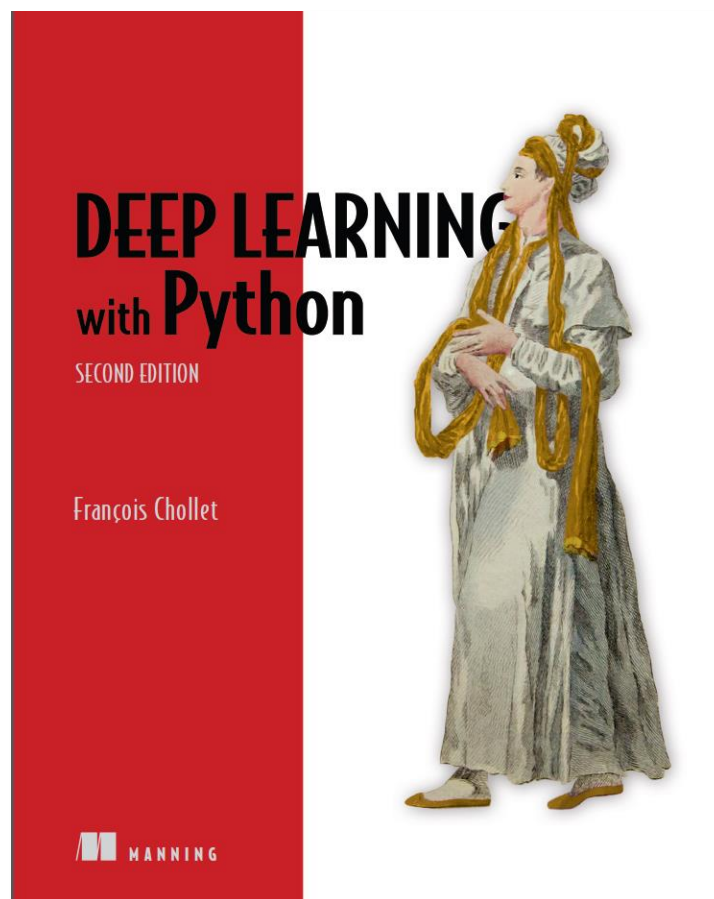


آموزش شبکه‌های عصبی عمیق (Training)



- مبانی یادگیری عمیق (Deep Learning)
- سعید محقق / زمستان 1400

• آموزش مبانی یادگیری عمیق (Deep Learning)

• آموزش شبکه‌های عصبی عمیق

DEEP LEARNING
with **Python**

SECOND EDITION

François Chollet

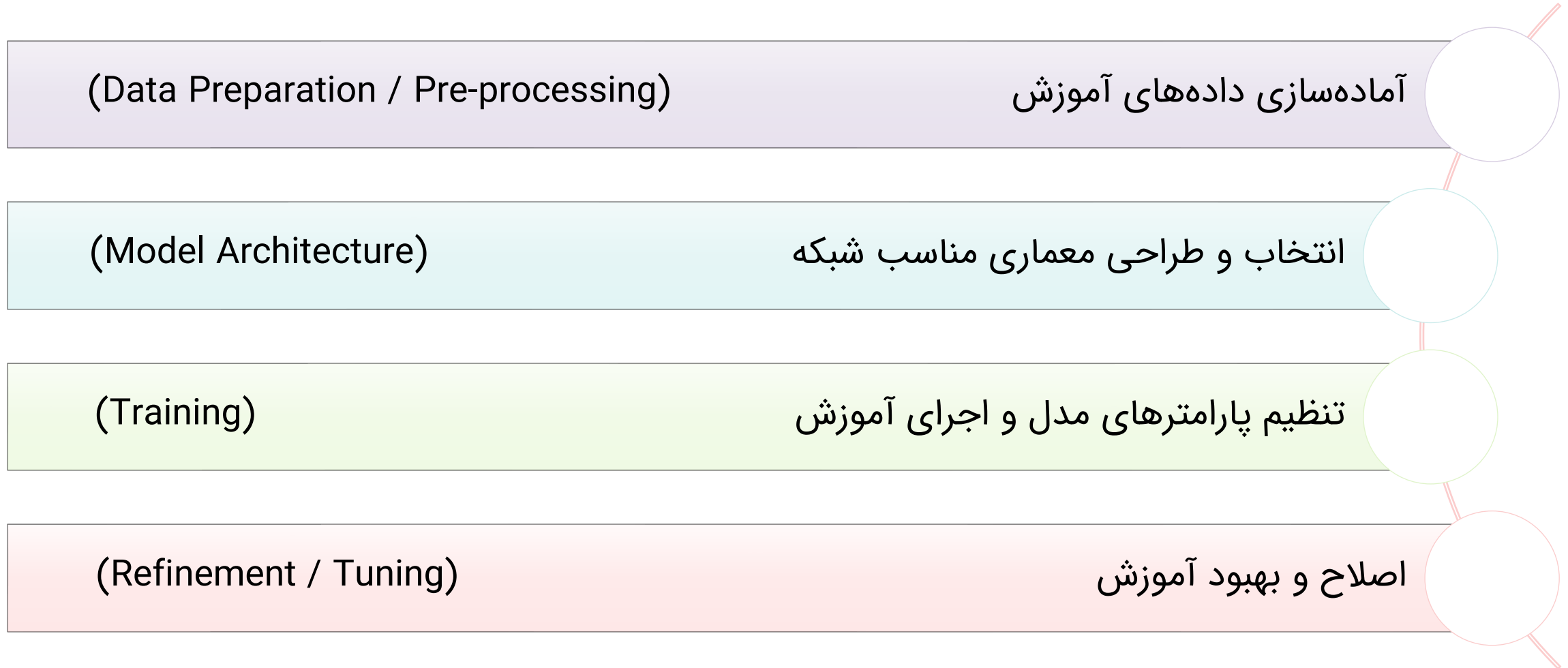


مدرس:

• سعید محققى

برنامه‌نویس و دکتری مهندسی پزشکی بیوالکتریک

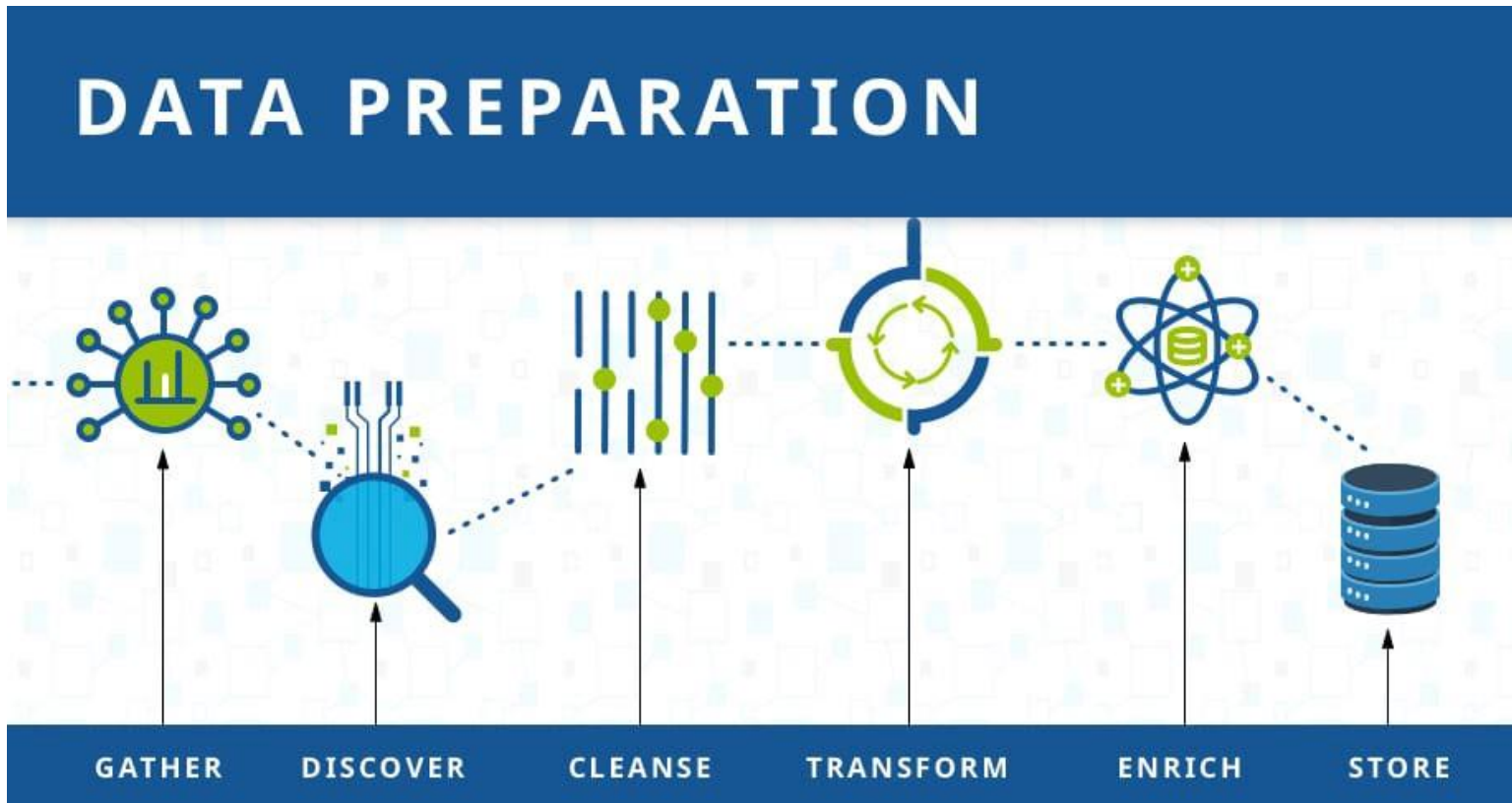
گام‌های اصلی فرآیند آموزش



اصول آماده‌سازی داده‌ها

• داده‌ها ← تنها منبع مدل برای یادگیری (منبع تغذیه و سوخت مدل)

• داده خوب ← آموزش خوب

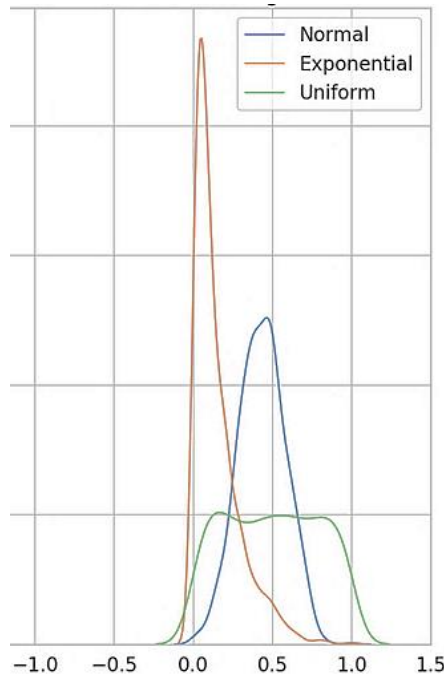
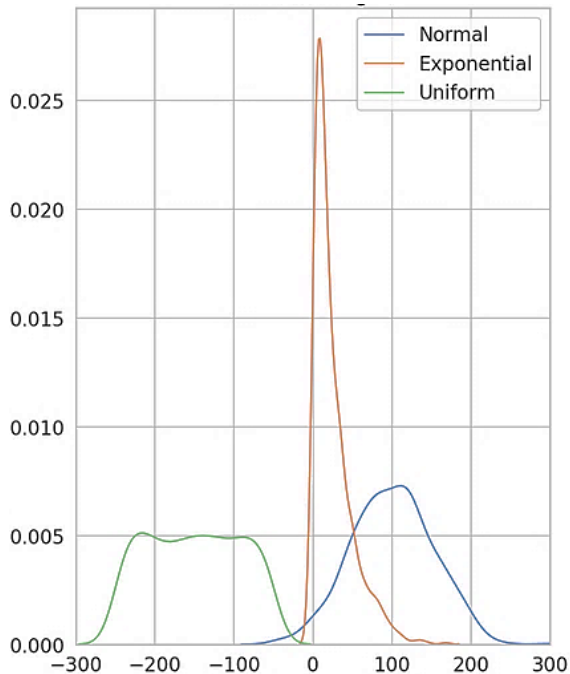


پیش‌پردازش داده‌ها

• نرمالیزه / استاندارد کردن داده‌ها ← حذف اطلاعات گمراه‌کننده

▪ انتقال محدوده مقادیر داده‌ها به محدوده $[0, 1]$

▪ تغییر داده‌ها برای داشتن میانگین ۰ و واریانس ۱



$$x_{normalized} = \frac{(x - \min(x))}{(\max(x) - \min(x))}$$

$$x_{standardized} = \frac{(x - \text{mean}(x))}{\text{std}(x)}$$

پیش‌پردازش داده‌ها

- بهبود کیفیت تصاویر



افزایش داده‌ها

• Data Augmentation

- ایجاد تغییرات بر روی داده‌های فعلی و ایجاد داده‌های جدید
- مثال: جابه‌جایی / تغییر اندازه / چرخش / تغییر شکل و رنگ



انتخاب معماری شبکه

نوع و معماری شبکه

- استفاده از معماری مدل‌های آماده

تعداد لایه‌ها / تعداد پارامترها

- شروع با تعداد کم ← افزایش، تا جایی که نتیجه بهبود پیدا نکند. (یا برعکس!)

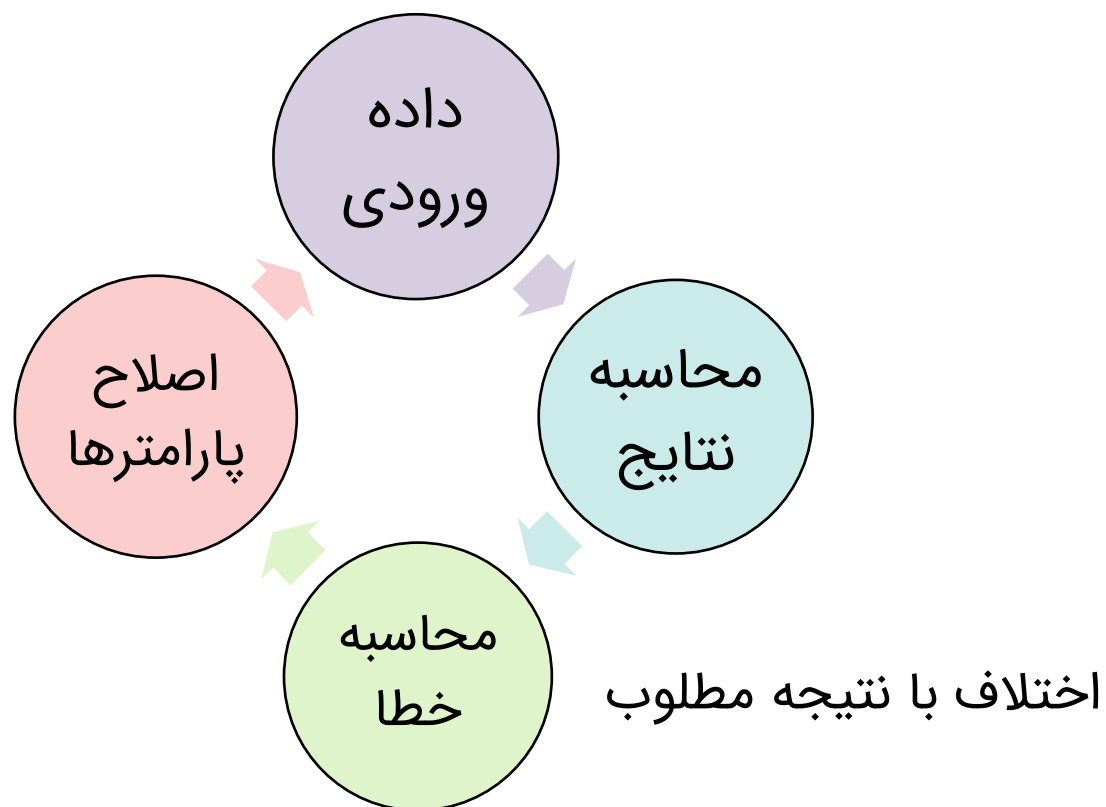
الگوریتم‌های جستجوی معماری (Neural Architecture Search)

- روش‌های AutoML (TPOT، Auto-Sklearn، Auto-Keras)

مدل‌های آماده برای Keras: <https://keras.io/api/applications/>

آموزش شبکه

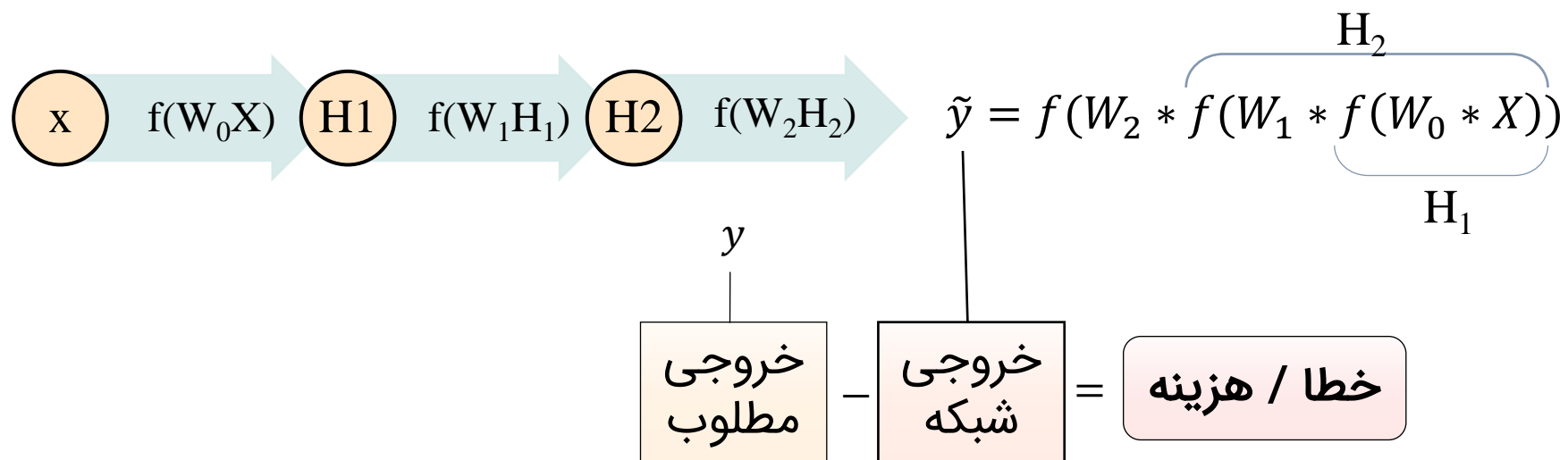
• چرخه آموزش



تکرار چرخه آموزش تا رسیدن به نتیجه مطلوب

آموزش شبکه

- عبور داده‌ها از شبکه و محاسبه خطا (Forward Pass)



محاسبه خطا (Loss)

- تابع هزینه ← نحوه محاسبه خطا

- نمونه‌های تابع هزینه (Cost / Loss function)

Mean Squared Error (MSE)

$$L(y, \tilde{y}) = \frac{1}{N} \sum_i (y_i - \tilde{y}_i)^2$$

Binary Cross-Entropy (BCE)

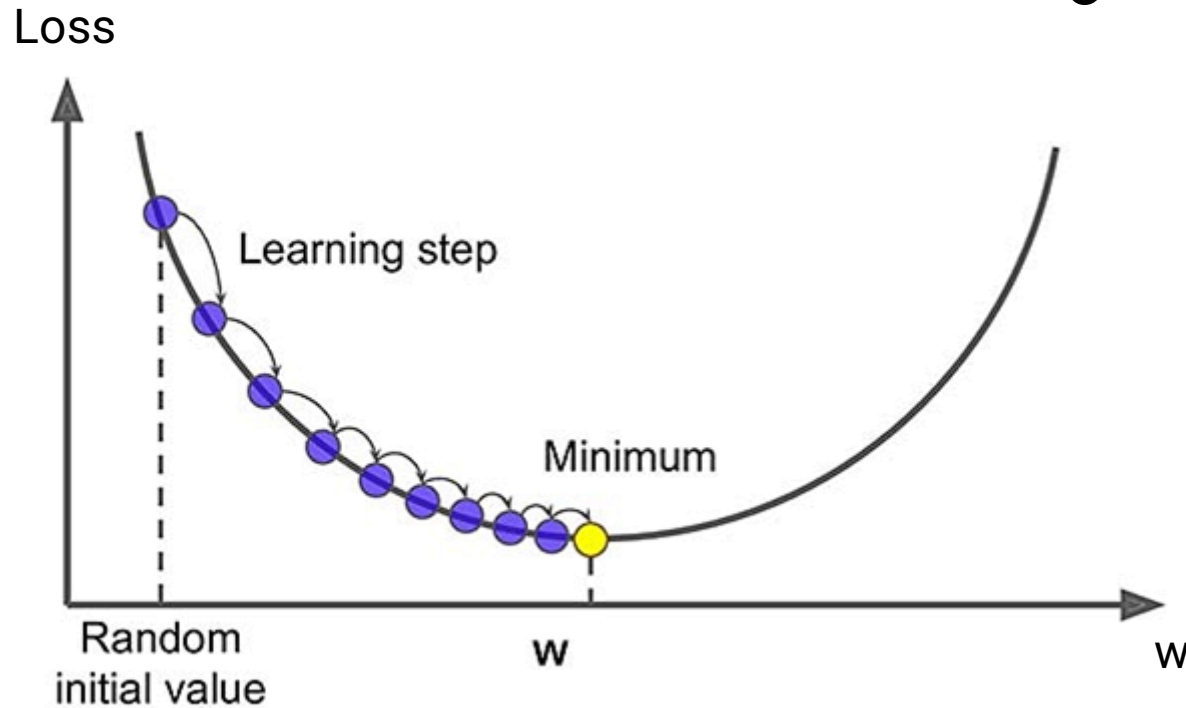
$$L(y, \tilde{y}) = \frac{1}{N} \sum_i y_i \cdot \log(\tilde{y}_i) + (1 - y_i) \cdot \log(1 - \tilde{y}_i)$$

Dice Similarity Coefficient (DSC)

$$L(y, \tilde{y}) = 1 - \frac{2 (y \cap \tilde{y})}{y \cup \tilde{y}}$$

بهینه‌سازی (Optimization)

- حرکت قدم‌به‌قدم به سمت کم‌ترین مقدار خطا
- اصلاح و به‌روزرسانی وزن‌ها برای رسیدن به حداقل خطا

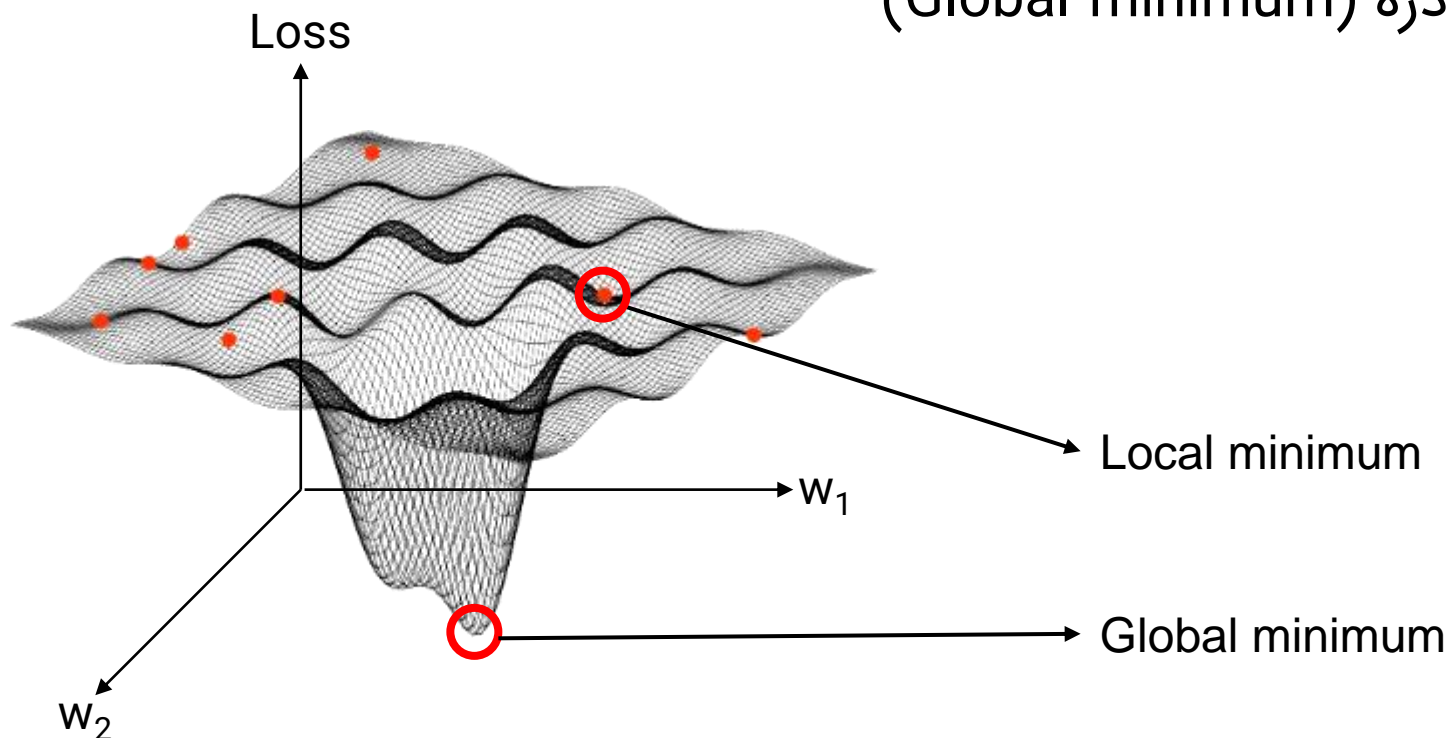


الگوریتم Gradient Descent

$$w_{i+1} = w_i - \alpha \frac{\partial Loss}{\partial w_i}$$

بهینه‌سازی (Optimization)

- مقادیر مختلف Loss در وزن‌های مختلف \leftarrow تشکیل یک سطح ناهموار (Loss surface)
- هدف \leftarrow رسیدن به عمیق‌ترین دره (Global minimum)



بهینه‌سازی (Optimization)

الگوریتم‌ها

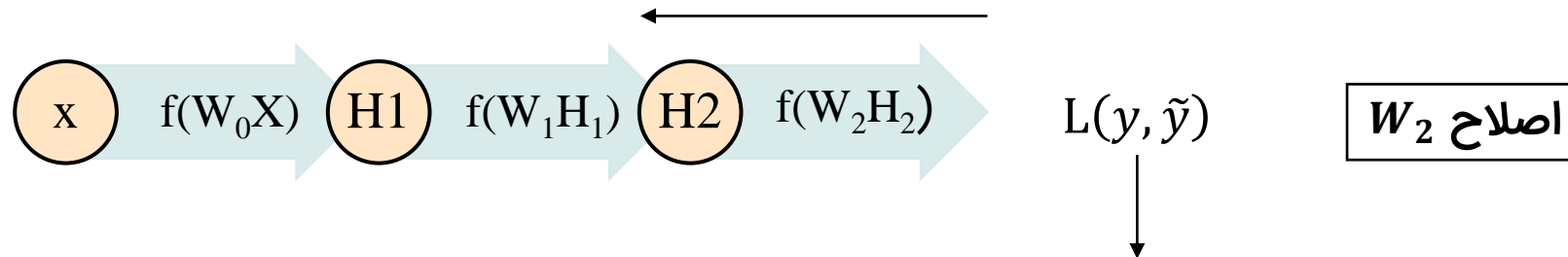
- SGD
- SGD + Momentum
- RMSprop
- Adagrad
- Adadelta
- Adam
- ...

• بررسی الگوریتم‌های بهینه‌سازی:

<http://ruder.io/optimizing-gradient-descent/>

اصلاح وزن‌ها

• روش Back-propagation (Backward Pass)



$$\Delta W_2 = \frac{\partial L}{\partial W_2}$$

$$W_2' = W_2 - \alpha(\Delta W_2)$$

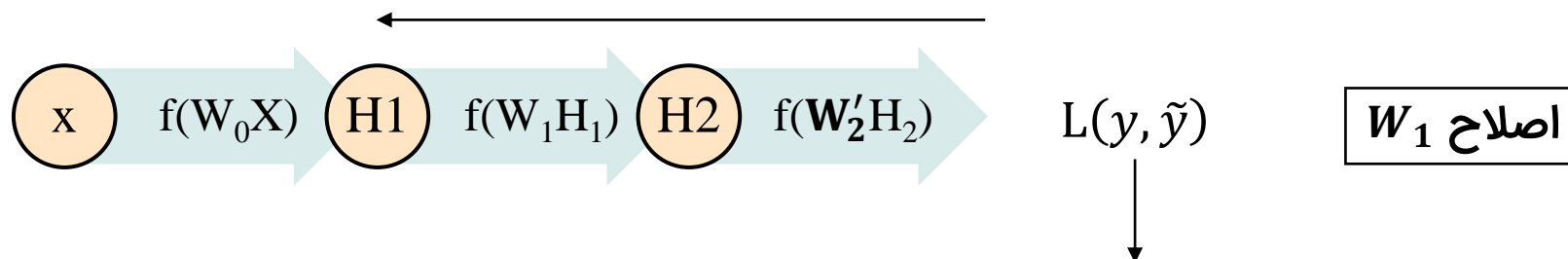
↓
Learning rate

• محاسبه گرادیان L نسبت به وزن W_2 :

• اصلاح W_2 در جهت کاهش گرادیان L (الگوریتم SGD):

اصلاح وزن‌ها

• روش Back - propagation



$$\Delta W_1 = \frac{\partial L}{\partial W_1}$$

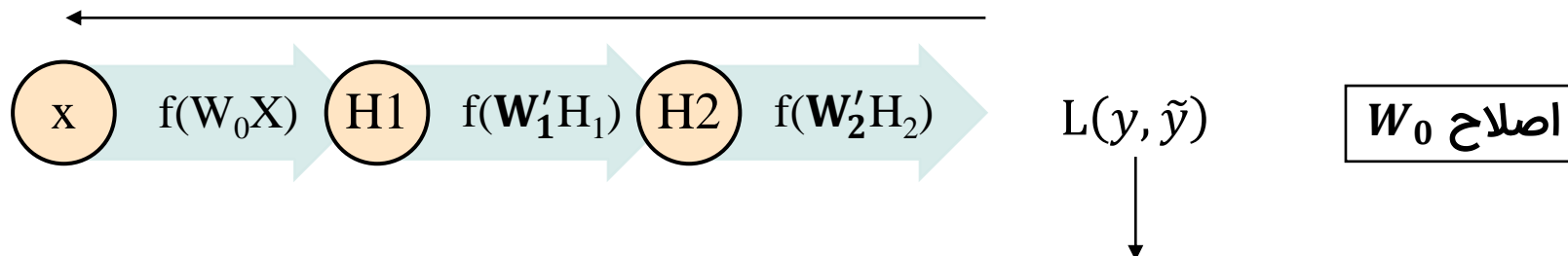
$$W_1' = W_1 - \alpha(\Delta W_1)$$

• محاسبه گرادیان L نسبت به وزن W_1 :

• اصلاح W_1 در جهت کاهش گرادیان L :

اصلاح وزن‌ها

• روش Back - propagation



$$\Delta W_0 = \frac{\partial L}{\partial W_0}$$

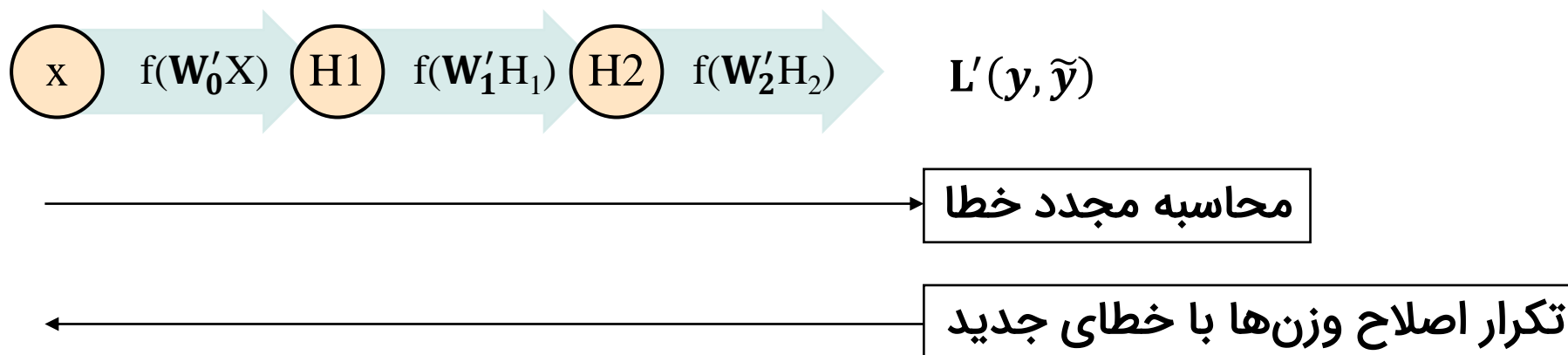
$$W'_0 = W_0 - \alpha(\Delta W_0)$$

• محاسبه گرادیان L نسبت به وزن W_0 :

• اصلاح W_0 در جهت کاهش گرادیان L :

اصلاح وزن‌ها

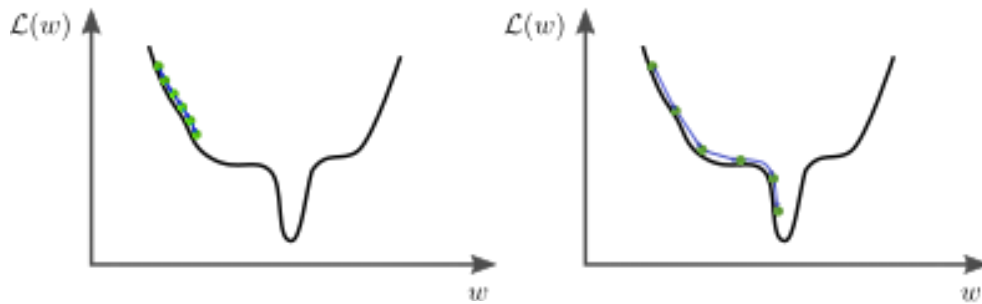
• روش Back - propagation



نرخ یادگیری

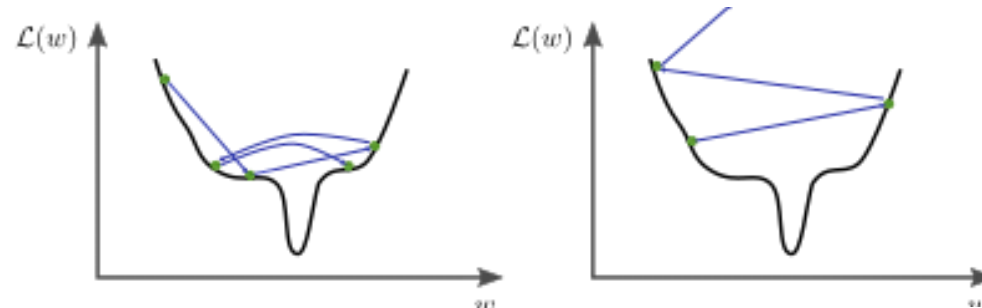
Learning Rate •

• اندازه گامها در حرکت به سمت حداقل خطا



Learning rate too low

Good learning rate



High learning rate

Learning rate much too high

