ML 27/7/2020 - Solutions

Q1 (Model Selection)

Answer to the following questions about the bias-variance decomposition, model selection, and related topics. Motivate your answers.

If your linear regression model underfits the training data (i.e., the model is not complex enough to explain the data), would you apply PCA to compute a more suitable feature space for your model?
NO, because if linear regression does not fit training data (underfitting), the features computed with PCA will not solve the problem as they are linear combinations of input variables.

If solving a regression problem, the design matrix (X^TX), is singular, would you apply PCA to solve this issue?
YES, as a singular design matrix is likely to be due to collinear features, I could apply PCA and select the K components associated to eigenvalues that are not zero (or close to zero).

You need to train a prediction model from a dataset of patients affected by one among a set of 5 very different diseases (that can be easily discriminated using the features available in the dataset for each patient). Assuming that we need to reduce the number of features, would you apply PCA?
NO, because the dataset seems to contain several clusters and this would not make very effective the representation of PCA. Moreover, PCA will not guarantee that the reduced set of features is the best choice to solve our prediction model.

Assuming a classifier fits very well the training data but underperforms on the validation set, would you apply Bagging or Boosting to improve it?
BAGGING, because it might reduce the variance of the model. In contrast, boosting allows to reduce bias without increasing (significantly) the variance (however, in this example we have a low bias and high variance).

Assuming that you trained a classifier with a K-fold cross-validation and it consistently has poor performances both on training and on validation folds, would you apply Bagging or Boosting to improve it?
BOOSTING, because it can successfully reduce bias of a stable learner without increasing the variance which seems to be the problem of the learner in this case.

You applied ridge regression to train a linear model using a rather large regularization coefficient, would you think that bagging would improve your model?
NO, because ridge regression with large regularization coefficient will be very stable (limited variance) and this would not allow to exploit significantly bagging.

You have been asked to implement a feature selection process on a system with very limited computational resources. Would you opt for a filter approach or for a wrapper approach?
FILTER, because a wrapper approach involves solving an optimization problem that requires training several models. In contrast, filter approaches only require to compute statistics on the features.

You have been asked to implement a feature selection process to improve as much as possible the performance of your model. Would you opt for a filter approach or for a wrapper approach?
WRAPPER, because filter approaches assume features are independent and might not find the best subset.

If you want to rank (in order of importance) the features of a classification problem, which kind of feature selection process would you use among the ones presented in the course?
FILTER, because it involves computing a metric for each feature (e.g., correlation or information gain) such that on the basis of this metric features can be ranked and selected from the most relevant to the least one.

You need to train a linear regression model using as input the readings of several sensors. Assuming that you know that some of these sensors might be faulty (i.e., resulting in meaningless readings), which linear regression approach would you use to train your model?
LASSO, because it implicitly performs a feature selection that will get rid of the faulty sensors.

A linear regression model, computed using ordinary least squares, has a validation error that is much larger than training error. Assuming that you do not want to change neither the input features nor the kind of model, what would you do to improve it?
REGULARIZATION can help me improve the performance by reducing the variance. In particular RIDGE regression would be the most obvious choice. Another solution, if viable, is to increase the number of samples used for training.

If you have to choose among a few models knowing only the training error (assuming you cannot retrain them or evaluate them on a different dataset), what would you do?
ADJUSTED COMPLEXITY METRICS can be used to "correct" the training error taking into account also the model complexity.

Q2 (MAB)

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

In a setting with Gaussian rewards we can use the UCB1 algorithm to have sublinear regret.
FALSE: the UCB1 is based on the Hoeffding inequality which holds for limited-support rewards.

In the presence of prior information on the arms rewards, we should use the EXP3 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. Moreover, in the case the setting is adversarial, there is no motivation to reuse past information on the arms since the reward is selected by an opponent.

In the case we do not know the nature of the MAB problem (stochastic or adversarial) it is a good idea to use the EXP3 algorithm.
TRUE: the adversarial setting is a more general model than the stochastic one and is able to get a sublinear regret in both cases.

An arm with a large upper confidence bound is likely to turn out to be the optimal one at the end of the MAB time horizon.
FALSE: it also might be that we have high uncertainty about its value but turn out to have small expected reward.

A MAB algorithm can be used in an MDP as long as the reward corresponding to the actions have the same distribution in every state.
TRUE: having the same reward distribution in each state is equivalent to say that we have a single state, assuming that all the actions can be performed in each state. Another option is to run a MAB algorithm where each arm corresponds to specific policy to converge to the optimal one over a finite set.

The Thompson Sampling algorithm can be applied only if we have bernoulli rewards (success/failures) as rewards.
FALSE: in generale we need a pair of conjugate distributions to make it work. (I am also accepting as an answer that during the course we only analysed the algorithm with bernoulli distributions)

The design of MAB algorithm different from UCB1 might provide 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).

An expert setting (e.g., weather forecasting) provides more information per round than the ones we have in a MAB one.
TRUE: we have a feedback for each arm, differently from the MAB setting in which the feedback is only for the selected arm.

The upper bound on the regret of UCB1 scales linearly with the number of arms.
TRUE: the upper bound has a summation over the arms, therefore if we increase the number of arms to consider, we also increase linearly the regret.

There exist situations in which we are able to obtain a regret lower than the one prescribed by the lower bound for MAB.

TRUE: the lower bound holds on average and on generic MAB problems. For specific realizations or for specific MAB settings we might get a smaller regret than the one prescribed by the bound.

It is generally a viable option to use an epsilon-greedy exploration strategy on a MAB setting.
TRUE: for the stochastic setting. If we have an adversarial setting we require to have enough randomization in the arm selection.
(I am also accepting a FALSE if the comment is that we do not have any theoretic result on the regret for such algorithms)

The selection of the arm provided by TS is stochastic.
TRUE: it relies on the selection of a sample from a posterior distribution, therefore is intrinsically randomized.

Since the TS is a randomized algorithm, it can be used in an adversarial MAB setting.
FALSE: the results on the regret we have for the TS algorithm are only valid in the stochastic setting.

Q3 (Kernel/SVM)

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

To tackle a classification problem, you split the dataset in three equal parts, and you train three separate but equivalent linear models on the splits. The resulting parameters are very similar across the models, but you are getting large training errors. In this case, employing a kernel method would NOT help.
FALSE: the description suggests the presence of low variance and high bias, thus employing a kernel method might help.

You face a regression problem on a dataset with N samples and M=N^2 features. It is likely that the dual representation of the linear regression would be more efficient than the original (primal) one.
TRUE: the dual representation requires to compute the inverse of an NxN matrix, while the primal requires to invert an MxM matrix.

The kernel trick allows to deal with complex classification problems by reducing the dimensionality of the feature space.
FALSE: the kernel trick allows for implicit computation of a high-dimensional scalar product, it is not a dimensionality reduction technique

Consider a function k(x, x') = exp(c_1 * k_1 (x, x') + c_2 * k_2 (x, x')), where k_1 and k_2 are valid kernel functions. We can say that k (x, x') represents a scalar product in some, possibly high-dimensional, feature space.
TRUE: under the assumption that c_1 and c_2 are greater than zero, k is a valid kernel function as it can be obtained by k_1, k_2 through common composition rules (linear combination and exponential), thus it represents a scalar product between implicit feature vectors of x and x'.

Consider to have trained a soft-margin SVM. You can safely say that any point which is associated with a parameter greater than zero is correctly classified by the SVM.
FALSE: every point associated to a parameter greater than zero is certainly a support vector, but in a soft-margin SVM it can be either correctly classified or misclassified.

Both a logistic regression and a SVM is a suitable choice to address a non-separable classification problem, but it is likely that the resulting separating hyper-planes will be different.
TRUE: we can use either a logistic regression or a SVM, but the objective functions of the two models are fairly different (maximum likelihood for logistic regression and margin maximization for SVM), thus we cannot guarantee that the hyper-planes will be equal.

To tackle a non-separable classification problem, it is a good idea to employ a soft-margin SVM. Especially, by setting a large value for the hyper-parameter C you can reduce the variance at the expense of some bias.
FALSE: the hyper-parameter C controls the penalization for misclassified samples, thus the larger is C the less misclassification we allow, and the less we reduce the variance of the trained model.

Similarly as any other memory-based method, such as k-NN, a support vector machine is lighting fast in training but fairly slow in prediction.
FALSE: the SVM training time scales with the power of three of the number of samples in the general case. Instead, being a sparse kernel method, it is usually fast in prediction.

If we train a hard-margin SVM over a linearly separable dataset, we cannot get any point inside the margins during the prediction phase.
FALSE: we will not have any point inside the margin in the training phase, we cannot control what will happen on new data points (prediction phase).

To address a 4-class classification problem with a one-against-one multi-class SVM, we have to train exactly 6 single-class SVMs.
TRUE: with one-against-one SVM k(k-1)/2 models are required, where k is the number of classes in the problem.

You have to buy a new machine to train general kernel methods, it is probably a good idea to focus on larger storage capabilities than computational power.

TRUE: kernel methods usually require to store all the training set to perform predictions, while computing kernel functions is unlikely to be particularly demanding in terms of computational resources.

You are trying to solve a non-separable classification problem with a linear model. You are not getting a good performance, but you can see a higher-dimensional feature space where the problem is linearly separable. Thus, it is a good choice to employ a kernel method.
FALSE: if you can see a feature space where the problem is linearly separable, probably the better choice is to train your model in that feature space, rather than employing a kernel method.
TRUE: If the explicit computation of the features is computationally expensive, one might use the kernel trick to avoid this problem and, at the same time, expand the feature space.

Q4 (Code Interpretation - RL)

The code snippet below implements the update loop of a well-known algorithm. Tell if the following statements are true or false and provide adequate motivations.
[Random choice between the two]
[Q-learning]

```
for i = 1:I
    a = epsilon_greedy_policy(s, Q, epsilon);
    [s_next, r] = transition_model(s, a);
    Q(s,a) = (1 - alpha) * Q(s,a) + ...
        alpha * (r + gamma * max(Q(s_next, :)));
    s = s_next;
end
```

[SARSA]

```
for i = 1:I
    [s_next, r] = transition_model(s, a);
    a_next = epsilon_greedy_policy(s_next, Q, epsilon);
    Q(s,a) = (1 - alpha) * Q(s,a) + ...
        alpha * (r + gamma * Q(s_next, a_next));
    s = s_next;
    a = a_next;
end
```

Describe the content of the lines provided in the snippet.
[Q-learning]
[SARSA]

The purpose of the reported algorithm is to get a reliable evaluation of the Q function for the considered policy.
[Q-learning] FALSE: Q-learning is not a policy evaluation algorithm but a control algorithm, thus the intended output is an optimal control policy

[SARSA] FALSE: SARSA is not a policy evaluation algorithm but a control algorithm, thus the intended output is an epsilon-optimal control policy. As a by-product, the algorithm also provides an estimate of the Q-function of the deployed policy.

The higher we set the value of alpha in line 4-5, the faster the algorithm will converge.
[Both] FALSE: the choice of a proper learning rate is problem-dependent. We can design a general learning rate to have a convergence guarantee, but not to control the rate of convergence.

With a sufficiently large value for I, the reported algorithm is guaranteed to converge to the optimal policy.
[Q-learning/SARSA] FALSE: even with a proper choice of the learning rate alpha, if we do not make the value of epsilon decay the algorithm will converge to an epsilon-optimal policy. (Only for Q-learning if we refer to the learned policy we are ensured to converge)

If we substitute the epsilon-greedy policy with any other stochastic policy ensuring sufficient exploration to all the possible actions the reported update loop would make sense as well.
[Q-learning] TRUE: Q-learning is an off-policy algorithm, if we ensure sufficient exploration we can learn an optimal target policy with any choice of the behavioral policy.
[SARSA] FALSE: SARSA is an on-policy algorithm, thus it requires to draw action with a greedy (or epsilon-greedy) policy.

The epsilon parameter allows to control the exploration-exploitation trade-off. Especially, the lower the value of epsilon, the more exploration we have.
[Both] FALSE: it is true that epsilon controls the exploration-exploitation trade-off, but a low value of epsilon will lead to less exploratory policies.

The update rule in line 4-5 is inspired by the Bellman expectation equation.
[Q-learning] FALSE: as in Value Iteration, the update rule of Q-learning is derived by the Bellman optimality equation.
[SARSA] TRUE: as in Policy Iteration, the update rule of SARSA is derived by the Bellman expectation equation.

If we would like to employ Monte-Carlo estimation in the update rule, we just need to substitute the term within the parenthesis in line 5 with the return.
[Both] FALSE: we cannot use Monte-Carlo in an online setting. Instead, we have to wait for an episode to end before updating the value function.

Q5 (Hypothesis Space Computation)

Questions

1) What is the cardinality of H?

2) How many samples N are necessary to guarantee that, with a probability at least of 0.99 the error of a consistent hypothesis (i.e., in the version space) is not greater than 0.05? Recall that Pr(\exists h \in H : error_D(h) = 0 & error_true(h) >= \epsilon) <= |H| e^(- \epsilon N).
3) If we assume the training error is not zero (i.e., we are in an agnostic learning setting), do we need less or more samples than the ones computed in (2) to guarantee that, with a probability at least of 0.99 the true error does not exceed the training error of more than 0.1?
4) Can you explain what would have changed if the cardinality of hypothesis space had been infinite instead (no computations are required in answering this question)?

Version A
Let us consider a classification problem in which the instance space is X = <x1, x2, x3, x4, x5> where each x_i is a boolean variable.
Consider also an hypothesis space H composed by rules of the following form:

if (x1=a1 or x2=a2 or x3=a3 or x4=a4 or x5=a5) then y=1, otherwise y=0

where a1, a2, …, a5 are parameters, defined for each hypothesis in the space, which can only be equal to 0 or 1.

Answer:

1) *Since, for each of the hypothesis there are 5 predicates connected by a boolean OR, and each of the predicate can assume two different meaning based on the value of the of it constant a_i, the cardinality of H is the following:*
   |H| = 2^5 = 32
2) *If the learner is consistent, we can compute the value of N using the corresponding bound:*
   *N >= 1 / \epsilon * (ln(|H|) + ln(1 / \delta)) = 20 * (5*ln(2) + 2ln(2) + 2ln(5)) ≈ 162*
3) *In the agnostic learning case we can compute the bound using the corresponding bound: N >= 1/(2*\epsilon^2) * (ln(|H|) + ln(1 / \delta)), therefore it would require a larger number of samples even as even doubling \epsilon (as in the agnostic bound it compares squared at denominator).*
4) *If the Hypothesis space is infinite, then, for the consistent case, an alternative bound is available, in which the VC-dimension of H is used instead of its cardinality. This bound is:*
   *N >= 1/\epsilon * (8 * VC(H) * log2(13/\epsilon)+ 4*log2(2 / \delta)).*

Version B
Consider a classification problem in which the instance space is X = <x1, x2, x3, x4, x5> where each x_i is a boolean variable.
Consider also an hypothesis space H composed by rules of the following form:

if (x1=1 <op1> x2=0 <op2> x3=1 <op3> x4=0 <op4> x5=1) then y=1, otherwise y=0

where <op_i> can be either the boolean operator "OR" or the boolean operator "AND" for each hypothesis.

Answer:

1) *Since, for each of the hypothesis there predicates are connected by two boolean operator, and each of the operators can assume two different meaning based on the value of the of its constant <op_i>, the cardinality of H is the following:*
   |H| = 2^4 = 16
2) *If the learner is consistent, we can compute the value of N using the corresponding bound:*

   *N >= 1/\epsilon * (ln(|H|) + ln(1 / \delta)) = 20 * (4*ln(2) + 2ln(2) + 2ln(5)) ≈ 148*
3) *In the agnostic learning case we can compute the bound using the corresponding bound: N >= 1/(2*\epsilon^2) * (ln(|H|) + ln(1 / \delta)), therefore it would require a larger number of samples even as even doubling \epsilon (as in the agnostic bound it compares squared at denominator).*
4) *If the Hypothesis space is infinite, then, for the consistent case, an alternative bound is available, in which the VC-dimension of H is used instead of its cardinality. This bound is:*
   *N >= 1/\epsilon * (8 * VC(H) * log2(13/\epsilon)+ 4*log2(2 / \delta))*

Version C

Consider a classification problem in which the instance space is X = <x1, x2, x3, x4> where each x_i is a boolean variable.
A decision tree is a binary tree that creates new nodes by splitting by the values of one of the variables at each level. This classifier assigns a boolean value to each of its leaves.
A 2-levels depth decision tree only splits over 2 of the 4 available variables.
Consider an hypothesis space H composed by all the possible 2-levels depth binary decision trees over the 4 given variables.

Answer:

1) *A 2-levels decision tree has 2^2 leaves, and for each leaf a different boolean value can be assigned. Since we can choose among all the possible subsets of two variables, the cardinality can be computed in this way:*
   *|H| = 2^(2^2) * n(n-1) / 2 = 2^(2^2) * 4 * 3 / 2 = 16 * 6 = 96*

2) *If the learner is consistent, we can compute the value of N using the corresponding bound:*

   *N >= 1/\epsilon * (ln(|H|) + ln(1 / \delta)) = 20 * (ln(96) + ln(100)) ≈ 184*
3) *In the agnostic learning case we can compute the bound using the corresponding bound: N >= 1/(2*\epsilon^2) * (ln(|H|) + ln(1 / \delta)), therefore it would require a larger number of samples even as even doubling \epsilon (as in the agnostic bound it compares squared at denominator).*
4) *If the Hypothesis space is infinite, then, for the consistent case, an alternative bound is available, in which the VC-dimension of H is used instead of its cardinality. This bound is:*

*N >= 1/\epsilon * (8 * VC(H) * log2(13/\epsilon)+ 4*log2(2 / \delta)).*

Version D

Consider a classification problem in which the instance space is X = <x1, x2, x3, x4, x5> where each x_i is a boolean variable.
A decision tree is a binary tree that creates new nodes by splitting by the values of one of the variables at each level. This classifier assigns a boolean value to each of its leaves.
A 2-levels depth decision tree only splits over 2 of the 5 available variables.
Consider an hypothesis space H composed by all the possible 2-levels depth binary decision trees over the 5 given variables.

Answer:

1) *A 2-levels decision tree has 2^2 leaves, and for each leaf a different boolean values can be assigned. Since we can choose among all the possible subsets of two variables, the cardinality can be computed in this way:*

   *|H| = 2^(2^2) * n(n-1) / 2 = 2^(2^2) * 5 * 4 / 2 = 16 * 10 = 160*

2) *If the learner is consistent, we can compute the value of N using the corresponding bound:*
   *N >= 1/\epsilon * (ln(|H|) + ln(1 / \delta)) = 20 * (ln(160) + ln(100)) ≈ 194*
3) *In the agnostic learning case we can compute the bound using the corresponding bound: N >= 1/(2*\epsilon^2) * (ln(|H|) + ln(1 / \delta)), therefore it would require a larger number of samples even as even doubling \epsilon (as in the agnostic bound it compares squared at denominator).*
4) *If the Hypothesis space is infinite, then, for the consistent case, an alternative bound is available, in which the VC-dimension of H is used instead of its cardinality. This bound is:*
   *N >= 1/\epsilon * (8 * VC(H) * log2(13/\epsilon)+ 4*log2(2 / \delta)).*

Q6 (Kernel KNN)

Questions
A KNN classifier classifies a new data point by applying a majority voting among its K-Nearest Neighbour.
1. Classify the new point according to a KNN classifier trained on the dataset reported below, assuming K = 3 and using the euclidean distance;
2. What happens if we use K = 10 instead? Do you think it is a good idea to choose such a parameter (hint: two pros if it is a good idea or two cons if it is not);
It is not a good idea since in this case we would have exactly the same results for each new

datapoint. Moreover, in general, it is a good practice to select an odd value for the K parameter to avoid ties.

3. Suggest a technique to set the parameter K.
We can use cross-validation (k-fold) to select the K having the lowest expected error.

Version A
Training Points
x_0 = (-2, -3, 0), y_0 = 1
x_1 = (-2, 2, 1), y_1 = 1
x_2 = (-2, -1, -3), y_2 = 1
x_3 = (2, 1, -3), y_3 = 1
x_4 = (2, -3, 2), y_4 = 1
x_5 = (1, 2, 1), y_5 = 0
x_6 = (-1, 1, -2), y_6 = 0
x_7 = (-1, 2, 2), y_7 = 1
x_8 = (1, -2, 0), y_8 = 0
x_9 = (-3, -3, -2), y_9 = 1

New Point
x = (0, 0, -1)

Using k=3
d(x, x_0) = d((0, 0, -1), (-2, -3, 0)) = sqrt(4 + 9 + 1) = sqrt(14) = 3.74
d(x, x_1) = d((0, 0, -1), (-2, 2, 1)) = sqrt(4 + 4 + 4) = sqrt(12) = 3.46
d(x, x_2) = d((0, 0, -1), (-2, -1, -3)) = sqrt(4 + 1 + 4) = sqrt(9) = 3.0
d(x, x_3) = d((0, 0, -1), (2, 1, -3)) = sqrt(4 + 1 + 4) = sqrt(9) = 3.0
d(x, x_4) = d((0, 0, -1), (2, -3, 2)) = sqrt(4 + 9 + 9) = sqrt(22) = 4.69
d(x, x_5) = d((0, 0, -1), (1, 2, 1)) = sqrt(1 + 4 + 4) = sqrt(9) = 3.0
d(x, x_6) = d((0, 0, -1), (-1, 1, -2)) = sqrt(1 + 1 + 1) = sqrt(3) = 1.73
d(x, x_7) = d((0, 0, -1), (-1, 2, 2)) = sqrt(1 + 4 + 9) = sqrt(14) = 3.74
d(x, x_8) = d((0, 0, -1), (1, -2, 0)) = sqrt(1 + 4 + 1) = sqrt(6) = 2.45
d(x, x_9) = d((0, 0, -1), (-3, -3, -2)) = sqrt(9 + 9 + 1) = sqrt(19) = 4.36

d(x, x_6) = 1.73 y_6 = 0
d(x, x_8) = 2.45 y_8 = 0
d(x, x_2) = 3.0 y_2 = 1
---
d(x, x_3) = 3.0 y_3 = 1
d(x, x_5) = 3.0 y_5 = 0
d(x, x_1) = 3.46 y_1 = 1

d(x, x_0) = 3.74 y_0 = 1
d(x, x_7) = 3.74 y_7 = 1
d(x, x_9) = 4.36 y_9 = 1
d(x, x_4) = 4.69 y_4 = 1
Actual Value: 0, KNN Classification: 0

Using k=10
Actual Value: 0, KNN Classification: 1

Version B
Training Points
x_0 = (-1, -3, -1), y_0 = 1
x_1 = (-3, -1, 1), y_1 = 1
x_2 = (-2, 1, -1), y_2 = 0
x_3 = (0, -1, 1), y_3 = 0
x_4 = (-1, 2, 2), y_4 = 1
x_5 = (-1, -2, 0), y_5 = 0
x_6 = (1, -1, -3), y_6 = 1
x_7 = (0, 2, 0), y_7 = 0
x_8 = (0, 2, 0), y_8 = 0
x_9 = (-2, -1, -3), y_9 = 1

New Point
x = (1, 0, -3)

****

Using k=3
d(x, x_0) = d((1, 0, -3), (-1, -3, -1)) = sqrt(4 + 9 + 4) = sqrt(17) = 4.12
d(x, x_1) = d((1, 0, -3), (-3, -1, 1)) = sqrt(16 + 1 + 16) = sqrt(33) = 5.74
d(x, x_2) = d((1, 0, -3), (-2, 1, -1)) = sqrt(9 + 1 + 4) = sqrt(14) = 3.74
d(x, x_3) = d((1, 0, -3), (0, -1, 1)) = sqrt(1 + 1 + 16) = sqrt(18) = 4.24
d(x, x_4) = d((1, 0, -3), (-1, 2, 2)) = sqrt(4 + 4 + 25) = sqrt(33) = 5.74
d(x, x_5) = d((1, 0, -3), (-1, -2, 0)) = sqrt(4 + 4 + 9) = sqrt(17) = 4.12
d(x, x_6) = d((1, 0, -3), (1, -1, -3)) = sqrt(0 + 1 + 0) = sqrt(1) = 1.0
d(x, x_7) = d((1, 0, -3), (0, 2, 0)) = sqrt(1 + 4 + 9) = sqrt(14) = 3.74
d(x, x_8) = d((1, 0, -3), (0, 2, 0)) = sqrt(1 + 4 + 9) = sqrt(14) = 3.74
d(x, x_9) = d((1, 0, -3), (-2, -1, -3)) = sqrt(9 + 1 + 0) = sqrt(10) = 3.16

d(x, x_6) = 1.0 y_6 = 1
d(x, x_9) = 3.16 y_9 = 1

d(x, x_2) = 3.74 y_2 = 0

---

d(x, x_7) = 3.74 y_7 = 0
d(x, x_8) = 3.74 y_8 = 0
d(x, x_0) = 4.12 y_0 = 1
d(x, x_5) = 4.12 y_5 = 0
d(x, x_3) = 4.24 y_3 = 0
d(x, x_1) = 5.74 y_1 = 1
d(x, x_4) = 5.74 y_4 = 1
Actual Value: 1, KNN Classification: 1

Using k=10
Actual Value: 1, KNN Classification: 0

Version C
Training Points
x_0 = (-2, -2, -3), y_0 = 1
x_1 = (2, 1, 1), y_1 = 0
x_2 = (1, -2, -2), y_2 = 1
x_3 = (-2, -3, -1), y_3 = 1
x_4 = (-1, 0, 0), y_4 = 0
x_5 = (0, -2, -2), y_5 = 1
x_6 = (-2, 2, 2), y_6 = 1
x_7 = (-2, -3, 2), y_7 = 1
x_8 = (2, -1, 1), y_8 = 0
x_9 = (-2, 2, -3), y_9 = 1

New Point
x = (0, -1, 0)

****

Using k=3
d(x, x_0) = d((0, -1, 0), (-2, -2, -3)) = sqrt(4 + 1 + 9) = sqrt(14) = 3.74
d(x, x_1) = d((0, -1, 0), (2, 1, 1)) = sqrt(4 + 4 + 1) = sqrt(9) = 3.0
d(x, x_2) = d((0, -1, 0), (1, -2, -2)) = sqrt(1 + 1 + 4) = sqrt(6) = 2.45
d(x, x_3) = d((0, -1, 0), (-2, -3, -1)) = sqrt(4 + 4 + 1) = sqrt(9) = 3.0
d(x, x_4) = d((0, -1, 0), (-1, 0, 0)) = sqrt(1 + 1 + 0) = sqrt(2) = 1.41
d(x, x_5) = d((0, -1, 0), (0, -2, -2)) = sqrt(0 + 1 + 4) = sqrt(5) = 2.24
d(x, x_6) = d((0, -1, 0), (-2, 2, 2)) = sqrt(4 + 9 + 4) = sqrt(17) = 4.12
d(x, x_7) = d((0, -1, 0), (-2, -3, 2)) = sqrt(4 + 4 + 4) = sqrt(12) = 3.46
d(x, x_8) = d((0, -1, 0), (2, -1, 1)) = sqrt(4 + 0 + 1) = sqrt(5) = 2.24
d(x, x_9) = d((0, -1, 0), (-2, 2, -3)) = sqrt(4 + 9 + 9) = sqrt(22) = 4.69

d(x, x_4) = 1.41 y_4 = 0
d(x, x_5) = 2.24 y_5 = 1
d(x, x_8) = 2.24 y_8 = 0
---
d(x, x_2) = 2.45 y_2 = 1
d(x, x_1) = 3.0 y_1 = 0
d(x, x_3) = 3.0 y_3 = 1
d(x, x_7) = 3.46 y_7 = 1
d(x, x_0) = 3.74 y_0 = 1
d(x, x_6) = 4.12 y_6 = 1
d(x, x_9) = 4.69 y_9 = 1
Actual Value: 0, KNN Classification: 0

Using k=10
Actual Value: 0, KNN Classification: 1

Version D
Training Points
x_0 = (2, 1, 1), y_0 = 0
x_1 = (-3, 1, 0), y_1 = 1
x_2 = (1, -3, 1), y_2 = 1
x_3 = (2, -1, 1), y_3 = 0
x_4 = (2, -3, -1), y_4 = 1
x_5 = (2, -1, -2), y_5 = 1
x_6 = (-3, -3, 2), y_6 = 1
x_7 = (0, 1, -1), y_7 = 0
x_8 = (2, -1, -2), y_8 = 1
x_9 = (-2, 0, 0), y_9 = 0

New Point
x = (-1, 0, 0)

****

Using k=3
d(x, x_0) = d((-1, 0, 0), (2, 1, 1)) = sqrt(9 + 1 + 1) = sqrt(11) = 3.32
d(x, x_1) = d((-1, 0, 0), (-3, 1, 0)) = sqrt(4 + 1 + 0) = sqrt(5) = 2.24
d(x, x_2) = d((-1, 0, 0), (1, -3, 1)) = sqrt(4 + 9 + 1) = sqrt(14) = 3.74
d(x, x_3) = d((-1, 0, 0), (2, -1, 1)) = sqrt(9 + 1 + 1) = sqrt(11) = 3.32
d(x, x_4) = d((-1, 0, 0), (2, -3, -1)) = sqrt(9 + 9 + 1) = sqrt(19) = 4.36
d(x, x_5) = d((-1, 0, 0), (2, -1, -2)) = sqrt(9 + 1 + 4) = sqrt(14) = 3.74
d(x, x_6) = d((-1, 0, 0), (-3, -3, 2)) = sqrt(4 + 9 + 4) = sqrt(17) = 4.12

d(x, x_7) = d((-1, 0, 0), (0, 1, -1)) = sqrt(1 + 1 + 1) = sqrt(3) = 1.73
d(x, x_8) = d((-1, 0, 0), (2, -1, -2)) = sqrt(9 + 1 + 4) = sqrt(14) = 3.74
d(x, x_9) = d((-1, 0, 0), (-2, 0, 0)) = sqrt(1 + 0 + 0) = sqrt(1) = 1.0

d(x, x_9) = 1.0 y_9 = 0
d(x, x_7) = 1.73 y_7 = 0
d(x, x_1) = 2.24 y_1 = 1
---
d(x, x_0) = 3.32 y_0 = 0
d(x, x_3) = 3.32 y_3 = 0
d(x, x_2) = 3.74 y_2 = 1
d(x, x_5) = 3.74 y_5 = 1
d(x, x_8) = 3.74 y_8 = 1
d(x, x_6) = 4.12 y_6 = 1
d(x, x_4) = 4.36 y_4 = 1
Actual Value: 0, KNN Classification: 0

Using k=10
Actual Value: 0, KNN Classification: 1