0.1 机器学习方法

1. 监督学习：通过已有的训练样本（即已知数据以及其对应的输出）去训练得到一个最优模型（这个模型属于某个函数的集合，最优则表示在某个评价准则下是最佳的），再利用这个模型将所有的输入映射为相应的输出。

2. 无监督学习：它与监督学习的不同之处，在于我们事先没有任何训练样本，而需要直接对数据进行建模。

3. 半监督学习：在训练阶段结合了大量未标记的数据和少量标签数据。与使用所有标签数据的模型相比，使用训练集的训练模型在训练时可以更为准确。

4. 强化学习：我们设定一个回报函数（reward function），通过这个函数来确认否越来越接近目标，类似我们训练宠物，如果做对了就给他奖励，做错了就给予惩罚，最后来达到我们的训练目的。

线性回归：对于输入 x 与输出 y 有一个映射 f，$y = f(x)$, 而 f 的形式为 $f(x) = ax + b$。

0.2 损失函数

1. nn.L1Loss: 输入 x 和目标 y 之间差的绝对值，要求 x 和 y 的维度要一样（可以是向量或者矩阵），得到的 loss 维度也是对应一样的 $loss(x, y) = 1/n \sum |x_i - y_i|$

2. nn.NLLLoss: 用于多分类的负对数似然损失函数 $loss(x, class) = -x[class]$ NLLLoss 中如果传递了 weights 参数，会对损失进行加权，公式就变成了 $loss(x, class) = -weights[class] * x[class]$

3. nn.MSELoss: 均方损失函数，输入 x 和目标 y 之间均方差 $loss(x, y) = 1/n \sum (x_i - y_i)^2$

4. nn.CrossEntropyLoss: 因为使用了 NLLLoss，所以也可以传入 weight 参数，这时 loss 的计算公式变为：$loss(x, class) = weights[class] * (-x[class] + log(\sum_j exp(x[j])))$ 所以一般多分类的情况会使用这个损失函数

5. nn.BCELoss: 计算 x 与 y 之间的二进制交叉熵。$loss(o, t) = -\frac{1}{n} \sum_i (t[i] log(o[i]) + (1 - t[i]) log(1 - o[i]))$ 与 NLLLoss 类似，也可以添加权重参数：$loss(o, t) = -\frac{1}{n} \sum_i weights[i](t[i] log(o[i]) + (1 - t[i]) * log(1 - o[i]))$ 用的时候需要在该层前面加上 Sigmoid 函数。

   欠拟合：- 增加网络结构，如增加隐藏层数目；- 训练更长时间；- 寻找合适的网络架构，使用更大的 NN 结构；

   过拟合：- 使用更多的数据；- 正则化（regularization）；- 寻找合适的网络结构；

但是 sigmod 由于需要进行指数运算（这个对于计算机来说是比较慢，相比 relu），再加上函数输出不是以 0 为中心的（这样会使权重更新效率降低），当输入稍微远离了坐标原点，函数的梯度就变得很小了（几乎为零）。在神经网络反向传播的过程中不利于权重的优化，这个问题叫做梯度饱和，也可以叫梯度弥散。这些不足，所以现在使用到 sigmod 基本很少了，基本上只有在做二元分类（0，1）时的输出层才会使用。使用 dropout 技巧在训练期间有选择性地忽略单个神经元，来减缓模型的过拟合；- 重叠最大池，避免平均池的平均效果；

0.3 经典模型

0.3.1 CNN

1. LeNet-5
2. AlexNet
3. VGG
4. GoogLeNet (Inception)
5. ResNet

0.3.2 RNN

1. LSTM
2. GRU
3. 循环网络的向后传播（BPTT）
4. 词嵌入（word embedding）
5. Beam Search（集束搜索）
6. 注意力模型

退化问题: 网络层数增加，但是在训练集上的准确率却饱和甚至下降了。这个不能解释为 overfitting，因为 overfit 应该表现为在训练集上表现更好才对。这个就是网络退化的问题，退化问题说明了深度网络不能很简单地被很好地优化 logistic 回归主要是进行二分类预测，我们在激活函数时候讲到过 Sigmod 函数，Sigmod 函数是最常见的 logistic 函数，因为 Sigmod 函数的输出的是是对于 0 1 之间的概率值，当概率大于 0.5 预测为 1，小于 0.5 预测为 0。迁移学习初衷是节省人工标注样本的时间，让模型可以通过一个已有的标记数据的领域向未标记数据领域进行迁移从而训练出适用于该领域的模型，直接对目标域从头开始学习成本太高，我们故而转向运用已有的相关知识来辅助尽快地学习新知识

针对于某个任务，自己的训练数据不多，那怎么办？没关系，我们先找到一个同类的别人训练好的模型，把别人现成的训练好了的模型拿过来，换成自己的数据，调整一下参数，再训练一遍，这就是微调（fine-tune）。

于不同的领域微调的方法也不一样，比如语音识别领域一般微调前几层，图片识别问题微调后面几层