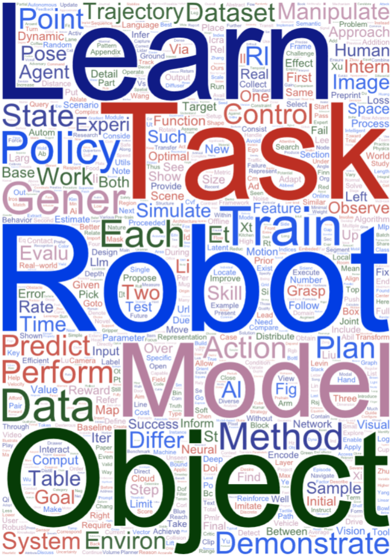


Robot Learning

Machine Learning Essentials

Marc Toussaint
Technical University of Berlin
Summer 2024

Machine Learning Essentials

- Supervised ML $f_\theta : x \mapsto y$

- Unsupervised ML $p_\theta(x)$ (and conditional $p_\theta(x|z)$)
  [Neglected here: Optimal embeddings, clustering]

Supervised ML

- Given data $D = \{(x_i, y_i)\}_{i=1}^n$ and a parameterized $f_\theta : x \mapsto y$, find θ

$$\min_\theta \underbrace{\sum_{i=1}^n \ell(y_i, f_\theta(x_i))}_{\text{(data) loss}} + \underbrace{R(\theta)}_{\text{regularization}}$$

Supervised ML

- Given data $D = \{(x_i, y_i)\}_{i=1}^n$ and a parameterized $f_\theta : x \mapsto y$, find θ

$$\min_\theta \underbrace{\sum_{i=1}^n \ell(y_i, f_\theta(x_i))}_{\text{(data) loss}} + \underbrace{R(\theta)}_{\text{regularization}}$$

done! That's (supervised) ML

Loss Functions

- Regularizations:
  - L_2 (Ridge): $R(\theta) = \|\theta\|_2^2$
  - L_1 (Lasso): $R(\theta) = \|\theta\|_1$

Loss Functions

- Regularizations:
  - L_2 (Ridge): $R(\theta) = \|\theta\|_2^2$
  - L_1 (Lasso): $R(\theta) = \|\theta\|_1$

- Regression $y \in \mathbb{R}^m$: Squared error: $\ell(y, \hat{y}) = (y - \hat{y})^2$

  [Robust variants: Huber loss, Forsyth]

Loss Functions

- Regularizations:
  - L_2 (Ridge): $R(\theta) = \|\theta\|_2^2$
  - L_1 (Lasso): $R(\theta) = \|\theta\|_1$

- Regression $y \in \mathbb{R}^m$: Squared error: $\ell(y, \hat{y}) = (y - \hat{y})^2$
  [Robust variants: Huber loss, Forsyth]

- Classification $y \in \{0, .., M\}$ (where $f : x \mapsto f(x) \in \mathbb{R}^M$ discriminative values)
  - Neg-Log-Likelihood: $\ell(y, f(x)) = -\log p(y|x)$ with $p(y|x) = \frac{e^{f_y(x)}}{\sum_{y'} e^{f_{y'}(x)}}$
  - Hinge: $\ell(y, f(x)) = \sum_{y' \neq y} [1 - (f_{y^*}(x) - f_{y'}(x))]_+$
  - Cross-Entropy: $\ell(y, f(x)) = -\sum_z h_y(z) \log p(z|x)$ *same as NLL for one-hot-encoding* $h_y(z) = [y = z]$

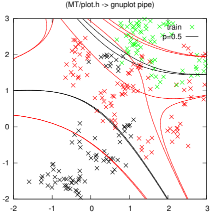
Parameterized Functions

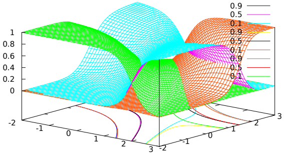
- Linear $f_\theta(x) = \theta_0 + \sum_{j=1}^d \theta_j x_j = \bar{x}^\top \theta$

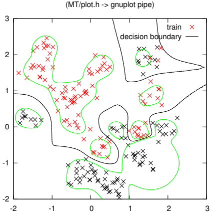
Parameterized Functions

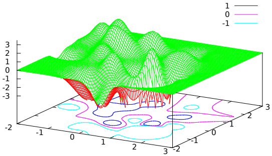
- Linear $f_\theta(x) = \theta_0 + \sum_{j=1}^d \theta_j x_j = \bar{x}^\top \theta$

- Linear in features: $f_\theta(x) = \phi(x)^\top \theta$ (or Hilbert space..)
  - Linear: $\phi(x) = (1, x_1, .., x_d) \ \in \mathbb{R}^{1+d}$
  - Quadratic: $\phi(x) = (1, x_1, .., x_d, x_1^2, x_1 x_2, x_1 x_3, .., x_d^2) \ \in \mathbb{R}^{1+d+\frac{d(d+1)}{2}}$
  - Cubic: $\phi(x) = (.., x_1^3, x_1^2 x_2, x_1^2 x_3, .., x_d^3) \ \in \mathbb{R}^{1+d+\frac{d(d+1)}{2}+\frac{d(d+1)(d+2)}{6}}$
  - Also: Radial-Basis Functions (RBF), piece-wise linear









Parameterized Functions

- Neural Nets: Repeating non-linear and linear parts: (this is a 3-layer NN):

$$f_\theta(x) = \underset{\uparrow \text{lin}}{W_3}\ \underset{\uparrow \text{nlin}}{\phi}\Big[\ \underset{\uparrow \text{lin}}{W_2}\ \underset{\uparrow \text{nlin}}{\phi}[\ \underset{\uparrow \text{lin}}{W_1}\ x + b_1\ ] + b_2\ \Big] + b_3$$

  - Non-linear parts:
    - rectified linear unit (ReLU): $\phi(x) = [x]_+ = \max\{0, x\}$
    - leaky ReLU: $\phi(x) = \max\{0.01x, x\}$
    - sigmoid, logistic: $\phi(x) = 1/(1 + e^{-x})$
    - max-pooling, soft-max, layer-norm
  - Linear parts:
    - Fully connected (W_i is a full matrix)
    - Convolutional
    - Transformer-like (cross-attentions)

- In essense
  - You define the parameterized function f_θ
  - You define the loss ℓ and regularization R
  - You provide the data set D

  - An optimizer (analytic for linear models, stochastic gradient otherwise) finds good parameters θ

- In essense
  - You define the parameterized function f_θ
  - You define the loss ℓ and regularization R
  - You provide the data set D

  - An optimizer (analytic for linear models, stochastic gradient otherwise) finds good parameters θ

- And you cross-validate to check your hyper-parameter choices

Unsupervised ML

- Given data $D = \{x_i\}_{i=1}^n$, learn "something" about $p(x)$

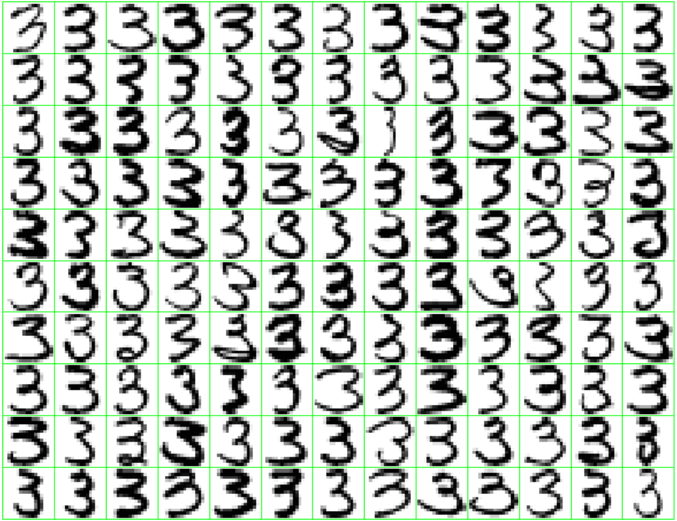
Unsupervised ML

- Given data $D = \{x_i\}_{i=1}^n$, learn "something" about $p(x)$
- Important setting: parameterized **autoencoder** $f_\theta : x \mapsto z \mapsto x'$, find θ

$$\min_\theta \underbrace{\sum_{i=1}^n \ell(x_i, f_\theta(x_i))}_{\text{autoencoding loss}} \quad + \quad \underbrace{R(\theta)}_{\text{regularization}}$$

  - You learn to reproduce x through a compact **latent code** $z \in \mathbb{R}^h$ (while $x \in \mathbb{R}^d$ is high-dimensional)
  - z has high entropy (typically Gaussian) distribution $\rightarrow$ you can **generate** $x' \sim p(x)$ by sampling z and decoding
  - If f is linear, this is called **Principle Component Analysis**
  - Better: Variational Autoencoder (VAC): Enforces $p(z)$ to have proper distribution.

Example: Digits



- There are other ideas in unsupervised learning, but the autoencoding objective is a major breakthrough
  - You “understand” the structure of data if you can compress and de-compress it
  - Autoencoders do this with powerful NN architectures

Diffusion Denoising Models

- Given data D, you want to learn a “system” that **generates** samples $x \sim p_\theta(x)$ where $p_\theta(x)$ models D

Diffusion Denoising Models

- Given data D, you want to learn a "system" that **generates** samples $x \sim p_\theta(x)$ where $p_\theta(x)$ models D

- Autoencoders are one approach, Diffusion Denoising Models another:
  - Train a stepwise stochastic process (Langevin dynamics) to generate samples $x \sim p_\theta(x)$
  - Has its origin in "energy-based models" and score matching
  - The step-wite sample generation process is very powerful

Conditional Generative Models

- Given data $D = \{(x_i, c_i)\}_{i=1}^n$ train a *conditional* distribution $p_\theta(x|c)$
  - We're actually back to Supervised ML $c \mapsto x$ (where c is the input)
  - But **if x is high-dimensional** (and c low-dim.), the generative model aspect is important:
  - The reconstruction objective enforces the system to find a good latent representation to generate high-dim. x
  - this is complemented by making conditional to c

$$f_\theta : \quad \begin{array}{c} x \mapsto z \mapsto x' \\ \uparrow \\ c \end{array}$$

A loss $\ell(x_i, f_\theta(x_i, c_i))$ jointly trains autoencoding $x \mapsto z \mapsto x'$ and conditional generation $c \mapsto z \mapsto x'$