SGD, stochastic gradient descent, 確率的勾配下降法

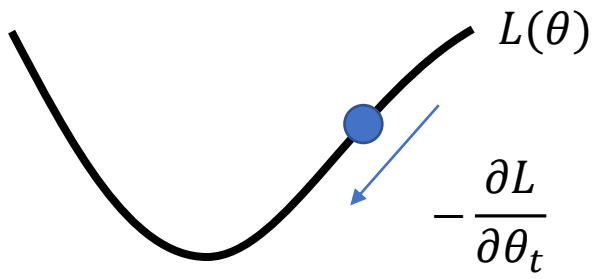
情報ソース：ゼロから作るDeep Learning , DEEP LEARNING 深層学習

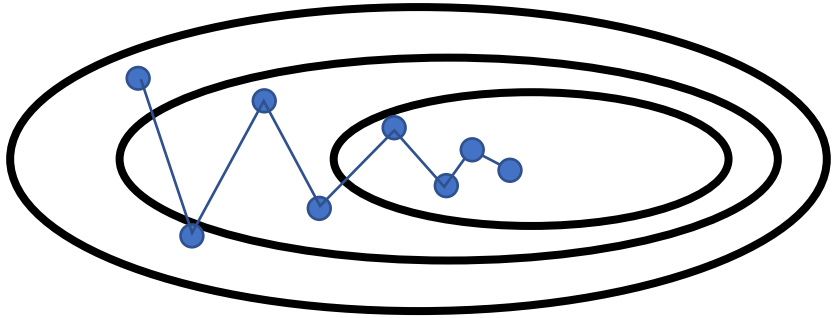
【定義】

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta_t}$$

η:学習率

【勾配下降イメージ】





```
class SGD:

    def __init__(self, lr=0.01):
        self.lr = lr

    def update(self, params, grads):
        for key in params.keys():
            params[key] -= self.lr*grads[key]
```

Momentum

情報ソース：ゼロから作るDeep Learning , DEEP LEARNING 深層学習

【定義】

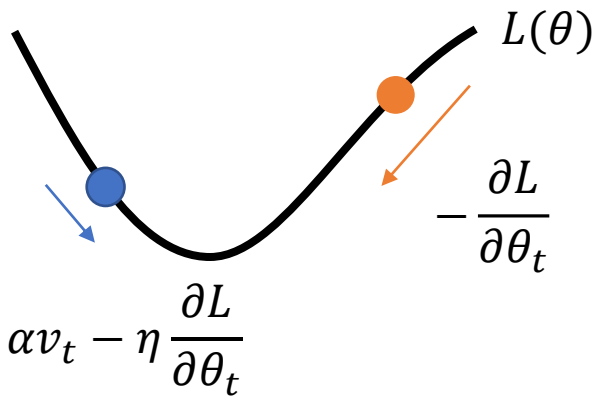
$$v_{t+1} = \alpha v_t - \eta \frac{\partial L}{\partial \theta_t}$$

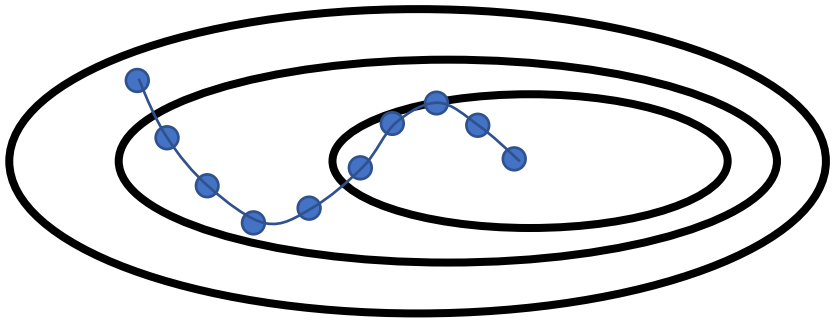
$$\theta_{t+1} = \theta_t + v_{t+1}$$

α:モーメンタム係数, η:学習率

前回の勾配の情報をv_tが持つ。v_tは速度ととらえる事ができる。
・前回と勾配が同じ：より下げる方向に更新される。
・前回と勾配が異なる：下げる方向と逆に減速が働く。

【勾配下降イメージ】





```
class Momentum:

    def __init__(self, lr=0.01, momentum=0.9):
        self.lr = lr
        self.momentum = momentum
        self.v = None # initialize

    def update(self, params, grads):
        if self.v is None:
            self.v = {}
            for key, val in params.items():
                self.v[key] = np.zeros_like(val)

        for key in params.keys():
            self.v[key] = self.momentum*self.v[key] - self.lr*grads[key]
            params[key] += self.v[key]
```

Nestrov

情報ソース：ゼロから作るDeep Learning , DEEP LEARNING 深層学習

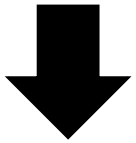
【定義】

$$v_{t+1} = \alpha v_t - \eta \frac{\partial L}{\partial(\theta_t + \alpha v_t)}$$

$$\theta_{t+1} = \theta_t + v_{t+1}$$

現在の速度が適用された後の勾配計算を行う。
標準のモーメンタムに修正要因を追加しようとする。

α:モーメンタム係数, η:学習率



このままでは実装が難しい。
実装用の定義

【定義】

$\Theta_{t+1} = \theta_t + v_{t+1}$ とおき

$$v_{t+1} = \alpha v_t - \eta \frac{\partial L}{\partial \Theta}$$

$$\Theta_{t+1} = \Theta_t + \alpha^2 v_t - (1+\alpha) * \eta \frac{\partial L}{\partial \Theta}$$

```
class Mestrov:

    def __init__(self, lr=0.01, momentum=0.9):
        self.lr = lr
        self.momentum = momentum
        self.v = None

    def update(self, params, grads):
        if self.v is None:
            self.v = {}
            for key, val in params.items():
                self.v[key] = np.zeros_like(val)

        for key in params.keys():
            params[key] += self.momentum * self.momentum * self.v[key]
            params[key] -= (1 + self.momentum) * self.lr * grads[key]
            self.v[key] *= self.momentum
            self.v[key] -= self.lr * grads[key]
```

AdaGrad

情報ソース：ゼロから作るDeep Learning , DEEP LEARNING 深層学習

【定義】

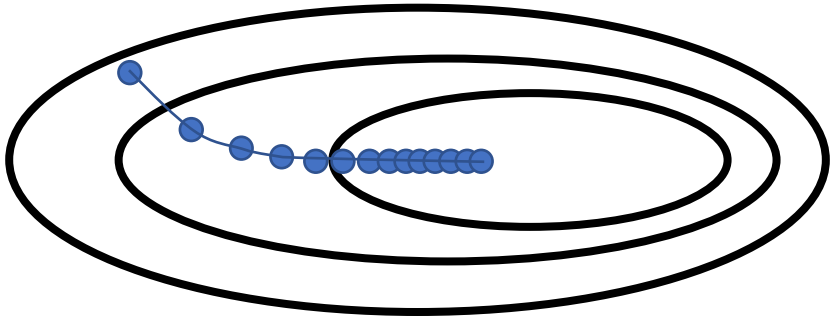
$$h_{t+1} = h_t + \eta \frac{\partial L}{\partial \theta_t} \odot \frac{\partial L}{\partial \theta_t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{1}{\varepsilon + \sqrt{h_{t+1}}} \odot \frac{\partial L}{\partial \theta_t}$$

η:学習率, ε:小さい定数

学習率を減衰させる。h_{t+1}は過去の勾配の2乗和となる。

【勾配下降イメージ】



```
class AdaGrad:

    def __init__(self, lr=0.01):
        self.lr = lr
        self.h = None

    def update(self, params, grads):
        if self.h is None:
            self.h = {}
            for key, val in params.items():
                self.h[key] = np.zeros_like(val)

        for key in params.keys():
            self.h[key] += grads[key] * grads[key]
            params[key] -= self.lr * grads[key] / (np.sqrt(self.h[key]) + 1e-7)
```

RMSProp

情報ソース：ゼロから作るDeep Learning , DEEP LEARNING 深層学習

【定義】

$$h_{t+1} = \rho h_t + (1-\rho)\frac{\partial L}{\partial \theta_t} \odot \frac{\partial L}{\partial \theta_t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{1}{\sqrt{\varepsilon + h_{t+1}}} \odot \frac{\partial L}{\partial \theta_t}$$

η:学習率, ε:小さい定数

AdaGradの修正モデル。
勾配の重みを指数関数的な重みをつけた移動平均に変更する事で
、非凸の条件下でAdaGradの性能を改善している

```
class RMSProp:

    def __init__(self, lr=0.01, decay_rate=0.99):
        self.lr = lr
        self.decay_rate = decay_rate
        self.h = None

    def update(self, params, grads):
        if self.h is None:
            self.h = {}
            for key, val in params.items():
                self.h[key] = np.zeros_like(val)

        for key in params.keys():
            self.h[key] *= self.decay_rate
            self.h[key] += (1 - self.decay_rate) * grads[key] * grads[key]
            params[key] -= self.lr * grads[key] / (np.sqrt(self.h[key] + 1e-7))
```

Adam

情報ソース：ゼロから作るDeep Learning, DEEP LEARNING 深層学習

【定義】

$$m_{t+1} = \rho_1 m_t + (1-\rho_1)\frac{\partial L}{\partial \theta_t}$$

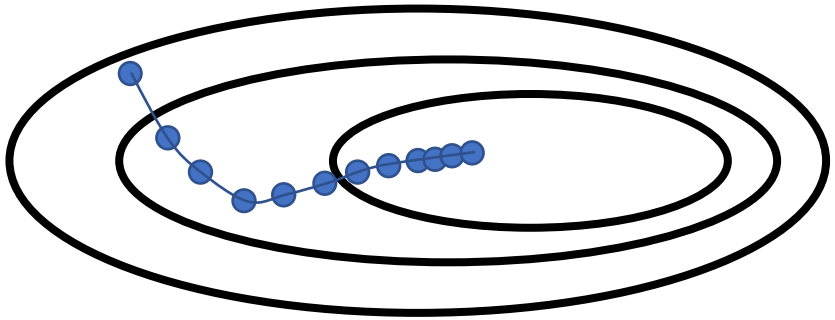
$$v_{t+1} = \rho_2 v_t + (1-\rho_2)\frac{\partial L}{\partial \theta_t}\odot\frac{\partial L}{\partial \theta_t}$$

$\hat{m}_{t+1} = \frac{m_{t+1}}{1-{\rho_1}^t}$, $\hat{v}_{t+1} = \frac{v_{t+1}}{1-{\rho_2}^t}$ として

$$\theta_{t+1} = \theta_t - \eta\frac{1}{\sqrt{\hat{v}_{t+1}}+\varepsilon}\odot\hat{m}_{t+1}$$

η:学習率, ε:小さい定数
ρ_1, ρ_2:モーメント推定に対する指数減衰率

MomentumとAdaGradを融合するアイデア



```
class Adam:

    def __init__(self, lr=0.001, rho1=0.9, rho2=0.999):
        self.lr = lr
        self.rho1 = rho1
        self.rho2 = rho2
        self.iter = 0
        self.m = None
        self.v = None
        self.epsilon = 1e-8

    def update(self, params, grads):
        if self.m is None:
            self.m, self.v = {}, {}
            for key, val in params.items():
                self.m = np.zeros_like(val)
                self.v = np.zeros_like(val)

        self.iter += 1

        for key in params.keys():
            self.m[key] = self.rho1*self.m[key] + (1 - self.rho1)*grads[key]
            self.v[key] = self.rho2*self.v[key] + (1 - self.rho2)*(grads[key]**2)

            m = self.m[key] / (1 - self.rho1**self.iter)
            v = self.v[key] / (1 - self.rho2**self.iter)

            params[key] -= self.lr * m / (np.sqrt(v) * self.epsilon)
```