2026 USA-North America AI Olympiad (USA-NA-AIO) Round 1 Problems

USA AI Olympiad (USAAIO)

January 30, 2026

Problem 1. (50 Points)

Part 1.1. (10 Points)

Which of the following tasks is an example of **unsupervised learning**?

A. Predicting house prices given historical data and labeled prices.

B. Classifying emails as spam or not spam.

C. Grouping customers based on purchasing behavior without predefined labels.

D. Predicting whether a tumor is malignant or benign.

E. Recognizing handwritten digits using labeled images.

Part 1.2. (10 Points)

Consider two regularized linear regression models trained on the same dataset: Model A uses l_1-regularization. Model B uses l_2-regularization. Both use the same regularization strength $\lambda > 0$.
Which statement is most likely true?

A. Models A and B have the same number of non-zero weights.

B. Model A has fewer non-zero weights than Model B.

C. Model B has fewer non-zero weights than Model A.

D. Neither model can produce zero weights.

E. There is no way to know which model has fewer non-zero weights.

Part 1.3. (10 Points)

Which of the following best describes the typical behavior of the error components attributed to bias and variance as you increase the complexity of a model (for example, by increasing the degree of a polynomial in a regression model),?

A. Both bias and variance increase.

B. Both bias and variance decrease.

C. Bias and variance remain constant regardless of complexity.

D. Bias decreases and variance increases.

E. Bias increases and variance decreases.

Part 1.4. (10 Points)

Under which circumstances is the $f1$ score generally preferred over classification accuracy?

A. When the dataset is very large.

B. When the classes are well balanced.

C. When the model is linear.

D. When the dataset is highly imbalanced.

E. When the number of features is large.

Part 1.5. (10 Points)

Which of the following statement is correct about **Uniform Manifold Approximation and Projection (UMAP)**?

A. UMAP preserves all pairwise distances exactly.

B. UMAP prioritizes preserving local neighborhood structure over global distances.

C. UMAP guarantees linear projections of the data.

D. UMAP minimizes reconstruction error like PCA.

E. UMAP forces all clusters to have equal size.

Problem 2. (15 Points)

Consider a sample data point $\mathbf{x} \in \mathbb{R}^d$. Let $\hat{\mathbf{e}} \in \mathbb{R}^d$ be a principal component of unit length.

Part 2.1. (5 Points)

Let $d = 3$. Consider the vector

$$\mathbf{v} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}.$$

Let $\hat{\mathbf{e}}$ point in the same direction as $\mathbf{v}$. We can write $\hat{\mathbf{e}}$ in the following form:

$$\hat{\mathbf{e}} = \frac{1}{\sqrt{a}} \begin{bmatrix} b \\ c \\ d \end{bmatrix},$$

where a is a positive integer, b, c, d are integers and $\gcd(a, b^2, c^2, d^2) = 1$. What is the value of $a + b + c + d$?

A. 0.

B. 12.

C. 14.

D. 16.

E. 20.

Part 2.2. (5 Points)

Let

$$\mathbf{x} = \begin{bmatrix} -1 \\ -4 \\ 6 \end{bmatrix}.$$

Compute the projection of $\mathbf{x}$ onto $\hat{\mathbf{e}}$ (the inner product of these two vectors). Your answer can be written as

$$\frac{b}{\sqrt{a}},$$

where a is a positive integer, b is an integer and $\gcd(a, b^2) = 1$. What is the value of $a + b$?

A. 14.

B. -6.

C. -11.

D. 34.

E. 39.

Part 2.3. (5 Points)

Let $\mathbf{r}$ be the residual after $\mathbf{x}$ is projected onto $\hat{\mathbf{e}}$. We can write $\mathbf{r}$ in the form

$$\mathbf{r} = \frac{1}{a} \begin{bmatrix} b \\ c \\ d \end{bmatrix},$$

where b, c, d are integers, a is a positive integer and $\gcd(a, b, c, d) = 1$. What is the value of $a + b + c + d$?

A. 14.

B. -22.

C. -36.

D. 50.

E. 64.

Problem 3. (10 Points)

Consider a matrix $\mathbf{A} \in \mathbb{R}^{3\times 4}$. For any vector

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

we have

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} x_2 - x_3 \\ x_3 \\ x_0 + 2x_1 \end{bmatrix}.$$

Part 3.1. (5 Points)

Compute $\mathbf{A}$. Reasoning is not required.

Part 3.2. (5 Points)

We can write $\mathbf{A}$ in the following decomposition form:

$$\mathbf{A} = \sum_{i=0}^{I-1} \mathbf{u}^{(i)} \mathbf{v}^{(i),\top},$$

where $\mathbf{u}^{(i)}$ and $\mathbf{v}^{(i)}$ are column vectors.

Compute

1. I
2. $\mathbf{u}^{(i)}$ and $\mathbf{v}^{(i)}$ for all $i \in \{0, 1, \cdots, I-1\}$.

Reasoning is required.

Problem 4. (20 points)

Part 4.1. (5 points)

This is a math task. Compute the following derivative:

$$\frac{d \tanh x}{dx},$$

where

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Reasoning is required.

Part 4.2. (15 points)

This is a coding task.
Run the following starter code:

```
import numpy as np
```

Write a function called `my_fun_derivative`:

1. Inputs:
   (a) NumPy array `X` with shape `(N,d)`.
   (b) NumPy array `y` with shape `(N,)`.
   (c) NumPy array `theta` with shape `(d,)`.
2. Output:

$$\nabla_\theta \frac{1}{N} \sum_{n=0}^{N-1} \left(y[n] - \theta^\top \mathbf{X}[n] \right)^2.$$

So the output is a NumPy array with shape `(d,)`.

In this task, you are **not** allowed to

1. import any additional libraries/modules/methods;
2. use autograd in PyTorch;
3. use any method in `np.linalg`, including `@` and `.T`;
4. use any loop.

You will receive 0 points if you violate any of these rules.

Problem 5. (90 Points)

In this problem, we study word/token embeddings. Each token is embedded as a vector of length 100.
Run the following starter code:

```
!pip install gensim

import gensim.downloader as api
from gensim.utils import simple_preprocess
import requests

import numpy as np
import matplotlib.pyplot as plt

glove = api.load("glove-wiki-gigaword-100")

url_part1 = "https://huggingface.co/datasets/usaaio-official/"
url_part2 = "2026_USAAIO_Round1_public/raw/main/"
url_part3 = "2026_USAAIO_Round1_NLP.txt"
url = url_part1 + url_part2 + url_part3

text = requests.get(url).text

tokens = simple_preprocess(text)
tokens = [token for token in tokens if token in glove]
print(len(tokens))
tokens = list(set(tokens))
print(len(tokens))
vectors = [glove[word] for word in tokens]
```

Do not import any additional libraries, modules, or methods beyond those included

in the given starter code.

Part 5.1. (5 Points)

In the starter code, we run the following line twice.

```
print(len(tokens))
```

Why do we get two different results?

Part 5.2. (5 Points)

Explain what the following code does:

```
tokens = list(set(tokens))
```

Part 5.3. (5 Points)

Write code to convert `vectors` into a NumPy array. Name it `vectors_arr`. So

$$\text{vectors\_arr} = \begin{bmatrix} \mathbf{v}^{(0),\top} \\ \mathbf{v}^{(1),\top} \\ \vdots \\ \mathbf{v}^{(N-1),\top} \end{bmatrix},$$

where N is the total number of tokens and $\mathbf{v}^{(n)}$ is the embedding of the nth token.

Part 5.4. (5 Points)

Define

$$\hat{\mathbf{v}}^{(n)} = \frac{\mathbf{v}^{(n)}}{||\mathbf{v}^{(n)}||_2}.$$

Thus, $\hat{\mathbf{v}}^{(n)}$ has unit length. Let

$$\mathbf{W} = \begin{bmatrix} \hat{\mathbf{v}}^{(0),\top} \\ \hat{\mathbf{v}}^{(1),\top} \\ \vdots \\ \hat{\mathbf{v}}^{(N-1),\top} \end{bmatrix}.$$

Write code to compute `W` from `vectors_arr`.
Do not use `np.linalg` or any loop.

Part 5.5. (5 Points)

Define the similarity between tokens i and j as

$$\text{sim}_{ij} = \hat{\mathbf{v}}^{(i),\top} \hat{\mathbf{v}}^{(j)}.$$

What is the range of sim_{ij}?
Reasoning is not required.

Part 5.6. (5 Points)

Define a similarity matrix

$$\mathbf{S} = \begin{bmatrix} \text{sim}_{00} & \text{sim}_{01} & \cdots & \text{sim}_{0(N-1)} \\ \text{sim}_{10} & \text{sim}_{11} & \cdots & \text{sim}_{1(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \text{sim}_{(N-1)0} & \text{sim}_{(N-1)1} & \cdots & \text{sim}_{(N-1)(N-1)} \end{bmatrix} \in \mathbb{R}^{N \times N}.$$

Express $\mathbf{S}$ in terms of $\mathbf{W}$.
Reasoning is not required. Coding is not needed.

Part 5.7. (5 Points)

Write one line of code to compute `S` using `W`.
Do not use `np.linalg`, `@` or any loop.

Part 5.8. (5 Points)

1. (Coding task) Print out the diagonal values of `S`.
2. (Non-coding task) Interpret your result.

Part 5.9. (10 Points)

Create a dictionary called `sim_dict`. Each token serves as a key.
For each given token (the key), its value should be the token (excluding itself) that has the highest similarity score with that key.
For example, if your solution is correct, you should obtain the following key-value pair:

```
"bright": "lights"
```

Part 5.10. (5 Points)

Is `S` invertible?
Mathematical reasoning is required. Coding is not allowed.

Part 5.11. (5 Points)

Let the singular value decomposition (SVD) of $\mathbf{W}$ be

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top.$$

Write code to compute

1. $\mathbf{U}$,
2. the diagonal of $\mathbf{\Sigma}$, sorted in decreasing order and denoted by σ,
3. $\mathbf{V}$.

Hint: You may use `np.linalg.svd`.

Part 5.12. (5 Points)

Denote by

$$\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$$

a spectral decomposition of $\mathbf{S}$, such that

1. $\mathbf{Q} \in \mathbb{R}^{N \times N}$ is an orthonormal matrix.
2. $\mathbf{\Lambda} \in \mathbb{R}^{N \times N}$ is a diagonal matrix whose diagonal entries are sorted in decreasing order.

Write code to compute $\mathbf{Q}$ and the diagonal of $\mathbf{\Lambda}$ (denoted by λ) using $\mathbf{U}$, σ, and $\mathbf{V}$ obtained in Part 5.11.

Part 5.13. (10 Points)

The Frobenius norm of the matrix $\mathbf{S}$ is defined as

$$||\mathbf{S}||_F = \sqrt{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} S_{ij}^2}.$$

Mathematically express $||\mathbf{S}||_F$ in terms of the SVD of $\mathbf{W}$.
Reasoning is required.

Part 5.14. (10 Points)

Consider a large N ($N > 100$). Recall that $\mathbf{S} \in \mathbb{R}^{N \times N}$ can be expressed in terms of $\mathbf{W} \in \mathbb{R}^{N \times 100}$.
Thus, the amount of data needed to store the relationship between each pair of tokens is not N^2, but $100N$.
However, this is still too much. Our goal is to approximate $\mathbf{S}$ using only rN numbers, where $r << 100$.

Denote by $\mathbf{W}_r \in \mathbb{R}^{N \times r}$ a matrix constructed from $\mathbf{W}$. Denote by $\mathbf{S}_r \in \mathbb{R}^{N \times N}$ a matrix constructed from $\mathbf{W}_r$.
To measure the error between $\mathbf{S}_r$ and $\mathbf{S}$, we use the following relative squared Frobenius metric:

$$\frac{||\mathbf{S} - \mathbf{S}_r||_F^2}{||\mathbf{S}||_F^2}.$$

For each given r, mathematically construct $\mathbf{W}_r$ and $\mathbf{S}_r$, such that the above metric is minimized.
Reasoning is not required.

Part 5.15. (5 points)

This is a coding task based on your findings in Part 5.14.
Generate a plot. The x-axis is $r \in \{1, 2, \cdots, 100\}$. The y-axis is the relative squared Frobenius metric.

Problem 6. (10 points)

Figure 1 is an image. We send it to a pretrained convolutional neural network.

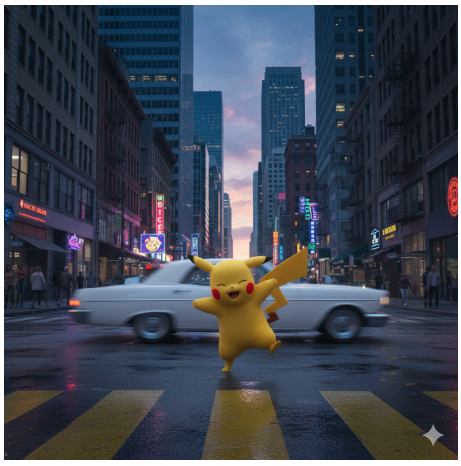

Figure 1: Original image

We obtain five feature maps that are outputs from five different layers (indexed from 0 to 4), as shown in Figure 2.

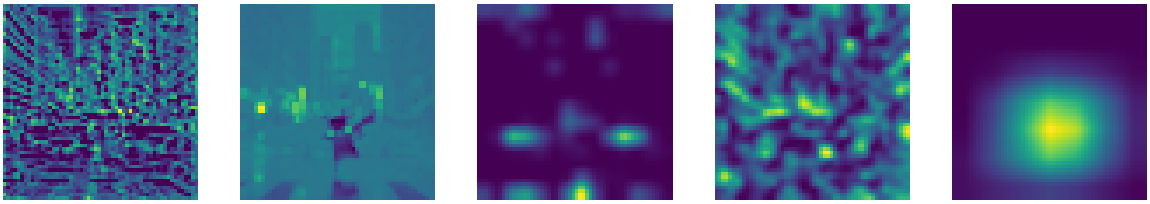

Figure 2: Feature maps from different layers

However, these feature maps are not given in order. For instance, feature map (1) is not necessarily the output from layer 0.

For each feature map, determine which layer it outputs from.
Reasoning is required.

Problem 7. (30 points)

Run the following starter code:

```
import torch
import torch.nn as nn
```

Part 7.1. (5 points)

Build a threshold module that subclasses `torch.nn.Module` and is named `My_Threshold`.
The threshold activation function is defined as

$$\Theta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}.$$

Part 7.2. (5 points)

Consider the following transformation

$$y = \sum_{c=0}^{C-1} w_c x_c.$$

Suppose $x_0, \cdots, x_{C-1}$ and $w_0, \cdots, w_{C-1}$ are all independent random variables. For $c \in \{0, \cdots, C-1\}$, $\mathrm{E}[w_c] = \mathrm{E}[x_c] = 0$, $\mathrm{Var}[w_c] = \sigma_w^2$ and $\mathrm{Var}[x_c] = \sigma^2$.
We want y to have the same variance as each input x_c. That is, we want $\mathrm{Var}[y] = \sigma^2$.
Compute σ_w that achieves this.
Reasoning is required.

Part 7.3. (5 points)

Build a linear module that subclasses `torch.nn.Module` and is named `My_Linear`.

1. The input argument `bias` is `True` by default. But it can be set to `False`. If `bias` is `True`, its initial value is 0.

2. The initial value of each entry in `weight` is drawn from a normal distribution with mean 0 and standard deviation σ_w that you find in Part 7.2.

Part 7.4. (15 points)

Use PyTorch to build a multi-layer perceptron (MLP) model that performs the following inference task.
On a 2-dim plane, consider a triangle with vertices $(0, 0)$, $(3, 1)$, $(1, 5)$. Each input sample to your model is a point in the 2-dim plane. The output is 1 if the point lies inside the triangle and 0 otherwise.

1. Define your neural network module as a Python class named `My_Triangle` that inherits from `nn.Module`, and then create an instance of that class named `my_triangle` to serve as your model.

2. Your module architecture may consist only of the threshold module you built in Part 7.1 and the linear module you built in Part 7.3. Any other module is prohibited.

3. In the forward method in `My_Module`, the input is a tensor of shape `(B,2)`, where `B` is the batch size. The output is a binary tensor (0 or 1) of shape `(B,)`.

4. In `my_triangle`, for each linear module, you must manually determine the values of `weight` and `bias`. So that the model is ready for inference.

   Note: This problem is inference only. There is no training.

5. In `my_triangle`, the attribute `requires_grad` for all learnable parameters must be set to `False`.

Problem 8. (25 points)

Run the following starter code:

```
import torch
import torch.nn as nn
from torchvision.models import resnet50
import torchsummary

model = resnet50(pretrained=True)
```

Part 8.1. (5 points)

What is the total number of learnable parameters in `model`?
Hint: For an object of type `torch.nn.parameter.Parameter`, you can use the method `numel()` to count the number of learnable parameters it contains.

Part 8.2. (5 points)

Suppose the input has shape `(B,3,224,224)`. What is the shape of the output from `model.layer2`?

Part 8.3. (5 points)

What is the total number of learnable parameters in all convolutional modules in `model.layer3[2]`?
To solve this problem, you are not allowed to use the method `numel` for any object of type `torch.nn.parameter.Parameter`.
Reasoning is required.

Part 8.4. (10 points)

We want to use a **truncated** pretrained resnet50 model (up to the output from `model.layer3[4]`) as a backbone for a classification task with 5 classes. Build such a model.
In your solution, all parameters in the backbone must be frozen (that is, fixed and not learnable).

Problem 9. (50 points)

In this problem, we study a tabular dataset for a binary classification task. All features are numeric.
You can access the **training dataset** by running the following starter code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

url_part1 = "https://huggingface.co/datasets/usaaio-official/"
url_part2 = "2026_USAAIO_Round1_public/raw/main/"
url_part3 = "2026_USAAIO_Round1_breast_cancer_train.csv"
url = url_part1 + url_part2 + url_part3

df = pd.read_csv(url)
X = df.drop("target", axis=1)
y = df["target"]
```

In this training dataset, `X` contains 30 input features, and `y` contains the binary target labels.
We also have a **hidden** test dataset that you **cannot access** during the competition. You must submit a single **Jupyter notebook (.ipynb)** containing your complete solution, including (but not limited to):

1. Data preprocessing (if needed)
2. Model construction
3. Model training
4. Inference logic

Inference requirements

For inference, you must define a function with the following signature:

```
def my_prediction(X_test):
    ###INSERT YOUR CODE HERE###
    return y_pred
```

In this function:

1. `X_test` is a pandas DataFrame containing all input features from the hidden test set.
2. `y_pred` must be a pandas Series containing your predicted labels.

After the competition, we will **execute all code** in your submitted notebook from top to bottom. During evaluation, we will load the hidden test features as `X_test` and call your function:

```
my_prediction(X_test)
```

Your predictions `y_pred` will be evaluated using the **macro-averaged f1 score (f1-macro)**.

Model constraints

1. In your solution, you must use **k-Nearest Neighbors (kNN)** for classification.
2. However, this does not mean you must directly apply kNN to the raw training and test data. You may use any data preprocessing, feature engineering, or pipeline you find appropriate.
3. Beyond kNN, you are not allowed to use any other supervised learning models.

4. You may import any module or method from scikit-learn, numpy, pandas, and matplotlib.

   One restriction is that in scikit-learn, among all supervised learning models, you can only import kNN.

5. Do not use deep neural network to solve this problem.

Notebook requirements
In your submitted notebook, please ensure the following:

1. Your code includes sufficient comments to make it readable.
2. At the end of the notebook, include a **text cell** summarizing:
   - Your overall approach.
   - The intuition behind your design choices.
   - Any alternative approaches or models you considered but did not pursue, or you tried but did not submit.

Your report does not need to be long, but it should be clear, concise, and well-reasoned.

Grading criteria
Your submission will be evaluated based on:

1. Whether the entire notebook runs successfully from start to finish.
2. Performance on the hidden test dataset.
3. The quality of your reasoning and problem-solving approach.

We **do not evaluate** code style or quality.