MLSM 2021 - Quiz 6
26/11/2021 - 08/12/2021 at 23:59

The code templates for programming exercises are given at the end of the pdf, or as python files separately at the course page.

Questions based on lecture 10: Feature learning, selection and sparsity

(1) (1.0 pt.) [*Programming exercise*] Consider the given code for classifying the breast cancer Wisconsin data. Which feature transformation technique gives the best accuracy?
   (a) No feature transformation
   (b) Centering
   (c) Standardisation
   (d) Unit range
   (e) Normalization of feature vectors

(2) (1.0 pt.) Which of the following claims is false?
   (a) If data contains highly correlated variables, l_1-norm regularization might result in only one of the corresponding coefficients being non-zero. In other words, if $x_i = x_j$ and $w_i = 1$, $w_j = 0$ is a valid solution for the lasso problem, then any $w_i = \alpha$, $w_j = 1 - \alpha$ with $0 \leq \alpha \leq 1$ is also.
   (b) l_∞-norm is the limit of l_p norm, defined as $l_\infty(x) = \max(|x_1|, |x_2|, ..., |x_d|)$. Like l_0 or l_1-norms, the l_∞ promotes sparsity.
   (c) Lasso model can offer computational gains: when some coefficients w_i are assigned zero weight, the corresponding features x_i can be subsequently ignored in the data during the prediction stage.

Questions based on lecture 11: Multi-class classification

(3) (1.0 pt) Consider the ECOC strategy for classifying the digits (0, 1, 2, ... ,9). What are the minimum and the maximum (without redundant codes) lengths of the codewords that can be defined for it?

(0.5 pt) Minimum length:
   (a) 4
   (b) 5
   (c) 10

(0.5 pt) Maximum length:
   (a) 1023
   (b) 767
   (c) 511

(4) (1.0 pt) [*Programming exercise*] Generate a multi-class classification dataset as in the example code with `make_blobs`. Using the perceptron algorithm from sklearn as the base classifier with its default parameters, apply the OVO, OVA and ECOC multi-class methods to this data. Use the sklearn's implementations imported in the example code; for ECOC use `random_state=42`.

Which multi-class strategy has the highest test accuracy?
   (a) OVO
   (b) OVA
   (c) ECOC

(5) (1.0 pt) [*Programming exercise*] Continuing within the setting of the previous exercise, investigate the effect that the length of the ECOC's codeword has to the ECOC's accuracy. Consider length parameters `np.arange(0.3, 4.1, 0.1)` for the `code_size`; the code book can be obtained from `code_book_` attribute. Fitting a linear regression model (`sklearn.linear_model.LinearRegression` with default settings) to the results (x: code length, y: accuracy), what is the model coefficient w (in `coef_`)? Use accuracies as percentages: scaled to between 0 and 100.

Note: you might encounter an error from linear regression requiring reshaping the data. In this case reshape the input data in vector `x` by `x.reshape(-1, 1)`.

(a) 0.4

(b) 2.0

(c) 6.7

(d) 12.1

Code template for exercise 1

```
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score

"""
More info about the attributes in the dataset:
https://scikit-learn.org/stable/datasets/toy_dataset.html#breast-cancer-wisconsin-diagnostic-dataset
"""

X, y = load_breast_cancer(return_X_y=True)
print("data shapes:", X.shape, y.shape, np.unique(y))

#-------------------------------------------------------------------------

# ....

#-------------------------------------------------------------------------
# linear classification

# divide into training and testing
np.random.seed(42)
order = np.random.permutation(len(y))
tr = np.sort(order[:250])
tst = np.sort(order[250:])

svm = LinearSVC(fit_intercept=False, random_state=2)
svm.fit(X[tr, :], y[tr])
preds = svm.predict(X[tst, :])
print("SVM accuracy:", np.round(100*accuracy_score(y[tst], preds), 1), "%")
```

Code template for exercises 4 and 5

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
# the data
from sklearn.datasets import make_blobs
# linear models
from sklearn.linear_model import Perceptron, LinearRegression
# multi-class models
from sklearn.multiclass import OneVsOneClassifier, OneVsRestClassifier, OutputCodeClassifier



# Create the dataset
C = 4
n = 800
X, y = make_blobs(n, centers=C, random_state=0)

np.random.seed(0)
order = np.random.permutation(n)
tr = order[:int(n/2)]
tst = order[int(n/2):]

Xt = X[tst, :]
yt = y[tst]
X = X[tr, :]
y = y[tr]

# use perceptron with default parameters as the base classifier for the multi-class methods
linear_classifier = Perceptron()

# ....
```