Exam Protocol
Machine Learning 1

1. Multiple Choice

Which of the following is false: Assume a boosted classifier consists of weak hypotheses (aka. weak classifiers) that are each of them implemented by a threshold neuron. In that case the boosted classifier:

- ☐ can be viewed as a two-layer neural network.
- ☒ can be trained by error backpropagation instead of AdaBoost.
- ☐ can represent nonlinear decision boundaries.
- ☐ can represent non-smooth decision boundaries.

Which of the following is true: A Product of Experts:

- ☐ is an extension of a mixture model where each mixture element is allowed to be non-Gaussian.
- ☐ is an extension of a mixture model where each mixture element can be Gaussian with non-isotropic covariance.
- ☒ allows to learn more global features compared to a mixture model.
- ☐ allows to learn more local features compared to a mixture model.

Which of the following is false: Gaussian kernel ridge regression:

- ☐ is an extension of ridge regression to non-linear models.
- ☐ admits a closed-form solution when minimized for least squares.
- ☐ learns smooth non-linear functions.
- ☒ assumes that the input data is drawn from a Gaussian distribution.

Which of the following is true: In learning theory, the VC (Vapnik-Chervonenkis) bound:

- ☐ is an upper bound to the generalization error of a trained ML classifier of any complexity.
- ☐ is a lower bound to the generalization error of a trained ML classifier of any complexity.
- ☒ is an upper bound to the generalization error of a trained ML classifier of limited complexity.
- ☐ is a lower bound to the generalization error of a trained ML classifier of limited complexity.

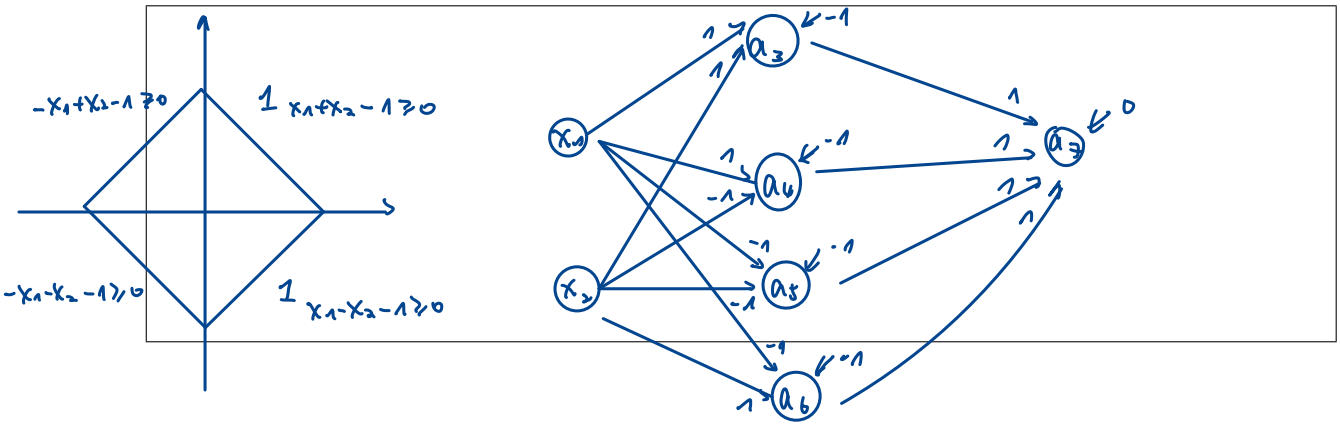
2. Neural Networks

Assume you would like to build a neural network that implements some decision boundary in $\mathbb{R}^d$. For this, you have at your disposal neurons of the type

$$a_j = \text{step}(\sum_i a_i w_{ij} + b_j)$$

where $\sum_i$ sums over the indices of the incoming neurons and where the step function is given by $\text{step}(t) = 1_{t \geq 0}$, i.e. one when the input is positive and zero otherwise. Denote by a_1 and a_2 the two input neurons (initialized to the values x_1 and x_2 respectively). Denote by a_3, a_4, a_5, a_6 the hidden neurons and by a_7 the output neuron.

2.1 Give the weights and biases associated to a neural network with the structure above and that implements the function $f(x) = step(|x_1| + |x_2| - 1)$.



2.2 Assuming a similar neural network architecture composed of one layer of hidden neurons, explain what would be the number of required hidden neurons if not taking two dimensions as input, but d dimensions and replacing x_1, x_2 by $x_1, x_2, ..., x_d$ in the formula above.

2^d

2.3 We define the objective to minimize to be the square error between the output neuron a_7 and some target variable t, i.e.

$$E = (a_7 - t)^2$$

Assume you observe the datapoint $x = (2, 3)$ with target $t = 0$. Give the value of the partial derivative $\frac{\partial E}{\partial w_{13}}$ for this data point.

$\frac{\partial E}{\partial w_{13}} = \frac{\partial E}{\partial a_7} \frac{\partial a_7}{\partial a_3} \frac{\partial a_3}{\partial w_{13}} = 2(a_7 - t)(1_{a_7 > 0})(1_{a_3 > 0}) x_1$

$a_3 = \text{step}(2 + 3 \cdot 1) = 1$ $= 2 \cdot 2 = 4$

$a_7 = 1$

3. Maximum Likelihood & Bayes Parameter Estimation

Clients are lining up in a post office. We record the time $t_1, ..., t_N$ in minutes required to serve the N consecutive clients. We distinguish between two types of clients, those that are coming to send a packet and those that are coming to send a letter (and whose service is typically twice faster). Service times for all clients are independent and drawn from an exponential distribution with rate dependent on whether the client sends a packet or a letter:

$$p(t_i|\theta) = \theta \exp(-\theta t - i) \quad (packet)$$
$$p(t_i|\theta) = 2\theta \exp(-2\theta t - i) \quad (letter)$$

and where θ is a parameter between 0 and ∞ to be learned.

3.1 Consider six clients, the first two wanted to send a packet and stayed at the post office for 2 and 5 minutes respectively. The last four clients wanted to send a letter and were served in 1 minute each.
State the likelihood function measuring the joint probability of observing all these events.

$$P(D|\theta) = \prod_{i=1}^{6} P(t_i|\theta) = \theta^2 \exp(-2\theta \cdot 1 - 5\theta \cdot 2)\, \theta^4 \cdot 16(-2\theta \cdot 4 - 3 \cdot 4 \cdot 4 - 6)$$

$$= \theta^6 \exp(-7\theta - 3) \cdot 16 \exp(-8\theta - 18)$$

$$= 16\, \theta^6 \exp(-15\theta - 21)$$

3.2 Give the optimal parameter θ in the maximum likelihood sense.

$$\max_\theta P(D|\theta) = \max_\theta \log P(D|\theta) = \max_\theta 6 \log\theta - 15\theta$$

$$\frac{\partial}{\partial\theta} 6\log\theta - 15\theta = \frac{6}{\theta} - 15 \Rightarrow \hat{\theta} = \frac{6}{15} = \frac{2}{5}$$

3.3 Give the expected time (according to the learned model) taken to serve the next three clients, each of them coming with a letter.

$$p(t_i|\theta) = 2\theta \exp(-2\theta t - i) = \underbrace{\exp(-i) \cdot 2\theta \exp(-2\theta t)}_{t \sim Exp(2\theta)}$$

$$E[t] = \frac{1}{2\theta} = \frac{1}{2 \cdot \frac{2}{5}} = \frac{5}{4}$$

$$\text{expected time} = 3 \cdot E[t] = \frac{15}{4}$$

3.4 We now take a Bayesian view on the problem. We consider the prior distribution for the parameter θ to be

$$p(\theta) = \exp(-\theta).$$

Give the equation for the posterior distribution $p(\theta|\mathcal{D})$ where $\mathcal{D}$ denotes the dataset of observations we have made, and give the parameter θ for which this posterior distribution is maximized. (Hint: You don't need to develop $p(\mathcal{D})$).

$p(\theta|D) = \frac{p(D|\theta)\,p(\theta)}{\int p(D|\theta)p(\theta)\,d(\theta)} = \frac{16\,\theta^6 \exp(-16\theta - 21)}{\int 16\theta^6 \exp(-16\theta - 21)\,d\theta}$

$\max_\theta p(\theta|D) = \max_\theta 6\log\theta - 16\theta \qquad \frac{\partial}{\partial\theta}(6\log\theta - 16\theta) = \frac{6}{\theta} - 16 = 0$

$\theta = \frac{6}{16}$

4. Explainable AI

Shapley Values porvide a way of attributing a prediction on the input features. The Shapley values $\phi_1, ..., \phi_d$ measuring the contribution of each feature are:

$$\phi_i = \sum_{\mathcal{S}: i \notin \mathcal{S}} \frac{|\mathcal{S}|!(d - |\mathcal{S}| - 1)!}{d!} \left[f(\vec{x}_{\mathcal{S}\cup\{i\}}) - f(\vec{x}_{\mathcal{S}}) \right]$$

Where $(\vec{x}_{\mathcal{S}})_{\mathcal{S}}$ are all possible subsets of features contained in the input $\vec{x}$. Shapley values assume a reference point which provides a replacement value for features that are removed. In this exercise we define the reference point to be $\vec{x}_0 = 0$, i.e. we set features to zero when removing them.

4.1 Compute the Shapley values ϕ_1, ϕ_2, ϕ_3 for the function $f(\vec{x}) = min(x_1, x_2, x_3)$ evaluated at the data point $\vec{x} = (4, 3, 4)$.

ϕ_2	ν_S	ΔS
$\{\}$	$\frac{2!}{3!} = \frac{1}{3}$	$\min(0,3,0) - \min(0,0,0) = 0$
$\{1\}$	$\frac{1}{6}$	$\min(4,3,0) - \min(1,0,0) = 0$
$\{3\}$	$\frac{1}{6}$	0
$\{1,3\}$	$\frac{1}{3}$	$\min(4,3,4) - \min(4,0,3) = 3$

$\phi_2 = \frac{1}{3} * 3 = 1$

$f(x) = \min(4,3,4) = 3 = \sum_i \phi_i$

$= 1 + \phi_1 + \phi_3$

$\phi_1 = \phi_2 = \phi_3$ Symmetric

Another approach for a explaining a prediction that does not involve evaluating the function multiple times is based on Taylor expansions. For example we choose a reference point $\vec{x}_0 = t \cdot \vec{x}$ with $0 < t < 1$ and compute

$$f(\vec{x}) = f(\vec{x}_0) + \sum_{i=1}^{d} \underbrace{[\nabla f(\vec{x}_0)]_i \cdot (x - x_{0i})}_{\phi_i} + ...$$

where ... denotes the higher order terms. Interestingly when the function is positive homogeneous, i.e. $\forall_{t \geq 0} : f(t\vec{x}) = tf(\vec{x})$, the Taylor expansion simplifies and in the limit of $t \to 0$, we get the scores $\phi_i = [\nabla f(\vec{x})]_i \cdot x_i$ also known as Gradient x input.

4.2 Show that the function $f(\vec{x})$ used above in the Shapley value exercise is positive homogeneous, i.e. show that:

$$f(t\vec{x}) = tf(\vec{x})$$

for all $\vec{x} \in \mathbb{R}^3$ and $t \geq 0$.

$f(t\vec{x}) = \min(tx_1, tx_2, tx_3) = t \min(x_1, x_2, x_3) = tf(\vec{x})$

4.3 Compute an explanation of the same prediction as in the Shapley values exercise, but this time using Gradient x input, i.e. compute the explanation scores $\phi_i = [\nabla f(\vec{x})]_i \cdot x_i$.

$$\nabla f(\vec{x}) = \begin{cases} \mathbb{1}\{\min(x_1, x_2, x_3) = x_1\} \\ \mathbb{1}\{\quad\quad\quad\quad = x_2\} \\ \mathbb{1}\{\quad\quad\quad\quad = x_3\} \end{cases}$$

5. Quadratic Programming

Consider an unsupervised dataset $\vec{x}_1, ..., \vec{x}_N \in \mathbb{R}^d$. We would like to learn a hyperplane that separates the data from the origin and that is as far as possible from the origin. This orientation of the hyperplane can be characterized by a vector $\vec{w} \in \mathbb{R}^d$ to which the hyperplane is orthogonal, and this vector can be found by constrained optimization.

The first part of the exercise will consist of writing a function that receives the dataset as input (as a numpy array X of dimensions N x d) and returns the vector $\vec{w}$ (a numpy array of dimension d).

Your implementation should make use of the cvxopt library internally. Specifically your function should consist of two parts, first, prepare the data structures required by cvxopt, then run cvxopt.solvers.qp (cf https://cvxopt.org/userguide/coneprog.html for the documentation) and return the desired output based on the output of the cvxopt function.

5.1 Write such function when the vector $\vec{w}$ is obtained by solving the constrained optimization problem:

$$\min_{\vec{w}} ||\vec{w}||^2 \text{ s.t. } \forall_{i=1}^{N} \vec{x}_i^T \vec{w} \geq 1$$

$w^T [2I] w$ $\quad -Xw \leq -1$ (d, N)

cvxopt.solvers.qp(*P, q* [, *G, h* [, *A, b* [, *solver* [, *initvals*]]]])

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

Solve(X):

```
N, d = X.shape
P = matrix(2 * np.eye(d))
q = matrix(np.zeros(d))
G = matrix(-X)
h = matrix(-1 * np.ones(N))
return qp(P, q, G, h)
```

5.2 Write a function that takes the output of the previous function and finds the point in the dataset that is farthest from the hyperplane.

```
find(X, w):
    dists = (X @ w) / np.linalg.norm(w)
    idx_farthest = np.argmax(dists)
    return X[idx_farthest]
```