1. 端到端的学习

非端到端：传统机器学习的流程往往由多个**独立**的模块组成。例如传统的语音识别系统，是由多个模块组成的，包括**声学模型、发音词典、语言模型**。其中**声学模型**和**语言模型**是需要训练的。这些**模块的训练一般都是独立进行的**，各有各的目标函数，例如声学模型的训练目标是最大化训练语音的概率，语言模型的训练目标是最小化perplexity。由于各个模块在训练时不能互相取长补短，训练的目标函数又与系统的整体性能指标有偏差，这样训练出的网络往往达不到最优性能。
针对这个问题，一般有两种解决方案:

- **端到端训练（end-to-end training):**一般指的是在训练好语言模型后，将声学模型和语言模型接在一起，以 WER 或它的一种近似为目标函数去训练声学模型。由于训练声学模型时要计算系统整体的输出，所以称为「端到端」训练。然而这种方法并没有彻底解决问题，因为语言模型还是独立训练的。

- **端到端模型（end-to-end models）**：系统中不再有独立的声学模型、发音词典、语言模型等模块，而是从输入端（语音波形或特征序列）到输出端（单词或字符序列）直接用一个神经网络相连，**把声学模型、发音词典、语言模型这些传统模块融合在一起**，让这个神经网络来承担原先所有模块的功能。而深度学习模型在训练过程中，从输入端（输入数据）到输出端会得到一个预测结果，与真实结果相比较会得到一个误差，这个误差会在模型中的每一层传递（反向传播），每一层的表示都会根据这个误差来做调整，直到模型收敛或达到预期的效果才结束。

2.批量梯度下降（BGD）与随机梯度下降（SGD）

2.1 定义

考虑需要优化的**目标函数（准则）**$J(\theta)$，参数 $\theta \in R^d$,当我们对其进行最小化时，我们也把它称为**代价函数、损失函数或误差函数。**
该目标函数对应的梯度为：$\nabla_\theta J(\theta)$，设学习速率为 η , 则：

- 对于**批量梯度下降-Batch gradient descent(BGD)**，其更新公式如下：

$$\theta := \theta - \eta \nabla \theta_J(\theta)$$

  注：对于BGD，batch等于所有训练样本数，即每次更新使用了所有的样本。

- 对于**随机梯度下降-Stochastic gradient descent(SGD)**，其更新公式如下：

$$\theta := \theta - \eta \nabla \theta_J(\theta, x^i, y^i)$$

  注：对于SGD，batch等于1，**每次迭代可以只用一个训练数据来更新参数。**

2.2 示例

考虑线性模型：

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

取平方损失函数作为目标函数：

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$

则 **BGD的求解思路如下：**

- 1. 将 $J(\theta)$ 对 θ_j 求偏导，得到每个 θ_j 对应的的梯度:

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m} \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$$

  注：x_j^i 代表第 i 个样本的第 j 个属性。

- 2. 由于是要最小化风险函数，所以按每个参数theta的梯度负方向，来更新每个 θ:

$$\theta_j^{'} = \theta_j + \frac{1}{m} \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$$

SGD的求解思路如下：

- 1. 将 $J(\theta)$ 对 θ_j 求偏导，得到每个 θ_j 对应的的梯度:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\ &= (h_\theta(x) - y) \, x_j \end{aligned}$$

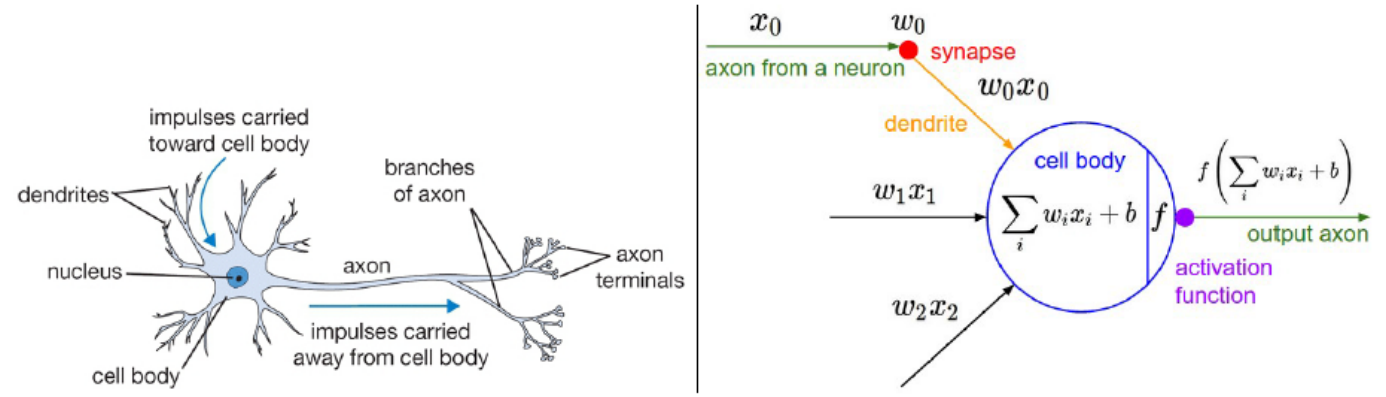
- 2. θ 更新规则如下：

$$\theta_j := \theta_j + (y - h_\theta(x)) x_j$$

2.3 对比

- **批量梯度下降** --- 最小化所有训练样本的损失函数，使得最终求解的是全局的最优解，即求解的参数是使得风险函数最小。得到的是一个全局最优解，但是每迭代一步，都要用到训练集所有的数据，如果m很大，那么迭代速度很慢。
- **随机梯度下降** --- 最小化每条样本的损失函数，虽然不是每次迭代得到的损失函数都向着全局最优方向， 但是大的整体的方向是向全局最优解的，最终的结果往往是在全局最优解附近。

3. 激活函数总结



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

激活函数：在神经网络的神经元上运行的函数，负责将神经元的输入映射到输出端。

$$y = f(\sum_{i=1}^{n} \omega_i x_i - \theta)$$

性质：

1. 当激活函数是线性的时候，一个两层的神经网络就可以逼近基本上所有的函数了。但是，如果激活函数是恒等激活函数的时候（即f(x)=x），就不满足这个性质了，而且如果MLP使用的是恒等激活函数，那么其实整个网络跟单层神经网络是等价的。

2. 可微性： 当优化方法是基于梯度的时候，这个性质是必须的。

3. 单调性： 当激活函数是单调的时候，单层网络能够保证是凸函数。f(x)≈x： 当激活函数满足这个性质的时候，如果参数的初始化是random的很小的值，那么神经网络的训练将会很高效；如果不满足这个性质，那么就需要很用心的去设置初始值。

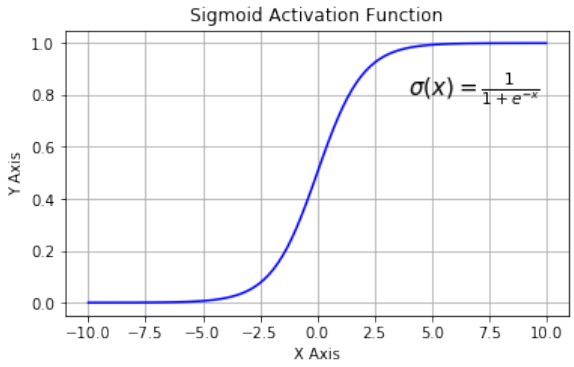
4. 输出值的范围： 当激活函数输出值**有限**的时候，基于梯度的优化方法会更加稳定，因为特征的表示受有限权值的影响更显著；当激活函数的输出是**无限**的时候，模型的训练会更加高效，不过在这种情况小，一般需要更小的learning rate.

3.1 Sigmoid函数

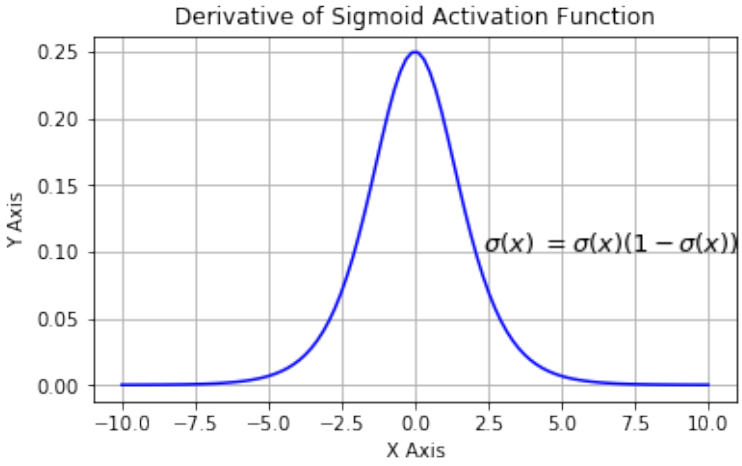
- Sigmoid函数表达式为:

$$y = \frac{1}{1 + e^{-x}}$$

- 函数图像如下:



- 函数导数的图像如下:



- 解析：在物理意义上最为接近生物神经元，能够把输入的连续实值“压缩”到[0,1]。此外，[0,1]的输出还可以被表示作概率，或用于输入的归一化。
- 缺点:

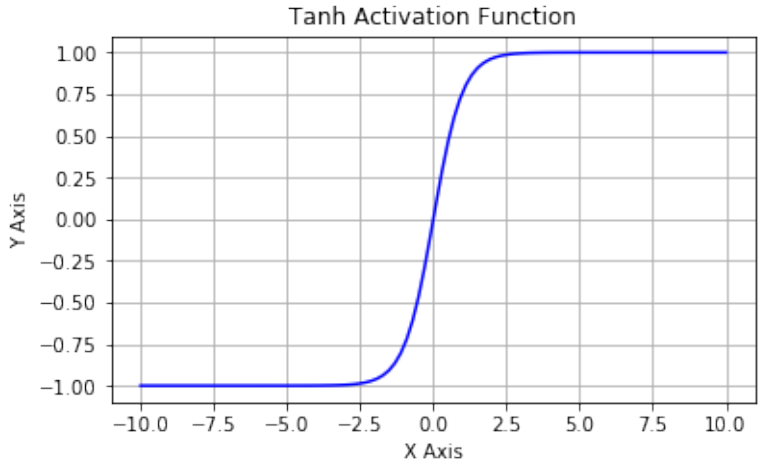
1. **梯度消失**： 注意：Sigmoid 函数趋近 0 和 1 的时候变化率会变得平坦，也就是说，Sigmoid 的梯度趋近于 0。神经网络使用 Sigmoid 激活函数进行反向传播时，输出接近 0 或 1 的神经元其梯度趋近于 0。这些神经元叫作饱和神经元。因此，这些神经元的权重不会更新。此外，与此类神经元相连的神经元的权重也更新得很慢。该问题叫作梯度消失。
2. **sigmoid函数的输出均大于0**：这使得输出不是0均值，这称为偏移现象，这会导致后一层的神经元将得到上一层输出的非0均值的信号作为输入。
3. **计算成本高昂**： exp() 函数与其他非线性激活函数相比，计算成本高昂。 鉴于以上几点，Sigmoid 函数现在已经很少使用。

3.2 Tanh函数

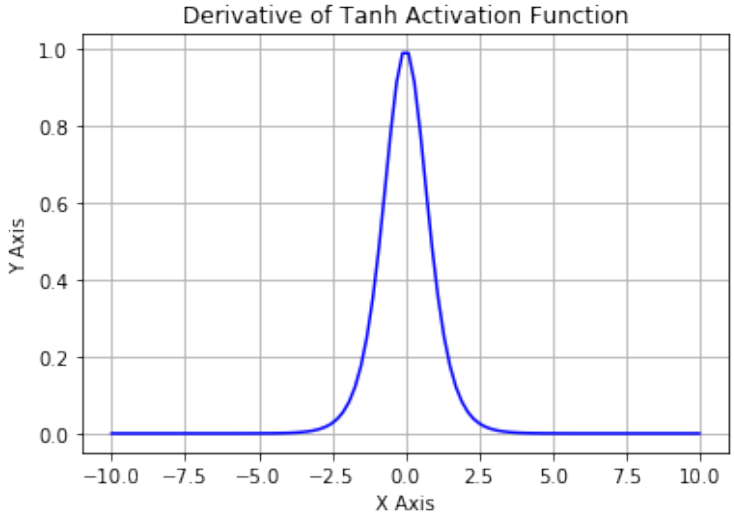
- Tanh函数表达式为:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

- 函数图像如下:



- 函数导数的图像如下:



- 解析：Tanh 函数将其压缩至-1 到 1 的区间内。与 Sigmoid 不同，Tanh 函数的输出以零为中心，因为区间在-1 到 1 之间。Tanh 函数的使用优先性高于 Sigmoid 函数。负数输入被当作负值，零输入值的映射接近零，正数输入被当作正值。与sigmoid相比，它的输出均值是0，使得其收敛速度要比sigmoid快，减少迭代次数。

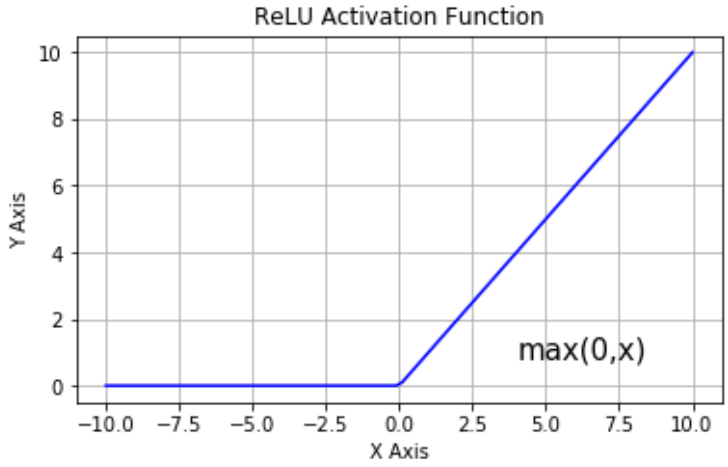
- 缺点：Tanh 函数也会有梯度消失的问题

3.3 修正线性单元（ReLU）

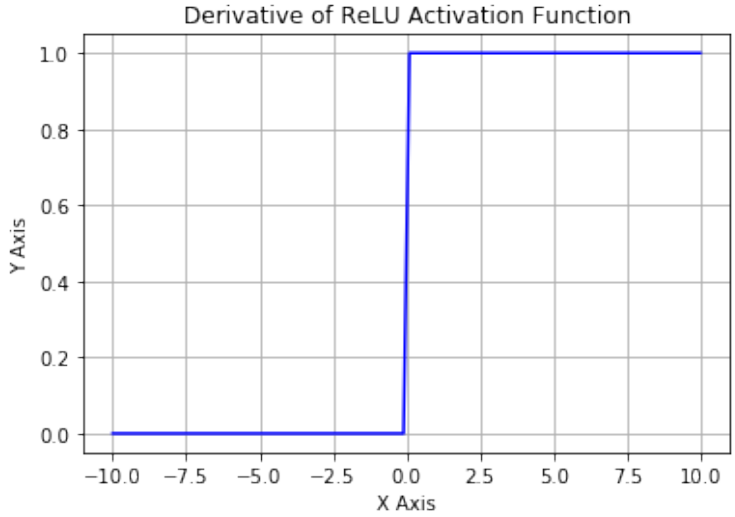
- ReLU函数表达式为:

$$ReLU(x) = max\{0, x\}$$

- 函数图像如下:



- 函数导数的图像如下:



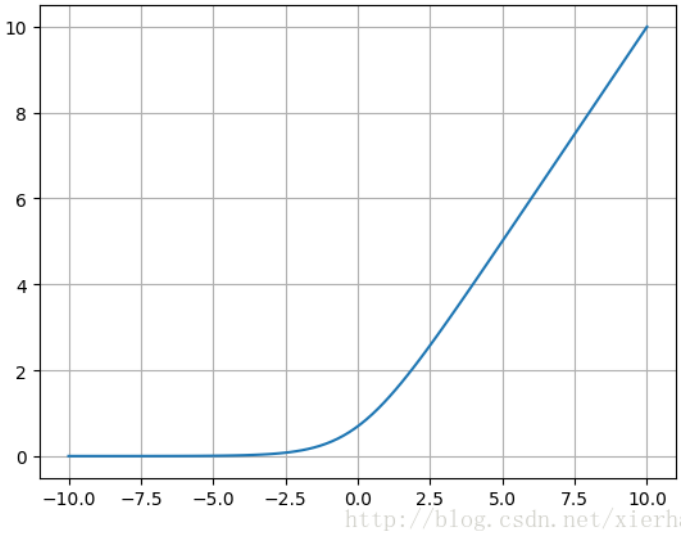
- 解析：最近几年卷积神经网络中，激活函数往往不选择sigmoid或tanh函数，而是选择relu函数。随着训练的推进，部分输入会落入 $x < 0$ 的区域，其梯度等于0，导致对应权重无法更新。这种现象被称为“神经元死亡”。与sigmoid类似，ReLU的输出均值也大于0，偏移现象和 神经元死亡会共同影响网络的收敛性。
- 优势：

1. 速度快：和sigmoid函数需要计算指数和倒数相比，relu函数其实就是一个max(0,x)，计算代价小很多。
2. 减轻梯度消失问题：relu函数在大于零的一侧其导数大于零，不会导致梯度变小。当然，激活函数仅仅是导致梯度减小的一个因素，但无论如何在这方面relu的表现强于sigmoid。使用relu激活函数可以训练更深的网络。

3.4 Softplus

- Softplus函数表达式为:

$$Softplus(x) = log(1 + e^x)$$



softplus可以看作是ReLu的平滑。

4. ML/MAP/BAYES

4.1 先验概率/后验概率/似然函数

考虑一个情景，小明要去公园，且有三种可选的交通方式，分别为自行车、步行和公交车，三种方式所需要的时间不同。

1. **后验概率（知果求因）**：如果小明去公园花费了2个小时，那么他很可能是走路过去的；如果小明去公园花费了半个小时，他很可能是骑车过去的。这种已知结果（花费的时间），然后根据结果推算原因（交通方式）的概率分布叫做**后验概率**。
   例子问题公式化：

   $$P(\text{交通方式}|\text{花费的时间})$$

   修改成一般的公式：

   $$P(\text{因}|\text{果})$$

   公式正规化：

   $$P(\theta|x)$$

   读作theta given x 的概率。
2. **先验概率（由历史求因）**：如果我们比较了解小明的喜好，例如是个健身爱好者，此时可以猜测他更可

能倾向于走路过去。这种在结果发生前就开始猜的，根据历史规律确定原因 （交通方式）的概率分布即**先验概率**。 例子问题公式化：

$$P(\text{交通方式})$$

修改成一般的公式：

$$P(\text{因})$$

正规化：

$$P(\theta)$$

3. **似然估计（由因求果）**：如果小明选择步行过去，那么一般情况下需要2个小时；很小的可能性是用了5分钟，这种根据原因来估计结果的概率分布即似然估计。 例子问题公式化：

$$P(\text{花费的时间}|\text{交通方式})$$

一般化：

$$P(\text{果}|\text{因})$$

正规化： 正规化：

$$P(x|\theta)$$

4. **Bayes 公式：**

$$P(\theta|x) = \frac{P(x|\theta) * P(\Theta)}{P(x)}$$

此处的$P(x)$即$evidence$小明去公园很多次，忽略交通方式是什么，只统计每次到达公园的时间 x，于是得到了一组时间的概率分布。这种不考虑原因，只看结果的概率分布即 evidence，它也称为样本发生的概率分布的证据。

$$\text{后验概率} = \frac{\text{似然函数} * \text{先验概率}}{evidence}$$

事实上evidence并不会影响分子概率分布的相对大小，因此：

$$Posteriori \propto Likelihood \times prior$$

5.贝叶斯变分推断(to be completed)

贝叶斯就是随着证据的增多逐渐改变你对一件事物的看法。

6. 全连接(to be completed)

n-1层的任意一个节点，都和第n层所有节点有连接。即第n层的每个节点在进行计算的时候，激活函数的输入是n-1层所有节点的加权。

7. Text Representation

7.1 词袋模型 (Bag of Words, BOW)

Bag-of-words model (BoW model) 最早出现在自然语言处理（Natural Language Processing）和信息检索（Information Retrieval）领域。该模型忽略掉文本的语法和语序等要素，将其仅仅看作是若干个词汇的集合，使用一组无序的单词(words)来表达一段文字或一个文档。 例如：

```
                    John likes to watch movies. Mary likes too.
                    John also likes to watch football games.
```

用BOW分别可以表示为：

```
       BoW1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
       BoW2 = {"John":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1
};
```

类似于python的dict,其中每个 key 代表一个单词， 每个 $value$ 代表该单词出现的频率，显然，词袋与每个单词出现的顺序无关。

- ***Cons.*** 有些常用的单词例如"the","a","to"出现的频率总会很高，因此单词的频率并不代表单词的重要性。为了解决这个问题，可以用文档频率的倒数(inverse of document frequency)或者 tf-idf。

2. tf-idf

tf-idf的主要思想是：如果某个词或短语在一篇文章中出现的频率(Term Frequency,TF)高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。 tf-idf是词频TF和逆文档频率(Inverse Document Frequency，IDF)的乘积:

$$if - idf = TF \times IDF$$

- 词频(Term Frequency，TF)计算: 最简单的方式是计算每个item在文档d中出现的次数出现的次数, $tf(t,d)$

Variants of term frequency (TF) weight

weighting scheme	TF weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

- 逆文档频率(Inverse Document Frequency，IDF)计算:

$$idf(t, D) = log \frac{\text{语料库文档总数}}{\text{包含该词的文档数} + 1}$$

Variants of inverse document frequency (IDF) weight

weighting scheme	IDF weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(1 + \frac{N}{n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

8. DBI（(Belief, Desire, Intention) 模型

BDI (Belief, Desire, Intention) 模型。起源于三篇经典论文：

- Cohen and Perrault 1979
- Perrault and Allen 1980
- Allen and Perrault 1980

1.信念（Belief）

信念是主体(agent)对世界的认知，包含描述环境特性的数据和描述自身功能的数据，是主体(agent)进行思维活动的基础。基于谓词 KNOW，如果 A 相信 P 为真，那么用 B(A, P) 来表示

2.愿望（Desire）

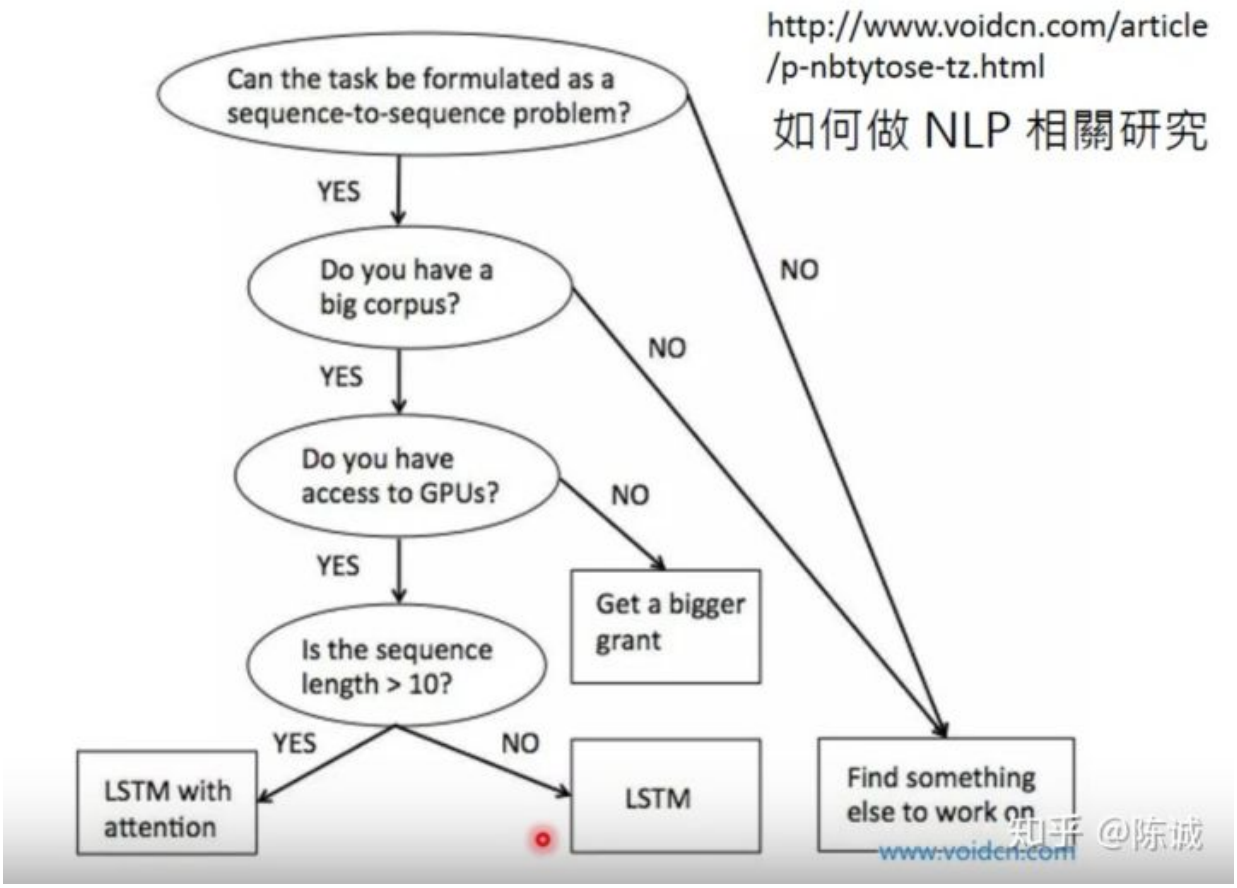
Agent希望达到的状态或者希望保持的状态，分别称作实现型愿望和维护型愿望。基于谓词 WANT，如果 S 希望 P 为真（S 想要实现 P），那么用 WANT(S, P) 来表示，P 可以是一些行为的状态或者实现，W(S, ACT(H)) 表示 S 想让 H 来做 ACT

3.意图（Intention）

是承诺实现的愿望中选取的当前最需要完成或者最适合完成的一个，是当前主体(agent)将要正在实现的目标，它是属于思维状态的意向方向。当前意图对主体(agent)的当前动作具有指导性的作用。

9.Sequence to Sequence 模型（to be completed）

sequence（sequence也有顺序的意思），指的是比如语音数据、文本数据、视频数据等一系列具有连续关系的数据,例如，机器翻译问题可以用RNN解决。



10.RNN/LSTM/GRU（to be completed）