CNS 187 Neural Computation - Problem Sheet 5

Ling Li, ling@cs.caltech.edu

November 6, 2000

5.1 Single Layer Networks

Let

$$s_i^\mu = \sum_{k=0}^{n} w_{ik} x_k^\mu$$

denote the linear portion (weighted sum) of a neuron. Thus the output is

$$y_i^\mu = \sigma(s_i^\mu).$$

1. From

   $$E = \frac{1}{2} \sum_{\mu=1}^{N} \sum_{i=1}^{m} \left(d_i^\mu - y_i^\mu\right)^2,$$

   we have

   $$\frac{\partial E}{\partial w_{ik}} = \sum_{\mu=1}^{N} \frac{\partial E}{\partial y_i^\mu} \frac{dy_i^\mu}{ds_i^\mu} \frac{ds_i^\mu}{dw_{ik}} = -\sum_{\mu=1}^{N} \left(d_i^\mu - y_i^\mu\right) \frac{dy_i^\mu}{ds_i^\mu} x_k^\mu. \tag{1}$$

   From

   $$\sigma(s) = \frac{1}{1+e^{-s}},$$

   we get

   $$\sigma'(s) = \frac{e^{-s}}{(1+e^{-s})^2} = \frac{1}{1+e^{-s}} \left(1 - \frac{1}{1+e^{-s}}\right) = \sigma(s)(1-\sigma(s)).$$

   Thus (1) becomes

   $$\frac{\partial E}{\partial w_{ik}} = -\sum_{\mu=1}^{N} \left(d_i^\mu - y_i^\mu\right) y_i^\mu \left(1 - y_i^\mu\right) x_k^\mu.$$

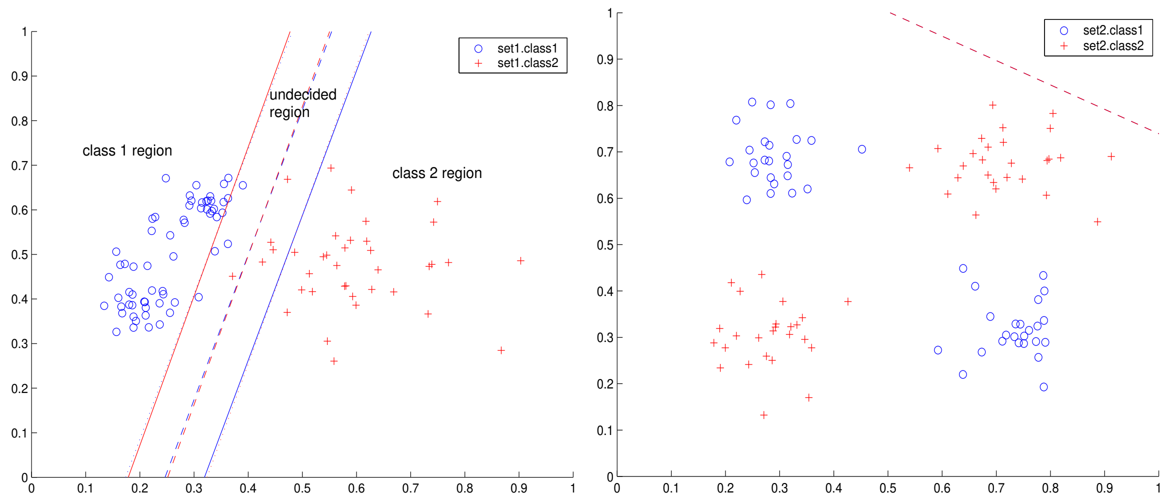
   So the learning rule that performs gradient descent could be

   $$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = \eta \sum_{\mu=1}^{N} \left(d_i^\mu - y_i^\mu\right) y_i^\mu \left(1 - y_i^\mu\right) x_k^\mu.$$

2. `dEdW = ((Y-D) .* Y .* (1-Y)) * Xp';`

3. Use Matlab code

```
[W,eh,ch] = bprop1(X, rand(2,3)*2-1, D, 0.1, 1000);
```



(a) `set1` (b) `set2`

Figure 1: Patterns and decision boundaries for set 1 and set 2.

to train the network with data `set1`. `ch` shows that 91% of the patterns can be correctly classified. The patterns and decision boundaries are shown in Figure 1(a).

The decision boundaries of the two output neurons almost coincide with each other, however, with opposite direction. Thus, one neuron will output '0.8–1' in the up-left region and output '0–0.2' in the down-right region, and the other neuron will do in a opposite way. So, the up-left region of the plot is labeled 'class 1', the down-right region is labeled 'class 2', and the region between the '0.2' and '0.8' boundaries is undecided.

4. The patterns and decision boundaries for `set2` are in Figure 1(b) or Figure 2. None of the patterns (0%) can be correctly classified. The wide-range plot in Figure 2 shows that the

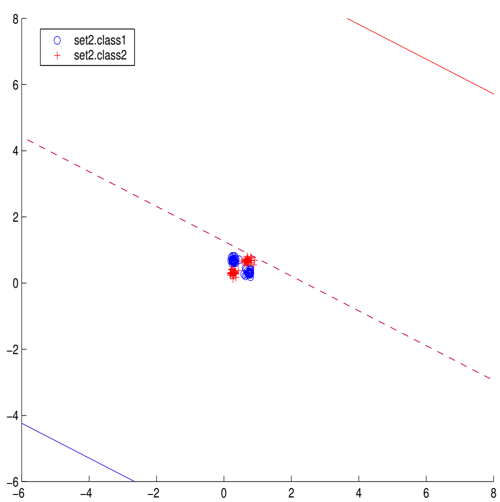


Figure 2: Boundaries of the single-layer network for `set2`.

outputs for all patterns are near 0.5, not preferable to either class.

The reason is that, like the XOR problem, this data set is not linearly separable. However, a single layer network can only classify data that can be linearly separated. Increasing `eta` or `iters` can not help.

5.2 Multilayer networks

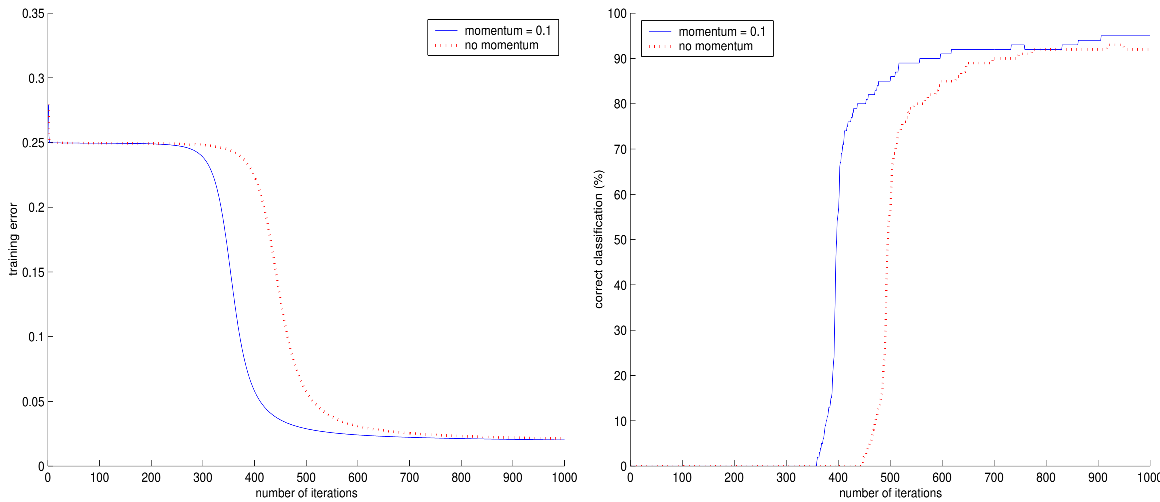
1. Use
$$E^\mu = \frac{1}{2}\sum_{k=1}^{m}\left(d_k^\mu - y_k^\mu\right)^2$$
and thus
$$E = \sum_{\mu=1}^{N} E^\mu.$$

   - `Xp = [-ones(1,size(X,2)); X]; Z = 1 ./ (1 + exp(-W*Xp));`
   - `Zp = [-ones(1,size(Z,2)); Z]; Y = 1 ./ (1 + exp(-V*Zp));`
   - `eY = (Y - D) .* Y .* (1 - Y)` is an $m \times N$ matrix;
   $$\frac{\partial E^\mu}{\partial s_k^\mu} = -\left(d_k^\mu - y_k^\mu\right) y_k^\mu \left(1 - y_k^\mu\right).$$
   - `eZ = (V(:,2:size(V,2))' * eY) .* Z .* (1 - Z)` is an $h \times N$ matrix;
   $$\frac{\partial E^\mu}{\partial r_j^\mu} = \sum_{k=1}^{m} \frac{\partial E^\mu}{\partial s_k^\mu}\frac{\partial s_k^\mu}{\partial z_j^\mu}\frac{dz_j^\mu}{dr_j^\mu} = \sum_{k=1}^{m}\frac{\partial E^\mu}{\partial s_k^\mu} V_{kj} z_j^\mu \left(1 - z_j^\mu\right).$$
   - `dEdV = eY * Zp'` is an $m \times (h+1)$ matrix;
   $$\frac{\partial E}{\partial V_{kj}} = \sum_\mu \frac{\partial E^\mu}{\partial s_k^\mu}\frac{\partial s_k^\mu}{\partial V_{kj}} = \sum_\mu \frac{\partial E^\mu}{\partial s_k^\mu} z_j^\mu.$$
   - `dEdW = eZ * Xp'` is an $h \times (n+1)$ matrix.
   $$\frac{\partial E}{\partial W_{ji}} = \sum_\mu \frac{\partial E^\mu}{\partial r_j^\mu}\frac{\partial r_j^\mu}{\partial W_{ji}} = \sum_\mu \frac{\partial E^\mu}{\partial r_j^\mu} x_i^\mu.$$

2. Using two-layer network for data set `set1` and `set2`, I got better results: 98% for `set1` and 95% for `set2` in one run.

   Plots in Figure 3 are training errors and percentages of correctly classified patterns for `set2`, with or without momentum. The one with momentum

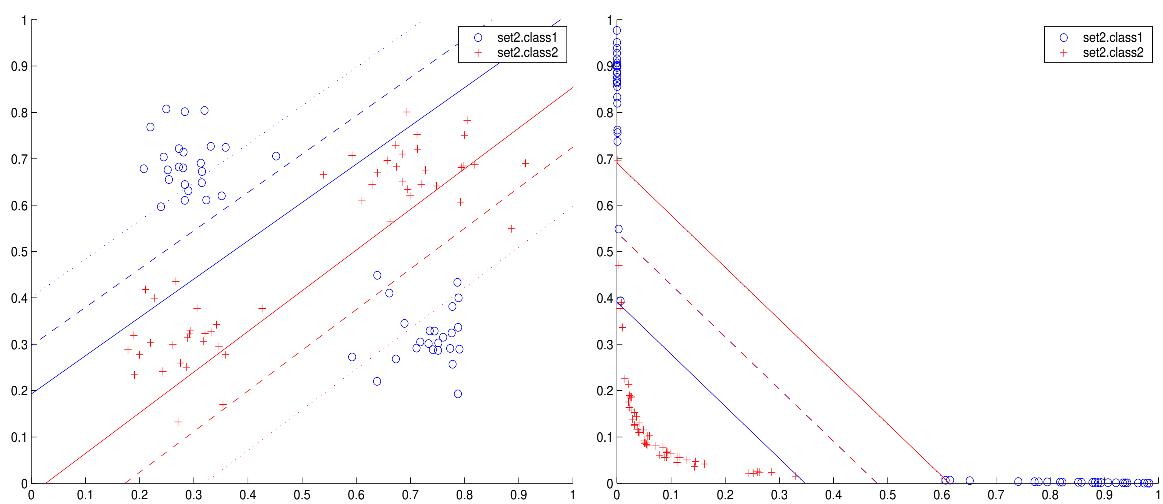
   The learning theory says that the complexity of learning model is an essential factor to the performance of the learning. The single layer network is simple, and effective, for the linear separable problems. However, the learning task of classifying `set2` is more difficult, and a more complicate model, such as a two-layer network whose decision regions can be convex open or closed[Lippmann, 1992], is needed.



(a) Training errors of the two-layer network

(b) Correct percentages of the two-layer network

Figure 3: Training with and without momentum.



(a) The boundaries of the hidder lay

(b) The boundaries of the output layer

Figure 4: Boundaries of two-layer neural network.

3. We can use the codes below to plot the boundaries of the hidden lay and the output layer in Figure 4.

```
load set2.dat;
X = set2(:, 1:2)'; D = set2(:, 3:4)';
[W,V,E,C] = bprop2(X, rand(2,3)*2-1, rand(2,3)*2-1, D, 0.1, 1000);
R = 1 ./ (1 + exp(-W * [-ones(1, size(X,2)); X]);
p1 = find(D(1, :)); p2 = find(D(2,:));
boundary(X(:,p1), X(:,p2), W); boundary(R(:,p1), R(:,p2), V);
```

From the plots, we can find that the outputs of the hidden layer for most patterns in class 1 are like $(0, 0.8)$ or $(0.8, 0)$, which can be seen in Figure 4(b); and the outputs for most patterns in class 2 are around $(0.1, 0.1)$. Thus the patterns are (nearly) linearly separable in the output layer. So, this two-layer network can correctly classify most of the patterns.

References

[Lippmann, 1992] Richard P. Lippmann, "An Introduction to Computing with Neural Nets" in Clifford Lau (Ed), *Neural Networks – Theoretical Foundations and Analysis*. IEEE Press:New York, 1992.