Traditional Programing Data, Program → ouput

Machine learning Data, output → Program

Supervised learning → training data labeled (train on output)

Unsupervised learning → train data unlabeled need to find hidden structure

semisupervised learning → only part of input is labeled

reinforcement learning → Learner interacts to find optimal behavior

Deep learning → Automatic learning without human interference

Regression → Predict a number

Classification → Predict categories

Unsupervised learning Application

- Clustering (group data)
- Dimensionality Reduction (compress data)
- Anomaly detection (find outliers)

when to split? max purity

limit depth of trees to → doesn't get to big/widely, less prone to overfit

stop splitting when node 100% one class, when purity improvement below threshold, when number of examples below threshold

Entropy (impurity) when =1 most impure, =0 most pure, p_1 = fraction of ex. are cats?

$H(p_1)$ = entropy, $p_0 = (1-p_1)$ fraction not in ex. not cats

Entropy with multiple → $H(p_1) = -\sum_i p_i \cdot \log_2(p_i)$

$H(p_1) = -p_1 \log_2(p_1) - p_0 \log_2(p_0) = -p_1 \log_2(p_1) - (1-p_1)\log_2(1-p_1)$

Information gain $H(p_1) - (p_0 H(p_1) + p_0(H_i))$

total - Nodal added

One hot encoding for k categories can have k values, 0 or 1

use in Reg tree for Male

ID3 Algorithm → start root calc entropy and IG for all start highest IG

Regression Tree → $H_{parent} - (w_i^l H(p_i^l) + w_i^R H(p_i^R))$

Variance $s^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$

Decision tree pros → easy understand, little to no data, problematic for convex

bias, more flexible | Decision trees cons → prone to overfit, high variance, costly

DT sensitive to small data changes, building multiple trees solves this issue

Sampling Replacement → take random but replace before each grab

IG is H(root) - the calc below

Bagged DT → given dataset of 21 to B → B between 64 → 128

when splitting entropy do by feature and weight is num cat in total on

$p_1^L = 3/10\ H(3/7)$ $p_1^R = 7/10 H(5)$

7 left root 3 cat

Random Forest Algo → at subset node pick random subset of k<n if n features available, the algo picks max IG typically k = √n

Sampling: Replacement, # Iteration, Items#

Boosted Trees

Same as bagged DT but must learn to pick examples previously failed

XG Boost → open source of boosted trees, fast efficient, built in step to overfitting

DT + Tree Ensembles → works on structured data, fast, not for unstruct data

works well with multiple models, works slower than DT

when splitting on leaf in regression tree use the mean as the root node

s^2 = sum variance of parent <= s^2 sum over child this = good

Bayes Rule $p(y|x) = \frac{P(x|y)\,p(y)}{P(x)}$

email representation given bag of words 1s if word show assigned

Laplace Smoothing $p(f_2 = b \mid y=1) = \frac{1 + (\#\text{ examples } f_2 = b \text{ \& } y = t)}{\#\text{possible value } f_2 + (\text{examples } y = t)}$

$p(y=1) = \frac{\#\text{ examples } y^{(i)} = 1}{m}$

k fold cross validation split into random subsets, validation set to test better train

$p(y=1|x) = \frac{P(x|y=1)\,p(y=1)}{P(x)} = \frac{p(x|y=1)\,p(y=1)}{p(x|y=1)\,p(y=1) + p(x|y=0)p(y=0)}$

$p(A|B) = \frac{P(A \cap B)}{P(B)}$ $P(B) = P(B|A)\,P(A) + P(B|\bar{A})\,P(\bar{A})$ value is greater the

for classification get $P(x_i|+)$ and multiply and negate the one

$F(x_j|y=0) = \frac{1}{\sqrt{2\pi}\,\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$ μ = mean σ = stddev = $\frac{1\Sigma(x_i - \mu)}{n}$

estimate for continuous

parameter → estimated by data, define shit, needed for pred

Hyperparameter → external can't be estimated → process tuned, specified by practitioner

Add out cross validation → train mi on S_{train} test on S_{val}

Report error on test only

k fold train on all S except S_i

Law of total probability

$S = B_1 \cup B_2 \cup \dots \cup B_n$

$B_i \cap B_j = \emptyset$ for ever i≠j

$P(B_i) > 0$ for $i = 1 \dots n$

we can say $B_1 \dots B_n$ partion S then

$P(A) = P(A|B_1)P(B_1) + \dots P(A|B_n)P(B_n)$