MLCC Laboratory 3: Dimensionality reduction, feature selection

In this laboratory we will address the problem of data analysis and dimensionality reduction with a reference to a classification problem.

Think hard before you call the instructors or you look at the solution file!

1 Warm up - data generation

You will generate a training and a test set of `d`-dimensional points (`n` points for each class), with `n = 100` and `d = 30`. Only two of those dimensions will be meaningful, the other one will be a variable we will modify.

- **1.A** For each point, the first two variables will be generated by `MixGauss`, extracted from two gaussian distributions with centroids `(0, 0)` and `(1,1)` and standard deviation `0.7` (the first one with labels `1`, the second with label `-1`):

```
Xtr, Ytr = MixGauss(means=[[0, 0], [1, 1]], sigmas=[[0.7], [0.7]], n=100)
Ytr = 2*np.mod(Ytr, 2)-1
Xts, Yts = MixGauss(means=[[0, 0], [1, 1]], sigmas=[[0.7], [0.7]], n=100)
Yts = 2*np.mod(Yts, 2)-1
```

- **1.B.** You may want to plot the relevant variables of the data:

```
fig, axs = plt.subplots(1, 1)
plt.scatter(Xtr[:, 0], Xtr[:, 1], s=30, c=np.squeeze(Ytr), alpha=0.8)
plt.scatter(Xts[:, 0], Xts[:, 1], s=30, c=np.squeeze(Ytr), alpha=0.1)
plt.title('train and test datasets')
```

- **1.C** The remaining variables will be generated as gaussian noise:

```
sigma_noise = 0.01
Xtr_noise = sigma_noise * np.random.randn(2*n, d-2)
Xts_noise = sigma_noise * np.random.randn(2*n, d-2)
```

  To compose the final data matrix, run:

```
Xtr = np.concatenate((Xtr, Xtr_noise), axis=1)
Xts = np.concatenate((Xts, Xts_noise), axis=1)
```

2 Principal Component Analysis (PCA)

- **2.A** Compute the data principal components (see "`help(PCA)`").
- **2.B** Plot the first two components of `X_proj` using the following line:

```
fig, axs = plt.subplots(1, 1)
plt.scatter(X_proj[:, 0], X_proj[:, 1], s=30, c=np.squeeze(Ytr), alpha
    =0.8)
plt.title('train dataset projected on first 2 components')
```

- **2.C** Plot the cummulative sum of the eigenvalues you found.

  Reason on the meaning of the results you are obtaining.

- **2.D** Display the `sqrt` of the first `10` eigenvalues and plot the coefficients (eigenvector) associated with the largest eigenvalue:

```
print(np.sqrt(D[:10]))
fig, axs = plt.subplots(1, 1)
plt.scatter(range(d), V[:, 0], s=30, alpha=0.8)
plt.title('Eigenvector of highest eigenvalue')
```

- **2.E** Repeat the above steps with dataset generated using different `sigma_noise = 0, 0.01, 0.1, 0.5, 0.7, 1, 1.2, 1.4, 1.6, 2`. To what extent data visualization by PCA is affected by the noise?

3 Variable selection

- **3.A** Use the data generated in part 1. Standardize the data matrix, so that each column has mean `0` and standard deviation `1`:

```
m = np.mean(Xtr, axis=0)
s = np.std(Xtr, axis=0)
Xtr = (Xtr - m) / s
```

  Do the same for `Xts`, by using `m` and `s` computed on `Xtr`.

- **3.B** Use the orthogonal matching pursuit algorithm (type "`help(OMatchingPursuit)`").

- **3.C** You may want to check the predicted labels on the training set:

```
Ypred = np.sign(Xts.dot(w))
error = calcErr(Yts, Ypred)
```

  and plot the coefficients `w` with `scatter(range(d), abs(w))`. How the error changes with the number of iterations of the method?

- **3.D** By using the method `holdoutCVOMP` find the best number of iterations with `intIter = 2,...,d` (and, for instance, `perc = 0.75, nrip = 20`).

  Moreover, plot the training and validation error with the following lines:

```
fig, axs = plt.subplots(1, 1)
plt.plot(intIter, Tm)
plt.plot(intIter, Vm)
plt.legend(['Training error', 'Validation error'])
plt.xlabel('number of iterations for OMP')
plt.ylabel('error')
```

What is the behavior of the training and the validation errors with respect to the number of iterations?

- **3.E** Try to increase the number of relevant variables `d = 3,5,...` (and the corresponding standard deviation of the Gaussians) around the centroids:

```
np.ones((d, 1)) # vector of all 1s
# and
-np.ones((d, 1)) # vector of all -1s
```

and see how this change is reflected in the cross-validation.

4 If you have time - more experiments

- **4.A** Analyse the results you obtain on sections 2 and 3 once you choose:
  - $\mathtt{n} \gg \mathtt{d}$
  - $\mathtt{n} \approx \mathtt{d}$
  - $\mathtt{n} \ll \mathtt{d}$,

  and evaluate the benefits of the two different analyses.

- **4.B** Dimensionality reduction is often used as a pre-procesing step to a learning (classification) algorithm. The idea is to perform classificatiton in a lower dimensional space and therefore safe computational time. You have the following task:
  - Generate a new training and test datasets as in Laboratory 1.
  - Perform dimensionality reduction on the training set.
  - Using the projection you just found, project the test set.
  - Perform kNN in the lower dimensional (projected) space. Compare the result (both accuracy and running time) with the one in Laboratory 1.