Decision Trees and Forests
Outline

1. Motivation
2. Recursive Splits and Trees
3. Prediction
4. Purity
5. Splitting
6. Forests: Bagging and Randomization
7. Forest Training and Inference
8. Out-of-Bag Statistical Risk Estimate
Linear Predictors $\rightarrow$ Trees $\rightarrow$ Forests

- Linear predictors:
  + Few parameters $\rightarrow$ Good generalization, efficient training
  + Convex risk $\rightarrow$ Unique minimum risk, easy optimization
  + Score-based $\rightarrow$ Measure of confidence
  - Few parameters $\rightarrow$ Limited expressiveness

- SVMs + kernels:
  + All of the advantages of linear predictors
  + Boundaries are nonlinear
  - Need to design kernels to shape the boundary

- Decision trees:
  - Arbitrarily expressive: Flexible, but generalizes poorly
  - Interpretable: We can audit a decision

- Random decision forests:
  - Ensembles of trees that vote on an answer
  - Expressive (somewhat less than trees), generalize well
Splitting $X$ Recursively
A Decision Tree

Choose splits to maximize purity

\[
\begin{align*}
\text{a: } j &= 2 \\
& \quad t = 0.265 \\
p &= [0, 1, 0] \\

\text{b: } j &= 1 \\
& \quad t = 0.41 \\
p &= [1, 0, 0] \\

\text{c: } j &= 2 \\
& \quad t = 0.34 \\
p &= [1, 0, 0] \\

\text{d: } j &= 1 \\
& \quad t = 0.16 \\
p &= [0, 0, 1] \\

\text{e: } j &= 2 \\
& \quad t = 0.55 \\
p &= [1, 0, 0] \\
p &= [0, 0, 1]
\end{align*}
\]
What’s in a Node

- Internal:
  - Split parameters: Dimension $j \in \{1, \ldots, d\}$, threshold $t \in \mathbb{R}$
  - Pointers to children, corresponding to subsets of $S$:
    
    \[
    L \overset{\text{def}}{=} \{(x, y) \in S \mid x_j \leq t\} \\
    R \overset{\text{def}}{=} \{(x, y) \in S \mid x_j > t\}
    \]

- Leaf: Distribution of training values $y$ in this subset of $X$:
  - $p$, discrete for classification, histogram for regression

At inference time, return a *summary* of $p$ as the value for the leaf

- Mode (majority) for a classifier
- Mean or median for a regressor
  (Remember $k$-NN?)
Why Store $p$?

- Can’t we just store $\text{summary}(p)$ at the leaves?
- With $p$, we can compute a confidence value
- (More important) We need $p$ at every node during training to evaluate purity
Prediction

function $y \leftarrow \text{predict}(x, \tau, \text{summary})$
  if leaf?($\tau$) then
    return $\text{summary}(\tau.p)$
  else
    return $\text{predict}(x, \text{split}(x, \tau), \text{summary})$
  end if
end function

function $\tau \leftarrow \text{split}(x, \tau)$
  if $x_{\tau.j} \leq \tau.t$ then
    return $\tau.L$
  else
    return $\tau.R$
  end if
end function
Design Decisions for Training

• How to define (im)purity
• How to find optimal split parameters $j$ and $t$
• When to stop splitting
Impurity Measure 1: The Error Rate

- Simplest option: \( i(S) = \overline{\text{err}}(S) = 1 - \max_y p(y|S) \)
- \( S \): subset of \( T \) that reaches the given node
- Interpretation:
  - Put yourself at node \( \tau \)
  - The distribution of training-set labels that are routed to \( \tau \) is that of the labels in \( S \)
  - If the distribution is representative:
    - The best the classifier can do is to pick the label with the highest fraction, \( \max_y p(y|S) \)
    - \( \overline{\text{err}}(S) \) is *the probability that the classifier is wrong at \( \tau \)* (empirical risk)
Impurity Measure 2: The Gini Index

- A classifier that always picks the most likely label does best at inference time.
- However, it ignores all other labels at training time.
  \[ p = [0.5, 0.49, 0.01] \] same error rate as \[ q = [0.5, 0.25, 0.25] \]
- In \( p \), we have almost eliminated the third label.
- \( q \) closer to uniform, perhaps less desirable.
- For evaluating splits (only), consider a stochastic predictor:
  \[ \hat{y} = h_{\text{Gini}}(x) = y \text{ with probability } p(y|S) \]
- The Gini index measures the empirical risk for the stochastic predictor (looks at all of \( p \), not just \( p_{\max} \)).
- Says that \( p \) is a bit better than \( q \): \( p \) is less impure than \( q \).
- \( i(S_p) \approx 0.51 \) and \( i(S_q) \approx 0.62 \).
The Gini Index

- **Stochastic predictor**: Draw answer at random from $p(y|S)$
  \[ \hat{y} = h_{\text{Gini}}(x) = y \text{ with probability } p(y|S) \text{ for } y \in Y \]
- What is the empirical risk for $h_{\text{Gini}}$?
- Answer $y$ is chosen with probability $p(y|S)$
- Answer $y$ is wrong with probability $1 - p(y|S)$
- Given that the answer is $y$ the error rate is $1 - p(y|S)$
- Therefore, the expected error rate is
  \[ i(S) = L_S(h_{\text{Gini}}) = \sum_{y \in Y} p(y|S)(1 - p(y|S)) = 1 - \sum_{y \in Y} p^2(y|S) \]
How to Split

- Split at training time:
  If training subset $S$ made it to the current node, put all samples in $S$ into either $L$ or $R$ by the split rule.
- Split at inference time: Send $x$ either to $\tau.L$ or to $\tau.R$.
- Either way:
  - Choose (training) or retrieve (inference) a dimension $j$ in $\{1, \ldots, d\}$.
  - Choose (training) or retrieve (inference) a threshold $t$.
  - Any data point for which $x_j \leq t$ goes to $\tau.L$.
  - All other points go to $\tau.R$.
- How to pick $j$ and $t$ at training time?
How to Pick $j$ and $t$ at Each Node?

- Try all possibilities and pick the best
- “Best:” Maximizes the decrease in impurity:
  $$\Delta i(S, L, R) = i(S) - \frac{|L|}{|S|}i(L) - \frac{|R|}{|S|}i(R)$$
- “All possibilities:” Choices are finite in number
  - Sorted unique values in $x_j$ across $T$: $x_j^{(0)}, \ldots, x_j^{(u_j)}$
  - Possible thresholds: $t = t_j^{(1)}, \ldots, t_j^{(u_j)}$
    where $t_j^{(\ell)} = \frac{x_j^{(\ell-1)} + x_j^{(\ell)}}{2}$ for $\ell = 1, \ldots, u_j$
- Nested loop: for $j = 1, \ldots, d$
  for $t = t_j^{(1)}, \ldots, t_j^{(u_j)}$
- Efficiency hacks are possible
Stopping too Soon is Dangerous

- Temptation: Stop when impurity does not decrease
When to Stop Splitting

• Possible stopping criteria
  • Impurity is zero
  • Too few samples in either $L$ or $R$
  • Maximum depth reached
• Overgrow the tree, then prune it
• There is no optimal pruning method
  (Finding the optimal tree is NP-hard)
  (Reduction from set cover problem, Hyafil and Rivest)
• Better option: *Random Decision Forests*
Summary: Training a Decision Tree

• Use exhaustive search at the root of the tree to find the dimension \( j \) and threshold \( t \) that splits \( T \) with the biggest decrease in impurity

• Store \( j \) and \( t \) at the root of the tree

• Make new children with \( L \) and \( R \)

• Repeat on the two subtrees until some criterion is met
Summary: Predicting with a Decision Tree

- Use $\tau.j$ and $\tau.t$ at the root $\tau$ to see if $x$ belongs in $\tau.L$ or $\tau.R$
- Go to the appropriate child
- Repeat until a leaf is reached
- Return $\text{summary}(p)$
- $\text{summary}$ is majority for a classifier, mean or median for a regressor
From Trees to Forests

- Trees are flexible $\rightarrow$ good expressiveness
- Trees are flexible $\rightarrow$ poor generalization
- Pruning is an option, but messy and heuristic
- *Random Decision Forests* let several trees vote
- Use the bootstrap to give different trees different views of the data
- Randomize split rules to make trees even more independent
Random Forests

- $M$ trees instead of one
- Train trees to completion (perfectly pure leaves) or to near completion (few samples per leaf)
- Give tree $m$ training bag $B_m$
  - Draw $|T|$ training samples independently at random with replacement out of $T$
  - $|B_m| = |T|$  
  - About 63% of samples from $T$ are in $B_m$
- Make trees more independent by randomizing split dim:
  - Original trees: for $j = 1, \ldots, d$
    - for $t = t_j^{(1)}, \ldots, t_j^{(u_j)}$
  - Forest trees: $j$ = random out of $1, \ldots, d$
    - for $t = t_j^{(1)}, \ldots, t_j^{(u_j)}$
Randomizing Split Dimension

\[ j = \text{random out of } 1, \ldots, d \]
\[ \text{for } t = t_j^{(1)}, \ldots, t_j^{(u_j)} \]

- Still search for the optimal threshold
- Give up optimality for independence
- Dimensions are revisited anyway in a tree
- Tree may get deeper, but still achieves zero training risk
- Independent splits and different data views lead to good generalization when voting
- Bonus: training a single tree is now \( d \) times faster
function \phi \leftarrow \text{trainForest}(T, M) \quad \triangleright M \text{ is the desired number of trees}
\phi \leftarrow \emptyset \quad \triangleright \text{The initial forest has no trees}
\text{for } m = 1, \ldots, M \text{ do}
\quad S \leftarrow |T| \text{ samples unif. at random out of } T \text{ with replacement}
\quad \phi \leftarrow \phi \cup \{\text{trainTree}(S, 0)\} \quad \triangleright \text{Slightly modified } \text{trainTree}
\text{end for}
end function
Inference

```
function y ← forestPredict(x, φ, summary)
    V = {} ▷ A set of values, one per tree, initially empty
    for τ ∈ φ do
        y ← predict(x, τ, summary) ▷ The predict function for trees
        V ← V ∪ {y}
    end for
    return summary(V)
end function
```
Out-of-Bag Statistical Risk Estimate

- Random forests have “built-in” training/validation or training/testing splits
- Tree $m$: $B_m$ for training, $V_m = T \setminus B_m$ for testing
- $h_{ooob}$ is a predictor that works only for $(x_n, y_n) \in T$:
  - Let tree $m$ vote for $y$ only if $x_n \notin B_m$
  - $h_{ooob}(x_n)$ is the summary of the votes over participating trees
  - Summary: majority (classification); mean, median (regression)
- Out-of-bag risk estimate:
  - $T' = \{ t \in T : \exists m \text{ such that } t \notin B_m \}$
    (samples that were left out of some bag, so some trees can vote on them)
  - Statistical risk estimate: empirical risk of $h_{ooob}$ over $T'$:
    $$L_{T'}(h_{ooob}) = \frac{1}{|T'|} \sum_{(x,y) \in T'} \ell(y, h_{ooob}(x))$$
Out-of-Bag Statistical Risk Estimate

\[ T' \approx T \]

- \( L_{T'}(h_{\text{oob}}) \) can be shown to be an unbiased estimate of the statistical risk
- No separate test set needed if \( T' \) is large enough
- How big is \( T' \)?
- \(|T'|\) has a binomial distribution over \( N \) points, \( p = 1 - (1 - 0.37)^M \approx 1 \) as soon as \( M > 20 \)
- \( p \) = probability that a sample is not included in all bags (so it gets an OOB prediction)
- Mean \( \mu = pN \), variance \( \sigma^2 = p(1 - p)N \)
- \( \sigma / \mu = \sqrt{\frac{1-p}{pN}} \rightarrow 0 \) quite rapidly with growing \( M \) and \( N \)
- For large \( M, N \), the size of \( T' \) is very predictably close to \( N \): All samples in \( T \) are also in \( T' \) nearly always
Summary of Random Forests

- Random views of the training data by bagging
- Independent decisions by randomizing split dimensions
- Ensemble voting leads to good generalization
- Number $M$ of trees tuned by OOB validation
- OOB estimate can replace final testing
- (In practice, that won’t fly for papers)
- More efficient to train than a single tree if $M < d$
- Still rather efficient otherwise, and parallelizable
- *Conceptually simple, easy to adapt to different problems*
- Lots of freedom about split rule
- Example: Hybrid regression/classification problems