Exercises
Introduction to Machine Learning
Spring 2022

Institute for Machine learning
Dept. of Computer Science, ETH Zürich
Prof. Dr. Andreas Krause, Prof. Dr. Fanny Yang

Homework 3
(Kernels and Neural Networks)

For questions, please refer to Moodle.
Released on **6 April 2022**

General Instructions

- Submission of solutions is not mandatory but solving the exercises are highly recommended. The master solution will be released next week.
- Part of the exercises are available on Moodle as a quiz. These problems are marked with [✓].

Exercise 1: Kernels

(a) [✓] Given a dataset $X = \{\boldsymbol{x}_i\}_{i=1,2} = \{(-3,4),(1,0)\}$, where we denote the components of the vector $\boldsymbol{x}_i \in \mathbb{R}^2$ as $(x^{(1)}, x^{(2)})$, and a feature map $\phi(\boldsymbol{x}) = [x^{(1)}, x^{(2)}, \|\boldsymbol{x}\|] \in \mathbb{R}^3$, choose the correct Gram matrix from the following:

$$1.\ \begin{pmatrix} 50 & 2 \\ 2 & 2 \end{pmatrix} \quad 2.\ \begin{pmatrix} 50 & 4 \\ 4 & 4 \end{pmatrix} \quad 3.\ \begin{pmatrix} -50 & 2 \\ 2 & 2 \end{pmatrix} \quad 4.\ \begin{pmatrix} 50 & 2 \\ 4 & 4 \end{pmatrix}$$

(b) [✓] Consider the following definitions of the kernel $k(x,y)$. Please state, taking into account the definition of a kernel and the kernel composition rules, if they are valid kernels (True) or not (False).

$k(x,y) = \frac{1}{1-xy}$, with $x,y \in (-1,1)$.	□ True	□ False
$k(x,y) = 2^{xy}$ with $x,y \in \mathbb{N}$.	□ True	□ False
$k(x,y) = cos(x+y)$ with $x,y \in \mathbb{R}$.	□ True	□ False
$k(x,y) = cos(x-y)$ with $x,y \in \mathbb{R}$.	□ True	□ False
$k(x,y) = max(x,y)$ with $x,y \in \mathbb{R}_+$.	□ True	□ False
$k(x,y) = \frac{min(x,y)}{max(x,y)}$ with $x,y \in \mathbb{R}_+$.	□ True	□ False

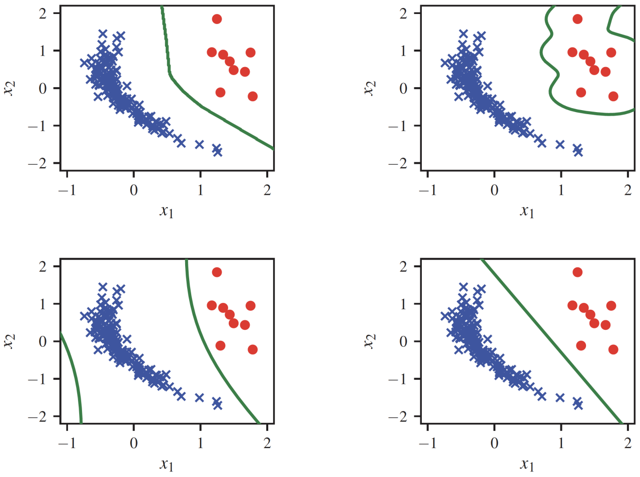
(c) [✓] Using the basic rules for kernel decomposition and assuming that $k(x,y)$ is a valid kernel, letting $f : \mathbb{R} \to \mathbb{R}$ for a) and b), $g : \mathbb{R} \to \mathbb{R}_+$ for d), $h : \mathcal{X} \to \mathbb{R}$ for e), and $\phi : \mathcal{X} \to \mathcal{X}'$ for f), state if each new kernel is a valid kernel (True) or not (False):

$k_a(x,y) = f(k(x,y))$, where $f : \mathbb{R} \to \mathbb{R}$ is a polynomial with non-negative coefficients.	□ True	□ False
$k_b(x,y) = f(k(x,y))$, if $f : \mathbb{R} \to \mathbb{R}$ is any polynomial.	□ True	□ False
$k_c(x,y) = \exp(k(x,y))$.	□ True	□ False
$k_d(x,y) = g(x)k(x,y)g(y)$, where $g : \mathbb{R} \to \mathbb{R}_+$.	□ True	□ False
$k_e(x,y) = h(x)k(x,y)h(y)$, where $h : \mathcal{X} \to \mathbb{R}$.	□ True	□ False
$k_f(x,y) = k(\phi(x), \phi(y))$, where $\phi : \mathcal{X} \to \mathcal{X}'$.	□ True	□ False

1.1 Kernelized Hinge Loss

(a) Let's assume you would like to kernelize the ℓ^2-regularized hinge loss. Knowing that the loss function can be written as $l_h(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - \boldsymbol{w}^\top \boldsymbol{x}_i y_i) + \lambda \boldsymbol{w}^\top \boldsymbol{w}$, how would the derivation of the kernelized hinge loss look like?

(b) [✓] Consider the classification problem with two classes, which is illustrated by circles and crosses in the plots below. In each of the plots, one of the following classification methods has been used, and the resulting decision boundary is shown:



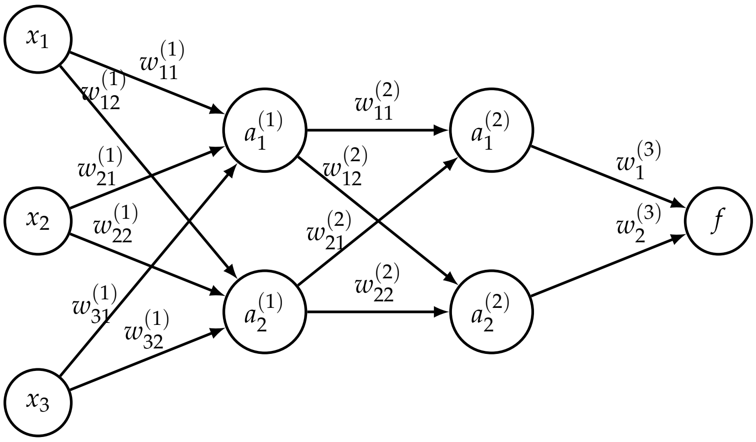
Assign each approach to the corresponding plot:

1. Linear SVM
2. Kernelized SVM (Polynomial kernel of order 2)
3. Kernelized SVM (Gaussian kernel)
4. Neural Network (1 hidden layer with 20 rectified linear units)

Exercise 2: Neural Networks

2.1 Grade Prediction

Xiaoming designed the following neural network to predict his grades in the exams. He bases the predictions on his degree of being nervous (x_1), his mood (x_2), and an indicator of the weather on the exam day (x_3). In the hidden layers, he uses a sigmoid function $\sigma(x) = 1/(1+e^{-x})$. In the output layer, no activation function is used. Similarly to many other regression tasks, he uses L^2 as the loss function: $L = (y - f)^2$.



By attending and completing the first "Introduction to Machine Learning" exam, Xiaoming collected one training example (x_1, x_2, x_3) with the corresponding grade (y). Help him to write down the sequence of computations for the forward-pass, and compute the loss by answering the following questions.

(a) [☑] What is $a_i^{(1)}$?

1. $\dfrac{1}{1+\exp(-\sum_k w_{ik}^{(1)} x_k)}$ 2. $\dfrac{1}{1+\exp(\sum_k w_{ki}^{(1)} x_k)}$ 3. $\sigma(\sum\limits_k w_{ki}^{(1)} x_k)$ 4. $\sigma(-\sum\limits_k w_{ik}^{(1)} x_k)$

(b) [☑] What is $a_i^{(2)}$?

1. $\dfrac{1}{1+\exp(-\sum_k w_{ki}^{(2)} a_k^{(1)})}$ 2. $\dfrac{1}{1+\exp(\sum_k w_{ki}^{(2)} a_k^{(1)})}$ 3. $\sigma(\sum\limits_k w_{ki}^{(2)} a_k^{(1)})$ 4. $\sigma(-\sum\limits_k w_{ki}^{(2)} a_k^{(1)})$

(c) [☑] What is f ?

1. $w_1^{(3)} a_1^{(2)} + w_2^{(3)} a_2^{(2)}$
2. $w_1^{(3)} a_1^{(2)} w_2^{(3)} a_2^{(2)}$
3. $\dfrac{1}{1+\exp(-\sum_i w_i^{(3)} a_i^{(2)})}$
4. $\dfrac{1}{1+\exp(\sum_i w_i^{(3)} a_i^{(2)})}$

After some semesters of attending exams, Xiaoming finds out he cannot collect enough training samples. So he decides to use a dropout technique to reduce overfitting of his model. In particular, he applies dropout for the second hidden layer, ($a_1^{(2)}$ and $a_2^{(2)}$), with the probability of the corresponding neuron being retained set to 0.4. Help him compute the expected value of the loss in this case, given a training example x_1, x_2, x_3 and grade y, by answering the following questions.

(d) [☑] What is $\mathbb{E}[f \mid (x_1, x_2, x_3)]$?

1. $0.4(w_1^{(3)} a_1^{(2)} + w_2^{(3)} a_2^{(2)})$
2. $0.6(w_1^{(3)} a_1^{(2)} + w_2^{(3)} a_2^{(2)})$
3. $0.4 w_1^{(3)} a_1^{(2)} + 0.6 w_2^{(3)} a_2^{(2)}$
4. $0.6 w_1^{(3)} a_1^{(2)} + 0.4 w_2^{(3)} a_2^{(2)}$

(e) [☑] What is $\mathrm{Var}(f \mid (x_1, x_2, x_3))$?

1. $0.24((w_1^{(3)} a_1^{(2)})^2 + (w_2^{(3)} a_2^{(2)})^2)$
2. $0.24(w_1^{(3)} a_1^{(2)} + w_2^{(3)} a_2^{(2)})$
3. $0.16((w_1^{(3)} a_1^{(2)})^2 + (w_2^{(3)} a_2^{(2)})^2)$
4. $0.16(w_1^{(3)} a_1^{(2)})^2 + 0.64(w_2^{(3)} a_2^{(2)})^2$

(f) [☑] What is $\mathbb{E}[L]$ (assuming that the inputs and the label are random variables)?

1. $Y^2 - 2Y\mathbb{E}(f) + \mathbb{E}(f^2)$
2. $Y^2 - 2Y\mathbb{E}(f) + \mathrm{Var}(f) + (\mathbb{E}(f))^2$
3. $Y^2 + 2Y\mathbb{E}(f) + \mathbb{E}(f^2)$
4. $Y^2 + 2Y\mathbb{E}(f) + \mathrm{Var}(f) + (\mathbb{E}(f))^2$

At a certain iteration of the training, Xiaoming inputs his training example x_1, x_2, x_3 and grade y, and looks into his neural network after the forward pass. He finds that $a_1^{(2)}$ gets dropped out and $a_2^{(2)}$ is kept. To perform an SGD update to the weight $w_{21}^{(1)}$, Xiaoming:

1) Executes forward pass according to answers to (1), (2) and (3) while setting $a_1^{(2)}$ to zero.

2) Runs backward pass computing the derivative, $\frac{dL}{dw_{21}^{(1)}}$ according to the above.

3) Update parameter according to SGD rule.

(g) [✓] Help him compute the derivative, $\frac{\partial L}{\partial w_{21}^{(1)}}$.

1. $2(f-y)w_2^{(3)}\sigma'(w_{21}^{(2)}a_1^{(1)} + w_{22}^{(2)}a_2^{(1)})w_{12}^{(2)}\sigma'(w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + w_{31}^{(1)}x_3)x_2$
2. $2(f-y)w_2^{(3)}\sigma'(w_{12}^{(2)}a_1^{(1)} + w_{22}^{(2)}a_2^{(1)})w_{12}^{(2)}\sigma'(w_{12}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{32}^{(1)}x_3)x_2$
3. $2(f-y)w_2^{(3)}\sigma'(w_{12}^{(2)}a_1^{(1)} + w_{22}^{(2)}a_2^{(1)})w_{12}^{(2)}\sigma'(w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + w_{31}^{(1)}x_3)x_2$

2.2 Expressiveness

Consider neural networks with sigmoid activation functions of the form $\sigma(z) = 1/(1+\exp(-z))$. Using the same notation as the previous question, we denote by $a_i^(l)$ the activation of the ith neuron on layer l, and by $w_{i,j}^l$ the weight of the connection between neuron i at layer $l-1$ and neuron j at layer l. Additionally, each neuron comes with a bias; we denote by $w_{0,i}^l$ the bias of the ith neuron on layer l. So, for example, we have for $l > 1$

$$a_i^l = \sigma\left(w_{0,i}^l + \sum_{j \in \text{Layer}_{l-1}} w_{j,i}^l a_j^{l-1}\right),$$

and for the first layer,

$$a_i^1 = \sigma\left(w_{0,i}^1 + \sum_{j=1}^{n} w_{j,i}^1 x_j\right),$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the input to the neural network.

In the following questions you will have to design neural networks that compute functions of two Boolean inputs x_1 and x_2, that is, $x_1, x_2 \in \{0,1\}$. Given that the outputs of the sigmoid units are real numbers $Y \in (0,1)$, we will treat the final output as Boolean by considering it as 1 if greater than or equal to 0.5 and 0 otherwise.

To remind you, the logical AND function is defined via

$$0 \wedge 0 = 0, \quad 1 \wedge 0 = 0, \quad 0 \wedge 1 = 0, \quad 1 \wedge 1 = 1,$$

the logical OR function is defined via

$$0 \vee 0 = 0, \quad 1 \vee 0 = 1, \quad 0 \vee 1 = 1, \quad 1 \vee 1 = 1,$$

and the logical XOR function is defined via

$$0 \oplus 0 = 0, \quad 1 \oplus 0 = 1, \quad 0 \oplus 1 = 1, \quad 1 \oplus 1 = 0.$$

(a) [✓] Consider a neural network with one layer and one neuron. Provide three weights w_0, w_1, w_2 that implements the logical OR function $Y = x_1 \vee x_2$. Note that $w_0 = w_{0,1}^1, w_1 = w_{1,1}^1, w_2 = w_{2,1}^1$ in the notation above. Also assume that w_0, w_1 and w_2 can only take values -0.5, 0, or 1.

1. w_0 = ?
2. w_1 = ?
3. w_2 = ?

(b) [✓] Can you implement the logical AND function $Y = x_1 \wedge x_2$ using a single unit? If so, give weights that achieve this. If not, set the weights to 0.

Please note that w_0, w_1 and w_2 can only take values -2, -1.5, -1, -0.5, 0, 0.5 or 1.

1. w_0 = ?
2. w_1 = ?
3. w_2 = ?

(c) Construct a neural network with the least number of neurons you can that implements the XOR function $Y = x_1 \oplus x_2$. Draw your network and show all the weights.