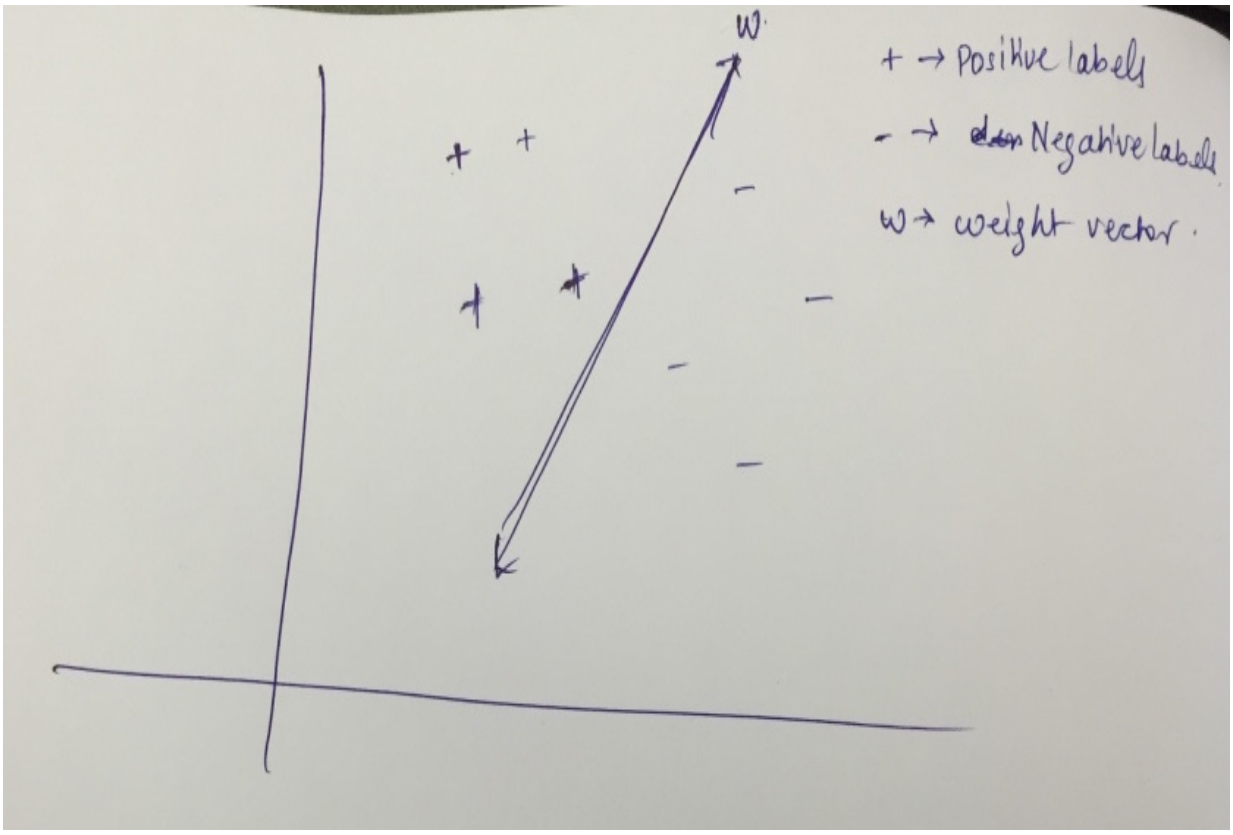
Process Book

Overview and Motivation:

In this project we are attempting to visualize one of the machine learning algorithms called “perceptron”. In machine learning domain this is a fundamental and a very powerful algorithm used to obtain a linear classifier of a data set.

Now taking a glimpse at the below diagram it must be evident that there are set of points labeled +, a set of point labeled – and a line (weight vector) w. The weight vector clearly separates the dataset into two parts. If we denote each example by (x, y , label) and the weight vector by (w1, w1, b), it follows the property that :

label == sign(w1*x + w2*y + b)



The algorithm attempts to find the weight vector w, given the data set and its labels.

So what's the use of the weight vector when we already have the data set and their labels?
If there are any new data points in future whose labels are unknown the weight vector can be used to obtain the label of that new point. There are many practical applications of this algorithm. The positive and negative labels could represent whether a mail is a spam of not, whether it will rain today or not, whether a given flower is a lily or jasmine etc.

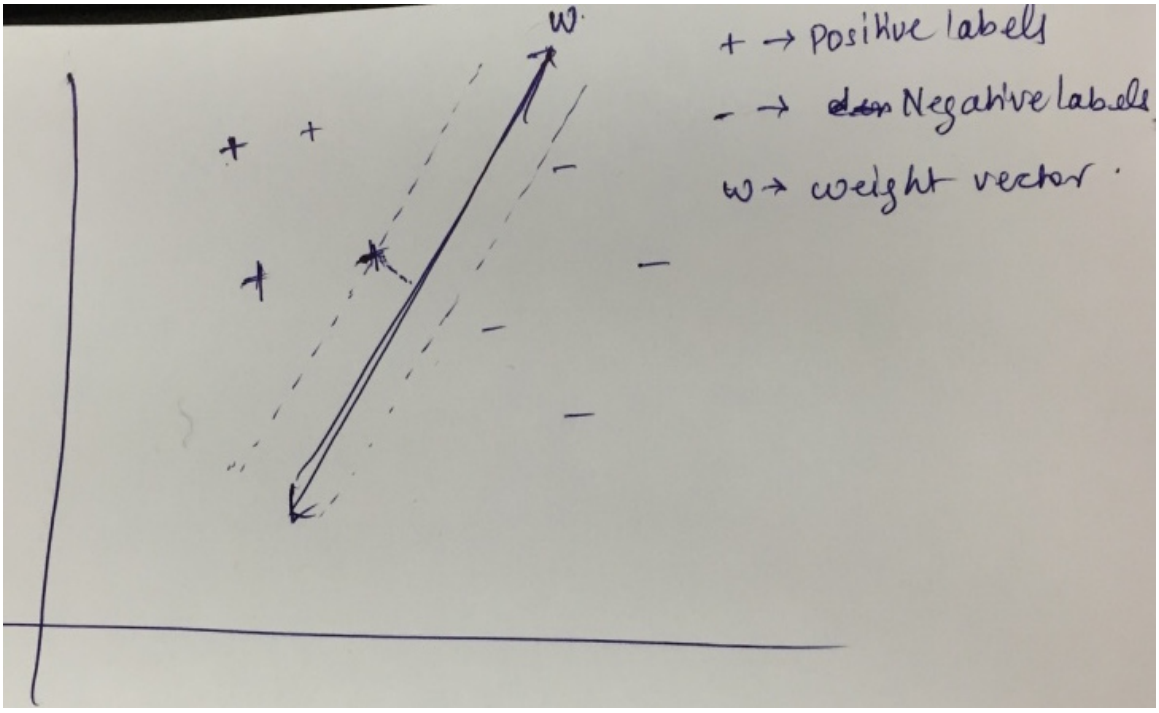
Though it is obvious, the mechanics of the algorithm are effectively understood only with the help of a well-designed visualization. It must be noted that the weight vector changes several times through out the course of the algorithm, but what is important is, how does it change?

Through this project we also address the question of how the perceptron would behave if we took only a subset of the given data.

Margins:

A margin is defined as the distance of the closest correctly classified point from a given weight vector.

One important aspect of the perceptron algorithm is the concept of margins. The concept of margins can be better understood by actually seeing the margin of a given weight vector than theorizing it. So, in this project we address the concept of margins and answer some questions like why larger margins are better.



Related Work:

What inspired us most was the fact that when we studied machine -learning course we looked up for interactive resources online, which would enable us to play around with the weight vector and see how it is updated?

In addition the home-works in this course enabled us to learn about d3 and interactive visualization techniques, which drove us towards this project.

Furthermore, we are hoping that this be used later on by students learning machine learning to better understand the perceptron algorithm.

The below website had some cool visualization of machine learning algorithms and it also influenced us in coming up with this project.

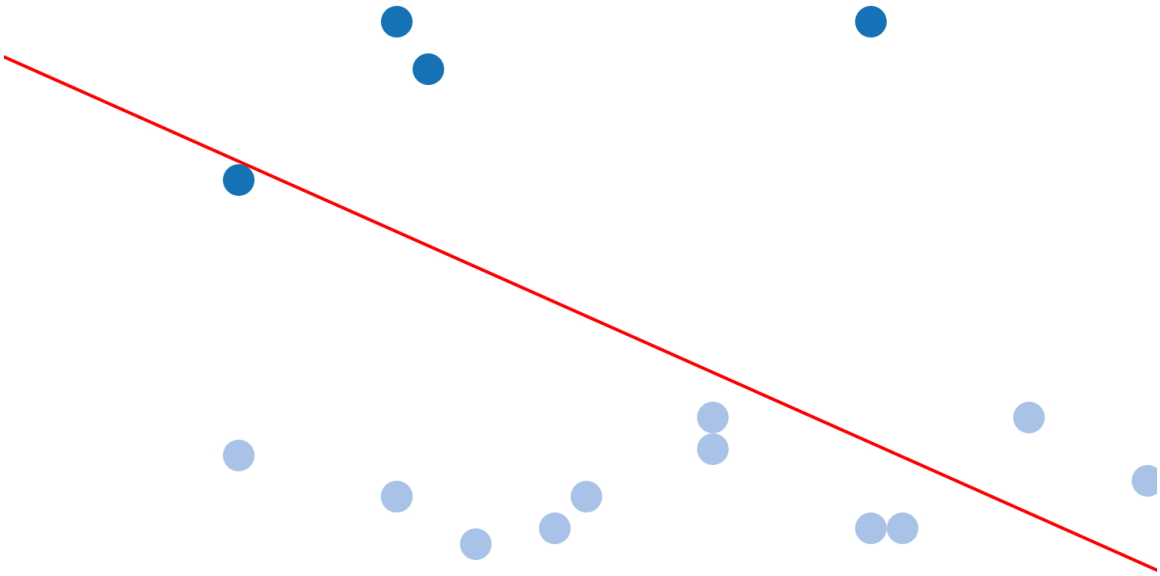
http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Data: Source, scraping method, cleanup, etc.

One of the most critical aspects of this project is find the right data set that can be used fit all our requirements. Though there are lots of datasets out there it was important to find a dataset that had binary classification, two features (as the visualizations are 2d in nature) and was linearly separable.

After searching and failing to find a good data set we decided to generate synthetic data set that catered to all our needs.

The below snapshot represents a subset of points with binary classification along with the weight vector with perceptron in action.



Later on we decided to be more creative and found a data set named iris data set.

The data set consists of 50 samples from three species of *Iris flowers* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

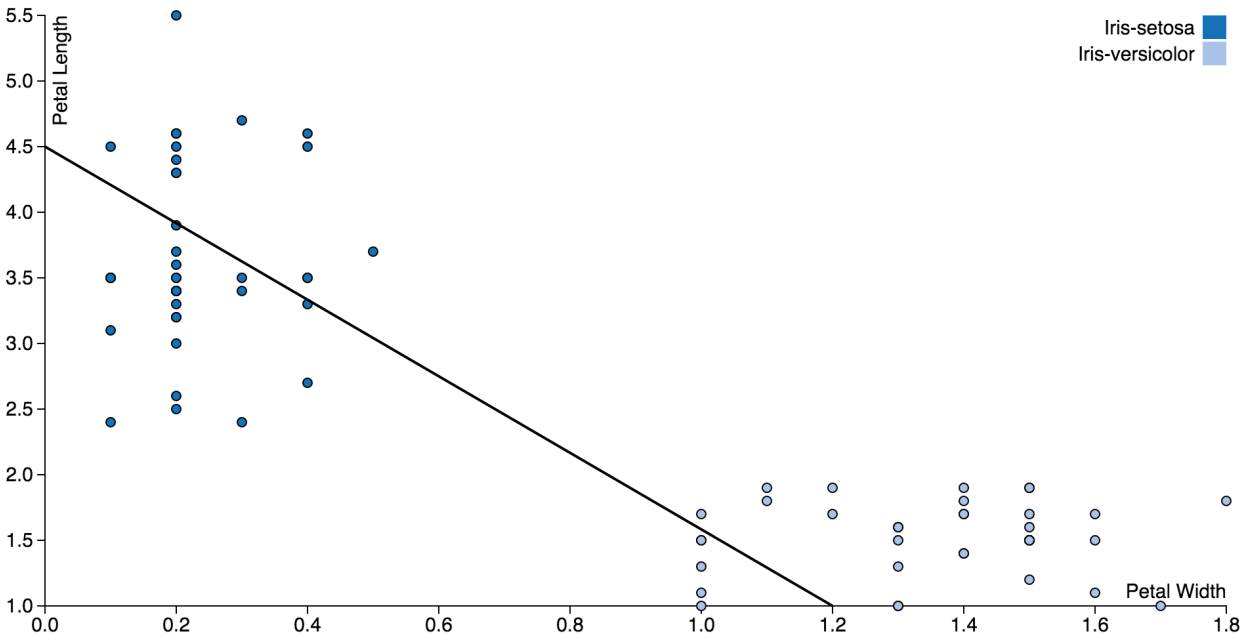
Exploratory Data Analysis:

Now, the data set clearly had three classes of flowers Iris-setosa, Iris virginica, Iris versicolor and perceptron works only on binary classified data-set. We initially decided to PCA of this data set and use the top two principal components to run our experiments. But on performing scatter plots for various combinations of the attributes we found that upon selecting two particular attributes named " petal length" and "sepal width" the data gives good clusters that could be suitable for perceptron algorithm.

We then went on to pick only two classes named Iris-setosa and Iris versicolor that looked promising to run our experiment.

This gave us an interesting idea of using d3.path to construct a flower and use flowers instead of dots while plotting a data points.

The below image shows the snapshot of the iris data set after cleaning up and processing it to be suitable for out project:



Design Evolution:

We though of 3 different designs for representing perceptron

a: We can draw a scatter plot and use d3 symbols to visualize different classes and use the a line to separate the two regions into different classes. With this approach we can potentially make use of the idea of the brush discussed in the class and define the region of the training data set on which perceptron algorithm must run. Thus, one can observe the accuracy with the variation of the data set.

b: We also though of using a 3d pie graph which is essential has two regions representing two classes and the axis of the pie can be used to represent the weight vector that separates the given data. Though this is one way of representing the perceptron model, it is ineffective in visualizing some key properties like representing the margin.

c: Since we are using a brush to select the region of interest , we thought we could use a bar graph which contains two bars corresponding to the two labels. However for the similar reasons as the pie chart, this is very in effective.

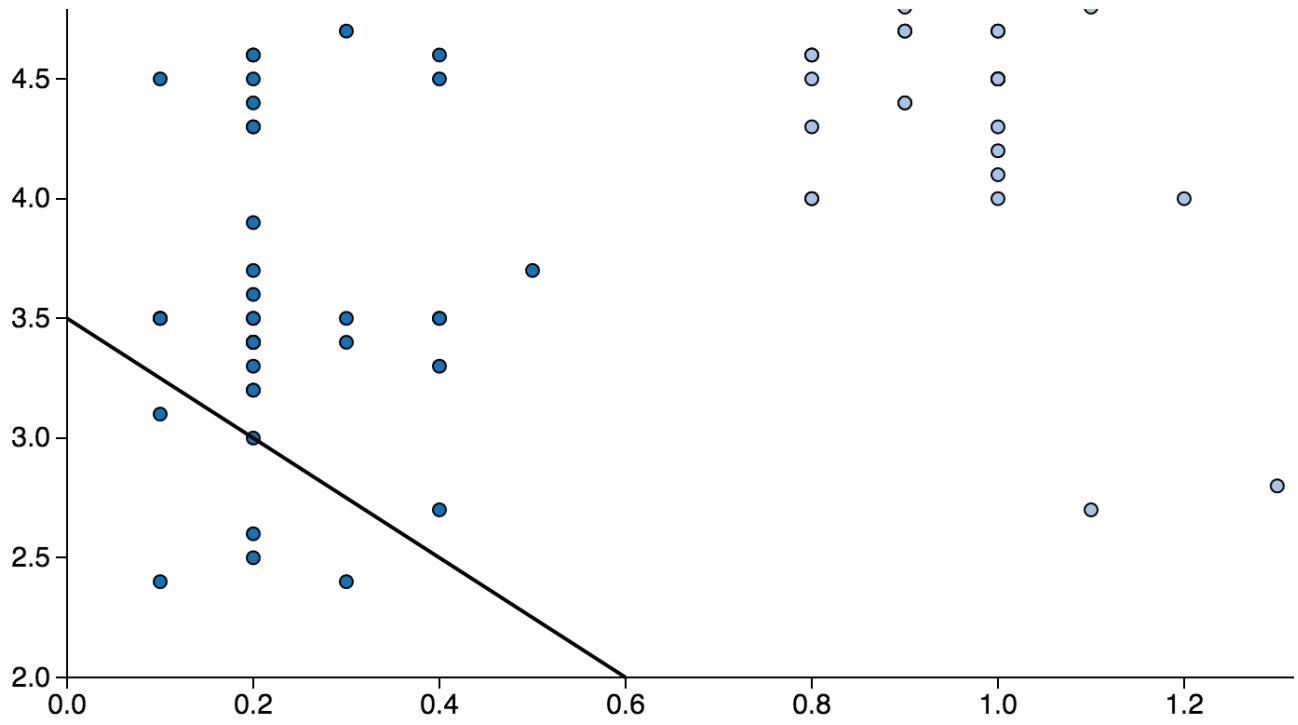
Implementation

The primary intent of this project is developing a well-perceived perceptron visualization that will enable the users to understand the important aspects of the algorithm. The points are divided into two classes as positive and negative and it is a known concept that any weight vector divides the space into two halves, i.e positive and negative. So, the user is given the flexibility to choose a point by clicking on it and observe how the weight vector moves from its current position. We have used d3 transitions to show the movement of the weight vector.

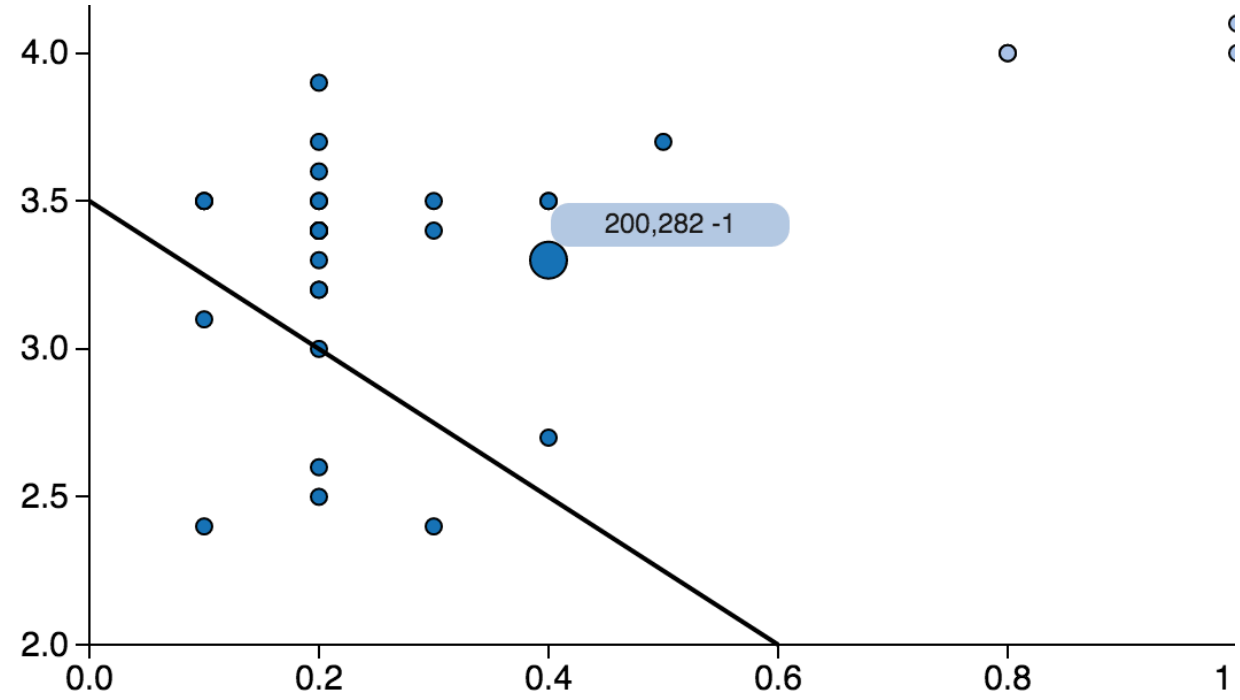
Furthermore, since this is a vanilla version of perceptron as opposed to its variants the weight vector upon clicking a misclassified point moves in the direction that classifies the point correctly but doesn't necessarily classify the point clicked correctly.

The images below illustrate the above idea:

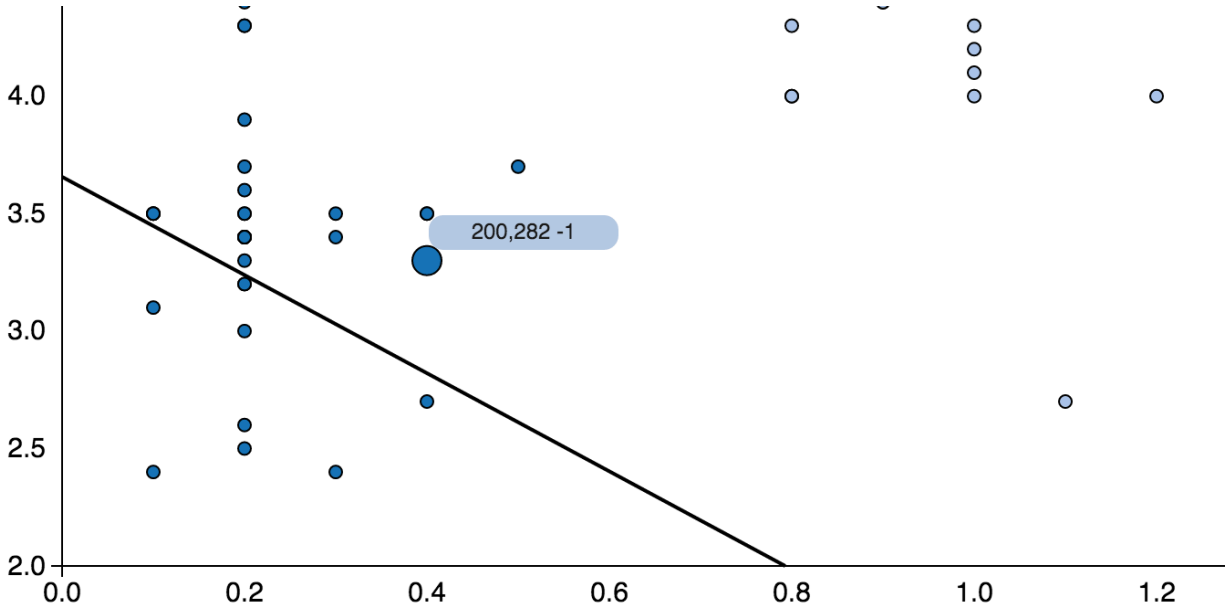
The below image shows the initial position of the weight vector which has a property that classifies every point below it as negative and all the positive above it as negative.



The below figure shows the point that has a negative label and present above the weight vector. Thus user upon clicking that point the weight vector should shift in a way that classifies that point correctly.

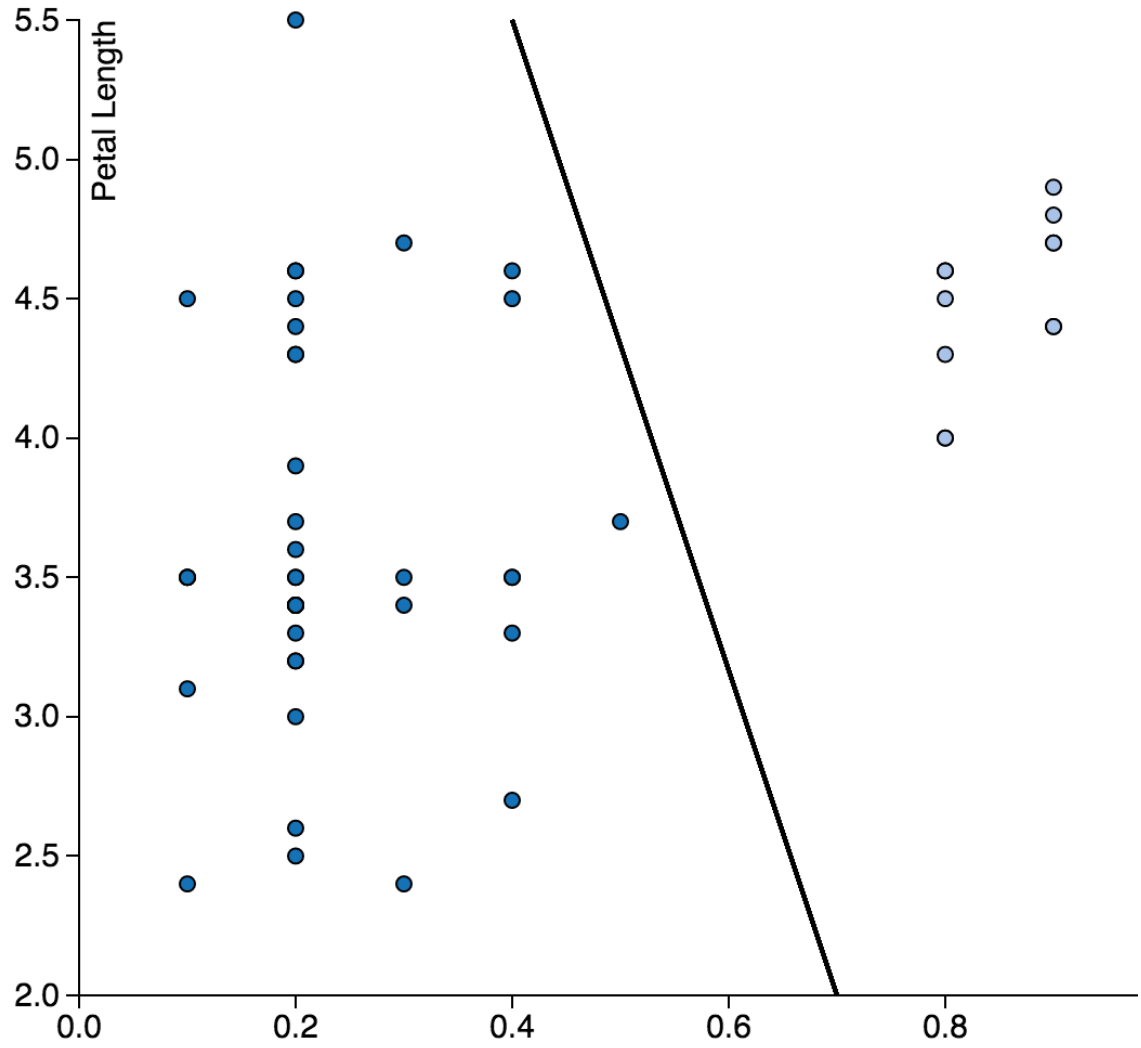


The below figure show that final position of the vector after the weight vector has shifted in favor the new point. As it can be observed, the new x and y intercepts are slightly larger



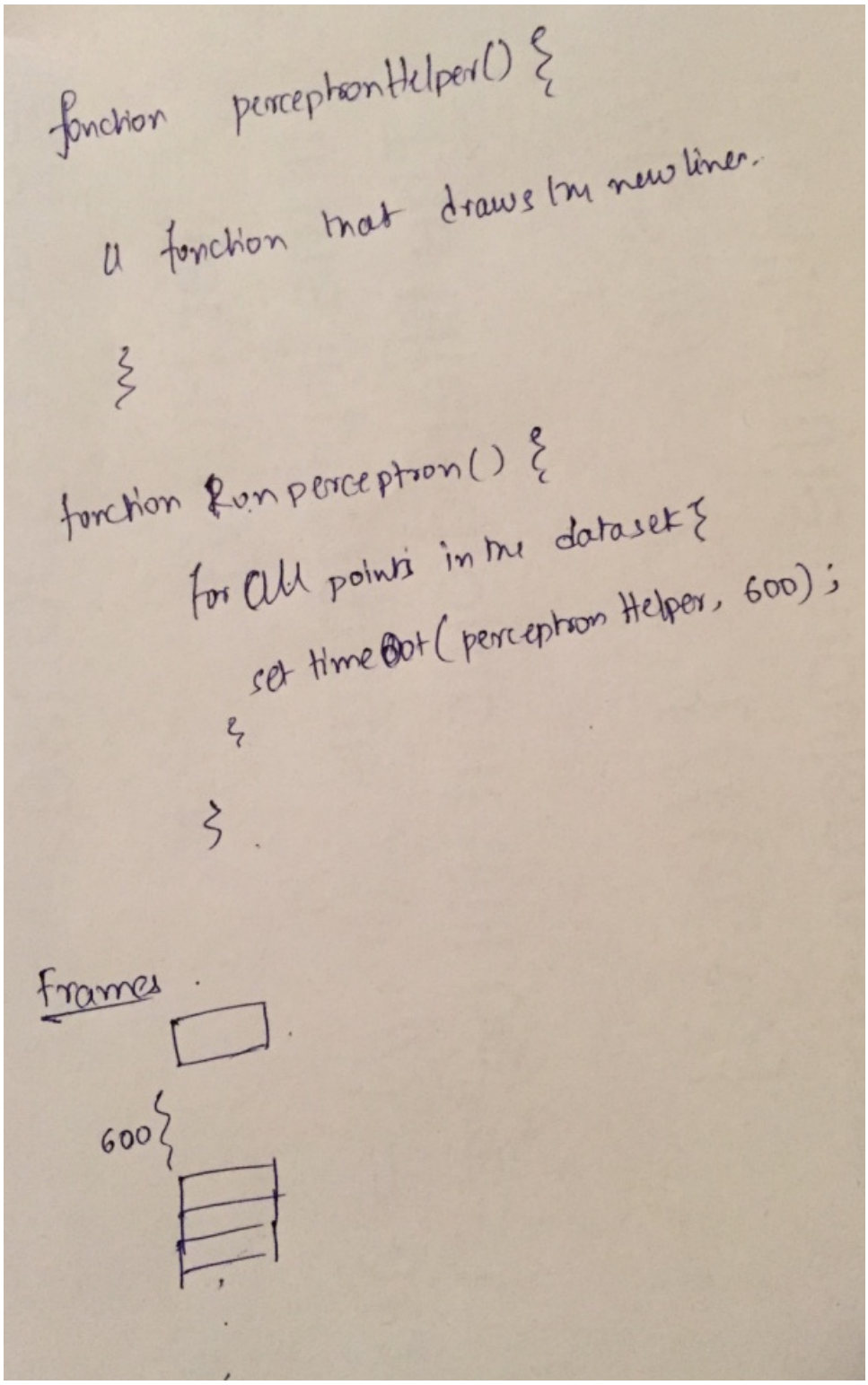
One more key property of the perceptron is the running of the algorithm itself as goes over all the data points one after the other and finally converges to a linear separator if the given data is linearly separable. For this purpose we decided to we decided to use set Timeout function from java script and wrote a small recursive function using lambdas.

Thus the users upon clicking the button named “run perceptron” the weight vector updates every 600 milliseconds and goes over all the points finally converging to a linear separator.
The below diagram represents the final position of the weight vector after it converges:

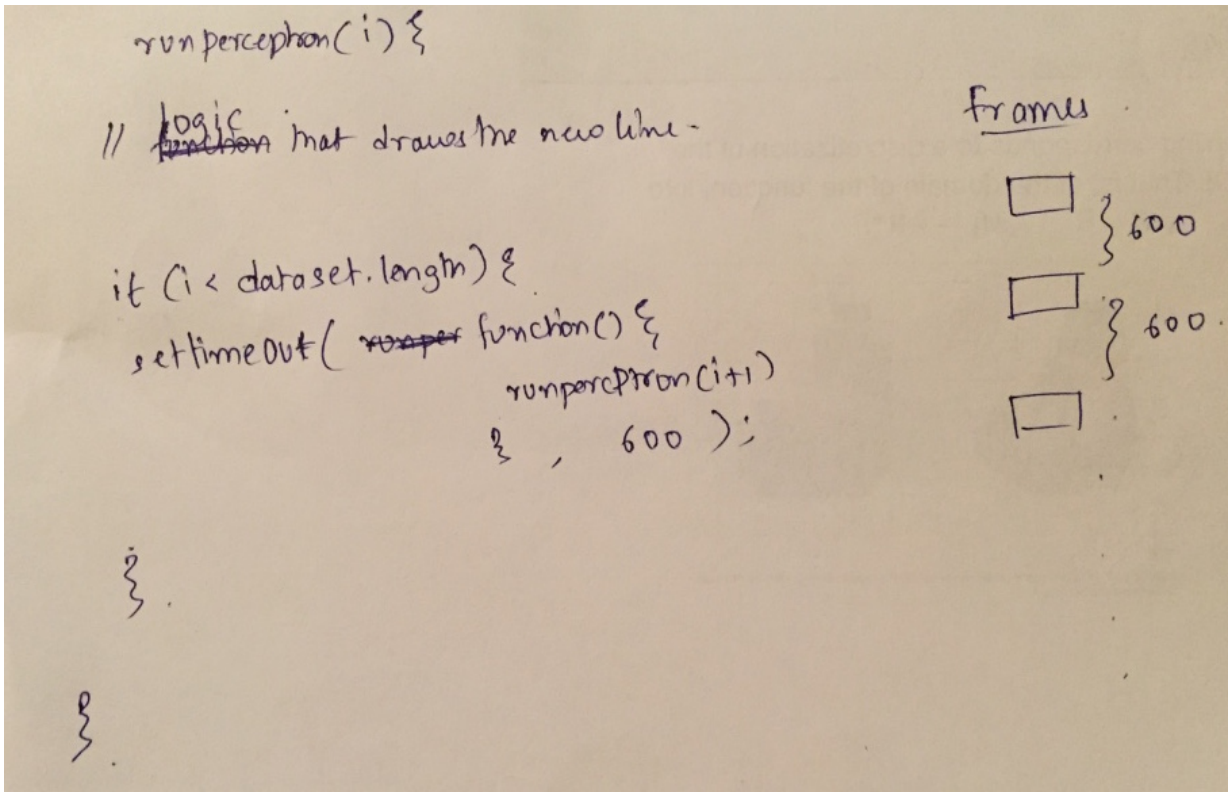


run perceptron Next

In implementing the above visualization we hit a road block when we where trying to use the below logic where in the frames where set one after the other because of the for loop instead of getting set at even 600 milliseconds interval.



We later resolved it writing a recursive function using lambdas that set frames at even 600 milliseconds.



- Design Evolution: What are the different visualizations you considered? Justify the design decisions you made using the perceptual and design principles you learned in the course. Did you deviate from your proposal?

The visualization we are attempting to realize is best achieved by the use of interactive scatter plots. Hence we decided to go with this option.

We wanted to demonstrate several concepts like:

1) Movement or update of the weight vector during the training phase
2) Running of the perceptron algorithm
3) The concept of margins
4) Visualize why larger margins are a better choice among the set of weight vectors that linearly separate the data.
5) The testing of the resultant weight vector on the test data set.

Hence we firstly needed to first decide on the number of visualizations we wanted to illustrate the above concepts. In addition, since we where using a data set in which we are trying to a separate two sets of flowers based on the properties of their petals namely petal width and petal length, we thought it will be very interesting to tell a story around it and hence we came up with the following idea:

A kid is trying to separate these two sets of flowers we have under consideration, namely Iris-setosa which looks something like the below:

and Iris-versicolor which looks the below:

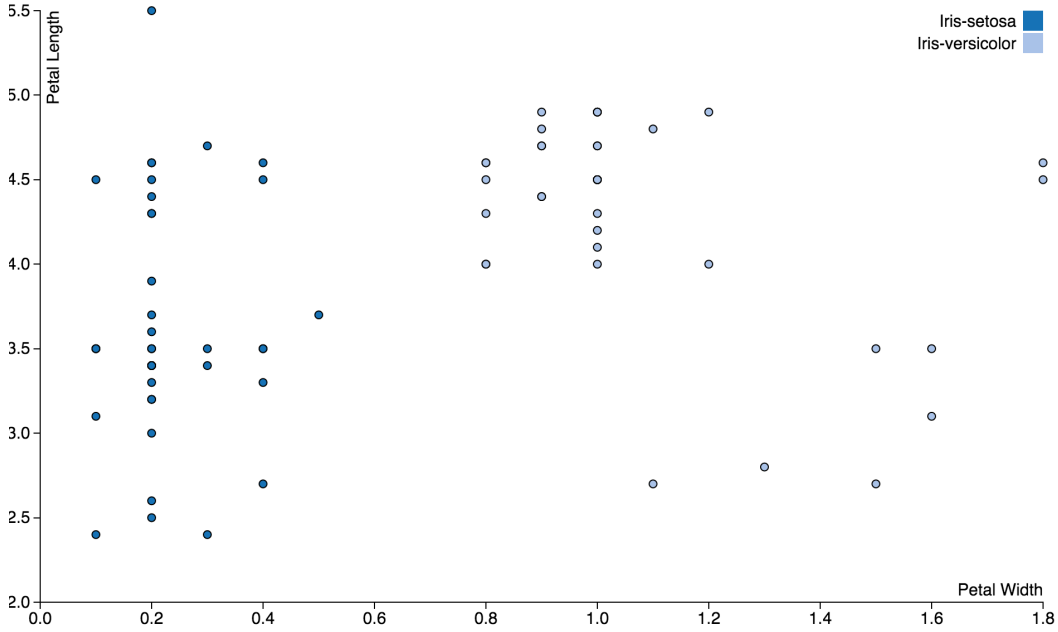
As it can be observed that the two sets of flowers are very similar to each other in terms of the number of petals and the color of the petals. Hence the perceptron algorithm becomes a very good candidate for separating such samples of data.

The kid thus uses the perceptron algorithm to initially train his weight vector to obtain to weight vector that will separate the training set perfectly and there after use the final weight vector to separate the test data set and check for the accuracy.
Here is the kid:

Further, the expression of the kid changes with the running of the algorithm and we have used various expression of this kid to capture that effect.

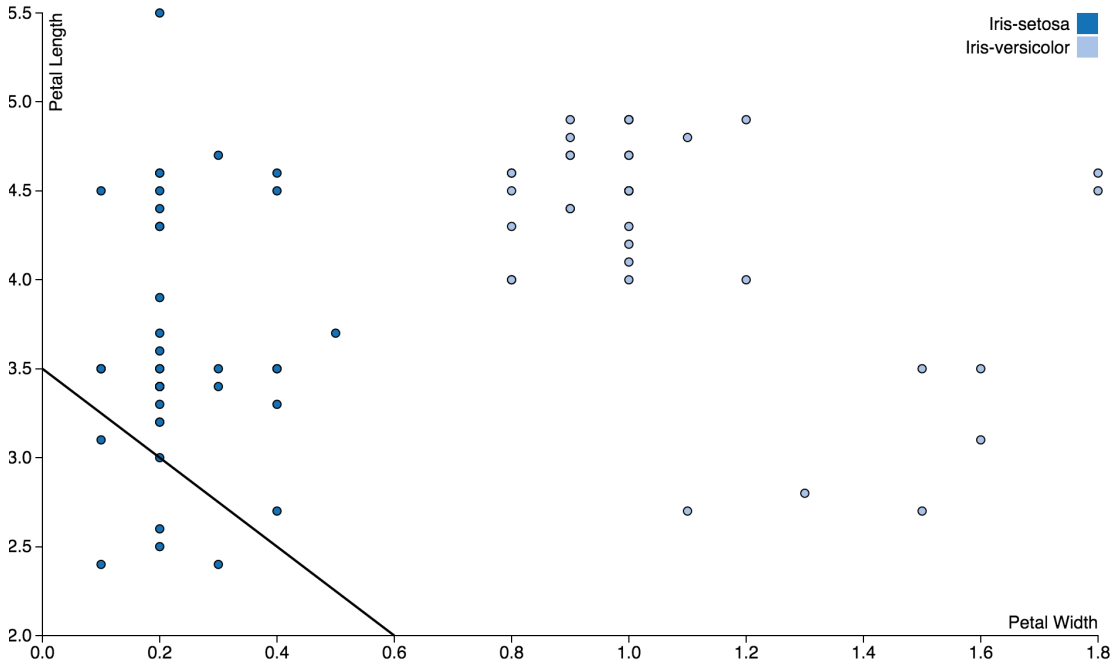
We have used in the first visualization/view to first introduce the data using a scatter plot. We have used d3 transition to obtain some nice transition effect in loading this data:

View1:



In view2 we needed to show the initial weight vector chosen at random and then the running of the perceptron. As explained above the user can check manually how the perceptron updates its vector upon clicking a particular point. The weight vector doesn't change its position if the point is already classified correctly, while it moves in the direction that classifies a given point correctly should we click a point that is incorrectly classified.

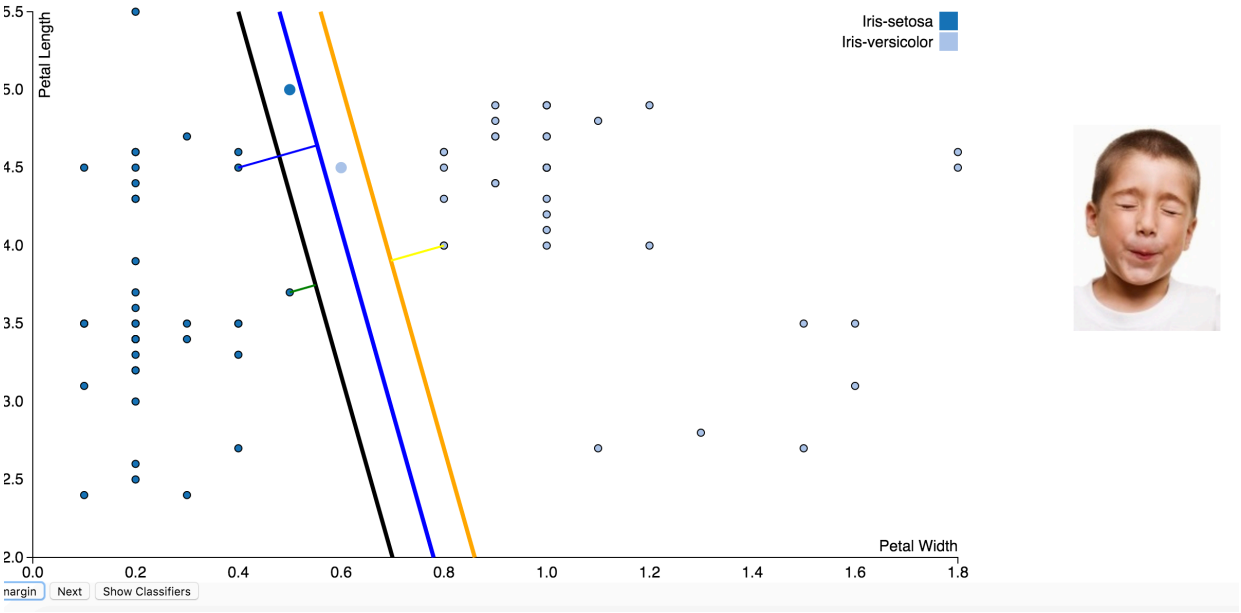
Furthermore, the run perceptron button runs the algorithm on all the points and gives the final weight vector:



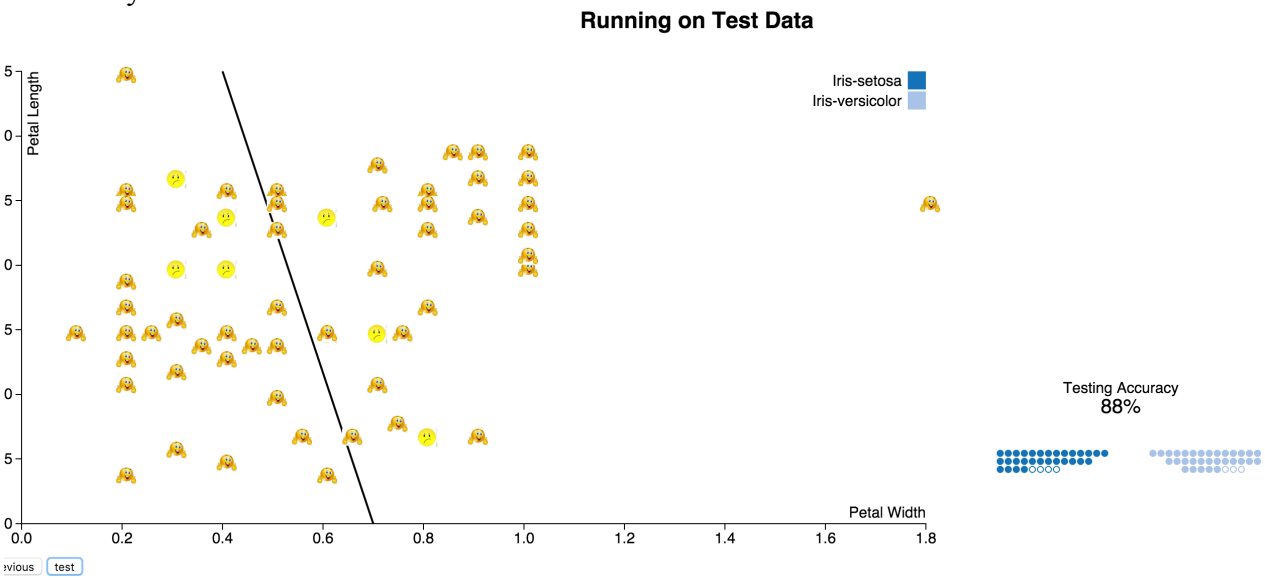
Training

The view3 then demonstrates the use of margins by show different margins possible and the use of the maximum margin:

The below diagram shows the screen shot of view3:



In the final view the weight vector is tested on the test dataset and the below diagram indicates the result. As it can be seen that the happy faces indicate that the points are correctly classified and the sad faces indicate that the points in the test data set are incorrectly classified:



Run perceptron:
The weight vector starts moving upon clicking the run perceptron button by training on the given data set. Since we needed to show which point is under consideration at any given point we have used a circle

Evaluation:
We evaluated the flowers data and found out that the data is linearly separable and the sequence of how a classifier can train itself on this data. With the help of visualization technique we were able to figure out the different set of margins possible on this dataset.
We are working on make it more appealing by replacing circles with actual shapes and bringing some more interaction elements in the canvas.