Decision Trees and Forests

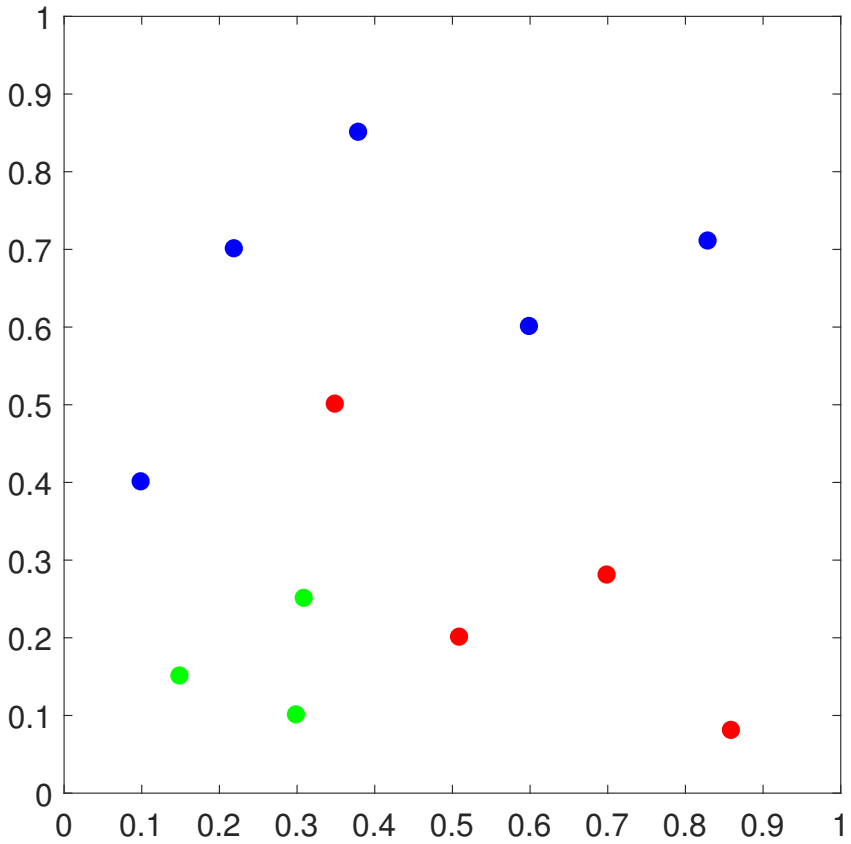
COMPSCI 371D — Machine Learning

Outline

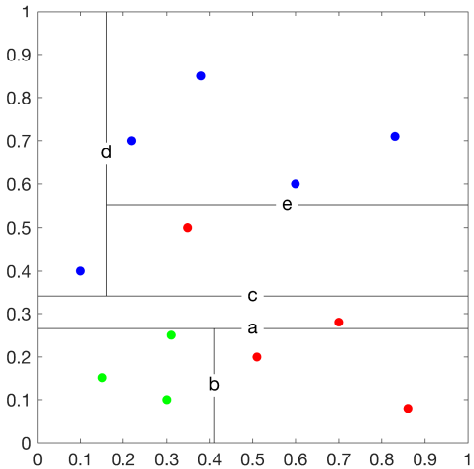
Linear Predictors → Trees → Forests

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

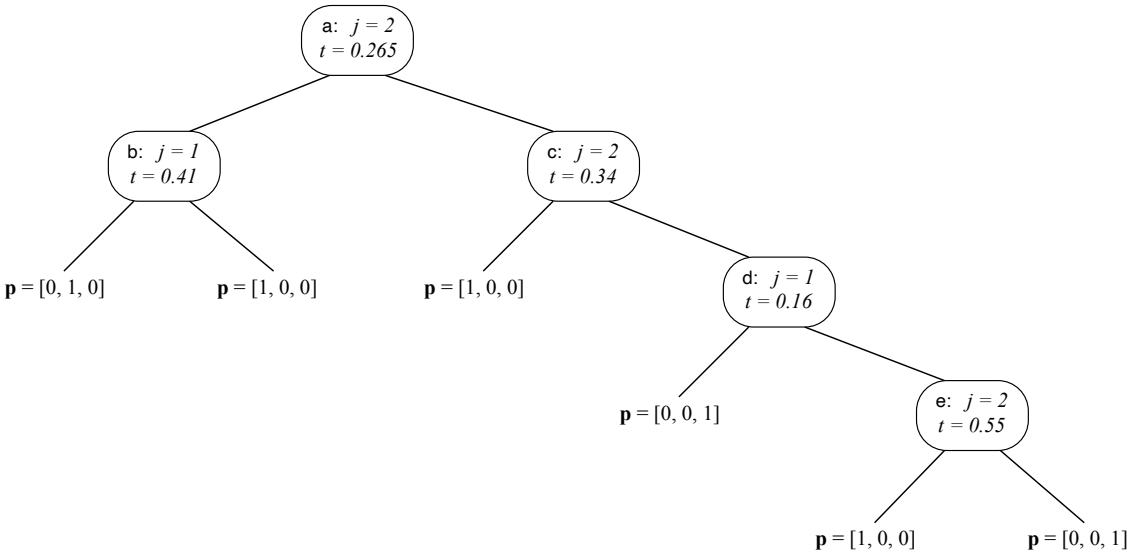
Splitting X Recursively



A Decision Tree



Choose splits to maximize *purity*



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:

$$L \stackrel{\text{def}}{=} \{(\mathbf{x}, y) \in S \mid x_j \leq t\}$$
$$R \stackrel{\text{def}}{=} \{(\mathbf{x}, y) \in S \mid x_j > t\}$$

- Leaf: Distribution of training values y in this subset of X:

  $\mathbf{p}$, discrete for classification, histogram for regression

- At inference time, return a *summary* of $\mathbf{p}$ 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 $\mathrm{summary}(\mathbf{p})$ 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 $y \leftarrow \text{predict}(\mathbf{x}, \tau, \text{summary})$
    if $\text{leaf?}(\tau)$ **then**
        return $\text{summary}(\tau.\mathbf{p})$
    else
        return $\text{predict}(\mathbf{x}, \text{split}(\mathbf{x}, \tau), \text{summary})$
    end if
end function

function $\tau \leftarrow \text{split}(\mathbf{x}, \tau)$
    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: $\mathrm{i}(S) = \overline{\mathrm{err}}(S) = 1 - \max_y p(y|S)$
- S: subset of T that reaches the given node
- Interpretation:
  - Put yourself at node τ
  - The distribution of training-set labels that are routed to τ 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, $\max_y p(y|S)$
    - $\overline{\mathrm{err}}(S)$ is *the probability that the classifier is wrong at* τ (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

  $\mathbf{p} = [0.5, 0.49, 0.01]$ same error rate as $\mathbf{q} = [0.5, 0.25, 0.25]$
- In $\mathbf{p}$, we have almost eliminated the third label
- $\mathbf{q}$ closer to uniform, perhaps less desirable
- For evaluating splits (only), consider a *stochastic predictor*:

  $\hat{y} = h_{\text{Gini}}(\mathbf{x}) = y$ with probability $p(y|S)$
- The *Gini index* measures the empirical risk for the stochastic predictor (looks at all of $\mathbf{p}$, not just $p_{\max}$)
- Says that $\mathbf{p}$ is a bit better than $\mathbf{q}$: $\mathbf{p}$ is less impure than $\mathbf{q}$
- $\mathrm{i}(S_{\mathbf{p}}) \approx 0.51$ and $\mathrm{i}(S_{\mathbf{q}}) \approx 0.62$

The Gini Index

- *Stochastic predictor*: Draw answer at random from $p(y|S)$
  $\hat{y} = h_{\text{Gini}}(\mathbf{x}) = y$ with probability $p(y|S)$ for $y \in Y$
- What is the empirical risk for h_{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
  $\mathrm{i}(S) = L_S(h_{\text{Gini}}) = \sum_{y \in Y} p(y|S)(1 - p(y|S)) =$
  $1 - \sum_{y \in Y} p^2(y|S)$

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 $\mathbf{x}$ 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?

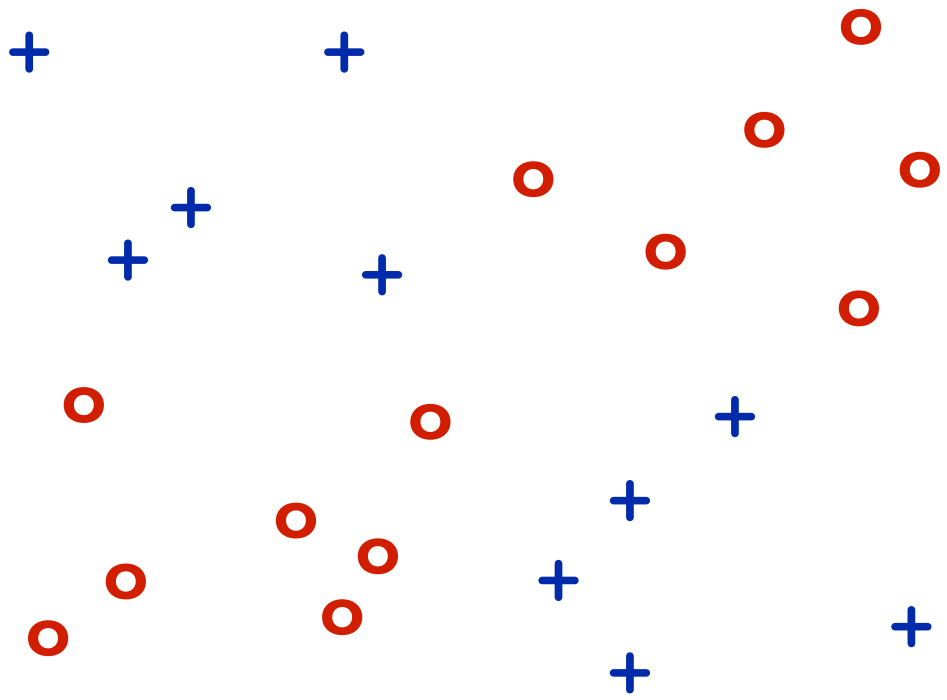
- Try all possibilities and pick the best
- "Best:" Maximizes the decrease in impurity:
  $\Delta \mathrm{i}(S, L, R) = \mathrm{i}(S) - \frac{|L|}{|S|}\mathrm{i}(L) - \frac{|R|}{|S|}\mathrm{i}(R)$
- "All possibilities:" Choices are finite in number
  - Sorted unique values in x_j across T: $x_j^{(0)}, \ldots, x_j^{(u_j)}$
  - 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

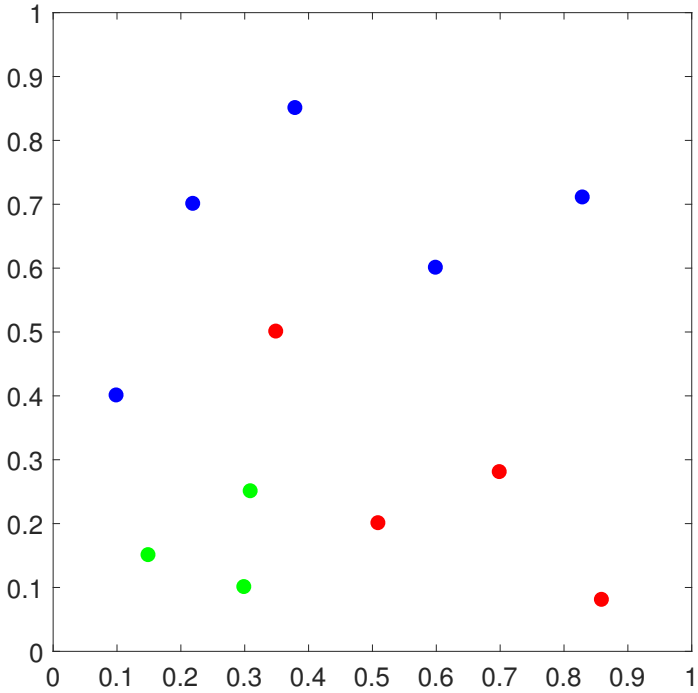
- Possible stopping criteria
  - Impurity is zero
  - Too few samples in either L or R
  - Maximum depth reached
- Overgrow the tree, then prune it
- There is no optimal pruning method

  (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 τ to see if $\mathbf{x}$ belongs in $\tau.L$ or $\tau.R$
- Go to the appropriate child
- Repeat until a leaf is reached
- Return $\text{summary}(\mathbf{p})$
- 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: $\texttt{for } j = 1, \ldots, d$
    $$\texttt{for } t = t_j^{(1)}, \ldots, t_j^{(u_j)}$$
  - Forest trees: $j = \texttt{random out of } 1, \ldots, d$
    $$\texttt{for } t = t_j^{(1)}, \ldots, t_j^{(u_j)}$$

Randomizing Split Dimension

$j = \texttt{random out of } 1, \ldots, d$

$\texttt{for } t = t_j^{(1)}, \ldots, t_j^{(u_j)}$

- 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

Training

function $\phi \leftarrow \text{trainForest}(T, M)$ ▷ M is the desired number of trees
  $\phi \leftarrow \emptyset$ ▷ The initial forest has no trees
  for $m = 1, \ldots, M$ **do**
    $S \leftarrow |T|$ samples unif. at random out of T with replacement
    $\phi \leftarrow \phi \cup \{\text{trainTree}(S, 0)\}$ ▷ Slightly modified trainTree
  end for
end function

Inference

function $y \leftarrow \text{forestPredict}(\mathbf{x}, \phi, \text{summary})$
  $V = \{\}$ ▷ A set of values, one per tree, initially empty
  for $\tau \in \phi$ **do**
    $y \leftarrow \text{predict}(\mathbf{x}, \tau, \text{summary})$ ▷ The predict function for trees
    $V \leftarrow V \cup \{y\}$
  end for
  return $\text{summary}(V)$
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_{oob} is a predictor that works only for $(\mathbf{x}_n, y_n) \in T$:
  - Let tree m vote for y only if $\mathbf{x}_n \notin B_m$
  - $h_{\text{oob}}(\mathbf{x}_n)$ is the summary of the votes over participating trees
  - Summary: majority (classification); mean, median (regression)
- Out-of-bag risk estimate:
  - $T' = \{t \in T \mid \exists\, m \text{ such that } t \notin B_m\}$ (samples that were left out of *some* bag, so some trees can vote on them)
  - Statistical risk estimate: empirical risk of h_{oob} over T': $L_{T'}(h_{\text{oob}}) = \frac{1}{|T'|} \sum_{(\mathbf{x},y)\in T'} \ell(y, h_{\text{oob}}(\mathbf{x}))$

$T' \approx T$

- $L_{T'}(h_{\text{oob}})$ 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 $\sigma^2 = p(1-p)N$
- $\sigma/\mu = \sqrt{\frac{1-p}{pN}} \to 0$ 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