PROBABILISTIC LEARNING

lunedì 1 maggio 2023 11:43

LIKELIHOOD FUNCTION: $p(D|x) = \prod_{n=1}^{N} p(c_n|x)$ ← probability of observing an event; ↳ random variable

it is a function of the parameter x and the observed data D:
it represents the probability of observing the data D given the value of the parameter x.

OPTIMIZATION PROBLEM: Find optimum value for x, given this data D.

Find each $x \in [0,1]$ that minimizes the (negative) log-likelihood function

$\min_{x\in[0,1]} -\log p(D|x)$

i) why? $\max f(x) = \min -f(x)$

ii) why log? $\log \prod = \sum \log$ and optimum is the same

minimizing the negative log-likelihood function is a common approach used to estimate the value of the parameter x that best fits the observed data D.

$\log p(D|x) = \log \prod_{n=1}^{N} p(c_n|x)$ Log-likelihood

$= \sum_{n=1}^{N} \log p(c_n|x)$ $\log \prod = \sum \log$

$= \sum_{n=1}^{N} \log x^{c_n}(1-x)^{1-c_n}$ Bernoulli distribution

$= \sum_{n=1}^{N} (c_n \log x + (1-c_n)\log(1-x))$ $\log a^b = b \log a$

↓ OPTIMUM $\log ab = \log a + \log b$

$\frac{\partial}{\partial x}\sum_{n=1}^{N} (c_n \log x + (1-c_n)\log(1-x)) = 0$ ↳ where gradient is zero! $* \frac{\partial}{\partial x}\log x = \frac{1}{x}$

- Determine
  - $p(y|x)$: your model
  - $p(D|x)$: find likelihood of dataset
- Check constraints
- Find best solution by minimizing $-\log p(D|x)$

SAMPLING METHODS

How to estimate π?

$A_O = \pi r^2$

$A_\square = (2r)^2$

$\frac{A_O}{A_\square} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4} \Rightarrow \pi = 4\frac{A_O}{A_\square}$

$\pi = 4\, \mathbb{E}_{(x,y)\sim Unif[-r,r]}\left[I[x^2+y^2 \le r^2]\right]$

$\approx 4\frac{1}{N}\sum_{n=1}^{N} I[x_n^2 + y_n^2 \le r^2]$ ↳ if point is in the circle

BASIC IDEA:

Generate random samples from a probability distribution that represents the problem of interest, and then use the k samples to estimate the solution to the problem.

how?

- simulating the behavior of a system over many random trials, then averaging the results to obtain an estimate of the expected value or other statistical properties of the system

MONTE CARLO METHODS: The underlying idea is to APPROXIMATE BY SAMPLING

$\frac{1}{N}\sum_n f(x_n) \xrightarrow{n\to\infty} \int f(x)p(x)dx$

- Analytically infeasible quantities
- optimization

Class of computational algorithms that use random sampling to obtain numerical solutions to problems in various fields.
Named after the Monte Carlo Casino known for its games of chance.

Problem with sampling: curse of dimensionality

MARKOV CHAIN MONTE CARLO APPROACH: → make new sample dependent on the past (Markov Chain)

Class of algorithms that use random sampling to generate a sequence of samples from a target distribution, which can be used to estimate the parameters of a Bayesian model or perform inference on complex systems.

PROPOSAL DISTRIBUTION $q(x_t|x_{t-1})$ to obtain a chain that corresponds to a sample from the original distribution

Conditions:
A) IRREDUCIBILITY: There is a positive probability of visiting all states
B) APERIODICITY: the chain should not get trapped in cycles

METROPOLIS-HASTINGS ALGORITHM

1. Initialize $x_t := x_0$.
2. For $t \in \{0,1,\dots,T-1\}$:
   (i) (**Generate**) Sample $x' \sim q(x|x_t)$ ← proposal distribution
   (ii) (**Evaluate**) Calculate acceptance probability:
   $$A(x', x_t) = \min\left\{1, \frac{p(x')\,q(x_t|x')}{p(x_t)\,q(x'|x_t)}\right\}$$ ← eliminates bias towards one direction
   (iii) (**Select**) Sample $u \sim \text{Unif}[0,1]$.
   If $A(x', x_t) > u$, then $x_{t+1} := x'$.
   Else $x_{t+1} := x_t$.

↳ pick a random variable from uniform distribution and see if greater than p(x') or not, and based on that, accept

$p(x) \propto \exp(-f(x))$

SIMULATED ANNEALING

1. Initialize $x_t := x_0$ and $T_t := T_0$.
2. For $t \in \{0,1,\dots,T-1\}$:
   (i) (**Generate**) Sample $x' \sim q(x|x_t)$.
   (ii) (**Evaluate**) Calculate acceptance probability:
   $$A(x', x_t) = \min\left\{1, \frac{p^{\frac{1}{T}}(x')\,q(x_t|x')}{p^{\frac{1}{T}}(x_t)\,q(x'|x_t)}\right\}$$
   (iii) (**Select**) Sample $u \sim \text{Unif}[0,1]$.
   If $A(x', x_t) > u$, then $x_{t+1} := x'$.
   Else $x_{t+1} := x_t$.
   (iv) Set T_{t+1} according to chosen cooling schedule.

IF $p(x) \propto \exp(-f(x))$ THEN $p^{\frac{1}{T}}(x) \propto \exp(-\frac{1}{T})$

It is a probability distribution that specify how to generate new candidate states for the Markov Chain — common choices: Gaussian distribution / uniform distribution

it describes how to move from the current state of the Markov chain to a new state

★ it determines the set of potential new states that the algorithm can explore
★ impact on efficiency of exploring the space of possible states

should be chosen in a way so that it generates new states that have a high probability of being accepted by the acceptance criterion of the MCMC algorithm.

IF proposed state has higher prob. vs current state, THEN it is always accepted.

IF it has a lower prob., it is accepted w/ a probability proportional to acceptance ratio.

ACCEPTANCE CRITERION: Rule that determines whether or not to accept a proposed new state in Markov Chain
= based on the ratio of the probability of the proposed state and the current state, known as acceptance ratio

TRADEOFF:
- exploring the space of possible states efficiently
- minimizing time spent in low-probability regions of the state space.

METROPOLIS-HASTINGS ALGORITHM = Generates a new parameter value from a proposal distribution, evaluates how well it fits the data, and accepts it or rejects it based on an acceptance criterion that balances the fit to the data with the probability of proposing the new value.

SIMULATED ANNEALING = Variant of MH algorithm that adds a temperature parameter that controls the probability of accepting a new parameter value.

TEMPERATURE PARAMETER:
- ↑ temperatures ↔ wide range of parameter values explored
- ↓ temperatures ↔ converge to most likely parameter value

controls the probability of accepting a new state that has a worse objective function value than the current state

Advantages:
- escape local minima
- explore new regions for the search space

COOLING SCHEDULE: function that determines the rate at which the temperature parameter is decreased over time during the search.
= determines the balance between exploration / exploitation

IF temperature ↓ too quickly = algorithm may converge prematurely to a local optimum
temperature ↓ too slowly = algorithm may spend too much time exploring unpromising regions of the search space.

optimum:
Should decrease the temperature slowly enough to allow for effective exploration of the search space, but quickly enough to allow for the algorithm to converge to a good solution in a reasonable amount of time.