Statistical Machine Learning
Assignment 4

Inez Wijnands (s4149696) & Guido Zuidhof (s4160703)

Radboud University Nijmegen

18/01/2015

The entire code listing is included in the zip-file. The listings shown here are merely code snippets.

1 Neural networks

1. Since all the weights have the same update function, if all the weights are 0, after the updates they will still remain the same as the other weights. This way you don't break the symmetry when backpropagating.

2. First, we take the weighted sum of the input x:

$$\begin{aligned}
a_j &= \sum_i w_{ji} x_i + b_j^{(1)} x_0 \\
a_1 &= w_1^{(1)} \cdot x_1 + b_1^{(1)} \cdot x_0 = 0.5 + 1 = 1.5 \\
a_2 &= w_2^{(1)} \cdot x_1 + b_2^{(1)} \cdot x_0 = 0.05 + 0 = 0.05 \\
a_3 &= w_3^{(1)} \cdot x_1 + b_3^{(1)} \cdot x_0 = -0.5 + 1 = 0.5
\end{aligned}$$

Where the input x_i given in the assignment $\in \{x_1, t_1\} = \{0.5, 0.25\}$, and w_{ji} is also given $\in w^{(1)} = [1, 0.1, -1]$. Now we calculate the activation z_j of unit j:

$$\begin{aligned}
z_j &= h(a_j) = \tanh(a_j) = \frac{e^{a_j} - e^{-a_j}}{e^{a_j} + e^{-a_j}} \\
z_1 &= \tanh(1.5) = 0.90515 \\
z_2 &= \tanh(0.05) = 0.04996 \\
z_3 &= \tanh(0.5) = 0.46212
\end{aligned}$$

Then, we calculate the output using the linear activation function given in the assignment:

$$\begin{aligned}
y &= \sum_j w_j^{(2)} z_j + b_1^{(2)} \\
&= w_1^{(2)} z_1 + w_2^{(2)} z_2 + w_3^{(2)} z_3 + b_1^{(2)} \\
&= -1 \cdot 0.90515 + 0.1 \cdot 0.04996 + -1 \cdot 0.46212 + 2 \\
&= -0.90515 + 0.004996 - 0.46212 + 2 \\
&= 0.63773
\end{aligned}$$

Second, we backpropagate the error $\delta = (y - t)$ to establish the derivatives $\frac{\partial E}{\partial w_{ji}}$, using a standard sum-of-squares function (with $K = 1$ since there is only one output value):

$$\begin{aligned}
\delta_k &= y_k - t_k \\
E_n &= \frac{1}{2} \sum_1^K \delta_k^2 \\
&= \frac{1}{2} \cdot (0.63773 - 0.25)^2 = \frac{1}{2} \cdot 0.37227^2 = \frac{1}{2} \cdot 0.15033 = 0.07517
\end{aligned}$$

Then we backpropagate these δ's for the hidden units:

$$
\begin{aligned}
\delta_j &= h'(a_j)\sum_1^K w_{kj}\delta_k \\
&= (1-h(a_j)^2)\sum_1^K w_{kj}\delta_k \\
&= (1-z_j^2)\sum_1^K w_{kj}\delta_k \\
\delta_1 &= (1-z_1^2)\cdot w_1^{(2)}\cdot 0.37227 \\
&= (1-0.90515^2)\cdot -1\cdot 0.37227 \\
&= (1-0.81930)\cdot -0.37227 \\
&= 0.18070\cdot -0.37227 = -0.06727 \\
\delta_2 &= (1-z_2^2)\cdot w_2^{(2)}\cdot 0.37227 \\
&= (1-0.04996^2)\cdot 0.1\cdot 0.37227 \\
&= (1-0.00250)\cdot 0.03723 \\
&= 0.99750\cdot 0.03723 = 0.03714 \\
\delta_3 &= (1-z_3^2)\cdot w_3^{(2)}\cdot 0.37227 \\
&= (1-0.46212^2)\cdot -1\cdot 0.37227 \\
&= (1-0.21355)\cdot -0.37227 \\
&= 0.78645\cdot -0.37227 = -0.29277
\end{aligned}
$$

Third, the derivatives with respect to the first-layer and second-layer weights are given by:

$$
\begin{aligned}
\frac{\partial E_n}{\partial w_{ji}^{(1)}} &= \delta_j x_i = \Delta w_j^{(1)} \\
\Delta w_1^{(1)} &= -0.06727\cdot 0.5 = -0.03364 \\
\Delta w_2^{(1)} &= \ \ 0.03714\cdot 0.5 = \ \ 0.01857 \\
\Delta w_3^{(1)} &= -0.29277\cdot 0.5 = -0.14639
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E_n}{\partial w_{kj}^{(2)}} &= \delta_k z_j = \Delta w_j^{(2)} \\
\Delta w_1^{(2)} &= 0.37277\cdot 0.90515 = 0.33741 \\
\Delta w_2^{(2)} &= 0.37227\cdot 0.04996 = 0.01862 \\
\Delta w_3^{(2)} &= 0.37277\cdot 0.46212 = 0.17226
\end{aligned}
$$

And to update the weights, we use learning rate $\eta = 0.5$:

$$
\begin{aligned}
w_1^{(1)} &:= w_1^{(1)} - \eta\Delta w_1^{(1)} = 1 - 0.5\cdot -0.03364 = 1.01682 \\
w_2^{(1)} &:= w_2^{(1)} - \eta\Delta w_2^{(1)} = 0.1 - 0.5\cdot 0.01857 = 0.09072 \\
w_3^{(1)} &:= w_3^{(1)} - \eta\Delta w_3^{(1)} = -1 - 0.5\cdot -0.14639 = -0.92681 \\
w_1^{(2)} &:= w_1^{(2)} - \eta\Delta w_1^{(2)} = -1 - 0.5\cdot 0.33741 = -1.16871 \\
w_2^{(2)} &:= w_2^{(2)} - \eta\Delta w_2^{(2)} = 0.1 - 0.5\cdot 0.01862 = 0.09159 \\
w_3^{(2)} &:= w_3^{(2)} - \eta\Delta w_3^{(2)} = -1 - 0.5\cdot 0.17226 = -1.08613
\end{aligned}
$$

Last, to check whether or not the updated network has improved the output (i.e. has a smaller error), we recalculate the first step given the same input.

We recalculate the weighted sum of the input with the new weights and use the activation function to obtain z_j:

$$
\begin{aligned}
z_1 &= \tanh w_1^{(1)}\cdot x_1 + b_1^{(1)}\cdot x_0 = \tanh(1.01682\cdot 0.5 + 1\cdot 1) = \tanh(1.50841) = 0.90666 \\
z_2 &= \tanh w_2^{(1)}\cdot x_1 + b_2^{(1)}\cdot x_0 = \tanh(0.09072\cdot 0.5 + 0\cdot 1) = \tanh(0.04536) = 0.04533 \\
z_3 &= \tanh w_3^{(1)}\cdot x_1 + b_3^{(1)}\cdot x_0 = \tanh(-0.92681\cdot 0.5 + 1\cdot 1) = \tanh(0.53660) = 0.49041
\end{aligned}
$$

We recalculate the output with the new activations:

$$\begin{aligned}
y &= \sum_j w_j^{(2)} z_j + b_1^{(2)} \\
&= w_1^{(2)} z_1 + w_2^{(2)} z_2 + w_3^{(2)} z_3 + b_1^{(2)} \\
&= -1.16871 \cdot 0.90666 + 0.09159 \cdot 0.04533 + -1.08613 \cdot 0.49041 + 2 \\
&= 0.41188
\end{aligned}$$

If we repeat the second step, we obtain the new error:

$$\begin{aligned}
\delta_k &= y_k - t_k \\
E_n &= \frac{1}{2} \sum_1^K \delta_k^2 \\
&= \frac{1}{2} \cdot (0.41188 - 0.25)^2 = \frac{1}{2} \cdot 0.16188^2 = \frac{1}{2} \cdot 0.02621 = 0.01310
\end{aligned}$$

The sum-of-squares error of the output is smaller than the previous error. Thus, it has improved.

3. If you only use linear activation functions for the hidden layer nodes, the network cannot model nonlinear combinations of the inputs. If the classification problem is not linearly separable, no matter how many hidden nodes you have, you can still only obtain linear separations and thus not solve this problem very well.

4. Due to the fairly high noise in the data we can not be very certain that it will converge. The sequential procedure is quite susceptible to noise, as for instance sequential points with high noise with the same sign may make for weights moving far away from an optimum. The batch procedure is less susceptible to this problem, as the noise gets averaged out for all 50 points.

   The weights of either approach do not have to end up with the same final weight distribution.

2 Neural network regression

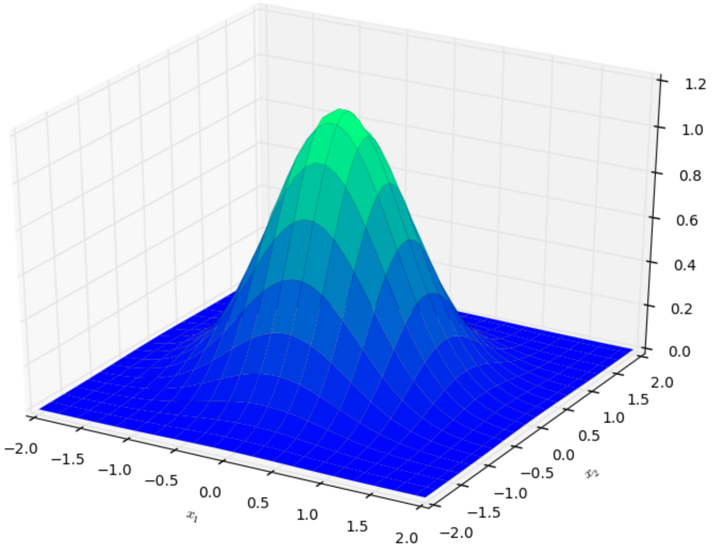
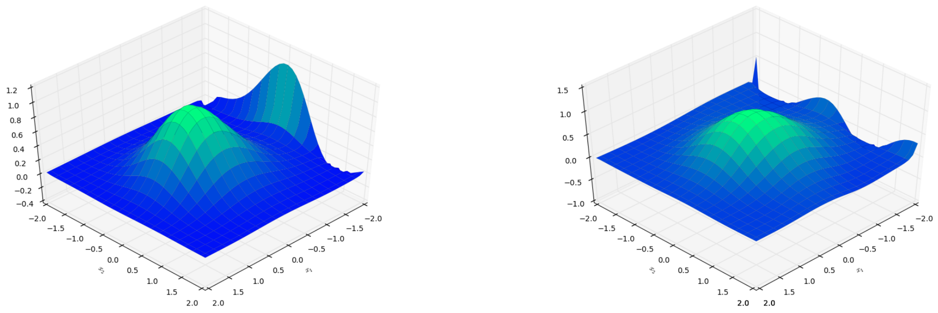


Figure 2.1: Isotropic 2D Gaussian $y = 3 \cdot \mathcal{N}(\boldsymbol{x} | \boldsymbol{0}, \frac{2}{5}\boldsymbol{I}_2)$.

1.

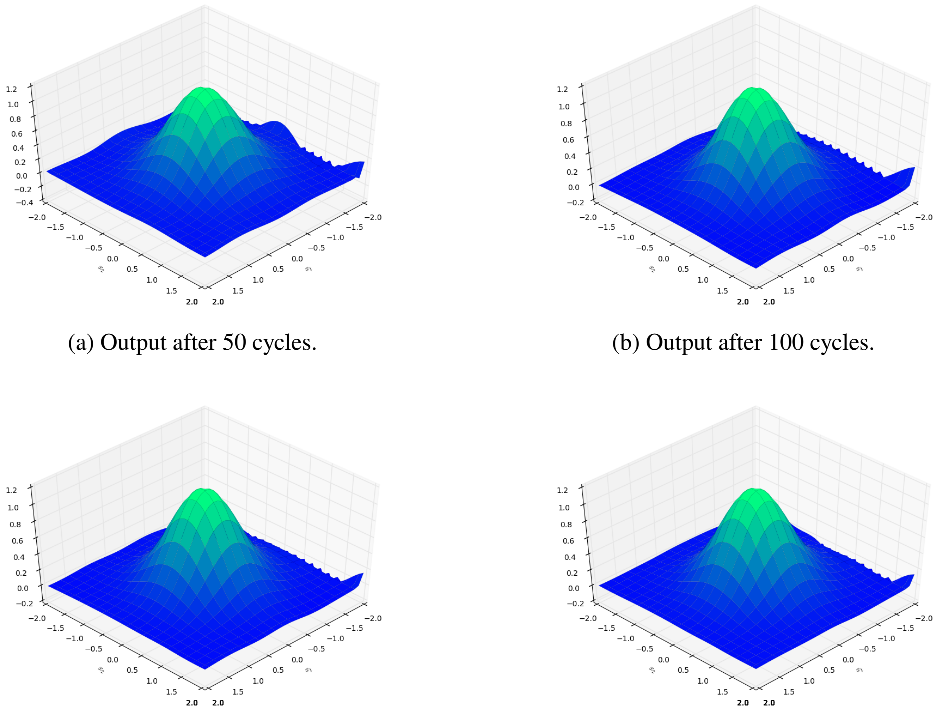
2. We have used the activation functions of the previous exercise. We also introduced a bias node in the input layer and in the hidden layer, similar to the previous exercise. The weights for these biases are initialized randomly, the same as the other weights in our network.



(a) Initial output.

(b) Output of the second cycle.

Figure 2.2: 2D graphs of initial output of network.



(a) Output after 50 cycles.

(b) Output after 100 cycles.

(c) Output after 200 cycles.

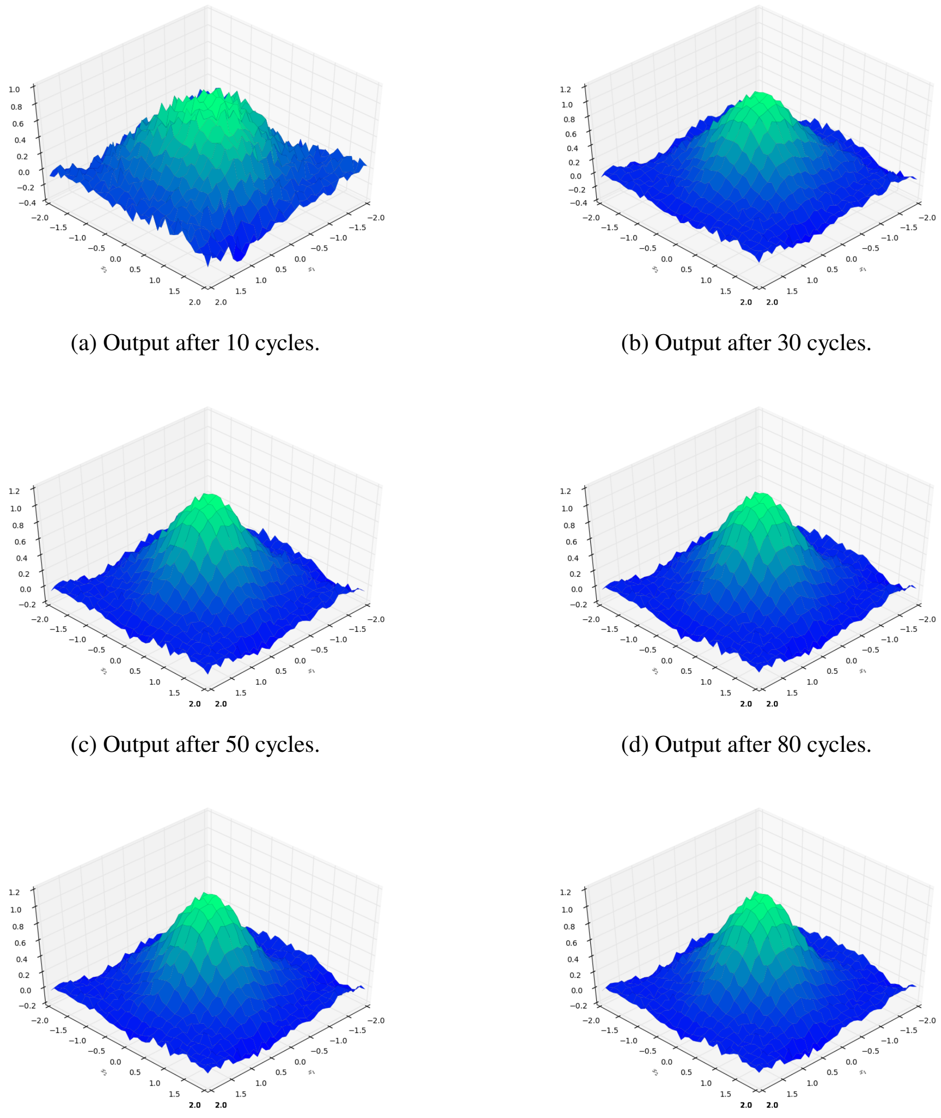
(d) Output after 500 cycles.

Figure 2.3: 2D graphs of the output of network.

3. In Figure 2.2 and Figure 2.3 is shown that the initial outputs still contain multiple peaks. After 100 cycles the output is nearly converged, only really small changes are visible. The plot already resembles the Gaussian density shown in Figure 2.1. Although the expected wobbles are not present.

4. In Figure 2.4 is shown what the output looks like when the dataset is permuted in a random order. Convergence is established after around 50-80 cycles, so a bit faster than in 2.3. However, we do see the wobbles in these plots, while we expected to see them as well in 2.3.

   Second, we trained the network using 16 hidden nodes instead of 8, but we did not see much difference.



(a) Output after 10 cycles.

(b) Output after 30 cycles.

(c) Output after 50 cycles.

(d) Output after 80 cycles.

(e) Output after 100 cycles.

(f) Output after 500 cycles.

Figure 2.4: 2D graphs of the output of network where the data set is in a randomized order

However, the network takes more time to train. When we tried 4 hidden nodes, the network performed quite badly, as can be seen in Figure 2.5. The ouput does not change after more cycles, thus convergence is fast.

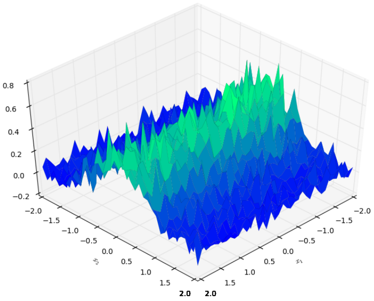
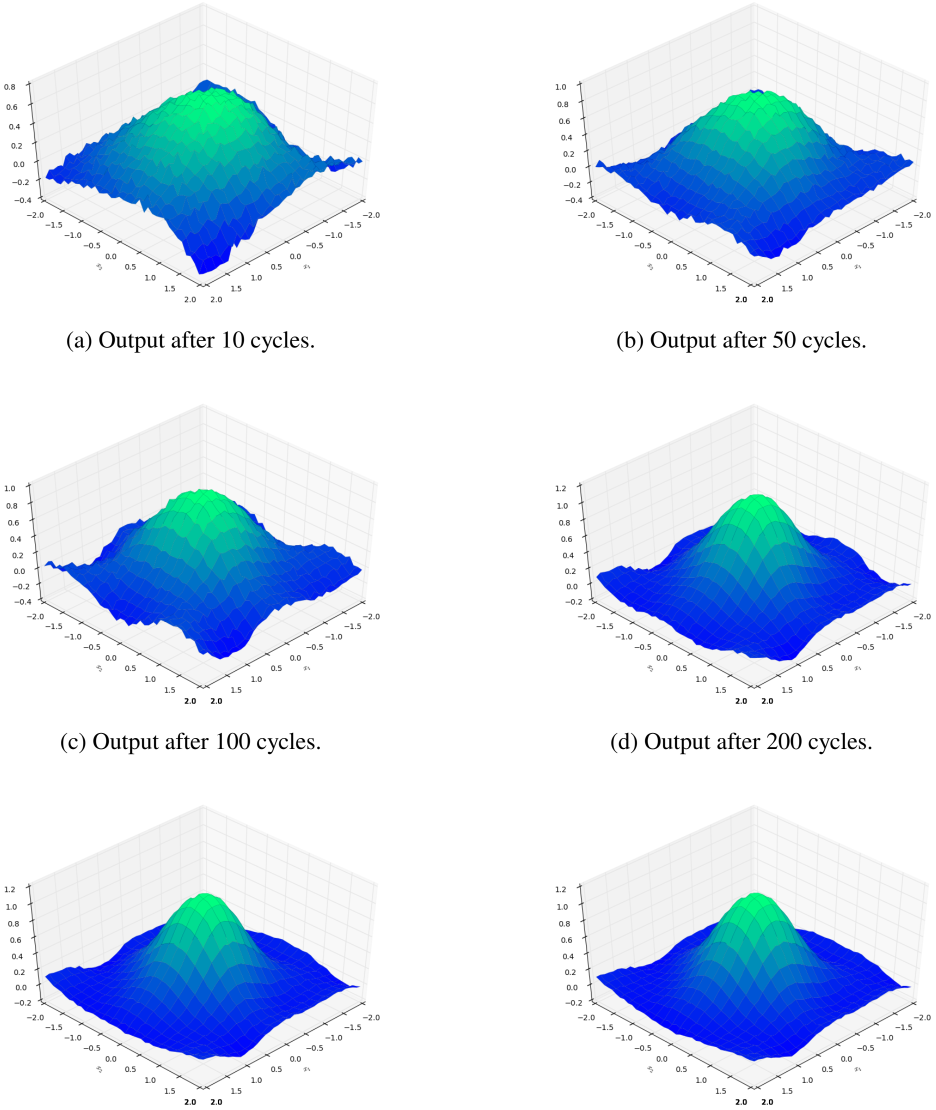


Figure 2.5: 2D graphs of the output of network after 500 cycles with a randomized data order, 4 hidden nodes

When we change the learning rate to 0.01 instead of 0.1, we get a smoother plot. Convergence is slower, around 250 cycles, as is shown in Figure 2.6.



(a) Output after 10 cycles.

(b) Output after 50 cycles.

(c) Output after 100 cycles.

(d) Output after 200 cycles.

(e) Output after 300 cycles.

(f) Output after 500 cycles.

Figure 2.6: 2D graphs of the output of network where the data set is in a randomized order, with $\eta = 0.01$

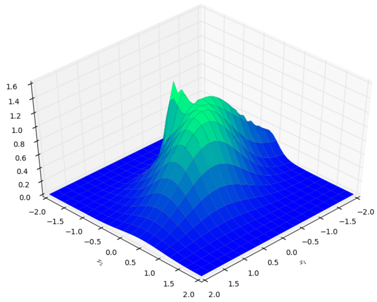
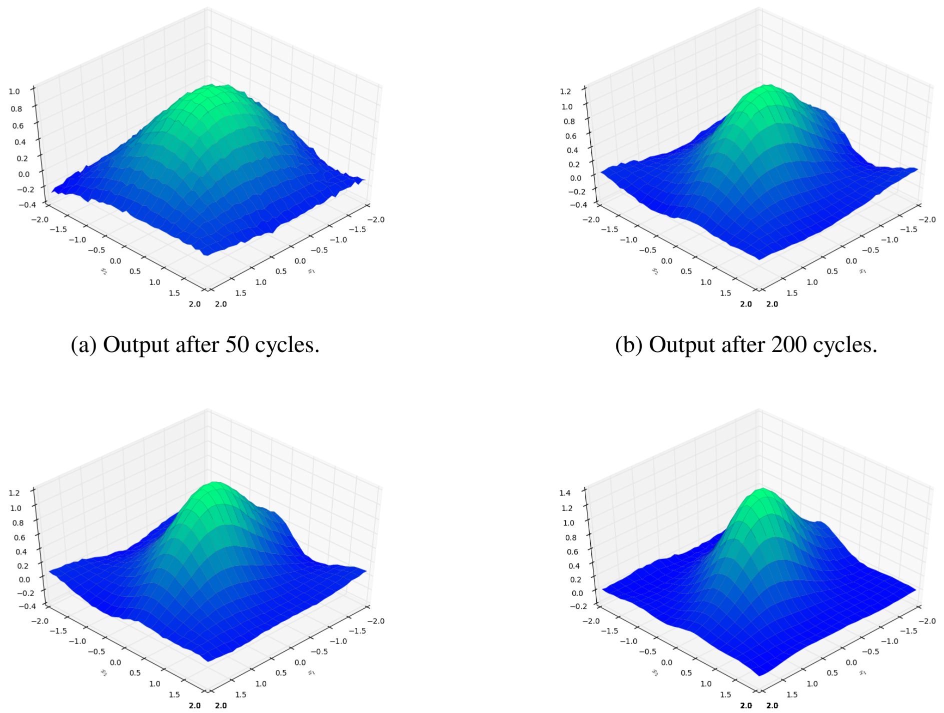


Figure 2.7: 2D graph of the target probability density function of the real data

5.

6. We used 60 hidden nodes, and a learning rate of 0.001, for 2000 cycles. The output is almost converged after around 200-300 cycles, only changing a bit towards 2000 cycles. We also tried a learning rate of 0.0001, to see what happens. This does not converge within 2000 cycles, it is too slow.



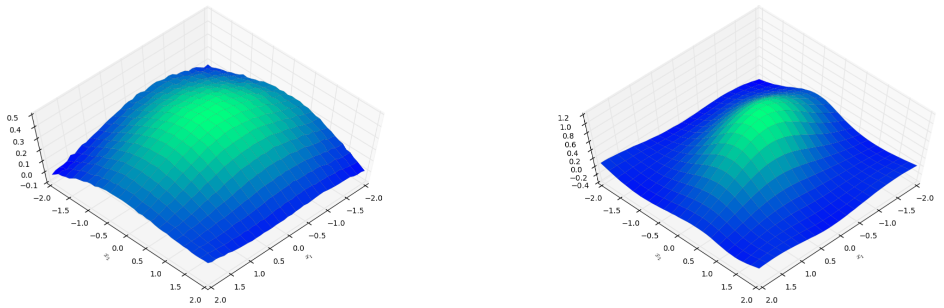
(a) Output after 50 cycles.

(b) Output after 200 cycles.

(c) Output after 300 cycles.

(d) Output after 2000 cycles.

Figure 2.8: 2D graphs of the output of network where the data set is in a randomized order, with $\eta = 0.001$ and 60 hidden nodes.



(a) Output after 50 cycles.

(b) Output after 2000 cycles.

Figure 2.9: 2D graphs of the output of network where the data set is in a randomized order, with $\eta = 0.0001$ and 80 hidden nodes

3 EM and doping

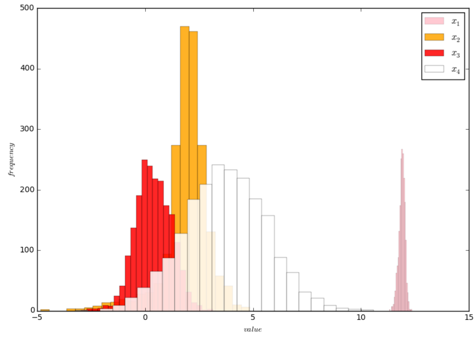
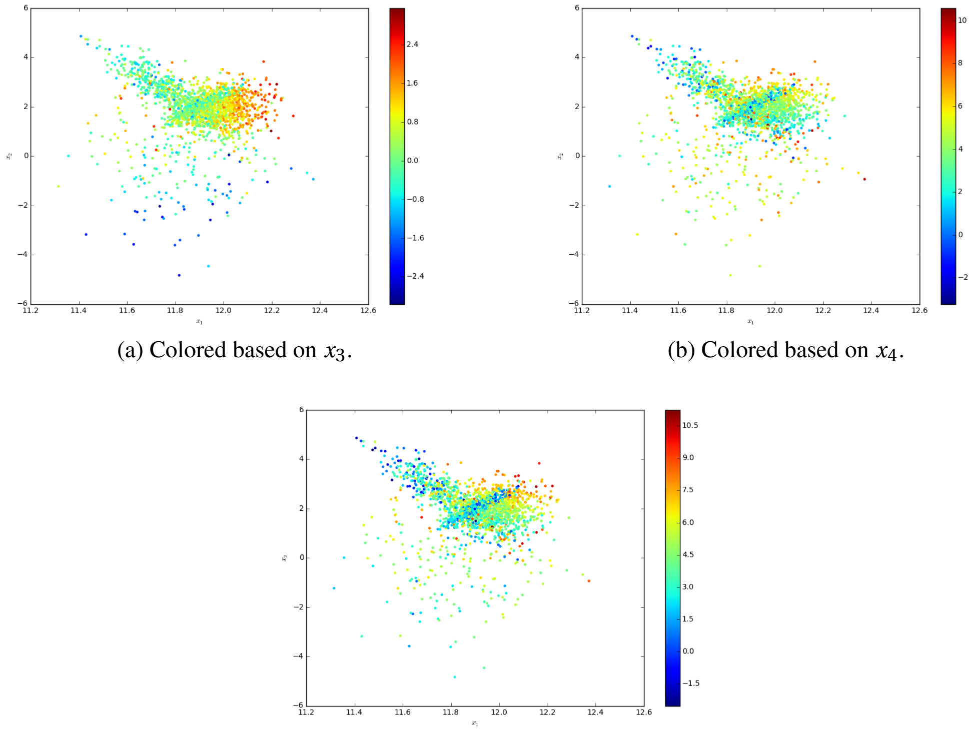


Figure 3.1: Histogram of the data, different colorings for different x_i



(a) Colored based on x_3.

(b) Colored based on x_4.

(c) Colored based on $x_3 + x_4$.

Figure 3.2: Scatter plot of data points on axes x_1 and x_2.

1. The histograms in Figure 3.1 show the distribution of each variable. We think there are 3-4 classes, as far as we can tell from the scatter plots in Figure 3.2. In Figure 3.3 our estimation of the clusters is shown. We chose these clusters because these are groups of similar sample size, with different densities but the same distance between data points within each group. In the smallest group, the most dense, there is a strong positive correlation between x_1 and x_2, and we believe this is the group of athletes that have substance X in their blood.

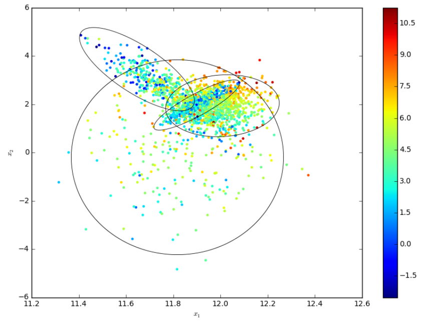


Figure 3.3: Figure 3.2(c) with our estimation of the class distributions drawn over it.

2. The EM algorithm is divided in the following steps:

   (a) Set initial values: K, π_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and evaluate the initial value for the log likelihood.

   (b) **E step**:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

(c) **M step:** Re-estimate parameters:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k}\sum_1^N \gamma(z_{nk})\boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_1^N \gamma(z_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

where

$$N_k = \sum_1^N \gamma(z_{nk})$$

(d) Evaluate the log likelihood:

$$\ln p(\boldsymbol{X}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}) = \sum_1^N \ln\left\{\sum_1^K \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right\}$$

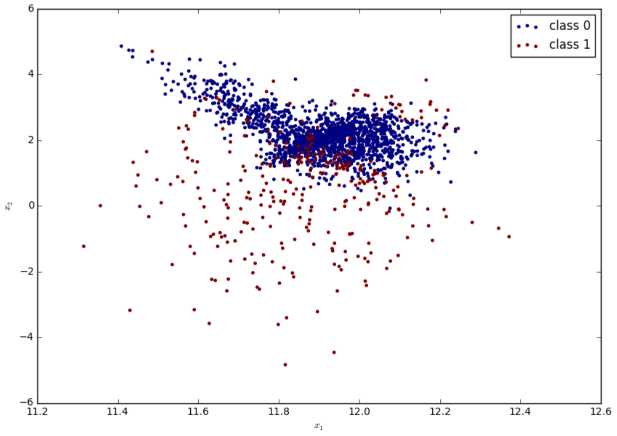


Figure 3.4: Scatter plot of data points colored according to most probable component, $K = 2$.

3. The correlation coefficients:

$$\rho_{12} = \frac{\text{cov}[x_1, x_2]}{\sqrt{\text{var}[x_1]\text{var}[x_2]}}$$

of the components are -0.5244 for class 0 and 0.1338 for class 1. None of these show the characteristic positive correlation for $\{x_1, x_2\}$. Different initializations do not alter the classes much, all cases we got similar clusters.

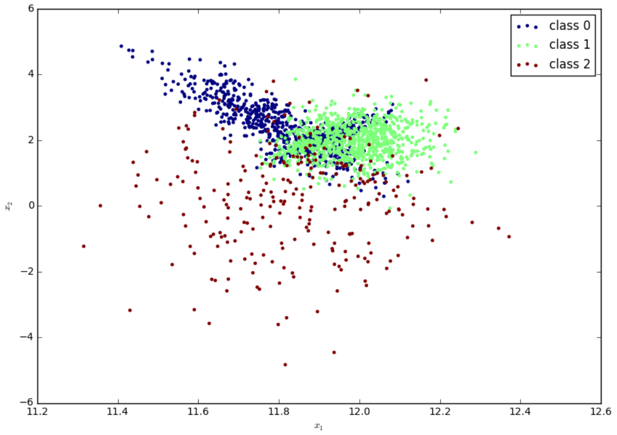


Figure 3.5: Scatter plot of data points colored according to most probable component, $K = 3$.

4. The correlation coefficients of the components are 0.0005 for class 0, 0.1146 for class 1 and -0.7181 for class 2. None of these show the characteristic positive correlation for $\{x_1, x_2\}$. The correlation coefficients

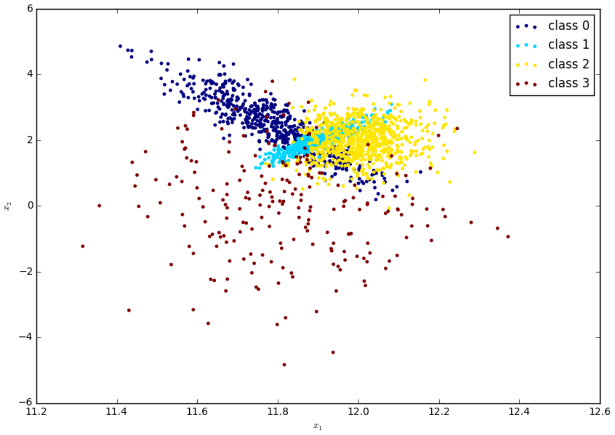
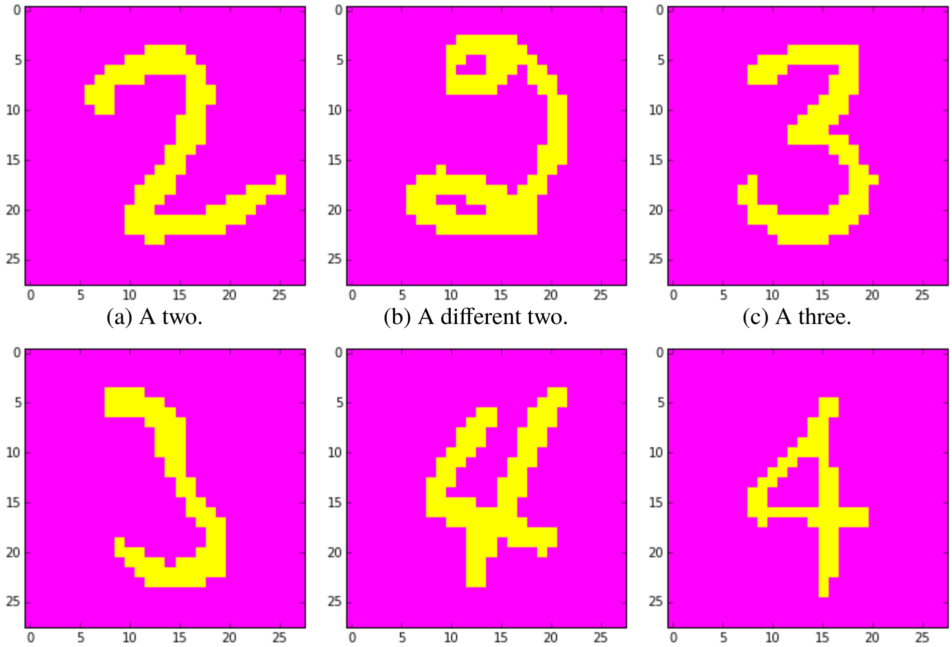


Figure 3.6: Scatter plot of data points colored according to most probable component, $K = 4$.

of the components are -0.9102 for class 0, 0.9102 for class 1, 0.0495 for class 2 and -0.1077 for class 3. Class 1 displays the characteristic positive correlation for $\{x_1, x_2\}$. There are 453 data points in class 1, which corresponds to 22.65% and thus around 1 in 5 users. We deem the rumoured 1-in-5 estimate for users of X credible.

5. Subject C is classified in class 1, which corresponds to the group of users of X. Subjects A and D are both classified in class 2, which is the largest group of users. Subject B is classified in class 0, the class with the strong negative correlation. We calculated the likelihoods for every subject, and found that while A, C and D had small probabilities for three classes but a high probability for the class they were put in, and corresponding with this also a likelihood of around 0.1-0.2, subject B was substantially different. Subject B had small probabilities for all classes, and a likelihood of 0.0000009, incredibly small, and was thus hard to classify. This is why we conclude that subject B is most likely the fraud.

4 Handwritten digit recognition



(a) A two.

(b) A different two.

(c) A three.

(d) A sloppy written three.

(e) A four.

(f) A different four.

Figure 4.1: Examples of handwritten digits in the dataset, binary images displayed in a happy springy colormap.

1. We would need class prototypes that represent the different ways of writing the same number. As shown in Figure 4.1, this might prove to be difficult, since some people write a 2 with a small loop at the bottom, or even at the top, while others do not. The number 3 can be really round, or more like a straight banana-shape. The number 4 can be either open or closed at the top. These different handwritings may complicate dividing the images into classes. However, with the right prototypes, we think at least most of the numbers can be classified correctly, because there are also more consistent differences.

2.