深度学习

深度学习的历史发展

- [ ] 考什么?

前馈神经网络

- 激活函数
  - sigmoid
  - tanh
  - relu

  特殊的激活的函数：softmax 一般用于多酚类

- 各个神经元接受前一级的输入，并输出到下一级，模型中没有反馈
- 层与层之间通过"fc"连接，层内神经元不相互连接

损失函数

- 均方误差损失函数 mse
- 交叉熵损失函数

$$\mathrm{H}(p,q) = -\sum_{x} p(x) * logq(x)$$

$$\text{熵} \quad H(X) = -\sum_{i=1}^{n} p(x_i) log(p(x_i))$$

$$\text{相对熵（KL散度）} \quad D_{KL}(p||q) = \sum_{i=1}^{n} p(x_i) log(\frac{p(x_i)}{q(x_i)})$$

> p是实际值，q是预测值

刻画两个概率分布之间的距离，交叉熵越小，两个概率分布p与q越接近

- [ ] 熵 + 相对熵 = 交叉熵

交叉熵与softmax函数结合，叫做softmax损失

损失函数

$$\text{softmax} \quad L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$$

$$\text{SVM} \quad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\text{Full loss} \quad L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)$$

R(W)是regularization term的loss

梯度下降：

- 梯度的反方向是函数值下降最快的方向，因此是损失函数求解的方向

目的是最小化损失函数，即$(\nabla f(x))^T \Delta x < 0$，增量x为正，只需要控制其方向恰和梯度方向相反成180°，就可以保证损失误差下降最多，下降最快

只要沿着损失函数梯度的反方向来更新参数，就可使得损失函数下降最快

$w^{new} = w - \eta \times \frac{dL}{dw}$

梯度下降算法

- BGD：批量梯度下降
- SGD：随机梯度下降
- MBGD：小批量梯度下降

> BGD:所有样本
>
> sgd:每次一个样本

batch_size增大的好处：

- 内存利用率变高，大矩阵乘法的并行化效率提高
- 跑完一次epoch（全数据集）所需迭代次数减少，对于相同数据量的处理速度进一步加快
- 在一定范围内，一般来说batch_size越大，其确定的下降方向越准，引起训练震荡越小

batch_size盲目增大的坏处：

- colab 内存不够用
- 跑完一次epoch（全数据集）所需的迭代次数减少，要想达到相同的精度，其所花费的时间大大增加，从而对参数的修正也就显得更加缓慢
- 增大到一定程度，确定的下降方向已经基本不再变化

CNN

卷积操作

如果卷积核**中心位置的权重系数越小**且**与其他卷积权重系数差别越小**，则卷积所得到图像滤波结果越模糊，这被称为图像平滑操作

每个输出点的取值仅依赖于其在输入图像中该点及其邻域区域点的取值，与这个区域之外的其他点取值均无关，该区域被称为感受野(receptive field)

池化操作

常用操作：

- 最大池化
- 平均池化

池化层用来大幅降低参数量级（降维）

- [ ] 计算公式

> padding = kernel / 2　向下取整

神经网络正则化

缓解过拟合，提高泛化能力

- dropout
- batch-normalization

  随着神经网络深度的增加，输入数据经过激活函数若干次非线性变换后，其整体分布逐渐向非线性函数的值域上下限两端发生偏移。例如，对于sigmoid函数，它所接受的输入往往是数值较大的正数或者负数，经过sigmoid非线性变换后，结果大多偏向于1或者-1。在1或者-1附近，sigmoid函数的梯度较小，导致在反向传播时，靠近输入端容易出现梯度消失问题，导致网络收敛很慢。

  批归一化就是通过规范化的手段，把神经网络每层中任意神经元的输入值分布改变到均值为0、方差为1的标准正态分布。
- L1-Norm & L2-norm
  - L1范数：模型参数W中各个元素的绝对值之和 $||W||_1 = \sum_{i=1}^{N} |w_i|$
  - L2范数：模型参数W中各个元素平方和的开方 $||W||_2 = \sqrt{\sum_{i=1}^{N} w_i^2}$

RNN

一种对h_t的隐式编码：

$$h_t = \Phi(\mathrm{U} \times x_t + \mathrm{W} \times h_{t-1})$$

为解决梯度消失问题：LSTM

LSTM

- input gate
- forget gate
- output gate
- cell state: 用到了input gate和forget gate，前者控制有多少信息流入当前时刻内部记忆单元c_t，后者控制上一时刻内部记忆单元c_{t-1}有多少信息可累积到当前时刻内部记忆单元c_t
- h_t: $h_t = o_t \odot tanh(c_t)$

三种门结构的输出i_t, f_t, o_t值域为(0, 1)，三个门结构所输出向量的维数、内部记忆单元的维数和隐式编码的维数均相等

克服梯度消失：

c_t对c_{t-1}求导

如果遗忘门选择保留旧状态，则这一求导结果就接近1，使得梯度存在

c_t：长时记忆

h_t：短时记忆

> forget趋向于0，过去状态对新的cell state的影响变小

GRU不再使用记忆单元来传递信息，仅使用hidden state进行信息传递

- [ ] GRU和LSTM的结构

深度学习的应用

词袋模型：维度灾难

Word2Vec

V个不同的单词，如何生成第k个单词的N维词向量

- 首先，把词表示为V维one hot向量X，第k个单词就只在第k个位置取值为1
- 该X(V * 1) 经过一个FC层转为h $h = W^T \times X$ W(V * N) 这个h就是单词的N维隐式表达
- h(N * 1) 再经过一个FC层转为输出层结果 $y = (W')^T \times h$ W'(N * V)

  y = [y1, y2, ..., yk, ..., yv] 将y进行softmax归一化，得到第k个单词对应的归一化概率输出，显然，第k个单词归一化的概率值应该远远大于输出层中其他位置所对应的归一化概率值

- [ ] 什么作为N维词向量？第一层fc中参数W的某一行

$$\text{word2vec model} = max\text{P}(\text{输出表达}|\text{输入表达})$$
$$= max \log\left(\frac{e^{y_k^*}}{\sum_{j=1}^{V} e^{y_j}}\right) = max(y_k^* - log \sum_{j=1}^{V} e^{y_j})$$

损失函数：

$$loss = -y_k^* + log \sum_{j=1}^{V} e^{y_j}$$

CBOW模型

给出包含c个单词的句子，可用x_k的前序k-1个单词和后续单词c-k个单词一起来预测xk。每个输入单词仍然是V维向量（可以是one-hot的表达），隐式编码为每个输入单词所对应编码的均值
$h = \frac{1}{c-1} W^T \times (X_1 + \ldots + X_{k-1} + X_{k+1} + \ldots + X_c)$

两个加快模型训练的方法：

- 层次化softmax
- 负采样

> 利用单词预测上下文：skip-gram

cnn for classification + localization

treat localization as a regression problem

分类的softmax loss + 预测box coordinates坐标的L2 loss