Imperial College London

Department of Mathematics
MATH60026 - MATH70026
Methods for Data Science
Years 3/4/5

Coursework 1 – Supervised learning

In this coursework, you will work with two different datasets of high-dimensional samples:

- **an engineering dataset measuring various mechanical properties of airfoils**
- **a medical dataset characterising risk of diabetes**

You will perform a **regression task** with the former, and a **binary classification task** with the latter.

Task 1: Regression (50 marks)

dataset: Your first task deals with an **engineering dataset**. It contains airfoils with various mechanical descriptors (the data features), and each set of descriptors is associated with a certain sound pressure level. Each sample in the dataset (rows) corresponds to an airfoil characterised by 7 mechanical descriptors (like chord length, thickness etc., each of them measured with appropriate units of measure, see the columns). We will consider the sound pressure level (column 'Sound Pressure') as the target variable to regress, while the other 6 variables are our predictors.

- This engineering dataset is made available to you on Blackboard as `airfoil_noise_samples.csv.`
- We also provide on Blackboard a test set in the file `airfoil_noise_test.csv.`

Questions:

1.1 Linear regression (8 marks)

1.1.1 - Starting from the dataset `airfoil_noise_samples.csv`, obtain a linear regression model to predict the 'Sound Pressure' as your target variable, using all the other features as predictors. To train the model on `airfoil_noise_samples.csv`, consider the loss function defined by:

$$L(\beta) = \frac{1}{2N}||\mathbf{y} - \mathbf{X}\beta - \beta_0||^2$$

where N is the size of the training set. Report the inferred values of the model parameters and the in-sample mean squared errors (MSE) and R^2 score for the dataset.

1.1.2 - Apply the model to the test data `airfoil_noise_test.csv` to predict the target variable, and compute the out-of-sample R^2 score and MSE on this test set. Compare the out-of-sample and the in-sample R^2 score and MSE, and explain your findings.

1.2 Lasso regression (12 marks)

1.2.1 - Using again the dataset `airfoil_noise_samples.csv`, repeat task **1.1.1** employing Lasso regression, i.e. by minimising the loss function:

$$L_{LASSO}(\beta) = \frac{1}{2N}||\mathbf{y} - \mathbf{X}\beta - \beta_0||^2 + \lambda||\beta||_1$$

via gradient descent with a learning rate that decreases during iterations, $l_r = 1/N_{iterations}$. Employ a 5-fold cross-validation to tune the Lasso penalty hyper-parameter λ. Using the average MSE over all folds, demonstrate with plots how you scan the penalty hyper-parameter λ to find its optimal value and report it explicitly. (Note that the regularisation does not apply to the intercept β_0).

1.2.2 - Choose one training/validation split and visualise the inferred parameters β as a function of the Lasso penalty, and explain their trend based on your knowledge of the bias-variance trade-off. Comment also on the effect of the Lasso penalty on the inferred parameters β.

1.2.3 - Fix the penalty hyper-parameter to the optimal value found in **1.2.1** and retrain the model on the entire dataset `airfoil_noise_samples.csv`. Obtain the in-sample MSE and R^2 score when applied to `airfoil_noise_samples.csv`, and compare it to the out-of-sample MSE and R^2 score on the test set `airfoil_noise_test.csv`. Use some of your results and plots to discuss the differences between the case $\lambda = 0$ and the optimal value of λ.

1.3 Elastic Nets (20 marks)

1.3.1 - Using the dataset `airfoil_noise_samples.csv` and $l_r = 1/N_{iterations}$, train a linear regression model implementing a regularisation that combines Lasso and Ridge penalties, called 'elastic net' linear regression. The cost function to be optimised is given by:

$$L_{EN}(\beta) = \frac{1}{2N}||\mathbf{y} - \mathbf{X}\beta - \beta_0||^2 + \lambda[\alpha||\beta||_1 + (1-\alpha)||\beta||^2]$$

where λ is the strength of the regularisation and $\alpha \in [0, 1]$ is a coefficient that controls the relative importance of the Ridge and Lasso terms.

1.3.2 - Conduct a grid search to find the optimal penalty hyper-parameter λ of the elastic net. Use for this a 5-fold cross-validation and fix α to 3 different values: 0.1, 0.5, and 0.9. For each value of α, repeat the grid search, re-train the model with the optimal λ on the full training set and report the out-of-sample MSE and R^2 score evaluated on the test set `airfoil_noise_test.csv`. Based on these metrics, report which value of α provides the best model.

1.3.3 - Visualise the inferred parameters with the optimal λ for each value of α, and comment explicitly on the cases $\alpha = 0$, $\alpha = 0.5$ and $\alpha = 1$.

1.4 kNN regression (10 marks)

1.4.1 - Train a k-Nearest Neighbour (kNN) regression model on the dataset `airfoil_noise_samples.csv`. Demonstrate that you have used a grid search with 5-fold cross-validation to find an optimal value of the hyper-parameter *k.*

1.4.2 - Fix the optimal *k* and retrain the model on the entire dataset `airfoil_noise_samples.csv`. Use the out-of-sample MSE on the test set `airfoil_noise_test.csv` to compare the performance of your kNN model to the performance of linear regression without and with regularisations (i.e., from tasks **1.1, 1.2, 1.3**). From this comparison, what can you conclude about the relationship between predictors and outcomes?

Task 2: Classification (50 marks)

dataset: Your second task deals with the classification of diabetes diagnosis based on patient data on a set of clinically relevant features. The column 'diabetes' corresponds to the diabetes diagnosis classification. The other 14 columns correspond to the patient's features.

- The dataset is available on Blackboard under file `diabetes_samples.csv`.
- The test set is in the file `diabetes_test.csv`.

Questions:

2.1 Random forest (20 marks)

2.1.1 - Train a random forest classifier on the dataset `diabetes_samples.csv` employing cross-entropy as your information criterion for the splits in the decision trees. Demonstrate that you have performed a grid search with 4-fold cross-validation to explore and optimise over suitable ranges the following hyper-parameters: (i) number of decision trees; (ii) depth of decision trees. Use *accuracy* as the measure of performance for this hyper-parameter optimisation.

2.1.2 - Re-train your optimal random forest classifier on the full dataset `diabetes_samples.csv` and compare its performance on `diabetes_samples.csv` to the performance on the test data `diabetes_test.csv`, using different measures computed from the confusion matrix, in particular commenting on accuracy, precision and F-score.

2.1.3 - Demonstrate that the dataset `diabetes_samples.csv` is unbalanced by computing the frequencies of diagnosis outcomes. Introduce appropriate weights for each data point that balance the diagnosis outcomes. Repeat tasks **2.1.1** and **2.1.2**, but now train the random forest using the appropriate weights in the bootstrap step. Using the ROC curve and the Precision-Recall curve, compare and discuss the performance of random forest classifiers with standard bootstrap (tasks **2.1.1** and **2.1.2**) versus the weighted bootstrap.

2.2 Support Vector Machine (SVM) (30 marks)

2.2.1 - Train a soft-margin *linear* SVM classifier by minimising the loss function:

$$\frac{1}{2}||\mathbf{w}||^2 + \lambda \sum_{i=1}^{N} \max\left\{0, 1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b)\right\}$$

on the dataset `diabetes_samples.csv`, using a 4-fold cross-validation to optimise the hardness hyper-parameter λ, which controls the boundary violation penalty. Use accuracy as a measure of performance for this optimisation. Display the average accuracy of the SVM classifiers as λ is varied, and discuss the limits of low hardness and high hardness.

2.2.2 - For each value of the hardness hyper-parameter you have examined in **2.2.1** you have found a hyperplane. Calculate the cosine of the angle between each pair of hyperplanes and report your results on a square heatmap as a function of the hardness hyper-parameters λ of each hyperplane.

2.2.3 - Fix the hardness hyper-parameter λ to its optimal value, re-train the linear SVM on the full dataset `diabetes_samples.csv` and evaluate its performance on the test data `diabetes_test.csv`, using accuracy, precision and F-score.

2.2.4 - Implement a soft-margin *kernelised* SVM classifier by minimising the loss function:

$$\frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u} + \lambda \sum_{i=1}^{N} \max\left\{0, 1 - y^{(i)}(\mathbf{K}^{(i)}\mathbf{u} + b)\right\}$$

with respect to $\mathbf{u}$ and b. The kernel is the *sigmoid* kernel:

$$[\mathbf{K}]_{ij} \equiv k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \tanh\left(\sigma(\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}) + 1\right)$$

Fix the hardness hyper-parameter to $\lambda = 10$ and train the soft margin kernelised SVM classifier on the dataset `diabetes_samples.csv`, considering the values of the coefficient $\sigma = 0.01, 0.1, 1, 2$. Report the inferred value of the intercept b for each value of σ.

2.2.5 - Evaluate accuracy, precision and F-score of the kernelised SVM classifier from **2.2.4** on the test set `diabetes_test.csv` for each value of σ. Compare these results with the linear SVM from **2.2.3**, commenting on: the conclusions you can draw on the distribution of the data; the hyper-parameters that should be optimised to further improve the performance of the kernelised SVM.

Task 3: Mastery component (25 marks)

This task is to be completed **only** by MSci (4th year) and MSc students.

3.1 Bias and variance in linear regression (12 marks)

Consider the linear model $f(x) = \beta x + \epsilon$ with parameters $\beta = 0.5$ and $\epsilon \sim \mathrm{Normal}(0, 1)$.

3.1.1 - Create a synthetic dataset $D_N = (x_i, f(x_i))_{i=1,2,\ldots,N}$ by sampling $N = 100$ times from the linear model with $x_i \sim \mathrm{Normal}(0, 1)$ (we shall call it D_{100}). Produce a scatter plot of the data D_{100} and perform linear regression to infer from this synthetic dataset an estimate of the parameter β.

3.1.2 - Evaluate bias and variance of the linear regression estimator of β relative to the true value 0.5 by generating 100 samples of D_{100}. Comment on the implications of bias and variance for the MSE of the estimator $\hat{f}$ of f.

3.1.3 - Repeat the analysis of task **3.1.2** for increasing sample sizes $D_{1,000}$, $D_{10,000}$ and $D_{100,000}$ and visualise bias and variance as a function of the sample size. Comment on your findings in connection with the Central Limit Theorem.

3.2 Comparison between Logistic Regression and Naive Bayes (13 marks)

3.2.1 - Consider again the classification task implemented in Task 2 on the diabetes dataset. Implement a Logistic Regression classifier and a Naive Bayes classifier to perform the same task, training the two models on the dataset `diabetes_samples.csv` and evaluating the out-of-sample accuracy on the test set `diabetes_test.csv`. For the Naive Bayes, you can assume that the features are normally distributed. How do the two methods compare? Comment on your results.

3.2.2 - Draw subsamples of the training dataset `diabetes_samples.csv` via bootstrapping with replacement, where the size M' of the bootstrapped samples is smaller than the size M of `diabetes_samples.csv`. Choose different values of M', and for each M' repeat the bootstrapping and the training of the two methods on the bootstrapped samples 10 times. Report the out-of-sample accuracy averaged over the 10 trainings as a function of M' (up to $M' = M$) for both methods on the same plot. Comment on: which method performs better at various M'; the sample size regime at which each method stops providing an accurate and reliable out-of-sample performance.

3.2.3 - As seen above, the dataset `diabetes_samples.csv` is unbalanced. Consider again the type of weights that balance the diagnosis outcomes (see task **2.1.3**) and use them to obtain a weighted loss function for the logistic regression classifier. Re-train the logistic classifier with this weighted loss for all the bootstrapped samples from task **3.2.2** and compare its average out-of-sample performance as a function of M' (up to $M' = M$) to the one obtained without weights in task **3.2.2.** Visualise appropriately and comment on the results of this comparison.