Maschine Learning Summary WS21/22

mastermakrela

January 29, 2022

Contents

kek

Introduction

Machine Learning Principles, methods, and algorithms for learning and prediction on the basis of past evidence

Goal of Machine Learning:

Machines that *learn* to *perform* a *task* from *experience*.

Learning most important aspect
We provide *data* and *goal* - machine figures rest out.

Learning Tools:

- statistics
- probability theory
- decision theory
- information theory
- optimization theory

Core Questions

Task: $y = f(x; w)$

Where:

- x Input
- y Output
- w Learned parameters

Regression	Classification
Continuous output	Discrete output

Core Problems

1. How to input data / how to interpret inputted data
2. Features
    - Invariance to irrelevant input variations
    - Selecting the "right" features is crucial
    - Encoding and use of "domain knowledge"
    - Higher-dimensional features are more discriminative.
3. Curse of Dimensionality
    - complexity increases exponentially with number of dimensions

Core Questions

1. Measuring performance of a model
    - eg. % of correct classifications
2. Generalization performance
    - performing on test data is not enough
    - model has to perform on new data
3. What data is available?
    - Supervised vs unsupervised learning

- mix: semi-supervised
- reinforcement learning - with feedback

Most often learning is an optimization problem

I.e., maximize $y = f(x; w)$ with regard to performance.

Bayes Decision Theory

Probability Theory

Probability

$$P(X = x) \in [0, 1] \tag{1}$$

Where $X \in \{x_1, ..., x_N\}$ an Occurence of something.

Assuming two random variables $X \in \{x_i\}$ and $Y \in \{y_i\}$. Consider N trials and let:

$n_{ij} = count\{X = X_i \wedge Y = yj\}$

$c_i = count\{X = x_i\}$

$r_i = count\{Y = y_j\}$

Then we can derive *The Rules of Probability*:

Joint Probability	$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$
Marginal Probability	$p(X = x_i) = \frac{c_i}{N}$
Conditional Probability	$p(Y = y_j \| X = x * i) = \frac{n*ij}{c_i}$
Sum Rule	$p(X) = \sum_Y p(X, Y)$
Product Rule	$p(X, Y) = p(X, Y)P(X)$

And *Bayes' Theorem*:

$$p(X|Y) = \frac{p(X|Y)P(X)}{P(X)} \tag{2}$$

where: $p(X) = \sum_Y p(X, Y)p(Y)$

Probability Densities

If the variable is continuous we can't just look at probability for x. We have to look for the interval the x is in, using *Probability Density Function* $p(x)$.

$p(x) = \int_a^b p(x)dx$

The probability that x lies in the interval is given by $(-\infty, z)$ the *cumulative distribution function*:

$P(z) = \int_{-\infty}^z p(x)dx$

Expectations

Expectation average value of some function $f(x)$ under a probability distribution $p(x)$

Discrete: $\mathbb{E}[f] = \sum_x p(x)f(x)$

Continuous: $\mathbb{E}[f] = \int p(x)f(x)dx$

For finite N samples we can approximate:

$\mathbb{E}[f] \backsimeq \frac{1}{N} \sum_{n=1}^{N} f(x_n)$

Conditional expectation:

$\mathbb{E}_x[f|y] = \sum_x p(x|y)f(x)$

Variances and Covariances

Variance measures how much variability there is in $f(x)$ around its mean value $\mathbb{E}[f(x)]$

$var[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$

Covariance for two random variables x and y

$cov[x, y] = \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]$

Bayes Decision Theory

Priors a priori probabilities

what can we tell about probability before seeing the data

sum of all priors is 1

Conditional probabilities $p(x|C_k)$ is **likelihood** for class C_k

where x measures/describes certain properties of input

Posterior probabilities $p(C_k|x)$

probability of class C_k given the measurement vector x

$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{p(x|C_k)p(C_k)}{\sum_i p(x|C_i)p(C_i)}$

Goal: Minimize the probability of a misclassification.

Decide for C_k when:

$p(C_k|x) > p(C_j|x) \forall j \neq k$

$p(x|C_k)p(C_k) > p(x|C_j)p(C_j) \forall j \neq k$

Classifying with Loss Functions

Motivation: Decide if it's better to choose wrong or nothing.

In general formalized as a matrix L_{jk}

$L_{jk} = loss for decision C_j if C_k is correct selection$

Minimizing the Expected Loss

Optimal solution requires knowing which class is correct - *this is unknown.* So we **minimize the expected loss**:

$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(x, C_k) dx$

This can be done by choosing the $\mathcal{R}_j$ regions such that:

$\mathbb{E}[L] = \sum_k L_{kj} p(C_k|x)$

Probability Density Estimation

How can we estimate (= learn) those probability densities?

In Supervised training case: data and class labels are known. So we can estimate the probability density for each class separately.

The Gaussian (or Normal) Distribution

One-dimensional

- Mean μ
- Variance σ^2

$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$

Multi-dimensional

- Mean μ
- Variance Σ

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\} \tag{3}$$

Properties

Central Limit Theorem "The distribution of the sum of N i.i.d. random variables becomes increasingly Gaussian as N grows."

`[There was some more stuff in slides but it was boring math]`

The marginals of a Gaussian are again Gaussians

Parametric Methods

Given some data X with parameters $\theta = (\mu, \sigma)$. What is the probability that X has been generated from a probability density with parameters θ.

$L(\theta) = p(X|\theta)$

Maximum Likelihood Approach

Single data point:

$p(x_n|\theta) = \frac{1}{\sqrt{2\pi}\sigma} exp\{-\frac{(x_n-\mu)^2}{2\sigma^2}\}$

Assumption all data points are independent:

$L($
$theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$

Log-Likelihood: $E(\theta) = -lnL(\theta) = -\sum_{n=1}^{N} lnp(x_n|\theta)$

Goal: Minimize $E(\theta)$.

How to:

1. Take the derivate of $E(\theta)$
2. Set it to zero
3. Quic Mafs

Warning

MLA is *biased* - it underestimates the true variance. I.e., *overfits to the observed data.*

Frequentist vs Bayesian

Frequentist	Bayesian
probabilities are frequencies of random, repeatable events	quantify the uncertainty about certain states or events

	Frequentist	Bayesian
	fixed, but can be estimated more precisely when more data is available	uncertainty can be revised in the light of new evidence

Non-Parametric Methods

Often the functional form of the distribution is unknown. So we have to estimate probability from data.

Histograms

Partition data into bins with widths Δ_i and count number of observations in each bin.

$p_i = \frac{n_i}{N\Delta_i}$

Often: $\Delta_i = \Delta_j \forall i, j$

Advantages	Disadvantages
works in any dimension D	curse of dimensionality
no need to store data after computation	rather brute force

Bin size:

- too large: too much smoothing
- too small: too much noise

Kernel Density Estimation

Parzen Window

Idea: Hypercube of dimension D with width edge length h.

We place a kernel window k at location x and count how many data points fall inside it. Crude solution because the chosen function k creates hard cuts around Hypercubes.

Kernel Function: $k(u) = \begin{cases} 1, |u_i * \frac{1}{2}, i = 1, ..., D \\ 0, else \end{cases}$

$K = \sum_{i=1}^{N} k(\frac{x - x_n}{h})$

$V = \int k(u)du = h^d$

Then probability density is:

$p(x) \simeq \frac{K}{NV} = \frac{1}{Nh_D} \sum_{i=1}^{N} k(\frac{x - x_n}{h})$

Gausian Kernel

Similar to Parzen Window, but with Gaussian kernel function k.

$k(u) = \frac{1}{(2 \times \pi h^2)^{1/2}} \exp\{-\frac{u^2}{2h^2}\}$

$K = \sum_{i=1}^{N} k(x - x_n)$

$V = \int k(u)du = 1$

Then probability density is:

$p(x) \simeq \frac{K}{NV} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{(2\pi h^2)^{D/2}} \exp\{-\frac{\|x - x_n\|^2}{2h^2}\}$

K-Nearest Neighbors

Similar to above but we fix K (the number of neighbors) and calculate V (size of the neighbourhood).

Then: $p(x) \simeq \frac{K}{NV}$

Warning: Strictly speaking, the model produced by K-NN is not a true density model, because the integral over all space diverges.

Bayesian Classification

$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$

Summary

- Very General
- Training requires no computation
- Requires storing and computing the entire dataset -> cost linear in the number of data points -> can be saved in implementation

Kernel size K in K-NN?

- Too large: too much smoothing
- Too small: too much noise