Random Forests

COMPSCI 371D — Machine Learning

Outline

1. Motivation
2. Bagging
3. Randomizing Split Dimension
4. Training
5. Inference
6. Out-of-Bag Statistical Risk Estimate

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
  - Training samples drawn 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 loss
- Independent splits and different data views
  lead to good generalization when voting
- Bonus: training a single tree is now d times faster
- Can be easily parallelized

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” test 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)
  - Statistical risk estimate: empirical risk over T':
    $e_{\text{oob}}(h, T') = \frac{1}{|T'|} \sum_{(\mathbf{x},y)\in T'} \ell(y, h_{\text{oob}}(\mathbf{x}))$

$T' \approx T$

- $e_{\text{oob}}(h, T')$ 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 with N points, $p = 1 - (1 - 0.37)^M \approx 1$ as soon as $M > 20$
- Mean $\mu \approx pN$, variance $\sigma^2 \approx p(1 - p)N$
- $\sigma/\mu \approx \sqrt{\frac{1-p}{pN}} \to 0$ quite rapidly with growing M and N
- For reasonably large N, the size of T' is very predictably about N: Practically all samples in T are also in T'

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 cross-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