Exercises
Introduction to Machine Learning
SS 2019

Institute for Machine Learning
Dept. of Computer Science, ETH Zürich
Prof. Dr. Andreas Krause
Web: `https://las.inf.ethz.ch/teaching/introml-s19`
Email questions to:
jingwei.tang@inf.ethz.ch, kjong.lehmann@inf.ethz.ch

Series 4, April 8th, 2019
(Solutions: Neural Networks)

Solution 1:

1. $\alpha_i^{(1)} = \frac{1}{1+exp(\sum w_{ki} x_k)}$
   $\alpha_i^{(2)} = \frac{1}{1+exp(\sum w_{ki} \alpha_k^{(1)})}$
   $f = w_1^{(3)} \alpha_1^{(2)} + w_2^{(3)} \alpha_2^{(2)}$
   $L = (f-y)^2$

2. $\mathbb{E}((f-y)^2$ with $f = w_1^{(3)} \alpha_1^{(2)} r_1 + w_2^{(3)} \alpha_2^{(2)} r_2$
   $\mathbb{P}(r_i = 1) = 0.4, \quad \mathbb{P}(r_i = 0) = 0.6.$
   $\mathbb{E}(L) = (Y^2 - 2Y\mathbb{E}(f) + \mathbb{E}(f^2)) = Y^2 - 2Y\mathbb{E}(f) + \mathsf{Var}(f) + (\mathbb{E}(f))^2.$
   where $\mathbb{E}(f) = 0.4(w_1^{(3)} \alpha_1^{(2)} + w_2^{(3)} \alpha_2^{(2)})$,
   and $\mathsf{Var}(f) = 0.24((w_1^{(3)} \alpha_1^{(2)})^2 + (w_2^{(3)} \alpha_2^{(2)})^2)$.

3. Backpropagation using chain rule and the fact that $\alpha_1^{(2)}$ was dropped out:
   $\frac{dL}{dw_{21}^{(1)}} = 2(f-y) w_2^{(3)} \phi'(w_{12}^{(2)} \alpha_1^{(1)} + w_{22}^{(2)} \alpha_2^{(1)}) w_{12}^{(2)} \phi'(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3) x_2$
   Now execute forward pass according to part 1 and setting $\alpha_1^{(2)}$ to zero. Then run backward pass to compute the derivative according to the above. Update parameter according to SGD rule.

Solution 2:

1. $\mathbf{W}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, and $w_2 = \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix}$.

2. $\hat{R}(\mathcal{D}, \mathbf{W}_1, \mathbf{w}_2) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w}_2^T \sigma(\mathbf{W}_1 \mathbf{x}_i) - y_i)^2 + \|\mathbf{w}_2\|_2^2$.

3. The question asks *What is the difference ...?*, hence a clear evidence of difference is required to achieve the full points. One point is for addressing the computational complexities, and the other is for pointing out the consequences and conclusion. Right conclusions based on wrong arguments were awarded zero points.

   SGD takes a single data point to calculate the gradient, while GD takes all data points. Complexity is $\mathcal{O}(1)$ for SGD and $\mathcal{O}(n)$ for GD per step. Other formulation taking into account more iterations were accepted as correct. Instead of asymptotic notation acceptable answers includes wording specifying limiting the optimization of SGD over a subset of data points from the whole dataset.

4. Simple permutation of rows yields the results. Both $\mathbf{W}_1$ and w_2 have to be permuted. $\mathbf{W}_1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, and $w_2 = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix}$ The activation function is not known, thus this is the only solution. Two points were

awarded for full solution. One point is awarded if mistake occurs with clear intent at arriving at the solution, namely wrong permutation of weights. Zero points otherwise.

5. $x_1 \leq 0$ and $x_1 + x_2 \leq 0 : \ f(x) = 0$
   $x_1 \geq 0$ and $x_1 + x_2 \leq 0 : \ f(x) = 1/4x_1$
   $x_1 \leq 0$ and $x_1 + x_2 \geq 0$: $f(x) = 3/4(x_1 + x_2)$
   $x_1 \geq 0$ and $x_1 + x_2 \geq 0$:$f(x) = x_1 + 3/4x_2$

Solution 3:

1. We consider the following network $w = 0.5$, $w_1 = 1$ and $w_2 = 1$. We check whether the output we get is desired OR function. The network looks as follows,

$$A \vee B = \text{round}(\varphi(w_0 + w_1 A + w_2 B))$$

$$A \vee B = \text{round}(\varphi(-0.5 + A + B))$$

A	B	$A \vee B$	Network	Round
1	1	1	≈ 0.81	1
0	1	1	≈ 0.62	1
1	0	1	≈ 0.62	1
0	0	0	≈ 0.37	0

2. $w_0 = -1.5$, $w_1 = 1$ and $w_2 = 1$. We check whether the output we get is desired AND function. The network looks as follows,

$$A \wedge B = \text{round}(\varphi(w_0 + w_1 A + w_2 B))$$

$$A \wedge B = \text{round}(\varphi(-1.5 + A + B))$$

A	B	$A \wedge B$	Network	Round
1	1	1	≈ 0.62	1
0	1	0	≈ 0.37	0
1	0	0	≈ 0.37	0
0	0	0	≈ 0.18	0

3. We find the weights by choosing the weights of the first layer and optimizing over the weights of the last layer s.t. the inequalities are satisfied.

   We use a network with one hidden layer and two states

$$A \oplus B = \text{round}(\varphi(-\varphi(A + B) + 0.84\varphi(2A + 2B))$$

A	B	$A \oplus B$	Network	Round
0	0	0	0.480010659844	0
0	1	1	0.502202727467	1
1	0	1	0.502202727467	1
1	1	0	0.486027265451	0

   For sketch check Figure 1.

4. There are multiple ways to proceed to solve this exercises. One universal way is to guess, however this might become too complicated for large expressions. Another approach is to use a numerical software to fit the network to match the logic. However, this might be too brute force approach for these general problems.

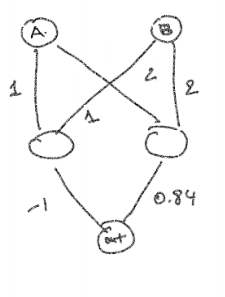

Figure 1

We provide here a third principled alternative to build such neural networks. I am not certain whether this method generalizes to any logical expressions (if you show it; good for you), however it works for this case.

Namely, we choose to transform the expression to *disjunctive normal form*, and then formulate a simplified classifier. Recall from your previous courses that any boolean algebra expression can be converted to a disjunctive normal form. In this form, the expression is cumulative OR of several expressions that contain just AND inside them. As AND, and OR operations can be modeled easily with our neural network, we can hope to perform the AND operation in one layer, and the OR operation in the second layer.

Specifically, after applying de Morgan laws, distributivity and symmetry to our expression many times, we arrive at the following minimal form,

$$(A \vee \neg B) \oplus (\neg C \vee \neg D) \iff (A \wedge C \wedge D) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg D) \vee (\neg B \wedge C \wedge D). \quad (1)$$

The detailed derivation works as follows.

$$\begin{aligned} & (A \vee \neg B) \oplus (\neg C \vee \neg D) \\ \iff & ((A \vee \neg B) \wedge (C \wedge D)) \vee ((\neg A \wedge B) \wedge (\neg C \vee \neg D)) \end{aligned}$$

applying the distributivity law $((X \vee Y) \wedge Z \iff ((X \wedge Z) \vee (Y \wedge Z))$, we get

$$\iff ((A \wedge C \wedge D) \vee (\neg B \wedge C \wedge D)) \vee ((\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg D))$$

applying associativity and symmetry, we get

$$\iff (A \wedge C \wedge D) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg D) \vee (\neg B \wedge C \wedge D).$$

Our expression decomposes to 4 expressions that are combined via logical ORs. Thus, we propose to choose 4 cells each modeling the respective expression in OR stream (AND here), and in the last cell we take OR of all of them. Implementation of AND can be done simply using the intuition from previous exercises.

(a) $(A \wedge C \wedge D)$: implements $h_1 = \varphi(\eta(A + C + D - 2.5))$

(b) $(\neg A \wedge B \wedge \neg C)$: implements $h_2 = \varphi(\eta(-A + B - C - 0.4))$

(c) $(\neg A \wedge B \wedge \neg D)$: implements $h_3 = \varphi(\eta(-A + B - D - 0.4))$

(d) $(\neg B \wedge C \wedge D)$: implements $h_4 = \varphi(\eta(-B + C + D - 1.4))$

In order to get output 0 or 1, we take $\eta \to$large, e.g.$\eta = 200$.

Subsequently to define OR, we require that at least one of these is activated, thus final layer becomes easy,

$$H(h1, h2, h3, h4) = \varphi(h1 + h2 + h3 + h4 - 0.5).$$

In full,

$$\text{output}(A, B, C, D) \quad = \quad H(h1(A, C, D), h2(A, B, C), h3(A, B, D), h4(B, C, D)). \tag{2}$$

We check the truth table,

A	B	C	D	$(A \vee \neg B) \oplus (\neg C \vee \neg D)$	Round
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

For a figure see Figure 2.

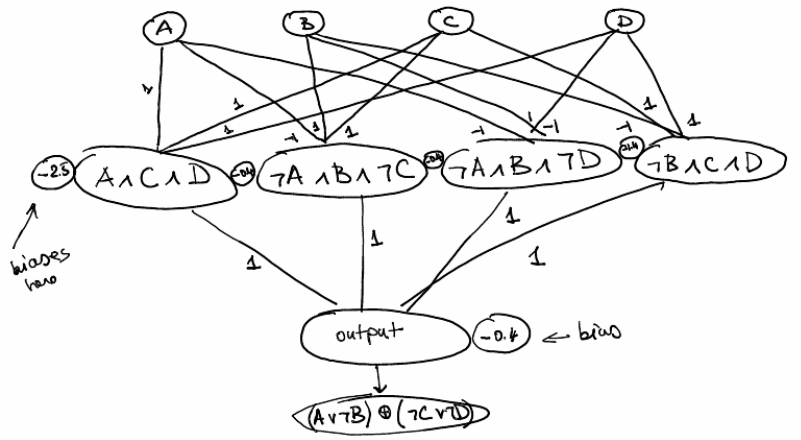

Figure 2