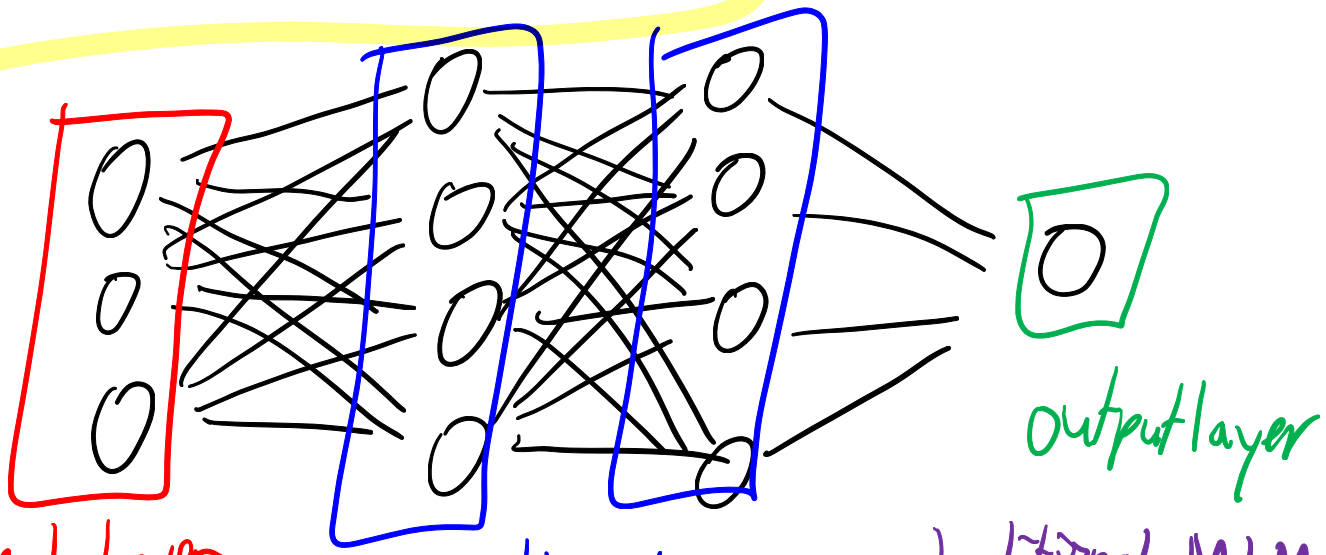
Multi-Layer Perceptron (MLP)

Fully-Connected Neural Network

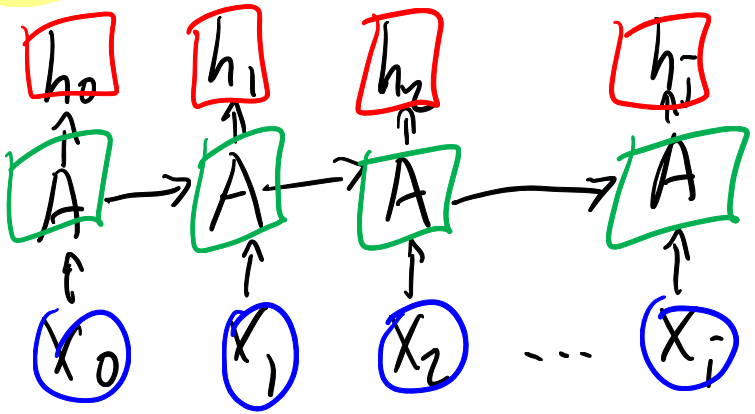


traditional MLPs only have input and output

Convolutional Neural Network

- each neuron is connected to only part of the previous neuron
- often used for image data

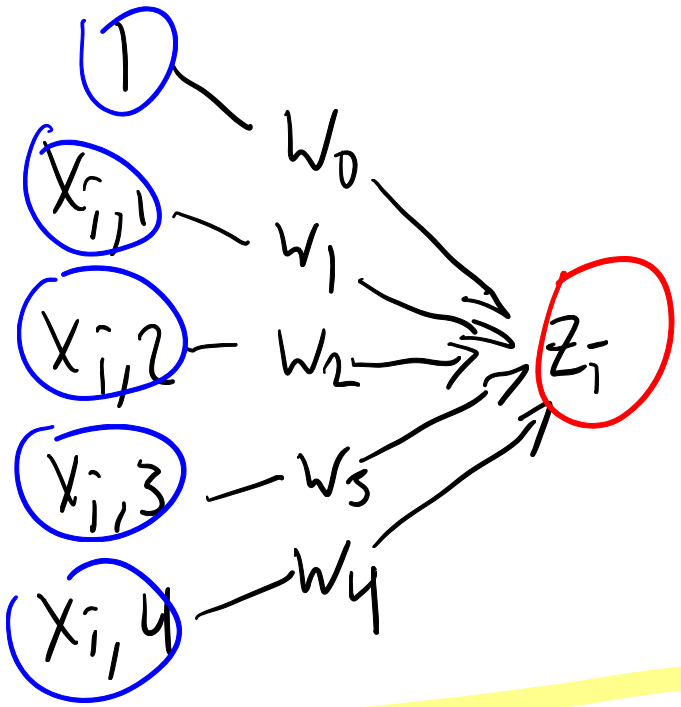
Recurrent Neural Network



- used when there are dependencies in sequential data

Logistic Regression

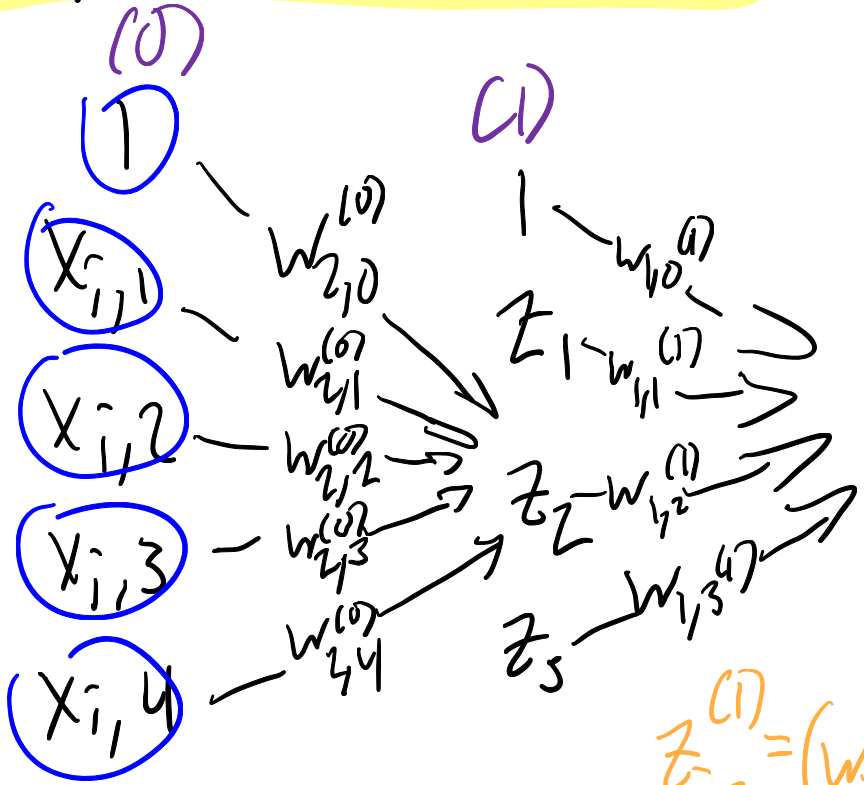
Given n samples: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$



$$z_i = w^T x_i = w_0 + w_1 x_1 + w_2 x_2 \ldots w_n x_n$$

There are no hidden layers: shallow learning

Single → Multiple Layers



$$z_1^{(0)} = W^{(1)} z_i^{(1)}$$

$$z_{i,2}^{(1)} = (w_2^{(0)})^T x_i =$$

$$z_i^{(1)} = W^{(0)} x_i \quad w_{2,0}^{(0)} + w_{2,1}^{(0)} x_{i,1} + w_{2,2}^{(0)} x_{i,2} + w_{2,3}^{(0)} x_{i,3} + w_{2,4}^{(0)} x_{i,4}$$

Layer 1: $z^{(1)} = W^{(0)} x_i$

Layer 2: $z^{(2)} = W^{(1)} x_i$

Layer L: $z^{(L)} = W^{(L-1)} x_i$

$W^{(0)} \in R^{d_1 \times d_0}$

$W^{(1)} \in R^{d_2 \times d_1}$

$W^{(L-1)} \in R^{d_L \times d_{L-1}}$

d_L = # of nodes in L

- Regression Task

$$f(x_i) = z_i^{(L)}$$

- Classification Task

$$f(x_i) = \sigma(z_i^{(L)})$$

σ for binary classification, use softmax for multiclass

$z_i^{(1)} = W^{(0)} x_i$

↓

$z_i^{(2)} = W^{(1)} x_i^{(1)}$

↓

$z_i^{(3)} = W^{(2)} x_i^{(2)}$

↓

...

↓

$z_i^{(L)} = W^{(L-1)} x_i^{(L-1)}$

From Linear to Non-linear Model

- stacking multiple linear models is still a linear model

$$z_i^{(L)} = W^{(L-1)} \cdots W^{(2)} W^{(1)} W^{(0)} x_i$$

deep linear networks are no more expressive than linear regression

Activation Function

- add (non-linear) activation functions to hidden layers
- multilayer fully-connected neural nets with nonlinear activation functions are universal approximators; they can approximate any function arbitrarily well

- sigmoid function

Linear Model

$$z_i^{(1)} = W^{(0)} x_i$$

$$z_i^{(2)} = W^{(1)} x_i^{(1)}$$

$$z_i^{(3)} = W^{(2)} x_i^{(2)}$$

...

$$z_i^{(L)} = W^{(L-1)} x_i^{(L-1)}$$

Non-linear model

$$z_i^{(1)} = W^{(0)} x_i, \quad h_i^{(1)} = \sigma(z_i^{(1)})$$

$$z_i^{(2)} = W^{(1)} h_i^{(1)}, \quad h_i^{(2)} = \sigma(z_i^{(2)})$$

$$z_i^{(3)} = W^{(2)} h_i^{(2)}, \quad h_i^{(3)} = \sigma(z_i^{(3)})$$

...

$$z_i^{(L)} = W^{(L-1)} h_i^{(L-1)}, \quad h_i^{(L)} = \sigma(z_i^{(L)})$$

- tanh function
- ReLu function — if the neuron is negative, it turns to 0; same other wise

4/7

Image Classification

- 60,000 training samples $\{(x_i, y_i)\}$

with Logistic Regression:

- vectorize each 28x28 image to 784-dim vector, $x_i \in R^{784}$
- add constant 1 to each x_i (bias), $x_i \in R^{785}$
- denote model parameter $W \in R^{10 \times 785}$
- $z_i = Wx_i$
- output prediction with softmax

with MLP:

input image $x_i \in R^{784}$

$z_i^{(1)} = W^{(0)} x_i \in R^{256}$

$h_i^{(1)} = relu(z_i^{(1)}) \in R^{256}$ hidden layer 1

$z_i^{(2)} = W^{(1)} h_i^{(1)} \in R^{128}$

$h_i^{(2)} = relu(z_i^{(2)}) \in R^{128}$ hidden layer 2

$z_i^{(3)} = W^{(2)} h_i^{(2)} \in R^{10}$

$\hat{y}_i = Softmax(z_i^{(3)}) \in R^{10}$ output layer

multiclassification → softmax

loss function: $L = -\sum_{i=1}^{60000} \sum_{j=1}^{10} y_{ij} \log(\hat{y}_{ij})$

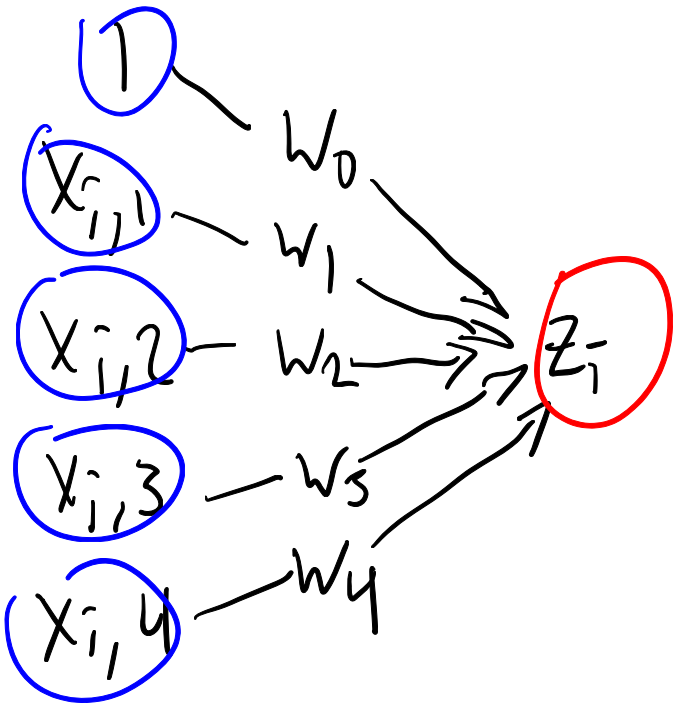
$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$

$W^{(0)} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1d} \\ w_{21} & & & w_{2d} \\ \vdots & & & \vdots \\ w_{d'1} & w_{d'2} & \cdots & w_{d'd} \end{bmatrix}$

Optimization

- Stochastic Gradient Descent to learn model parameters

Single layer:

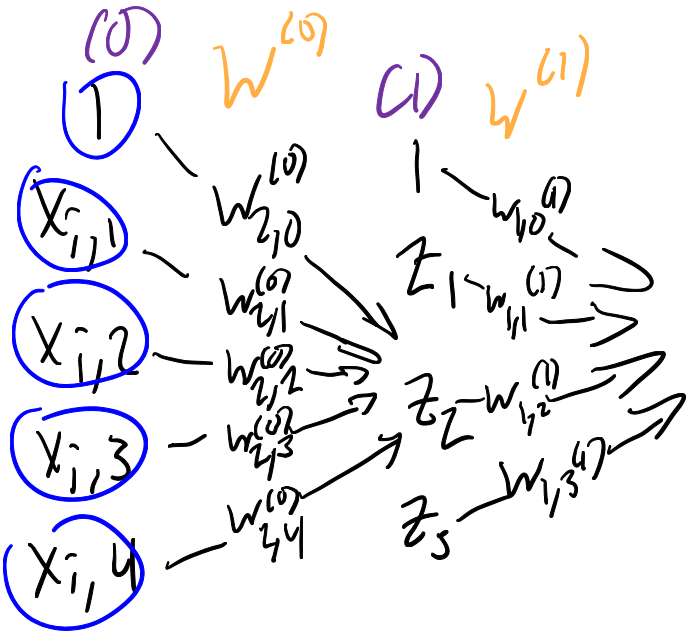


$L = \text{loss}(f(x_i), y_i)$

w

$w_{t+1} = w_t - \eta \left.\frac{\partial L}{\partial w}\right|_{w=w_t}$

multiple layer:



It's easy to calculate $W^{(1)}$.

Calculate $W^{(0)}$ through backpropagation

Calculate features going forward.
Calculate gradients going backward.

chain rule:

$$h(x) = f(g(x))$$

gradient: $\frac{\partial h(x)}{\partial x} = \frac{\partial f(g)}{\partial g} \frac{\partial g(x)}{\partial (x)}$

$f(y) = y^2$

$g(x) = 3x + 5$

$\frac{\partial f(g)}{\partial g} = 2g$

$\frac{\partial g(x)}{\partial x} = 3$

$h(x) = f(g(x))$

$\frac{\partial h(x)}{\partial x} = (2(3x+5)) \times 3$

Back propagation

Loss function is function of $W^{(i)}$ and $z^{(i)}$

1) Compute gradients of last layer

Compute gradients as regular models $\frac{\partial L}{\partial W^{(i)}}$ $\frac{\partial L}{\partial z^{(i)}}$

(2) Compute gradients of hidden layers based on chain rule

$z^{(1)}$ is function of $W^{(0)}$: $z^{(1)} = W^{(0)} x$

$$\frac{\partial L}{\partial W^{(0)}} = \frac{\partial L}{\partial z^{(1)}} \boxed{\frac{\partial z^{(1)}}{\partial W^{(0)}}} = x$$

Forward Pass

$z_i^{(1)} = W^{(0)} x_i \in R^{256}$

$z_i^{(2)} = W^{(1)} x_i \in R^{128}$

$z_i^{(3)} = W^{(2)} \text{relu}(z_i^{(2)}) \in R^{10}$

Backward Pass

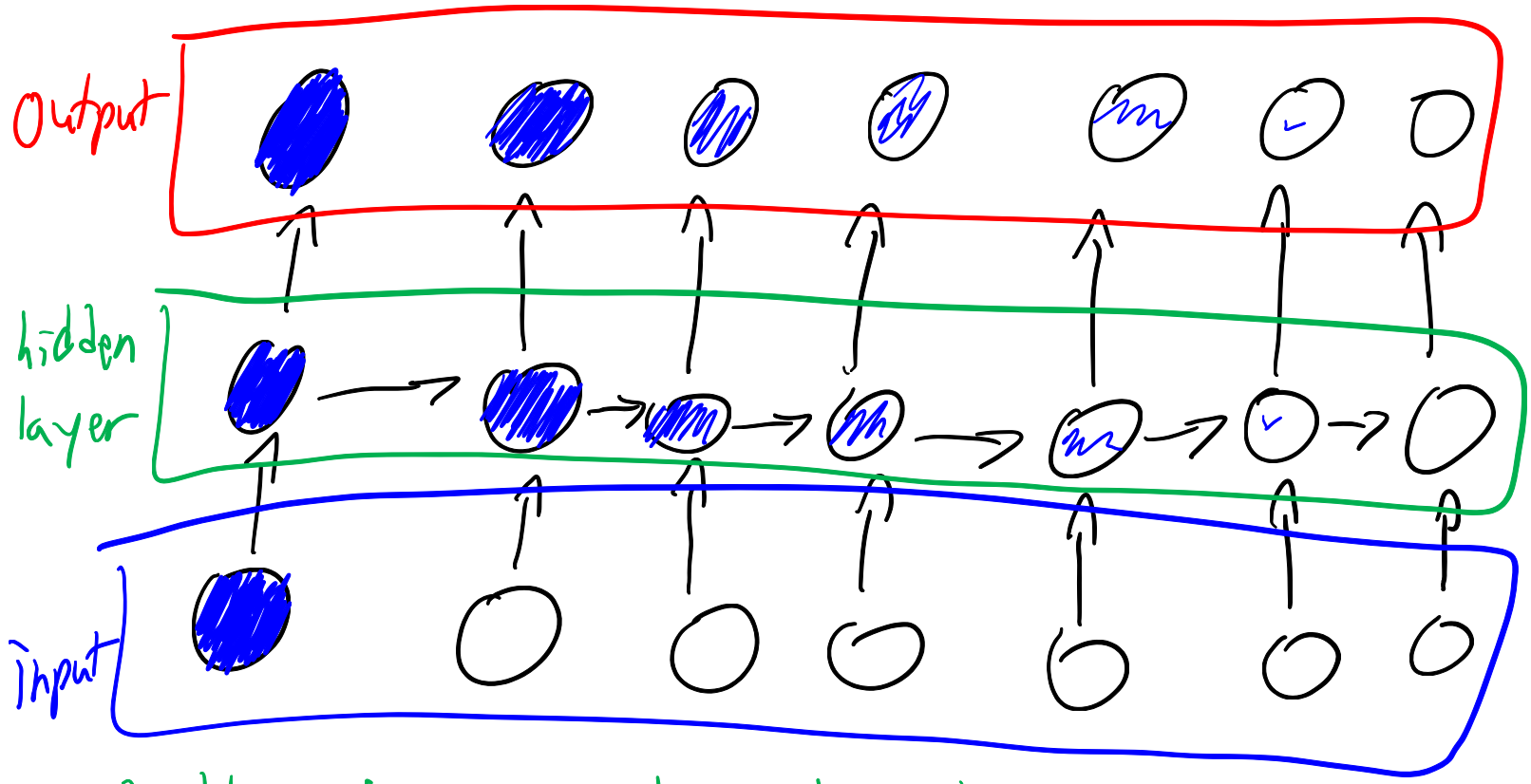
- $\frac{\partial L}{\partial W^{(0)}} = \frac{\partial L}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(0)}}$
- $\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W^{(1)}}$
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial z^{(1)}}$
- $\frac{\partial L}{\partial W^{(2)}}$
- $\frac{\partial L}{\partial z^{(2)}}$

Recurrent Neural Networks

4/14

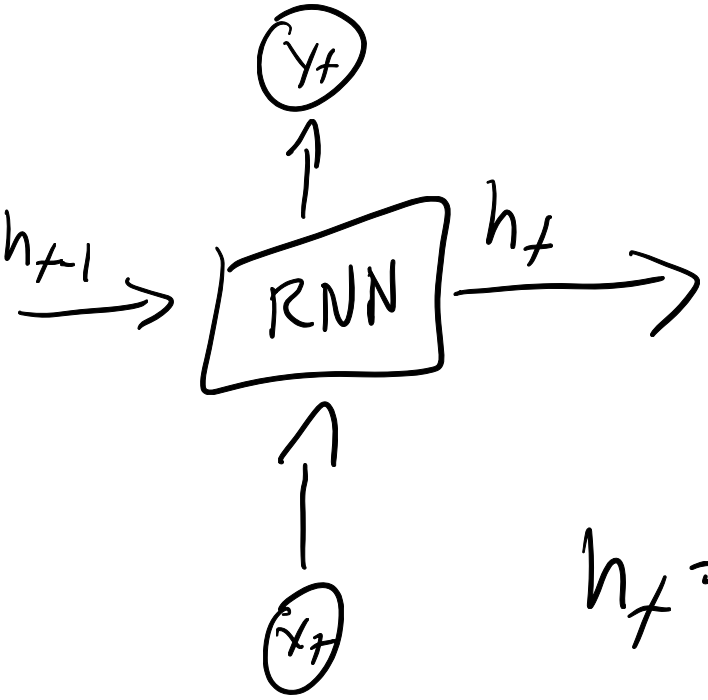
- an "internal state" that is updated as a sequence is processed

The last input includes all of the previous ones.



hidden layer captures dependence across words

- process a sequence of vectors x by applying a recurrence formula at every time step
  - 1) update hidden state



some function with parameters W

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

new state | old state | input vector at some time step

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

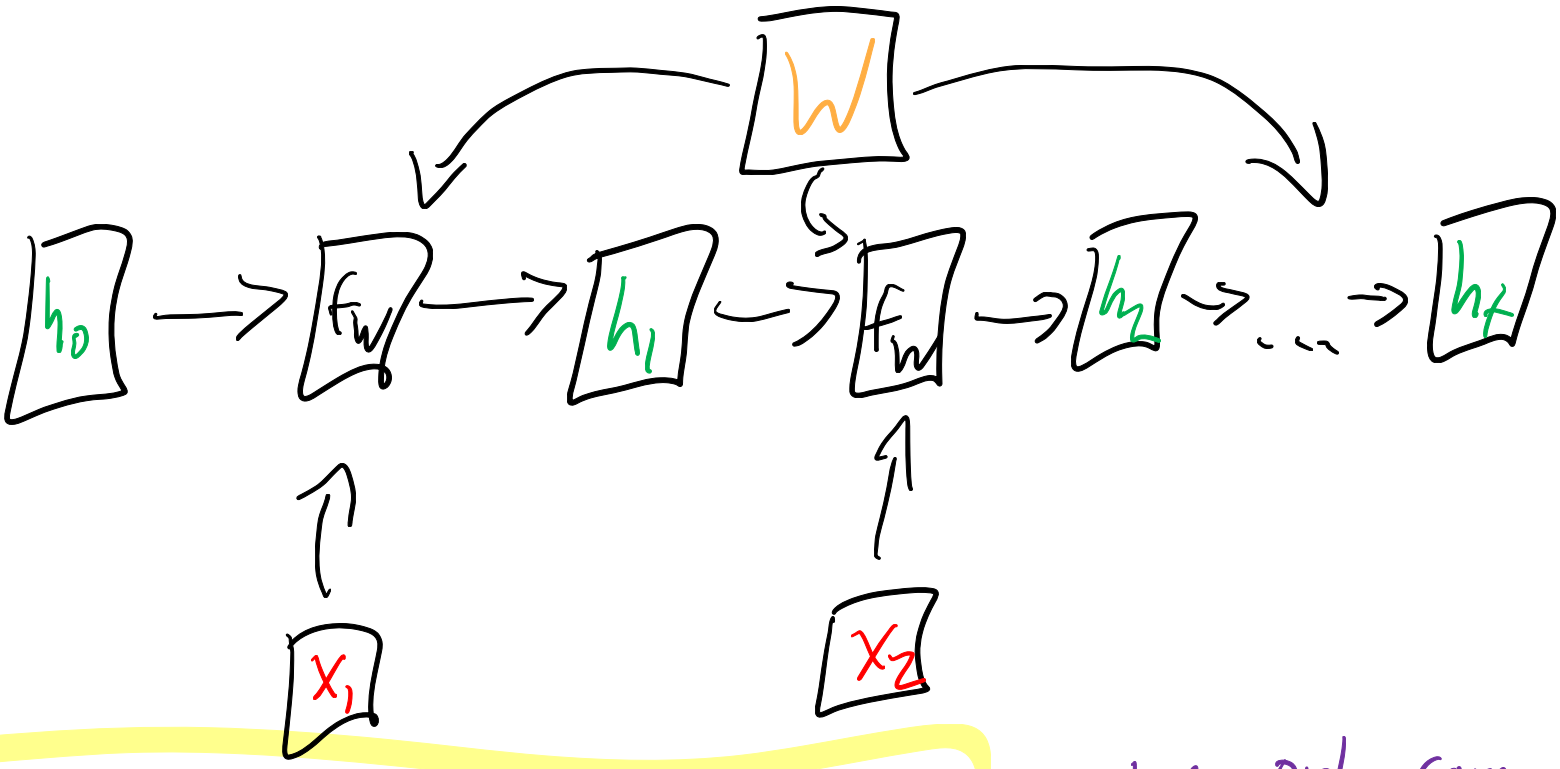
  - update output

$$y_t = W_{hy} h_t$$

- Forward pass

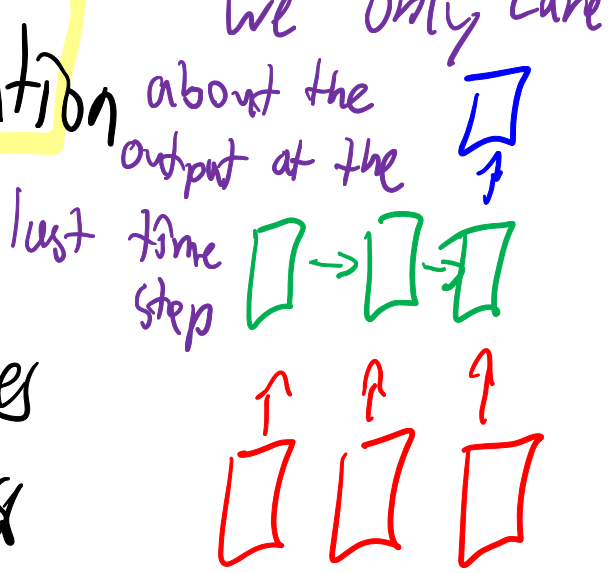
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} X_t)$$

  - weight sharing: same for all states

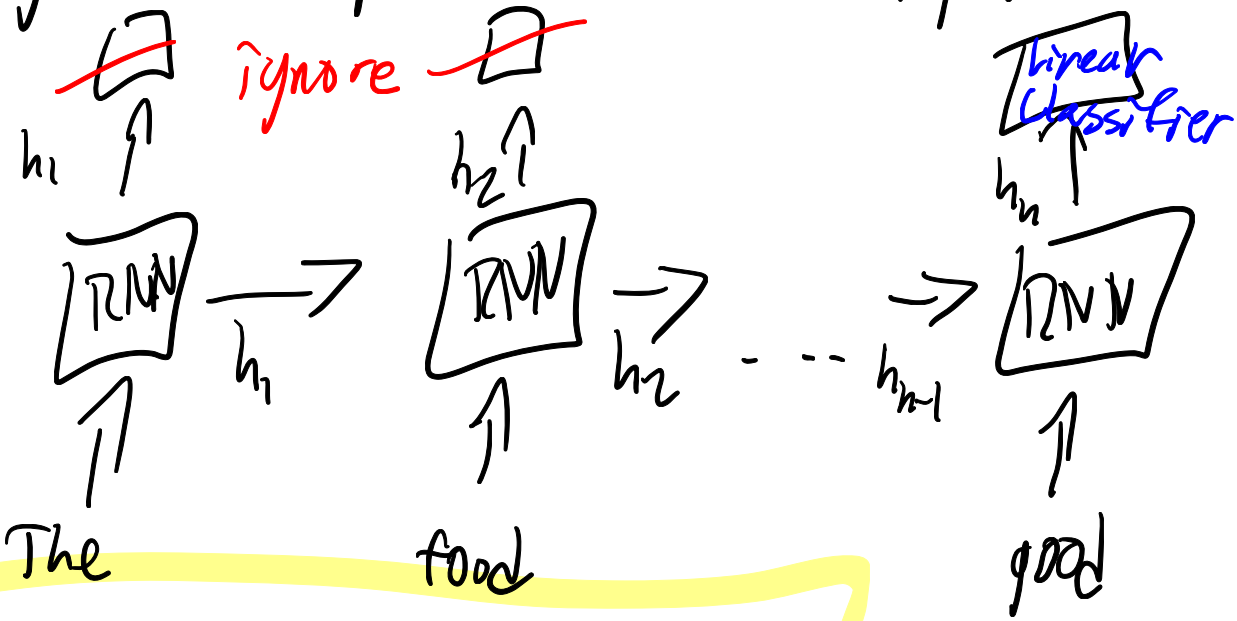


Example - Sentiment Classification

- Many to one
  - input: one or more sentences
  - output: positive/negative class



- ignore output of internal input



Example - Image Captioning

- One to many
  - input: image features (from CNN)
  - output: sentence describing image