Solutions to the exam in Neural Networks and Learning Systems - TBMI26 Exam 2013-03-14

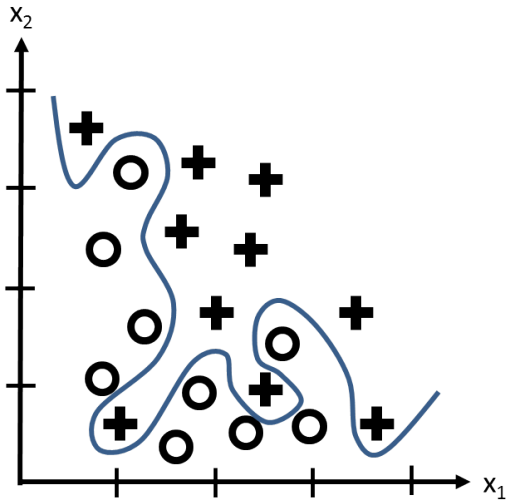
Part 1

1. If $\text{sign}(\mathbf{w}^T\mathbf{x}) > 0$, then class 1, otherwise class 2 (or vice versa).
2. In classification we want to learn to predict a discrete class label. In regression we want to learn to predict a continuous variable, for example a temperature, probability, stock price etc.
3. 5 times, each time with 80 examples used for training and 20 for evaluation.
4. Advantages are that k-nearest neighbor requires no training and it is easy to implement. Disadvantages is that one must store all training examples and it takes a long time to classify if the training data set is large, as the distance to all training examples must be calculated.
5. Gradient ascent follows the gradient direction of the cost function to find its (local) maximum. Gradient descent follows the negative gradient direction to find the (local) minimum.
6. The falsely labeled training example can become an outlier which will gain too much weight in the AdaBoost training because it will be misclassified often by the weak classifiers. Thus, this training example may ruin the classifier.
7. The number of clusters k.
8. A policy is usually represented as a lookup-table, i.e., for each state there is an action defined.
9. The cost function maximized in PCA is $\text{var}(\mathbf{w}^T\mathbf{x})$ where $\mathbf{x}$ represents the training data. As we can maximize this function just by increasing the magnitude of $\mathbf{w}$, the size of $\mathbf{w}$ must be constrained, for example so that $\|\mathbf{w}\| = 1$.
10. The accuracy is usually calculated as the number of correctly classified examples divided with the total number of training examples.

Part 2

11. With supervised learning, one could train a classifier (linear classifier, neural network, Support Vector Machine, etc) to classify if a person has the disease or not based on the measured parameters. With unsupervised learning techniques (PCA, ICA, clustering, etc), one can try to discover relationships among the measured parameters. This could for example be used identify parameters that carry the same information so that the number of parameters that must be measured can be reduced.

12. The image below shows an example of a classifier that has overtrained. Linear classifiers have few parameters and can only produce linear class boundaries. They therefore do not tend to overtrain.

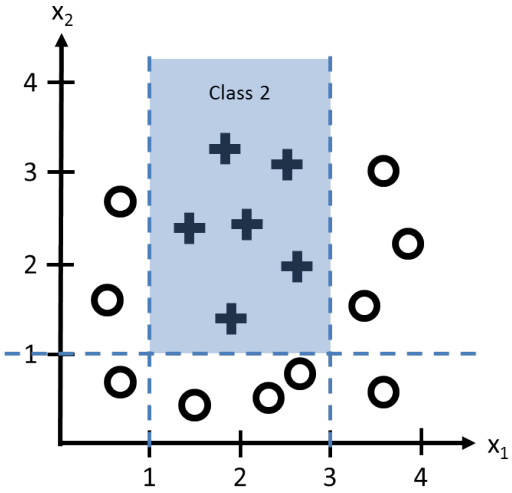


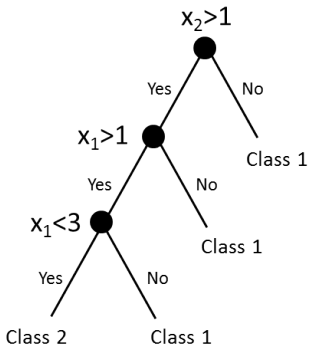
13. A kernel function defines the inner product $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2)$ in the new feature space. Thus, $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ specifies the feature space by defining how distances and angles are measured, instead of explicitly stating the mapping function $\Phi(\mathbf{x})$.

    The distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ in the new feature space is

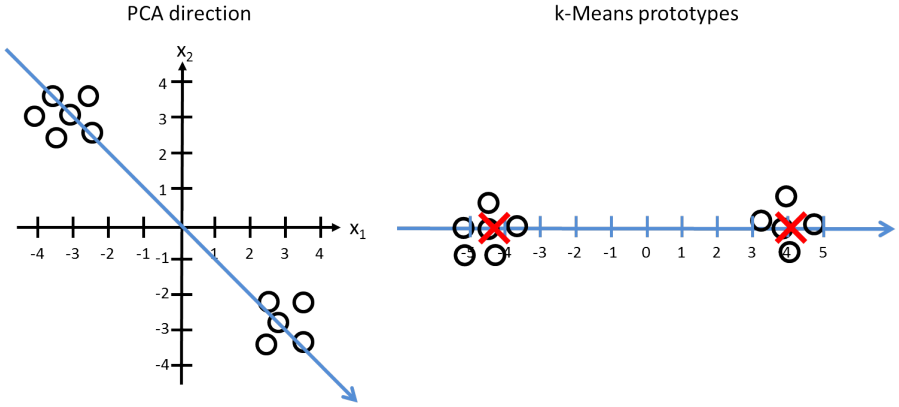
$$\begin{aligned}
\|\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)\| &= \sqrt{(\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2))^T (\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2))} \\
&= \sqrt{\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_1) - 2\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2) + \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_2)} \\
&= \sqrt{\kappa(\mathbf{x}_1, \mathbf{x}_1) - 2\,\kappa(\mathbf{x}_1, \mathbf{x}_2) + \kappa(\mathbf{x}_2, \mathbf{x}_2)} \\
&= \sqrt{1 - 2\,e^{-\frac{1}{4}} + 1} = 0.6651
\end{aligned}$$

14. The figure below shows one possible solution. Note that there are other similar solutions.





15. The left image below show the major PCA direction for dimensionality reduction. The right image shows the prototype vector positions at the center of the clusters (if we use k=2 clusters). The original clusters are centered around (3,3) and (-3,-3). After projection on the PCA direction, the clusters are centered around $\sqrt{3^2+3^2} \approx 4.2$ and -4.2, which should be the output from the k-Means algorithm



Part 3

16.
- The number of CCA solutions are limited to the smallest dimensionality between $\mathbf{x}$ and $\mathbf{y}$. The number of solutions therefore will be:
  a) 2
  b) 2
  c) 3
- The first correlation finds the strongest linear relatoinship between the two sets of feature vectors. All CCA solutions needs to be uncorrelated to each other. Since the norm of $\mathbf{w}_x$ and $\mathbf{w}_y$ does not affect the result, we choose $||\mathbf{w}_x||_2 = ||\mathbf{w}_x||_2 = 1$
  a) The first correlation: $\mathbf{w}_x = (1,1)^T$ and $\mathbf{w}_y = (1,0)^T$. The second correlation: $\mathbf{w}_x = (1,-1)^T$ and $\mathbf{w}_y = (0,1)^T$.
  b) The first correlation: $\mathbf{w}_x = (1,0)^T$ and $\mathbf{w}_y = (0,0,1)^T$. For the second correlation, there are several equal good solutions possible, one example is $\mathbf{w}_x = (0,1)^T$ and $\mathbf{w}_y = (1,0,0)^T$.
  c) The first correlation:$\mathbf{w}_x = (0,0,1)^T$ and $\mathbf{w}_y = \frac{1}{\sqrt{5}}(0,2,1)^T$. For the second and third correlation, several answers is possible.
- The canocial correlation ρ tells how much of the signal that is shared between $x = \mathbf{w}_x^T\mathbf{x}$ and $y = \mathbf{w}_y^T\mathbf{y}$.
  a) For the first correlation, $0 < |\rho| < 1$ will hold, in practice at least. In theory, you could have correlation $= 1$. For the second correlation, no signal is shared so $\rho = 0$.
  b) For the first correlation, $0 < |\rho| < 1$ will hold. For each of the possible solution for the second correlation, no signal will be shared, so $\rho = 0$.
  c) The same as above holds: For the first correlation, $0 < |\rho| < 1$ will hold. For each of the possible solutions for the second and third correlation, no signal will be shared, so $\rho = 0$.

17. a) In figure 1 the classification problem is sketched along with the initial weights. In figure 1 we have chosen one of these lines as an solution example. We get $\alpha_1 = \frac{1}{2}\ln\frac{1-\epsilon_1}{\epsilon_1} = \frac{1}{2}\ln\frac{4/5}{1/5} = \frac{\ln 4}{2}$. Now we update (decreasing) the weights of the correctly classified samples with $e^{-\alpha_1} = \frac{1}{2}$. The weight of the erroneous classified sample (upper left in the solution example) is instead increased with $e^{\alpha_1} = 2$. The sum of the weights are now $4\frac{1}{10} + \frac{2}{5} = \frac{4}{5}$. After normalizing the weights (dividing with the sum) we get the resulting weights as shown in the right part of figure 1.

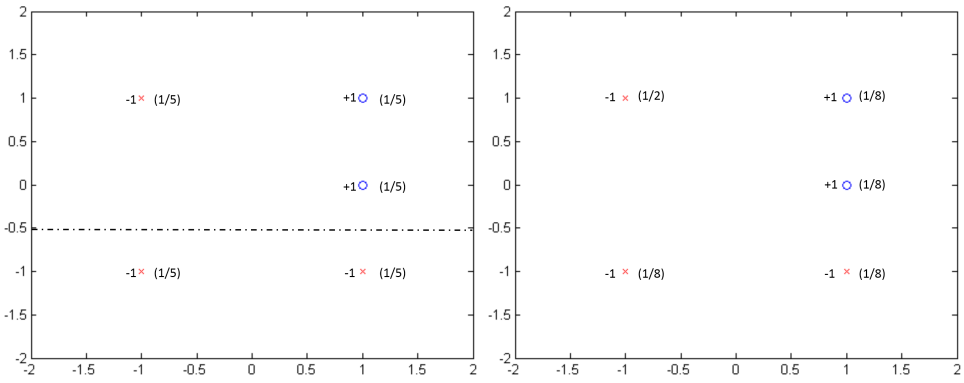


Figure 1: Classification problem 17a, weights before/after first iteration

b) In figure 2 the classification problem is sketched along with the weights from a). The smallest error possible give the error $\epsilon = \frac{1}{8}$. In figure 2 we have chosen one of these lines as an solution example. We get $\alpha_2 = \frac{\ln 7}{2}$. Now we update (decreasing) the weights of the correctly classified samples with $e^{-\alpha_1} = \frac{1}{\sqrt{7}}$. The weight of the erroneous classified sample (upper left in the solution example) is instead increased with $e^{\alpha_1} = \sqrt{7}$. The sum of the weights are now $\frac{3}{8\sqrt{7}} + \frac{1}{2\sqrt{7}} + \frac{\sqrt{7}}{8} = \frac{\sqrt{7}}{4}$. After normalizing the weights (dividing with the sum) we get the resulting weights as shown in the right part of figure 2.

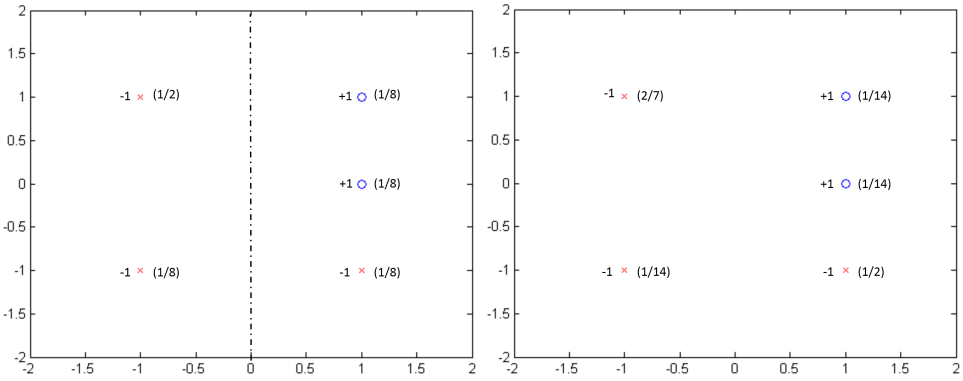


Figure 2: Classification problem 17b, weights before/after second iteration

c) The final strong classifier is illustrated in figure 3. As we can see, the AdaBoost algorithm will need more iterations in order to work properly, 2 iterations is not enough. However, after only three iterations the strong classfier will be able to classify all samples correctly.

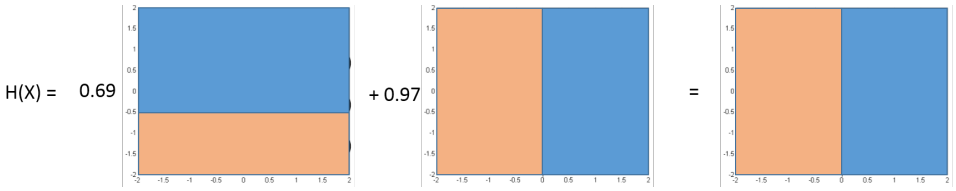


Figure 3: The final strong classifier after 2 iterations with AdaBoost

18. The formula for the optimal Q-function is given by:

$$\begin{aligned} Q^*(x,a) &= r(x,a) + \gamma V^*(g(x,\mu^*(x))) \\ &= r(x,a) + \gamma \max_b Q^*(g(x,\mu^*(x)),b) \end{aligned}$$

And the update rule:

$$\begin{aligned} \hat{Q}(x,a) &= (1-\alpha)\hat{Q}(x,a) + \alpha(r(x,a) + \gamma \hat{V}(g(x,\mu^*(x)))) \\ &= (1-\alpha)\hat{Q}(x,a) + \alpha(r(x,a) + \gamma \max_b \hat{Q}(g(x,\mu^*(x)),b) \end{aligned}$$

a) Since only the action "right" and state 1 and 3 are used the non zero Q values with $\alpha = 0.5$ after each move are:

move$\backslash\hat{Q}(x,right)$	$x=1$	$x=3$
$1 \to 3$	0	0
$3 \to 5$	0	75
$1 \to 4$	-75	75
$1 \to 4$	-112.5	75
$1 \to 3$	-18.75	75
$3 \to 5$	-18.75	112.5

When $\alpha = 1$ only the last move is of importance so the end result becomes $\hat{Q}(1,right) = 150$ and $\hat{Q}(3,right) = 150$:

move$\backslash\hat{Q}(x,right)$	$x=1$	$x=3$
$1 \to 3$	0	0
$3 \to 5$	0	150
$1 \to 4$	-150	150
$1 \to 4$	-150	150
$1 \to 3$	150	150
$3 \to 5$	150	150

b) Counting backwards we get:

$$\begin{aligned} Q^*(3,right) &= 150 + 0 \cdot \gamma = 150 \\ Q^*(2,down) &= \gamma Q^*(3,right) - 50 = 150\gamma - 50 \\ Q^*(1,up) &= \gamma Q^*(2,down) - 50 = 50(3\gamma^2 - \gamma - 1) \\ Q^*(1,right) &= 0.5 \cdot \gamma Q^*(3,right) + 0 - 0.5 \cdot 150 = 75(\gamma - 1) \end{aligned}$$

$$\begin{aligned} V^*(5) &= 0 \\ V^*(4) &= 0 \\ V^*(3) &= Q^*(3,right) = 150 \\ V^*(2) &= Q^*(2,down) = 150\gamma - 50 \\ \frac{1}{3} \le \gamma \le \frac{1}{2} \Rightarrow V^*(1) &= Q^*(1,right) = 75(\gamma - 1) \\ else \Rightarrow V^*(1) &= Q^*(1,up) = 50(3\gamma^2 - \gamma - 1) \end{aligned}$$

$V(1)^*$ is calculated by solving $Q^*(1,right) = Q^*(1,up) \to \gamma = 5/12 \pm 1/12$

19. a) We start by introducing the input matrix $\mathbf{X}$ with C rows (all features and one bias input) and N columns. We store the weights of neurons in the matrix $\mathbf{W}$ with M rows and C columns.
Now the task is to find $W_{ij}^{t+1} = W_{ij}^{t} - \gamma \frac{\delta \epsilon}{\delta W_{ij}}$
First we form functions for the different operations of the network.

$$\begin{aligned} \epsilon &= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{M}(y_i^n - u_i^n)^2 \\ u_i^n &= \sigma(x_i^n) = \frac{1}{1+e^{-z_i^n}} \\ z_i^n &= \sum_{j=1}^{C} W_{ij}X_{jn} \end{aligned}$$

Then we calculate their partial derivatives

$$\begin{aligned} \frac{\delta \epsilon}{\delta u_i^n} &= \frac{-2}{N}\sum_{n=1}^{N}(y_i^n - u_i^n) \\ \frac{\delta u_i^n}{\delta z_i^n} &= \frac{e^{-z_i^n}}{(1+e^{-z_i^n})^2} \\ \frac{\delta z_i^n}{\delta W_{ij}} &= X_{jn} \end{aligned}$$

Thus

$$\frac{\delta \epsilon}{\delta W_{ij}} = \frac{\delta \epsilon}{\delta u_i^n}\frac{\delta u_i^n}{\delta z_i^n}\frac{\delta z_i^n}{\delta W_{ij}} = \frac{-2}{N}\sum_{n=1}^{N}\frac{(y_i^n - u_i^n)e^{-z_i^n}X_{jn}}{(1+e^{-z_i^n})^2}$$

b) We can reuse most of the results from above. But since we want on-line updating we can throw away N and n and rewrite the error function. The quirk is that all outputs are depending on all weights due to the normalization. For the sake of simplicity we also set $N = 2$.

$$\begin{aligned} \epsilon &= \sum_{i=1}^{2}(y_i - u_i)^2 = \sum_{i=1}^{2}\epsilon_i \\ u_1 &= h_1 = \frac{\sigma_1}{\sigma_1 + \sigma_2} \\ u_2 &= h_2 = \frac{\sigma_2}{\sigma_1 + \sigma_2} \\ \sigma_i &= \frac{1}{1+e^{-z_i}} \\ z_i &= \sum_{j=1}^{C} W_{ij}x_j \end{aligned}$$

Using gradient descent we now want to find the gradient of every weight W_{ij}.

$$\frac{\delta \epsilon}{\delta W_{ij}} = \sum_{k=1}^{2}\frac{\delta \epsilon_k}{\delta u_k}\frac{\delta u_k}{\delta z_i}\frac{\delta z_i}{\delta W_{ij}}$$

With the partial derivatives:

$$
\begin{aligned}
\frac{\delta \epsilon_k}{\delta u_k} &= -2(y_k - u_k) \\
\frac{\delta u_1}{\delta z_1} &= \frac{\sigma_2 \frac{\delta \sigma_1}{\delta z_1}}{(\sigma_1 + \sigma_2)^2} = -\frac{\delta u_2}{\delta \sigma_1} \\
\frac{\delta u_2}{\delta z_2} &= \frac{\sigma_1 \frac{\delta \sigma_2}{\delta z_2}}{(\sigma_1 + \sigma_2)^2} = -\frac{\delta u_1}{\delta \sigma_2} \\
\frac{\delta z_i}{\delta W_{ij}} &= x_j
\end{aligned}
$$

Thus

$$
\begin{aligned}
\frac{\delta \epsilon}{\delta W_{1j}} &= \frac{-2}{(\sigma_1 + \sigma_2)^2} \sum_{k=1}^{2} (-1)^{k+1} (y_k - u_k) \sigma_2 \sigma_1 (1 - \sigma_1) x_j \\
\frac{\delta \epsilon}{\delta W_{2j}} &= \frac{-2}{(\sigma_1 + \sigma_2)^2} \sum_{k=1}^{2} (-1)^{k} (y_k - u_k) \sigma_1 \sigma_2 (1 - \sigma_2) x_j
\end{aligned}
$$