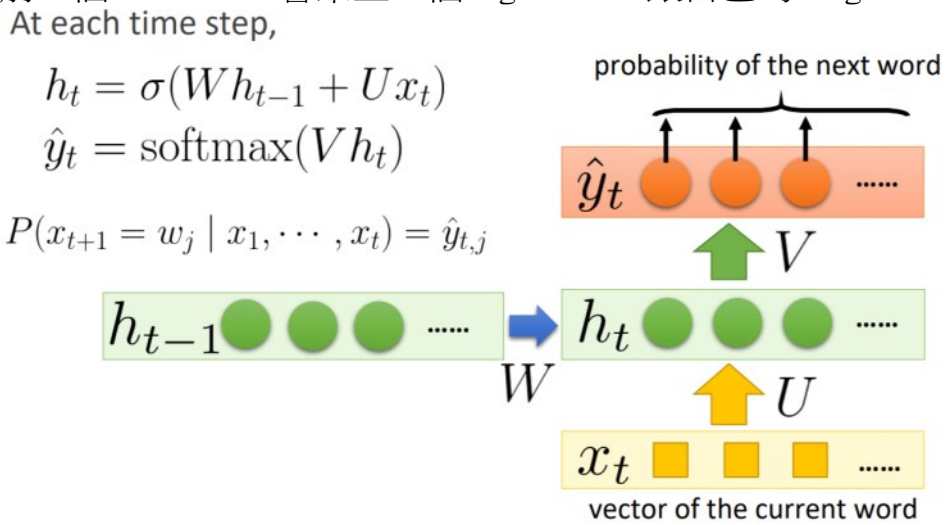
ADL—RNN

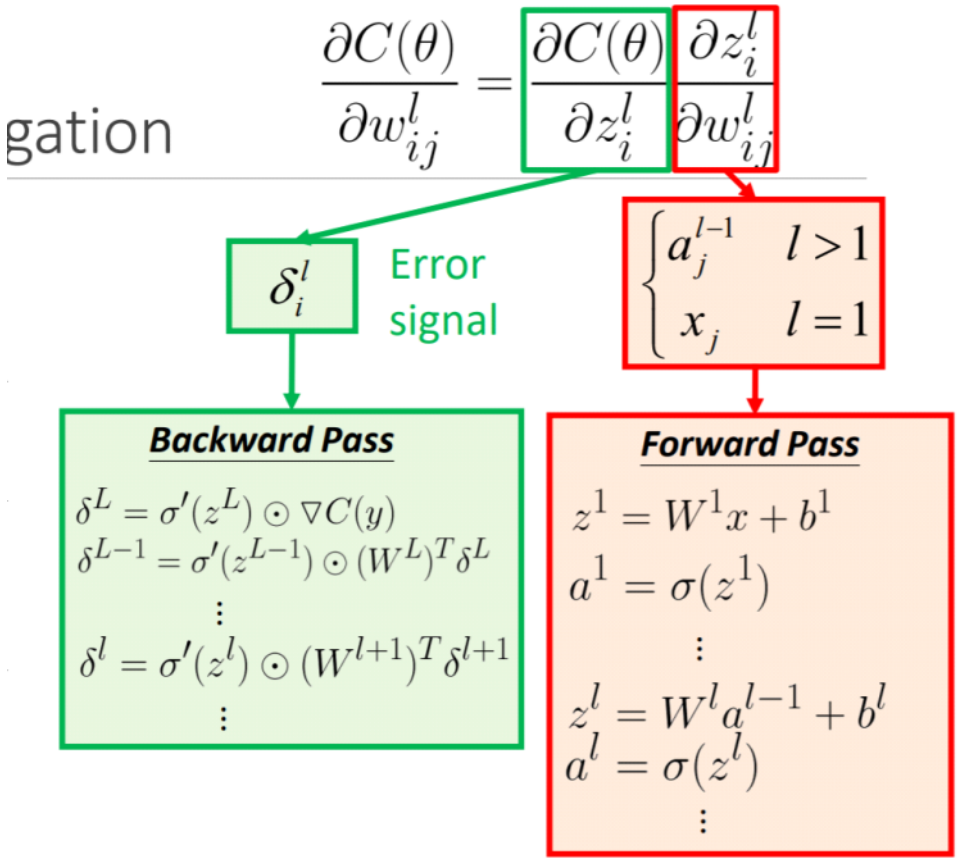
2019年3月19日　　上午 09:17

hidden state往下傳，從前一個時間點到下一個時間點，所以會考慮前文
softmax: 機率分布
前一個hidden state會乘上一個wight matrix跟自己的weight matrix相拼在一起

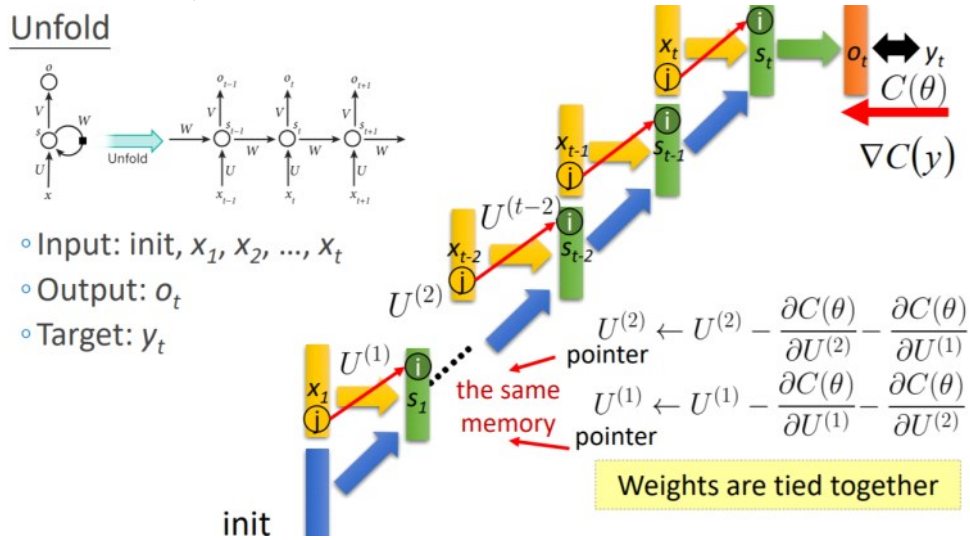


要學的參數只有三個matrix: U、V、W
(假設shared weight，減少參數，不同input相同weight就會吐出結果，句子長度不同參數仍是一樣的)
=

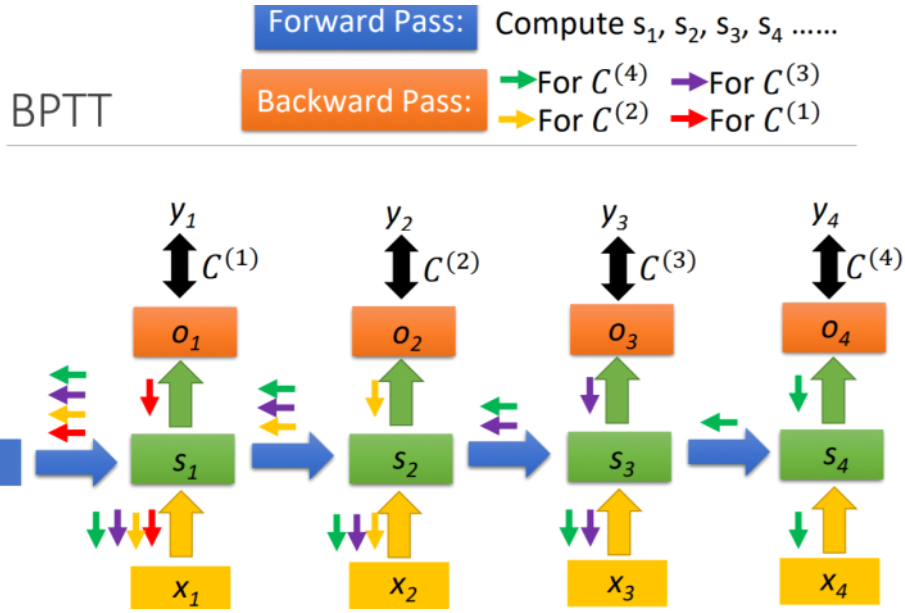
- BP: loss function對偏微分的結果，要讓cost function下降
  - Backward pass: gradient，每一層的output gradient。從最後一層往前推 (error signal，從最後面往前推的結果)
  - Forward pass: 照著network的順序走



- BPTT: 時間上面的layer-wise更新
  - U2更新的時候U1也要跟新，所以最後要更新的時候是全部U的方向更新 (pointer去改同一塊memory的資訊，所以最後的memory就是最終資訊)



  - Backward pass是從4開始做，然後3的greadient再加近來，再加2的1的…最後把四個error全部一起加起來一起update



- RNN training很麻煩，有很多matrix相乘，相乘很多次最後不是vanishing就是exploding (gradient可能會不見或是非常大)
  - cost很不穩定 ，gradient接近0或是太大
- exploding gradient解法
  - clipping: 每次都control在某個範圍裏面，不會飛太遠
- Vanishing 解法
  - IRNN: 用identical matrix來initialize，用ReLU

- 為了要handle很長的文章=>gate
  - 傳統的gradient經過很長的gradient相加以後的error就已經被消失了
  - 可以靠一些門，不經過相乘就可以直接過去=>LSTM
    - encode long distance info

- BiRNN
  - hidden state除了左到右，也會考慮右到左
  - 但有些task不是左右兩邊都可以拿到
    - 有些是說完才做判斷，那就可以
    - 但如果是股票預測，想預測明天會拿不到後天的資訊
  - 時間軸上的deep，layer-wise的deep，希望每一層學到越來越high level的資訊

Application:

- input如果是sequence就可以把它用RNN aggregate在一起
  - 傳統是相加或是取平均
  - 這邊是RNN，最後一層hidden state的輸出vector，就可以根據此vector判斷 EX: FC，把後面的
    - temporal資訊encode變成vector拿來分類
- POS tagging、speech recognition、machine translation
  - output : sequence labeling

    - information 跟input是bind在一起的
    - 一個input對應一個output
- Natural Language Understanding (NLU)
    - 理解語意: 一句話講完以後要知道她要做甚麼EX: 主旨、內文
    - 所以每一個字都給一個tag，最後就變成api call
- I/O 不是align在一起的時候，順序完全不一樣
    - 需要一個RNN把input aggregate起來，後面會decode出答案變成很大的RNN
    - seq2seq