REPORT **Date** November 21, 2017 **Reference number**

Created by
Manuel Osvaldo Olguín

Final Assignment

DD3431 Machine Learning (PhD variant)

This final report for the DD3431 Machine Learning course describes the application of Linear Support Vector Machines to a multi-class, multi-output classification problem modeling the resource allocation strategies for basestations in a 5G environment. The classifier was adapted to the multi-class nature of the output.

1 Theoretical description of the problem

1.1 Description of the data

The data used for this study corresponds to samples of the resource allocation for a communication system composed of exactly three basestations and up to a maximum of 20 user devices. The area of interest studied within this communication scenario is modeled as a map; more specifically, it represents an area of 2400 m^2 divided into a grid of 2m×2m cells. This grid is encoded as a matrix, with values between 0 and 2 indicating the occupancy status of each cell:

- 0 for empty cells,
- 1 for cells occupied by a user device,
- 2 for cells occupied by a basestation.

The occupancy data for each user could be either perfect or have an inaccuracy defined by a normal distribution with three possible standard deviations: 0.1, 0.25 or 0.4. Thus, the model had to be trained a total of four times, one for each variation of the positioning inaccuracy (we'll call them "scenarios").

The input data for the learning algorithm corresponds then to the vectorized form of this matrix, i.e. a vector of 600[1] elements, each with a value $v_i \in \{0, 1, 2\}$.

On the other hand, the output data for the learning algorithm corresponds to encoded variables representing the basestation–user device association and the resource allocation information for each sample. The exact details of this encoding and the information carried within escape the scope of this report, but it is of importance to note that:

[1] $\frac{2400}{(2\times2)} = 600$

- there are 9 of these output variables per input sample (3 for each basestation in the model);
- these output variables are mutually independent;
- they have discrete integer values that range between 0 and 12 288;
- these values can be repeated across variables;
- finally, the order in which these values appear in the output is relevant (i.e. permutations of the same values correspond to different output classes!).

In summary, the structure of an arbitrary sample looks like the following:

$$\underbrace{0, 1, 0, 0, 2, 0, \ldots 0, 0, 0, 0, 0, 1}_{\text{600 input features}}, \underbrace{4567, 23, \ldots, 1337}_{\text{9 output labels}}$$

In total, 72 000 samples in this format for each scenario were provided for the training of the model, with two thirds (48 000) of these used for fitting and the rest (24 000) used for validation purposes. The complete dataset was also used for 10-fold cross-validation, again for each scenario.

1.2 Adapting the data

As described in the previous section, the data represents a multi-class, multi-label problem with a high dimensionality both in terms of input features and classes, with the additional restriction that the order of the output labels matters. [1]

1.3 Choice and adaptation of classifier

The dataset in question is part of ongoing research at the Department of Information Science and Engineering of the School of Electrical Engineering, and has already been modeled with great success using Random Forests and Neural Networks. The application of Support Vector Machines was then a natural step given the models' popularity in networking research literature; the specific application of the Linear Kernel for SVMs was a result of experimentation with the dataset, where initial experiments exposed the linearly separatable nature of the output variables.

Support Vector Machines are binary classifiers though, and thus additional modifications are required to adapt these classifiers to multiclass problems as the one in question. Specifically, the following techniques for adapting binary classifiers to multiclass applications were identified from literature [2, 3]:

- One-vs-One Method: For n classes, this method constructs $n(n-1)/2$ classifiers, each comparing a pair of classes from the training set. For prediction, all $n(n-1)/2$ classifiers are applied to an unseen sample, whose final class corresponds

to the class with the most "votes" after processing. In case of a tie, it selects the class with the highest total confidence, obtained by aggregating the confidence scores of each binary classifier.

- One-vs-Rest Method: Constructs n classifiers, each comparing one class in the training set with the rest (i.e. each classifier determines if a sample belongs to a specific class or not). At prediction time, these n classifiers are applied to the unseen sample and it is once again classified according to the majority vote.

Other multiclass adaptations of binary classifiers, like DAGSVM [4] and DDAG [5] were considered as well, but were ultimately considered overly complex for the problem at hand and dismissed.

2 Implementation

2.1 Language and libraries used

The language chosen for this project was Python 3.6, given its extensive support for scientific programming, data analysis and machine learning in the form of libraries. In particular, the libraries **scikit-learn** (and its multilabel extension, **scikit-multilearn**), **scipy**, **matplotlib** and **numpy** were used for the implementation [6, 7, 8, 9, 10].

2.2 Choice of classifier

Based on the analysis detailed in section 1.3, **sklearn.svm.LinearSVC** was selected as the base classifier to be used for this problem, as it corresponds to an implementation of a multi-class Support Vector Machine using the *One-vs-Rest* method and a linear kernel. Tests were also conducted with **sklearn.svm.SVC**, which implements a multi-class SVM using the *One-vs-One* method and a RBF-kernel, but its runtime proved to be cumbersomely large[2].

The classifier was then extended to work on multilabel outputs through the **skmultilearn.problem_transform.LabelPowerSet** class, which implements the *Label Powerset* problem transformation as detailed in 1.2, thus treating every distinct label combination in the training data as a separate class to pass to the multiclass classifier.

2.3 Parsing and adapting the data

The data provided for building the model consisted of 72 000 samples for each of the four positional accuracy values detailed in the problem description (section 1.1), divided into two files each: two thirds (48 000) of the samples for training, and one third (24 000) for validation. The format of the input files was one sample per line: 600

[2]Not a fair comparison, but for the unoptimized SVC case the runtime was over 3 hours for fitting and predicting, whereas the final optimized runtime for the LinearSVC is, on average, 110 seconds.

integers with values in $\{0, 1, 2\}$ representing the input features, followed by 9 integers with values in $[0, 12288]$ representing the output variables (everything separated by commas, see example below).

```
0,0,0,...,0,2,0,...,3968,3841,528,8080,4209,6273,9201,9073,8592
0,0,0,...,0,2,0,...,80,2545,2417,6145,8176,4112,8443,11761,8736
0,0,0,...,0,2,0,...,3680,0,0,4112,8017,7408,10241,8272,11505
```

Listing 1: Abbreviated example of input samples.

This data was read parsed **numpy**'s **loadfromtxt()** function, and then split into separate arrays for the input and output data (labels). The input data was passed "as-is" to the classifier, but the output data required additional processing

References

[1] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. "Mining multi-label data". In: *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 667–685.

[2] Chih-Wei Hsu and Chih-Jen Lin. "A comparison of methods for multiclass support vector machines". In: *IEEE transactions on Neural Networks* 13.2 (2002), pp. 415–425.

[3] D. M. J. Tax and R. P. W. Duin. "Using two-class classifiers for multiclass classification". In: *Object recognition supported by user interaction for service robots*. Vol. 2. 2002, 124–127 vol.2. DOI: 10.1109/ICPR.2002.1048253.

[4] P. Chen and S. Liu. "An Improved DAG-SVM for Multi-class Classification". In: *2009 Fifth International Conference on Natural Computation*. Vol. 1. Aug. 2009, pp. 460–462. DOI: 10.1109/ICNC.2009.275.

[5] John C Platt, Nello Cristianini, and John Shawe-Taylor. "Large margin DAGs for multiclass classification". In: *Advances in neural information processing systems*. 2000, pp. 547–553.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[7] P. Szymański and T. Kajdanowicz. "A scikit-based Python environment for performing multi-label classification". In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.01460 [cs.LG].

[8] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed <today>]. 2001–. URL: http://www.scipy.org/.

[9] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[10] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. “The NumPy Array: A Structure for Efficient Numerical Computation”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30. DOI: 10.1109/MCSE.2011.37. eprint: http://aip.scitation.org/doi/pdf/10.1109/MCSE.2011.37. URL: http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37.