深度学习

Zebediah

2024 年 7 月 23 日

选自 Michael Nielsen 的 Neural Network and Deep Learning 第六章 6.1, 6.4, 6.5, 6.6 部分.

目录

1 卷积神经网络 (CNN)

我们之前使用了全连接的邻接关系的网络来完成识别手写数字的工作.

我们的输入层神经元为 28×28 像素的图像, 然后训练网络的权重和偏置, 使得网络输出正确地辨认输入图像. 然而, 这样的一个网络架构没有考虑图像的空间结构, 例如它完全平等地对待相距很远和彼此接近的输入像素. 因此我们设想: 如果我们使用一个设法利用空间结构的网络架构, 而不是从一个白板状态的网络架构开始会如何? 下面我们将介绍 CNN 这一网络.

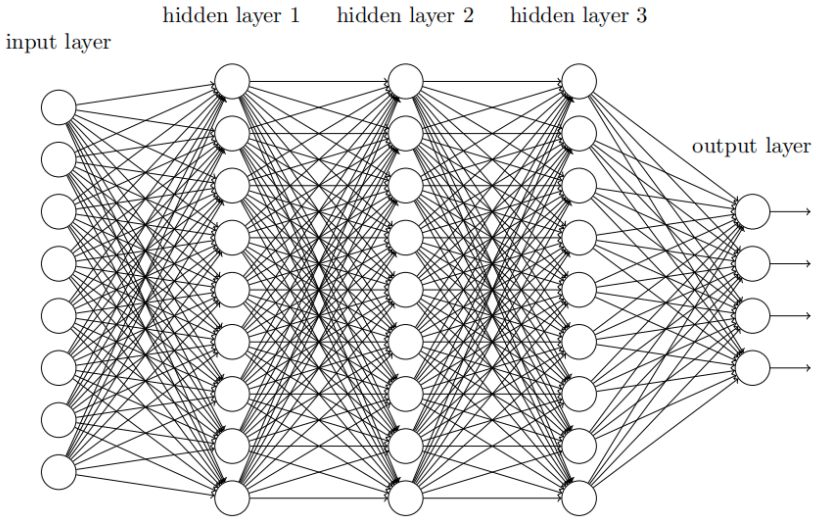


图 1: 全连接神经网络

CNN 采用了三种基本概念: 局部感受野 (local receptive fields), 共享权重 (shared weights), 和池化 (pooling).

1.1 局部感受野

在全连接层的网络中, 输入为纵向排列的神经元, 但在一个卷积网络中, 把输入看作是一个 28×28 的方形排列的神经元更有帮助. 和通常一样, 我们把输入像素连接到一个隐藏神经元层, 但是不会把每个输入像素连接到每个隐藏神经元, 相反只是把输入图像进行局部区域的连接. 确切地说, 第一个隐藏层中的每个神经元会连接到一个输入神经元的一个小区域, 例如一个 5×5 的区域, 对应于 25 个输入像素. 所以对于一个特定的隐藏神经元, 我们有这样的连接:

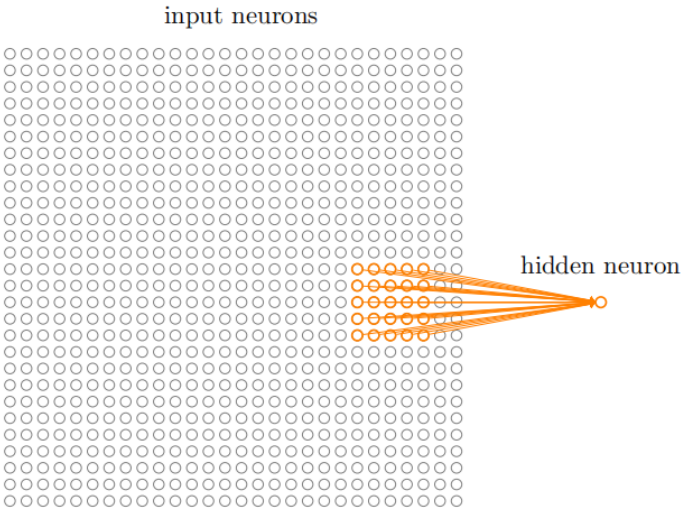
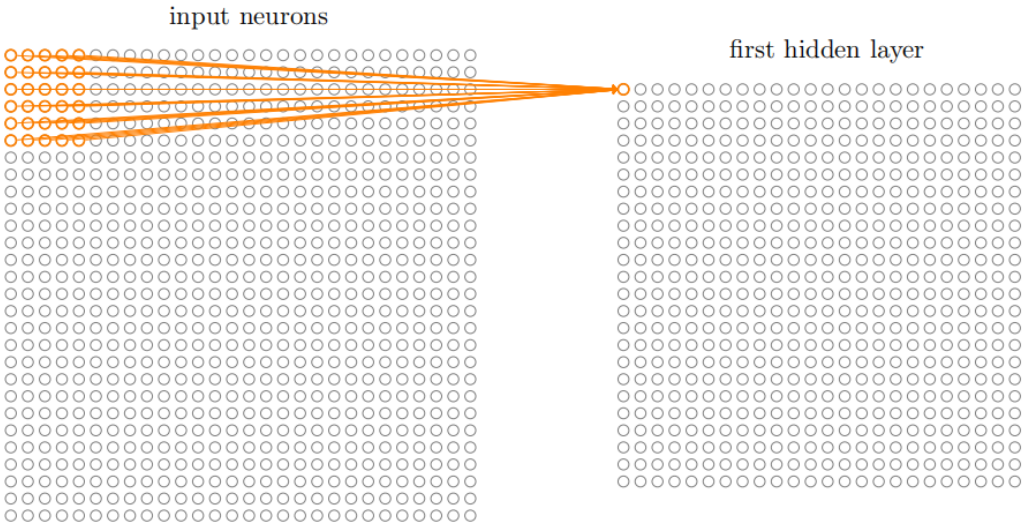
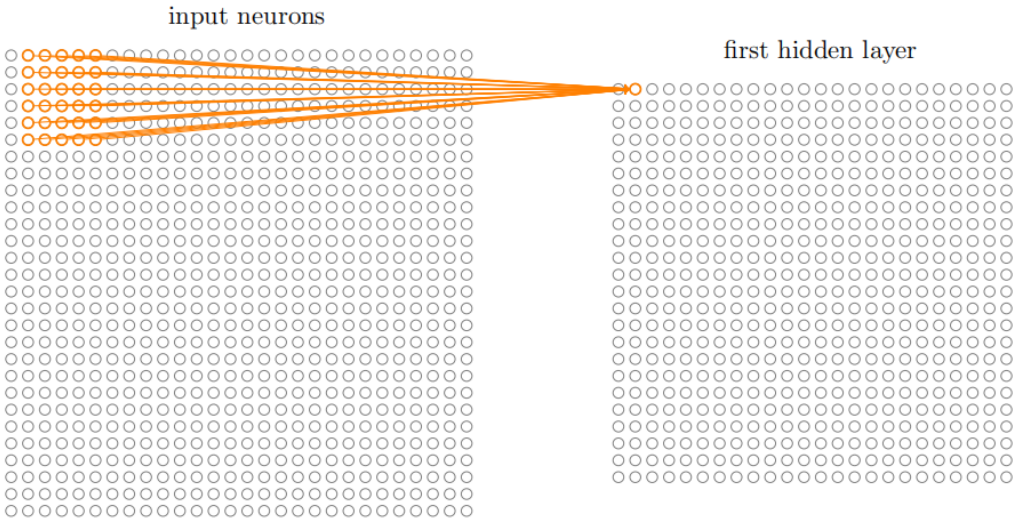


图 2: 一个隐藏神经元与输入层的连接

这个输入图像的区域被称为隐藏神经元的**局部感受野**. 它是输入像素上的一个小窗口, 每个连接学习一个权重, 而隐藏神经元同时也学习一个总的偏置. 然后我们在整个输入图像上交叉移动局部感受野, 从左上角开始:



然后往右一个像素 (即一个神经元) 移动局部感受野, 连接到第二个隐藏神经元:



如此重复, 就能构建起第一个隐藏层 (有 24×24 个神经元). 这里我们的局部感受野每次移动一个像素, 当然实际中, 有时候会使用不同的跨距.

1.2 共享权重和偏置

前面已经说过每个隐藏神经元具有一个偏置和连接到它的局部感受野的 5×5 权重, 但没有提及的是我们打算对 24×24 隐藏神经元中的每一个使用相同的权重和偏置. 换句话说, 对第 j, k 个隐藏神经元, 输出

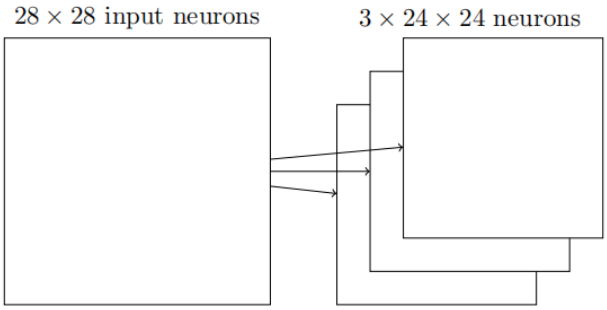
为:

$$\sigma\left(b+\sum_{l=0}^{4}\sum_{m=0}^{4}w_{l,m}a_{j+l,k+m}\right),$$

其中 σ 是激活函数, b 是共享偏置, $w_{l,m}$ 是共享权重的 5×5 数组, $a_{x,y}$ 表示位置为 x,y 的输入激活值.

我们把从输入层到隐藏层的映射称为一个**特征映射**, 定义特征映射的权重和称为**共享权重**和**共享偏置**. 共享权重和偏置经常被称为一个**卷积核**或者**滤波器**. 而共享权重和偏置的优点有很多, 如可以让参数减少、使得 CNN 具有平移不变性等.

目前我们所描述的网络结构只能检测一种局部特征的类型, 为了完成图像识别我们需要多个特征映射, 所以一个完整的**卷积层**由几个不同的特征映射组成, 下图展示了 3 个特征映射的例子:



下面我们使用具有 20 个特征映射的卷积层. 图 3 中的 20 幅图像对应于 20 个不同的特征映射 (或滤波器、核). 每个映射有一幅 5×5 块的图像表示, 对应于局部感受野中的 5×5 权重. 白色块意味着一个小权重, 更暗的块意味着一个更大的权重, 较小 (大) 权重的特征映射对相应的输　像素有更小 (大) 的响应.

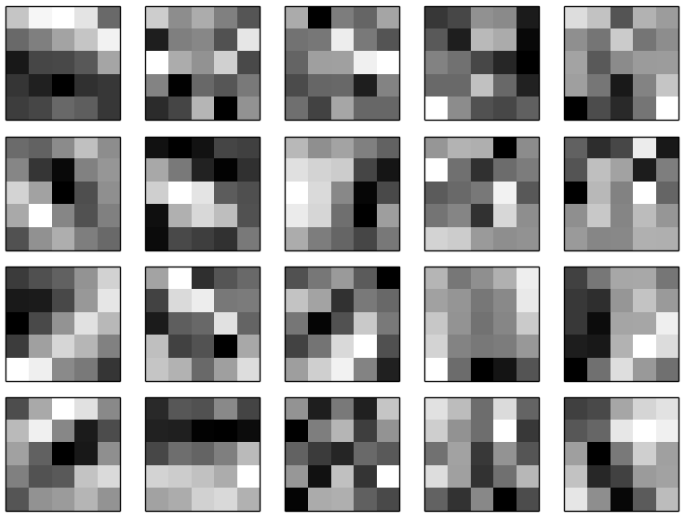
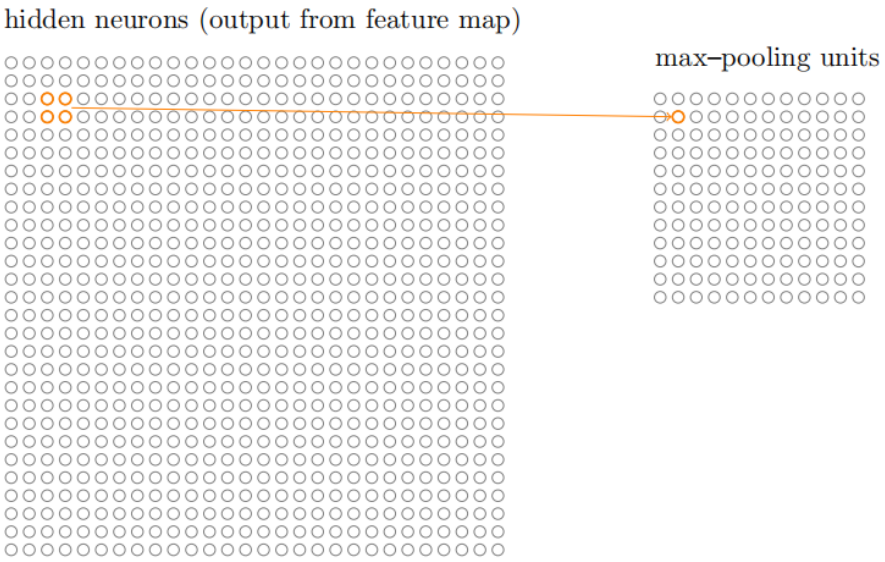


图 3: 20 个不同的特征映射

很明显这些特征有许多亮和暗的子区域, 这表示我们的网络正在学习和空间结构相关的东西, 然而我们很难看清这些特征检测器在学习什么. 实际上, 现在有许多关于通过 CNN 来更好理解特征的工作成果, 如 Matthew Zeiler, Rob Fergus 的 Visualizing and Understanding Convolutional Networks (2013).

1.3 池化层

池化层通常紧接着在卷积层之后使用, 它的作用是简化从卷积层输出的信息. 详细地说, 一个池化层从卷积层输出的每个特征映射中提取数据, 并将它们凝缩成一个新的特征映射. 例如, 池化层的每个单元可能概括了前一层的一个 2×2 的区域. 此外, 一个常见的池化操作是最大池化 (max-pooling). 在最大池化中, 一个池化单元简单地输出其 2×2 输入区域的最大激活值, 进而得到 12×12 个神经元:



另一个常用的方法是 L2 池化 (L2 pooling). 我们取 2×2 区域中激活值的平方和的平方根, 而非最大激活值. 虽然细节不同, 但其直观上和最大池化是相似的, 实际中我们可以对比不同方法并择优选择.

最后我们构建一个完整的 CNN 如图 4 所示. 注意, 网络中最后连接的层是一个全连接层, 这里一个箭头只是为了简化起见.

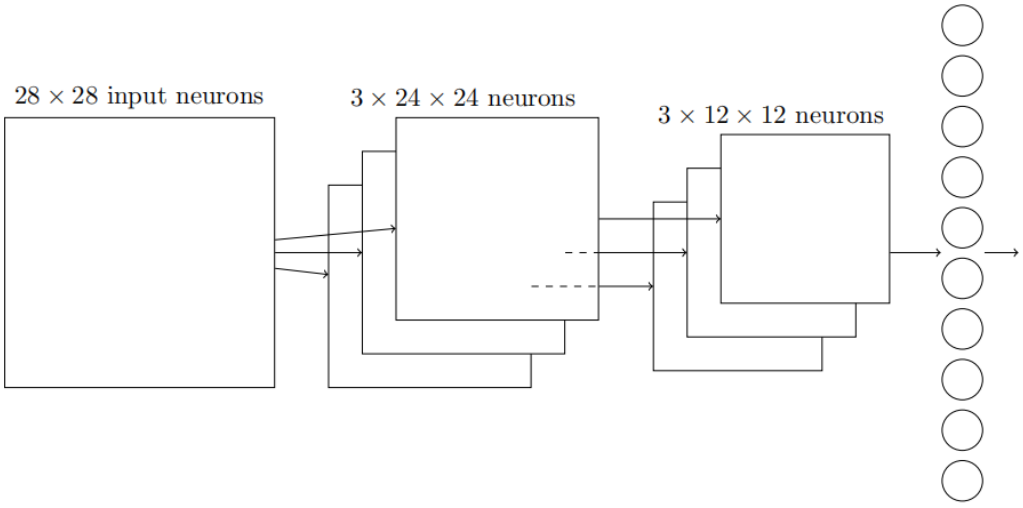


图 4: 一个完整的卷积神经网络

2 图像识别领域中的近期进展

2.1 2012 LRMD 论文

原文链接; 论文复述 1; 论文复述 2.

这是一篇关于特征学习的论文, 该文章说明了即使在没有监督的情况下训练神经网络, 高层次的神经元仍然能够对一些语义对象 (如人类的头部和脸、猫等) 做出强烈的反应.

文章主要考虑: 仅从未标记的数据构建高级的、特定类别的特征检测器. 稀疏编码可以在未标记的数据上训练产生感受野, 但稀疏编码结构比较浅, 只能获取 low-level 的特征. 文章构建了一个稀疏深度自编码网络, 包含三个特征: 局部感受野, 池化和局部对比度归一化. 深度自编码由三个相同的步骤组成: 局部滤波, 局部池化和局部对比度归一化, 共 9 层, 如图 5 所示.

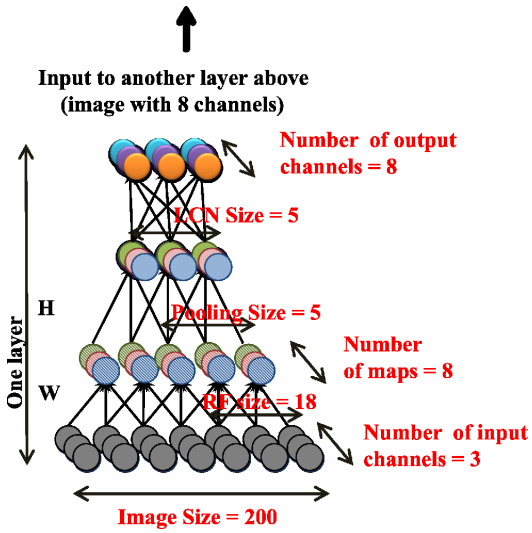


图 5: 深度自编码网络

网络的特点是节点的局部连接, 第一层的接受域为 18×18 像素, 且神经元连接到所有输入通道中的像素, 而第二层中的神经元只连接到一个通道. 当第一层输出线性滤波器响应时, 池化层输出其输入平方和的平方根, 因此称为 L2 池. 与 CNN 不同, 这里权值不共享. 该网络节点达到了庞大的 10 亿个, 但依然比人的视觉系统节点小 10^6 数量级.

学习: 在学习过程中, 将第二子层的参数固定为统一的权值, 第一层的编码权值 W_1 和解码权值 W_2 使用以下方法优化:

$$\underset{W_1,W_2}{\text{minimize}} \sum_{i=1}^{m} \left(\left\| W_2 W_1^T x^{(i)} - x^{(i)} \right\|_2^2 + \lambda \sum_{j=1}^{k} \sqrt{\epsilon + H_j \left(W_1^T x^{(i)} \right)^2} \right).$$

其中 λ 为稀疏性和重构之间的一个权衡参数; m, k 分别为样本数和池单元数; H_j 是第 j 个池单元的权值向量. 目标中的第一项确保表示对数据的重要信息进行编码, 即它们可以重建输入数据; 第二项鼓励池特征将相似的特征组合在一起以实现不变性.

最优化: 为了训练模型, 文章通过将局部权值 W_1、W_2 和 H 分配给不同的机器来实现模型的并行性.

研发组建立了一个软件框架 DistBelief, 为了加速训练, 系统使用了异步 SGD(Asynchronous SGD). 将训练分为 5 部分, 并在每个部分上运行模型的 copy. 模型通过一组集中的“参数服务器”进行更新, 这些服务器将模型的所有参数的当前状态保存在一组分区的服务器中. 简单的说, 在处理前, 每个 model replica 向参数服务器请求一份 mini-batch 模型参数的拷贝, 之后它计算 minibatch 的参数梯度, 之后将梯度发送至合适的参数服务器, P 步更新一次, G 步上传一次可以减少通信开销. 异步 SGD 比标准 SGD 对故障更具鲁棒性.

实验结果: 网络中表现最好的神经元在人脸检测中的准确率达到了 81.7%. 一个单层网络中表现最好的神经元仅能达到 71% 的准确率, 而从训练集中随机抽取的 100,000 个滤波器中选择的最佳线性滤波器仅能达到 74% 的准确率. 同时在从 ImageNet 识别 22,000 个对象类别时获得了 15.8% 的准确率, 比以前的最先进水平提高了 70%.

2.2 2012 KSH 论文

原文链接; 论文复述 1; 论文复述 2; 论文复述 3.

文章大致内容如下: (1) 训练了一个很大的深度卷积神经网络, 在 ImageNet 数据集上 ILSVRC-2012 比赛中达到远超第二名的最好效果; (2) **网络结构**; (3) **加速训练:** 卷积的 GPU 实现; (4) **降低过拟合:** dropout.

这篇论文追随了 LRMD 的成果. KSH 训练和测试一个深度卷积神经网络, 他们使用的数据集来自一个流行的机器学习竞赛————ImageNet Large-Scale Visual Recognition Challenge(ILSVRC). ILSVRC-2012 训练集包含有大约 1,200,000 幅 ImageNet 图像, 取自 1,000 个种类. 验证和测试集分别包含有 50,000 和 150,000 幅图像, 各自取自同样的 1,000 个种类.

ILSVRC 竞赛一个困难的地方是很多 ImageNet 图像包含了多个物体. 假设一幅显示一条拉布拉多犬追逐一个足球的图像. 这幅图像所谓“正确的”ImageNet 分类也许是一条拉布拉多犬. 如果一个算法把这个图像标记为一个足球, 它应该被扣分吗? 由于这种歧义性, 如果实际的 ImageNet 分类在一个算法认为最有可能的 5 个分类中, 那么这个算法就被认为是正确的. 在这个标准下, KSH 的深度卷积网络达到了一个 84.7% 的准确率, 大大好于次优的参赛者 (73.8%). 使用更严格的标准, KSH 的网络达到了 63.3% 的准确率.

KSH 使用一个深度卷积神经网络, 在两个 GPU 上训练, 网络示意图如图 6 所示. KSH 网络有 7 个隐藏神经元层. 前 5 个隐藏层是卷积层 (有些具有 Max Pooling), 而接下来的 2 层是全连接层. 输出层是一个 1,000 个单元的 softmax 层, 对应于那 1,000 个图像类别.

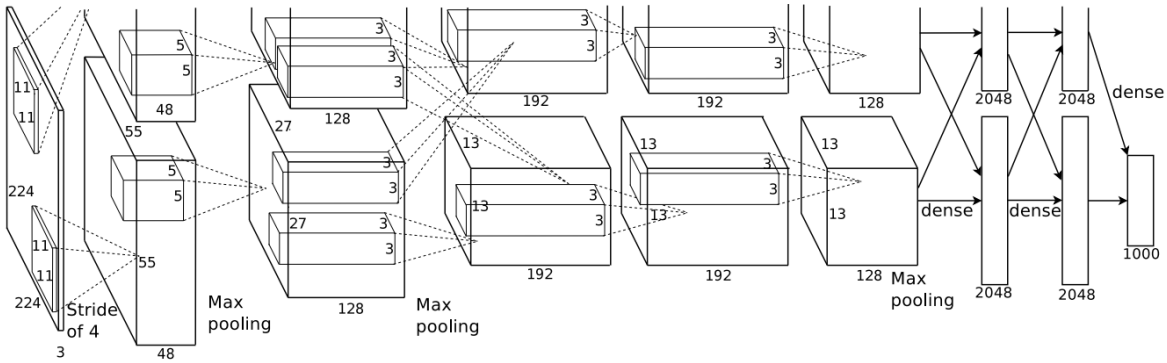


图 6: KSH 网络

输入层包含 $3 \times 224 \times 224$ 个神经元, 对应于一幅 224×224 图像的 RBG 值. 注意到 ImageNet 包含有不同分辨率的图像, 但一个神经网络的输入层通常是固定大小. 于是 KSH 缩放每幅图像使得长和宽中短的

长度为 256, 然后从缩放后的图像中裁剪出一个 256 × 256 的区域, 最后从 256 × 256 的图像中随机提取出 224 × 224 的子图像 (和水平反射). 他们把这个随机的裁剪用作扩展训练数据的方式, 可以减少过度拟合.

2.3 2014 ILSVRC 竞赛

和 2012 一样, 这次也包括了一个 120,000 张图像, 1,000 种类别, 而优值系数是前 5 个预测是否包含正确的分类。获胜团队使用了包含 22 层神经元的深度卷积网络 (GoogLeNet), GoogLeNet 达到了 93.33% 的前 5 准确率, 远超 2013 年的获胜者 (Clarifai，88.3%) 和 2012 年的获胜者 (KSH，84.7%). 那么 GoogLeNet 93.33% 的准确率又是多好呢? 事实上, 一个专家级别的人类, 非常细心地检查图像, 付出很大的努力才能够略微胜过深度神经网络.

2.4 其它结果

上面均关注的是 ImageNet, 但是也有一些其他的使用神经网络进行图像识别的工作.

- Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. 这篇论文应用深度卷积网络在识别 Google 的街景图像库中街景数字上. 他们对近 100,000,000 街景数字的自动检测和自动转述已经能与人类不相上下, 同时系统速度很快: 在一个小时内可将法国所有的街景数字全部转述. 该模型能够显著提高 Google Maps 在一些国家的地理精准度, 尤其是那些缺少地理编码的地区, 同时也解决了很多应用中字符短序列的 OCR 问题.
- Intriguing properties of neural networks. 该论文指出, 深度神经网络可能存在盲点问题, 即在某些情况下, 网络无法正确分类图像. 如图 7 所示, 左侧是被网络正确分类的 ImageNet 图像, 右边是一幅稍受干扰的图像 (使用中间的噪声进行干扰), 结果就没有能够正确分类. 作者发现对每幅图片都存在这样的“对手”图像, 而非少量的特例. 论文使用了 KSH 代码, 尽管这样的神经网络计算的函数在理论上都是连续的, 并且对这种不连续性出现的原因尚且未知. 幸运的是, 尽管对手图像会出现, 但是在实际场景中也不常见, 而这样的结果也可以催生出一系列的研究工作. 因此我们也不能盲目地说我们已经接近最终图像识别问题的答案了.

(a) (b)

图 7: ImageNet 图像示例

3 其他的深度学习模型

3.1 递归神经网络 (RNN)

RNN 是两大类人工神经网络之一 (另一类是 FNN). 与 FNN 不同, RNN 是一种双向人工神经网络, 可以体现出随时间动态变化的特性, 具有时间记忆性, 因此在处理时序数据和过程上效果很好. 这里仅简单介绍 RNN 的相关理论, 初步了解可以参考这篇笔记, 想深入学习可参考花书的第十章.

这里假设使用的激活函数为 tanh 函数, 其图像如图 8 所示.

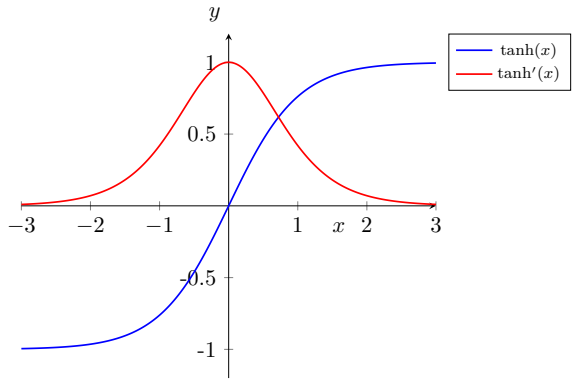


图 8: $\tanh(x)$ 及 $\tanh'(x)$ 图像

前向传播: 从 $t=1$ 到 $t=\tau$ 的每个时间步, 更新方程为:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \tag{3.1}$$

$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{a}^{(t)}\right), \tag{3.2}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \tag{3.3}$$

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{o}^{(t)}\right), \tag{3.4}$$

其中 $\boldsymbol{b}$ 和 $\boldsymbol{c}$ 分别对应于输入到隐藏、隐藏到输出的偏置向量, $\boldsymbol{U}, \boldsymbol{V}$ 和 $\boldsymbol{W}$, 分别对应于输入到隐藏、隐藏到输出和隐藏到隐藏的权重矩阵. 图 9 是一个示例 (当然也还有其他结构的 RNN).

沿时反向传播 (BPTT): 与 BP 算法中的反向传播是类似的, 利用链式法则可以求解出 L 的各个梯度, 然后直接利用梯度下降的理论进行更新即可, 这里不再赘述, 梯度的详细计算可参考花书的 10.2.2.

3.2 长短期记忆网络 (LSTM)

在标准的 RNN 中, 当进行 BPTT 时, 梯度会通过时间步骤向后传播. 如果序列很长, 梯度在每个时间步都会乘以权重矩阵. 这个过程会导致: 如果权重矩阵的特征值小于 (大于) 1, 反复相乘会导致梯度变小 (大), 从而学习表现不佳 (数值不稳定), 即梯度消失 (爆炸) 问题. 同时 RNN 在处理长序列时, 较早时间步的信息可能会在传递到后面的时间步时逐渐消失或被覆盖, 即短时记忆问题. 但 LSTM 可以解决这些问题, 下面 LSTM 和之后的 DBN 的内容主要参考邱锡鹏《神经网络与深度学习》. LSTM 主要改进在以下两个方面:

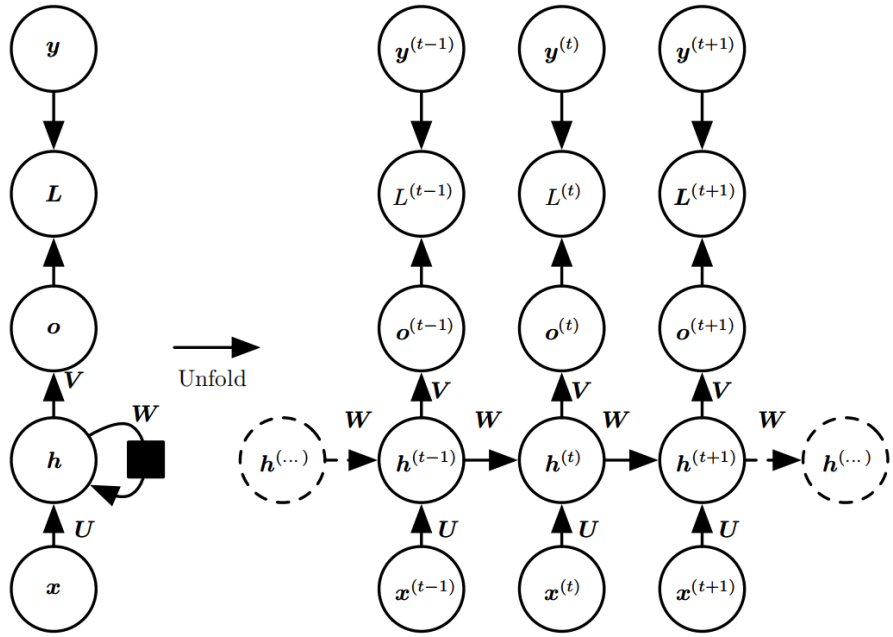


图 9: RNN 图示. $\boldsymbol{x}$ 为输入, $\boldsymbol{o}$ 为输出, L 为损失, $\boldsymbol{U}$ 为输入层到隐藏层的权重矩阵, $\boldsymbol{W}$ 为隐藏层到隐藏层的权重矩阵, $\boldsymbol{V}$ 为隐藏层到输出层的权重矩阵.

新的内部状态: LSTM 网络引入一个新的内部状态 (internal state) $c_t \in \mathbb{R}^D$ 专门进行线性的循环信息传递，同时 (非线性地) 输出信息给隐藏层的外部状态 $\boldsymbol{h}_t \in \mathbb{R}^D$. 内部状态 $\boldsymbol{c}_t$ 通过下面公式计算:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \tag{3.5}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right), \tag{3.6}$$

其中 $\boldsymbol{f}_t \in [0,1]^D \boldsymbol{i}_t \in [0,1]^D$ 和 $\boldsymbol{o}_t \in [0,1]^D$ 为三个门 (gate) 来控制信息传递的路径; $\odot$ 为 Hadamard 乘积; $\boldsymbol{c}_{t-1}$ 为上一时刻的记忆单元; $\tilde{\boldsymbol{c}}_t \in \mathbb{R}^D$ 是通过非线性函数得到的候选状态:

$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right). \tag{3.7}$$

其中 $\boldsymbol{W}_*, \boldsymbol{U}_*, \boldsymbol{b}_*, * \in \{\boldsymbol{i}, \boldsymbol{f}, \boldsymbol{o}, \boldsymbol{c}\}$ 为可学习的网络参数 (下同). 在每个时刻 t, LSTM 网络的内部状态 $\boldsymbol{c}_t$ 记录了到当前时刻为止的历史信息.

门控机制: 在数字电路中, 门为一个二值变量 $\{0,1\}$, 0 代表关闭状态, 不许任何信息通过; 1 代表开放状态, 允许所有信息通过.

LSTM 网络引入门控机来控制信息传递的路径. 公式 (3.5) 和 (3.6) 中三个“门”分别为输入门 $\boldsymbol{i}_t$, 遗忘门 $\boldsymbol{f}_t$ 和输出门 $\boldsymbol{o}_t$. 设 $\boldsymbol{x}_t$ 为当前时刻的输入, $\boldsymbol{h}_{t-1}$ 为上一时刻的外部状态, 则这三个门的作用为

- 遗忘门 $\boldsymbol{f}_t$ 控制上一个时刻的内部状态 $\boldsymbol{c}_{t-1}$ 需要遗忘多少信息:

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right). \tag{3.8}$$

- 输入门 $\boldsymbol{i}_t$ 控制当前时刻的候选状态 $\tilde{\boldsymbol{c}}_t$ 有多少信息需要保存:

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right). \tag{3.9}$$

- 输出门 $\boldsymbol{o}_t$ 控制当前时刻的内部状态 $\boldsymbol{c}_t$ 有多少信息需要输出给外部状态 $\boldsymbol{h}_t$:

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right). \tag{3.10}$$

LSTM 网络中的“门”是一种“软”门, 取值在 $(0, 1)$ 之间, 表示以一定的比例允许信息通过. 当 $\boldsymbol{f}_t = 0, \boldsymbol{i}_t = 1$ 时, 记忆单元将历史信息清空, 并将候选状态向量 $\tilde{\boldsymbol{c}}_t$ 写入. 但此时记忆单元 c_t 依然和上一时刻的历史信息相关. 当 $\boldsymbol{f}_t = 1, \boldsymbol{i}_t = 0$ 时, 记忆单元将复制上一时刻的内容, 不写入新的信息. 图 10 给出了 LSTM 网络的循环单元结构.

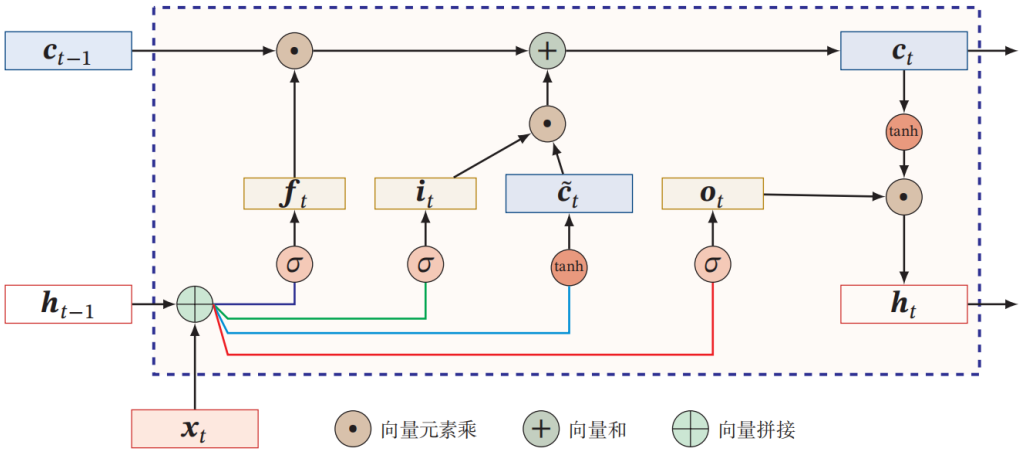


图 10: LSTM 网络的循环单元结构

前面所涉及到的公式可以简洁地描述为

$$\begin{bmatrix} \tilde{\boldsymbol{c}}_t \\ \boldsymbol{o}_t \\ \boldsymbol{i}_t \\ \boldsymbol{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\boldsymbol{W} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{bmatrix} + \boldsymbol{b} \right), \tag{3.11}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \tag{3.12}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right), \tag{3.13}$$

其中 $\boldsymbol{x}_t \in \mathbb{R}^M$ 为当前时刻的输入, $\boldsymbol{W} \in \mathbb{R}^{4D \times (M+D)}$ 和 $\boldsymbol{b} \in \mathbb{R}^{4D}$ 为网络参数.

3.3 深度信念网络, 生成模型和 Boltzmann 机

尽管深度信念网络 (DBN) 作为一种深度学习模型已经很少使用, 但其在深度学习发展进程中的贡献十分巨大, 并且其理论基础为概率图模型, 有非常好的解释性, 依然是一种值得深入研究的模型.

Boltzmann 机: Boltzmann 机是一个随机动力系统, 每个变量的状态都以一定的概率受到其他变量的影响. 玻尔兹曼机可以用概率无向图模型来描述. 一个具有 K 个节点 (变量) 的玻尔兹曼机满足三个性质: 每个随机变量是二值的; 所有节点之间是全连接的; 每两个变量之间的互相影响是对称的. 图 11 给出了一个包含 3 个可观测变量和 3 个隐变量的 Boltzmann 机.

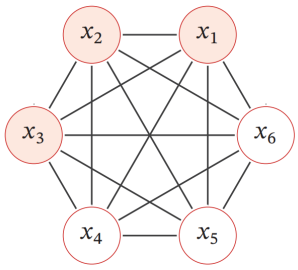


图 11: 一个有六个变量的 Boltzmann 机

生成模型: 生成模型是一种机器学习模型, 其主要任务是生成与训练数据相似的新数据. 与判别模型不同, 生成模型不仅能够进行分类, 还能够理解数据的分布, 并基于这种理解生成新的数据样本.

受限 Boltzmann 机 (RBM): RBM 是 DBN 的基本构建块, 是一种随机神经网络, 包含一个可见层和一个隐藏层, 两层之间的节点全连接, 但同一层内节点之间没有连接. RBM 通过调整权重和偏置来学习输入数据的概率分布. 图 12 给出了一个有 7 个变量的 RBM.

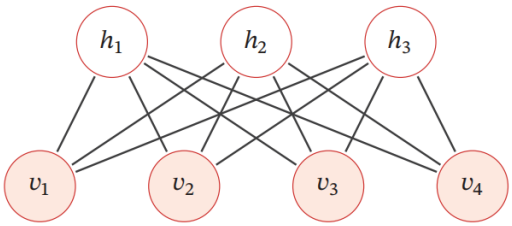


图 12: 一个有 7 个变量的 RBM

深度信念网络 (DBN): DBN 是一个生成模型, 它由多个 RBM 堆叠而成, 每一层的隐藏层作为下一层的可见层输入. 图 13 给出了一个有 4 层结构的 DBN. DBN 的训练过程可分为**逐层预训练**和**精调**两个阶段.

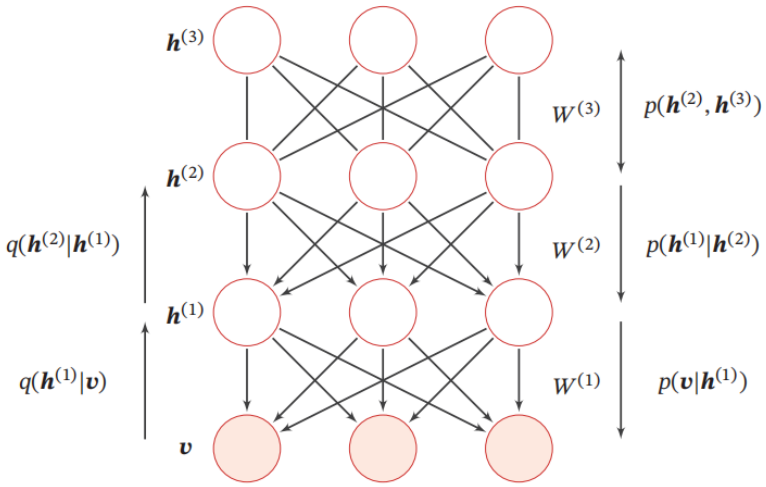


图 13: 一个有 4 层结构的 DBN

- **逐层预训练:** 先训练浅层网络, 再逐步增加网络的深度. 具体来说, 首先训练第一层 RBM, 第一层训练完毕后将其参数固定, 再将其隐藏层的输出作为下一层的输入, 依此类推.
- **精调:** 在预训练完成后, 首先将逐层预训练得到的参数作为整个网络的初始参数, 然后使用带标签的数据对整个网络进行训练, 最后通过传统的全局学习算法对网络进行精调. DBN 一般采用 Contrastive Wake-Sleep 算法.

3.4 其他想法

在神经网络和深度学习领域, 除了基础研究, 还有许多其他正在进行的热门研究和应用, 如自然语言处理、机器翻译和音乐信息学等. 一个特别有趣的研究是使用深度卷积网络和强化学习技术来玩电子游戏. 系统从原始像素数据中学习, 在不知道游戏规则的情况下, 却能在各种复杂场景中做出高质量决策.