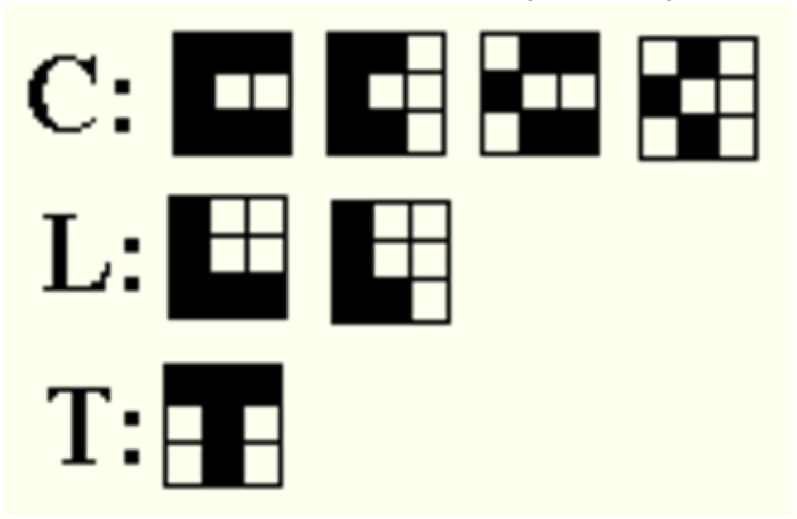
Tutorial 7

Thursday 29th September, 2016

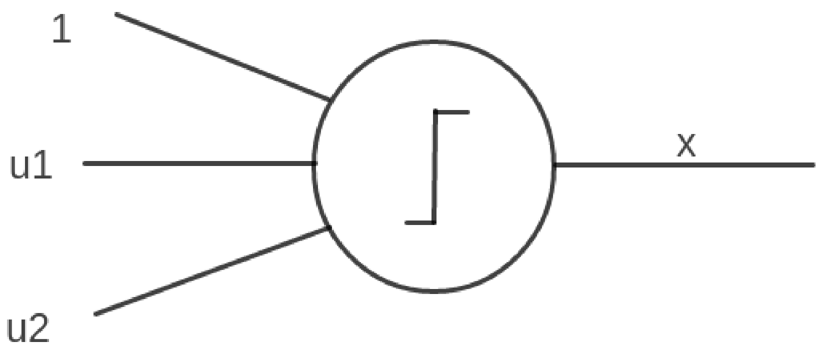
Problem 1. Design a multilayer perceptron which will learn to recognize various forms of the the letters C,L,T placed on a 3x3 grid through backpropagation algorithm.



1. Design a one layer network indicating what should be applied at the input layer and what should be expected at the output layer showing the number of neurons, the connections between them and the neurons output function.
2. repeat (a) for two layer network by adding a hidden layer

Problem 2. Consider a perceptron for which $u \in R^2$ and

$$f(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}$$



Let the desired output be 1 when elements of class A = {(1,2),(2,4),(3,3),(4,4)} is applied as input and let it be -1 for the class B = {(0,0),(2,3),(3,0),(4,2)}. Let the initial connection weights $w_0(0) = +1, w_1(0) = -2, w_2(0) = +1$ and learning rate be h = 0.5.

This perceptron is to be trained by perceptron convergence procedure, for which the weight update formula is $w(t+1) = w(t) + \eta(y^k - x^k(t))u^k$

1. (a) Mark the elements belonging to class A with x and those belonging to class B with o on input space.
   (b) Draw the line represented by the perceptron considering the initial connection weights w(0).
   (c) Find out the regions for which the perceptron output is +1 and 1
   (d) Which elements of A and B are correctly classified, which elements are misclassified and which are unclassified?
2. If u=(4,4) is applied at input, what will be w(1) ?
3. Repeat a) considering w(1).
4. If u=(4.2) is then applied at input, what will be w(2)?
5. Repeat 1) considering w(2).
6. Do you expect the perceptron convergence procedure to terminate? Why?

Problem 3. Recall the Regularized (Logistic) Cross-Entropy Loss function (minimized wrt $\mathbf{w} \in \Re^p$):

$$E(w) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_w\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2m}\|\mathbf{w}\|_2^2 \tag{1}$$

Now prove that minimizing the following dual kernelized objective[1]
(minimized wrt $\alpha \in \Re^m$) is equivalent to minimizing the regularized cross-entropy loss function:

$$E_D(\alpha) = \left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)}K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\alpha_j + \frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1+\sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\right)\right] \tag{2}$$

where, decision function $f_{\mathbf{w}}(\mathbf{x}) = \dfrac{1}{1+\displaystyle\sum_{j=1}^{m}\alpha_j K\left(\mathbf{x},\mathbf{x}^{(j)}\right)}$

Problem 4. In the class, we discussed the probabilistic binary (class) logistic regression classifier. How will you extend logistic regression probabilistic model to multiple (say K) classes? Are their different ways of extending? What is the intuition behind each? Discuss and contrast advantages/disadvantages in each.

Problem 5. Suppose you are provided a multi-layer neural network for a binary classification problem. Can you convert this network into an equivalent kernel perceptron (single node neural network only)? What will be the exact steps?

[1] `http://perso.telecom-paristech.fr/~clemenco/Projets_ENPC_files/kernel-log-regression-svm-boosting.pdf`