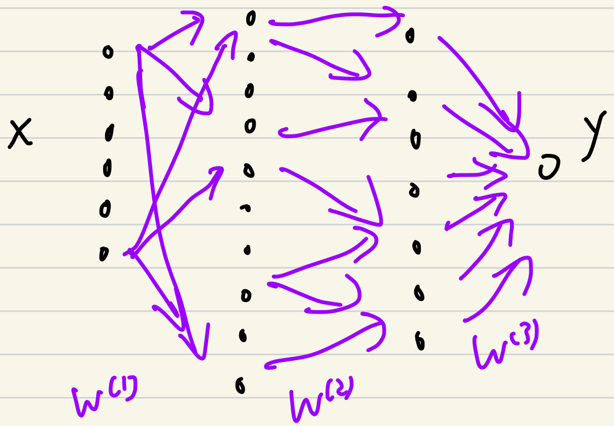
L10

midterm solution will be posted tomorrow.

* Recap
* Convolutional Neural Network (CNNs)
* Recurrent Neural Network (RNNs)



"Fully Connected"

$$y = W^{(3)} \sigma(W^{(2)} \sigma(W^{(1)} x))$$

problems of this picture :

* Computational Cost

e.x. large processing

Input : RGB images of 400 x 400

1 layer of neurons : 1000 neurons

Output : 10 classes

edges (weights)

$= 3 \times 400 \times 400 \times 1000 + 1000 \times 10$

RGB

bias

$=$ 500 million weights

Generally: L layer network

$O(L \cdot d^2)$

- ✱ Sample complexity
  - Require a very large training dataset
- ✱ Loss of context
  - all input is flattened into vector
  - e.g images lose special information
  - e.g NLP/speech

    have long-range temporal dependencies
- 混淆的

  ✱ Confounding features:
  -
  - Background clutter
  - illumination
  -

?

Convolution Neural Network (CNNs) 最常见 image classification

Two main principles:

① Locality: The model should look at local regions of your input image 如图像识别，应该只看物品那部分

② shift-invariance: The model should make the same prediction to the same object no matter where it appears

$\boxed{\text{Convolutional layers}}$

<u>Convolution</u>

$X[t]$ ← input

$W[t]$ called filter / kernel

$y(t) = (x * w)[t]$ 卷积表达式

$$= \sum_{\tau} X[t-\tau]\, w(\tau)$$

"Translate - and - scale"

Not hard to prove: $y(t) = \sum_{\tau} X(\tau)\, w(t-\tau)$

"Flip and dot"

* linear, associative, commu~ ?

* shift-invariance //

2D-images $\quad X(s,t) \quad w[s,t]$

2维 $$(x * w)[s,t] = \sum_{\sigma} \sum_{\tau} x[s-\sigma, t-\tau] \, w[\sigma,\tau]$$

Convolution layers

same as convolution, except no flipping.

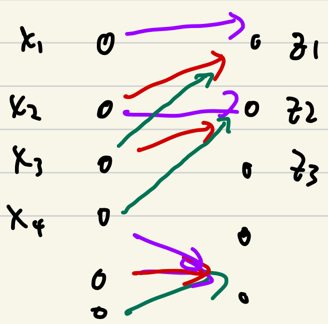
$$(x * w)[s,t] = \sum_{\tau=-\Delta}^{\Delta} \sum_{\sigma=-\Delta}^{\Delta} x[s+\sigma, t+\tau] \, w[\sigma,\tau]$$

$\Delta \rightarrow$ locality //

example $\Delta : 2 \longrightarrow$ restrict to $x[-2:2, -2:2]$ $\quad 5 \times 5 = 25$ indices of input

$$h[s,t] = \phi(z[s,t])$$

$$h[s,t] = \phi((x * w)[s,t])$$



☆ import to understand.

Consequences: 1) sparsely connected network

2) weights sharing

detect some part satisfy feature we want.

it is like a certain kernel determined to fine the feature can be used at any location of the image

In practice, we have D input channels

F ouput channels.

Input $X_1 \cdots X_2 \cdots X_D$

Output $Z_1, Z_2 \cdots Z_F$

$$\boxed{\begin{aligned} Z_i &= \sum_j X_j * W_{ij} \\ h_i &= \phi(Z_i) \end{aligned}}$$

Computational benefits:

$$(2\Delta+1)^2 \cdot D \cdot F = O(\Delta^2 \cdot D \cdot F)$$

个输入由于卷积处涉 $(2\Delta+1)$ 个 W

Eg. $(2\sigma+1)^2 \cdot D_F = 25 \cdot 3 \cdot 6 \approx \leq 600$

conv. RGB output 6 channel

different CNNs:

- LeNet - 5
- AlexNet (2012) similar but deeper
- VGG Net (2014)
- Google Net (2014)
- Residual Networks

Training: Back propagation

Recurrent Neural Networks (Useful for Time Series / NLP analysis)

NLP
- Document retrieval
- Speech to text
- Language translation
- Sentiment prediction

— short range
— long range

dependencies eg: 句子长短

Classical: Markov Models:

往前 dependency

$$p(\{w_1, w_2, \dots, w_d\})$$

$$= p(w_1)\, p(w_2 | w_1) + p(w_3 | w_2) \cdots p(w_d | w_{d-1})$$

long range dependency

$p(w_3 | (w_1, w_2))$ 第三个word依赖 w_1, w_2 → 更复杂]

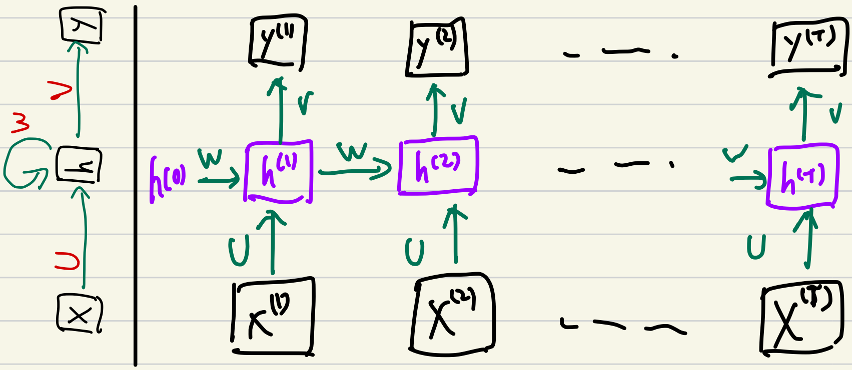
Recurrent neural nets: introduce feed back into neural nets

$$\begin{tikzcd} \boxed{X} \arrow[r, "U"] & \boxed{h} \arrow[r, "V"] \arrow[loop above, "W"] & \boxed{Y} \end{tikzcd}$$

X^t indexed with time

$$h^t = \phi(Ux^t + Wh^{t-1})$$

$$y^t = Vh^t$$



U. V. W don't change over time

∴ weight sharing

Examples:

- Basic RNN (h is single layer)
- GRU (Gated recurrent units)
- LSTM (Long-short term memory)
- Attention networks