Machine Learning - Problem Set 1

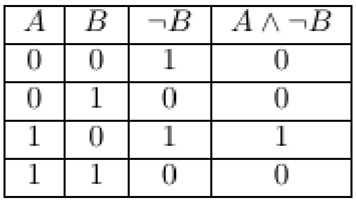
Question 1

Question 2

Design a two-input perceptron that implements the boolean function $A \land \neg B$. Design a two-layer network of perceptrons that implements $A \oplus B$ (where $\oplus$ is XOR).

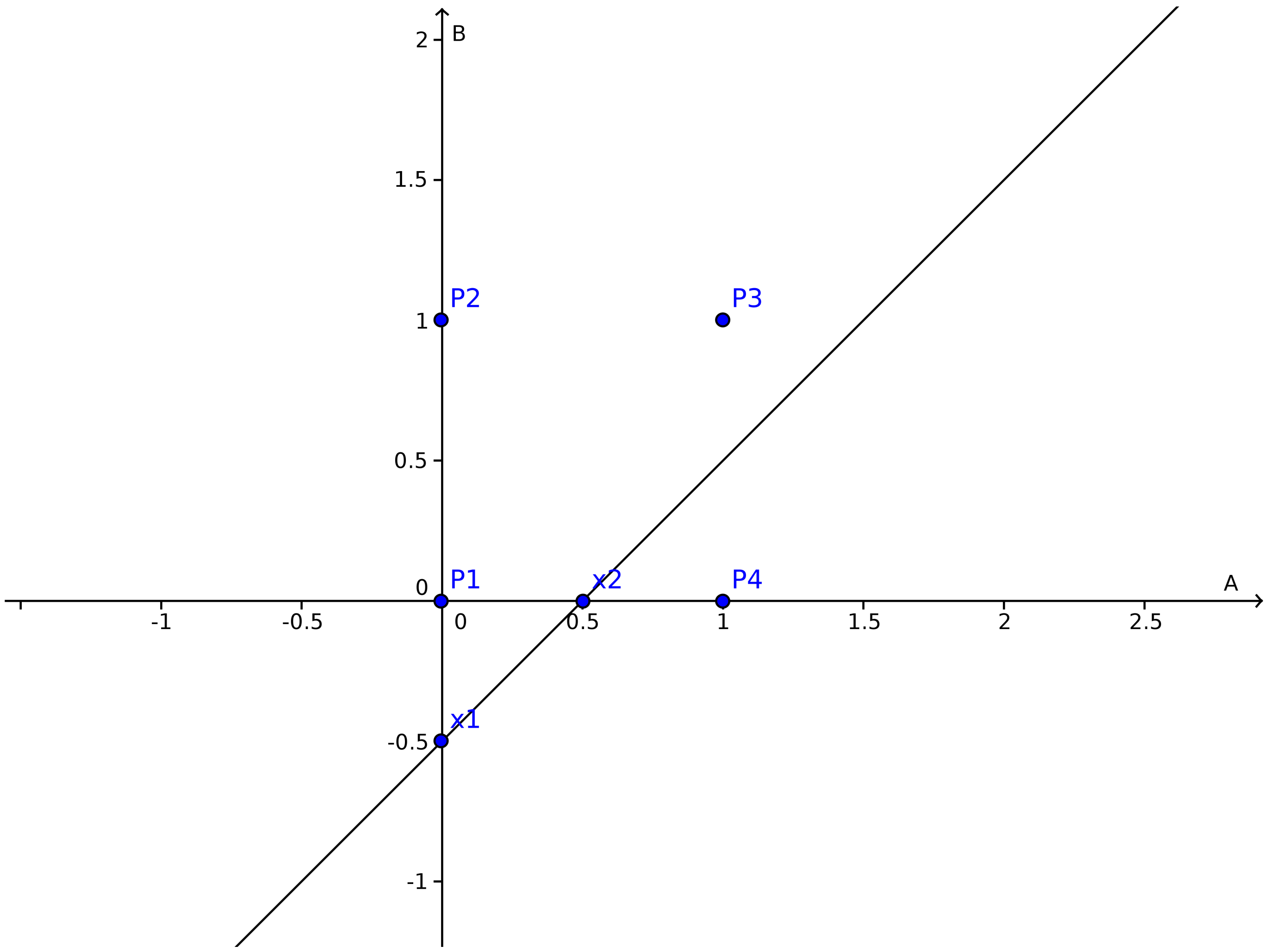
$A \land \neg B$

Let's first explicit the truth table of $A \land \neg B$.



A	B	$\neg B$	$A \land \neg B$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

Let's draw the 4 different cases in a 2D space in figure \ref{fig:2}. The point $P1$ represents the case where $A = 0$ and $B = 0$. The point $P2$ represents the case where $A = 0$ and $B = 1$. The point $P3$ represents the case where $A = 1$ and $B = 1$. The point $P4$ represents the case where $A = 1$ and $B = 0$.

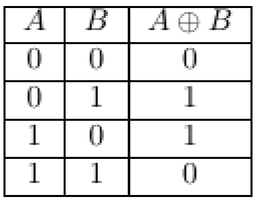


A perceptron that implements the boolean function $A \land \neg B$ is a perceptron that computes an halfplane that contains $P4$ but not $P1$, $P2$ and $P3$.

Therefore we can parametrize that perceptron as follow: $$ y = w_a x_a + w_b x_b $$ $$ \theta = 1/2 $$ $$ w_a = 1/2 $$ $$ w_b = -1/2 $$

$A \oplus B$

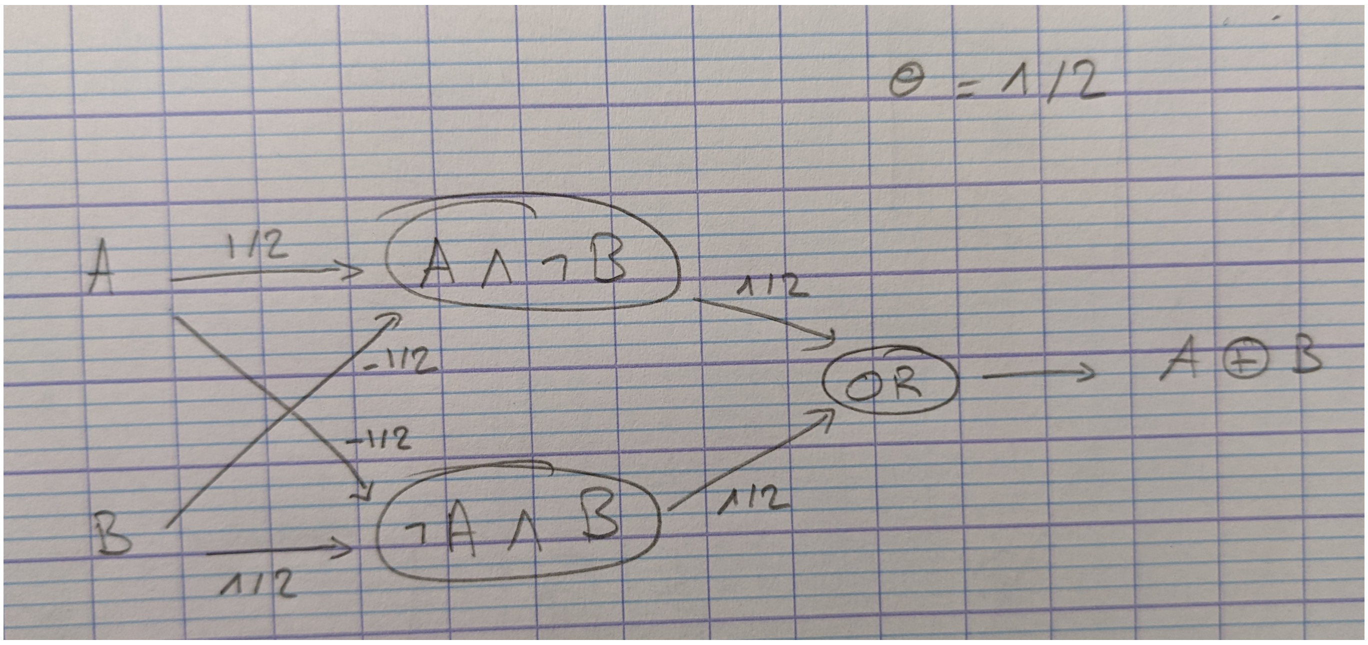
Let's first explicit the truth table of $A \oplus B$.



A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

We notice that XOR is the same as $A \land \neg B$ or $\neg A \land B$.

Then a 2-layer network of perceptrons that implements $A \oplus B$ should be like the one in following figure.



Question 3

Derive the perceptron training rule and gradient descent training rule for a single unit with output, where $\omicron = \omega_0 + \omega_1 x_1 + \omega_1 x_1^2 + \dots + \omega_n x_n + \omega_n x_n^2$. What are the advantages of using gradient descent training rule for training neural networks over the perceptron training rule?

Perceptron Rule

We define $$\hat{y} = (\omicron \ge 0)$$ Then we use $$\Delta \omega_i = \eta (y - \hat{y}) x_i$$ Then we adjust the weights, $$\omega_i \leftarrow \omega_i + \Delta \omega_i$$

Gradient Descent Rule

We define $$E(\omega) = \frac{1}{2} \sum_{x, y \in D} (y - \omicron)^2$$ Then we derive $$\frac{\partial E}{\partial \omega_i} = \frac{\partial \omicron}{\partial \omega_i} \frac{1}{2}\sum_{x, y \in D} (y - \omicron)^2 $$ $$ = \frac{1}{2}\sum_{x, y \in D} (y - \omicron) \frac{\partial \omicron}{\partial \omega_i} * - \sum_j \omega_j x_j$$ $$ = \sum_{x, y \in D} (y - \omicron) (-x_i)$$ $$ = - \sum_{x, y \in D} (y - \omicron) (x_i)$$ So we have $\Delta \omega_i - \eta (y = \omicron) x_i$.

Advantages of Gradient Descent over Perceptron

The gradient descent gives us more information for the update process of the weights. The gradient descent is more robust than the perceptron rule.

Question 4

Explain how one can use Decision Trees to perform regression? Show that when the error function is squared error that the expected value at any leaf is the mean. Take the Boston Housing dataset and use Decision Trees to perform regression

In Decision Trees for Classification, we saw how the tree asks right questions at the right node in order to give accurate and efficient classifications. The way this is done in Classification Trees is by using 2 measures

, namely Entropy and Information Gain. But since we are predicting continuous variables, we cannot calculate the entropy and go through the same process. We need a different measure now. A measure that tells us how much our predictions deviate from the original target and that's the entry-point of mean square error.

Question 5

Question 6

Imagine you had a learning problem with an instance space of points on the plane and a target function that you knew took the form of a line on the plane where all points on one side of the line are positive and all those on the other are negative. If you were constrained to only use decision tree or nearest-neighbor learning, which would you use? Why?

I think KNN should be good. As we know that the data is linearly separable, we can use KNN without feer to outliers. Also KNN is very fast.

Question 7

Give the VC dimension of these hypothesis spaces, briefly explaining your answers:

1. An origin-centered circle (2D)
2. An origin-centered sphere (3D)

1. An origin-centered circle (2D)

The VC dimension of an origin-centered circle is 3.