COMP3223: Coursework

1 Introduction

1.1 General remarks

The exercises are meant to help you understand the course material in depth. You will find that using tools such as `scikit-learn`, which are there for you to use in practical contexts, will make the task of completing your coursework much simpler. However, for developing a thorough understanding of those implementations you will need to explore how the data gets transformed using mathematical methods, and you need to implement the algorithms yourself. You may use the `scikit-learn` libraries to check your implementations if you wish. However, for many of the data preparation steps such as loading, splitting data into training and test tests randomly and so on, you can use canned routines such as those in `scikit-learn`. For some tasks I have also given you pieces of code that you can reuse, and you can repurpose the `jupyter` lab notebooks as well.

There are umpteen references on the web you may wish to consult. However, for most of the exercises [1], [2] and [3] should give you food for thought and influence the way you write your report.

1.2 Submission guidelines

The report should focus on the key conceptual steps that underpin each algorithm, illustrating your understanding by pointing to the results you obtain that should be summarised in graphical or tabular form. Try to present the results as answers to specific questions that you have posed. Some of these are asked for explicitly in the tasks below, others are left to your judgement. All else being the same, a well articulated report that shows evidence of depth

of understanding will get a higher mark than one where the same results are described, but less cogently. The Handin page has the following guide:

Learning Outcomes

1. A1 Underlying mathematical principles from probability, linear algebra and optimisation
2. F1 Characterise data in terms of explanatory models
3. D1 Systematically work with data to learn new patterns or concepts
4. D2 Gain facility in working with algorithms to handle data sets in a scientific computing environment

Marking Scheme

Criterion	Description	Outcomes	Marks
Display depth of understanding	Explain the rationale behind mathematical formulations that bring out patterns in data	1,2	33
Facility with data transformation	Extract statistics from data in informative numerical and visual summaries	1,3,4	33
Clarity of presentation	Express the key ideas systematically, backed up by evidence from data analysis	1,2,3	33

The report should be between 6 and 8 pages, but if you can squeeze the information into less than 6 pages, that would be perfect.

2 Linear Regression with non-linear functions

In this exercise you have to perform the task of fitting polynomial functions to $y(x) = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma)$ is a noise term and the function $f(x)$ is

$$f(x) = e^{-x/2}\sin(\pi x) + \sin\left(\frac{3\pi x}{2}\right).$$

You have to generate a number N of points (work with fewer than 30 points for training) chosen randomly from $-2 \leqslant x \leqslant 2$. Your training and test sets will be taken from these (x_n, y_n) pairs, for $n = 1, \ldots, N$.

```
rn = np.random.uniform(0, 1, n)
x = np.reshape(np.sort(rn, axis=0), (n, 1))
```

The task is to model each (input, target) pair (t_n, x_n) by a linear combination of basis functions:

$$y(x; \boldsymbol{w}) = \sum_{j=0}^{p} w_j \phi_j(x).$$

You will take **two** classes of functions $\phi_j(x)$ for your basis elements to fit the training and test data sets that you generate:

1. polynomials: $\phi_j(x_n) = x_n^j$ where x_n is an input value.

2. Gaussian radial basis functions (RBF): for some choices of *centres* x_j,

$$\Phi(x; x_j) = \exp\left(-\frac{1}{2}\left(\frac{x - x_j}{s}\right)^2\right).$$

Construct a design matrix $\mathbf{A}$ by concatenating to a column of ones p columns $\phi_1(x_n), \ldots, \phi_p(x_n)$. The column entries are $\phi_j(x_n)$, $n = 1, \ldots, N$.

```
def gaussian_basis_fn(x, mu, sigma=0.1):
    return np.exp(-0.5 * (x - mu) ** 2 / sigma ** 2)

def polynomial_basis_fn(x, degree):
    return x ** degree

 def make_design(x, basisfn, basisfn_locs=None):
    if basisfn_locs is None:
        return np.concatenate([np.ones(x.shape), basisfn(x)], axis=1)
    else:
        return np.concatenate([np.ones(x.shape)] + \
        [basisfn(x, loc) for loc in basisfn_locs], axis=1)
```

The weights that minimise the loss function

$$\sum_{n=1}^{N}\left(y_n - w_0 - \sum_{j=1}^{p} w_j \phi_j(x_n)\right)^2 + \lambda\|\boldsymbol{w}\|_2^2$$

are given by the analytical expression

$$\boldsymbol{w} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbb{I}_{p+1})^{-1}\mathbf{A}^T\mathbf{y}, \tag{1}$$

where $\mathbb{I}_m$ stands for a $m \times m$ identity matrix.

2.1 Tasks

This section gets you to explore the generalisation performance of models learned by linear regression. You have to vary the size of the basis function set p, the locations of the centres and the regularisation parameter λ to evaluate the nature of the fit to the training and test data. To create your training and test sets, you could call

```
from sklearn.model_selection import train_test_split
```

and look up the function description of `train_test_split`. Read the paragraph starting at the bottom of page 149 of Bishop [2] until page 151 for a discussion of how to work with training and test data to explore the generalisation performance of your models.

- *For this data set, set up the evaluation so that you can construct the equivalent of Tables 1.1 and 1.2 and Figures 1.4 to 1.8 and 3.5 of Bishop [2]. This is what you must submit, with brief commentary.*
- Each of your figures must have its axes labelled, and a detailed caption provided so that it is possible to understand the message conveyed by the graph. Your tables, too, must have their own explanatory captions.

3 Classification

You will explore two methods for some toy datasets in order to get familiar with some of the issues to do with projections and data transformations. You will generate one of the two datasets yourself using a random number generator. The other is an old classic – Ronald Fisher's Iris dataset that can be loaded using `scikit-learn`'s data loader.

You will first explore Fisher's LDA for binary classification for class labels a and b. In Fisher's method a direction defined by vector $\boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ is chosen so that the data $\boldsymbol{x}^n$ projected on to $\boldsymbol{w}$ maximises the separation between the a and b type distributions. (n is a data index, ranging from 1 to n_a for class a, and from 1 to n_b for class b, here used in superscript; it is not the n-th power of $\boldsymbol{x}$.) Setting $y_c^n \triangleq \boldsymbol{w} \cdot \boldsymbol{x}_c^n$ for class label $c \in \{a, b\}$ the scalar means and standard deviations of the projected data become

$$\mu_c = \frac{1}{n_c} \sum_{n=1}^{n_c} y_c^n, \ \sigma_c^2 = \frac{1}{n_c} \sum_{n=1}^{n_c} (y_c^n - \mu_c)^2, \ c \in \{a, b\}.$$

The direction $\boldsymbol{w}$ is chosen in order to maximise the Fisher ratio $F(\boldsymbol{w})$: Fisher ratio

$$F(\boldsymbol{w}) \triangleq \frac{(\mu_a - \mu_b)^2}{\frac{n_a}{n_a + n_b}\sigma_a^2 + \frac{n_b}{n_a + n_b}\sigma_b^2}.$$

3.1 Data 1: separate 2 Gaussians

You will generate your own data for this part. Generate data from two 2-dimensional Gaussian distributions

$$\mathbf{x}_a \sim \mathcal{N}(\mathbf{x}|\mathbf{m}_a, \mathbf{S}_a),\ \mathbf{x}_b \sim \mathcal{N}(\mathbf{x}|\mathbf{m}_b, \mathbf{S}_b)$$

where $\mathbf{m}_a, \mathbf{m}_b$ are the (2×1) mean vectors and $\mathbf{S}_a$ and $\mathbf{S}_b$ are the (2×2) covariance matrices that define normal distributions. Let the number of data points from each type a, b be n_a and n_b. For reference, you should read Section 5.2 of FCML [1], Section 4.2 of Bishop [2], Section 4.4 of [4] or Section 4.3 of [3] for guidance and ideas.

Note: *If the classes are widely separated or if they overlap significantly it will be hard for you to explore and demonstrate the consequences of choosing different directions, and thus learn from this exercise. Hence, make the differences of the means of the same order of magnitude as the standard deviations. The precise values will affect the results of the following tasks and you can experiment with numerical values that help you appreciate the pros and cons of the classification method. This will determine the evidence you provide in the report that demonstrate your insights on the nature of the learning algorithm.*

Task: This part is for visual exploration of the consequences of projecting data onto a lower dimension. Your report must present figures with labelled axes and explanatory captions.

1. Make *two* illustrative choices for the direction $\boldsymbol{w}$ and plot the histograms of the values y_a^n and y_b^n. You should make choices that make a difference to the nature of the resulting histograms.

2. Alter the values of $\mathbf{S}_a$ and $\mathbf{S}_b$ to show how the histograms in the previous task change. Choose $2-4$ matrices to visually demonstrate how class separation is altered.

3. Plot the dependence of the Fisher ratio $F(\boldsymbol{w})$ on the direction of $\boldsymbol{w}$ by rotating some random starting weight vector $\boldsymbol{w}(0) = (w_1, w_2)^T$ and rotating it by angles θ to get $\boldsymbol{w}(\theta) = R(\theta)\boldsymbol{w}(0)$ where

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

Find the maximum value of $F(\boldsymbol{w}(\theta))$ and find the corresponding direction $\boldsymbol{w}^*$ for the optimal projection:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, F(\boldsymbol{w}) = \underset{\theta}{\operatorname{argmax}}\, F(\boldsymbol{w}(\theta)).$$

3.2 Data 2: Iris data

In this section you will perform the same LDA task on the famous Iris dataset `https://en.wikipedia.org/wiki/Iris_flower_data_set` which can be downloaded from `http://archive.ics.uci.edu/ml/datasets/Iris`. While the previous exercise was done by finding the weight vector to project on by direct search, here you follow Fisher and solve the generalised eigenvalue condition for optimal weights. For an introduction to the method, read Section 4.1.4 of Bishop [2].

With more features and classes, you will need to compute the between-class and within-class covariance matrices Σ_B and Σ_W:

$$\Sigma_B = \sum_c \frac{n_c}{N}(\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^T, \text{where } \boldsymbol{\mu} \text{ is the mean of class means,}$$

and Σ_W is the sum of the covariance matrices for each class. n_c is the number of data samples in class c and $N = \sum_c n_c$. In case there are different numbers of training data points from each class, you have to scale any class dependence by the corresponding fraction of class members in the population. (In the Iris data set all three classes have 50 members, so you can skip this step.)

1. Find the optimal direction $\boldsymbol{w}^*$ for projecting the data onto. You will need to solve the generalised eigenvalue problem $\Sigma_B \boldsymbol{w} = \lambda \Sigma_W \boldsymbol{w}$. Section 16.2, 16.3 of [5] and eq. (4.15) of [3] has further details.

   Suggestions: *The standard libraries in scipy (and others) can give you the generalised eigenvalues and eigenvectors. Since the covariance matrix is symmetric, the function you should call in* `numpy/scipy` *is* `eigh`*, and not* `eig`*, although for problems of this scale it won't make a difference and you can eyeball the eigenvalues. In particular, the eigenvalues returned by* `eigh` *are* sorted. You must also check the answer provided by verifying that the generalised eigenvalue condition $\Sigma_B \boldsymbol{w} = \lambda \Sigma_W \boldsymbol{w}$ holds. This will clarify the notational conventions of the software used. Sometimes it is the transpose of the returned matrix of vectors that contains the eigenvectors, so please make sure you understand what is being returned. *You should also discover that the rank of Σ_B is limited by the number of classes.*

2. Draw a histogram of the projections along each of the eigenvectors of the data from each class. Comment on the nature of the class overlap.

3. Take the dot product of each Iris data point with the two eigenvectors corresponding to the two largest generalised eigenvalues. These are the new co-ordinates for the data. Display a 2-D scatter-plot of these projected points, colour-coded by class label.

4. Perform softmax regression on the same dataset to classify the points.

5. Make sure you divide the data into training and test sets and compare and contrast the results from the two methods used on this dataset.

References

[1] Simon Rogers and Mark Girolami. *A First Course in Machine Learning, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2016.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2nd edition, 2009.

[4] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[5] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 04-2011 edition, 2011.