

Week3 - CNN & Self-Attention

2.1 CNN

图像的表达形式

一张输入的图像其实是一个三维的张量, **三个维度分别表示图像的宽, 高和Channel数目**, 彩色图像的每一个像素都是由RGB三个颜色所组成的, 所以3个Channel就分别代表了RGB三个颜色, 宽和高表示了这张图像中像素的数目. 将这个三维的张量拉直成一个向量, 每一维里存的数值表示某一个像素在某个颜色上的强度.

使用 `Fully Connected Network` 来实现目标, **随着网络层数增多, 参数也增多, 网络的弹性增强, 更容易出现过拟合问题, 训练的效率也会变低**. 因此 `Fully Connected Network` 并不适合直接拿来处理图像信息, 需要做进一步的简化.

识别图像的特征(Pattern)

用于图像分类的类神经网络是通过综合检测到的一些重要的 patterns 来进行分类, 因此没必要将一整张图像的每个神经元的输入. 只需要将一小部分输入到神经元就可以检测到某些重要的 patterns.

解决方法: 感受野 (Receptive Field)

一般在CNN中设置有感受野, 并由一组神经元来处理这个感受野提取的图像信息. 一般感受野是要考虑所有Channels上的信息的.

在描述感受野时, 只要说明宽和高(Kernel Size).

选取感受野:

1. 感受野可以有大有小
2. 可以指考虑一个Channel (对于CNN模型并不常用)
3. 感受野可以为长方形

通常的设置:

1. 通常考虑所有的 Channel (kernel size: 3x3)
2. 移动的值 `Stride` 通常设为1或2
3. 超出的范围做 `Padding` 处理(一般为补0)
4. 图像的每一个位置都应被 `neural` 照顾到

图像的Pattern出现在不同位置的处理方式

解决方法: 共享参数.

例如存在两个 `neural` ,

1. **neural 1**: $\sigma(w_1 x_1 + w_2 x_2 + \dots)$
2. **neural 2**: $\sigma(w_1 x'_1 + w_2 x'_2 + \dots)$

都采用相同的 $(w_1, w_2, \dots)$

通常的设置:

- 每个感受野有一组 `neural` 照顾到
- 公用的参数称之为Filter[1,2,3,...]

因此, 相较于 全连接网络, 感受野 和 参数共享 限制了网络的弹性. 感受野 和 参数共享 组成了 `Convolution Layer`, 拥有较大的 `Model bias`

2.2 Self-attention (自注意力机制)

输入可以是一个向量, 也可以是一组向量. 例如

- 处理一段文字时, 每个单词作为一组向量. 通常有 `One-hot encoding` 和 `Word embedding` 两种方法处理文字
- 处理一段音频时, 将一段音频分为若干个帧, 一个帧作为一组向量.
- 处理一个图(Graph)时, 一个Node作为一个向量

通常多向量输入的模型输出分为三种情况:

- 每个向量对应一个输出Label
- 只输出1个Label
- 自己决定输出多少个Label (Seq2Seq, 如翻译)

对于每个向量对应一个输出Label

假设存在 $a = (a_1, a_2, a_3, a_4)$ 四个输入, 采用 `Self-attention` 的步骤如下

1. 计算 a_i, a_j 的关联程度
 - `dot-product`: $q = a^i W^q, k = a^j W^k, \alpha = q \cdot k$
 - 通常对 α 做 `soft-max`, 得到 α'
2. 根据 α' 抽取特征
 - $v^i = W^v a^i$
 - $b^1 = \sum \alpha'_{1,i} v^i$

- α 称为 `attention score`
- q 称为 `query`
- k 称为 `key`

矩阵方式表达

令 $Q = (q_1, q_2, q_3, q_4)$, $I = (a_1, a_2, a_3, a_4)$, $V = (v_1, v_2, v_3, v_4)$, $O = (b_1, b_2, b_3, b_4)$

$$\begin{aligned}Q &= W^q I \\K &= W^k I \\V &= W^v I\end{aligned}$$

$$A = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,4} \\ \vdots & & \vdots \\ \alpha_{4,1} & \cdots & \alpha_{4,4} \end{bmatrix} = k^T Q \rightarrow^{softmax} A'$$

$$O = (v_1, v_2, v_3, v_4) \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,4} \\ \vdots & & \vdots \\ \alpha_{4,1} & \cdots & \alpha_{4,4} \end{bmatrix} = V A'$$

因此, W^q, W^k, w^v 为需要被学习的参数

Position Embedding

如果输入的各个向量对位置有所要求, 通常做 `Position Embedding`

1. 为每一个不同位置设置向量: e^i
2. $a^i \leftarrow e^i + a^i$