

رسالة محمد



یادگیری عمیق

مدرس: محمدرضا محمدی

زمستان ۱۴۰۱

الگوریتم‌های بهینه‌سازی

Optimization Algorithms

رویکرد ۳: حرکت در مسیر شیب

- تقریب مرتبه اول یک تابع یک‌بعدی به صورت زیر تعریف می‌شود:

$$f(x + \epsilon) = f(x) + \epsilon f'(x) + \mathcal{O}(\epsilon^2)$$

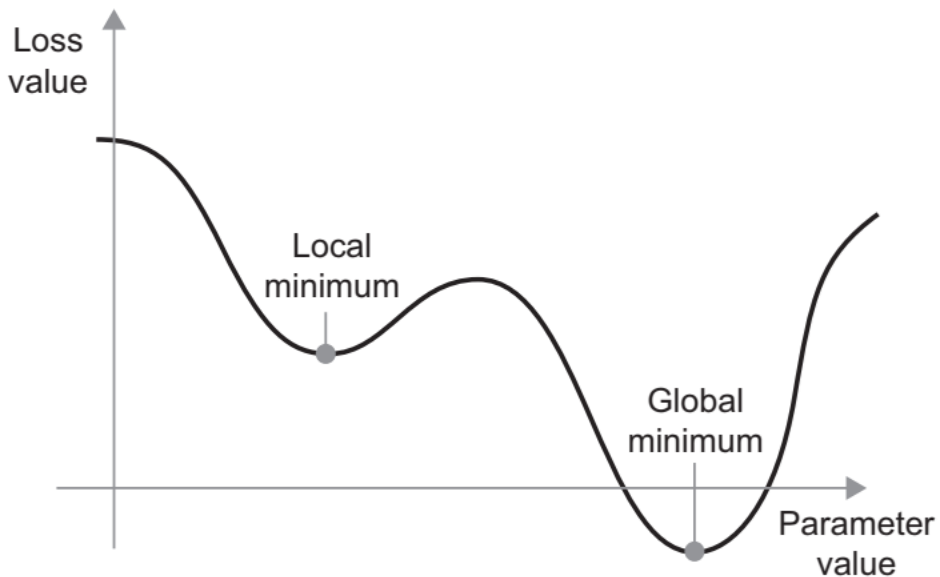
- اگر $\epsilon = -\eta f'(x)$ با $\eta > 0$ انتخاب شود

$$f(x - \eta f'(x)) = f(x) - \eta f'^2(x) + \mathcal{O}(\eta^2 f'^2(x))$$

- اگر $f'(x) \neq 0$ باشد و η کوچک باشد:

$$f(x - \eta f'(x)) \lesssim f(x)$$

$$x \leftarrow x - \eta f'(x)$$

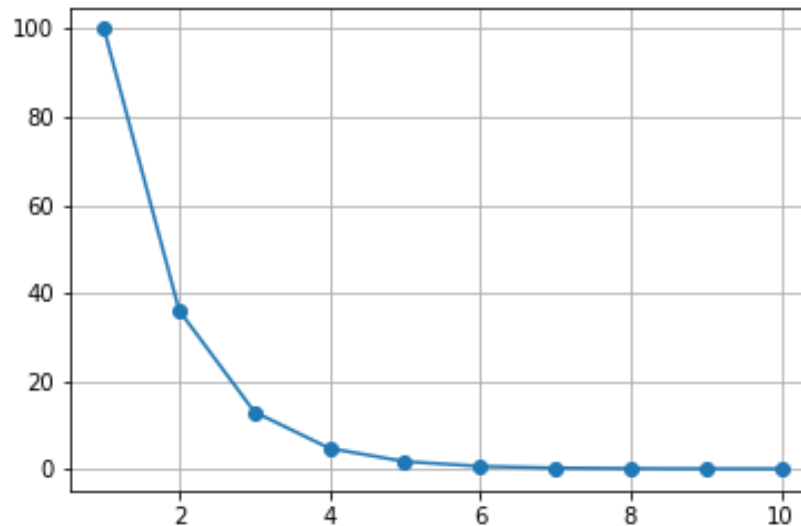
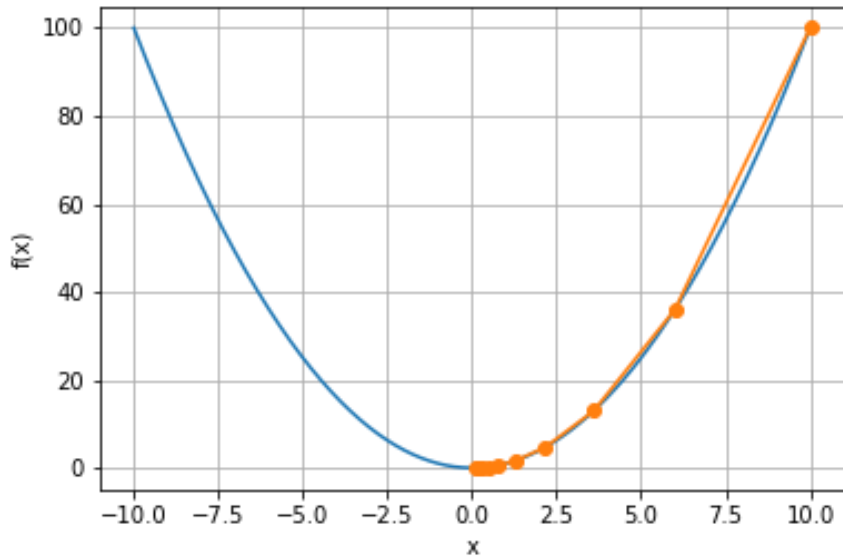


الگوریتم گرادیان کاهشی (Gradient Descent)

```
# Vanilla Gradient Descent

while True:
    weights_grad = evaluate_gradient(loss_fun, data, weights)
    weights += - step_size * weights_grad # perform parameter update
```

$$\eta = 0.20$$

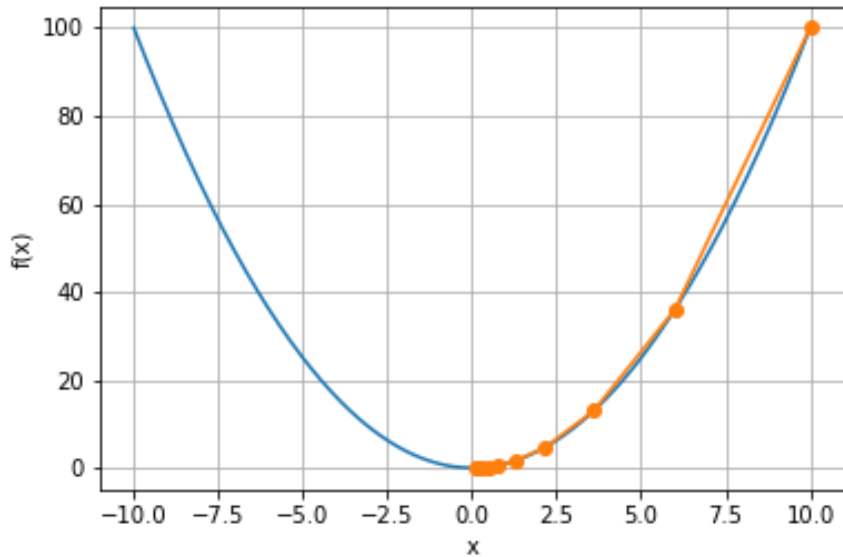


الگوریتم گرادیان کاهشی (Gradient Descent)

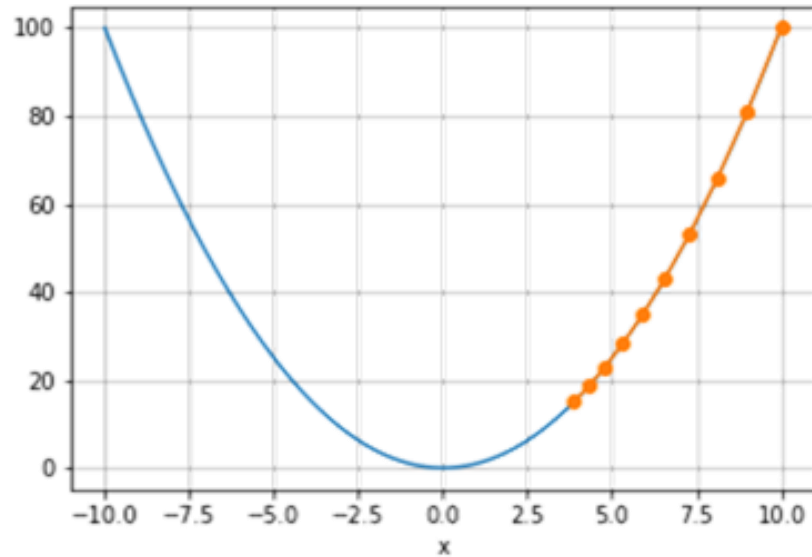
```
# Vanilla Gradient Descent

while True:
    weights_grad = evaluate_gradient(loss_fun, data, weights)
    weights += - step_size * weights_grad # perform parameter update
```

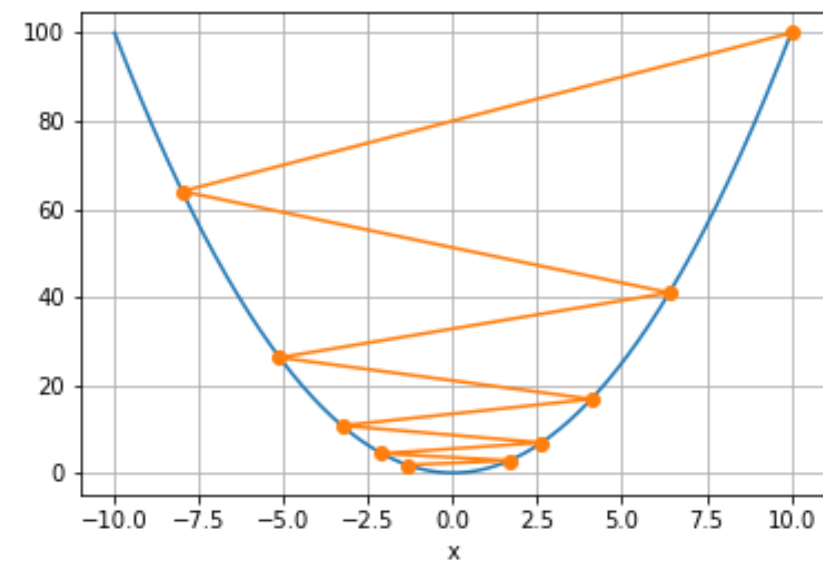
$\eta = 0.20$



$\eta = 0.05$

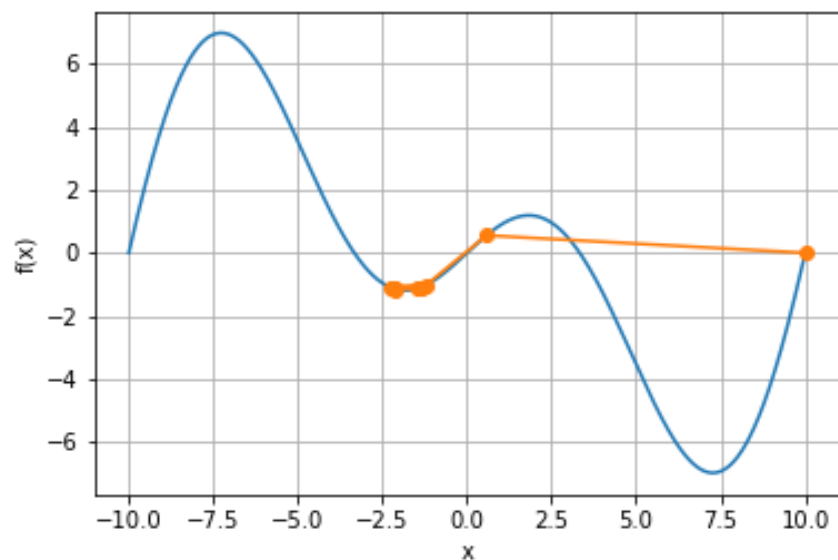


$\eta = 0.90$



الگوریتم گرادیان کاهشی (Gradient Descent)

- الگوریتم GD به سادگی ممکن است به بهینه محلی همگرا شود



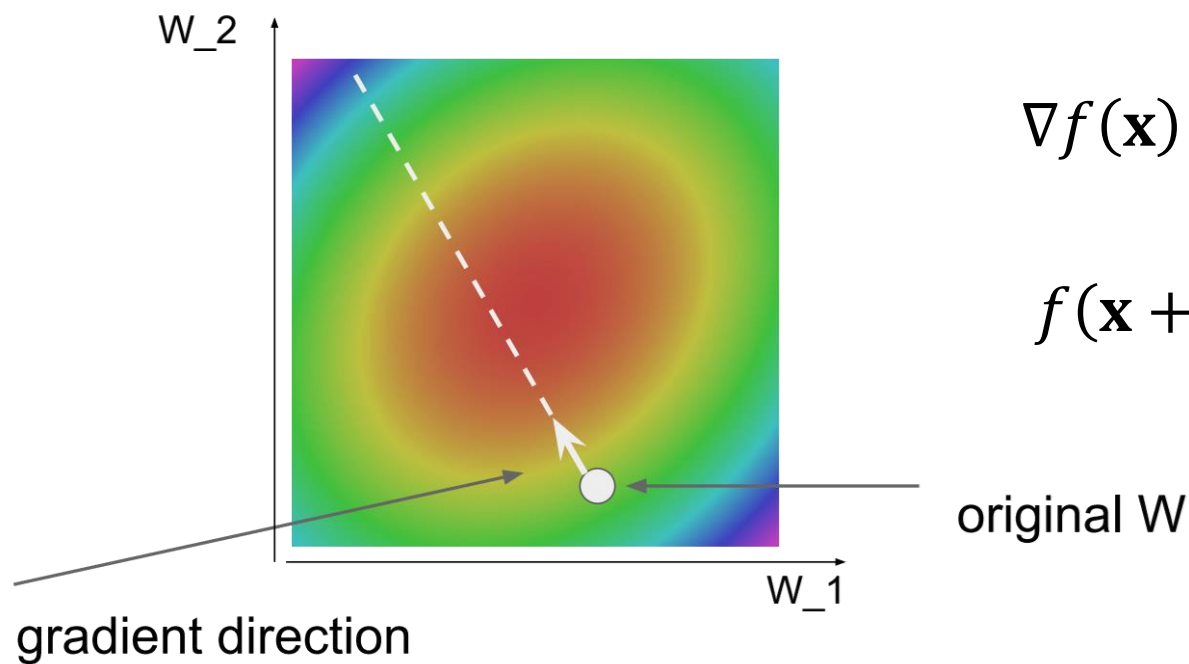
الگوریتم گرادیان کاهشی (Gradient Descent)

- در چند بعد، گرادیان تعریف می‌شود که برداری است شامل مشتق‌های جزئی در هر بُعد
- شیب در هر جهت دلخواه برابر است با ضرب داخلی جهت با بردار گرادیان
- جهت تندترین کاهش تابع برابر با منفی گرادیان است

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right]^T$$

$$f(\mathbf{x} + \epsilon) = f(\mathbf{x}) + \epsilon^T \nabla f(\mathbf{x}) + \mathcal{O}(\|\epsilon\|^2)$$

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f(\mathbf{x})$$



روش‌های افقی

- انتخاب مقدار درست برای نرخ آموزش η بسیار مهم و البته مشکل است
 - آیا می‌توان η را به صورت خودکار انتخاب کرد یا اصلاً به آن نیاز نداشت؟
- می‌توان از انحنای منحنی (curvature) هم استفاده کرد
- روش نیوتن: در بسط تیلور از مرتبه دوم هم استفاده کنیم

$$f(\mathbf{x} + \boldsymbol{\epsilon}) = f(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla f(\mathbf{x}) + \frac{1}{2} \boldsymbol{\epsilon}^T \nabla^2 f(\mathbf{x}) \boldsymbol{\epsilon} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^3)$$

- ماتریس Hessian تابع f با ابعاد $d \times d$ به صورت $H \triangleq \nabla^2 f(\mathbf{x})$ نمایش داده می‌شود
 - برای d های بزرگ از لحاظ مصرف حافظه و محاسبات سنگین خواهد بود
- تقریب مرتبه ۲ دارای یک بهینه سراسری و قابل محاسبه است

روش نیوتن

$$f(\mathbf{x} + \boldsymbol{\epsilon}) = f(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla f(\mathbf{x}) + \frac{1}{2} \boldsymbol{\epsilon}^T \mathbf{H} \boldsymbol{\epsilon} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^3)$$

- برای یافتن مقدار بهینه این تابع تقریبی، می‌توان مشتق آن نسبت به $\boldsymbol{\epsilon}$ را برابر با صفر قرار داد

$$\nabla f(\mathbf{x}) + \mathbf{H} \boldsymbol{\epsilon} = 0 \Rightarrow \boldsymbol{\epsilon} = -\mathbf{H}^{-1} \nabla f(\mathbf{x})$$

- بجای نرخ آموزش از معکوس ماتریس Hessian استفاده شده است

$$f(x) = x^2 - 2x + 3 \Rightarrow f'(x) = 2x - 2 \Rightarrow f''(x) = 2$$

- مثال:

$$x_0 = 12 \Rightarrow x \leftarrow x + \epsilon = 12 - \frac{1}{2} \times 22 = 1$$

- در این مثال خاص در یک تکرار از هر نقطه شروعی به جواب بهینه می‌رسد

گرادیان کاهشی تصادفی (SGD)

- در یادگیری عمیق، تابع هدف به طور معمول میانگین تابع ضرر برای تمام نمونه‌های مجموعه داده آموزشی است

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x}) \Rightarrow \nabla f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\mathbf{x})$$

- این محاسبات به صورت خطی با بزرگ شدن مجموعه داده افزایش می‌یابد
- در روش SGD، در هر تکرار گرادیان تنها برای یک نمونه محاسبه می‌شود
- هزینه هر گام به مقدار ثابت $\mathcal{O}(1)$ می‌رسد
- گرادیان تصادفی $\nabla f_i(\mathbf{x})$ یک تخمین بدون بایاس از $\nabla f(\mathbf{x})$ است

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x})$$

$$\mathbb{E}_i[\nabla f_i(\mathbf{x})] = \nabla f(\mathbf{x})$$

گرایان کاهش تصادفی (SGD)

- بجای ۱ نمونه، می توان $\nabla f(\mathbf{x})$ آن را با استفاده از یک minibatch از نمونه ها تقریب زد
- ۳۲/۶۴/۱۲۸ متداول هستند

```
# Vanilla Minibatch Gradient Descent
```

```
while True:
```

```
    data_batch = sample_training_data(data, 256) # sample 256 examples
```

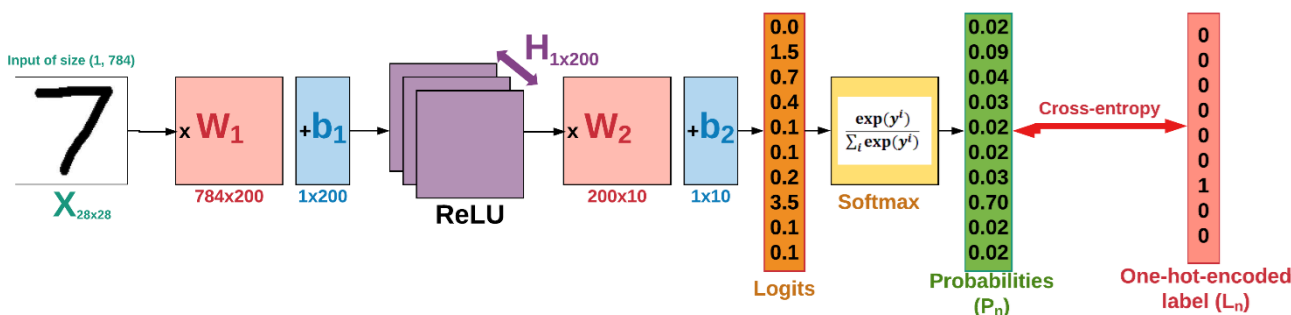
```
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

چرخه آموزش

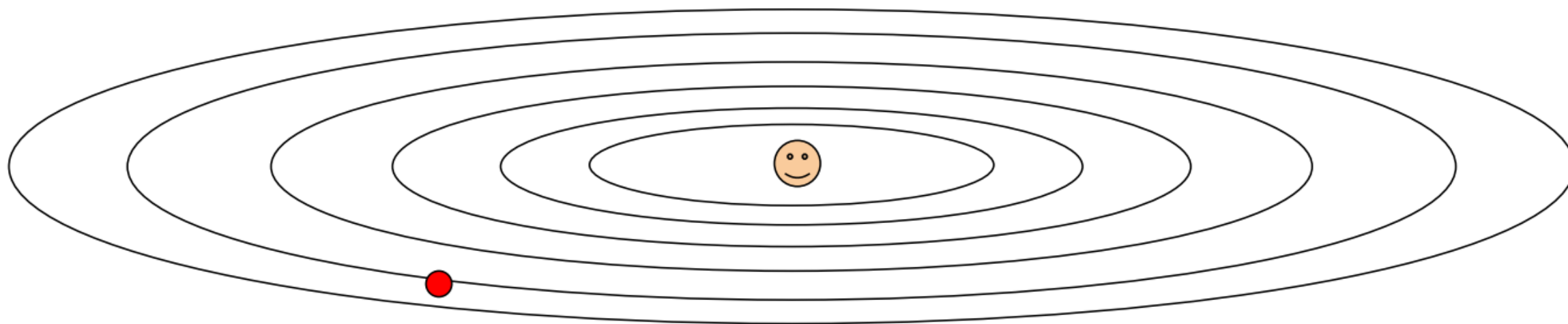
- این تنظیم تدریجی، که آموزش نیز نامیده می شود، پایه یادگیری در بسیاری از الگوریتم های یادگیری ماشین است

- یک batch از نمونه های آموزشی x و خروجی های مربوطه y انتخاب می شود
- شبکه بر روی x اعمال می شود (forward pass) تا \hat{y} بدست بیاید
- تابع ضرر شبکه برای این batch از مقایسه y و \hat{y} محاسبه می شود
- تمام وزن های شبکه به گونه ای به روز می شوند که مقدار ضرر برای این batch کمی کاهش بیابد



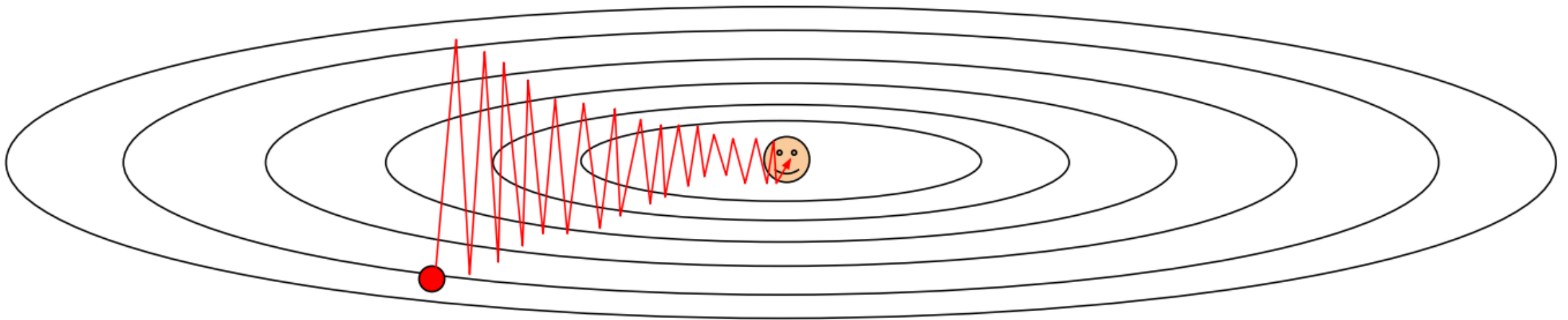
مشکلات SGD

- اگر تابع ضرر در یک راستا تغییرات بسیار سریعتری نسبت به یک راستای دیگر داشته باشد چه می‌شود؟
- الگوریتم GD چگونه عمل می‌کند؟ اندازه گام چه مقدار باشد؟



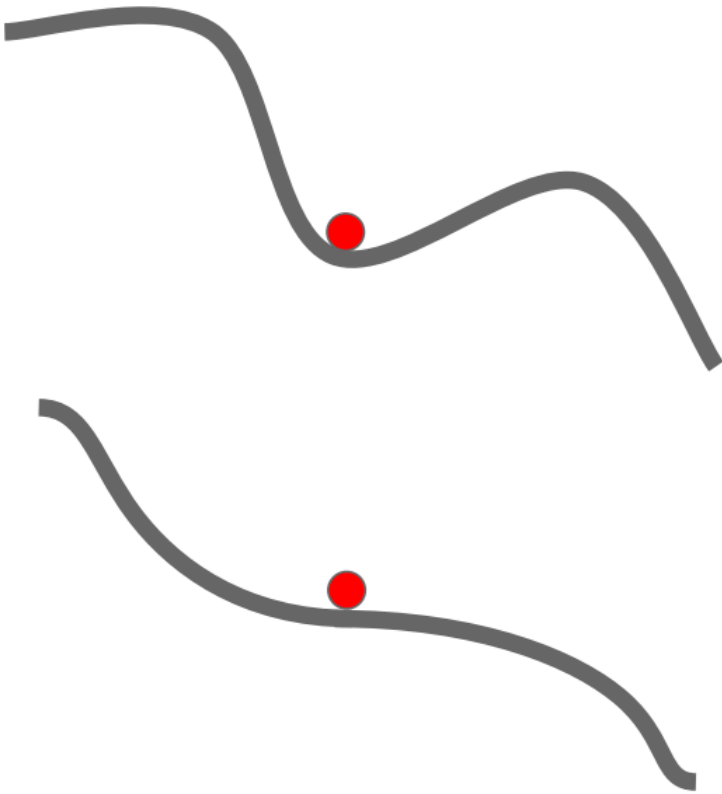
مشکلات SGD

- اگر تابع ضرر در یک راستا تغییرات بسیار سریعتری نسبت به یک راستای دیگر داشته باشد چه می‌شود؟
- الگوریتم GD چگونه عمل می‌کند؟
 - در یک راستا خیلی آهسته پیش می‌رود و در یک راستای دیگر نوسان می‌کند
 - اندازه گام چه مقدار باشد؟



مشکلات SGD

- اگر تابع ضرر دارای مینیمم محلی یا نقطه زینی باشد؟
- گرادیان صفر باعث توقف GD می شود
- این مشکل در SGD کمتر است زیرا بعید است گرادیان یک نقطه در تمام minibatchها صفر باشد



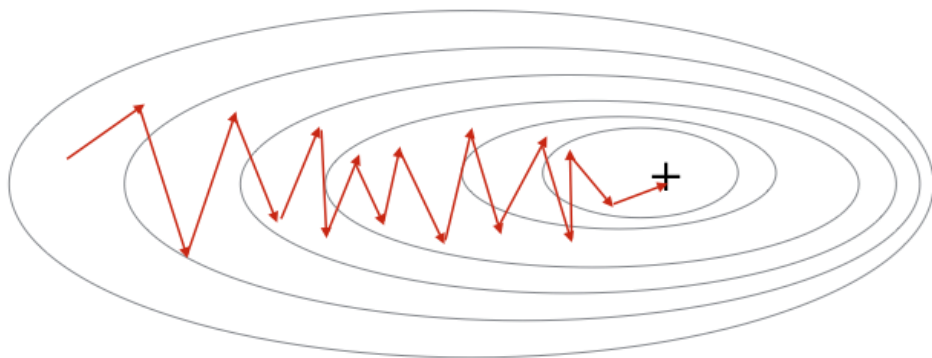
مشکلات SGD

- گرادیان‌ها از minibatch محاسبه می‌شوند و می‌توانند نویزی باشند!

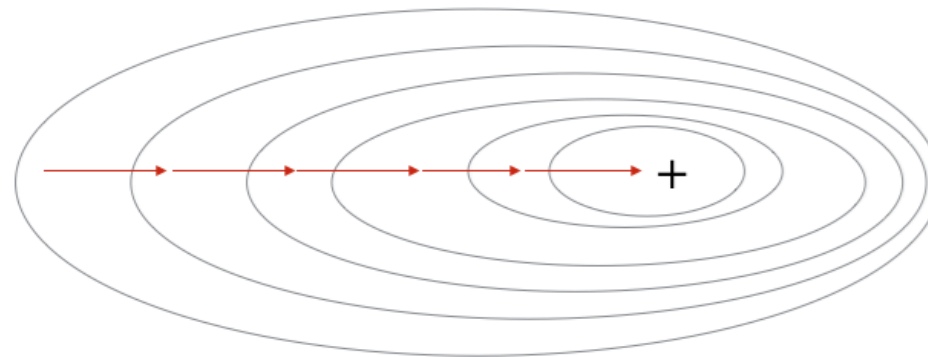
$$\nabla f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\mathbf{x})$$

- با وجود نویزی بودن گرادیان‌ها، به طور میانگین در مسیر درستی حرکت می‌کند

Stochastic Gradient Descent



Gradient Descent



SGD + Momentum

- تابع ضرر می تواند مشابه با ارتفاع یک زمینه تپه ای در نظر گرفته شود
- مقداردهی اولیه تصادفی برای پارامترها معادل با قرار دادن یک توپ با سرعت اولیه صفر در یک مکان است

- فرآیند بهینه سازی می تواند معادل با حرکت توپ به سمت عمیق ترین نقطه این زمین در اثر جاذبه مدل سازی شود



SGD + Momentum

SGD + Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
vx = 0
while True:
    dx = compute_gradient(x)
    vx = rho * vx + dx
    x -= learning_rate * vx
```

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

```
while True:
    dx = compute_gradient(x)
    x -= learning_rate * dx
```

- با معادل فرض کردن گرادیان با شتاب، "سرعت" محاسبه می شود
- پارامتر ρ اصطکاک را شبیه سازی می کند
 - به طور معمول ۰.۹ یا ۰.۹۹ است
- یک پیاده سازی معادل هم به صورت روبرو است