Perceptron

Balazs B Ujfalussy

13/11/2017

This is a demo for illustrating the calssification by a single neuron - now called the perceptron. We will first consider a linearly separable problem, which can be solved by a single neuron with linear input integration corresponding to single layer perceptron. Next, we will see how single neurons fail for more challenging, non linealy separable tasks. Finally, we will illustrate how we can fix the problem by introducing another layer of neurons, a so called hidden layer.

Linearly separable classification

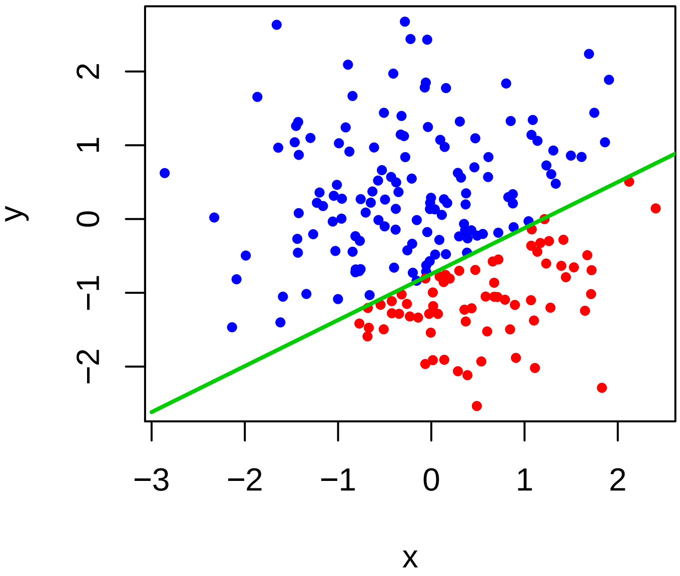
Here we will use a randomly chosen neuron, a teacher, to define the target classification for another neuron, the student. This choice gurantees that the problem is learnable.

```
source('./perceptron.R', chdir=T)
set.seed(14) # initialising random seed
n.input <- 2 # number of presynaptic neurons - the dimensionality of the problem
n.patterns <- 200 # total number of patterns
e <- 10

## input patterns: a matrix of 200x3. First and second column are Gaussian rundom numbers. The third is
patterns <- cbind(matrix(rnorm(n.input*n.patterns), n.patterns), rep(1, n.patterns))
## the teacher - random initializaton:
ww <- rnorm(3)
x <- seq(-3,3)
separatrix <- (-1) * (ww[1] * x  + ww[3]) / ww[2]
target <- (sign(patterns %*% ww) + 1) / 2
```

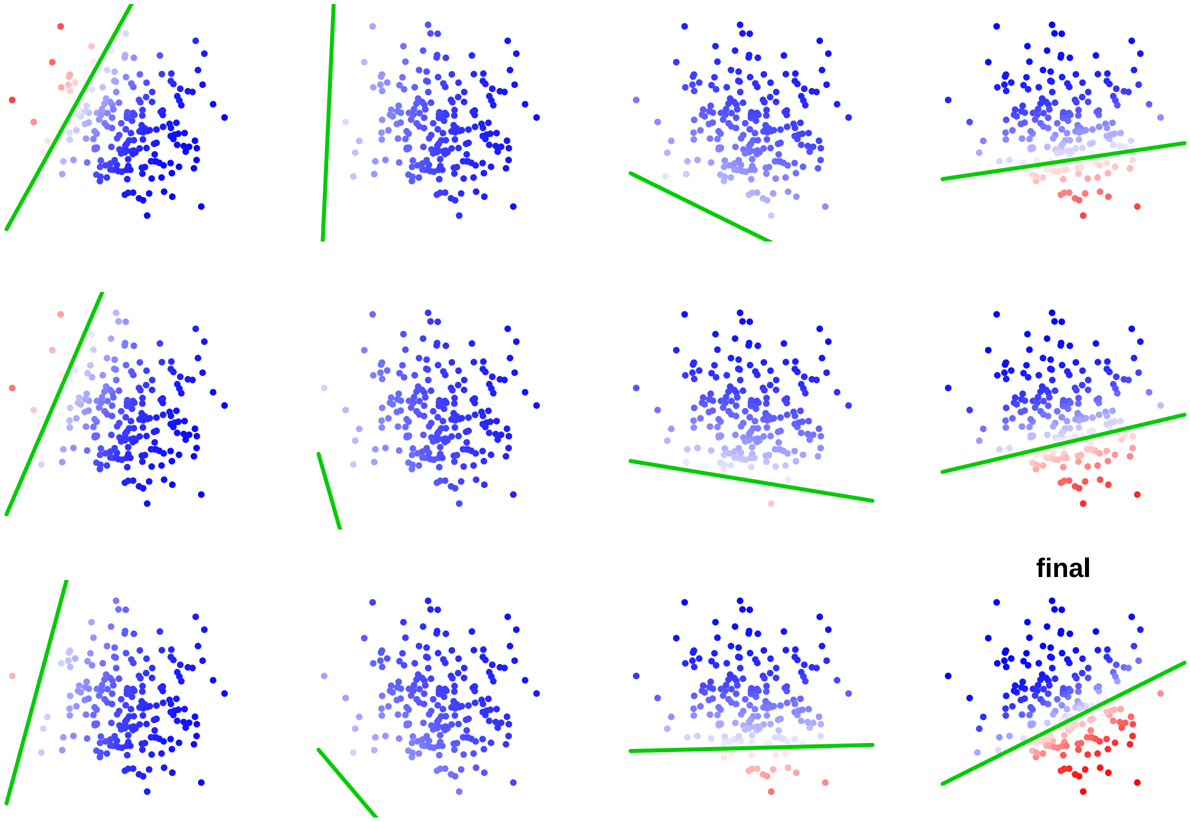
This is the teacher's classification of its inputs. It corresponds to a single line separating the 2D plane into two halves: a red and a blue half. Importantly, the weights of a single neuron always correspond to such a linear decision boundary in the N-dimensional space of the inputs.

```
plot(patterns[,1], patterns[,2], pch=16, col=blueRed2(target), xlab="x", ylab="y", cex=0.7)
lines(x, separatrix, col=3, lwd=2)
```



Now the student is initialised randomly and aims to learn the correact classification of the inputs, i.e., the position and the orientation of the boundary between the red and the blue-coloured region, represented by the green line. The following figure shows the progress of the learning process. The color code indicates the uncertainty of the classification - darker colours are more certain, whereas white means that the student has no idea.

```
set.seed(20)
w.init <- rnorm(3)
w.final <- train.perceptron(w.init, patterns, target, 20, T)
```



The misclassification rate is 2% - but only a few cases are misclassified near the boundary.

Random target pattern

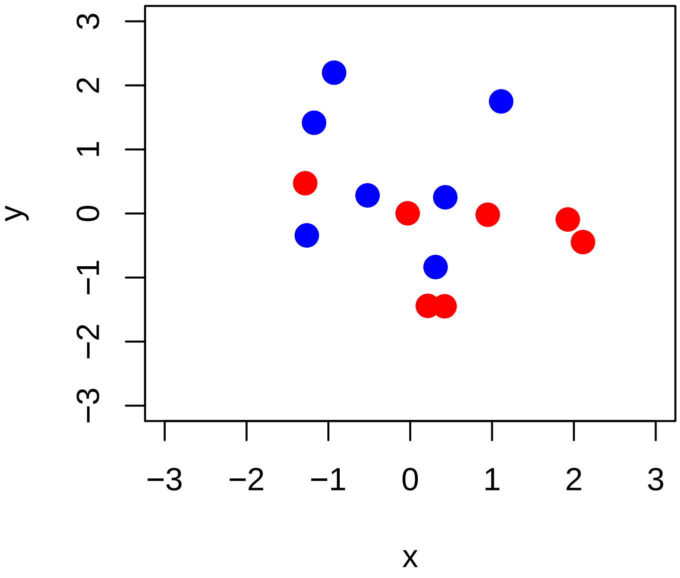
Now, instead of learning a classification that the teacher neuron already solved, let's try to do something new and more exciting: Try to learn a mapping from 14 input patterns to random outcomes.

```r
set.seed(29)
n.patterns <- 14
sp <- 0.5

patterns <- cbind(matrix(rnorm(n.input*n.patterns), n.patterns), rep(1, n.patterns))
target <- rep(0, n.patterns)
target[sample(1:14, 7)] <- 1
```

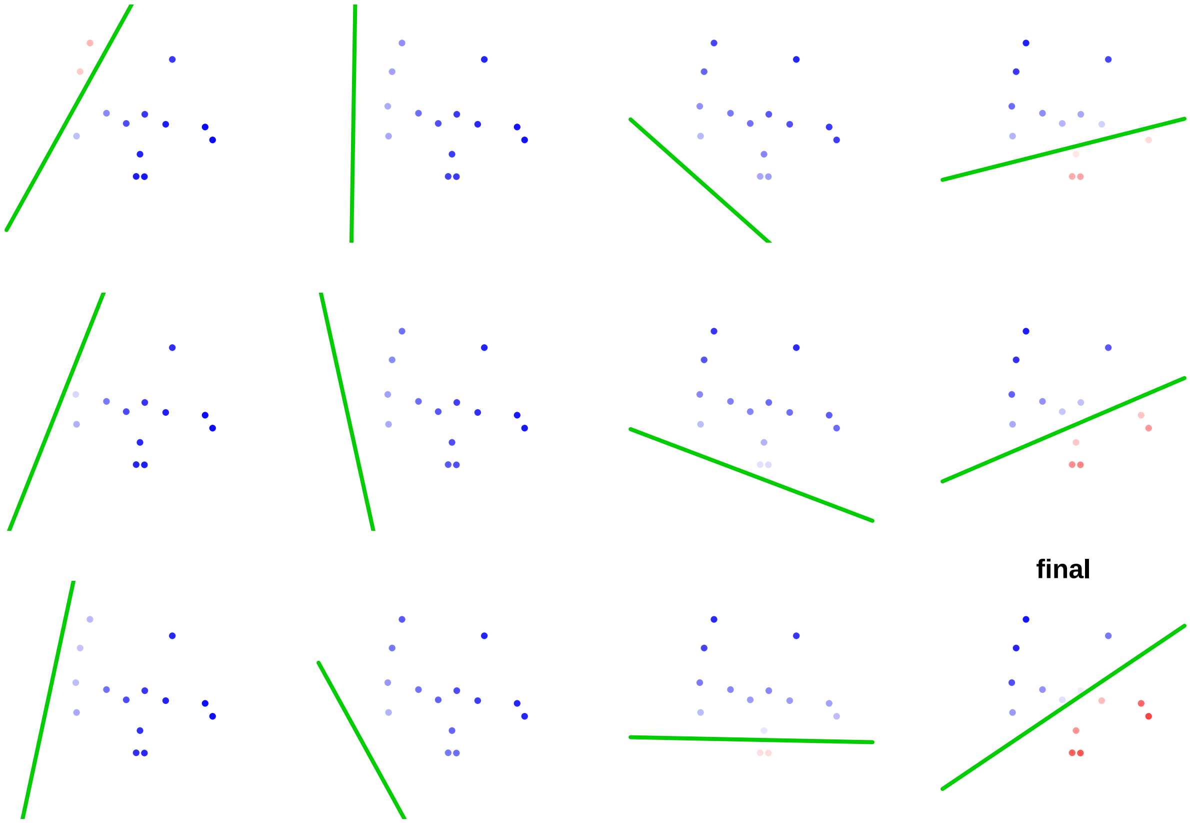
Here I plot the target mapping. It is clearly impossible to separate the red dots from the blue using a straight line - so most likely the simple neuron will not be able to learn the task.

```r
m <- ceiling(max(abs(patterns)))
plot(patterns[,1], patterns[,2], pch=16, col=blueRed2(target), xlab="x", ylab="y", cex=1.7, xlim=c(-m,
```



Let's observe how the student neuron will try to implement the mapping:

```r
set.seed(20)
w.init <- rnorm(3)
w.final <- train.perceptron(w.init, patterns, target, 20, T)
```

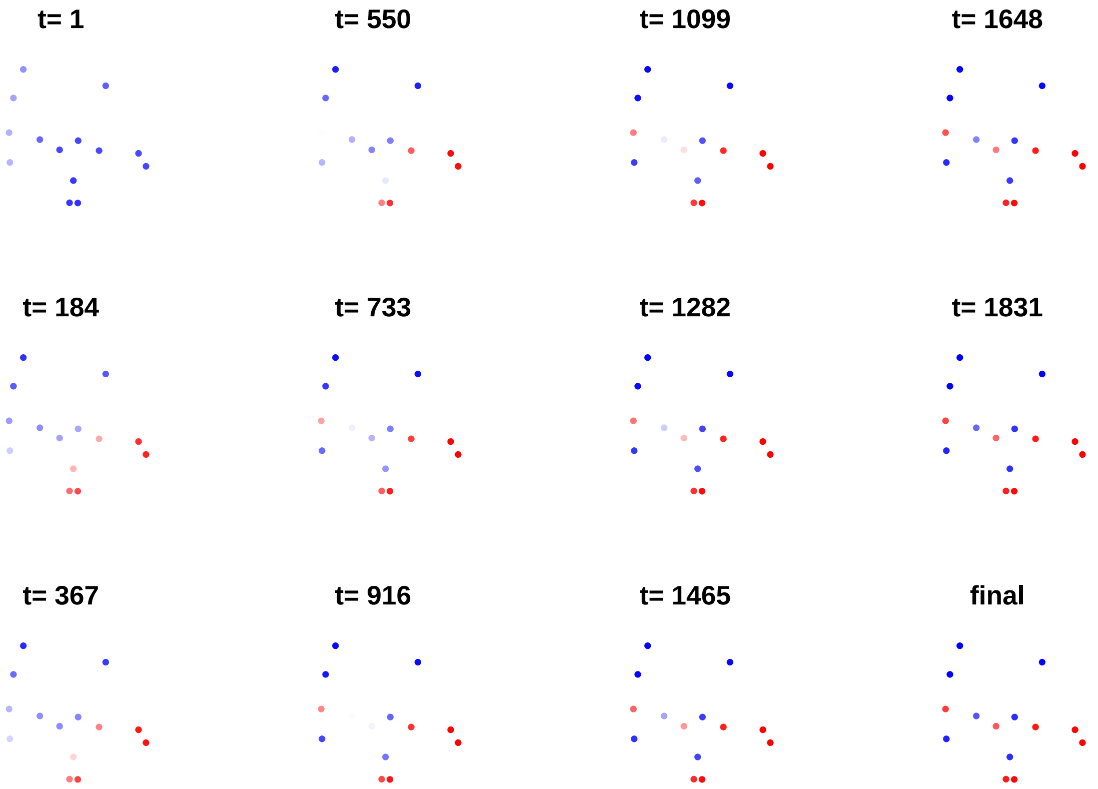


The misclassification rate is 21%. It is not that bad after all, but there are still 3 patterns clearly misclassified.

Solving the non-separable problem - multiple layers

Now we will introduce 4 additional neurons. These neurons are not participating in the output, and their activity is not constrained by the input - they consist of an additional, hidden layer between the input and the output unit. This is why we call this network as a two-layer neural network, or perceptron in this case. These new neurons provide the network with sufficient flexibility to learn more complex problems.

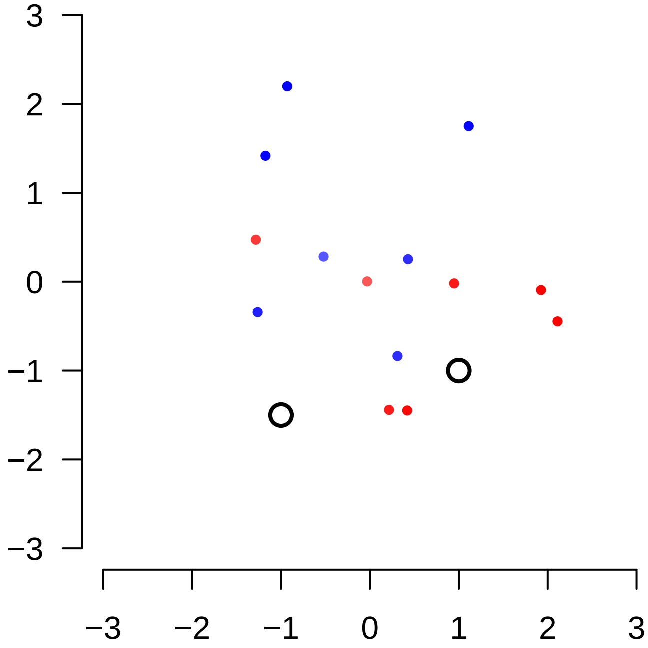
```r
n.hidden <- 10 # number of hidden neurons
set.seed(114)
w.in <- matrix(rnorm(3*n.hidden), 3)
w.out <- matrix(rnorm(n.hidden+1), n.hidden+1)
w.final.2L <- train.perceptron.2L(w.in, w.out, patterns, target, Tmax=2000, e=10, graphics=T, verbose=F
```



Indeed, at the end all points are correctly classified.

Homework

```
v.u <- eval.perceptron.2L(w.final.2L$w.in, w.final.2L$w.out, patterns, T)
points(1,-1, pch=21, col=1, cex=1.5, lwd=2)
points(-1,-1.5, pch=21, col=1, cex=1.5, lwd=2)
axis(1); axis(2, las=2)
```



Consider a new point in the above classification task, shown now in black. These are two novel, test points that the newtork has never seen before. Now the question is, what the network believes - without retraining it - about the color of these novel points?

- Would you colour it blue or red?
- What is the classification of the 2-layer network? Try to use different initial conditions or different number of hidden neurons (between e.g., 3 and 20), but always train the network using the original 14 points, and not the two new points! The new points are only added when you evaluate the network!
- What is the problem here? What would be the solution to this problem? How the brain can possibly solve this problem?