Problem 1: Intelligent agents, rationality and descriptions of the environment.

(a) (ref: CHATGPT)
(i) The vacuum-cleaner agent function described in Figure 2.3 is rational because it selects actions based on the current percept, which includes the current location of the agent and the state of the environment. The function uses a simple rule to decide which direction to move in order to clean the most amount of dirty squares. This rule is based on the assumption that the environment is static and does not change while the agent is making its decisions.

(ii) If each movement of the cleaner generates a unit cost, then a rational agent needs to maintain an internal state to keep track of its current location and the state of the environment. This is because the agent needs to minimize its cost while maximizing its cleaning performance. By keeping track of its current location and the state of the environment, the agent can make informed decisions about which direction to move in and when to clean a square.

(iii) To modify the rational agent for the case where there is a naughty pet dog in the environment, we can add a new percept to the agent's current percept that indicates whether the current square has been dirtied by the dog or not. We can also modify the fun ction to take into account the probability that each clean square has a 50% chance of becoming dirty at each time step. One way to do this is to assign a higher priority to cleaning squares that have not been dirtied by the dog since these squares are more likely to remain clean for a longer period of time. Another approach would be to incorporate a probabilistic model of the environment and use Bayesian inference to update the agent's beliefs about the state of the environment based on the observed percepts.

(b) Give a PEAS description of the task environment and characterize it in terms of the properties listed in textbook Section 2.3.2 -

1. **Playing soccer as a robot -**

   PEAS description:

   - Performance measure: Winning the soccer game by scoring more goals than the opponent team.
   - Environment: Soccer field, ball, opponent team players, teammates, referee, weather conditions, etc.
   - Actuators: Motors, sensors, camera, microphone, etc.
   - Sensors: Cameras, microphones, GPS, etc.

   Properties:

(a) **Stochastic:** It's stochastic as while playing soccer the next state of the game cannot be accurately determined by the agent. There are many possible outcomes. Sequential: It's sequential because the action agent takes now will shape the future events of the game. In soccer to score or save a goal, the agent has to perform a sequence of actions which depend on one another.

(b) **Dynamic:** It's dynamic because the environment keeps changing. The opponent and ball positions are continuously changing and the agent has no say in it. Continuous: It's continuous because the agent has to decide parameters like force required to hit the ball, direction of the throw,speed with which it has to run are all continuous. There are infinite possibilities for scoring, defending and passing the ball to a player. All these attributes are continuous variables to be decided by the agent.

(c) **Partially, Observable:** It's again depends on our assumptions. With no assumptions about what sort of camera is being used to monitor the game, it is partially observable because the agent can't observe the whole environment at once. The sensors of the agent can not surely decide what the opponent is thinking and what its next step is, the agent can't sense the all the changes in the environment. It could also be fully observable if there is a camera mounted with the god's view that views the movements of all robot soccer players on the field.

(d) **Multi-Agent:** It's multiagent because the game of soccer can't be played by a single agent. There has to be 2 teams.

2. **Internet laptop-shopping agent.** (an intelligent agent that helps a customer to shop for a laptop on a website.)

PEAS description:

- Performance measure: Maximizing the customer's satisfaction with the purchased laptop within the budget.
- Environment: Laptop seller's website, laptop models, customer's budget, customer's preferences, etc.
- Actuators: Mouse, keyboard, browser, interact with the website by clicking links, filling out forms, and submitting searches, web browsing software, search engines, and decision-making algorithms for finding and recommending laptops to the customer.
- Sensors: Website interface, user input, read the website's HTML code, customer reviews and ratings

Properties:

- **Partially observable**: as the agent can only observe information available on the website and customer feedback.

- **Stochastic**: as the agent cannot predict with certainty how the customer will react to different laptops and information. Also, because laptop products and reviews can change over time.
- **Sequential**: as the agent must take into account the customer's preferences and constraints over multiple steps in the shopping process.
- **Episodic**: Usually each customer's laptop shopping session can be considered an episode with a clear start and end. But this can change depending on our assumptions. So it is important to be aware of what assumptions are we working with to decide on this property.
- **Multi-agent**: the agent is the intelligent entity involved in the task, it may engage a human to ask its a preference or a customer service associate to ask for some help. Depends.

Problem 2. Classifiers and Machine Learning.

(a) (Ref: ChatGPT)

P,E,A,S descriptions of a classifier agent -

1. Performance Measure: The classifier agent's performance measure is the accuracy of its predictions.
2. Environment: A stream of new examples for the classifier to classify.
3. Actuators: The classifier agent's actuator is to classify, i.e., predict the class labels.
4. Sensors: The sensors of the classifier agent are the input features of the data points that it is classifying.

(b) (ref: ChatGPT)

Modelling: Two components: 1. Feature engineering 2. specifying the hypothesis class — which classifier family to use,

Learning: Selecting a classifier from the family that works well. This can be done using brute-force minimization of the training error, gradient descent or SGD...

Inference: This component involves using the trained model to predict the output for new input data. The inference process can be done in real-time, and it is the main function of the classifier agent.

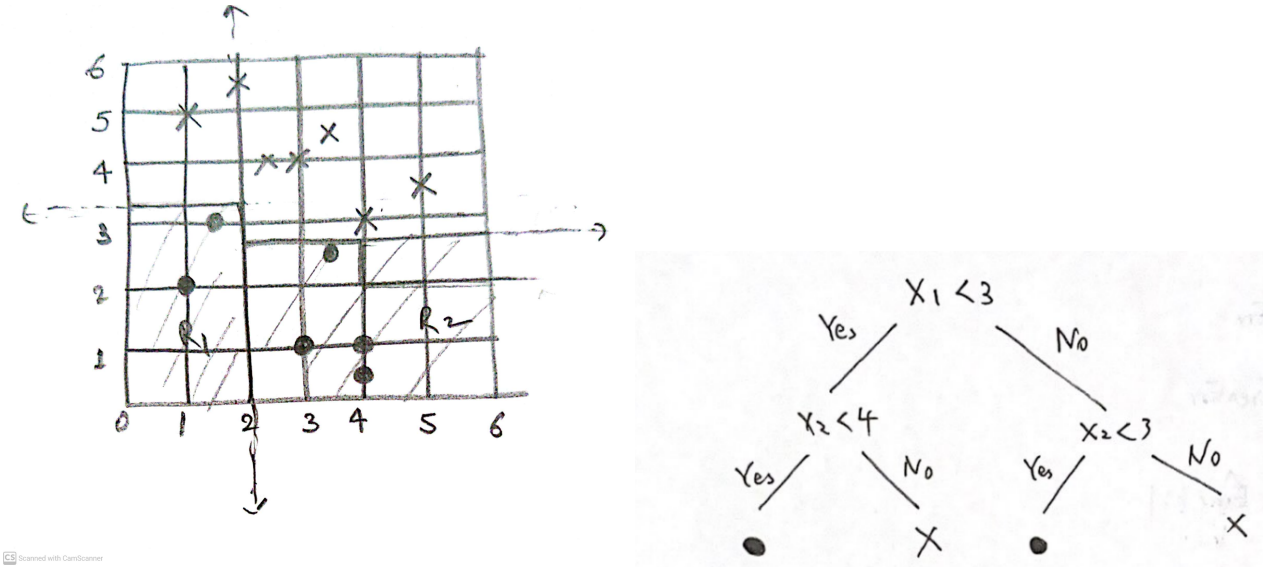

Figure 1: Decision boundary for depth 2 decision tree classifier.

(c) (i) For a decision tree of depth=2, the possible boundaries representable by this classifier on a 2D plane use a line at which a classifier changes its prediction from one class to another. Its decision boundaries divide the input space into axis-parallel rectangles and label each rectangle with one of the K classes.

(ii) Yes, there is a depth 2 decision tree that gives perfect classification in the example below.

(d) The decision boundary of a 1-nearest neighbor classifier in the example above is piecewise linear (see Figure 2 for the sketch). Each piece is a hyperplane that is perpendicular to the bisector of pairs of points from different classes. The boundaries are concatenated segments of the Voronoi tessellation [reference 1]. Each of the 1-NN boundaries generally should have sharp edges and corners as we are drawing bisectors between any two neighbours. [reference 2]

Problem 3: Holdout and K-Fold Cross Validation.

(solutions to a,b,c, and d are from HW1 commented block)

- (a) **Holdout Method**: Divide the dataset D into two disjoint sets: a training set D_{train} and a validation set D_{val}. Explain the purpose of these two sets and their roles in training and validating a classifier.

  Solution:

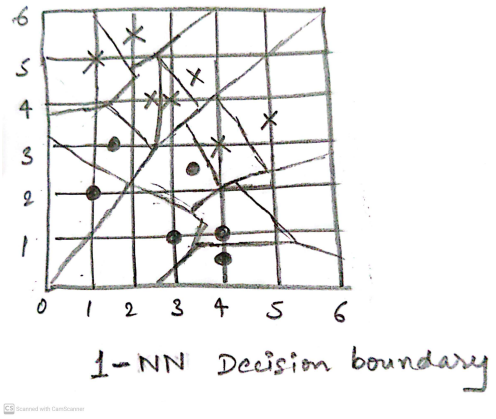

Figure 2: Decision boundary for 1-nearest neighbor classifier.

In the holdout method, the dataset D is divided into two disjoint sets: a training set D_{train} and a validation set D_{val}. The purpose of this division is to train the classifier on D_{train} and evaluate its performance on unseen data, which is represented by D_{val}. By evaluating the classifier on data it has not been trained on, we can get a more accurate estimate of its performance on new data.

- (b) **K-Fold Cross Validation**: Explain the concept of k-fold cross-validation, including the process of creating k partitions of the dataset, and how it helps in evaluating the performance of a classifier.

  Solution:

  In k-fold cross-validation, the dataset D is divided into k equally-sized partitions or "folds". The classifier is then trained and evaluated k times, with each iteration using a different fold as the validation set and the remaining $k-1$ folds as the training set. The performance of the classifier is averaged over the k iterations to provide a more robust estimate of its performance on unseen data.

  This method helps in evaluating the performance of a classifier by utilizing the entire dataset for both training and validation while still maintaining a separation between the two. By averaging the performance over multiple iterations, k-fold cross-validation reduces the variance in the performance estimate and provides a more reliable measure of the classifier's ability to generalize to new data.

- (c) **Comparison**: Compare the holdout method and k-fold cross validation in terms of their advantages and disadvantages. Discuss the situations in which you would prefer one method over the other.

  Solution:

  Holdout Method:

- Advantages: Simpler and faster to implement, requires only one training and validation process, suitable for very large datasets.
- Disadvantages: Performance estimates may have high variance depending on the random split, and may not utilize the entire dataset effectively.

k-Fold Cross Validation:

- Advantages: Provides a more robust performance estimate, utilizes the entire dataset for both training and validation and reduces the impact of random splits on the performance estimate.
- Disadvantages: Computationally more expensive, requires k training and validation processes, may not be suitable for very large datasets due to computational constraints.

In general, k-fold cross-validation is preferred when the dataset size is not too large and a more robust performance estimate is desired. The holdout method may be preferred for very large datasets or when computational resources are limited.

- (d) **Understanding**: Explain why it is not appropriate to use the training dataset to evaluate the performance of a classifier. Discuss the concept of overfitting and its relationship to evaluating classifiers.

  Solution:

  Using the training dataset to evaluate the performance of a classifier is not appropriate because it can lead to an overly optimistic estimate of the classifier's performance. During the training process, the classifier learns to minimize the error on the training dataset, and as a result, it may fit the noise in the training data. This phenomenon is known as overfitting.

  Overfitting occurs when a classifier learns the noise in the training data rather than the underlying patterns, which reduces its ability to generalize to new, unseen data. When a classifier overfits the training data, it will perform well on the training data but poorly on unseen data. Evaluating the classifier on the training dataset does not provide a reliable estimate of its performance on new data, as the classifier's performance will be inflated due to overfitting.

  To obtain a more accurate estimate of the classifier's performance on new data, it is necessary to evaluate the classifier on a separate dataset that it has not been trained on, such as a validation set. This is why methods like holdout and k-fold cross-validation are used to evaluate the performance of classifiers.

Problem 4. (More on training error, test error, Hold-out data, and cross-validation)

(a) The basic question is: what does the equation say? The above equation states that the expected error on the learned classifier is bounded by the expected error on the optimal classifier and twice the generalization error. In other words, the equation says that the expected error of a learned model is no worse than the expected error of the best possible model plus some additional error that arises from the model's inability to perfectly generalize to new data.

if $\hat{h}$ be a learned classifier and h^* be the optimal classifier, then generalization error is GenErr $= \max_{h \in \mathbb{H}} |Err(h) - \hat{Err}(h)|$

$$Err(\hat{h}) - Err(h^*) = Err(\hat{h}) - Err(h^*) + \hat{Err}(\hat{h}) - \hat{Err}(\hat{h})$$
$$Err(\hat{h}) - Err(h^*) = (Err(\hat{h}) - \hat{Err}(\hat{h})) - (Err(h^*) - \hat{Err}(\hat{h}))$$
$$Err(\hat{h}) - Err(h^*) \leq (Err(\hat{h}) - \hat{Err}(\hat{h})) + (\hat{Err}(h^*) - Err(h^*))$$
$$Err(\hat{h}) - Err(h^*) \leq |Err(\hat{h}) - \hat{Err}(\hat{h})| + |Err(h^*) - \hat{Err}(h^*)|$$

In the first line, we just add and subtract $\hat{Err}(\hat{h})$. In the second line, we collect the terms. In the third line, From ERM, we know that $\hat{h}$ is our best optimal hypothesis & hence we can replace with it h^* (using the hint in the question). In the fourth line, the inequality can be rewritten in terms of absolute value without loss of generality.

From the definition of generalization error which is applicable to any hypothesis in the hypothesis class $\mathbb{H}$, we can rewrite the last equation/line without any loss of inequality as-

$$Err(\hat{h}) - Err(h^*) \leq 2GenErr$$

Hence, proved.

(Additional note: ERM: Empirical risk minimization (ERM) is a principle in statistical learning theory that defines a family of learning algorithms and is used to give theoretical bounds on their performance.)

(b) What does the equation mean? - The expectation of the empirical error on the holdout validation set is the same as the expected error over the population. In other words, the

holdout estimator of the classifier performance is unbiased.

$$
\begin{aligned}
E[\hat{Err}(\tilde{h}, D_{val})|\tilde{h}] &= E\Big[\frac{1}{|D_{val}|}\Sigma_{j=1}^{|D_{val}|}\mathbb{I}(\tilde{h}(X_j) \neq y_j)\Big|\tilde{h}\Big] \\
&= \frac{1}{|D_{val}|}\Sigma_{j=1}^{|D_{val}|}E\Big[\mathbb{I}(\tilde{h}(X_j) \neq y_j)\Big|\tilde{h}\Big] \\
&= \frac{1}{|D_{val}|}\Sigma_{j=1}^{|D_{val}|}\underset{(X_j,y_j)\sim D}{E}\Big[\mathbb{I}(\tilde{h}(X_j) \neq y_j)\Big|\tilde{h}\Big] \\
&= \underset{(X,y)\sim D}{E}\Big[\mathbb{I}(\tilde{h}(X) \neq y)\Big|\tilde{h}\Big] \\
&= Err(\tilde{h})
\end{aligned}
$$

In the first line, we just substitute the definition of the Error function. The second line is by the linearity of expectation. The third line is due to the fact that $\tilde{h}$ is a function of D_{train} (classifier trained on training dataset) and is independent of D_{val}. The fourth line is due to the i.i.d assumption of the dataset and the fifth line, is by definition of the error function.

A few additional points to notice are that 1) $\tilde{h}$ can arbitrarily depend on the D_{train} (example: Gradient descent, SGD, Empirical risk minimization). 2) There is no need for any assumption on the distribution of data for what we need for this proof is that $(x_i, y_i \sim D)$ (where D-distribution) for $(x_i, y_i) \in D_{val}$

(c) The error estimates we get from each of the K-fold in the cross-validation setup are not independent of each other. Each fold k uses $(K-1)/K$ fraction of the data as training data and the remaining fraction as validation data. Thus, the validation sets overlap, and the error estimates are not independent.

what does this equation mean? - The cross-validation error, i.e., the average over the K-folds, is an unbiased estimator of the expected error rate of a classifier trained on $n(K-1)/K$ data points drawn i.i.d. Notice that the RHS the expectation is over $\tilde{h}_1$.

$$
\begin{aligned}
\mathbb{E}[\frac{1}{K}\sum_{k=1}^{K}E_k] =& \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[E_k] && (1)\\
=& \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\mathbb{E}[E_k|\tilde{h}_k]\right] && (2)\\
=& \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\mathrm{Err}(\tilde{h}_k)] && (3)\\
=& \mathbb{E}[\mathrm{Err}(\tilde{h}_1)], && (4)
\end{aligned}
$$

The first line (eq. 1) uses the linearity of expectation. The second line (eq. 2) uses the so-called "law of total expectation" by first conditioning on $\tilde{h}_k$ (the expectation in the outside averages over $\tilde{h}_k$, the expectation inside averages over the validation data in the kth fold. The third line (eq. 3) is using the result of Part (b) (that the expectation of the empirical error on the holdout validation set is the same as the expected error over the population). The last line (eq. 4) uses the fact that $\tilde{h}_1$ and $\tilde{h}_j$ for $j = 2, 3, ..., K$ has the same distribution due to the iid assumption — therefore they have the same expected value. They have the same distribution because they are all trained on a dataset with $(K-1)n/K$ i.i.d data points.

Therefore, the expected value of the average error estimate overall folds is equal to the expected error of the classifier trained on the first fold.