周报

2018年6月4日

看过机器学习公开课之后，我发现人工智能领域对于数学知识的重视程度还是比较高的，尽管这点刚来大学的时候就听学长说过，但是怎么也没有自己接触到印象来的深刻。之前向别人了解过相关内容，知道主要的部分是一个算法，但是这个算法怎么实现，当时的确没有任何的思路，终于开始接触了，内心还是有点小兴奋的。在简单的接触之后我也有了些想法，注意到了以前没有注意到的点：

- 深度学习怎么解决现实生活中很普遍的偶然性问题。系统在抛出错误之后是有着同样的算法尝试去解决这个错误，还是直接终止程序。如果系统尝试去解决问题，会不会与我们人类的远景发生冲突，给我们带来巨大的危害，毕竟系统自己学习的过程中也是有着随机性存在的。
- 在线性代数中我已经接触到了向量，当时就有些好奇将其推广到无限维下的意义，看视频的时候发现了用无限维的坐标来表示一个数据的情况，自己还是有点小兴奋的。不过向量如何映射到内存这个困难的问题是要等到之后的视频才能解决了。
- 对于强化学习算法，计算机不是人，如何才能知道什么是正确的回报，这点是不是还要程序自己来用程序界定。
- 自己以前没有想到机器学习能够解决分离音频这么困难的问题。

另外，我也对计算机视觉的开始有了些自己的看法。在我看来，计算机视觉领域的确是人工智能领域里面重要的一环，我认为计算机和人进行交互，主要的方面有两个：视觉、听觉。现有的技术已经将听觉发展到了一个相当先进的程度，智能音像，手机的语音助手的功能现在已经相对完善并且逐渐大众化，但是计算机视觉和我们的生活的距离还是相对较远的，计算机视觉想要贴近生活还要考虑相对较大的硬件成本。

在识物识人方面，计算机视觉给我们带来了便利但是也同样带来了隐私安全问题，信息化时代下人变得越来越透明。

> 当我们在某个平台上上传我们的照片之后，现有的计算机识别技术就能够把照片中的人或物识别，平台依赖统计出的众多数据就可以绘制出一个人的社交网络。

这种言论当然不是危言耸听，在我们的生活中已经完全可以看到。尽管现在平台是凭借我们的文本信息来绘制我们的社交网络的，但是我相信发展到更高层面之后因为计算机视觉的加入，这个网络会绘制的更加清晰。

最近看到了有关扰乱人脸识别系统算法的介绍,经过其处理过的照片尽管人眼看起来没有区别，但是机器却无法做到识别，这套破坏性AI如何实现文中没有给出，但是它的研发的确能给人的隐私加上一层保险，不过它的公布，也同样促进了识别算法的升级，一个比较混乱的关系…

其他课程

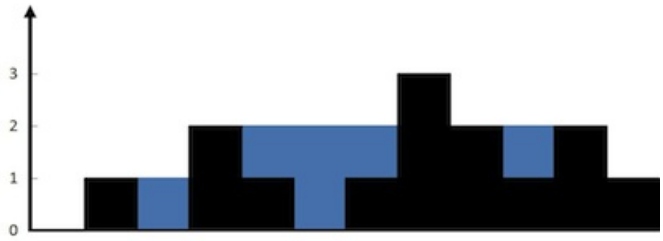
- 数据结构的课程我选择放一放，如果不出意外第四周就可以完全结束python的网课，因为前期的内容涉及python特性的方面很少，学起来比较轻松，可以预知后面课程的难度绝对会有增加。
- 我打算用atom来实现latex的书写，不过所需环境目前还在配置中。

---

leetcode题目

题目 42. Trapping Rain Water

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.



The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped. Thanks Marcos for contributing this image!

Example:

```
Input: [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
```

分析

我的想法是假设有一块木板，放到左边柱体上面，如果能够闭合(右边的高度大于等于左边)，意味着下面都有水。对应的，木板的右边所接触的位置就是下一次放木板开始的位置。事实证明，这个方法是错误的…因为有类似 `[4,1,3]` 这种情况出现。

不过想要解决也很简单，按照这个过程，反过来操作一次，尽管有些复杂，但能够解决这个问题。

```
class Solution {
public:
   int trap(vector<int>& height) {
        int len=height.size(),index=0,h;
        int sum=0;
        //正向遍历
        while(index<len-1){
          for(int i=index+1;i<len;i++){
            if(height[i]>=height[index]){
              h=height[index];
              for(int j=index+1;j<i;j++){
                sum+=h-height[j];
                height[j]=h;
              }
              index=i-1;
                break;
              }
          }
          index++;
        }
        //反向遍历
        while(index>0){
        for(int i=index-1;i>=0;i--){
          if(height[i]>=height[index]){
            h=height[index];
            for(int j=index-1;j>i;j--){
              sum+=h-height[j];
            }
            index=i+1;
            break;
          }
        }
        index--;
      }
      return sum;
    }
};
```

觉得上面的算法相对麻烦，查看解答之后有了更好的方法，类似双指针，左右两边同时操作，更新最大高度，同时不断填满空缺。代码也很简单易懂。

```
class Solution {
public:
    int trap(vector<int>& height) {
    //int len=height.size();
    int l=0,r=height.size()-1;
    int l_max=0,r_max=0,sum=0;
    while(l<r){
      if(height[l]<height[r]){    //注意比较的对象是当前指向的左右两个值
        height[l]>l_max    ? (l_max = height[l]) : (sum+=l_max-height[l]);
        l++;
      }
      else{
        height[r]>r_max ? (r_max = height[r]) : (sum += r_max-height[r]);
        r--;
      }
    }
    return sum;
    }
};
```

pdf是用atom直接导出的，效果可能不是很好