MLSM 2022 - Quiz 6
18/11/2022 - 30/11/2022 at 23:59

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

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

(1) (1.0 pt.) Select the true statements *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

(a) Selecting a subset of features most correlated to the output values is guaranteed to find the most useful subset of features for the supervised learning task.

(b) Backward elimination can be expected to produce better results than forward selection.

(c) If data contains highly correlated variables, l_1-norm regularization on them 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.

(d) 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.

(2) [*Programming exercise*] Consider the code template loading the breast cancer Wisconsin data from sklearn. Investigate various feature transformation techniques: centering, standardisation, unit range and normalisation with l_2-norm as defined one the slides "Examples of feature transformations". Fit Lasso from sklearn (use `tol=1e-3` and `max_iter=1e5`) to the transformed training sets and investigate the accuracies and sparsities obtained with different levels of regularisation. In the sklearn's implementation the regularisation parameter, called λ in the lecture slides, is set with `alpha`.

Note: while feature transformation techniques can be (and often are) applied consecutively, in this question only one feature transformation technique is applied at a time. Also recall good machine learning practices about testing: nothing about the test set should be leaked to the training stage. This means that the feature transformations should be also defined only based on the training data.

Hints: The following numpy functions might be useful (but by no means mandatory): count_nonzero, logspace. If you obtain multiple results with the same level of sparsity, you can consider only the results with maximum accuracy.

(a) (0.5 pt) With which feature transformation technique(s) can you obtain at least 90% accuracy on the test set with fewest selected features? *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

(a) No feature transformation
(b) Centering
(c) Standardisation
(d) Unit range
(e) Normalization of feature vectors

(b) (0.5 pt) With which feature transformation technique(s) can you obtain over 85% accuracy on the test set already with one feature? *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

(a) No feature transformation
(b) Centering
(c) Standardisation
(d) Unit range
(e) Normalization of feature vectors

(3) (1.0 pt.) [*Programming exercise*] Consider the code template sampling the two moons toy dataset from sklearn. Investigate the effect of adding noisy features (as given in the template) to the original data with support vector machine using RBF kernel, and Lasso imported in the template. For all noise amounts, perform 5-fold cross-validation to select the optimal parameters from the lists of parameters given in template

(for all other parameters use the defaults) with a grid search. See slide "n-Fold Cross-Validation" from lecture 4 to recall the procedure; do not shuffle the data here but consider them in order as illustrated there. Use accuracy as the metric. If ties occur during the CV, select the parameters with lowest index.

Hints: The following numpy functions and sklearn modules might be useful (but by no means mandatory): argmax, mean, KFold.

What is the lowest amount of noisy features added to the original data, when Lasso accuracy on test set is better than SVM? _____

Questions based on lecture 11: Multi-class classification

(4) (1.0 pt) Select the true statements *Note: Multiple choices may be correct. Marks are not awarded for partially correct solutions*

(a) Both OVA and OVO can be seen as special cases of ECOC approach
(b) OVA scheme always produces the optimal error rate for a classifier
(c) One-vs-one is more robust to uneven classes than one-vs-all
(d) Comparing classifier scores in one-vs-all scheme is a meaningful way to break the possible ties
(e) One-vs-one model is more prone to overfitting than one-vs-all

(5) Consider the ECOC strategy for classifying cats, dogs, parrots, axolotls, kiwis and pangolins. What are the minimum and the maximum lengths of the codewords? Redundant codes should not be included for maximum length.

(0.5 pt) Minimum length: _____ (0.5 pt) Maximum length: _____

Code template for exercise 1

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import Lasso
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_original, y_original = load_breast_cancer(return_X_y=True)
y_original[y_original==0] = -1
print("data shapes:", X_original.shape, y_original.shape)

# return_X_y=True is an easy option here if you just want to quickly apply ML algorithms
# with return_X_y=False, a pandas dataframe is returned instead
# in this dataframe there is more information about the data, for example the feature names:
bc_pandas_frame = load_breast_cancer(return_X_y=False)
print("\nfeature names:")
for ii in range(X_original.shape[1]):
    print(ii, bc_pandas_frame.feature_names[ii])

# divide into training and testing
np.random.seed(7)
order = np.random.permutation(len(y_original))
tr = np.sort(order[:250])
tst = np.sort(order[250:])
```

Code template for exercise 3

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.linear_model import Lasso
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.datasets import make_moons

n = 300
n_tr = 200
X, y = make_moons(n, shuffle=True, noise=0.2, random_state=112)
y[y==0] = -1

# standardise the data
trmean = np.mean(X[:n_tr, :], axis=0)
trvar = np.var(X[:n_tr, :], axis=0)
X = (X - trmean[np.newaxis, :]) / np.sqrt(trvar)[np.newaxis, :]

# take first n_tr as training, others as test.
Xtr = X[:n_tr, :]
ytr = y[:n_tr]
Xt = X[n_tr:, :]
yt = y[n_tr:]

# inspect the dataset visually
plt.figure()
plt.scatter(Xtr[:, 0], Xtr[:, 1], c=ytr, marker='x')
plt.scatter(Xt[:, 0], Xt[:, 1], c=yt, marker='o')
plt.show()

# the parameters to consider in cross-validation
# order lasso params from largest to smallest so that if ties occur CV will select the one with more sp
params_lasso = [1, 1e-1, 1e-2, 1e-3, 1e-4]
params_rbf = [1e-3, 1e-2, 1e-1, 1, 10, 100]
params_svm = [1e-3, 1e-2, 1e-1, 1, 10, 100, 1000]  # smaller C: more regularisation

# the noisy features to be considered; for n noisy features, add the first n columns
n_tot_noisy_feats = 50
np.random.seed(92)
X_noise = 2*np.random.randn(X.shape[0], n_tot_noisy_feats)
Xtr_noise = X_noise[:n_tr, :]
Xt_noise = X_noise[n_tr:, :]
```