Decision trees continued

picking up from last time:

For 3 out of 4 leaves any instance that fall in them can be automatically labeled without need to do NN → all 3 of them are pure

Advantage → for those 3 leaves you don't need to store the $\vec{x}$

coordinate for any of the instances. Since you skip doing NN for any instance that falls

that tak further. for S_{NE}, $Pr(\text{"x"}) = \frac{5}{8}$ and $Pr(\text{"o"}) = \frac{3}{8}$

So, in this case we may say that a new instances in that leaf is labeled "x" with $Pr = \frac{5}{8}$

OR we could continue to split S_{NE} until we reach pure leaves

Question: is that always possible?

Answer: in the general case of noise (two instances with some $\vec{x}$ have differently)

NO.

Adding Noise to Resolve Label conflict

Counterintritively, we add a negligible amount of

zero-mean Gaussian noise $\sigma^2 = b\vec{q}$ to all $\vec{x}$.

→ as assumed as conflict resolves we can split until

without noise, you can split data until you reach pure leaves

↳ you must have to split until you have one instance per leaf → BAD because one instance per leaf means you are overfitting the training data (high variance)

1. if tree is too deep (many layers and leaves) then we have high variance (overfitting, high test error)

2. if tree is too shallow (low depth), then we have high bias (underfitting, high error on both training set and test set)

New Goal → Build a tree that is maximally compact and only has pure leaves.

This is an NP-hard problem.

(No solution can be found in polynomial-time algorithms)

Fortunately, we can find suitable solution using a greedy algorithm

greedy algorithm = any algorithm that chooses to make local optimum choice at each stage, without considering the global optimum.

We keep spliting the data to minimize an impurity function that measures purity among subsets

How do we decide borders for splits?

We test which borders lead to greater purity using impurity functions

- Entropy
- Gini impurity index

Impurity Functions

We want to measure ambiguity of labels within a leaf.

Ex) A leaf has 100 "x" and 1 "o" → low ambiguity, so label a new instance "x"

Ex) A leaf has 500 "x" and 500 "o" → high ambiguity, so unsure how to label a new instance

Given a data set: $S = \{(\vec{x}_1, y_1), \cdots (\vec{x}_n, y_n)\}$

$y_i \in \{1, 2, \cdots, C\}$, C = number of classes (distance labels)

Let $S_k \subseteq S$ where $S_k = \{(x,y) \in S$ such that $y = k\}$

(subset)

$\Rightarrow S$ is  titioned $\Rightarrow S = S_1 \cup S_2 \cdots \cup S_c$

(union)

$S_k \cap S_j = \emptyset \quad \forall k \neq j$

(intersection; empty set; for all)

Def Gini impurity function of set S

$= G(S)$

$= \sum_{k=1}^{c} p_k(1 - p_k)$

$p_k = \frac{|S_k|}{|S|} = \frac{\text{number of instances with label } k}{\text{number of instances in } S}$

Ex Case of $c = 2$ labels

$p_r(\text{"1"}) = p_1$

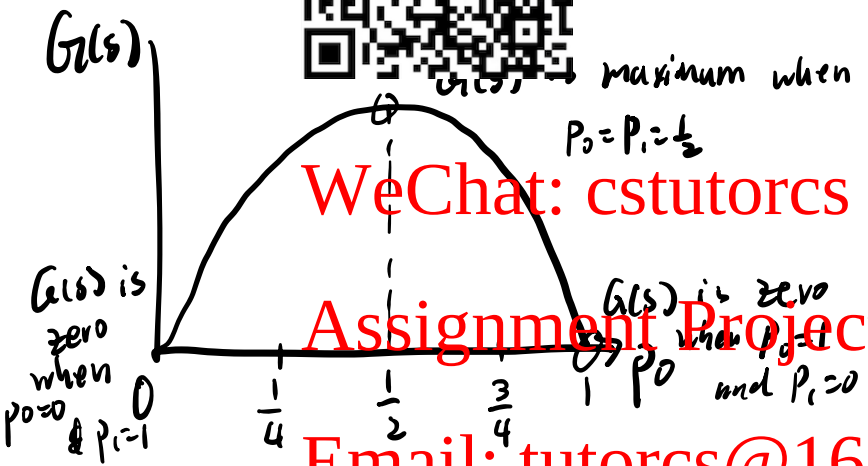
$p_r(\text{"0"}) = p_0 = 1 - p_1$

$\Rightarrow G(s) = p_0(1-p_0) + p_1(1-p_1)$

$= (1-p_0)\,p_0$

$= p_1(1-p_1)$

$= 2(1-p_0)\,p_0$



We want G(leads) to be low (at best, zero)

At worst G(leads) = 0.5 (large) when $p_0 = p_1 = \frac{1}{2}$

$\Rightarrow$ this tell us nothing about a new instance falling in that leaf #

In general, the worst case scenario is when the label are uniformly distributed

$$\Pr(\text{label } k) = \frac{1}{C} \quad \forall k = 1, 2, \cdots C$$

$$= q_k$$

{ q_k } will minimize the Gini impurity

fancy

Ideally, we $\sum_{k=1} p_{j \neq k} = 0$

A our goal is that to decide about a split, we select $\min_{\{S,P\}} G(s)$

at least new leaf we want to have a dominant label

$p_k >> p_j \; j \neq k$

To measure our success, we use a tool called the kullback-Leibler (KL) divergence

Def) The kullback-Leibler (KL) divergence between two distributions $\{p_k\}$ and $\{q_k\}$ is $KL(p||q) = \sum_{k=1}^{c} p_k \log\left(\frac{p_k}{q_k}\right)$

this can be $\log_a(\cdot)$, $\log_2(\cdot)$, $\ln(\cdot)$ need $\lim_{a \to 0^+} a \log a = 0$

Ex: $KL(p\|q) = \sum_{k=1}^{c} p_k \log\left(\frac{p_k}{q_k}\right)$ etc

$\log(1) = 0$

#

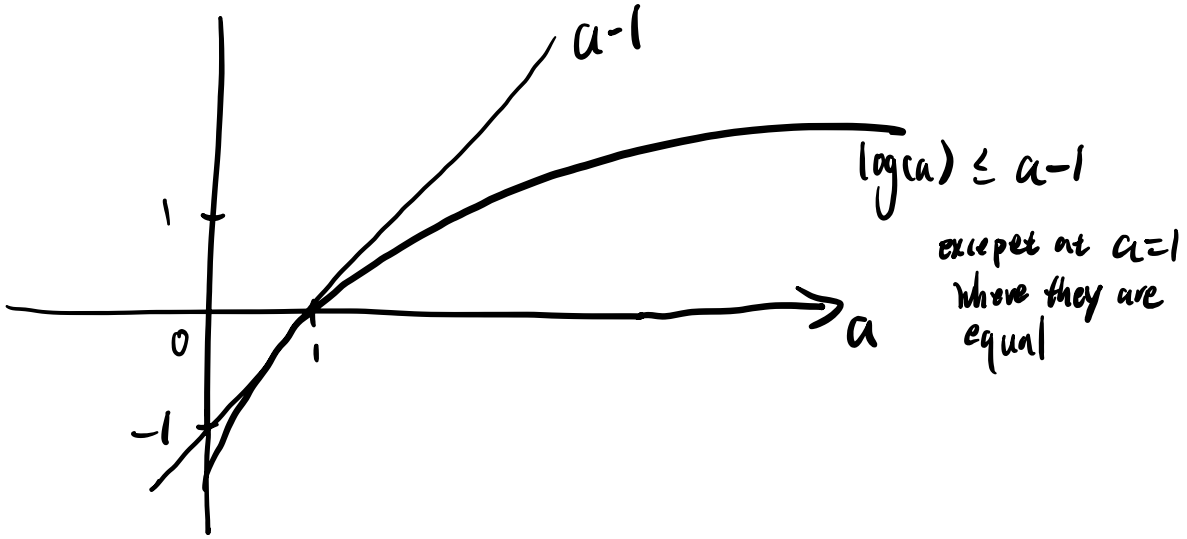
property of KL:

- KL is NOT a distance measure, since

$KL(p\|q) \neq KL(q\|p)$

- $KL(p\|q) \geq 0$ with equality $\Leftrightarrow p = q$

proof: we show $-KL \leq 0$



$$-KL(p\|q) = -\sum_{k=1}^{C} p_k \log\left(\frac{p_k}{q_k}\right) = \sum_{k=1}^{C} p_k \log\left(\frac{q_k}{p_k}\right)$$

$$\leq \sum_{k=1}^{C} p_k \left(\frac{q_k}{p_k} - 1\right) = \sum_{k=1}^{C} (q_k - p_k)$$

$$= \underbrace{\sum_{k=1}^{C} q_k}_{=1} - \underbrace{\sum_{k=1}^{C} p_k}_{=1} = 1 - 1 = 0$$

$$\Rightarrow -KL(p\|q) \leq 0$$

with equality if $\frac{p_k}{q_k} = 1 \ \forall k$

$$\Rightarrow KL(p\|q) \geq 0$$

equality if both distributions are identical

$p_k \neq q_k \Rightarrow KL(p\|q) > 0$

$p_k = q_k \Rightarrow KL(p\|q) = 0$

For a given leaf, we measure the quality of purity as the KL divergence from the worst case,

q_k (uniform)

$KL(p \| q) = \sum_{k=1}^{c} p_k \log\left(\frac{p_k}{q_k}\right) = \sum_{k=1}^{c} p_k \log\left(\frac{p_k}{\frac{1}{c}}\right)$

want for from q

worst $q_k = \frac{1}{c}\ \forall k$

$= \sum_{j=1}^{c} p_k \log(c) = \sum_{k=1}^{c} p_k \log(c) = \log(c) \underbrace{\sum_{j=1}^{c} p_j}_{=1}$

$= \sum_{k=1}^{c} p_k (-\log(p_k)) + \log(c)$

this is a constant independent of our choice of split

In deciding about a split in a leaf, our goal is to maximize $KL(p \| q)$ which is the same as

q uniform

minimizing $-\sum_{k=1}^{c} p_k \log(p_k)$

Def: the ENTROPY of a set with probability distribution $p = \{p_1, p_2, \ldots p_c\}$

is designed as

$$H(S) = -\sum_{k=1}^{C} p_k \log(p_k)$$

☆ our goal for every split the leaves have min $H(S)$ $\{S_1, S_2\}$

Impurity applied to a tree

Gini Impurity of a tree ☆	Entropy of a tree ☆
we seek the split that results in minimum G(tree)	= H(tree)
step 1: choose a split. It partitions S. For this, you'll need to isolate feature (dimension) x_i and sove all inseances along dimension i.	step 1: same
step 2: For each of the two subsets, estimate Gini impurity index $P_{w,x} = \frac{\lvert S_{w,x} \rvert}{\lvert S \rvert}$, $P_{w,0} = \frac{\lvert S_{w,0} \rvert}{\lvert S \rvert} \Rightarrow G(S_w)$	step 2: same, except we p's to get entropy $H(S_w)$

and

$P_{E,x} = \frac{|S_{E,x}|}{|S_E|}$ $P_{E,o} = \frac{|S_{E,o}|}{|S_E|}$

Step3: G(tree) = Weighted average of Gini index

Step3:

$H(tree) = \frac{|S_w|}{|S|} H(S_w) + \frac{|S_E|}{|S|} H(S_E)$

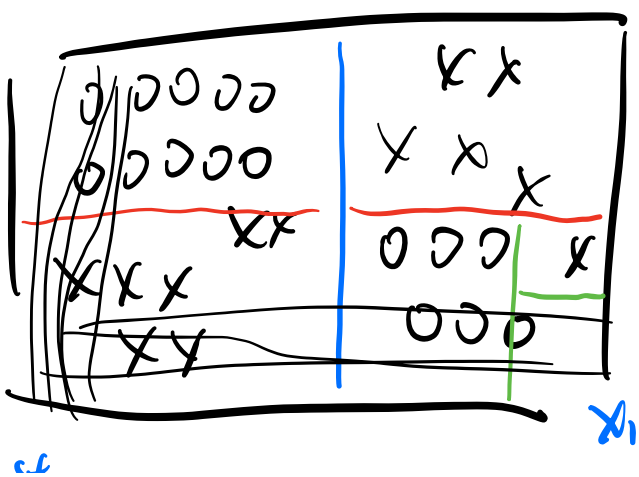
($|S_w|$: number of instances in S_w; $|S|$: number of instances in original before split)

step 4: try all features (dimensions) and all possible N-1 splits

Pick the split that minimizes impurity

step4: Save

Ex 5



1st split along x_1

2nd split yields better entropy purity

3rd yields purity in all leaves

keep splitting until entropy (or Gini) is zero

try every single coordinate split, all $N-1$ of them in V_1 } in all, you will test $d(N-1)$ coordinate splits

then go to V_2 and try all $N-1$ coordinate splits

For each coordinate:

calculate $p_{N,0} = \frac{3}{4}$, $p_{N,1} = \frac{1}{4}$ → $H(S_N)$

$p_{S,0} = \frac{2}{4}$, $p_{S,1} = \frac{2}{4}$ → $H(S_S)$

→ $H(\text{tree})$

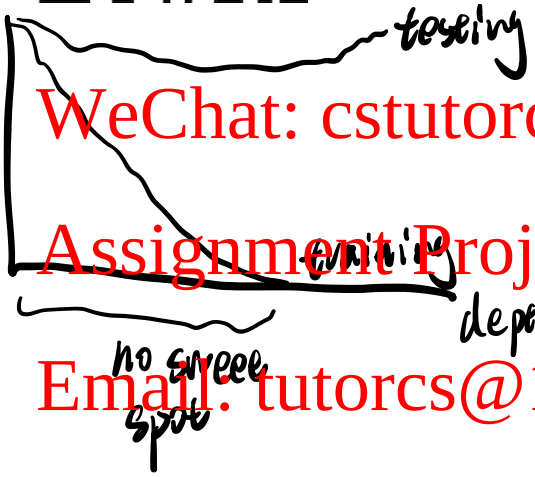
$$= \frac{4}{8} G(S_N) + \frac{4}{8} G(S_S)$$ (or you could use H instead of G)

Danger of too few splits: high bias (underfitting)

" " "many" " : high variance (overfitting)

When tree is too shallow ⇒ you'll see both training and testing error are high

⇒ you'll see training error low and testing high

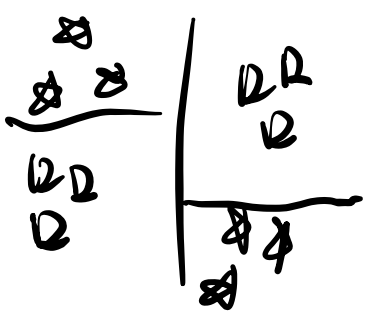


What if I go through splits along X; but entropy not improve?

Do I stop?

Answer is NO.

Ex XOR function. binary addition



1st split does not improve the entropy

2nd split round I get purity

NOTE: Decision Trees are plagued with high variance (why we use bagging)

Classification Trees (CART)

The algorithm is the same, but instead of using entropy or Gini we use square loss

$$L(S) = \frac{1}{|S|}\sum_{(\vec{x},y)\in S}(y-\bar{y}_S)^2$$

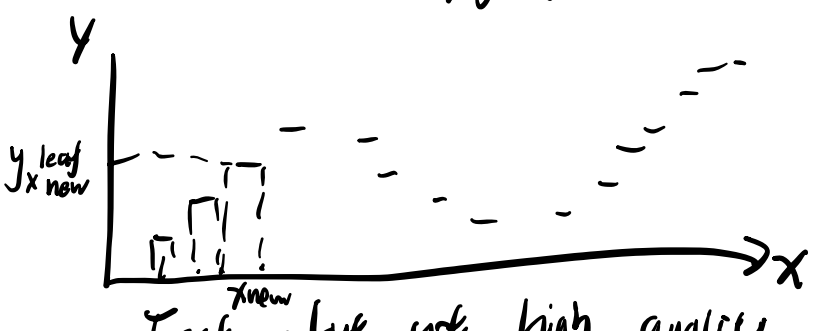
→ average square distance from label

$$\bar{y}_S = \frac{1}{|S|}\sum_{(\vec{x},y)\in S} y$$

→ average label

EX) we test splitting the x feature space into 2

you continue splitting along x to get a piece-wise constant estimate of underlying function



fast, but not high quality

Bagging (Bootstrap Aggregating)

Used to [illegible] variance in machine learning algorithms

We'll try to [illegible] x) closer to $\bar{h}(x)$

by using the weak Law of large Numbers

(the sample average of independent and identically distributed sample converges to the expectation of mean)

$$\frac{1}{N}\sum_{i=1}^{N} X_i \xrightarrow[N\to\infty]{} E(X) = \lambda$$

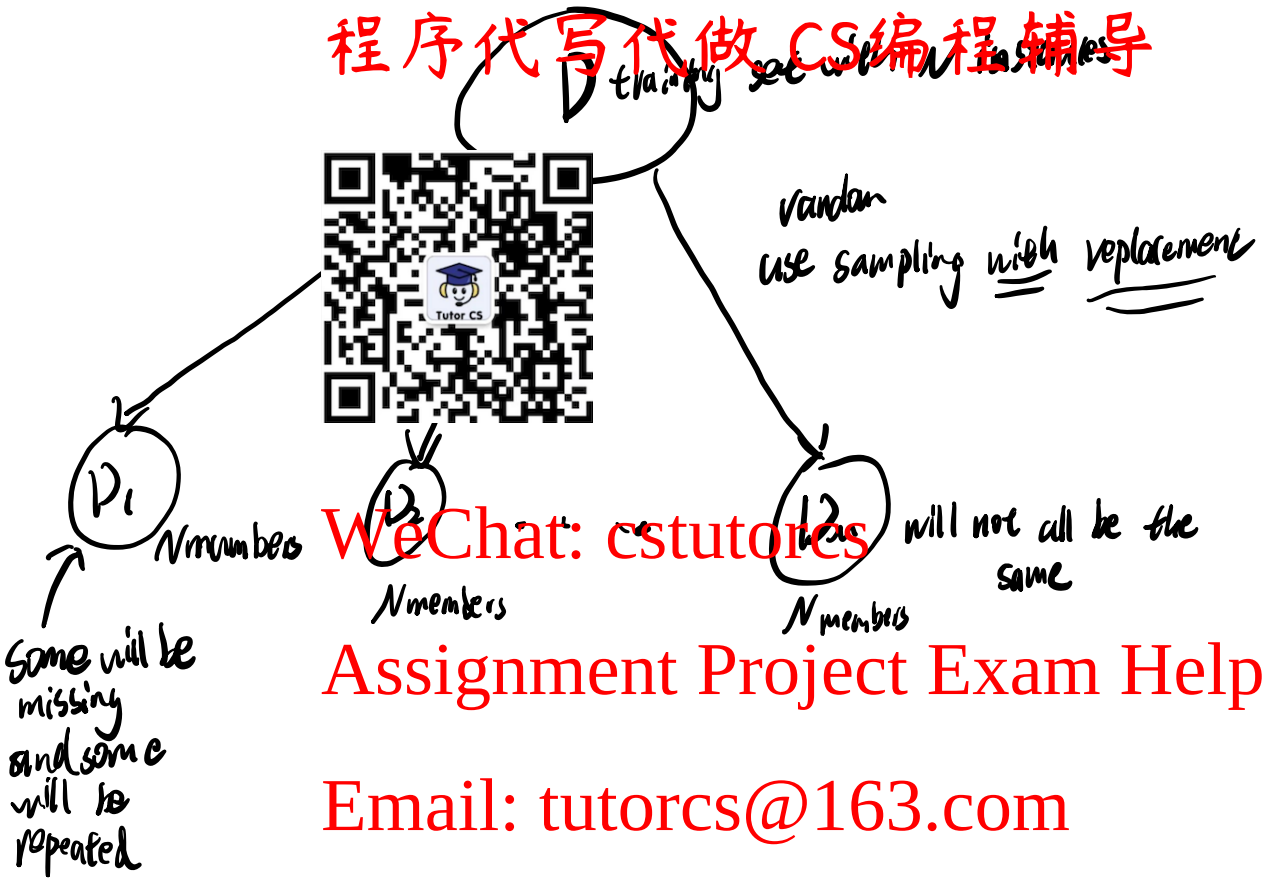
If I have M data sets, then I can

train M trees, each one based on sets D_i, $i=1\cdots M$

Each yields a classifier $h_{D_i}(x)$

Next, average them for regression $h_{avg}(x) = \frac{1}{M}\sum_{i=1}^{M} h_{D_i}(x) \xrightarrow{M\to\infty} h(x)$

Ok, if you are working with classifiers, then take the mode among all $h_{D_i}(x)$



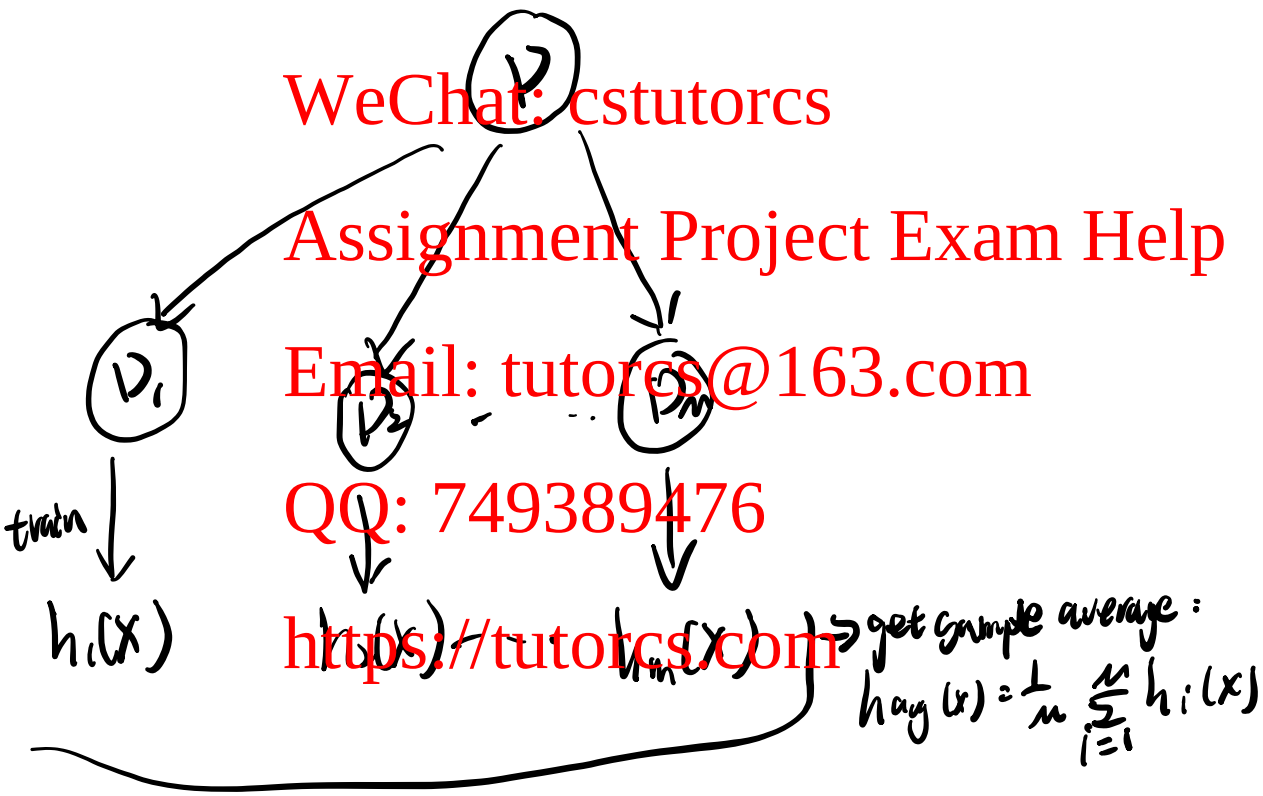
☆ Because we are allowed to sample with replacement randomly with N instances we pick from S....

some instances get repeated, and some get skipped

This mitigates overfitting

Def | "Bootstrapping" = resampling technique that involves repeatedly drawing samples from our source data with replacement.

we'll try to bring $h_{avg}(x)$ closer to $\bar{h}$ by using the weak Law of Large Numbers (the average of independent and identically distributed converges to the expectation or mean



Note, there sets are NOT independent ⇒ Weak law of Large Numbers does not apply and sample average will not converge to $\bar{h}$

But bagging effectively improves Variance!