Started on	Saturday, 27 July 2024, 10:25 PM
State	Finished
Completed on	Saturday, 3 August 2024, 2:30 PM
Time taken	6 days 16 hours
Marks	43/43
Grade	**10** out of 10 (**100**%)

Question 1
Correct
Mark 1 out of 1

Decision trees are machine learnable models that learn a mapping from an input space to an output space based on a finite amount of training samples exemplifying this mapping. Concretely, a decision tree model is

- ☐ a. a special kind of deep neural network
- ☑ b. defining an axis-aligned decision boundary
- ☐ c. restricted to categorical output features
- ☐ d. restricted to categorical inputs
- ☑ e. a directed tree of conditions as nodes and discrete outputs as leaves
- ☑ f. a partition of the input space

Your answer is correct.

Correct
Marks for this submission: 1/1.

Question 2
Correct
Mark 5 out of 5

A decision tree consists of

connection of a superset to its subsets which are obtained via splitting according to a splitting condition	edges
subset of the input space together with a hierarchy of conditions to further break it down	subtree
subsets of the input space that are not further split; these can be associated with an output class label	leaf nodes
subsets of the input space together with a splitting condition that uses one feature	internal nodes
the first splitting on the complete input space, called	root node

Your answer is correct.

Correct
Marks for this submission: 5/5.

Question 3
Correct
Mark 1 out of 1

A decision tree is built by iterative splitting of nodes. What are typical stop criteria for quitting the splitting at a node?

- ☑ a. low number of training samples in the node
- ☐ b. further splitting would exceed a maximum width
- ☐ c. low variance reduction for any possible split
- ☑ d. further splitting would exceed a maximum depth
- ☐ e. low information gain for any possible split
- ☐ f. low Gini impurity of the node

Your answer is correct.

Correct
Marks for this submission: 1/1.

Question 4

Correct

Mark 4 out of 4

To decide whether a proposed split is better than another, decision tree learning algorithms utilize one of several methods to quantify "purity" of a subset with respect to the output classes/values.

Let's first **recap the formulas:**
For output variable Y, feature X, and set to split S, what is the score $C(X;Y)$ for improvement of purity when splitting at X?

- [Gini impurity] : $C(X=x) = 1 - \sum_y P(y|x)^2; C(X;Y) = C(X=any) - \sum_x P(X=x)C(X=x) = -\sum_y P(y)^2 + \sum_{y,x} P(x)P(y \mid x)^2$
- [information gain] : $C(X;Y) = H(Y) - H(Y|X) = -\sum_y P(y)\log(P(y)) + \sum_{y,x} P(x)P(y \mid x)\log(P(y \mid x))$
- [variance reduction] : $C(X;Y) = \frac{1}{|S|^2}\left(\sum_{s,s' \in S} k(y_s, y_{s'}) - \sum_x \sum_{s,s' \in S, x_s = x} k(y_s, y'_s)\right)$

Note: Notice the similarity of the first two! In essence, they both sum some strengthened form of the conditional probabilities; one $P(y \mid x)P(y \mid x)$, the other $P(y \mid x)log(P(y \mid x))$. Thus, there is not much of a difference to be expected in the outcomes when using either of the two.

Now we can have a closer look at **their properties**.
Which of the following holds true about the purity measures that we have seen in the lecture?

- [none] works both for regression and classification tasks; [variance reduction] works for regression tasks, and the rest for classification.
- [Gini impurity] measures the probability of correctly guessing the class of a sample drawn from the split.
- [variance reduction] measures how much the training sample output values within each split deviate from one another.
- [information gain] measures how far the respective distributions are from being random.
- Entropy lies at the heart of [information gain] : It measures the change in entropy when splitting.
- [information gain] was the first used metric, while [Gini impurity] was first introduced in CART for classification trees.

[information gain] [variance reduction] [none] [Gini impurity]

Your answer is correct.

Correct

Marks for this submission: 4/4.

Question 5

Correct

Mark 4 out of 4

Let's recapitulate the decision tree learning algorithm step by step, and start with the overview here.

In the following pseudo-code, insert the available comments at the correct position which describe the rational of the respective steps.

```
Given: input space S, training samples t, validation samples v, features X, output feature Y
Goal: tree consisting of
* inner nodes Node(feature, input space region),
* leaves Leaf(input space region), and
* connecting edges represented by tests Test("point[{some feature}] </>=/== {some value of feature}"),
where each node defines a subtree.

# start at the complete input space
tree <- Leaf(S)
# 1. [iteratively split each leaf at the optimal split if permissible]
FOR leaf in tree.leaves:
    subtree <- split_leaf(leaf, t, X, Y)
    replace leaf in tree by subtree
    tree.leaves.append( leaves in splitting_feature.split)

# 2. [define the tree inference mapping]
FOR leaf in tree.leaves:
    # [assign each leaf the majority vote / average of its training points]
    leaf.output <- determine_output(leaf.samples)

# 3. [make sure to get rid of unnecessary splits that overfitted the training data]
all_subtrees <- sort(nodes in tree)  # go through subtrees bottom-up or top-down
FOR subtree in subtrees:
    replace subtree in tree by prune_subtree(subtree, v)
```

Your answer is correct.

Correct

Marks for this submission: 4/4.

Question 6

Correct

Mark 4 out of 4

Let's take a closer look at the iterative splitting of the tree nodes. Fill in the comments at the right position.

```
FUNCTION split_leaf(Leaf leaf, TrainingData t, InputFeatures X, OutputFeature Y, Bool is_part_of_random_forest = true):
    # 1. check stop criterion: depth or minimum number of samples
    IF maximum depth reached or |leaf| < minimum samples:
        return leaf  # no splitting

    X' <- X
    # 2. sample features for creating a random forest member
    IF is_part_of_random_forest:
        X' <- randomly sample from features X

    # 3. for each feature F determine the subtree and its quality that would be defined by splitting at F
    leaf.samples <- points in t that lie in leaf.subset # if not already defined
    FOR feature F in X':
        F.split <- determine_subtree_for_feature(leaf, F)
        # calculate the improvement in purity, e.g., information gain, variance reduction, for the candidate split
        F.improvement <- calculate purity improvement on training set Y-values if applying F.split

    # 4. pick optimal splitting feature
    splitting_feature <- argmax_F F.improvement
    RETURN splitting_feature
END FUNCTION
```

```
FUNCTION determine_subtree_for_feature(Leaf leaf, feature F):
    # in the categorical case: set the subnodes to the splits defined by the categorical feature values
    IF F categorical:
        edges = [ Test("point[{F}] == {value}") for value in F.values ]
        split <- Node(F, leaf.samples) connected via edge to Leaf({p in leaf | p passes edge.test}) for edge in edges

    # in the numerical case: find optimal split point for binary split
    ELSEIF F numerical:
        # 1. determine possible splitting points as splitting between training samples
        F.possible_splitting_points <- F-values halfway between points

        # optional: prune splitting points
        F.possible_splitting_points <- splitting points pruned to define more consistent intervals

        # 2. determine the subtree each splitting point would define
        FOR split_point in F.possible_splitting_points:
            edges = [ Test("point[{F}] <= {split_point}"), Test("point[{F}] > {split_point}") ]
            F.possible_splits.append( Node(F) connected via edge to Leaf({p in leaf | p passes edge.test}) for edge in edges )

        # select best splitting point
        split <- split subtree in F.possible_splits with maximum quality gain

    RETURN split
END FUNCTION
```

```
FUNCTION determine_output(TrainingData training_data_subset):
    IF Y categorical:
        output <- most common Y-value of points in training_data_subset
    IF Y numerical:
        output <- average Y-value of points in training_data_subset
    return output
END FUNCTION
```

Your answer is correct.

Correct

Marks for this submission: 4/4.

Question 7

Correct

Mark 4 out of 4

Lastly, fill in the comments that describe the steps for finally pruning the tree. The following function is the one that returns for a given subtree its pruned version (i.e., the original subtree, if no pruning should be applied, or else its pruned version).

```
FUNCTION prune_subtree(DecisionTree subtree, ValidationSet v):
    root <- root node of subtree

    # 1. determine the output the node would be assigned to if made a leaf
    root.v_samples <- samples in v that lie in the subspace defined by this root node
    root.output <- determine_output(root.v_samples)
    mapping_without_splitting <- function g on root.v_samples: g(point) = root.output
    mapping_with_splitting <- function g' on root.v_samples: g(point) = leaf.output for leaf in subtree with point in leaf's subspace

    # 2. calculate performance indicators of the subtree's mapping with and without further splitting; e.g., distribution or error rate
    performance_without_splitting <- performance indicator on root.v_samples of mapping_without_splitting
    performance_with_splitting <- performance indicator on root.v_samples of mapping_with_splitting

    # 3. if the change introduced by the splitting is too small, prune
    IF performance_with_splitting similar to performance_without_splitting:
        RETURN Leaf(root.set)
    ELSE:
        RETURN subtree

END FUNCTION
```

Your answer is correct.

Correct

Marks for this submission: 4/4.

Question 8

Correct

Mark 4 out of 4

Assume we are building a decision tree for binary classification into $\mathrm{blue}, \mathrm{pink}$. We have reached a decision tree node which only holds the sample set S of size $K = |S|$, and we would like to break it further down. As a candidate for splitting we consider the X feature, splitting at the split point x to get a left side (K_r samples in S, P_r of which are blue) and a ride side (K_l samples in S, P_l of which are blue) subset.

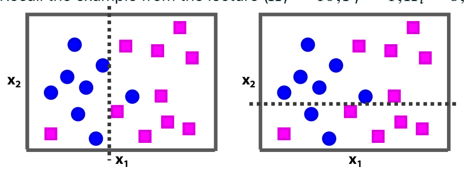
Recap of the formulas

Let's calculate the decrease in impurity with respect to different purity measurements. For that, recall the formulas for this simple 2 input, 2 output class case:

- Information Gain [bits]:
  $$I(X;Y) = H(Y) - H(Y \mid X) = -\left(\frac{P_r+P_l}{K}\log_2\left(\frac{P_r+P_l}{K}\right) + \left(1-\frac{P_r+P_l}{K}\right)\log_2\left(1-\frac{P_r+P_l}{K}\right)\right) + \left(\left(\frac{K_r}{K}\frac{P_r}{K_r}\log_2\left(\frac{P_r}{K_r}\right)\frac{K_r}{K}\left(1-\frac{P_r}{K_r}\right)\log_2\left(1-\frac{P_r}{K_r}\right)\right) + \left(\frac{K_l}{K}\frac{P_l}{K_l}\log_2\left(\frac{P_l}{K_l}\right)\frac{K_l}{K}\left(1-\frac{P_l}{K_l}\right)\log_2\left(1-\frac{P_l}{K_l}\right)\right)\right)$$
- Gini impurity decreases as
  $$\mathrm{Gini}(X = any) - (P(X \leq x)\mathrm{Gini}(X \leq x) + P(X \geq x)\mathrm{Gini}(X \geq x)) = \left(1 - \left(\left(\frac{P_r+P_l}{K}\right)^2 + \left(1-\frac{P_r+P_l}{K}\right)^2\right)\right) - \left(\frac{K_r}{K}\left(1-\left(\left(\frac{P_r}{K_r}\right)^2 + \left(1-\frac{P_r}{K_r}\right)^2\right)\right) + \frac{K_l}{K}\left(1-\left(\left(\frac{P_l}{K_l}\right)^2 + \left(1-\frac{P_l}{K_l}\right)^2\right)\right)\right)$$

A concrete example

Recall the example from the lecture ($K_r = 10, P_r = 1, K_l = 8, P_l = 7$):



Let's manually go through the example using the formula for information gain in order to get a better gut feeling about their meaning.

- $H(Y) = -(P(\mathrm{blue})\log_2(P(\mathrm{blue})) + P(\mathrm{pink})\log_2(P(\mathrm{pink}))) = -(8/18)\log_2(8/18) - (10/18)\log_2(10/18) \approx 0.991$
- Split at $X_1 = x_1$ obtaining the new binary variable $X_1' := (X_1 \geq x_1)$ (write $(X_1 \leq x_1) \Leftrightarrow$: left, $(X_1 > x_1) \Leftrightarrow$: right):
  - $H(Y \mid X_1') = -P(\mathrm{right})\big(P(\mathrm{blue} \mid \mathrm{right})\log_2(P(\mathrm{blue} \mid \mathrm{right})) + P(\mathrm{pink} \mid \mathrm{right})\log_2(P(\mathrm{pink} \mid \mathrm{right}))\big) - P(\mathrm{left})\big(P(\mathrm{blue} \mid \mathrm{left})\log_2(P(\mathrm{blue} \mid \mathrm{left})) + P(\mathrm{pink} \mid \mathrm{left})\log_2(P$

    0.511
  - $I(X_1'; Y) =$ 0.488
- Split at $X_2 = x_2$ obtaining the new binary variable $X_2' := (X_2 \geq x_2)$ (write $(X_2 \leq x_2) \Leftrightarrow$: bottom, $(X_2 > x_2) \Leftrightarrow$: top):
  - $H(Y \mid X_2') = -P(\mathrm{bottom})(P(\mathrm{blue} \mid \mathrm{bottom})\log_2(P(\mathrm{blue} \mid \mathrm{bottom})) + P(\mathrm{pink} \mid \mathrm{bottom})\log_2(P(\mathrm{pink} \mid \mathrm{bottom}))) - P(\mathrm{top})(P(\mathrm{blue} \mid \mathrm{top})\log_2(P(\mathrm{blue} \mid \mathrm{top})) + P(\mathrm{pink} \mid \mathrm{top}$

    0.94
  - $I(X_2'; Y) =$ 0.049

This comparison clearly shows that splitting at feature X_1 is preferrable, which is coherent with the visualization.

Correct

Marks for this submission: 4/4.

Question **9**

Correct

Mark 4 out of 4

In the lecture we have visited bagging and boosting as means to build ensembles that reduce variance and bias compared to their single members. While the result of both is an ensemble, it is important to differentiate between them, because the follow quite different ideas and have different effects on the bias-variance tradeoff. Which of the following properties holds for which of the methods?

bagging	boosting	
●	○	Learning of ensemble members can be done in parallel.
●	○	Aims to decrease variance.
○	●	Aims to decrease bias.
○	●	Can be formulated to weight each training data point according to whether it is erroneously treated. Utilizes the complete data set (but weighted) for training each ensemble member.
○	●	Tries to circumvent instability of the learning method by active search for complementary solutions.
●	○	A special application to decision trees is random forests.
○	●	Learning of ensemble members is done sequentially.
●	○	Utilizes bootstrapping, i.e., resampling from a sample to obtain new sample sets. Every ensemble member is trained on a different sample set.
○	●	Builds a strong learner as an ensemble of weak learners.
●	○	Captures instability of the learning method by randomization.

Correct

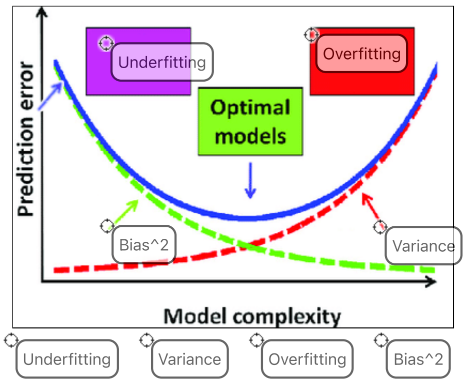
Marks for this submission: 4/4.

Question **10**

Correct

Mark 4 out of 4

The bias, also the training error, and variance together make up the final testing error as depicted below. Correctly insert the right labels.



Your answer is correct.

Correct

Marks for this submission: 4/4.

Question 11

Correct

Mark 8 out of 8

Recap: Why is the bias-variance tradeoff really important?

Recall that one of the main learning goals in this semester is to be able to do informed model selection for your problem. For supervised models, this culminates in the bias-variance tradoff: The bias vs. variance tradeoff is one of the most important factors that should be taken into account for selecting the machine learning model to apply. Training a too complex/flexible model on training data for a simple model will almost for sure lead to overfitting, i.e., a high variance between solutions for different training sets; and on the other hand, selecting a too simple model for a complex problem cannot lead to a trustworthy solution.

Recap: How to influence bias and variance?

We have seen several ways how one can directly influence the **bias**:

- **By the choice of model**: fewer parameters respectively less *flexibility,* as we called it, means less bias. In particular, different models assume different shapes of problems that they work best for: Try to find the matching one.
- By adding further constraints to the model like regularization
- By accepting remaining model errors and simply fix it with error-fixing models, building an ensemble, which is boosting.

And for a large bias, the **variance** can only be reduced by

- decreasing the bias,
- random ensembling such as bagging, or
- by drastically increasing the amount of training samples.

Recap: Know your bias and variance!

To finalize the remark: Always make sure that you measure your bias and variance by applying testing on a separate test dataset to measure the generalization error, and cross-validation! Cross-validation will give you a glimpse on the variance of your model, while the generalization error for the average model gives you (by definition) an estimation for the bias.

Now let's finally come to the actual exercise: Try yourself on a couple of examples.

Rather high bias	Rather low bias		
●	○	Regression: Polynomial regression $f(x) = \theta_2 x^2 + \theta_1 x + \theta_0$	
●	○	Classification: k-NN with $k = 18$	
●	○	Regression: Linear regression model $f(x) = \theta^T x + b$	
○	●	Classification: k-NN with \(k=3\)	
○	●	Classification: Decision tree of depth 8	
○	●	Natural language processing: Llama 2 70B language model	This *definitely* has a minuscule bias, and variance is (again not measured and) only controlled by the immense amount of training data!
●	○	Classification: Quadratic discriminant analysis	
●	○	Regression/classification: Deep neural network with 1 hidden layer and 5 neurons.	
○	●	Regression: Polynomial regression \(f(x) = \sum_{i=0}^7 \theta_i x^i \)	
●	○	Image classification: AlexNet convolutional neural network for image classification on ImageNet classes	While AlexNet already features quite a couple of parameters, it still must be considered small and biased in the context of complex tasks like the classification of 1000 image classes.
●	○	Classification: Naive Bayes classifier	Note that this even has a higher bias than logistic regression, as it adds the conditional independence assumption on the inputs.
●	○	Classification: Logistic regression	
○	●	Object detection: YOLOv8 nano	Note that the creators of the YOLOv8 models do not provide variance measures; this is most probably due to the high training effort and the availability of sufficient data for proper cross-validation, not because the models have a low variance!

Correct

Marks for this submission: 8/8.

◀ 08. Quiz - Version Space Learning

Jump to...

10. Quiz - Unsupervised Clustering ▶