# HSE 2021: Mathematical Methods for Data Analysis.
# Assignment 6: optional

December 10, 2021

## Disclaimer

- This is an optional homework, which contains of **6** theoretical problems, 2 points each.
- You may choose any 5 to get 10 points. Getting more that 10 points is not possible.
- We encourage you to use LaTeXto write the solution. Overleaf is a nice online editor, if you don't want to install it locally. Hand-written solutions will be also accepted, but only if you provide high quality **scans** in the form of a single pdf file. Please, make sure that TAs can read what you've submitted, otherwise, the submission will not be graded.
- Please, give as much details in your derivation as possible.

# Problem 1. Boosting. [2 points]

In this task you will be working with gradient boosting algorithm. Let's firstly recap the notation and the algorithm itself.

$$b_m(x) := \text{the best base model from the family of the algorithms } \mathcal{A} \tag{1}$$

$$\gamma_m(x) := \text{scale or weight of the new model} \tag{2}$$

$$a_M(x) = \sum_{m=0}^{M} \gamma_m b_m(x) := \text{the final composite model} \tag{3}$$

Consider a loss function $L(y, z)$ for the target $y$ and prediction $z$, and let $\{x_n, y_n\}_{n=1}^N$ be the train dataset with $N$ observations for a regression task. Then gradient boosting algorithm is the following:

1. Initialize $a_0(x) = \hat{z}$ with the constant prediction $\hat{z} = \arg\min_{z \in \mathbb{R}} \sum_{n=1}^N L(y_n, z)$
2. For $m$ from `1` to `M` do:
   Solve the current subproblem $G_m(b, \gamma) = \sum_{n=1}^N L\big(y_n, a_{m-1}(x_n) + \gamma b(x_n)\big) \to \min\limits_{b,\gamma}$ , using the following method:
   - Compute the residuals
   $$s_n = -\frac{\partial}{\partial z} L(y_n, z)\Big|_{z=a_{m-1}(x_n)}, n = 1, \ldots, N. \tag{4}$$
   - Train the next base algorithm
   $$b_m(x) = \arg\min_{b \in \mathcal{A}} \sum_{n=1}^{N} \big(b(x_n) - s_n\big)^2. \tag{5}$$
   - Find its weight
   $$\gamma_m = \arg\min_{\gamma} G_m(b_m, \gamma). \tag{6}$$
   - Update the mixture
   $$a_m(x) = a_{m-1}(x) + \gamma_m b_m(x). \tag{7}$$
3. Return $a_M(x) = a_0(x) + \sum_{m=1}^M \gamma_m b_m(x)$.

**Finally, the task**

Consider Poisson loss, namely $L(y, z) = -yz + \exp z$.

- Derive formula for the residuals at a step $m$
- Derive first-order conditions for $\gamma$ at a step $m$

**Solution**

`YOUR SOLUTION HERE`

# Problem 2. Bayesian ML. [2 points]

Consider a univariate Gaussian likelihood:

$$p(x|\mu,\tau) = \mathcal{N}(x|\mu,\tau^{-1}) = \left(\frac{\tau}{2\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^2\tau\right). \tag{8}$$

Let's define the following prior for the parameters $(\mu,\tau)$:

$$p(\mu,\tau) = \mathcal{N}(\mu|\mu_0,(\beta\tau)^{-1}) \cdot \text{Gamma}(\tau;a,b) \tag{9}$$

$$= \left(\frac{\beta\tau}{2\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2}(\mu-\mu_0)^2\beta\tau\right) \cdot \frac{b^a\tau^{a-1}\exp(-b\tau)}{\Gamma(a)}. \tag{10}$$

**The task**

Find the posterior distribution of $(\mu,\tau)$ after observing $N$ i.i.d. samples $X = (x_1,\ldots,x_N)$ from the $p(x|\mu,\tau)$.

1. Use Bayes formula to express $p(\mu,\tau|\{x_1,\cdots,x_N\})$ using prior and likelihood.
2. The hardest part in your equation above is to compute normalization constant (denominator). We will avoid that. Write down the formula for the logarithm of the numerator and get rid of all the terms that do not depend on $\mu$ or $\tau$. This is logarithm of our posterior up to normalization constant.
3. Write down logarithm of the prior $p(\mu,\tau)$ and get rid of all of the terms that do not depend on $\mu$ and $\tau$.
4. Compare equations in 2 and 3 and conclude what is the family of the posterior distribution, write down its parameters.
5. Calculate the normalization constant of the posterior distribution

**Solution**

`YOUR SOLUTION HERE`

## Problem 3. Gaussian Processes. [2 points]

Assume, that the function $y(x)$, $x \in \mathbb{R}^d$, is a realization of a Gaussian Process with the kernel $K(a, b) = \exp(-\gamma \|a - b\|_2^2))$:

$$y(x) \sim GP\big(0; K(x, x)\big). \tag{11}$$

Namely, for a given $x$, $y$ has a Gaussian distribution $\mathcal{N}(y|0, K(x, x))$
Suppose two datasets were observed: **noiseless** and **noisy**:

$$D_0 = \{x_n, y(x_n)\}_{n=1}^N \,, \tag{12}$$

$$D_1 = \{x'_m, y(x'_m) + \varepsilon_m\}_{m=1}^M \,, \tag{13}$$

where $\varepsilon_m$ are i.i.d. Gaussian: $\varepsilon_m \sim \mathcal{N}(\varepsilon_m|0, \sigma^2)$.

### The task

Derive the conditional distribution for a new point $y^* = y(x^*)$, given observed data: $p(y^* \big| D_0, D_1)$.

### Hint

You can find useful properties of the Gaussian distribution for this task in the Matrix Cookbook

### Solution

YOUR SOLUTION HERE

# Problem 4. MLP. [2 points]

Consider a multiclass classification task. Dataset consists of feature vectors $x \in \mathbb{R}^d$ and target variables $y \in \{1, 2, \ldots, K\}$.

Assume you are using a neural network with 1 linear layer and a softmax activation to predict a class of a given object. For a single object, forward pass through this network will look like this:

$$h = W^T x \tag{14}$$

$$t_i = \frac{\exp h_i}{\sum_{j=1}^{K} \exp h_j} \tag{15}$$

where $W \in \mathbb{R}^{d \times K}$ is an unknown weight matrix, $h \in \mathbb{R}^K$ is a hidden state vector, $t_i \in [0, 1]$ is a predicted probability for an object to be from class $i$.

To train a neural network, cross-entropy loss will be used. For a single object it looks like this:

$$l = -\log t_y \tag{16}$$

where $t_y$ is an output of the network with input $y$ (index of the correct class).

**The task**

1. Work out derivative $\frac{\partial l}{\partial t_i} \quad \forall i$.
2. Work out derivative $\frac{\partial t_i}{\partial h_j} \quad \forall i, j$.
   **Hint:** note that $t_i$ depends on all of the $h_j$, not only on the $h_i$. It might be convenient to consider $\frac{\partial t_i}{\partial h_i}$ and $\frac{\partial t_i}{\partial h_j}$ (when $i \neq j$) separately.
3. Use results from q1-2 to obtain $\frac{\partial l}{\partial h_i}$.
4. Explain how $\frac{\partial l}{\partial h_i}$ will be used in a backward pass of a neural network.

**Solution**

YOUR SOLUTION HERE

## Problem 5. Generative Model. [2 points]

We observe a dataset $\{x_1, \dots, x_N\}$, in other words, we consider an empirical distribution over $x$: $p_e(x) = \frac{1}{N}\sum_{n=1}^{N} \delta_{x_n}(x)$. We want to infer a latent representation $z$ for a point $x$ from the dataset. Thus, we consider the following generative model with parameters $\theta$:

$$z \sim p(z), \quad x \sim p_\theta(x|z). \tag{17}$$

We choose our generative model to be a linear and assume the presence of the normal noise:

$$p_\theta(x|z) = \mathcal{N}(x|W_p z + \mu_p, \Lambda_p^{-1}), \theta := \{W_p, \mu_p, \Lambda_p^{-1}\}, \tag{18}$$

$$p(z) = \mathcal{N}(z|0, I). \tag{19}$$

We want to infer parameters from data as an MLE solution:

$$\theta^* = \arg\max_\theta \mathbb{E}_x \log \int p_\theta(x|z)p(z)dz. \tag{20}$$

Also, we would like to have the ability to find the latent representation $z$ for a new datapoint $x$. Thus, we will use variational approach to solve the optimization problem:

$$\max_\theta \mathbb{E}_x \log \int p_\theta(x|z)p(z)dz \geq \max_{\theta,\phi} \mathbb{E}_x \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}dz. \tag{21}$$

Since generative process is linear, we would like to use similar structure for the inference:

$$q_\phi(z|x) = \mathcal{N}(z|W_q x + \mu_q, \Lambda_q^{-1}), \phi := \{W_q, \mu_q, \Lambda_q^{-1}\}. \tag{22}$$

Finally, note that taking expectations w.r.t empirical distribution is the same as averaging, which gives us the following objective:

$$\mathcal{L} = \mathbb{E}_x \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}dz = \frac{1}{N}\sum_{n=1}^{N} \int q_\phi(z|x_n) \log \frac{p_\theta(x_n|z)p(z)}{q_\phi(z|x_n)}dz. \tag{23}$$

**Finally, the task**

- Use first-order conditions (FOC) to find: $W_p, \mu_p$, given $W_q, \mu_q$, $\Lambda_q$ using objective (23). Note that in the final formula $W_p$ may depend on $\mu_p$ and vice versa.
- Is it enough to check the FOC for $\mu_p$? Check the convexity over $\mu_p$.

**Solution**

`YOUR SOLUTION HERE`

## Problem 6. Linear Model for Recommendation. [2 points]

In this task we will use a linear model, to build a simple recommendation system. Assume that you have $N$ products and $M$ users. Each product is represented by a feature vector $u_i \in \mathbb{R}^k$ and each user is represented by a feature vector $v_j \in \mathbb{R}^k$. These vector are not given to use, we will learn them from the data.

For some pairs of users and products you observe values $r_{i,j} \in [0, 1]$ - how much user $i$ likes product $j$ (we will call it rating).

We will use the following model to predict rating for a given pair product-user:

$$r_{i,j} = \text{Softmax}(u_i^T v_j) = \frac{\exp(u_i^T v_j)}{\sum_q \exp(u_i^T v_q)}, \tag{24}$$

$$\log r_{i,j} = u_i^T v_j - \log \sum_q \exp(u_i^T v_q). \tag{25}$$

Since $r_{i,j} \in [0, 1]$ we will interpret it as a probablity and use maximum likelihood to find users and products representations.

$$\sum_{(i,j)} \log r_{i,j} \to \max_{U,V}, \tag{26}$$

where $U \in \mathbb{R}^{M \times k}$, $V \in \mathbb{R}^{N \times k}$.

### The task

- Find the gradient of (26) with respect to $u_i$.
- Find the gradient of (26) with respect to $v_i$.
- Let's add regularization (with hyperparameters $\lambda_1, \lambda_2$ to the model:

$$\sum_{(i,j)} \log r_{i,j} + \lambda_1 \|V^T V - I\|_2^2 + \lambda_2 \|U^T U - I\|_2^2 \to \max_{U,V}. \tag{27}$$

  Find the gradient of a regularized objective (27) w.r.t. $U$ and $V$.
- Explain how you will used computed gradient to train the model
- Assume that you found the solution $U^*$ and $V^*$ of the optimization problem (27). Which product will you recommend to the user with the index $m$ based on the train model?

### Solution

YOUR SOLUTION HERE