1. Suppose we are using filtering (the Forward Algorithm) to compute $P(X_T|e_{1:T})$ in a Hidden Markov Model (HMM).

   (a) If we double the number of time steps, how much longer does it take us to run filtering?

   (b) If we double the number of possible states, how much longer does it take us to run filtering?

   (c) Imagine we have computed $P(X_T|e_{1:T})$ and now a new observation at the next time step comes in, e_{T+1}. How long would it take to compute $P(X_{T+1}|e_{1:T+1})$, given that we have already computed $P(X_T|e_{1:T})$?

2. You are doing Naive Bayes, and a particular feature occurs k_+ times among the positive examples, k_- times among the negative examples, and that there are p positive examples and n negative examples. Assume that by positive examples, we mean spam.

   (a) Assume no smoothing. What is $P(+\text{feature}| + \text{spam})$?

   (b) Assume Laplace smoothing. What is $P(-\text{feature}| + \text{spam})$?

3. Consider the propositional logic sentence $(\neg P) \implies (\neg Q)$. Your task is to construct a neuron using the logistic activation function that is equivalent to this sentence. It will have two inputs: the first will be the truth value of P, given as 1 or 0 if its value is true or false respectively, and the second is similarly the truth value of Q. The neuron's output should be less than 0.5 if the value of $(\neg P) \implies (\neg Q)$ is false, and greater than 0.5 if the value of $(\neg P) \implies (\neg Q)$ is true. For example, if given the input $P = Q =$ false, the neuron would receive the input $x = [0.0, 0.0]$ and should output a value greater than 0.5. What are the weights and bias for your neuron?

4. Imagine we have a training set for a neural network, $(x_1, y_1), ..., (x_N, y_N)$. Let w be the learnable parameters of the neural network, and $\text{Loss}(w, x_n, y_n)$ be the loss of the network with parameters w on the n^{th} example. We could train with gradient descent to either minimize the total loss:

$$\text{Loss}_{\text{Total}}(w) = \sum_{n=1}^{N} \text{Loss}(w, x_n, y_n)$$

Or the average loss:

$$\text{Loss}_{\text{average}}(w) = \frac{1}{N} \sum_{n=1}^{N} \text{Loss}(w, x_n, y_n)$$

The gradient descent update rule derived from the loss function $\text{Loss}_{\text{Total}}$ has a learning rate α_{Total}. Imagine you instead derived a gradient descent update rule using the loss function $\text{Loss}_{\text{average}}$, which would have its own learning rate α_{Average}. What must α_{Average} be so that the resulting updates would be identical to those for the original update rule based on $\text{Loss}_{\text{Total}}$? Your answer should look like $\alpha_{\text{Average}} = \cdots$, where you have to fill in $\cdots$. **Show/explain your work.**

5. (a) True/False: Gradient descent is guaranteed to find the minimum of a function. **Explain**

   (b) True/False: Gradient descent is guaranteed to find the minimum of a function, even if it is not differentiable. **Explain**

6. Imagine we have a SAT formula of the form:

$$x_1 \wedge (x_1 \implies x_2) \wedge (x_2 \implies x_3) \wedge (x_3 \implies x_4) \wedge \cdots \wedge (x_{N-1} \implies x_N)$$

Explain why DPLL will find a satisfying assignment to this formula in time polynomial with respect to N.

7. Convert the following statement to first order logic:

For every state in a board game, there is a move which causes the next state to be a winning state.

To get the next state, assume that there is a predicate Next(m,s,s') which is true whenever move m in state s results in the next state being s'.

8. Consider the sentence in propositional logic $(A \wedge B) \implies C$. Your task is to construct a linear classifier which outputs the truth value of $(A \wedge B) \implies C$, when it takes as input the truth values of A, B, and C. Truth values of False correspond to 0.0, and truth values of True correspond to 1.0. Let the input to the neural network be $x = [A, B, C]$, where $x \in \mathbb{R}^3$. For example, if A is true, B is false, and C is true, then $x = [1, 0, 1]$. Assume that all neurons are implemented with Logistic activations, so you can think of them as outputting true whenever the weighted input to the neuron is at least 0.

What are the weights and biases of such a linear classifier?

9. Imagine we have a training set for linear regression, $(x_1, y_1), ..., (x_N, y_N)$, where $x_n \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$. We have trained a weight vector $w \in \mathbb{R}^D$ and a bias $b \in \mathbb{R}$ to approximately fit the training data: $y_n \approx w \cdot x_n + b$. Notice that we are *explicitly* writing out the constant bias, b. Unfortunately the data that we are going to deploy our linear predictor on has been corrupted! Write $(x^{\text{bad}}, y^{\text{bad}})$ for one of these corrupted data points, which came from corrupting $(x^{\text{good}}, y^{\text{good}})$.

   (a) If the data was corrupted by adding the same constant c to each feature in x^{good}, then $x_d^{\text{bad}} = x_d^{\text{good}} + c$, and $y^{\text{bad}} = y^{\text{good}}$. How should we transform w and b so that we still correctly predict y_{good} given x^{bad}?

   (b) If the data was corrupted by multiplying the same constant c to each feature in x^{good}, then $x_d^{\text{bad}} = cx_d^{\text{good}}$, and $y^{\text{bad}} = y^{\text{good}}$. How should we transform w and b so that we still correctly predict y_{good} given x^{bad}?

10. Suppose we are trying to learn to predict $y \in \mathbb{R}^M$ from input $x \in \mathbb{R}^D$. Notice that we are predicting not a single value but M values. One way of doing this is to have M linear predictors. Another way is to have a neural network with H hidden neurons and a single hidden layer. What must H be less than in order for the neural network to have fewer learnable parameters than the M linear predictors?

11. You will be working with the following STRIPS/PDDL planning problem. It is meant to model you in a hallway, as you walk along a 1-dimensional grid world with 10 different locations: locations 0, 1, 2, 3, .., 9. You start at location 1, indicated by the fluent At(You, 1). There is a locked door at location 5, indicated by the fluent LockedDoor(5). There is a key at location 0, indicated by the fluent At(Key, 0). Your goal is to make it to the end of the hallway and be holding the key. The fluent Adjacent(x, y) indicates that the locations x and y are adjacent, and holds whenever $|x - y| = 1$.

    - Objects: You, Key, Door, 0, 1, 2, 3, ..., 9
    - Fluents: At(x, y), Holding(x, y), Adjacent(x, y), LockedDoor(x)
    - Initial state: At(Key, 0), At(You, 1), LockedDoor(5), Adjacent(0, 1), Adjacent(1, 0), Adjacent(1, 2), Adjacent(2, 1), ..., Adjacent(8, 9), Adjacent(9, 8)
    - Goal state: Holding(You, Key), At(You, 9)
    - Actions:
        - Move(x, y)
            * Precondition: At(You, x), Adjacent(x, y), ¬LockedDoor(y)
            * Effect, add: At(You, y)
            * Effect, delete: At(You, x)
        - Unlock(x, y)
            * Precondition: Holding(You, Key), LockedDoor(y), At(You, x), Adjacent(x, y)
            * Effect, add: At(Key, y)
            * Effect, delete: LockedDoor(y), Holding(You, Key)
        - Drop(x)
            * Precondition: Holding(You, Key), At(You, x)
            * Effect, add: At(Key, x)
            * Effect, delete: Holding(You, Key)
        - Pickup(x)
            * Precondition: At(You, x), At(Key, x)
            * Effect, add: Holding(You, Key)
            * Effect, delete: At(Key, x)

    (a) What is plan which achieves the goal in the fewest number of actions?

    (b) The ignore precondition heuristic works, for a given state s, by creating a relaxed planning problem where all the actions have no preconditions, and finding the shortest path to a goal starting from s in this relaxed planning problem. Using the ignore preconditions heuristic, what is the heuristic value of the initial state?