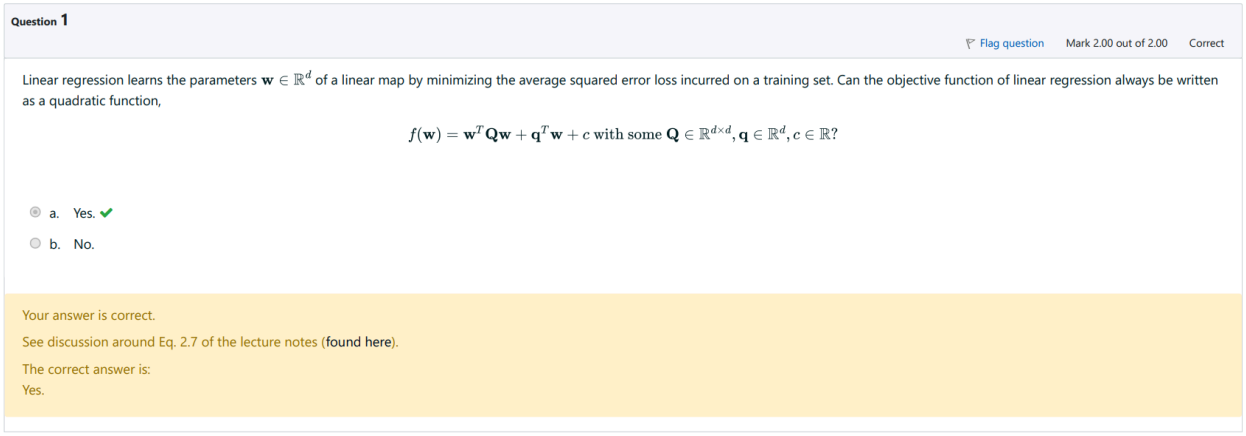
TABLE OF CONTENTS

1. ML Basics



Question 1

Flag question Mark 2.00 out of 2.00 Correct

Linear regression learns the parameters $\mathbf{w} \in \mathbb{R}^d$ of a linear map by minimizing the average squared error loss incurred on a training set. Can the objective function of linear regression always be written as a quadratic function,

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{q}^T\mathbf{w} + c \text{ with some } \mathbf{Q} \in \mathbb{R}^{d\times d}, \mathbf{q} \in \mathbb{R}^d, c \in \mathbb{R}?$$

a. Yes. ✔

b. No.

Your answer is correct.

See discussion around Eq. 2.7 of the lecture notes (found here).

The correct answer is:
Yes.

Lecture - "ML Basics", Three Components and a Design Principle (Equation 2.7)

Figure 2.1: ERM (2.1) for linear regression minimizes a convex quadratic function $\mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q}$.

Inserting (2.4) into (2.3) allows to formulate linear regression as

$$\widehat{\mathbf{w}}^{(\mathrm{LR})} \in \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q} \qquad (2.7)$$

$$\text{with } \mathbf{Q} := (1/m)\mathbf{X}^T\mathbf{X}, \mathbf{q} := -(2/m)\mathbf{X}^T\mathbf{y}.$$

The matrix $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is psd with corresponding eigenvalue decomposition (EVD),

$$\mathbf{Q} = \sum_{j=1}^{d} \lambda_j(\mathbf{Q})\mathbf{u}^{(j)}\left(\mathbf{u}^{(j)}\right)^T. \qquad (2.8)$$

Question 2

Flag question Mark 2.00 out of 2.00 Correct

Ridge regression learns the parameters $\mathbf{w} = \left(w_1, \ldots, w_n\right)^T$ of a linear map by minimizing the sum of the average squared error loss on a training set and the penalty term $\alpha \|\mathbf{w}\|_2^2$. Is it possible to formulate ridge regression as the minimization of a quadratic function of the form

$$f(\mathbf{w}) = \mathbf{w}^T \mathbf{Q} \mathbf{w} + \mathbf{q}^T \mathbf{w} \, ?$$

a. Yes. ✔

b. No.

Your answer is correct.

See Sec. 2.6. of the lecture notes (found here).

The correct answer is:
Yes.

Lecture - "ML Basics", Regularization (Equation 2.27)

To study the computational aspects of ridge regression, let us rewrite (2.25) as

$$\widehat{\mathbf{w}}^{(\alpha)} \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \mathbf{w}^T \mathbf{Q} \mathbf{w} + \mathbf{w}^T \mathbf{q}$$
$$\text{with } \mathbf{Q} := (1/m)\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I}, \, \mathbf{q} := (-2/m)\mathbf{X}^T \mathbf{y}. \qquad (2.27)$$

Thus, like linear regression (2.7), also ridge regression minimizes a convex quadratic function. A main difference between linear regression (2.7) and ridge regression (for $\alpha > 0$) is that the matrix $\mathbf{Q}$ in (2.27) is guaranteed to be invertible for any training set $\mathcal{D}$. In contrast, the matrix $\mathbf{Q}$ in (2.7) for linear regression might be singular for some training sets.[3]

Question 3

Flag question Mark 2.00 out of 2.00 Correct

Consider some ERM-based ML method that learns model parameters by minimizing the average loss on a training set. After completing this training process, we compute the resulting average loss on the training set (the training error) and the average loss on a validation set (the validation error). Can it happen that the validation error is smaller than the training error?

a. Yes, for some training and validation sets. ✔

b. No, the validation error is **always (for any dataset)** at least as high as the training error.

c. Yes. Moreover, the validation error is **always (for any dataset)** smaller than the training error.

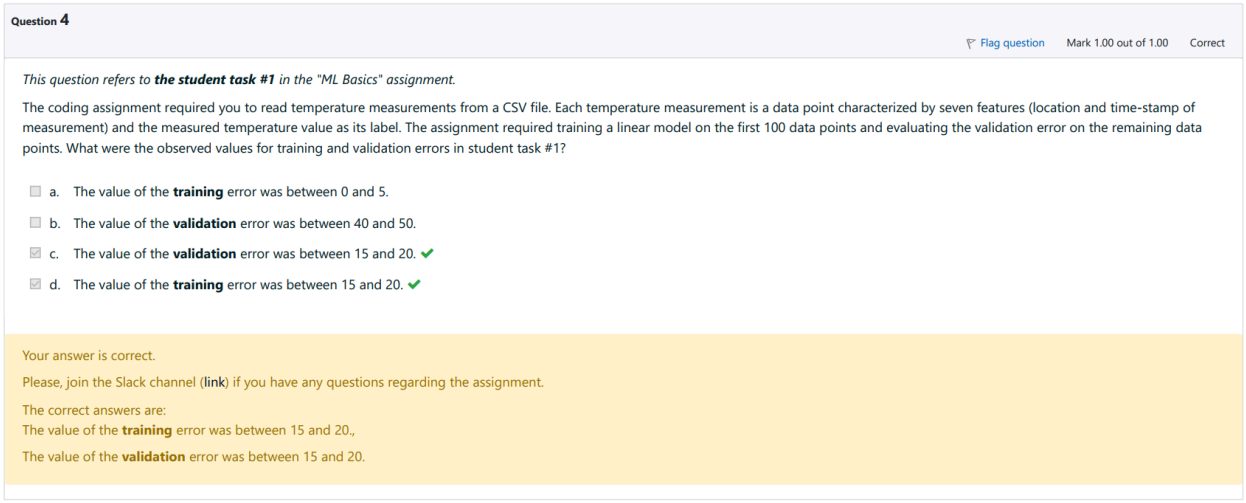
Your answer is correct.

See discussion around Figure 2.3. of the lecture notes (found here).

The correct answer is:
Yes, for some training and validation sets.

Lecture - "ML Basics", Validation and diagnosis of ML

- $E_t \gg E_v$: The training error is significantly larger than the validation error. The idea of ERM (2.24) is to approximate the risk (2.12) of a hypothesis by its average loss on a training set $\mathcal{D} = \{(\mathbf{x}^{(r)}, y^{(r)})\}_{r=1}^{m}$. The mathematical underpinning for this approximation is the law of large numbers which characterizes the average of (realizations of) i.i.d. RVs. The accuracy of this approximation depends on the validity of two



Question 4

Flag question Mark 1.00 out of 1.00 Correct

*This question refers to **the student task #1** in the "ML Basics" assignment.*

The coding assignment required you to read temperature measurements from a CSV file. Each temperature measurement is a data point characterized by seven features (location and time-stamp of measurement) and the measured temperature value as its label. The assignment required training a linear model on the first 100 data points and evaluating the validation error on the remaining data points. What were the observed values for training and validation errors in student task #1?

- a. The value of the **training** error was between 0 and 5.
- b. The value of the **validation** error was between 40 and 50.
- c. The value of the **validation** error was between 15 and 20. ✔
- d. The value of the **training** error was between 15 and 20. ✔

Your answer is correct.

Please, join the Slack channel (**link**) if you have any questions regarding the assignment.

The correct answers are:
The value of the **training** error was between 15 and 20.,

The value of the **validation** error was between 15 and 20.

```
********** Linear Regression Diagnosis **********
Training error: 17.082721305711885
Validation error: 18.25410763805204
```

Question 5

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #2** in the "ML Basics" assignment.*

The coding assignment required you to study the effect of augmenting the given features (latitude, longitude, day, month, year, hour, minute of measurement) by their polynomial combinations up to a given maximum degree d.
For each choice of $d = 1, 2, 3$, you had to learn the parameters of a linear model and then compute the training and validation errors (average squared error loss) of the learnt model parameters. How do the training and validation errors change with increasing polynomial degrees d?

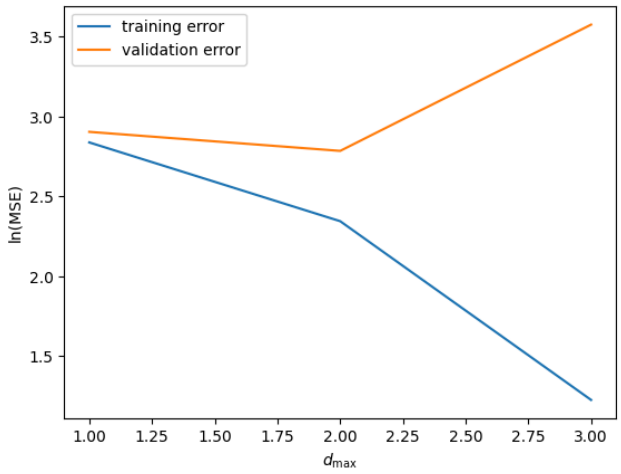
- [x] a. The validation error might increase with increasing d. ✔
- [ ] b. The training error might increase with increasing degree d.
- [ ] c. The validation error never increases with increasing d.
- [x] d. The training error never increases with increasing d. ✔

Your answer is correct.

Increasing the degree d results in a larger hypothesis space and, in turn, can only result in a decrease of the training error but never (ignoring imperfections such as numerical errors during optimization) in an increase of the training error. However, increasing d might result in increased validation error due to overfitting. More discussion can be found in Ch. 6 and 7 of A.J, "Machine Learning: The Basics", Springer, 2022 (preprint here).

The correct answers are:
The training error never increases with increasing d.,

The validation error might increase with increasing d.



Question 6

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #3** in the "ML Basics" assignment.*

The coding assignment required you to use ridge regression to learn the parameters of a linear model using the original features and their polynomial combinations. You had to determine the resulting training and validation errors for different choices for the regularization parameter α of ridge regression. How do they vary with increasing values of α?

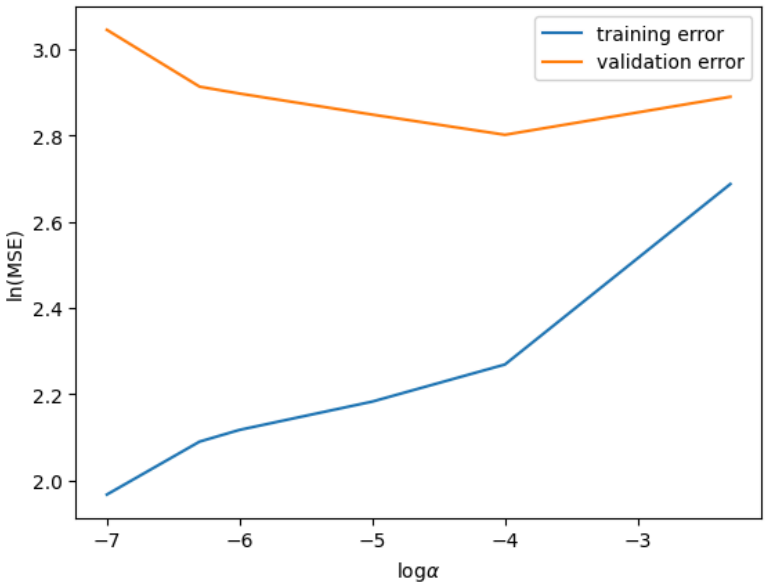
- a. The training error always decreases with increasing α.
- b. The validation error always increases with increasing α.
- c. The training error might increase with increasing α. ✔
- d. The validation error might decrease with increasing α. ✔
- e. The training error does not change with increasing α.

Your answer is correct.

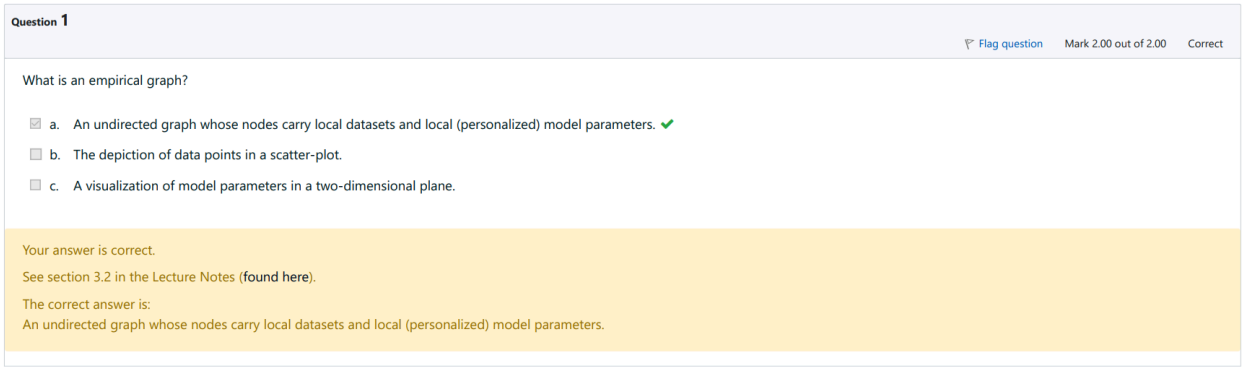
Increasing the regularization parameter α in ridge regression corresponds to a shrinking of the effective hypothesis space and, in turn, can only result in an increase of the training error. However, increasing α might result in a reduced validation error due to making the training more robust against overfitting. More discussion can be found in Ch. 7 of A J, "Machine Learning: The Basics", Springer, 2022 (preprint here).

The correct answers are:
The training error might increase with increasing α.

,
The validation error might decrease with increasing α.



2. FL Design Principle



Question 1

Flag question Mark 2.00 out of 2.00 Correct

What is an empirical graph?

- a. An undirected graph whose nodes carry local datasets and local (personalized) model parameters. ✔
- b. The depiction of data points in a scatter-plot.
- c. A visualization of model parameters in a two-dimensional plane.

Your answer is correct.

See section 3.2 in the Lecture Notes (found here).

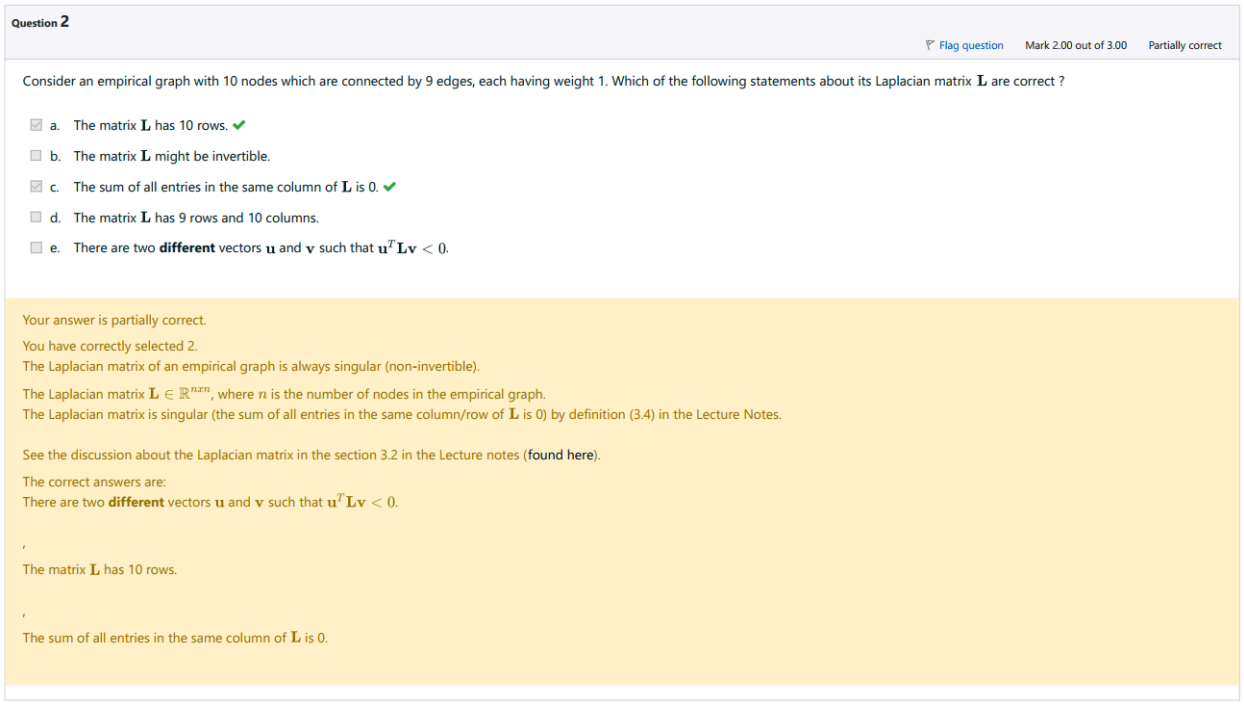
The correct answer is:
An undirected graph whose nodes carry local datasets and local (personalized) model parameters.

Lecture - "FL Design Principle", Empirical graphs and their Laplacian

The empirical graph $\mathcal{G}$ is an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} := \{1, \ldots, n\}$. Each node $i \in \mathcal{V}$ of the empirical graph $\mathcal{G}$ carries a local dataset

$$\mathcal{D}^{(i)} := \left\{ \left(\mathbf{x}^{(i,1)}, y^{(i,1)}\right), \ldots, \left(\mathbf{x}^{(i,m_i)}, y^{(i,m_i)}\right) \right\}. \tag{3.1}$$

Here, $\mathbf{x}^{(i,r)}$ and $y^{(i,r)}$ denote, respectively, the features and the label of the rth data point in the local dataset $\mathcal{D}^{(i)}$. Note that the size m_i of the local dataset might vary between different nodes $i \in \mathcal{V}$.



Question 2

Flag question Mark 2.00 out of 3.00 Partially correct

Consider an empirical graph with 10 nodes which are connected by 9 edges, each having weight 1. Which of the following statements about its Laplacian matrix $\mathbf{L}$ are correct ?

- a. The matrix $\mathbf{L}$ has 10 rows. ✔
- b. The matrix $\mathbf{L}$ might be invertible.
- c. The sum of all entries in the same column of $\mathbf{L}$ is 0. ✔
- d. The matrix $\mathbf{L}$ has 9 rows and 10 columns.
- e. There are two **different** vectors $\mathbf{u}$ and $\mathbf{v}$ such that $\mathbf{u}^T\mathbf{L}\mathbf{v} < 0$.

Your answer is partially correct.

You have correctly selected 2.
The Laplacian matrix of an empirical graph is always singular (non-invertible).

The Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$, where n is the number of nodes in the empirical graph.
The Laplacian matrix is singular (the sum of all entries in the same column/row of $\mathbf{L}$ is 0) by definition (3.4) in the Lecture Notes.

See the discussion about the Laplacian matrix in the section 3.2 in the Lecture notes (found here).

The correct answers are:
There are two **different** vectors $\mathbf{u}$ and $\mathbf{v}$ such that $\mathbf{u}^T\mathbf{L}\mathbf{v} < 0$.

,
The matrix $\mathbf{L}$ has 10 rows.

,
The sum of all entries in the same column of $\mathbf{L}$ is 0.

Lecture - "FL Design Principle", Empirical graphs and their Laplacian

The Laplacian matrix is defined element-wise as (see Fig. 3.2)

$$L^{(\mathcal{G})}_{i,i'} := \begin{cases} -A_{i,i'} & \text{for } i \neq i', \{i,i'\} \in \mathcal{E} \\ \sum_{i'' \neq i} A_{i,i''} & \text{for } i = i' \\ 0 & \text{else.} \end{cases} \tag{3.6}$$

Number of i is number of nodes, which is 10 => The matrix L has 10 rows

One immediate consequence of (3.8) is that any collection of identical local model parameters

$$\mathbf{w} = \text{stack}\{\mathbf{c}\} = \left(\mathbf{c}^T, \ldots, \mathbf{c}^T\right)^T, \text{ with some } \mathbf{c} \in \mathbb{R}^d, \tag{3.11}$$

is an eigenvector of $\mathbf{L}^{(\mathcal{G})} \otimes \mathbf{I}$ with corresponding eigenvalue $\lambda_1 = 0$ (see (3.10)). Thus, the Laplacian matrix of any empirical graph is singular (non-invertible).

=> The matrix X may be invertible is False

The Laplacian matrix is defined element-wise as (see Fig. 3.2)

$$L^{(\mathcal{G})}_{i,i'} := \begin{cases} -A_{i,i'} & \text{for } i \neq i', \{i,i'\} \in \mathcal{E} \\ \sum_{i'' \neq i} A_{i,i''} & \text{for } i = i' \\ 0 & \text{else.} \end{cases} \tag{3.6}$$

L = D - A, where D is degree matrix and A is adjacency matrix, which has 1 if there is an edge from node i to node j and 0 if these two nodes are not connected. Naturall the degree of a node (in matrix D) is equal to the number of the edges a node has (in matrix A), **which makes the column of L always has sum of 0** due to the formula D - A

As like before, L is a square matrix of dimension num nodes x num nodes => **it should be 10 x 10, not 9 x 10 (False)**

Don't be tricked by the last question. It is true that Laplacian matrix is positive semidefinite, which means uT L u is always non negative given any real value column vector u. **However the question asks about two separate vector u and v (THEY ARE NOT THE SAME VECTOR), so the answer naturally is True**

Question 3

Flag question Mark 0.00 out of 2.00 Incorrect

*This question refers to **the student task #1** in the "FL Design Principle" assignment.*

The assignment requires to you generate the Laplacian matrix, build the matrix Q and the vector q for the quadratic objective function in GTVMin. What is the training error and the total variation for the computed solution for the GTVMin instance?

- a. **The training error** is around 32.77.
- b. **The total variation** is around 203.29.
- c. **The training error** is around 35.23. ✖
- d. **The total variation** is around 198.78.
- e. **The training error** is around 38.55.
- f. **The total variation** is around 207.85. ✖

Your answer is incorrect.

See equations 3.17-3.22 from the Lecture Notes (found here). Please, join Slack channel (link) if you have any questions.

The correct answers are:
The training error is around 32.77.,

The total variation is around 203.29.

```
Training error: 32.77241810495208
Total variation: 203.28666126559543
```

```python
# Generate the Laplacian matrix for the empirical graph.
L_FMI = nx.laplacian_matrix(G_FMI).toarray()

# Build matrix Q and vector q  (Eq. (3.18)) for the quadratic objective
# function in GTVMin (see Eq. (3.17)) in the Lecture Notes.
Q = np.eye(num_stations) + gtvmin_alpha * L_FMI
q = np.zeros(num_stations)
for iter_node in range(num_stations):
    q[iter_node] = -2*np.sum(G_FMI.nodes[iter_node]['y']) /
G_FMI.nodes[iter_node]['samplesize']

# Use the zero-gradient condition (see Lecture Notes) to compute a
solution for the GTVMin instance.
hat_w = LA.inv(Q).dot(-0.5*q)
```

Inserting (3.16) into (3.15), yields the following instance of GTVMin to train local linear models,

$$\big\{\widehat{\mathbf{w}}^{(i)}\big\} \in \operatorname*{argmin}_{\{\mathbf{w}^{(i)}\}_{i=1}^{n}} \sum_{i\in\mathcal{V}} (1/m_i) \left\|\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i)}\right\|_2^2 + \alpha \sum_{\{i,i'\}\in\mathcal{E}} A_{i,i'} \left\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\right\|_2^2. \quad (3.17)$$

The identity (3.8) allows to rewrite (3.17) using the Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$ as

$$\widehat{\mathbf{w}}^{(i)} \in \operatorname*{argmin}_{\mathbf{w}=\mathrm{stack}\left\{\mathbf{w}^{(i)}\right\}} \sum_{i\in\mathcal{V}} (1/m_i) \left\|\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i)}\right\|_2^2 + \alpha \mathbf{w}^T \big(\mathbf{L}^{(\mathcal{G})} \otimes \mathbf{I}_d\big)\mathbf{w}. \quad (3.18)$$

Let us rewrite the objective function in (3.18) as

$$\mathbf{w}^T \left(\begin{pmatrix} \mathbf{Q}^{(1)} & 0 & \cdots & 0 \\ 0 & \mathbf{Q}^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{Q}^{(n)} \end{pmatrix} + \alpha \mathbf{L}^{(\mathcal{G})} \otimes \mathbf{I} \right) \mathbf{w} + \big(\big(\mathbf{q}^{(1)}\big)^T, \ldots, \big(\mathbf{q}^{(n)}\big)^T \big)\mathbf{w} \quad (3.19)$$

with $\mathbf{Q}^{(i)} = (1/m_i)\big(\mathbf{X}^{(i)}\big)^T\mathbf{X}^{(i)}$, and $\mathbf{q}^{(i)} := (-2/m_i)\big(\mathbf{X}^{(i)}\big)^T\mathbf{y}^{(i)}$.

Thus, like linear regression (2.7) and ridge regression (2.27), also GTVMin (3.18) (for local linear models $\mathcal{H}^{(i)}$) minimizes a convex quadratic function,

$$\big\{\widehat{\mathbf{w}}^{(i)}\big\}_{i=1}^{n} \in \operatorname*{argmin}_{\mathbf{w}=\mathrm{stack}\left\{\mathbf{w}^{(i)}\right\}_{i=1}^{n}} \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{q}^T\mathbf{w}. \quad (3.20)$$

Here, we used the psd matrix

$$\mathbf{Q} := \begin{pmatrix} \mathbf{Q}^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q}^{(n)} \end{pmatrix} + \alpha \mathbf{L}^{(\mathcal{G})} \otimes \mathbf{I} \text{ with } \mathbf{Q}^{(i)} := (1/m_i) \big(\mathbf{X}^{(i)}\big)^T \mathbf{X}^{(i)} \tag{3.21}$$

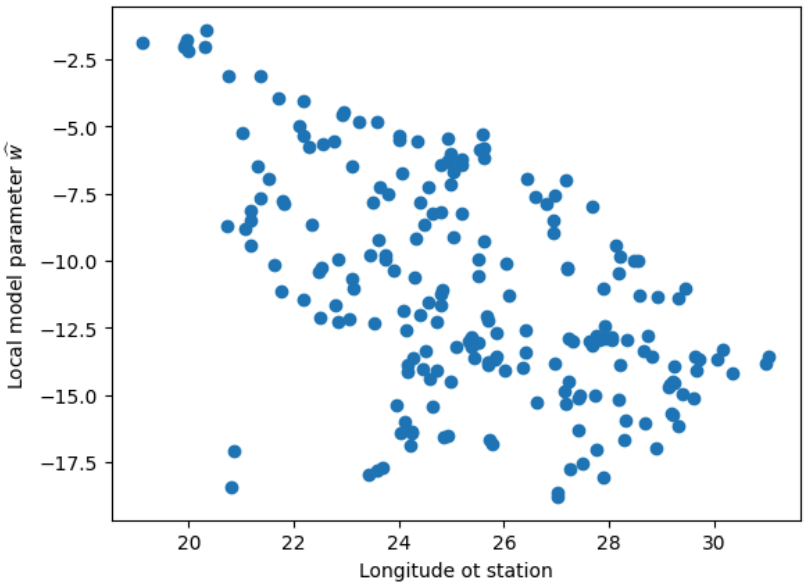
and the vector

$$\mathbf{q} := \big(\big(\mathbf{q}^{(1)}\big)^T, \ldots, \big(\mathbf{q}^{(n)}\big)^T \big)^T, \text{ with } \mathbf{q}^{(i)} := (-2/m_i) \big(\mathbf{X}^{(i)}\big)^T \mathbf{y}^{(i)}. \tag{3.22}$$

Let us now consider the computational aspects of GTVMin (3.17) to train local linear models. As discussed above, this instance is equivalent to solving (3.20). Any solution $\widehat{\mathbf{w}}$ of (3.20) (and, in turn, (3.17)) is characterized by the zero-gradient condition

$$\mathbf{Q}\widehat{\mathbf{w}} = -(1/2)\mathbf{q}, \tag{3.23}$$

with $\mathbf{Q}, \mathbf{q}$ as defined in (3.21) and (3.22). If the matrix $\mathbf{Q}$ in (3.23) is invertible, the solution to (3.23) and, in turn, to the GTVMin instance (3.17) is unique and given by $\widehat{\mathbf{w}} = (-1/2)\mathbf{Q}^{-1}\mathbf{q}$.



Question 4

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #2** in the "FL Design Principle" assignment.*

The coding assignment requires you to study the connectivity of the empirical graph whose nodes are FMI weather stations. For which value of the parameter "nrneighbors" is the empirical graph connected?

- a. 1
- b. 2
- c. 3
- d. 4 ✔

Your answer is correct.

Please, join Slack channel (link) if you have any questions regarding the assignment.

The correct answer is:
4

```
The minimum number of nearest neighbors is 1, the graph is connected: False
The minimum number of nearest neighbors is 2, the graph is connected: False
The minimum number of nearest neighbors is 3, the graph is connected: False
The minimum number of nearest neighbors is 4, the graph is connected: True
```

Question 5

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #3** in the "FL Design Principle" assignment.*

The coding assignment required you to learn local model parameters by solving an instance of GTV minimization for different values of the regularization parameter α. How does the average local loss (training error) of the learnt model parameter vary with increasing α ?

- a. The training error cannot increase with increasing α.
- b. The training error might increase with increasing α. ✔

Your answer is correct.

Increasing the GTVMin parameter $alpha$ forces the solutions of GTVMin to have a smaller total variation, i.e., to be approximately constant at connected nodes of the empirical graph. This restriction of local model parameters might counter-act the minimization of the local average loss (training error). Note that GTVMin is an instance of regularized empirical risk minimization. For more discussion of the effect of increasing regularization strength see Section 7.1 of this book.

The correct answer is:
The training error might increase with increasing α.

```
Alpha value: 1
Training error: 33.72827289106725
Total variation: 293.6653233718288

Alpha value: 10
Training error: 36.96204335171303
Total variation: 66.95593556679525

Alpha value: 100
Training error: 45.5749141510435
Total variation: 5.513418883106896

Alpha value: 1000
Training error: 50.849808139889184
Total variation: 0.0889046331357756
```

3. Gradient Methods

Question 1

Flag question Mark 3.00 out of 3.00 Correct

Consider the objective function

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{q}^T\mathbf{w}$$

with some positive semi-definite matrix $\mathbf{Q}$ and some vector $\mathbf{q}$. This function is differentiable with gradient $\nabla f(\mathbf{w})$. Which of the following options are correct expressions for $\nabla f(\mathbf{w})$?

- a. $\mathbf{Q}\mathbf{w}$
- b. $2\mathbf{Q}\mathbf{w} + \mathbf{q}$ ✔
- c. $\mathbf{Q}\mathbf{q}$

Your answer is correct.

See the matrix cookbook: https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

The correct answer is:
$2\mathbf{Q}\mathbf{w} + \mathbf{q}$

Lecture - "Gradient Methods", Gradient Descent

Gradient-based methods are iterative algorithms for finding the minimum of a differentiable objective function $f(\mathbf{w})$ of vector$\mathbf{w}$ (which could be model parameters in a ML method). Unless stated otherwise, we consider an objective function of form

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{q}^T\mathbf{w}. \tag{4.1}$$

Given model parameters $\mathbf{w}^{(k)}$, we want to update them towards a minimum of (4.1). To this end, we use the gradient $\nabla f\left(\mathbf{w}^{(k)}\right)$ to locally approximate $f(\mathbf{w})$ (see Figure 4.1). The gradient $\nabla f\left(\mathbf{w}^{(k)}\right)$ indicates the direction in which the function $f(\mathbf{w})$ maximally increases. Therefore, it seems reasonable to update $\mathbf{w}^{(k)}$ in the opposite direction of $\nabla f\left(\mathbf{w}^{(k)}\right)$,

$$\begin{aligned}\mathbf{w}^{(k+1)} &:= \mathbf{w}^{(k)} - \eta\nabla f\left(\mathbf{w}^{(k)}\right)\\ &\overset{(4.1)}{=} \mathbf{w}^{(k)} - \eta\left(2\mathbf{Q}\mathbf{w}^{(k)} + \mathbf{q}\right).\end{aligned} \tag{4.2}$$

Question 2

Flag question Mark 3.00 out of 3.00 Correct

Consider gradient descent using a fixed **positive** learning rate η to learn the parameters of a linear model by minimizing the average loss on a training set. In other words, we use gradient descent to solve ERM for linear regression. Which statements are correct?

- a. Gradient descent will always converge, for any training set, to a solution of ERM as long as $\eta < 1$.
- b. For a given training set, there is a threshold γ such that gradient descent converges for all learning rates $\eta < \gamma$. This threshold depends only on the label values of the data points in the training set but not on their features.
- c. Gradient descent can only converge if there is a unique solution to ERM, i.e., only if there is a unique choice for the model parameters resulting in minimum training error.
- d. For a given training set, there is a threshold γ such that gradient descent converges for all learning rates $\eta < \gamma$. This threshold depends only on the features of the data points in the training set but not on their label values. ✔

Your answer is correct.

See the discussion in Section 4.4. of the lecture notes.

The correct answer is: For a given training set, there is a threshold γ such that gradient descent converges for all learning rates $\eta < \gamma$. This threshold depends only on the features of the data points in the training set but not on their label values.

a. is False because there is no specific real value that always guarantee convergence if learning rate is smaller than this threshold. Convergence depends a lot on the labels and dataset, and the loss function.

c. is False. Imagine an objective function that has two local minima that minimize empirical risk (ERM). Now, which minima that the GD converges to would highly depend on the initial position of the descending point. So GD can converge even when there are many solutions to ERM.

b and d have the same idea in the first part, which is correct. Proof is inside
Lecture - "Gradient Methods", Learning rate

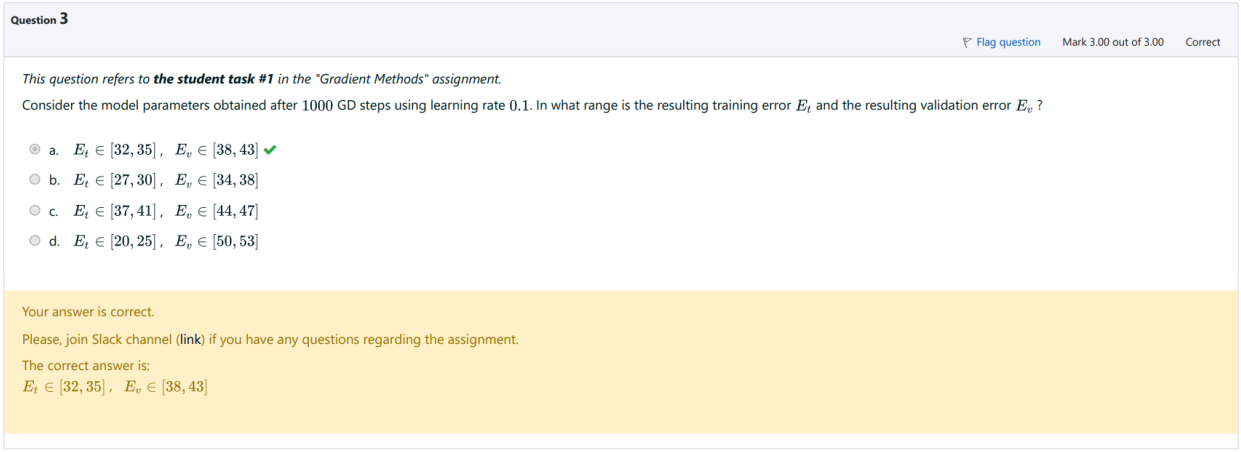
Carefully note that the formula (4.10) is valid only if the matrix $\mathbf{Q}$ in (4.1) is invertible, i.e., if $\lambda_1 > 0$. If the matrix $\mathbf{Q}$ is singular ($\lambda_1 = 0$), the convergence of (4.2) towards a solution of (4.1) is much slower than the decrease of the bound (4.10). However, we can still ensure converge of gradient steps by using a fixed learning rate $\eta_k = \eta$ that satisfies [47, Thm. 2.1.14]

$$0 < \eta < 1/\lambda_d(\mathbf{Q}). \tag{4.11}$$

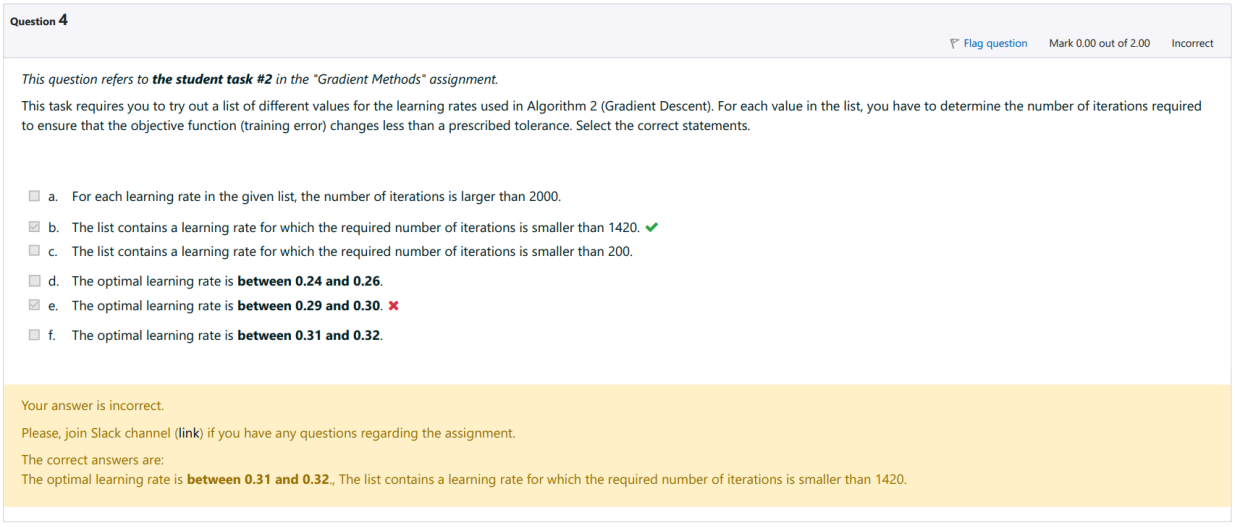
It is interesting to note that for linear regression, the matrix $\mathbf{Q}$ depends only on the features $\mathbf{x}^{(r)}$ of the data points in the training set (see (2.17)) but not on their labels $y^{(r)}$. Thus, the convergence of gradient steps is only affected by the features, whereas the labels are irrelevant. The same is true for ridge regression and GTVMin (using local linear models).

where gamma in question (b) and (d) is 1/ eigenvalue_d of matrix Q. This matrix Q only depends on the feature sets and not on the labels

=> (b) is False and (d) is True




```
Xtrain shape is: (100, 7)
ytrain shape is: (100, 1)
w shape is: (7, 1)
Training error (GD): 34.714123131613945
Validation error (GD): 41.646992336370175
```




```
lrate: 0.28, Converging k: 1567, Obj. value: 38.12430754444085, Etrain: 34.48, Eval: 41.37
lrate: 0.28210526315789475, Converging k: 1556, Obj. value: 38.124307544368875, Etrain: 34.48, Eval: 41.37
lrate: 0.2842105263157895, Converging k: 1545, Obj. value: 38.124307544313545, Etrain: 34.48, Eval: 41.37
lrate: 0.28631578947368425, Converging k: 1534, Obj. value: 38.12430754427454, Etrain: 34.48, Eval: 41.37
lrate: 0.28842105263157897, Converging k: 1523, Obj. value: 38.1243075442516, Etrain: 34.48, Eval: 41.37
lrate: 0.2905263157894737, Converging k: 1513, Obj. value: 38.124307544145495, Etrain: 34.48, Eval: 41.37
lrate: 0.29263157894736846, Converging k: 1502, Obj. value: 38.124307544153595, Etrain: 34.48, Eval: 41.37
lrate: 0.2947368421052632, Converging k: 1492, Obj. value: 38.12430754407772, Etrain: 34.48, Eval: 41.37
lrate: 0.2968421052631579, Converging k: 1482, Obj. value: 38.1243075440166, Etrain: 34.48, Eval: 41.37
lrate: 0.29894736842105263, Converging k: 1472, Obj. value: 38.12430754396992, Etrain: 34.48, Eval: 41.37
lrate: 0.3010526315789474, Converging k: 1463, Obj. value: 38.12430754383858, Etrain: 34.48, Eval: 41.37
lrate: 0.3031578947368421, Converging k: 1453, Obj. value: 38.1243075438196, Etrain: 34.48, Eval: 41.37
lrate: 0.30526315789473685, Converging k: 1443, Obj. value: 38.124307543814346, Etrain: 34.48, Eval: 41.37
lrate: 0.30736842105263157, Converging k: 1434, Obj. value: 38.1243075437234, Etrain: 34.48, Eval: 41.37
lrate: 0.30947368421052635, Converging k: 1425, Obj. value: 38.12430754364555, Etrain: 34.48, Eval: 41.37
lrate: 0.31157894736842107, Converging k: 1416, Obj. value: 38.12430754358045, Etrain: 34.48, Eval: 41.37
lrate: 0.3136842105263158, Converging k: 1407, Obj. value: 38.124307543527756, Etrain: 34.48, Eval: 41.37
lrate: 0.3157894736842105, Converging k: 5036, Obj. value: 38.12430755842261, Etrain: 34.48, Eval: 41.37
```

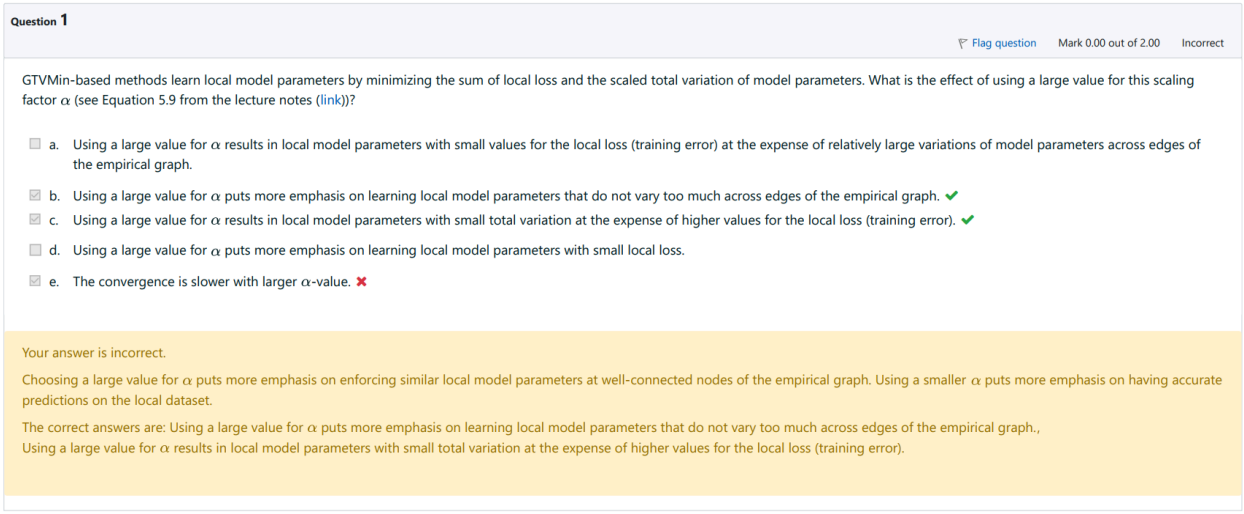
a) False, there are many converging k < 2000

b) True, which is where lrate = 0.31368 and 0.31157

c) False, all of them having k converge >= 1407

(d), (e), (f), only (f) is correct, where optimal learning rate is the one corresponding to smallest converge k, which is 1407. The optimal learning rate is thus 0.313684, inside [0.31, 0.32]

4. FL Algorithms



Question 1

Flag question Mark 0.00 out of 2.00 Incorrect

GTVMin-based methods learn local model parameters by minimizing the sum of local loss and the scaled total variation of model parameters. What is the effect of using a large value for this scaling factor α (see Equation 5.9 from the lecture notes (link))?

a. Using a large value for α results in local model parameters with small values for the local loss (training error) at the expense of relatively large variations of model parameters across edges of the empirical graph.

b. Using a large value for α puts more emphasis on learning local model parameters that do not vary too much across edges of the empirical graph. ✔

c. Using a large value for α results in local model parameters with small total variation at the expense of higher values for the local loss (training error). ✔

d. Using a large value for α puts more emphasis on learning local model parameters with small local loss.

e. The convergence is slower with larger α-value. ✖

Your answer is incorrect.

Choosing a large value for α puts more emphasis on enforcing similar local model parameters at well-connected nodes of the empirical graph. Using a smaller α puts more emphasis on having accurate predictions on the local dataset.

The correct answers are: Using a large value for α puts more emphasis on learning local model parameters that do not vary too much across edges of the empirical graph.,
Using a large value for α results in local model parameters with small total variation at the expense of higher values for the local loss (training error).

We can deal with (e) first. **(e) is False** because larger alpha value means that the GD will quickly tries to reduce the total variation between the edges. Choosing a large value for α puts more emphasis on enforcing similar local model parameters across the edges. This means that the large α can lead to faster convergence in terms of reaching a consensus across the network, because the regularization term that penalizes the difference between connected nodes' parameters has a stronger influence. The nodes are 'encouraged' to agree with each other more strongly, which might lead to a quicker agreement across the network. A large α can smooth out variations due to noisy local gradients, which might otherwise lead to large, oscillating updates. This smoothing effect could help stabilize convergence, as the updates become more consistent across iterations.

Lecture - "FL Algorithms", Message Passing Implementation

The iterate $\mathbf{w}^{(k)}$ contains local model parameters $\mathbf{w}^{(i,k)}$,

$$\mathbf{w}^{(k)} =: \mathrm{stack}\big\{\mathbf{w}^{(i,k)}\big\}_{i=1}^{n}. \tag{5.8}$$

Using (5.1), we can express the gradient $\nabla f\big(\mathbf{w}^{(k)}\big)$ and the corresponding gradient step as (see Figure 5.1)

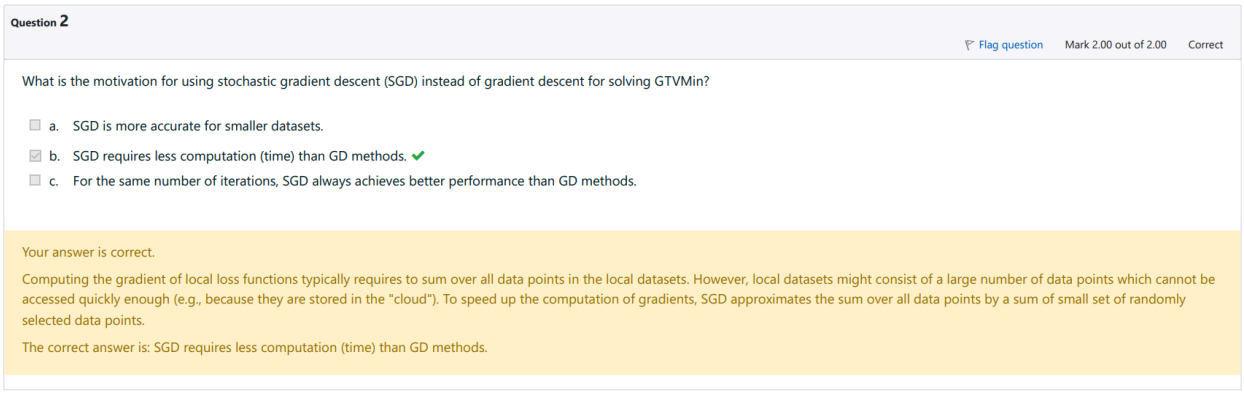
$$\mathbf{w}^{(i,k+1)} := \mathbf{w}^{(i,k)} + \eta\Bigg[\underbrace{(2/m_i)\big(\mathbf{X}^{(i)}\big)^T\big(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i,k)}\big)}_{\text{(I)}} + \underbrace{2\alpha\sum_{i' \in \mathcal{V}\setminus\{i\}} A_{i,i'}\big(\mathbf{w}^{(i',k)} - \mathbf{w}^{(i,k)}\big)}_{\text{(II)}}\Bigg]. \tag{5.9}$$

The update (5.9) consists of two components, denoted (I) and (II). The purpose of component (I) is to minimize the local loss, i.e., to learn local model parameters with small deviation between labels $y^{(i,r)}$ and the predictions $\left(\mathbf{w}^{(i,k+1)}\right)^{T}\mathbf{x}^{(i,r)}$. Note that we can rewrite the component (I) in (5.9) as

$$(2/m_i)\sum_{r=1}^{m_i}\mathbf{x}^{(i,r)}\left(y^{(i,r)}-\left(\mathbf{x}^{(i,r)}\right)^{T}\mathbf{w}^{(i,k)}\right). \tag{5.10}$$

The purpose of component (II) in (5.9) is to force the local model parameters to be similar across an edge $\{i,i'\}$ with large weight $A_{i,i'}$. We control the relative importance of (II) and (I) using the GTVMin parameter α: Choosing a large value for α puts more emphasis on enforcing similar local model parameters across the edges. Using a smaller α puts more emphasis on learning local model parameters delivering accurate predictions (incurring a small loss) on the local dataset.

From this passage, it is clear that **(a)(d) are False and (b)(c) are True**.



Lecture - "FL Algorithms", FedSGD

For some applications, it might be infeasible to compute the sum (5.10) exactly for each gradient step. For example, local datasets might consist of a large number of data points which cannot be accessed quickly enough (stored in the cloud). It might then be useful to approximate the sum by

$$\underbrace{(2/B)\sum_{r\in\mathcal{B}} \mathbf{x}^{(i,r)}\left(y^{(i,r)} - \left(\mathbf{x}^{(i,r)}\right)^{T}\mathbf{w}^{(i,k)}\right)}_{\approx(5.10)}. \tag{5.11}$$

The approximation (5.11) uses a subset (so called "batch")

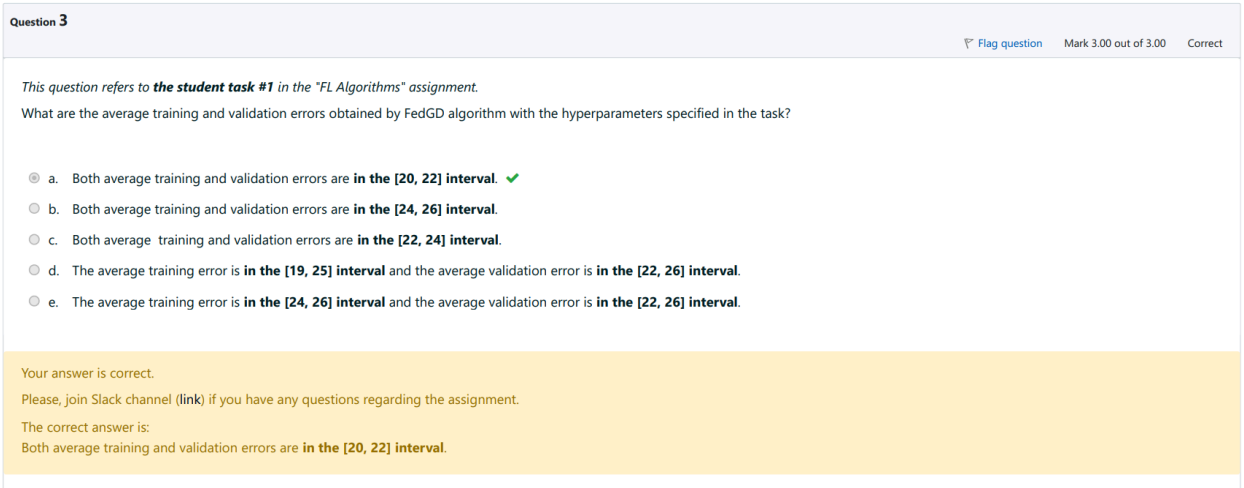
$$\mathcal{B} = \left\{\left(\mathbf{x}^{(r_1)}, y^{(r_1)}\right), \ldots, \left(\mathbf{x}^{(r_B)}, y^{(r_B)}\right)\right\}$$

of B randomly chosen data points from $\mathcal{D}^{(i)}$. While (5.10) requires summing over m data points, the approximation requires to to sum over B (typically $B \ll m$) data points.

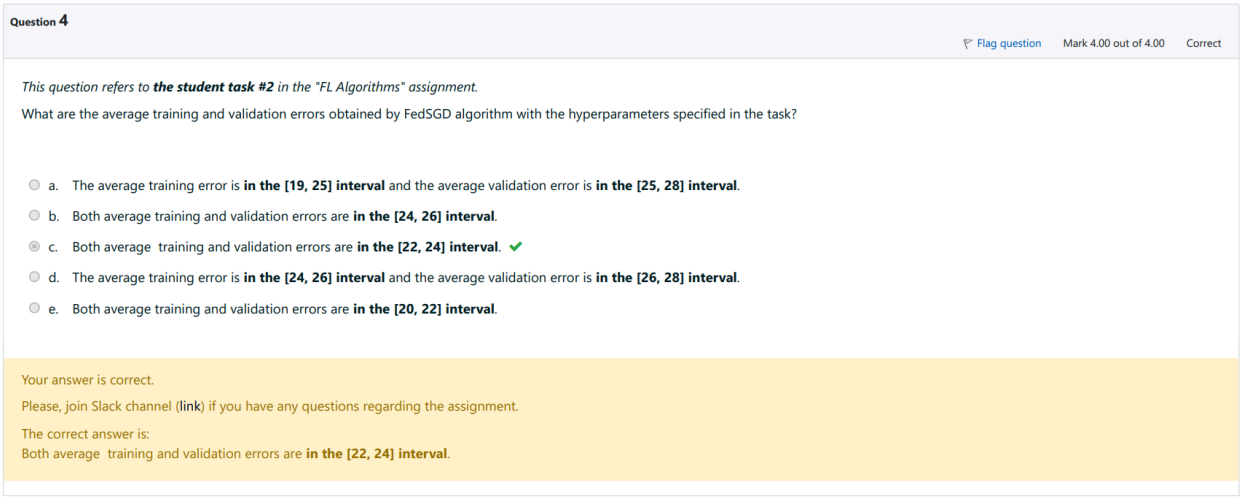
From this, **(b) is clearly True.** However, (a) and (c) are not mentioned anywhere, so we need some logic to think about them

a is False. The accuracy of SGD compared to GD is not inherently better for smaller datasets. In fact, because SGD estimates the gradient based on a single sample or a small batch of samples, it can be noisier than the gradient computed over the entire dataset, as is done in GD.

c is False. It is not guaranteed that SGD will always outperform GD for the same number of iterations. SGD can sometimes converge faster to a good solution or escape local minima due to its noisy gradient updates, but it doesn't always lead to better performance, especially if we define performance in terms of reaching the exact minimum of the loss function.



```
The average training error: 21.065277715408104
The average validation error: 21.336245897848386
```



Question 4

Flag question Mark 4.00 out of 4.00 Correct

*This question refers to **the student task #2** in the "FL Algorithms" assignment.*

What are the average training and validation errors obtained by FedSGD algorithm with the hyperparameters specified in the task?

a. The average training error is **in the [19, 25] interval** and the average validation error is **in the [25, 28] interval**.
b. Both average training and validation errors are **in the [24, 26] interval**.
c. Both average training and validation errors are **in the [22, 24] interval**. ✔
d. The average training error is **in the [24, 26] interval** and the average validation error is **in the [26, 28] interval**.
e. Both average training and validation errors are **in the [20, 22] interval**.

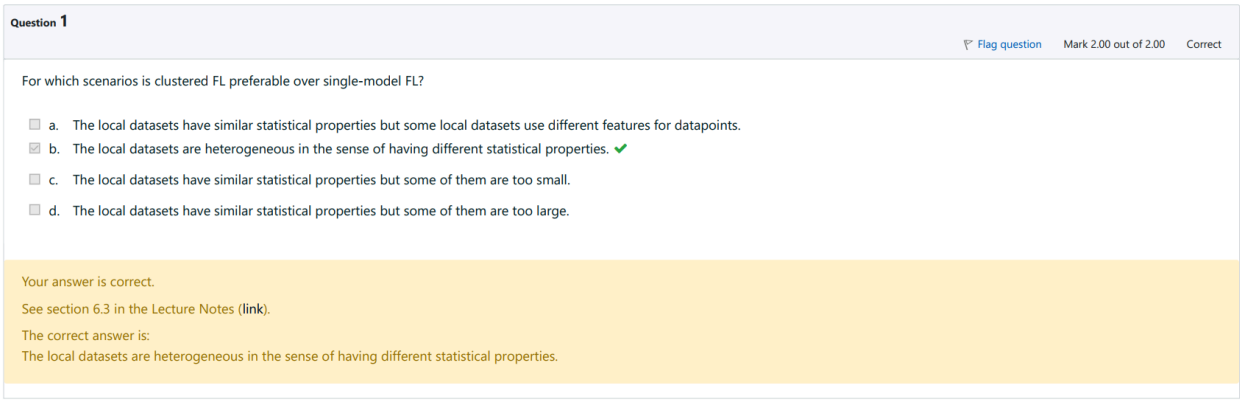
Your answer is correct.

Please, join Slack channel (link) if you have any questions regarding the assignment.

The correct answer is:
Both average training and validation errors are **in the [22, 24] interval**.

```
The average training error: 21.897922405451236
The average validation error: 22.103452061523004
```

5. FL Flavors



Question 1

Flag question Mark 2.00 out of 2.00 Correct

For which scenarios is clustered FL preferable over single-model FL?

a. The local datasets have similar statistical properties but some local datasets use different features for datapoints.
b. The local datasets are heterogeneous in the sense of having different statistical properties. ✔
c. The local datasets have similar statistical properties but some of them are too small.
d. The local datasets have similar statistical properties but some of them are too large.

Your answer is correct.

See section 6.3 in the Lecture Notes (link).

The correct answer is:
The local datasets are heterogeneous in the sense of having different statistical properties.

Lecture - "FL Flavors", Clustered FL

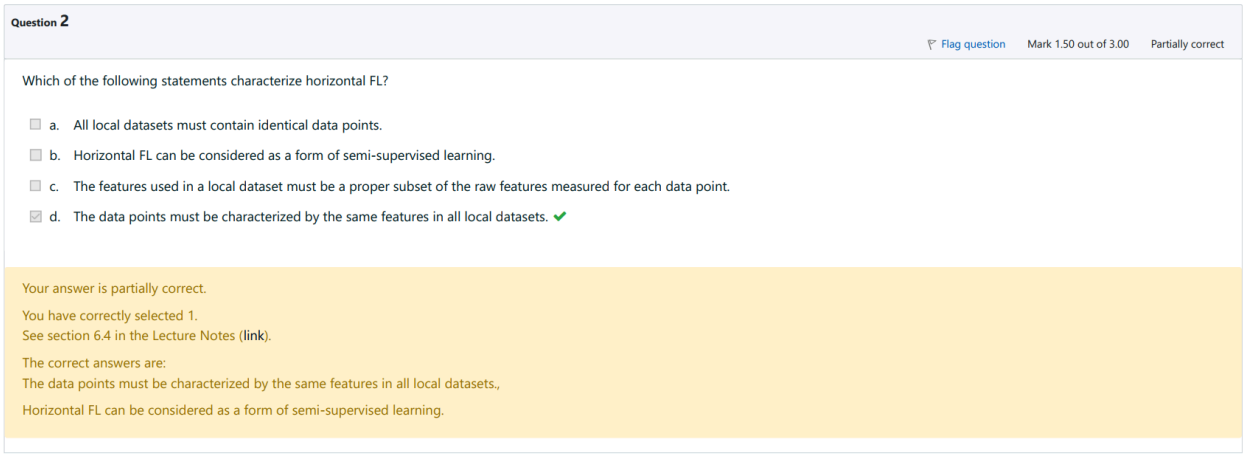
a) is False. The local datasets have similar statistical properties but some local datasets use different features for datapoints. When local datasets use different features for datapoints, the scenario typically uses vertical FL, where different datasets Di hold different pieces of information about the same entities.

b) is True. The local datasets are heterogeneous in the sense of having different statistical properties.

Single-model FL systems require the local datasets to be well approximated as i.i.d. realizations from a common underlying probability distribution. However, requiring homogeneous local datasets, generated from the same probability distribution, might be overly restrictive. Indeed, the local datasets might be heterogeneous and need to be modelled using different probability distribution [17, 32].

c) is False. The local datasets have similar statistical properties but some of them are too small.
d) is False. The local datasets have similar statistical properties but some of them are too large.

The size of the local datasets (whether too small or too large) does not directly influence the preference for clustered FL over single-model FL. Instead, the key factor is the heterogeneity in the statistical properties of the datasets.



Lecture - "FL Flavors", Horizontal FL

a) is False. All local datasets may have different data points. This is a characteristic of HFL. In fact, HFL specifically deals with the scenario where datasets across different nodes contain different data points but share the same feature space.

Horizontal FL uses local datasets $\mathcal{D}^{(i)}$, for $i \in \mathcal{V}$, that contain data points characterized by the same features [52]. We can think of each local dataset $\mathcal{D}^{(i)}$ as being a subset (or batch) of an underlying global dataset

$$\mathcal{D}^{(\text{global})} := \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(m)}, y^{(m)}\right) \right\}.$$

b) is True. Horizontal FL can be considered as a form of semi-supervised learning.

We can interpret horizontal FL as a generalization of semi-supervised learning (SSL) [53]: For some local datasets $i \in \mathcal{U}$ we might not have access to the label values of data points. Still, we can use the features of the data

c) is False. The features used in a local dataset must contain all raw features measured for each data point, not just a proper subset. In Horizontal FL, the datasets across all participating nodes are expected to have the same set of features, not subsets of features.

d) is True. The data points must be characterized by the same features in all local datasets.

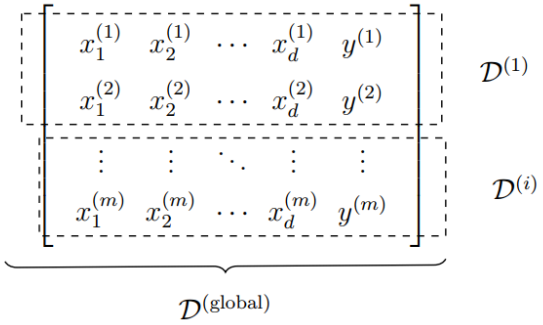


Figure 6.4: Horizontal FL uses the same features to characterize data points in different local datasets. Different local datasets are constituted by different subsets of an underlying global dataset.

Question 3

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #1** in the "FL Main Flavors" assignment.*

What is the average squared loss over all nodes for a K-Means clustering with the latitude and longitude values of the FMI stations as a representative vector?

- a. The average squared loss lies **in [41, 50]**.
- b. The average squared loss lies **in [70, 80]**.
- c. The average squared loss lies **in [35, 40]**. ✔
- d. The average squared loss lies **in [51, 60]**.
- e. The average squared loss lies **in [30, 33]**.
- f. The average squared loss lies **in [80, 90]**.

Your answer is correct.

Please, join Slack channel (link) to discuss the assignment questions.

The correct answer is:
The average squared loss lies **in [35, 40]**.

```
The average squared loss over all datapoints is 36.419995057609505
```

Question 4

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #2** in the "FL Main Flavors" assignment.*

What is the average squared loss over all nodes for a K-Means clustering with the fitted GMM's parameters as a representative vector?

- a. The average squared loss lies **in [60, 70]**.
- b. The average squared loss lies **in [36, 45]**. ✔
- c. The average squared loss lies **in [15, 25]**.
- d. The average squared loss lies **in [4, 10]**.
- e. The average squared loss lies **in [50, 60]**.
- f. The average squared loss lies **in [0, 5]**.

Your answer is correct.

Please, join Slack channel (link) to discuss the assignment questions.

The correct answer is:
The average squared loss lies **in [36, 45]**.

```
The average squared loss over all datapoints is 37.87456560645458
```

Question 5

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #3** in the "FL Main Flavors" assignment.*

What is the average squared loss over all nodes for a K-Means clustering with the eigenvectors of the Laplacian matrix as a representative vector?

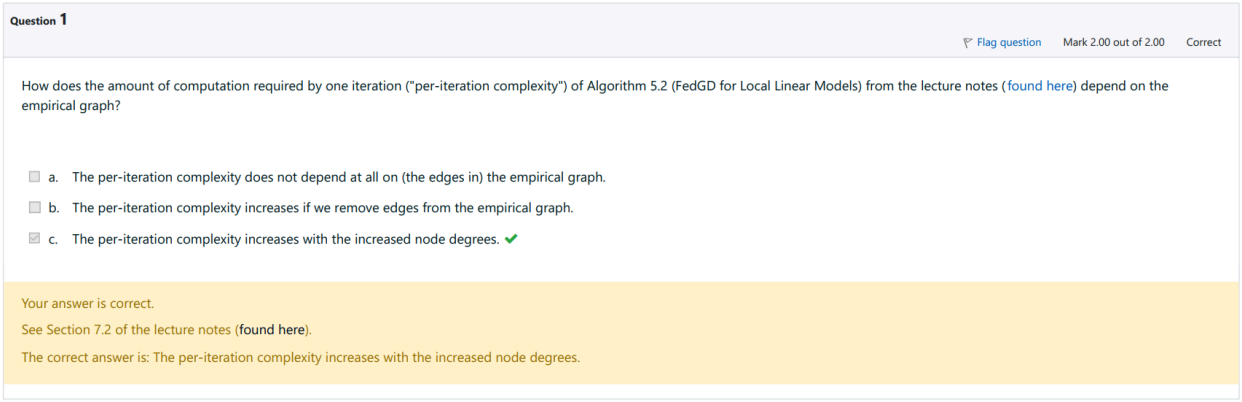
- a. The average squared loss lies **in [25, 30]**.
- b. The average squared loss lies **in [35, 40]**. ✔
- c. The average squared loss lies **in [90, 100]**.
- d. The average squared loss lies **in [80, 90]**.
- e. The average squared loss lies **in [50, 55]**.
- f. The average squared loss lies **in [45, 50]**.

Your answer is correct.

The correct answer is:
The average squared loss lies **in [35, 40]**.

```
The average squared loss over all datapoints is 36.539662249802106
```

6. Graph Learning



Question 1

Flag question Mark 2.00 out of 2.00 Correct

How does the amount of computation required by one iteration ("per-iteration complexity") of Algorithm 5.2 (FedGD for Local Linear Models) from the lecture notes (found here) depend on the empirical graph?

- a. The per-iteration complexity does not depend at all on (the edges in) the empirical graph.
- b. The per-iteration complexity increases if we remove edges from the empirical graph.
- c. The per-iteration complexity increases with the increased node degrees. ✔

Your answer is correct.

See Section 7.2 of the lecture notes (found here).

The correct answer is: The per-iteration complexity increases with the increased node degrees.

Algorithm 5.1 FedGD for Local Linear Models

Input: empirical graph $\mathcal{G}$; GTV parameter α; learning rate $\eta_{k,i}$ local dataset $\mathcal{D}^{(i)} = \left\{\left(\mathbf{x}^{(i,1)}, y^{(i,1)}\right), \ldots, \left(\mathbf{x}^{(i,m_i)}, y^{(i,m_i)}\right)\right\}$ for each i; some stopping criterion.

Output: linear model parameters $\widehat{\mathbf{w}}^{(i)}$ for each node $i \in \mathcal{V}$

Initialize: $k := 0$; $\mathbf{w}^{(i,0)} := \mathbf{0}$

1: **while** stopping criterion is not satisfied **do**
2: **for** all nodes $i \in \mathcal{V}$ (simultaneously) **do**
3: share local model parameters $\mathbf{w}^{(i,k)}$ with neighbours $i' \in \mathcal{N}^{(i)}$
4: update local model parameters via (5.9)
5: **end for**
6: increment iteration counter: $k := k+1$
7: **end while**
8: $\widehat{\mathbf{w}}^{(i)} := \mathbf{w}^{(i,k)}$ for all nodes $i \in \mathcal{V}$

Inserting (5.1) into (5.7), we obtain the gradient step

$$\mathbf{w}^{(i,k+1)} := \mathbf{w}^{(i,k)} + \eta \Big[\underbrace{(2/m_i)\big(\mathbf{X}^{(i)}\big)^T\big(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i,k)}\big)}_{\text{(I)}} + \underbrace{2\alpha \sum_{i' \in \mathcal{V}\setminus\{i\}} A_{i,i'}\big(\mathbf{w}^{(i',k)} - \mathbf{w}^{(i,k)}\big)}_{\text{(II)}} \Big]. \tag{5.9}$$

The update (5.9) consists of two components, denoted (I) and (II). The component (I) is nothing but the negative gradient $-\nabla L_i\left(\mathbf{w}^{(i,k)}\right)$ of the local loss $L_i\left(\mathbf{w}^{(i)}\right) := (1/m_i)\left\|\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i)}\right\|_2^2$. Component (I) drives the local model parameters $\mathbf{w}^{(i,k+1)}$ towards the minimum of $L_i(\cdot)$, i.e., having a small deviation between labels $y^{(i,r)}$ and the predictions $\left(\mathbf{w}^{(i,k+1)}\right)^T\mathbf{x}^{(i,r)}$. Note that we can rewrite the component (I) in (5.9), as

$$(2/m_i)\sum_{r=1}^{m_i}\mathbf{x}^{(i,r)}\big(y^{(i,r)} - \big(\mathbf{x}^{(i,r)}\big)^T\mathbf{w}^{(i,k)}\big). \tag{5.10}$$

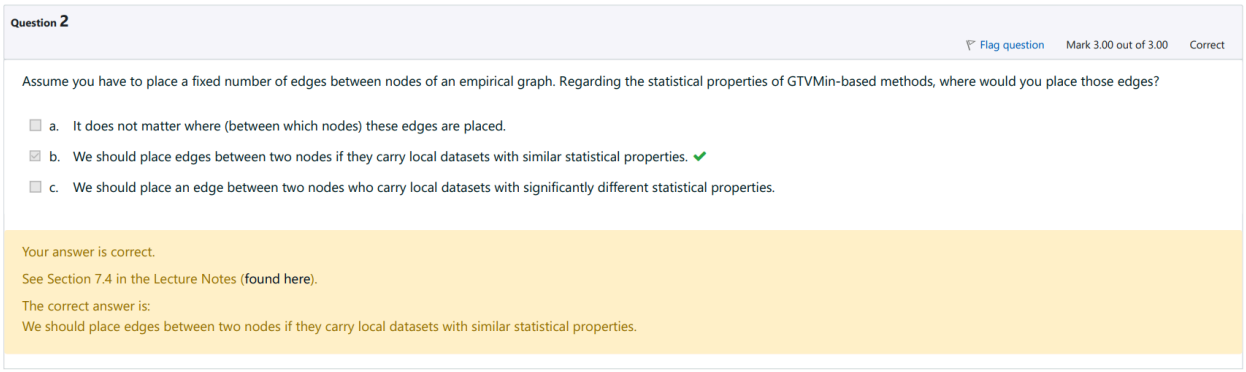
The purpose of component (II) in (5.9) is to force the local model parameters to be similar across an edge $\{i, i'\}$ with large weight $A_{i,i'}$. We control the relative importance of (II) and (I) using the GTVMin parameter α: Choosing a large value for α puts more emphasis on enforcing similar local model parameters across the edges. Using a smaller α puts more emphasis on learning local model parameters delivering accurate predictions (incurring a small loss) on the local dataset.

Computational Properties. The computational complexity of Algorithm 5.1 depends on the amount of computation required by a single iteration of its steps (3) - (4). Clearly, this "per-iteration" complexity of Algorithm 5.1 increases with increasing node degrees $d^{(i)}$. Indeed, the step (3) requires to communicate local model parameters across each edge of the empirical graph. This communication can be implemented by different communication channels such as a short-range wireless link or an optical fibre cable [60, 61].

According to (4.6), the convergence speed of the gradient steps (5.9) used in Algorithm 5.1 depends on the condition number $\lambda_{nd}(\mathbf{Q})/\lambda_1(\mathbf{Q})$ of the matrix $\mathbf{Q}$ in (5.2). Algorithm 5.1 tends to require fewer iterations when the ratio between the largest and smallest eigenvalues of $\mathbf{Q}$ is small (closer to 1). The condition number of $\mathbf{Q}$ tends to be smaller for a smaller ratio between the maximum node degree $d_{\max}$ and eigenvalue $\lambda_2\big(\mathbf{L}^{(\mathcal{G})}\big)$ (see (5.5) and (5.6)).

To summarize, the per-iteration complexity of Algorithm 5.1 increases with the node degrees $d^{(i)}$ of the empirical graph $\mathcal{G}$. On the other hand, the number of iterations required by Algorithm 5.1 decrease with increasing $\lambda_2\big(\mathbf{L}^{(\mathcal{G})}\big)$. Some recent work studies graph constructions that maximize $\lambda_2\big(\mathbf{L}^{(\mathcal{G})}\big)$ for a given (prescribed) maximum node degree $d_{\max} = \max_{i \in \mathcal{V}} d^{(i)}$ [62, 63].

The per iteration complexity of algorithm 5.1 increases with node degrees of empirical graph G => (a) (b) are False and (c) is True



The whole idea of GTVMin is to enforce similar model parameters at two nodes i, i' that are connected by an edge $\{i, i'\}$ with (relatively) large edge weight $A_{i,i'}$. In general, the edges (and their weights) of the empirical graph are a design choice. However, the edges should somehow reflect the actual similarities between local datasets. We next discuss different approaches to measuring the similarity or, equivalently, the discrepancy (the lack of similarity) between two local datasets.

Assume we have constructed a useful measure $d^{(i,i')} \in \mathbb{R}_+$ for the discrepancy between any two local datasets $\mathcal{D}^{(i)}, \mathcal{D}^{(i')}$. We could then construct an empirical graph by connecting each node i with its nearest neighbours. The nearest neighbours of i are those other nodes $i' \in \mathcal{V} \setminus \{i\}$ with smallest discrepancy $d^{(i,i')}$. We next discuss an alternative to the nearest-neighbour graph construction. This alternative approach formulates graph learning as a constrained linear optimization problem.

From these two paragraphs, (a) and (c) are False, and (b) is True

Question 3

Flag question Mark 3.00 out of 3.00 Correct

*This question refers to **the student tasks #1-3** in the "Graph Learning" assignment.*

For which *node_degree* values (*node_degree* parameter in the *add_edges* function) the graphs ($\mathcal{G}^{(I)}$, $\mathcal{G}^{(II)}$, and $\mathcal{G}^{(III)}$) are connected?

P.S. The empirical graph notation corresponds to the Section 7.5 in the Lecture Notes.

- [x] a. The connectivity of $\mathcal{G}^{(II)}$ depends on the random seed. ✔
- [ ] b. The connectivity of $\mathcal{G}^{(II)}$ does not depend on the random seed.
- [ ] c. $\mathcal{G}^{(III)}$ is connected for **some** *node_degree* < 2.
- [x] d. $\mathcal{G}^{(I)}$ is connected for **all** *node_degree* ≥ 8. ✖
- [x] e. $\mathcal{G}^{(I)}$ is connected for for **some** *node_degree* values < 8. ✖
- [ ] f. $\mathcal{G}^{(III)}$ is connected for **all** *node_degree* ≥ 2.

Your answer is correct.

Unfortunately, the correctness of $\mathcal{G}^{(I)}$ and $\mathcal{G}^{(III)}$ connectivity cannot be checked due to the different environments. Meanwhile, $\mathcal{G}^{(II)}$ connectivity depends on the random seed - for different seeds the minimum node degree required for a graph to be connected is different (from 30 to 70).

Please, join Slack channel (**link**) if you have any questions regarding the coding assignment.

The correct answer is:
The connectivity of $\mathcal{G}^{(II)}$ depends on the random seed.

```
The current seed is: 1000
G_FMI_2 is connected: True with 4 neighbors.
The current seed is: 2000
G_FMI_2 is connected: True with 3 neighbors.
The current seed is: 3000
G_FMI_2 is connected: True with 4 neighbors.
The current seed is: 4000
G_FMI_2 is connected: True with 5 neighbors.
The current seed is: 5000
G_FMI_2 is connected: True with 4 neighbors.
```

```
G_FMI_1 is connected: False with 1 neighbors.
G_FMI_1 is connected: False with 2 neighbors.
G_FMI_1 is connected: False with 3 neighbors.
G_FMI_1 is connected: False with 4 neighbors.
G_FMI_1 is connected: True with 5 neighbors.
G_FMI_1 is connected: True with 6 neighbors.
G_FMI_1 is connected: True with 7 neighbors.
G_FMI_1 is connected: True with 8 neighbors.
G_FMI_1 is connected: True with 9 neighbors.
G_FMI_1 is connected: True with 10 neighbors.
G_FMI_1 is connected: True with 11 neighbors.
G_FMI_1 is connected: True with 12 neighbors.
G_FMI_1 is connected: True with 13 neighbors.
G_FMI_1 is connected: True with 14 neighbors.
G_FMI_1 is connected: True with 15 neighbors.
```

```
G_FMI_3 is connected: False with 1 neighbors.
G_FMI_3 is connected: False with 2 neighbors.
G_FMI_3 is connected: False with 3 neighbors.
G_FMI_3 is connected: True with 4 neighbors.
G_FMI_3 is connected: True with 5 neighbors.
G_FMI_3 is connected: True with 6 neighbors.
G_FMI_3 is connected: True with 7 neighbors.
G_FMI_3 is connected: True with 8 neighbors.
G_FMI_3 is connected: True with 9 neighbors.
G_FMI_3 is connected: True with 10 neighbors.
G_FMI_3 is connected: True with 11 neighbors.
G_FMI_3 is connected: True with 12 neighbors.
G_FMI_3 is connected: True with 13 neighbors.
G_FMI_3 is connected: True with 14 neighbors.
G_FMI_3 is connected: True with 15 neighbors.
```

Question 4

Flag question Mark 3.00 out of 3.00 Correct

*This question refers to **the student tasks #1-3** in the "Graph Learning" assignment.*

What interval contains training and validation errors of $\mathcal{G}^{(I)}$, $\mathcal{G}^{(II)}$, and $\mathcal{G}^{(III)}$ graphs ($2 * 3 = 6$ errors in total).

P.S. The empirical graph notation corresponds to the Section 7.5 in the Lecture Notes.

- a. [51, 55]
- b. [27, 31]
- c. [15, 19]
- d. [23, 27]
- e. [19, 23] ✔
- f. [43, 47]
- g. [55, 59]
- h. [35, 39]
- i. [39, 43]
- j. [59, 63]
- k. [31, 35]
- l. [47, 51]

Your answer is correct.

Please, join Slack channel (**link**) if you have any questions regarding the coding assignment.

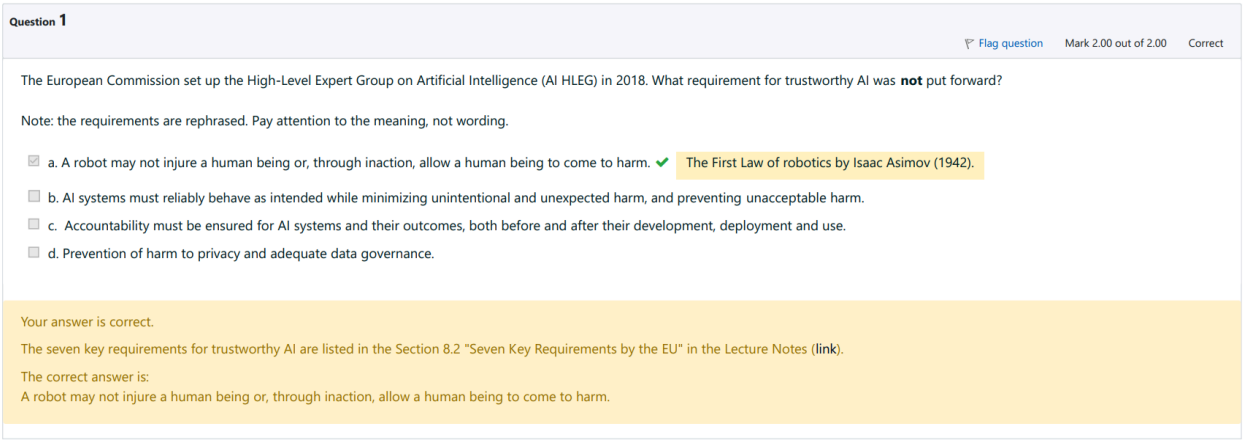
The correct answer is:
[19, 23]

```
The seed is 4740
The average training error for G_FMI_1: 20.68635695529578
The average validation error for G_FMI_1: 21.386552008386015

The average training error for G_FMI_2: 20.042493743748498
The average validation error for G_FMI_2: 20.68321527579531

The average training error for G_FMI_3: 20.05213223348533
The average validation error for G_FMI_3: 20.645772957128727
```

7. Trustworthy AI



Question 1

Flag question Mark 2.00 out of 2.00 Correct

The European Commission set up the High-Level Expert Group on Artificial Intelligence (AI HLEG) in 2018. What requirement for trustworthy AI was **not** put forward?

Note: the requirements are rephrased. Pay attention to the meaning, not wording.

- a. A robot may not injure a human being or, through inaction, allow a human being to come to harm. ✔ The First Law of robotics by Isaac Asimov (1942).
- b. AI systems must reliably behave as intended while minimizing unintentional and unexpected harm, and preventing unacceptable harm.
- c. Accountability must be ensured for AI systems and their outcomes, both before and after their development, deployment and use.
- d. Prevention of harm to privacy and adequate data governance.

Your answer is correct.

The seven key requirements for trustworthy AI are listed in the Section 8.2 "Seven Key Requirements by the EU" in the Lecture Notes (link).

The correct answer is:
A robot may not injure a human being or, through inaction, allow a human being to come to harm.

(a) is made up
(b) is 8.2.2
(c) is 8.2.7
(d) is 8.2.3

8.2.1 KR1 - Human Agency and Oversight.

"*..AI systems should support human autonomy and decision-making, as prescribed by the principle of respect for human autonomy. This requires that AI systems should both act as enablers to a democratic, flourishing and equitable society by supporting the user's agency and foster fundamental rights, and allow for human oversight...*" [71, p.15]

8.2.2 KR2 - Technical Robustness and Safety.

"*...Technical robustness requires that AI systems be developed with a preventative approach to risks and in a manner such that they reliably behave as intended while minimising unintentional and unexpected harm, and preventing unacceptable harm. ...*' [71, p.16].

8.2.3 KR3 - Privacy and Data Governance.

"*..privacy, a fundamental right particularly affected by AI systems. Prevention of harm to privacy also necessitates adequate data governance that covers the quality and integrity of the data used...*" [71, p.17].

8.2.4 KR4 - Transparency.

Traceability. This key requirement includes the documentation of design choices (and underlying business models) for a GTVMin-based FL system. This includes the source for the local datasets, the local models, the local loss function as well as the construction of the empirical graph. Moreover, the documentation should also cover the details of the implemented optimization method used to solve GTVMin. This documentation might also require to periodically store the current model parameters along with a time-stamp ("logging").

8.2.5 KR5 - Diversity, Non-Discrimination and Fairness.

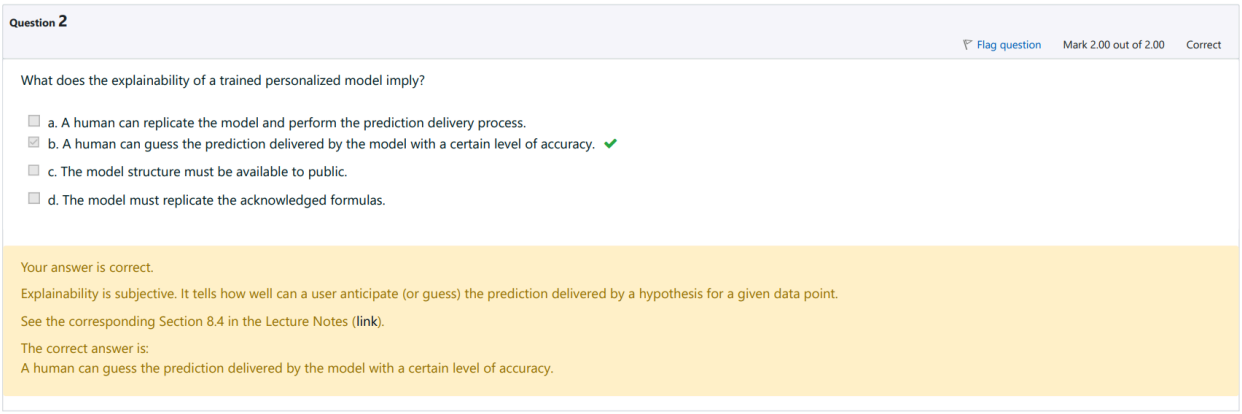
"*...we must enable inclusion and diversity throughout the entire AI system's life cycle...this also entails ensuring equal access through inclusive design processes as well as equal treatment.*" [71, p.18].

8.2.6 KR6 - Societal and Environmental Well-Being.

"*...Sustainability and ecological responsibility of AI systems should be encouraged, and research should be fostered into AI solutions addressing areas of global concern, such as for instance the Sustainable Development Goals.*" [71, p.19].

8.2.7 KR7 - Accountability.

"*...mechanisms be put in place to ensure responsibility and accountability for AI systems and their outcomes, both before and after their development, deployment and use.*" [71, p. 19].



Question 2

Flag question Mark 2.00 out of 2.00 Correct

What does the explainability of a trained personalized model imply?

- a. A human can replicate the model and perform the prediction delivery process.
- b. A human can guess the prediction delivered by the model with a certain level of accuracy. ✔
- c. The model structure must be available to public.
- d. The model must replicate the acknowledged formulas.

Your answer is correct.

Explainability is subjective. It tells how well can a user anticipate (or guess) the prediction delivered by a hypothesis for a given data point.

See the corresponding Section 8.4 in the Lecture Notes (link).

The correct answer is:
A human can guess the prediction delivered by the model with a certain level of accuracy.

Explainability. The transparency of a GTVMin-based FL system also includes the explainability of the trained local models. Section 8.4 discusses quantitative measures for the subjective explainability of a learnt hypothesis. We will also use this measure as a regularizer to obtain GTVMin-bases systems that guarantee subjective explainability "by design".

8.4 Subjective Explainability of FL Systems

Let us now discuss how to ensure key requirement **KR4 - Transparency** in GTVMin-based FL systems. This key requirement includes also the explainability of a trained personalized model $\hat{h}^{(i)} \in \mathcal{H}^{(i)}$ (and their predictions). It is important to note that the explainability of $\hat{h}^{(i)}$ is subjective: A given learnt hypothesis $\hat{h}^{(i)}$ might offer high degree of explainability to one user (a graduate student at a university)but a low degree of explainability to another user (a high-school student). We must ensure explainability for the specific user, which we will also denote by i, that "consumes" the predictions at node $i \in \mathcal{V}$ of the empirical graph.

One particular useful approach to the explainability of trained ML models is based on its simulatability [85–87]: How well can a user anticipate (or guess) the prediction $\hat{y} = \hat{h}^{(i)}(\mathbf{x})$ delivered by $\hat{h}^{(i)}$ for a data point with features $\mathbf{x}$. We can then measure the explainability of $\hat{h}^{(i)}(\mathbf{x})$ to the user at node i by comparing the prediction $\hat{h}^{(i)}(\mathbf{x})$ with the corresponding "guess" (or "simulation") $u^{(i)}(\mathbf{x})$.

We can enforce (subjective) explainability of FL systems by modifying the local loss functions in GTVMin. For ease of exposition we will focus on the GTVMin instance (5.1) for training local (personalized) linear models. For each node $i \in \mathcal{V}$, we construct a test-set $\mathcal{D}_t^{(i)}$ and ask user i to deliver a guess $u^{(i)}(\mathbf{x})$ for each data point in $\mathcal{D}_t^{(i)}$.[17]

We measure the (subjective) explainability of a linear hypothesis with

model parameters $\mathbf{w}^{(i)}$ by

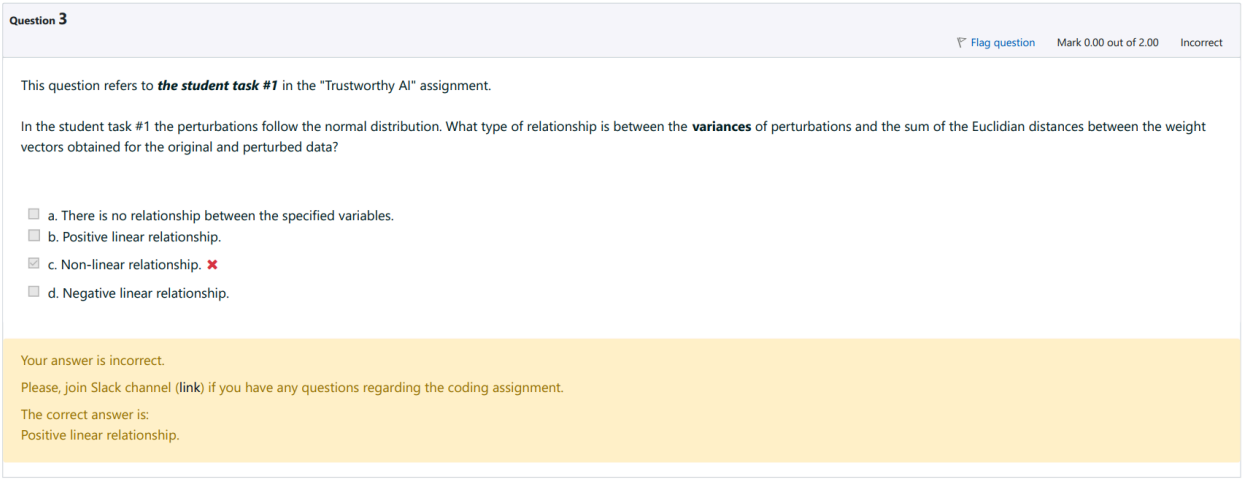
$$(1/|\mathcal{D}_t^{(i)}|) \sum_{\mathbf{x} \in \mathcal{D}_t^{(i)}} \left(u^{(i)}(\mathbf{x}) - \mathbf{x}^T \mathbf{w}^{(i)} \right)^2 . \tag{8.11}$$

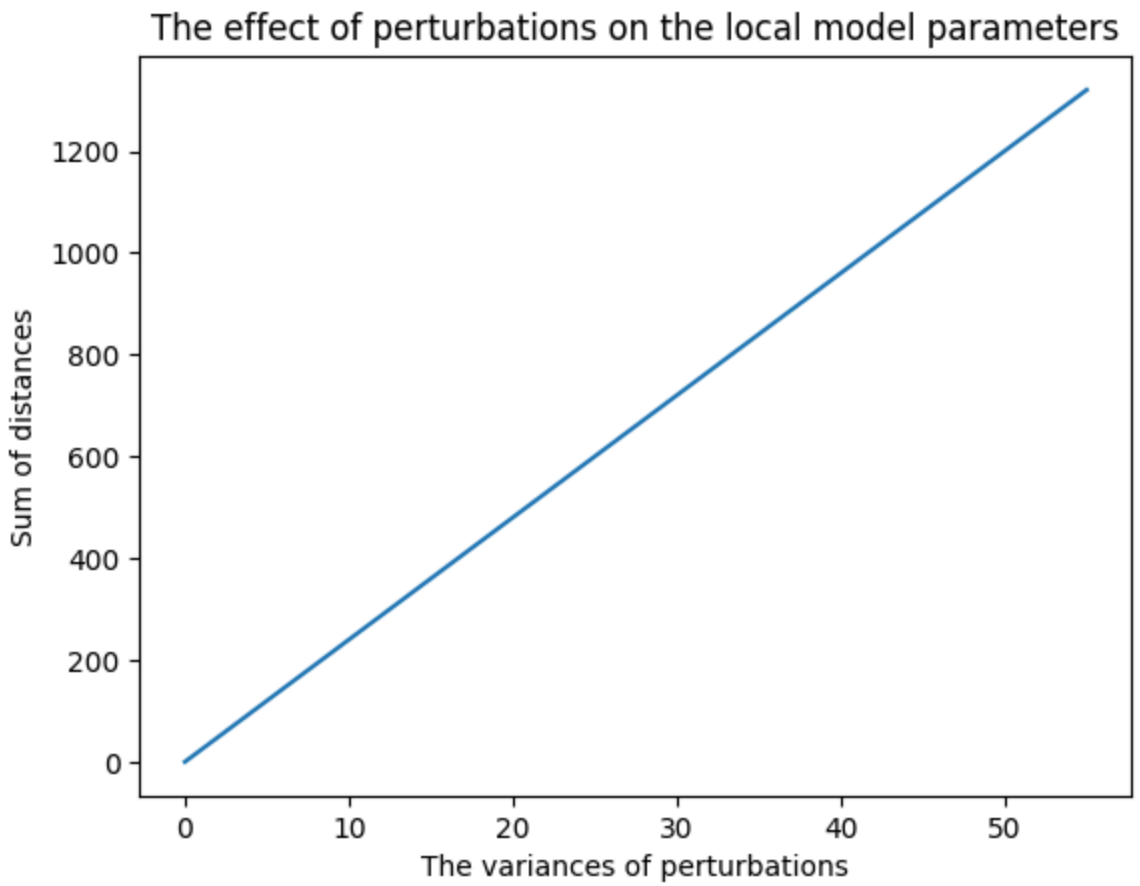
It seems natural to add this measure as a penalty term to the local loss function in (5.1), resulting in the new loss function

$$L_i\left(\mathbf{w}^{(i)}\right) := \underbrace{(1/m_i) \left\| \mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i)} \right\|_2^2}_{\text{training error}} + \rho \underbrace{(1/|\mathcal{D}_t^{(i)}|) \sum_{\mathbf{x} \in \mathcal{D}_t^{(i)}} \left(u^{(i)}(\mathbf{x}) - \mathbf{x}^T \mathbf{w}^{(i)} \right)^2}_{\text{subjective explainability}} . \tag{8.12}$$

The regularization parameter ρ controls the preference for a high subjective explainability of the hypothesis $h^{(i)}(\mathbf{x}) = \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}$ over a small training error [87]. It can be shown that (8.12) is the average weighted squared error loss of $h^{(i)}(\mathbf{x})$ on an augmented version of $\mathcal{D}^{(i)}$. This augmented version includes the data point $\left(\mathbf{x}, u^{(i)}(\mathbf{x})\right)$ for each data point $\mathbf{x}$ in the test-set $\mathcal{D}_t^{(i)}$.

From here, we can see that (c) and (d) are irrelevant. However, (a) and (b) are more difficult to to tell apart. From the text, they say that explainability means how certain a human can make a prediction for a datapoint with feature X. It does not mention that human must have access to the physical model in (a) to make a prediction, but they are only required to guess the output with some certainty. Therefore, only (b) is correct.





Question 4

Flag question Mark 0.00 out of 2.00 Incorrect

This question refers to **the student task #1** in the "Trustworthy AI" assignment.

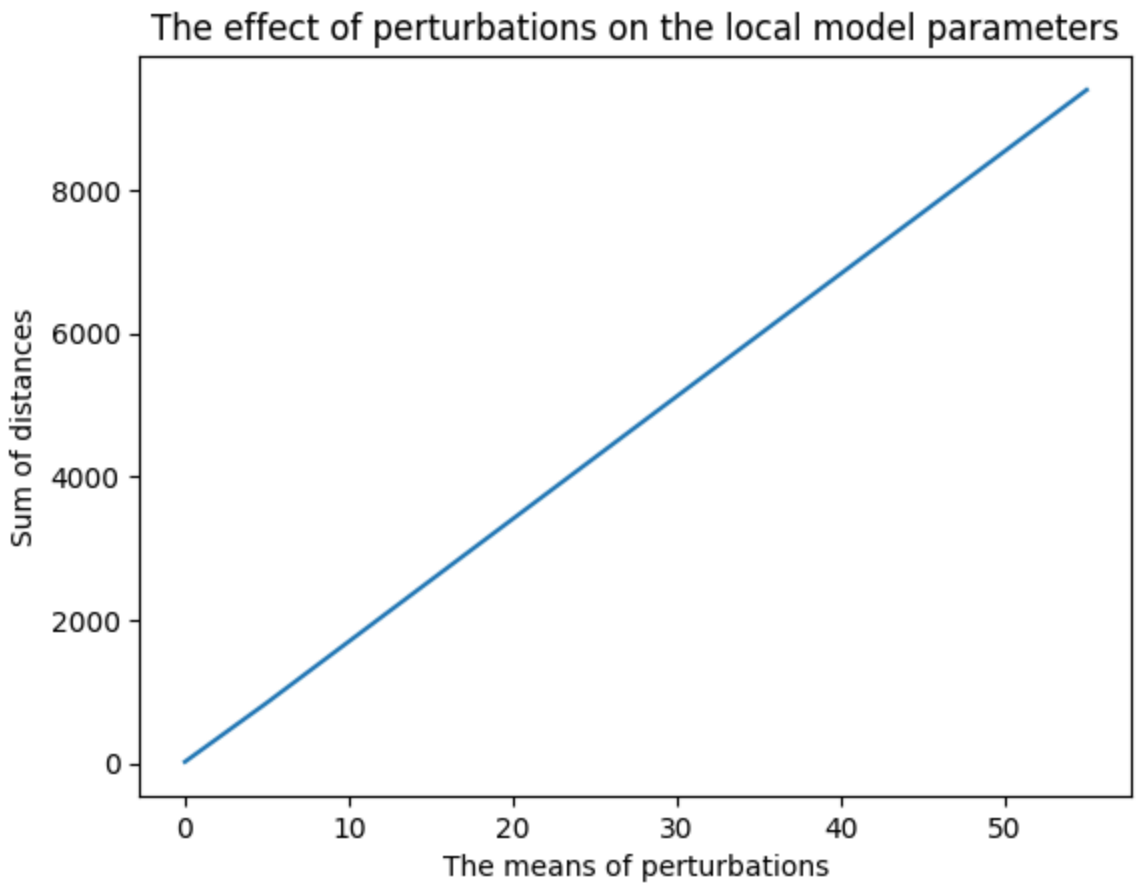
In the student task #1 the perturbations follow the normal distribution. What type of relationship is between the **means** of perturbations and the sum of the Euclidian distances between the weight vectors obtained for the original and perturbed data?

- a. Positive linear relationship.
- b. Negative linear relationship.
- c. There is no relationship between the specified variables.
- d. Non-linear relationship. ✖

Your answer is incorrect.

Please, join Slack channel (**link**) if you have any questions regarding the coding assignment.

The correct answer is:
Positive linear relationship.



Question 5

Flag question Mark 0.00 out of 3.00 Incorrect

This question refers to **the student task #1** in the "Trustworthy AI" assignment.

In the student task #1 the perturbations follow the normal distribution. Consider two cases:

1. the perturbations $\sim \mathcal{N}(1, 10)$, and
2. the perturbations $\sim \mathcal{N}(10, 1)$.

For what case does the the sum of the Euclidian distances between the weight vectors obtained for the original and perturbed data **is larger**?

- [ ] a. For both cases the sum Euclidian distances is the same.
- [x] b. 1 ✖
- [ ] c. 2

Your answer is incorrect.

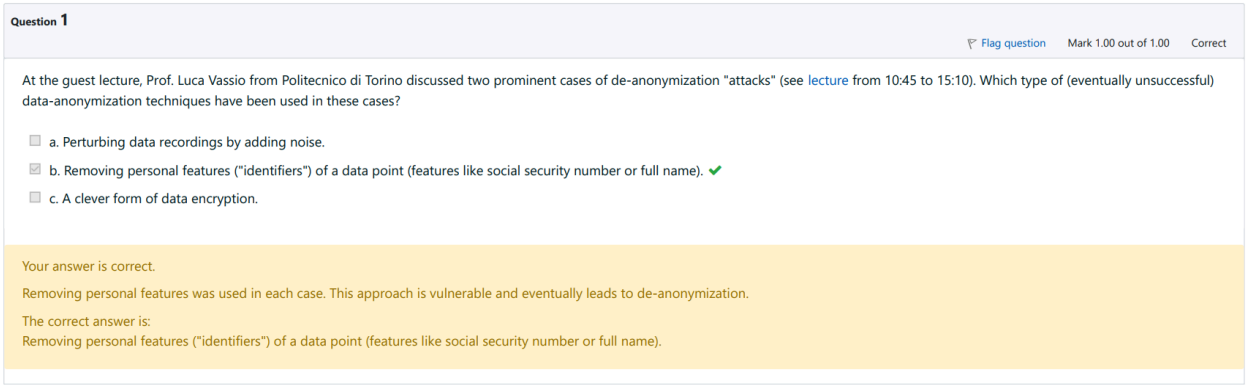
Please, join Slack channel (link) if you have any questions regarding the coding assignment.

The correct answer is:
2

```
Train on the perturbed data with mean = 1 and variance = 10...
Original weights shape: (207, 7)
Perturbed weights shape: (207, 7)
Train on the perturbed data with mean = 10 and variance = 1...
Original weights shape: (207, 7)
Perturbed weights shape: (207, 7)
Sum of distances for perturbations ~ N(1, 10): 4554.260546820076
Sum of distances for perturbations ~ N(10, 1): 3642.3096079211155
```

Which is wrong, as N(10, 1) should have larger perturbation. This question is unsure.

8. Privacy in AI



Question 1

Flag question Mark 1.00 out of 1.00 Correct

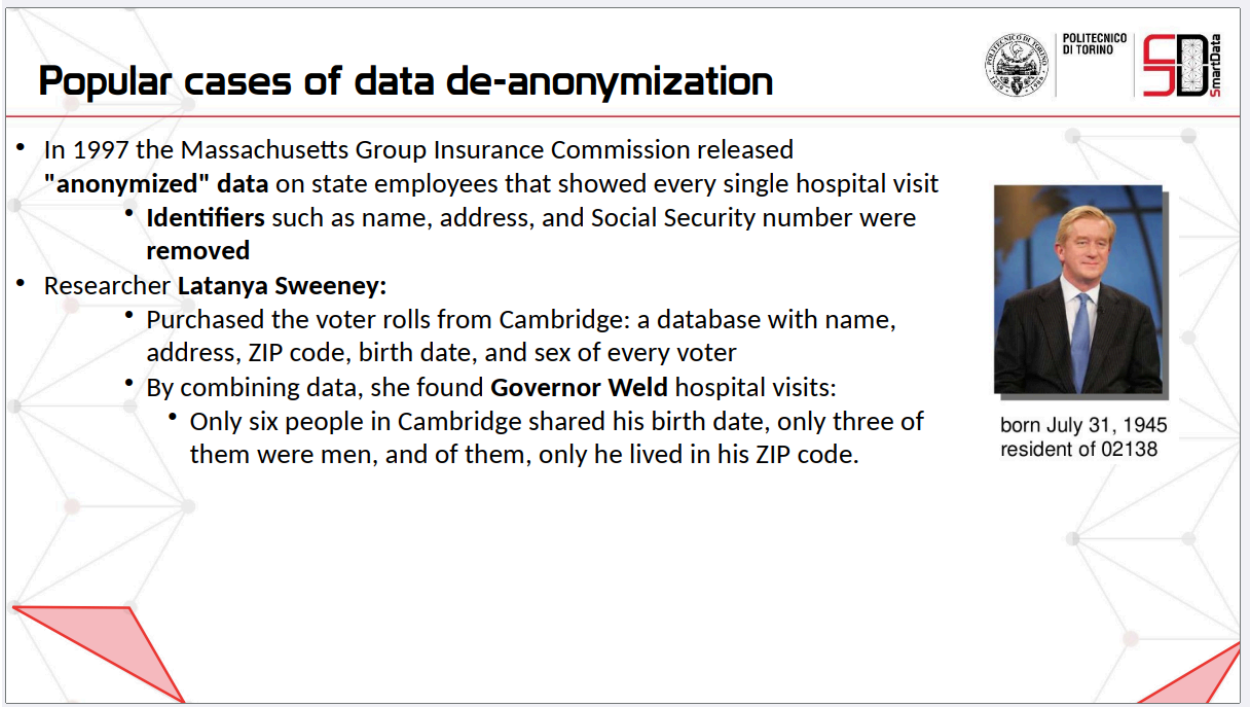
At the guest lecture, Prof. Luca Vassio from Politecnico di Torino discussed two prominent cases of de-anonymization "attacks" (see lecture from 10:45 to 15:10). Which type of (eventually unsuccessful) data-anonymization techniques have been used in these cases?

- a. Perturbing data recordings by adding noise.
- b. Removing personal features ("identifiers") of a data point (features like social security number or full name). ✔
- c. A clever form of data encryption.

Your answer is correct.

Removing personal features was used in each case. This approach is vulnerable and eventually leads to de-anonymization.

The correct answer is:
Removing personal features ("identifiers") of a data point (features like social security number or full name).



Question 2

Flag question Mark 1.00 out of 1.00 Correct

At the guest lecture, Prof. Luca Vassio from Politecnico di Torino discussed a prominent case of privacy leakage from public data (see lecture from 10:45 to 15:10). In this case, how was it possible to determine private information about an individual ?

- a. A bug in a database software.
- b. Using unencrypted data for communication.
- c. Using advanced ML models that predict the private data.
- d. Data aggregation from multiple sources. ✔

Your answer is correct.

Data aggregation eventually leads to de-anonymization by grouping the known attributes.

The correct answer is:
Data aggregation from multiple sources.

Popular cases of data de-anonymization

POLITECNICO DI TORINO

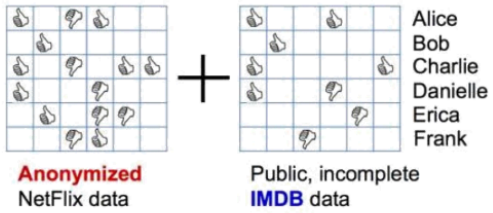
- Problem: can combine "private" ratings from Netflix with public reviews from IMDB to identify users in dataset
- May expose private info about members…

IMDb Earth's Biggest Movie Database™

Popular cases of data de-anonymization

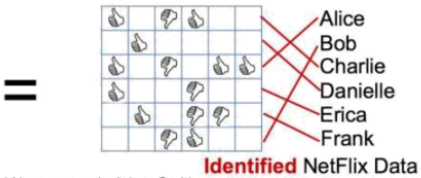
POLITECNICO DI TORINO

Use Public Reviews from IMDB.com



Anonymized NetFlix data

Public, incomplete **IMDB** data



Identified NetFlix Data

Credit: Arvind Narayanan via Adam Smith

Narayanan, Shmatikov, Robust De-anonymization of Large Datasets (How to Break Anonymity of the Netflix Prize Dataset), 2008

Question 3

Flag question Mark 2.00 out of 2.00 Correct

The amount of privacy-leakage of an algorithm is related to the level of its "non-invertibility". One popular approach to measure this non-invertibility is referred to as differential privacy (DP). What is the basic idea behind DP as a measure for privacy-leakage of an algorithm.

- a. DP is based on upper bounding the norm of the gradient of the algorithm, viewed as a map from training dataset to learnt model parameters.
- b. DP is based on the Lipshitz constant of the algorithm, viewed as a map from training dataset to learnt model parameters.
- c. DP measures the dis-similarity (or divergence) between probability distributions of the algorithm output, obtained for small changes in the input dataset. ✔
- d. DP is based on the (estimated) covariance matrix of the datapoints' features.

Your answer is correct.

See Section 9.2 in the Lecture Notes (found here).

The correct answer is:
DP measures the dis-similarity (or divergence) between probability distributions of the algorithm output, obtained for small changes in the input dataset.

Why is anonymization hard?

It's hard to guess what capabilities attackers will have, especially in the future

- Future datasets
- Future techniques
- Future computational power

Analogy with cryptography: cryptosystems today are designed based on what quantum computers might be able to do in 30 years

Why Differential Privacy (DP)?

- Strong, quantifiable, composable mathematical privacy **guarantee**
- **It is (by design) resilient** to known and unknown attack modes!
- DP enables many computations with personal data while preserving personal privacy

What to learn from data?

- **Differential privacy**: no harm in **participation**
  - Outcome of any analysis is essentially equally likely, **independent of whether any individual joins or not the dataset**
- **Usually, we want our observations to generalize to other data points → avoid overfitting**

Differential Privacy: Scenario

- **Worst-case scenario**: an attacker that knows everything, but not if the user is in the dataset
  - The attacker knows the value of every record, **even all the target user's one**
- The attacker can perform **queries** to the dataset
  - The answer of the dataset should not hint the presence of the user

Question 4

Flag question | Mark 2.00 out of 2.00 | Correct

Consider a dataset that is represented as a table, whose rows are individual data points and the columns represent the available features (or attributes) of a data point. When does such a dataset satisfy "k-anonymity" with some given value for k (e.g., k=5)?

- [x] a. For any given data point (= row in the table) there are at least K individuals that match the known features (= columns of the table) of the data point. ✔
- [ ] b. The dataset consists of at least K data points.
- [ ] c. The dataset includes at least K different features (columns) for each data point.
- [ ] d. The dataset consists of at most K data points.

Your answer is correct.

See the guest lecture by Prof. Luca Vassio from Politecnico di Torino (found here). The discussion about K-anonymity starts at 18:20.

The correct answer is: For any given data point (= row in the table) there are at least K individuals that match the known features (= columns of the table) of the data point.

Achieving k-Anonymity

- Goal of k-Anonymity
  - Each record is indistinguishable from at least k-1 other records
  - These k records form an equivalence class
- **Generalize, modify, or supress** quasi-identifier values so that no individual is uniquely identifiable from a group of k

Question 5

Flag question Mark 0.00 out of 1.00 Incorrect

The question refers to the notebook for the "Privacy-Protection in FL" assignment.

Sections 3.2 and 3.3 enforced differential privacy (DP) by pre- and by post-processing, respectively. Assuming the tested configurations (seeds, noise distribution, etc.), does pre-processing or post-processing ensure higher level of privacy protection (reflected by larger training and validation errors)?

- a. Post-processing yields higher level of privacy protection.
- b. Pre-processing yields higher level of privacy protection. ✖
- c. Both techniques ensures approximately the same level of privacy protection.

Your answer is incorrect.

Please, join Slack channel (link) to discuss the coding assignment.

The correct answer is:
Post-processing yields higher level of privacy protection.

Properties of Differential Privacy

POLITECNICO DI TORINO

SmartData

Composability

- Applying the sanitization several time yields a graceful degradation
- If A_1 satisfies ε_1-DP, and A_2 satisfies ε_2-DP, then outputting both A_1 and A_2 satisfies $(\varepsilon_1+\varepsilon_2)$-DP

Robustness to side information

- No need to specify **exactly** what the adversary knows
- Any **post-processing cannot improve** the attacker's knowledge

Question 6

Flag question Mark 2.00 out of 2.00 Correct

The question refers to the notebook with the "Privacy-Protection in FL" assignment.

Section 3.4.2 studies the usefulness of privacy preserving features by training a model for predicting a private attribute. High level of privacy protection should result in relatively large validation errors of the trained predictor.

Choose the interval that contains the normalized validation error obtained after training a model for predicting the private attribute using privacy-preserving features. The validation error is normalized by the (sample) variance of the private attribute values.

- a. [0, 0.5]
- b. [0.5, 1.5] ✔
- c. [1.5, 2.5]
- d. [2.5, 3.5]
- e. [3.5, 4.5]
- f. [4.5, 5.5]
- g. [5.5, 6.5]
- h. [6.5, 7.5]

Your answer is correct.

Please, join Slack channel (link) to discuss the coding assignment.

The correct answer is:
[0.5, 1.5]

```
Train/Val errors obtained when using privacy-preserving features Z
Training Error: 1.003611902314061
Validation Error: 0.9857399699602243

************************************************************

Train/Val errors obtained when using original features X
Training Error: 3.719283221546538e-28
Validation Error: 3.665935665838089e-28
```

Question 7

Flag question Mark 2.00 out of 2.00 Correct

The question refers to the notebook for the "Privacy-Protection in FL" assignment.

Section 3.4.3 evaluates the utility of privacy-preserving features in terms of how well they allow to predict the temperature of a weather recording (= label of data point). In particular, you had to compute the validation error of a trained linear model using the privacy-preserving features. How much larger is this validation error compared to the validation error obtained from a trained linear model using the original features?

- a. Approximately 5 times.
- b. Approximately 6 times.
- c. Approximately 3 times.
- d. Approximately 2 times. ✔
- e. Approximately 4 times.

Your answer is correct.

Please, join Slack channel (**link**) to discuss the coding assignment.

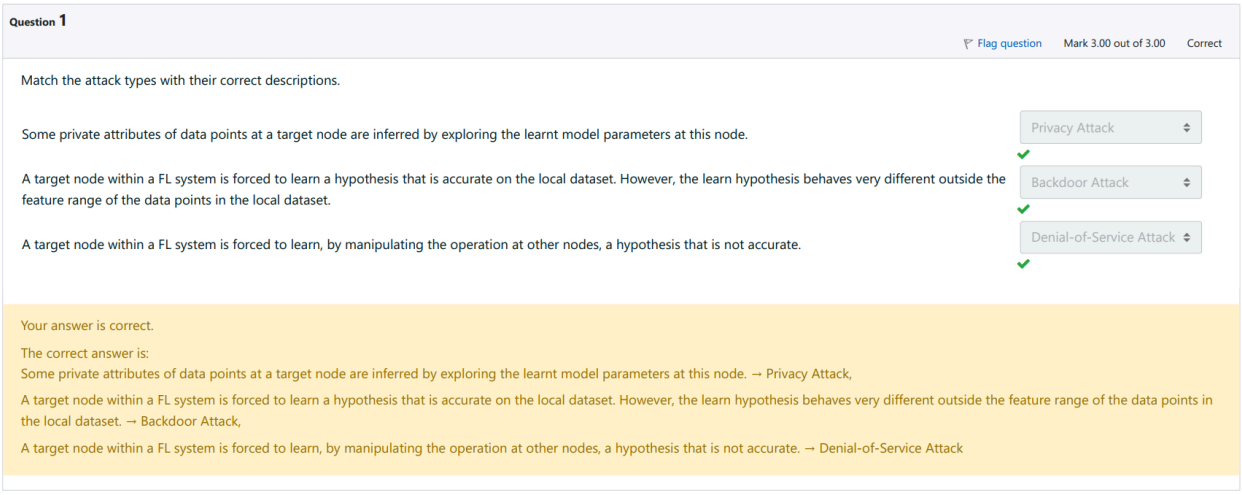
The correct answer is:
Approximately 2 times.

```
Train/Val errors obtained when using new features Z to predict tempature y
Training Error: 31.08297194420889
Validation Error: 30.04272420998213

************************************************************

Train/Val errors obtained when using original (privacy-leaking) features X to predict tempature y
Training Error: 16.735639571838544
Validation Error: 16.67841632785604
```

9. Data poisoning in FL



Question 1

Flag question Mark 3.00 out of 3.00 Correct

Match the attack types with their correct descriptions.

Some private attributes of data points at a target node are inferred by exploring the learnt model parameters at this node. — Privacy Attack

A target node within a FL system is forced to learn a hypothesis that is accurate on the local dataset. However, the learn hypothesis behaves very different outside the feature range of the data points in the local dataset. — Backdoor Attack

A target node within a FL system is forced to learn, by manipulating the operation at other nodes, a hypothesis that is not accurate. — Denial-of-Service Attack

Your answer is correct.

The correct answer is:
Some private attributes of data points at a target node are inferred by exploring the learnt model parameters at this node. → Privacy Attack,

A target node within a FL system is forced to learn a hypothesis that is accurate on the local dataset. However, the learn hypothesis behaves very different outside the feature range of the data points in the local dataset. → Backdoor Attack,

A target node within a FL system is forced to learn, by manipulating the operation at other nodes, a hypothesis that is not accurate. → Denial-of-Service Attack

- **Denial-of-Service Attacks.** The goal of a denial-of-service attack is to make the learnt hypothesis $\bar{h}^{(i)}$, at some target node i, useless in the sense of having unacceptable large prediction errors. We can detect a denial-of-service attack by continuously monitoring the performance of the learnt model parameters $\mathbf{w}^{(i)}$. Denial-of-service attacks on GTVMin-

- **Backdoor Attacks.** A backdoor attack tries to make a target node i learn a hypothesis $\tilde{h}^{(i)}$ that behaves well on the local dataset $\mathcal{D}^{(i)}$ but highly irregular for a certain range of feature values. The goal of the attacker is to exploit this irregular behaviour by preparing a data point

- **Privacy Attacks (see Lecture 9).** The goal of a privacy attack is to determine private attributes of the data points in the local dataset at some target node i. To this end, an attacker might try to enforce some other (vulnerable) node i' to learn a copy of the model parameters $\mathbf{w}^{(i)}$ at node i. For GTVMin-based methods, this could be achieved by using

Question 2

Flag question Mark 3.00 out of 3.00 Correct

Which of the following loss functions typically results in ERM being **least robust** against perturbations of data points in the training set.

- [x] a. The squared error loss. ✔
- [ ] b. The absolute error loss.
- [ ] c. The linear least squares function regularized by l2-norm.

Your answer is correct.

The correct answer is:
The squared error loss.

From a GTVMin-perspective, the effect of a data poisoning attack is that the original local loss functions $L_i(\cdot)$ in GTVMin (3.17) are replaced by perturbed local loss functions $\tilde{L}_i(\cdot)$. The extend of perturbation depends on the fraction of data points that are poisoned as well as on the loss function used to measure the prediction errors.

Different choices for the underlying loss function offer different levels of robustness against poisoning. For example, using the absolute error loss yields higher robustness against perturbations of the label values of few data points compared to the squared error loss. Another class of robust loss functions is obtained by including a penalty term (as in regularization).

Question 3

Flag question Mark 3.00 out of 3.00 Correct

*This question refers to **the student task #1** in the "Data Poisoning in FL" assignment.*

The implemented Denial-of-Service attack affects the validation error of the attacked node's model parameters via adding noise to (randomly selected) other nodes in the empirical graph.

Match the random seed with the corresponding minimum required number of poisoned nodes to increase the validation error of the attacked node's model parameters by 20%. Attack the node indexed 1 (in the code: *attacked_node = 1*).

P.S. The intervals with the number of poisoned nodes are given to allow for variations caused by using different Python environments. If - for a specific seed value - you find that at least 3 poisoned nodes are required, then the intervals [1, 3], [2, 5], or [3, 10] are all correct.

Seed = 10001	[21, 23]	✔
Seed = 101	[4, 6]	✔
Seed = 4	[8, 10]	✔
Seed = 1	[4, 6]	✔
Seed = 4740	[17, 19]	✔

Your answer is correct.

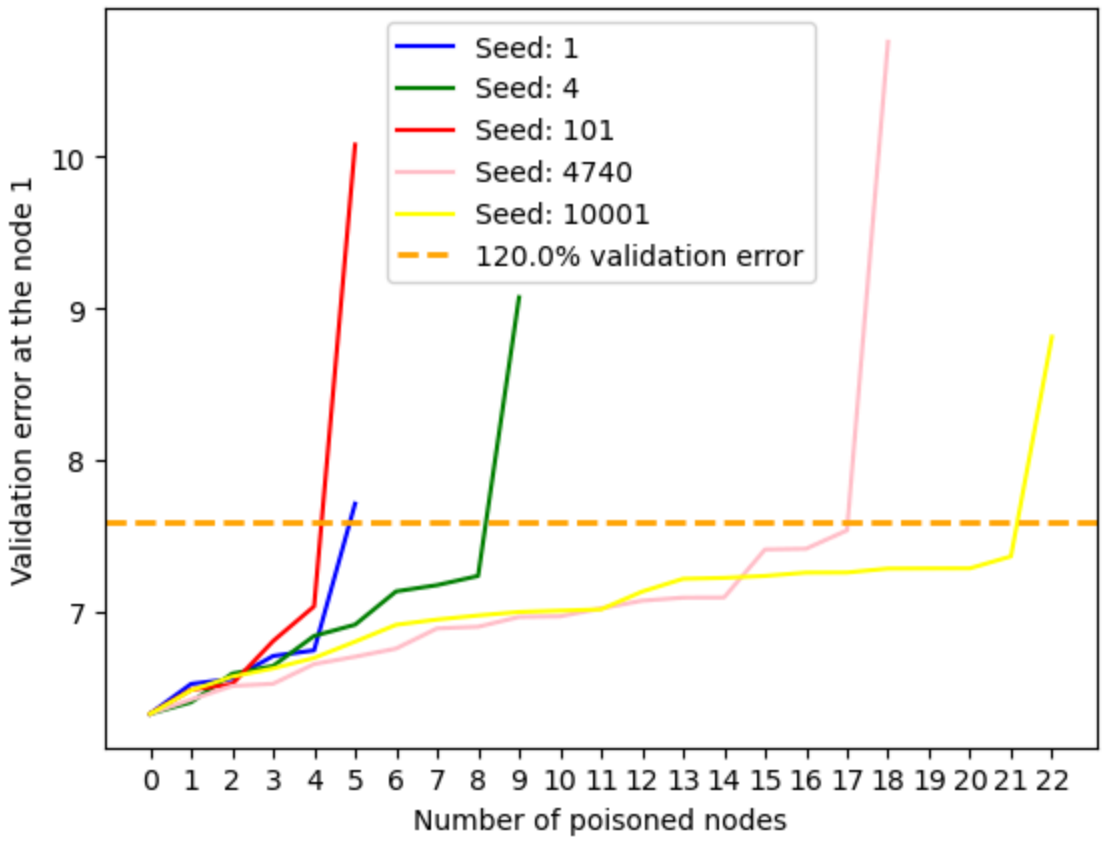
The correct answer is:
Seed = 10001 → [21, 23],

Seed = 101 → [4, 6],

Seed = 4 → [8, 10],

Seed = 1 → [4, 6],

Seed = 4740 → [17, 19]



Question 4

Flag question Mark 2.00 out of 2.00 Correct

*This question refers to **the student task #2** in the "Data Poisoning in FL" assignment.*

How many data points of the attacked node have been poisoned to plant the backdoor into the trained linear model? Attack the node indexed 1 (in the code: *attacked_node = 1*). The backdoor trigger is 4 (in the code: *trigger = 4*).

P.S. Consider all data points of the attacked node (training + validation).

- a. 0
- b. 1
- c. 2
- d. 3
- e. 4 ✔
- f. 5
- g. 6
- h. 7

Your answer is correct.

The correct answer is:
4

```
The training set contains 2 poisoned data points.
The validation set contains 2 poisoned data points.
```