网络优化与正则化

https://nndl.github.io/

深度学习
~~机器学习~~的矛与盾

优化
经验风险最小

正则化
降低模型复杂度

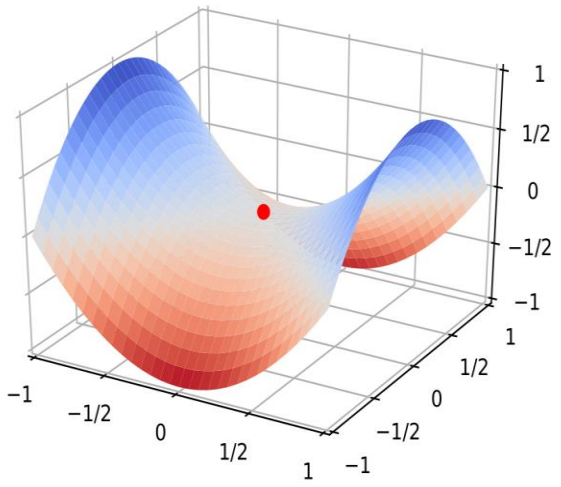
网络优化

网络优化的难点

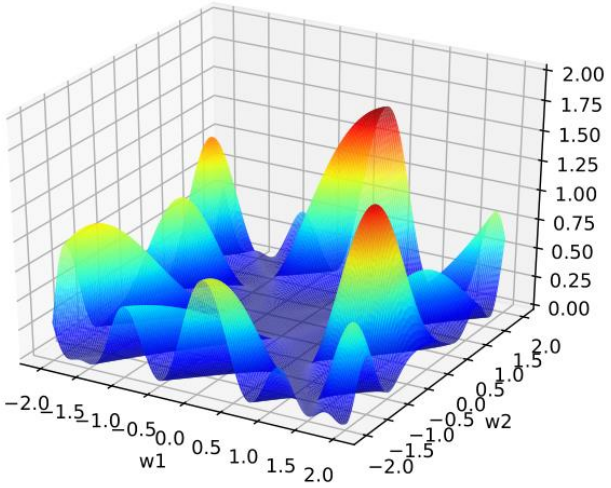
- 结构差异大
  - 没有通用的优化算法
- 非凸优化问题
  - 参数初始化
  - 逃离局部最优

高维空间的非凸优化问题

- 鞍点

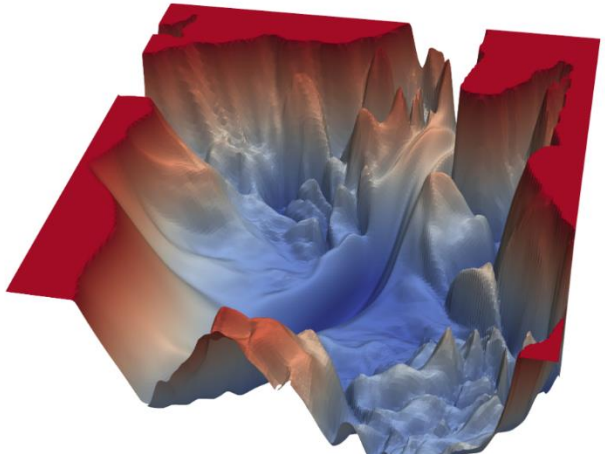


- 平摊底部

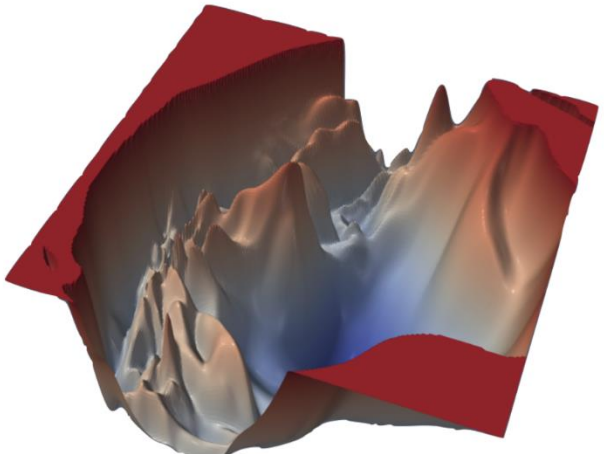


VISUALIZING THE LOSS LANDSCAPE OF NEURAL NETS

VGG-56

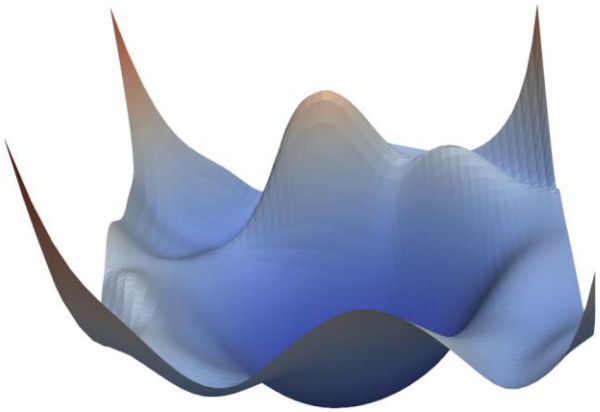


VGG-110

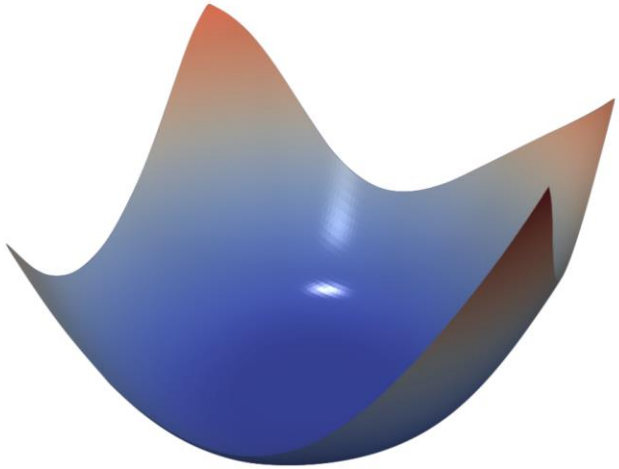


without skip connect

Renset-56



Densenet-121



with skip connecti

优化算法：随机梯度下降

算法 2.1: 随机梯度下降法

输入: 训练集 $\mathcal{D}=\{(\mathbf{x}^{(n)},y^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 α

随机初始化 θ;

repeat

  对训练集 $\mathcal{D}$ 中的样本随机重排序;

  for $n=1\cdots N$ **do**

    从训练集 $\mathcal{D}$ 中选取样本 $(\mathbf{x}^{(n)},y^{(n)})$;

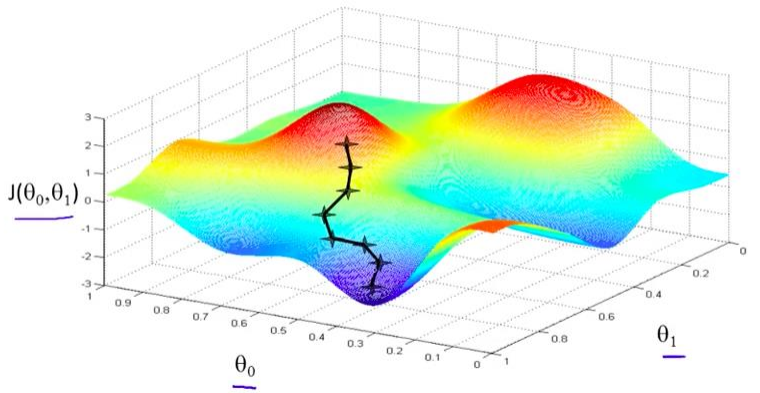
    // 更新参数

    $\theta\leftarrow\theta-\alpha\dfrac{\partial\mathcal{L}\left(\theta;x^{(n)},y^{(n)}\right)}{\partial\theta}$;

  end

until 模型 $f(\mathbf{x},\theta)$ 在验证集 $\mathcal{V}$ 上的错误率不再下降;

输出: θ



优化算法：
小批量随机梯度下降 MiniBatch

- 选取K个训练样本$\{\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\}_{k=1}^{K}$，计算偏导数

$$\mathbf{g}_t(\theta) = \frac{1}{K} \sum_{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \in \mathcal{I}_t} \frac{\partial \mathcal{L}(\mathbf{y}^{(k)}, f(\mathbf{x}^{(k)}, \theta))}{\partial \theta}$$

- 定义梯度

$$\mathbf{g}_t \triangleq \mathbf{g}_t(\theta_{t-1})$$

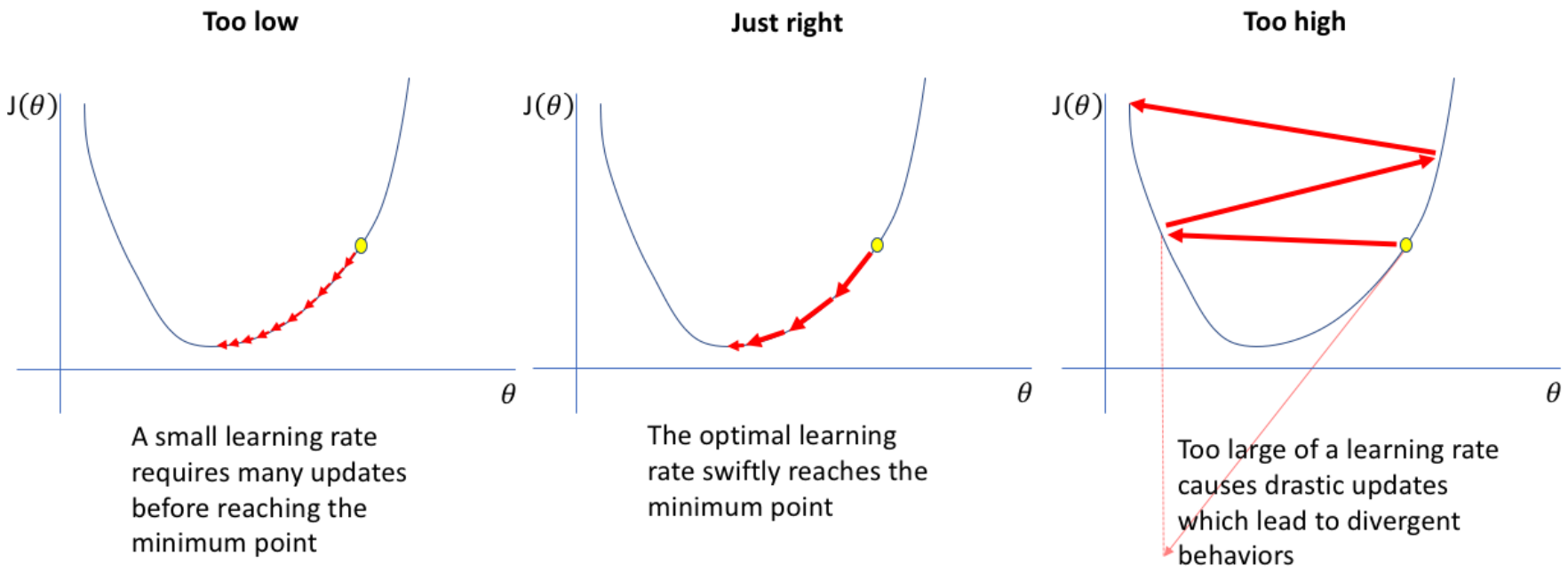
- 更新参数

$$\theta_t \leftarrow \theta_{t-1} - \alpha \mathbf{g}_t$$

其中$\alpha > 0$为学习率

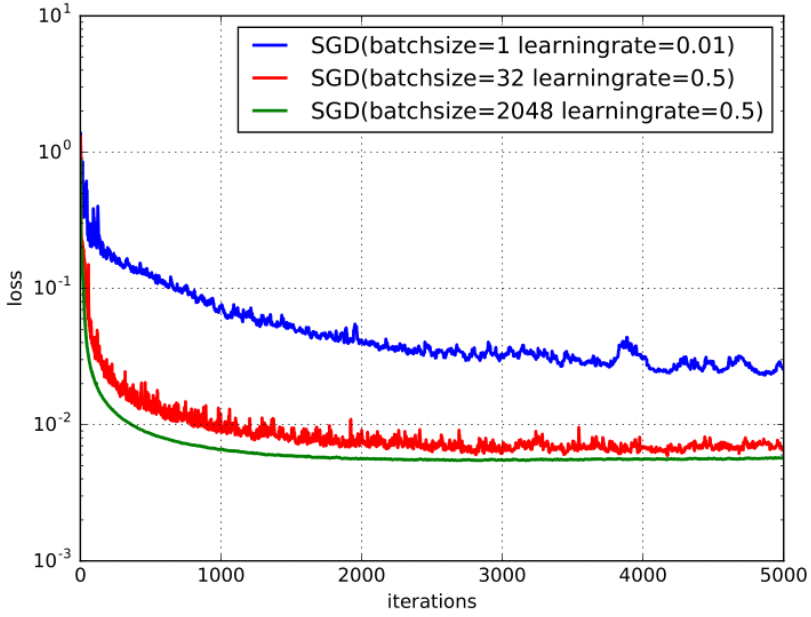
几个关键因素：
- 小批量样本数量
- 梯度
- 学习率

学习率的影响

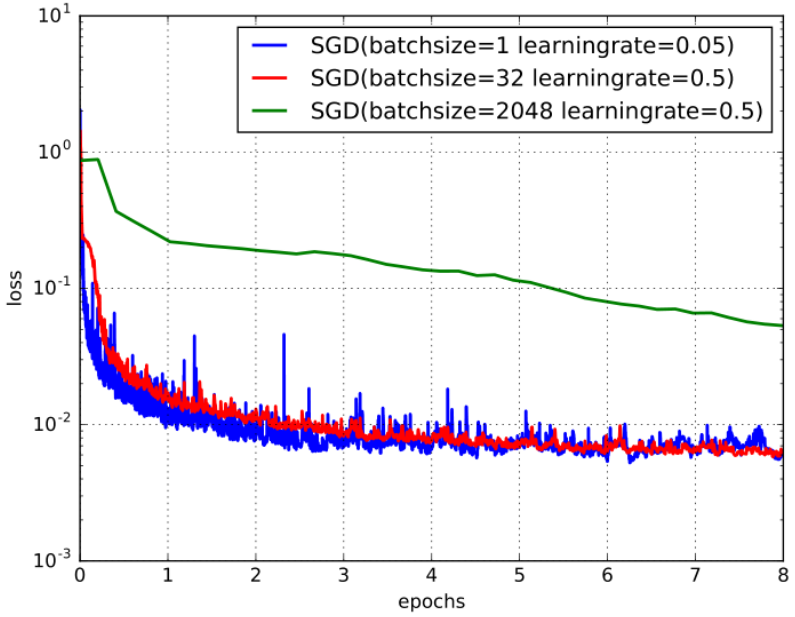


https://www.jeremyjordan.me/nn-learning-rate/

批量大小的影响



(a) 按每次小批量更新的损失变化



(b) 按整个数据集迭代的损失变化

小批量梯度下降中，每次选取样本数量对损失下降的影响。

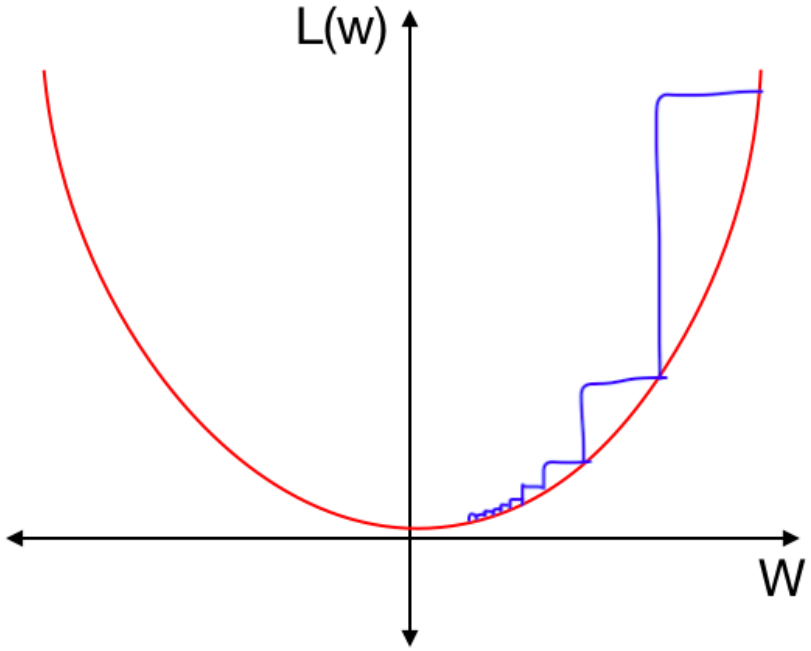
如何改进?

Reference:
1. An overview of gradient descent optimization algorithms
2. Optimizing the Gradient Descent

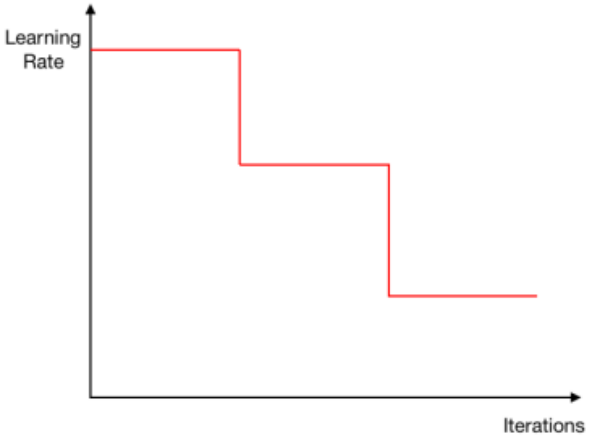
- 学习率
  - 学习率衰减
  - Adagrad
  - RMSprop
  - Adadelta
- 梯度
  - Momentum
  - Nesterov accelerated gradient
  - 梯度截断
- 综合方法
  - Adam≈动量法+RMSprop
  - Nadam

Adam is better choice!

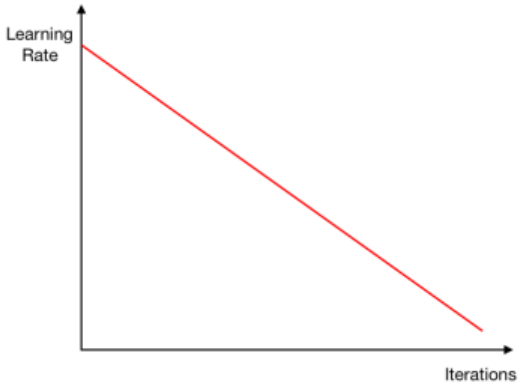
学习率衰减



学习率衰减



梯级衰减（step decay）



线性衰减（Linear Decay）

学习率衰减

假设初始化学习率为α_0，在第t次迭代时的学习率α_t。常用的衰减方式为可以设置为按迭代次数进行衰减。比如逆时衰减（inverse time decay）

$$\alpha_t = \alpha_0 \frac{1}{1 + \beta \times t}, \tag{7.5}$$

或指数衰减（exponential decay）

$$\alpha_t = \alpha_0 \beta^t, \tag{7.6}$$

或自然指数衰减（natural exponential decay）

$$\alpha_t = \alpha_0 \exp(-\beta \times t), \tag{7.7}$$

其中β为衰减率，一般取值为0.96。

学习率衰减

- 动态衰减率
  - Adagrad
  - RMSprop
  - Adadelta

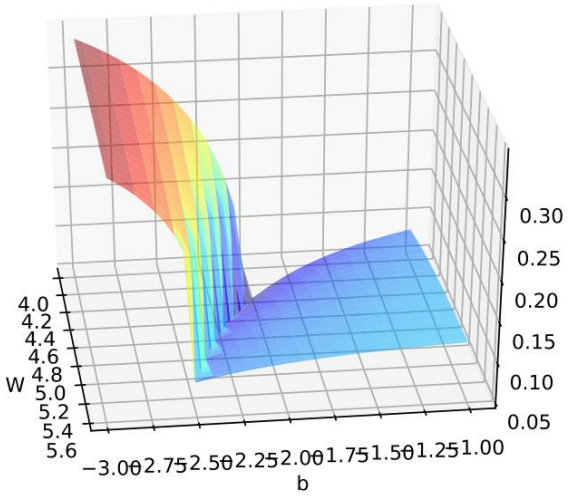
梯度截断

- 梯度截断是一种比较简单的启发式方法，把梯度的模限定在一个区间，当梯度的模小于或大于这个区间时就进行截断。
  - 按值截断

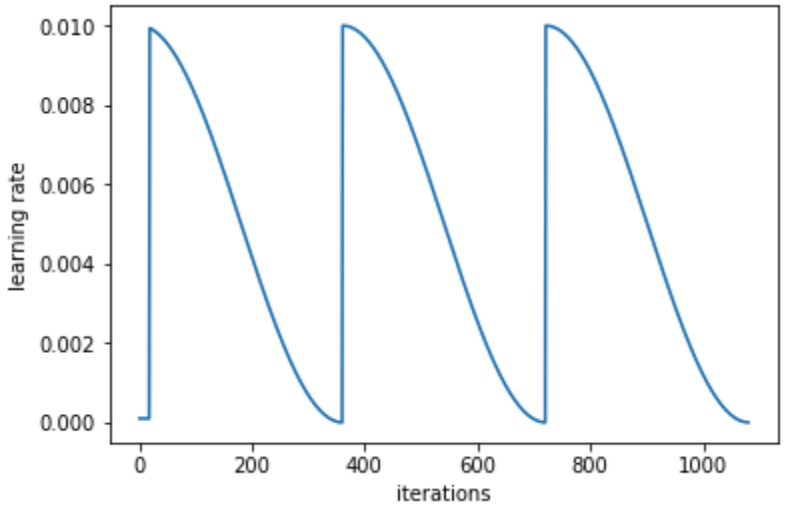
$$\mathbf{g}_t = \max(\min(\mathbf{g}_t, b), a)$$

  - 按模截断

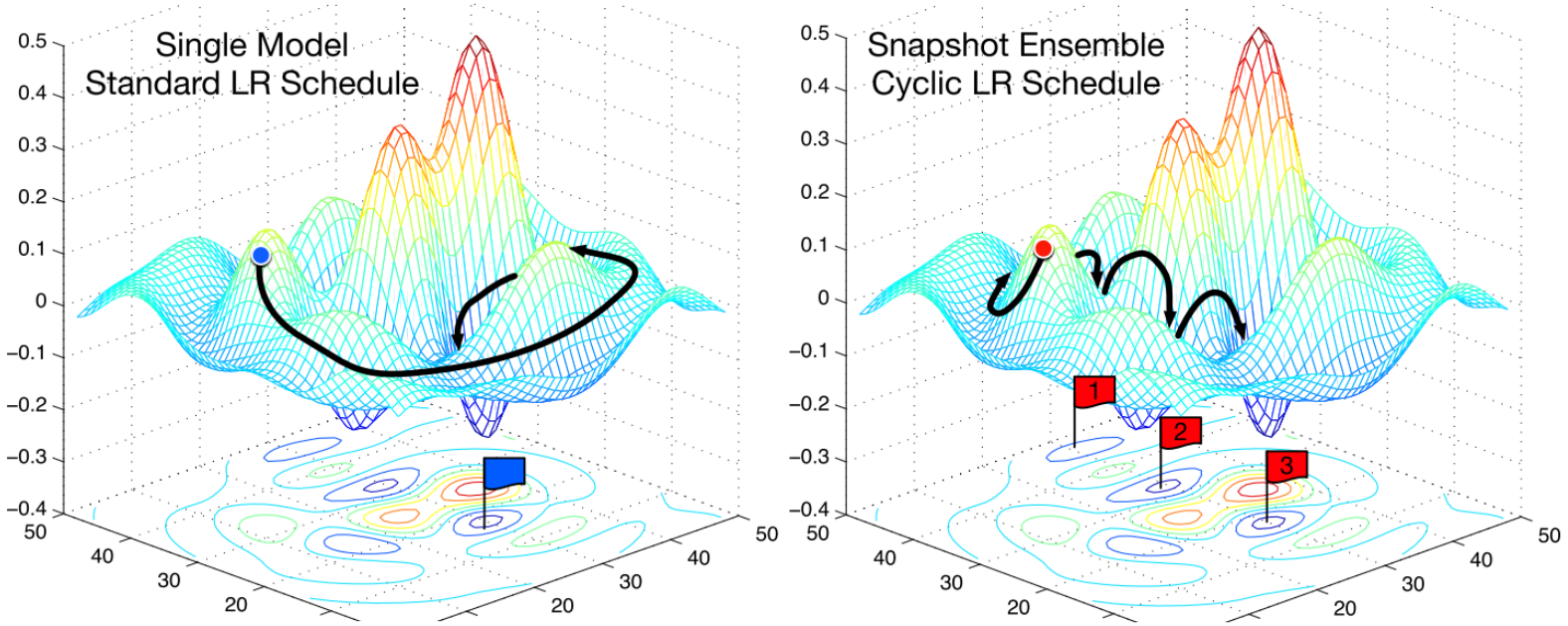
$$\mathbf{g}_t = \frac{b}{\|\mathbf{g}_t\|}\mathbf{g}_t$$



Cyclical Learning Rates

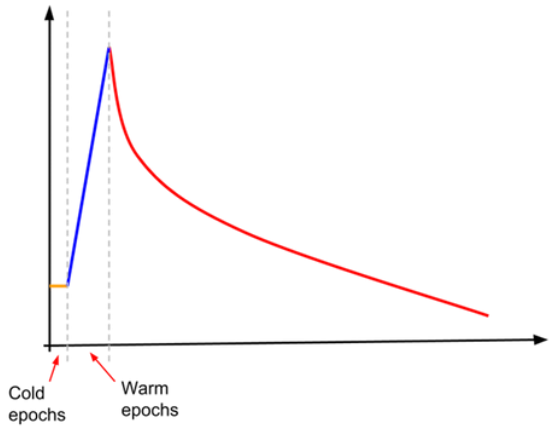


Cyclical Learning Rates



Others

- Don't Decay the Learning Rate, Increase the Batch Size
  - https://openreview.net/pdf?id=B1Yy1BxCZ
- Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour
  - Warmup
  - https://arxiv.org/abs/1706.02677



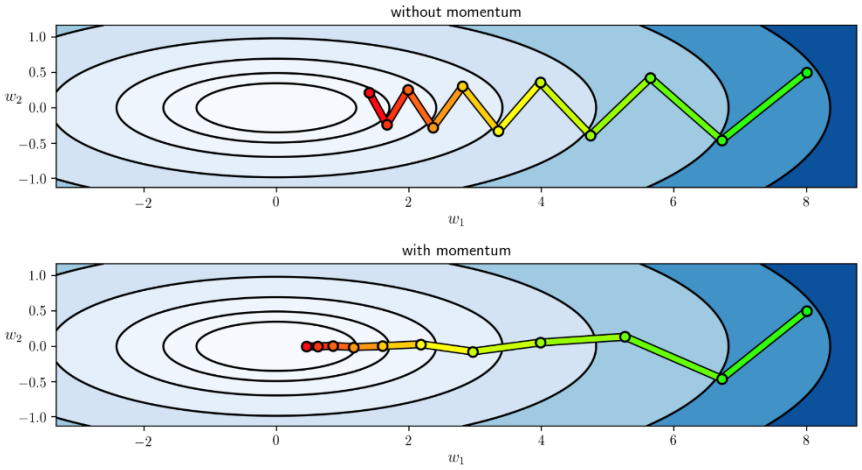
梯度方向优化

- 动量法（Momentum Method）
  - 用之前积累动量来替代真正的梯度。每次迭代的梯度可以看作是加速度。

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，

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \alpha\mathbf{g}_t, = -\alpha\sum_{\tau=1}^{t}\beta^{t-\tau}\mathbf{g}_\tau, \tag{7.15}$$

其中ρ为动量因子，通常设为0.9，α为学习率。

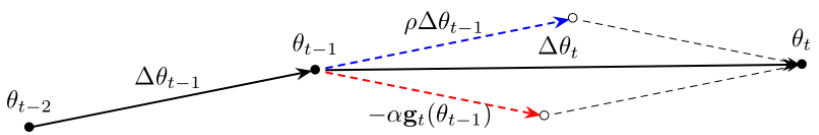


https://www.quora.com/What-exactly-is-momentum-in-machine-learning

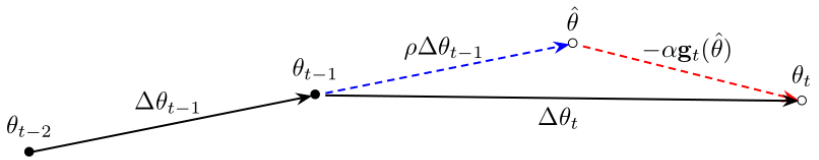
梯度方向优化

- Nesterov加速梯度

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \alpha\mathbf{g}_t(\theta_{t-1} + \rho\Delta\theta_{t-1})$$



(a) 动量法



(b) Nesterov 加速梯度

梯度方向优化+自适应学习率

- Adam算法≈动量法+RMSprop
  - 先计算两个移动平均

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)\mathbf{g}_t,$$
$$G_t = \beta_2 G_{t-1} + (1-\beta_2)\mathbf{g}_t \odot \mathbf{g}_t,$$

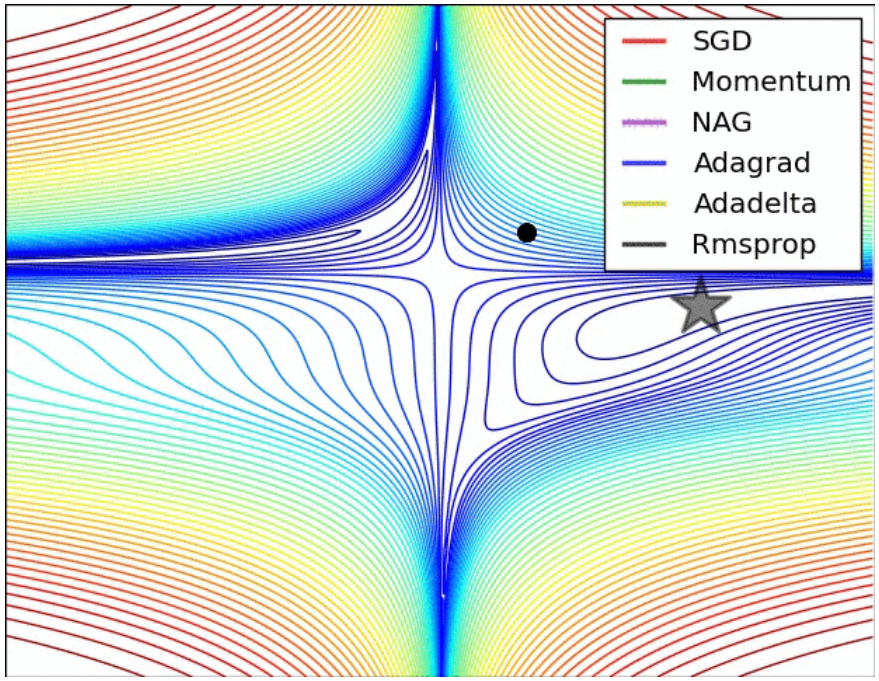
  - 偏差修正

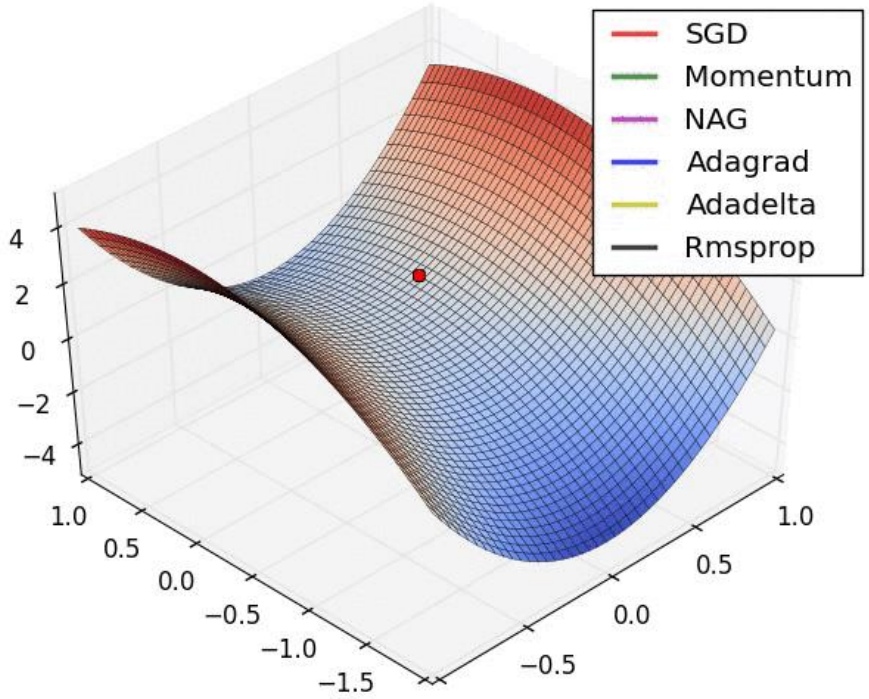
$$\hat{M}_t = \frac{M_t}{1-\beta_1^t},$$
$$\hat{G}_t = \frac{G_t}{1-\beta_2^t}.$$

  - 更新

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{\hat{G}_t + \epsilon}}\hat{M}_t$$

优化


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

鞍点

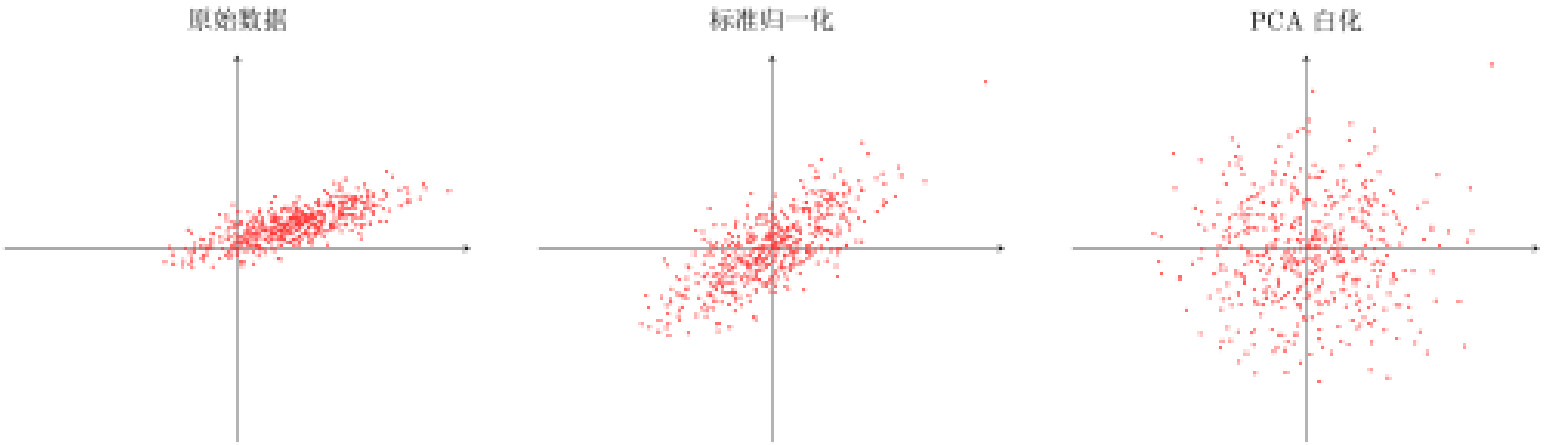
参数初始化

- 参数不能初始化为0！为什么？
  - 对称权重问题！
- Gaussian初始化方法
  - Gaussian初始化方法是最简单的初始化方法，参数从一个固定均值（比如0）和固定方差（比如0.01）的Gaussian分布进行随机初始化。
- Xavier初始化
  - 参数可以在区间[−r,r]内采用均匀分布进行初始化。

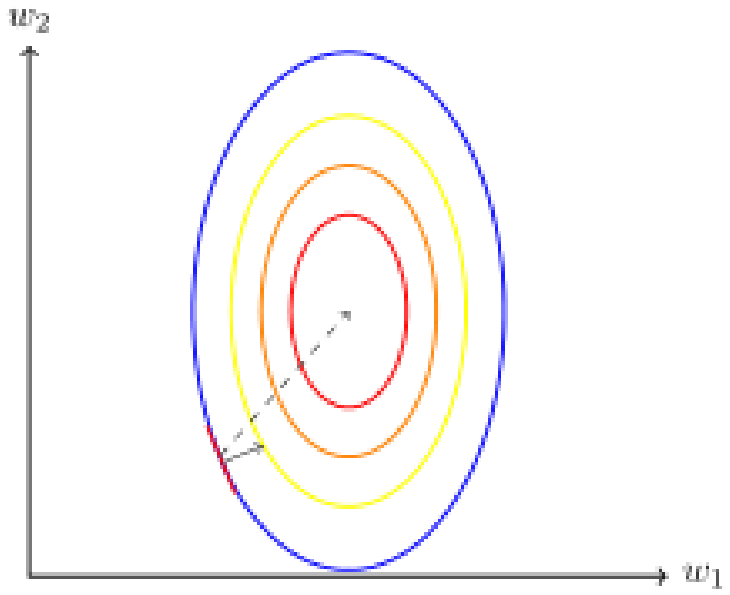
$$r = \sqrt{\frac{6}{n^{l-1} + n^{1}}},$$

数据预处理

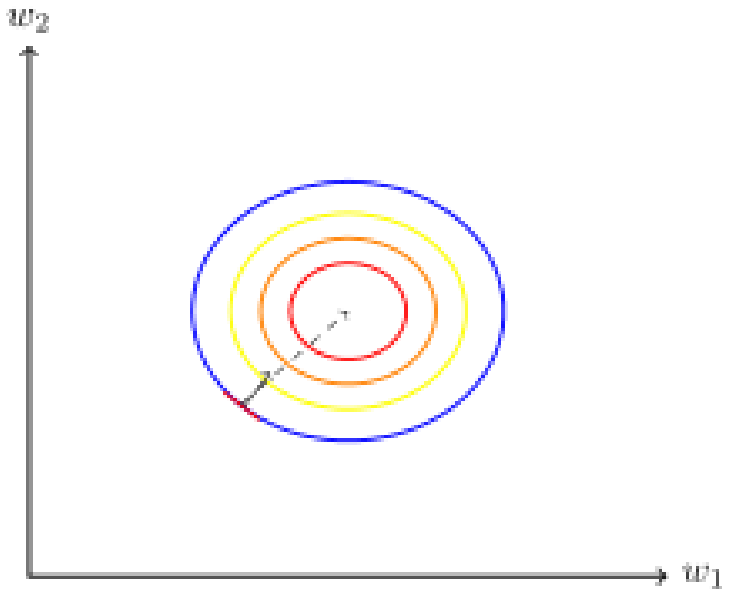
- 数据归一化
  - 标准归一化
  - 缩放归一化
  - PCA



数据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

逐层归一化

逐层归一化

- 内部协变量偏移（Internal Covariate Shift）

- 归一化方法
  - 批量归一化（Batch Normalization，BN）
  - 层归一化（Layer Normalization）
  - 权重归一化（Weight Normalization）
  - 局部响应归一化（Local Response Normalization，LRN）

批量归一化

对于一个深层神经网络，令第l层的净输入为$\mathbf{z}^{(l)}$，神经元的输出为$\mathbf{a}^{(l)}$，即

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}) = f(W\mathbf{a}^{(l-1)} + \mathbf{b}), \tag{7.42}$$

其中$f(\cdot)$是激活函数，W和$\mathbf{b}$是可学习的参数。

- 给定一个包含K 个样本的小批量样本集合，计算均值和方差

$$\mu_{\mathcal{B}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{z}^{(k,l)},$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{K}\sum_{k=1}^{K}(\mathbf{z}^{(k,l)} - \mu_{\mathcal{B}}) \odot (\mathbf{z}^{(k,l)} - \mu_{\mathcal{B}}).$$

- 批量归一化

$$\hat{\mathbf{z}}^{(l)} = \frac{\mathbf{z}^{(l)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \odot \gamma + \beta$$

$$\triangleq \mathrm{BN}_{\gamma,\beta}(\mathbf{z}^{(l)}),$$

超参数优化

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

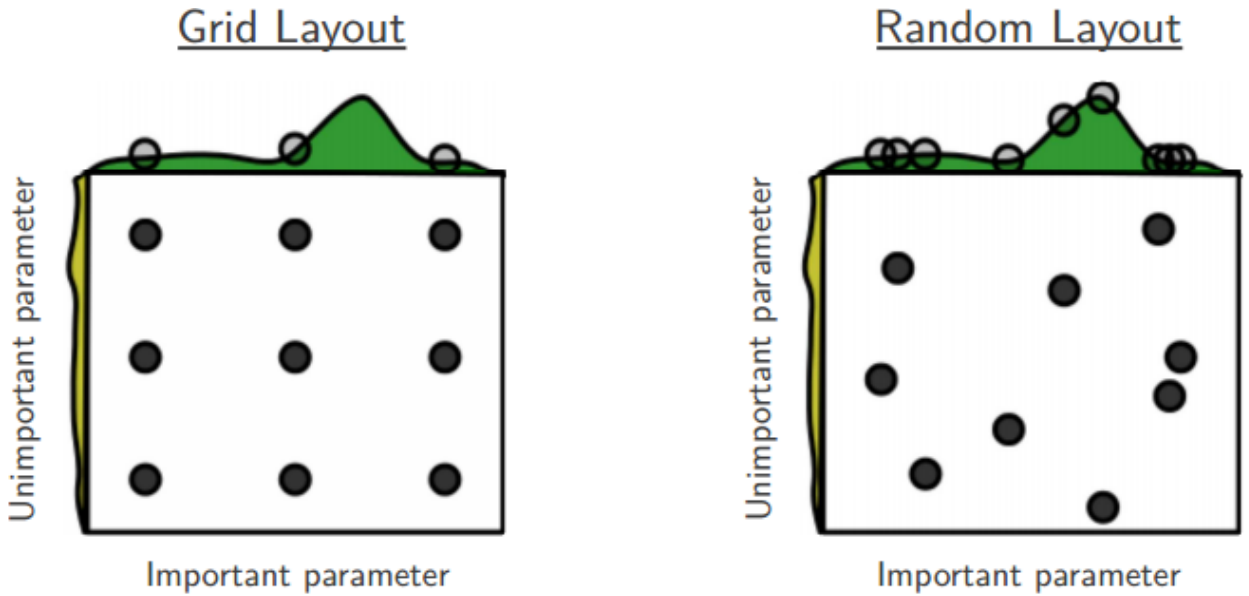
超参数优化

- 网格搜索（Grid Search）
  - 假设总共有K 个超参数，第k个超参数的可以取m_k 个值。
  - 如果参数是连续的，可以将参数离散化，选择几个“经验”值。比如学习率α，我们可以设置

$$\alpha \in \{0.01, 0.1, 0.5, 1.0\}.$$

  - 这些超参数可以有 $m_1 \times m_2 \times \cdots \times m_K$ 个取值组合。

超参数优化



Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

超参数优化

- 贝叶斯优化
- 动态资源分配
- 神经架构搜索

网络正则化

重新思考泛化性

- 神经网络
  - 过度参数化
  - 拟合能力强

⇩

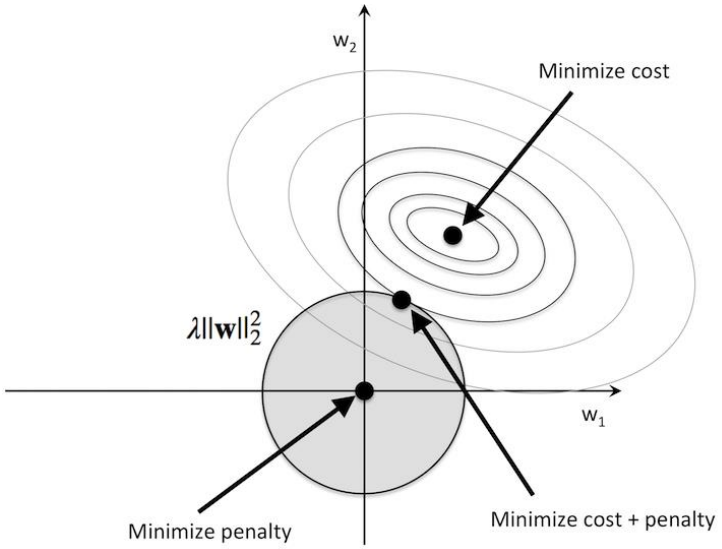
~~泛化性差~~

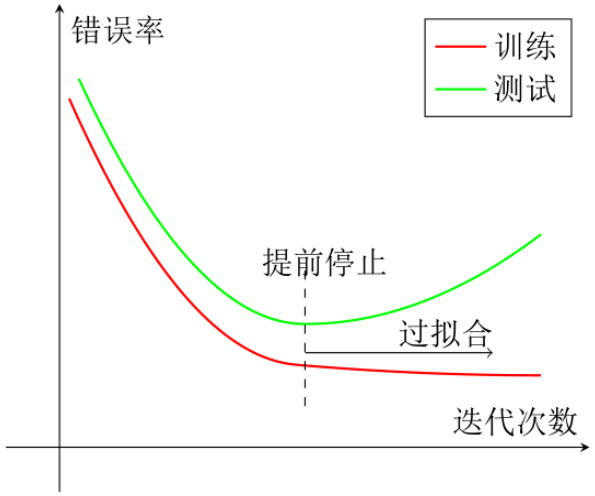
Zhang C, Bengio S, Hardt M, et al. Understanding deep learning requires rethinking generalization[J]. arXiv preprint arXiv:1611.03530, 2016.

正则化（regularization）

所有损害优化的方法都是正则化。

增加优化约束	干扰优化过程
L1/L2约束、数据增强	权重衰减、随机梯度下降、提前停止





正则化

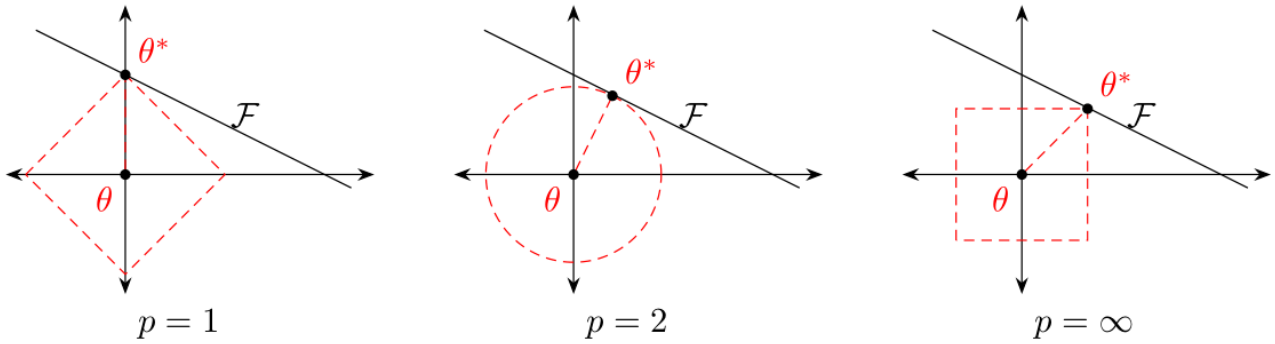
- 如何提高神经网络的泛化能力
  - ℓ_1和ℓ_2正则化
  - early stop
  - 权重衰减
  - SGD
  - Dropout
  - 数据增强

ℓ_1和ℓ_2正则化

- 优化问题可以写为

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left(y^{(n)}, f(\mathbf{x}^{(n)}, \theta)\right) + \lambda \ell_p(\theta)$$

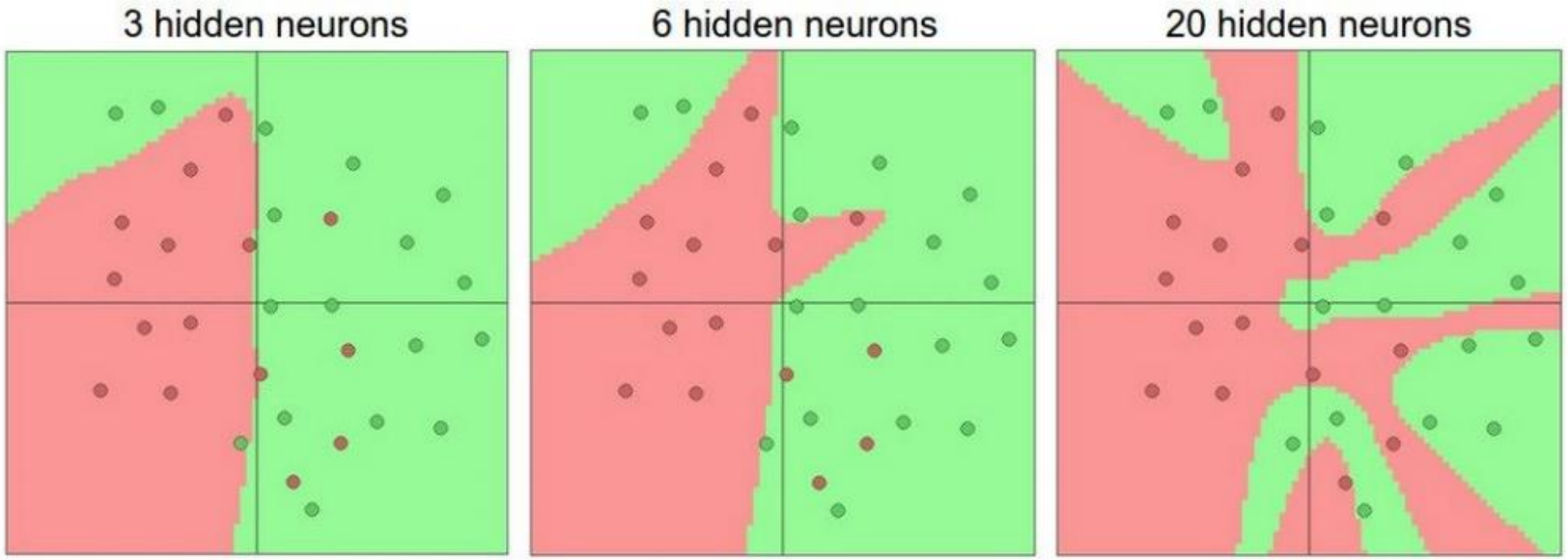
  - ℓ_p 为范数函数，p的取值通常为{1,2}代表ℓ_1和ℓ_2范数，λ为正则化系数。



神经网络示例

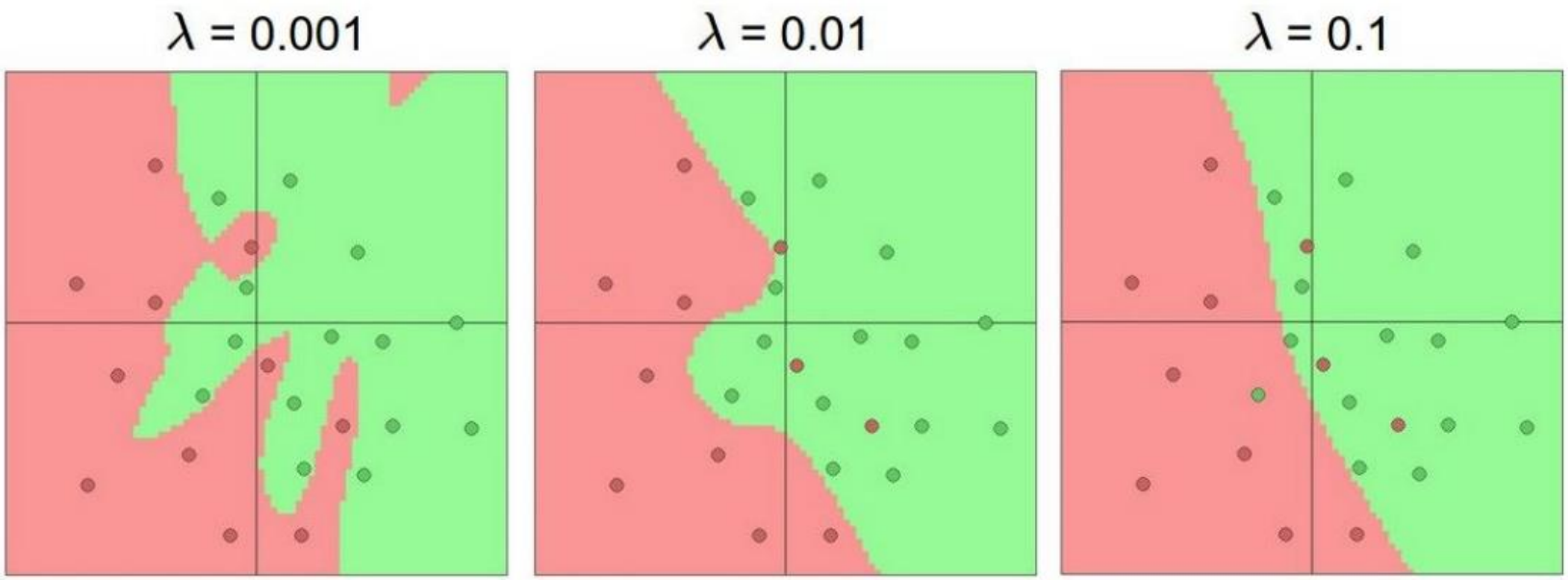
http://playground.tensorflow.org/

- 隐藏层的不同神经元个数



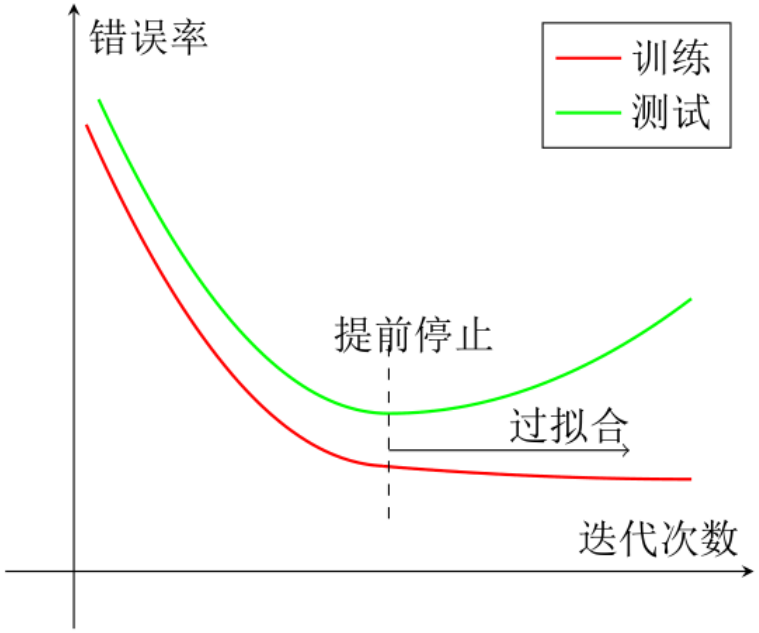
神经网络示例

- 不同的正则化系数



提前停止

- 我们使用一个验证集（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。



权重衰减（Weight Decay）

- 在每次参数更新时，引入一个衰减系数w。

$$\theta_t \leftarrow (1-w)\theta_{t-1} - \alpha \mathbf{g}_t$$

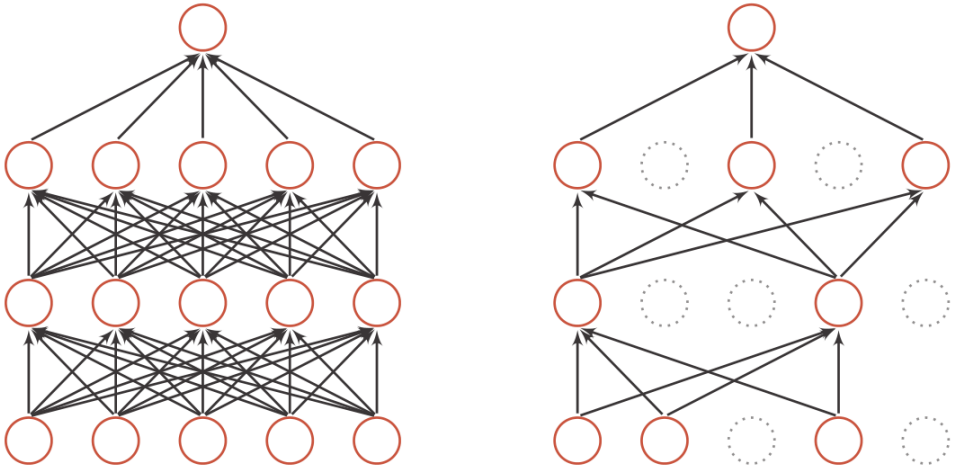
- 在标准的随机梯度下降中，权重衰减正则化和ℓ_2正则化的效果相同。
- 在较为复杂的优化方法（比如Adam）中，权重衰减和ℓ_2正则化并不等价。

丢弃法（Dropout Method）

- 对于一个神经层$y = f(Wx + b)$，引入一个丢弃函数$d(\cdot)$使得$y = f(Wd(x) + b)$。

$$d(\mathbf{x}) = \begin{cases} \mathbf{m} \odot \mathbf{x} & \text{当训练阶段时} \\ p\mathbf{x} & \text{当测试阶段时} \end{cases}$$

  - 其中$m \in \{0,1\}^d$是丢弃掩码（dropout mask），通过以概率为p的贝努力分布随机生成。



(a) 标准网络　　(b) Dropout后的网络

Dropout意义

- 集成学习的解释
  - 每做一次丢弃，相当于从原始的网络中采样得到一个子网络。如果一个神经网络有n个神经元，那么总共可以采样出2^n个子网络。

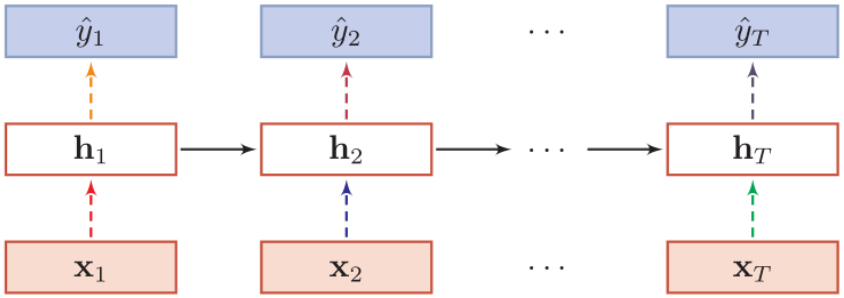
- 贝叶斯学习的解释

$$\mathbb{E}_{q(\theta)}[y] = \int_q f(\mathbf{x}, \theta) q(\theta) d\theta$$
$$\approx \frac{1}{M} \sum_{m=1}^{M} f(\mathbf{x}, \theta_m),$$

  - 其中$f(x,\theta_m)$为第m次应用丢弃方法后的网络。

循环神经网络上的丢弃法

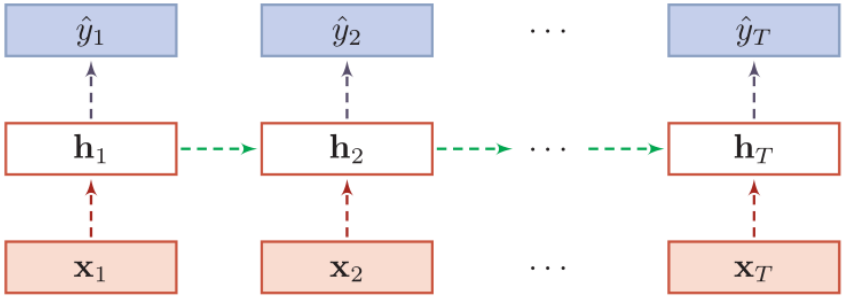
- 当在循环神经网络上应用丢弃法，不能直接对每个时刻的隐状态进行随机丢弃，这样会损害循环网络在时间维度上记忆能力。



虚线边表示进行随机丢弃，不同的颜色表示不同的丢弃掩码。

变分Dropout

- 根据贝叶斯学习的解释，丢弃法是一种对参数θ的采样。
  - 每次采样的参数需要在每个时刻保持不变。因此，在对循环神经网络上使用丢弃法时，需要对参数矩阵的每个元素进行随机丢弃，并在所有时刻都使用相同的丢弃掩码。



相同颜色表示使用相同的丢弃掩码

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转变，引入噪声等方法来增加数据的多样性。
  - 图像数据的增强方法：
    - 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。

标签平滑（Label Smoothing）

- 在输出标签中添加噪声来避免模型过拟合。
- 一个样本x的标签一般用onehot向量表示

$$\mathbf{y} = [0, \cdots, 0, 1, 0, \cdots, 0]^{\mathrm{T}}$$ 硬目标（Hard Targets）

- 引入一个噪声对标签进行平滑，即假设样本以ϵ的概率为其它类。平滑后的标签为

$$\tilde{\mathbf{y}} = [\frac{\epsilon}{K-1}, \cdots, \frac{\epsilon}{K-1}, 1-\epsilon, \frac{\epsilon}{K-1}, \cdots, \frac{\epsilon}{K-1}]^{\mathrm{T}}$$

总结

- 模型
  - 用 ReLU 作为激活函数
  - 分类时用交叉熵作为损失函数
- 优化
  - SDG+mini-batch（Adam算法优先）
  - 每次迭代都重新随机排序
  - 数据预处理（标准归一化）
  - 动态学习率（越来越小）
- 正则化
  - ℓ_1和ℓ_2正则化（跳过前几轮）
  - Dropout
  - Early-stop
  - 数据增强

谢 谢

https://nndl.github.io/