ML Exam Questions 15/1/2021

Question 1 (Bias/Variance Tradeoff)

Tell if the following statements about the bias-variance dilemma (and related topics) are true or false. Motivate your answers.

Regularization techniques are likely to increase the bias of a model.
TRUE: regularization acts as constraints or penalty on parameters variability and, hence, reduces the variance of the model and at the same time increases its bias.

If we focus on a specific task, it does not exist an ML algorithm that performs better than the others.
FALSE: If we focus on a specific task, a single ML algorithm can outperform others. This is not true if we, instead, consider all the possible problems (no free lunch theorem).

If a logistic regression classifier does not achieve the desired performance (on a test set) one might consider an SVM with a linear kernel as a method to improve the performances.
TRUE: Even if SVM with linear kernel still finds a linear boundary, it might find a better solution in terms of test error, because it will find the maximum margin separating hyperplane and soft margins can be used to improve generalization. Nonetheless, we are expecting that the two methods would provide similar results.

If the performance on the training is matching the desired performance, while the one on a test set is not satisfactory the model might be too complex for the task.
TRUE: we might have provided a model which is overfitting the available data. Using a simpler model seems a proper choice.

Increasing the training set size always improves the model performance.
FALSE: if the model selected presents a bias w.r.t. the real process, increasing the training set size would provide only limited benefits. In general, the more data we have, the more we are reducing the variance of the model.

It is a good idea to increase the number of samples used for training if we decided to increase the model complexity.
TRUE: the more complex is the model, the more data I usually need to train it in order to avoid overfitting.

Removing features randomly to a given model might help if the model has a large variance due to limited availability of data.

FALSE: selecting a subset of features might be a good idea to reduce the variance, but removing them at random might exclude relevant information from the model, decreasing even more its performance on a test set. Notice that there exists some techniques which include randomicity in the selection of the feature to use in the model selection phase.

The use of the error on a validation set is suggested when we have a small dataset and we want to discriminate the performance of a set of computationally expensive models.
FALSE: with computationally hard models and only a few data one should prefer the use of an adjustment technique, like AIC, to get an estimate of the test error.

Question 2 (MAB)

Tell if the following statements about MAB are true or false. Motivate your answers.

A MAB setting provides more information per round than the ones we have in an expert one.
FALSE: we have feedback only for a single arm, differently from the expert setting in which the feedback for each available arm.

In a setting with Gaussian rewards we can use the TS algorithm to have sublinear regret.
FALSE/TRUE: if we use the Beta prior, the conjugate distribution is the Bernoulli, thus the rewards should be {0,1}. Instead, if we use a Gaussian prior, its conjugate is Gaussian itself and it is possible to execute the TS algorithm with Gaussian rewards.

Since the TS is a randomized algorithm, it provides sublinear guarantees on the regret also in an adversarial MAB setting.
FALSE: the results on the regret we have for the TS algorithm are only valid in the stochastic setting.

The selection of the arm provided by UCB1 is deterministic.
TRUE: it relies on the selection of the largest arm upper bound, whose computation is deterministic.

In the presence of prior information on the arms rewards, we should use the UCB1 algorithm.
FALSE: the best option to exploit the prior information is the use of Bayesian algorithm, that for the MAB setting is the use of TS.

In the case we do not know the nature of the MAB problem (stochastic or adversarial) it is a good idea to use a stochastic algorithm, like the TS.
FALSE: the stochastic setting is a more specific model than the adversarial one and, therefore, we are not assured the algorithms designed for it are able to get a sublinear regret.

The upper bound on the regret of UCB1 and TS scales logarithmically with the time horizon.

TRUE: it has been shown that their upper bound on the regret is dependent on log(T), where T is the time horizon.

The design of MAB algorithms different from TS might provides an expected pseudo-regret of order log(log(T)), T being the time horizon.
FALSE: this is prevented from the theorem on the lower bound for the regret of stochastic MAB setting, providing a lower bound of log(T).

Question 3 (Linear Regression)

Tell if the following statements about Linear Regression are true or false. Motivate your answers.

Linear regression can be the ideal choice also when the target is a nonlinear function of the problem variables.
TRUE, in fact linear regression models can be extended by mapping the problem space into a feature space to account for nonlinearity.

The feature space for linear regression can be the result of any nonlinear mapping of the problem space.
TRUE, linear regression models require the target to be linear only with respect to the model parameters and not to the input.

The easiest approach to train the model parameters in linear regression is to minimize the mean absolute error of the model.
FALSE, the sum of squared error is the most convenient loss function because it allows us to compute a closed form solution.

As long as we assume the targets are generated by a linear model with a zero-mean Gaussian noise, the model parameters with the maximum likelihood always minimize the residual sum of squares (RSS).
TRUE, we can prove that under the above assumption, maximizing the log-likelihood of the model parameters lead exactly to minimize the RSS.

In linear regression, the larger the values of the parameters the higher is the probability of overfitting the training data.
TRUE, larger parameters lead to larger variance and, hence, might lead to overfitting the training data.

Lasso and ridge regression require to minimize the same loss function.
FALSE, despite they both extend RSS, in ridge regression the regularization term is the norm-2 of the parameters vector while in lasso it is the norm-1.

The probabilistic interpretation of ridge regression is that it corresponds to choosing the model parameters that maximize the posterior probability, while assuming an infinitely broad Gaussian prior.
FALSE, it corresponds to assuming a zero mean Gaussian prior with a finite variance.

Bayesian Linear Regression framework is less suitable than Ordinary Least Squares to perform sequential training due to the higher computational complexity.
FALSE, sequential training is possible in Bayesian Linear Regression while it is not with OLS. An alternative solution for OLS would be to apply the recursive least square algorithm to reduce the computational complexity by processing each new datum in an efficient way.

Code 1 (PCA)

The code snippet below implements the training procedure of three different models. Tell if the following statements are true or false and provide adequate motivations.

```
1     % the iris dataset is a set of 150 records with 4 features and 3 targets.
2     % iris_dataset contains: irisInputs 150x4 double, irisTargets 150x3 double.
3     % find(condition, 1) returns first index for which condition holds.
4 -   load iris_dataset;
5
6 -   model1 = mnrfit(irisInputs, irisTargets);
7
8 -   [loadings, scores, variance] = pca(irisInputs);
9 -   irisInputs2 = scores(:,1:3);
10 -  model2 = mnrfit(irisInputs2, irisTargets);
11
12 -  perc_variance = cumsum(variance) / sum(variance);
13 -  vv = find(perc_variance >= 0.95, 1);
14 -  irisInputs3 = scores(:,1:vv);
15 -  model3 = mnrfit(irisInputs3, irisTargets);
```

It is likely that model1 has a lower training error than the other two models.
TRUE, model1 is trained on the full feature space, so it is more likely to overfit the training set, and to achieve a smaller training error.

The model2 is more prone to overfitting than model1.
FALSE, model2 is trained on a reduced feature space, so it is less prone to overfitting.

The pca function at line 8 provides a useful methodology to identify the most significant features in the dataset.
FALSE, pca implements the principal component analysis, which is not a feature selection method. PCA provides a new set of features that are obtained through linear combinations of the original features.

The loadings matrix contains the principal components of the irisInputs, thus it contains four orthogonal vectors.

TRUE, principal component analysis provides a new basis for the feature space, which is four-dimensional.

We can guarantee that model3 is trained on a feature space that explains at least 95 percent of the variance of the original feature space.
TRUE, line 13 in the snippet above identifies the number of principal components for which the cumulative variance is at least 95 percent of the original variance.

We can say with certainty that the last element of the vector perc_variance is 1.
TRUE, the snippet stores in the n-th element of perc_variance the percentage of the cumulative variance explained by the first n principal components. The full set of principal components explains the whole variance.

The first column of the loadings matrix identifies the direction of greatest variability in the iris dataset feature space.
TRUE, the first column of the loadings matrix is the first principal component, which identifies the direction with greatest variability in the dataset.

The matrix irisInputs can be perfectly reconstructed from irisInputs2 and the loadings matrix.
FALSE, we can reconstruct the irisInputs with minimal reconstruction loss, but it will be more than zero in general.

Exercise 1 (MC/TD)

VERSION A
Consider the set of trajectories below, which are obtained by running a given policy in an MDP with three states S={A, B, C} (C is terminal) and two actions A={left, right}. A trajectory is denoted by a sequence (S_t, A_t, R_t).

(A,right,-1) -> (A,left,4) -> (B,left,1) -> (C)
(B,left,4) -> (A,right,-3) -> (C)
(A,left,1) -> (B,right,-2) -> (A,left,1) -> (B,left,1) -> (C)

1) Compute the state-action value function Q(A,right) by resorting to TD evaluation. Assume to start from zero values for each state, alpha=0.5, gamma=1.
TD update rule: $Q(S, A) = Q(S, A) + \text{alpha} * (R + \text{gamma} * Q(S',A') - Q(S,A))$

$Q(A,right) = Q(A,right) + \text{alpha} * (R + \text{gamma} * Q(A,left) - Q(A,right)) = 0 + 0.5*(-1) = -0.5$
$Q(A,right) = Q(A,right) + \text{alpha} * (R + \text{gamma} * Q(C,\bullet) - Q(A,right)) = -0.5 + 0.5*(-2.5) = -1.75$

2) Compute the state-action value function for every meaningful state-action pair by resorting to first-visit MC evaluation.
Q(A,left) = (5 + 1) / 2 = 3
Q(A,right) = (4 - 3) / 2 = 0.5
Q(B,left) = (1 + 1 + 1) / 3 = 1
Q(B,right) = 0

3) What does the greedy policy prescribe according to the MC first-visit evaluation?
pi (left | A) = 1, pi (right | A) = 0
pi (left | B) = 1, pi (right | B) = 0

4) Assume to have performed the MC first-visit evaluation with an infinite number of trajectories from the same policy. What can we say about the optimal policy?
We cannot say anything about the optimal policy. MC first-visit is an unbiased estimator, so the evaluation converges to the exact state-action values. But these values only refer to the current policy.

VERSION B
Consider the set of trajectories below, which are obtained by running a given policy in an MDP with three states S={A, B, C} (C is terminal) and two actions A={left, right}. A trajectory is denoted by a sequence (S_t, A_t, R_t).

(A,right,1) -> (A,left,4) -> (B,left,1) -> (C)
(A,left,1) -> (B,right,-2) -> (A,left,1) -> (B,left,1) -> (C)
(B,left,4) -> (A,right,3) -> (C)

1) Compute the state-action value function Q(A,right) by resorting to TD evaluation. Assume to start from zero values for each state, alpha=0.5, gamma=1.
TD update rule: Q(S, A) = Q(S, A) + alpha * (R + gamma * Q(S',A') - Q(S,A))

Q(A,right) = Q(A,right) + alpha * (R + gamma * Q(A,left) - Q(A,right)) = 0 + 0.5*(1) = 0.5
Q(A,right) = Q(A,right) + alpha * (R + gamma * Q(C,•) - Q(A,right)) = 0.5 + 0.5*(2.5) = 1.75

2) Compute the state-action value function for every meaningful state-action pair by resorting to first-visit MC evaluation.
Q(A,left) = (5 + 1) / 2 = 3
Q(A,right) = (6 + 3) / 2 = 4.5
Q(B,left) = (1 + 1 + 7) / 3 = 3
Q(B,right) = 0

3) What does the greedy policy prescribe according to the MC first-visit evaluation?
pi (left | A) = 0, pi (right | A) = 1
pi (left | B) = 1, pi (right | B) = 0

4) Assume to have performed the MC first-visit evaluation with an infinite number of trajectories from the same policy. What can we say about the optimal policy?
We cannot say anything about the optimal policy. MC first-visit is an unbiased estimator, so the evaluation converges to the exact state-action values. But these values only refer to the current policy.

VERSION C

Consider the set of trajectories below, which are obtained by running a given policy in an MDP with three states S={A, B, C} (C is terminal) and two actions A={left, right}. A trajectory is denoted by a sequence (S_t, A_t, R_t).

(A,right,3) -> (A,left,4) -> (B,left,1) -> (C)
(A,left,1) -> (B,right,-2) -> (A,left,1) -> (B,left,1) -> (C)
(B,left,-4) -> (A,right,-1) -> (C)

1) Compute the state-action value function Q(A,right) by resorting to TD evaluation. Assume to start from zero values for each state, alpha=0.5, gamma=1.
TD update rule: $Q(S, A) = Q(S, A) + \text{alpha} * (R + \text{gamma} * Q(S',A') - Q(S,A))$

$Q(A,right) = Q(A,right) + \text{alpha} * (R + \text{gamma} * Q(A,left) - Q(A,right)) = 0 + 0.5*(3) = 1.5$
$Q(A,right) = Q(A,right) + \text{alpha} * (R + \text{gamma} * Q(C,\bullet) - Q(A,right)) = 1.5 + 0.5*(-2.5) = 0.25$

2) Compute the state-action value function for every meaningful state-action pair by resorting to first-visit MC evaluation.
$Q(A,left) = (5 + 1) / 2 = 3$
$Q(A,right) = (8 - 1) / 2 = 3.5$
$Q(B,left) = (1 + 1 - 5) / 3 = -1$
$Q(B,right) = 0$

3) What does the greedy policy prescribe according to the MC first-visit evaluation?
pi (left | A) = 0, pi (right | A) = 1
pi (left | B) = 0, pi (right | B) = 1

4) Assume to have performed the MC first-visit evaluation with an infinite number of trajectories from the same policy. What can we say about the optimal policy?
We cannot say anything about the optimal policy. MC first-visit is an unbiased estimator, so the evaluation converges to the exact state-action values. But these values only refer to the current policy.

VERSION D

Consider the set of trajectories below, which are obtained by running a given policy in an MDP with three states S={A, B, C} (C is terminal) and two actions A={left, right}. A trajectory is denoted by a sequence (S_t, A_t, R_t).

(A,right,-3) -> (A,left,4) -> (B,left,1) -> (C)
(B,left,-4) -> (A,right,1) -> (C)
(A,left,1) -> (B,right,-2) -> (A,left,1) -> (B,left,1) -> (C)

1) Compute the state-action value function Q(A,right) by resorting to TD evaluation. Assume to start from zero values for each state, alpha=0.5, gamma=1.
TD update rule: $Q(S, A) = Q(S, A) + \alpha * (R + \gamma * Q(S',A') - Q(S,A))$

$Q(A,right) = Q(A,right) + \alpha * (R + \gamma * Q(A,left) - Q(A,right)) = 0 + 0.5*(-3) = -1.5$
$Q(A,right) = Q(A,right) + \alpha * (R + \gamma * Q(C,\bullet) - Q(A,right)) = -1.5 + 0.5*(2.5) = -0.25$

2) Compute the state-action value function for every meaningful state-action pair by resorting to first-visit MC evaluation.
Q(A,left) = (5 + 1) / 2 = 3
Q(A,right) = (2 + 1) / 2 = 1.5
Q(B,left) = (1 - 3 + 1) / 3 = -1/3
Q(B,right) = 0

3) What does the greedy policy prescribe according to the MC first-visit evaluation?
pi (left | A) = 1, pi (right | A) = 0
pi (left | B) = 0, pi (right | B) = 1

4) Assume to have performed the MC first-visit evaluation with an infinite number of trajectories from the same policy. What can we say about the optimal policy?
We cannot say anything about the optimal policy. MC first-visit is an unbiased estimator, so the evaluation converges to the exact state-action values. But these values only refer to the current policy.

Exercise 2 (Perceptron)

VERSION A
Consider a binary classifier defined by parameters w = [2, -1, 1] with quadratic features (i.e., phi([a, b]) = [a^2, a*b, b^2]). Answer the following questions related to the perceptron algorithm. Provide full calculations and clear motivations.

1) Predict the class of the data point x_1 = [0, 2] with the given classifier (+1 represents the positive class, -1 the negative class).
y_1 = w' * phi(x_1) = [2, -1, 1]' * [0, 0, 4] = 4 > 0 → +1 positive class.

2) Consider the training data points:
(x_2, t_2) = ([1, 1], -1),
(x_3, t_3) = ([0, 1], +1),
(x_4, t_4) = ([1, 0], +1).
Compute the value of the perceptron loss over the dataset.
y_2 = w' * phi(x_2) * t_2 = [2, -1, 1]' * [1, 1, 1] * -1 = -2 --> misclassified
y_3 = w' * phi(x_3) * t_3 = [2, -1, 1]' * [0, 0, 1] * 1 = 1
y_4 = w' * phi(x_4) * t_4 = [2, -1, 1]' * [1, 0, 0] * 1 = 2
L = 2 (since the loss only accounts for misclassified samples)

3) Consider training the perceptron via stochastic gradient descent, under which assumptions the algorithm is guaranteed to converge? If you are likely to have outliers in your dataset, do you think that the perceptron provides a good modeling technique?
- The perceptron algorithm trained with stochastic gradient descent is guaranteed to converge in a finite number of steps if and only if the dataset is linearly separable in the feature space.
- Probably not, the perceptron would not converge. It would be better to employ a model that allows for a soft boundary, such as SVM or logistic regression.

VERSION B
Consider a binary classifier defined by parameters w = [2, 1, -1] with quadratic features (i.e., phi([a, b]) = [a^2, a*b, b^2]). Answer the following questions related to the perceptron algorithm. Provide full calculations and clear motivations.

1) Predict the class of the data point x_1 = [0, 2] with the given classifier (+1 represents the positive class, -1 the negative class).
y_1 = w' * phi(x_1) = [2, 1, -1]' * [0, 0, 4] = -4 < 0 → -1 negative class.

2) Consider the training data points:
(x_2, t_2) = ([1, 1], +1),
(x_3, t_3) = ([0, 1], +1),
(x_4, t_4) = ([1, 0], -1).
Compute the value of the perceptron loss over the dataset.
y_2 = w' * phi(x_2) * t_2 = [2, 1, -1]' * [1, 1, 1] * 1 = 2
y_3 = w' * phi(x_3) * t_3 = [2, 1, -1]' * [0, 0, 1] * 1 = -1 --> misclassified
y_4 = w' * phi(x_4) * t_4 = [2, 1, -1]' * [1, 0, 0] * -1 = -2 --> misclassified
L = 3 (since the loss only accounts for misclassified samples)

3) Consider training the perceptron via stochastic gradient descent, under which assumptions the algorithm is guaranteed to converge? Suppose it does not converge on a given dataset, what would you try to increase its chance of convergence?
- The perceptron algorithm trained with stochastic gradient descent is guaranteed to converge in a finite number of steps if and only if the dataset is linearly separable in the feature space.
- Probably not, the perceptron would not converge. It would be better to employ a model that allows for a soft boundary, such as SVM or logistic regression.

VERSION C

Consider a binary classifier defined by parameters w = [-2, -1, 1] with quadratic features (i.e., phi([a, b]) = [a^2, a*b, b^2]). Answer the following questions related to the perceptron algorithm. Provide full calculations and clear motivations.

1) Predict the class of the data point x_1 = [3, 0] with the given classifier (+1 represents the positive class, -1 the negative class).
y_1 = w' * phi(x_1) = [-2, -1, 1]' * [9, 0, 0] = -18 < 0 → -1 negative class.

2) Consider the training data points:
(x_2, t_2) = ([1, 1], +1),
(x_3, t_3) = ([0, 1], -1),
(x_4, t_4) = ([1, 0], -1).
Compute the value of the perceptron loss over the dataset.
y_2 = w' * phi(x_2) * t_2 = [-2, -1, 1]' * [1, 1, 1] * 1 = -2 --> misclassified
y_3 = w' * phi(x_3) * t_3 = [-2, -1, 1]' * [0, 0, 1] * -1 = -1 --> misclassified
y_4 = w' * phi(x_4) * t_4 = [-2, -1, 1]' * [1, 0, 0] * -1 = 2
L = 3 (since the loss only accounts for misclassified samples)

3) Consider training the perceptron via stochastic gradient descent, under which assumptions the algorithm is guaranteed to converge? Suppose it does not converge on a given dataset, what would you try to increase its chance of convergence?
- The perceptron algorithm trained with stochastic gradient descent is guaranteed to converge in a finite number of steps if and only if the dataset is linearly separable in the feature space.
- Probably not, the perceptron would not converge. It would be better to employ a model that allows for a soft boundary, such as SVM or logistic regression.

VERSION D

Consider a binary classifier defined by parameters w = [1, -1, -2] with quadratic features (i.e., phi([a, b]) = [a^2, a*b, b^2]). Answer the following questions related to the perceptron algorithm. Provide full calculations and clear motivations.

1) Predict the class of the data point x_1 = [3, 0] with the given classifier (+1 represents the positive class, -1 the negative class).
y_1 = w' * phi(x_1) = [1, -1, -2]' * [9, 0, 0] = 9 > 0 → +1 positive class.

2) Consider the training data points:
(x_2, t_2) = ([1, 1], +1),
(x_3, t_3) = ([0, 1], -1),
(x_4, t_4) = ([1, 0], -1).
Compute the value of the perceptron loss over the dataset.
y_2 = w' * phi(x_2) * t_2 = [1, -1, -2]' * [1, 1, 1] * 1 = -2 --> misclassified
y_3 = w' * phi(x_3) * t_3 = [1, -1, -2]' * [0, 0, 1] * -1 = 2
y_4 = w' * phi(x_4) * t_4 = [1, -1, -2]' * [1, 0, 0] * -1 = -1 --> misclassified
L = 3 (since the loss only accounts for misclassified samples)

3) Consider training the perceptron via stochastic gradient descent, under which assumptions the algorithm is guaranteed to converge? Suppose it does not converge on a given dataset, what would you try to increase its chance of convergence?
- The perceptron algorithm trained with stochastic gradient descent is guaranteed to converge in a finite number of steps if and only if the dataset is linearly separable in the feature space.
- Probably not, the perceptron would not converge. It would be better to employ a model that allows for a soft boundary, such as SVM or logistic regression.