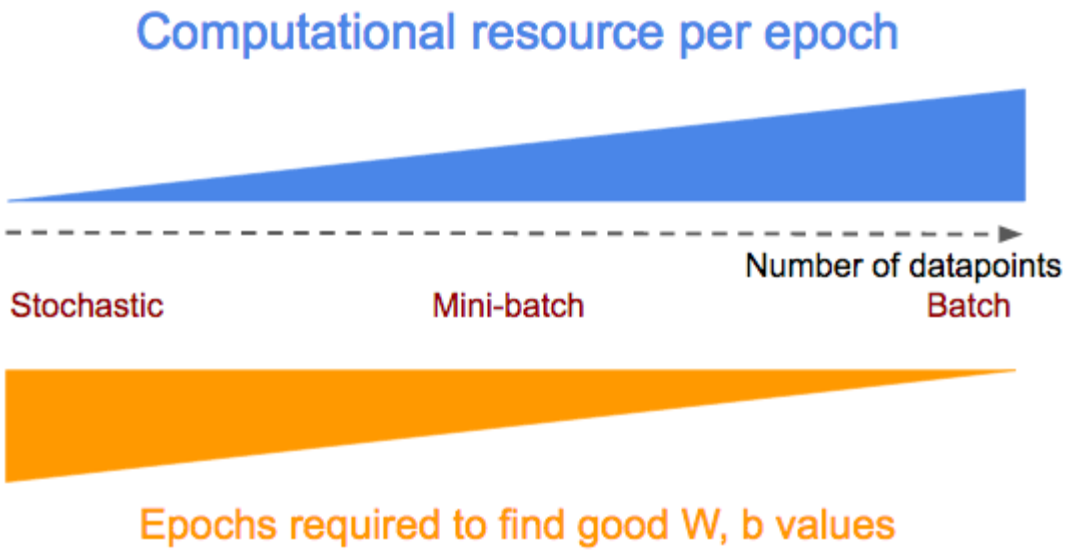
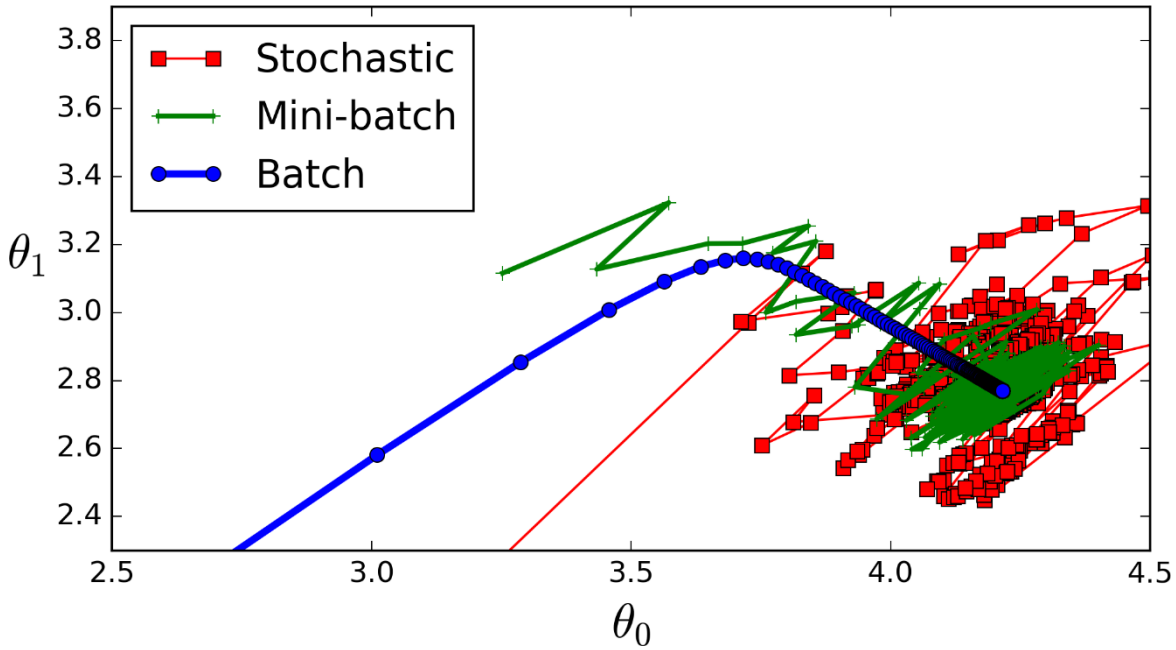
梯度下降训练策略

- 批次梯度下降 (Batch Gradient Descent)
- 随机梯度下降 SGD (Stochastic Gradient Descent)
- 小批次梯度下降 (Mini-batch Gradient Descent)


Computational resource per epoch
Number of datapoints
Stochastic
Mini-batch
Batch
Epochs required to find good W, b values


Stochastic
Mini-batch
Batch
θ_1
θ_0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.5
3.0
3.5
4.0
4.5

批次梯度下降

(Batch Gradient Descent)

利用全部训练数据集计算损失函数的梯度来执行一次参数更新

$$\theta \Leftarrow \theta - \eta \cdot \nabla J(\theta)$$

- 更新较慢
- 不能在线更新模型
- 对凸的损失函数可保证收敛到全局最小值；对非凸的损失函数可收敛到局部最小值

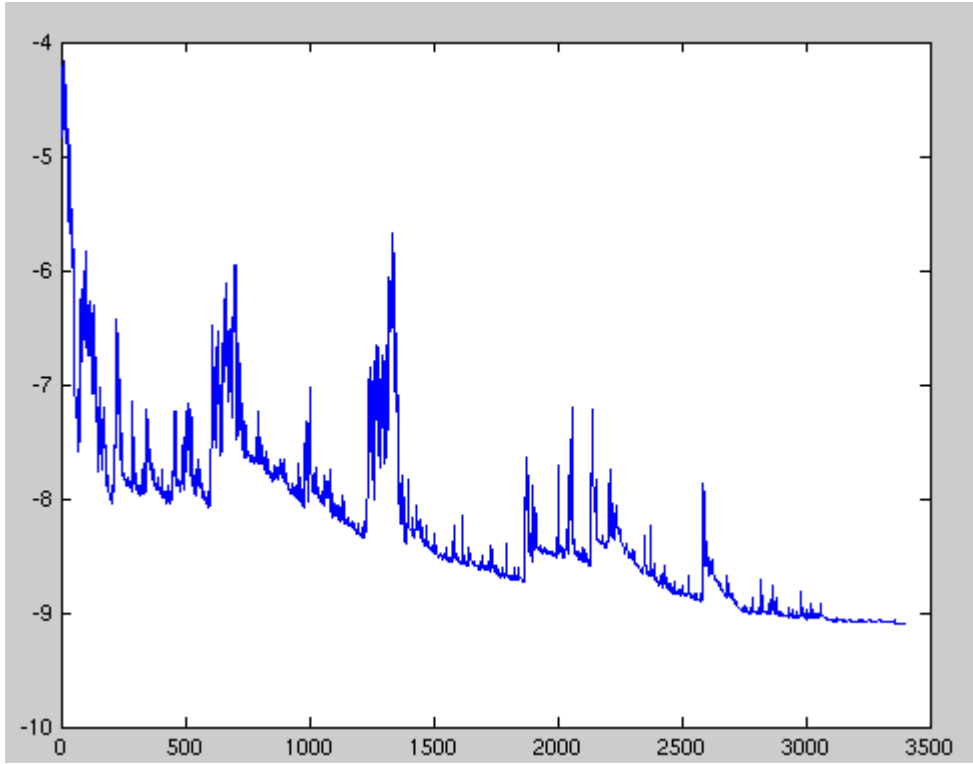
随机梯度下降

SGD (Stochastic Gradient Descent)

对每一个训练样本点和标签执行参数更新

$$\theta \Leftarrow \theta - \eta \cdot \nabla J(\theta; x^{(i)}; y^{(i)})$$

- 速度快，可在线学习
- 梯度精度差，目标函数下降过程出现大幅波动



小批次梯度下降
(Mini-batch Gradient Descent)

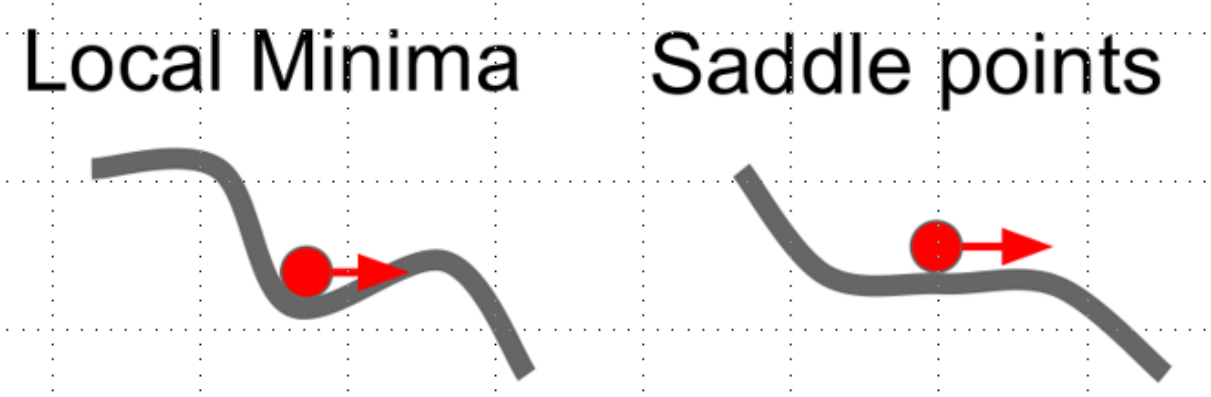
每n个训练样本点，进行一次参数更新

$$\theta \Leftarrow \theta - \eta \cdot \nabla J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

- Batch-GD和单样本SGD方法的折衷
- 减小了参数更新的方差，可平稳收敛
- 速度快，可利用优化的矩阵运算库来高效的计算梯度
- Batch大小根据问题来定。一般而言设为32、64、128或256即可。

存在的问题

- 确定合适的学习率比较困难
  - 小的学习率可保证收敛性，但收敛过程很慢
  - 大的学习率会导致损失函数优化过程在最小值附近波动甚至发散
- SGD难以跳出局部极小值点和鞍点
  - 神经网络对应的函数具有高度非线性
  - 众多局部极小值点或鞍点
  - 在极小值点或鞍点附近，目标函数值几乎不变，导致梯度近似为零



梯度下降改进优化算法

- Momentum
- NAG
- Adagrad
- Adadelta
- RMSprop
- Adam

SGD+动量(Momentum)

动量（Momentum）可用来加速SGD

$$v_1 = \eta \nabla J(\theta_1)$$

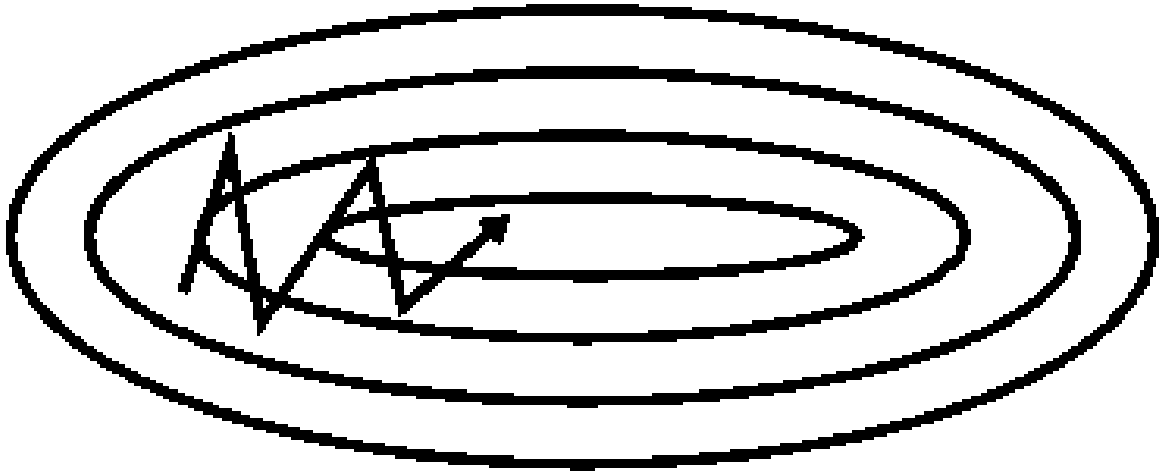
$$v_k = \gamma v_{k-1} + \eta \nabla J(\theta_{k-1}), \quad \gamma \in (0,1)$$

$$\theta_k = \theta_{k-1} - v_k$$

- 把过去时间步骤更新矢量的一部分（γ）加到当前更新矢量
- 速度的逐渐增加是作为梯度的移动平均
- 动量项γ 一般设为0.9。可理解为“摩擦”效果

SGD+动量(Momentum)

SGD without momentum



SGD with momentum

- 相比SGD，移动平均提供更好的梯度下降方向估计
- 动量有助于在正确方向上加速梯度，从而越过沟壑(ravine)

Adam(Adaptive Moment Estimation)

Adam是一种为每一参数计算自适应学习率的方法

存储过去梯度平方的指数衰减平均值v_t；保留了过去梯度的指数衰减平均值m_t，类似动量 。

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$g_{t,i} = \nabla J(\theta_{t,i})$$

m_t 和v_t分别是一阶矩(均值) 和梯度二阶矩(有偏方差) 的估计值

参数 β_1、$\beta_2 \in [0, 1)$ 控制了这些移动均值（moving average）指数衰减率。

Adam(Adaptive Moment Estimation)

计算含有偏差校正（bias-corrected）的一阶矩和二阶矩 的估计:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Adam更新规则:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

YOLOv3 梯度下降策略与优化算法

- mini Batch GD+Momentum（CPU训练）
- Adam（GPU训练）