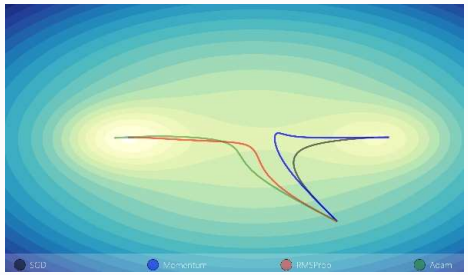
媒体与认知

Media and Cognition



V 1.0 2022.3.17

清华大学电子工程系

王生进 彭良瑞 李亚利

Email: {wgsgj, penglr, liyali13}@tsinghua.edu.cn

第 4 讲 深度学习

一、深度学习概述

二、深度学习的优化方法

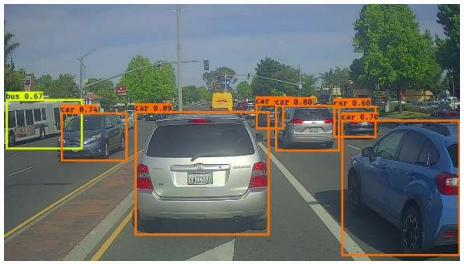
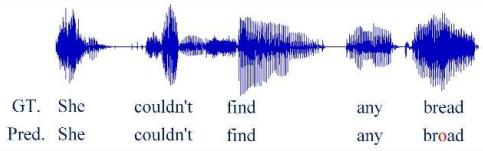
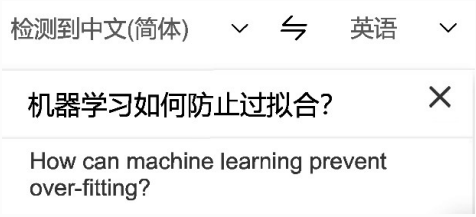
三、深度学习的正则化方法

问题

- 深度学习是什么？
- 如何训练深度学习模型？
- 深度学习如何防止过拟合？

一、深度学习概述

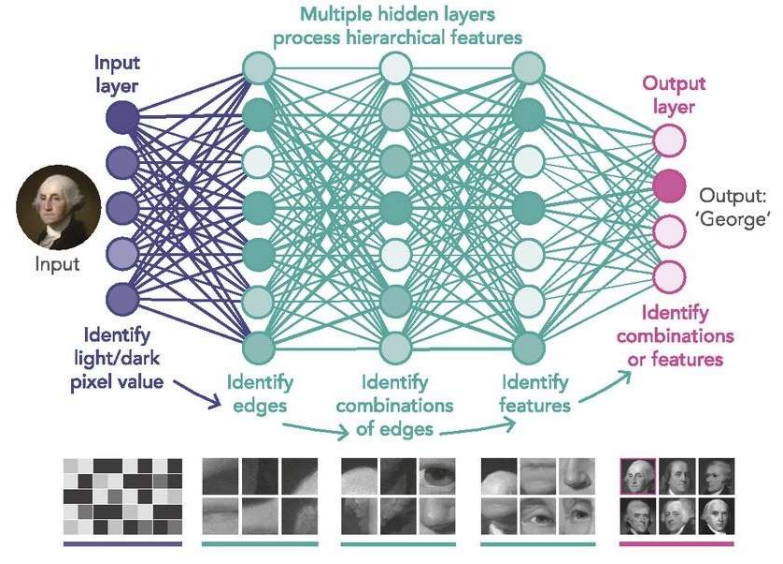
➢ 目前深度学习技术在机器学习领域显著优于传统方法

图像分类	图像分割	目标检测
		
语音识别	**机器翻译**	**对话系统**
		

深度学习是什么？

- 深度学习(Deep Learning, DL)是基于人工神经网络的机器学习技术，通过构建具有多个隐含层的深层神经网络，来实现数据驱动的模型参数学习

- 深度学习通过组合低层特征形成更抽象的高层特征表示，模拟大脑的认知机制来处理图像、声音、文本等数据



深度学习与传统方法的区别与联系

➢ 区别

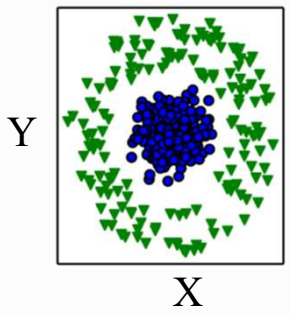
	传统方法（如 统计模式分类）	深度学习（深层神经网络）
特征提取	人工设计特征提取方法	自动学习特征提取方法
模型参数	样本统计分布参数 等	神经网络节点的权值系数
模式分类训练准则	最小错误率 等	交叉熵极小化
训练过程	多步骤，分模块或分阶段训练	端到端网络迭代训练，直接优化任务总体目标
计算规模	适中	处理数据量大，训练迭代次数多，需要并行计算硬件（如 Graphics Processing Unit, GPU）的支持

➢ 联系

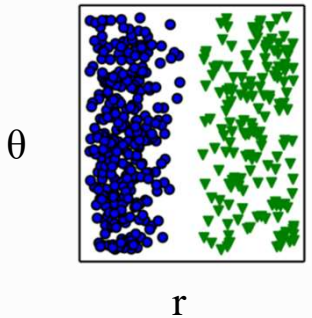
- 单个神经元节点与传统线性分类模型类似
- 深度学习模型中可以结合传统的统计方法
- 传统的统计方法增加模型复杂度也可以提升性能

深度学习的非线性表示能力

- 数据的不同表示方法

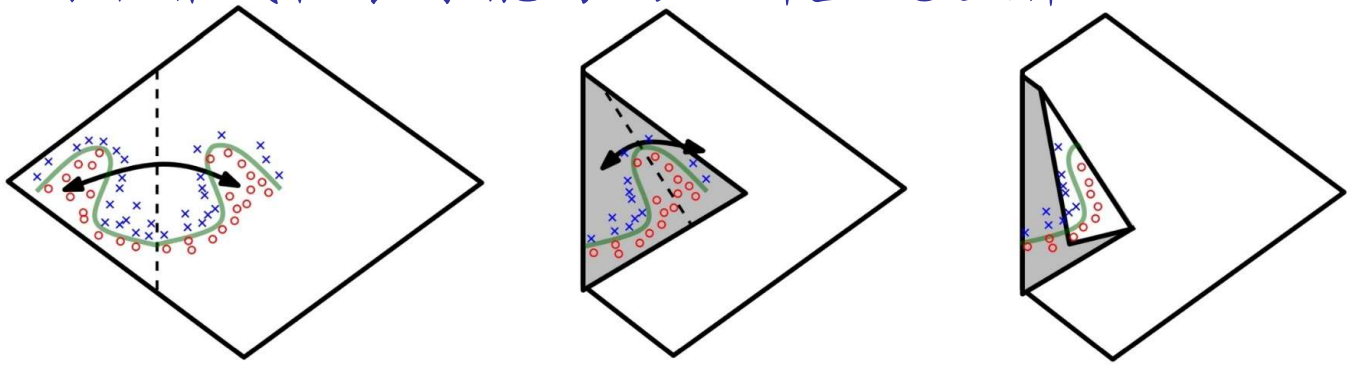


平面直角坐标系 X - Y



极坐标系 r - θ

- 深层神经网络非线性表示能力的一种直观理解

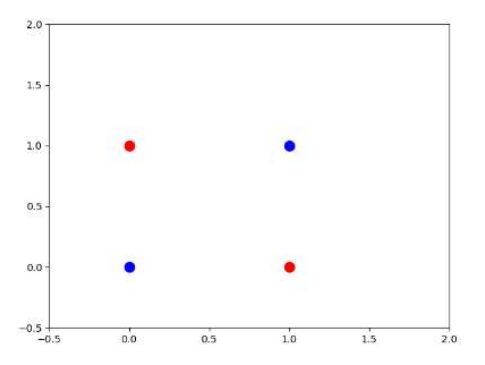


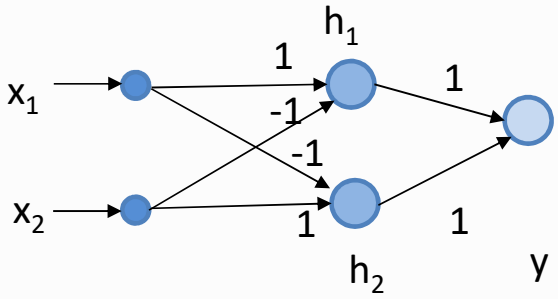
采用绝对值整流函数的神经元对其输入中的每对镜像点有相同的输出。
镜像的对称轴由神经元的权值和偏置量定义的超平面给出

Montúfar G, et al. On the number of linear regions of deep neural networks. NIPS, 2014

深度学习的非线性表示能力

- 非线性表示能力得益于神经网络层数、每层节点数、非线性激活函数
  - 异或问题求解
    - 假设隐含层神经元不用偏置量
    - 采用ReLU激活函数

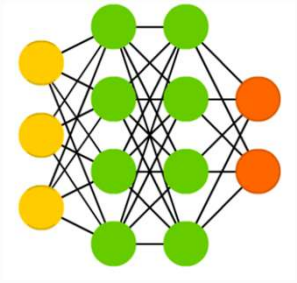




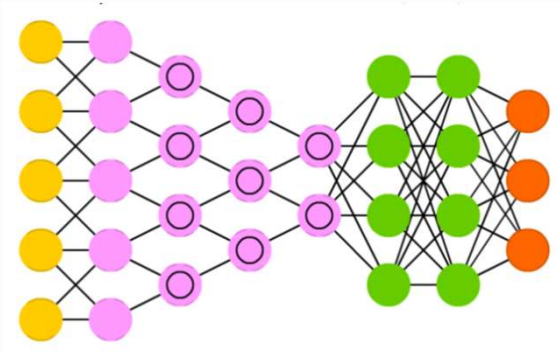
x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	0	1	0.5
1	0	1	0	0.5
1	1	0	0	0

基本深度学习网络架构

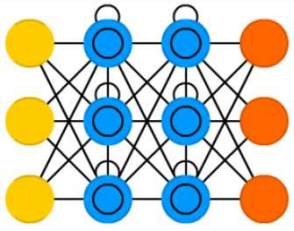
- 前馈神经网络



- 卷积神经网络



- 循环神经网络



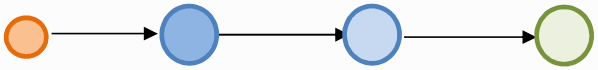
常用深度学习开源软件

- Caffe https://caffe.berkeleyvision.org/
  - 快速卷积神经网络实现， 支持CPU以及GPU
- TensorFlow https://www.tensorflow.org/
  - Tensor(张量)表示N维数组，Flow(流)表示数据流图
- Torch http://torch.ch/
  - 基于Lua语言的科学计算平台，支持CPU/GPU运算
- PyTorch https://pytorch.org/
  - 基于Torch的Python机器学习库，支持自动求导、动态图
- MxNet https://mxnet.apache.org/
- 图神经网络工具包Deep Graph Library (DGL) https://www.dgl.ai/
- Paddle Paddle 飞浆 https://www.paddlepaddle.org.cn/
- Jittor 计图 https://cg.cs.tsinghua.edu.cn/jittor/
  - 清华大学计算机系于2020年3月推出

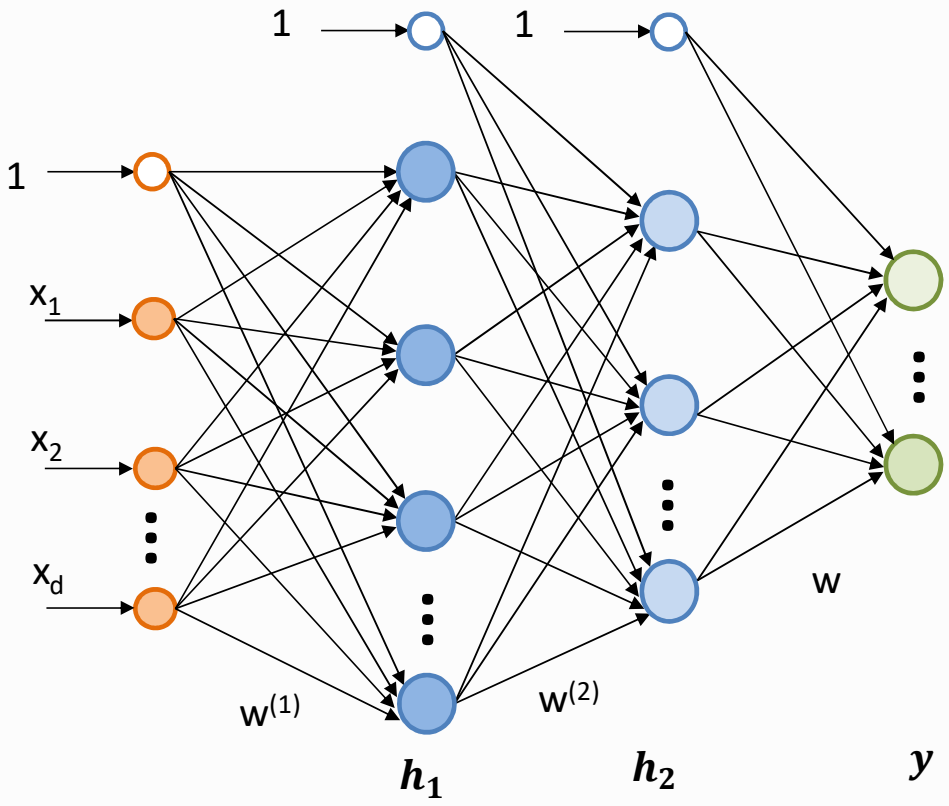
深度学习模型

- 设有一个用于模式分类任务的前馈神经网络
  - 具有2个隐含层$\boldsymbol{h_1}, \boldsymbol{h_2}$，节点分别为$d_1, d_2$，激活函数均为ReLU
  - 输出层节点数为C，激活函数为Softmax
  - 训练采用的目标函数为交叉熵
  - 数据、权值向量可用列向量或行向量表示

列向量 $\mathbf{x} = (\boldsymbol{x_1}, \ldots \boldsymbol{x_d})^{\boldsymbol{T}}$
行向量 $\mathbf{x} = (\boldsymbol{x_1}, \ldots \boldsymbol{x_d})$

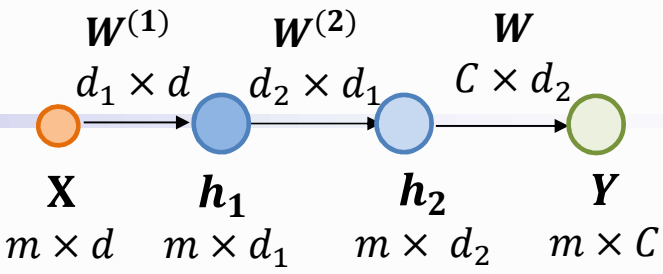


计算图



网络结构示意图

基于矩阵及向量的表示



- 输入一个批量m个样本$\{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(m)}\}$，对应one-hot向量类别真值为$\{\mathrm{y}^{*(1)},\ldots,\mathrm{y}^{*(m)}\}$，可将每个样本数据表示为行向量，数据矩阵$\boldsymbol{X}$的行数为样本个数，维度为$m \times d$；真值矩阵$\boldsymbol{Y}^*$维度为$m \times C$

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x_1}^{(1)} & \cdots & \boldsymbol{x_d}^{(1)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_1}^{(m)} & \cdots & \boldsymbol{x_d}^{(m)} \end{pmatrix}$$

- 第一个隐含层节点个数为d_1，输出为$\boldsymbol{h_1}$，每个节点权值向量为行向量，对应的权值矩阵$\boldsymbol{W}^{(1)}$维度为$d_1 \times d$；第二个隐含层节点个数为d_2，输出为$\boldsymbol{h_2}$，权值矩阵$\boldsymbol{W}^{(2)}$维度为$d_2 \times d_1$；两个隐含层向量形式的偏置量分别为$\boldsymbol{b_1}$，$\boldsymbol{b_2}$

$$\boldsymbol{h_1} = \boldsymbol{ReLU}(\,(\boldsymbol{X}(\boldsymbol{W}^{(1)})^{\boldsymbol{T}} + \boldsymbol{b_1})\,,\ \text{维度为} m \times d_1$$

$$\boldsymbol{h_2} = \boldsymbol{ReLU}\big(\,(\boldsymbol{h_1}(\boldsymbol{W}^{(2)})^{\boldsymbol{T}} + \boldsymbol{b_2})\,,\ \text{维度为} m \times d_2$$

$$\boldsymbol{W}^{(1)} = \begin{pmatrix} \boldsymbol{w_{1,1}}^{(1)} & \cdots & \boldsymbol{w_{1,d}}^{(1)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{w_{d_1,1}}^{(1)} & \cdots & \boldsymbol{w_{d_1,d}}^{(1)} \end{pmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{pmatrix} \boldsymbol{w_{1,1}}^{(2)} & \cdots & \boldsymbol{w_{1,d_1}}^{(2)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{w_{d_2,1}}^{(2)} & \cdots & \boldsymbol{w_{d_2,d_1}}^{(2)} \end{pmatrix}$$

- 输出层权值矩阵$\boldsymbol{W}$维度为$C \times d_2$，偏置向量为$\boldsymbol{b}$

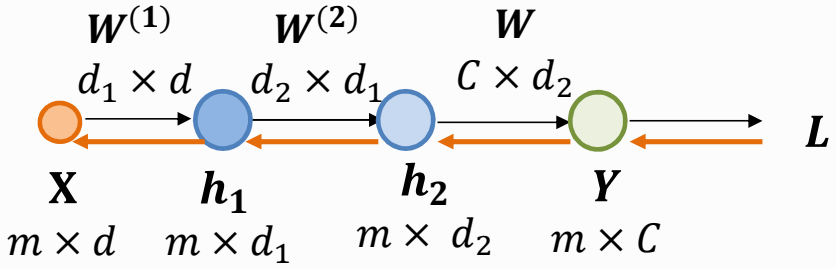
预测值$\boldsymbol{Y} = \boldsymbol{softmax}(\mathrm{Z}) = \boldsymbol{softmax}\big(\boldsymbol{h_2}\boldsymbol{W}^{\boldsymbol{T}} + \boldsymbol{b}\big)$

$\mathbf{Z}$和$\boldsymbol{Y}$的维度均为$m \times C$

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{w_{1,1}} & \cdots & \boldsymbol{w_{1,d_2}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{w_{C,1}} & \cdots & \boldsymbol{w_{C,d_2}} \end{pmatrix}$$

- 目标函数采用交叉熵，利用预测值$\boldsymbol{Y}$及真值$\boldsymbol{Y}^*$计算，并对该批量样本求平均得到误差$\boldsymbol{L}$

误差反向传播



- 利用目标函数计算批量数据的平均误差L，对输出层及隐含层权值矩阵和偏置量求导，得到$\frac{\partial L}{\partial W}$及$\frac{\partial L}{\partial b}$，$\frac{\partial L}{\partial W^{(2)}}$及$\frac{\partial L}{\partial b_2}$，$\frac{\partial L}{\partial W^{(1)}}$及$\frac{\partial L}{\partial b_1}$，通过梯度下降法调整模型参数。此过程中需要计算$\frac{\partial L}{\partial h_2}$与$\frac{\partial L}{\partial h_1}$，使得误差在层间反向传播
- 计算过程涉及矩阵及向量求导，由于矩阵相乘要求维度匹配，在应用复合函数求导的链式法则时，应根据具体情况进行矩阵转置、交换矩阵相乘顺序的处理，例如：

$$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial Z} \cdot \frac{\partial Z}{\partial h_2} = \frac{\partial L}{\partial Z} \cdot W \qquad \frac{\partial L}{\partial W} = \left(\frac{\partial L}{\partial Z}\right)^T \cdot \frac{\partial Z}{\partial W} = \left(\frac{\partial L}{\partial Z}\right)^T \cdot h_2$$

其中，$Z = h_2 W^T + b$，$\frac{\partial L}{\partial Z}$为标量$L$对矩阵$Z$求导，$\frac{\partial L}{\partial Z}$与矩阵$Z$维度相同，均是$m \times C$的矩阵

矩阵及向量求导示例

- $f = \boldsymbol{a}^T X \boldsymbol{b}$, 其中，$f$为标量，$\boldsymbol{a}$是$m$维列向量，$\boldsymbol{b}$是$n$维列向量，$X$是$m \times n$的矩阵，求 $\frac{\partial f}{\partial X}$。

解：标量f对矩阵X求导，输出为维度与X相同的矩阵。

标量f对矩阵X的任意一个元素X_{ij}求导：

$$\frac{\partial a^T X b}{\partial X_{ij}} = \frac{\partial \sum_{p=1}^{m} \sum_{q=1}^{n} a_p X_{pq} b_q}{\partial X_{ij}} = \frac{\partial a_i X_{ij} b_j}{\partial X_{ij}} = a_i b_j$$

在矩阵X的(i, j)位置求导结果是向量$\boldsymbol{a}$的第i个分量和向量$\boldsymbol{b}$第j个分量的乘积，将求导结果排列成一个$m \times n$的矩阵，得到：

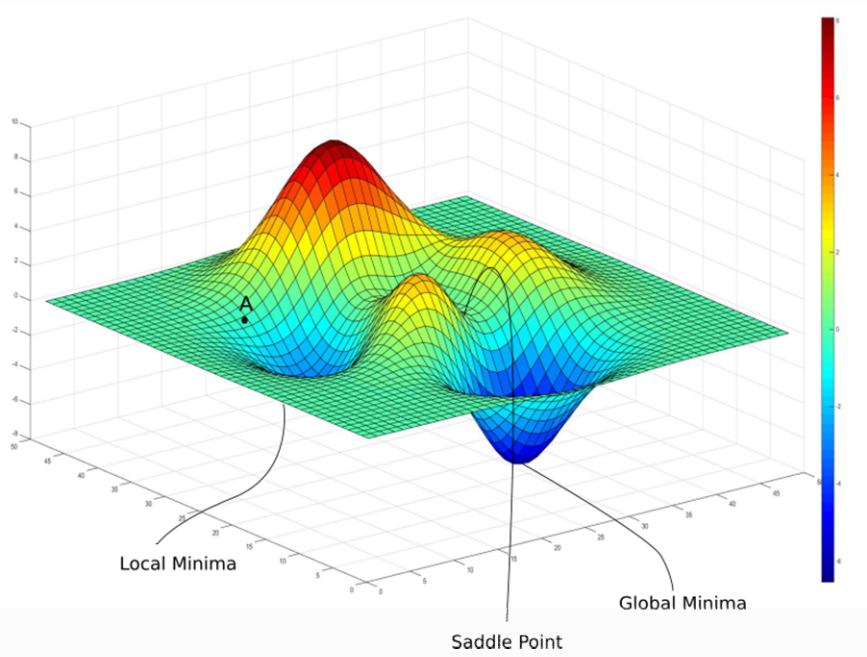
$$\frac{\partial f}{\partial X} = \boldsymbol{a}\boldsymbol{b}^{\boldsymbol{T}}$$

- 对于 $\boldsymbol{Z} = \boldsymbol{h_2}\boldsymbol{W}^{\boldsymbol{T}} + \boldsymbol{b}$，类似推导可得$\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{h_2}} = \boldsymbol{W}$，$\frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}} = \boldsymbol{h_2}$

二、深度学习的优化方法

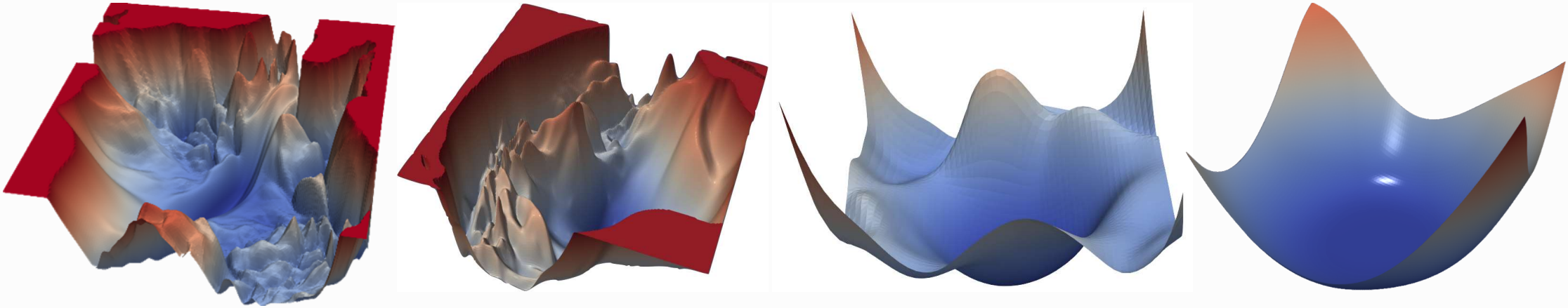
- 神经网络相当于一个复杂的函数
  - 模型的训练为优化问题求解
  - 模型参数为神经元的权值和偏置量

- 高维空间的非凸优化问题
  - 鞍点（Saddle Point）
  - 局部最小值（Local Minima）



神经网络的目标函数

- 不同深层神经网络的目标函数示例



Hao Li, et al. Visualizing the Loss Landscape of Neural Nets. NeurIPS, 2018.

https://www.cs.umd.edu/~tomg/projects/landscapes/

优化方法

- 神经网络模型的训练过程是找到使损失函数的值尽可能小的模型参数
  - 解决这个问题的过程称为最优化（optimization）
  - 数值分析中常见的最优化方法
    - 梯度下降法 $$w_{t+1} \leftarrow w_t - \eta \frac{\partial L}{\partial W}$$
    - 牛顿法：
      - 需要求解目标函数的二阶偏导数构成的方阵(Hessian矩阵)的逆矩阵

$$w_{t+1} \leftarrow w_t - \eta\, H^{-1} \frac{\partial L}{\partial W}$$

    - 拟牛顿法：用正定矩阵来近似Hessian矩阵的逆矩阵
    - 共轭梯度法：通过迭代下降的共轭方向来避免Hessian矩阵求逆计算

优化方法

- 梯度下降法：使用目标函数对模型参数的梯度，沿梯度反方向迭代更新参数
  - 批量梯度下降（batch gradient descent），使用所有的训练样本计算梯度
    - 梯度计算稳定，但计算过慢
  - 随机梯度下降（stochastic gradient descent, SGD），每次只取一个样本进行梯度的计算
    - 梯度计算相对不稳定，容易震荡，整体上趋近于全局最优解的
  - 折衷方案：从训练集中随机取一部分样本进行迭代，即mini-batch gradient descent

- 深度学习中常用的优化方法
  - SGD
  - 动量法
  - RMSProp
  - Adam

随机梯度下降法 (stochastic gradient descent, SGD)

利用梯度下降法的模型参数更新公式：

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta L_t(\theta)$$

其中，η是学习率，θ_t是第 t次迭代的参数，$L(\theta)$是目标函数，$\nabla_\theta L_t(\theta)$是目标函数$L(\theta)$在第 t次迭代时对参数θ的梯度

为表示简便，令 $g_t=\nabla_\theta L_t(\theta)$：

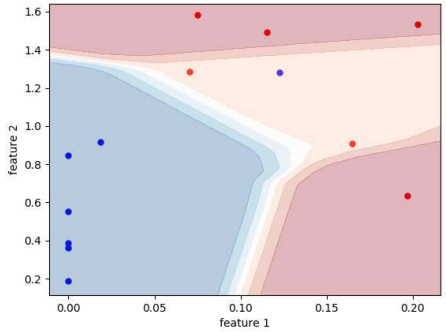
$$\theta_t = \theta_{t-1} - \eta g_t$$

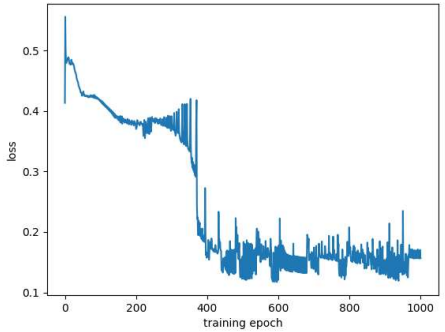
即：

$$\Delta\theta_t = \theta_t - \theta_{t-1} = -\eta g_t$$

不同学习率比较

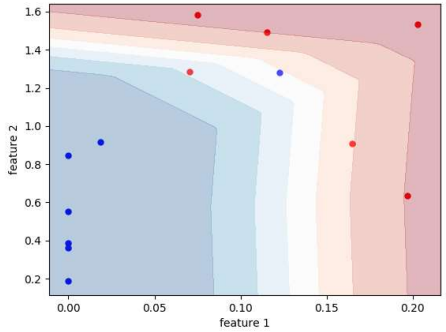
- 1层隐含层，节点数32，激活函数ReLU
  - 优化器：SGD，学习率：0.3，训练轮数：1000

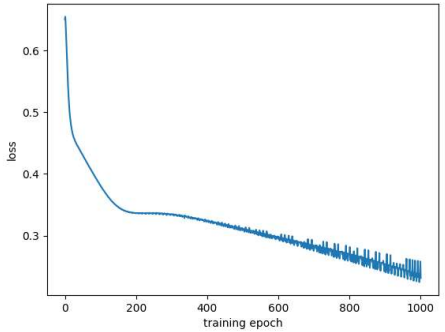




loss = 0.158

  - 优化器：SGD，学习率：0.05，训练轮数：1000

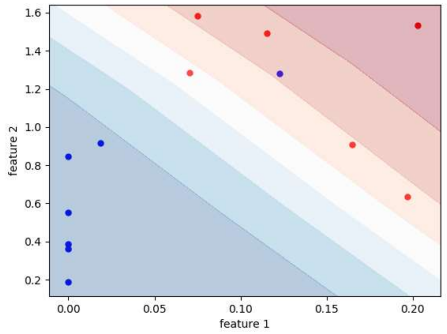


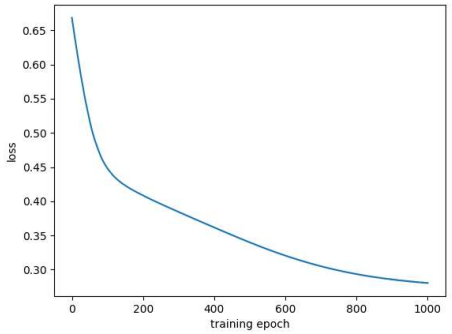


loss = 0.258

不同学习率比较

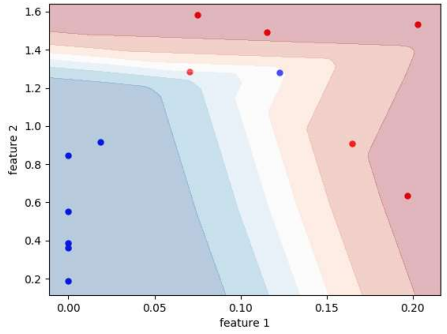
- 1层隐含层，节点数32，激活函数ReLU
  - 优化器：SGD，学习率：0.01，训练轮数：1000

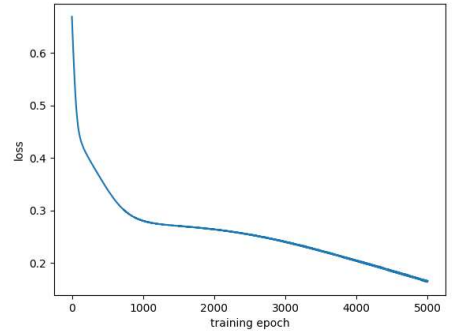




loss = 0.281

  - 优化器：SGD，学习率：0.01，训练轮数：5000





loss = 0.166

梯度估计调整

➢ 动量法（Momentum Method）

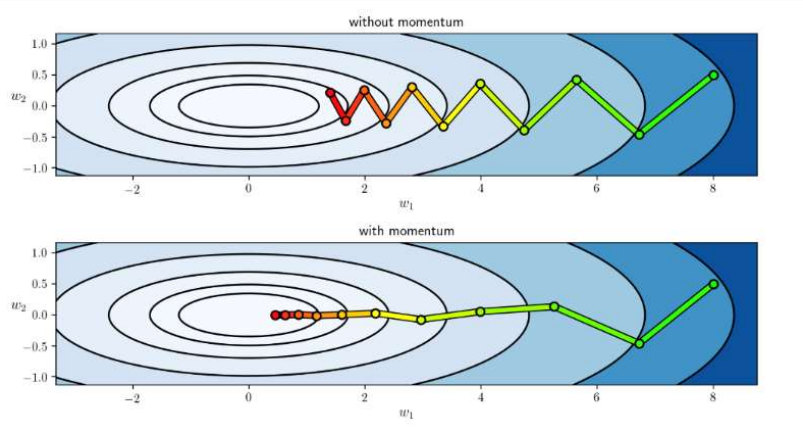
◆ 利用前一次迭代的参数调整量，得到更为稳定的数值

• 在第t次迭代时，计算负梯度的“加权移动平均”作为参数的更新方向

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \eta g_t$$

$$= -\eta \sum_{\tau=1}^{t} \rho^{t-\tau} g_\tau$$

其中，ρ为动量因子，通常设为0.9，η为学习率，g_t表示梯度



RMSProp (root mean square propagation)

- RMSProp在优化过程中自适应地调整学习率
  - $h \leftarrow \alpha\, h + (1-\alpha) g_t \cdot g_t$
  - $\mathrm{W} \leftarrow \mathrm{W} - \eta \frac{1}{\sqrt{h}} g_t$
  - g_t较小时，$\frac{1}{\sqrt{h}}$较大，能够放大梯度
  - g_t较大时，$\frac{1}{\sqrt{h}}$较小，能够约束梯度

Adam

- Adam(Adaptive Moment Estimation)算法，结合了动量法和自适应学习率调整的方法
  - 计算m_t、v_t，分别是对梯度的一阶和二阶矩估计，即对期望$E[g_t]$和$E[g_t^2]$的估计

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad ，其中，t=0时，m_0=0，v_0=0$$

  - 对m_t、v_t进行校正，近似为对期望的无偏估计

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \qquad \widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

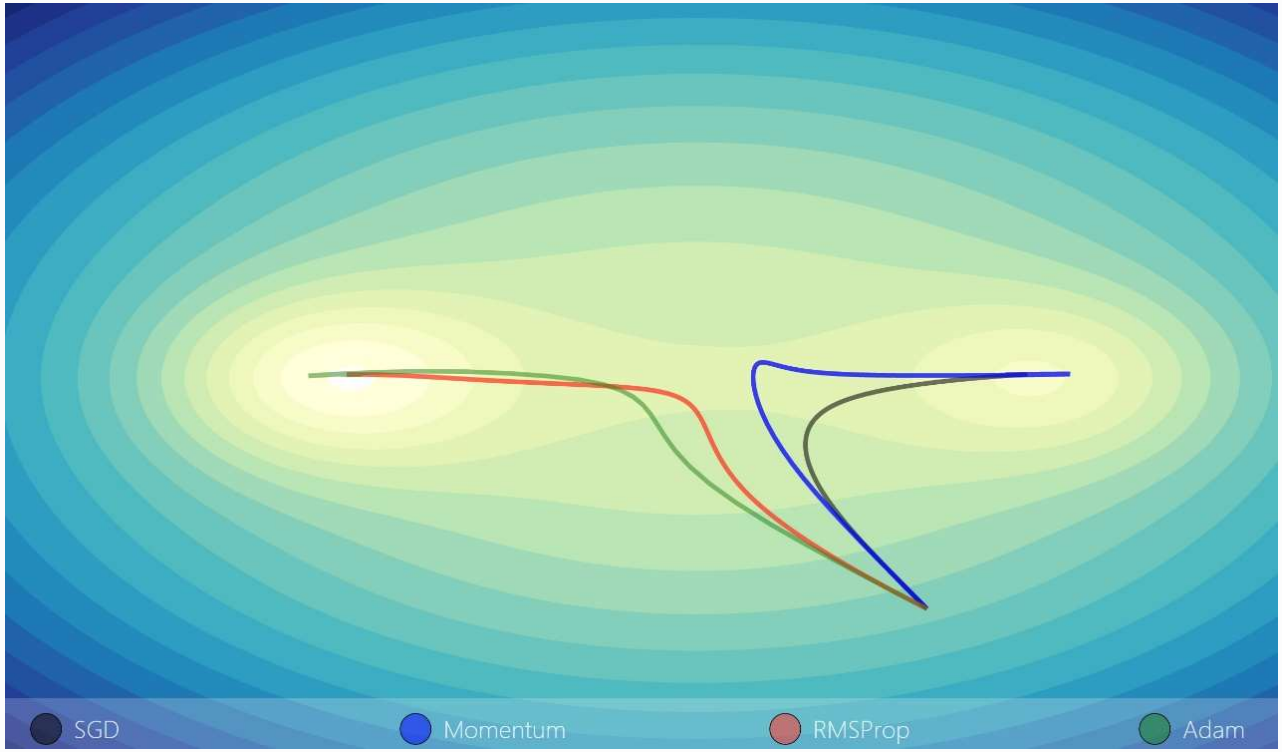
  - 参数更新公式：$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v_t}}+\epsilon}\widehat{m_t}$

通常设 η=0.001， β_1 和 β_2 都是接近 1 的数，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 1e^{-8}$，是为了防止分母为 0

Diederik P K, Jimmy B. ADAM: A method for stochastic optimization. ICLR. arXiv preprint arXiv:1412.6980, 2014.

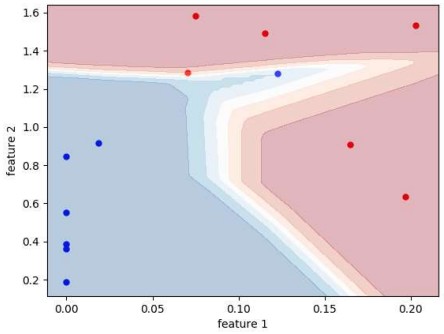
优化方法在线演示

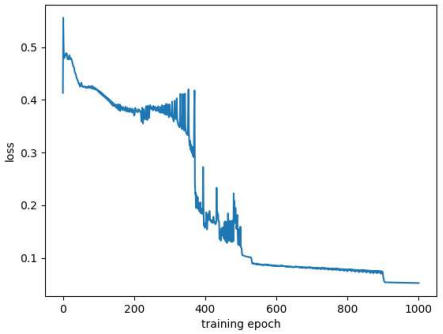
- 比较SGD，ADAM等方法
  - https://bl.ocks.org/EmilienDupont/aaf429be5705b219aaaf8d691e27ca87



不同优化方法比较

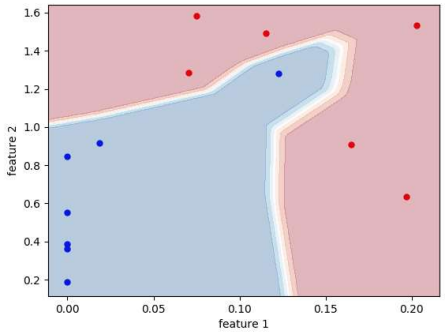
- 1层隐含层，节点数32，激活函数ReLU
  - 优化器：SGD，学习率：0.3 (1-500轮)，0.05 (501-900轮)，0.01 (901-1000轮)

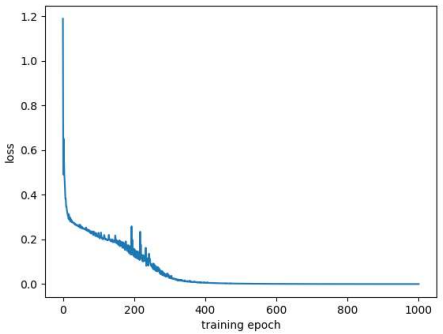




loss = 0.053

  - 优化器：Adam，学习率：0.03，训练轮数：1000





loss = 0.000

深度学习的超参数

- 超参数
  - 层数
  - 每层神经元个数
  - 学习率（以及动态调整算法）
  - 正则化系数
  - mini-batch 大小

模型参数初始化

- 初始化方法
  - 预训练初始化
  - 随机初始化
  - 固定值初始化
    - 偏置（Bias）通常用 0 来初始化

随机初始化

➢Gaussian分布初始化

- Gaussian初始化方法是最常用的方法
  - 按固定均值（比如0）和固定方差（比如0.01）的Gaussian分布进行随机初始化

➢均匀分布初始化

- 参数可以在区间[−r, r]内采用均匀分布进行初始化

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

深度学习模型参数初始化如果全部设为0，会出现什么问题?

A 无法计算损失

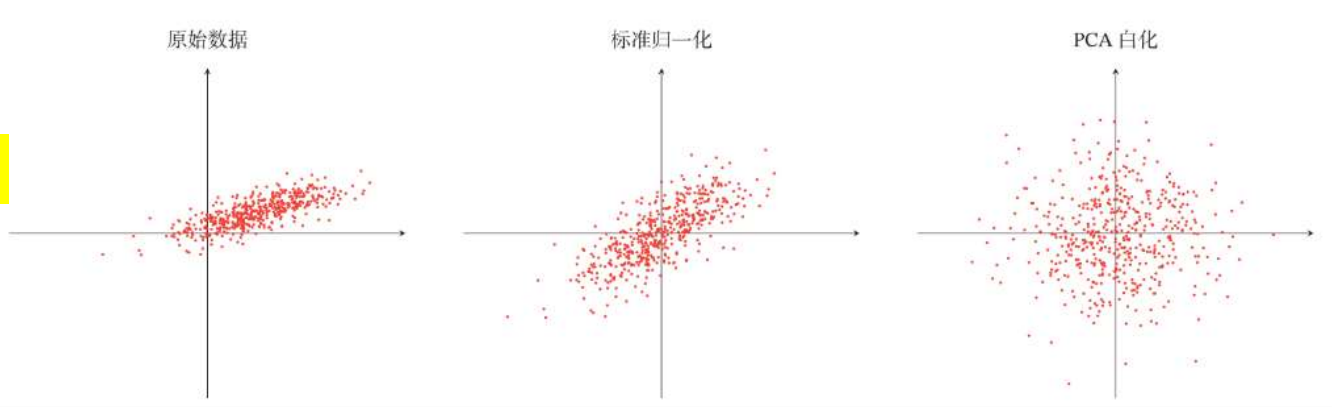
B 无法调整输出层参数

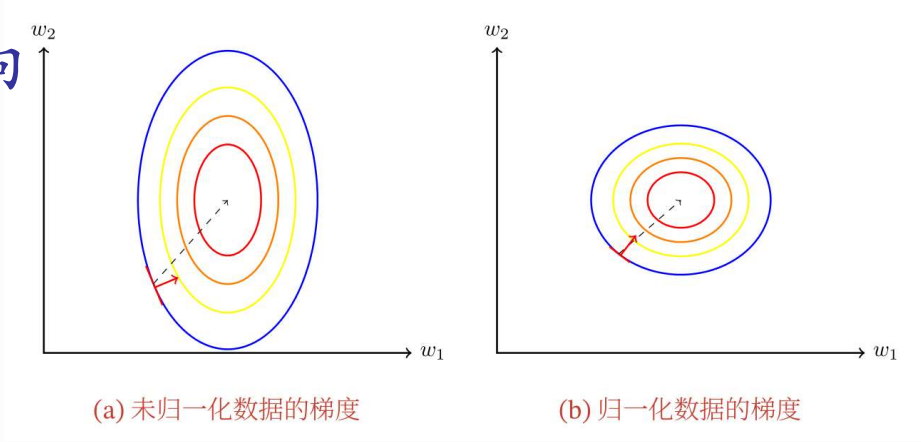
C 无法调整隐含层参数

提交

数据预处理

- 图像大小归一化
- 数值动态范围归一化
  - 最小最大值归一化
  - 标准化
  - PCA
- 数值动态范围归一化对梯度的影响

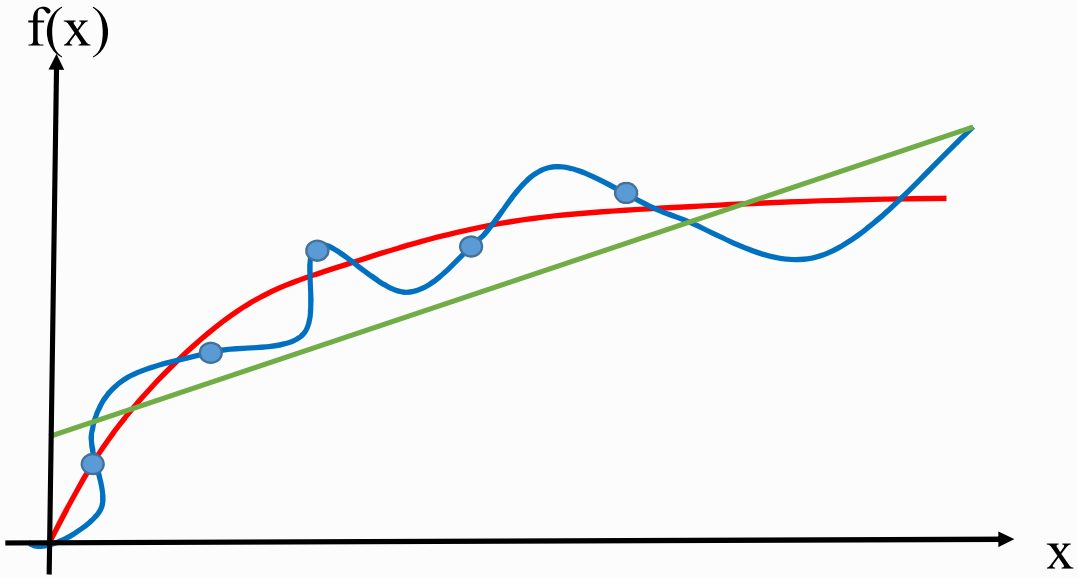


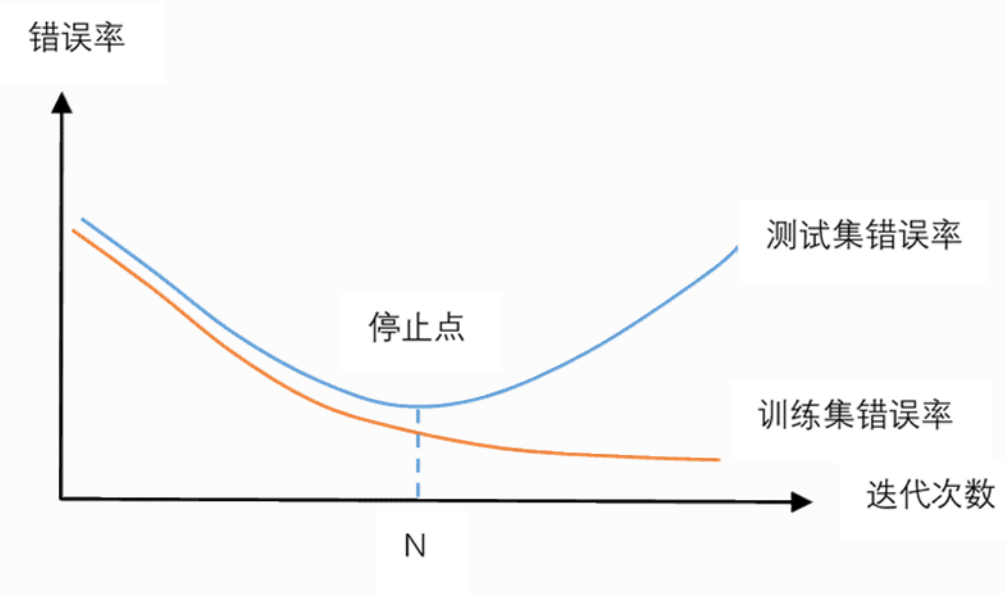


(a) 未归一化数据的梯度　(b) 归一化数据的梯度

三、深度学习的正则化（regularization）方法

➢什么是过拟合问题？



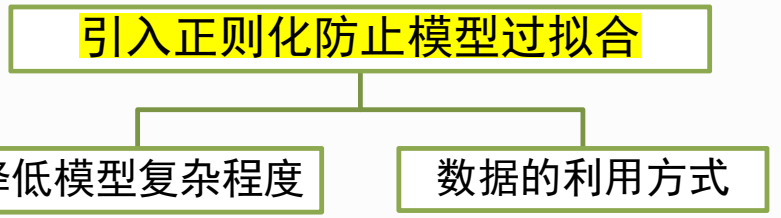


➢解决方案：引入正则化方法

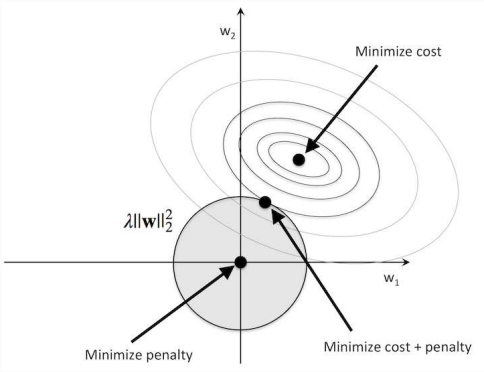
防止模型过拟合

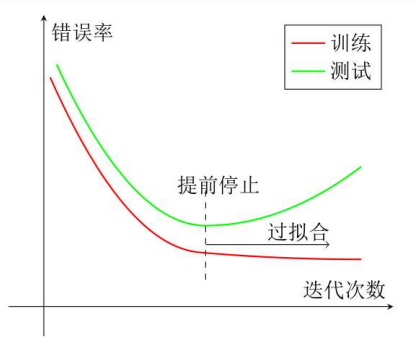
➢ 如何防止模型过拟合?

- ◆ 降低模型复杂度
  - L_1或L_2约束
  - 权值衰减
  - 舍弃法Dropout
- ◆ 数据的利用方式
  - 提前停止 Early stopping
  - 数据增强



L_1或L_2约束、权值衰减、 、Dropout　　提前停止、数据增强

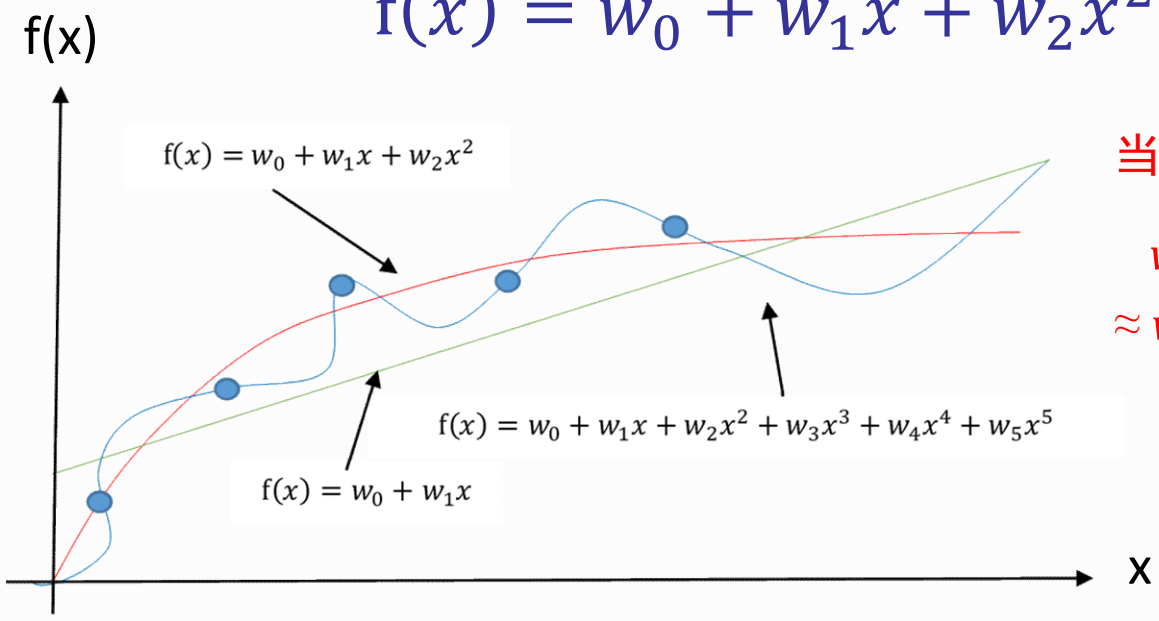




正则化项的作用：降低模型复杂度

曲线的多项式表示：

$$f(x) = w_0 + w_1 x + w_2 x^2 + \cdots w_n x^n$$



当w_3、w_4、w_5比较小时：

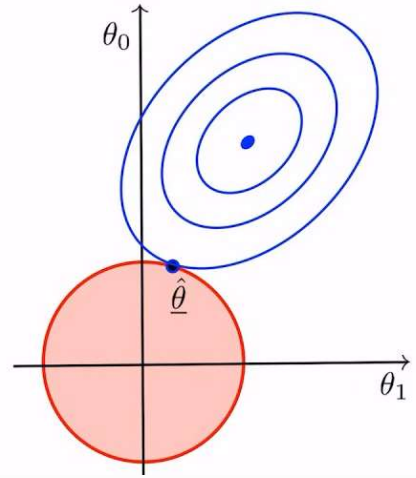
$$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 \approx w_0 + w_1 x + w_2 x^2$$

L_1或L_2约束

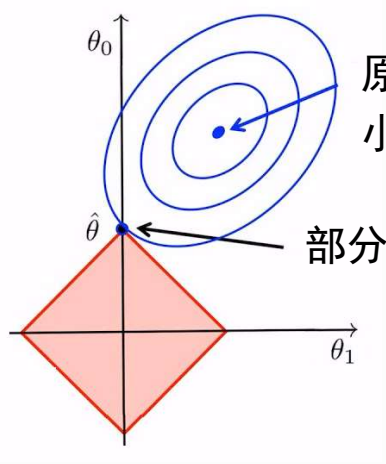
➢优化问题可以表示为：

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{m} \frac{1}{m} L\left(y^{*(i)}, f\left(\mathbf{x}^{(i)}, \theta\right)\right) + \lambda L_p(\theta)$$

L_p为范数，p取值通常为1或2，λ为正则化系数。



L_2范数



原有目标函数全局极小值点参数不为0

部分参数为0

L_1范数

向量范数

假设$\mathbf{x}$是n维向量，常见范数定义为：

L_p (p-范数) $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$ 向量元素绝对值p次方的1/p次幂

L_1 (1-范数) $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$ 向量元素绝对值之和

L_2 (2-范数) $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ 向量元素平方和的平方根

L_∞ (无穷-范数) $\|\mathbf{x}\|_\infty = \max_i |x_i|$ 向量元素绝对值最大值

L_0 (0-范数) 向量中非0元素的个数

权值衰减（Weight Decay）

- 参数更新过程中引入一个衰减系数λ

$$\theta_t = (1-\lambda)\theta_{t-1} - \eta g_t$$

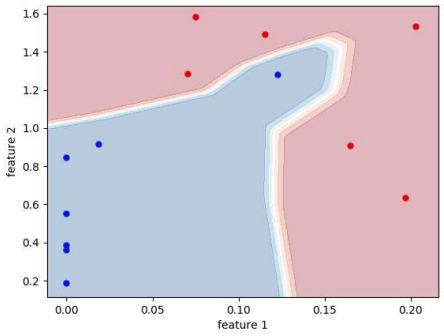
- 在标准的随机梯度下降中，权值衰减和L_2约束的效果相同

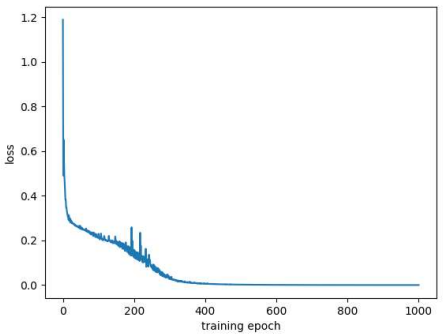
$$\begin{aligned}\theta_t &= (1-\lambda)\theta_{t-1} - \eta g_t \\ &= \theta_{t-1} - (\eta g_t + \lambda\theta_{t-1})\end{aligned}$$

- 在较为复杂的优化方法（比如Adam）中，权值衰减和L_2约束并不等价

正则化方法：L_2约束

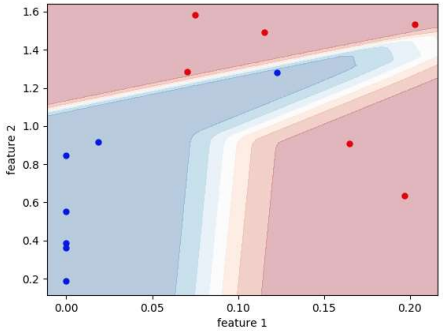
- 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000
  - L_2约束权值：0

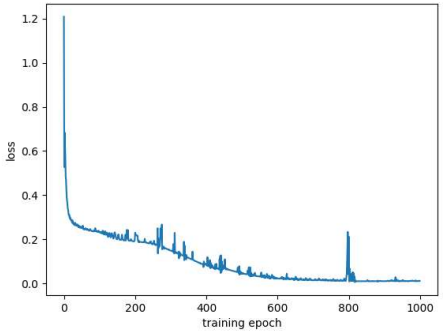




loss = 0.000

  - L_2约束权值：$1e^{-4}$

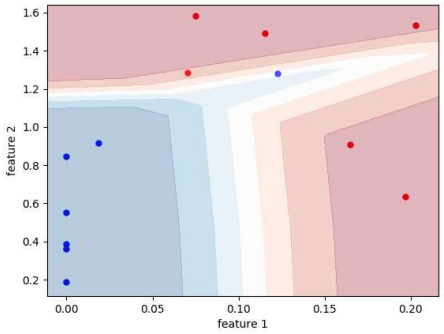


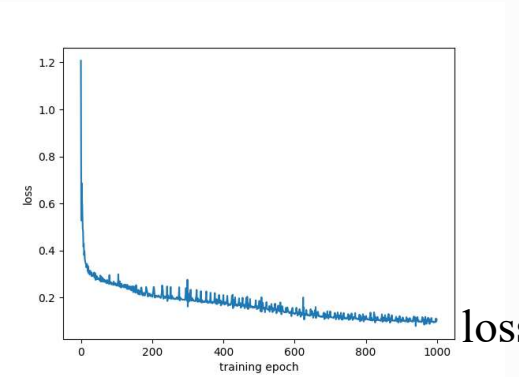


loss = 0.012

正则化方法：L_2约束

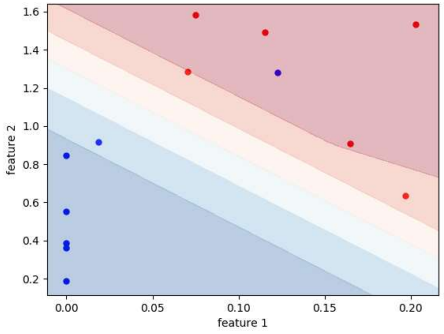
- 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000
  - L_2约束权值：$1e^{-3}$

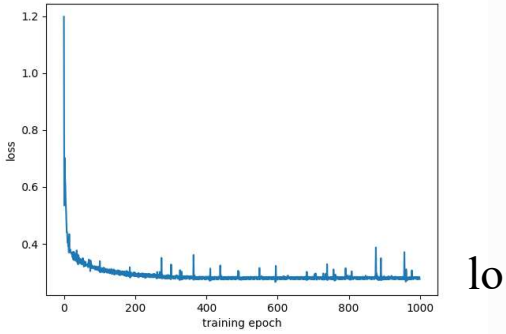




loss = 0.107

  - L_2约束权值：$5e^{-3}$



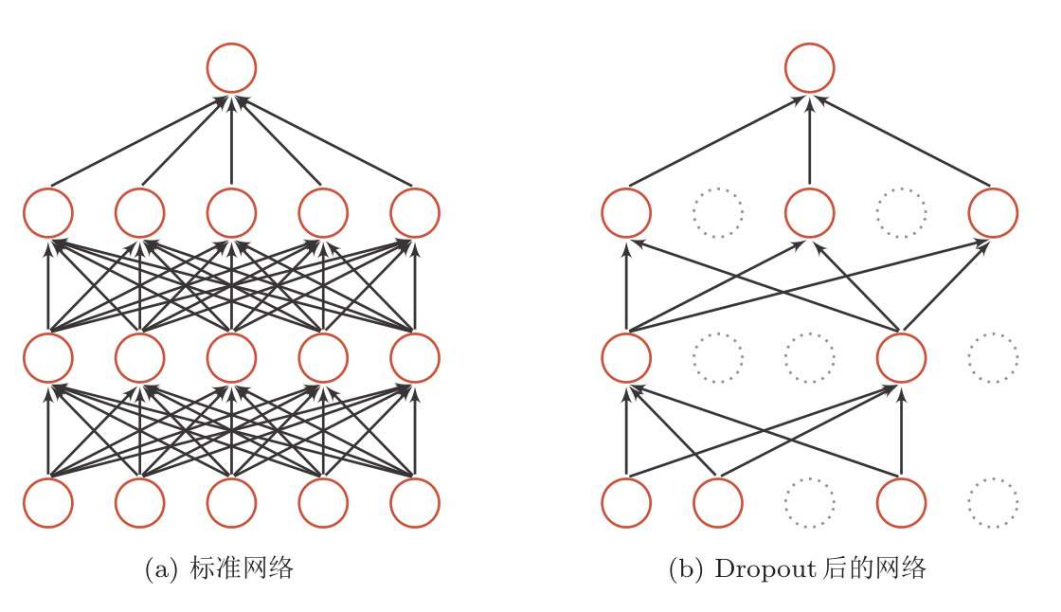


loss = 0.276

丢弃法（Dropout Method）

➢ Dropout

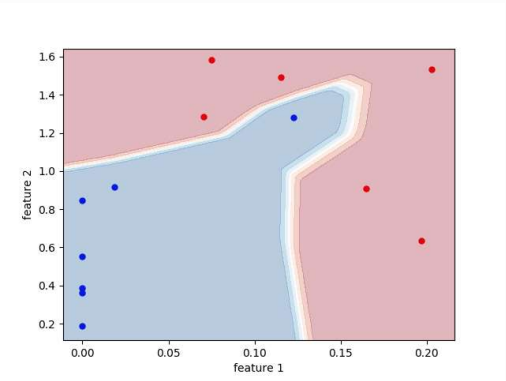
- ◆ 在训练过程中每次迭代时，将各层的输出节点以一定概率随机置为0

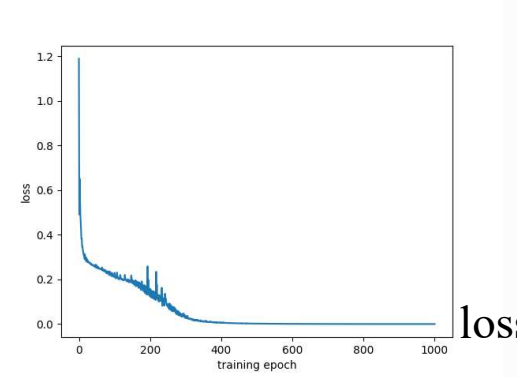


Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 2014, 15(1): 1929-1958.

正则化方法：Dropout

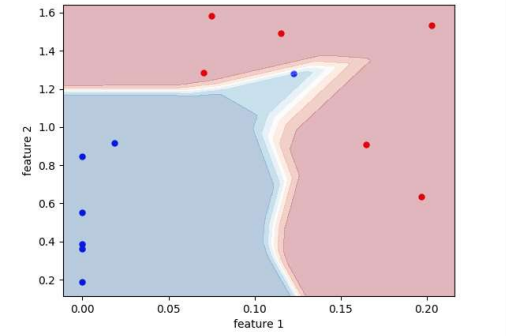
- 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000
  - Dropout概率：0

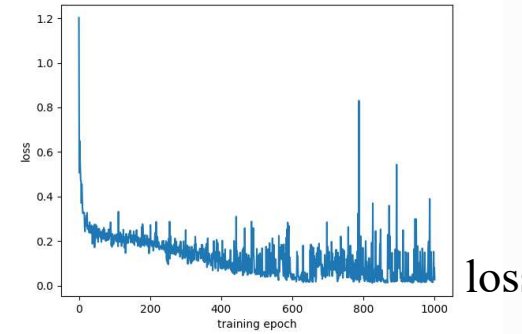




loss = 0.000

  - Dropout概率：0.1

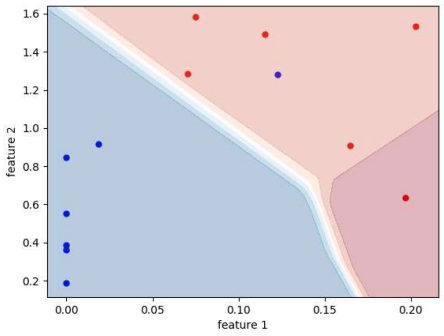


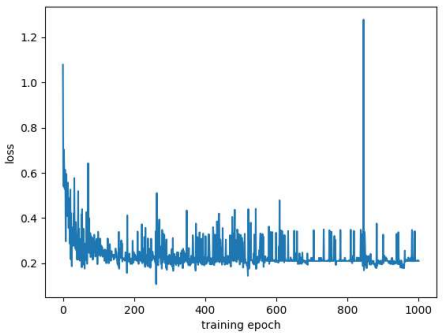


loss = 0.028

正则化方法：Dropout

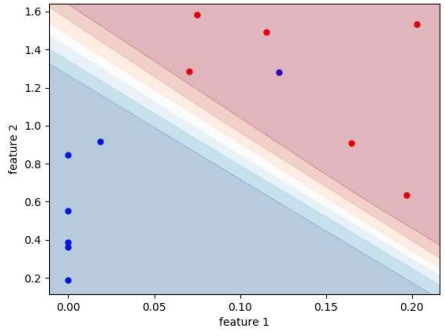
- 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000
  - Dropout概率：0.5

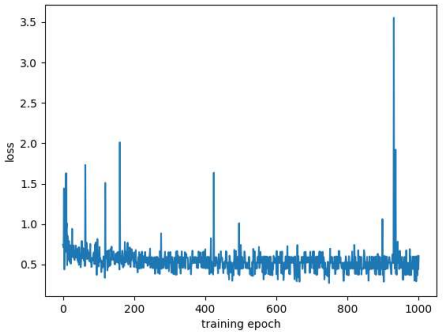




loss = 0.210

  - Dropout概率：0.9

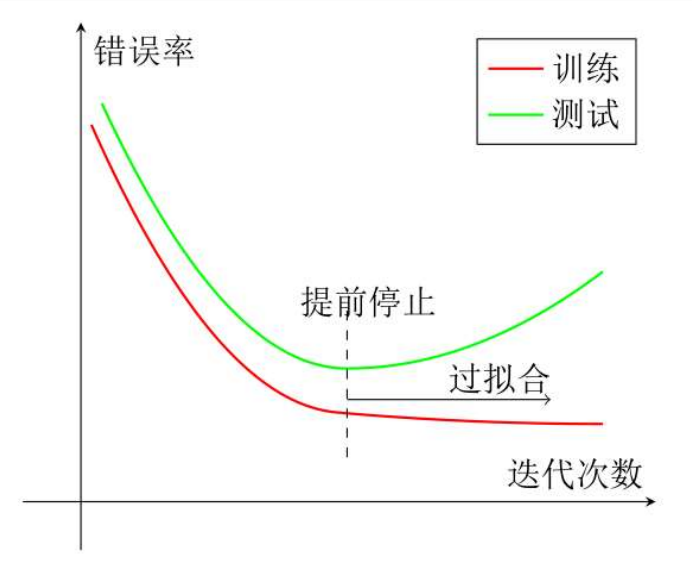




loss = 0.602

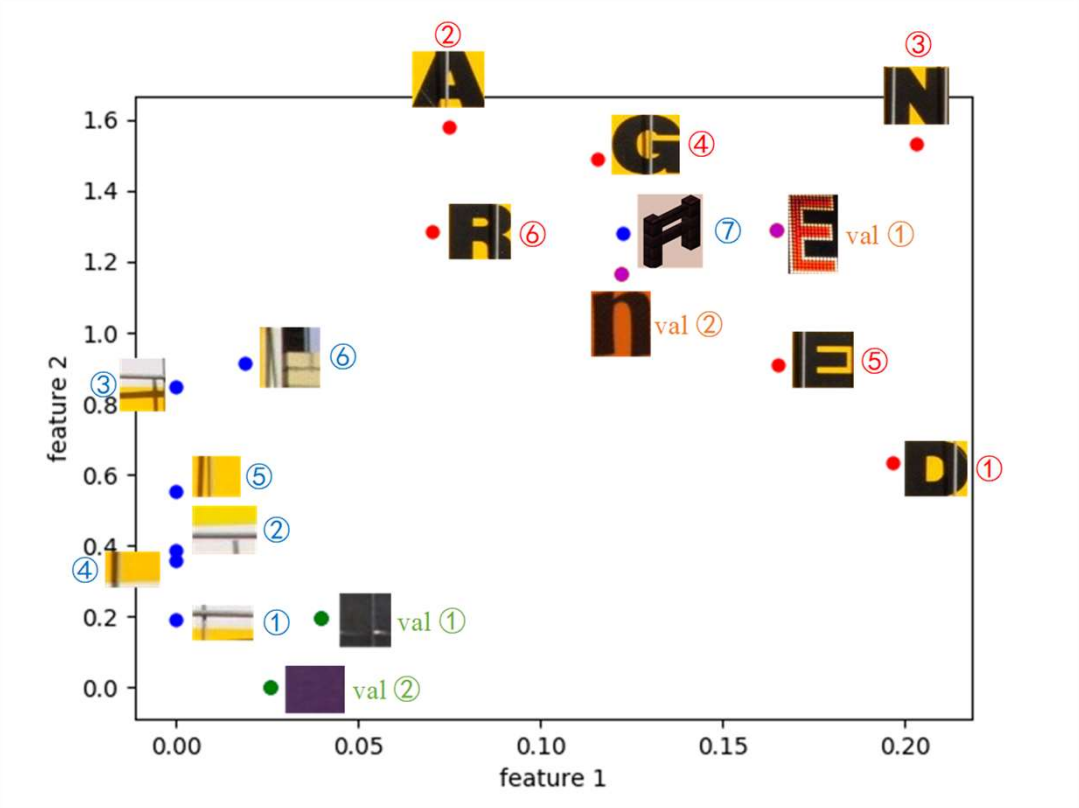
正则化方法：提前停止 (Early Stopping)

- 使用一个验证集（Validation Dataset）来测试每一次迭代的参数在验证集上是否最优
  - 如果在验证集上的错误率不再下降，就停止迭代



正则化方法：提前停止 (Early Stopping)

- 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000
  - 为“字符”和“背景”类别各加入两张图像作为验证集

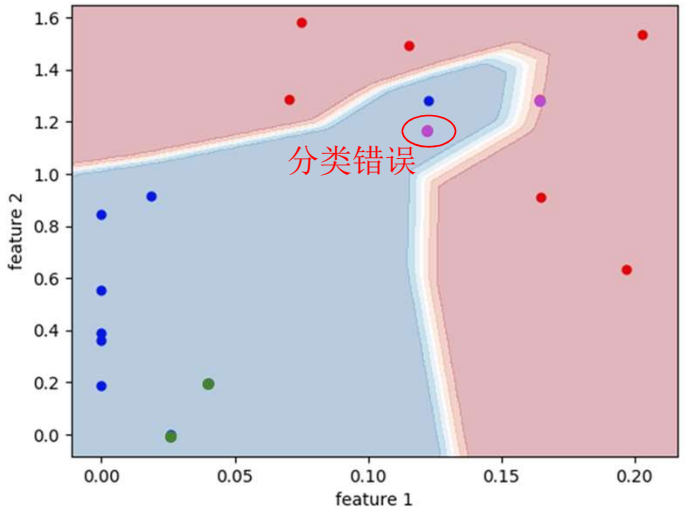


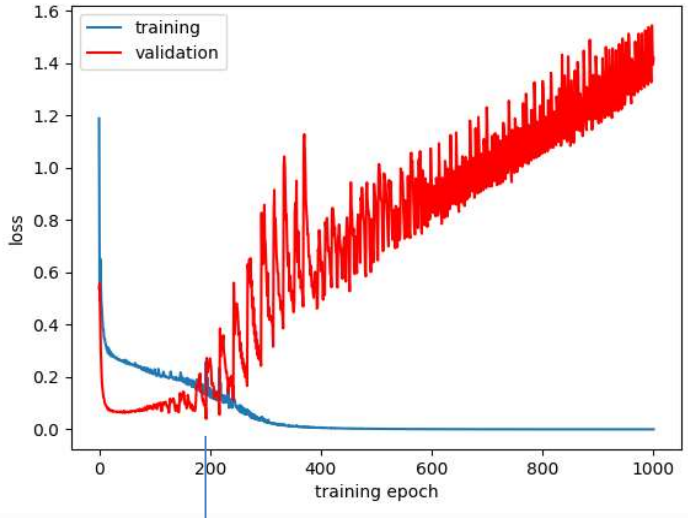
正则化方法：提前停止 (Early Stopping)

➢ 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000

◆ 无Early Stopping：训练1000轮

• 训练集loss = 0.000，验证集loss = 1.420





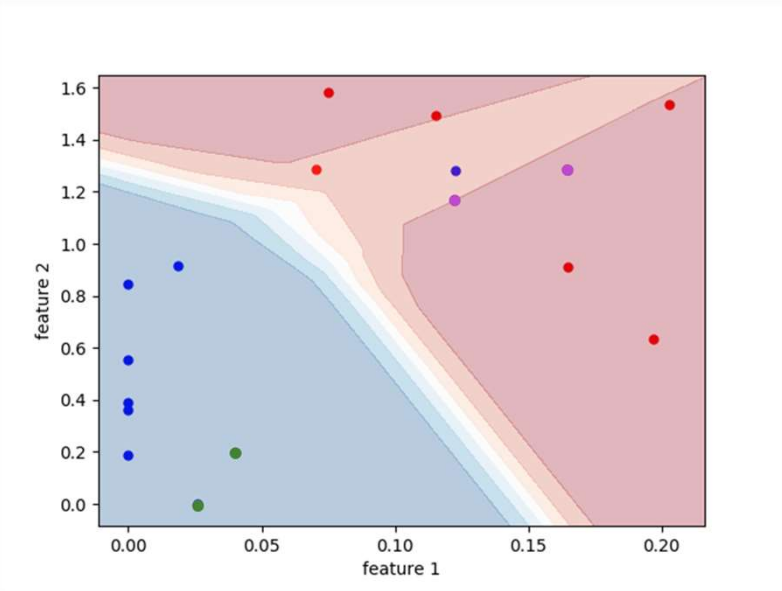
验证集loss在第194轮取得最小值

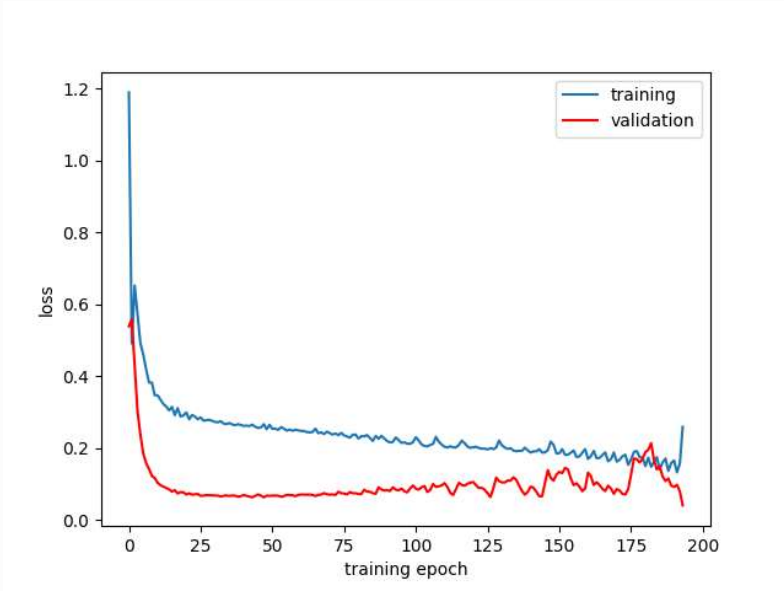
正则化方法：提前停止 (Early Stopping)

➢ 1层隐含层，节点数32，激活函数ReLU，优化器Adam，学习率0.03，训练轮数1000

◆ 有Early Stopping：训练194轮即停止

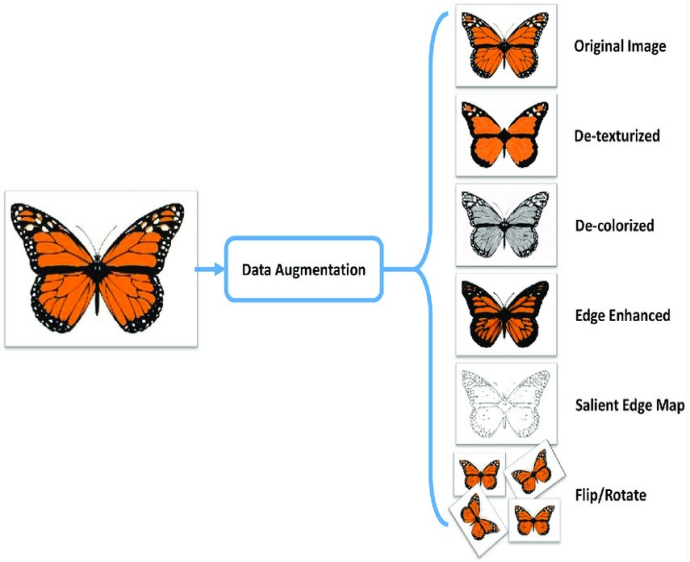
- 训练集loss = 0.259，验证集loss = 0.040





数据增强（Data Augmentation）

- 图像数据的增强主要是通过算法对图像进行处理增加数据的多样性，例如：
  - 旋转（Rotation）：将图像按顺时针或逆时针方向随机旋转一定角度
  - 翻转（Flip）：将图像沿水平或垂直方法随机翻转一定角度
  - 缩放（Zoom In/Out）：将图像放大或缩小一定比例
  - 平移（Shift）：将图像沿水平或垂直方法平移一定步长
  - 加噪声（Noise）：加入随机噪声



小结

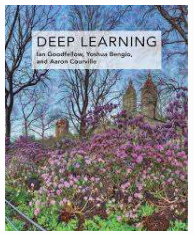
- 深度学习是一种非线性建模方法
  - 常见网络架构
  - 优化方法
    - 动态学习率、Adam算法
  - 正则化方法
    - 对权值施加L_2约束
    - 权值衰减
    - 舍弃法Dropout
    - 提前停止Early-stopping
    - 数据增强

本周需要掌握的内容

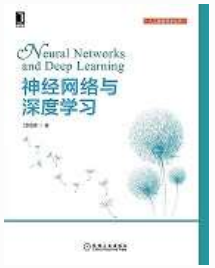
- 原理
  - 深度学习常见网络结构
  - 深度学习模型的训练
    - 数据结构与模型定义
    - 优化方法 (SGD，动量法， Adam)
    - 正则化方法（对权值施加 L_2 约束、权值衰减、Dropout，提前中止，数据增强）
- 数学基础
  - 矩阵及向量求导
  - 向量范数
- 编程实践
  - 网络学堂“课程文件”→“编程实践”栏目中的“第四周教程”
    - tutorial-4.ipynb

参考书

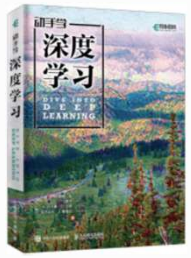
- Ian Goodfellow and Yoshua Bengio et al., Deep Learning, MIT Press, 2016.
  http://www.deeplearningbook.org
  - Chapter 7 Regularization for Deep Learning
  - Chapter 8 Optimization for Training Deep Models



- 邱锡鹏，神经网络与深度学习，机械工业出版社，2020年
  https://nndl.github.io/
  - 第7章 网络优化与正则化
  - 附录 B.3矩阵微积分、B.4 常见函数的导数



- 动手学深度学习(PyTorch版)
  - 第三章第3.11-3.15节，第四章，第七章
  - https://tangshusen.me/Dive-into-DL-PyTorch/#/
  - https://github.com/ShusenTang/Dive-into-DL-PyTorch



谢谢大家！