1 Introduction

Learning algorithms are tools that allow us to solve problems from many domains without domain-specific knowledge. The task of a learning algorithm is to classify a set of objects. For example, classifying different types of motor vehicles as cars, trucks, or motorcycles based on distinguishing properties such as number of doors and vehicle length. If there are $d$-features, then each object can be represented as a $d$-dimensional vector, which we will refer to as a feature vector. Each component gives the value for one feature. Our objective is to design a prediction algorithm that given a vector will correctly predict which vehicle type it is. A non-learning rule-based approach could simply have a domain expert provide a series of rules such as two wheels $\rightarrow$ motorcycle.

Learning rule development is automated. We use domain knowledge to select features and have a domain expert provide classification labels for an example set for training. The task of the learning algorithm: generate a set of rules that correctly labels all training examples. Since we are trying to create rules that all work on unseen data, we will say the rules must be more succinct than just outputting the table of features and training labels itself.

The simplest rule in any $d$-dimensional space is going to be the generalization of a line in a plane and we can take inspiration from nature for an approach to generating such rules.

2 Perceptrons

![Neuron Diagram](image1)

Figure 1: Neuron
Our brain consists of a massive ($10^{11}$) set of neurons. Each individual neuron consists of a cell body with a long axon on one end and a set of smaller dendrites on the other. The dendrites connect to other axons, and when there is enough stimulation on the dendrites of our neuron, a pulse will travel down the neuron through the axon to the next connecting dendrite. We can model the behaviour of neurons using perceptrons.

![Figure 2: Perceptron](image)

Perceptrons take a set of values $x_1, ..., x_n$ as input, weighted by a set of weights $w_1, ..., w_n$. These weighted inputs are summed, and if the sum $\sum_i w_i x_i$ is greater than some threshold, the perceptron fires with a value of 1, otherwise it returns zero (or -1).

### 3 Perceptron Learning Algorithm

The problem of learning a linear separator consists of $n$ labeled examples, $a_1, ..., a_n$ in $d$-dimensional space. The task is to find a $d$-dimensional vector $w$ and a threshold $b$ s.t.

- $w^T a_i > b$ for each $a_i$ labelled +1
- $w^T a_i < b$ for each $a_i$ labelled -1

This pair $(w,b)$ is called a linear separator.

We first add an extra value 1 to each $a_i$ and -b to $w$ to represent the threshold, so that now we are only working with the vector of weights with an additional dimension. We can also rewrite the above equations as:

- $(w^T a_i) l_i > 0$ for all $1 \leq i \leq n$

To find a solution to this, we start with $w = l_1 a_1$. Then we pick any $a_i$ where $(w^T a_i) l_i \leq 0$ and replace $w$ by $w + l_i a_i$. We repeat until $(w^T a_i) l_i > 0$ is satisfied for all $i$. 