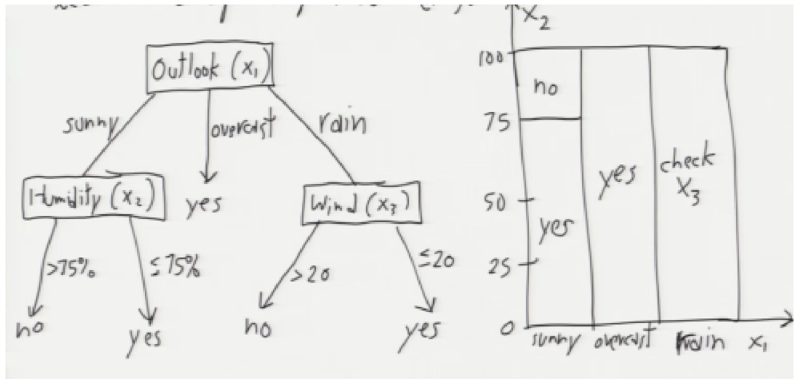
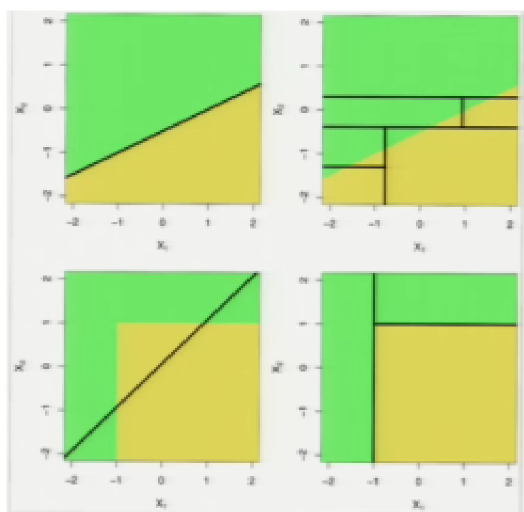
03/14/2016

Decision Trees

- Non-linear method for classification.
- Uses trees with two node types:
  - internal nodes test feature values (usually just one) and branch accordingly.
  - leaf nodes specify class $h(x)$.



- Cuts x-space into rectangular cells.
- Works well with both categorical and quantitative features.
- Interpretable result (inference).
- Decision boundary can be arbitrarily complicated.
  - Can really reduce bias if boundary is truly complicated.
  - A lot of opportunity for over-fitting.

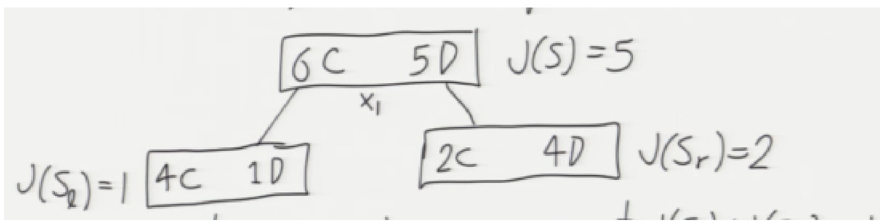


- Leaf is pure if every training sample in it has the same class.
- Consider classification first: Greedy, top-down learning heuristic:
- Let $S \subseteq \{1, 2, \ldots, n\}$ be a list of sample indices.

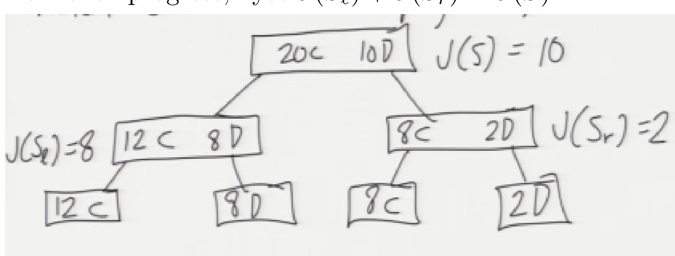
- Top level call: $S = \{1, 2, \ldots, n\}$

```
GrowTree(S):
    if (y_i = c ∀i ∈ S and some class C):
        return new leaf(C)
    else:
        choose best splitting feature j and splitting point β (*)
        S_ℓ = {i : X_ij < β}
        S_r = {: X_ij ≥ β}
        return new node(j,β,GrowTree(S_ℓ),GrowTree(S_r))
```

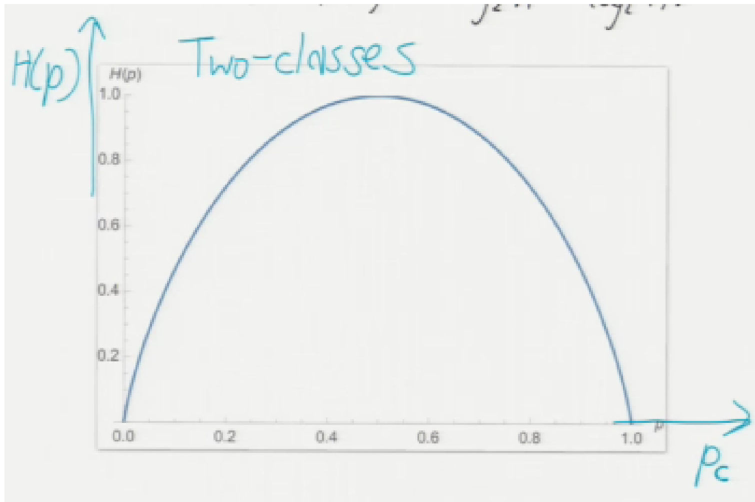
- (*) How to choose best split?
  - Try all splits.
  - For a set S, let $J(S)$ be the cost of S.
  - Choose the split that minimizes $J(S_\ell)+J(S_r)$; or, the split that minimizes the weighted average $\frac{|S_\ell| J(S_\ell)+|S_r| J(S_r)}{|S_\ell|+|S_r|}$
- How to choose cost $J(S)$?
  - Idea 1 (bad): Label S with the class C that labels the most samples in S. $J(S) \leftarrow$ number of samples in S not in class C.



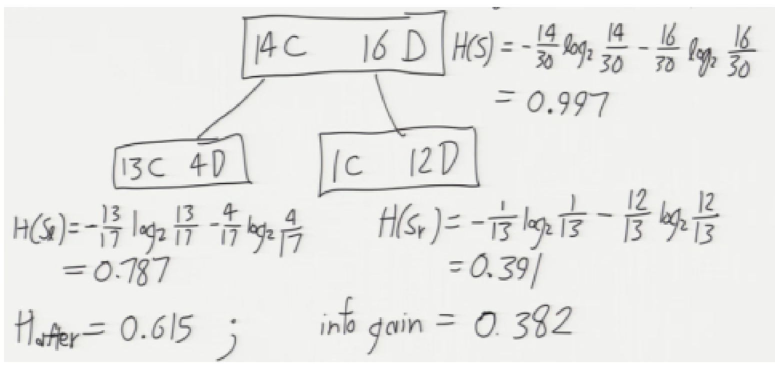
    - Problem: sometimes we make "progress," yet $J(S_\ell) + J(S_r) = J(S)$.



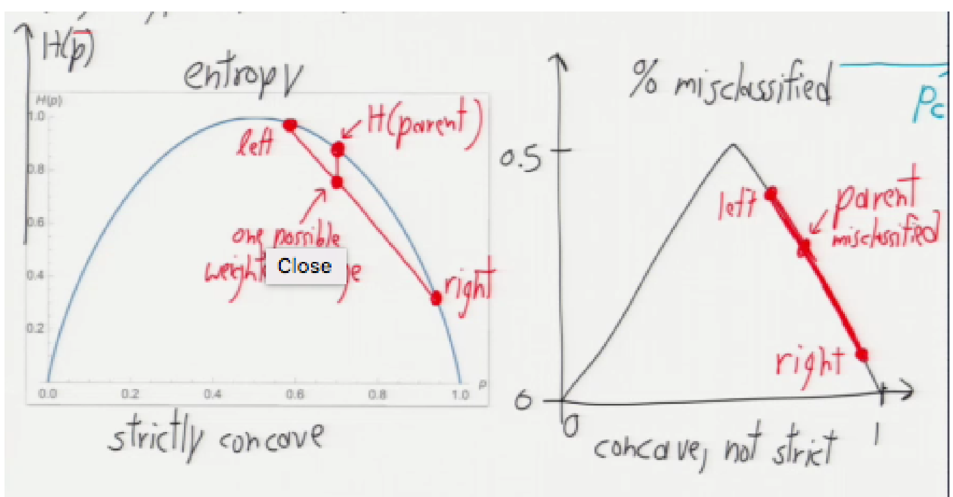
  - Idea 2 (good): Measure entropy.
    - Let Y be a random class variable, and suppose $P(Y = C) = p_c$.
    - The surprise of Y being class C is $S(Y = C) = -log_2\ p_c$.
      - event with probability 1 gives us zero surprise.
      - event with probability 0 gives us infinite surprise.
    - The entropy of an index set S is the average surprise: $H(S) = -\sum_c p_c\ log_2\ p_c$, where $p_c = \frac{|\{i \in S : y_i = C\}|}{|S|}$
    - If all samples in S belong to same class? $H(S) = -1\ log_2\ 1 = 0$.
    - Half class C, half class D? $H(S) = -\frac{1}{2}\ log_2 \frac{1}{2} - \frac{1}{2}\ log_2 \frac{1}{2} = 1$.
    - n samples, all different classes? $H(S) = -log_2\ \frac{1}{n} = log_2\ n$.



- Weighted average entropy after split is $H_{after} = \frac{|S_\ell| H(S_\ell) + |S_r| H(S_r)}{|S_\ell| + |S_r|}$
- Choose split that maximizes information gain $H(S) - H_{after}$.

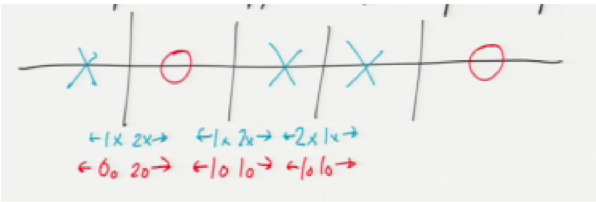


- Information gain always positive expect when one child is empty or $\forall C, P(y_i = C | i \in S_\ell) = P(y_i = C | i \in S_r)$



- More on choosing a split.
  - For binary feature x_i, children are $x_i = 0$ and $x_i = 1$.
  - If x_i has 3+ discrete values, split depends on application.

- If x_i is quantitative, sort samples in S by feature x_i; remove duplicates try splitting between each pari of consecutive samples.
- Clever Bit: As you scan sorted list from left to right, you can update entropy in $\mathcal{O}(1)$ time per sample!



- Algorithms and running times:
  - Test point: walk down tree until leaf. Return it's label.
    - Worst-case time is $\mathcal{O}$(depth tree).
    - For binary features, that's $\leq d$.
    - Usually (not always) $\leq \mathcal{O}(\log n)$.
  - Training:
    - For binary features, try $\mathcal{O}(d)$ splits at each node.
    - For quantitative features, try $\mathcal{O}(n'd)$ splits; n' = samples in node.
    - Either way, $\Rightarrow \mathcal{O}(n'd)$ time at this node.
  - Each sample participates in $\mathcal{O}$(depth) nodes, cost $\mathcal{O}(d)$ time in each node.
  - Running time $\leq \mathcal{O}(dn\text{depth})$.