BP神经网络算法程序实现鸢尾花数据集分类

郑力南

算法推导

基本的梯度下降公式如下，其中α表示学习率（更新步长）

$$\theta \leftarrow \theta - \alpha \frac{\partial L}{\partial \theta}$$

在进行梯度下降更新参数时，关键在于损失函数关于参数θ 偏导数的计算

算法推导

根据链式法则，损失函数关于参数 θ 的偏导数（以第q层的神经元为例）为

$$
\begin{aligned}
\frac{\partial L}{\partial \theta} &= \frac{\partial L}{\partial \mathbf{W}^{(q)}} \\
&= \frac{\partial L}{\partial \sigma(\mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)})} \frac{\partial \sigma(\mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)})}{\partial \mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}} \frac{\partial \mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}}{\partial \mathbf{W}^{(q)}} \\
&= L'(\sigma(\mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}))\sigma'(\mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}) \frac{\partial \mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}}{\partial \mathbf{W}^{(q)}}
\end{aligned}
$$

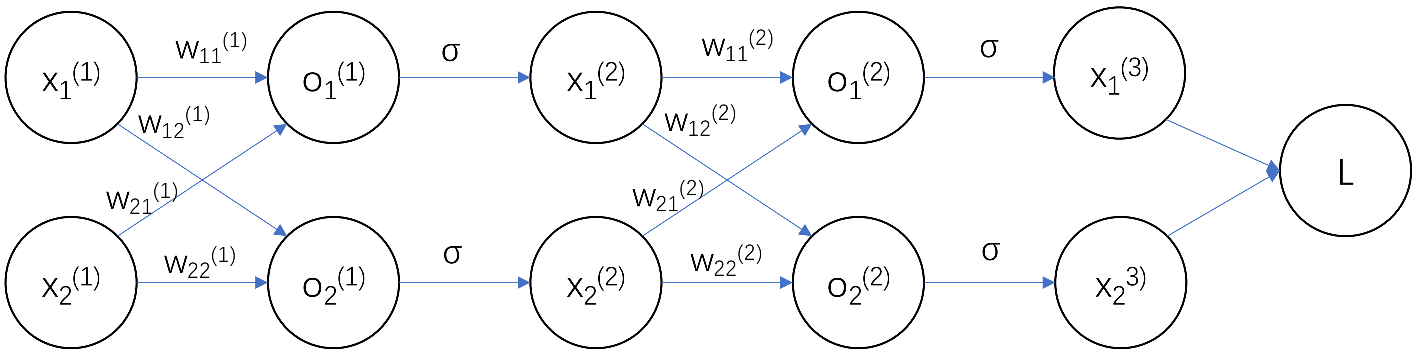
算法推导

其中 $\frac{\partial \mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}}{\partial \mathbf{W}^{(q)}}$ 继续用链式法则展开，可以得到一个关于$\frac{\partial \mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)}}{\partial \mathbf{W}^{(q)}}$的表达式，将其继续用链式法则推导，直到第q层结束

$$
\begin{aligned}
\frac{\partial \mathbf{W}^{(Q)^T}\mathbf{X}^{(Q)}}{\partial \mathbf{W}^{(q)}} &= \frac{\partial \mathbf{W}^{(Q)^T}\sigma(\mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)})}{\partial \mathbf{W}^{(q)}} \\
&= \frac{\partial \mathbf{W}^{(Q)^T}\sigma(\mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)})}{\partial \sigma(\mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)})}\frac{\partial \sigma(\mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)})}{\partial \mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)}}\frac{\partial \mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)}}{\partial \mathbf{W}^{(q)}} \\
&= \mathbf{W}^{(Q)}\sigma'(\mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)})\frac{\partial \mathbf{W}^{(Q-1)^T}\mathbf{X}^{(Q-1)}}{\partial \mathbf{W}^{(q)}}
\end{aligned}
$$

算法推导

以一个两层神经网络为例



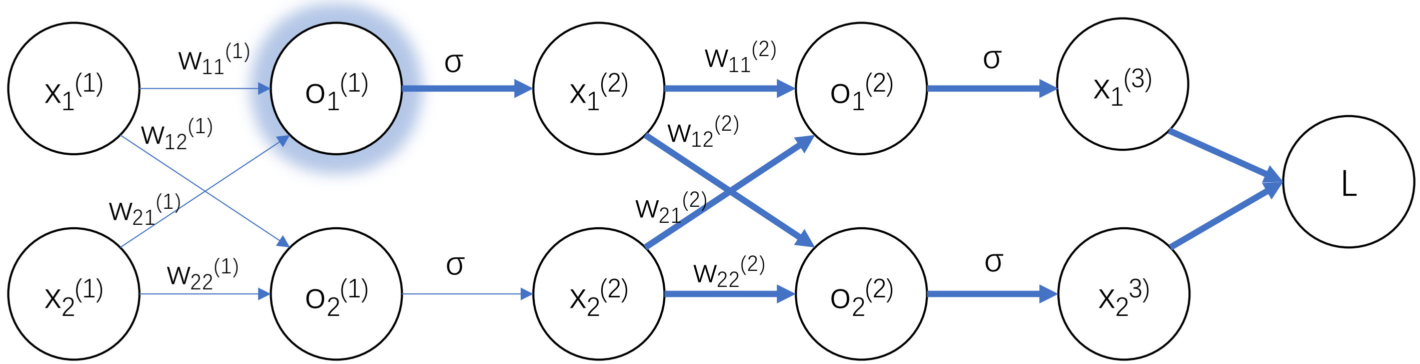
算法推导

例如要计算损失L关于$w_{11}^{(1)}$的偏导数，根据链式计算方式为

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}_{11}^{(1)}} &= \frac{\partial L}{\partial \mathbf{x}_1^{(3)}}\frac{\partial \mathbf{x}_1^{(3)}}{\partial \mathbf{o}_1^{(2)}}\frac{\partial \mathbf{o}_1^{(2)}}{\partial \mathbf{x}_1^{(2)}}\frac{\partial \mathbf{x}_1^{(2)}}{\partial \mathbf{o}_1^{(1)}}\frac{\partial \mathbf{o}_1^{(1)}}{\partial \mathbf{w}_{11}^{(1)}} \\ &\quad + \frac{\partial L}{\partial \mathbf{x}_2^{(3)}}\frac{\partial \mathbf{x}_2^{(3)}}{\partial \mathbf{o}_2^{(2)}}\frac{\partial \mathbf{o}_2^{(2)}}{\partial \mathbf{x}_1^{(2)}}\frac{\partial \mathbf{x}_1^{(2)}}{\partial \mathbf{o}_1^{(1)}}\frac{\partial \mathbf{o}_1^{(1)}}{\partial \mathbf{w}_{11}^{(1)}} \\ &= (\frac{\partial L}{\partial \mathbf{x}_1^{(3)}}\frac{\partial \mathbf{x}_1^{(3)}}{\partial \mathbf{o}_1^{(2)}}\frac{\partial \mathbf{o}_1^{(2)}}{\partial \mathbf{x}_1^{(2)}}\frac{\partial \mathbf{x}_1^{(2)}}{\partial \mathbf{o}_1^{(1)}} \\ &\quad + \frac{\partial L}{\partial \mathbf{x}_2^{(3)}}\frac{\partial \mathbf{x}_2^{(3)}}{\partial \mathbf{o}_2^{(2)}}\frac{\partial \mathbf{o}_2^{(2)}}{\partial \mathbf{x}_1^{(2)}}\frac{\partial \mathbf{x}_1^{(2)}}{\partial \mathbf{o}_1^{(1)}})\frac{\partial \mathbf{o}_1^{(1)}}{\partial \mathbf{w}_{11}^{(1)}}\end{aligned}$$

算法推导

从计算图中可以直观地看出，损失L关于$w_{11}^{(1)}$的偏导数实际上是$w_{11}^{(1)}$这条边指向的节点，到终点的路径（偏导数的乘积）的和再乘上改节点对$w_{11}^{(1)}$的偏导数



$$\frac{\partial L}{\partial \mathbf{w}_{11}^{(1)}} = \left(\frac{\partial L}{\partial \mathbf{x}_1^{(3)}} \frac{\partial \mathbf{x}_1^{(3)}}{\partial \mathbf{o}_1^{(2)}} \frac{\partial \mathbf{o}_1^{(2)}}{\partial \mathbf{x}_1^{(2)}} \frac{\partial \mathbf{x}_1^{(2)}}{\partial \mathbf{o}_1^{(1)}} + \frac{\partial L}{\partial \mathbf{x}_2^{(3)}} \frac{\partial \mathbf{x}_2^{(3)}}{\partial \mathbf{o}_2^{(2)}} \frac{\partial \mathbf{o}_2^{(2)}}{\partial \mathbf{x}_1^{(2)}} \frac{\partial \mathbf{x}_1^{(2)}}{\partial \mathbf{o}_1^{(1)}}\right) \frac{\partial \mathbf{o}_1^{(1)}}{\partial \mathbf{w}_{11}^{(1)}}$$

算法推导

对于$\frac{\partial L}{\partial x_i}$，在鸢尾花分类问题中最后一层是softmax，并且使用交叉熵作为损失函数。首先先求解损失函数关于softmax输出a_j的偏导数

$$\frac{\partial L}{\partial a_j} = \frac{\partial\left(-\sum_j y_j \ln a_j\right)}{\partial a_j} = -\sum_j y_j \frac{1}{a_j}$$

算法推导

对于$\frac{\partial L}{\partial x_i}$， 有 $\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial a_j}\frac{\partial a_j}{\partial x_i} = \left(-\sum_j y_j \frac{1}{a_j}\right)\frac{\partial a_j}{\partial x_i}$

当i=j时，

$$\frac{\partial a_i}{\partial x_i} = \frac{\partial\left(\frac{e^{x_i}}{\sum_k e^{x_k}}\right)}{\partial x_i} = \frac{\sum_k e^{x_k} e^{x_i} - \left(e^{x_i}\right)^2}{\sum_k \left(e^{x_k}\right)^2} = \left(\frac{e^{x_i}}{\sum_k e^{x_k}}\right)\left(1 - \frac{e^{x_i}}{\sum_k e^{x_k}}\right) = a_i(1 - a_i)$$

当i≠j时，

$$\frac{\partial a_j}{\partial x_i} = \frac{\partial\left(\frac{e^{x_j}}{\sum_k e^{x_k}}\right)}{\partial x_i} = -e^{x_j}\left(\frac{1}{\sum_k e^{x_k}}\right)e^{x_i} = -a_i a_j$$

算法推导

综合可得

$$\frac{\partial L}{\partial x_i} = a_i - y_i$$

即softmax输出概率与ground thruth的差

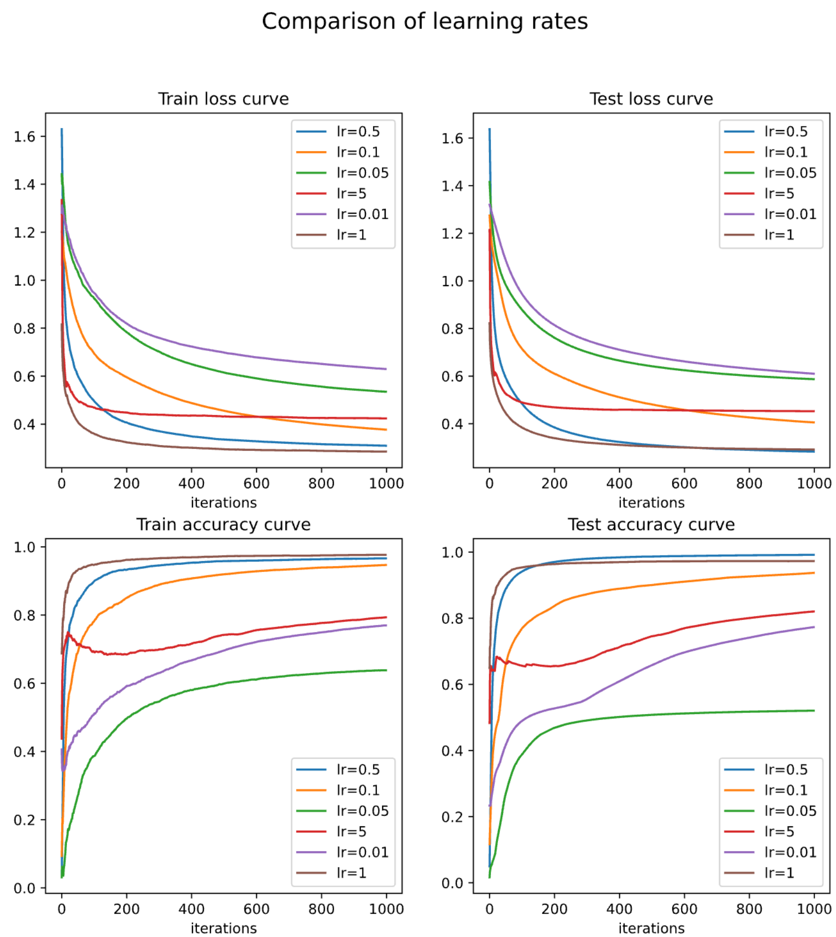
实验

默认设置：

网络结构为输入层为5个神经元（含增广的1），隐藏层为8个神经元，输出层为3个神经元（对应三个类别的概率）。

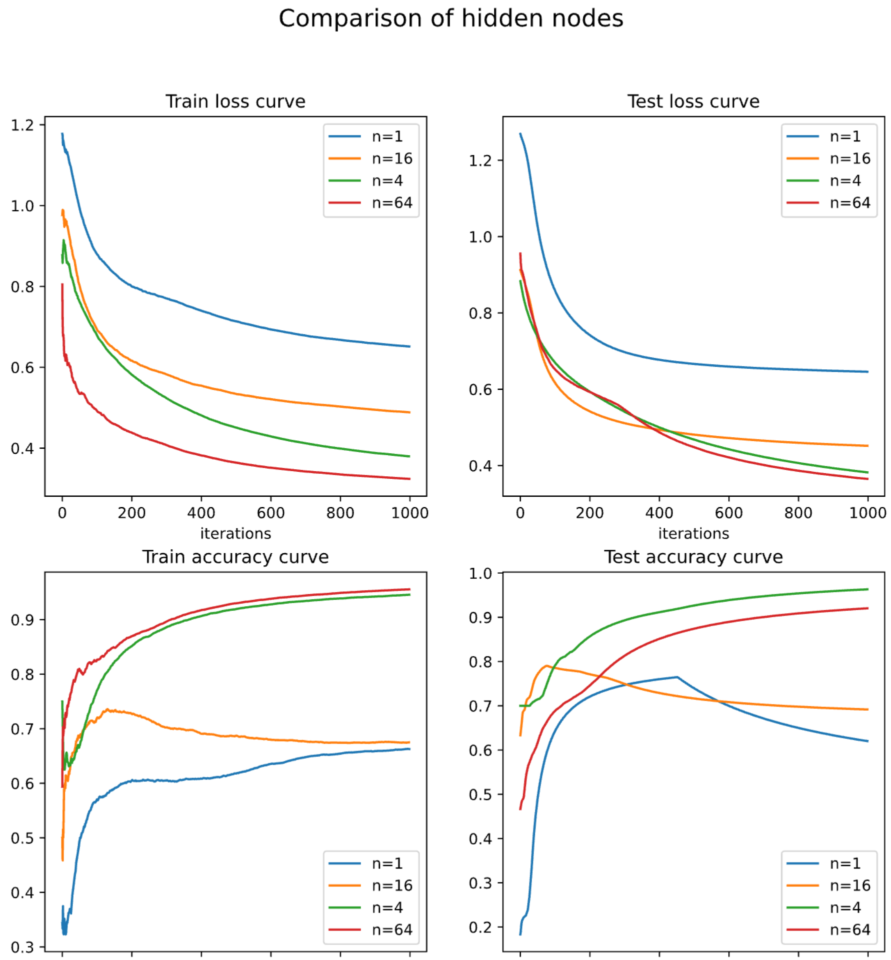
batch_size为16，学习率为0.2，γ为0（默认不适用动量），迭代次数为1000，参数初始化方式为-1至1的均匀分布，训练测试划分比为8:2，输入数据标准化方式使用z-score，损失函数使用交叉熵，激活函数使用tanh

实验——学习率对实验结果的影响



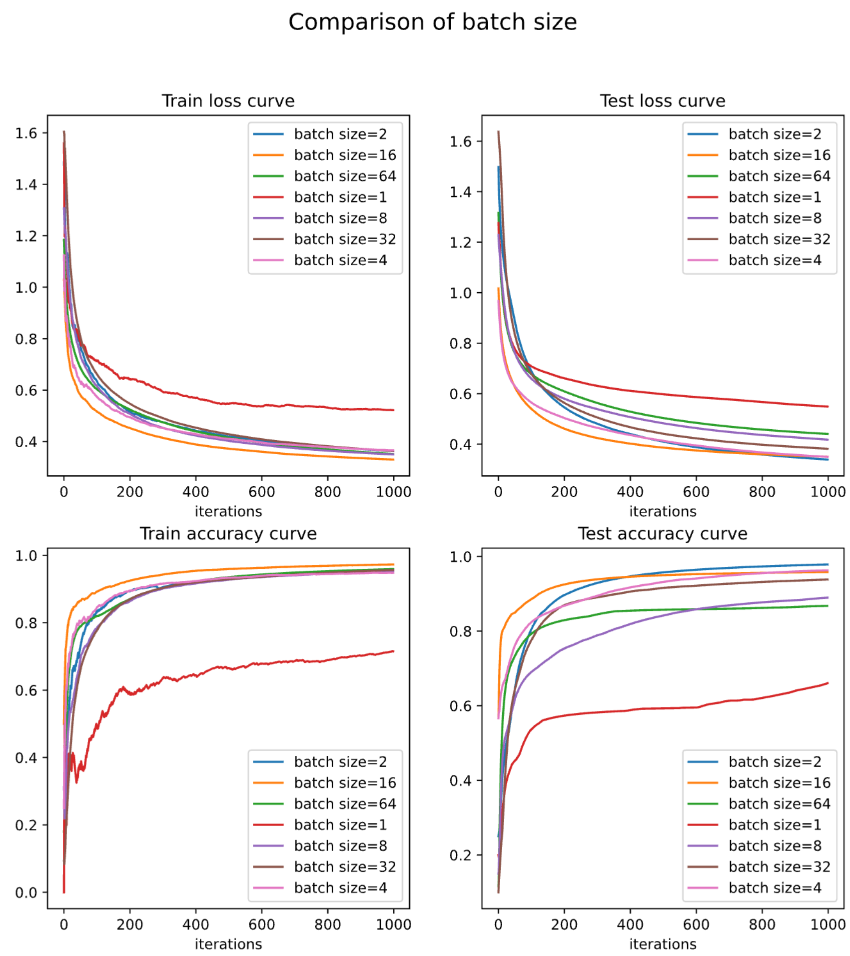
根据曲线可以发现，当学习率过小时（lr=0.01），曲线收敛很慢；当学习率过大时（lr=5），曲线能够很快收敛，但是效果较差，在测试数据上loss很早地收敛于一个较高的值；当学习率的值适中时（ lr=1，lr=0.5），收敛效果较好。

实验——隐藏层神经元个数对实验结果的影响



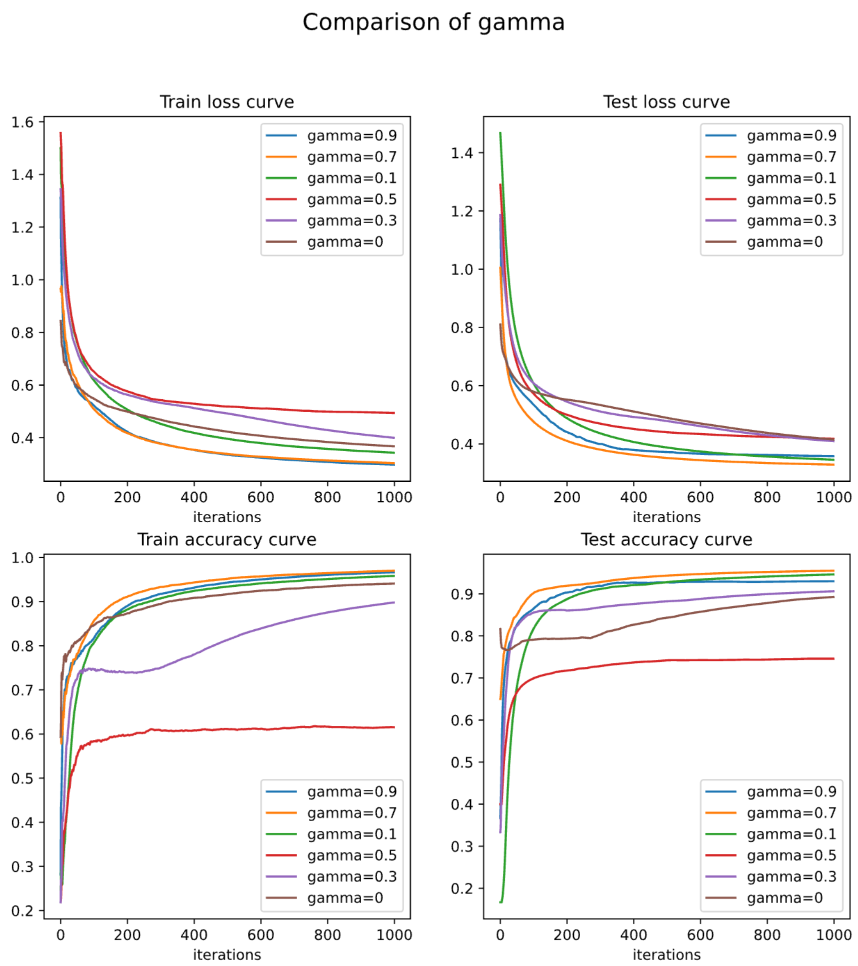
当隐藏层神经元个数较小时（n=1），神经网络表达能力受限，收敛结果很差；增加隐藏层神经元个数，能够有效的改进loss收敛效果，但是当n过大时，效果提升不明显，而又额外增加了计算量，训练耗时更长。

实验——batch size对实验结果的影响



根据曲线可以发现，当batch size为1时，即每次只根据单个样本的梯度进行更新参数，曲线收敛到的结果较差；当batch size稍稍增大到大于1的数时，loss和准确率就可以收敛到一个较为满意的值，由此可见，使用batch的方式训练相比于使用单个样本训练具有很大优势。

实验——动量对实验结果的影响



根据曲线可以发现，当加入动量后，训练效果和收敛速度多数情况下能够得到提升（gamma=0.7, gamma=0.9），但是有时不能保证加入动量后训练效果能够更好（gamma=0. 5），因此控制动量衰减的参数gamma可能需要人为多次调试。