
بسم الله الرحمن الرحيم

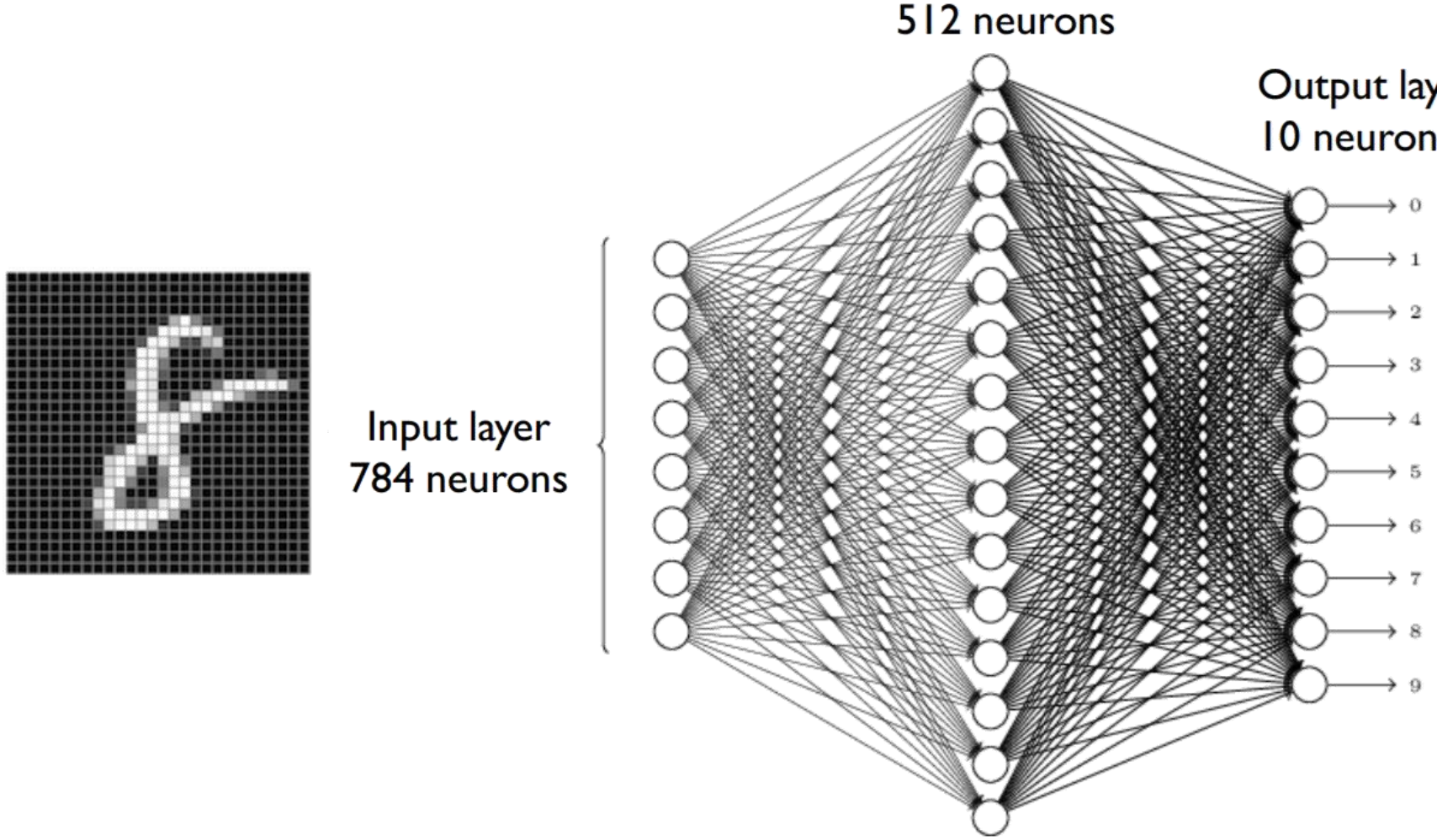
مبانی یادگیری عمیق

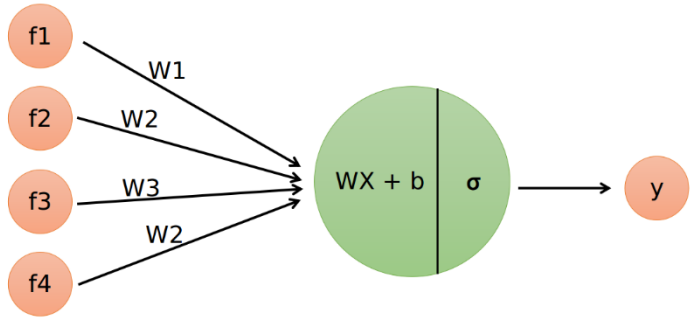
مدرس: محمدرضا محمدی

۱۴۰۰

پرسپترون چندلایه (MLP)

- بخش‌های ۲.۲ و ۲.۳ در رابطه با تنسورها در پایتون مطالعه شود

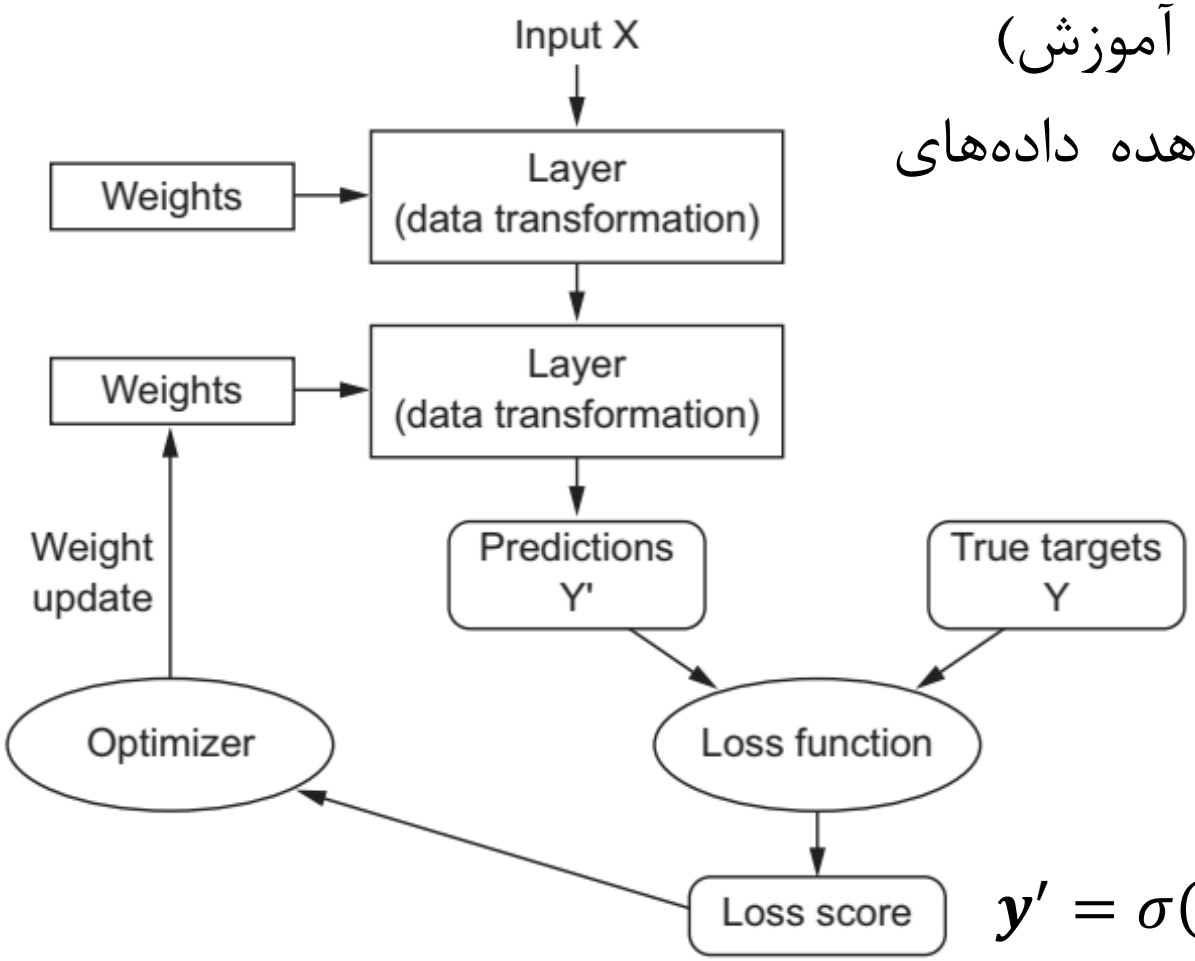




بهینه‌سازی

بهینه‌سازی

- Wها و $\boldsymbol{b}$ها وزن‌های هر لایه هستند (پارامترهای قابل آموزش)
- این وزن‌ها شامل اطلاعاتی هستند که شبکه از مشاهده داده‌های آموزشی آموخته است
- چگونه مقادیر بهینه را بدست بیاوریم؟



$$\boldsymbol{y}' = \sigma(W_2\sigma(W_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$$

رویکرد ۱: جستجوی تصادفی

```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)
# assume Y_train are the labels (e.g. 1D array of 50,000)
# assume the function L evaluates the loss function

bestloss = float("inf") # Python assigns the highest possible float value
for num in range(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
  print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)

# prints:
# in attempt 0 the loss was 9.401632, best 9.401632
# in attempt 1 the loss was 8.959668, best 8.959668
# in attempt 2 the loss was 9.044034, best 8.959668
# in attempt 3 the loss was 9.278948, best 8.959668
# in attempt 4 the loss was 8.857370, best 8.857370
# in attempt 5 the loss was 8.943151, best 8.857370
# in attempt 6 the loss was 8.605604, best 8.605604
# ... (trunctated: continues for 1000 lines)
```

- پاسخ بسیار ضعیف است!
- دقت نهایی تنها ۱۵.۵٪ است
- نتایج روش‌های جدید بالای ۹۹٪ است

```
# Assume X_test is [3073 x 10000], Y_test [10000 x 1]
scores = Wbest.dot(Xte_cols) # 10 x 10000, the class scores for all test examples
# find the index with max score in each column (the predicted class)
Yte_predict = np.argmax(scores, axis = 0)
# and calculate accuracy (fraction of predictions that are correct)
np.mean(Yte_predict == Yte)
# returns 0.1555
```

رویکرد ۲: جستجوی محلی تصادفی

- در هر مرحله بهترین وزن‌ها را ذخیره می‌کنیم و جستجو را با یک گام محدود در اطراف آن انجام می‌دهیم
- با استفاده از این تغییر ساده، دقت بر روی داده‌های آزمون به ۲۱.۴٪ افزایش می‌یابد
- همچنان دقت خیلی پائین است!

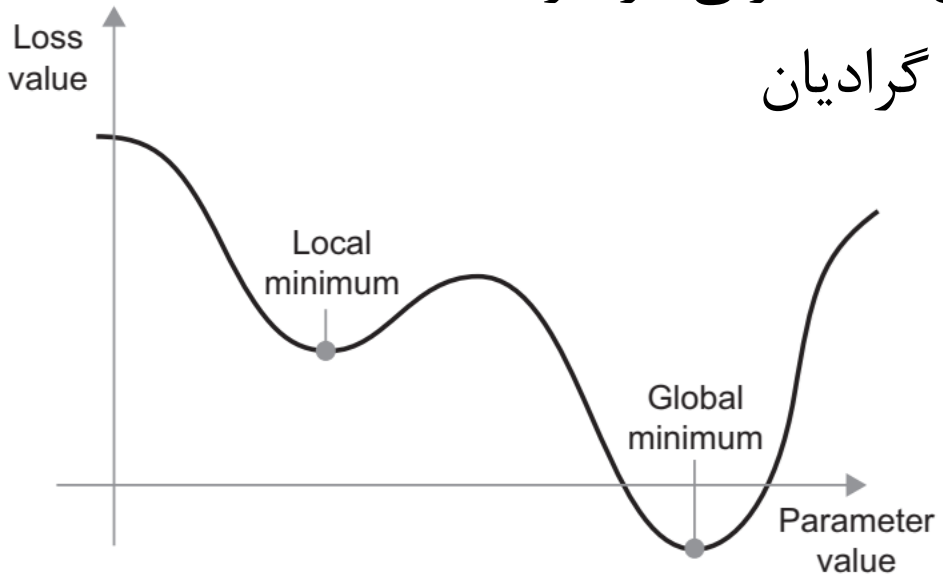
```
W = np.random.randn(10, 3073) * 0.001 # generate random starting W
bestloss = float("inf")
for i in range(1000):
  step_size = 0.0001
  Wtry = W + np.random.randn(10, 3073) * step_size
  loss = L(Xtr_cols, Ytr, Wtry)
  if loss < bestloss:
    W = Wtry
    bestloss = loss
  print 'iter %d loss is %f' % (i, bestloss)
```

رویکرد ۳: حرکت در مسیر شیب

- برای ۱ بعد، مشتق تابع به صورت زیر تعریف می‌شود:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- 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



محاسبه عددی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

محاسبه عددی گرادیان

current W:	**W + h** (first dim):	**gradient dW:**
[0.34, -1.11, 0.78, 0.12, 0.55, 2.81, -3.1, -1.5, 0.33,…]	[0.34 + **0.0001**, -1.11, 0.78, 0.12, 0.55, 2.81, -3.1, -1.5, 0.33,…]	[?, ?, ?, ?, ?, ?, ?, ?, ?,…]
loss 1.25347	**loss 1.25322**	

محاسبه عددی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25322

gradient dW:

[**-2.5**,
?,
?,
(1.25322 - 1.25347)/0.0001
= -2.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,…]

محاسبه عددی گرادیان

current W:	**W + h** (second dim)**:**	**gradient dW:**
[0.34, -1.11, 0.78, 0.12, 0.55, 2.81, -3.1, -1.5, 0.33,…]	[0.34, -1.11 + **0.0001**, 0.78, 0.12, 0.55, 2.81, -3.1, -1.5, 0.33,…]	[-2.5, ?, ?, ?, ?, ?, ?, ?, ?,…]
loss 1.25347	**loss 1.25353**	

محاسبه عددی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,…]

محاسبه عددی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (third dim):

[0.34,
-1.11,
0.78 + **0.0001**,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

gradient dW:

[-2.5,
0.6,
?,
?,
?,
?,
?,
?,
?,…]

محاسبه عددی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (third dim):

[0.34,
-1.11,
0.78 + **0.0001**,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

gradient dW:

[-2.5,
0.6,
0,
?,
?,…]

(1.25347 - 1.25347)/0.0001
= 0

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

محاسبه عددی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (third dim):

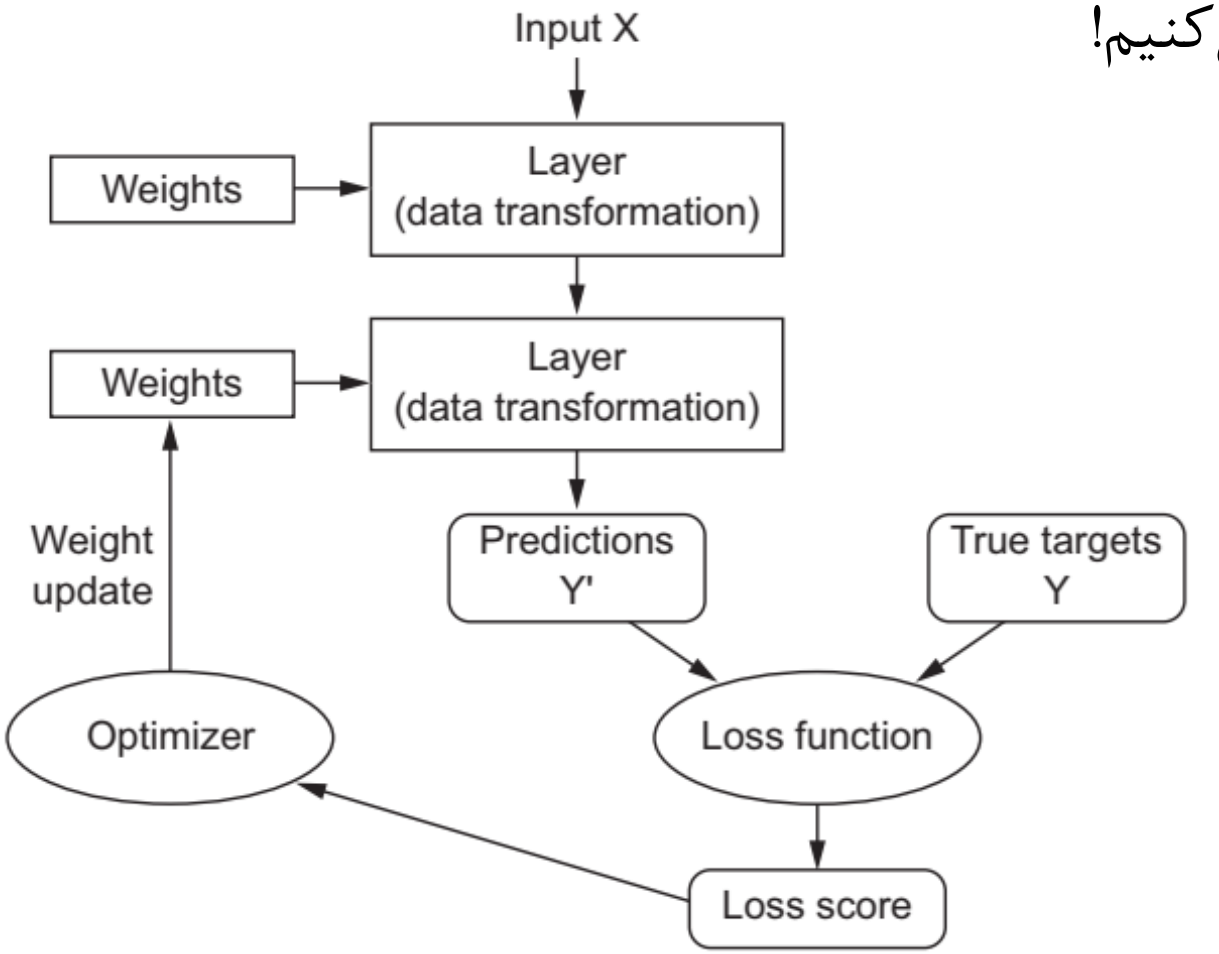
[0.34,
-1.11,
0.78 + **0.0001**,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

gradient dW:

[-2.5,
0.6,
0,
?,
?,…]

Numeric Gradient
- Slow! Need to loop over all dimensions
- Approximate

محاسبه تحلیلی گرادیان



- از ریاضیات برای محاسبه تحلیلی گرادیان استفاده می‌کنیم!

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(s_i, y_i)$$

$$s_i = f(x_i, W)$$

- باید $\nabla_W L$ را محاسبه کنیم
  - تابعی است از W و مجموعه داده
  - تابع L باید مشتق‌پذیر باشد

محاسبه تحلیلی گرادیان

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

dW = ...
(some function
data and W)

gradient dW:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,…]

محاسبه گرادیان

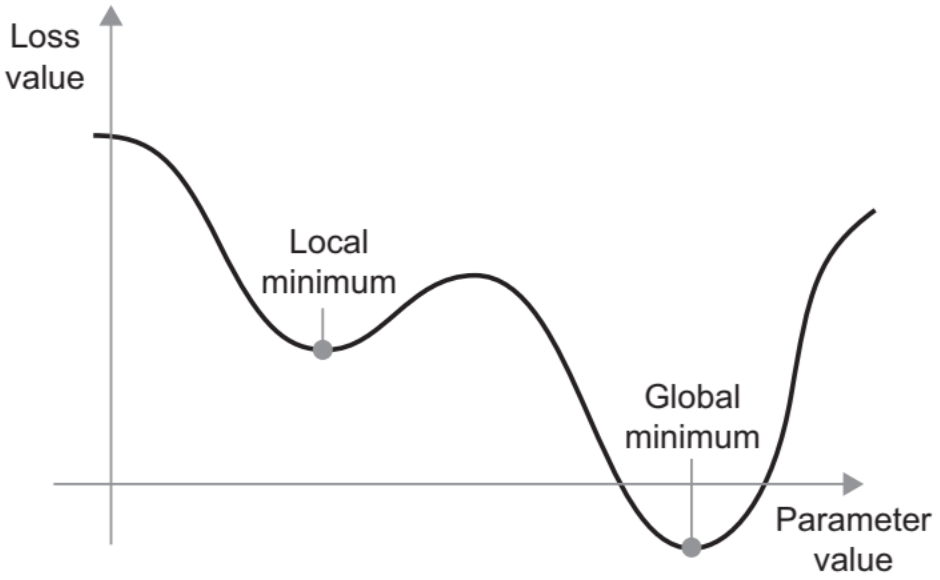
- به طور خلاصه:
  - محاسبه عددی گرادیان تقریبی، بسیار کند، اما به سادگی قابل پیاده‌سازی است
  - محاسبه تحلیلی گرادیان دقیق، سریع، اما مستعد خطا در محاسبات و پیاده‌سازی است

- در عمل:
  - حتما از محاسبه تحلیلی گرادیان استفاده می‌کنیم
  - اما پیاده‌سازی آن را با نتیجه حاصل از محاسبه عددی گرادیان چک می‌کنیم

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
```

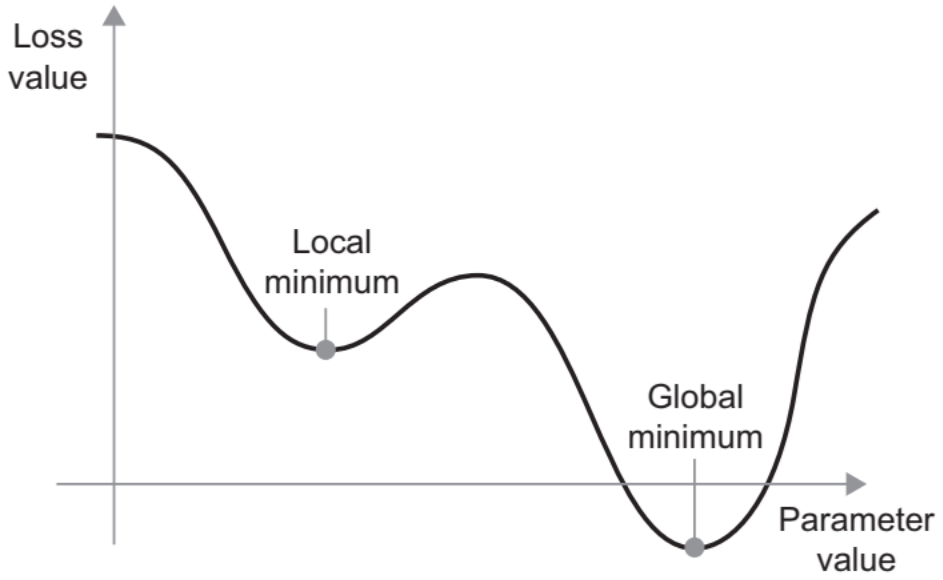


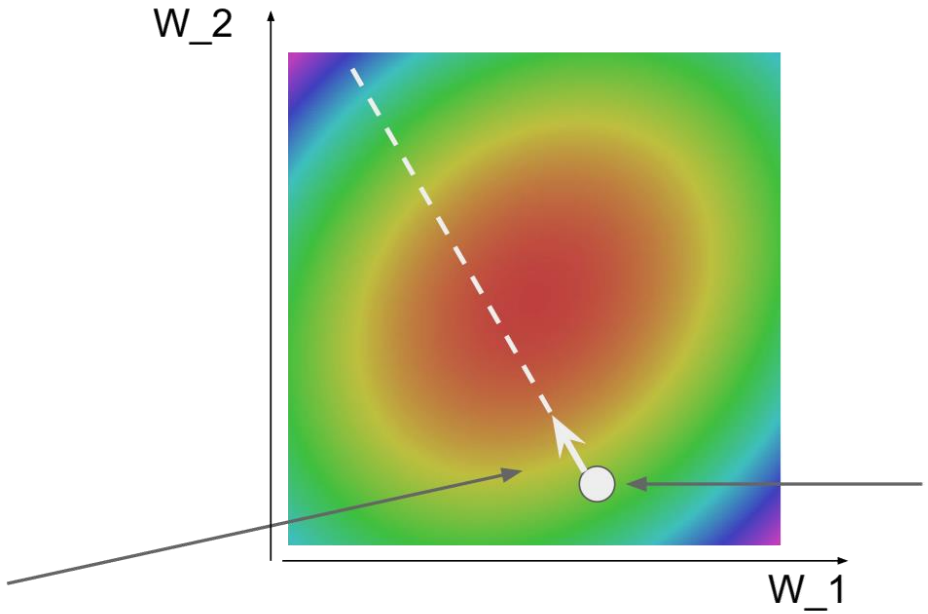
$$L(w+\Delta w) = f(w) + \Delta w \frac{\partial L(w)}{\partial w} + \frac{(\Delta w)^2}{2!}\frac{\partial^2 L(w)}{\partial w^2} + \cdots$$

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
```





original W

negative gradient direction

گرادیان کاهشی تصادفی (SGD)

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(s_i, y_i)$$

$$\nabla_W L = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(s_i, y_i)$$

$$s_i = f(x_i, W)$$

$$W = W - \eta \nabla_W L$$

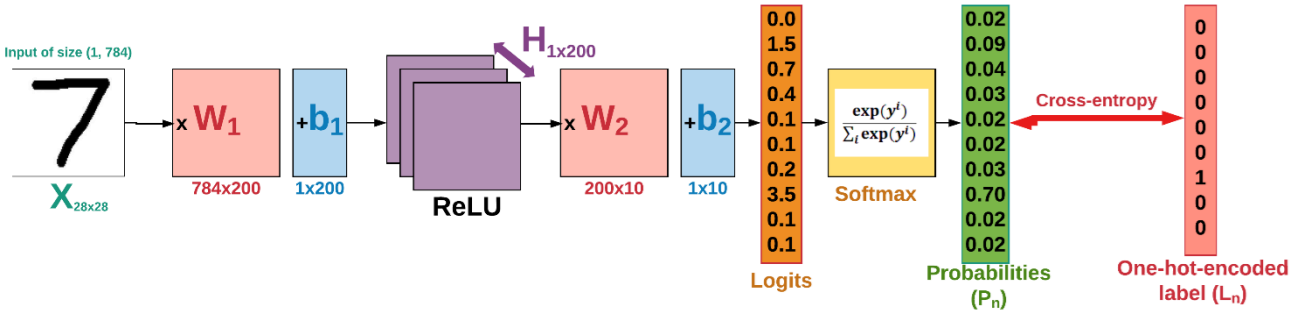
- محاسبه مجموع کامل اگر N بزرگ باشد بسیار پر هزینه است
- آن را با استفاده از یک minibatch از نمونه‌ها تقریب می‌زنیم
  - ۳۲/۶۴/۱۲۸ متداول هستند

```
# 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
```

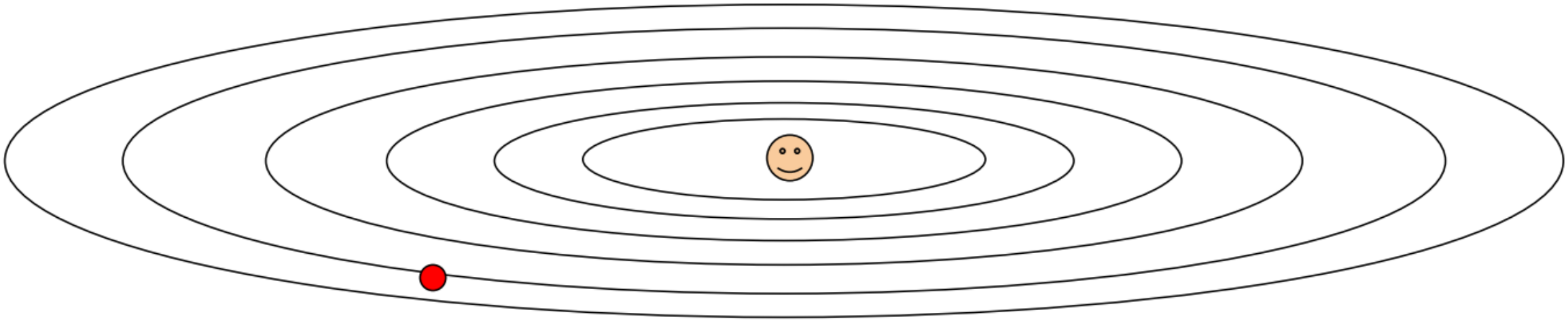
چرخه آموزش

- این تنظیم تدریجی، که آموزش نیز نامیده می شود، پایه یادگیری در بسیاری از الگوریتم‌های یادگیری ماشین است
  - یک batch از نمونه‌های آموزشی x و خروجی‌های مربوطه y انتخاب می‌شود
  - شبکه بر روی x اعمال می‌شود (forward pass) تا y_{pred} بدست بیاید
  - تابع ضرر شبکه برای این batch از مقایسه y و y_{pred} محاسبه می‌شود
  - تمام وزن‌های شبکه به گونه‌ای به روز می‌شوند که مقدار ضرر برای این batch کمی کاهش بیابد



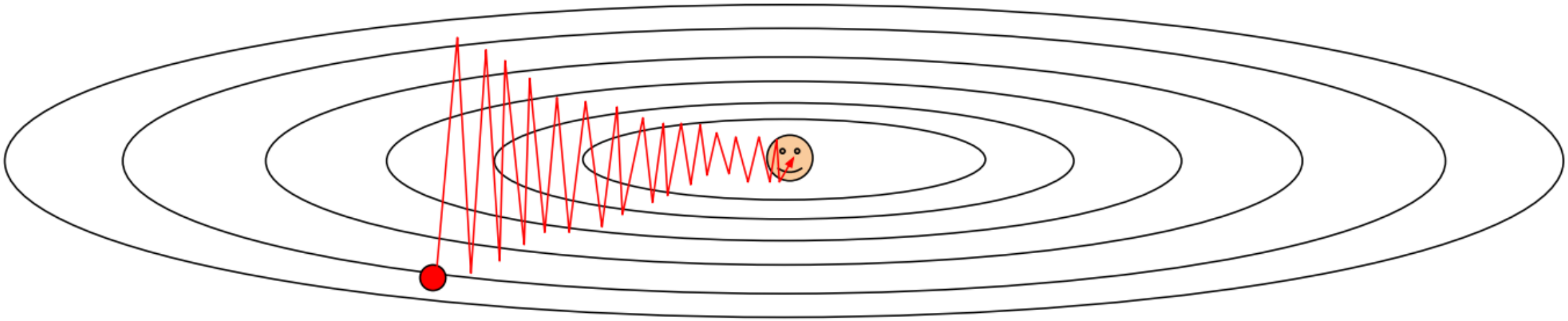
مشکلات SGD

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



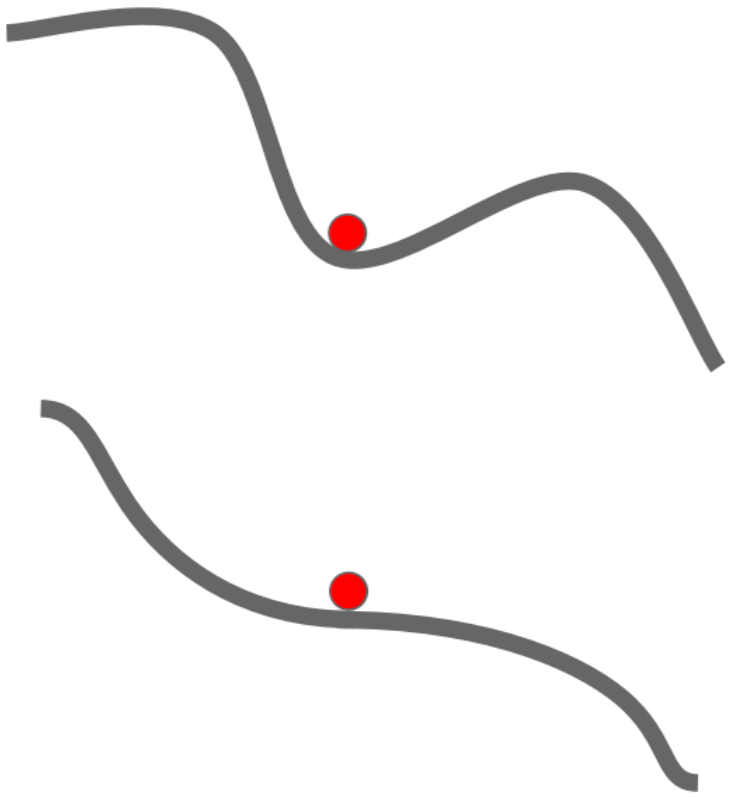
مشکلات SGD

- 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 می‌شود
  - این مشکل در SGD کمتر است زیرا بعید است گرادیان یک نقطه در تمام minibatchها صفر باشد

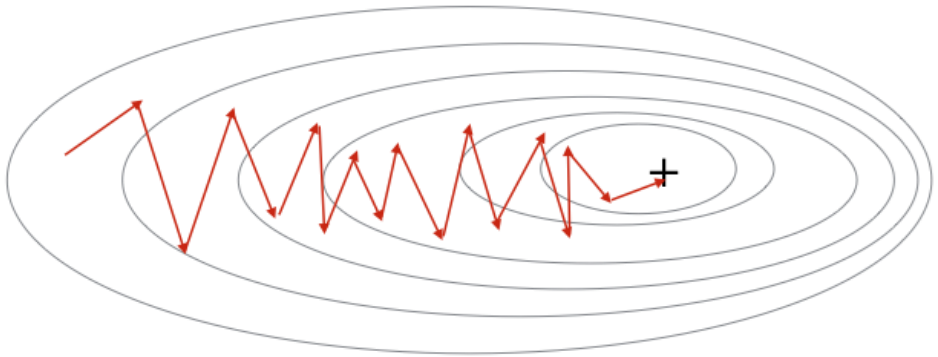


مشکلات SGD

$$\nabla_W L = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(s_i, y_i)$$

- گرادیان‌ها از minibatchها محاسبه می‌شوند و می‌توانند نویزی باشند!
- با وجود نویزی بودن گرادیان‌ها، به طور میانگین در مسیر درستی حرکت می‌کند

Stochastic Gradient Descent



Gradient Descent