STATS 507
Data Analysis in Python

Week13-1: Deep Sequential Modeling (RNN, LSTM)

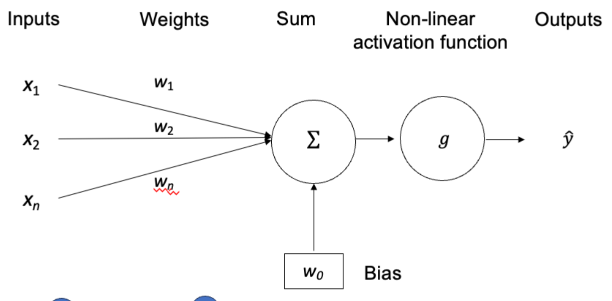
Dr. Xian Zhang

Adapted from slides by Ava Amini

MIT Introduction to Deep Learning

Recap: Core Foundation Review

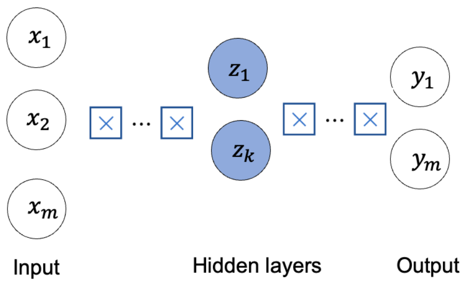
The perceptron



$$\hat{y} = g(w_0 + \sum_{i=1}^{n} x_i w_i) = g(z)$$

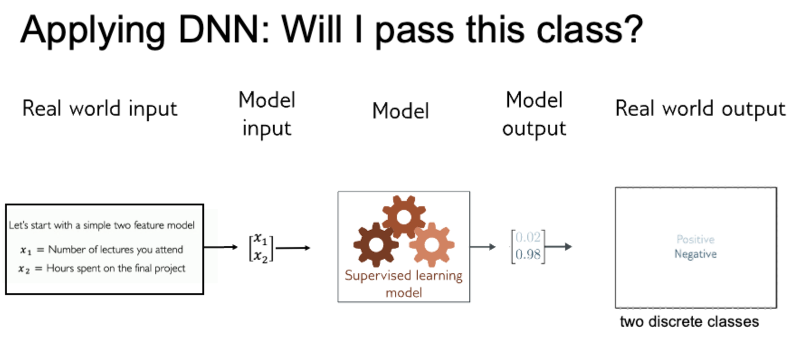
- Structural building blocks
- Numerical operator
- Nonlinear activation functions

Neural Networks



- Stacking Perceptions to form neural networks (MLP)
- Optimization through backpropagation

Training in Practice



- Adaptive learning rate
- Batching
- Regularization

Recap: Gradient descent

Full Batch

- Batch size = N
- 1 update per epoch
- Use all the data

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial L_i(\boldsymbol{W})}{\partial \boldsymbol{W}}$$

- Very slow updates
- Most stable
- Can stuck in local minima

Mini-batch GD

- Batch size = bs
- N/32 updates per epoch
- Use sub dataset

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{1}{B}\sum_{i=1}^{B} \frac{\partial L_i(\boldsymbol{W})}{\partial \boldsymbol{W}}$$

- More accurate estimation of gradient
- Smoother convergence
- Allow for larger learning rates

Stochastic gradient descent

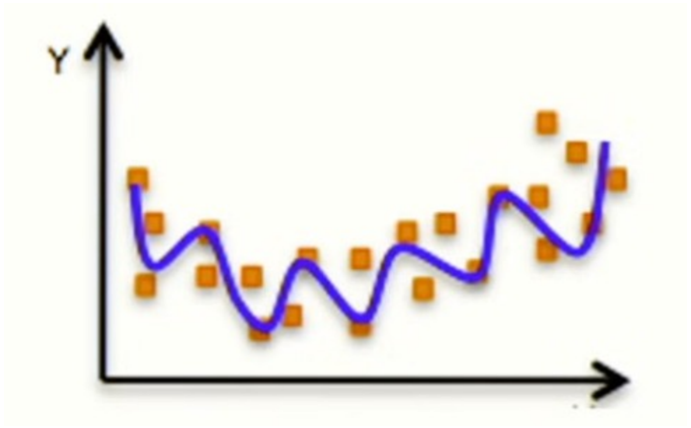
- Batch size = 1
- N updates per epoch
- Use single sample

$$\frac{\partial L_i}{\partial \boldsymbol{W}}$$

- Most efficient
- Very noisy updates
- Unstable

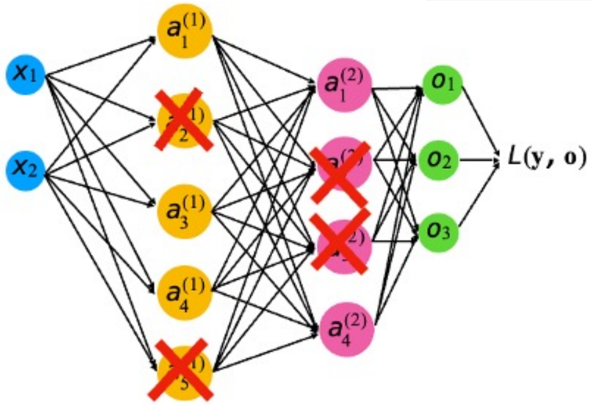
Recap: Training in practice

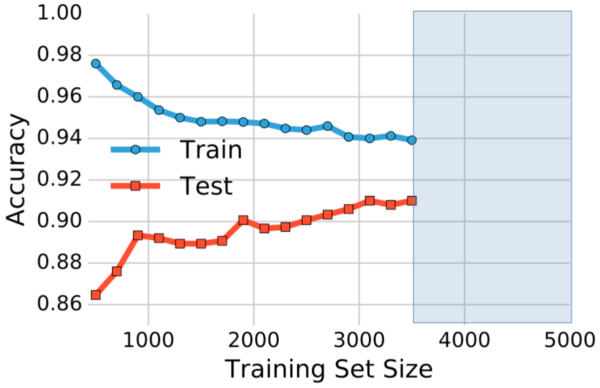
Model is doing really well on training data, but very badly on test data



Overfitting

Use regulation to discourage complex models





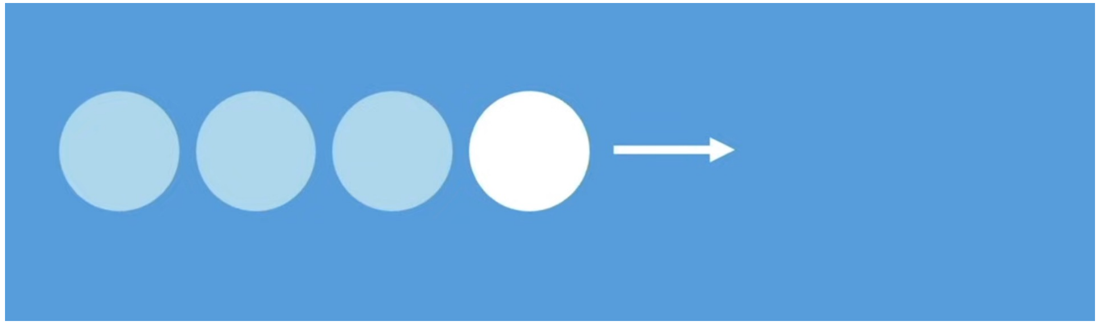
Slide Credit: Alexander Amini

Modified from MIT open course: 6.S191

1. What is sequence modeling
2. RNN
3. LSTM

Intro: Sequence Data?

Given an image of a ball, can you predict where it will go next?



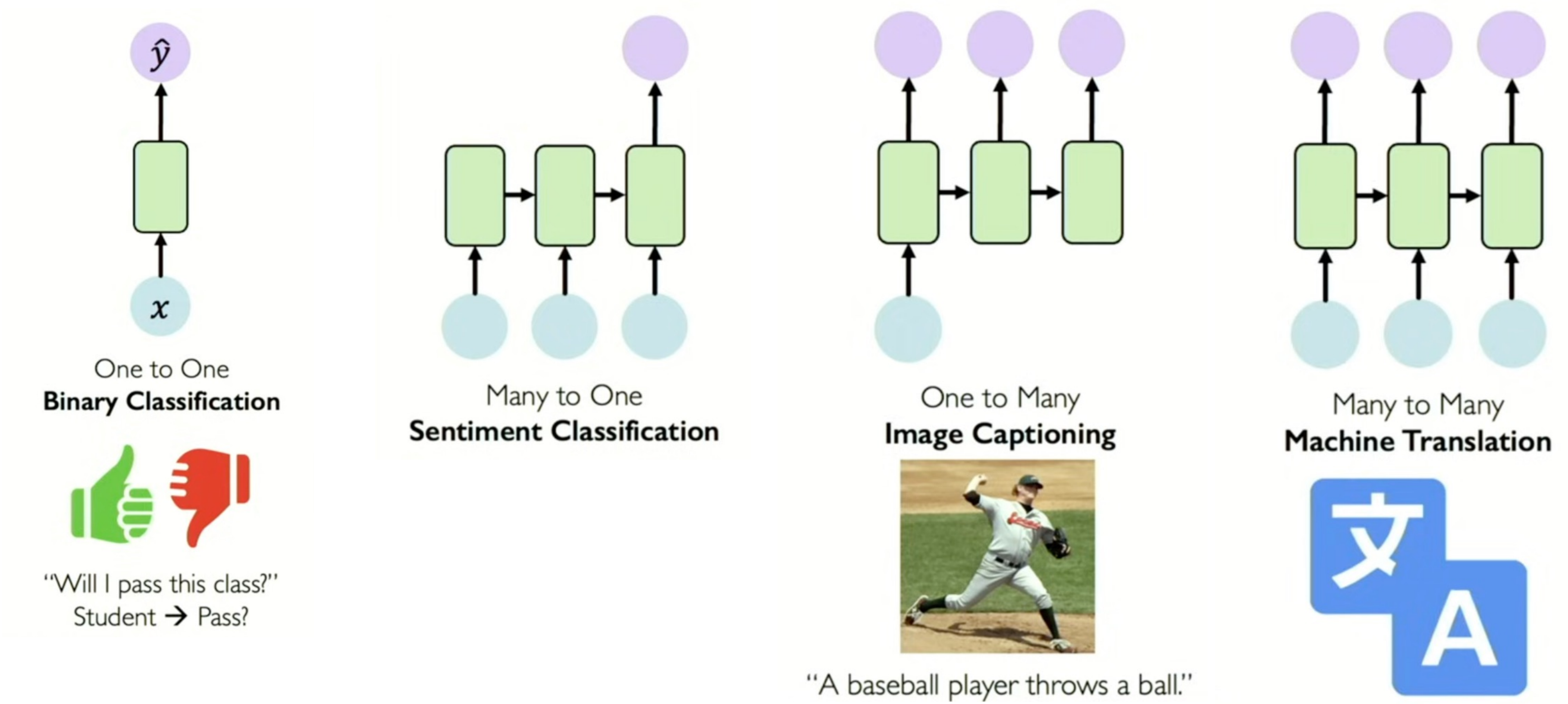
Audio: 

Stock Market: 

Language: This is Stats 507, where we learn data analytics using Python…

Slide Credit: Ava Amini

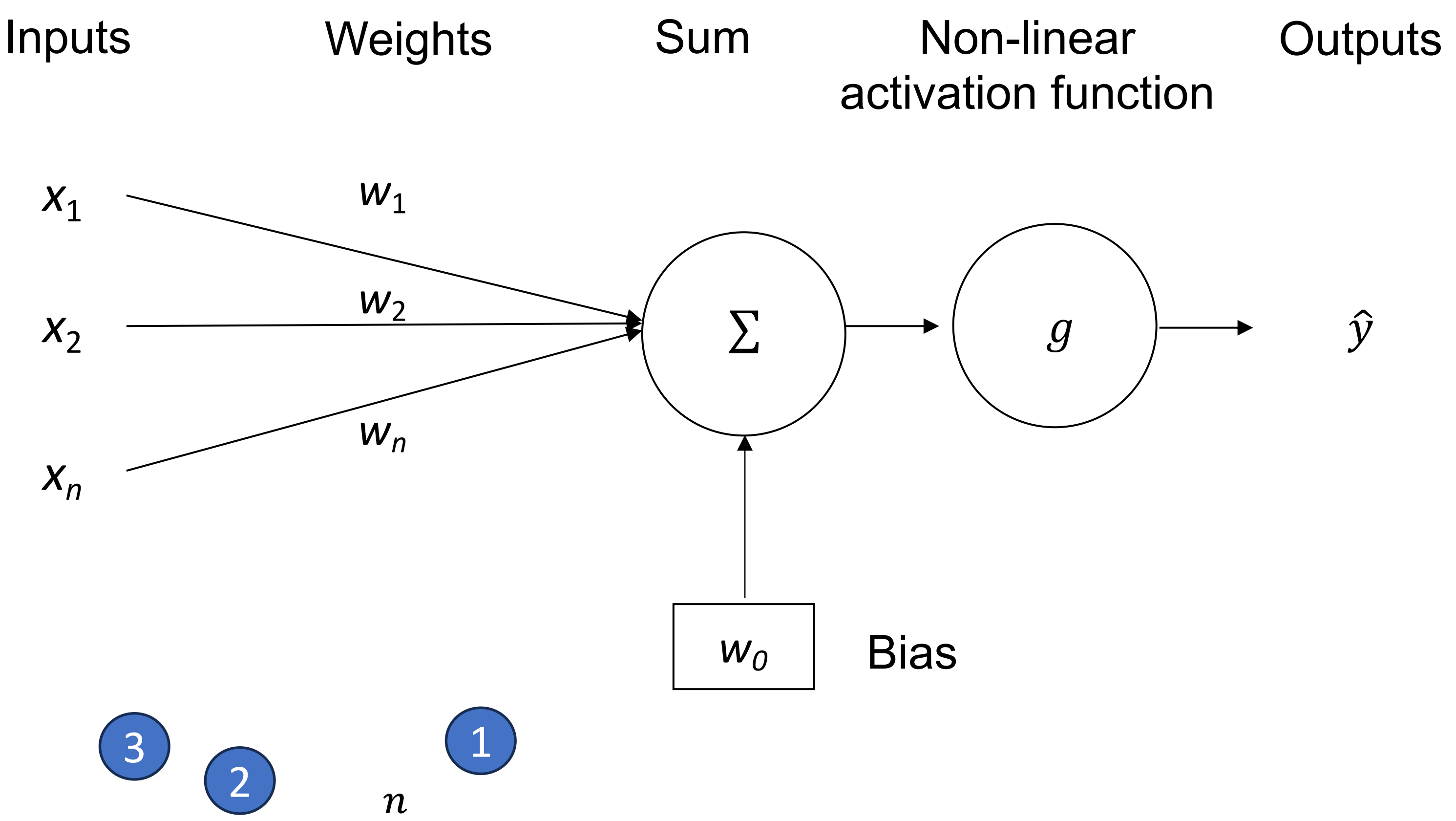
Intro: Sequence modeling applications



Slide Credit: Ava Amini

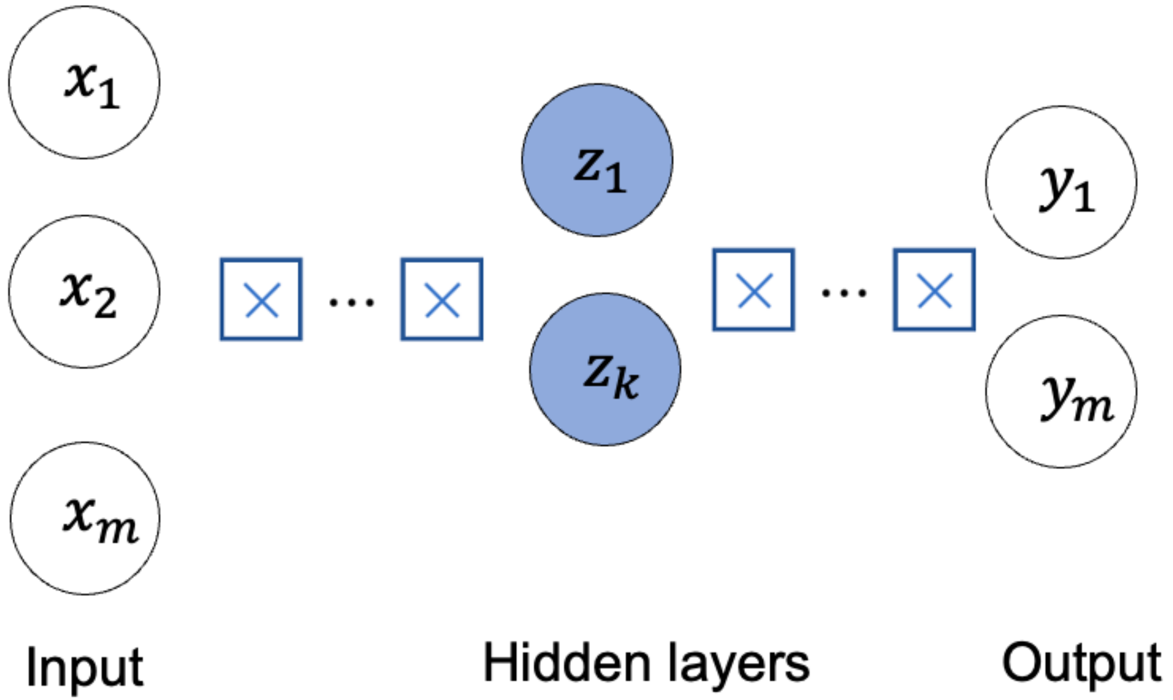
How to build models for sequence?

The perceptron (revisited)

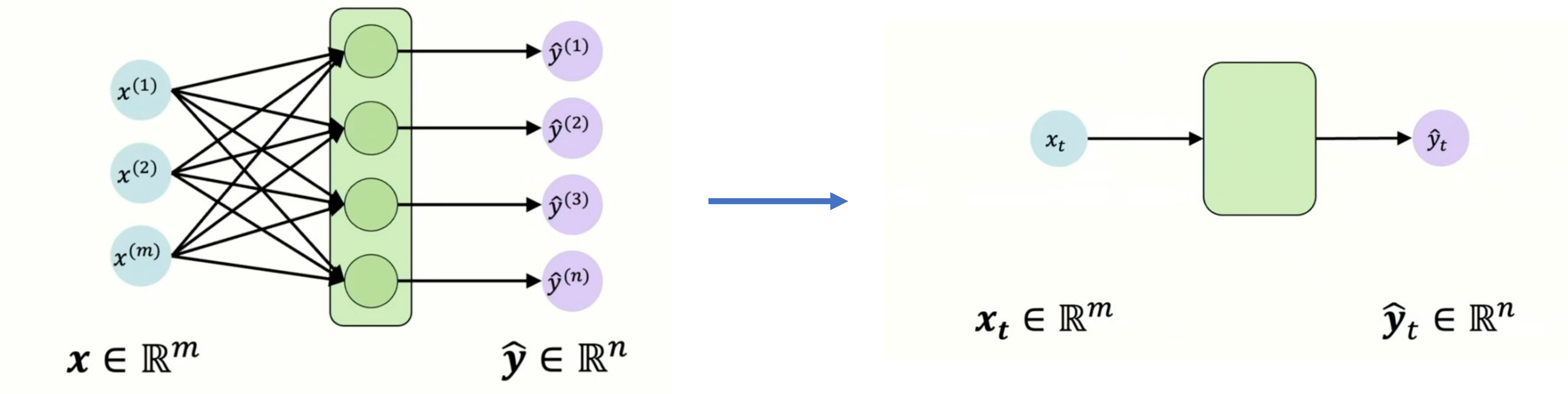


$$\hat{y} = g(w_0 + \sum_{i=1}^{n} x_i w_i) = g(z)$$

Feed-forward Networks (revisited)



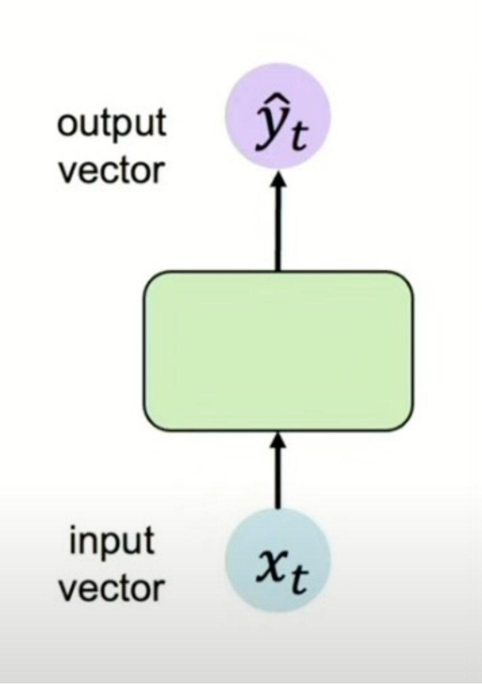
Adding the notion of time

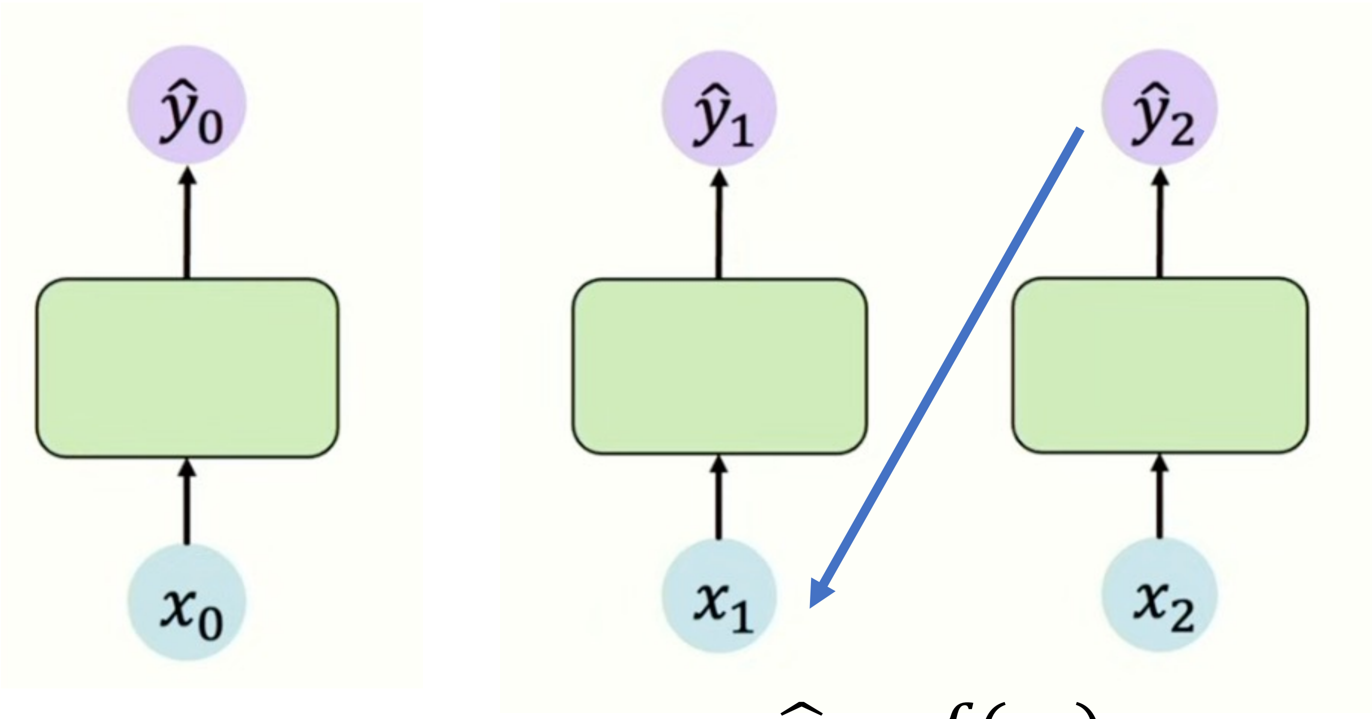


Slide Credit: Ava Amini

Handling individual time steps

Apply the same model stepwise to **time** slice





$$\hat{y}_t = f(x_t)$$

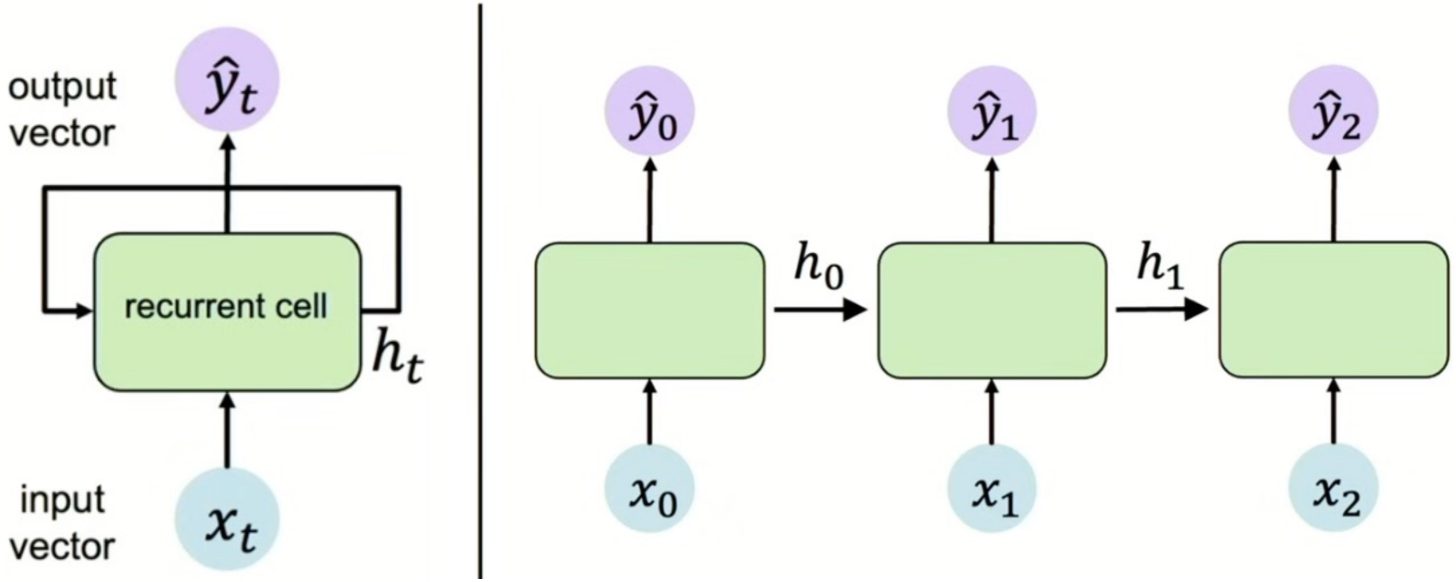
What could the issue?

There is no link between time steps.

Slide Credit: Ava Amini

Neurons with Recurrence

We need to build a neural network that can explicitly model the time step to time step relation: neuron with **recurrence**.



$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

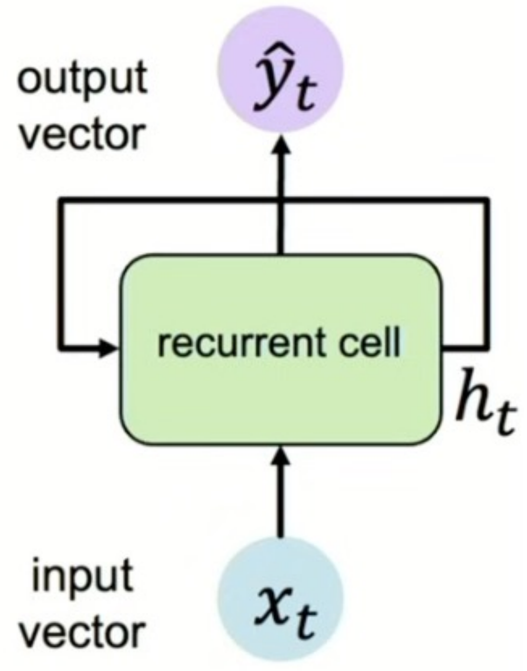
Slide Credit: Ava Amini

1. What is sequence modeling

2. RNN

3. LSTM

Recurrent Neural Networks (RNNs)



Add a cell state: where we apply a recurrence relation at very time step to process sequence data.

$$h_t = f_W(\underline{x_t}, \underline{h_{t-1}})$$

Hidden State

input

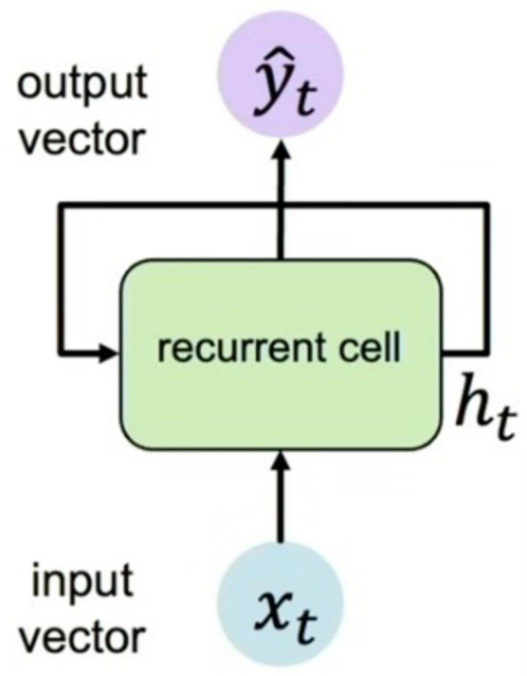
Past memory

Function weights

Note: in RNN, we use the SAME function and set of parameters at every time step.

Slide Credit: Ava Amini

State update and output update



Update hidden state (cell state)

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

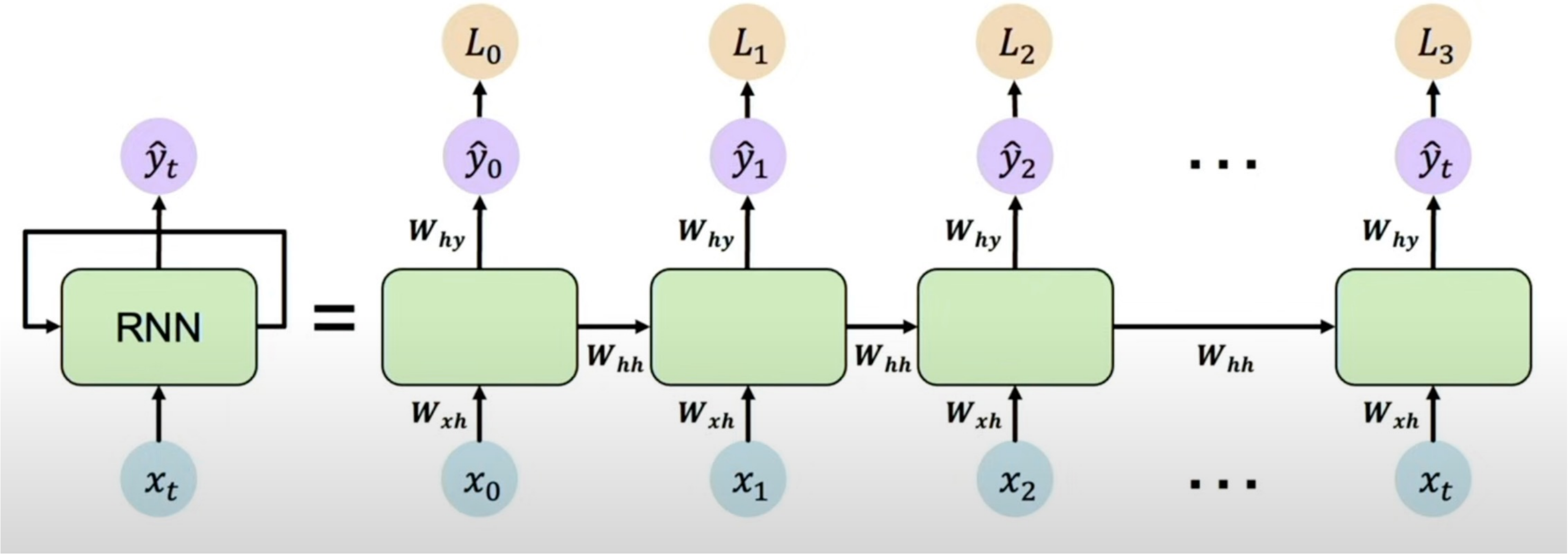
Hidden State Activation function Previous hidden state Input vector

$$\hat{y}_t = W_{hy}^T h_t$$

Note: in RNN, we use the SAME function and set of parameters at every time step.

Slide Credit: Ava Amini

RNNs: computational Graph



Forward Pass

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

$$\hat{y}_t = W_{hy}^T h_t$$

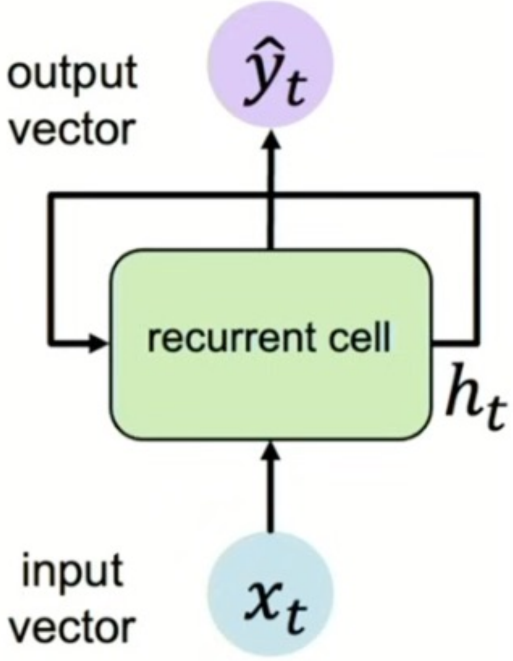
Backpropagation to update the weights

Slide Credit: Ava Amini

In-class practice

RNNs from Scratch

RNNs from Scratch

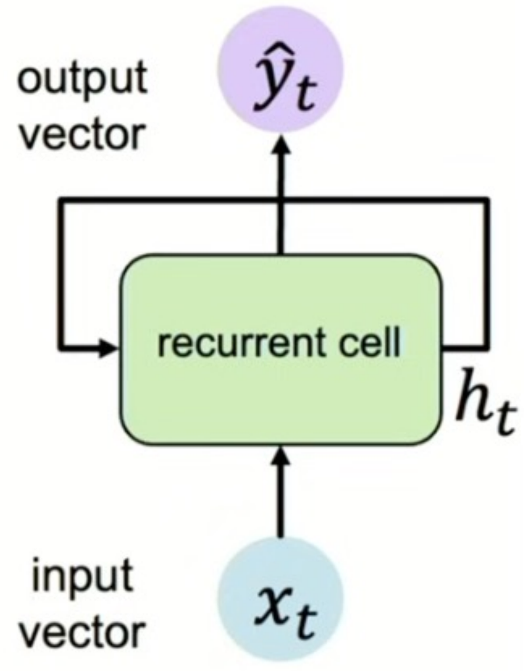



```python
class SimpleRNNCell(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super().__init__()

        # Define the weights as Parameters so they're tracked by PyTorch
        self.Wxh = nn.Parameter(torch.randn(hidden_size, input_size) * 0.01)
        self.Whh = nn.Parameter(torch.randn(hidden_size, hidden_size) * 0.01)
        self.Why = nn.Parameter(torch.randn(output_size, hidden_size) * 0.01)
        self.hidden_size = hidden_size
        self.h = None

    def forward(self, x):
        if self.h is None:
            self.h = torch.zeros(x.size(0), self.hidden_size, device=x.device)

        self.h = torch.tanh(
            x @ self.Wxh.t() +      # Changed F.linear to matrix multiplication
            self.h @ self.Whh.t()   # Changed F.linear to matrix multiplication
        )
        output = F.linear(self.h, self.Why)
        return output
```

Simple one-line RNN using predefined module

```python
self.rnn = nn.RNN(input_size=input_size, hidden_size=hidden_size, num_layers=1)
self.fc = nn.Linear(hidden_size, output_size)
```

RNNs: design criteria



- Handle variable-length sequences
- Track long-term dependencies
- Maintain information about order
- Share parameters across the sequence

RNN meets those criteria.

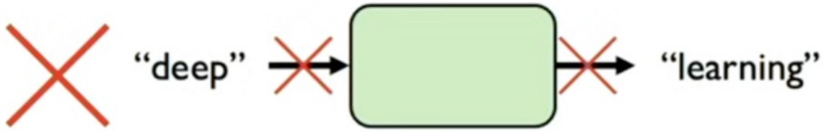
Slide Credit: Ava Amini

Example: predict the next word

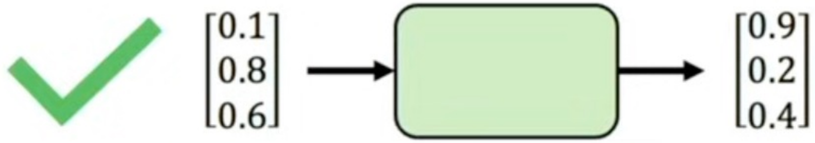
This morning, I took my cat for a __

walk

Representing the word (encoding language)



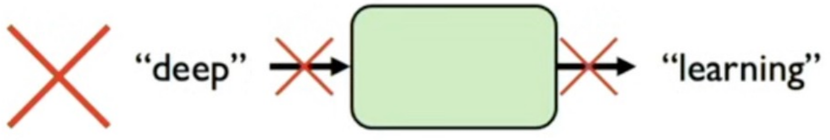
Neural networks cannot interpret words



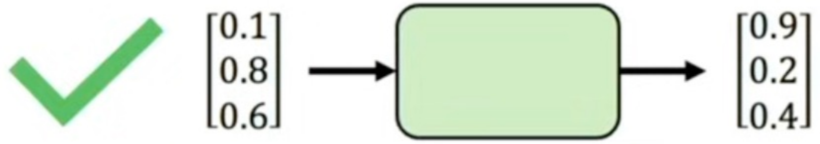
Neural networks require numerical inputs

Slide Credit: Ava Amini

Encoding language for NN

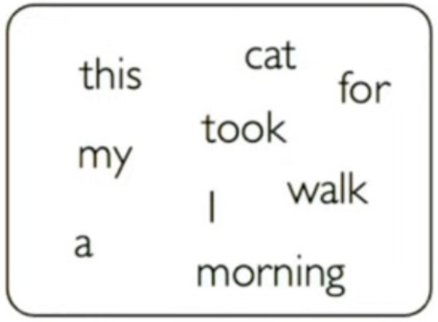


Neural networks cannot interpret words

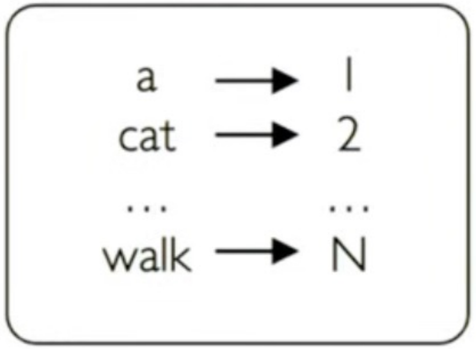


Neural networks require numerical inputs

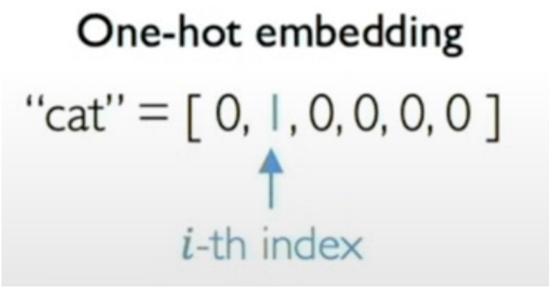
Embedding: transform **indexes** into a vector of fixed size

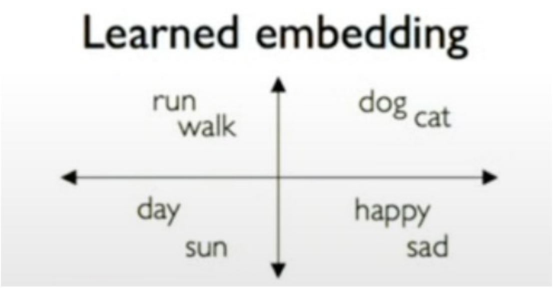


1. Vocabulary:
Corpus of words



2. Indexing:
Word to index





Slide Credit: Ava Amini

Capture differences in **order**

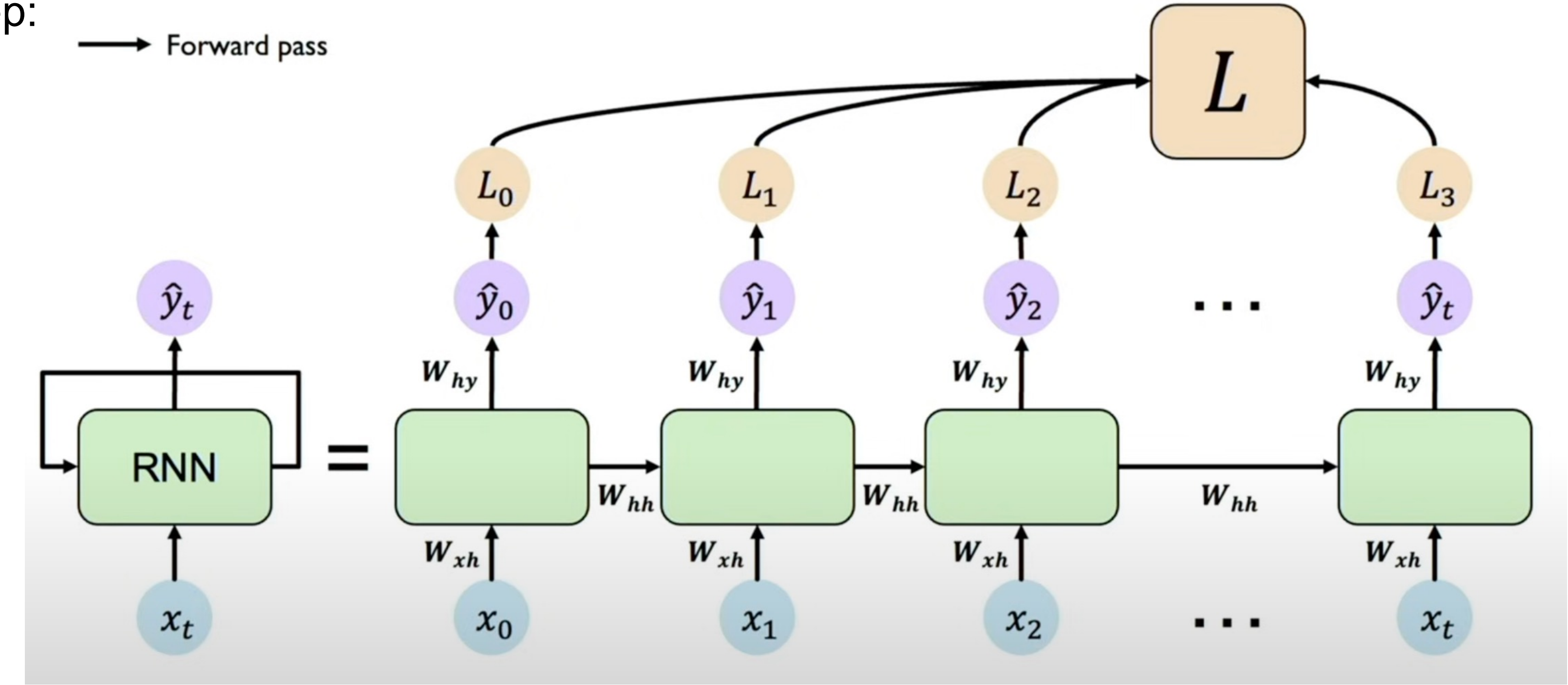
The food was good, not bad at all.

The food was bad, not good at all.

Slide Credit: Ava Amini

Define the loss

In the case of a RNN, the loss function of all time steps is defined based on the loss at every time step:

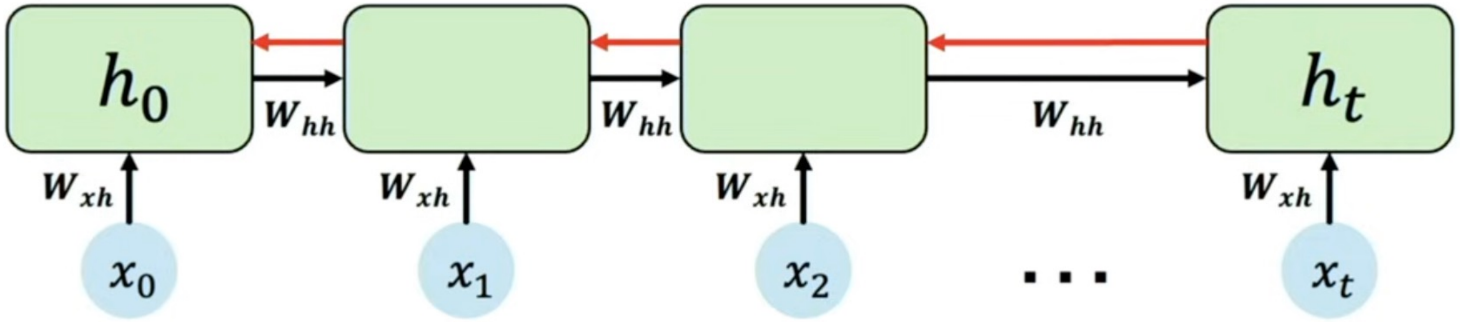


$$J(\boldsymbol{W}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^i; \boldsymbol{W}\right), y^i\right)$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

Slide Credit: Ava Amini

Backpropagation through time



Backpropagation is done at each point in time. At timestep T, the derivate of the loss with respect to weight matric W is expressed as:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}\bigg|_{(t)}$$

Slide Credit: Ava Amini

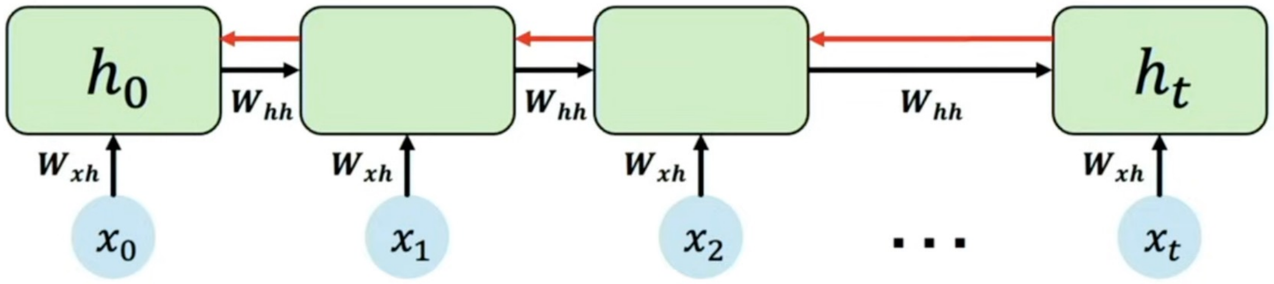
In-class practice

Predict the next word in PyTorch

- Implement the training loop

Problem with gradients

It is difficult to capture long term dependencies because of multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers.



Exploding gradient

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

Vanishing gradient

Many values < 1:
vanishing gradients

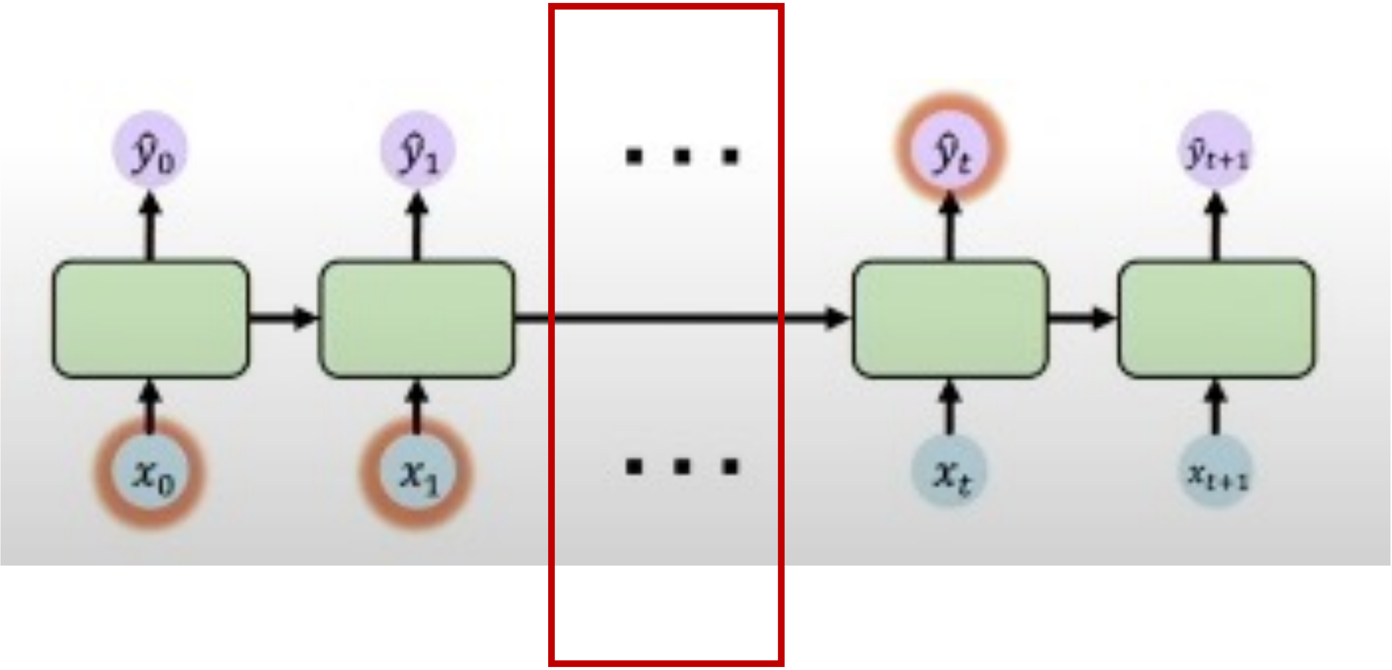
1. Activation function
2. Weight initialization
3. Network architecture

Slide Credit: Ava Amini

The problem of long-term dependencies

Why is vanishing gradient is a problem?

Vanishing gradient is a problem because if we multiply many small numbers together, we would have smaller and smaller gradient, and the bias parameters are only there to capture short-term dependencies.

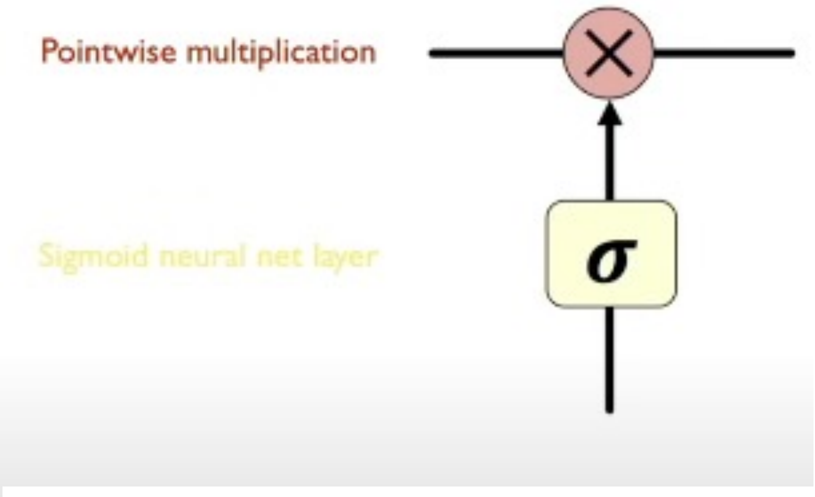


Slide Credit: Ava Amini

The idea of gated cells

To resolve vanishing gradient, one of the ideas is to use **gates** to selectively add or remove information within each recurrent unit with:

Gates can optionally let information through the cell



$$\boxed{\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)}$$

Gated Recurrent Unit (GRU) and Long Short-Term Memory units (LSTM) use gated cells

Slide Credit: Ava Amini

The idea of gated cells

In order to remedy the vanishing gradient problem, specific gates are used and each has a well-defined purpose.

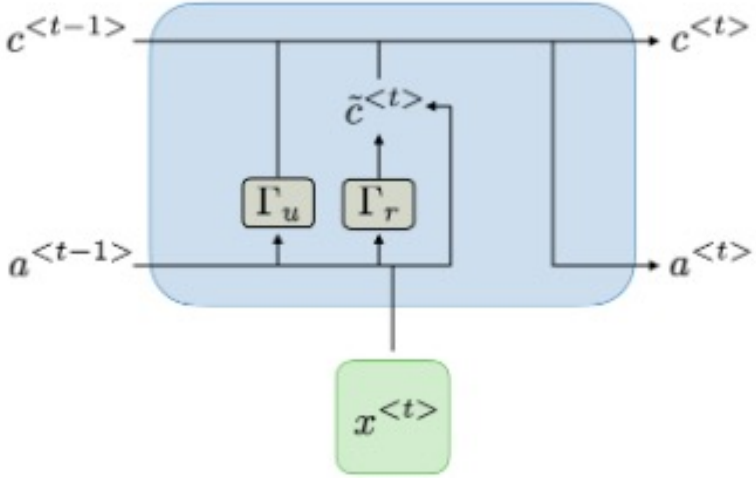
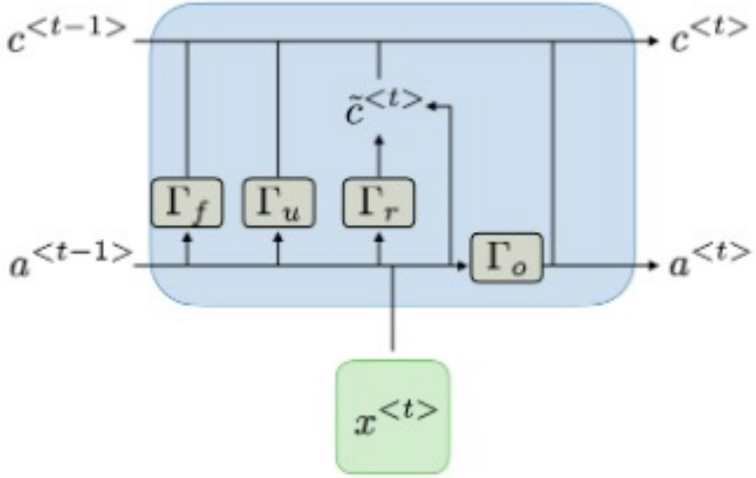
Type of gate	Role	Used in
Update gate Γ_u	How much past should matter now?	GRU, LSTM
Relevance gate Γ_r	Drop previous information?	GRU, LSTM
Forget gate Γ_f	Erase a cell or not?	LSTM
Output gate Γ_o	How much to reveal of a cell?	LSTM

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Slide Credit: Ava Amini

LSTM

Use gates to control the flow of information. (Forget, store, update and output …)

Characterization	**Gated Recurrent Unit** (GRU)	**Long Short-Term Memory** (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Dependencies		

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Slide Credit: Ava Amini

Other things

HW8 out.

Final project guideline out (**start early**)

Coming next:

Deep Sequential model (RNN, LSTM, Transformer…)