Speaker's notes on Trees

Note: these were the speaker's notes I used in class. They are unpolished material distilled from the different sources listed in the "bibliography"

The elements of a tree structured classifier are:

Nodes correspond to regions of the feature space.

Internal nodes are associated with tests

Leaves correspond to decision regions, associated to labels

Edges correspond to outcomes of test

The main advantage of tree classifiers is the speed during the classification phase:
(Have $(\mathbf{X}, Y)$, observe $\mathbf{X}$, want Y.

- Start at root.
- When reach an internal node, apply the associated test to the sample to be labeled.
- Follow the branch corresponding to the test outcome.
- When a leaf is reached, assign to the sample the label of the leaf.

The idea is the same as the 20-question game

0.1 Main Questions

1. What do we use to split?
2. How many splits per node?
3. Where do we split?
4. When do we stop splitting?

5. Do we stop splitting? (Maybe we can build up the tree and then prune!)
6. How do we assign labels to leaves?
7. Are trees any good?

NOTE: the key question is "how do we assign labels?". Using that we can answer 3,4,5. So, we will answer 1,2,6,3,4,5 and 7 in the order

1 What do we use to split?

- Hyperplanes perpendicular one coordinate axis (split along one feature) - most of the lecture is based on this type of splits.
- General hyperplanes (multivariate splits)
- Spheres (sphere trees)
- More complex surfaces (these miss the point)

2 How do we split?

Have $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$

- Methods using only $\mathbf{X}_1, \ldots, \mathbf{X}_n$ to split.
- Methods using both observ. and labels

The label assignment to the leaves is the same for both cases.

Note The former methods can do better than the latter. If you are interested, look at the technical report on multidimensional indexing structures.

Most of lecture is devoted to the latter methods.

3 How many splits per node?

ONE ! Binary Trees! But:

- we will briefly discuss multiway splits.
- we must be careful with > 2 classes.

First Assume we know the distributions.

4 How do we assign labels?

Step 1: *loss*: $\ell(y, \hat{y})$: measure "cost" incurred in guessing $\hat{y}$ when truth is y.

- zero-one loss: $\ell(y, \hat{y}) = 1_{y \neq \hat{y}}$ (class.)
- squared error: $\ell(y, \hat{y}) = (y - \hat{y})^2$ (regression)
- if $y \in \{1, \dots, M\}$, $\hat{\mathbf{y}} = [\hat{P}(y=1), \dots, \hat{P}(y=M)]$ (probability vector): $\ell(y, \hat{\mathbf{y}}) = -\log \hat{P}(y)$ (log likelihood) (interpretation: the nr of bits to specify y with code for $\hat{\mathbf{y}}$).

Step 2:

- Leaves: $1, \dots, L$, $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(L)}$, decide $\hat{y}(1), \dots, \hat{y}(L)$.
- $(\mathbf{X}, Y)$, $\mathbf{X} \in \mathcal{X}^{(j)}$, decide $\hat{y} = \hat{y}(j)$.
- **Notation** $E\left[\ell(Y, \hat{y}(j)) \mid X \in \mathcal{X}^{(j)}\right] = E[\ell(Y, \hat{y}(j)) \mid j]$.

Def: ***Centroid***

$$y^*(j) \overset{\triangle}{=} \arg\min_{\hat{y}(j)} \{E[\ell(Y, \hat{y}(j)) \mid j]\}.$$

Def: ***Leaf Impurity*** [1]

$$\mathbb{I}(j) \overset{\triangle}{=} E[\ell(Y, y^*(j)) \mid j].$$

Interpret as Average Loss, conditional on $\mathbf{X} \in$ leaf.

Theorem
The centroid is the best possible assignment of a label to a leaf.
Proof Any other assignment results in a larger risk.

[1]Others use $\neq$ definition: $y^*(j) \overset{\triangle}{=} E[Y \mid j]$, let $\phi(\cdot)$ convex, then $\mathbb{I}(j) \overset{\triangle}{=} \phi(y^*(j))$.

$$
\begin{aligned}
R_{opt} &= \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} \mathbb{I}\left(j\right) \\
&= \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} E\left[\ell\left(Y, y^*\left(j\right)\right) \mid j\right] \\
&= \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} E\left[\ell\left(Y, y^*\left(j\right)\right) \mid j\right] \\
&\quad + \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} \left(E\left[\ell\left(Y, \hat{y}\left(j\right)\right) \mid j\right] - E\left[\ell\left(Y, \hat{y}\left(j\right)\right) \mid j\right]\right) \\
&= \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} E\left[\ell\left(Y, \hat{y}\left(j\right)\right) \mid j\right] \\
&\quad + \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} \left(E\left[\ell\left(Y, y^*\left(j\right)\right) \mid j\right] - E\left[\ell\left(Y, \hat{y}\left(j\right)\right) \mid j\right]\right) \\
&= \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} E\left[\ell\left(Y, \hat{y}\left(j\right)\right) \mid j\right] \\
&\quad - \sum_{j=1,\ldots,L} \Pr\left\{\mathbf{X} \in \mathcal{X}^{(j)}\right\} d\left(j, \hat{y}\left(j\right)\right) \qquad (1)
\end{aligned}
$$

Def: ***Divergence:*** $d\left(j, \hat{y}\left(j\right)\right)$ Increase in Risk
when $\hat{y}\left(j\right)$ is used instead of $y^*\left(j\right)$

Lemma Divergence is non-negative.
Why? look at definition of $y^*\left(j\right)$!

The lemma, applied to Equation 1 concudes the proof of the theorem.

5 Where and how do we split?

Impurity is "convex" in the following sense: split j into jR, and jL,

$$
\begin{aligned}
\mathbb{I}\left(j\right) &= P(jR \mid j)E\left[\ell\left(Y, y^*\left(j\right)\right) \mid jR\right] \\
&\quad +P(jL \mid J)E\left[\ell\left(Y, y^*\left(j\right)\right) \mid jL\right] \\
&\quad \text{Note that we use the same label as the original node} \\
&\geq P(jR \mid j)E\left[\ell\left(Y, y^*\left(jR\right)\right) \mid jR\right] \\
&\quad +P(jL \mid j)E\left[\ell\left(Y, y^*\left(jL\right)\right) \mid jL\right] \\
&= P(jR \mid j)\mathbb{I}\left(jR\right) + P(jL \mid j)\mathbb{I}\left(jL\right)
\end{aligned}
$$

Splitting never increases impurity. (Hence, splitting never increases the overall risk of the classifier.)

NOTE: impurity is not convex in other senses: in particular, it is not convex in the splitting point or splitting surface!

Most splitting algorithms are greedy:

- Find "best" split in each dimension;
- split along dimension with best "best split".

where "best split" maximizes the reduction in

- *Impurity*
  $\Delta\mathbb{I} = \mathbb{I}(j) - P(jR \mid j)\mathbb{I}(jR) + P(jL \mid j)\mathbb{I}(jL)$.
- *Overall risk*
  $\Delta R = P(j)\mathbb{I}(j) - P(jR)\mathbb{I}(jR) + P(jL)\mathbb{I}(jL)$.

6 When do we stop splitting ?

- *No improvement*: When the best split does not reduce risk/impurity.
- *Size Threshold method*: when the number of nodes (or the maximum depth) reaches a threshold.
- *Risk Threshold method*: when the risk of the tree is within T of the Bayes Risk.

In practice, we do not know the distributions!

Have $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$.

7 How do we assign labels?

We *estimate* $\mathbb{I}(j)$ and centroid using the samples falling in $\mathcal{X}^{(j)}$:

$$E\left[\ell\left(Y, \hat{y}(j)\mid j\right)\right] = \frac{1}{n_j}\sum_{i=1}^{n_j} \ell\left(Y_i, \hat{y}(j)\right)$$

where n_j is nr samples in $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}\bigcap \mathcal{X}^{(j)}$.

Then: centroid minimizes cond. expected loss, impurity is min of cond. loss.

7.1 Where and how do we split?

- For each dimension d, sort $X_i[d]$'s $\rightarrow X'_i[d]$'s;
- For $i = 1, \ldots, n_j - 1$: split at $\left(X'_{i+1}[d] + X'_i[d]\right)/2$; compute δ impurity of split.
- Find best splitting point for dimension d, and $\Delta\mathbb{I}_d$.
- Select dim d^* where the best split yields highest impurity reduction.

7.2 Now we know how to split, so?

Split Until Termination Condition is reached.

- Can split all nodes that can be split at each iteration, until no more are left (favors a wider tree)
- Can split only the best node of the entire tree (might have a deep tree)

Note: these are greedy algorithm: no guarantee that a split will produce children that split well

7.3 When do we stop splitting?

- **That's it**: all samples have the same feature vector.
- **Pure node**: all samples have the same label.
- **Test Set**: Training + Test set.
  Split with Training, test with Test.
  Stop when no improvement on Test set.
  (Of course, one could use Cross validation: multiple test sets...)
- **Threshold method**: no test set
  Stop when $\Delta\mathbb{I} \leq T$ for all splits.
  But: what is a good T?
- **Model Selection**: penalize for number of nodes, and for impurity: minimize
  $$\alpha \cdot \text{nr nodes} + \mathbb{I}_{\textit{all leaves}}$$
  Note: if the impurity is based on entropy, the criterion is analogous to MDL.
- **Independece between X and Y**: Test: data supports rejecting hypothesis that observations and class labels are independent?

To keep in mind: with few samples: is the split meaningful or are we deceiving ourselves?

7.4 Notes

- Much more difficult with multivariate splits (easiest with misclassification impurity);
  However: steepest descent exist that yield linear time splitting. (see Chou's paper).
- Much more difficult with non-binary trees: We do not optimize anymore on a single point but on multiple points at the same time.
- For categorical variable: consider all possible ways of dividing att. values into 2 groups.
  However: steepest descent also exist that yield linear time splitting.
- For more than 2 classes: consider assignment of classes to 2 superclasses that maximizes reduction of $\mathbb{I}$.

8 Do we need to stop splitting?

Might not Split When we should: Example XOR with 4 examples in training set.

Maybe: a good idea is

- Splitting until only 1 example/observation per node;
- Prune the tree.

8.1 Common Pruning Methods

ASSUME: $0 - 1$ **loss function.**

Reduced Error Pruning (REP) (Quinlan, 87)

- use "pruning set", separate from training set
- build full tree
- For each internal node t, with subtree $\mathcal{T}(t)$: compare $\mathbb{I}(\mathcal{T}(T))$ with $\mathbb{I}(t)$; Estimate impurities using pruning set.

- Starting from the leaves: prune branch rooted at t if $\mathbb{I}\left(\mathcal{T}\left(T\right)\right) \geq \mathbb{I}\left(t\right)$.
- Iterate until impurity starts increasing.

Properties

1. *REP finds the smallest version of the most accurate subtree with respect to the pruning set, when the* $0-1$ *loss is used.*
2. The method is linear in the number of nodes: each node is visited only once.
3. The method tends to overprune: it ignores totally the training set. Severe problem when tranining set is small.

Pessimistic Error Pruning (PEP) (Quinlan 87)

Let $n(t)$ = nr samples at node t,
$e(t)$ = nr errors at node t on training set,
$\mathcal{L}\left(t\right)$ = the set of leaves rooted at t.

- use same set for growing and pruning
- instead of "apparent" error rate $r(t)$ (error on training set) uses:
  $r(\mathcal{T}\left(t\right)) \triangleq \frac{\sum_{j\in\mathcal{L}(t)} e(j)}{\sum_{j\in\mathcal{L}(t)} n(j)}$ before pruning
  $r(t) \triangleq e(t)/n(t)$ after pruning
- Apply Binomial correction $r(t) \triangleq (e(t)+1/2)/n(t) \triangleq e^{'}(t)$, same for $r(\mathcal{T}\left(t\right))$, where
  $$e^{'}(\mathcal{T}\left(t\right)) \triangleq \left|\mathcal{L}\left(t\right)\right|/2 + \sum_{j\in\mathcal{L}(t)} e(j)$$
  need because e(original leaf) $= 0$ otherwise.
- Then pruning is better if $e^{'}(t) \leq e^{'}\left(\mathcal{T}\left(t\right)\right)$ but: too stringent
- Prune if
  $$e^{'}(t) \leq e^{'}\left(\mathcal{T}\left(t\right)\right) + SE\left[e^{'}\left(\mathcal{T}\left(t\right)\right)\right]$$
  where Standard Error (binomial error model)
  $$SE\left[e^{'}\left(\mathcal{T}\left(t\right)\right)\right] = \left[e^{'}\left(\mathcal{T}\left(t\right)\right)\cdot\frac{n(t)-e^{'}\left(\mathcal{T}\left(t\right)\right)}{n(t)}\right]^{1/2}$$

Properties

1. Where does the continuity correction come from? It can induce underpruning and overpruning.

2. Linear complexity.

Minimum Error Pruning (Niblett and Bratko)

- *Estimate* expected error rate: use estimate of probability that $(\mathbf{X}, y)$ reaches node t as

$$p_y(t) = \frac{n_y(t) + p(y) * m}{n(t) + m}$$

  where $n_y(t) = \#$ samples of class y in node t
  $p(y) = n(y)/n$ estimate of prior of class y
  m is a parameter discussed later
- Use this to compute the expected error rate for each node.
- Compare the expected error rate at a node with the expected error rate of its children (not the leaves).

Properties

1. Larger m yields severe pruning. What is a good value of m?
2. Dependence on number of classes.

Cost-Complexity Pruning (Breiman 87) (CART)

- **Step 1:**
  - build full tree
  - Compute change in apparent error rate (use training set): $r(t) - r(T_t)$.
  - Divide by the number of leaves -1:

$$\alpha_t = \frac{r(t) - r(T_t)}{\mathcal{L}_t - 1}$$

  - Prune node(s) with smallest α; yields T^0
  - Iterate to construct family of trees $[T^0, T^1, ..., T^L]$. Note: nodes with same α are pruned during the same iteration.
- **Step 2**: Select tree with maximum predictive capability.

Two approaches:

1. Use **cross-validation** sets.
   - Split training set $\mathcal{T}$ into v subsets $\mathcal{T}^1$ to $\mathcal{T}^v$.
   - $\forall i$ construct auxiliary training set $\mathcal{T} \setminus \mathcal{T}^i$
   - Construct family of classifiers as above, identify the best point at which to stop pruning (record α).
   - Combine the results to find the best overal point.

2. Use **independent pruning** set, to find which tree in the family $[T^0, T^1, ..., T^L]$ has best predictive accuracy.

Properties

1.
2. The pruning set is used to select among a family of already pruned trees: thus, the search is limited to an already selected family of trees.

Error-Based Pruning (Quinlan, '93), C4.5

- Use same training set for training & pruning
- Visits nodes bottom-up, in post-order
- Use a more pessimistic estimate of error rate than PEP
  - At each node t, use data to construct *confidence interval* for error rate. Use binomial approximation.
  - Note that, for same apparent error rate, confidence interval is wider for nodes with fewer samples
  - Use upper limit of confidence interval as estimate of error rate
- Compare error rate at a node with
  - mean error rate of leaves
  - error rate of child with largest branch
- Leave alone, prune entire branch, or graft largest subbranch.

Comments on the method:

- Great idea: grafting
- However, might end up grafting a leaf in place of a node ... not too smart
- Grafting a subtree (say, left) implies that we believe that the test in the subtree are good for the part of the space described by the right subtree (says who ?)
- In particular, we assume that the error rate would still be binomial with the same parameter.

9 Missing Features

Sometimes: sample in training set/samples to be labeled have a missing feature.

Training

- Ignore sample (lame answer).
- Ignore sample ONLY for creating split for missing feature, use for all other features.

Classify

- Define "default value"
  - Mean value
  - Median value
  - Mode
- Define *surrogate splits*:
  - Find best split
  - For each remaining feature, find split that divides the training set in the most similar way as the best split (e.g.: use number of samples that "go the wrong way");
  - Sort remaining features in "goodness of split" order.