COMP6245: Coursework

1 Due date

The hand-in date for the assignment is **Monday, December 3, 2018** .

2 Introduction

You should implement all the exercises for yourself. Tools such as `scikit-learn` are there for you to use in practical contexts, but for understanding the underpinnings of many of those implementations you need to work them out for yourself. 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`.

A useful blog for many of the machine learning algorithms you will encounter in the module is at `https://jermwatt.github.io/mlrefined/index.html` based on the book [5]. There are umpteen references on the web you may wish to consult. However, for most of the exercises [2] and [3] should give you food for thought and influence the way you write your report.

3 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. Some of these are asked for explicitly in the questions below, others are left to your judgement. All else being the same, a well articulated report will get a higher mark than one where the same results are described, but less cogently. There are page limits signposted in some of the sections and the other sections should merit a

similar allocation of space. These limits are not going to be policed rigorously but are intended as a guide for you to gauge how much is too much or too little. There will be a handin page for you to submit a report (mandatory) and code files (optional, but recommended, to address ambiguities in the report).

4 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. (Just as you loaded the `diabetes` dataset.)

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

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

4.1 Data 1: separate 2 Gaussians

The data for this part are to be generated by you. Generate data from two 2-dimensional Gaussian distributions

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

where $\boldsymbol{m}_a, \boldsymbol{m}_b$ are the (2×1) mean vectors and $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ 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. 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 the experience. Hence, make the differences of the means of the order of magnitude of 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.

The following tasks are broken up into two chunks to demarcate bow the marks are to be allotted, even though they are all connected. *You will write up your observations in no more than 2 pages, providing evidence based on your experiments, and discuss what you have learned.* You should read Section 4.4 of [4] (or Section 4.3 of [3]) for guidance and ideas.

1. This part is for visual exploration of the consequences of projecting data onto a lower dimension.

   [3 marks]

   (a) Make a few (between $2-4$) illustrative choices for the direction $\boldsymbol{w}$ and plot the histograms of the values y_a^n and y_b^n.

   (b) Plot the dependence of $F(\boldsymbol{w})$ 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

   The expression for F was provided in the introduction to the Classification section.

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

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

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

2. This part makes you look at probability distributions visually by drawing contour plots and/or histograms.

   [4 marks]

   (a) Since the generating distributions for the classes are known, plot the equi-probable contour lines for each class and draw the direction of the optimal choice vector $\boldsymbol{w}$.

(b) Use Bayes' theorem to write down the logarithm of the ratio of conditional probabilities (called log-odds)

$$\ln\left(\frac{P(c=a|\boldsymbol{x}^n)}{P(c=b|\boldsymbol{x}^n)}\right)$$

and plot the decision boundary where this quantity vanishes. In other words, for each point $\boldsymbol{x} \in \mathbb{R}^2$ check whether the log-odds is positive, negative or zero. The decision boundary is the set of points where the log-odds vanishes.

Do this for the two cases $\boldsymbol{S}_a = \boldsymbol{S}_b$ and $\boldsymbol{S}_a \neq \boldsymbol{S}_b$.

(c) Explore the consequences of using a formula for the discriminant

Contrast with the Fisher ratio introduced above.

$$F_{\text{unbalanced}}(\boldsymbol{w}) \triangleq \frac{(\mu_a - \mu_b)^2}{\sigma_a^2 + \sigma_b^2}$$

that does not account for the different fractions of data in each class. How is this accounted for in the application of Bayes' rule?

4.2 Data 2: Iris data

[5 marks]

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.

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

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

Correction: $\frac{N_c}{N}$ was missing.

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

   NB: *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* $(\Sigma_B - \lambda \Sigma_W) w = 0$ *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.*

   Rank condition

2. Display the histograms of the three classes in the reduced dimensional space defined by w^*.

3. What would happen if, instead of choosing the generalised eigenvector w^* you project on a different vector $w = w^* + a$? You should consider a to be constructed out of the other generalised eigenvectors (why?).

   Verify optimality condition holds for w^*.

4. Present your results with reflections and evidence in no more than 2 pages. You need to discuss the relation between the class separation task and the generalised eigenvector formulation taking into account the question in the last part. You should comment on how this method compares with the method in the 2-gaussian separation exercise above.

5 Linear Regression with non-linear functions

[4 marks]

In this exercise you have to perform the task of fitting polynomial functions to $y(x) = \sin(x) + \epsilon$ where ϵ is a noise term. You have to generating a number N of points for $0 \leqslant x < 2\pi$ and a small noise term.

5.1 Performing linear regression

Fit polynomial functions ϕ of different degrees of your choice to this data.

1. Learn the weights $\boldsymbol{w} = (w_0, \ldots, w_p)$ by minimising the loss function

$$\sum_{n=1}^{N} \left(y^n - w_0 - \sum_{j=1}^{p} w_j \phi_j(x^n) \right)^2 + \lambda C(\boldsymbol{w})$$

   by gradient descent. $C(\boldsymbol{w})$ is a penalty on the norm of the weights. Choose $C(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$, the L_2 norm penalty.

2. For the L_2 norm penalty, obtain the weights from the analytical expression

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

   where $\mathbb{I}_{p+1}$ stands for a $(p+1) \times (p+1)$ identity matrix.

3. Split up your data into a training $\mathcal{S}^{tr}$ and test set $\mathcal{S}^{ts}$ and measure the mean of the squared residuals on the test set $\mathcal{S}^{ts}$ as a performance metric for each model. Plot this measure as a function of:

   (a) the complexity of the model. This could be the number of basis components ϕ_j (and therefore, the number of weights p or the degree of the polynomial);

   (b) the strength of the regularisation coefficient λ.

5.2 How does linear regression generalise?

[4 marks]

This section gets you to explore the models learned by linear regression, by looking at the dependence of the optimal weights on training data and the scale of regularisation. When writing up the results of this section about the ability of the models to generalise using the tasks below, try to relate the weights from eq.(1) to your analysis.

1. Split up your data into 10 (roughly) equal parts for 10-fold cross-validation. Evaluate the models learned on each of the 9/10 of the N data points on the remaining $N/10$. The results of this evaluation relates to the generalisation performance of the model.

References

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

[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] Jeremy Watt, Reza Borhani, and Aggelos K. Katsaggelos. *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge University Press, 2016.