CS342 Machine Learning: Lab #4
ANN: Perceptron

Labs on February 11 & 12, 2016

Week 5 of Term 2

Office Hours:
CS 3.07, Monday & Friday 10:00-11:00

Instructor: **Dr Theo Damoulas** (T.Damoulas@warwick.ac.uk)
Tutors: **Helen McKay** (H.McKay@warwick.ac.uk), **Shan Lin** (Shan.Lin@warwick.ac.uk)

In the fifth Lab we will code up a simple Perceptron (Heaviside step function, linearly-separable problems) and apply it to some toy data. If there is time left (else you can continue on your own time) we will code up a linear unit perceptron with batch-mode gradient descent.

1 Standard Perceptron (step activation function, no gradient info)

Here is the pseudocode of the Perceptron from the lecture:

Algorithm 1 Perceptron

Initialise $\mathbf{w}$ randomly
$\eta = 0.1$ (can try other values to see the effect)
while non-zero error component **do**
    for i=1 to N (number of training examples) **do**
        Choose the i$^{\text{th}}$ training example $\mathbf{x}, t$ (really I mean $\mathbf{x}_i, t_i$)
        Compute dot product $\mathbf{xw}$
        Compute error(i) = t - sign($\mathbf{xw}$)
        Update $\mathbf{w}$ += $\eta\cdot$error(i)$\cdot\mathbf{x}^{\mathrm{T}}$

Tips: You can use math.copysign(1,a) to implement the sign(a) function. You can check for non-zero entries in tuple named "values" with all(v == 0 for v in values). **Don't forget to include the bias term in X by adding a column of 1s to your data**.

→ Implement the Perceptron in Python and test it with some linearly separable data from Table 2.

x_1 (first attribute)	x_2 (second attribute)	t (Class Variable)
1	-1	-1
2	1	1
1.5	0.5	-1
2	-1	-1
1	2	1

Table 1: A toy dataset with 5 observations (rows), 2 attributes (x_1, x_2) and the target class (Class Variable)

→ Create your own linearly-separable data and test your Perceptron
→ What is the effect of the learning rate η?
→ Is the algorithm affected by different initialisation? Plot the data and the resulting linear separator

2 Linear-unit Perceptron (batch gradient descent)

Here is the pseudocode of the Linear-unit Perceptron with gradient descent from the lecture:

Algorithm 2 Linear-unit perceptron with batch-mode gradient descent

Initialise $\mathbf{w}$ randomly
$\eta = 0.1$ (can try other values to see the effect)
while not converged (e.g. for 10 or 100 iterations) **do**
    Update $\mathbf{w}$ += $\eta\mathbf{X}^{\mathrm{T}}(\mathbf{t} - \mathbf{Xw})$

→ Implement the linear-unit perceptron in Python and run it on the data above
→ Create some non-linearly separated data and study the behaviour of the algorithm
→ What is a better convergence criterion to use for termination?