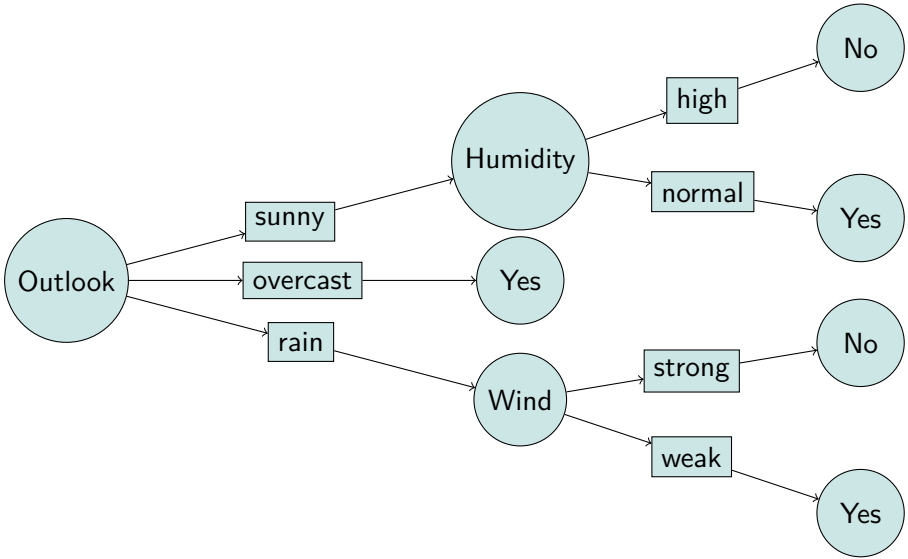
Lecture 24: Other (Non-linear) Classifiers: Decision Tree Learning, Boosting, and Support Vector Classification

Instructor: Prof. Ganesh Ramakrishnan

Decision Trees: Cascade of step functions on individual features



Use cases for Decision Tree Learning

The Canonical Playtennis Dataset

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision tree representation

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

Top-Down Induction of Decision Trees

Main loop:

1. $\phi_i \leftarrow$ the "best" decision attribute for next *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?

Top-Down Induction of Decision Trees

Main loop:

1. $\phi_i \leftarrow$ the "best" decision attribute for next *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?

Answer: That which brings about maximum reduction in impurity $\mathbf{Imp}(S_v)$ of the data subset $S_v \subseteq \mathcal{D}$ induced by $\phi_i = v$.

Top-Down Induction of Decision Trees

Main loop:

1. $\phi_i \leftarrow$ the "best" decision attribute for next *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?

Answer: That which brings about maximum reduction in impurity $\mathbf{Imp}(S_v)$ of the data subset $S_v \subseteq \mathcal{D}$ induced by $\phi_i = v$.

- S is a sample of training examples, p_{C_i} is proportion of examples belonging to class C_i in S
- Entropy measures impurity of S: $H(S) \equiv \sum_{i=1}^{K} -p_{C_i} \log_2 p_{C_i}$
- $Gain(S, \phi_i) =$ expected reduction in entropy due to splitting/sorting on ϕ_i
  $Gain(S, \phi_i) \equiv H(S) - \sum_{v \in Values(\phi_i)} \frac{|S_v|}{|S|} H(S_v)$

Common Impurity Measures (Tutorial 9)

$$\phi_s = \arg\max_{V(\phi_i),\phi_i} \left(Imp(S) - \sum_{v_{ij} \in V(\phi_i)} \frac{|S_{v_{ij}}|}{|S|} Imp(S_{v_{ij}}) \right)$$

where $S_{ij} \subseteq \mathcal{D}$ is a subset of dataset such that each instance x has attribute value $\phi_i(x) = v_{ij}$.

Name	$Imp(S)$
Entropy	$-\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$
Gini Index	$\sum_{i=1}^{K} Pr(C_i)(1 - Pr(C_i))$
Class (Min Prob) Error	$\operatorname{argmin}_i (1 - Pr(C_i))$

Table: Decision Tree: Impurity measurues

These measure the extent of spread /confusion of the probabilities over the classes

Alternative impurity measures (Tutorial 9)

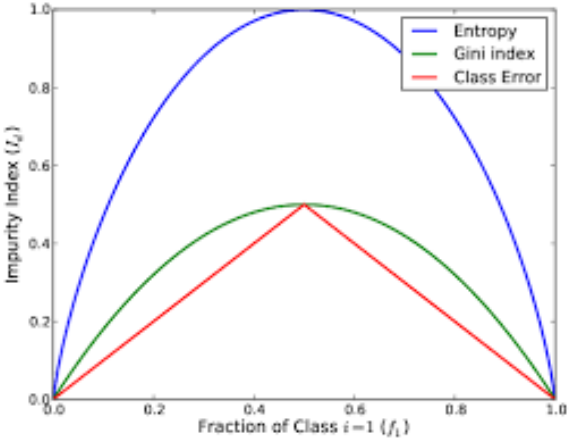


Figure: Plot of Entropy, Gini Index and Misclassification Accuracy. Source: `https://inspirehep.net/record/1225852/files/TPZ_Figures_impurity.png`

Regularization in Decision Tree Learning

- Premise: Split data into *train* and *validation* set[1]

[1] *Note:* The *test set* still remains separate
[2] Like we discussed in the case of Convolutional Neural Networks
[3] Prefer the shortest hypothesis that fits the data

Regularization in Decision Tree Learning

- Premise: Split data into *train* and *validation* set[1]
- Structural Regularization[2] based on Occam's razor[3]
  1. stop growing when data split not statistically significant
     - ⋆ Use parametric/non-parametric hypothesis tests

[1] *Note:* The *test set* still remains separate

[2] Like we discussed in the case of Convolutional Neural Networks

[3] Prefer the shortest hypothesis that fits the data

Regularization in Decision Tree Learning

- Premise: Split data into *train* and *validation* set[1]
- Structural Regularization[2] based on Occam's razor[3]
  1. stop growing when data split not statistically significant
     - ⋆ Use parametric/non-parametric hypothesis tests
  2. grow full tree, then post-prune tree
     - ⋆ *Minimum Description Length* (MDL): minimize $size(tree) + size(misclassifications_{val}(tree))$
     - ⋆ Achieved as follows: Do until further pruning is harmful
       (1) Evaluate impact on *validation* set of pruning each possible node (plus those below it)
       (2) Greedily remove the one that most improves *validation* set accuracy

[1] *Note:* The *test set* still remains separate
[2] Like we discussed in the case of Convolutional Neural Networks
[3] Prefer the shortest hypothesis that fits the data

Regularization in Decision Tree Learning

- Premise: Split data into *train* and *validation* set[1]
- Structural Regularization[2] based on Occam's razor[3]
  1. stop growing when data split not statistically significant
     - ⋆ Use parametric/non-parametric hypothesis tests
  2. grow full tree, then post-prune tree
     - ⋆ *Minimum Description Length* (MDL): minimize $size(tree) + size(misclassifications_{val}(tree))$
     - ⋆ Achieved as follows: Do until further pruning is harmful
       (1) Evaluate impact on *validation* set of pruning each possible node (plus those below it)
       (2) Greedily remove the one that most improves *validation* set accuracy
  3. convert tree into a set of rules and post-prune each rule independently (C4.5 Decision Tree Learner)

[1] *Note:* The *test set* still remains separate
[2] Like we discussed in the case of Convolutional Neural Networks
[3] Prefer the shortest hypothesis that fits the data

General Minimum Description Length

- Data is D and theory about the data is T.
- MDL principle: Define $I(D|T)$ and $I(T)$ and choose T such that it minimizes $I(D|T) + I(T)$.
- Also aligned with the *Occam Razor* principle.
- *Bayes Estimation*: $I(D|T) = \log P(D|T)$ and $I(T) = \log P(T)$

General Feature Selection based on Gain

- S is a sample of training examples, p_{C_i} is proportion of examples with class C_i in S
- Entropy measures impurity of S: $H(S) \equiv \sum_{i=1}^{K} -p_{C_i} \log_2 p_{C_i}$
- Selecting R best attributes: Let $\mathcal{R} = \emptyset$
- $Gain(S, \phi_i)$ = expected **Gain** due to choice of ϕ_i Eg: Gain based on entropy - $Gain(S, \phi_i) \equiv H(S) - \sum_{v \in Values(\phi_i)} \frac{|S_v|}{|S|} H(S_v)$

Do:

1. $\phi^* = \underset{\phi_i}{\text{argmax}}\, Gain(S, \phi_i)$
2. $\mathcal{R} = \mathcal{R} \cup \{\phi^*\}$

Until $|\mathcal{R}| = R$

General Feature Selection based on Gain

- S is a sample of training examples, p_{C_i} is proportion of examples with class C_i in S
- Entropy measures impurity of S: $H(S) \equiv \sum_{i=1}^{K} -p_{C_i} \log_2 p_{C_i}$
- Selecting R best attributes: Let $\mathcal{R} = \emptyset$
- $Gain(S, \phi_i)$ = expected **Gain** due to choice of ϕ_i Eg: Gain based on entropy - $Gain(S, \phi_i) \equiv H(S) - \sum_{v \in Values(\phi_i)} \frac{|S_v|}{|S|} H(S_v)$

  Do:

  1. $\phi^* = \underset{\phi_i}{\text{argmax}}\, Gain(S, \phi_i)$
  2. $\mathcal{R} = \mathcal{R} \cup \{\phi^*\}$

  Until $|\mathcal{R}| = R$

Q: What other measures of **Gain** could you think of?

Injecting Randomness: Bagging and Ensemble

Main loop:

1. $\phi_i \leftarrow$ **"best" decision attribute for next** *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. **Sort training examples to leaf nodes**
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Steps (1) and (4) prohibitive with large numbers of attributes (1000s) and training examples (100000s). **Alternatives?**

Injecting Randomness: Bagging and Ensemble

Main loop:

1. $\phi_i \leftarrow$ **"best" decision attribute for next** *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. **Sort training examples to leaf nodes**
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Steps (1) and (4) prohibitive with large numbers of attributes (1000s) and training examples (100000s). **Alternatives?**

- Uniformly at random (with replacements), sample subsets $\mathcal{D}_s \subseteq \mathcal{D}$ of the training data, $\Phi_s \subseteq \Phi$ of the attribute set and construct decision tree T_s for each such random subset.
- Random Forest Algorithm:

Injecting Randomness: Bagging and Ensemble

Main loop:

1. $\phi_i \leftarrow$ **"best" decision attribute for next** *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. **Sort training examples to leaf nodes**
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Steps (1) and (4) prohibitive with large numbers of attributes (1000s) and training examples (100000s). **Alternatives?**

- Uniformly at random (with replacements), sample subsets $\mathcal{D}_s \subseteq \mathcal{D}$ of the training data, $\Phi_s \subseteq \Phi$ of the attribute set and construct decision tree T_s for each such random subset.
- Random Forest Algorithm: For $s = 1$ to B repeat:
  1. **Bagging:** Draw a bootstrap sample $\mathcal{D}_s$ of size m_s from the training data $\mathcal{D}$ of size m
  2. Grow a random decision tree T_s to $\mathcal{D}_s$ by recursively repeating steps (1) - (5) of decision tree construction algorithm,, with following difference to step (1)

Injecting Randomness: Bagging and Ensemble

Main loop:

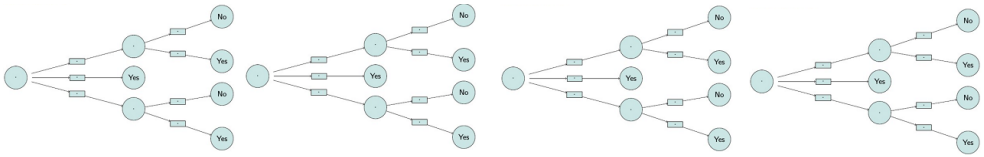
1. $\phi_i \leftarrow$ **"best" decision attribute for next** *node*
2. Assign ϕ_i as decision attribute for *node*
3. For each value of ϕ_i, create new descendant of *node*
4. **Sort training examples to leaf nodes**
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Steps (1) and (4) prohibitive with large numbers of attributes (1000s) and training examples (100000s). **Alternatives?**

- Uniformly at random (with replacements), sample subsets $\mathcal{D}_s \subseteq \mathcal{D}$ of the training data, $\Phi_s \subseteq \Phi$ of the attribute set and construct decision tree T_s for each such random subset.
- Random Forest Algorithm: For $s = 1$ to B repeat:
  1. **Bagging:** Draw a bootstrap sample $\mathcal{D}_s$ of size m_s from the training data $\mathcal{D}$ of size m
  2. Grow a random decision tree T_s to $\mathcal{D}_s$ by recursively repeating steps (1) - (5) of decision tree construction algorithm,, with following difference to step (1)
     1. $\phi_i \leftarrow$ **'best" decision attribute for next** *node* **from** Φ_s **where** $\Phi_s \subseteq \Phi$ **is sample of size** n_s
- **Output:** Ensemble of Trees $\{T_s\}_1^B$

Random Forest applied to Query (Test) data

Output of Random forest Algorithm: Ensemble of Trees $\{T_s\}_1^B$

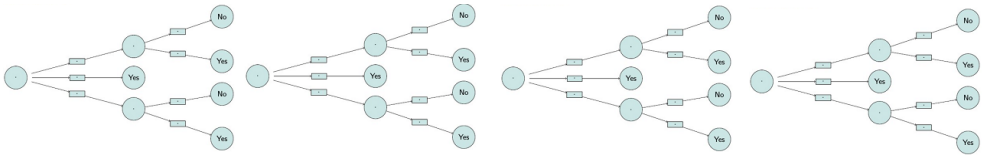


- Consider $\Pr_t(c \mid \mathbf{x})$ for each each tree $t \in T$ for each class $c = [1..K]$ based on the proportion of training points in class c of the leaf node determined by the path of query point $\mathbf{x}$ on tree t
- Decision for a new test point $\mathbf{x}$:

[4] Brieman et. al. http://www.jmlr.org/papers/volume9/biau08a/biau08a.pdf and https://www.microsoft.com/en-us/research/publication/decision-forests-a-unified-framework-for-classification-regression-density-estimation-manifol for several other results on random forests

Random Forest applied to Query (Test) data

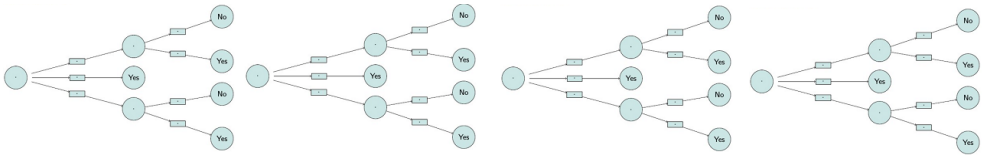
Output of Random forest Algorithm: Ensemble of Trees $\{T_s\}_1^B$



- Consider $\Pr_t(c \mid \mathbf{x})$ for each each tree $t \in T$ for each class $c = [1..K]$ based on the proportion of training points in class c of the leaf node determined by the path of query point $\mathbf{x}$ on tree t
- Decision for a new test point $\mathbf{x}$: $\Pr(c \mid \mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} \Pr_t(c \mid \mathbf{x})$
- For m data points, with $|T| = \sqrt{m}$, consistency results have been proved[4]

[4] Brieman et. al. http://www.jmlr.org/papers/volume9/biau08a/biau08a.pdf and https://www.microsoft.com/en-us/research/publication/decision-forests-a-unified-framework-for-classification-regression-density-estimation-manifol for several other results on random forests

Random Forest: Balancing Bias and Variance



- Decision for a new test point $\mathbf{x}$: $\Pr(c \mid \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \Pr_t(c \mid \mathbf{x})$
- Each single decision tree, viewd as an estimator of the *ideal* tree has high variance, with very less bias (assumptions)
- But since the decision trees T_i and T_j are uncorrelated, when decision is averaged out across them, it tends to be very accurate.

Extra Reading: Bias Variance Trade-off

Instructor: Prof. Ganesh Ramakrishnan

Bias and Variance

- Bias and Variance are two important properties of a machine learning model.
- They help us measure the accuracy of the model and the dependence between the trained model and the training data set. (Q: Is greater dependence good?)
- **Variance** of a model is the variance in the prediction of the models trained over different training data. (Is high variance good?)
- **Bias** of a model is the difference between the expected prediction of the model and the true values which we are trying to predict. (Is low bias good?)
- In this lecture we will talk about the trade-off between the two.

Bias and Variance

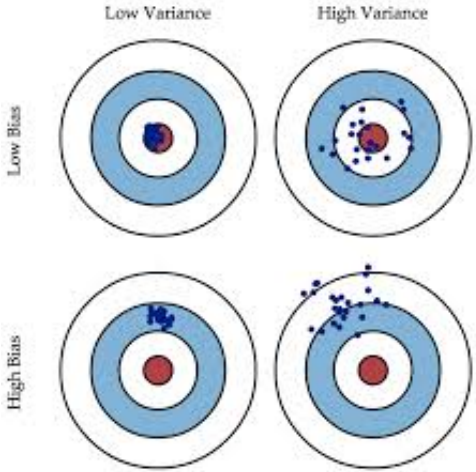


Figure: The distance of the cluster from the eye represents bias and the spread of the cluster represents variance.

Expected loss of a model

- Say, we are given the training data T_D containing values for x and the target variable is y. $P(x, y)$ is the joint distribution over x and y. $f(x)$ is our target function (as this function will be dependent on T_D as well it is more appropriate to call it $f(x, T_D)$).
- To find the expected loss of the model over the distribution of the training data, we first simplify the expected loss expression. For square loss we get,

$$E_{P(x,y)}[(f(x) - y)^2] = \int_x \int_y (f(x) - y)^2 P(x, y) dxdy$$

$$
\begin{aligned}
&E_{P(x,y)}[(f(x)-y)^2] \\
&= \int_x \int_y (f(x)-y)^2 P(x,y)\,dxdy \\
&= \int_x \int_y (f(x)-E(y/x)+E(y/x)-y)^2 P(x,y)\,dxdy \\
&= \int_x \int_y (f(x)-E(y/x))^2 P(x,y)\,dxdy + \int_x \int_y (E(y/x)-y)^2 P(x,y)\,dxdy \\
&\quad + 2\int_x \int_y (f(x)-E(y/x))(E(y/x)-y)P(x,y)\,dxdy
\end{aligned}
$$

We will rewrite the 3rd term in the final equation as:

$$
\begin{aligned}
&2\int_x \int_y (f(x)-E(y/x))(E(y/x)-y)P(x,y)\,dxdy \\
&= 2\int_x (f(x)-E(y/x))(\int_y (E(y/x)-y)P(y|x)\,dy)P(x)\,dx
\end{aligned}
$$

By definition $\int_y yP(y|x)\,dy = E(y/x)$. Therefore the inner integral is 0.

Finally we get,
$E_{P(x,y)}[(f(x) - y)^2] = \int_x \int_y (f(x) - E(y/x))^2 P(x,y) dxdy + \int_x \int_y (E(y/x) - y)^2 P(x,y) dxdy$

The 2nd term is independent of f. Can you think of a situation when the 2nd term will be 0?
Q: For what value of f will this loss be minimized?

The minimum loss will be achieved when $f(x) = E(y/x)$
Now let us find the expected loss over the training data. Using our previous analysis we see that only the $(f(x) - E(y/x))^2$ component can be minimized. (Remember f is dependent on T_D)
(Simple Q: Why is integrating over T_D and (x, y) the same)

$\int_{T_D}(f(x, T_D) - E(y/x))^2 P(T_D) dT_D$
$= E_{T_D}[(f(x, T_D) - E_{T_D}[f(x, TD)] + E_{T_D}[f(x, TD)] - E(y/x))^2]$
$= E_{T_D}[(f(x, T_D) - E_{T_D}[f(x, TD)])^2 + (E_{T_D}[f(x, TD)] - E(y/x))^2$
$-2(E_{T_D}[f(x, TD)] - E(y/x))(f(x, T_D) - E_{T_D}[f(x, TD)])]$

The last term vanishes (WHY?) and we get:
$E_{T_D}[(f(x, T_D) - E_{T_D}[f(x, TD)])^2] + (E_{T_D}[f(x, TD)] - E(y/x))^2$

Bias and Variance

$E_{T_D}[(f(x, T_D) - E_{T_D}[f(x, TD)])^2] + (E_{T_D}[f(x, TD)] - E(y/x))^2$
$= Variance + Bias^2$
Finally we say the expected loss of the model is:
$Variance + Bias^2 + Noise$

The noise in the measurement can cause errors in prediction. That is depicted by the third term.

Interpret with example - Linear Regression

If we were to take the linear regression with a low degree polynomial, we are introducing a bias that the dependency of the predicted variable is simple.
Similarly when we add a regularizer term, we are implicitly biased towards weights that are not big.
By being biased towards a smaller class of models the predicted values will have smaller variation when trained over different samples (Low Variance) and may fit poorly as compared to a complex model (High Bias).
The low variance makes model generalizable over the samples.

Interpret with example - Linear Regression

Suppose we complicate our regression model by increasing degree of the polynomial used.
As we have seen before this will lead to complex curves and will tend to pass through all points.
Here we have put fewer restrictions on our model and hence have less bias.
For a given training data our prediction could be very good (Low Bias).
Although if we consider different Training Sets are models could vary wildly (High Variance).
This reduces the generalizability of the model.

Conclusion

This is the Bias-Variance Tradeoff in action. Simple models usually have low variance but high bias and complex models usually have high variance and low bias.
Food for Thought: So how should we choose our model?
Also whenever you learn about a new algorithm it would be a good exercise to see how the tradeoff works there.
For example, think how the tradeoff manifests itself in the K Nearest Neighbor algorithm.