ECEN 649 Pattern Recognition – Spring 2016

Problem Set 4

Due on: Apr 27

1. Consider a linear discriminant $g(x) = a^t x + b$.

   (a) Use the method of Lagrange multipliers to show that the distance of a point x_0 to the hyperplane $g(x) = 0$ is given by $|g(x_0)|/||a||$.

   Solution: We need to minimize the distance $||x - x_0||$ of a point x on the hyperplane to the given point x_0. The point x is on the hyperplane if and only if $g(x) = a^T x + b = 0$. Therefore, we can find the minimum distance by solving the following optimization problem:

$$\min ||x - x_0||^2$$
$$\text{s.t. } a^T x + b = 0\,.$$

   We can do this by using the Lagrange multiplier method that we used in class for SVMs. First, we change this into an unconstrained optimization problem by introducing one Lagrange multiplier λ and coding the constraint into the objective function, which gives the functional

$$L(x, \lambda) = ||x - x_0||^2 - \lambda(a^T x + b)\,. \tag{1}$$

   We need to minimize this with respect to the unconstrained variable x; therefore, we set $\partial L/\partial x = 0$, and get

$$2(x - x_0) - \lambda a = 0 \;\Rightarrow\; x - x_0 = \frac{\lambda}{2} a \;\Rightarrow\; x = x_0 + \frac{\lambda}{2} a\,. \tag{2}$$

   At this point, we could proceed as in the case of SVMs: substitute (2) back into (1) to eliminate x and obtain a quadratic functional that depends only on λ, solve this dual problem to find λ, and substitute into (2). It turns out that in this case it is easier to employ the following equivalent approach: use the constraint $a^T x + b = 0$ in combination with (2) to solve for λ:

$$a^T \left(x_0 + \frac{\lambda}{2} a \right) + b = 0 \;\Rightarrow\; a^T x_0 + b + \frac{\lambda}{2} ||a||^2 = 0 \;\Rightarrow\; \lambda^* = -\frac{2 g(x_0)}{||a||^2}$$

   and then substitute into (2) to get the optimal minimum distance :

$$(x - x_0)^* = \frac{\lambda^*}{2} a = -\frac{g(x_0)}{||a||^2}\, a \;\Rightarrow\; ||x - x_0||^* = \frac{|g(x_0)|}{||a||}\,.$$

(b) Use the previous result to show that the margin in a linear SVM $g(x) = a^t x + b = 0$ is given by $1/||a||$.

Solution: If $g(x) = a^T x + b = 0$ is the solution of the linear SVM, then a point x_0 on the margin hyperplane satisfies

$$g(x_0) = a^T x_0 + b = 1\,.$$

Using the result in part (a), we obtain that the distance of x_0 to the SVM hyperplane, i.e., the margin, is given by $g(x_0)/||a|| = 1/||a||$.

2. Consider the following training data consisting of 4 points:

$$x_1 = (-1, 1), y_1 = 1,\ x_2 = (1, 1), y_2 = 1,\ x_3 = (-1, -1), y_3 = 0,\ x_4 = (1, -1), y_4 = 0.$$

(a) Run manually the perceptron algorithm for these training data, considering the initial parameters to be $a(0) = (1, 0)$ and $a_0(0) = 0$, and a fixed step length $\ell = 1$. Plot the designed perceptron classifier.

Solution: First we need to transform the vectors by adding a constant unit coordinate

$$x_1' = (1, -1, 1),\ \ x_2' = (1, 1, 1),\ \ x_3' = (1, -1, -1),\ \ x_4' = (1, 1, -1)$$

and then negating the vectors from class 0, which gives the vectors z_i for use in the perceptron algorithm:

$$z_1 = (1, -1, 1),\ \ z_2 = (1, 1, 1),\ \ z_3 = (-1, 1, 1),\ \ z_4 = (-1, -1, 1)$$

The transformed initial vector is $b(0) = (0, 1, 0)$. We start by determining the set $\mathcal{Y}_1$ of misclassified points, i.e., the z_i such that $b(0)^T z_i < 0$. This yields $\mathcal{Y}_1 = \{z_1, z_4\}$. The perceptron update is

$$b(1) = b(0) + \ell \sum_{z_i \in \mathcal{Y}_1} z_i = (0, 1, 0) + 1 \times [(1, -1, 1) + (-1, -1, 1)] = (0, -1, 2) \qquad (3)$$

We now observe that $b(1)^T z_i \geq 0$ for all z_i, that is, $\mathcal{Y}_2 = \emptyset$. Therefore, $b(1)$ is in the solution region and and the perceptron algorithm stops. From $b(1)$ we get the solution

$$a = (-1, 2),\ \ a_0 = 0$$

The corresponding perceptron classifier is depicted in Figure 1.

(b) By assuming the same initial parameters, find the condition on the fixed step length ℓ that allows the perceptron algorithm to find a solution after a single iteration.

Solution: The perceptron algorithm stopped after just one iteration. For this to happen, we need $b(1)$ to be in the solution region. From (3), we can see that for a general step length ℓ, we have

$$b(1) = (0, 1, 0) + \ell \times [(1, -1, 1) + (-1, -1, 1)] = (0, 1 - 2\ell, 2\ell)$$

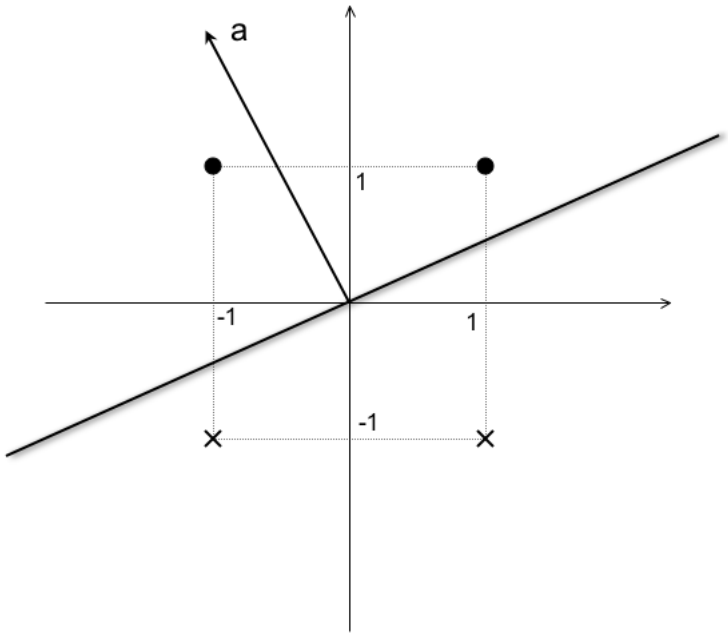


Figure 1: Perceptron Classifier.

We need to check that $b(1)^T z_i \geq 0$ for all z_i. For z_2 and z_3, this is true for any ℓ. For z_1 and z_4, we get a single condition:

$$b(1)^T z_1 = b(1)^T z_4 = -(1-2\ell) + 2\ell = 4\ell + 1 \geq 0 \;\Rightarrow\; \ell \geq \frac{1}{4}$$

Therefore, a short step length ($\ell < 0.25$) will lead to slower convergence (Note: this is not necessarily true in general; is some cases, too large a step length can lead to overshooting and slow convergence).

(c) By assuming the same initial parameters, and a fixed step length $\ell = 1$, show what happens with the perceptron algorithm if the training data are instead:

$$x_1 = (-1,-1), y_1 = 1,\; x_2 = (1,1), y_2 = 1,\; x_3 = (-1,1), y_3 = 0,\; x_4 = (1,-1), y_4 = 0.$$

Can you fix this by changing ℓ or the initial parameters?

Solution: In this case, the data for the perceptron algorithm becomes:

$$z_1 = (1,-1,-1),\;\; z_2 = (1,1,1),\;\; z_3 = (-1,1,-1),\;\; z_4 = (-1,-1,1)$$

With initial vector $b(0) = (0,1,0)$, we have $\mathcal{Y}_1 = \{z_1, z_4\}$ as before. The perceptron update is

$$b(1) = b(0) + \ell \sum_{z_i \in \mathcal{Y}_1} z_i = (0,1,0) + 1 \times [(1,-1,-1) + (-1,-1,1)] = (0,-1,0)$$

We observe now that $b(1)^T z_i < 0$ for z_2 and z_3, while z_1 and z_4 are correctly classified, that is, $\mathcal{Y}_2 = \{z_2, z_3\}$. The next update is:

$$b(2) = b(1) + \ell \sum_{z_i \in \mathcal{Y}_2} z_i = (0, -1, 0) + 1 \times [(1, 1, 1) + (-1, 1, -1)] = (0, 1, 0)$$

Therefore, $b(2) = b(0)$ and we are in a cycle! The perceptron algorithm will cycle indefinitely between these two vectors, alternately misclassifying z_1, z_4 and z_2, z_3.

Now, it is known (e.g. see Thm 5.1 in DHS) that the perceptron algorithm with fixed step length must converge to a solution if the problem is linearly separable. Therefore, the logical implication is that the present problem must be non-linearly separable. In fact, this problem is the XOR problem, the minimal non-linearly separable problem in two-dimensional space. Obviously, the perceptron algorithm cannot reach a solution for this data, regardless of any changes to the initial vector or step length.

3. Show that the polynomial kernel $K(x, y) = (1 + x^T y)^p$ satisfies Mercer's condition.

Solution: Since $x^T y = y^T x$, K is symmetric: $K(x, y) = (1 + x^T y)^p = (1 + y^T x)^p = K(y, x)$. We need to show that K is positive semi-definite, i.e., for any square-integrable function g

$$\int g^2(x)\, dx < \infty$$

we need to show that

$$\int K(x, y) g(x) g(y)\, dx\, dy \geq 0 \tag{4}$$

Using the binomial theorem, we can expand $K(x, y) = (1 + x^T y)^p$ as

$$K(x, y) = \sum_{k=0}^{p} \binom{p}{k} (x^T y)^k$$

Therefore, to satisfy (4), it suffices to show that

$$\int (x^T y)^k g(x) g(y)\, dx\, dy \geq 0, \quad \text{for } k = 0, \ldots, p \tag{5}$$

Now, using the multinomial theorem we can expand $(x^T y)^k = (x_1 y_1 + \ldots + x_d y_d)^k$ as

$$(x^T y)^k = \sum_{\substack{k_1, \ldots, k_d \geq 0 \\ k_1 + \ldots + k_d \leq k}} \frac{k!}{k_1! \ldots k_d!} x_1^{k_1} y_1^{k_1} \ldots x_d^{k_d} y_d^{k_d}$$

Therefore, in order to satisfy (5), and thus (4), it suffices to show that

$$\int x_1^{k_1} y_1^{k_1} \ldots x_d^{k_d} y_d^{k_d} g(x) g(y)\, dx\, dy \geq 0, \quad \text{for } k_i \geq 0$$

But this follows from factorization of the integral:

$$\begin{aligned}
&\int x_1^{k_1} y_1^{k_1} \dots x_d^{k_d} y_d^{k_d} g(x) g(y)\, dx\, dy \\
&= \int x_1^{k_1} \dots x_d^{k_d} g(x)\, dx \int y_1^{k_1} \dots y_d^{k_d} g(y)\, dy \\
&= \left(\int x_1^{k_1} \dots x_d^{k_d} g(x)\, dx \right)^2 \geq 0
\end{aligned}$$

4. Consider a network with l and m neurons in two hidden layers (see Figure 30.3 in DGL). This network is specified by:

$$\zeta(x) = c_0 + \sum_{i=1}^{l} c_i \xi_i(x)$$

where $\xi_i(x) = \sigma(\phi_i(x))$, for $i = 1, \dots, l$, and

$$\phi_i(x) = b_{i0} + \sum_{j=1}^{m} b_{ij} \upsilon_j(x)\,, \quad i = 1, \dots, l$$

where $\upsilon_j(x) = \sigma(\chi_j(x))$, for $j = 1, \dots, m$, and

$$\chi_j(x) = a_{j0} + \sum_{k=1}^{d} a_{jk} x_k\,, \quad j = 1, \dots, m$$

Determine the backpropagation algorithm updates for the coefficients c_i, b_{ij}, and a_{jk}. Find the backpropagation equation(s) for this problem.

Solution: By including bias units (neurons with constant unit output) where necessary, we can write the output of the network as:

$$\zeta(x) \;=\; \sum_{i=0}^{l} c_i \sigma \left[\sum_{j=0}^{m} b_{ij} \sigma \left(\sum_{k=0}^{d} a_{jk} x_k \right) \right]$$

with criterion function

$$J(w) = \frac{1}{2}[t - \zeta(x)]^2$$

where w is the weight vector of coefficients c_i, b_{ij}, a_{jk}, and t is the target

$$t = \begin{cases} 1, & y = 1 \\ -1, & y = 0 \end{cases}$$

For the weights c_i, the problem is essentially equal to what was done in class for the one-hidden-layer:

$$\frac{\partial J}{\partial c_i} \;=\; \frac{\partial J}{\partial \zeta} \frac{\partial \zeta}{\partial c_i} = -[t - \zeta(x)]\, \xi_i(x)$$

so that $\Delta c_i = \ell\, \delta^o \xi_i(x)$, where

$$\delta^o \equiv -\frac{\partial J}{\partial \zeta} = t - \zeta(x) \tag{6}$$

For the weights b_{ij}, we have

$$\frac{\partial J}{\partial b_{ij}} = \frac{\partial J}{\partial \phi_i}\,\frac{\partial \phi_i}{\partial b_{ij}} = \frac{\partial J}{\partial \phi_i}\,\upsilon_j(x)$$

where

$$\frac{\partial J}{\partial \phi_i} = \frac{\partial J}{\partial \zeta}\,\frac{\partial \zeta}{\partial \xi_i}\,\frac{\partial \xi_i}{\partial \phi_i} = -\delta^o\, c_i\, \sigma'(\phi_i(x))$$

so that $\Delta b_{ij} = \ell\, \delta_i^s \upsilon_j(x)$, where

$$\delta_i^s \equiv -\frac{\partial J}{\partial \phi_i} = \sigma'(\phi_i(x))\, c_i\, \delta^o \tag{7}$$

For the weights a_{jk}, we have

$$\frac{\partial J}{\partial a_{jk}} = \frac{\partial J}{\partial \chi_j}\,\frac{\partial \chi_j}{\partial a_{jk}} = \frac{\partial J}{\partial \chi_j}\,x_k$$

where

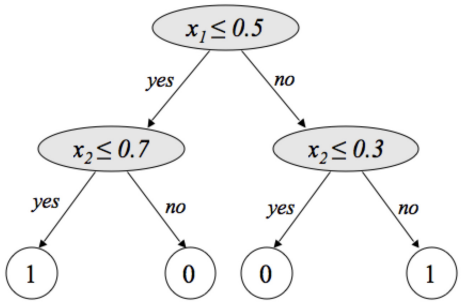
$$\frac{\partial J}{\partial \chi_j} = \sum_{i=1}^{l} \frac{\partial J}{\partial \phi_i}\,\frac{\partial \phi_i}{\partial \upsilon_j}\,\frac{\partial \upsilon_j}{\partial \chi_j} = -\sum_{i=1}^{l} \delta_i^s\, b_{ij}\, \sigma'(\chi_j(x))$$

so that $\Delta a_{jk} = \ell\, \delta_j^f x_k$, where

$$\delta_j^f \equiv -\frac{\partial J}{\partial \chi_j} = \sigma'(\chi_j(x)) \sum_{i=1}^{l} b_{ij}\, \delta_i^s \tag{8}$$

Equations (6), (7), and (8) are the backpropagation equations for this problem. Given the neuron outputs $\zeta(x)$, $\phi_i(x)$ and $\chi_j(x)$, they allow one to compute δ^o, δ_i^s, and δ_j^f, in this order (from the output backwards), and this in turns allows one to find the weight updates Δc_i, Δb_{ij}, and Δa_{jk}.

5. Consider the simple CART classifier in R^2 depicted below, consisting of three splitting nodes and four leaf nodes.



Design an equivalent two-hidden-layer neural network with threshold sigmoids, with three neurons in the first hidden layer and four neurons in the second hidden layer (note the correspondence with the numbers of splitting nodes and leaf nodes).

Solution: The CART classifier corresponds to an arrangement classifier, which can be implemented by a neural network with threshold nonlinearities in the first hidden layer. Each hyperplane split correponds to one perceptron in the first hidden layer. In addition, we can implement the desired labels for each leaf region by means of a second hidden layer, where each leaf region corresponds to one perceptron in the second hidden layer, as we will show below.

Perceptron i in the first hidden layer implements a hyperplane decision of the form

$$a_{i1}x_1 + a_{i2}x_2 + a_{i0} \geq 0$$

It is easy to check that the top split node "$x_1 \leq 0.5$" can be implemented by perceptron 1 with coefficients:

$$a_{11} = -1,\ a_{12} = 0,\ a_{10} = \boxed{0.5}$$

Similarly, the split node "$x_2 \leq 0.7$" can be implemented by perceptron 2 with coefficients:

$$a_{21} = 0,\ a_{22} = -1,\ a_{20} = \boxed{0.7}$$

and the split node "$x_2 \leq 0.3$" can be implemented by perceptron 3 with coefficients:

$$a_{31} = 0,\ a_{32} = -1,\ a_{30} = \boxed{0.3}$$

Let us number the leaf nodes L_1 to L_4 from left to right in the tree diagram. Let y_i be the output of perceptron i in the first hidden layer. It is easy to see that we have

$$\begin{aligned}
x \in L_1 &\Leftrightarrow (y_1, y_2, y_3) = (1, 1, \times) \\
x \in L_2 &\Leftrightarrow (y_1, y_2, y_3) = (1, 0, \times) \\
x \in L_3 &\Leftrightarrow (y_1, y_2, y_3) = (0, \times, 1) \\
x \in L_4 &\Leftrightarrow (y_1, y_2, y_3) = (0, \times, 0)
\end{aligned}$$

where "$\times$" indicates "don't care." (the corresponding output can be 0 or 1.)

We assign leaf node L_i to perceptron i in the second hidden layer. We denote the output of this perceptron by z_i. Our strategy will be to have the second hidden layer implement an indicator function for the leaf nodes, that is, we want

$$x \in L_i \ \Rightarrow\ z_i = 1 \text{ and } z_j = 0, \text{ for } j \neq i \tag{9}$$

Therefore, we want the second hidden layer to implement the following Boolean function:

y_1	y_2	y_3	z_1	z_2	z_3	z_4
1	1	0	1	0	0	0
1	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	1	0	1	0	0
0	0	1	0	0	1	0
0	1	1	0	0	1	0
0	0	0	0	0	0	1
0	1	0	0	0	0	1

The values (y_1, y_2, y_3) can be seen as the vertices of a cube in R^3. Perceptron i in the second hidden layer implements a hyperplane decision of the form

$$b_{i1}y_1 + b_{i2}y_2 + b_{i3}y_3 + b_{i0} \geq 0$$

We need the hyperplane to be oriented in such a way that the two vertices for which $z_i = 1$ in the above table be on the positive side of the hyperplane and all other vertices be on the negative side.

By visualizing the hypercube in 3-D space, it is not difficult to come up with the necessary hyperplanes. For perceptron 1, we have

$$y_1 + y_2 \geq 1.5 \;\Rightarrow\; b_{11} = 1,\; b_{12} = 1,\; b_{13} = 0,\; b_{10} = -1.5$$

Similarly, for perceptron 2,

$$y_1 - y_2 \geq 0.5 \;\Rightarrow\; b_{21} = 1,\; b_{22} = -1,\; b_{23} = 0,\; b_{20} = -0.5$$

For perceptron 3,

$$y_3 - y_1 \geq 0.5 \;\Rightarrow\; b_{31} = -1,\; b_{32} = 0,\; b_{33} = 1,\; b_{30} = -0.5$$

For perceptron 4,

$$y_3 + y_1 \leq 0.5 \;\Rightarrow\; b_{41} = -1,\; b_{42} = 0,\; b_{43} = -1,\; b_{40} = 0.5$$

The output weights c_i must be determined in such a way that

$$x \in L_i \;\Rightarrow\; \begin{cases} c_0 + \sum_{j=1}^{4} c_j z_j \geq 0, & \text{if } \psi(L_i) = 1 \\ c_0 + \sum_{j=1}^{4} c_j z_j < 0, & \text{if } \psi(L_i) = 0 \end{cases}$$

where $\psi(L_i)$ denotes the decision over L_i. From (9), it is clear that this can be accomplished by letting $c_0 = 0$ and assigning $c_i = 1$ or $c_i = -1$ according to whether $\psi(L_i) = 1$ or $\psi(L_i) = 0$, respectively. Therefore, we have

$$c_0 = 0,\; c_1 = 1,\; c_2 = -1,\; c_3 = -1,\; c_4 = 1$$

We have now specified all the weights of the desired neural network.