Short introduction to ML and DL

May 07, 2025

Learning Tasks

Supervised learning

- Task: to learn a mapping f from inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$.
  - x: features
  - y: label, target.
  - $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$: training set.
- Model: $f(x; \theta)$, where θ is a parameter.
- Goal: find the optimal model $f(x; \theta^*)$ with the optimal parameter θ^*.

Classification (pattern recognition)

- $\mathcal{Y} = \{1, 2, \ldots, C\}$
- empirical risk: $\mathcal{L}(\theta) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n; \theta))$
- training: solve

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta)\,. \tag{1}$$

Classification (pattern recognition)

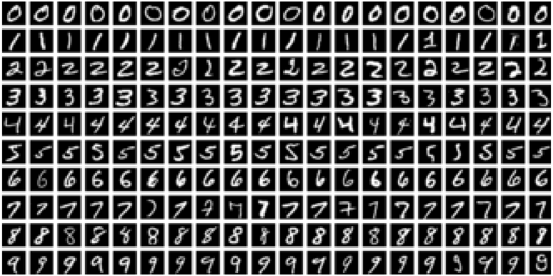
- $\mathcal{Y} = \{1, 2, \ldots, C\}$
- empirical risk: $\mathcal{L}(\theta) := \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n; \theta))$
- training: solve

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta)\,. \tag{1}$$

Example:

1. image classification
2. misclassification rate: $\mathcal{L}(\theta) := \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(y_n \neq f(x_n; \theta))$
3. maximum likelihood estimation: $\mathcal{L}(\theta) := -\frac{1}{N}\sum_{n=1}^{N} \log p(y_n | f(x_n; \theta))$

Image classification

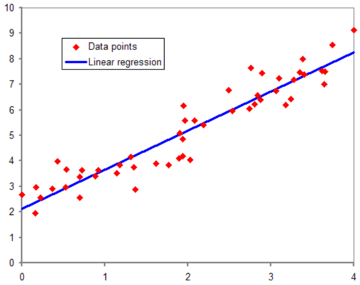


Regression

- $\mathcal{Y} = \mathbb{R}$
- mean square error: $\mathrm{MSE}(\theta) = \frac{1}{N}\sum_{n=1}^{N} |y_n - f(x_n; \theta)|^2$

Regression

- $\mathcal{Y} = \mathbb{R}$
- mean square error: $\mathrm{MSE}(\theta) = \frac{1}{N} \sum_{n=1}^{N} |y_n - f(x_n; \theta)|^2$



Regression

Examples:

- linear regression:

$$f(x;\theta) = b + w^\top x, \quad \theta = (b, w).$$

- polynomial regression:

$$f(x;\theta) = w_0 + \sum_{i=1}^{D} w_i x^i, \quad \theta = (w_0, w_1, \ldots, w_D).$$

- deep neural networks:

$$f(x;\theta) = f_L(f_{L-1}(\cdots(f_1(x))\cdots)).$$

Unsupervised learning

Data: $\mathcal{D} = \{x_n\}_{n=1}^N$.

Examples:

- clustering: k-means
- discovering latent structure: PCA
- generative modeling

Training algorithms

Gradient descent

Optimization objective:

$$\mathcal{L}(\theta) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n; \theta)).$$

Gradient descent

Optimization objective:

$$\mathcal{L}(\theta) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n; \theta)).$$

Gradient descent:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) \\ &= \theta_t - \frac{\eta_t}{N} \sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n; \theta_t)). \end{aligned} \tag{2}$$

Gradient descent

Optimization objective:

$$\mathcal{L}(\theta) := \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n; \theta)).$$

Gradient descent:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) \\ &= \theta_t - \frac{\eta_t}{N}\sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n; \theta_t)). \end{aligned} \tag{2}$$

Euler scheme for the ODE $\frac{d\theta_t}{dt} = -\nabla_\theta \mathcal{L}(\theta_t)$.

Gradient descent

Optimization objective:

$$\mathcal{L}(\theta) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n; \theta)).$$

Gradient descent:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) \\ &= \theta_t - \frac{\eta_t}{N} \sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n; \theta_t)). \end{aligned} \tag{2}$$

Euler scheme for the ODE $\frac{d\theta_t}{dt} = -\nabla_\theta \mathcal{L}(\theta_t)$.

Complexity per step: N.

Stochastic gradient descent

$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n; \theta))$.

Stochastic gradient descent

$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n;\theta))$.

Let $\mathcal{B}_t$ be a set of randomly chosen indices with uniform probability.

min-batch SGD:

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{n \in \mathcal{B}_t} \nabla_\theta \ell(y_n, f(x_n;\theta_t))$$

Stochastic gradient descent

$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n;\theta))$.

Let $\mathcal{B}_t$ be a set of randomly chosen indices with uniform probability.

min-batch SGD:

$$
\begin{aligned}
\theta_{t+1} &= \theta_t - \frac{\eta_t}{|\mathcal{B}_t|}\sum_{n\in\mathcal{B}_t} \nabla_\theta \ell(y_n, f(x_n;\theta_t)) \\
&= \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) + \eta_t\Big(\nabla_\theta \mathcal{L}(\theta_t) - \frac{1}{|\mathcal{B}_t|}\sum_{n\in\mathcal{B}_t} \nabla_\theta \ell(y_n, f(x_n;\theta_t))\Big) \\
&= \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) + \eta_t e_t
\end{aligned}
$$

Stochastic gradient descent

$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \ell(y_n, f(x_n; \theta))$.

Let $\mathcal{B}_t$ be a set of randomly chosen indices with uniform probability.

min-batch SGD:

$$
\begin{aligned}
\theta_{t+1} =& \theta_t - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{n \in \mathcal{B}_t} \nabla_\theta \ell(y_n, f(x_n; \theta_t)) \\
=& \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) + \eta_t \left(\nabla_\theta \mathcal{L}(\theta_t) - \frac{1}{|\mathcal{B}_t|} \sum_{n \in \mathcal{B}_t} \nabla_\theta \ell(y_n, f(x_n; \theta_t)) \right) \\
=& \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t) + \eta_t e_t
\end{aligned}
$$

non-biased estimator:

$$
\mathbb{E}(e_t) = 0, \quad \mathrm{Var}(e_t) = \mathcal{O}(|\mathcal{B}_t|^{-1}) .
$$

Feedforward neural networks

$$f(x;\theta) = f_L(f_{L-1}(\cdots(f_1(x))\cdots)) \tag{3}$$

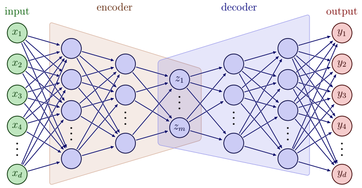
Recursive definition:

$$\begin{aligned} z_0 &= x \\ z_l &= f_l(z_{l-1}) = \varphi(b_l + W_l z_{l-1}), \quad l = 1, 2, \ldots, L-1\,, \\ z_L &= f_L(z_{l-1}) = b_L + W_L z_{L-1}\,. \end{aligned} \tag{4}$$

where

- $z_l, b_l \in \mathbb{R}^{d_l}$, $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$
- φ: activation function
- $\theta = (b_1, b_2, \ldots, b_L, W_1, \ldots, W_L)$

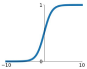
Feedforward neural networks



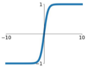
Feedforward neural networks

Activation Functions

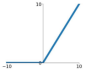
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



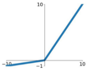
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

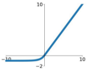


Leaky ReLU
$\max(0.1x, x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Implementation

PyTorch

- Load and store data
- define functions as deep neural networks
- Automatic differentiation by backpropgation
- Training algorithms, e.g. ADAM

Questions?