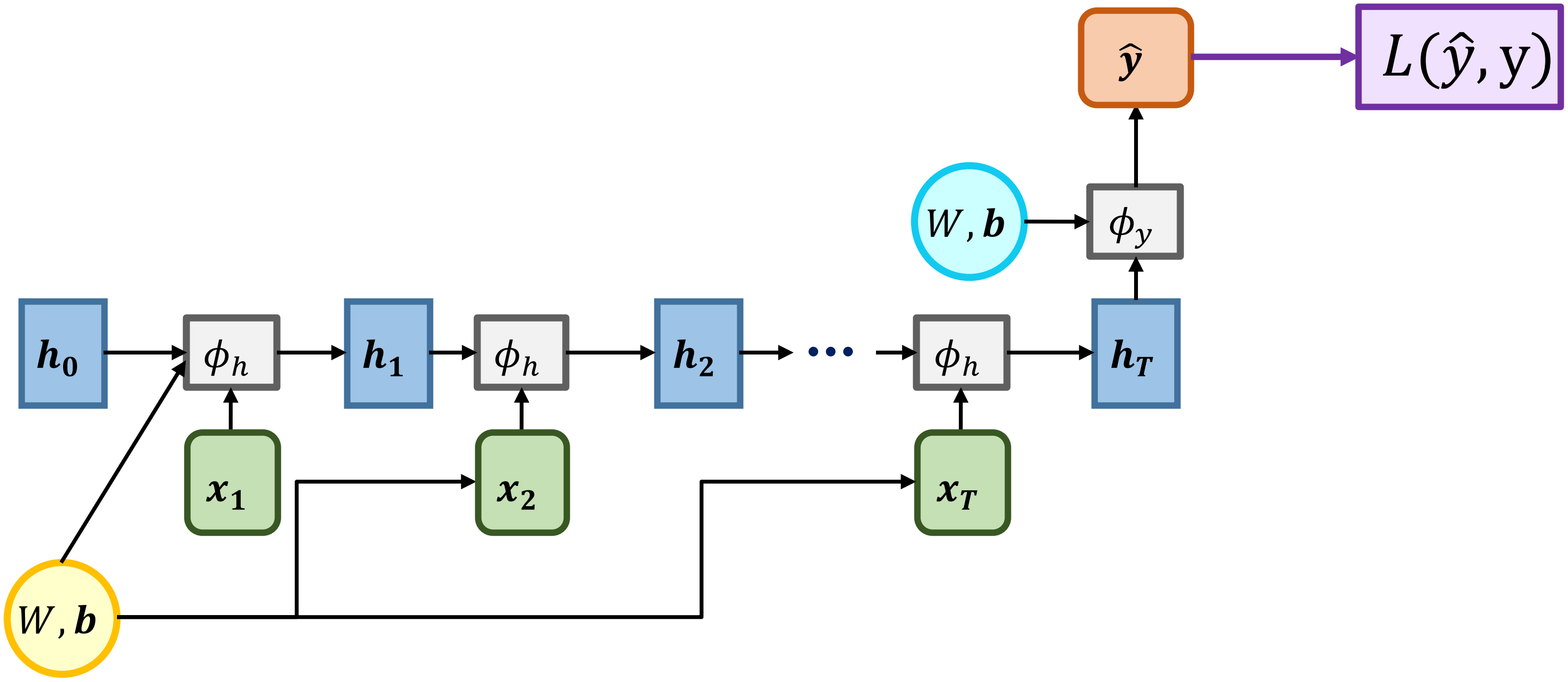


Many-to-one aka the Time-Series Prediction Task

- Prediction of a target variable $\boldsymbol{o}$ after $t \leq T$ time steps, where T is the total number of time steps



Computational Graph – Many-to-one



Today – Recurrent Neural Networks

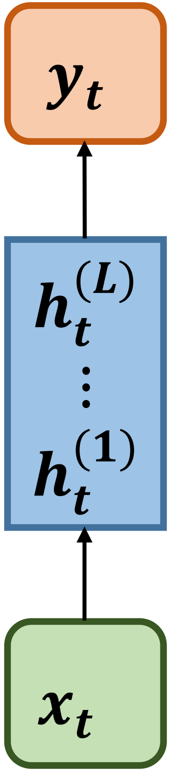
- Deep Knowledge Tracing
- Parameters and hyperparameter tuning
- Different architectures
- **Different tasks**:
  - “Many-to-many” versus “Many-to-one”
  - **Classification versus Regression**

Classification vs. Regression

Output Layer

Hidden Layer(s)

Input Layer



$$\boldsymbol{y}_t = \phi_y(W_{yh}\boldsymbol{h}_t^L + \boldsymbol{b}^{(y)})$$

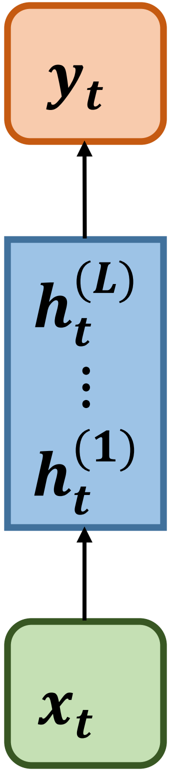
$$\boxed{L(\widehat{\boldsymbol{y}_t}, \boldsymbol{y}_t)}$$

Classification vs. Regression: Output Layer

Output Layer

Hidden Layer(s)

Input Layer



$$\boldsymbol{y_t} = \phi_y(W_{yh}\boldsymbol{h}_t^L + \boldsymbol{b}^{(y)})$$

$$L(\widehat{\boldsymbol{y_t}}, \boldsymbol{y_t})$$

Classification

Regression

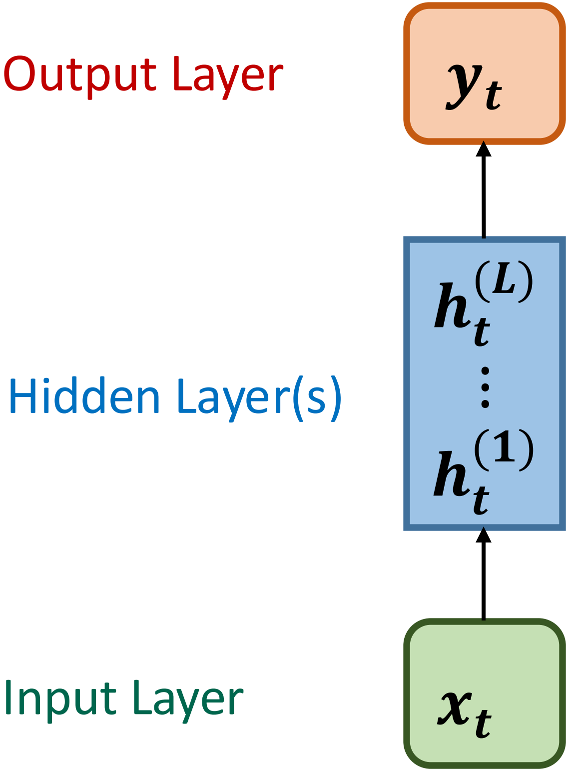
Sigmoid activation:

$$\boldsymbol{y_t} = \sigma(W_{yh}\boldsymbol{h}_t^L + \boldsymbol{b}^{(y)})$$

Linear activation:

$$\boldsymbol{y_t} = W_{yh}\boldsymbol{h}_t^L + \boldsymbol{b}^{(y)}$$

Classification vs. Regression: Training Loss



$$\boldsymbol{y_t} = \phi_y(W_{yh}\boldsymbol{h}_t^L + \boldsymbol{b}^{(y)})$$

$$\boxed{L(\widehat{\boldsymbol{y_t}}, \boldsymbol{y_t})}$$

Classification

- Binary crossentropy
- Categorical crossentropy

Regression

Mean squared error

Your Turn

- Given:
  - Data from a MOOC
  - An LSTM for predicting quiz performance of a student for every week of the course (tracing task)
- Your Task:
  1) Adjust the create_model function in order to predict pass/fail after 5 weeks of the course (time series prediction task) and send us the binary accuracy + AUC
     - Hint 1: return_sequences=False
     - Hint 2: what does TimeDistributed(…) do?
  2) Tune hyperparameters of your choice and send us binary accuracy and AUC

Summary

- Deep Knowledge Tracing
- Parameters and hyperparameter tuning
- Different architectures
- Different tasks:
  - “Many-to-many” versus “Many-to-one”
  - Classification versus Regression