1A

Battery life is long lasting but not infinite; it should be a valid performance component. We don't want the robot to run out of battery mid-operation.

Battery being finite is something that the robot should consider. However, there is nowhere in the description that states that the robot should *maximize* battery (maximize here is the important word). Moreover, the fact that the robot is equipped with *very* long-lasting battery life should further make optimizing for maximizing battery life make very little sense. If the option states that the performance measure is reducing the number of instances of empty battery mid-operation, then it makes sense.

To give an analogy, suppose that your laptop battery lasts for 10 years; does it make sense for you to try to maximize the battery life? Probably not. However, it makes sense to reduce instances of the empty battery during critical work, maybe by charging the battery every 9 years. Note that optimizing this doesn't imply optimizing for maximum battery life (e.g., by using it only to do light browsing instead of running ML models).

Safe operation is interpreted as safety towards humans, not safety of medicine.

"A hospital employs autonomous robots to <u>deliver medications to patient</u> rooms. The robots must navigate corridors, avoid people or equipment in their path, and prioritize urgent deliveries for critical patients while maintaining safe operation."

From this description, we can see that the main goal of the robot is to deliver medications to patients—intact, well-preserved medicines. There is no point in delivering destroyed medicines. Limiting the interpretation of safe operation only to safety towards humans seems illogical, especially since the human safety factor has already been somewhat considered in the "avoid people ... in their path".

Regarding query: "I selected D, E, F, G, and H based on the following interpretations: ..."

The explanation given in the write-up should have already answered this. Please have a look at it.

It was mentioned that the robot would be moving in the corridors to deliver the medications. In view of this, can the environment involve just a grid of corridors?

The keyword here is “only” and “fixed”. It’s not true that the environment *only* consists of predefined grids with *fixed* obstacles.

1B

Goal-based/model-based reflex agents are more appropriate because their main goal is delivering medicines for critical patients, while safe operation is an additional goal. The constraints described are also suitable for rule-based planning.

Utility-based agent has no issue encoding constraints by incorporating them into its utility function. As you’ve seen in the mini-project, humans can encode sophisticated constraints in the utility function. It can also be learned using machine learning.

The point here is that there are multiple objectives that the agent needs to balance: time and safety.

It’s true that goal/model-based agent can, to some extent, do the job of delivering medicine safely and on time. However, there are limitations to handling the constraints as hard constraints using rules. One of the limitations is that it can’t handle tradeoffs. Sometimes, we want to deliver the medicines while operating in a relatively “unsafe” manner.

For example, let’s say there is a patient that requires a medicine in 10 minutes; otherwise, the patient will die.

- Goal/model-based (reflex) agent will treat the problem as business-as-usual, carefully navigating the corridors, stopping and waiting when the corridors are crowded, following the rules and programming that dictate the constraints.
- Utility-based agent will realize the utility of urgently delivering the medicine while the safe operation, although still important, is deemphasized. The agent can behave more “recklessly” but get to the patient faster and save lives.

There is an argument that says that model-based reflex agents may possess sophisticated reflexes based on their current perceptions and internal models. However, the term “reflex” agent is used because this type of agent is limited to simple condition-action mapping, hence the name “reflex.”

1C

Why the environment is not fully-observable?

The reason is that the agent lacks access to all the necessary information to make an optimal decision. Even if the robot is equipped with the most advanced 3D camera and lidar, or if its camera and lidar are positioned in the most optimal position, there are still certain pieces of information that the robot will not have access to in order to make an optimal decision. To make an optimal decision, the robot needs to be aware of every single object (big or small) in the area where the decision is being made (i.e., a hospital) at all times: their states, positions, etc. This is because any object could potentially affect the optimal decision. For example, if you pass a corridor with a person holding a laptop with a ruptured battery that will explode in 10 seconds, you would be at risk of exploding with them. You wouldn't know this (the laptop's battery state) with your camera or lidar. Therefore, the environment is not fully observable.

For this question, it's actually quite straightforward to draw a relation with self-driving cars example that we have in class because it's very similar problem. In self-driving cars, you already know that it's partially observable.

2A

Output is 0 and 1, SVM output is -1,+1:

The mismatching output domain is not an issue. It is easy to associate 0 <-> -1 and 1 <-> +1.

Why not logistic regression over SVM?

Note the sentences in the problem statement "Deployment of your machine learning method will have many imperfections. " and "Here, **best** is defined as being the most appropriate for the specific properties of the new data.". These sentences and other sentences in the problem statement speak against arguments for logistic regression:

- loss function (deployment does not use a loss function like BCE),
- zero-offset SVM (SVM offset was discussed conceptually in the tutorial),
- interpretability (We want to deploy, i.e., classify correctly, not interpret),
- label noise (there is no label noise in training, f is the ground truth).

The maximum margin property is what makes the SVM robust to noise in the features and should be the best choice among the given options and from what we know from the stated problem.

2B

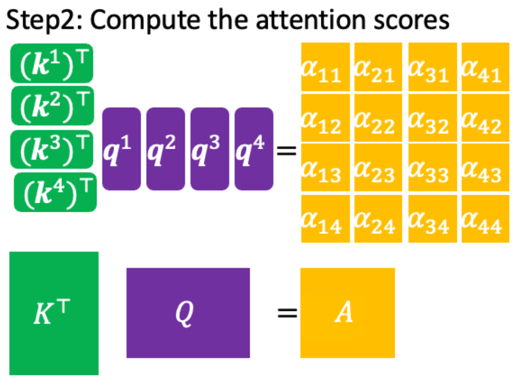
REVISED ANSWER: **B** only or **D** only

Can we consider the output of self-attention layer as hidden state?

Since we didn't explicitly specify what is the hidden state in this question, we accept B.

K^T Q can be considered a weighted sum of query vectors. I interpreted this question to mean what steps are taken in the process of obtaining the output of the self attention layer.

$K^T Q$ is not the weighted sum of query vectors. It is the multiplication of each query vector with each key vector. For example, $(k^1)^T q^1$, which generates α_{11}, is not the weighted sum of query vectors.


Step2: Compute the attention scores
$(k^1)^T$
$(k^2)^T$
$(k^3)^T$
$(k^4)^T$
q^1 q^2 q^3 q^4
=
α_{11} α_{21} α_{31} α_{41}
α_{12} α_{22} α_{32} α_{42}
α_{13} α_{23} α_{33} α_{43}
α_{14} α_{24} α_{34} α_{44}
K^T
Q
=
A

3A

REVISED ANSWER: *** can denote inner product between vectors or standard multiplication between matrices.**

I use an asterisk (*) to denote the dot product (inner product).

Since we didn’t explicitly specify the type of product we’re using for *, we’ll allow this.

Are these correct?

- **[log(u_1), log(u_2)]^T[log(v_1), log(v_2)]**
    - This is incorrect. The matrix multiplication is between a column vector (2x1) and a row vector (1x2), resulting in a 2x2 matrix instead of a scalar.
- **(u_j)^T * log(v_j)**
    - This is clearly incorrect. The u components are not passed through the log function. The notation is unclear (see below). Additionally, it does not produce a scalar.
- **[log(u_j]^T*[log(v_j)]**
    - The question clearly stated that u and v are of dimension 2. Moreover, the current notation is unclear. If you want to make it general, you should’ve written [log(u_1), …, log(u_N)] or with expansion [log(u_j)]_{j=1}^{N}. Additionally, it does not produce a scalar.
- **[log(u1),log(u2)]^T*[log(v1),log(v2)]**
    - This does not produce a scalar.
- **k(u, v) = …**
    - If your expression is valid, then it’s okay. Since no full equation is provided then we can’t tell.
- **Use u_x1 and u_x2 to represent features of the vectors**
    - This doesn’t make any sense. The description doesn’t mention anywhere that subscript x1 refers to a component of u or v.
- **log(u_i) * log(v_i) + log(u_j) * log(v_j)**
    - There is no reason to write it this way. This probably indicates a lack of understanding.
- **k(u,v) = [Ilog(u_1), log(u 2)|T/^T * [log(v_1), log(v_2)|^T = |log(u_1), log(u_2)] * [log(v_ 1), log(v_2)]^T**
    - Oh, we must have missed this. We will mark this as correct.

Please note that this question is worth only one mark. You must either get it correct or it will be marked incorrect.

4A

L1 regularization better meets clinicians' need for model transparency and trust compared to L2 regularization by eliminating irrelevant features and improving interpretability, so option c should also be correct.

The question asks to explain why Experiment A achieved higher validation accuracy than Experiment B. While it's true that L1 eliminates (zeroes out) irrelevant features where L2 may not, it is not a valid ground to summarily dismiss L2 and conclude that L2 is not suited for medical use.

L1 simplifies the model and increases bias as part of the bias-variance tradeoff, which improved generalization in this case.

First, we would like to clarify that making stronger assumptions and constraints on a model doesn't always lead to an increase in bias. Bias is defined as the expected difference between model predictions and the true values. While we can increase the constraints of a model, it may still align well with the true function, resulting in no increase in bias.

While L1 regularization can increase model bias in theory, the key here is that **validation accuracy improved** (58% → 76%), which directly reflects a **reduction in generalization error**. Generalization error = Bias + Variance.

The initial model (94% training, 58% validation) had **high variance** (overfitting). L1 regularization reduced variance by eliminating irrelevant/noisy features (B and D). Even if L1 may slightly increased bias, the **net effect** was **a large reduction in variance**, leading to lower generalization error and higher validation accuracy.

Thus, **the dominant factor explaining Experiment A's superior performance is variance reduction, not bias**. The question asks for factors that explain why A outperformed B, not whether L1 inherently increases bias. While A is technically true in isolation, it does not directly account for the observed improvement in validation accuracy.

4B

SVMs are preferable over logistic regression for handling noisy clinical data.

The question specifically seeks an experiment that aids in identifying lifestyle factors for targeted patient intervention. In this context, noise in the data is not the primary determining factor. While Support Vector Machines (SVMs) can handle noise, they lack inherent interpretability, particularly when using non-linear kernels. Logistic regression, on the other hand, explicitly quantifies the importance of features through coefficients, which is crucial for clinical diagnostics and intervention as per the requirement.

Removing features using L1 could lead to the overlooking of certain factors. Therefore, all features should be retained.

The key is that L1 regularization doesn't "remove" lifestyle factors arbitrarily—it selects the strongest predictors (including lifestyle factors if they are statistically significant). For example:

- If exercise/diet are truly impactful, L1 will retain them as non-zero weights.
- If 28 features are zeroed out, they likely contribute minimally to prediction (e.g., redundant/noisy measurements).

Note that L1 will retain interdependence factors and joint factors as long as they are significant.

Clinicians require a shortlist of actionable factors (e.g., "increase exercise" vs. "all 40 factors could be relevant") to determine appropriate interventions for their patients. L1 achieves this by isolating the most critical predictors, even if some weaker lifestyle factors are excluded. Retaining all 40 features (L2) would obscure priorities.

The question's wording led me to believe that all lifestyle factors should be included for better prediction, it would be clearer if the question specified selecting only the most critical lifestyle factors.

The question specifically seeks an experiment that aids in identifying lifestyle factors for targeted patient intervention. It should be immediately clear that what is needed is actionable factors for interventions, not an exhaustive list. Here, we are not interested in making predictions, but rather in finding out what interventions (i.e., actions) could be done to the patient. Listing out all features would be useless.

4C

REVISED ANSWER: **A, B, D** or **B, D**

Option A: The word "too small" is not well-defined and can be interpreted as excessive. Furthermore, there is no clear boundary as to when a model can be said to fail to capture patterns.

It is true that the sentence could be interpreted in different ways. Therefore, we decided to allow you to choose whether to include or exclude Option A.

Option B: We are taught that regularization should make a model less overfit and more underfit. However, I didn't really perceive it as prioritizing one aspect over another.

Yes, regularization helps with overfitting. However, excessive regularization can lead to the model being overly constrained (i.e., trying to minimize the weights) and failing to fit the data/trends, resulting in underfitting.

Option D: L1 regularization revealed that a substantial portion of the features are irrelevant, considering all 40 should be considered overfitting.

Overfitting is not determined solely by the number of features in a model—a model can have many features and still fit the data appropriately, depending on the complexity and the relationship between features and target. Underfitting, on the other hand, occurs when a model fails to capture underlying patterns in the data.

In this case, the model previously achieved a 71% validation accuracy. After increasing the regularization penalty, the validation accuracy dropped to 65%. Since the only change was the increase in regularization, which constrains the model further, the drop in performance suggests that the model is now underfitting—it has become too simple to capture the patterns in the data, leading to reduced accuracy.

Also see explanation for option B.

5B

Why can't x(1) and x(4) be the support vectors instead of including x(2)?

This does <u>not</u> lead to an SVM with <u>maximum</u> margin. Selecting only points 1 and 4 gives partial credit.

Wouldnt x(1) , x(4) and x(5) be possible too?

The margin of using these support vectors would be 1, which is smaller than the margin when including point 2.

5D

REVISED ANSWER: 527 or 30.

Why can't we just take 3 points (that contains negative and positive support vectors) out of the 17 support vectors, and store their features and classification, since 3 support vectors can uniquely identify the decision boundary?

All support vectors define the resulting SVM. Notice that for this problem we are given 30 features, which implies that the support vector machine operates in this higher-dimensional space.

Is it possible for us to just store the final length 30 vector from summing all the alpha weighted support vectors instead?

The question was intended to store the dual SVM, such that the kernel method can be applied. As we did not specify that kernels should be used, we now allow both answers: 527 and 30.

6A

Included partially correct solutions.

6C

Assume that the 64x64 computed means matrix also undergoes compression?

To our knowledge, it is not correct to compress the mean feature vector as well. One would not be able to mean-center the new data properly.

-To compress/reconstruct a matrix, all we need to store is X + the means (as mentioned in the question), then we can automatically decompose it into the SVD matrix and contain U tilde and Z for compression/reconstruction.

-You can simply store the mean map (64 * 64) and then store the magnitude of the 50 most significant features for the 500 images (50 * 500). This works, because from the mean map, you can always deterministically recompute the identical SVD that you used to compress, so there's no need to store the final result if you can store the inputs.

Notice the statement, “First, you choose to represent each pixel by a single feature.” which fixes the features. All subsequent steps and the question are based on this choice.

It is unclear what the “mean map” is. If this suggestion refers to the means, we are not aware how we can use only the means to perform the SVD properly.

If the suggestion refers to the mean-centred data matrix or the sample covariance matrix, then notice that their sizes are larger than the answer to this question.

Mean-centred data matrix: 4096 * 500 = 2048000.

Sample covariance matrix: 4096 * 4096 = 16777216.

9B

REVISED ANSWER: 4 or ` 4 or

dL⁄dw_1 = dL⁄dy(hat) * dy(hat)⁄dz_3 * dz_3⁄dz_1 * dz_1⁄dw_1 = (y(hat)-y)g[2]'(z_3)x_1. For first sample: 4 x 1 x 1 = 4 For second sample: -2 x 0 x 0 = 0 Taking the derivative of RELU g[2]'(z_3) as 0 for z_3 =0.

I calculated the first sample to be 4.

Accept the answer that computes the gradient for individual samples and provides explanations.

9C

I indicated C instead of A, as I was considering the case where the weights are all 0, leading to the output of the sigmoid activation always predicting a single class. There seems to be no clear linear decision boundary in this case (undefined) and so I considered that non-linear.

If all weights are zero, the model collapses to a constant output and no decision boundary exists. In other words, decision boundary cannot be generated by a model with all weights set to zero. If the model can generate a decision boundary (i.e., if not all weights are zero), the generated decision boundary will always be linear. Thus, option A is correct: The decision boundary generated by this neural network is always linear.

10A

I accidentally swapped the rows and columns. Is it possible to consider partial mark in that aspect?

No partial mark for this.

10C

REVISED ANSWER: 4 or 4x4

I accidentally wrote “4x4” instead of “4” for Question 29. I meant to enter just “4” as the value of N, but mistakenly wrote the full receptive field size.

We understand how '4x4' might have been included, reflecting the full receptive field size instead of just '4' for the value of N. To be lenient, we've decided to accept your answer.

10D

REVISED ANSWER: 135 or 3(input channels) x 5(output channels) x 9(weights for each kernel) = 135

I assumed there was bias, so I got 140 instead of 135. Could this be accepted as well?

It was announced during the exam that there is no bias.

Is it possible that the same kernel is applied to all 3 channels to generate one output channel, so in total there will only be 5 "unique" kernels, and hence the total number of weights is 3x3x5 = 45?

It is stated in the question that you need to apply the convolution operation introduced in the lecture. So, for each channel, different weight matrices should be applied.

I calculated the answer to be 135.

Accept the answer that computes the number of weights and explains how the result is computed.

10E

REVISED ANSWER: **B** only or **E** only

The phrasing says "which settings would help you generate the required output," which I interpreted as allowing for intermediate steps, rather than requiring a single-step solution.

Option B: One kernel with height set to 1 and width set to 1, all weight values set to 1, stride 1, no padding.

Option B's configuration can compute the sum of corresponding elements across the three input channels. Dividing the output of Option B by 3 would then produce the required result. Since we did not explicitly require directly generating the output, we accept B.

11A

This question asks which models **can** be used, rather than which are best suited for use.

Option A: CNN

CNNs use filters/kernels that scan across the image to detect features. One possible way to extract features is by using the output from one of the convolution layers.

Option B: RNN

As mentioned in Tutorial 10, Question D3, an image can actually be converted into a sequence and then processed by an RNN. Therefore, an RNN can also be used to extract features from an image.