手写字符识别

赵耀

手写字符识别

- 属于机器学习中的回归任务，学习算法需要输出一个函数，该函数能对给定的输入预测数值。
- 属于监督学习。样本（图片）包含一组特征（像素点）和一个标签（图片对应的数字）

知识回顾

Key Questions for Learning

- In AI, a learning process is implemented by algorithms and programs.
- Given some observations (data) from the environment, and (possibly) some prior knowledge, how could an agent improve its agent function?
  - What is the format of the data? (data representation)
  - What is the prior knowledge?
  - What does the agent function look like? (model representation)
  - How to measure the "improvement"? (objective function)
  - What is the learning algorithm?

Search in a model/hypothesis space

Representation + Algorithm + Evaluation

What is the format of the data ?

- 此次实验的数据集是手写数字图片数据集 MNIST 的一个子集
- 每个样本都是一张 28x28 像素的黑白图，图片上是一个手写数字，同时还有对应的标签表明是数字几

- 为了方便处理，我们可以将一张图片处理成1*784的矩阵，矩阵的每个元素为该图片的每个像素点的值。每个像素点的值处理为一个特征，一共有784个特征
- 图片库中的图片，其像素点范围为[0,255]

What is the prior knowledge

- Not required

What does the agent function look like?

- Model representation：
- 本实验练习的是人工神经网络中应用的比较广泛也比较简单的一种，叫做BP神经网络。其网络结构为4层全连接神经网络。
- 一个model的核心：1、其数学表达形式；2、如何训练模型，使之能相对准确的根据输入预测一个结果。

How to measure the improvement

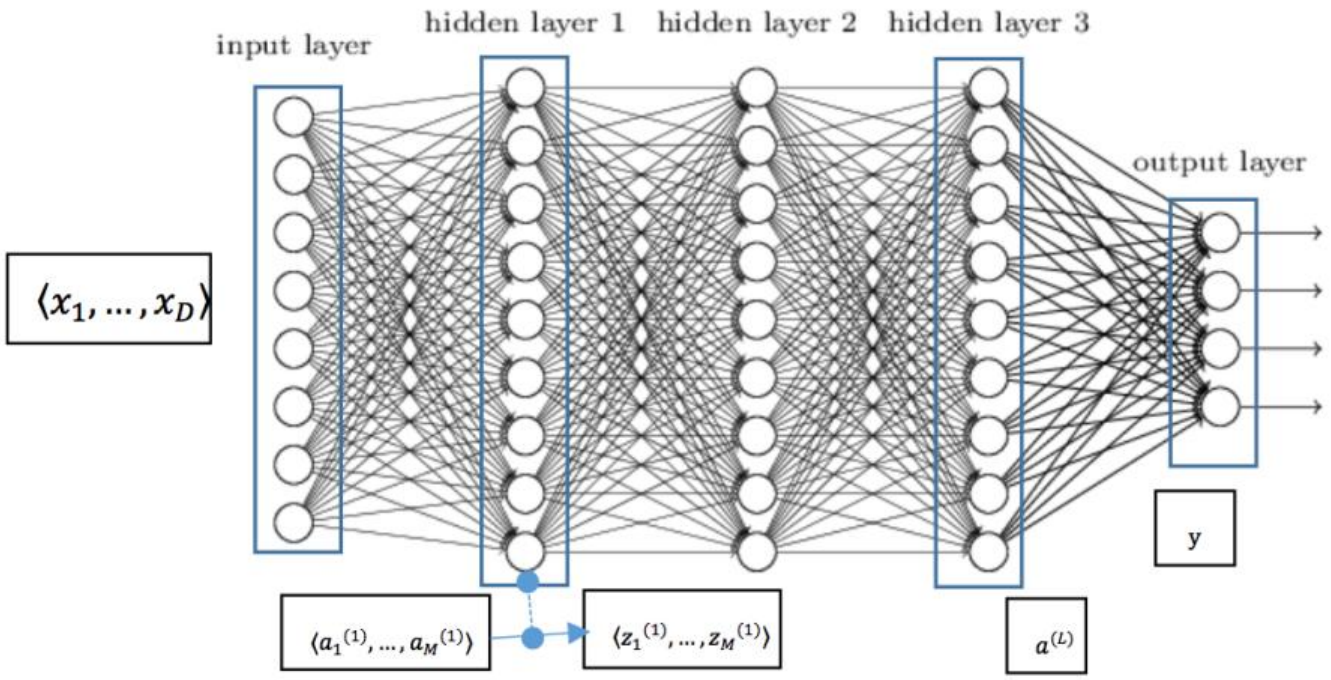
- 精确度: 正确识别的图像数目 / 总测试图像数目

模型的选择

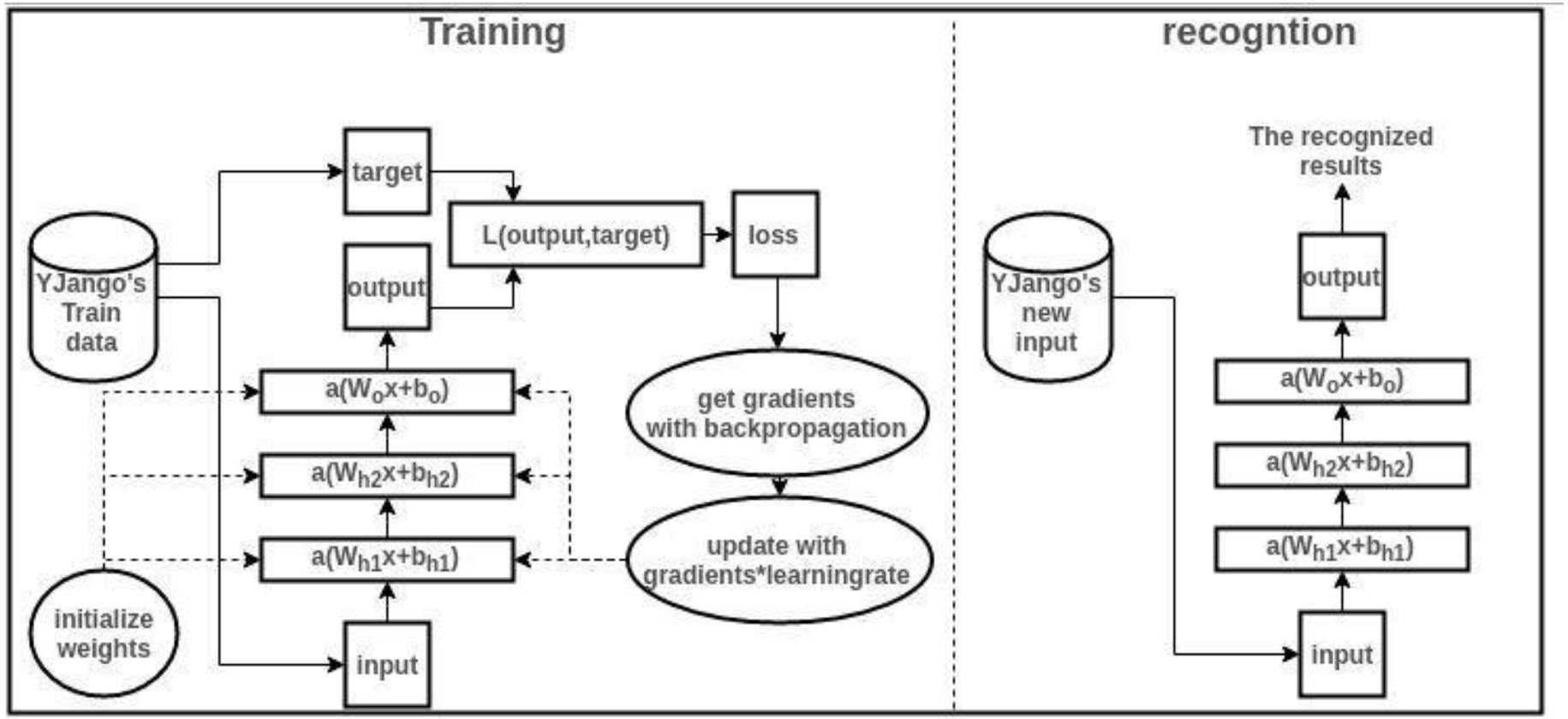
- 为什么选择BP神经网络？
- BP神经网络具有较强的非线性映射能力，其实质上实现了一个从输入到输出的映射功能，从数学上证明了三层神经网络就能以任意精度逼近任何非线性连续函数。
- 具有自学习和自适应能力：在训练时，BP算法能够通过学习提取数据间的规则，并将自适应的将学习内容记忆于网络的权值中。
- 泛化能力：泛化能力是指在设计模式分类器时，即要考虑网络在保证对所需分类对象进行正确分类，还要关心网络在经过训练后，能否对未见过的模式或有噪声污染的模式，进行正确的分类
- 容错能力：BP神经网络在其局部的或者部分的神经元受到破坏后对全局的训练结果不会造成很大的影响，也就是说即使系统在受到局部损伤时还是可以正常工作的。
- 为什么选择双隐层？
- 理论上，单隐层神经网络可以逼近任何连续函数（只要隐层的神经元个数足够）
- 虽然从数学上看多隐层和单隐层表达能力一致，但多隐层的神经网络比单隐层神经网络工程效果好很多
- 对于一些分类数据，层数加的太多，对最后的结果的帮助没有那么大的跳变

模型的表达及如何训练

- 通常一个神经网络由一个input layer，多个hidden layer和一个output layer构成。
- 图中圆圈可以视为一个神经元（又可以称为感知器）
- 设计神经网络的重要工作是设计hidden layer，及神经元之间的权重
- 添加少量隐层获得浅层神经网络SNN；隐层很多时就是深层神经网络DNN
- 四层神经网络，包含输入层，2个隐藏层和输出层



训练的流程图


Training
recogntion
target
L(output,target)
loss
YJango's Train data
output
$a(W_o x+b_o)$
$a(W_{h2} x+b_{h2})$
$a(W_{h1} x+b_{h1})$
get gradients with backpropagation
update with gradients*learningrate
initialize weights
input
The recognized results
YJango's new input
output
$a(W_o x+b_o)$
$a(W_{h2} x+b_{h2})$
$a(W_{h1} x+b_{h1})$
input

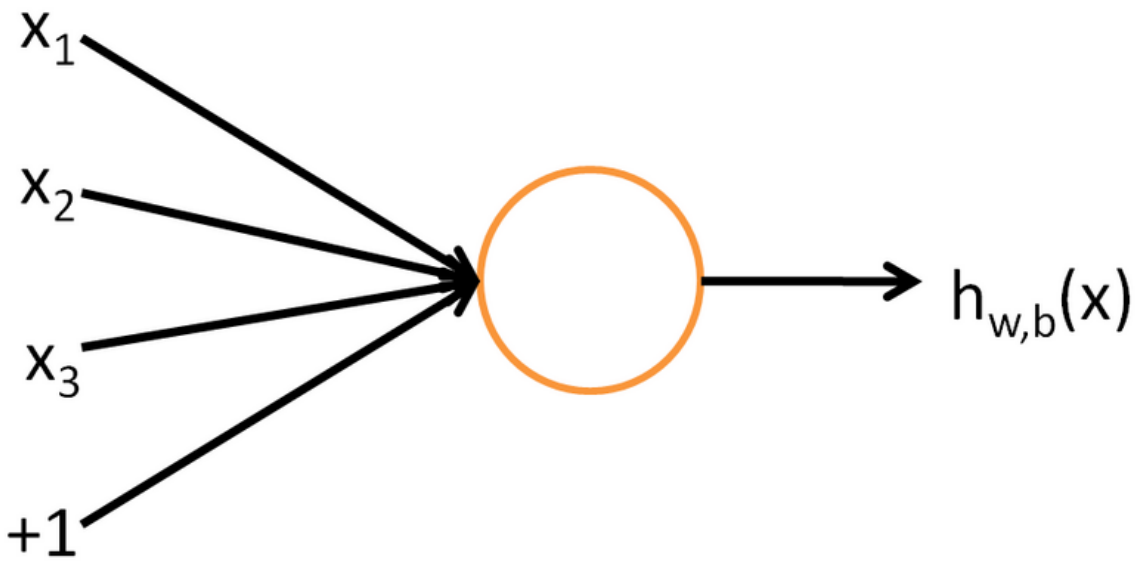
一个人工神经元的数学模型

这个神经元是一个以X_1, X_2, X_3以及截距+1为输入的运算单元，其输出为$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$

其中，f为激活函数。

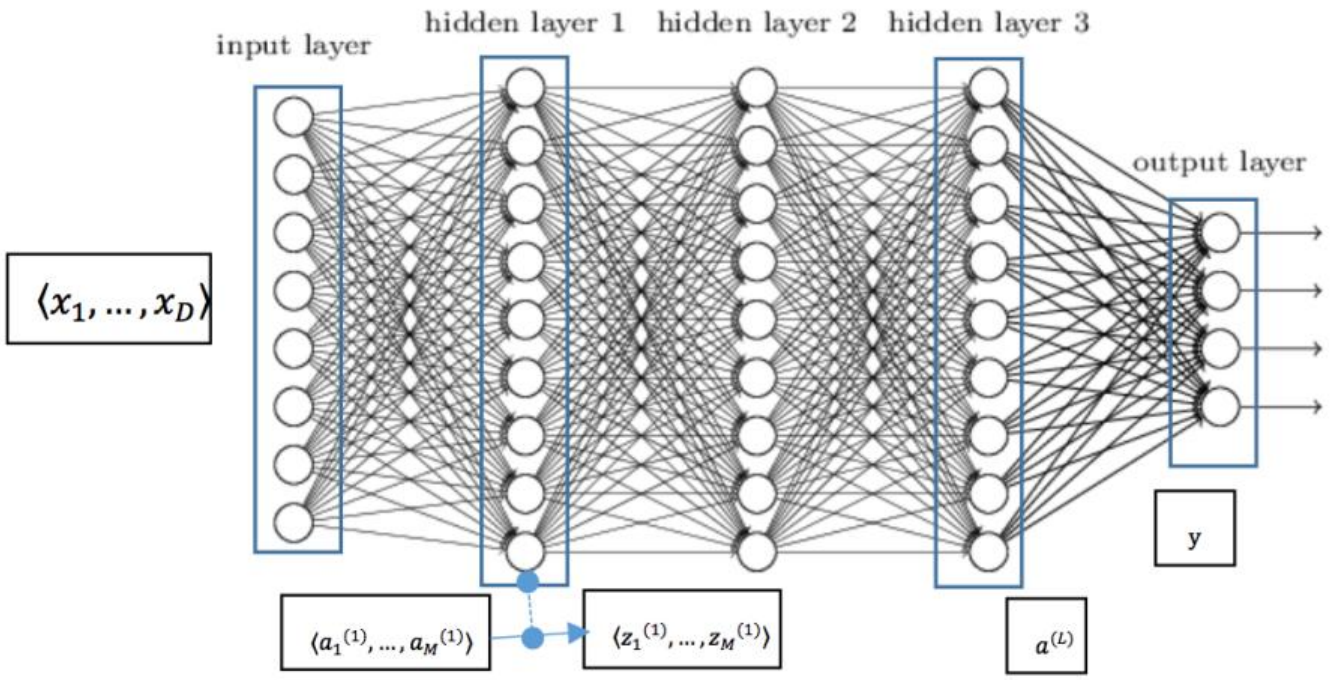
上述这个单一的神经元可以看出是一个逻辑回归

常用的激活函数有sigmoid、relu等



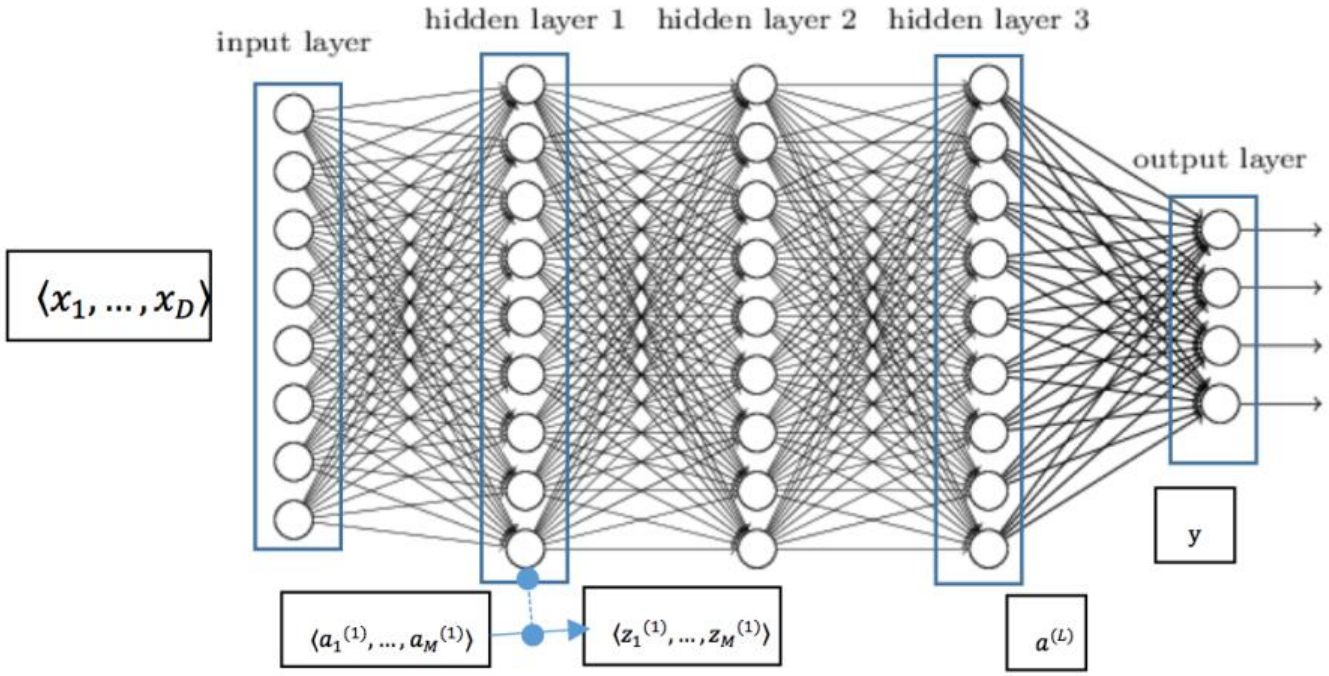
神经网络模型-输入层到隐层1

- 输入层：假设我们每次进行训练的样本数为100，每个样本的特征为784，则输入为100*784
- 输入层经过线性变换成为隐藏层1的输入：隐藏层1的输入$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j$,此处D为样本的特征数目784，j为隐藏层1的神经元结点数目，假设设置隐藏层1的结点数目M=300，则W权重矩阵为784*300，B偏置矩阵为100*300，最终得到的$a^{(1)}$为100*300的矩阵。
- 隐藏层1对$a^{(1)}$通过激活函数f得到激活值$z^{(1)}$: $z_j^{(1)} = f(a_j^{(1)})$，最终得到的$z^{(1)}$也为100*300的矩阵。



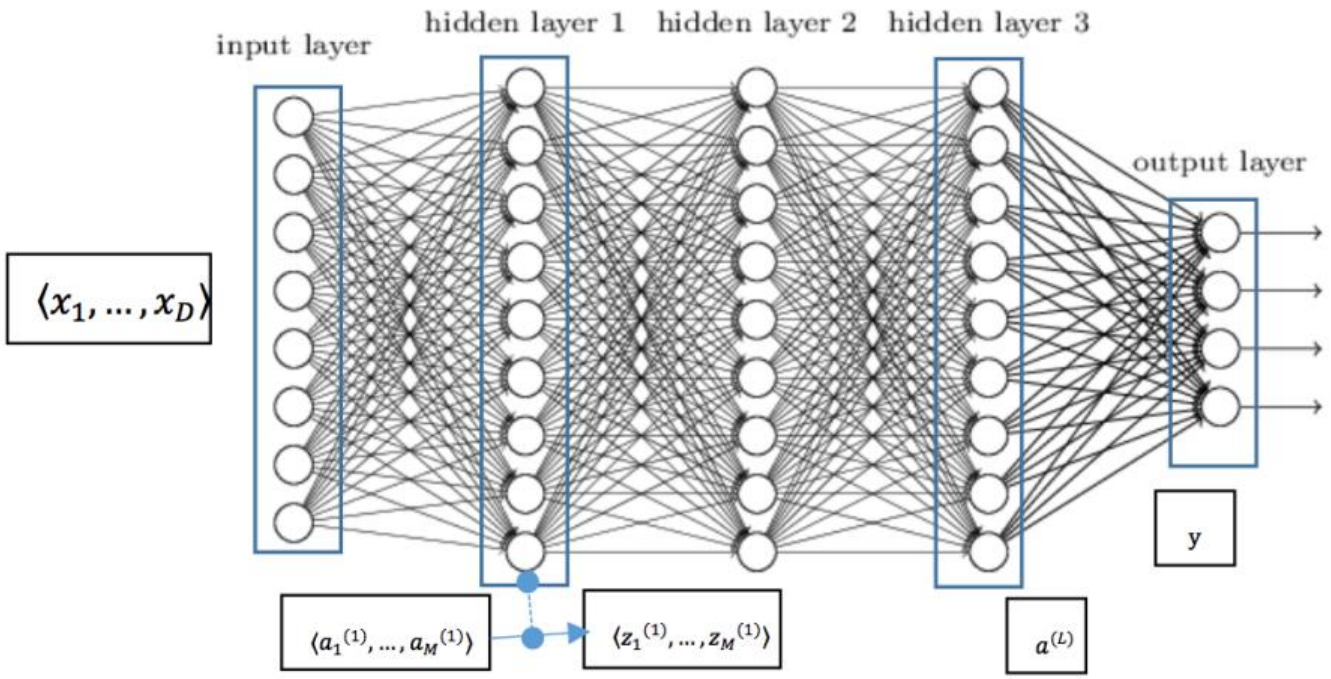
神经网络模型-隐层1到隐层2

- 隐藏层1的输出经过线性变换成为隐藏层2的输入：隐藏层2的输入$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + b_k$,此处M为隐藏层1的神经元数目，比如300，k为隐藏层2的神经元结点数目，假设设置隐藏层1的结点数目N=100，则W权重矩阵为300*100，B偏置矩阵为100*100，最终得到的$a^{(2)}$为100*100的矩阵。
- 隐藏层2对$a^{(2)}$通过激活函数f得到激活值$z^{(2)}$: $z_j^{(2)} = f(a_j^{(2)})$，最终得到的$z^{(2)}$也为 100*100（样本数*神经元数）的矩阵。
- 本实验中，隐藏层1和隐藏层2的激活函数f都选择relu函数



神经网络模型-隐层2到输出层

- 同样的，隐藏层2的输出经过线性变换成为输出层的输入：输出层的输入$a_l^{(3)} = \sum_{k=1}^{N} w_{lk}^{(3)} z_k + b_l$,此处N为隐藏层2的神经元数目，比如100，l为输出层的神经元结点数目，因为输出为0~9的数字，所以输出层的结点数目O=10，则W权重矩阵为100*10，B偏置矩阵为100*10，最终得到的$a^{(3)}$为100*10的矩阵。
- 输出层对$a^{(3)}$经过激活函数得到输出值y:$y_l = \sigma\left(a_l^{(3)}\right)$
- 输出层的激活函数设置σ为softmax函数



前馈神经网络

- 将上面的计算过程通用化成N层的神经网络，第1层是输入层，第N层是输出层，中间的每个层L与层L+1紧密相联。这种模式下，要计算神经网络的输出结果，我们可以按照之前描述的步骤，按部就班，进行前向传播，逐一计算第2层的所有激活值，然后是第3层的激活值，以此类推，直到第N层。这是一个前馈神经网络的例子，因为这种联接图没有闭环或回路。

前向传播

- 隐层将输入经过线性变换wx+b后还用到了激活函数（传递函数）。常用的激活函数除了sigmoid函数，还有tanh、relu等别的激活函数。激活函数使线性的结果非线性化。本次Project用到的是relu函数。
- 为什么需要激活函数？简单理解上，如果不加激活函数，无论多少层隐层，最终的结果还是原始输入的线性变化，这样一层隐层就可以达到结果，就没有多层感知器的意义了。 所以每个隐层都会配一个激活函数，提供非线性变化。
- 输出层经过线性变换后， 用softmax函数。

为什么要用relu函数？

- http://blog.csdn.net/leo_xu06/article/details/53708647

权重矩阵W 以及 偏移向量B

- 仔细观察上面的计算过程，可以发现，W和B取值的不同，决定了神经网络的输出值。
- 训练模型的目的，是找到一组合适的W和B的取值，使得输入值经过一系列变换之后能够接近预测值。
- 神经网络的学习过程就是**学习怎么去控制权重矩阵**

如何训练？

- 既然希望神经网络的输出尽可能的接近真正想要预测的值，那么就可以通过比较当前网络的**预测值**与**目标值**，根据两者的差异情况来更新每一层的权重矩阵。
- 如何比较差异情况？ 损失函数(loss function)
- 如何更新权重矩阵？梯度下降(Gradient descend)

损失函数(loss function)

- 损失函数，又称为误差函数（error function ），也有叫cost function。是用于衡量预测值和目标值差异的函数。
- loss function的输出值越高表示差异性越大，那么神经网络的训练目标就是尽可能的缩小差异
- 本次实验中采用的loss function为；$E(w) = \sum_{n=1}^{N} \sum_{k=1}^{k} -t_k^n ln y_k(x_n, w)$（推导过程详见准备知识.pdf文档）。训练目标是调整w，使得$E(w)$最小

梯度下降(Gradient descend)

- 梯度下降：通过使loss值向当前点对应梯度的反方向不断移动，来降低loss。一次移动多少是由学习速率（learning rate）来控制的。
- 难点1：有可能梯度下降只能找到局部最小值，找不到全局最小值
- 难点2：如何快速的计算梯度？如何更新隐藏层的权重？本次实验用到的是反向传播算法。（反向传播算法是一种计算梯度的方法，其贡献如同FFT）

本实验中神经网络的训练过程

- 数据加载
- 数据处理，归一化：像素值/256
- 权重初始化：xavier方法
- 将10000个数据，分为100个epochs
- 每个epochs的处理流程：
  - 将以下过程迭代100次：
    - 正向传播，计算loss（记录最后一次迭代计算的loss）
    - 反向传播
    - 梯度下降
    - 在最后一次迭代，用测试集计算该epochs的精确度
- 在界面上画出每个epochs的loss变化曲线图，以及epochs对应的精确度变化曲线图

权重初始化-xavier方法

- 产生m*n维度的w
- 初始的w采用随机化的方式产生，使得w均匀分布在$(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}})$范围内

权重初始化-xavier方法-为什么？

- Glorot Xavier 的论文“Understanding the difficulty of training deep feedforward neural networks.”
- https://zhuanlan.zhihu.com/p/22028079