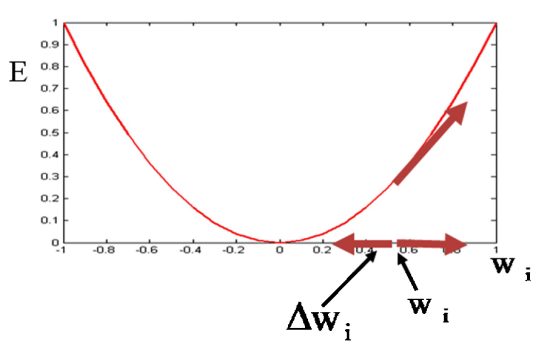
EXAMPLE Machine Learning (C395) Exam Questions

(1) **Question:** Explain the principle of the gradient descent algorithm. Accompany your explanation with a diagram. Explain the use of all the terms and constants that you introduce and comment on the range of values that they can take.

Solution: Training can be posed as an optimization problem, in which the goal is to optimize a function (usually to minimize a cost function E) with respect to a number of free variables, usually weights w_i. The gradient decent algorithm begins from an initialization of the weights (e.g. a random initialization) and in an iterative procedure updates the weights w_i by a quantity Δw_i, where $\Delta w_i = -\alpha (\partial E / \partial w_i)$ and $(\partial E / \partial w_i)$ is the gradient of the cost function with respect to the weights, while α is a constant which takes small values in order to keep the updates low and avoid oscillations.



(2) **Question:** Derive the gradient descent training rule assuming that the target function representation is:

$$o_d = w_0 + w_1 x_1 + \ldots + w_n x_n.$$

Define explicitly the cost/error function E, assuming that a set of training examples D is provided, where each training example $d \in D$ is associated with the target output t_d.

Solution: The error function: $E = \sum_{d \in D} (t_d - o_d)^2$
The gradient decent algorithm: $\Delta w_i = -\alpha (\partial E / \partial w_i)$
First represent $(\partial E / \partial w_i)$ in terms of the unit inputs x_{id}, outputs o_d, and target values t_d:
$(\partial E / \partial w_i) = (\partial \sum_{d \in D} (t_d - o_d)^2) / \partial w_i = \sum_{d \in D} 2(t_d - o_d) (\partial (t_d - o_d) / \partial w_i) =$
$\sum_{d \in D} 2(t_d - o_d) (-\partial o_d / \partial w_i) = -\sum_{d \in D} 2(t_d - o_d) (\partial (w_0 + \ldots + w_i x_{id} + \ldots + w_n x_{nd}) /$
$\partial w_i) = -\sum_{d \in D} 2(t_d - o_d) (x_{id})$
$\Rightarrow \Delta w_i = \alpha \sum_{d \in D} 2(t_d - o_d) x_{id}$

(3) **Question:** Prove that the LMS training rule performs a gradient descent to minimize the cost/error function E defined in (2).

Solution: Given the target function representation
$o_d = w_0 + w_1x_1 + \ldots + w_nx_n$,
LMS training rule is a learning algorithm for choosing the set of weights w_i to best fit the set of training examples $\{< d, t_d >\}$, i.e., to minimize the squared error $E \equiv \sum_{d \in D} (t_d - o_d)^2$.
LMS training rule works as follows:
$(\forall < d, t_d >)$ use the current weights w_i to calculate o_d
$(\forall w_i)\ w_i \leftarrow w_i + \eta(t_d - o_d)x_{id}$ (*)

From (2) $\rightarrow$ $(\partial E / \partial w_i) = -\sum_{d \in D} 2(t_d - o_d)x_{id} \rightarrow -(1/2x_{id})(\partial E / \partial w_i) = (t_d - o_d)$
Substitute this in (*) $\rightarrow$ $(\forall w_i)\ w_i \leftarrow w_i + (\eta/2)(-\partial E / \partial w_i)$
This shows that LMS alters weights in the very same proportion as does the gradient descent algorithm (i.e., $-\partial E / \partial w_i$), proving that LMS performs gradient descent.

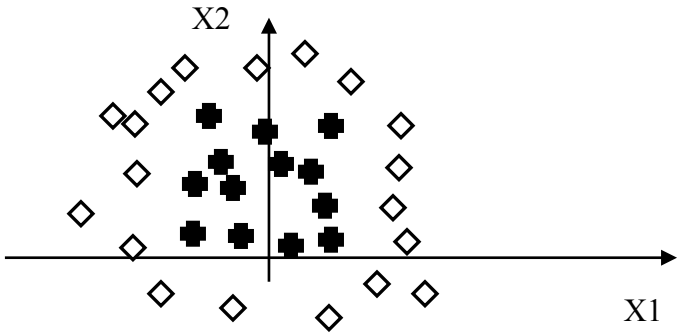
(4) **Question:** Consider the following set of training examples:

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

What is the information gain of a2 relative to these training examples? Provide the equation for calculating the information gain as well as the intermediate results.

Solution:
Entropy $E(S) = E([3+, 3-]) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$.
Gain $(S, a2) = E(S) - (4/6)E(T) - (2/6)E(F) = 1 - 4/6 - 2/6 \approx 0$.
$E(T) = E([2+, 2-]) = 1$.
$E(F) = E([1+, 1-]) = 1$.

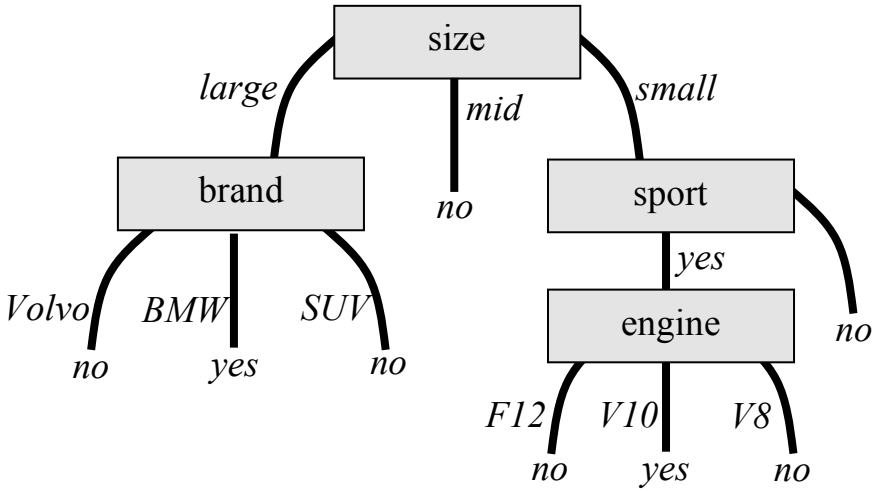
(5) **Question:** Suppose that we want to build a neural network that classifies two dimensional data (i.e., X = [x1, x2]) into two classes: diamonds and crosses. We have a set of training data that is plotted as follows:



Draw a network that can solve this classification problem. Justify your choice of the number of nodes and the architecture. Draw the decision boundary that your network can find on the diagram.

Solution:
A solution is a multilayer FFNN with 2 inputs, one hidden layer with 4 neurons and 1 output layer with 1 neuron. The network should be fully connected, that is there should be connections between all nodes in one layer with all the nodes in the previous (and next) layer. We have to use two inputs because the input data is two dimensional. We use an output layer with one neuron because we have 2 classes. One hidden layer is enough because there is a single compact region that contains the data from the crosses-class and does not contain data from the diamonds-class. This region can have 4 lines as borders, therefore it suffices if there are 4 neurons at the hidden layer. The 4 neurons in the hidden layer describe 4 separating lines and the neuron at the output layer describes the square that is contained between these 4 lines.

(6) **Question:** Suppose that we want to solve the problem of finding out what a good car is by using genetic algorithms. Suppose further that the solution to the problem can be represented by a decision tree as follows:



What is the appropriate chromosome design for the given problem? Which Genetic Algorithm parameters need to be defined? What would be the suitable values of those parameters for the given problem? Provide a short explanation for each.
What is the result of applying a single round of the prototypical Genetic Algorithm? Explain your answer in a clear and compact manner by providing the pseudo code of the algorithm.

Solution:
size = {large, mid, small} → 100, 010, 001, 011, …, 111, 000
brand = {Volvo, BMW, SUV} → 100, 010, 001, 011, …, 111, 000
sport = {yes, no} → 10, 01, 11, 00
engine = {F12, V12, V8} → 100, 010, 001, 011, …, 111, 000
GoodCar = {yes, no} → 10, 01, 11, 00
→ chromosome design:

size	brand	sport	engine	GoodCar
100	100	11	111	01

Fitness function for the given problem can be defined as a Sigmoid function $f(x) = 1 / (1+ e^{-x})$, where x is the percentage of all training examples correctly classified by a specific solution (chromosome).
Selection method – e.g., rank selection method can be used;
Crossover technique – 2-point crossover can be used for the given problem with a crossover mask 1111110000011; the reason is that either size + brand or sport + engine define the solution
Crossover rate – usually k = 60%
Mutation rate – usually 1%
Termination condition – e.g., all training examples are correctly classified

GA pseudo code:
Step 1: Choose initial population.
Step 2: Evaluate the fitness of individuals in the population.
Step 3: Select k individuals to reproduce; breed new generation through crossover and mutation; evaluate the individual fitness of offspring; replace k worse ranked part of population with offspring.
Step 4: Repeat step 3 until the termination condition is reached.

s1/2: 1010101001010, *101111110101*, 0100011111101, 0011111001010, *101101110101*
s3: 1010101110110 (fit 1), 0011111001001 (fit 0), 1010101111110 (fit 2), 1011011001001 (fit 1), 0011111110101 (fit 2), 1011011110101 (fit 3)
result: *101111110101*, *101101110101*, 1011011110101, 0011111110101, 1010101111110