15 April 2017

ACTIVE LEARNING

Lucas Lefevre
Ecole Polytechnique de Bruxelles, ULB

ABSTRACT

This is a report on the development and implementation of an active learning add-on for GRASS GIS, a free Geographic Information System (3). The aim of this add-on is the classification of remote sensing images based on land cover with a semi-supervised machine learning algorithm. The classification itself is performed with a support vector machine (SVM) and a heuristic query to a supervisor the labels of the samples that would be the most informative to the classifier. This heuristic is based both on an uncertainty criterion and a diversity criterion.

1. INTRODUCTION

A common problem in geography is to determine land cover from remote sensing images. This can be achieved very accurately by humans but the time and cost constraints often limits the number of samples that can be labeled manually. Therefore, automatic classification of remote sensing images is very often used. It is generally performed by a supervised machine learning algorithm that requires a lot of labeled data to train the classifier. Gathering the training set is tedious and can be expensive particularly if you have to go on site to label an area. This is why we want to minimize the task of manually labeling samples. This is of course in contradiction with the accuracy of the classifier because the quantity and quality of labeled samples is crucial to achieve an accurate classification. If we want a training set as small as possible while still reaching a good classification score, a semi-supervised approach can be used. Such an approach known as active learning iteratively expands the original training set by querying a supervisor to obtain the true label of samples that would improve the most the overall score. The number of labeled examples to learn the classification is often much lower than the number of examples needed in normal supervised algorithms.

2. BACKGROUND ON ACTIVE LEARNING

Let's first define formally an active learning system. It can be represented by a quintuple $(\mathcal{C}, \mathcal{Q}, \mathcal{S}, \mathcal{L}, \mathcal{U})$ (1). Where $\mathcal{C}$ is a supervised classifier trained on a labeled data set $\mathcal{L}$. $\mathcal{Q}$ is a query function used to select samples from a large unlabeled set $\mathcal{U}$. These samples are usually found in areas of uncertainty of the classifier. A supervisor $\mathcal{S}$ (e.g. a human expert) is able to label them.

Active learning algorithms are iterative schemes. For a given iteration, h samples are selected from $\mathcal{U}$ that would maximize performance and reduce the uncertainty of the classifier if it knew their true label. Once labeled by the supervisor those samples are added to the training set $\mathcal{L}$ and remove from the pool of candidates $\mathcal{U}$. By iterating this process, the training set is expanding after each iteration only by the most informative examples. This leads to a training set with a much better quality than if is was composed of randomly selected samples. But in order to choose the best samples to add to the training set, we need a strategy to rank the candidates in $\mathcal{U}$. Multiple strategies (heuristics) can be used to perform this ranking grouped in three main families (10) :

- Committee-based heuristics
- Large margin-based heuristics
- Posterior probability-based heuristics

These three families are briefly explained below.

2.1. *Committee-based active learning*

The strategy to quantify the uncertainty of a sample is to consider a committee of multiple learners. The members of the committee use different hypothesis to label the samples in the pool. Every learner is trained separately and assign a class to each candidate in the unlabeled pool. The candidates showing maximum disagreement between the learners in the committee are chosen to be labeled by the supervisor. A disadvantage of this family of heuristics is the computational overhead of training multiple classifiers.

2.2. *Large-margin-based active learning*

These heuristics are specific to margin-based classifier. SVMs are naturally a good example of such methods. The key idea is to consider the distance to the separating hyperplane (absolute value of the decision function) to quantify the uncertainty of a sample. Intuitively, a sample away from the decision boundary has a high confidence in the class assignment and will not be of much help to the classifier. Conversely a sample close to the decision boundary has a low confidence in the class prediction. It is probably the easiest heuristic to implement when working with SVM.

2.3. *Posterior probability-based active learning*

The third family of heuristics uses the estimation of posterior probabilities of class membership ($p(y|x)$). The probabilities give a level a confidence in the class assignment. The most uncertain samples will have an near uniform probability of belonging to each class. SVM does not directly provide probability estimates but they can be computed via Platt's method (7).

3. TECHNIQUE USED

This section presents the strategy that is used here to select the most informative samples from the unlabeled pool of samples. The supervised classifier $\mathcal{C}$ used is a SVM which naturally leads to an active learning technique based on large-margin. The heuristic used here

goes further than the ideas presented above and combines two criteria to select samples from the pool of candidates. The first criterion is based on the uncertainty of class assignments and the second is based on the diversity among uncertain samples. The query function $\mathcal{Q}$ selects h samples at each iteration and works in two stages. First it will find the $m > h$ most uncertain samples from the unlabeled pool $\mathcal{U}$. Next it will extract h samples out of the m selected samples at the first stage such that only the most diverse samples are kept. This has the effect of removing the most redundant samples. The two stages are detailed in the following subsections.

3.1. *Uncertainty criterion*

The most straight forward way to quantify uncertainty in a binary classification problem is to consider the distance of a sample $\mathbf{x}_i$ from the SVM hyperplane (8)

$$f(\mathbf{x}_i) = \sum_{j=i}^{n} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + b$$

where $\mathbf{x}_j$ are the support vectors (samples with nonzero α_j Lagrangian multipliers. $K(\mathbf{x}_j, \mathbf{x}_i)$ is a kernel function which is a similarity measure between $\mathbf{x}_i$ and a support vector $\mathbf{x}_j$. $b \in \mathbb{R}$ is the intercept term and the labels y_j of the support vectors are $+1$ if $\mathbf{x}_j$ belongs to the positive class and -1 if it belongs to the negative class. The closer a sample is to the separating hyperplane, the more uncertain its classification is.

Unfortunately in the case of determining land cover, instances are classified between more than two classes and this method must be generalized to multiclass problems (MC). In a multiclass problem with N classes $\omega = 1, \ldots, N$ and a "One-versus-All" setting, there is actually N hyperplanes, one per class. Each hyperplane separates samples belonging to one class from the rest of the samples. With that in mind, we can define $f(\mathbf{x}_i, \omega)$ as the distance between the sample $\mathbf{x}_i$ and the separating hyperplane for the class ω.

Using this new distance function we can extend the idea from binary to multiclass classifiers to find the most uncertain sample. The strategy is to consider the difference between the first and the second largest distance values to the hyperplanes (2) :

$$\hat{\mathbf{x}} = \underset{\mathbf{x}_i \in \mathcal{U}}{\arg\min}\, f(\mathbf{x}_i)^{MC}$$

with

$$f(\mathbf{x}_i)^{MC} = \max_{\omega \in N} f(\mathbf{x}_i, \omega) - \max_{\omega \in N \setminus \{\omega^+\}} f(\mathbf{x}_i, \omega)$$

Where ω^+ is the class predicted by the classifier for $\mathbf{x}_i$.

Intuitively, $f(\mathbf{x}_i)^{MC}$ computes the uncertainty between the two most likely classes. If this value is high, it means that the classifier has a high confidence in the predicted class ω^+. On the contrary, if it is small, there is a possible conflict with the two most probable classes e.i. the sample $\hat{\mathbf{x}}$ is close to the boundary between the two classes. Thus, this sample is selected by the query function because it is considered uncertain.

The query function selects a subset $X_{\mathcal{Q}}$ of m samples with this uncertainty criterion.

Note that this exact strategy can be used with a "One-versus-One" setting if you change the distances to the hyperplanes by the class membership probabilities. The probabilities can be estimated with Platt's method (7) but it involves a computation overhead.

3.2. *Diversity criterion*

Labeling at each iteration a batch of h samples selected on the basis of uncertainty alone is not necessarily better than labeling a single sample because they may be very similar and not bring much more information to the model. This is why the query function first selects m uncertain samples (with $m > h$) and then reduces this number to keep only the h most diverse samples.

This process iteratively removes the most redundant sample $\hat{\mathbf{x}}_d$ from $X_{\mathcal{Q}}$ until only h samples are left. To find this sample $\hat{\mathbf{x}}_d$ that is more similar to all other samples we consider both the distance to its closest neighbour and the average distance to all others (1).

$$\hat{\mathbf{x}}_d = \underset{\mathbf{x}_j \in X_{\mathcal{Q}}}{\arg\max} \left[\lambda \max_{\mathbf{x}_i \in X_{\mathcal{Q}}} K(\mathbf{x}_j, \mathbf{x}_i) + (1-\lambda) \frac{1}{m} \sum_{\mathbf{x}_k \in X_{\mathcal{Q}} \setminus \{\mathbf{x}_j\}} K(\mathbf{x}_j, \mathbf{x}_k) \right]$$

λ is a parameter that tunes the weight between the distance to the closest neighbour and the average distance to other samples.

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The algorithm presented above was implemented using the python package *scikit-learn* for SVM computations (9) and Numpy (6) for the active learning heuristic. The implementation was integrated into a GIS GRASS add-on (3). This enable to launch this algorithm directly from GRASS and easily specify input parameters from an automatically generated graphical user interface. You can find the user manual accompanying this add-on in the appendix section. It briefly describes its purpose and its input parameters and options.

The input data is composed of features extracted from the different region areas (also called objects) that the algorithm have to classify. The *i.segment* (5) add-on for GRASS is used to identify the objects from the imagery data (groups similar pixels into unique objects). Object features are statistical measures (such as spectral channel means and variances, min/max values, region size and shape, , ...) computed from the different regions with the *i.segment.stats* add-on (4).

To assess the performance of this solution and the benefits of iteratively expanding the training set with the most informative samples, several experiments were carried out. In particular the advantage of active learning compared to normal supervised learning (e.i. the training set is randomly generated) and the advantage of adding the diversity criterion over the uncertainty criterion alone. The experiments were conducted with a data set of 1450 samples of 52 features each classified in 9 classes. The initial training set was composed of 60 labeled samples with at least 5 samples per class. At each iteration the training set grows by $h = 5$ samples and the number of pre-selected samples before applying the diversity criterion is $m = 15$. In order to obtain statistically significant results, 80 trials were run varying the

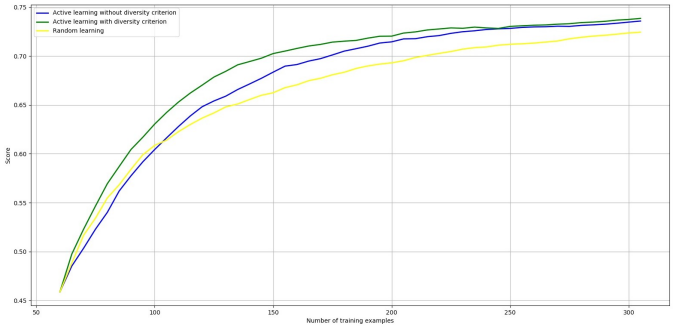

Figure 1. Comparison between normal supervised learning (yellow), active learning with only the uncertainty criterion (blue) and active learning with both the uncertainty and diversity criterion (green).The curves show classification scores as a function of the size of the training set. $m = 15$, $h = 5$, $\lambda = 0.25$

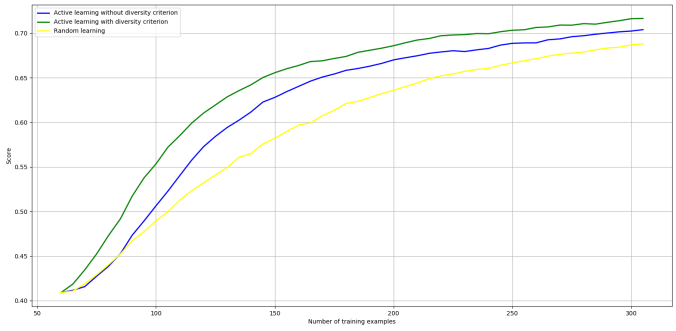

Figure 2. Same experiment as in Figure 1, but the data has been linearly scaled instead of centered to the mean and component wise scaled to unit variance.

initial training set (randomly selecting samples from the pool).

The results in Figure 1 show that the size to reach the same classification score can be much smaller with the technique used compared to normal supervised learning. For example the training set needed to reach a score of 0.7 is reduces by 32% from around 220 labeled samples to 150. The benefits of the diversity criterion is also clearly visible.

Note that a preprocessing stage scales the data such that all features are centered around zero and have unit variance. Linearly scaling the data such that the 5^{th} percentile goes to 0 and the 95^{th} to 1 was also tired but it resulted in a lower score as shown in Figure 2.

5. CONCLUSION

In this report we presented an active learning algorithm to perform multiclass SVM classification of land cover in remote sensing images. The algorithm is able to reduce the number of labeled sample needed to train the classifier compared to normal supervised learning techniques. The proposed strategy successfully selects the most informative samples from the unlabeled pool based on their class prediction uncertainty and the diversity among them. Experimental results proved the effectiveness of this strategy applied to remote sensing images but it could also be applied to many other types of data. Especially the diversity criterion could be adapted with classifier other than SVM and could be the goal of a future work.

REFERENCES

[1] Bruzzone, L., and Persello, C. Active learning for classification of remote sensing images. Tech. rep., Department of Information Engineering and Computer Science, University of Trento, 2009.

[2] Demir, B., Persello, C., and Bruzzone, L. Batch-mode active-learning methods for the interactive classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing 49*, 3 (2011).

[3] GRASS. Add-ons. https://grass.osgeo.org/download/addons/. [Online; accessed 10-April-2017].

[4] Lennert, M. i.segment.stats. https://grass.osgeo.org/grass72/manuals/addons/i.segment.stats.html. [Online; accessed 15-April-2017].

[5] Momsen, E. i.segment. https://grass.osgeo.org/grass73/manuals/i.segment.html. North Dakota State University, [Online; accessed 15-April-2017].

[6] Numpy. http://www.numpy.org. [Online; accessed 10-April-2017].

[7] Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS* (1999), MIT Press, pp. 61–74.

[8] Schölkopf, B., and Smola, A. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, Dec. 2002.

[9] Scikit-Learn. Support vector machines. http://scikit-learn.org/stable/modules/svm.htmlsupport-vector-machines. [Online; accessed 10-April-2017].

[10] Tuia, D., Volpi, M., Copa, L., Kanevski, M., and Munoz-Mari, J. A survey of active learning algorithms for supervised remote sensing image classification. *IEEE Journal of Selected Topics in Signal Processing 5*, 3 (2011), 606–617.

APPENDIX

USER MANUAL

DESCRIPTION

This module uses SVM from the scikit-learn python package to perform classification on regions of raster maps. These regions can be the output of i.segment or r.clump.

The module enables learning with only a small initial labeled data set via active learning. This semi-supervised learning algorithm interactively query the user to label the regions that are most useful to improve the overall classification score. With this technique, the number of examples to learn the classification is often much lower than the number of examples needed in normal supervised algorithms. You should start the classification with a small training set and run the module multiple times to label new informative samples to improve the classification score.

The samples that are chosen to be labeled are the ones where the class prediction is the most uncertain. Moreover, from the more uncertain samples, only the most different samples are kept. This diversity heuristic takes into account for each uncertain sample the distance to its closest neighbour and the average distance to all other samples. This ensures that newly labeled samples are not redundant with each other.

The learning data should be composed of features extracted from the regions, for example with the i.segment.stats module. The features of the training set, the test set and the unlabeled set should be in three different files in csv format. The first line of each file must be a header containing the features' name. Every regions should be uniquely identified by the first attribute. The classes for the training and test examples should be the second attribute.

Example of a training and test files :

```
cat,Class\_num,attr1,attr2,attr3
167485,4,3.546,456.76,6.76
183234,6,5.76,1285.54,9.45
173457,2,5.65,468.76,6.78
```

Example of an unlabeled file :

```
cat,attr1,attr2,attr3
167485,3.546,456.76,6.76
183234,5.76,1285.54,9.45
173457,5.65,468.76,6.78
```

The training set can be easily updated once you have labeled new samples. Create a file to specify what label you give to which sample. This file in csv format should have a header and two attributes per line : the ID of the sample you have labeled and the label itself. The module will transfer the newly labeled samples from the unlabeled set to the training set, adding the class you have provided. This is done internally and does not modify your original files. If the user wants to save the changes in new files according to the updates, add the -u flag and new files will be created with the new labeled samples added to the training file and removed from the unlabeled file. You can specify the path of those output files in the parameters (training_updated, unlabeled_updated, update_updated) or leave the default paths. If the transfer of a sample is successful, the corresponding line is deleted in the new update file.

Example of an update file :

```
cat,Class_num
194762,2
153659,6
178350,2
```

Here are more details on a few parameters :

- learning_steps : This is the number of samples that the module will ask to label at each run.
- nbr_uncertainty : Number of uncertain samples to choose before applying the diversity filter. This number should be lower than learning_steps.
- diversity_lambda : Parameter used in the diversity heuristic. If close to 0 only take into account the average distance to all other samples. If close to 1 only take into account the distance to the closest neighbour.
- c_SVM : Penalty parameter C of the error term. If it is too large there is a risk of overfitting the training data. If it is too small you may have underfitting.
- gamma_SVM : Kernel coefficient. 1/features is often a good value to start with. If C or gamma is left empty (or both), a good value based on the data distribution is found by cross-validation (at least 3 samples per class in the training set is needed, more is better). This automatic parameter tuning requires more computation time as the training set grows.
- search_iter :Number of parameter settings that are sampled in the automatic parameter search (C, gamma). search_iter trades off runtime vs quality of the solution.

EXAMPLES

The following examples are based on the data files found in this module repository.

SIMPLE RUN WITHOUT AN UPDATE FILE

```
>>>r.objects.activelearning training_set=/path/to/training_set.csv

test_set=/path/to/test_set.csv unlabeled_set=/path/to/unlabeled_set.csv
```

```
Parameters used : C=146.398423284, gamma=0.0645567086567, lambda=0.25
12527959
9892568
13731120
15445003
13767630
Class predictions written to predictions.csv
Training set : 70
Test set : 585
Unlabeled set : 792
Score : 0.321367521368
```

WITH AN UPDATE FILE

The five samples output at the previous example have been labeled and added to the update file.

```
>>>r.objects.activelearning training_set=/path/to/training_set.csv test_set=/path/to/test_set.csv

unlabeled_set=/path/to/unlabeled_set.csv update=/path/to/update.csv
```

```
Parameters used : C=101.580687073, gamma=0.00075388337475, lambda=0.25
Class predictions written to predictions.csv
Training set : 75
Test set : 585
Unlabeled set : 787
Score : 0.454700854701
8691475
9321017
14254774
14954255
15838185
```

NOTES

This module requires the scikit-learn python package. This module needs to be installed in your GRASS GIS Python environment. Please refer to r.learn.ml's Notes to install this package. The memory usage for 1450 samples of 52 features each is around 650 kb. This number can vary due to the unpredictablity of the garbage collector's behaviour. Everything is computed in memory; therefore the size of the data is limited by the amount of RAM available.