۲- آموزش شبکه‌های عصبی عمیق (Training)



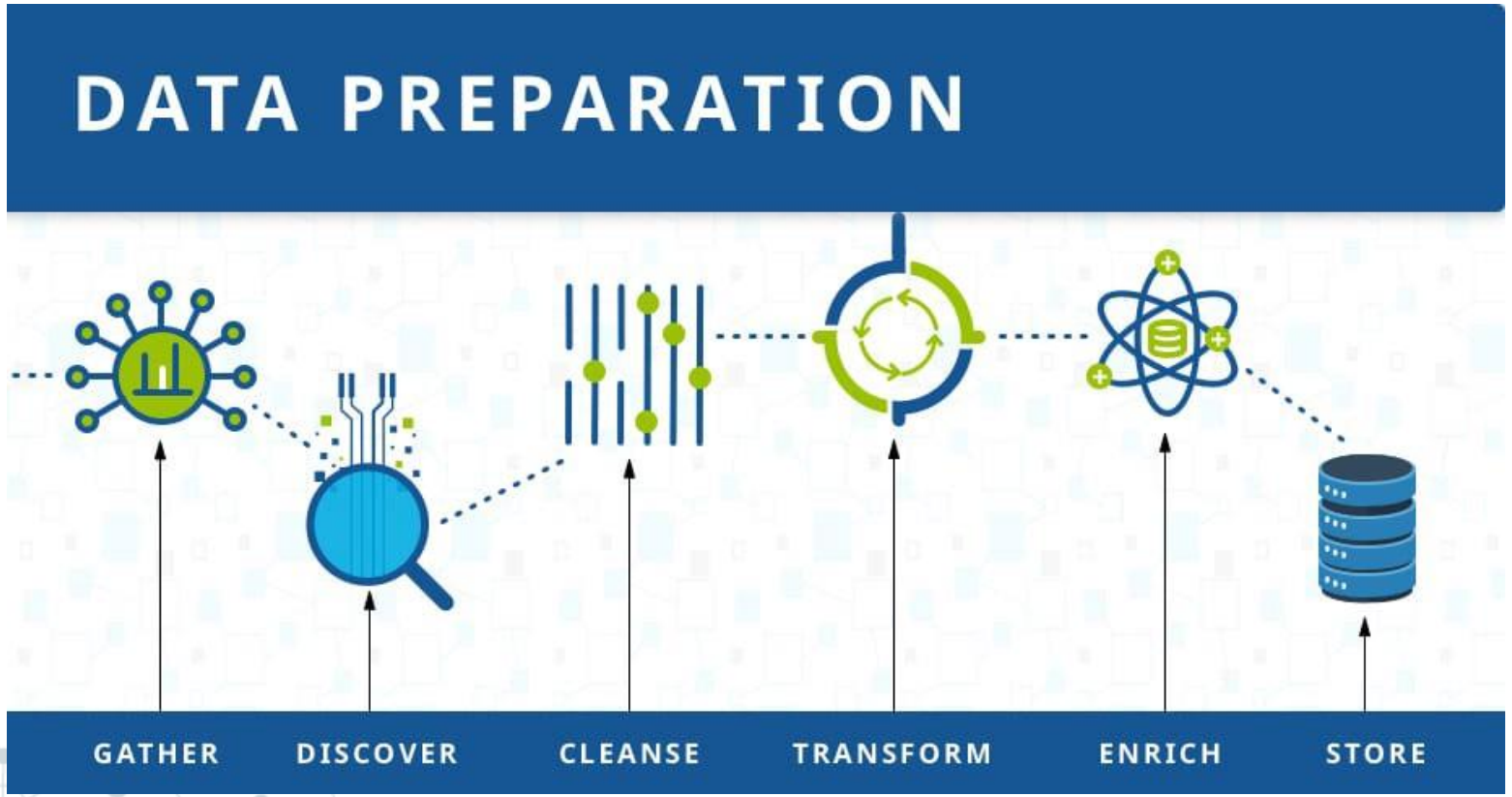

گام‌های اصلی فرآیند آموزش

آماده‌سازی داده‌های آموزش	(Data Preparation / Pre-processing)
انتخاب / طراحی معماری مناسب شبکه	(Model Architecture)
تنظیم پارامترهای مدل و اجرای آموزش	(Training)
اصلاح و بهبود آموزش	(Refinement / Tuning)

اصول آماده‌سازی داده‌ها

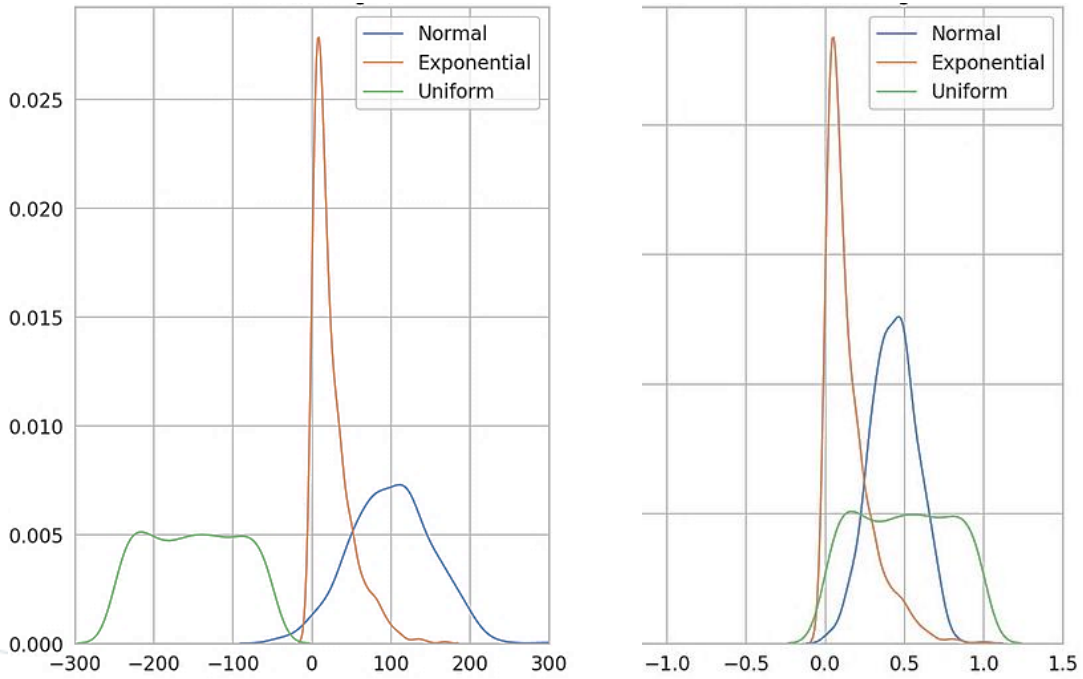
داده‌ها ← تنها منبع مدل برای یادگیری (منبع تغذیه و سوخت مدل)

داده خوب ← آموزش خوب



پیش‌پردازش داده‌ها

نرمالیزه / استانداردکردن داده‌ها ← حذف اطلاعات گمراه‌کننده

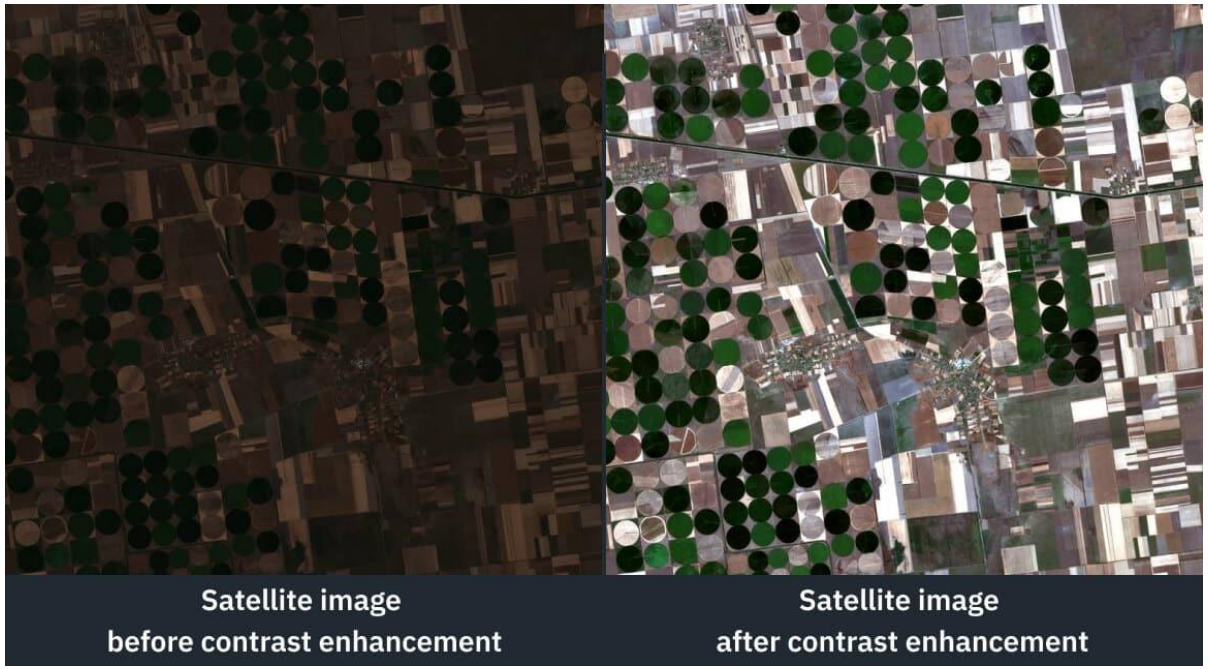


$$x_{normalized} = \frac{(x - \min(x))}{(\max(x) - \min(x))}$$

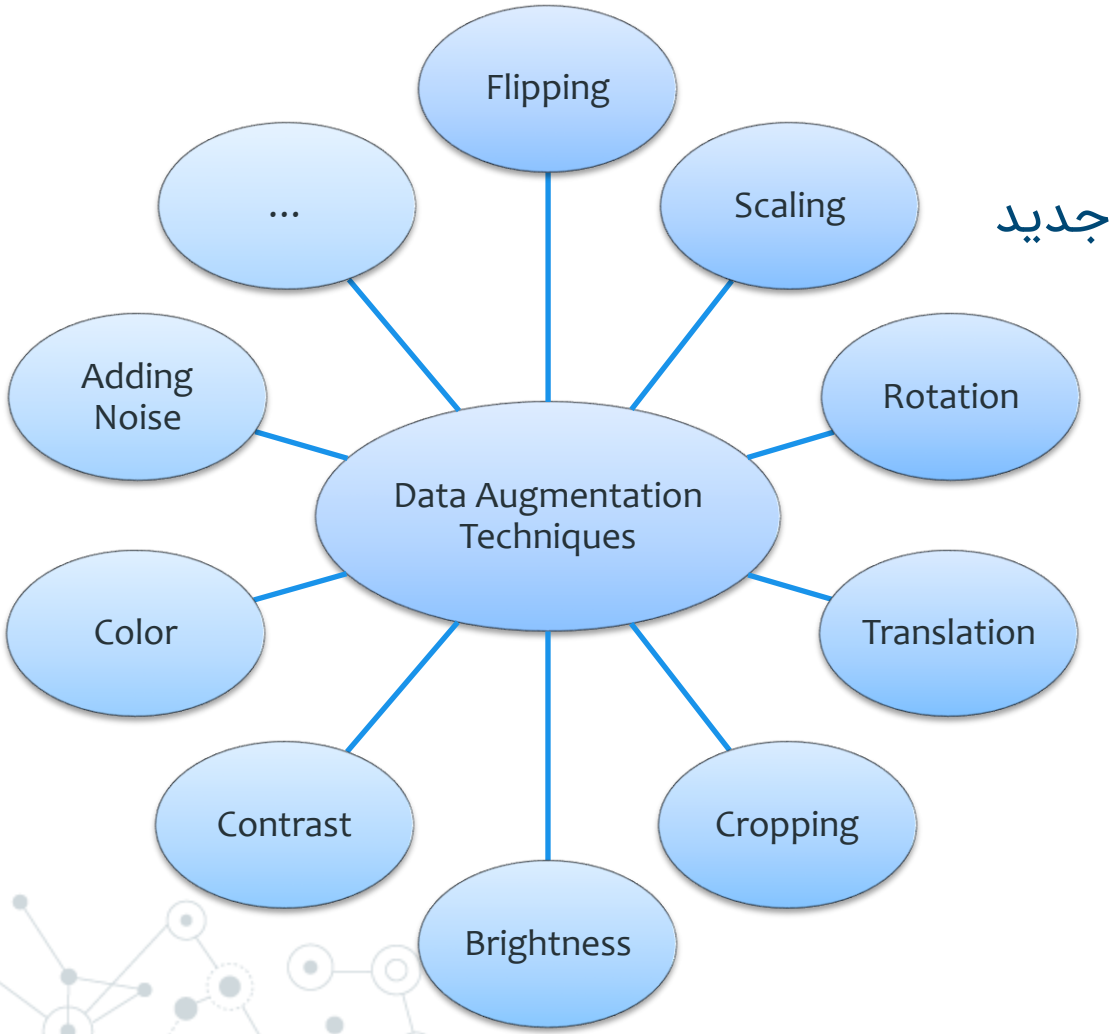
$$x_{standardized} = \frac{(x - \text{mean}(x))}{\text{std}(x)}$$

پیش‌پردازش داده‌ها

بهبود کیفیت تصاویر

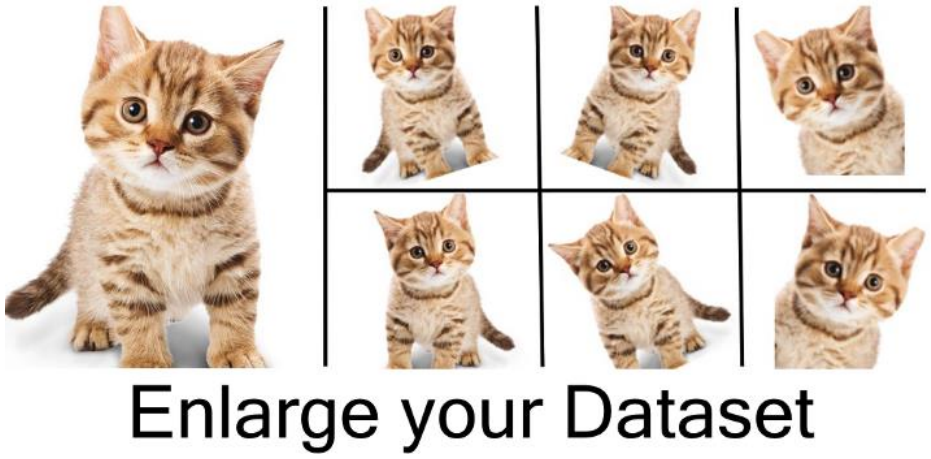


افزایش داده‌ها



Data Augmentation

ایجاد تغییرات بر روی داده‌های فعلی و ایجاد داده‌های جدید



انتخاب / طراحی معماری مدل

نوع و معماری شبکه

- با توجه به نوع داده های آموزش
- قابلیت استفاده از معماری مدل‌های آماده

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

تعداد لایه‌ها / تعداد پارامترها

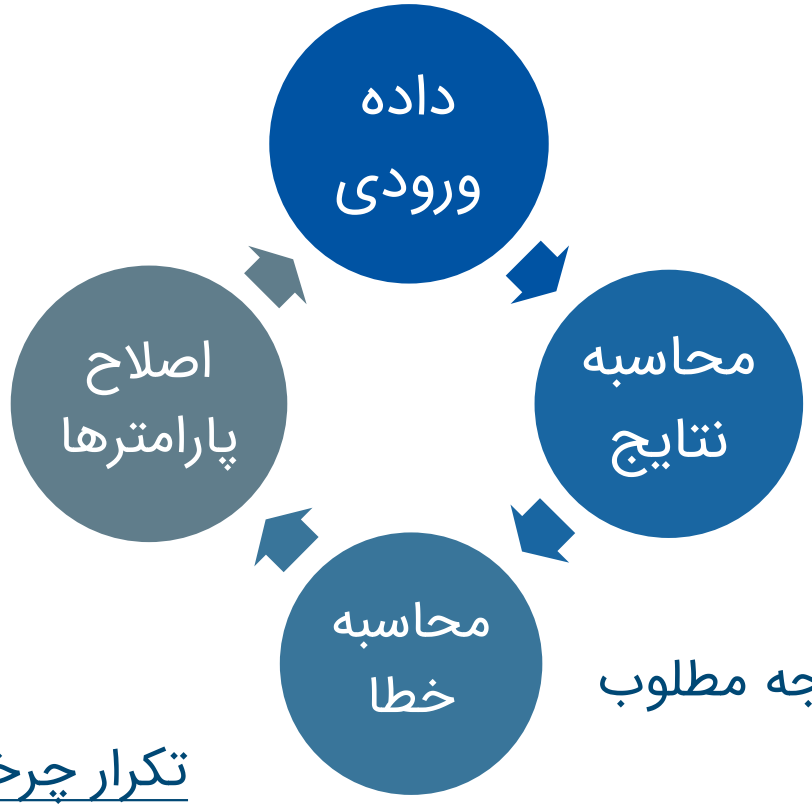
- با توجه به حجم داده های آموزش
- شروع با تعداد کم ← افزایش، تا جایی که نتیجه بهبود پیدا نکند. (یا برعکس!)

الگوریتم‌های جستجوی معماری (Neural Architecture Search)

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

آموزش شبکه

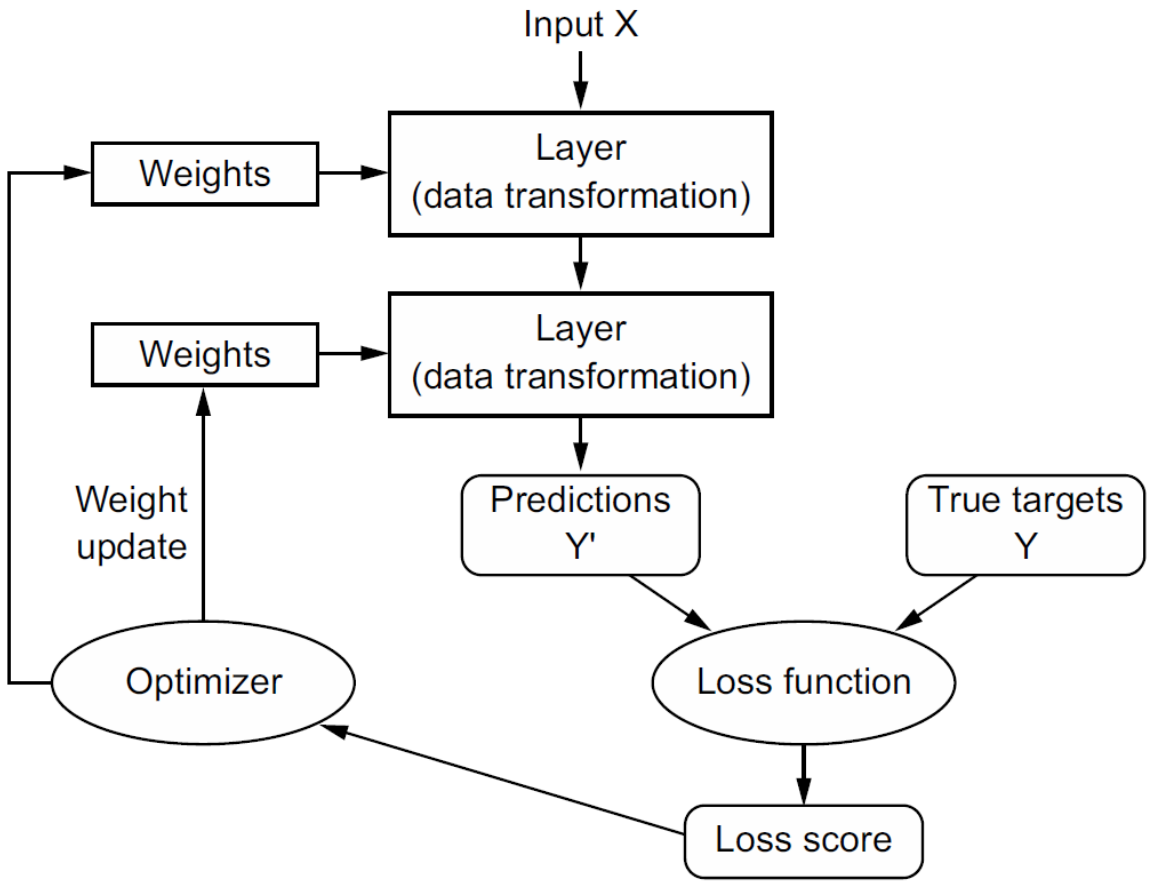
چرخه آموزش



اختلاف با نتیجه مطلوب

تکرار چرخه آموزش تا رسیدن به نتیجه مطلوب

آموزش شبکه

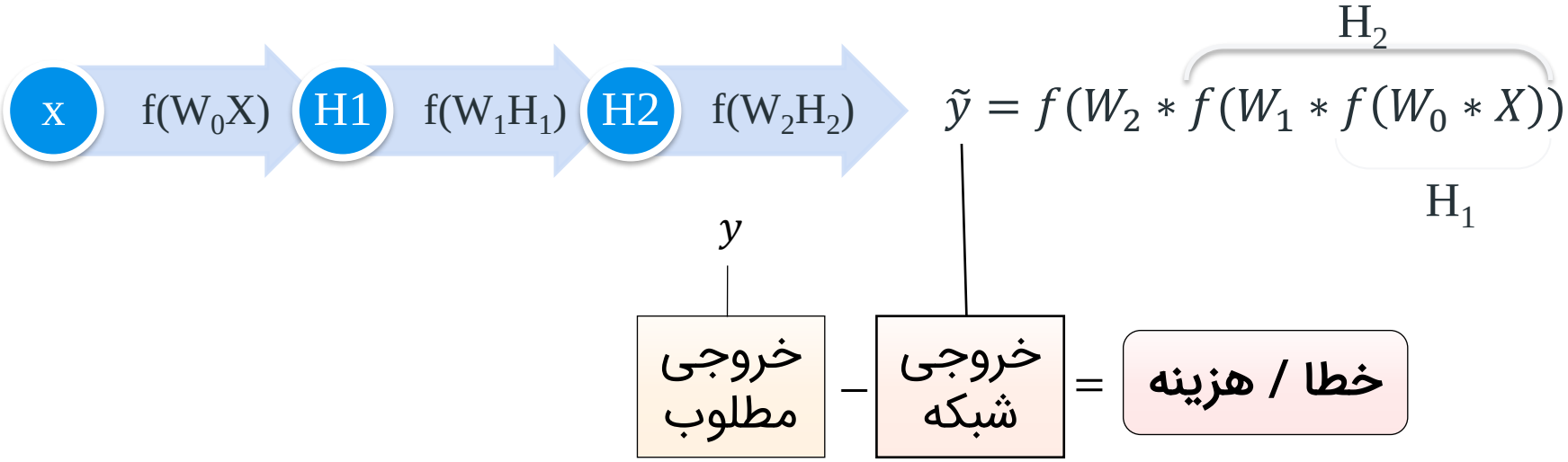


چرخه آموزش

- ورودی / خروجی
- لایه ها
- تابع هزینه
- الگوریتم بهینه سازی

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)

$$L(y, \tilde{y}) = \frac{1}{N}\sum_{i} y_i . \log(\tilde{y}_i) + (1 - y_i) . \log(1 - \tilde{y}_i)$$

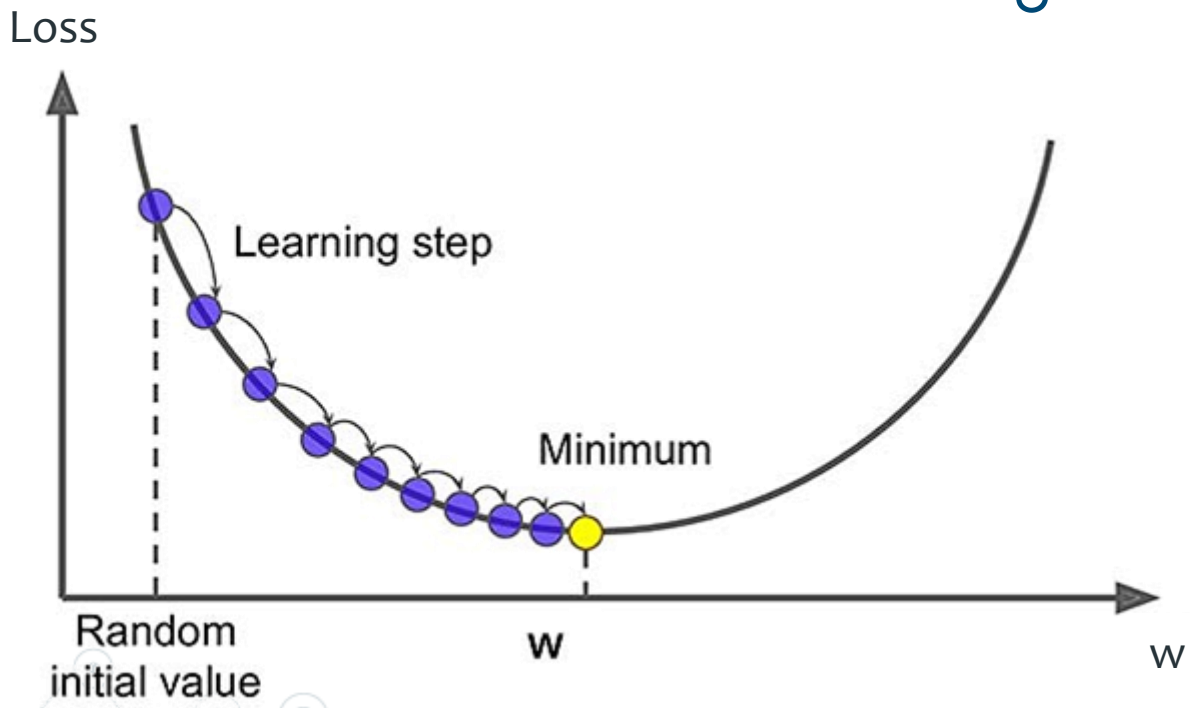
Dice Similarity Coefficient (DSC)

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

بهینه‌سازی (Optimization)

حرکت قدم‌به‌قدم به سمت کم‌ترین مقدار خطا

اصلاح و به‌روزرسانی وزن‌ها برای رسیدن به حداقل خطا

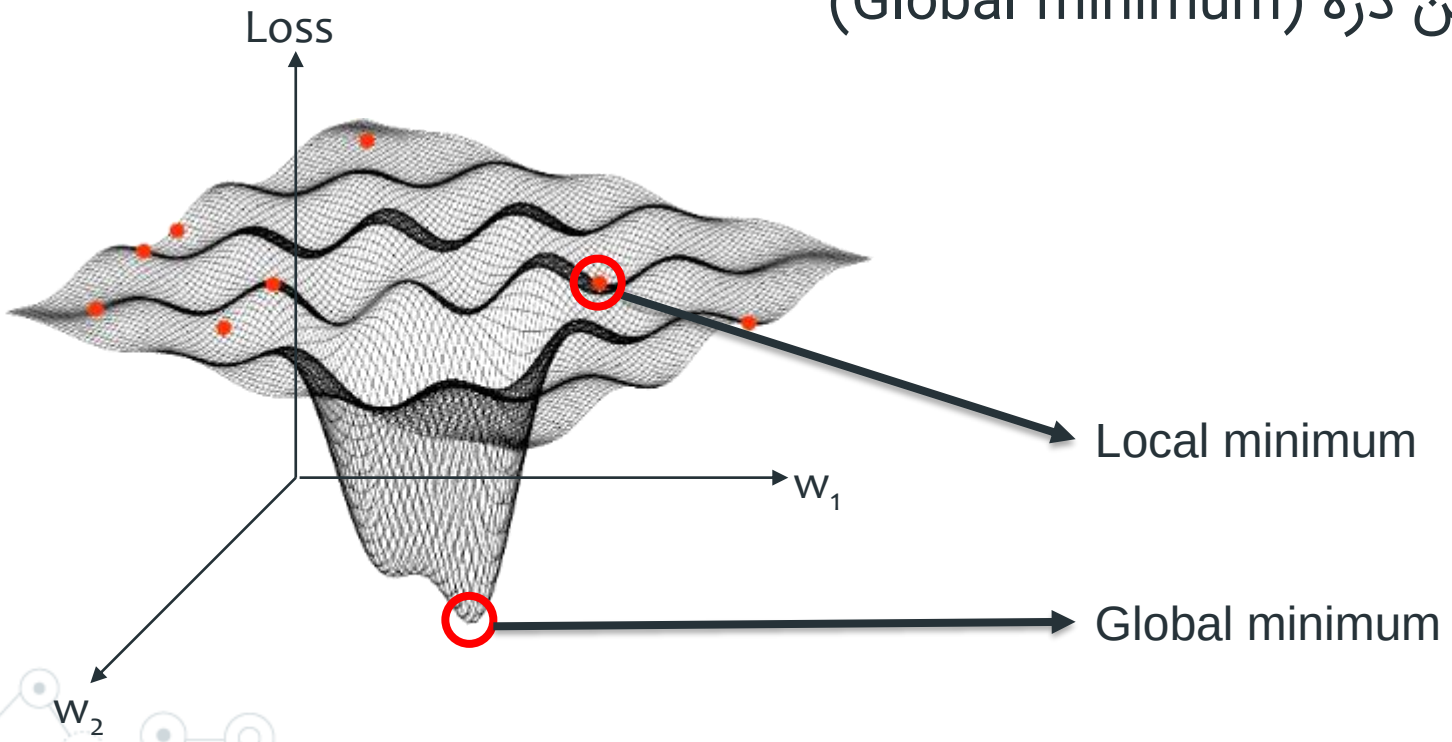


الگوریتم Gradient Descent

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

بهینه‌سازی (Optimization)

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



بهینه‌سازی (Optimization)

الگوریتم‌ها

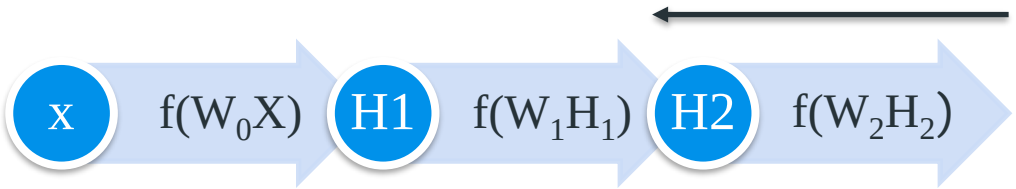
- SGD
- SGD + Momentum
- RMSprop
- Adagrad
- Adadelta
- Adam
- ...

- بررسی الگوریتم‌های بهینه‌سازی:

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

اصلاح وزن‌ها

روش Back–propagation (Backward Pass)



$L(y, \tilde{y})$ اصلاح W_2

- محاسبه گرادیان L نسبت به وزن W_2:

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

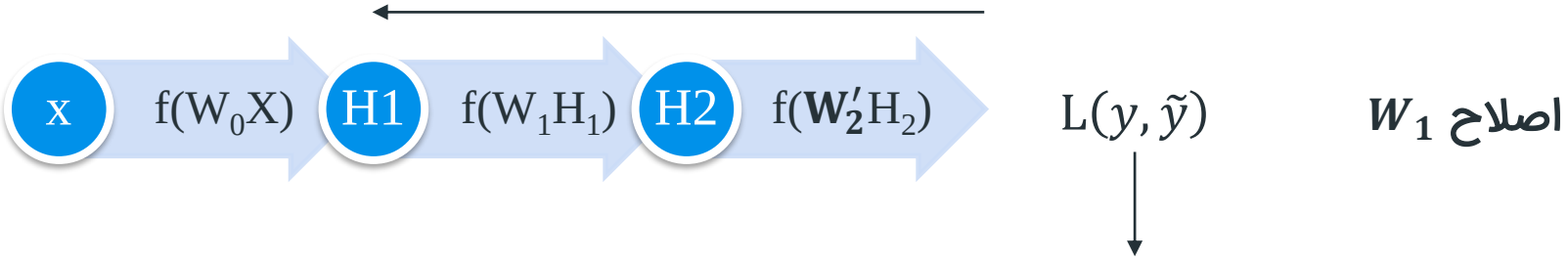
- اصلاح W_2 در جهت کاهش گرادیان L (الگوریتم SGD):

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

Learning rate

اصلاح وزن‌ها

روش Back–propagation (Backward Pass)



- محاسبه گرادیان L نسبت به وزن W_1:

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

- اصلاح W_1 در جهت کاهش گرادیان L:

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

اصلاح وزن‌ها

روش Back–propagation (Backward Pass)

$L(y, \tilde{y})$

اصلاح W_0

- محاسبه گرادیان L نسبت به وزن W_0 :

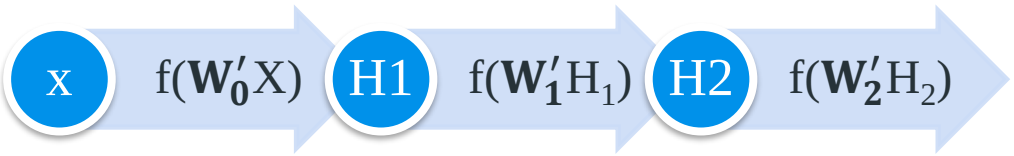
$$\Delta W_0 = \frac{\partial L}{\partial W_0}$$

- اصلاح W_0 در جهت کاهش گرادیان L:

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

اصلاح وزن‌ها

روش Back–propagation



محاسبه مجدد خطا (Forward Pass) ←

اصلاح وزن‌ها با خطای جدید (Backward Pass) →

نرخ یادگیری

Learning Rate