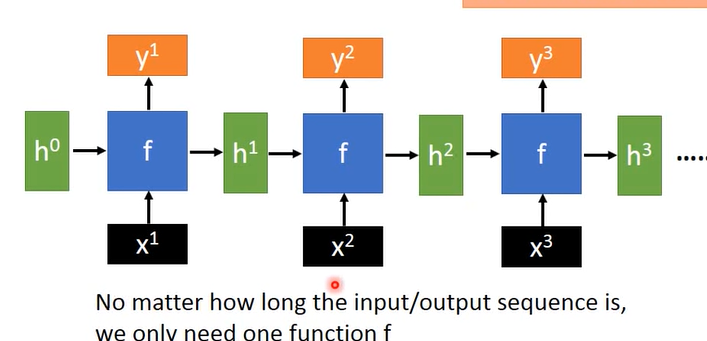
1.另一个方面来理解RNN: Recurrent Structure: 同样的结构反复使用， 不管input 多长的sequence, 所用到的weights 数目是不变的？

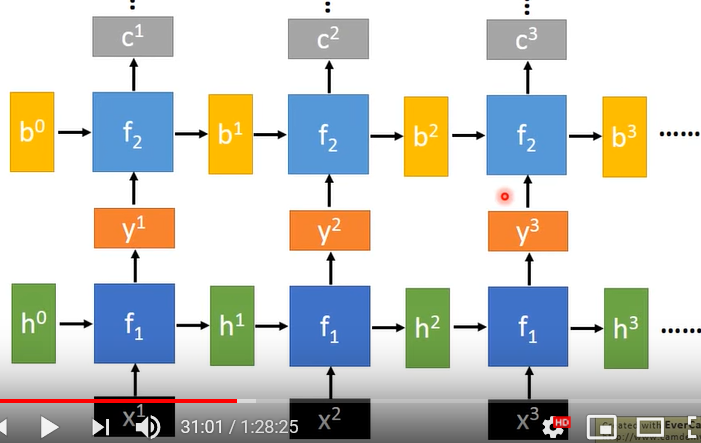
Give function:

f(h,x) -> h’,y : recurrent network: 同样的结构重复利用，h 与h’的dimension必须是相同的。 单层的rnn 如下：



RNN 比较难train， 但是train好了之后能够得到比较好的处理结果。

2层的RNN如下：



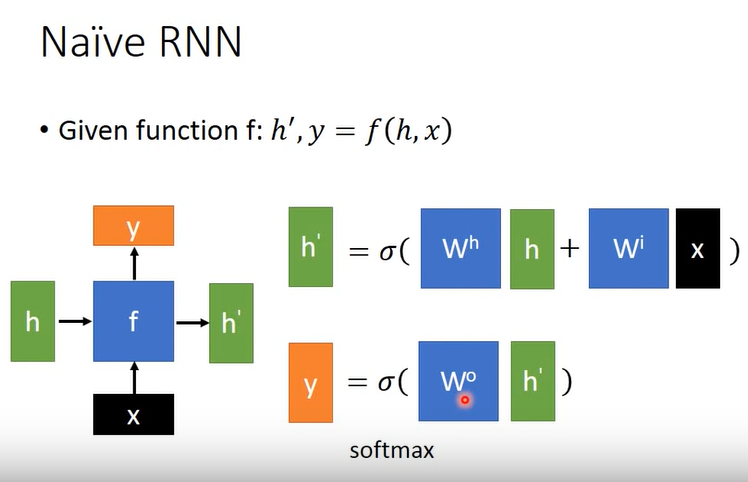
Bidirectional RNN:

一个正向的,一个反向，第三个函数结合两个不同向网络的输出形成新的输出。

Pyramidal RNN： sequence to sequence(硬train): 后一层每一个吃的输入是前一层几个的输出。

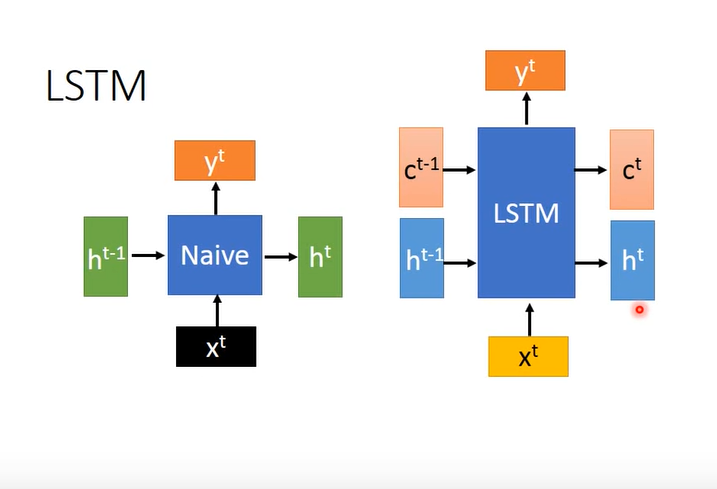
Naïve RNN： f最naïve 的方式： 线性组合 + sigmoid 函数(或其他non-linean function)

而y 是根据h’ 来得出，线性组合

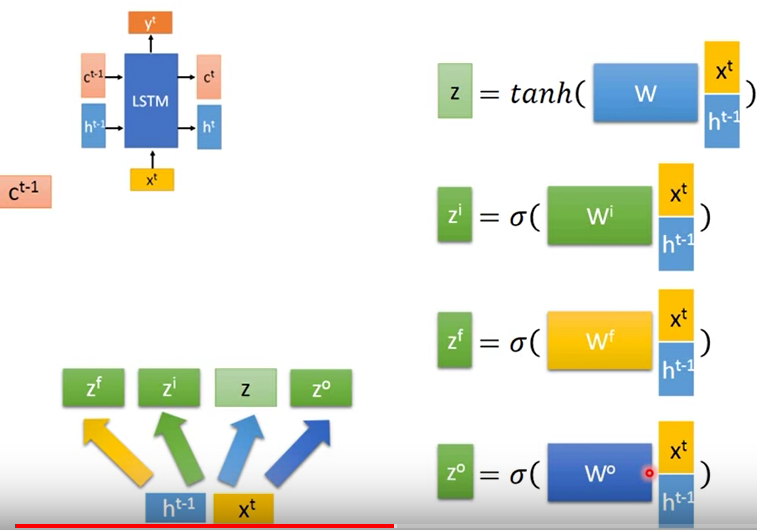


LSTM 详解

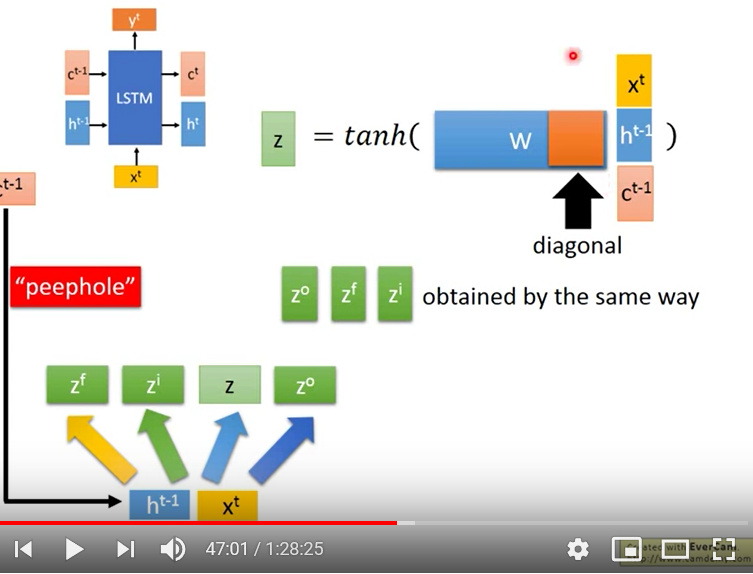
LSTM 的基本单元:

C的变化是很慢的，C\_t-1 加了什么之后变成了C\_t, 而h 的变化是很快的，ht与ht-1之间的关系很大。

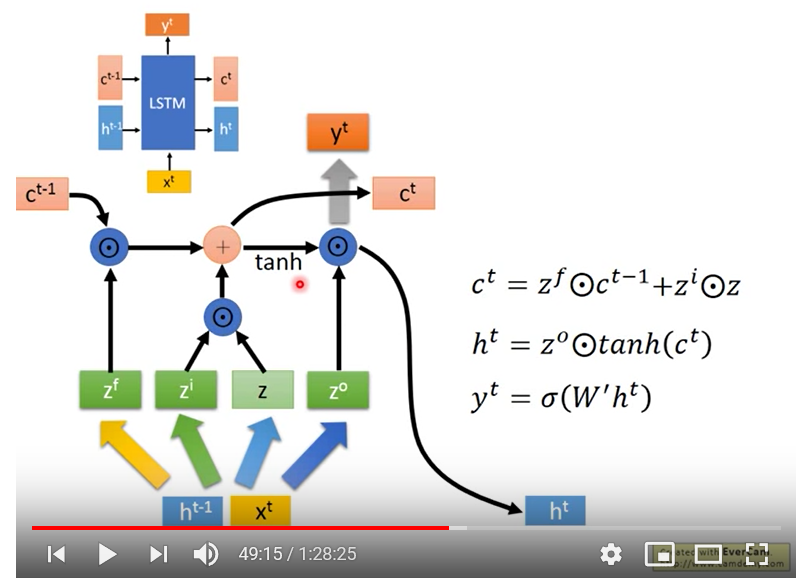
C 的information 传递的慢，可以记得long term 的info



有时也可以使用“peephole”的方法

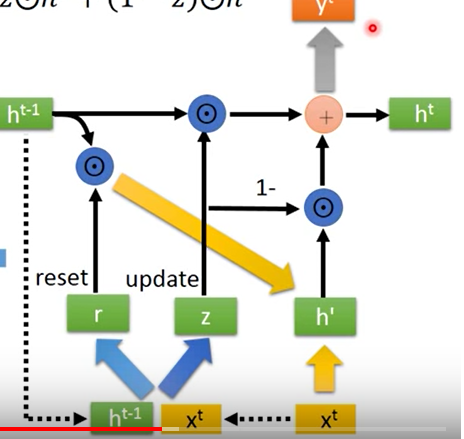


之后Z 与Z^I element wise 乘, C\_t-1 与 Z^f element wise 想乘，之后二者想乘的结果相加。



GRU：没有变化很快的h和变化很慢的C

GRU： 将h\_(t-1) 与 输入x\_t 拼接起来， 通过weight 矩阵后经过sigmoid函数， 产生r: reset gate , z: update gate ， 其流程示意图如下：

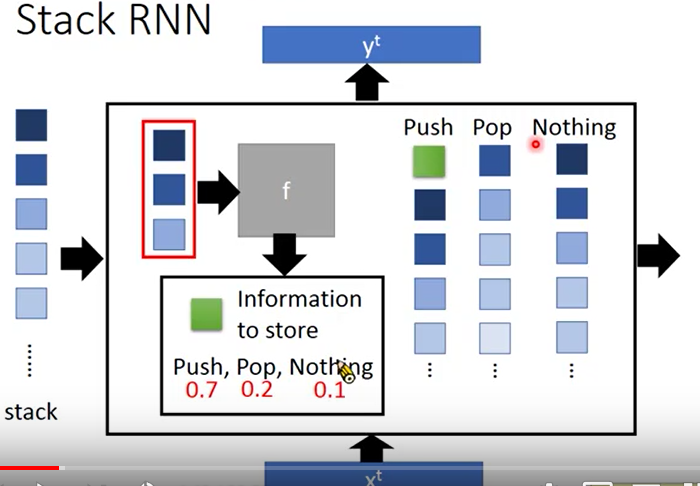


GRU仅仅用到3个参数matrix， 直接去掉了一个gate

GRU 与 LSTM的关系: h 可以看成是memory

在语音辨识时，可以使用target delay的方法，能够让RNN 读完整个phonics.

Stack RNN: 需要的参数量与input的vector的长度是无关的，因为仅仅pick input的前几个element，丢进function f(同时接受输入x), 输出要store的information， 同时output 其他三个value：push-把information push 到 stack 中,pop – 丢掉stack 上的元素,nothing-不做任何操作. 则push, pop, nothing 各做义词，并按照他们的值加权



CNN / POOLING 复习

特性:

1. Sparse connectivity

2. parameter sharing (filter)

GAN

Structural learning: 输出的结果是sequence / tree /matrix 等等, GAN是一种通向structural learning的引路人.

Structural learning 的方式:

1.bottom up: learn to generate the object at the component level

2.top down：evaluating the whole object, and find the best one – 但仅仅学会evaluate，如何产生？

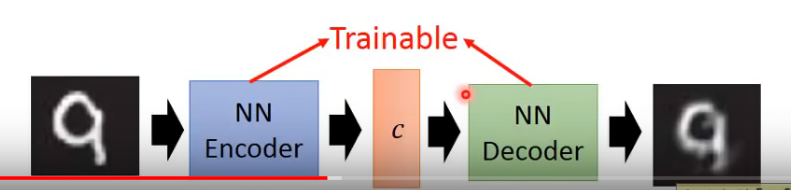
在GAN中， generator 就是bottom-up的方式，而discriminator是top-down的方法。

在generate 动漫头像的例子中，那么为什么不直接将图像feed给generator呢？

因为在实际中，只有图像output,并不知道什么样子的vector 对应什么样的图像.

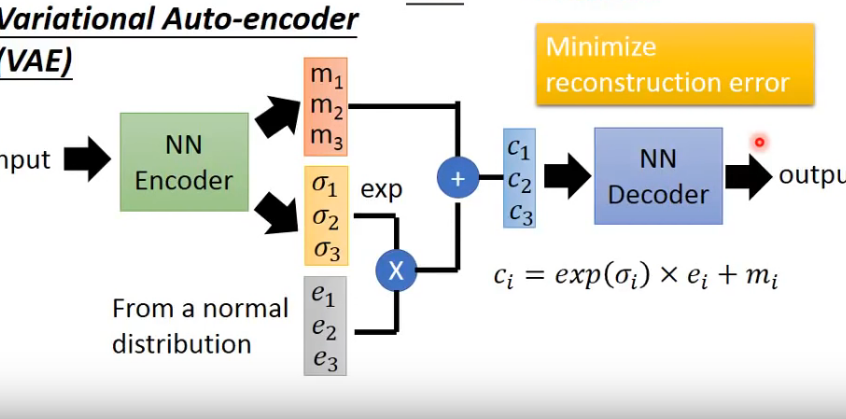
一个解决方案: 用auto-encoder 对图像进行降维, 产生的code/vector 作为输入

Auto-encoder: 一个encoder，一个decoder，同时train，目标使得输出图像与输入图像越接近越好。

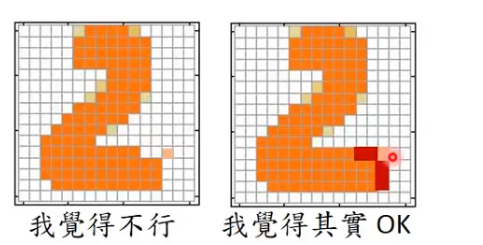


在此过程中，拿出的decoder可以作为GAN中的generator来使用。

在以上训练encoder以及decoder的过程中，可能会产生过拟合的问题。那么解决方案是：VAE. 即在训练过程当中，encoder产生噪声并加到产生的vector上，再进行训练。（noise太小也应该有惩罚）



而我们并不希望generator 完全的copy 目标图像，因此可以要求generator 犯一些错误。Pixel 与pixel之间的关联是十分重要的，



例如以上，我们宁可6个pixel的错误，也不希望只有1个pixel的错误。

但是想让网络学习到pixel与pixel之间的关系是比较困难的。

Discriminator: 输入一副图像，输出一个数值表示图像的好坏。

能够通过discriminator 来产生图像么？可以。 并且discriminator是比较容易抓住pixel与pixel之间的关系。 那么如何让discriminator 取判断好坏呢？

目前只有好的图片， 那么必须要有一些差的图像。 差的图像最好能够“以假乱真”，所以random 生成的图像并不可靠。

那么如何生成假的图像呢(realistic negative examples)？

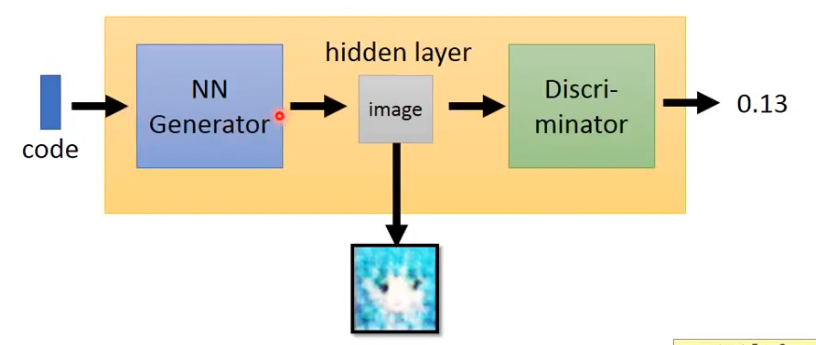
可以利用: General Algorithm

一些positive example: label(1), 一些random examples：label(0). 同时进行徐连，然后拿训练好的discrimiator 取生成新的negative example， 然后把这些图像feed回网络不断进行下去, 直到discriminator的参数收敛。

Generator + Discriminator

Discriminator: 要求解 x = argmax D(x), 而x 实际上可以用generator 的输出来代替。

实际上generator 和 discriminator 串联如下图:



GAN 的流程：

Train的目标是希望discriminator 的output 越大越好，因此使用gradient ascent 调整generator 的参数（但注意此时discriminator的参数固定）；当generator的参数能够使得discriminator的输出足够大的时候，将generator 输出的中间结果(图像)标记为0分，然后单独train discriminator。（generator 目的在于讨好”Discriminator”）

Conditional generation: 根据指令产生图片

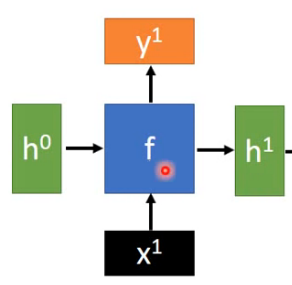
# Advanced topic In Deep Learning

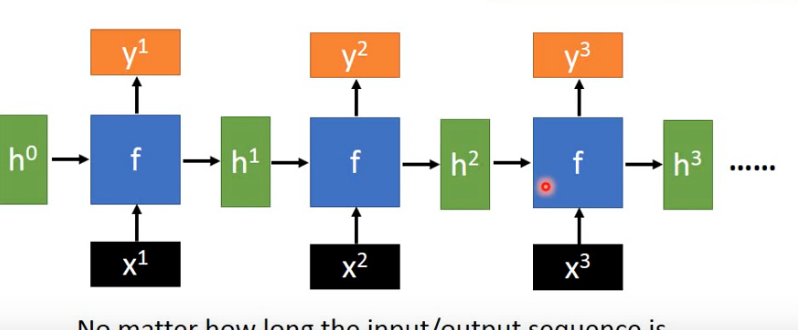
Recurrent Structure: 将同样的network反复的使用。

Recurrent Neural network: function:

h’, y = f(h,x ), 注意f 吃2个vector 作为input: h, x, 同时输出两个vector 作为output.

那么将function f 不断的反复使用(要求h 与 h’的维度是始终一致的).



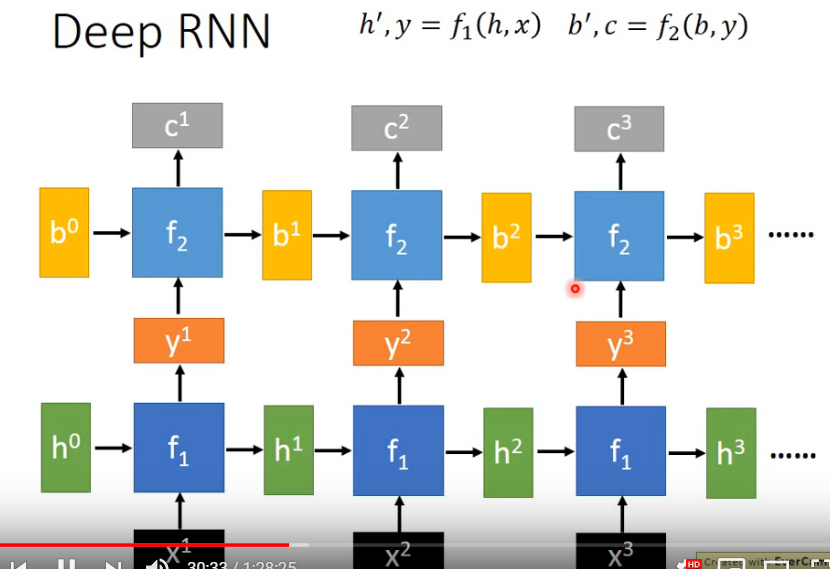


当sequence 又长有短的时候,后面补0. 当sequence很长的时候，仅仅需要比较少量的参数，因为仅需要一种结构f。

feed forward network 对sequence的处理：比较容易overfitting， 因为需要比较多的参数。

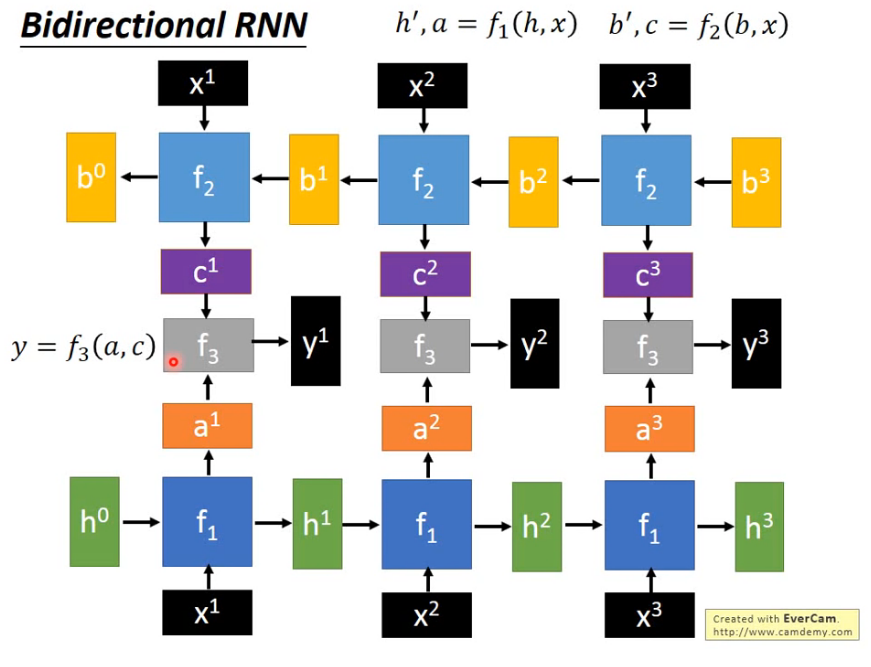
RNN 对sequence的处理：比较难train， 但是train好以后会具有比较好的泛化能力。

Deep RNN：



新设计能够吃y的 f2 ，f3 等等即可。

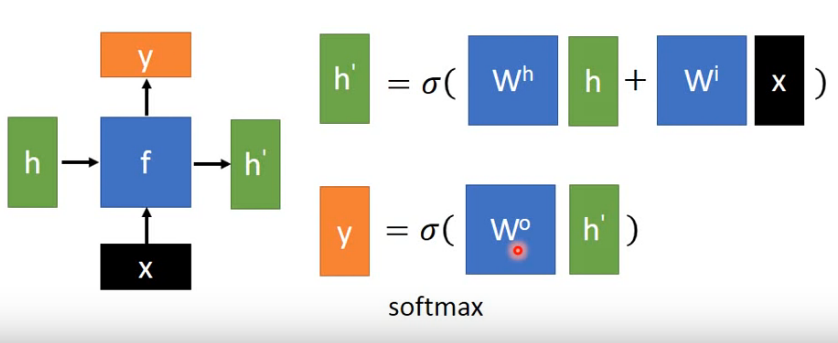
Bidirectional RNN



Bidirectional 的需要一个back 的网络来进行。

那么RNN 的function 应该长什么样子？

h’ 是通过 x 与前一个隐藏阶段的 h 的各自线性变换的加和，在通过sigmoid函数得到，而 y 是根据h’ 的线性变化通过sigmoid函数得到。 Naïve 的RNN cell 如下所示：



如果最后的输出希望是概率的时候，那么加一个softmax 函数。

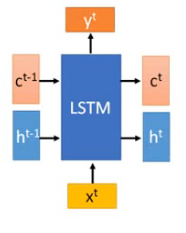
LSTM：

Lstm 的基础结构： LSTM 吃 3个input，其中包含c, h, 以及当前的input x;

C: change slow , c^t 与 c^(t-1) 通常是加什么东西的关系

H: change faster, h^t 与h^(t-1) 可以非常不同

LSTM cell 的基础结构如下图所示:



计算流程：(注意以下w都并不相同)

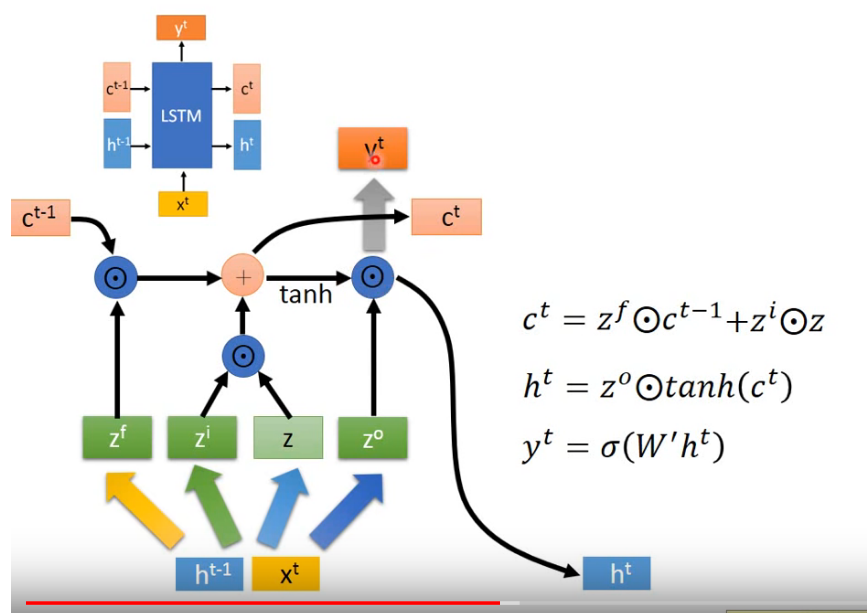
Z = tanh(wx^t + wh^(t-1))

Z^i=sigmoid(wx^t+w\*h^(t-1))

Z^f=sigmoid(wx^t+w\*h^(t-1))

Z^o =sigmoid(wx^t+w\*h^(t-1))

然后进行elment-wise 的乘法运算

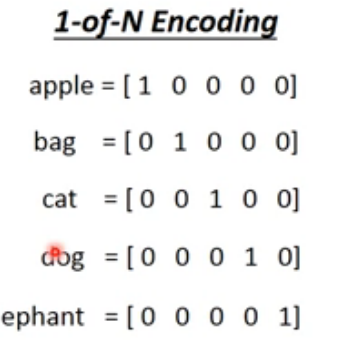


GRU – 相对来说参数量较少。

# Unsupervised Learning – Word Embedding

是一种降维手段。

如果使用one-hot 的方式(1 of N Encoding)，那么可能相近之间的word的距离会产生缺失。

例如cat 和 dog 之间的关系无法显示

另一个方法： Word class – 将相似的word放在一个class中

Word embedding： 将word 映射到一个高维空间(远比1 of N Encoding 的维度低), 并且每一个dimension 都有其代表的含义。 是一种unsupervised的方式。

问题描述： training data： 许多的text，但没有word对应的embedding

期望结果：给定出一个word，NN输出其embedding的vector

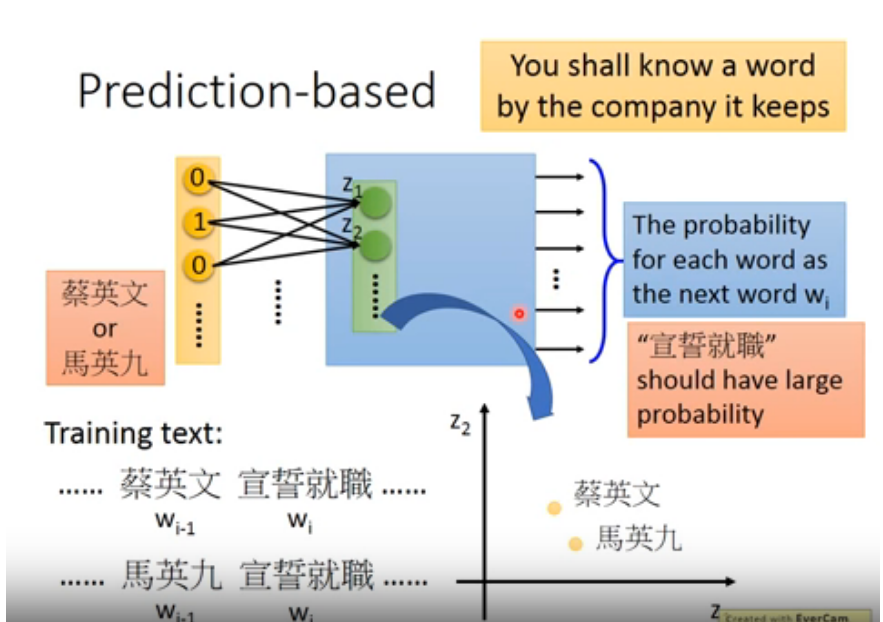
（不能用auto-encoder做，那么auto-encoder的机制？）

Word embedding 的机制:

当机器阅读大量文章之后，发现两个word的context很相似，那么则具有相似的属性。

例如： Glove vector.

Prediction-based: 根据上文的word(n-1) 的one-hot encoding 作为input vector, 将word(i) 的one-hot encoding作为目标向量，中间通过一个神经网络。当训练完成后，仅取第一个hidden layer的输出作为word embedding 的结果.



Prediction-based 的方法也可以看前面的多个word(一般10个左右).

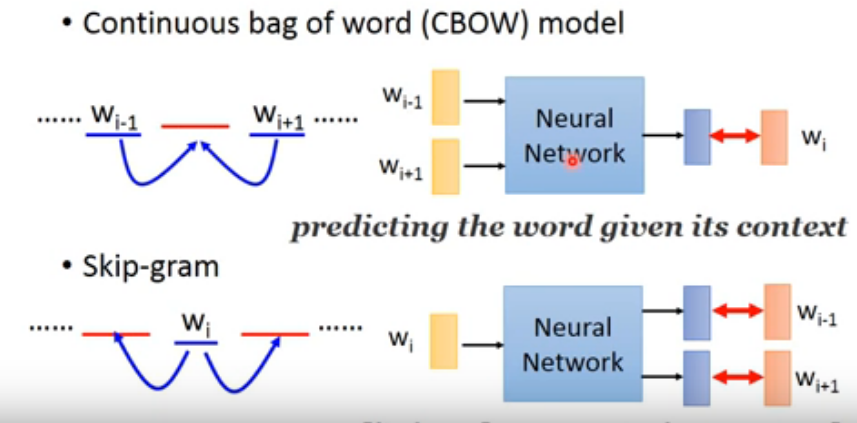
做法： 将前面的word的vector拼接，作为NN的输入即可。但是强制让每一个word的对应的W都相等，这就等同于 z = W(x(i-1) + x(i-2))

公用参数做法： Wi,Wj给定同样的初始值，在梯度下降时下降的梯度等于Wi的梯度+Wj的梯度，这样确保训练过程中Wi与WJ相等。

Prediction-based embedding 的变形：

CBOW: 用word(i-1)与word(i+1) 取predict word(i)

Skip-gram: 用word(i) 去predict W(i-1)和W(i+1)



而实际上embedding的model 并不是deep 的，通常只要一层NN，因为一层的NN也可以取得非常好的效果，不需要deep。

Neighbour Embedding:

Manifold ：绝大多数数据都是分布在一个低维空间中，而被扭曲到高维空间来表示。

因此，欧式距离在低维的时候才比较有效，而在高维空间中将会失效。

Locally Linear Embedding(LLE)

原理：

首先对原有空间的点xi, 对于其所有的邻居xj, 都有wxj, 那么我们minimize：

Xi – sum(wij\*xj), 也就是让所有的neighbour的加权和都越接近xi 越好。

那么对原空间降维，xi，xj用zi,zj表示。 那么对于同样的weights，我们同样

的要求 minimize: zi – sum(wijzj)， 满足这样的降维变换x -> z就是我们想要的降维。

LLE中neighbour 的数量也需要调整。

Auto-encoder – 一种un-supervise learning的方式

在没有label的情况下：

Image -> Encoder-decoder -> Image, 来训练

Auto-encoder 可以用来降维，将高纬度的image/sentence/document 可以变成低维的vector。

而使用auto-encoder 的方式，由于是无监督的，数据集一般不缺。

Auto-encoder出的结果，也可以用作参数的初始化(作为pre-train)-使用layer-wise的方法。

（强调- L1的regularization 是sparse的）

CNN auto-encoder：

还原image要de-convolution：

实际上，de-convolution就padding之后的是convolution

Decoder的妙用： train好的decoder，给定一个输入的向量，可以产生图像的结果。

# Structural Learning