MLSM 2023 - Quiz 1

05/09/2023 - 13/09/2023 at 23:59

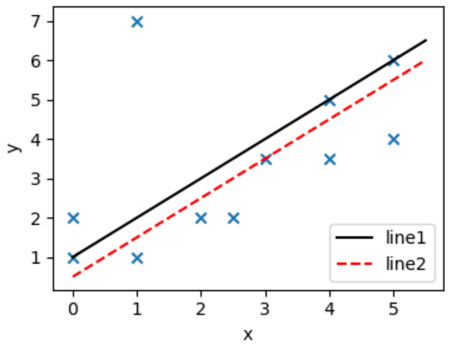
Questions based on lecture 1

(1) (1 pt.) Connect a description of a machine learning task with a correct category.

(a) Given an athlete's training data, predict the 100m running time in a competition	(a) Binary classification
(b) From location and time of the year, predict whether it is going to rain or not.	(b) Multi-label classification
(c) Learn to predict which drug from a given set of drugs would be the best to use for a patient against a disease.	(c) Ranking
(d) Predict which animal or animals are present in an image	(d) Regression

(2) (1.0 pt) Consider the data and the lines $l_1(x) = x + 1$ and $l_2(x) = x + 0.5$ shown below.

```
X = [1, 0, 0, 1, 2, 2.5,   3,   4, 4, 5, 5]
Y = [7, 1, 2, 1, 2,   2, 3.5, 3.5, 5, 4, 6]
line1 = lambda x: x+1
line2 = lambda x: x+0.5
```



Which line is a better fit to the data with respect to mean squared error, $mse(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$? What about mean absolute error, $mae(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i|$?

(a) (0.5 pt) MSE
   (i) Line 1 is a better fit
   (ii) Line 2 is a better fit

(b) (0.5 pt) MAE
   (i) Line 1 is a better fit
   (ii) Line 2 is a better fit

(3) (1.0 pt) Which of the following claims is true?

(a) When choosing a hypothesis $h \in \mathcal{H}$ from a hypothesis class, no matter what optimisation procedure or loss function is used, the same hypothesis will be recovered.

(b) Training a machine learning model selects the best hypothesis class $\mathcal{H}$ among the set of all possible hypothesis classes $\{\mathcal{H}\}$.

(c) Loss function $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is used to approximate empirically the error of the model during learning

(4) (1.0 pt) [*Programming exercise*] Consider the given subset of Breast cancer Wisconsin dataset available on the Assignments course page, and the division to training and test sets given in the code template. With this data, train a linear regression model introduced in the lecture 1 (without the intercept term), and a linearSVM imported from sklearn (with settings as given in the template). What is the difference between the test accuracies of the model, $accuracy(\mathbf{y}_{true}, \mathbf{y}_{SVM}) - accuracy(\mathbf{y}_{true}, \mathbf{y}_{LR})$?

Note: the real-valued predictions from linear regression model should be transformed to class labels in order to compute accuracy; a helper function is provided for this

(a) 0.098

(b) 0.035

(c) 0.102

(d) 0.032

(5) (1.0 pt) [*Programming exercise*] Continuing with the same setting: it is important to catch all the serious cancer cases, while it is not a big deal if some benign cases are predicted as serious. Considering the malignant cases as the ”positive” class, what evaluation metric should be used to find which model is better, and which model outperforms the other with respect to this measure?

(a) Precision is the best evaluation metric, LR performs better than SVM

(b) Precision is the best evaluation metric, SVM performs better than LR

(c) Recall is the best evaluation metric, LR performs better than SVM

(d) Recall is the best evaluation metric, SVM performs better than LR

```python
import numpy as np
from collections import Counter


"""
The data in this exercise is a subset of the much used Breast cancer Wisconsin dataset:
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

Documentation states the class distribution as: 212 - Malignant, 357 - Benign
"""

X = np.load("quiz1_X.npy")
y = np.load("quiz1_y.npy")

n = len(y)
print("There are %d samples in total in the dataset" % n)
print("The shape of X:", X.shape)

print("Unique labels in y:", np.unique(y))
print("Counts of labels in y:", Counter(y))

# shuffle the data and randomly divide it to training and testing
# give the random generator a seed to get reproducable results
np.random.seed(0)
order = np.random.permutation(n)
#-----------------------------------------------------------------------------------------------
# Note! Random number generator between old an new numpy versions might not be the same!
# In this course we use up-to-date versions.
# Check that the generated order should be the same as what is given in the materials:
# order = np.load("quiz1_sample_order.npy")
#-----------------------------------------------------------------------------------------------
tr_samples = order[:int(0.5*n)]
tst_samples = order[int(0.5*n):]
print("The data is divided into %d training and %d test samples" % (len(tr_samples), len(tst_samples)))
Xtr = X[tr_samples, :]
Xtst = X[tst_samples, :]
ytr = y[tr_samples]
ytst = y[tst_samples]

# For exercise 1, use the sklearn's LinearSVM with these settings:
from sklearn.svm import LinearSVC
svm = LinearSVC(dual=False)
# to train the svm, call svm.fit(...) and to predict, call svm.predict(...)
# with suitable arguments in place of ...
# this is a helper function for transforming continuous labels to binary ones
# works with both 0&1 and -1&1 labels
def get_classification_labels_from_regression_predictions(unique_labels, y_pred):
    assert len(unique_labels) == 2  # this function is meant only for binary classification

    meanval = np.mean(unique_labels)

    transformed_predictions = np.zeros(len(y_pred))
    transformed_predictions[y_pred < meanval] = np.min(unique_labels)
    transformed_predictions[y_pred >= meanval] = np.max(unique_labels)

    return transformed_predictions
```