Machine Learning

Covered topic — KNN not work for simbolic or non-numeric test

- □ Identification trees
  - Arranging tests
  - Measures of disorder
  - Numeric tests ← some test are expensive
  - Rules
  - Transformations

Note: a good 2D tree has lower depth and simple structure

— Disorder Set (P: positive, T: total, N: negative)

$$D(set) = -\frac{P}{T}\log_2\frac{P}{T} - \frac{N}{T}\log_2\left(\frac{N}{T}\right)$$

If we have a unbiased data set:

$\frac{P}{T} = \frac{1}{2}$: $D(set) = -\left(\frac{1}{2}\log\left(\frac{1}{2}\right)\right) \times 2$ e.g. ▷ △ □ □

If we have a fully total data:

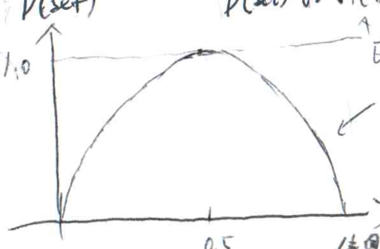
$\frac{P}{T} = 1$: $D(set) = \underbrace{-1\log_2(1)}_{0} - \underbrace{0\log(0)}_{0} = 0$, e.g. △△ △△

Disorder test:

$$D(test) = \sum_{Branches} Dis(set) \times \frac{set\ size}{Total}$$

— Occam's razor: the simplest explanation is often the best explanation

$D(set)$ or $H(X) = -\left[\sum_{x \in X} P(x)\ln(P(x))\right]$ ← Entropy in information theory



Note: P 越大，说明某事件经常发生，因此对应的信息量越少。

1111 N种可能 = 2 信息量 = $2^{4\ bit}$

$\Rightarrow$ 信息量 $= \log_2(N)$ ∵ $P = \frac{1}{N}$

$\Rightarrow = \log_2\left(\frac{1}{P}\right)$ ∴ $N = \frac{1}{P}$

— Decision Tree Example

Data: Vampire Data in Romanian

Vampire	有无影子 shadow	Garlic	肤色 Complexion	Accent
No	?	Y	Pale	None
No	Yes	Y	Ruddy	None
Yes	?	N	Ruddy	None
Yes	N	N	Average	Heavy
Yes	?	N	Average	Odd
No	Y	N	Pale	Heavy
No	Y	N	Average	Heavy
No	?	Y	Ruddy	Odd

Goal: Use some test result of characteristic to determine whether it's an Vampire or not.

Step1: Compute the disorder of each feature test. (Actual, try to use your intuition at first).

Shadow ④ (# of homogeneous samples) (0.5 — Disorder score)

Yes 3/8: − − − | No 1/8: + | ? 1/2: + + − −

D(set): 0, 0, 1

$D(test) = \frac{3}{8}(0) + \frac{1}{8}(0) + \frac{1}{2}(1) = \frac{1}{2} = 0.5$

Garlic ③ (0.5625)

Y 3/8: − − − | N 5/8: + + + − −

0, ~0.9

$D(test) = 3/8(0) + 5/8(0.9) = 0.6$

Completion ② (0.)

Pale 1/4: − − | Ruddy 3/8: − − + | Avg 3/8: − + +

0, ~0.9, ~0.9

Accent ⓪ (0.925)

Heavy 3/8: + − − | Odd 2/8: + − | None 3/8: − − +

0.9, 1, 0.9

$D(Pale) = -[0\log(0) + 1\log(1)] = 0$

$D(Ruddy) = -\left[\frac{1}{3}\log\left(\frac{1}{3}\right) + \frac{2}{3}\log\left(\frac{2}{3}\right)\right] \approx 0.9$

$D(Avg) = -\left[\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right] \approx 0.9$

$D(test) = 1/4(0) + 3/8(0.9) + 3/8(0.9) = 0.675$

Step2: eliminating the root feature and built the next level for Decision tree:

~~Shadow~~

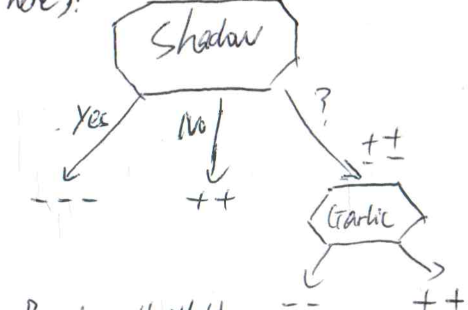
Garlic ④ (0): Y 1/2: − − | N 1/2: + +; D(set): 0, 0

Completion ② (0.5): Pale 1/4: − | Ruddy 1/2: + − | Avg 1/4: +; D(set): 0, 1, 0

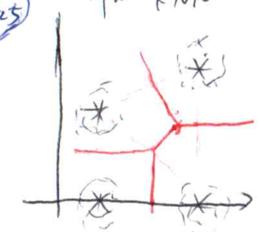
Accent ⓪ (1.0): ~~Heavy~~ | Odd 1/2: + − | None 1/2: + −; D(set): 1, 1

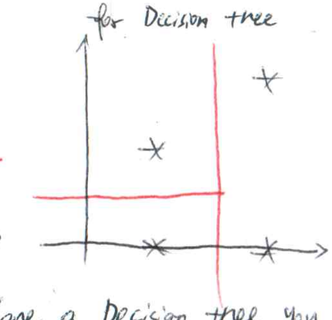
Note: You can see the cost is reduced, because each step we only take the test that seems most useful.

Step3: Construct the complete disorder-minimizing Identification/Decision tree (Note: In practice, the tree can be deeper than 2 level, so you might need to take more step to reach here):



— Boundary threshold for KNN / for Decision tree

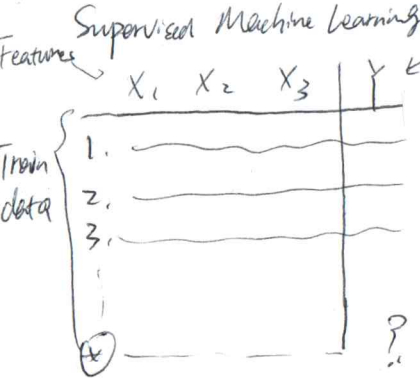




Note: After you have a Decision tree, you can use it to develop some rules/policies with if-else clauses.

Machine Learning

Supervised Machine Learning



Classification
- Boolean: Y/N
- Discrete: e.g. color, party, ...
- Continuous: Height, Weight

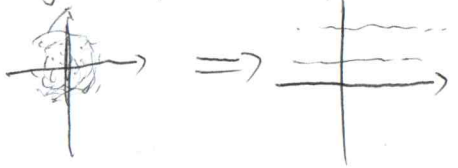
Supervised ML objective:

Given traing data with known features (x_i) and known classification (Y), and produce some rule for classifing new points with know features and unknow classification.

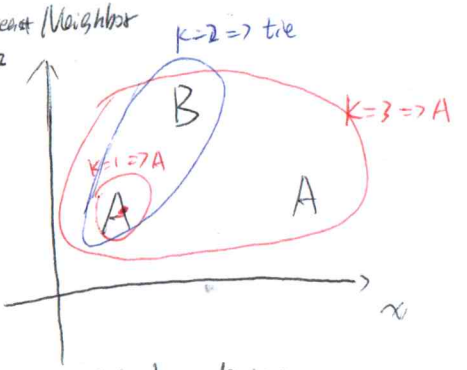
You can also working backword to figure out the traing data:

1NN: X
3NN: Y
5NN: X

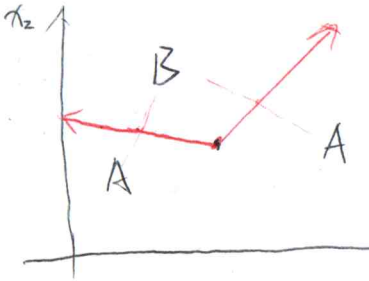
5) Transform to polar coord

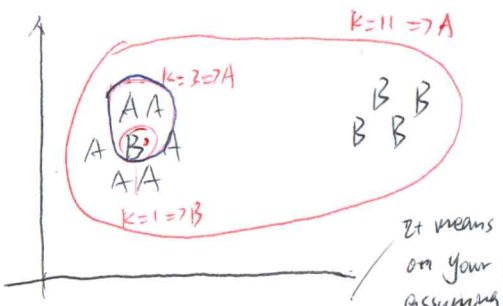


K-Nearest Neighbor



1NN boundaries





It means you overly relying on your data. so you're assuming even small fluctuations in the data are significant.

k=1 Overfitting

k=3

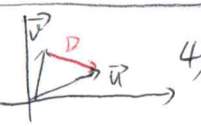
k=11 Underfitting → Smoothing over too much, so you're not relying enough on the the data. As a result You're ignoring the differences in the data should be real trend.

How to find a good value of K? ⇒ CrossValidation.

Idea: set aside part of traing data, then training using the remaining data, then we'll test whether the data we set aside gets classified correctly. e.g Leave-one out CV, k-CV.

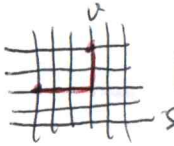
Other thing we have control over for KNN is Distance Metric:

1.) Euclidean distance: $D(\vec{u},\vec{v}) = \sqrt{\sum (u_i - v_i)^2}$



(Note: this the default method, which simply find the stright line distance between two point, aka geometric distance)

2) Manhattan distance: $D(\vec{u},\vec{v}) = \sum_i |u_i - v_i|$

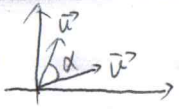


(Note; aka texicab distance, used in case like classifying people based on what grocery store are they lived closest to, for advertising purpose, so people can't go across the block but have to go along the streets.)

3) Hamming distance: $D(\vec{u},\vec{v}) = \sum_i \text{diff}?(u_i, v_i)$

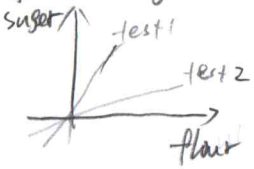
(Note: It's useful for comparing symbolic things. It's just looking at whether the vector components are same or not. e.g. like comparing some children's toys: $\vec{u} = (\triangle, \text{red}, \text{soft})$, $\vec{v} = (\triangle, \text{blue}, \text{soft})$ $\Rightarrow 0+1+0=1$

4) Cosine distance:



$$D(\vec{u},\vec{v}) = 1 - \cos(\text{angle}) = 1 - \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\cdot\|\vec{v}\|}$$

(Note: cosine is a similarity metric, so smaller angle is actually a bigger number, so we use the negation, so that's effective larger angle means large dist. It's useful if you care about the ratio between the features than actual quantities. Eg Recipe. flour : suger = 1:1.5



Or if you want to cluster starts into galaxy on which sky they were in